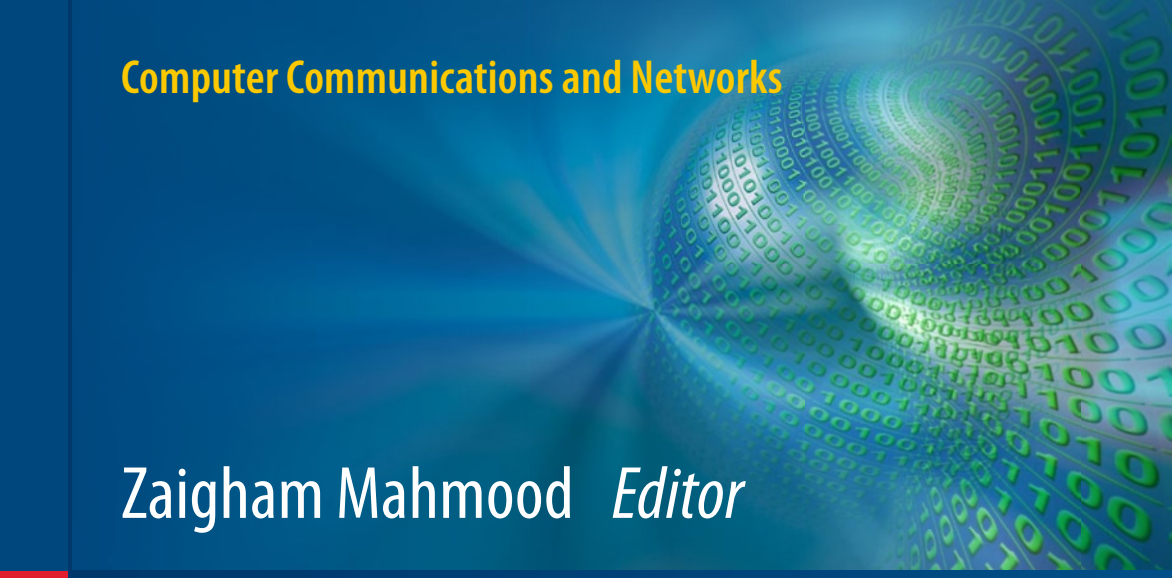
Computer Communications and Networks



Zaigham Mahmood *Editor*

Cloud Computing

Methods and Practical Approaches

Springer

Computer Communications and Networks

For further volumes:
http:// www.springer.com/series/4198

The Computer Communications and Networks series is a range of textbooks, monographs and handbooks. It sets out to provide students, researchers and non-specialists alike with a sure grounding in current knowledge, together with comprehensible access to the latest developments in computer communications and networking.

Emphasis is placed on clear and explanatory styles that support a tutorial approach, so that even the most complex of topics is presented in a lucid and intelligible manner.

Zaigham Mahmood
Editor

Cloud Computing

Methods and Practical Approaches

Springer

Editor
Zaigham Mahmood
School of Computing
University of Derby, UK
North West University, Potchefstroom, South Africa

Series Editor
A.J. Sammes
Centre for Forensic Computing
Cranfield University
Shrivenham campus
Swindon, UK

ISSN 1617-7975
ISBN 978-1-4471-5106-7 ISBN 978-1-4471-5107-4 (eBook)
DOI 10.1007/978-1-4471-5107-4
Springer London Heidelberg New York Dordrecht

Library of Congress Control Number: 2013939609

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

To

Zoya and Imran:
Happy 2nd Anniversary

Preface

Overview

Cloud computing is an attractive paradigm that allows consumers to self-provision cloud-based resources, application services, development platforms and virtualized infrastructures. The benefits associated with cloud computing are enormous; however, there still are numerous inherent issues due to the dynamic, virtualized, distributed and multi-tenant nature of the cloud environment. Some of the major concerns relate to the security and privacy of consumer data, interoperability and portability of services across different clouds, inappropriate service level agreements and lack of open standards as well as cloud provider and vendor lock-in.

Although the technologies underlying the cloud paradigm have been in existence for some time, the frameworks and methodologies for construction, deployment and delivery of cloud services and environments are in the process of being further developed and refined. Newer approaches to such methodologies and mechanisms are required to ensure that the software developed is scalable and suitable for virtualized distributed environments, the deployment of platforms is secure and exhibits the inbuilt characteristic of multi-tenancy, there are clear open standards with respect to the interoperability and portability across different clouds and the new breed of security threats that now exist due to the shared trust boundaries in the cloud environment are, at least, minimized. It is also important that the chosen frameworks and mechanisms are suitable and appropriate for the distributed and virtualized environments.

This book, *Cloud Computing: Methods and Practical Approaches*, aims to capture the state of the art and present discussion and guidance on the methods, approaches, technologies and frameworks relevant to the emerging cloud paradigm. In this text, 35 researchers and practitioners from around the world have presented their research, practice reports, case studies and suggestions for further development of the cloud computing paradigm.

Objectives

The aim of this text is to present the current research and development with respect to the frameworks, methods and practical approaches relevant to cloud computing. The key objectives for the book include:

- Capturing the state of the art in cloud technologies, infrastructures and service delivery and deployment models
- Discussing the relevant theoretical frameworks, practical approaches and suggested methodologies
- Providing guidance and best practices for development of cloud-based services and infrastructures
- Advancing the understanding of the emerging new methodologies relevant to the cloud paradigm
- Presenting suggestions for future developments and research directions

Organization

There are 15 chapters in *Cloud Computing: Methods and Practical Approaches*. These are organized in four parts, as follows:

- Part I: *Cloud Infrastructures and Frameworks*. This section has a focus on cloud frameworks and deployment approaches. There are four chapters. Two of these discuss metadata-based cloud frameworks and architecture of scientific data systems. The other two chapters present architecture, solutions and services for social, dynamic and consumer clouds.
- Part II: *Cloud Enablement and Management*. This comprises three chapters that consider management aspects of cloud computing. One of these chapters discusses the role of cloud service brokers, and the other two discuss resource and scheduling management of cloud applications and present management infrastructures for power-efficient cloud environments.
- Part III: *Cloud Perspectives and Patterns*. There are four chapters in this part that focus on consumer perspectives on mobile cloud computing, potential and concerns relating to cloud-based enterprise resource planning and cloud solution patterns. The patterns refer to the entire set of software, platforms and infrastructure solutions.
- Part IV: *Cloud Modernization and Migration*. This final section also consists of four chapters. The first two chapters present modernization, migration and premigration assessment methodologies. The final two chapters discuss software performance testing and open-source cloudware support with respect to software portability.

Target Audiences

This text has been developed to support a number of potential audiences, including the following:

- *Enterprise architects*, *business analysts and software developers* who wish to adapt to newer approaches to developing and deploying cloud-based services and infrastructures
- *IT infrastructure managers and business leaders* who need to clearly understand the requirement for newer methodologies and frameworks in the context of cloud paradigm
- *Students and lecturers* of cloud computing who have an interest in further enhancing the knowledge of the cloud-related methodologies, mechanisms and frameworks
- *Researchers* in this field who wish to further increase their understanding of the current practices, methodologies, mechanisms and frameworks to further develop the same

Suggested Uses

Cloud Computing: Methods and Practical Approaches can be used as a primer and textbook on university courses in cloud computing. It can also be used as a reference text by software architects, infrastructure specialists and other practitioners in the field of cloud computing.

For adoption as a course text, the following programme of study is suggested for a 12-week teaching semester format:

- Weeks 1–4: Part I
- Weeks 4–7: Part II
- Weeks 7–9: Part III
- Weeks 9–12: Part IV

Acknowledgements

The editor acknowledges the help and support of the following colleagues during the review and editing phases of this text:

- Dr. Wasif Afzal, Bahria University, Islamabad, Pakistan
- Mr. Daniel Crichton, Jet Propulsion Laboratory, California Inst Tech, USA
- Dr. Ashraf Darwish, Helwan University, Cairo, Egypt

- Dr. Shehzad Khalid, Bahria University, Islamabad, Pakistan
- Prof. Francisco Milton Mendes, Rural Federal Univ. Semi-Arid, Brazil
- Prof. Mahmood Niazi, King Fahd Univ. of Petroleum and Minerals, Dhahran
- Dr. S. Parthasarathy, Thiagarajar College of Engineering, Madurai, India
- Dr. Pethuru Raj, Wipro Technologies, Bangalore, India
- Dr. Muthu Ramachandran, Leeds Metropolitan University, Leeds, UK
- Dr. C. R. Rene Robin, Jerusalem College of Engineering, Chennai, India
- Dr. Lucio Agostinho Rocha, State University of Campinas, Brazil
- Dr. Lloyd G. Waller, University of the West Indies, Kingston, Jamaica
- Dr. Fareeha Zafar, GC University, Lahore, Pakistan

I would also like to thank the contributors of this book: 35 authors and coauthors, from academia as well as industry from around the world, who collectively submitted 15 chapters. Without their efforts in developing quality contributions, conforming to the guidelines and meeting often the strict deadlines, this text would not have been possible.

Grateful thanks are also due to the members of my immediate family—Rehana, Zoya, Imran, Hanya and Ozair—for their support and encouragement.

University of Derby, UK Zaigham Mahmood
North West University, Potchefstroom, South Africa
January 2013

Contents

Contributors

Ira Agrawal Cloud Computing CoE, Infosys Labs, Infosys Ltd., Bangalore, Karnataka, India

Juncal Alonso European Software Institute Division, TECNALIA ICT, Zamudio, Spain

Omer Muhammad Ayoub Faculty of IT, University of Central Punjab, Lahore, Pakistan

Yolande Berbers iMinds-DistriNet, Department of Computer Science, KU Leuven, Leuven, Belgium

George Chang Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA

Luca Cinquini Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA

Antonio Corradi DEIS, University of Bologna, Bologna, Italy

Daniel Crichton Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA

Shyam Kumar Doddavula Cloud Computing CoE, Infosys Labs, Infosys Ltd., Bangalore, Karnataka, India

Marisa Escalante European Software Institute Division, TECNALIA ICT, Zamudio, Spain

Mario Fanelli DEIS, University of Bologna, Bologna, Italy

Luca Foschini DEIS, University of Bologna, Bologna, Italy

Eduardo Martins Guerra Laboratory of Computing and Applied Mathematics (LAC), National Institute for Space Research (INPE), São José dos Campos, SP, Brazil

Sean Hardman Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA

Sudhanshu Hate Infosys Ltd., Bangalore, Karnataka, India

Dolly Kandpal Qamlo Information Services Limited, Bangkok, Thailand

Naveen Kumar Infosys Ltd., Bangalore, Karnataka, India

Emily Law Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA

Krishna Markande Department of Engineering Services, Infosys Ltd., Bangalore, Karnataka, India

Chris A. Mattmann Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA

Department of Computer Science, University of Southern California, Los Angeles, CA, USA

Constandinos Mavromoustakis Department of Computer Science, University of Nicosia, Nicosia, Cyprus

Sridhar J. Murthy Department of Engineering Services, Infosys Ltd., Bangalore, Karnataka, India

Nayyab Zia Naqvi iMinds-DistriNet, Department of Computer Science, KU Leuven, Leuven, Belgium

Ednelson Oliveira Department of Computer Science, Aeronautical Institute of Technology (ITA), Praça Marechal Eduardo Gomes, São José dos Campos, São Paulo, Brazil

Leire Orue-Echevarria European Software Institute Division, TECNALIA ICT, Zamudio, Spain

Katerina Papanikolaou School of Sciences, European University Cyprus, Nicosia, Cyprus

S. Parthasarathy Department of Computer Applications, Thiagarajar College of Engineering, Madurai, Tamil Nadu, India

Dana Petcu Institute e-Austria Timisoara, West University of Timisoara, Timisoara, Romania

Davy Preuveneers iMinds-DistriNet, Department of Computer Science, KU Leuven, Leuven, Belgium

Pethuru Raj Wipro Technologies, Bangalore, Karnataka, India

Massimiliano Rak Department of Industrial and Information Engineering, Second University of Naples, Naples, Italy

Vikas Saxena Cloud Computing CoE, Infosys Labs, Infosys Ltd., Bangalore, Karnataka, India

Rajaraajeswari Settu Department of Master of Computer Applications, Raja Rajeswari College of Engineering, Bangalore, Karnataka, India

Khawaja Shams Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA

Fareeha Zafar School of Computing, University of Derby, Derby, UK

Department of Computer Science, GC University Lahore, Lahore, Pakistan

Editor

Zaigham Mahmood
Professor Zaigham Mahmood is a researcher and author. He has an M.Sc. in Mathematics, M.Sc. in Computer Science and Ph.D. in Modelling of Phase Equilibria. He is a Senior Technology Consultant at Debesis Education, UK, a researcher at the University of Derby UK and Professor Extraordinaire at the North West University in South Africa. He is, currently, also a Foreign Professor at NUST Islamabad, Pakistan.

Professor Mahmood has published over 100 articles in international journals and conference proceedings in addition to two reference texts on e-government and four books on cloud computing viz. *Cloud Computing: Concepts, Technology and Architecture*; *Cloud Computing for Enterprise Architectures*, *Software Engineering Frameworks for the Cloud Computing Paradigm* and *Cloud Computing: Methods and Practical Approaches*. He is also the Editor-in-Chief of the *Journal of E-Government Studies and Best Practices* and Editor-in-Chief of a book series on e-government for the IGI Global publishers.

Dr. Mahmood is an active researcher; he also serves as editorial board member of several journals, books and conferences; guest editor for journal special issues; organiser and chair of conference tracks and workshops; and keynote speaker at conferences. His research interests encompass subject areas including software engineering, project management, enterprise computing, e-government studies and cloud computing.

Professor Mahmood can be reached at *z.mahmood@debesis.co.uk*. He welcomes your views and comments.

Part I
Cloud Infrastructures and Frameworks

Chapter 1
Metadata-Based Frameworks in the Context of Cloud Computing

Eduardo Martins Guerra and Ednelson Oliveira

Abstract In the context of cloud computing, one server is usually responsible to run multiple applications and a single application is spread across multiple servers. On the one hand, the applications need to be able to determine how the cloud environment should handle its execution, or even the execution of each one of its components. Yet, on the other hand, the applications should be decoupled from the middleware that executes them, enabling each one to evolve independently. Based on this scenario, it is possible to state that metadata-based frameworks are a suitable option for the interaction between the application and the services provided by the cloud, since it decouples the application from the environment and allows a transparent individual configuration of each class. The goal of this chapter is to describe the essence of metadata-based frameworks and how they can be applied to cloud computing. It brings several examples of cloud computing frameworks and describes some design practices for the framework structure, scenarios that are suitable for the metadata-based approach and best practices for metadata configuration. As a result, after reading this chapter, the reader should be able to understand the basic functioning of a metadata-based framework and why it is suitable for cloud applications.

Keywords Metadata • Metadata-based frameworks • Cloud-based services • Cloud patterns

E.M. Guerra (✉)
Laboratory of Computing and Applied Mathematics (LAC),
National Institute for Space Research (INPE), P. O. Box 515 – 12227-010,
São José dos Campos, SP, Brazil
e-mail: eduardo.guerra@inpe.br

E. Oliveira
Department of Computer Science, Aeronautical Institute of Technology (ITA),
Praça Marechal Eduardo Gomes, 50 Vila das Acassias,
São José dos Campos, São Paulo, Brazil
e-mail: ednelsonoliveira@gmail.com

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_1,

1.1 Introduction

Cloud computing applications are executed across multiple servers, and this fact should be transparent to the developers, especially when the platform is provided as a service. Additionally, the application should interact with some cloud services, such as persistence and session management. To allow this transparency, the application should be decoupled from the cloud environment services. Since servers also execute many applications, another important requirement is to enable each application, or even each component, to configure how it should be handled by the cloud. Combining these two requirements, it is possible to conclude that the cloud provider should provide frameworks that, at the same time, abstract the cloud environment and allow a flexible configuration of how each application class should be handled.

Many frameworks for developing cloud applications, such as Gaelyk [5], Objectify [6], Play Framework [7] and Guice [8], use metadata to allow a fine-grained configuration of the application classes in the cloud environment. Gaelyk is a Groovy toolkit for web application development for Google App Engine (GAE) [5], which makes a classic use of metadata to map classes to persistent entities. Objectify implements the same mapping, but covering all features of the Google App Engine Datastore, using metadata also to define entities, relationships and listeners.

Another very interesting use for metadata can be found in Play Framework [7] where a web framework is deployed in GAE. It uses an annotation to schedule asynchronous jobs, which can run periodically or in an instant defined by expressions. Google Guice [8] uses metadata to inject dependencies from cloud services into application classes. It allows implementation classes to be programmatically bound to an interface and then injected into constructors, methods or fields using the @Inject annotation.

Metadata-based frameworks can be defined as frameworks that consume custom metadata from application classes to customize its behaviour [1]. Common approaches for metadata definition are XML documents, code annotations, code conventions and even databases. From the developer's perspective, the focus is on declarative metadata definition and not on method invocation or class extension. Recent studies reveal that one benefit in this approach is the decoupling between framework and application compared to other techniques [2]. This characteristic makes this kind of solution suitable for cloud application, where it is desirable to decouple the business rules from the infrastructure.

The goal of this chapter is to explore the characteristics of metadata-based frameworks and how they can be used for the development of cloud applications, considering the concept of platform as a service. Examples of real cloud frameworks are used to illustrate the practices presented. Additionally, it also presents some patterns that can be used to structure internally this kind of framework, architectural scenarios where this approach is used in cloud applications and best practices for metadata schema definition and metadata configuration. In brief, this chapter presents a complete set of practices considering distinct perspectives about metadata-based frameworks for cloud applications.

1.2 Frameworks and Metadata

To understand how a metadata-based framework can be useful on a cloud environment, it is important to understand the concepts about frameworks and metadata definition. That knowledge is important to understand how frameworks can be internally structured to allow behaviour specialization and extension using classic object-oriented techniques and metadata. This section also explores the alternatives on metadata definition and the basic functioning of a metadata-based framework.

1.2.1 Framework Concepts

A framework can be considered an incomplete software with some points that can be specialized to add application-specific behaviour, consisting in a set of classes that represents an abstract design for a family of related problems. It is more than well-written class libraries, which are more application independent and provide functionality that can be directly invoked. A framework provides a set of abstract classes that must be extended and composed with others to create a concrete and executable application. Those classes can be application-specific or taken from a class library, usually provided along with the framework [9].

The main purpose of a framework is to provide reuse in the application, but in a larger granularity than a class. This reuse of the design provided by a framework is defined by its internal interfaces and the way that the functions are divided among its components. It can be considered more important than the source code reuse. According to Jacobsen and Nowack [10], the reuse in a framework is performed in three levels: analysis, design and implementation. The flexibility which makes possible the application behaviour specialization is important to enable its usage in multiple contexts.

Another important characteristic of a framework is the inversion of control [11]. Framework runtime architecture enables the definition of processing steps that can call applications handlers. This allows the framework to determine which set of application methods should be called in response to an external event. The common execution flow is an application to invoke the functionality on an external piece of software and not the opposite. On the other hand, using the inversion of control approach, the framework, and not the application, is responsible for the main execution flow. This is also known as the Hollywood Principle [12]: *Don't call us, we'll call you*.

A framework can contain points, called hot spots, where applications can customize their behaviour [13]. Each type of behaviour which can be customized in a framework is called variability, and they represent domain pieces that can change among applications. Points that cannot be changed are called frozen spots. Those points usually define the framework general architecture, which consists in its basic

components and the relationships between them. This section presents the two different types of hot spots that respectively use inheritance and composition to enable the application to add behaviour. New approaches in framework development can use other kinds of hot spots, such as reflective method invocation [14] and metadata definition [1].

1.2.2 Metadata Definition

Metadata is an overloaded term in computer science and can be interpreted differently according to the context. In the context of object-oriented programming, metadata is information about the program structure itself such as classes, methods and attributes. A class, for example, has intrinsic metadata like its name, its superclass, its interfaces, its methods and its attributes. In metadata-based frameworks, the developer also must define some additional application-specific or domain-specific metadata.

Even in this context, metadata can be used for many purposes. There are several examples of this, such as source code generation [15], compile-time verifications [16, 17] and class transformation [18]. The metadata-based components consume metadata at runtime and use it for framework adaptation. This distinction is important because the same goal could be achieved using different strategies [19].

The metadata consumed by the framework can be defined in different ways. Naming conventions [20] use patterns in the name of classes and methods that have a special meaning for the framework. To exemplify this, there are the JavaBeans specification [21], which uses method names beginning with 'get' and 'set', and the JUnit 3 [22], which interprets methods beginning with 'test' as test cases implementation. Ruby on Rails [23] is an example of a framework known by the naming conventions usage. Other information can also be used on conventions, such as variable types, method parameters and other class characteristics.

Conventions usage can save a lot of configurations, but it has a limited expressiveness. For some scenarios, the metadata needed are more complex and naming conventions are not enough. An alternative can be setting the information programmatically in the framework, but it is not used in practice in the majority of the frameworks. Another option is metadata definition in external sources, like XML files and databases. The possibility to modify the metadata at deploy time or even at runtime without recompiling the code is an advantage of this type of definition. However, the definition is more verbose because it has to reference and identify program elements. Furthermore, the distance that configuration keeps from the source code is not intuitive for some developers.

Another alternative that has become popular in the software community is the use of code annotations, which is supported by some programming languages like Java [24] and C# [25]. Using this technique, the developer can add custom metadata

elements directly into the class source code, keeping this definition less verbose and closer to the source code. The use of code annotations is a technique called attribute-oriented programming [26].

Fernandes et al. [4] presented a study about how the different types of metadata definition are suitable for different framework requirements. Their study analyses how simple it is to use and develop a framework with some requirements about metadata. The requirements considered are metadata extension, existence of more than one metadata schema per class in different contexts and runtime metadata modification.

1.2.3 Metadata-Based Frameworks

Metadata-based frameworks can be defined as frameworks that process their logic based on the metadata of the classes whose instances they are working with [1]. In these frameworks, the developer must define, into application classes, additional domain-specific or application-specific metadata to be consumed and processed by the framework.

The use of metadata changes the way frameworks are built and how they are used by software developers. In an interview motivated by the 15 years of the book *Design Patterns: Elements of Reusable Object-Oriented Software* [27], when asked about how the metadata approach replaces or complements the patterns in the book, Erich Gamma answered the following [28]:

> While they complement the patterns in DP (referring to the patterns in the book) it can indeed be the case that meta-programming can replace the design pattern used in a design. The evolution of JUnit 3 to JUnit 4 comes to mind. JUnit 3 was a small framework that used several patterns like Composite, Template Method and Command. JUnit 4 leverages the Annotations meta-programming facilities introduced in J2SE 5.0. The use of the patterns disappeared and the framework evolved into a small set of annotations plus a test runner infrastructure that executes the annotated Java code.

In metadata-based frameworks, some variable points in the framework processing are determined by class metadata. Reflective algorithms must be generic and, in some cases, they cannot be applied due to more specific requirements for some classes. Metadata can be used to configure specific behaviours when the framework is working with that class.

The developer's perspective in the use of those frameworks has a stronger interaction with metadata configuration than with method invocation or class specialization. In traditional frameworks, the developer must extend its classes, implement its interfaces and create hook classes for behaviour adaptation. He also has to create instances of those classes, setting information and hook class instances. Using metadata-based frameworks, programming focus is on declarative metadata configuration and the method invocations in framework classes are smaller and localized.

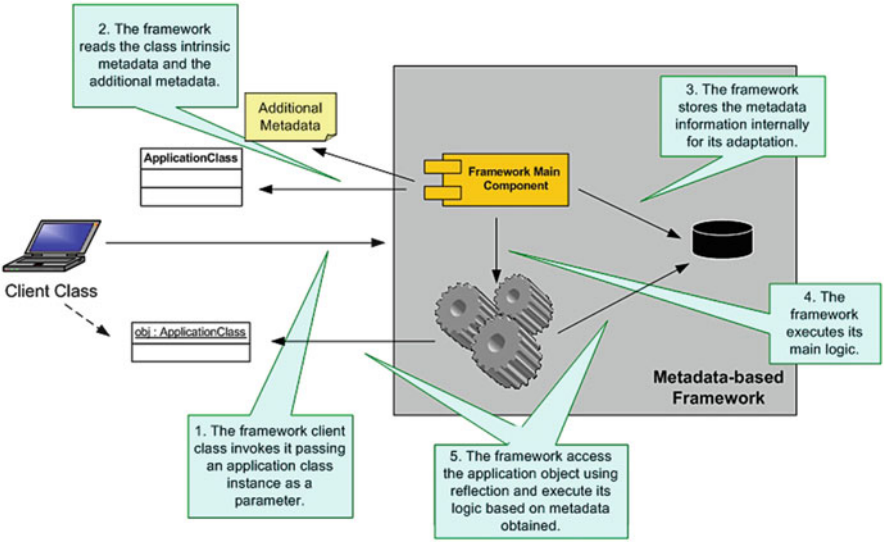


Fig. 1.1 Basic execution process of a metadata-based framework

Figure 1.1 presents the basic processing steps in a metadata-based framework. It starts with the framework main component being called by the application and passing an application object as a parameter. It is important to notice that this first step can be triggered more transparently using aspects or dynamic proxies. Then, the framework reads class-intrinsic metadata using introspection and additional metadata, like in annotations or in XML files. Cached information can also be used to avoid unnecessary readings.

This information is somehow stored inside the framework for a further use. It can store the meta-information or create and configure hook classes based on them. After that, the framework calls its main logic, which uses the read metadata to adapt its behaviour and introspection to access and modify the application object. Not all the metadata-based frameworks follow exactly this process, but it captures a good abstraction of how they work.

In the metadata-based approach, the metadata can be considered as one kind of hot spot since the framework changes its behaviour based on it. Usually a framework has only one defined behaviour for each instantiation; however, using metadata it can have a distinct behaviour for each application class received. Internally, the framework can use the other presented techniques for behaviour adaptation, but it configures them based on each class metadata. One advantage of this approach is to allow a granular and manageable configuration of the framework variabilities.

An important drawback of such kind of framework is the indirection caused by the usage of metadata. Since the behaviour is generated by metadata configuration, an error or inconsistence in class metadata can cause an unexpected result. Since metadata has a declarative nature, this error is hard to debug and find if the framework does not provide a comprehensive error message.

1.3 Cloud Framework Examples

The goal of this section is to present some examples of metadata-based frameworks that can be applied to cloud architectures to exemplify the usage of metadata for such frameworks. It is not in the scope of this chapter to perform a comparative study about these frameworks or to present all their features.

Among many metadata-based frameworks and APIs designed for regular enterprise applications, there are some that are not supported by GAE. However, there are others that are supported with restrictions, such as JPA [40], and those that work fully without any change, such as Guice [8].

Because of the wide use of GAE, frameworks for the exclusive use of this platform raised, such as Gaelyk and Objectify. But there is a trend of creating frameworks that abstracts the particularities of a cloud architecture and work also in other environments, such as Play Framework. The following sections present some existing metadata-based frameworks used for cloud architectures, focusing on how metadata is used in the client code and how they are consumed.

1.3.1 Gaelyk

Gaelyk Framework [5] is a Groovy Web Framework for GAE, which makes use of metadata to map classes to persistent entities and inject various services in client classes. Listing 1.1 illustrates the usage of annotation @GaelykBinding, which indicates this class should be injected by the framework.

The framework uses dynamic features of the Groovy Language to get all classes with @GaelykBinding and, at compile time, injects GAE services, such as DatastoreService, MemcacheService and MailService. Gaelyk implements Active Record Pattern [29], which adds the data access logic in the domain object. As it is shown in Listing 1.2, for an ordinary class to become a persistent entity, it needs to add annotations in the client class and in their fields, identifying keys, not indexed and transient fields. The @Entity annotation is consumed by the compiler, which injects CRUD methods, such as save, delete, find and others. When some of these methods are called, the framework uses the

```
//One needs only use this annotation
//to Inject GAE services
@GaelykBindings
class WeblogService {
   def numberOfComments(post) {
      // the datastore service is available
      datastore.execute {
         select count from comments where postId == post.id
      }
   }
}
```

Listing 1.1 Usage example of @GaelykBinding

```
@Entity
class Person {
   @Key String login
   String firstName
   String lastName
   @Unindexed String bio
   @Ignore String getFullName() {"$firstName $lastName"}
}
```

Listing 1.2 Example of how to define an entity using Gaelyk

```
public class Address {
   @Id Long id;
   String street;
   String city;
}

public class Person {
   @Id Long id;
   String name;
   @NotSaved String street;
   @NotSaved String city;
   Key<Address> address;

   @PrePersist
   void onSave(Objectify ofy){
      if (this.street != null || this.city != null){
         this.address =
           ofy.put(new Address(this.street, this.city));
      }
   }
}
```

Listing 1.3 Example of how to define an entity using Objectify

field annotations to convert the object into a Google DataService entity. Only after this conversion, the persistence operation is really executed.

1.3.2 Objectify

An alternative to develop a persistence layer at GAE is the framework Objectify [6], which implements a metadata mapping between classes and the GAE persistent storage. Its main differential is that it covers all the Google DataService features. As previous example, it is necessary to add annotations in client code to identify fields with a distinct behaviour from default. Listing 1.3 shows an example of entities defined with Objectify. In this example, it is possible to observe that annotations are used to indicate how the framework should handle each field on persistence operations. It is also possible to define callback methods, which are called at certain times by the framework. In Listing 1.3, the method with the @PrePersist annotation will be called before it is saved on the data storage.

```
@OnApplicationStart
public class Initializer extends Job {
   public void doJob() {
      //do something
   }
}

@On("0 0 6 * * ?")
public class DailyReportJob extends Job {
   public void doJob() {
      //create a report
   }
}
```

Listing 1.4 Example of how to schedule jobs in Play Framework

To configure a class as an entity, it is not necessary to add an annotation in the class, nor to configure the class name in an XML file. On Objectify the class needs to be registered previously to its usage. This registration can be done by the invocation of the method register() in the class ObjectifyService.

When the application code registers a persistent class, ObjectifyService reads all annotations and stores them in memory. When a CRUD method is called, the framework uses the metadata to convert the client entity into a data store entity. After that, Objectify invokes the Google DataService methods passing the parameters according to the metadata retrieved.

1.3.3 Play Framework

Play Framework [7] is Java and Scala [30] web framework that enables to deploy applications on the cloud application platforms Heroku [31] and GAE. This framework abstracts the characteristics of the cloud environment, allowing the application to be deployed also on dedicated servers. Regardless of the deployment option, it provides a single way to schedule asynchronous jobs with annotations. Listing 1.4 shows examples of how to do that. For instance, to schedule a job to run at start time, it is necessary to mark the class with @OnApplicationStart annotation. It is also possible to schedule a job to run at a specific instance, like the example presented in Listing 1.4 that creates a daily report at 6:00 AM.

1.3.4 Miscellaneous

The examples of metadata-based framework presented in the previous sections have focused at persistence, dependence injection and scheduling in framework designed specifically to execute in cloud architectures. This section enumerates some other examples that can be applied to cloud application. Jersey [32], for instance, can be used to map using metadata class methods to restful web services that can be accessed

remotely. On the web tier, the framework VRaptor [33] can be used in the development of web controllers, using annotations to determine which requests each method should handle. Hibernate Validator [34] uses metadata, defined as annotations or XML documents, to define constraints to validate application class instances. At last, JAXB [35] is an API which maps application classes to XML documents, also using metadata.

Finally, it is important to emphasize that the main goal of the Java EE 7 specification [36], which is in development, is to allow enterprise applications to be deployed in dedicated servers or in cloud environments. It is for this reason it is possible to speculate that some specifications that already integrate the stack will be adjusted, and others will be created in order to support the cloud requirements. It is expected that applications that follow the standard should be able to be ported between different application servers and cloud providers. Since the current Java EE specification provides a metadata-based API, the techniques presented in this chapter will be very important to develop its features for cloud providers.

1.4 Internal Structure

The internal architecture of a framework is composed of hot spots and frozen spots which respectively represent points with fixed functionality and points where behaviour can be extended and adapted. The potential of reuse of a framework is directly related to its capacity to adapt to different requirements and applications. This is achieved by providing hot spots at the right places, allowing the application to extend its behaviour when necessary.

In frameworks that aim to provide functionality for cloud-based applications, the flexibility requirements can cover different kinds of needs. The first one is to enable each application to adapt the framework behaviour to its needs, considering that many applications will share the same resources. The second is to enable an application to be built independently from the cloud providers, allowing each one to adapt the implementation according to its infrastructure. And finally, the third is to enable the evolution of the cloud services without affecting the deployed applications.

This section is based on a pattern language that studied several metadata-based frameworks and identified recurrent solutions on them [1]. The practices presented focus mainly on metadata reading and processing, providing alternatives to extend behaviour on each mechanism. It is important to state that, despite all practices can be used successfully on the same framework, they should be introduced according to the framework needs.

1.4.1 Metadata Reading and Processing Decoupling

Some metadata-based frameworks consume metadata and execute its behaviour at the same time. The coupling between these two concerns can prevent the introduction of extensions on both mechanisms. So, when designing this kind of framework,

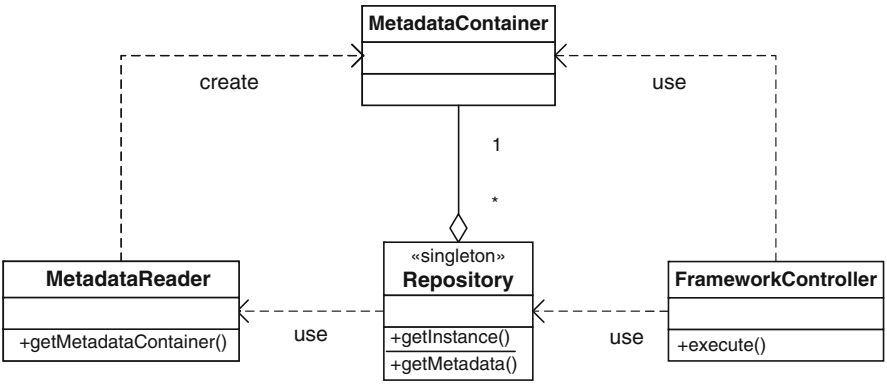


Fig. 1.2 Decoupling between metadata reading and processing

the first requirement to be considered is the decoupling between metadata reading and execution. To achieve this, a solution can be the introduction of a class that represents metadata at runtime and is used to exchange data between these two mechanisms. A representation of this solution is presented in Fig. 1.2.

The MetadataReader is the class responsible to read metadata wherever it is defined and to create an instance of MetadataContainer representing metadata. Further, the MetadataContainer is accessed by the FrameworkController, which in this scenario has the role to receive the framework client calls and execute the main functionality. The MetadataContainer became the protocol between the other components, allowing their decoupling.

This decoupling is also important to allow the MetadataContainer to be stored and reused, avoiding unnecessary metadata readings. In Fig. 1.2 this solution is represented by the class Repository, which can intermediate the communication between FrameworkController and MetadataReader. This class can create a cache of the instances of MetadataContainer already retrieved, improving framework performance after the first call. The Repository can also be a central component where metadata can be retrieved easily by all the framework components.

1.4.2 Flexibility on Metadata Reading

By the introduction of a component responsible to read metadata, it is possible to apply solutions to this mechanism transparently from the other parts of the framework. As presented previously in this chapter, there are several ways to define metadata, such as code conventions, code annotations and external sources (XML documents, databases). Depending on the application requirements, a different metadata definition strategy can be more suitable. For instance, the usage of code annotations are less verbose and closer to the source code, but an external source should be used when you need to be able to change configurations at deploy time without recompiling the code.

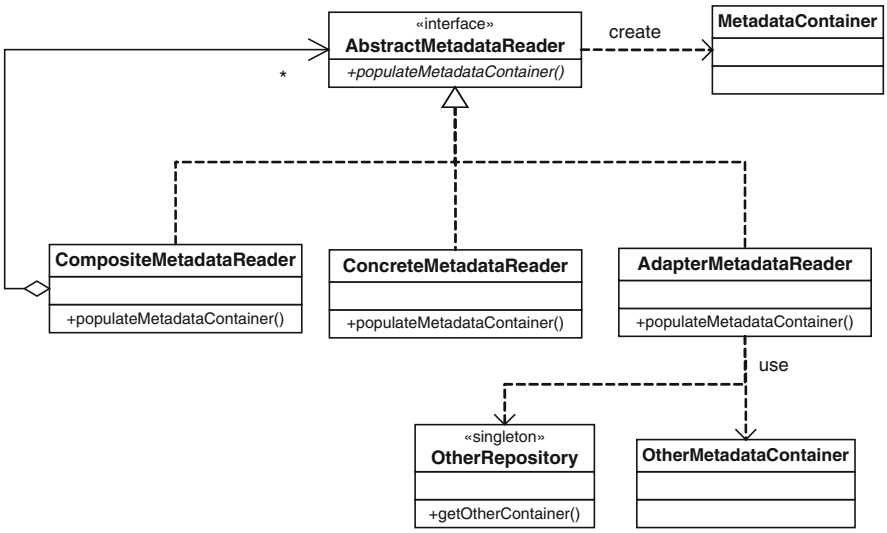


Fig. 1.3 Providing flexibility on metadata reading

To allow the flexibility on metadata reading, the metadata reader can be defined by an interface which can have more than one implementation. Each implementation can provide logic to read metadata from a different kind of source or with a distinct data schema. The structure of this solution is presented in Fig. 1.3. The interface AbstractMetadataReader represents an abstraction of a metadata reader component, and the ConcreteMetadataReader represents an implementation of it.

Based on the presented structure, it is also possible to create a chain of metadata readers, enabling more than one metadata source to be considered at the same time. In Fig. 1.3, the pattern Composite [27] is used on the class CompositeMetadataReader to implement this reading sequence. A Chain of Responsibility [27] is another option for this implementation. That solution enables the introduction of metadata readers that reads only a partial portion of metadata. This enables the application to create metadata readers that can consider domain-specific code conventions to infer part of the information. That also allows the existence of classes like the AdapterMetadataReader, which obtain metadata from the Repository of other frameworks, avoiding metadata redundancy and duplication.

1.4.3 Metadata Schema Extension

An important flexibility requirement that a metadata-based framework can have is to enable the extension of the metadata schema associated to an extension on the framework behaviour. In other words, the application should be able to create new types of metadata elements and to execute application classes when that

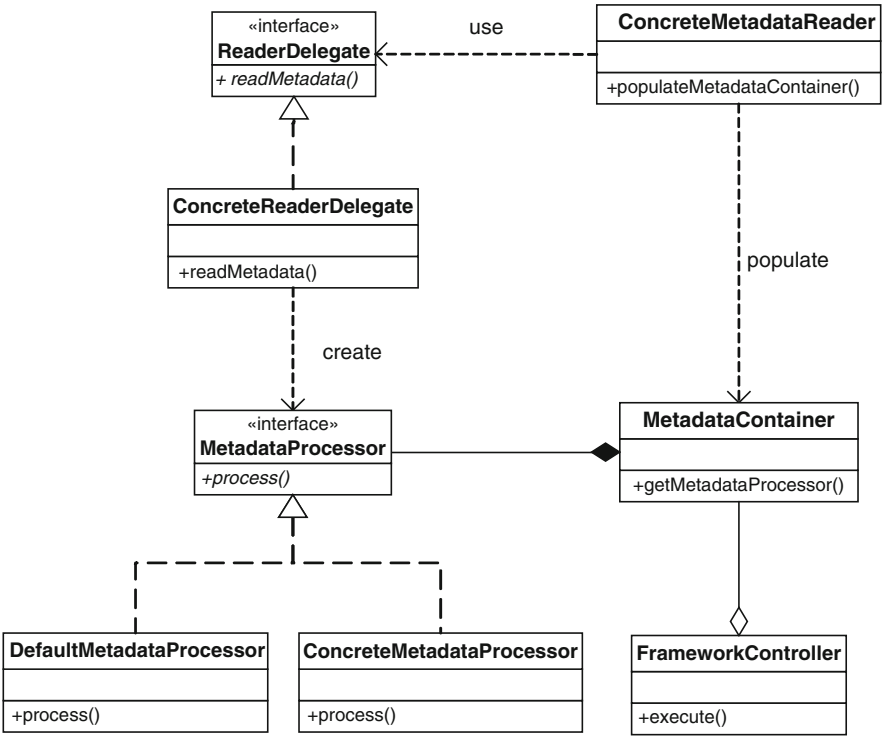


Fig. 1.4 Framework structure to enable metadata extension

```
@AssociatedDelegateReader(AnnotationReader.class)
public @interface ApplicationAnnotation{
    String value();
}
```

Listing 1.5 Example of association between annotation and ReaderDelegate

piece of metadata is processed by the framework. The structure necessary to enable metadata extension as a hot spot is presented in Fig. 1.4.

During the metadata reading process, a ConcreteMetadataReader should delegate the metadata reading of each piece of metadata to an associated class, which in this work is called ReaderDelegate. The concept of metadata piece can vary with the context and with the metadata definition strategy. For instance, if code annotations are being used, the metadata piece can be a single annotation. As another example, if metadata is defined in an XML document, a metadata piece can be an XML element.

Listing 1.5 presents an example of how an annotation can be associated to its delegate metadata reader. A framework annotation, in this example @AssociatedDelegateReader, can be used to define the ReaderDelegate implementation which

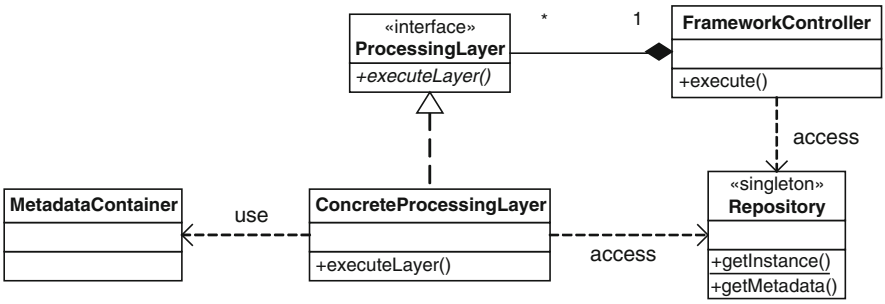


Fig. 1.5 Metadata processing layer structure

should be used to interpret the metadata. To find these custom annotations, the ConcreteMetadataReader should search on all class annotations searching for the ones annotated with @AssociatedDelegateReader. Then, an instance of the associated delegate reader class should be created and used to interpret the annotation.

As a result, the DelegateReader should return an instance responsible to execute the behaviour associated with that piece of metadata, which is called a MetadataProcessor. This MetadataProcessor is added to the MetadataContainer associated to its respective code element. During the metadata processing, part of the framework execution is delegated to the Metadata Processor.

So, based on this solution, an application which needs to extend metadata should define the new metadata type, a metadata reader delegate and a Metadata Processor. The created metadata type should be associated to the metadata reader delegate, which should return the Metadata Processor as the result of the reading.

1.4.4 Metadata Processing Layers

The behaviour extension by defining new metadata types can be appropriate for some scenarios, but in other situations it can be necessary to add application-specific logic on the entire metadata processing. There are also some framework domains in which it is hard to isolate the processing for each piece of metadata. In these cases, it is important to provide an extension point that can interfere with the whole metadata processing.

A solution to this issue found for some frameworks is to divide the processing logic on different layers. Each layer is responsible for part of the framework logic, and the FrameworkController is responsible to coordinate their execution. The solution is represented in the diagram in Fig. 1.5. The interface ProcessingLayer should be implemented by the ConcreteProcessingLayers and represent a framework extension point. Following this structure, new layers with application-specific logic can be easily introduced, and their execution order can also be customized.

1.5 Architectural Scenarios

As presented in the previous section of this chapter, there are several frameworks for cloud architectures which use the metadata-based approach. This section presents some architectural scenarios where the usage of metadata as a hot spot is a suitable solution. Some of the scenarios presented here are based on documented architectural patterns for metadata-based frameworks [3], but contextualized for cloud environments.

The uses presented here are not only based on the existing frameworks designed for cloud architecture but also for other kind of software. Even if the usage of some of these solutions is restricted to the cloud environment, their successful usage in similar scenarios but on other reference architectures can indicate a potential usage in cloud frameworks.

1.5.1 Dependency Injection Indication

Dependency injection [37] is a pattern where the object dependencies are injected externally by some class that creates the instance or manages its life cycle. Common ways of dependency injection are by constructor, by assessor or by an interface method. This practice decouples the class for its dependence, since the concrete instance that is injected is defined and created externally.

Metadata can be used to indicate which fields should be injected in a class. Additionally, metadata can also be used to indicate characteristics of the instance that should be injected. Spring framework [38] uses metadata defined in an XML file to inject the dependencies; however, new versions also support annotation-based injection. Java EE specification [39] also uses annotations to indicate the fields which should be used for injection. An example of the cloud environment is Gaelyk [5], which uses annotations to indicate that a class should receive instances that represent the GAE services.

The decoupling is a consequence of dependency injection that is enhanced by the usage of annotations, which defines which fields should be injected. That strategy can be very useful for cloud frameworks to enable different ways for creating and making available to the application classes the services provided by the cloud, such as for sending emails and connecting to data stores.

As a consequence, the provider can have the same application deployed in different environments and inject in its classes different implementations of the same service. Additionally, the cloud provider can evolve the implementation which is injected in the application class without impacting on it. In brief, this practice allows the cloud framework to use different strategies to create and handle the life cycle of its services instances.

Figure 1.6 illustrates how the metadata-based dependency injection happens. The cloud service should provide an interface which should be referenced by the components in the application. The fields which should receive the service injection

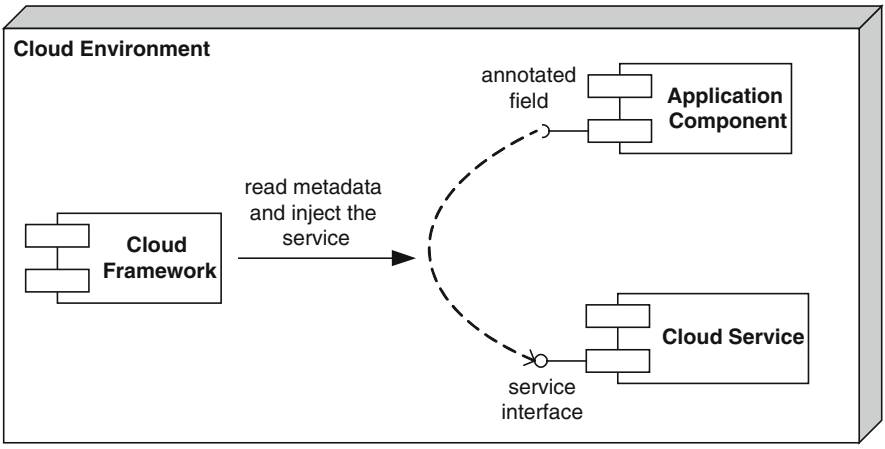


Fig. 1.6 Dependency injection based on metadata

use metadata to indicate that to the framework. In turn, the framework reads the application class metadata, obtains a cloud service instance by the most appropriate way and injects it in the application class instances in their initialization.

1.5.2 Entity Mapping

It is very common for applications to have different representations of the same domain entity. For instance, a business concept represented as a class at runtime can be persisted in a database, presented in a user interface or sent to other applications on web services. The conversion between the different entity representations can generate a very repetitive and error-prone code for the several entities of the system. This practice also couples the application to the other specific representation, for instance, a database schema or an XML format.

Metadata can be associated with one representation of an entity, configuring how it can be mapped to another representation. It should add information about how each characteristic is mapped and converted to the other representation. The most common use is to map between classes and databases, following the pattern metadata mapping [29]. Examples of cloud frameworks which use this kind of mapping are Gaelyk [5] and Objectify [6]. The framework Spring Data [41] also proposes the mapping between interface methods to database queries using code conventions.

It is important to state that this solution is not exclusive for mapping to persistence storages. For instance, when mapping to a web service, a method could be mapped to a service and an entity could be mapped to a parameter.

Cloud providers usually support persistence by using nonrelational databases. The mechanisms of such storages may be different according to different kinds of

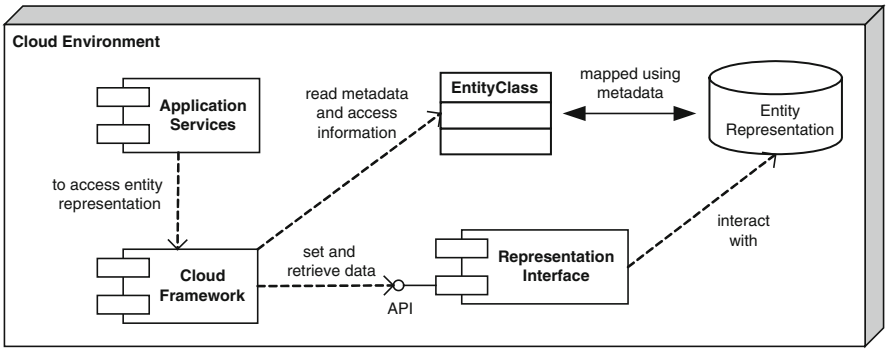


Fig. 1.7 Metadata-based entity mapping

parameters. Metadata can configure the persistent classes with constraints about how it should be persisted, such as which fields are unique identifications, which should not be persisted and even the data format that each one should be stored. That mapping can help in the decoupling between the cloud application and how the data is actually stored. As a consequence, it can improve application portability among cloud providers.

Figure 1.7 depicts the characteristic of an architecture that uses a metadata-based entity mapping. The class that represents an entity in the application should be configured with custom metadata, which maps it to the other representation. When the application components need to access the information from the other representation, they invoke functionality in an API provided by the framework. The EntityClass is used by the application to interact with the framework. After, the framework should access the API from a component which interacts with the other representation, making calls according to the entity metadata to set and retrieve information. For instance, the component called Representation Interface could be a native driver to access a database.

1.5.3 Configured Callback Methods

When an application executes embedded on a server, sometimes it needs to know about events that happen on the server. A design pattern appropriate for this scenario is Observer [27], in which an application class needs to realize an interface and receives invocations on its methods when the events happen. That can be a solution to enable application classes to receive notifications from events that happen on the cloud provider side. The problem of this approach is that the application needs to be coupled with the cloud framework interface, and consequently with the parameter types of its methods. That is specially a problem when this interface needs to evolve and when the application needs to be adapted to different kinds of cloud environments.

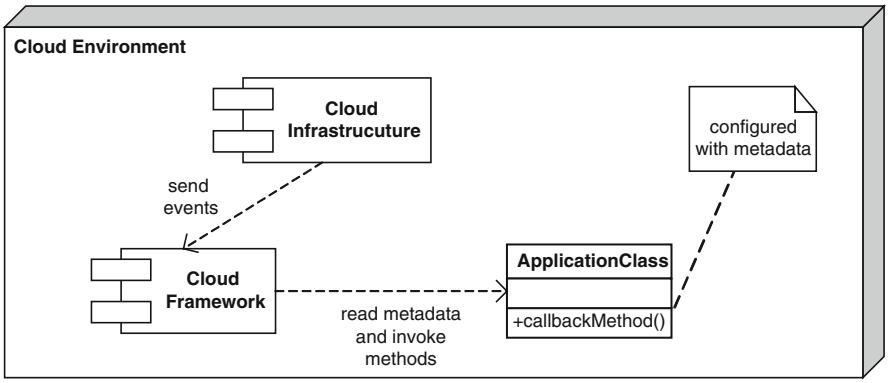


Fig. 1.8 Invoking callback methods configured with metadata

Another common use of metadata is to indicate callback methods in the application classes. By using an annotation or other kind of metadata, the application class indicates which methods should be invoked, instead of implementing an interface. Then the framework looks into the class methods and invoked the configured ones when appropriate.

A very common use of this solution is in application frameworks that handle HTTP requests, such as JSF [42] and VRaptor [33], which were initially designed for regular web applications but can also be used in cloud architectures. This solution is also applied on persistence frameworks, such as Objectify [6], to callback application classes before or after persistence operations, such as saving or loading. Play Framework [7] uses annotations in classes to execute them when the application starts or on scheduled jobs. The scheduling specification is an instance of how metadata can define fine-grained conditions for method execution.

Usually cloud applications abstract the environment in which it is deployed, since it does not have much knowledge about where it is located. For instance, it can use the data storage service available on the cloud provider without actually knowing what the implementation is used. However, sometimes it is important for the application to know when some events happen on the server. For instance, the events can be related to persistence operations, like loading and persisting, or to session migration, like when a session is transferred among servers.

By using a metadata-based approach, the application classes became decoupled from the framework interfaces and only need to handle necessary events. That increases the application portability and even allows the cloud provider to evolve its event model without breaking existing applications. The usage of metadata also allows the addition of constraints, which enables a granular model for event handling. For instance, suppose that a method should be invoked before the migration of a session to another server, the addition of constraints in the metadata can configure for that method to be invoked only when the session attribute "logged" is true.

Figure 1.8 illustrates the structure of this solution. The framework should read the metadata from the application classes and identify which methods should be

invoked and in what conditions. After that, the framework should observe the cloud infrastructure events and delegate the execution to the application class method when appropriate.

1.5.4 Other Architectural Uses

The previous sections presented the uses of metadata for frameworks developed specifically for cloud architectures. However, there are other scenarios where metadata can be applied that are found more often on general purpose framework or on the ones design for other kind of architecture. The goal of this section is to describe more briefly these other scenarios, which can potentially be applied in cloud frameworks in the future.

Sometimes, requirements demand that the application iterates through the fields of some classes, executing some logic for each one. In this context, metadata can be applied to allow the creation of a general algorithm, which processes each field according to the metadata found for each one. Examples of this are Hibernate Validator [34], which validates the constraints of each field from an instance, and Esfinge Comparison [43], which compares all the fields from two instances of the same class. Metadata is used respectively in each framework to define each field constraint and to define the comparison criteria.

On server-based applications, it is common for a class to be managed by the server, having its life cycle controlled by it. In these scenarios, usually it is used as a proxy, a decorator [27] or even an aspect to intercept the method invocation and transparently add some kind of functionality. Accordingly, metadata can be used to configure parameters about the functionality that should be executed. For instance, in Java EE standard [36], annotations are used to configure constraints about security and transaction management. For cloud providers, for instance, metadata could configure a method-level cache which should use the memory cache available as a cloud service.

Finally, metadata can also be used in application classes to enable an automatic generation of customized graphical interfaces. Metadata can be used to define constraints about how each field should be represented on screen. This use is appropriate when the service model is "application as a service" and the user is able to edit the domain entities structure. This is often enabled in dynamic languages and on Adaptive Object Model architectural style [44]. SwingBean [45] is an example of a framework that uses metadata to generate tables and forms.

1.6 Final Considerations

One cloud computing service model is known as platform as a service. In this model, it is provided a computing platform and a solution stack where applications should be deployed. The cloud provider should have available tools, libraries and

frameworks, which should be used by the applications to access services and resources.

This chapter presents how metadata-based frameworks can be used in the construction of cloud-based applications, helping to decouple the application from cloud provider details. Many examples of cloud framework which use metadata were presented, along with some details about how metadata is consumed and how it is used. A set of practices to develop the internal structure of this kind of framework were also presented, in order to introduce the main kinds of hot spots which can be provided. At last, the chapter presented some architectural scenarios in which the usage of metadata is suitable to.

Despite the metadata approach used in several frameworks designed for the cloud, there are many applications that can still be explored. The goal to make transparent for the application the environment where it is deployed is far from being reached. However, some advances were made, like the some features from Play Framework [7]. The new standard for Java enterprise applications [36], which when this chapter was written was a work in progress, represents another effort to achieve this goal. In this context, to use metadata to define framework hot spots can be a good strategy to achieve the decoupling needed for this portability.

References

1. Guerra, E., Souza, J., Fernandes, C.: A pattern language for metadata-based frameworks. In: Proceedings of the 16th Conference on Patterns Languages of Programs, Chicago, 28–30 August 2009
2. Guerra, E.: A conceptual model for metadata-based frameworks. Ph.D. thesis, Aeronautical Institute of Technology, São José dos Campos (2010)
3. Guerra, E., Fernandes, C., Silveira, F.: Architectural patterns for metadata-based frameworks usage. In: Proceedings of the 17th Conference on Pattern Languages of Programs, Reno, 16–18 October 2010
4. Fernandes, C., Guerra, E., Nakao, E., Ribeiro, D.: XML, annotations and database: a comparative study of metadata definition strategies for frameworks. In: XML: Aplicações e Tecnologias Associadas (XATA 2010), Vila do Conde, 19–20 May 2010
5. Zahariev, A.: Google app engine. In: Seminar on Internetworking, Espoo, 27 April 2009
6. Google: Objectify Framework. http://code.google.com/p/objectify-appengine/ (2012). Accessed 9 June 2012
7. Reelsen, A.: Play Framework Cookbook. Packt Publishing, Birmingham (2011)
8. Venbrabant, R.: Google Guice: Agile Lightweight Dependency Injection Framework. Apress, New York (2008)
9. Johnson, R., Foote, R.: Designing reusable classes. J. Object-Oriented Program **1**(2), 22–35 (1988)
10. Jacobsen, E., Nowack, P.: Frameworks and patterns: architectural abstractions. In: Fayad, M., Schmidt, D., Johnson, R. (eds.) Building Application Frameworks: Object-Oriented Foundations of Frameworks Design. Wiley, New York (1999)
11. Fayad, M., Schmidt, D., Johnson, R.: Application frameworks. In: Fayad, M., Schmidt, D., Johnson, R. (eds.) Building Application Frameworks: Object-Oriented Foundations of Frameworks Design. Wiley, New York (1999)
12. Bosch, J., et al.: Framework problems and experiences. In: Fayad, M., Schmidt, D., Johnson, R. (eds.) Building Application Frameworks: Object-Oriented Foundations of Frameworks Design. Wiley, New York (1999)

13. Pree, W.: Design Patterns for Object-Oriented Software Development. Addison Wesley, Reading (1995)
14. Foote, B., Yoder, J.: Evolution, architecture, and metamorphosis (Chap. 13). In: Vlissides, J., Coplien, J., Kerth, N. (eds.) Pattern Languages of Program Design 2, pp. 295–314. Addison-Wesley Longman, Boston (1996)
15. Damyanov, I., Holmes, N.: Metadata driven code generation using .NET framework. In: International Conference on Computer Systems and Technologies, 5, 2004, Rousse. pp. 1–6 (2004)
16. Quinonez, J., Tschantz, M., Ernest, M.: Inference of reference immutability. In: 22nd European Conference on Object-Oriented Programming, 2008, Paphos. pp. 616–641 (2008)
17. Ernest, M.: Type annotations specification (JSR 308). http://types.cs.washington.edu/jsr308/specification/java-annotation-design.pdf (2011). Accessed 15 May 2012
18. Hel, J., Eichhorn, P., Zwitserloot, R., Grootjans, R., Spilker, R., Koning, S.: Project Lombok. http://projectlombok.org/ (2012). Accessed 15 May 2012
19. Fowler, M.: Using metadata. IEEE Softw. **19**(6), 13–17 (2002)
20. Chen, N.: Convention over configuration. http://softwareengineering.vazexqi.com/files/pattern.html (2006). Accessed 15 May 2012
21. Java Community Process: JavaBeans(TM) specification 1.01 Final release. http://download.oracle.com/otndocs/jcp/7224-javabeans-1.01-fr-spec-oth-JSpec/ (1996). Accessed 15 May 2012
22. Massol, V., Husted, T.: JUnit in Action. Manning, Greenwich (2003)
23. Ruby, S., et al.: Agile Web Development with Rails, 3rd edn. Pragmatic Bookshelf, Raleigh (2009)
24. Java Community Process: JSR 175: a metadata facility for the java programming language. http://www.jcp.org/en/jsr/detail?id=175 (2003). Accessed 15 May 2012
25. Miller, J.: Common Language Infrastructure Annotated Standard. Addison-Wesley, Boston (2003)
26. Schwarz, D.: Peeking inside the box: attribute-oriented programming with Java 1.5. http://missingmanuals.com/pub/a/onjava/2004/06/30/insidebox1.html (2004). Accessed 15 May 2012
27. Gamma, E., et al.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading (1994)
28. O'Brien, L.: Design patterns 15 years later: an interview with Erich Gamma, Richard Helm and Ralph Johnson. InformIT. http://www.informit.com/articles/article.aspx?p=1404056 (2009). Accessed 15 May 2012
29. Fowler, M.: Patterns of Enterprise Application Architecture. Addison-Wesley, Boston (2002)
30. Odersky, M. et al.: An overview of the Scala programming language. Technical report IC/2004/640. EPFL, Lausanne (2004)
31. Heroku: Heroku cloud application platform. http://www.heroku.com/ (2012). Accessed 9 June 2012
32. Java.net: Jersey API. http://jersey.java.net/ (2012). Accessed 9 June 2012
33. Freire, A., Silveira, P.: Vraptor – simple and quick web framework. In: Anais do 5o Workshop sobre Software Livre, Porto Alegre, pp. 39–42 (2004)
34. RedHat: Hibernate validator. http://www.hibernate.org/subprojects/validator.html (2012). Accessed 8 June 2012
35. Java Community Process: JSR 222: JavaTM Architecture for XML Binding (JAXB) 2.0. http://jcp.org/en/jsr/detail?id=222 (2009). Accessed 8 June 2012
36. Java Community Process: JSR 342: JavaTM Platform, Enterprise Edition 7 (Java EE 7) specification. http://jcp.org/en/jsr/detail?id=342 (2012). Accessed 8 June 2012
37. Fowler, M.: Inversion of control containers and the dependency injection pattern. http://www.martinfowler.com/articles/injection.html (2004). Accessed 8 June 2012
38. Walls, C., Breidenbach, R.: Spring in Action, 2nd edn. Manning, Greenwich (2007)
39. Java Community Process: JSR 318: Enterprise JavaBeansTM 3.1. http://jcp.org/en/jsr/detail?id=318 (2010). Accessed 8 June 2012
40. Java Community Process: JSR 317: JavaTM Persistence 2.0. http://jcp.org/en/jsr/detail?id=317 (2009). Accessed 8 June 2012
41. Spring Source: Spring projects – Spring data. http://www.springsource.org/spring-data (2012). Accessed 8 June 2012
42. Java Community Process: JSR 344: JavaServerTM Faces 2.2. http://jcp.org/en/jsr/detail?id=344 (2011). Accessed 8 June 2012

43. Esfinge: Esfinge Framework. http://esfinge.sf.net/ (2012). Accessed 15 May 2012
44. Yoder, J., Johnson, R.: The adaptive object-model architectural style. In: Proceedings of the IFIP 17th World Computer Congress – TC2 Stream/3rd IEEE/IFIP Conference on Software Architecture: System Design, Development and Maintenance, Montreal, 25–29 August 2002
45. Sourceforge: SwingBean. http://swingbean.sf.net/ (2012). Accessed 15 May 2012

Chapter 2
Architecting Scientific Data Systems in the Cloud

Daniel Crichton, Chris A. Mattmann, Luca Cinquini, Emily Law, George Chang, Sean Hardman, and Khawaja Shams

Abstract Scientists, educators, decision makers, students, and many others utilize scientific data produced by science instruments. They study our universe, make new discoveries in areas such as weather forecasting and cancer research, and shape policy decisions that impact nations fiscally, socially, economically, and in many other ways. Over the past 20 years or so, the data produced by these scientific instruments have increased in volume, complexity, and resolution, causing traditional computing infrastructures to have difficulties in scaling up to deal with them. This reality has led us, and others, to investigate the applicability of cloud computing to address the scalability challenges. NASA's Jet Propulsion Laboratory (JPL) is at the forefront of transitioning its science applications to the cloud environment. Through the Apache Object Oriented Data Technology (OODT) framework, for NASA's first software released at the open-source Apache Software Foundation (ASF), engineers at JPL have been able to scale the storage and computational aspects of their scientific data systems to the cloud – thus achieving reduced costs and improved performance. In this chapter, we report on the use of Apache OODT for cloud computing, citing several examples in a number of scientific domains. Experience, specific performance, and numbers are also reported. Directions for future work in the area are also suggested.

D. Crichton (✉) • L. Cinquini • E. Law • G. Chang • S. Hardman • K. Shams
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, USA
e-mail: crichton@jpl.nasa.gov; cinquini@jpl.nasa.gov; elaw@jpl.nasa.gov; gchang@jpl.nasa.gov; shardman@jpl.nasa.gov; kshams@jpl.nasa.gov

C.A. Mattmann
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, USA

Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA
e-mail: mattmann@jpl.nasa.gov

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*, Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_2,

Keywords Cloud computing • Distributed computing • Data-intensive systems • Scientific data systems • Object-oriented computing • e-Science

2.1 Introduction

Cloud computing promises substantial, on-demand computing and storage capabilities, scalable by design and by agreed-upon levels of service. Within science, scalability is an important need since the instruments capable of producing massive amounts of data have emerged. These instruments capture a variety of observations generating copious amounts of data, be it remote observations of Mars, in situ ground sensors measuring greenhouse gases, or information related to feedback loops for oil and gas drilling areas. These measurements and the instruments that take them require highly scalable data-intensive systems, which capture, generate, manage, and distribute data to the worldwide scientific community. These same scientific disciplines have routinely resourced their own computing infrastructures including provisioning large computing centers that provide the storage and computational capacity necessary to generate and manage terabytes to petabytes of scientific data collections. The emergence of the cloud promises an alternative methodology for science groups to leverage internally for their scientific data management systems. Leveraging the cloud, however, requires a data management system to abide by a set of key architectural principles whereby the storage and computation can be provisioned remotely within a distributed environment – principles that we will report on in this chapter.

We have researched and demonstrated the use of cloud computing across a number of scientific disciplines using a common software framework known as Apache OODT [1]. These disciplines include planetary and Earth science, lunar science, astronomy, and biomedicine. Employing the cloud within these disciplines presents many questions and subsequent knowledge gained from studying the answers to those questions that must be factored into the decision to use cloud services. For example, whether the cloud is provisioned through an external vendor or resourced through a private cloud – there are many other examples, a relevant subset of which we will describe later.

In this chapter, we discuss the reference data management framework routinely deployed at the NASA's Jet Propulsion Laboratory to support scientific data management for a number of NASA projects. We discuss how the framework can be extended to integrate cloud computing resources. Rather than re-architecting the system, our scientific data management framework is loosely coupled allowing for integrating distributed computing and data storage resources. This architectural notion has been critical in allowing for integration of cloud infrastructures, either from commercial vendors or resourced from internal clouds. We then discuss the practical applications and challenges addressed by cloud computing for these projects and how the framework has been scaled. Finally, we present the outlook for cloud computing at JPL within the scientific discipline.

2.2 The Emerging Science Paradigm: Opportunities and Challenges

Instrument technologies for making science observations have evolved considerably with data sets now in the petabyte range for many of the science disciplines that NASA and other researchers are working with. The scale and complexity of current and future science data changes the nature of data processing, management, sharing, analyzing, and archiving extremely large-scale science data sets. This becomes one of the grand challenges of the twenty-first century. Cloud computing has emerged as one of the decade's chief technologies that can help meet these new compute- and data-intensive science challenges.

Today, within science, there is a trend toward ensuring that high-capacity science data pipelines are established to produce science data products as efficiently and cost effectively as possible. Pipelines are used to take data, acquired from scientific observations, and generate data in structures that are useful for scientific analysis. These pipelines employ complex algorithms that can often be compute-intensive. Many of these pipelines produce intermediate products that must be captured. In addition, reprocessing and rerunning data through pipelines generates newer data. Because of the importance of capturing the provenance of the data, multiple versions of science data are routinely kept. The end result is that massive amounts of data and computation must be made available for these science disciplines. Due to the disparate nature of how these projects are funded, they are often housed and developed in isolation of one another. Local laboratories and ground systems both acquire dedicated systems to support these compute-intensive jobs and data storage needs.

Capturing and archiving data for the long term is also critical for preserving scientific knowledge and distributing that knowledge to the worldwide science community. In many scientific disciplines, archives have been doubling in size about every 5 years. Local laboratories and institutions that have routinely instantiated their own storage to archive data need to scale up to construct petabyte scale archives. In addition, addressing issues of data redundancy, integrity, and security is critical to providing a robust storage infrastructure.

These two scenarios are well suited for the cloud environment. Many of these systems are highly distributed where both storage and computation can be resourced externally. However, the movement of massive data, the provisioning of reliable services, and the policies and security related to access to data are issues that must be resolved when integrating cloud services and technologies into what we are now building as an emerging scientific data ecosystem of distributed software services and data.

One of the most attractive aspects of cloud computing is the elastic nature which allows for services to be provisioned "on demand." Many scientific projects have differing requirements and needs relative to the use of operational capabilities the cloud can offer. Instantiating computation and data internally by each project often leaves quite a bit of untapped computing power. Using Earth scientific research as an example, often data processing and image data tiling for efficient retrieval,

involves advanced computational algorithms and parallelism that demand high-performance computing resources such as supercomputers and computer clusters that are expensive to establish and maintain. Many of these jobs require processing systems that approach the teraflops region to support the computational demands. Within airborne science missions, many of the compute-intensive jobs are only performed on demand since the airborne campaigns occur infrequently. Sharing of computational resources among airborne science missions is an effective way to achieve better economies of scale. The cloud computing model is ideal for such jobs. Likewise, many scientific disciplines have similar needs that can achieve better economies of scale through cloud computing.

In addition to lowering cost and scalable capacity, science data systems can benefit from other services with cloud computing. The cloud provides a virtual environment that allows users to share both science data and services without knowledge, expertise, nor control over the technology infrastructure that supports them. This promotes collaboration of scientific research by making it possible for researchers across laboratories, institutions, and disciplines globally to work together on common projects. Researchers need access to data that can be integrated to generate science products for multidisciplinary experiments. This is especially important in data-intensive science, where the power of discovery lies in applying computational approaches to collection of large data sets that need to be sharable in advancing research.

On the other hand, there are barriers to the success of integration of cloud and science. There are a number of challenges that inhibit the leveraging of the full potential that cloud computing promises. These challenges are related to security, data movement, and reliability. Virtualization, cyber, and data security have always been a key concern to most cloud enterprises, and they are just starting to grasp and not fully understand all the issues. This contributes to skepticism of data integrity and privacy since users often do not trust cloud providers' effectiveness of their security (e.g., hacker attack) and privacy controls (e.g., poor encryption key management, user management).

Another challenge that directly affects science is the ability to move massive data from data providers to the cloud and from the cloud to the users. Science applications (e.g., biomedical research) require frequently upload to or download very large amounts of data to and from the cloud. Often, there are data transfer bottlenecks affecting performance and reliability because of physical networking bandwidth limitation, and no single data movement protocol can support all data movement tasks. Despite the exponential growth in network and hardware speeds, data movement continues to exceed the capability. It is one of the broad areas where research and development are needed to minimize the time spent on data transfer and to bolster the ability to recover from failures and incomplete transfers.

Recent outages at Amazon, Google and Microsoft have shown that cloud services still have glitches and raise reliability concerns. In addition to service disruption, some data were permanently lost. This possibility of data loss and losing access to the data is unsettling for mission-critical and science data, as well as processing

algorithms and research results. It is critical for science data pipelines to take extra measures in order to provide reliable services.

Cloud computing promises to provide more efficiency and reduced expenditure on IT services to providers of science services. It presents potential opportunities for integration of science and the cloud as well as improving science data pipelines even though there are still some challenges to fostering this new science paradigm on the cloud.

2.3 Scaling Scientific Data Systems with Cloud Computing

Moving scientific data systems to the cloud requires adherence to critical architectural principles. We have developed a reusable science data system framework whereby key capabilities including storage and computation are decoupled from the rest of the system. This critical architectural principle has been critical to being able to move these functional capabilities outside the local confines of the system. This architectural framework has been validated through implementation on several different projects and through several different scientific disciplines. While the sciences are different, at the infrastructure level, there are many opportunities to reuse the pattern and to deliver common infrastructure capabilities that are now routinely available in the cloud.

As we have defined reference architectures for our scientific data systems, we have identified a number of key principles that have guided the development of our systems and ultimately a common data management framework. First and foremost is the need to decouple the system into discreet components that can be deployed across highly distributed, multi-institutional environments. This decoupling is key in order to allow us to deploy our infrastructures in various distributed environments. Second is the separation of our data and technology architectures that allows us to instantiate common software for different scientific disciplines. While not directly applicable to cloud computing, it is applicable to developing the common framework discussed below. Third is the need to run as an open source infrastructure. Sharing of software, data, and services is key since many of our science data systems span multiple institutions. While there are other principles, these three are key in considering the implications of deploying scalable science data systems for multiple disciplines.

At the Jet Propulsion Laboratory, we have developed a distributed data management framework called Apache OODT [1] that has been released to the open source community through the Apache Software Foundation. This framework, which implements many of our architectural principles, recognizes first and foremost the need to distribute the processing and storage components of science data systems. As a result, Apache OODT has directly embraced the cloud and demonstrated that systems built using its framework can easily leverage various cloud offerings from storage to computation.

As Fig. 2.1 shows, the Apache OODT scientific data management framework provides a set of services for generation of scientific data processing pipelines and

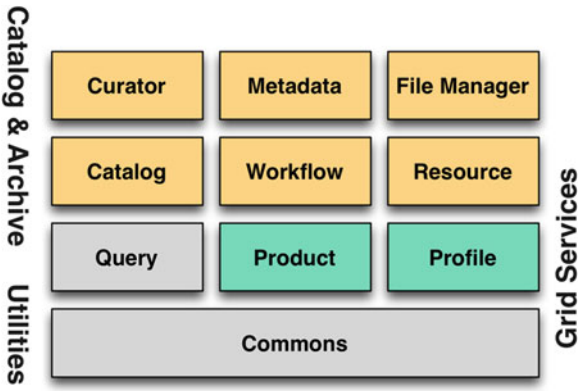


Fig. 2.1 The Apache OODT scientific data management framework

management of science data (grid services shown in the lower right portion of Fig. 2.1, and Catalog/Archive services shown in the upper portion of the figure). Across physical and biological science data systems, there are common patterns that can be applied which makes reuse of a common data management framework, such as Apache OODT, useful for quickly building new data systems. Underlying such systems are the storage and computing services. Bringing in cloud computing allows science systems running the Apache OODT infrastructure to be positioned to scale their storage and computing capabilities.

One of the major components of Apache OODT is the Catalog and Archive Service as shown in Fig. 2.2. It provides a set of processing components that have also been referred to as the Process Control System. The CAS provides a number of services that support ingestion, workflow, catalog management, storage management, and resource management. Each of these is decoupled allowing for the CAS data management system to be one virtual system deployed across many services. The storage and resource management components of OODT are prime candidates to be run on the cloud since the design of the Apache OODT CAS component has allowed for storage and computation of data to be distributed. This is routinely done for processing satellite data from Earth orbiting missions where the processing is done on high-end clusters.

2.4 Comparing Cloud Computing Implementations Across Different Scientific Data Systems

We have developed a number of systems that have now taken advantage of the cloud using Apache OODT. These systems include airborne and spaceborne missions for Earth and planetary environments, the analysis of massive climate science data and models, radio astronomy, and biomedical research. While each of these environments is unique, we have demonstrated that our architectural framework

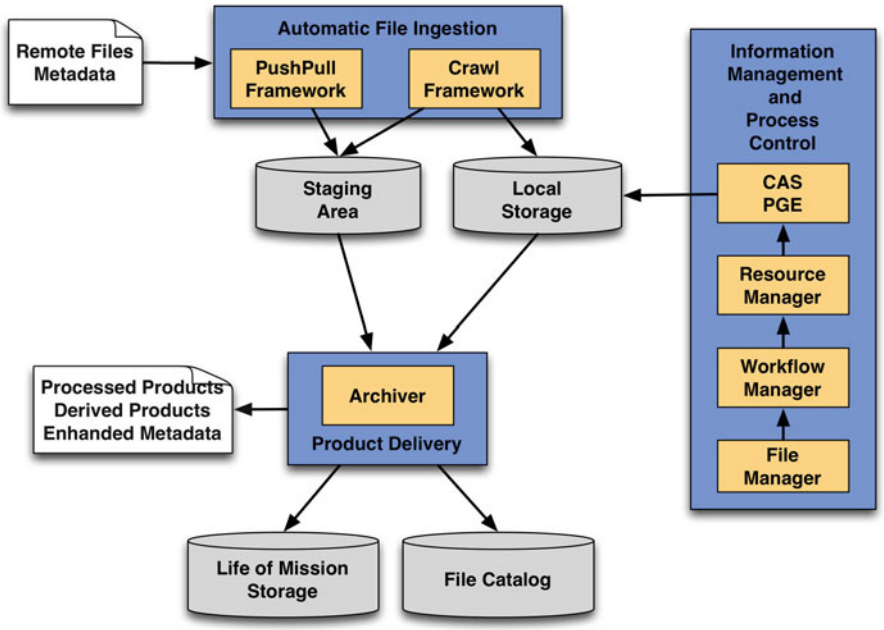


Fig. 2.2 Apache OODT Catalog and Archive Service

coupled with the Apache OODT software has allowed us to quickly construct scalable systems without requiring us to stand up large data centers. As a result, we have been able to respond to the scientific research demands of each of these communities.

2.4.1 Cloud Computing Architecture for DESDynI Earth Science Mission

2.4.1.1 Science Goals and Data Processing Requirements

DESDynI (Deformation, Ecosystem Structure and Dynamics of Ice) is one of four Tier-1 missions planned by NASA as part of the next Earth Science Decadal Survey, a large scientific endeavor directed by the US National Research Council to assess the current state of our planet (geographic realms, climate, ecosystem) and answer key scientific questions about its future. In particular, DESDynI scientific goals include evaluating the stress level of known earthquake faults and active volcanoes, studying how the energy released by such violent events affects the ecosystem and carbon cycle on a large scale, and measuring the change in Arctic and Antarctic ice masses as both a cause and an effect of

climate change. To achieve these goals, DESDynI will employ two instruments (an L-band inSAR and multibeam Lidar) hosted on a single geo-orbiting satellite, taking high-resolution interferometry images that are downloaded in real time to the mission ground data system.

When compared to other NASA remote-sensing missions, two unique aspects characterize DESDynI. First, the sheer volume of expected data output is far larger than anything produced by current Earth observing systems. It is estimated that around 5 TB/day of raw output will be transmitted to the ground data system, which will need to be post-processed through several science algorithms resulting in approximately 44 TB/day of derived data output, organized in 21 proposed scientific products. Second, the DESDynI data products and scientific algorithms will span three major disciplines: solid Earth deformation, ecosystem structure, and dynamics of ice. Because of its multidisciplinary nature, one of the key questions about the mission is whether it makes sense to process and archive the data at a single location or to spread the load over multiple centers that are separately run and controlled by domain experts. Indeed, a distributed architecture might actually not be an option; rather it might be a necessary consequence of the large data volumes involved.

In the 2010–2011 timeframe, the NASA's Jet Propulsion Laboratory (JPL) conducted an internal study aimed at assessing the feasibility of processing the data output expected from DESDynI. The study had two major goals: (1) evaluate the performance and flexibility of the Apache OODT framework for supporting I/O and CPU intensive data processing pipelines and (2) evaluate architectural trade-offs between a central topology (all data processed and archived at a single location) and a distributed topology (data processed and archived at multiple centers). The study leveraged cloud computing as an ideal environment for temporary hosting of data processing services, duplication of identical software stack configurations, and simulation of distributed processing and archiving.

2.4.1.2 Cloud Computing Setup

The study used the Amazon Web Service Elastic Cloud Computing (AWS-EC2) infrastructure together with the Simple Storage Service (S3) to deploy a simulated DESDynI data processing pipeline onto geographically distributed servers (see Fig. 2.3: servers B1 and B2, physically located on the AWS-West (Northern California) region, and server C in the AWS-East (Northern Virginia) region). An Amazon Machine Image (AMI) was built starting from a high-end server configuration that would offer considerable I/O and CPU processing performance (m2.4xlarge type, 68.4 GB of memory, 26 EC2 Compute Units, 1.7 GB of attached local storage with high I/O performance), then augmented with additional software components (OODT and data transfer packages), and deployed onto the cloud servers. This experimental setup was used to conduct two kinds of experiments: measurements of data transfer speed between JPL and cloud servers and comparison of overall data processing time across different topologies.

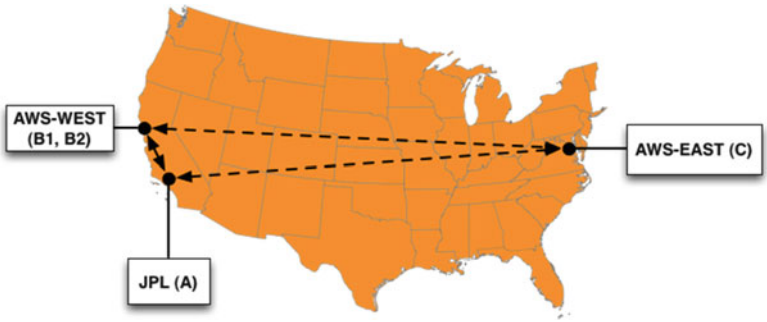


Fig. 2.3 Geographic location of JPL server (*A*) and Amazon EC2 servers (*B1*, *B2*, *C*) used in the simulation study for the DESDynI data processing pipeline

2.4.1.3 Data Transfer Studies

Data transfer performance is a critical factor when considering the deployment of a data-intensive processing pipeline on a distributed topology. In fact, an important part of the DESDynI pre-mission studies consisted in using the deployed array of cloud servers to evaluate available data transfer technologies, both in terms of speed and easiness of installation and configuration options. Several data transfer toolkits were compared: FTP (most popular, used as performance baseline), SCP (ubiquitous, built-in SSH security, potential encryption overhead), GridFTP (parallelized TCP/IP, strong security, but complex installation and configuration), bbFTP (parallelized TCP/IP, easy installation, standalone client/server), and UDT (reliable UDP-based bursting technology). Benchmarking was accomplished by transferring NetCDF files (a highly compressed format commonly used in Earth sciences) of two representative sizes (1 and 10 GB) between JPL, two cloud servers in the AWS-West region, and one cloud server in the AWS-East region. The result was that UDT and GridFTP offered the overall best performance across transfer routes and file sizes: UDT was slightly faster and easier to configure than GridFTP, but it lacked the security features (authentication, encryption, confidentiality, and data integrity) offered by GridFTP. It was also noticed that the measured transferred times varied considerably when repeated for the same file size and transfer end points, most likely because of concurrent use of network and hardware resources by other projects hosted on the same cloud. Additionally, using the Amazon internal network, when transferring data between servers on the same AWS-West region consistently, yielded much better performance than when using the publicly available network between the same servers.

2.4.1.4 Architecture Studies

The second and most important part of the study was to leverage the cloud environment to evaluate different possible architectures for the overall DESDynI data processing pipeline (also know and Process Control System or PCS). To this

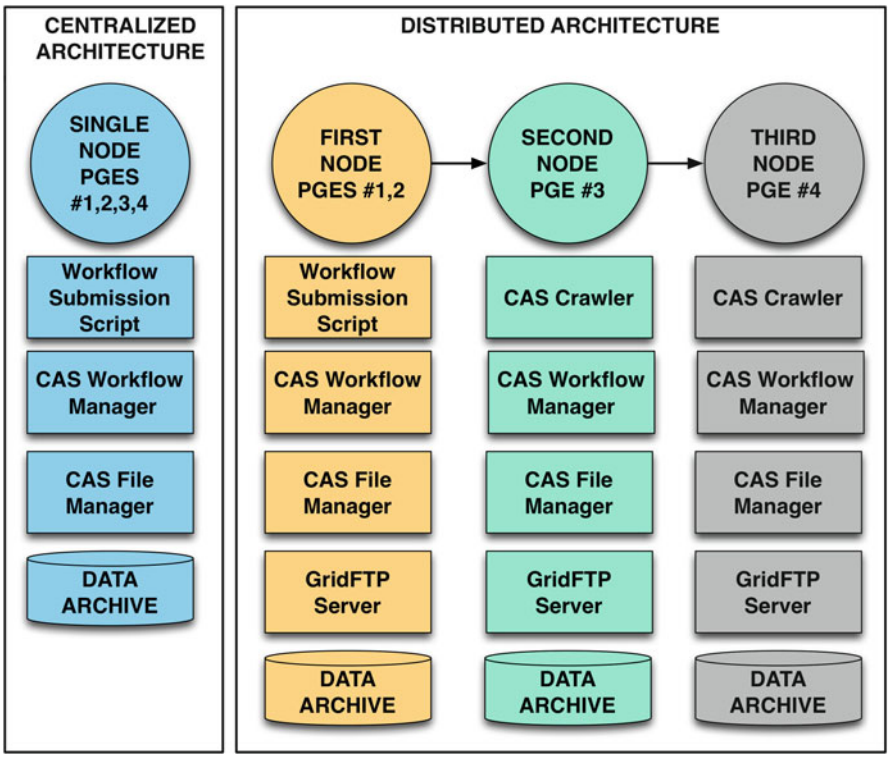


Fig. 2.4 Simulated DESDynI PCS (Process Control System) for the centralized and distributed architecture

purpose, we designed a simulated scientific workflow composed of four tasks (program generation executables or PGEs), designed to closely mimic the actual data processing tasks expected during the mission. Each PGE consisted in reading input files from the previous PGE (or from the simulated ground system feed), running a compute-intensive scientific algorithm, and producing output files for the next PGE. Standard OODT service components (the File Manager, Workflow Manager, Resource Manager, and Crawler) were deployed onto the cloud servers to execute the workflow in two different configurations (see Fig. 2.4):

- In a single-server centralized topology, the workflow was run on a single cloud server. A batch script was charged with submitting the same workflow N times to the Workflow Manager. For each workflow, the Workflow Manager runs the four PGEs sequentially, each time submitting the PGE to the Resource Manager and waiting for that PGE to be completed before archiving it with the File Manager and starting the next PGE.
- In a multi-server distributed topology, the workflow was split such that it run on three identical cloud servers (all deployed in the AWS-West region and connected

by the internal Amazon network). The first two PGEs were run on the first server, after which the output from the second PGE was transferred via GridFTP to the second server. A Crawler demon on the second server monitored the ingest disk area and, when the output was fully transferred, took care of notifying the Workflow Manager on that server to start the third PGE. Similarly, the output from the third PGE was transferred to the third server and triggered the submission of the fourth PGE.

Figure 2.4 shows the standard OODT data processing components that were orchestrated to execute the data processing workflow on the cloud.

The two architectures were compared by measuring the total clock time necessary to execute the same workflow *N* times, thus simulating the continuous data processing occurring during mission execution. It was measured that the distributed topology represented a considerable improvement over the centralized topology, yielding approximately a 30 % reduction in overall execution time. This is obviously because the heavy scientific processing was spread across multiple servers, while the latency caused by moving data between servers and by job monitoring and starting was kept to a minimum, thanks to the choice of high-speed data transfer technology and low overhead of the OODT data management services.

2.4.1.5 Conclusions

The DESDynI pre-mission studies constitute a poster example of how the cloud computing paradigm can be leveraged to design, evaluate, and refine the data processing pipeline for a NASA mission long before it leaves the ground. The cloud offers the ability to configure, save, and duplicate a machine image that contains all the standard service components necessary to design a mission full data processing pipeline, plus additional custom software to generate mission-specific science products. This image can be used to refine and test the science algorithms, evaluate I/O and CPU requirements, and benchmark overall performance versus different possible deployment architectures. The cloud resources need not be allocated in the long term, since it might be more cost-effective to use NASA-owned servers and storage to execute the data processing during mission operations and for long-term archiving purposes. Nevertheless, pre-mission studies are best executed onto temporary resources that allow the mission architects to experiment with different technologies and configurations.

2.4.2 Cloud Computing Architecture for Lunar Mapping and Mission Program

LMMP (Lunar Mapping and Modeling Portal) is tasked by the NASA's Human Exploration and Operations Mission Directorate (HEOMD) to develop a single, common, consistent, uniform, and intuitive NASA web portal for users to access lunar

mapping and modeling products, tools, and data. It provides valuable information to scientists, researchers, and engineers in the planning of future lunar missions, aiding in tasks such as evaluating and selecting potential landing sites, identifying locations to place rovers and other assets, and developing computer systems to navigate the surface and perform advanced scientific analysis. It has a secondary purpose as a tool to support the dissemination of lunar data to future missions and projects, international partners, commercial entities, education, and the general public. The data sources for LMMP range from historical missions, such as Apollo, Lunar Orbiter, Clementine, Lunar Prospector, and Earth-based observations, to the recent Lunar Reconnaissance Orbiter (LRO) and the Lunar Crater Observation and Sensing Satellite (LCROSS) missions. LMMP is currently housing over 2 TB of lunar data, including image mosaics, digital elevation maps, mineralogy maps, lighting models, gravity models, and thermal models.

LMMP contains both publicly released data sets as well as private and embargoed information. When new data is received from spacecraft, it is typically embargoed for several months to allow principal investigators and sponsoring scientists to have the opportunity to perform research and publish findings before the data is disseminated to other scientists and the general public. Despite providing storage for historical as well as these new products in the same infrastructure, LMMP is required to segregate and control access to sensitive data. This was a major consideration in utilizing cloud storage and computing. Our solution was to only place publicly released data sets on the cloud while keeping sensitive data at JPL, protected behind a login-based security model. However, we were able to integrate the cloud-hosted and JPL-hosted data sets together in one unified user interface so that users were not specifically aware of the source of the data, yet would notice more data sets available to them once they logged in.

Once we identified the data that we could put on the cloud, we looked into how we could utilize cloud services for our needs (Fig. 2.5).

2.4.2.1 Data Processing

One of the first uses of cloud computing in LMMP was for creating image tiles from large image mosaics. LMMP archives images and digital elevation maps that are several gigapixels in resolution and gigabytes in size. A global image mosaic from the Clementine spacecraft has a screen resolution of 92,160×46,080 pixels (4 gigapixels). A mosaic from the Apollo 15 metric camera has a file size of 3.3 GB. While modern desktop computers, with some effort, can display these images as a whole in their native resolutions, older machines and resource-constrained devices such as mobile phones and tables do not have the processing power to render the images. If the image is not stored on a local file system, the image data has to be transferred from a remote machine, thus adding a dependency on network bandwidth. To work around these limitations, the image has to be broken up into smaller parts. This process is called image tiling and building a tile pyramid. The goal is to create small image tiles, typically a few hundred pixels high and wide (512×512 in

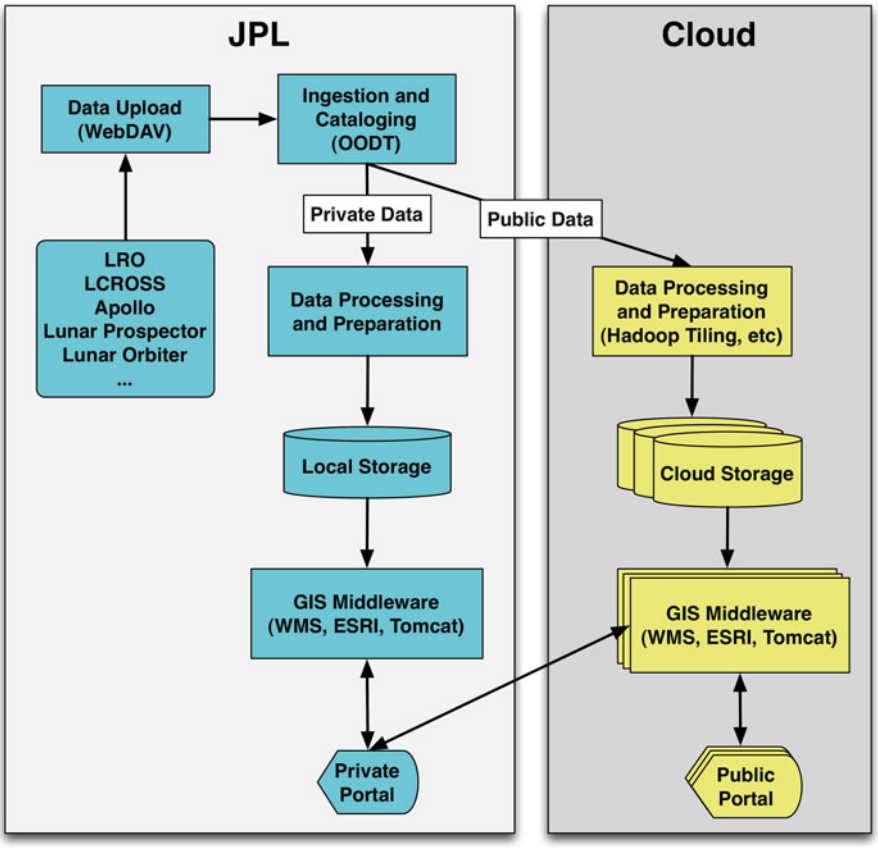


Fig. 2.5 Hybrid use of private and cloud infrastructure in LMMP

LMMP), and to send those tiles to the client as needed depending on the location and the zoom level that the client is viewing. This method of presenting image data is very common in mapping software and is used in many commercial mapping packages.

The process of tiling is straightforward but computationally intensive. It involves subsetting a portion of the original image and performing a bicubic resizing of the subset to a standard tile size. At the lowest level of this tile pyramid, the tiles are (or very close to) a 1:1 mapping to the original image. At each level, the tiles regenerated at ¼ size of the previous tile until the last level, where the image is contained in exactly one tile. Despite the seemingly sequential process, image tiling is incredibly parallelizable and cloud computing enabled us to distribute this image processing across multiple machines.

With the help of the Hadoop framework and the scalability of cloud computing [2], we were able to process and tile large amounts of data in a relatively short amount

of time. We set up and configured Hadoop on several Amazon EC2 nodes and optimized to find the number and type of EC2 machines that would yield the best performance. We tested 20 large instances along with 4 Cluster Compute Instances. Because of the nature of our image processing and the nuances of the Hadoop framework, large amounts of binary data were transferred between the nodes in the cluster. Since the Cluster Compute Instances were tuned for network performance, that smaller cluster yielded better performance and was able to complete the image tiling process faster. However, both solutions performed significantly better than the previous solution of processing the image on a single local machine. The cost for processing each image was only a few dollars.

2.4.2.2 Data Storage, Transfer, and Hosting

In addition to image processing, LMMP heavily utilizes cloud storage and hosting for serving publicly available data. The project is required to handle at least 400 concurrent users using the system. Users must be able to access all of 2 TB of the data currently stored in the system as well as perform real-time data visualization and image manipulation. The cloud allows LMMP to scale up when demand for its tools and data is high and to scale down to minimize cost when demand is low.

Upon receiving data from the data providers, LMMP uses Apache OODT to ingest information about the product. Once the data has been properly catalogued, it can be stored in its original form for on-demand processing, such as rendering 3D animations, or can be tiled and served as static data such as the image tiles mentioned previously. Depending on the type of data, different cloud storage schemes are utilized.

LMMP stores its public data in Amazon‘s bucket store infrastructure, S3, as well as its elastic block storage, EBS, which is attached to an EC2 instance. Based on the features of these two types of storage, we store different types of files. Files stored on S3 are primarily static files that will not be used in subsequent processing such as the individual tiles generated from an image mosaic. S3 files are accessed via URLs, which makes them easily accessible from users online. In addition, we can use Amazon CloudFront to automatically host the data at different data centers around the world to reduce download times for our international users. However, S3 files are not suitable for dynamic computations within EC2. For data sets that will be used to dynamically generate products, we will store them in EBS. EBS is attached to EC2 instances and can be readily accessible to processes using standard file I/O functions

Moving the large amounts of data between JPL and the cloud has been a challenge, and we have developed a specific tool that enables higher throughput by utilizing parallel connections to cloud systems. Rather than transferring files through a single connection, we could transmit multiple files concurrently and even parts of files and assembling the pieces at the destination. We used the partial GET feature in the HTTP protocol and an S3-specific feature for multi-part uploads to maximize bandwidth utilization.

The portal operates on a multi-tiered, heterogeneous architecture consisting of Apache HTTPD, Apache Tomcat Application Server, ESRI ArcGIS Server, and a JPL internal map server. The HTTPD server, running on EC2, distributes the requests to the other servers and returns the responses to the client. The Tomcat server, also running on EC2, performs catalog lookups, 3D animation rendering, sun angle lookups, and dynamic image subsetting. The ESRI ArcGIS server handles the distribution of some images as well as nomenclature data. This server runs on a Windows instance on EC2. Finally, the JPL map server hosts other images and runs within JPL. This architecture, coupled with the scalability of cloud computing, gives fine-grained flexibility in what to scale capacity. If users are rendering multiple 3D animations, multiple Tomcat servers can be instantiated to satiate the demand. If users are requesting more image data, we can instantiate more Windows instances and have multiple ArcGIS servers. We can also run these servers are larger virtual machine instances to handle larger loads.

2.4.2.3 Conclusions

LMMP is an operational example of how a project can utilize cloud computing resources not only for data processing but also as an efficient means for the storage and presentation of the data. The ability to start virtual machines as needed allows for the quick setup and teardown of distributed computing environments, especially with the help of frameworks such as Hadoop. The virtual machine software image can be instantiated on numerous compute "sizes" to emphasize memory capacity, CPU power, and/or network connectivity to maximize algorithm performance while decreasing cost. The cloud also provides a convenient platform to host and serve data to customers, whether they are scientists or the general public. The project can optimize its resource utilization based on actual demand. This frees up resources and allows the project to focus more on its primary goal of providing unique and interesting data.

2.4.3 Cloud Computing Architecture to Support Airborne Science Missions

Airborne missions are geared toward maximizing science with budgets on the order of a few million dollars for the entire mission. The low cost of these missions is appealing to scientists, and they are becoming more popular across NASA. The ACCE (Airborne Cloud Computing Environment) team aims to minimize the cost of development and deployment of a data processing pipeline for these missions by identifying the common patterns, developing a framework to address them, and by executing the computations on the cheapest available cloud computing infrastructure that can sustain the requirements of the mission. ACCE provides an end-to-end science data system for airborne missions by utilizing Apache OODT as its core

framework with extension to cloud infrastructure to support processing as well as access to the collected or processed data to the mobile science team during tactical operations.

The ACCE team has evolved its framework and demonstrated its readiness by initiating work with several airborne missions. For example, the ACCE team has worked closely with the CARVE (Carbon in the Arctic Reservoir Vulnerability Experiment) mission which collects detailed measurements of greenhouse gases on local to regional scales in the Alaskan Arctic and provides an integrated set of data that will allow for unprecedented experimental insights into Arctic carbon cycling. Specifically, ACCE helps CARVE to select the most cost-effective and high-performing cloud solution for processing the Level 2 Full Physics (L2FP) algorithm on the collected data. The L2FP algorithm is a core component across numerous Earth science missions such as Orbiting Carbon Observatory 2 (OCO-2) which studies carbon dioxide in the atmosphere and provides scientists with a better idea of the chemical compound's impacts on climate change. It is a converging algorithm that runs a forward model to predict spectral values produced by an instrument. The forward model searches for and estimates starting parameters and converges when observed values from the instrument are within a certain threshold of predicted values.

The compute-to-storage ratio of L2FP is high: L2FP algorithms are computationally intense, but they don't require or produce a significant amount of data. CARVE is also a classic example of a bursty processing environment: it flies a few times in a year, and scientists are eager to obtain processed data from each flight as soon as it is completed. If CARVE makes an investment in local infrastructure, the resources would be idle for most of the year while the mission awaits the next flights, and the processing would be confined to the speed and throughput of the local capacity. With bursty operations, low volume of data, and computationally intense requirements for rapid data processing, CARVE has chosen to use a commercial cloud solution, Amazon Web Services. This particular selection helps CARVE minimize the risk associated with making infrastructure investments too early, and it is also predicted to save CARVE several hundred thousand dollars over the course of the mission.

2.4.3.1 ACCE Architecture

ACCE is providing CARVE with the interface to schedule and orchestrate jobs on the commercial cloud through Polyphony [3]. For CARVE, Polyphony is a workflow orchestration framework that tightly integrates with Apache OODT to provide scheduling and execution of L2FP jobs in the cloud. Relying heavily on Amazon SWF (Simple Workflow), Polyphony makes no assumption about the cloud in which data are processed. Thus, the ACCE architecture is still not bound to infrastructure on Amazon Web Services, and it is versatile enough to take advantage of another provider if a more cost-effective alternative is available.

The L2FP processing in ACCE employs a poll-based mechanism to schedule jobs on processing nodes. In this architecture, each of the available nodes polls a

collection of distributed workflow systems for scheduled jobs. If a job is available, it is assigned to the polling node for a specific duration. If the node finishes the job during that time, it informs the scheduler of its success. Otherwise, the job times out and is automatically assigned to another node. We employ two types of time-outs: lifetime time-outs and heartbeat time-outs. The lifetime time-out encompasses the total amount of time a node has to finish the job, which can range from a few seconds to a few hours. However, relying on lifetime time-outs in isolation implies that failed nodes would not be detected until the entire workflow times out, which could be hours. Therefore, we added heartbeat time-outs, which are heartbeats that nodes send on each workflow that they are currently processing. The heartbeat time-outs allow us to quickly detect workflows that are not being processed so that they can be scheduled elsewhere. Both of these time-outs are available natively in SWF. Another key benefit of the polling approach is that it allows an elegant and simple approach to expand or shrink the computational capacity with heterogeneous nodes: when more capacity is desired, we simply turn on more nodes that poll the queue, and when we shut nodes down, they simply stop polling. This approach enables utilization of internal infrastructure as well as infrastructure from multiple cloud providers in concert to maximize the throughput of our processing pipeline.

Once data are ingested, ACCE stores them on local machines behind the JPL firewall. To harness the capacity available to a mission in the cloud without opening firewall holes, the data is securely uploaded to a location that is accessible by the cloud servers. As data are available in the local file system, they are uploaded to the Amazon S3 buckets over an optimized channel to maximize throughput. Once data are on S3, they can be accessed by nodes at JPL, nodes across multiple data centers at Amazon, or other cloud providers. Upon receiving a job, a node pulls the data from S3, processes them, and uploads the results back to S3. The workflow concludes with a job by retrieving the result from S3 and placing it back on the local file system at JPL.

2.4.3.2 Conclusions

ACCE is another operational science data system solution that allows any airborne mission to seamlessly utilize cloud computing resources as part of its data migration, processing, and access workflow pipeline. We also believe that the integration of Apache OODT with distributed cloud computing and storage is a generic workflow that is relevant to numerous science data systems that span beyond airborne missions.

2.4.4 Cloud Architecture to Support Cancer Research Studies

The Early Detection Research Network (EDRN) funded by the US National Cancer Institute is comprised of a number of laboratories focused on the discovery of cancer biomarkers as indicators for disease. The EDRN has employed the Apache OODT

infrastructure as its core framework for the deployment of a data management system supported by the EDRN Informatics Center [4] at the Jet Propulsion Laboratory. The system supports the management, sharing, and analysis of data generated during EDRN biomarker studies. The data acquired is processed at the local laboratories. This includes ensuring that the data is high quality and ready for broader research review. Once data has met certain criteria, including allowing ample time for the local laboratory to generate and analyze the data themselves, it is captured and distributed to the general science community.

Cloud computing has been employed by the EDRN in two different paradigms. First, the EDRN has used Amazon S3 services for sharing data via the cloud through a study focused on nonsmokers who have acquired lung cancer. The second is a data processing and computing infrastructure which has been developed for the laboratories using the Apache OODT data processing infrastructure. This is deployed on a private cloud at the Jet Propulsion Laboratory and is provided to the EDRN research community so they do not need to deploy their own infrastructure.

For the nonsmokers study, data was shared as an agreement between the US National Cancer Institute and the Canary Foundation [2]. The agreement was to capture and distribute the data long term to the broad research community through the EDRN infrastructure with the data residing long term in the EDRN archive known as "eCAS." However, the initial capture of data was done using tools provisioned by the Canary Foundation. In support of this, a joint biorepository was developed at Amazon S3. This allowed the Canary Foundation to upload data into an online, accessible storage location, while the EDRN infrastructure was able to pull data into its biorepository using Amazon Web Services.

In addition to pulling data into EDRN from other systems, such as Canary Foundation for the Never Smoker's Lung Cancer study, the EDRN has developed a complete, packaged solution for data management and processing, for each local laboratory. Rather than deploying the system locally, the EDRN is providing a cloud-oriented infrastructure whereby scientists can upload, process, and manage data within their local laboratory. This is an attractive model because it reduces the amount of effort required to use such systems.

JPL, as the Informatics Center for EDRN, has stood up the Apache OODT infrastructure and delivered it as a web-based capability providing the storage, computation, and application hosting as a service known as the LabCAS (Laboratory Catalog and Archive Service). Storage and computation are delivered from an internal cloud shared with a number of projects at JPL in Earth and planetary science. The Apache OODT infrastructure provides the Catalog and Archive Service (CAS) component that has been deployed on top of the cloud infrastructure and delivered specifically for the laboratories enabling LabCAS. Data is uploaded to LabCAS through a variety of means, including WebDAV and iRODS, which provide high-speed data transfer for massive data. Individual workflows on the uploaded can be executed which employ specific algorithms, for such data types as proteomics. The LabCAS, via the Apache OODT CAS infrastructure, will pull the data from its storage repository and execute the algorithms on a scalable compute host employing the same cloud computing capabilities as has been identified by a number of the projects. The adherence

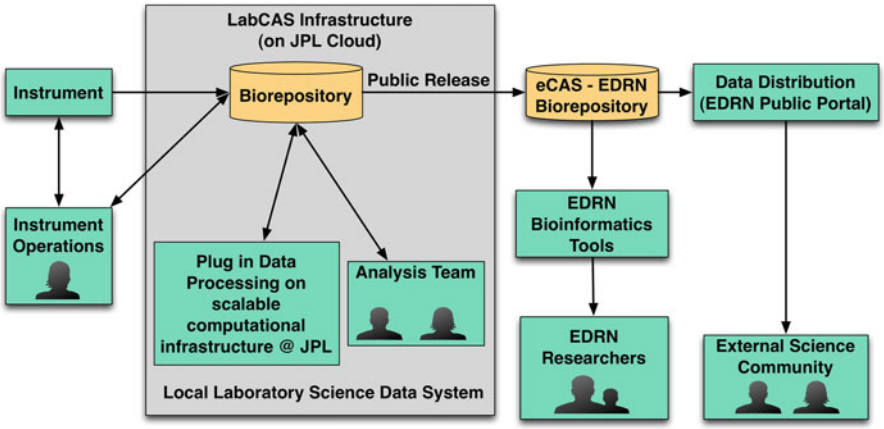


Fig. 2.6 EDRN informatics infrastructure

to the reference architecture has been critical because different storage and computational services can be brought in and swapped out without disturbing the LabCAS system (Fig. 2.6).

2.5 Related Works

Our work is directly related to generalized cloud frameworks that provide the underlying substrate and API to communicate with cloud storage and processing services. In addition, several of the case studies described in this chapter were first presented at a workshop held during the 2011 *International Conference on Software Engineering* (ICSE11) [5] to discuss the software engineering aspects of cloud architectures, frameworks, and the like. Finally, our work is similar to an open source software effort for science data portals (or "gateways") called Airavata [6].

Cloud frameworks are an underlying application programming interface (API) and substrate that arose to deal with the proliferation of cloud providers and technologies that have emerged over the past 5 years, e.g., Amazon (S3 and EC2) [7], Rackspace [8], and Google App Engine [9]. Furthermore, entire collaborative software systems and frameworks have emerged that mitigate the differences between the cloud providers – OpenStack [10] and CloudStack [11] are recent examples of this trend – and a small representative subset of the larger number of these frameworks that exist.

Though our early experience has focused primarily on the use of the Amazon provider's cloud services (EC2, S3, EMR [12]), we have tried as best as possible to architect our solutions to be cloud provider neutral. In certain cases, we have also designed our cloud case studies in such a way that it can leverage compute and processing services provided by other providers, e.g., NASA's Nebula [13]. Expansion

in this area is also a focus of our future work, as is integration with other cloud framework APIs, such as OpenStack and CloudStack.

In 2011, we held a special workshop at the International Conference on Software Engineering [5], to discuss software engineering, architecture, design, and case studies of cloud computing in science data systems. Several of the case studies presented in this chapter were first described at the workshop *SECLOUD*, or Software Engineering for Cloud Computing, in particular, the Lunar Mapping and Modeling (LMMP) project use case [14], and the DESDynI science data processing system use case [15].

Several of the science data system projects described in the aforementioned case studies utilize software frameworks (e.g., Apache OODT [16, 17]) similar to the Apache Airavata project [6] whose focus is on science gateways and portal software. Recently through the Google Summer of Code project [18], efforts are underway to extend Airavata to interact with cloud frameworks and APIs.

2.6 Conclusions

Cloud computing has provided a synergistic approach for many of our science projects at NASA's Jet Propulsion Laboratory. Given the massive data and computation demands of our projects, coupled with the distributed nature, cloud computing directly integrates with our current and future science data system plans. While there are still challenges in data movement, security, and governance that can affect choices between using public, hybrid, or private clouds, our systems, through the use of the Apache OODT infrastructure, are positioned and proven that they can use and take advantage of cloud computing services.

Many of the challenges that now affect the use and selection of cloud computing are generally cultural. Groups are used to local control. However, the cloud offers a number of very attractive capabilities in a cost-effective manner that can substantially help scientific computing projects. The era of "big data" in science is placing new demands on the systems that support scientists. They must provide robust computing capabilities that scale and support generation production, management, and analysis of massive data systems. The combination of Apache OODT with the cloud offers a viable solution for addressing big data management challenges. The decoupled nature of the infrastructure allows for construction of data systems where services, such as computation and storage can be resourced remotely. In addition, Apache OODT is well positioned to take advantage of other services that are provided through the cloud. This includes Apache Hadoop where instances on the cloud can be integrated with Apache OODT to provide data management and computational services within a highly distributed environment.

At JPL, we believe the era of "big data" is just beginning. With major vendors embracing cloud computing, the services become commodities that are routinely integrated into our systems. While there are still issues to be addressed in terms of

standardizing the delivery of the systems, our decoupled approach allows us to embrace both cloud computing and its specific vendor implementations overtime. As a result, our focus can be shifted away from having to provision our own storage and computational services toward delivering more advanced data analytic services that help our science community in analyzing these massive data sets. It is this combination that will truly help us unlock the scientific mysteries of our time.

Acknowledgments The authors would like to thank the many project sponsors and collaborators that have supported this effort from the National Aeronautics and Space Administration, National Cancer Institute, and the Jet Propulsion Laboratory. This includes Elizabeth Kay-Im, Gary Lau, Sudhir Srivastava, Christos Patriotis, Shan Malhotra, Dana Freeborn, Andrew Hart, Paul Ramirez, Brian Foster, and Brian Chafin.

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract to the National Aeronautics and Space Administration.

References

1. Crichton, D., Mattmann, C., Hughes, J.S., Kelly, S., Hart, A.: A multi-disciplinary, model-driven, distributed science data system architecture. In Yang, X., Wang, L.L., Jie, W. (eds.) Guide to e-Science: Next Generation Scientific Research and Discovery, 1st edn. XVIII, 558 pp., 184 illus. Springer, London, ISBN 978-0-85729-438-8 (2011)
2. Chang, G., Malhotra, S., Wolgast, P.: Leveraging the Cloud for robust and efficient lunar processing. In: Proceedings of the IEEE Aerospace Conference, Big Sky, March 2011
3. Khawaja, S., Powell, M.W., Crockett, T.M., Norris, J.S., Rossi, R., Soderstrom, T.: Polyphony: a workflow orchestration framework for Cloud Computing. In: Proceedings of CCGRID, pp. 606–611 (2011)
4. Crichton, D., Mattmann, C., Thornquist, M., Hughes, J.S., Anton, K.: Bioinformatics: biomarkers of early detection. In: Grizzle, W., Srivastava, S. (eds.) Translational Pathology of Early Cancer. IOS Press, Amsterdam (2012)
5. ICSE 2011 Software Engineering for Cloud Computing Workshop (SECLOUD): https://sites.google.com/site/icsecloud2011/ (2011)
6. Marru, S., Gunathilake, L., et al.: Apache Airavata: a framework for distributed applications and computational workflows. In: Proceedings of the SC 2011 Workshop on Gateway Computing Environments, Seattle, 18 November 2011
7. Garfinkel, S.: An evaluation of Amazon's grid computing services: EC2, S3 and SQS. Tech. Rep. TR-08-07, Harvard University, August 2007
8. Acosta N.: Big Data on the Open Cloud. Rackspace US, Inc. (2012)
9. Ciurana, E.: Developing with Google App Engine. Firstpress, Berkeley (2009)
10. OpenStack: http://openstack.org/ (2012)
11. CloudStack – Open Source Cloud Computing: http://cloudstack.org/ (2012)
12. Amazon Elastic MapReduce (Amazon EMR): http://aws.amazon.com/elasticmapreduce/ (2012)
13. Nebula Cloud Computing Platform – NASA: http://nebula.nasa.gov (2012)
14. Chang, G., Law, E., Malhotra, S.: Demonstration of LMMP lunar image processing using Amazon E2C Cloud Computing facility. In: Proceedings of the ICSE 2011 Software Engineering for Cloud Computing (SECLOUD) Workshop, Honolulu, May 2011
15. Tran, J., Cinquini, L., Mattmann, C., Zimdars, P., Cuddy, D., Leung, K., Kwoun, O., Crichton, D., Freeborn, D.: Evaluating Cloud Computing in the NASA DESDynI ground data system. In: Proceedings of the ICSE 2011 Workshop on Software Engineering for Cloud Computing – SECLOUD, Honolulu, 22 May 2011

16. Mattmann, C., Crichton, D., Medvidovic, N., Hughes, S.: A software architecture-based framework for highly distributed and data intensive scientific applications. In: Proceedings of the 28th International Conference on Software Engineering (ICSE06), Software Engineering Achievements Track, Shanghai, 20–28 May 2006, pp. 721–730 (2006)
17. Mattmann, C., Freeborn, D., Crichton, D., Foster, B., Hart, A., Woollard, D., Hardman, S., Ramirez, P., Kelly, S., Chang, A.Y., Miller, C.E.: A reusable process control system framework for the orbiting carbon observatory and NPP Sounder PEATE missions. In: Proceedings of the 3rd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT 2009), 19–23 July 2009, pp. 165–172 (2009)
18. Google Summer of Code – Google Code: http://code.google.com/soc/ (2012)

Chapter 3
Social, Dynamic and Custom-Based Clouds: Architecture, Services and Frameworks

Fareeha Zafar and Omer Muhammad Ayoub

Abstract The fundamental areas of Information Technology are being revolutionised by cloud computing—offering versatility of services in the form of infrastructure (IaaS), software (SaaS) and platforms (PaaS). This chapter focuses on existing cloud computing technology and architectural frameworks for three different cloud environments, namely, social clouds, dynamic clouds and custom-based clouds. It also provides a brief discussion about the need for cloud development frameworks along with the services rendered by different clouds. Since the endowment of services to different enterprises being served by the same cloud depends upon the type of consumer requirements, this chapter suggests various software engineering techniques and principles to design different types of cloud frameworks. In this context, the process model of *Extreme Programming* is proposed to build custom-based frameworks for dynamically changing cloud-based requirements of clients. Software engineering process models are also identified to cater clouds with streaming data, keeping in view the variable business needs and scientific advancements. Through this, a new integrated process model is also brought under discussion to develop and deploy dynamic clouds.

Keywords Social cloud • Dynamic cloud • Custom-based cloud • Software process model • Cloud deployment

F. Zafar (✉)
School of Computing, University of Derby, Derby, UK

Department of Computer Science, GC University Lahore, Katchery Road, Lower Mall, Lahore 5400, Pakistan
e-mail: dr.f.zafar@gcu.edu.pk

O.M. Ayoub
Faculty of IT, University of Central Punjab, Lahore, Pakistan

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_3,

3.1 Introduction

Cloud computing evolution has successfully provided secure, versatile and an enormous amount of scalable resources as per the corporate and business needs. Cloud-based applications are also serving a huge interest for several potential and immediate returns on investments. Whether it is entertainment services that include streaming audio and video, live chats, photo trading or online gaming, cloud computing services are rendered as most suitable solutions. Cloud computing is in vogue, say, a status representation over the Internet, and at the same time it invites busty traffic to the network. It is changing the future of enterprise architectures. It is being seen by IT and business think tanks as the backbone of scalable, distributed, heterogeneous and diverse networks. It is noticeable that cloud computing is seen as a computing model and not technology and as amalgamation of many different technologies which undertake components of software engineering (i.e. software development models), resources virtualization and network privatisation together which form this new model. Cloud computing promises a different way to architect and remotely manage computing services.

Cloud computing provides a model in which the Information Technology resources and services are abstracted from the principal infrastructure and provided on demand to the consumers. It can be saleable and adapted at a multi-tenant environment. It integrates multi-communication platforms, all submerged and presented to the customers as one. Social clouds act as the next-generation platforms for customer service rendered via technology. Changed business requirements demand for cloud services that can adjust to the new requirements. Organisations, big and small, are requiring to get shifted onto cloud environments; however, existing cloud frameworks may act as workable solutions only that may not be very efficient. That is why *Extreme Programming* process model along with some other software engineering process frameworks is being proposed, in this chapter, for the development of customised cloud-based solutions.

This chapter discusses, in some detail, cloud computing, cloud-based service models, cloud deployment approaches, cloud architectures and dynamic cloud computing. A software process model has also been proposed for dynamically changing requirements of client from the cloud.

3.2 Cloud Service Delivery Models

Cloud offers services according to three fundamental service models as shown in Fig. 3.1. The models are:

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

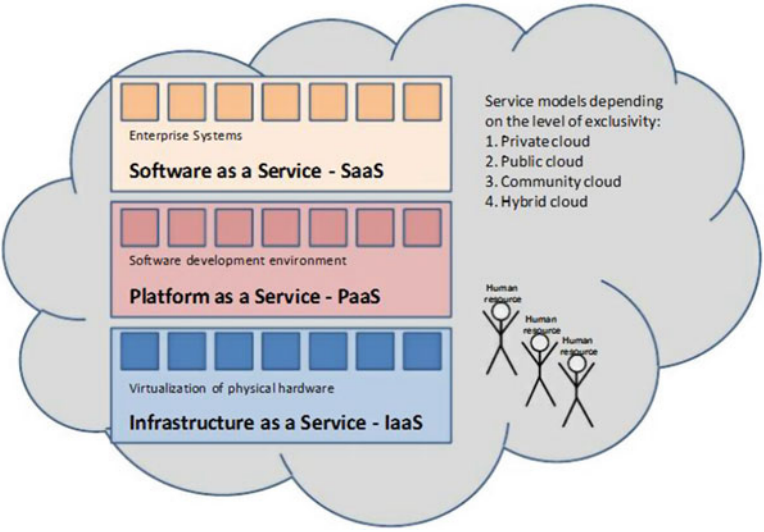


Fig. 3.1 Cloud computing service models

3.2.1 Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) is a type of cloud computing services in which physical computing infrastructure is provided on lease to the organisation keeping in view their business needs and requirements. Virtual machines (VM), operating systems, runtime components, network, storage, data and applications are included in computing infrastructure. Underlying physical hardware is provided by cloud vendors. Users work at their end according to their business infrastructure and can provision cloud resources when needed [1].

In IaaS, back-end hardware is administered and owned by the cloud service provider [2]. Infrastructure as a Service is less expensive [1]. It allows utilisation of hardware without purchase, thus saving on the capital costs. IaaS allows to access and manage systems from anywhere around the globe and thus helps organisation to grow or shrink its business infrastructure. Mobility, stability, agility, availability and elasticity in business are made possible while incorporating IaaS [1].

3.2.2 Platform as a Service (PaaS)

Platform as a Service (PaaS) provides platforms for developing applications without installation of development tools and libraries. The cloud provider also provides networks, storage and servers. PaaS helps business users to lower operational costs and enhance their productivity. Instead of focusing on infrastructure setup,

PaaS enables its users to focus on innovation which is more important. In PaaS, there is no budget pressure as all and no licensing fees as tools are being self-provisioned from the cloud.

3.2.3 Software as a Service (SaaS)

Software as a Service (SaaS) is a model in which the cloud service provider or a vendor hosts applications to be run within the cloud. Software as a Service has been a part of strategic planning of many enterprise companies. Many of the service providers offer application programming interface (API) that exposes the applications data and functionality to developers. Many different mechanisms are used to keep data and storages secure. SaaS provides quick implementation for getting the application up and running. It includes maintenance and support and the user does not need any hardware [3]. Software as a Service (SaaS) is easy to use and is benefiting many trades. It is much less expensive as there is no purchase or licensing costs.

3.3 Deployment Models for Cloud Computing

There are four deployment models [1] which can be classified depending upon the requirements provided by the client. Each model has specific services and fulfils the needs in a particular way. The four models, as shown in Fig. 3.2, are as follows:

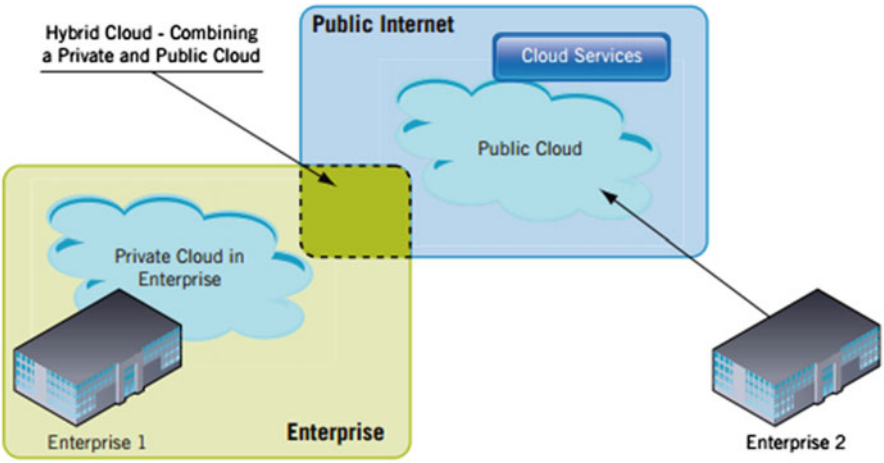


Fig. 3.2 Deployment models

- Private cloud
- Community cloud
- Public cloud
- Hybrid cloud

3.3.1 Private Cloud

Private cloud is a cloud computing environment based on several physical servers and other IT resources including software. The resources act as dedicated resources and can be leased by the customers [4]. Data resources and applications in the private cloud can only be accessed by permitted users. Private clouds are always protected behind firewalls and generally used by the organisations that own them.

3.3.2 Community Cloud

A community cloud can be referred to as a multi-tenant environment which ensures a common computing environment shared by all the organisations/consumers of the cloud. Audit requirements and hosting applications can be common computing concerns for various organisations. Community cloud supports a certain community (having similar requirements), and it can be made available on and off as per consumer requirements [5].

3.3.3 Public Cloud

The cloud infrastructure is available to the public on commercial basis. This enables user to develop and deploy applications in the cloud with very little financial outlay. Public clouds, as the name implies, are meant for general public or large group of organisations. Unlike private and community clouds, public clouds are managed and owned by external cloud providers.

3.3.4 Hybrid Cloud

A hybrid cloud is a blend or combination of two or more of the above variety of clouds, generally a combination of a private and a public cloud. A hybrid cloud provides a cloud computing environment in which some of the resources are managed internally whereas some of them are public. Often, it stores sensitive data internally and takes its backup externally in the public cloud so that if the system fails, backup data is available somewhere.

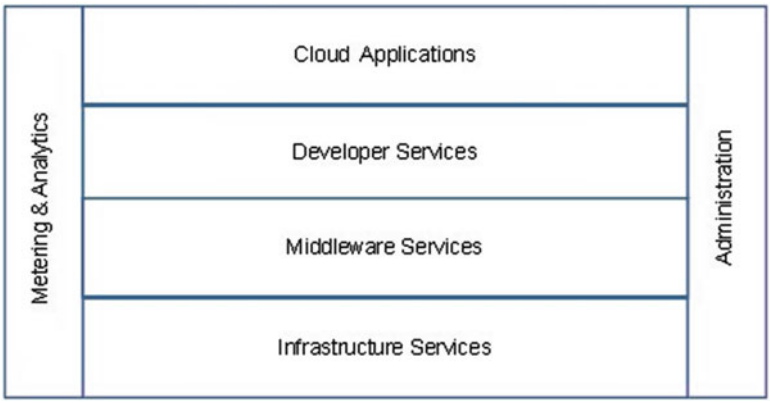


Fig. 3.3 Cloud development framework

3.3.5 Basic Cloud Development Framework

Cloud solutions demand structure and frameworks. Figure 3.3 presents a cloud development framework.

Layers of this basic cloud development framework are elaborated as follows:

- *Infrastructure services layer*
  This is the layer that has core capabilities of IaaS. It defines the services of hardware, software and allied infrastructure that is needed for the deployment of developed cloud solution. Powerful infrastructure is essentially needed for a robust and reliable solution.
- *Middleware services layer*
  Capability of this layer is to make the development work compatible with desired application. Middleware service layer usually does this through APIs. Thus, this layer helps to shape the result from development effort.
- *Development services layer*
  This is the development layer where the actual implementation takes place.
- *Application layer*
  It is the top layer of the model and consists of two types of applications. One is developed and the other that are deployed in the cloud. These are called as Native Cloud Applications and Ported Cloud Applications. Every development model needs management capabilities for effective and useful provision.
- *Marketing*, *analytics* and *administration*
  This includes management capabilities.
- *Metering and analytics*
  These are the tools for managing the usage of cloud solution. Using these tools, the developer measures performance and efficiency of cloud solution and tracks runtime cost models, and the result information can be used by billing engines.

- *Administration*
  This provides capabilities of configuration management/version control system and data dictionary services. Related tools ensure the efficient development in cloud environment.

3.4 Social Clouds: Services and Framework Architecture

3.4.1 Social Cloud

Social cloud is a virtual cloud service backboned by cloud computational model wherein virtualized resources provided by some users are used by a group of 'friends'. Users can be given a freedom to use these resources or they can follow some other techniques; in this regard social cloud provides a new way to communicate by providing a blend of social network and cloud computing thus forming a social cloud. Social cloud enables people to share resources within the context of social networks as seen in Fig. 3.4 [6]. In simple words, we can refer social clouds to be a place where people over the Internet can socialise with others across their respective platforms.

Considering the massive shift in communication, news and media, this brings something different to the table. Remember that the social consumer evaluates and

Fig. 3.4 An example of social cloud

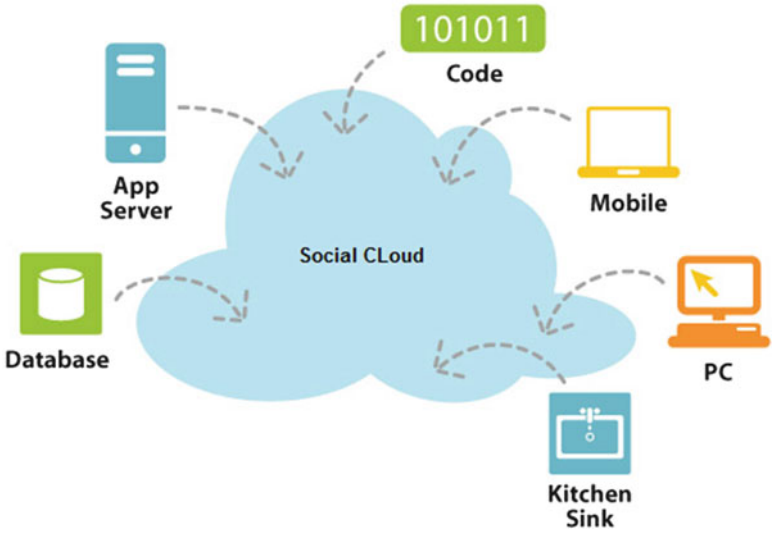


Fig. 3.5 An example of connectivity to social cloud

absorbs information differently from the traditional consumer. The conversation is now a two-way street; it is necessary to monitor your community and respond with the established 'voice' or keeping ourselves 'updated' about everything related just as shown in Fig. 3.5. Across all social media, there is a need to maintain consistency in the way various companies present their brands. Because this relationship is about interaction, develop and position your message in a way that invites your community to relate and connect with other platforms.

3.4.2 Framework Architecture

Google is the largest search engine at present and Facebook is an example of a largest social network cloud as Software as a Service (SaaS). In order to support huge requests from the world through each personal computer, mobile device and smart phone, traditional frameworks are scaling up architecture. But in a social cloud network, the scaling up is in general a non-trivial venture, and the problem is discriminating particularly due to the rapid growth that they can potentially experience. Best practice for social cloud architecture starts with a fully distributed architecture to avoid traditional scalability problems. However, this is not always an option due to resource scarcity: there is a trade-off between the application functionality versus future scalability. Consequently, common practice is to implement a prototype of a social network application to run in a scaling out paradigm and then scale it up whenever the application takes off. Refer also to Fig. 3.6.

The typical distributed architecture consists of a number of layers: application logic (Ruby on Rails, Scala), caching (Memcache, SQL query caching) and database back end (RDBMS clusters, Couch DB, Google's Big Table or Amazon's Dynamo)

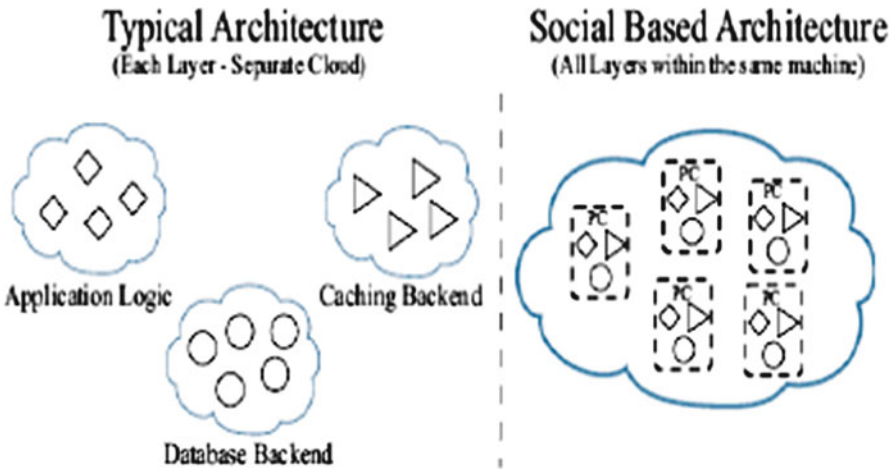


Fig. 3.6 A typical database architecture versus social-based architecture

that interact through asynchronous message passing. In such distributed architecture, each layer consists of a number of machines devoted to perform their respective tasks, and scaling up can be addressed by increasing the number of servers in each layer.

3.4.3 Architecture Comparison

In order to know the future of cloud technology trend in the social network domain, here is a comparison between existing business models of two famous social cloud service providers: Google and Facebook.

3.4.3.1 Software as a Service (SaaS) Business Model

As shown in Table 3.1, these two cloud service providers have the same or similar revenue sources. However, in February 2010, Facebook toppled Google as the top online destination from Internet traffic report. That suggests that the social cloud and network is more attractive than a search engine cloud. Similar illustration is shown in comparative graphs in Fig. 3.7.

3.5 Cloud Building and Software Process Models

3.5.1 Building Cloud Environment

One of the most important factors in building a cloud is service analysis. Here, the entire infrastructure is analysed, understanding how application workloads,

Table 3.1 SaaS business model comparison

Items	Google	facebook
Service	Search engine	Social network
Source of Revenue	Online advertisement	Online advertisement
Targeted Customer	General customers	Selected customers (Filter by location, interest, and age etc)
Charging Model	Cost-per-click(CPC): USD$0.6	Cost-per-click(CPC): USD$0.6

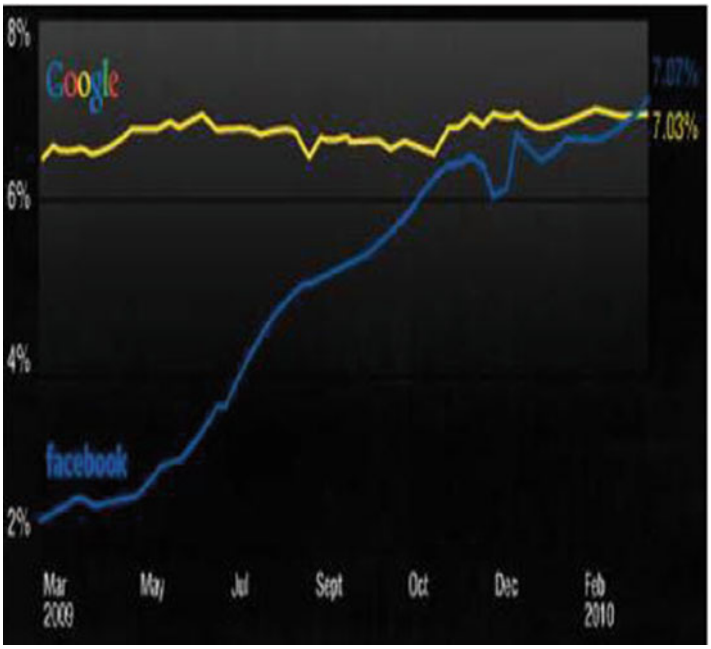


Fig. 3.7 SaaS business model comparison

Fig. 3.8 Approach to cloud building

utilisation levels and resources interact. After the analysis, requirements of the client are mapped with the cloud services and then a particular service is chosen for the client [7]. Building a cloud requires standardisation, automation and shared resources. Standardised and documented procedures are needed for operating, deploying and maintaining the cloud environment. These standardised procedures are the building blocks of the next step which is automation. Automating deployment for the cloud is necessary because it then provides self-service capabilities. Through automation different workflows are viewed and then resources are chosen which the cloud must have. Once approved, cloud platform automatically deploys that chosen environment. Building a cloud requires the assessment of different types of workloads and the risk, benefit and cost of moving each to different cloud models. Shared infrastructure, self-service automation portal, scalable, rich application container, programmatic control, 100 % virtual hardware abstraction, strong multi-tenancy and chargeback are the eight ingredients in building a cloud. Cloud storage gives the opportunity to the organisations to manage rising file storage costs and burden under control. It helps organisations in reducing the funds spent on purchasing file storage capacity, and it also cuts down the overheads to manage them. Storage required in the cloud can be increased depending upon the needs and requirements of the client [8].

The following steps can be considered in cloud building. These are also shown in Fig. 3.8.

Virtualise Server virtualisation is a cloud prerequisite. Virtualisation standardises IT infrastructure assets and resources. It can also be extended to storage and network for achieving benefits of higher utilisation, lower costs and standardisation and a requirement for multi-tenancy.

Integrate After the process of virtualisation, IT infrastructure is integrated and unified. Individual IT infrastructure components can be integrated into unified platforms: an integrated package of virtualization, storage and network access for a complete computing platform and a unified fabric for interconnecting computing and storage resources. These platforms can be packaged into infrastructure building blocks.

Automate With virtualisation and integration, infrastructure can only be viewed as pools of resources. Automation provides rapidly deployable service by automating those operations.

Deliver After automation, IT capabilities can be viewed as a catalogue of services. These services depend upon the requirements of client and keep on changing with dynamic needs of business.

3.6 Software Engineering Practices and Cloud Frameworks

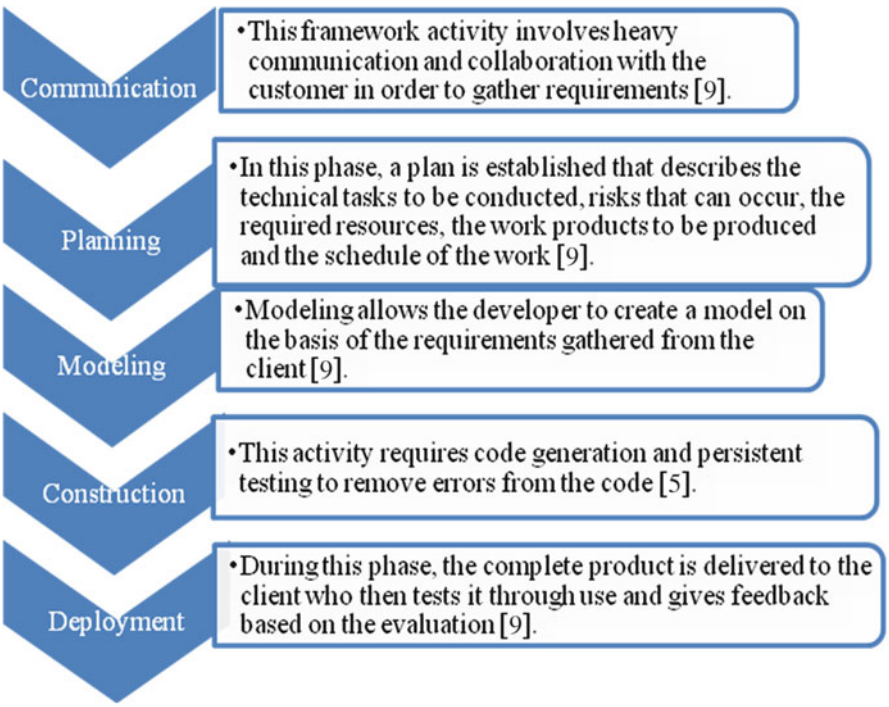
3.6.1 Phases Within a Software Process Model

A process model is a unique set of framework activities for the development of software processes [9]. All process models undergo five important phases, namely:

- Communication
- Planning
- Modelling
- Construction
- Deployment

The working ability of each phase is presented in Table 3.2.

Table 3.2 Software process model



3.7 Software Process Models for Cloud Deployment

3.7.1 Custom-Based Cloud

This section provides emphasis on establishing a cloud environment, where individual business requirements are treated and a customised cloud model is provided to meet the specific needs of the working environment within organisation. Change in the business world requires parallel changes in technological measurements and with individual business; some requirements might also need customised solutions. So, why not develop a customised model of a cloud which can meet the changed requirement of the client?

Custom-based cloud development framework prompts the idea that one solution cannot fit in all situations. Businesses may be large or working environments can be small, or these may stand at intermediate scale and hence would need to satisfy requirements accordingly. Same patterned cloud models might present a workable solution but can prove to never be an efficient solution to cater every individual business environment need. Similarly every business at initial level might possess needs, which are different from those which come across later or may need extension of existing requirement. In all such cases, it is essentially required that a custom-based cloud development framework shall exist to fulfil the changing requirements of such clients.

3.7.1.1 Process Model for Custom-Based Cloud

We suggest involving software engineering principles of Extreme Programming in order to develop a customised cloud to cater for the needs of individual customers. At present, several cloud deployment models exist; however, interestingly due to technological barriers, they have to get for all types of business needs. In the era of dynamic changed requirements of business, Extreme Programming can be deployed to provide a custom-based cloud. Figure 3.9 provides a flow chart of how it shall work for individual customer needs.

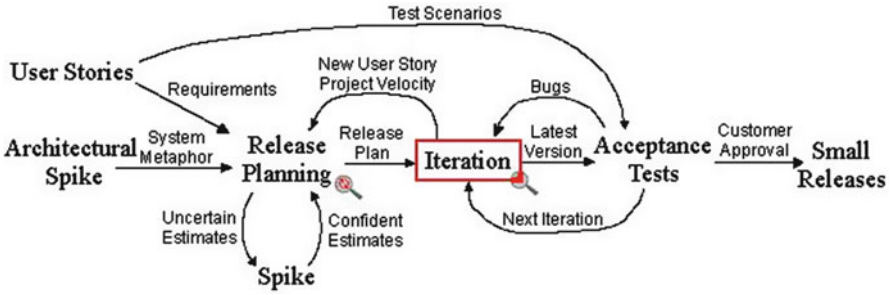


Fig. 3.9 Extreme Programming activity flow chart

The flow chart in Fig. 3.9 suggests how Extreme Programming rules work together in order to cater the business client. Customers enjoy being partners in the cloud development process, developers actively contribute regardless of experience level, and managers concentrate on communication and relationships. Unproductive activities have been trimmed to reduce costs and frustration of everyone involved. In the end, a small release of workable cloud is handed over to clients to start their business.

3.7.1.2 Extreme Programming for Cloud Development Architecture

Extreme Programming-based cloud development emphasises teamwork. Cloud vendors, customers and developers are all equal partners in a collaborative team. Extreme Programming implements a simple yet effective environment enabling vendors and cloud development managers to become highly productive. The organisation that wants to deploy cloud services becomes part of team, and all stakeholders self-organise around the problem to solve it as efficiently as possible until they find a real efficient cloud instead of deploying a workable one.

Extreme Programming improves a development of a project in five essential ways: communication, simplicity, feedback, respect and courage. Extreme Programmers constantly communicate with their business customers and fellow programmers. They keep their design simple and clean. They get feedback by testing their cloud starting on day one. They deliver the cloud to the customers as early as possible and implement changes as suggested. Every small success deepens their methodology of producing an effective solution in the form of the required cloud. With this foundation Extreme Programming courageously responds to changing requirements and technology of business clients.

XP integrates vendors and end users, i.e. business client onto development iteration. The business environment is asked to identify the acceptance tests, and the XP team at the cloud vendors' end works to automate these tests so they can be run throughout the iteration. An overview of XP is shown in Fig. 3.10.

Some of the characteristics of XP are as follows:

- *Specific practice*—Unlike Scrum, XP may act more specifically about the practices that should be used during a cloud development project. These practices include pair programming, TDD, continuous integration, refactoring and collective code ownership along with security concerns of shared network.
- *Modelling*—XP teams frequently use modelling to better understand the required tasks and cloud architecture needed to support individual user story.
- *Simplicity*—XP helps to perform the minimum and only focused work needed to meet requirements.
- *Automation*—Unit and functional tests are automated for each release of cloud.
- *Quality through testing*—Features are tested constantly; pair programming and iterative testing help to formulate a more customised shape of cloud for individual business.

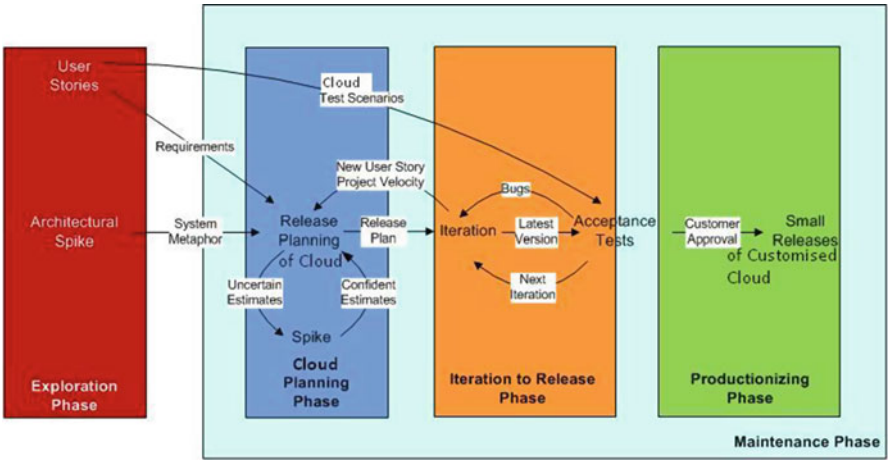


Fig. 3.10 Extreme Programming life cycle for customised cloud development

These are some of the strengths of XP:

- Customer-focused cloud (it is all about user stories)
- Quality via frequent testing of released cloud
- Constant focus on identifying and delivering the critical business dynamics
- High visibility on development status
- Great support for volatile requirements

It also has weaknesses such as:

- *Need for team maturity*—Practices such as pair programming and TDD require responsible developers, and they are not always easy to obtain.
- *Dependency on testing*—If developers know that significant testing will take place downstream, they may be less than diligent when they are creating designs.
- *Scalability*—XP may not work well for large cloud frameworks.
- *Dependency on team member co-location*—The team usually has a team room, in case of development of cloud, vendors, XP team members and client need to get located at the same place.

XP supports many of the critical goals of an agile process, such as dealing with volatile requirements, delivering prioritized custom based solutions and working with cloud frameworks as early as possible.

3.7.2 Clouds with Streaming Media

Cloud with streaming media refers to those entertainment clouds which include streaming audio and video, live chats, photo trading or online gaming.

Table 3.3 Steps for component-based cloud development

Planning and research	Integration issues	Design	Integration	Testing
Research of available component-based products Application domain is evaluated	Component integration issues are taken into consideration for further architecture	Cloud architecture is designed	Components that pass the integration test are integrated to the cloud-designed architecture	Rigorous testing is done to ensure proper functionality of cloud

3.7.2.1 Process Model for Streaming Media in the Cloud

This section discusses a process model to incorporate streaming media in the cloud. Streaming media is in common use. In order to deploy and develop this feature within a cloud, a component-based process model may be considered to be the best. A component-based process model encompasses applications from prepackaged software components. When streaming media is used over the Internet, its components are available and can be reused to deploy the same underlying architecture in the cloud. These components provide targeted functionality with well-defined interfaces that enable the component to be integrated into the software.

3.7.2.2 Component-Based Process Model for Cloud with Streaming Media

Along with underlying architecture of streaming media over the cloud, the client can have his own requirements for the cloud. They can be incorporated in the architecture using component-based process model. This process model leads to software reuse, and reusability reduces much of the development cycle time which in turn benefits both the developer and the client. Moreover, the overall production cost is also reduced by using component-based process model [9]. Component-based development incorporated steps as shown in Table 3.3.

3.7.3 Software Process Model for Dynamic Clouds

3.7.3.1 Dynamic Cloud

Dynamic and changing business requirements need dynamic cloud services. As IT technology is taking over the traditional business activities, cloud computing has

transfigured the market. Business requirements are becoming more agile and scalable which give a competitive edge in the market. Due to their changed dynamic nature, dynamic services must be provided. Such services are mostly needed by those enterprises where challenging and more knowledgeable clients are being dealt with. This is the very reason to introduce and launch dynamic clouds. Dynamic cloud has the capability of handling dynamic and changing requirements of businesses and individuals. It can react to market's ups and downs effectively which makes it popular in the industry [10].

Dynamic cloud helps in improving business agility, flexibility, efficiency and cost-effectiveness. It increases utilisation and reduces cost; service delivery is enhanced and is provided on time. Dynamic cloud is also responsible for business innovation [7]. Dynamic cloud allows its clients to build customised servers whenever they are needed by them, and they are provided with a user-friendly control panel to configure, manage and maintain their virtual data centre in the cloud depending upon their variable needs. Dynamic clouds are inexpensive in regard to their services. It is also to be noted that in order to handle a dynamic cloud, security must be dynamic and should be changed with workload and data. It must also be able to configure itself in new environment in a consistent and automated manner.

The benefits of dynamic cloud computing are apparent through social network clouds like Facebook, MySpace, Twitter and other popular web-based applications. They serve dynamic content to millions of users, whose access and interaction patterns are hard to predict. In addition, their features are very dynamic in the sense that new features can be created by independent developers, added to the main system and used by other users. In several situations vast amount of memory could be used, for example, whenever new system features become popular (e.g. use of timeline on Facebook) or a new plug-in application is installed. As these popular social networks are used all over the world at different areas any time, load spikes can take place. In order to handle unpredictable seasonal changes in system workload, an automatic scaling scheme is vital to keep the quality of service and resource consumption at appropriate levels [11]. Dynamic cloud offers many advantages. It provides full control with safety and security. It offers easy extensions. High availability and inactive servers are also part of dynamic clouds.

3.7.3.2 Process Model for Cloud to Meet Dynamic Requirements of Clients

Cloud design depends upon enterprise requirements. According to it, various process models, deployment models and service models are chosen for building a cloud. So, there is no need to stick on one such cloud because requirements vary from enterprise to enterprise and every one of them wants a customised cloud for their business needs. The main focus is on the *dynamically changing requirements of client*. This means that requirements can change at any point of time during the development of a cloud for a particular client. Hence, process model must be chosen wisely.

Changing requirements of a client has a risk factor that must be taken into account. Risk analysis is chosen from spiral process model to deal with the risk factors involved and to overcome them. Time factor is taken from RAD model which has a short development cycle and focuses on time constraints. The rest of cloud development is done under the component-based process model. Component-based process model is chosen as software that can be reused and integrated in the proposed architecture of the cloud which saves both time and money [9].

Spiral model is one good choice as it demands considerable risk assessment and relies on this expertise for success. The main reason to use this approach is to practice risk analysis factor; however, Rapid Application Development (RAD) typically focuses on time factor. Time constraints are picked up from this model, as the business client might come with dynamically changing requirements. Therefore, to deal with all such issues, one set of requirements cannot last long over a longer period of time. Short development life cycle enables the user to focus on other requirements of the client. Component-based approach saves time and money as already made components are added and integrated with the required architecture of the cloud.

The process model thus proposed in this section is a combination of three of them: RAD, spiral and component-based process model. Each of these is selected based on various specialised core factors which they possess and are required.

3.7.3.3 Diagrammatic Illustration of Process Model for Dynamic Cloud

Various phases are picked up and integrated from Spiral, RAD and component-based process models in order to formulate a new stepwise process model to cater for the development of a dynamic cloud. A diagram for the said process model is illustrated in Fig. 3.11.

3.8 Conclusion

Software engineering process frameworks are a great help to identify and deploy solutions for each type of cloud framework. Social, dynamic or custom-based cloud; whichever category a client may select, cloud brings an essential shift for businesses around the globe. Cloud computing, in its every form, has provided cost-effectiveness, elasticity, agile, flexible and scalable solution to the business world. It provides the users with opportunities to build custom applications on a user-friendly interface without hardware hiccups. However, there is an immense need to identify correct process model for the deployment of cloud-centric environments in order to meet changed business requirement of clients. One solution can never fit all problems; likewise, there is a need of customised cloud for individual businesses and dynamically changed requirements of clients.

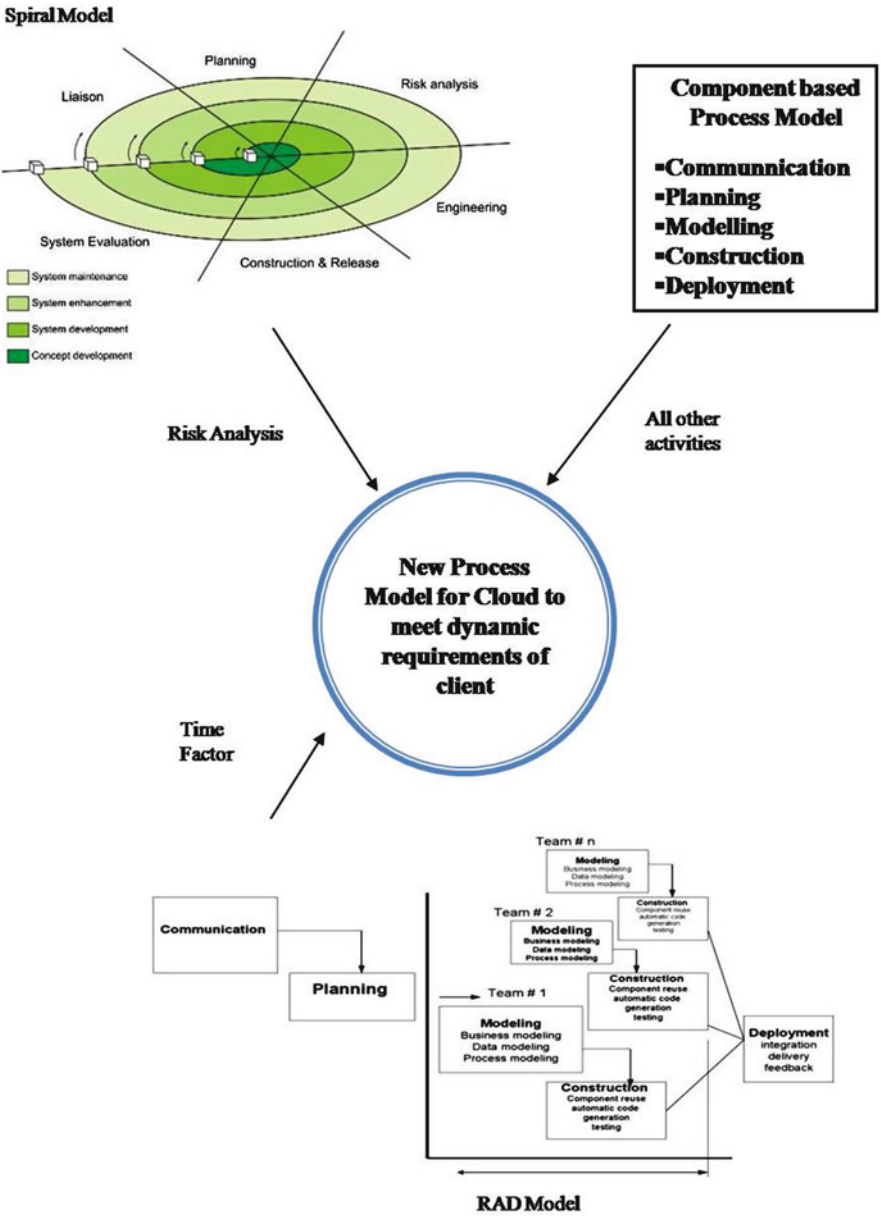


Fig. 3.11 Process model for dynamic requirements of client

References

1. Hess, K.: The frugal networker, why you need Infrastructure as a Service (IaaS). http://frugal-networker.com/2012/01/07/why-you-need-infrastructure-as-a-service-iaas/ (2012)
2. Warren, B.: Ezine Mark, Infrastructure as a Service – Benefits of IaaS services. http://infra-structure.ezinemark.com/infrastructure-as-a-service-benefits-of-iaas-services (2011)
3. Mahmood, Z., Hill, R.: Cloud Computing for Enterprise Architectures, pp. 52–53. Springer, London/New York (2011). ISBN 978-1-4471-2236-4
4. Schubert, P., Adisa, F.: Informatik, cloud computing for standard ERP systems reference framework and research agenda. Paper, pp. 11–14 (2011)
5. Hefner, K.: Search cloud storage, community cloud. http://searchcloudstorage.techtarget.com/definition/community-cloud (2012)
6. Chard, K. et al.: Cloud times, social cloud: social networks and cloud computing connections. http://cloudtimes.org/social-cloud/ (2012)
7. Guerrero, J., Louis, A., Vail, H.: NetApp, building a cloud on FlexPod: validated management solutions. www.netapp.com/us/communities/tech-ontap/tot-building-a-cloud-on-flexpod-1203.html (2011)
8. Grundy, J., Kaefer, G., Keung, J., Liu, A.: Software engineering for the Cloud. IEEE Softw. **29**, 26–29 (2012)
9. Pressman, R.: Software Engineering, 6th edn. pp. 78–92. Prentice Hall (2008)
10. Fripp, C.: IT News Africa, the dynamic cloud has arrived. http://www.itnewsafrica.com/2010/10/the-dynamic-cloud-has-arrived/ (2010)
11. Rubin, E.: Cloud switch, dynamic cloud fitting – the future in automated cloud management. http://www.cloudswitch.com/page/dynamic-cloud-fitting--the-future-in-automated-cloud-management (2009)

Chapter 4
Consumer Cloud: Concepts, Vendor Solutions, Industry Applications and Way Forward

Naveen Kumar and Sudhanshu Hate

Abstract Over the years, consumers have been accessing the World Wide Web to perform one or more of the personal functions such as accessing email via messenger/phone communication; managing social and professional identities using networking sites; education and learning via the web; entertainment through listening to music and watching movies; and managing documents, organizing finances and going to the extent of managing small businesses over the web. The traditional discrete web-based solutions deal with consumer needs in a highly fragmented manner, through multiple discrete point solutions and devices. This has given rise to challenges in cross-leveraging data, missing contextual intelligence and lack of unified experience across applications leading to loss of productivity and opportunities. Solutions leveraging the cloud provision overcome several of these challenges and are becoming popular, e.g. Apple iCloud platform. Apple, Google, Microsoft and Amazon are fast building strategic partnerships with product vendors to bridge platform gaps and are bringing in innovations to grab a sizeable pie to emerge as a leader in the digital sphere of the consumer ecosystem. This chapter looks at the interesting scenarios for consumer clouds, vendors, technologies, industries and trends currently shaping up as they start to take centre stage in the consumer's activities.

Keywords Consumer cloud • Cloud vendors • iCloud • Cloud challenges • Web-based solutions • Enterprise cloud

N. Kumar • S. Hate (✉)
Infosys Ltd., Bangalore, Karnataka, India
e-mail: nkumar@infosys.com; Sudhanshu_Hate@infosys.com

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_4,

4.1 Introduction

Earlier in the year, on April 2, 2011, I recorded the Cricket World Cup final match between India and Sri Lanka that India won. I was happy thinking I would get to watch this high-definition (HD) recorded moments at my will rather than relying on the TV channels to repeat the recordings.

A couple of months after, ironically, my TV setup box started malfunctioning. The service provider graciously agreed to replace the setup box at free of cost, as it was within a year's warranty. However, later I came to know that with the setup box, I was going to lose all my recorded programs along with the recorded 2011 World Cup victory, which was closer to my heart like many Indian fans. This was because all the programs were saved on the local disk of the setup box. I thought if the service provider had given me the option to save these programs on cloud and in turn charge small fees, I would not have landed in this situation.

While this is a very simple use case or scenario of marrying the consumer electronics (TV) with Cloud, it has a significant impact on our day-to-day lives. This scenario and solution is obvious, and I am sure supported by some service providers. However, the intent of this example is to seed the possibilities and benefits of cloud in the consumer sphere of things.

4.2 Consumerization of IT

With the advent and evolution of the WWW (web) over the last decade, several day-to-day functions which otherwise were long cycles became fast and easy for the web consumers. Consumer's reliance on the web increased to perform some of the day-to-day core functions as depicted in Fig. 4.1 and elaborated below:

- Communication using the web became ultrafast and affordable through email, messenger chats/phone, fax and tweets.
- Establishing social and professional identity and connections using Orkut, Facebook, LinkedIn, etc. became a common norm.
- Entertainment – Successful business models like iTunes made it possible to easily download and listen to music. YouTube simplified sharing and watching videos online.
- Education over the web is much simplified through connected virtual classrooms, webcasts and Internet articles. Sharing knowledge and collaborating on presentations using sites like SlideShare.com became common wisdom.
- Consumers started relying on the web for managing several personal functions such as managing finances, planning travel, managing personal documents and doing personal work/tasks/calendars to the extent of managing personal small businesses.

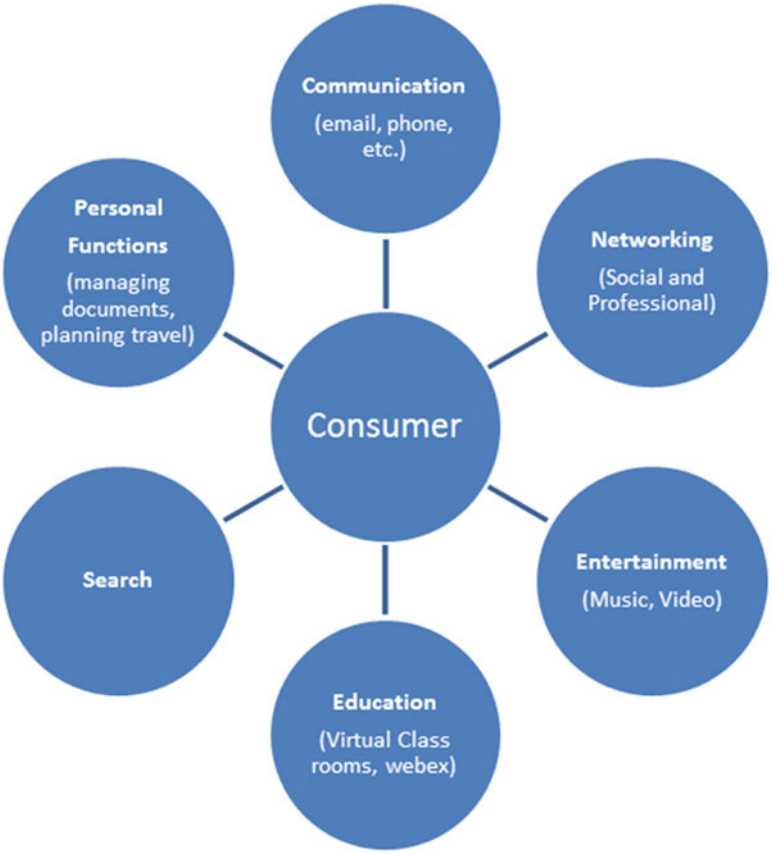


Fig. 4.1 Consumer's core functions on the web (discrete manner)

- Amongst all of this, search (semantic, contextual) on the web became critical to find any information in fraction of a second.

4.3 Challenges and Limitations

The abovementioned web-based solutions hold good and are still serving; however, it has not been able to resolve several impeding issues and therefore falls short on several fronts such as the following:

- Discrete functions – Traditional approaches discussed above included providing point-to-point solutions rather than looking at holistic needs of customer.

- Contextual experience – The context (business process, application and device) was not understood, stored and transferred across applications and devices, leading to loss of continuity.
- Lack of intelligence – Since the traditional sites store only transactional data, it didn't have the ability to capture and analyse the entire user-generated data, leading to loss of intelligence.
- Support for multiple devices – For most of the functionality, web becomes the de facto channel. Limited functionality is made available through other channels such as smartphones due to limitations of form factor, device processing capacity/hardware, etc. issues.
- Inefficient and counterproductive – Traditional solution works in more stateless manner treating every interaction as fresh expecting the user to perform repetitive steps every time, leading to loss of user productivity and interest.
- Data privacy and user identity – Privacy concerns around managing user credentials existed for long time. Several users avoided dealing with any confidential information using web as they don't have full control on their data with respect to storage, duration, controls, etc.
- Managing data growth – Ability to store, manage and process growing unstructured data on a local device in a cost-effective manner became a challenge.

4.4 Solution

Consumer expectations and behaviour has changed significantly over the last few years. In today's era, the consumer demands the ability to seamlessly move across applications and devices. They want to perform various functions in an integrated state-full manner, as depicted in Fig. 4.2.

This is a significant change unlike the isolated one to one dealing with individual functionality as depicted in an earlier Fig. 4.1.

Cloud-based solution offerings promise to deal with most of the challenges mentioned above and provide necessary unified integrated experience to the end user. In the following sections, we put forth what we call 'consumer cloud', together with the various building blocks and implementation options, and analyse corresponding solutions from the viewpoint of the key players operating in this segment.

4.4.1 Defining Consumer Cloud

We define consumer cloud as follows: Traditionally, integrated services that would have required a lot of on-premise storage, computing and device hardware that can now be delivered (synchronized – read/write) from remote elastic storage, computing through lightweight devices such as smartphones, notebooks and PCs as an integrated, unified service offering with super enriched, elevated, immersive user experience, can be considered as a consumer cloud.

Fig. 4.2 Consumer's core functions on the web (integrated manner)

4.4.2 Consumer Cloud Solution Stack

For simplicity, the consumer cloud solution stack can be abstracted and envisioned as shown in Fig. 4.3 and as mentioned below:

- Platform to store and publish information. It is elastic in nature and provides core infrastructure services such as storage, computing and network.
- Applications to support specific point functions as well as delivered integrated functionality such as emails, social apps and video streaming.
- Devices (device hardware + OS) – smartphones, PCs, etc. They are used to access and perform specific functions.
- Integrated service offering – to deliver experiences/results such as contextual, integrated unified experience with ability to pause song from iPod and continue listening the same song from any other Apple devices.

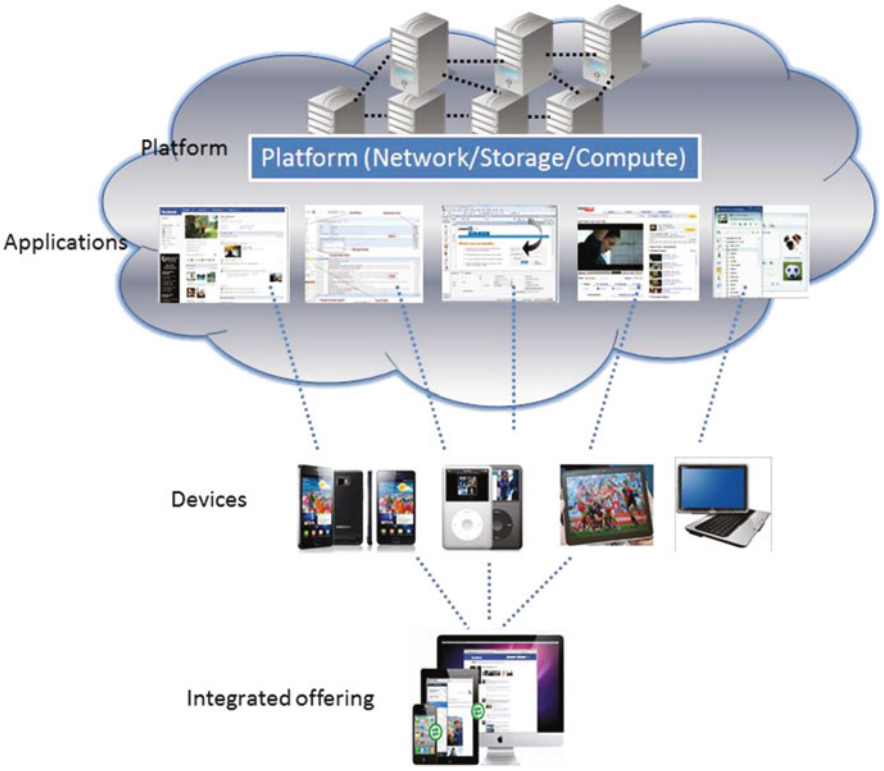


Fig. 4.3 Consumer cloud solution stack

4.4.3 Characteristics of Consumer Cloud

Key characteristics of consumer cloud-based solutions are the following:

- The functionality is available anytime anywhere.
- Have the ability to deal (render, store and process) large amount of structured and unstructured data.
- Works in an integrated cohesive manner in context to the user helping improve user productivity.
- Runs on cloud platform to combine multiple individual use cases to deliver intelligent behaviour for systems and devices improving user experience.

4.5 Consumer Cloud Implementation Options

Apple, Google, Microsoft, Amazon and others have offerings that qualify and excel in consumer cloud space. There are more vendors as listed in [1], but we consider the following four as they are the leading platform providers and have

necessary consumer cloud constituents. We look into each of their offerings in further detail.

4.5.1 Apple iCloud

Apple Inc. announced its iCloud offerings to deliver highly contextual synchronous information through its various devices (iPhone, iPad, Mac, PC, Apple TV) to various form factors. The result is the user gets unified view into his/her calendars, contacts, mails, messages, music, pictures, videos and books and can work with them anytime anywhere without needing to manually transfer/synch up data across devices. It does all this by providing 5 GB of free storage on the cloud. If you need more, you can purchase. Some of the interesting things that can be done are:

- Watch movies on the move, pause and play from other device
- Take picture and share it with friends and family in near real time
- Have a video conference call
- Buy and read books from Kindle
- Bookmark a page in the book to return for reading at later point from other device

More exhaustive list of possibilities with iCloud can be read from here: http://www.apple.com/icloud/what-is.html.

To summarize, Apple has provided integrated offerings as iCloud to tap in consumers; however, it suffers from the lack of openness as iCloud services are available only on Apple devices. If you are using non-Apple device, you won't be able to make use of it. Of all the vendors offering services in consumer cloud space, Apple's iCloud is a forerunner and can be considered as a defining phenomenon in the space of consumer cloud. According to Juniper Research, the mobile consumer cloud market is expected to touch $6.5 billion by year 2016 [2].

4.5.2 Google

Google has strength in its platform through Google App Engine. Google provides various applications for email (Gmail), collaboration using documents, spreadsheets and powerpoint presentations (Google Docs), creating and managing photo albums on the web (Picasa web albums), text and voice messenger (Google Talk), publishing videos (YouTube), maps (Google Maps), social networking (Orkut) and many more free apps along with browser as Chrome. For a complete application list, refer to [3].

In devices strategy, Google has partnered with hardware vendors for the devices and owns OS platform – Android.

Google Chromebook is similar to iPad or Tablet and has several innovative interesting features to improve user productivity and experience. It runs on very thin operating system which boots in as low as 8 s. Applications get updated automatically

when connected. It can be used to work with documents (files, photos, etc.), and the same can be published to cloud. With Chromebook, you can play music from cloud or locally. It has very small local storage and can leverage storage from cloud (provides 5 GB of free storage on Cloud). Offerings like Google Apps for education [4] are offered as an integrated offering for a specific user segment for college and university students.

To summarize, Google has strong platform, strong application base, support for open standards/compliance and good device base and an integrated/unified rich solution offerings in consumer clouds to serve specific user segments (education industry).

4.5.3 Microsoft

Microsoft has strong evolving platform offerings as Microsoft Windows Azure and Microsoft Business Productivity Online Services (BPOS).

Microsoft Windows Azure provides platform APIs to build and host applications on cloud. Microsoft BPOS has Microsoft Exchange, SharePoint, Office Communications and Office Live Meeting online services to meet the online needs of businesses around email, calendaring, voice and video communication, portals, content management and collaboration, instant messaging and live meeting to name a few. Microsoft XBOX Kinect offers online gaming and recently opened up its beta of SDK for game programming [5].

Microsoft has fairly good application base for consumers to perform several core online functions such as Windows Mail, Windows Live, Windows Live Mesh (to synchronize files across devices, 5 GB free storage on cloud/SkyDrive), Windows Live Messenger, Bing (search), Microsoft MapPoint (location intelligence), Microsoft Media Player (music), Microsoft Money (consolidated integrated view into bank accounts, loan accounts, credit cards) and browser as Internet Explorer (IE) to name a few.

In devices strategy, Microsoft provides OS for Windows Phone and has partners like Nokia for providing hardware.

Windows Phone 8 called 'Mango' has several features meant to deal with several consumer challenges and limitations listed earlier. Mango has support for conversation threads so that a conversation with the same person is seamlessly integrated across various channels such as MSN messenger chat, SMS and Facebook. It provides integration with Twitter, Facebook and LinkedIn for working with collective contacts and viewing their respective updates in respective apps, all at one place without needing to go to individual apps.

Multitasking is possible with apps and games with the ability to run them in the background. Augmented Reality using Bing vision by aiming phone camera to the book to get all the information about books such as places to buy, download and also read from Amazon Kindle.

Microsoft Windows 8 Tablet is gearing up for providing competition to iPad or Google Chromebook. Microsoft Windows 8 released in late 2012 plays a pivotal

role in consumer cloud with the set of interesting features it has on offer for solving the pain points in consumer cloud scenarios.

Microsoft Windows 8 is an OS for tablets, laptops and desktops. Windows 8's new touch-based user interface will be replacing traditional icons. Touch-based interfaces will facilitate adoption of Windows 8 in kiosk-based applications on airport terminals, ATMs, etc. Windows 8 is integrated with Windows Live (along with other cloud based apps), and users will be able to log on to Windows 8 using their Windows Live ID. Browser history, settings, themes, email accounts, and various other settings will be saved in the cloud and can be shared across various devices running Windows 8. A connected address book, a Metro-styled Windows 8 app, allows users to combine their contacts from various email accounts and social networks such as Facebook and LinkedIn. Similarly, a photo app using Windows Live account can share photos from other services such as Facebook and Flickr.

Windows 8 also leverages Microsoft's SkyDrive, a cloud-based storage service that provides every Windows Live user with 25 GB of storage capacity. Users can access files in SkyDrive just as they do in the local file system. Further, using the Live APIs for SkyDrive, developers can build their own cloud-connected Metro apps.

In summary, Microsoft has strong evolving platform, good applications, emerging device base footprint and fairly open standards/compliance for interoperability with non-Microsoft applications and devices. Windows 8 as an OS for multiple devices (except smartphone) along with integration of cloud apps makes Microsoft platform ecosystem as one of the strong choices in consumer cloud.

4.5.4 Amazon

Amazon has a strong platform offering as Amazon Web Services; they are leader in the platform space. In applications space, Amazon Cloud Drive provides 5 GB free storage for uploading and managing files (documents, music, videos, etc.). Amazon Cloud Player allows you to play music from the web onto desktop or Android [6].

Amazon app stores provide users options to buy applications online for playing music, games, etc. Various applications available on app store can also be bought from [7]. Amazon's app store for Android works in the same manner as the Amazon website for other devices, where user can view recommendations about apps and buy it using credit cards.

In devices strategy, Amazon has Kindle for reading books (buy once and read anywhere). Kindle helps consumer do a lot of online functions improving overall productivity. On the device you can carry to the tune of 3,500 books (ebooks library in itself). While reading you can highlight content, mark online notes with the ability to clip and add notes, bookmark pages and review documents (pdf). Online you can purchase and read magazines, news, books and blogs. Once purchased books are backed up in Kindle which can be downloaded on any Kindle device. You can also share clipped messages on Facebook and Twitter from Kindle.

Amazon's new 'Fire' Android Tablet is already out in the market competing with Apple iPad. It provides free cloud storage for all the Amazon content. It allows you to play movies, games, songs, read books and browse the web using its silk browser.

In summary, Amazon has strong platform, vast application based through Amazon app store, supports open standards and compliance and has a nice emerging device strategy.

4.5.5 Integrated Vendor Landscape View

All the companies we have discussed and depicted in Tables 4.1 and 4.2 have more or less similar building blocks (platform, applications and devices) to deliver Apple iCloud like Integrated Service offering (turn ON intelligent unified integrated experiences) in the consumer segment.

Additionally, each of them has one or more product offering providing 'integrated experience' similar to the aspects mentioned in Fig. 4.2.

While Apple is leading innovations in consumer cloud with its iCloud platform, others like Google, Microsoft and Amazon are not far behind with new product offerings to establish themselves in consumer cloud.

4.6 Factors Influencing Consumer Choices

There are many factors influencing customer choices. We are not touching upon various factors that are common for choosing any particular brand. Assuming that all brands are reliable and more or less equal, following are some of the key influencers.

4.6.1 Costs

In consumer cloud, marketplace cost is an important element in the complete equation. New vendors into this space will try to gain customer share by offering consumer cloud services and products at relatively lower costs.

4.6.2 Loyalty and Brand Name

It is difficult to make Apple customer switch to Google or Microsoft offering and vice a versa. Consumer loyalty to a specific brand is an important factor and will drive choices.

Table 4.1 Company-specific offerings for each dimension

	Dimensions			
Company	Platform	Applications	Devices (h/w, OS or both)	Integrated service offering
Apple	Runs on Microsoft Windows Azure and Amazon AWS	iTunes, iBooks, QuickTime, Safari (browser), etc.	iOS iPod iPhone iPad Mac PC	iCloud
Google	Google app engine	Gmail, Calendar, Google Docs, Picasa web albums, Google Talk, YouTube, Google Maps, Orkut, Chrome (browser), etc.	Google Android, Google Chromebook	Google apps for education
Microsoft	Windows Azure, Microsoft BPOS	Windows Live, Windows Live Mesh, Windows Mail, Windows Live Messenger, Bing (search), Microsoft MapPoint, Microsoft Media Player, Microsoft Money, IE9(browser), etc.	Windows 8, XBox Kinect, Windows Phone 8 (Mango)	Windows Live-connected apps
Amazon	Amazon Web Services	Amazon Cloud Drive, Amazon App Store, etc.	Amazon Kindle, Amazon Fire	

Table 4.2 Company offering ratings in specific dimension

	Dimensions				
Company	Platform	Applications	Devices (h/w, OS or both)	Integrated service offering	Open standards/ interoperability (openness)
Apple	**	****	*****	*****	**
Google	****	*****	****	****	****
Microsoft	*****	****	****	***	***
Amazon	*****	****	****	***	****

*****Strong (relatively best offerings in market); *Weak

4.6.3 Service Offering Variations

Considering today there are variations in experience, interoperability and integrated services of each of the vendor stack, initially consumer's preferences in this would drive choices.

4.6.4 What Happens to First-Mover Advantage?

Very often, we come across the discussion of the 'first/early'-mover advantage. Definitely the 'first/early' mover has an advantage, but the one catching up need not be written off as most often we tend to do. In smartphone marketplace, early market leader (April 2010) RIM has to shell out its market share to Google Android and Apple iPhone [8].

4.6.5 Enterprise or Consumer?

Consumer may want to seamlessly manage office (enterprise) and personal information from the same device than having to maintain independent device, one for office and the other for personal work. To cater for such segments, companies having offerings in both enterprise and consumer segment may get preference. Hybrid players (players having offerings in consumer and enterprise space) such as Amazon, Google and Microsoft may command an edge as they have offerings in both the segments as against Apple.

4.7 Consumer Cloud Versus Enterprise Cloud

Consumer cloud provides a testing ground for a lot of innovations in consumer computing space before a specific aspect in consumer cloud is matured and starts getting adopted in enterprise cloud. This is because the stakes of adoption within enterprise cloud in terms of success or failure are always high. Additionally:

- Enterprise cloud demands stringent rules for governance, security, privacy, control, scalability and SLAs as against consumer cloud.

- Enterprise cloud has significant influence on enterprise top line and bottom line, while in the early days consumer cloud will need business case justification which may not be straightforward.
- The penalties of breach in contract with enterprise would be very high as the complete enterprise is at stake as against consumer cloud where it is dealing with individual consumers one to one.

But selective adoption of consumer cloud aspects in enterprise cloud would help to significantly enhance information worker (enterprise consumer) user experience and productivity which can have a direct impact on enterprise bottom line.

4.8 Opportunities in Various Industries

Consumer cloud is already changing lots of conventional ways of doing things in various industries such as education, media, automobile and health care. Let us take a look at a few industries in this context.

4.8.1 Media

Ability to store TV programs on cloud is one of the core needs we have seen in the beginning of this chapter. Having it on cloud helps you personalize programs for viewing anytime anywhere using any device. Microsoft's Mediaroom makes this possible to a large extent [9]. Similarly as discussed earlier, Amazon's 'Fire' and Kindle make it possible to access and read digital content (newspapers, magazines, books) anytime anywhere. Listening music, watching videos and using Apple iCloud platform is a completely different experience. Thus, several functions which otherwise a consumer needed to perform from his designated location using a specific device (watching TV from drawing room or reading books) are possible in a ubiquitous manner.

A media cloud can help to enhance consumer experience with on-demand anytime anywhere any device interaction, rich analytics to understand reading, listening, viewing tastes and behaviours for contextualizing specific campaigns, program archival for future viewing and with digital rights management to avoid piracy.

4.8.2 Education

Virtual classrooms and games aided learning and simulations-based learnings are some of the things that can be made more effective by leveraging Cloud to make anytime, anywhere using any device. IBM has set up Cloud Academy for education [10].

4.8.3 Insurance and Health Care

More and more insurance companies are moving to B2C self-service channels (portals, mobiles, kiosks) to sell their products and pass on the benefits to customers.

Insurance companies want to capture and track all the information about their customers, their histories and demographics which help them in not only selling the right products but also ensuring right price coverages. Consumer cloud can help achieve this in a very fluent and cost-efficient manner. Similarly, in health care, managing electronic patient records is an important trend; having consumer cloud would create several opportunities for consumers, such as collaborating with 'patients like me'; pharmacies/clinics providing services such as sending reminders/alerts for medications and tracking patients understand which medicines are prescribed most by doctors and their impacts.

4.8.4 Automobile

There are also early signs of cloud adoption for consumer applications in the automobile industry.

Ford and Toyota are building several cloud-based applications by opening up and integrating the car dashboard and cloud platform. Ford's MyFord Touch platform links mobile apps like Pandora with the car's digital screens and voice-control capabilities [11]. Ford Evos is a cloud-connected concept car run using smartphone apps. It is an Internet-connected vehicle and would provide several Internet-enabled services like scan weather and traffic reports in real time, route suggestions based on road conditions and fuel efficiency on the specific road based on past driving and traffic data, to name a few [12]. According to [13], 'A social network designed exclusively for Toyota, connecting owners, dealers, customer service representatives, etc. As a Toyota customer, this should greatly enhance your driving experience by allowing you to interact with other Toyota owners, your dealer and the Toyota customer service seamlessly. For the company, not only this will lead to greater customer satisfaction, it will also allow optimization of resources to ensure those customers are satisfied with maximum efficiency and minimum cost'.

4.8.5 Retail

The retail industry can benefit tremendously from a consumer cloud as the success in retail industry depends on attributes such as competitive pricing, real-time information availability of inventory stock levels and running targeted

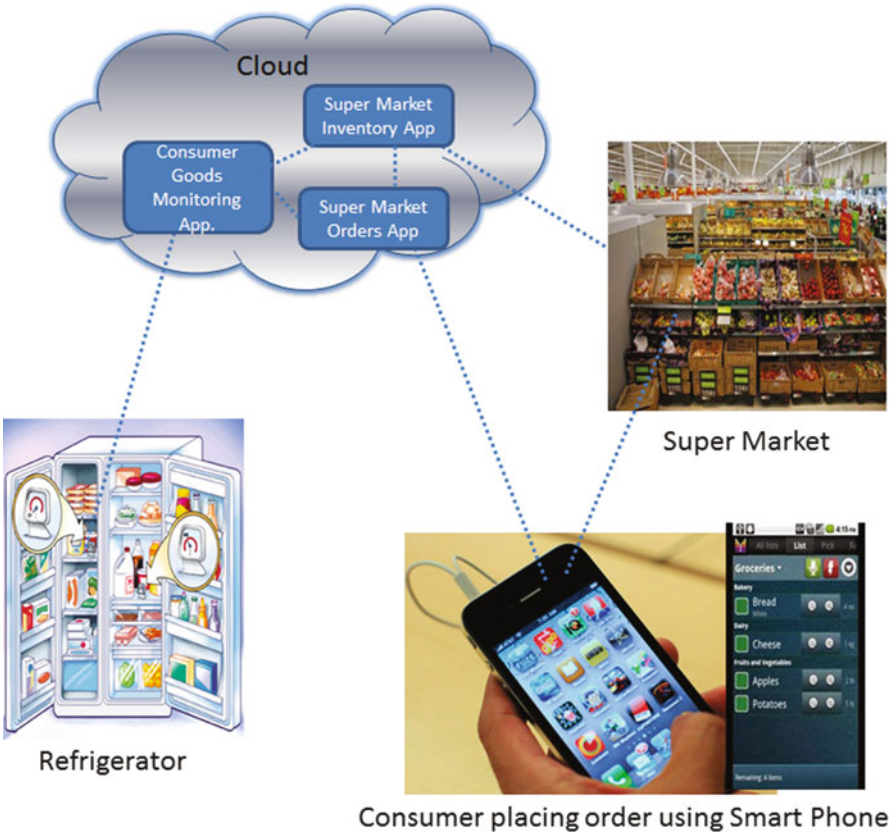


Fig. 4.4 Consumer cloud for a retail scenario

campaigns. There are several scenarios which can benefit from a consumer cloud. Let us take a look at one of the day-to-day scenarios. The workflow can be seen as in Fig. 4.4.

Home refrigerator is designed with the ability to communicate with consumer 'goods monitoring app' hosted on the cloud. In turn this app communicates with a supermarket (e.g. Walmart) app, such as supermarket 'inventory app' which checks for the status of inventory. Supermarket 'order app' on cloud sends an email or SMS to the consumer based on supermarket stock/inventory status suggesting items finishing in the consumer's refrigerator to buy so that they can be shipped at the door. The complete interaction is depicted in Fig. 4.3.

This workflow can be further personalized by the supermarket's ability to weave additional intelligent marketing campaigns based on consumer consumption pattern of vegetables, fruits, etc. suggesting items with discounts by giving offers, targeting customer to advertise supermarket's own products similar to the branded ones that they often buy, etc.

4.8.6 Home Scenarios

Next set of consumer services could possibly be around providing integrated experience across various devices at home such as boilers, air conditioning, refrigerator, washing machine, microwave, lights, TV, phone, security systems and car, and possibly more as they start having a meaningful dialogue by leveraging cloud capabilities to deliver you a customized unified experience. Note the device interactions will happen through cloud as hub and devices as spoke and not peer to peer. Some of the scenarios are:

- Ability to switch on (control) your home lights and air conditioning from your car as you approach near to your home
- As you get up in the morning, watch news, boiler warms up water, signals your water is ready and switches off TV as soon as you leave the room
- Control home security systems from remote locations

4.9 Delivering Wireless Data Through LED Lights

Harald Haas, a professor of engineering at Edinburgh University, has recently demonstrated 'delivering wireless data through LED lights', and he calls it as 'Li-Fi' for Light Fidelity. Benefit from application of this concept would be humongous. In Haas own words: 'In hospitals, for new medical instruments; in streets for traffic control. Cars have LED-based headlights, LED-based back lights, and cars can communicate with each other and prevent accidents in the way that they exchange information. Traffic lights can communicate to the car and so on. And then you have these millions of street lamps deployed around the world. And every street lamp would be a free access point' [14].

Similarly, in enterprise, you may have various applications and devices interacting to deliver similar self-service, intelligent experiences.

4.10 Regulations and Standards

As more and more devices like refrigerators, washing machines, boilers and home alert systems need to talk to each other through cloud to deliver intelligent integrated experience, it would mean a few things such as the following:

- Need for building applications/devices to communicate
- Establishing standards for devices to communicate in terms of data interoperability and communication protocol
- Establish partnerships across a diverse set of vendors and service providers
- Regulations to strengthen privacy and security concerns

4.11 Summary

Adopting consumer cloud will mean more efficiency through saving time, cost and improving productivity for consumers. Apple, through its iCloud offerings, has set the ball rolling. Google, Microsoft and Amazon are rolling out various product-specific offerings as discussed in this chapter. From the example of Ford and Toyota, there are early evidences of consumer cloud adoption in the automobile industry. The media and education industry have started utilizing consumer cloud for penetrating their services and providing true anytime, anywhere, any device experience. Retail, health care and other industries which directly deal with consumers can significantly leverage consumer cloud for improving customer engagements to improve bottom lines in competitive markets.

It would be a matter of time when consumer cloud starts spreading to more electronic devices and industries to deliver cohesive integrated intelligent experiences. However, it would mean more hardware, software vendor partnerships and work for standard bodies.

Acknowledgements Authors would like to acknowledge the effort of Ramesh Ramchandran, Lead Client Solutions, Infosys Labs, Infosys Ltd., for his help in shaping this up.

References

1. Top 15 Consumer Cloud services/applications. http://www.devx.com/architect/Article/46190
2. Consumer cloud criteria. http://www.juniperresearch.com/analyst-xpress-blog/2011/06/22/what-is-a-consumer-cloud-service/
3. Google Apps: http://www.google.com/apps/intl/en/group/index.html
4. Google Apps for education. http://www.google.com/a/help/intl/en-GB/edu/index.html
5. Microsoft Xbox Kinect SDK release announcement. http://www.microsoft.com/Presspass/press/2011/jun11/06-16MSKinectSDKPR.mspx?rss_fdn=Custom
6. Amazon Cloud Player: https://www.amazon.com/clouddrive/learnmore; http://www.pcworld.com/article/223604/amazon_cloud_drive_and_cloud_playerahandson_tour.html
7. Amazon App store: http://www.amazon.com/mobile-apps/b?ie=UTF8&node=2350149011&tag=vglnk-c43-20
8. Mobile subscriber market share. http://www.prnewswire.com/news-releases/comscore-reports-march-2011-us-mobile-subscriber-market-share-121396514.html
9. Microsoft Media room. http://www.microsoft.com/mediaroom/default.aspx
10. IBM cloud for education. www-03.ibm.com/press/us/en/pressrelease/28749.wss
11. Ford Car – Cloud Connected Dashboard: http://www.simplysecurity.com/2011/04/20/consumer-cloud-taking-shape/
12. FordEvos:http://www.cloudtweaks.com/2011/09/cloud-computing-and-your-car-%E2%80%93-part-1/
13. Toyota'sConnectedCarvision.http://www.cloudtweaks.com/2011/09/cloud-computing-and-your-car-%E2%80%93-part-1/
14. Delivering wireless data through light, Prof. Hass: http://www.smartplanet.com/blog/business-brains/wireless-data-can-be-delivered-by-led-lights-anywhere-call-it-8216li-fi/18180

Part II
Cloud Enablement and Management

Chapter 5
Role of Service Brokers in Cloud Computing

Dolly Kandpal

Abstract Cloud computing is currently the most hyped and popular paradigm in the domain of distributed computing. Multiple vendors are providing infrastructure, platform, and software-related cloud products or solutions, and the number of such vendors is growing rapidly. But the reality is that there is very little in the form of standards in general and interoperability standards in particular. This leads to a challenging situation for cloud service consumers as they need to use solutions provided by multiple cloud vendors in order to fulfill their business requirements, and the lack of standardization leads to lack of interoperability among the solutions provided by varied vendors. Service brokers act as intermediaries between the cloud service providers and consumers. The role of service brokers becomes critical in situations that need integration among multiple systems as they bridge the interoperability gaps among multiple cloud solutions and thus compensate for the lack of standards. They do so by acting as agents adding value on behalf of the service consumers. The usage of such cloud brokers also helps to future-proof the solution as service brokers add a layer of indirection, thus enabling loose coupling among the cloud service consumers and providers. The aim of this chapter is to review the role of cloud service brokers in cloud computing. The review includes consolidation of the definitions and characteristics of cloud service brokers, review of the current brokerage solutions, and interfacing requirements. Also presented are a high-level cloud service broker functional architecture, criteria for selection of a cloud service broker, future trends and review of benefits, and challenges of using cloud service brokers.

Keywords Cloud computing • Cloud service broker • Service brokerage • Service aggregation • Service arbitrage • Service intermediation

D. Kandpal (✉)
Qamlo Information Services Limited, 468/530 Petchburi Road, 10400 Bangkok, Thailand
e-mail: dolly.kandpal@qamlo.com

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_5,

5.1 Introduction and Motivation

Cloud computing is fast gaining popularity as the preferred way of accessing computing resources (infrastructure and services) over the Internet for an enterprise. According to Gartner [1], by 2015, 50 % of Global 1000 enterprises will rely on external cloud computing services for at least one of their top ten revenue-generating processes, and by 2016, all Global 2000 companies will use some level of public cloud services. Also according to Gartner [2], IT Spending Forecast Summary, Q2 2012 Update, the migration to public cloud services is one of the hottest topics in IT, and forecast spending in this area, at an annual average growth rate of 18 %, is growing more than four times faster than spending on overall IT.

What does the above mean for the IT industry, specifically for the segment of the industry focusing on cloud computing? The competition is intense among the cloud solution providers, and there are multiple solution providers focusing on provision of infrastructure, platforms, or software-related cloud services. The number of such providers is increasing very rapidly, and each provider is trying to differentiate itself from the other by adding some features or capability not provided by others. The other reality is that although there is an increasing focus on cloud interoperability, including a number of interoperability initiatives like the Open Virtualization Format [3] from Distributed Management Task Force, Inc. (DMTF) [4], Unified Cloud Interface [5] from Cloud Computing Interoperability Forum [6], Open Cloud Computing Interface [7] from Open Grid Forum [8], Cloud Security Alliance Security Guidance [9] from Cloud Security Alliance [10], Cloud Storage Initiative [11] from Storage Networking Industry Association (SNIA) [12], and the OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA-v1.0) [13] from Organization for the Advancement of Structured Information Standards (OASIS) [14], most of these standardization efforts are at very early stages of development, and there is very little currently available in terms of actual standardization among the cloud service providers. Each provider may use a different platform, application programming interface (API), programming language, and architecture. This leads to low interoperability among the individual solutions.

What precisely is the impact of the above factors on an enterprise attempting to migrate its applications to the cloud? The typical situation within a medium-to-large-sized enterprise is that it has multiple heterogeneous platforms for the fulfillment of its business requirements, and it needs integration among its already existing applications to fulfill its crosscutting business processes. This is the situation even without taking into account the migration of some of these processes to the cloud environment. In order to effectively fulfill even a single business process, an enterprise consuming cloud services may practically need to interoperate with multiple cloud service providers to take advantage of the specific offerings and the best-of-breed solutions provided by these different providers. This is especially true for large-scale scenarios where the consumer is building up their entire business processes using the services provided by various cloud service providers and vendors.

In order to make use of the cloud services provided by even a single cloud vendor, the enterprise needs to integrate its existing business processes to the offerings of

the cloud vendor by mapping its existing implementation to the interfacing and functional requirements of the vendor. This may mean integrating at an infrastructure, platform, or API level with that provider. This is a challenging exercise on its own, considering that this integration impacts the entire lifecycle of the processes and actual running, live systems and processes. This challenge is compounded by the fact that it may be necessary for an enterprise to integrate with a number of different cloud service vendors for the fulfillment of its business processes. This means creation of a number of such integrations, as each of the cloud vendor will most likely use different interfaces for the services it provides. This leads to a paradoxical situation, as one of the main motivators of usage of cloud computing is to allow the employees of an enterprise to focus on their core business, rather than solve the information technology-related issues. In case the enterprise migrates its business processes to services provided by various cloud vendors, employees of such an enterprise are faced with the situation of having to understand the (probably different) ways of invoking multiple cloud services and implementing such integrations for fulfilling a single business process. In some cases, the enterprise will not have the technical or commercial capability to perform such multiple integrations with the multiple cloud service providers. In other cases, the commercial and technical capability may be there, but the willingness to perform the tasks may not be there, as the resources may be foreseen for performing core business tasks.

Usage of cloud service brokers serving as intermediaries between the cloud service consumer and the service providers is a pragmatic way of resolving the above paradoxical situation.

This chapter aims to elaborate on the concept of cloud brokers or cloud service brokers and the role they play in cloud computing. Section 5.2 provides an overview of the concept of cloud service brokers including definition and specific characteristics of cloud service brokers that distinguish them from the service providers and the consumers. Section 5.3 reviews current brokerage solutions based on categories of services provided, and Sect. 5.4 details the value of using cloud service brokers. Section 5.5 describes the metrics needed for interfacing to the cloud service brokers; the metrics are of the business and technical types. Section 5.6 details the role played by semantic technologies in provision of cloud services by a service broker by reviewing a few cases. Section 5.7 provides possible high-level cloud service broker architecture in terms of the functionality provided by a service broker. Section 5.8 lists the criteria for selection of a cloud service broker, and Sect. 5.9 reviews the future trends. Section 5.10 concludes the chapter by reviewing the benefits as well as challenges of using cloud service brokers.

5.2 Overview of Cloud Service Brokers

Cloud service brokers (CSB) are intermediaries that are positioned between a cloud service consumer and one or more cloud service providers. They form a layer of abstraction between the cloud service consumers and cloud service providers such

that the consumers see a single cohesive view of all the provided services. The cloud service brokers thus mainly provide brokerage services to the service consumers. They may also perform consolidation and value addition to the services provided by one or more cloud service providers.

5.2.1 Definitions

National Institute of Standards and Technology (NIST) [15] defines a cloud service broker as an entity that manages the use, performance, and delivery of cloud services and negotiates relationships between cloud providers and cloud consumers. According to Frank Kenney, research director at Gartner [16], a broker might be software, person, appliances, platforms, or suites of technologies that enhance the base services available through the cloud.

Forrester [17] defines cloud broker as a business model that leverages skills and capabilities from all three of the traditional business models of software, consulting, and infrastructure to offer a wide value contribution in the emerging cloud space.

5.2.2 Characteristics

Cloud service brokers are not service providers themselves and do not host any applications/services. They also do not provide infrastructure or other resources directly to the clients. Their job is to perform value-added brokerage between cloud service consumers and providers.

There may be a one-to-one, many-to-one, one-to-many, or many-to-many relationships between the cloud service broker and the cloud service provider and the cloud service broker and the cloud service consumer.

Thus, the key characteristics of a cloud service broker are that a service broker:

- Is an intermediary
- Provides a single interface for dealing with multiple clouds
- May be software, appliances, platforms, or suites of technologies – it does not necessarily have to be software
- Does not provide any services on its own
- Is not able to modify an existing service provided by a cloud service provider
- Does not own the service provided by a cloud service provider
- Does not control the actions of either the cloud service consumer or the cloud service provider
- Manages the relationship between cloud service consumer and provider
- Performs consolidation
- Adds some value to the interaction

5.3 Categorization of Cloud Service Brokers

There may be multiple ways of categorization of cloud service brokers. The categorization used here is according to the functionality provided by the cloud service brokers. According to NIST [15], there are three categories of cloud service brokers based on the functionality provided by the service brokers. They are the following:

- Service aggregation: A cloud service broker combines and integrates multiple services into one or more new services. The service broker will provide data integration and ensure the secure data movement between cloud consumer and multiple cloud providers.
- Service intermediation: A cloud service broker enhances a given service by improving some specific capability and provides the value-added service to cloud consumers. Examples of capability improvements include identity and access management, performance reporting, and security enhancements.
- Service arbitrage: Service arbitrage is similar to service aggregation, with the difference that the services being aggregated are not fixed. Service arbitrage allows flexible and opportunistic choices for the service broker. For example, the cloud service broker can use a credit-scoring service and select the best score from multiple scoring agencies.

Gartner [18] provides a way of categorization based on roles played by the cloud service broker that help define how information technology (IT) service providers accommodate cloud computing in their service offerings. The roles are the following:

- Aggregation brokerage: A cloud service broker can bundle many individual services together and present them as a unified service.
- Customization brokerage: Cloud service broker customizes cloud services for individual customers by adding identity and access management or security- and policy-related features.
- Integration brokerage: An enterprise will often rely on a cloud service broker to bring integrate multiple services, collectively providing new functionality, say, in the form of a fulfilled business process.

Other additional roles include the following:

- Arbitration brokerage: Similar to service arbitrage above
- Governance brokerage: Cloud service broker that specializes in governance-related brokerage activities like quality of service (QoS) and SLA management

5.4 Value Proposition for Using Cloud Service Brokers

As mentioned in the Introduction and Motivation section of this chapter, as the cloud offerings become widely used and increasing number of cloud services are available, there is increased value in using an intermediary. This is because the

service consumer has to deal with a number of aspects like integration, identity and access management, governance, service management, service-level agreement management, and security per service provider.

An intermediary/cloud service broker performs a number of services/activities on behalf of the service consumer and thus takes over some of the complexity from the cloud service consumer. The usage of service broker thus simplifies the usage of cloud services for the cloud consumer.

The cloud service brokers keep track of the latest offers and simplify the job of interfacing with one or more cloud offerings from the point of view of the consumer. This results in overall simplification, not just in terms of interfacing and integration but also of the overall maintenance, management, accounting, billing, and SLA fulfillment for the solution. The cloud service brokers thus provide specialized expertise to allow an enterprise to focus their limited resources including capital on the core activities. They also allow the employees of an enterprise to focus on their core competencies and core business rather than having to deal with information technology-related issues. The enterprise can be more agile and respond to changes due to the introduction of services provided by cloud service brokers and also gets the collective benefit of the latest advances provided by the individual cloud services providers.

The following subsections provide a list of high-level benefits (divided into functional categories) of using a cloud service brokers. The categorization below is mainly for the purpose of defining the major benefits of the corresponding cloud service brokers, whereby the categories themselves may have interrelationships and intersections among each other.

5.4.1 Simplification and Abstraction

The key value provided by the cloud service broker is in the area of simplification of consumption of services by a cloud service consumer. Rather than dealing with multiple different APIs and interfaces, the service consumer sees a single interface/API. This helps with the existing problem of lack of standards as the service brokers provide a single interface for interacting with possibly multiple clouds. In addition to easing the job of consuming services, it also makes future refactoring easy as the service broker forms a layer of indirection between the service provider and the consumer and shields the consumer from future changes by the service provider. An example of this type of cloud service broker is Apache [19] Deltacloud which provides an API that abstracts the differences between clouds.

5.4.2 Aggregation

A cloud service broker may provide an aggregation of multiple services by various cloud vendors – by means of cloud services composition as a single service to the service consumer. Thus, the composition (which may again be dynamic) of services

provided by different cloud service providers and the associated complexity and orchestration is performed by the service broker rather than the consumer. This may be thought of as cloud aggregation/orchestration. An example of this is Aepona's Services Aggregation and Merchandizing platform [20] which provides tools that enable service providers to aggregate, merchandize, and deliver a wide range of cloud services. One of the key benefits of aggregation is also that the cloud service brokers are able to provide the combined demand for services for a number of cloud service consumers, and this increased buying power helps them to negotiate better conditions on behalf of their clients with the cloud service provider. The cloud service providers, on the other hand, get an increased number of clients as a result of this integration.

5.4.3 Value Addition

A cloud service broker may perform the function of value addition to the services provided by the cloud service provider. The following are the major areas in which the value addition may be provided.

5.4.3.1 Security, Privacy, and Trust Management

Security, privacy, and trust are the key issues that enterprises consider when moving to the clouds, and the lack of these is a major deterrent for organizations planning to use cloud computing. Cloud service brokers may act on behalf of the providers and/or consumers to provide the intermediary services that alleviate one or more of the above concerns.

5.4.3.2 Compliance and Auditing

Compliance to location-specific and/or domain-specific regulatory requirements like those pertaining to Sarbanes-Oxley Act (SOX) [21], Health Information Portability and Accountability Act of 1996 (HIPAA) [22], Payment Card Industry Data Security Standard (PCI DSS) [23], and European Data Privacy Framework [24] is a functionality that may be provided by a cloud service broker.

5.4.3.3 Management and Monitoring

Management and monitoring of services provided by one or multiple clouds may be performed by a cloud service broker on behalf of the service consumer. Such services may include provisioning, monitoring, job scheduling, policy-based automation, analytics, resource management and virtualization, and workflow and workload management.

5.4.3.4 SLA Management

SLA management with possibly multiple providers on behalf of a service consumer is another functionality provided by the service brokers. This helps in the SLA management from the point of view of the service consumer. Some of the examples of cloud service brokers in the value-addition category are Novell's Cloud Security Service [25] and CloudSwitch [26].

5.4.4 Access to Most Suitable Services

As the cloud service providers are proliferating, it may be difficult for the service consumer to keep track of the latest cloud services offered and to find the most suitable cloud service providers based on their criteria. In such cases, the service broker performs the cost calculation of the service(s), thus performing the analysis on behalf of the consumer and providing the most competitive service to the consumer from the palette of available services. This may lead to consumption of the service from a new service provider providing the service at better conditions (based on matching criteria like SLA, costs, fit, security, energy consumption). Thus, the service broker may be able to move system components from one cloud to another based on user-defined criteria such as cost, availability, performance, or quality of service. Cloud service brokers will be able to automatically route data, applications, and infrastructure needs based on key criteria such as price, location (including many legislative and regulatory jurisdictional data storage location requirements), latency needs, SLA level, supported operating systems, scalability, backup/disaster recovery capabilities, and regulatory requirements. There are a number of frameworks and solutions that provide examples of this functionality. Some of them are the RESERVOIR [27], a framework that allows efficient migration of resources across geographies and administrative domains, maximizing resource exploitation and minimizing their utilization costs; the Intercloud [28] environment which supports scaling of applications among multiple vendor clouds; and Just in Time [29] broker which adds value by offering cloud computing without needing to take care of capacity planning but simply discovering, recovering, and reselling resources already amortized and idle. Another approach is provided by FCM [30], a meta-brokering component providing transparent service execution for the users by allowing the system to interconnect the various cloud broker solutions, based on the number and the location of the utilized virtual machines for the received service requests.

5.4.5 Reliability, Failure Handling, and Disaster Recovery

Service brokers may detect cloud failures and react in an appropriate way to those failures. An example of this is that they may move infrastructure elements from one cloud (public or private) to another in case of failure. A cloud service broker can

improve reliability and decrease vendor lock-in by spreading infrastructure across multiple clouds and can enable disaster recovery into a secondary cloud. Nebulas [31] is an example of a context-aware system that provides failure handling over heterogeneous environments. Business service management solution from HP [32] collects system information in real time using alerts instead of scanning and combines it with tools for determining the root causes of a problem and automated remediation capability to provide reliability.

5.4.6 Environment Protection

Cloud service brokers may be used to reduce the carbon footprint by optimizing distribution of resources and making data centers energy efficient by using consolidation and virtualization. Carbon Aware Green Cloud Architecture [33] is an example of such a broker-based architecture which promotes carbon efficiency and addresses the environmental problem from the overall usage of cloud computing resources.

5.5 Metrics for Interfacing to Cloud Service Brokers

The cloud service brokers provide mediation, optimization, and value-addition functionality based on either business or technical requirements of the corresponding cloud consumers. It is based on these business requirements, specified by means of defined metrics, that the cloud service brokers provide the best fitting service to the consumers and/or perform necessary selection and filtering of the cloud service providers.

The following subsections list the relevant metrics for interfacing with the cloud service brokers by categories.

5.5.1 Business Metrics

Here is a list of a number of relevant metrics:

- Cost of service: Is one of the most important criteria and may be quite difficult to specify and calculate. Cost of service may be divided into one-type and ongoing fixed costs. The cost may be based on per hour/per transaction/volume of transfer, quota, time of request, availability of resources, and so on. Other criteria may be commercial vs. noncommercial provider usage, time, and/or cost optimization scheduling. It is one of the most important but also the most difficult parameter to monitor and regulate, as it involves comparison of costing data for existing services to newly available services as well. Costing data itself is complicated

and may involve multiple sub-parameters like fixed one-time costs and variable costs depending on time and demand.
- Regulatory criteria-driven requirements: Regulatory requirements which need to be fulfilled by the cloud consumer are further taken over to the cloud service broker; examples of such data are geographical constraints like location of data, logging, audit trails, and privacy.
- Data-related requirements, e.g., data security, data location, backup, and storage.
- Maximum tolerable latency: This is typically a technical metrics, but in case a business process has some specific response time-related constraint, then it forms part of the business parameter.
- Business SLA requirements, including availability guarantees, availability penalties, and penalty exclusions.
- Role-based access control: These are business-related access control metrics pertaining to hierarchy of the consumer organization.
- On-demand business process deployment and user or application constraint management like management of budget and deadline.
- Governance criteria like quality of service, policy enforcement, and management.
- Environmental constraints, e.g., preference for Green providers and incentives for Green providers.

5.5.2 Technical Metrics

Here is a list of a number of relevant metrics:

- Virtual machine requirements, say, those related to central processing unit (CPU), memory, and operating system
- Security-related requirements like details related to encryption, digital key management, and identity and access management
- Technical SLA requirements, including alerts and notifications when SLA terms reach a critical point
- Maximum provisioning time
- Redundancy and disaster recovery metrics for critical applications or functionality
- Environment safety and protection-related metrics like carbon emission rate and CPU power efficiency

5.6 Role of Semantic Technologies to Support CSB

There are various cloud service providers who provide specialized services. Any medium- to large-scale enterprise would need to interface with one or more such cloud service providers in order to fulfill their computing and/or resource requirements. Different vendors use different platforms, APIs, and languages – also the

semantics, including data semantics, data definitions, meaning, and data model as well as granularity level of the services provided by such vendors, are different. Thus, interoperability and portability is a challenging issue among cloud service vendors.

As discussed in the first section of this chapter, cloud service brokers are entities that support the cloud service consumers perform these tasks. Such cloud service brokers are currently mainly able to provision resources and infrastructure dynamically. The activities of service discovery, negotiation, provision, aggregation, monitoring, analysis, and deployment of security, QoS, and other policies still require human intervention and have to be done statically.

Semantic technologies help in performing data modeling mapping and capturing definitions and relationship among entities by means of ontologies [34]. This helps to create a common understanding of the entities among the various roles (cloud consumer, broker, and provider), and this creates a common understanding and facilitates dynamic behavior.

In this section, two initiatives that perform intermediary semantic functionality to improve interoperability and portability are discussed. The first such initiative is mOSAIC – an open-source cloud application programming interface and a platform targeted for developing multi-cloud-oriented applications [35]. It provides a platform which:

- Enables interoperability among different cloud services
- Eases the portability of developed services on different platforms
- Enables intelligent discovery of services
- Enables services composition
- Allows for management of SLAs

The key entity which allows for interoperability and resources description in the above framework is the *Cloud Ontology*. It is developed in Web Ontology Language (OWL) [36] and OWL-S languages [37] and can be used for semantic retrieval and composition of cloud services. Such a Cloud Ontology provides a common definition of concepts related to cloud domains and describes cloud components like infrastructure, platform, and services and thus bridges the gap between domain-related functionalities and cloud resources and services. OWL and OWL-S are ontology languages which provide a general method for the conceptual description or modeling of information that is implemented by actual resources.

Another interesting initiative is Semantic Cloud Services Tool [38], a semantically rich, policy-based framework that facilitates the automation of the lifecycle of virtualized services. The main component of this framework is a service lifecycle ontology [39] – an ontology written in OWL2 [40] that conceptualizes the five phases of requirements, discovery, negotiation, composition, and consumption of an IT service lifecycle on a cloud. A tool including a cloud service broker component has been developed using Semantic Web technologies like Simple Protocol and RDF Query Language (SPARQL) [41], Resource Description Framework (RDF) [42], and OWL to represent and reason about services and service requirements.

Currently this tool automatically discovers, negotiates, and consumes services from the cloud for a specific use case around storage services. It is built upon the service lifecycle ontology.

These two initiatives show that the future direction of the cloud services brokering is semantics-enabled to enable interoperability and portability by creating a common understanding in the form of ontology based on standards like OWL, OWL-S, RDF, and SPARQL.

5.7 High-Level Functional Architecture for CSB

The cloud service broker architecture must enable and facilitate provision of optimal scheduling strategies and deployment of virtual services across multiple clouds. These services may be based on different optimization criteria like cost optimization or performance optimization and fulfill different user constrains like budget, performance, instance types, load balancing, instance prices, and service workload.

The high-level conceptual architecture presented in Fig. 5.1 is a depiction of the functionality that a cloud service broker may provide. All the cloud service brokers may not provide all the functionality, and additional functionality may be added if the need arises.

The cloud service broker provides varied capabilities to the cloud consumer. In order to fulfill its task, the following components are needed:

- Service management: This is the most critical component of the cloud service broker and performs the tasks related to service discovery, service selection (in conjunction with other functions like semantic engine and SLA and QoS management), service provisioning, and service de-provisioning. The allocation of services is done in a manner to preserve on-demand and elastic nature of cloud services provisioning.
- Metering/billing: This functionality keeps track of the services consumed along with any aggregations and discounts and other pricing-related information (in conjunction with rules management and monitoring engine). This module may have integration with external third-party payment gateways and will meter data from various resources and aggregate them so that they can be rated and billed.
- Data management: Also an important functionality dealing with data and its storage and security (in conjunction with integration, transformation, security management, and SLA and QoS management modules).
- Orchestration: Business transactions are often executed by coordinating or arranging existing applications and infrastructure in the cloud to implement business processes. This involves usage of technologies and tools like ESBs, process engines, middleware, legacy, and packaged applications. Event processing may be a useful functionality to have as it provides asynchronous resource coordination and automated policy-based orchestration.

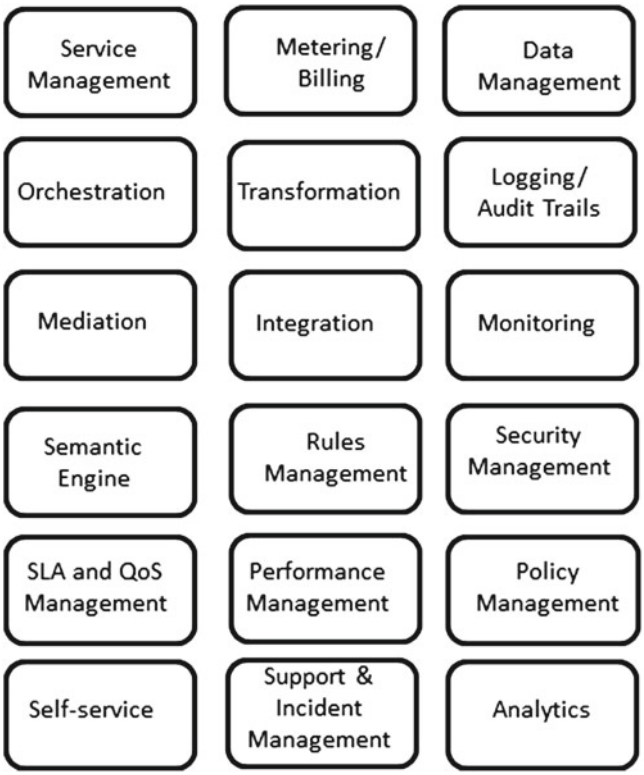


Fig. 5.1 High-level cloud service broker functional architecture

- Transformation: Involves changing an entity in one form to another at run-time, for example, transformation of an entity from one data model to another or transformation of message from one protocol to another.
- Logging/audit trails: Performs creation of logs and audit trails. The module is essential for fulfilling regulatory compliance and also for interfacing to incident management and essential for security management, SLA and QoS management, and support and incident management modules.
- Mediation: This module helps to resolve the differences between two or more systems in order to integrate them seamlessly. Mediation can happen at different levels such as security, transport, message, API, and protocol.
- Integration: Is used to facilitate the combination or aggregation of separately produced components or services to perform a larger task, ensuring that the problems in their interactions are addressed by using some intermediary tool, say mediation. This module is necessary for interfacing to the multiple service providers who do not follow the same standards in terms of protocols, technology, APIs, etc.

- Monitoring: Monitors the business activities, SLAs, holistic service status, outstanding alerts, and policy violations. This module interfaces with most other modules and provides them with relevant information.
- Semantic engine: A specialized entity (could be optional) that will support the creation of a common understanding of and relationships among entities in a domain by means of creation and usage of ontologies. This module thus helps in easier mapping and understanding of services provided by different cloud vendors and helps in creating ease of interoperability and common understanding among cloud services provided by different vendors.
- Rules management: Is a support functionality that is utilized by many other modules to perform complex decision-making and evaluation functionality and also to map the business requirements in a declarative, easy-to-use manner that allows easy update and changes.
- Security management: May include identity and access management functionality for handling user roles and access issues and security services like authentication, authorization, auditing, encryption, wire-level security, and other conventional security measures required in a distributed environment. Privacy-related issues are also handled by this module.
- SLA and QoS management: Makes use of metrics in the relevant areas, some of them being legal metrics, SLA and QoS requirements pertaining to regulatory, privacy, data security, and penalties management; interfaces with the security management, policy management, rules management, logging/audit trails, monitoring, and performance management modules; defines metrics for usage and assessment of charge-backs, promotions, and discount-related information management.
- Performance management: Handles the performance-related aspects of business processes and services and also of the underlying resources. It interfaces with certain other modules like monitoring and support and incident management.
- Policy management: Policy handling including policy creation and assessment, mapping, attachment, and deployment is performed as part of this module. Policy enforcement and escalations must also be applied as appropriate as part of functionality of this module.
- Self-service: Provides the customer with the ability to self-register and perform self-service functions including provisioning and management tasks and also administration. This module ties in with the security management module and may tie in with other modules like rules management, SLA and QoS management, and policy management.
- Support and incident management: Performance and utilization of diagnostic information at multiple levels to troubleshoot and resolve issues. Cloud management infrastructure must provide diagnostics capabilities for the full stack. Incident management aims to restore normal cloud operation as quickly as possible and minimize the adverse effect on business operations. This may include resolution of the root causes of incidents and thus minimizes the adverse impact of incidents and problems on business and prevents recurrence of incidents related to these errors. If resolution is not possible, then alternative service deployment may be needed in conjunction with service management module.

- Analytics: This module collects and makes use of historical data and provides analytical information for both internal usages of the cloud service broker but also for the cloud consumers. It provides insight into business trends and user preferences, transaction visibility, business key performance indicator (KPI) monitoring, reporting functionality, and dashboards.

5.8 Criteria for Selection of CSB

Planning for the involvement of a cloud service broker as part of the cloud strategy is necessary and important for an enterprise when there is a need for using more than one cloud service providers. There are a number of generic criteria for selection of cloud service broker and some specifically for the cloud service brokers. The criteria are as given below:

- Cost/price: This is one of the main deciding factors for the consumers of the service. This should include all the components of the cost, including one-time and ongoing costs. The transparency of the costs is also important to ensure that there are no hidden costs and so is the review of the historical price trend to see how the cost of services has changed in the past.
- Viability and reputation of the provider: Since this is a nascent field and the consumers build their business processes based on the services provided by the cloud service broker, it is very important to look at the reputation of the cloud service provider/service broker and assess their references, the number of successful projects delivered, and the duration of time for which they have been in business. It is also important to assess the long-term stability of the broker's business.
- Transparency of operational information and data: It is necessary to get the insight into the basic operational information and data related to one's operation from the point of view of the cloud service consumer, and the service broker that provides this information should be preferred.
- Security and privacy: Security provision as a value addition is one of the reasons for using the cloud service brokers in the first place. It is important to ensure that the service brokers provide an end-to-end security (including physical, application, and personnel related) for the services they provide. Equally important (also from the point of view of regulatory requirements fulfillment) is the review of how a service broker manages privacy identity and access management, and single sign-on feature provision is important. It is important that the service broker provides details of security certifications that their services may fulfill.
- Regulatory requirements fulfillment: Depending on the domain, the cloud consumers have to fulfill varied requirements (say audit trail provision, logging, etc.). The cloud service broker on behalf of the consumer must ensure that the compliance requirements are met.
- Data management: Data is an important asset of the enterprise. Functions related to data like data storage and backup, confidentiality and privacy of data, and geographical location of data are met. Data availability is important for the

consumer's business, and, thus, it is important to review the contingency plan that the service broker has in the event of a data center/service failure (either own or of the service provider's).

- SLA management: The service broker's readiness in SLA to provide some transparency into the vendor's operations in the areas of audits and compliances and its flexibility to meet SLA requirements including penalties in case of noncompliance is a key deciding factor as it impacts the fulfillment of many other requirements. The flexibility of the SLA to reflect the service consumer's business requirement, including a match to inputted translated business metric of the cloud service consumer, must be reviewed.
- Contingency and disaster recovery plans: It is important to review the contingency and disaster recovery plans that a service broker provides.
- Features of services provided: Ultimately the distinguishing factor among different service brokers may be the features of the service they offer including the ability to find the best services for the consumer based on the consumer's business and its businesses location.
- Service broker references: A review of the references provided by the service broker is a necessary step. This should ideally include the possibility to communicate with actual users of the services of the service broker.

5.9 Future of Cloud Service Brokers

The scenario of workloads being redirected across different cloud providers for real-time optimized cost/performance based on business or technical requirements is the vision toward which the cloud service brokers are working. Due to the lack of standards and also practical experience, this scenario is not yet a reality, but as seen from the use cases above, the first successful steps are being taken in that direction. This dynamism is being fulfilled currently to a certain extent, but in future, as the capabilities of the cloud service brokers improve, this will lead to finer granularity of workloads and increased dynamic behavior.

Another trend is for the cloud service brokers to become more specialized, e.g., for certain domains, like banking or insurance, or for certain specialized type of computing, like big data-related computing.

Data consolidation, specifically context based (over multiple applications and providers), is another functionality that cloud service brokers are going to provide soon. In fact, Appirio [43] SalesWorks already provides access to Salesforce Chatter [44] that feeds on contacts, accounts, and opportunities directly within Google Apps [45] for the enterprise functionality. A related area would be to provide consolidated business intelligence in a context-dependent manner as a next step to data consolidation.

Service brokers based upon semantic technologies are going to be needed to enable creation of a common understanding of the varying semantics, concepts, platforms, policies, nonfunctional requirements, and APIs.

Standardization among the cloud service providers and service brokers is not yet present, but in the future, the interoperability requirements may lead to standardization in some areas of service provision.

5.10 Conclusion

A cloud service broker has no cloud resources of its own but matches cloud consumers and one or more cloud service providers based on the SLA and other metrics required by the cloud consumers. The consumer does not know that the broker does not control the resources.

A cloud service broker provides multiple benefits in the form of simplification and aggregation of services, security and privacy enhancement, integration of services, and fulfillment of business processes in an optimal manner. The above is done in a manner that fulfills the user-defined metrics and constraints. There are some challenges associated with the usage of cloud service brokers as well. The whole domain is still in a nascent state of development. There may be very little in terms of references or proven delivery of services by some new and/or very specialized service brokers.

The service brokers themselves have to deal with a number of challenges as they have to now perform the complex integration with multiple service providers on behalf of the consumers – so the complexity moves from the consumers to the service brokers. They need to have specialized resources, tools, and infrastructure to perform the complex interoperations. They also take the risks in terms of SLA conditions provision and possible penalties for noncompliance. The creation of such services is associated with large costs.

Despite the above challenges, the area is growing rapidly as all the parties – the cloud service consumers, service providers, and service brokers – see value in the model. The cloud service consumers see the value of simplification and provision of best terms and conditions in terms of SLAs, costs, and features. They also have a single point of contact. The cloud service providers have better coverage in terms of an additional channel. They can focus on the core business and client handling is managed by the service brokers. The service brokers have a niche area where they act and provide relevant services to the consumers by performing value addition, aggregation, etc., to the services provided by the pure-play cloud service providers.

References

1. Gartner: Cloud Computing innovation key initiative overview. http://my.gartner.com/portal/server.pt?open=512&objID=260&mode=2&PageID=3460702&id=1745015&ref=seo (2012). Accessed 8 July 2012
2. Gartner: Gartner worldwide IT spending forecast summary, Q2 2012 update. http://www.gartner.com/technology/research/it-spending-forecast/ (2012). Accessed 8 July 2012

3. Open Virtualisation Format: http://dmtf.org/sites/default/files/standards/documents/DSP0243_2.0.0c.pdf (2012). Accessed 9 July 2012
4. DMTF: http://www.dmtf.org/ (2012). Accessed 9 July 2012
5. Unified Cloud Interface: http://code.google.com/p/unifiedcloud/ (2012). Accessed 9 July 2012
6. Cloud Computing Interoperability Forum: http://www.cloudforum.org/ (2012). Accessed 9 July 2012
7. Open Cloud Computing Interface: http://www.ogf.org/documents/GFD.183.pdf (2012). Accessed 9 July 2012
8. Open Grid Forum: http://www.ogf.org/ (2012). Accessed 9 July 2012
9. Cloud Security Alliance Security Guidance: https://cloudsecurityalliance.org/guidance/csaguide.pdf (2012). Accessed 9 July 2012
10. Cloud Security Alliance: https://cloudsecurityalliance.org/ (2012). Accessed 9 July 2012
11. Cloud Storage Initiative: http://www.snia.org/forums/csi (2012). Accessed 9 July 2012
12. SNIA: http://www.snia.org/ (2012). Accessed 9 July 2012
13. TOSCA-v1.0 Topology and Orchestration Specification for Cloud Applications Version 1.0: OASIS Committee Specification Draft 02, 5 April 2012. http://docs.oasis-open.org/tosca/TOSCA/v1.0/csd02/TOSCA-v1.0-csd02.html (2012). Accessed 10 July 2012
14. OASIS: https://www.oasis-open.org/news/pr/tosca-tc (2012). Accessed 10 July 2012
15. NIST SP 500-292: NIST Cloud Computing reference architecture. http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/ReferenceArchitectureTaxonomy/NIST_SP_500-292_-_090611.pdf (2012). Accessed 9 July 2012
16. Gartner: Gartner says Cloud consumers need brokerages to unlock the potential of Cloud services. http://www.gartner.com/it/page.jsp?id=1064712 (2012). Accessed 9 July 2012
17. Forrester: Cloud broker – a new business model paradigm August 10, 2011 | Updated: September 22, 2011 by Stefan Ried, Ph.D. http://www.forrester.com/Cloud+Broker+A+New+Business+Model+Paradigm/fulltext/-/E-RES57809?docid=57809 (2012). Accessed 9 July 2012
18. Gartner: Cloud services brokerage is dominated by three primary roles. http://my.gartner.com/portal/server.pt?open=512&objID=260&mode=2&PageID=3460702&docCode=226509&ref=docDisplay (2012). Accessed 9 July 2012
19. Apache – Deltacloud: http://deltacloud.apache.org/ (2012). Accessed 16 July, 2012
20. Aepona:http://www.aepona.com/news-events/news/aepona-announces-new-services-aggregation-merchandizing-platform/ (2012). Accessed 16 July 2012
21. Sarbanes-Oxley Act (SOX): http://searchcio.techtarget.com/definition/Sarbanes-Oxley-Act (2012). Accessed 16 July 2012
22. Health Information Portability and Accountability Act of 1996 (HIPAA): http://www.hhs.gov/ocr/privacy/hipaa/understanding/summary/privacysummary.pdf (2012). Accessed 16 July 2012
23. Payment Card Industry Data Security Standard (PCI DSS): https://www.pcisecuritystandards.org/security_standards/getting_started.php (2012). Accessed 16 July 2012
24. European Data Privacy Framework: http://ec.europa.eu/justice/data-protection/index_en.htm (2012). Accessed 16 July, 2012
25. Novell: http://www.novell.com/docrep/2009/10/Provide_Connection_Access_and_Compliance_with_Novell_Cloud_Security_en.pdf (2012). Accessed 16 July 2012
26. CloudSwitch: http://www.cloudswitch.com/page/enterprise-cloud-computing-product-overview (2012). Accessed 16 July 2012
27. RESERVOIR – Resources and Services Virtualization Without Barriers: http://62.149.240.97/uploads/Training/RESERVOIR_Framework_Guide_Website.pdf (2012). Accessed 16 July 2012
28. Buyya, R., Ranjan, R., Calheiros, R.N.: InterCloud: utility-oriented federation of cloud computing environments for scaling of application services. In: Algorithms and Architectures for Parallel Processing. Lecture notes in computer science, vol. 6081, pp. 13–31. Springer, Berlin/Heidelberg (2010)
29. Costa, R., Brasileiro, F., de Souza Filho, G.L., Sousa, D.M.: Just in time Clouds: enabling highly-elastic public Clouds over low scale amortized resources. Tech. Rep. TR-3, Federal University of Campina Grande/Federal University of Paraíba, Brazil (2010)

30. Marosi, A.C., Kecskemeti, G., Kertesz, A., Kacsuk, P.: FCM: an architecture for integrating IaaS cloud systems. In: Proceedings of the Second International Conference on Cloud Computing, GRIDs, and Virtualization (Cloud Computing 2011), pp. 7–12. IARIA, Rome (2011)
31. Chandra, A., Weissman, J.: Nebulas: using distributed voluntary resources to build Clouds. In: HotCloud09 USENIX Workshop on Hot Topics in Cloud Computing, San Diego, June 2009
32. Business Service Management HP: http://h71028.www7.hp.com/enterprise/w1/en/solutions/service-management-business.html?jumpid=reg_r1002_usen_c-001_title_r0001 (2012). Accessed 16 July 2012
33. Garg, S.K., Yeo, C.S., Buyya, R.: Proceeding Euro-Par'11 Proceedings of the 17th international conference on Parallel processing – Volume Part I, pp. 491–502. Springer, Berlin/Heidelberg (2011). ISBN 978-3-642-23399-9
34. Guarino, N., Oberle, D., Staab, S.: What is an ontology?. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, 2nd edn., pp. 1–17. Springer, Heidelberg (2009)
35. Moscato, F., Aversa, R., Di Martino, B., Fortis, T., Munteanu, V.: An analysis of mOSAIC ontology for Cloud resources annotation. In Computer Science and Information Systems (FedCSIS), 2011, pp. 973–980 (2011)
36. OWL Web Ontology Language Reference: http://www.w3.org/TR/owl-ref/ (2012). Accessed 25 July 2012
37. Martin, D., et al.: Bringing semantics to web services: the OWL-S approach. In: Cardoso, J., Sheth, A. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 26–42. Springer, Berlin/Heidelberg (2004)
38. Preprint: Joshi, K.P., Finin, T., Yesha, Y., Joshi, A., Golpayegani, N., Adam, N.R.: A policy-based approach to smart Cloud services. In: Annual Service Research and Innovation Institute Global Conference, San Jose, July 2012
39. Joshi, K.P.: DC proposal: automation of service lifecycle on the Cloud by using semantic technologies. In: Aroyo, L., et al. (eds.) ISWC 2011, Part II. LNCS, vol. 7032, pp. 285–292. Springer, Berlin/Heidelberg (2011)
40. Knowledge representation for the semantic web part 1: OWL2. http://semantic-web-book.org/w/images/b/b0/KI09-OWL-Rules-1.pdf (2012). Accessed 25 July, 2012
41. SPARQL Query Language for RDF: http://www.w3.org/TR/rdf-sparql-query/ (2012). Accessed 25 July 2012
42. Resource Description Framework (RDF): http://www.w3.org/RDF/ (2012). Accessed 25 July 2012
43. Appirio SalesWorks: www.appirio.com/company/pdf/AppirioSalesWorks_archive.pdf (2012). Accessed 25 July 2012
44. Chatter: http://www.salesforce.com/chatter/overview/ (2012). Accessed 25 July 2012
45. Google Apps: http://www.google.com/intl/en/enterprise/apps/business/products.html (2012). Accessed 25 July, 2012

Chapter 6
Resource and Scheduling Management in Cloud Computing Application Paradigm

Katerina Papanikolaou and Constandinos Mavromoustakis

Abstract In the 'Era of Peta' processing level, which supports several concepts and practices in order to build highly scalable applications, there is a great need to have the offered services provided to the end-users, with the minimum possible reduction in quality in terms of time and availability. This chapter addresses some of these traditional concepts combined in a 'multi-sharing' cloud application environment and discusses how these concepts evolve in the context of cloud computing. The chapter also presents some unprecedented issues, such as resource allocation and management and the potential inefficiencies in the currently utilised APIs in the cloud computing application paradigm that have emerged through time. As the size of the cloud environment increases, efficient resource allocation and management become even more challenging, whereas the problem of optimisation in a large distributed environment under QoS constraints needs a potentially utility-oriented solution. Current methods, although efficient, need to be re-engineered encompassing methods suitable for distributed system optimisation for managing the future clouds. It also exposes the state-of-the-art techniques to provide efficient replicated storage according to the data semantic context and resource batching for satisfying users' requests originating from anywhere anytime using both static and moving environments. The upcoming and current computing paradigm is envisioned to be offering synchronous and asynchronous services in the context of the cloud computing services paradigm. Cloud computing has evolved through the creation of a new service paradigm by utilising data centres and assembling services of networked virtual machines. Therefore, the availability of the requested resources by users

K. Papanikolaou
School of Sciences, European University Cyprus, 6 Diogenous Str., 1516 Nicosia, Cyprus
e-mail: K.Papanikolaou@euc.ac.cy

C. Mavromoustakis (✉)
Department of Computer Science, University of Nicosia,
46 Makedonitissas Avenue, 1700 Nicosia, Cyprus
e-mail: mavromoustakis.c@unic.ac.cy

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_6,

poses a crucial parameter for the adequacy of the service provided. One of the major deployments of the cloud application paradigm is the virtual data centres (VDC). These are utilised by service providers which enable a virtual infrastructure in a distributed manner in various remotely hosted locations worldwide to provide accessibility and backup services in order to ensure reliability. This chapter also presents resource considerations and paradigms in the cloud environment describing the fundamental operations that are required for faster response time by distributing workload requests to multiple VCDs using certain resource manipulation techniques such as the Virtual Engine Migration (VEM) for resource availability and the associated resource management in VDCs using user-based VEM resource migration or virtual-to-virtual (V2V) resource migration.

Keywords Scheduling management • API • Virtual data centre • Resource allocation • Cloud management • Virtual engine migration • Virtual-to-virtual resource migration • High resource availability

6.1 Introduction

In cloud computing and the relevant architectures proposed in recent research work, there are many different levels of abstraction with reference to the deployment architectural models. The concept of utility computing [1] which is offering synchronous and asynchronous services in the context of cloud computing services paradigm, likewise the services offered through the cloud computing paradigm, emerged mainly from the appearance of public computing utilities. These services vary in the physical location and the distribution of requested data and resources. In this sense, regardless of its service class, a cloud can be classified as public, private, community-based or so-called social, or hybrid or a combination of all the above [2] based on the model of deployment as shown in Fig. 6.1. Cloud computing services are divided into three classes, according to the abstraction level of the capability provided and the service model of providers, namely: (1) Infrastructure-as-a-Service (IaaS), (2) Platform-as-a-Service (PaaS) and (3) Software-as-a-Service (SaaS) [2]. Figure 6.1 depicts the conceptual resource organisation of the cloud stack from physical infrastructure to applications.

A key challenge faced by providers when building a cloud infrastructure is managing physical and virtual resources according to user-resources' demands, with respect to the access time to the associated servers, the capacity needed for storage and the heterogeneity aspects traversing different networks in a holistic fashion [3]. The organisation or, namely, the orchestration of resources must be performed in a way that rapidly and dynamically provides resources to applications [4]. There are many software tools that are aiming to reduce the related to user's service aggravation issues, in contrast to the resources' manipulation in an end-to-end manner. The software toolkit responsible for this orchestration is called a Virtual Infrastructure Manager (VIM) [4] or Virtual Infrastructure Resource Manager (VIRM). This type

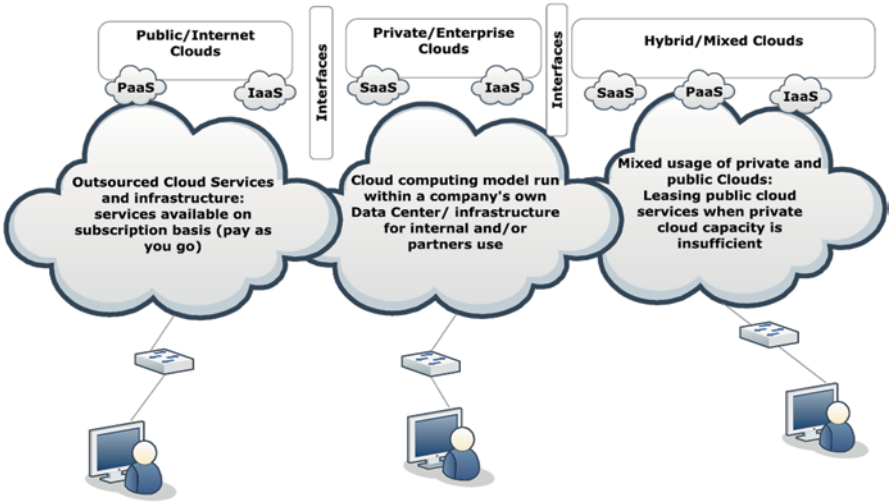


Fig. 6.1 Types of clouds based on deployment models

of software tool resembles a traditional operating system—but instead of dealing with a single computer, it aggregates resources from multiple computers, superpositioning them in a content-oriented uniform view to the user. Recent researches make use of the term 'cloud operating system' which is basically the technical explanation of the resource manipulation concept in the cloud paradigm [3]. Other terms include 'infrastructure sharing software [4]' and 'virtual infrastructure engine' [5] as mechanisms that enable efficient resource manipulation. On the basis of resource manipulation and sharing in a cloud environment, there are many challenges such as the update formula of the requested resources [6] and the informational notation about these updates to the end-users. The following sections analyse these mechanisms that enable resource manipulation in an efficient and network-context-oriented way aiming to present the most significant benefit that the cloud computing offers: the elasticity of resources, services and applications, which is the ability to automatically scale-out based on demand and users' quality of service requests.

6.2 'Under the Hood' Resource Challenges

One can easily argue whether a company or an organisational body would be offering better services if these services can be easily migrated to the cloud. It is undoubtedly true that the cloud services present a simplistic view of IT in the case of IaaS or a simplistic view of programming notations in the case of PaaS or even a simplistic view of resources manipulation and utilisation in the case of SaaS. However, the underlying communicating mechanisms comprising of heterogeneous systems are

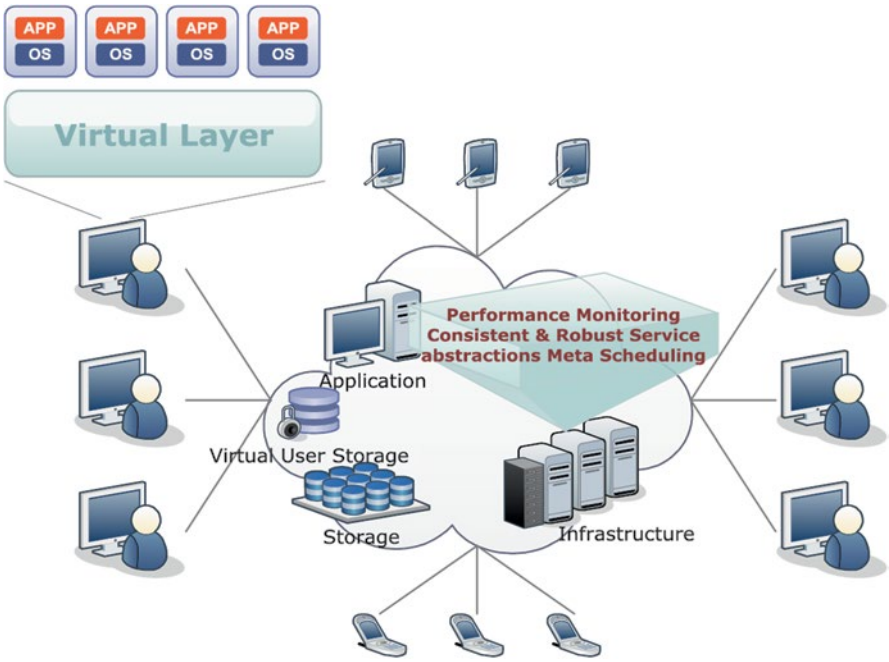


Fig. 6.2 Reliability via performance monitoring, consistent and robust service abstractions and meta-scheduling

formed in a highly complex way. The underlying supportive mechanisms and systems should not be failure-prone supporting scalability and heterogeneity, whereas overcoming resource or virtual resources dismissals. In this way the mechanism should support 'zero' network latency or perhaps support 'infinite' bandwidth.

As is typical in distributed systems, cloud-based systems have many parameterised operational issues which seem to play a sophisticated role in the service offered to the end-user (to the end-recipient). Therefore, by enabling a mechanism that will be monitoring the performance (see Fig. 6.2) of the shared/requested resources in terms of reliability and resource scheduling according to the 'system-based' information, requested by the virtual infrastructure for the end-users, is in essence the pathway to enable the migration of all services to the cloud. On one hand, the challenges in implementing cloud computing services for both statically located users and mobile users are numerous. On the other hand, there are many different aspects in the process of the implementation, as follows:

- Interoperability between the different services and the supporting platforms: Enabling user-customisable cloud profile: As clouds develop into application platforms (see, e.g. Google Chrome OS [7]), context such as user device properties or location becomes more and more relevant. Therefore, there is a need to designate the execution capabilities (even if remote) by being aware of the

connectivity requirements (bandwidth and volume of requested resources) and the associated restrictions, as well as the location [8].
- Batching different user demands in order to 'request many, transfer once': Users may request resources which may be common for a group of users. The resource manipulation mechanism may batch the requests, and using a delay sensitive mechanism (which will minimise the delay of the response), the users will have the requested resources.
- Taking the advantages of using the Communication-as-a-Service (CaaS) [9] and the communication extracted information through the CaaS: Furthermore, the Monitoring-as-a-Service (MaaS) [10] will also enable the awareness of information related to the resource manipulation based on the requests and responses of the systems to the users.
- Aspects that exist in utility-computing environments, where users assign a 'utility' value to their jobs and where utility is a fixed or time-varying valuation that captures various QoS constraints (like response deadline, service importance, end-users' satisfaction). Usually in resource-constraint systems (wireless networks and ad-hoc connected networks), the context-aware utility computing becomes very scarce as resources are not available at all times and at the desired available manner (bandwidth and acquire response for every service the users demand).

Cloud services are focused on the support of large software package usage leveraging cloud benefits. For a majority of these packages, there is a noncontrollable ignorance of the underlying cloud support, where in fact most, if not all, do not consider the different mechanisms as they act in the service provision as a 'black box'. For example, in *Gmail*, users hardly consider or are even aware of either the storage space taken up or whether an email needs to be deleted using a natural or a virtual storage location. It is undoubtedly true that these reflect the cloud underneath, where storage (most of the applications are not aware on which system these applications or resources are set) is not easily definable and scalable. Therefore, there is a need to define different interfaces between different platforms where the interfaces will serve as middleware connectors (cloud middleware, also referred to as cloud OS) between the API of each application and services provided.

6.2.1 Mind the Gap: API 'Efficiencies' and 'Insufficiencies'

Cloud development environments are built on top of infrastructure services to offer application development and deployment capabilities for the offered services. For these services, various programming models, application-specific libraries and APIs providing different application programming interfaces (APIs) enable the creation of a wide range of business, Web and scientific applications. In this respect, there were many SaaS providers who have responded to the integration challenge by designing and developing specific APIs. There has been a great challenge to application developers regarding the management of data requested by users via an API which requires a significant amount of coding, as well as maintenance due to frequent API modifications and

updates. In addition, there is no certain mechanism which is responsible for defining the updates for a specific API, and for this reason, companies without a software methodology for synchronising data between multiple services are at a serious disadvantage in terms of maintaining accurate data and enabling automation for key business processes. Real-time API update maintainability and data-service access for functionality sharing is an essential ingredient for cloud environments.

Notwithstanding the above, there is a great potential to maintain API updates and maintenance through web services. Moreover, the lack of standardisation or consensus on the structure or format of SaaS APIs creates a significant inefficiency and gap in developing mechanisms that will provide an API structure acceptable by all platforms and services. This issue results in a waste of resources and time for developers who need to develop and maintain a unique method of communication regarding the API support and update of each SaaS application deployed within an organisation.

In contrast to the resource considerations and the APIs of each of the services provided by cloud environments, cloud computing suffers a number of limitations which can be seen on the basis of specific situations and scenarios. Some of these limitations are keeping the cloud paradigm as a 'solo' organisational instrument, where only the company knows the precise structure of the API. Indicative limitations are:

- The controllability of the response of the offered services under network-restrictive cases
- The transparency, visibility and flexibility of the services through heterogeneous clouds
- The security and privacy
- The high performance and availability
- The integration and composition
- Distinctive standards for the various scenarios

For the above issues, the implementation and provision of adequate solutions that fit to the services offered and the requested demands can be performed through the collaboration of interfaces. These collaborations will be commonly used by services' vendors to enable smooth transition from one application interface to another. Another proposed solution for this gap is the integration of different APIs into a 'solid' single, integrated API for supporting SaaS. Integration is not easier either to implement or to enable many application developers to adopt it. The constraining attributes of SaaS applications include the following [10]:

- Dynamic nature of the SaaS interfaces that constantly change
- Dynamic nature of the metadata native to a SaaS provider such as Salesforce.com [11] paradigm
- Managing assets that exist outside of the firewall
- Massive amounts of information that need to move between SaaS and on-premise systems on a daily basis and the need to maintain data quality and integrity through time

Therefore, the API integration should also encompass mechanisms that are controllable and manageable in contrast to the scalability of the utilised scenarios,

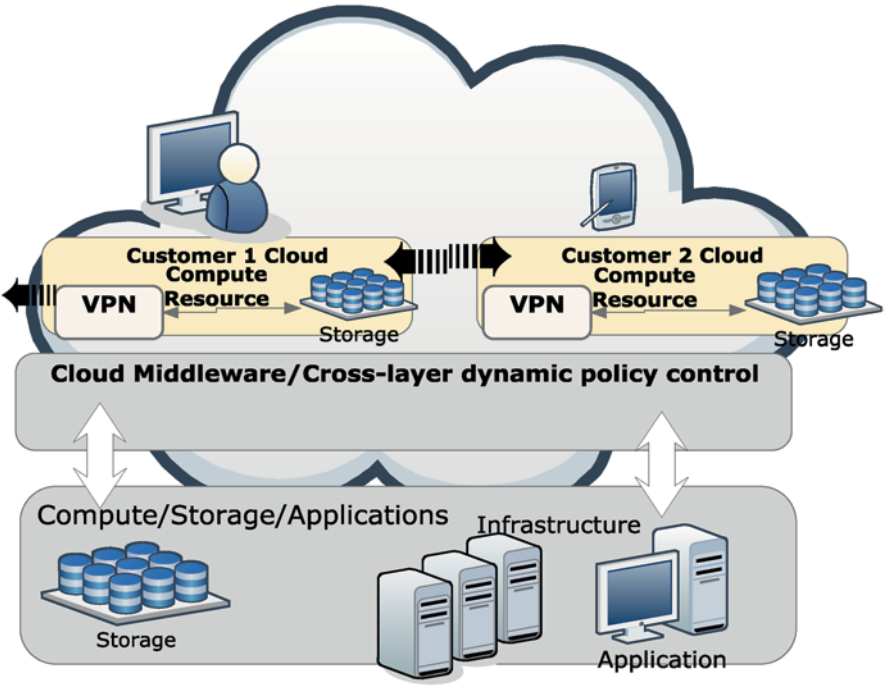


Fig. 6.3 High-level cloud middleware architecture for resource manipulation

whereas they should enable dynamic access to resources regardless of their physical location. This access will resolve many performance issues related to accessing the requested by users cloud resources (SaaS, PaaS and the infrastructures), which are not faced when dealing with local applications. Once applications are moved to the cloud, custom applications and supported APIs must be designed to support integration. Enterprises putting their applications in the cloud are dependent on the vendor to provide the integration interfaces and APIs. As a case scenario, the SalesForce.com [11] web services API does not support transactions against multiple records, resulting to the need for the development of an integration API code (between different platforms) to handle the process. However, another open issue for filling the gap of integrating APIs in SaaS is the platform-to-platform integration. There is a tentative agreement for SaaS providers, where a limited set of APIs will improve the incompatibility and manipulate the consistency among APIs to a certain extent [9]. As most of the third-party cloud providers do not submit their infrastructures for third-party audit, the visibility factor is yet another factor that is lost-out, in case of this transition. The main reason for providing an integration of the API is to allow (both static and mobile end-users) to orchestrate their virtual resources.

Figure 6.3 shows the high-level cloud middleware architecture for resource manipulation and the management mechanism, which constitutes of a key factor for the development of scalable cloud computing architectures. Network resources are

statically provisioned, but computing, storage and application resources in clouds must be capable of being provisioned dynamically and on demand, to both statically located and mobile users. Additionally, network resources must be capable of being provisioned separately from application resources [12]. In a cloud environment (where cloud instances must be created on demand and dynamically with minimum turnaround time), the administrative resource manipulation boundaries become a major issue which may result in decreasing the overall Quality of Service (QoS) and Quality of Experience (QoE) to the end-users. The resource management process and the users' workflow requests hosted in the cloud must be fully automated with minimum turnaround time, so that users can have resources within specified timeline and within minimised duration.

6.3 Resource Management in the Cloud Paradigm

Cloud resource management refers to the allocation of cloud resources, such as processor power, memory capacity and network bandwidth. The resource management system is the system responsible for allocating the cloud resources to users and applications. For any resource management system to become successful, it needs to flexibly utilise the available cloud resources whilst maintaining service isolation [13, 14]. The resource management system is expected to operate under the predefined QoS requirements as set by the customers. For this, resource management at cloud scale requires a rich set of resource management schemes that are capable to manage efficiently the provision of QoS requirements whilst maintaining total system efficiency. The greatest challenge for this optimisation problem is that of scalability.

A number of such management systems exist for both private and public clouds such as the (1) IaaS solutions like OpenNebula [15], OpenStack [16] and Eucalyptus [17] or (2) the PaaS solutions such as AppleScale [18], WebSphere XD [19] and Cloud Foundry [20] for private clouds and the public cloud solutions such as those provided by Amazon and Microsoft. Some public cloud systems (e.g. Amazon EC2) are using a one-to-one mapping between a virtual and a physical CPU and memory resources. This results to poor consolidation ratios and poor exploitation of the benefits of statistical multiplexing that is used in private clouds. In private clouds, resource management solutions like VMware Distributed Resource Scheduler (DRS) [1] and Microsoft PRO [21] have produced better performance isolation, higher utilisation of underlying hardware resources via overcommitment and overall lower cost of ownership. The negative aspect of this is that the rich set of controls and methods provided by these solutions is not easily scalable for covering large cloud environments.

For a more efficient and scalable resource management, a number of issues need to be addressed:

- Efficient and dynamic application placement. Efficient load balancing among cloud resources whilst maintaining total system efficiency

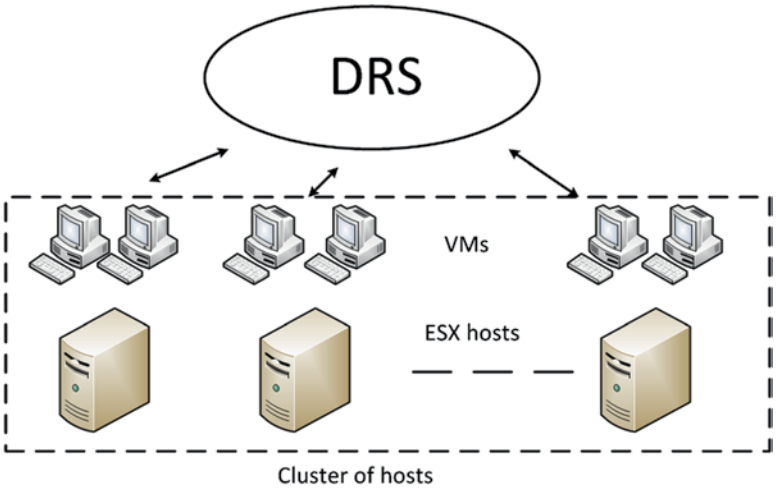


Fig. 6.4 DRS cluster architecture

- Network bandwidth allocation, as clients' content for cloud resources, they also content for network resources.
- Efficient task scheduling, resulting in a parallel and more efficient use of cloud resources.

6.3.1 Existing Schemes and Weaknesses of Resource Management

Currently a number of different methods and techniques are employed in order to provide resource management controls in the cloud environment. We shall examine VMware's Distributed Resource Scheduler (DRS) [22] in more detail as a representative system. All systems perform two basic functions: (1) application placement and (2) load balancing. The DRS scheduler has a rich set of controls used for efficient multi-resource management whilst providing differentiated QoS to groups of VMs, but this is obtainable for a small scale compared to a typical cloud. DRS is currently supporting 32 hosts and 3,000 VMs approximately, in a management domain called a cluster (see Fig. 6.4).

Having a rich set of resource management controls, in a large-scale cloud environment, lightens the noisy-neighbour problem for tenants if, like in the case of DRS, the underlying management infrastructure natively supports automated enforcement of guarantees. Additionally, such controls provide support to the cloud service provider from overcommitting hardware resources safely, allowing better efficiency from statistically multiplexing resources without sacrificing the expected guarantees.

6.3.1.1 Basic Resource Controls in DRS

VMware enterprise software hypervisors (ESX) and DRS provide resource controls allowing administrators and users to express allocations in terms of their absolute VM service rate requirements or relative VM importance. The same controls are provided for CPU and memory allocations for both host and cluster levels. Similar controls are under development for input and output resources that in conjunction with VMware's Distributed Power Management product power on/off hosts whilst respecting these controls.

- *Reservations*: A reservation specifies the minimum guaranteed resources required, i.e. the lower bound that applies even when the system is over committed. Reservations are specified in absolute units, such as megahertz (MHz) for CPU and megabytes (MB) for memory. Admission control during VM power on ensures that the sum of reservations for a resource does not exceed the total capacity.
- *Limits*: The limit specifies an upper bound on consumption required, even when a system is undercommitted. A VM can be prevented from consuming more than its pre-specified upper bound, even if resources remain idle. Like reservations, limits are specified in concrete absolute units, such as MHz and MB.
- *Shares*: Shares are used for specifying relative importance and are in contrast specified as abstract numeric values. A VM is allowed to consume resources proportional to its allocated share value; it is guaranteed a minimum resource fraction equal to its fraction of the total shares in the system. Shares are used to represent relative resource rights that depend on the total number of shares contending for the resource.

Reservations and limits have an important role in managing cloud resources. Without the provision of these guarantees, users would suffer from performance unpredictability, unless the cloud provider took the non-work-conserving approach of simply statically partitioning the physical hardware, such a measure would lead to the inefficiency of over-provisioning. This is the main reason for providing such controls for enterprise workloads running on top of VMware ESX and VMware DRS.

6.3.1.2 Resource Pools

Additional flexibility is provided to administrators and users with extra resource management policies for groups of VMs. This is facilitated by introducing the concept of a logical resource pool. The logical resource pool is a container that can be used to specify an aggregate resource allocation for a set of VMs. Admission control is now performed at the resource pool level. The pool operates under the constraint of the sum of the reservations for a pool's children should not exceed the pool's own reservation. Separate, per-pool allocations can be used to provide both isolation between pools and sharing within pools. For example, if some VMs within a pool are idle, their unused allocation will be reallocated preferentially to other

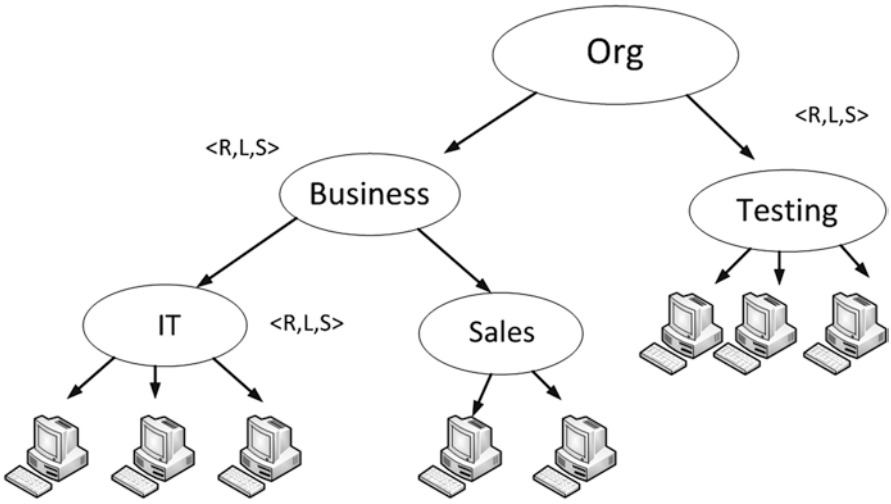


Fig. 6.5 Resource pool tree: R, L and S denote reservation, limit and share values, respectively

VMs within the same pool. Such resource pools may be organised using a flexible hierarchical tree-shaped scheme as shown in Fig. 6.5: each pool having an enclosing parent pool and children that may be VMs or sub-pools. Resource pools can be used in dividing large capacity to logically grouped users. Resource pool hierarchies can be used by organisational administrators to mirror human organisational structures and to support delegated administration.

The resource pool construct can be used in the cloud setting as organisational administrators typically buy capacity in bulk from providers and run several VMs. It is not adopted by several large-scale cloud providers despite its usefulness for thousands of enterprise customers using VMware DRS and VMware ESX, each managing the resource needs of thousands of VMs.

6.3.1.3 DRS Load Balancing

The DRS scheduler performs load balancing by using three key resource-related operations: (1) computes the amount of resources that each VM should get based on the reservation, limit and shares settings for VMs as well as resource pool nodes; (2) performs the initial placement of VMs on to hosts, so that a user doesn't have to make a manual placement decision; and (3) recommends and performs live VM migrations to do load balancing across hosts in a dynamic environment where the VMs' resource demands may change over a period of time.

DRS cluster management of distributed hosts is done by providing the illusion that the entire cluster is operating as a single host with the aggregate capacity of all individual hosts. This is implemented by breaking up the user-specified resource

pool hierarchy into per-host resource pool hierarchies with appropriate host-level resource pool settings. Once the VMs are placed on a host, the local schedulers on each ESX host allocate resources to VMs fairly based on host-level resource pool and VM resource settings. DRS is invoked every 5 min by default, but can also be invoked on demand.

DRS load balancing uses its own load-balancing metric [22]. In particular, it does not use host utilisation. In DRS, load reflects VM importance, as captured by the concept of dynamic entitlement. Dynamic entitlement is computed based on the resource controls and actual demand for CPU and memory resources for each VM. The entitlement is higher than the reservation and lower than the limit; its actual value depends on the cluster capacity and total demand. Dynamic entitlement is equivalent to demand when the demands of all the VMs in the cluster can be met, if this is not the case it is a scaled-down demand value with the scaling dependent on cluster capacity, the demands of other VMs, the VM's place in the resource pool hierarchy, and its shares, reservation and limit. Dynamic entitlement is computed using a pass over the resource pool hierarchy tree to allocate to all VMs and resource pools their CPU and memory reservations and to constrain their demand by their limits, followed by another pass over the tree to allocate spare resources to address limit-constrained demand above reservation in accordance with the associated share values. DRS currently uses normalised entitlement as its core per-host load metric. For a host h, normalised entitlement N_h is defined as the sum of the per-VM entitlements E_i for all VMs running on h, divided by the host capacity C_h available to VMs: $N_h = \sum E_i / C_h$. If $N_h \leq 1$, then all VMs on host h would receive their entitlements, assuming that the host-level scheduler is operating properly. If $N_h > 1$, then host h is deemed to have insufficient resources to meet the entitlements of all its VMs, and as a result, some VMs would be treated unfairly. After calculating N_h for each host, the centralised load balancer computes the cluster-wide imbalance, I_c, which is defined as the standard deviation over all N_h.

In a simplified description [23], DRS load-balancing algorithm uses a greedy hill-climbing optimisation technique. This approach, as opposed to an exhaustive approach that would try to find the best target balance, is driven by the practical considerations that the VMotion operations needed to improve load balancing have a cost and that VM demand is changing over time, so highly optimising for a particular dynamic situation is not worthwhile. DRS minimises I_c by evaluating all possible migrations, many filtered quickly in practice, and selecting the move reducing I_c the most. The selected move is applied to the algorithm's current internal cluster snapshot so that it then reflects the state that would result when the migration is completed. This move-selection step is repeated until no additional beneficial moves remain or there are enough moves for this pass or the cluster imbalance is at or below the threshold T specified by the DRS administrator. The actual implementation of load-balancing algorithm considers many other factors, including the risk-adjusted benefit of each move given the range and stability of VMs' dynamic demand over the last hour, as well as the cost of the migration and any potential impact of the migration on the workload running in the VM. DRS is currently able to handle a small cloud of 32 hosts and 3,000 VMs; for extending its application, a

number of issues need to be resolved: (1) inventory management, efficient collection of host and VM data as the cloud grows, (2) efficient management of a cluster of heterogeneous host, (3) efficient adaptation to a fast changing environment and (4) survivability to resource management system failure.

6.3.1.4 The Future of Resource Management

The expansion of resource management to larger clouds comprised of thousands of machine and sites beyond those offered in private level is a complex optimisation task. Most existing systems do not, in a combined and integrated form, (1) dynamically adapt existing placements, (2) dynamically scale resources and (3) scale beyond a few thousands of physical machines. To overcome these constraints, research in the area is moving towards employing optimisation techniques for large distributed environments, such as stochastic processes, gossip protocols or even multiplayer gaming.

The use of a gossip protocol for dynamic resource management is proposed in [24]. A middleware architecture is presented with key element the gossip protocol that is used to ensure fair resource allocation among sites, whilst being able to dynamically adapt the allocation to load changes complying with the scalability requirement. In the proposed system, architecture is comprised of three managing entities: the *machine*, *resource* and *overlay* managers. Each machine runs a *machine manager* component that computes the resource allocation policy, which includes deciding the module instances to run. The resource allocation policy is computed by a protocol that runs in the *resource manager* component. This component takes as input the estimated demand for each module that the machine runs. The computed allocation policy is sent to the module scheduler for implementation/execution, as well as the site managers for making decisions on request forwarding. The *overlay manager* implements a distributed algorithm that maintains an overlay graph of the machines in the cloud and provides each resource manager with a list of machines to interact with. The protocol is simulated with varying input and a large number of machines 160,000 and sites 368,000 and maintains its efficiency showing a high degree of scalability.

The use of load prediction algorithms for cloud platforms is proposed in [24], using a two-step approach of load trend tracking followed by load prediction, using *cubic spline interpolation* and hotspot detection algorithm for sudden spikes. Their results indicate that the autonomic management framework is able to respond to both synthetic and real-world load trends accurately.

The use of a hybrid P2P cloud computing architecture for massively multiplayer online games is proposed in [25]. The proposed architecture uses a two-level load management system, multi-threshold load management for each game server and load management among game servers. The resulting architecture can support more whilst reducing response time.

Finally, an area that should not be overlooked in resource management is the area of network resources such as bandwidth allocation. With the use of multiple streams

for multiple applications, some fairer TCP stream handling is needed. The work on [26] focuses on fair bandwidth allocation among the cloud users. The proposed mechanism prevents the use of additional flows or non-compliant protocols from gaining an uneven share of the network resources of the shared environment. The use of the 'fair-share' dropping mechanism builds on this by ensuring an equal division of bandwidth among the TCP flows of the local entity. The results show that the small overhead incurred justifies the improvement in efficiency. The trends presented here are a clear indication for the direction of things to come in terms of resource allocation and management in the cloud environment.

6.3.2 Resource Management in Virtual Data Centres (VDCs)

Virtualisation technology appeared several years ago; it comes in many types, all focusing on control and usage schemes that emphasise efficiency. This efficiency is seen as a single terminal being able to run multiple machines or a single task running over multiple computers via idle computing power. Adoption within data centres and adoption by service providers is increasing rapidly and encompasses different proprietary virtualisation technologies. Again, the lack of standardisation poses a barrier to an open standards cloud that is interoperable with other clouds, and a broad array of computing and information resources is fundamentally implementable. As the availability of requested resources by users poses a crucial parameter for the adequacy of the service provided, one of the major deployments of the cloud application paradigm is the virtual data centres (VDC), utilised by service providers [21] by enabling a virtual infrastructure (Fig. 6.6) in a distributed manner in various remotely hosted locations worldwide to provide accessibility [27] and backup services and ensure reliability in case of a potential single site failure. In the case of resource saturation or resource dismissal, where a certain location-based resource cannot be accessed, the VDC claims the resource in order to enable availability to potential requests/users. Additionally, these services with globally assigned operations require faster response time by distributing workload requests to multiple VCDs using certain scheduling and load-balancing methodologies. Therefore, as an optimal approach to resource availability, a k-rank model [45] can be applied in order to rank the requests and resources and create outsourcing 'connectivity' to potential request.

Currently, most computing servers in VDCs comprise of clusters of computers as they offer high performance, high availability of resources. Notwithstanding the high-throughput response is of vital essence, it should be offered at a lower cost compared to conventional high-performance computing (HPC) systems [28]. Resource manipulation in these clusters needs an adaptively dynamic decision making, in order to provide the service requirements' response to users. By considering the dynamic amendments of the original service requests that should take place in order to avoid the resources' deficiencies expressed by the traditional resource management systems such as Condor [29], LoadLeveler [30], Portable Batch System and

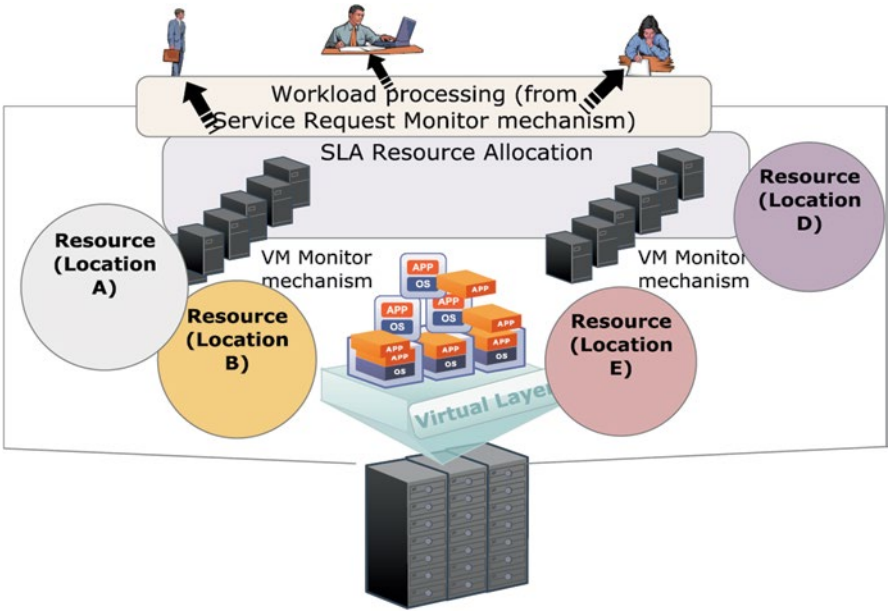


Fig. 6.6 VM workload processing (from service request monitor mechanism from different locations) for resource manipulation

Load Sharing Facility [28], someone needs to express the host-system parameters in the basis of resource-requests and the overall volume of needed resources. However, these types of systems adopt a system-centric resource allocation approaches that focus on optimising overall cluster performance in a selfish way.

To overcome the selfishness obstacle of VDC clusters, the utilised random segmentation policies of the requested resources need to be replaced by a new strategy which takes: (a) the semantic contents of the datasets, (b) the requirements of users/applications into account, i.e. data shall be distributed according to the interest in the data/information per time, and (c) the semantic resource applicability of the requested data for location-aware resource accessibility. For this reason, users, devices and applications need to be modelled by capturing relevant context parameters (e.g. the actual position and network properties) as well as analysing application states with respect to upcoming data retrieval and/or processing needs [31].

It is undoubtedly true that in a cloud environment, performance penalties may appear at two different levels: at the virtual engine (VE) level for which the performance is significantly reduced due to faulty or dismissed (non-updated) approach of the virtualisation mechanism and the overall cloud environment (CE) which causes losses to be introduced at a higher level by the cloud environment, and they are mainly due to overheads for the communication resources and due to the sharing or resource unavailability. The approach for facing this inefficiency is shown in Fig. 6.7, where the concept of virtual cluster is introduced which can execute

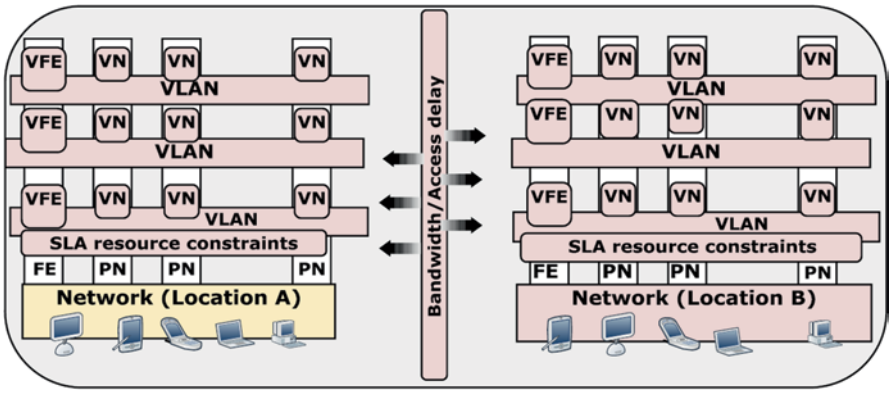


Fig. 6.7 Bandwidth and access restrictions based on the SLA by service operator when accessing remotely hosted resources stored in virtualised server farms using virtual front end

multiple jobs in parallel, by assigning to every job a subset of the total number of CPUs. To exploit all the available computing resources and be able to fulfil requests, the application should use at least a number of processes equal to the number of available CPUs (or, in case of concurrent jobs, equal to the number of CPU exclusively reserved for the job [28]).

A virtual cluster is composed of a virtual front end and a number of virtual nodes (Fig. 6.7). Virtual front ends are obtained by virtualisation of a physical front-end machine, and virtual nodes are obtained by virtualisation of physical processing nodes. The advantages of cluster virtualisation are the simplicity that each application can set up a proper execution environment, which does not interfere with all other applications and virtual clusters running in the VLAN. Moreover, the volume of traffic in every virtual cluster is encapsulated in a separate VLAN of a different location. In this way, all VLANs will share the network resources in an efficient way by minimising the request/dismissal rate.

Given the scale at which physical resources exist in such environments, the likelihood of having failures is relatively high rather than being an exception [20]. However, when problems such as power outage or network switch failures occur, a whole rack can fail, leading to multiple resources and servers being unavailable. Failures at such a scale are very likely to significantly reduce the efficiency, reliability offered to the end-recipient and consistency of the system. According to the literature, there are three possible ways of addressing the problem of multiple serves being unresponsive:

- Design dynamic resource allocation (DRA) policies that are failure-aware
- Outsource the resources according to the forecasting and predictability policy used by a service operator
- Execute a DRA policy when a failure is encountered

The issue of resource allocation is also addressed in research such as [20], where periodic balanced server provisioning was investigated. In contrast to experimental

research, analytical approaches to dynamic resource allocation have also been explored in work such as [32], [33] and [34], where application modelling and queuing theory were applied, respectively. Resource failures in the context of data centres are well addressed by existing literature [35, 36]. The coverage of resource failures has mostly concerned failures affecting a single hardware resource, excluding the possibility of large-scale failures. The authors of [36] give an insight into the failure rates of servers in a large *data centre* and attempt to classify them using a range of criteria. The issue of resource failures in cloud computing is addressed in [35], where the authors develop a policy to partition a resource between high-performance computing application and web services where these resources can be accessed and found on a broker-based format. The algorithm proposed in [37] depicts that under a balanced resource allocator, which uses a parameterised features of the metrics that are affecting the resource-sharing process (e.g. the capacity loss metric as shown in Eq. 6.1), can effectively assign resources to users/devices, at the same time minimising the potential capacity loss in the event of unpredictable server failure occurs. The capacity loss metric is considered as follows:

$$\text{Capacity Loss}_{r_i, a_j} = \frac{s_i^j}{S_{a_j}} \tag{6.1}$$

where r_i is the rack, a_j is an application, s_i^j is the number of servers in rack r_i hosting application a_j and S_{aj} is the number of servers hosting application across all racks. It is used to capture the impact the failure of rack r_i will have on the application a_j in terms of capacity. The above metric, as the performed estimations and experiments show, can effectively allocate requested resources to users, even if an unpredictable event occurs like a rack of massive server failures.

6.3.3 Energy Efficient Resource Management in VDCs

With the rapid growth of server farms and data centres, the demand of electric power increases. Despite the improvements in efficient energy use by hardware, overall energy consumption will continue to grow as current electronics technology power consumption cannot be reduced beyond a certain limit. The efficient and intelligent use of power from data centres becomes important.

A number of practices can be applied to achieve energy efficiency, such as improvement of applications' algorithms, energy efficient hardware, Dynamic Voltage and Frequency Scaling (DVFS) [38], terminal servers and thin clients, and virtualisation of computer resources [39]. Virtualisation technology allows one to create several virtual machines (VMs) on a physical server and, therefore, reduces amount of hardware in use and improves the utilisation of resources. Among the benefits of virtualisation, we have improved fault and performance isolation between applications sharing the same resource (a VM is viewed as a dedicated resource to the customer), the ability to relatively easily move VMs from one physical host to

another using live or off-line migration and support for heterogeneous hardware and software.

The statement of the problem of continuous consolidation as a sequential optimisation and is addressed in [40] by using Limited Lookahead Control (LLC). The proposed model requires simulation-based learning for the application-specific adjustments. Due to complexity of the model, the optimisation controller's execution time reaches 30 min even for a small number of nodes (e.g. 15) that is not suitable for large-scale real-world systems.

Another system is proposed in [41] addressing the problem of request scheduling for multi-tiered web applications in virtualised heterogeneous systems in order to minimise energy consumption, whilst meeting performance requirements. To handle the optimisation over multiple resources, the authors have proposed a heuristic for multidimensional bin-packing problem as an algorithm for workload consolidation. Resource allocation to applications according to their priorities in multi-application virtualised cluster is proposed in [42]. The approach requires machine learning to obtain utility functions for the applications and defined application priorities.

The problem of power-efficient allocation of VMs in virtualised heterogeneous computing environments is explored in [43]. The 'min', 'max' and 'shares' parameters of VMM are leveraged representing minimum, maximum and proportion of CPU allocated to VMs sharing the same resource. The approach suits only enterprise environments or private clouds as it does not support strict SLA and requires knowledge of applications priorities to define the 'shares' parameter.

Another system is proposed in [44], the underlying infrastructure is represented by a large-scale cloud data centre comprising n heterogeneous physical nodes. Each node has a CPU, which can be multicore, with performance defined in Millions Instructions Per Second (MIPS). Besides that, a node is characterised by the amount of RAM and network bandwidth. Users submit requests for provisioning of m heterogeneous VMs with resource requirements defined in MIPS, amount of RAM and network bandwidth. SLA violation occurs when a VM cannot get the requested amount of resource, which may happen due to VM consolidation.

The software system architecture is tiered comprising a dispatcher, global and local managers. The local managers reside on each physical node as part of a virtual machine monitor (VMM). They are responsible for observing current utilisation of the node's resources and its thermal state. The local managers choose VMs that have to be migrated to another node in the following cases:

- The utilisation of some resource is close to 100 % that creates a risk of SLA violation.
- The utilisation of resources is low; therefore, all the VMs should be reallocated to another node and the idle node should be turned off.
- A VM has intensive network communication with another VM allocated to a different physical host.
- The temperature exceeds some limit and VMs have to be migrated in order to reduce load on the cooling system and allow the node to cool down naturally.

The local managers send to the global managers the information about the utilisation of resources and VMs chosen to migrate. Besides that, they issue commands for VM resizing, application of DVFS and turning on/off idle nodes. Each global manager is attached to a set of nodes and processes data obtained from their local managers. The global managers continuously apply distributed version of a heuristic for semi-online multidimensional bin packing, where bins represent physical nodes and items are VMs that have to be allocated. The decentralisation removes a Single Point of Failure (SPF) and improves scalability. Each dimension of an item represents the utilisation of a particular resource. After obtaining allocation decision, the global managers issue commands for live migration of VMs.

The system operation consists of the following steps:

1. New requests for VM provisioning. Users submit requests for provisioning of VMs.
2. Dispatching requests for VM provisioning. The dispatcher distributes requests among global managers.
3. Intercommunication between global managers. The global managers exchange information about utilisation of resources and VMs that have to be allocated.
4. Data about utilisation of resources and VMs chosen to migrate. The local managers propagate information about resource utilisation and VMs chosen to migrate to the global managers.
5. Migration commands. The global managers issue VM migration commands in order to optimise current allocation.
6. Commands for VM resizing and adjusting of power states. The local managers monitor their host nodes and issue commands for VM resizing and changes in power states of nodes.
7. VM resizing, scheduling and migration actions. According to the received commands, VMM performs actual resizing and migration of VMs as well as resource scheduling.

The evaluation of the system shows that substantial savings of up to 83 % can be achieved. With rising energy costs and environmental consciousness playing an increasingly important role in decision making, adopting energy efficient policies in the cloud is becoming a key factor.

6.3.4 Virtual Engine Migration

In the cases where the resources can become a sharpening bottleneck, in order to allow access to the end-users, a Virtual Engine Migration (VEM) can be used as a potential mechanism that can isolate such problematic situations. A virtual machine consideration can be hosted at high level of the cloud abstraction to encapsulate access to a set of physical resources. Providing ‘live’ migration of these VMs in a

clustered server environment, leads to focus on the physical resources used in such environments. These are, namely, the memory, connectivity of the network and access to the disk. This section addresses the design decisions that have to be made for live VM migration through remote access.

OpenNebula [45], Enomaly [46] and Eucalyptus [47] are open-source software layers that provide a service-oriented interface on the top of existing virtual engines (mainly, VMWare and Xen). Virtual workspaces [48, 49] and related projects (Nimbus, Kupa, WISPY mentioned in [21]) build up the service-oriented interface for the virtual engines by exploiting a tightly coupled grid infrastructure.

Moving the contents of a VM's memory from one physical host to another can be approached in any number of ways. However, when a VM is running a live service, it is important that this transfer occurs in a manner that balances the requirements of minimising both downtime and total migration time. The latter is the period during which the service is unavailable due to situations that occur as there are no available executed instances of the VM; this time period is directly visible to clients of the VM as service interruption. The duration between when migration is initiated and when the original VM may be finally discarded is also considered very crucial, and therefore, there is a need to assess the VM migration. VM migration may be of vital significance as some times it smoothes out the delays exposed due to access downtime and total migration time. Memory transfer can be therefore adopted and can be implemented into three phases as follows:

- *Push phase*: The source VM continues running whilst certain pages are pushed across the network to the new destination for replication and availability purposes. To ensure consistency, pages modified during this process must be re-sent.
- *Stop-and-copy phase*: The source VM is paused and/or stopped according to the level of migration, whereas the pages are copied across to the destination VM, and then the new VM is started.
- *Pull phase*: The new VM executes with the requested contents and, if it accesses a page that has not yet been copied, then the requested page is being pulled from a VM source across the network.

The above migrations can be performed in a VDC conversion mode which performs both physical-to-virtual (P2V) and virtual-to-virtual (V2V) migrations, as shown in Fig. 6.8. This mechanism enables severe recoverability to all resources migrated to VMs from any physical data centre using the memory transfer implemented into the three phases mentioned above.

6.3.4.1 Virtual-to-Virtual (V2V) Resource Migration

As virtual-to-virtual (V2V) is a term that refers to the migration of application programs and data from a virtual machine or disk partition to another virtual machine or disk partition, it is important to mention that these mechanisms occur in a VE level in cloud computing. The target of this migration is to enable VMs to recover

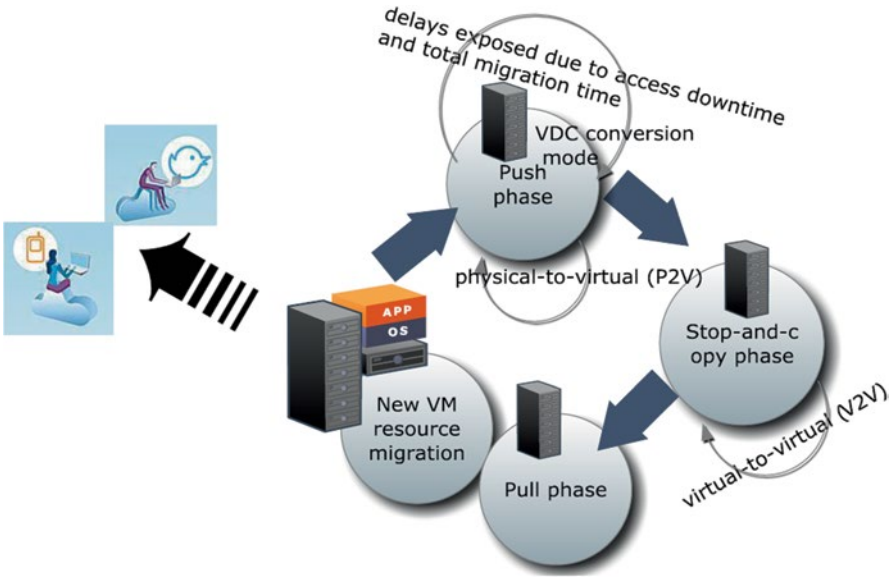


Fig. 6.8 VM migration and resource and memory transfer is implemented into three phases

any requested data from other VMs that are running on remotely hosted VDCs. To streamline the recoverability operation and enable the availability of the requested resources, part or all of the migration can be carried out automatically by means of specialised programs known as migration or virtual synchronisation tools. V2V can be used to copy or restore end-users' requested files, programs or OS components to a virtual machine from a different virtual machine. V2V migration tools can also be employed, in conjunction with physical-to-virtual (P2V) and virtual-to-physical (V2P) migrations, in order to copy the OS and applications from a computer's hard disk to a virtual machine and from there to an unlimited number of other virtual machines or computers. Even though there are many doubts and concerns about privacy and security regarding this migration, we have research indications that the resource allocation can be effectively balanced encompassing less failed requests due to the fact that it creates V2V resources' replicas that minimise the capacity loss for each application.

6.4 Future Trends and Research Issues

There are many open-ended issues in the cloud environment with a recent concept gaining sparkly ground as the mobile cloud computing. Mobile cloud computing was introduced in order to enable mobile users to have the requested information accessed, without client's mobile resources being utilised. This means

that in mobile cloud computing, both the data storage and the data processing occur outside of the mobile device. Mobile cloud applications move the computing power and data storage away from mobile phones and into the cloud, bringing applications and mobile computing to not just smart phone users but a much broader range of mobile subscribers. Moreover, mobile cloud computing is known to be a promising solution for mobile computing due to many reasons, e.g. mobility, communication and portability [52].

Some of the advantages of mobile cloud computing are figured with the battery lifetime extensibility, the improvement in data storage capacity and processing power where mobile users can store/access the large data on the cloud as well as with the improvement in reliability for many innovative applications. Some of the applications' paradigm that a fertile ground has been developed are the mobile commerce (m-commerce), mobile learning (m-learning) and mobile healthcare (m-healthcare) with the latter aiming to provide to mobile users with convenient access to health-related resources (e.g. patient health records). Further applications include the mobile game (m-game) and other practical applications like the keyword-based searching, voice-based searching and tag-based searching where through the mobile-cloud collaborative application environment one can contribute to detecting traffic lights and monitoring different areas in a landscape through a mobile device.

In addition, with the gigantic growth of social-enabled web during the past years and the creation of new cloud paradigms and platforms that encompass the social communication, a new application paradigm is utilised. This cloud application paradigm is enforced with the users' tendency to associate the social behaviour with their daily demands and utilise their social profile record [50]. As this paradigm can be utilised in all communication environments (wired and wireless), there is a need to eliminate the opportunistic aggravation of the resource-sharing process in the wireless communication and enable a mechanism which will consider the problem of resource sharing and content distribution across heterogeneously diffused resources and cloud paradigms. As mobility is one of the most distinguishing traits of today's devices having characteristics of the running 3As (anywhere, anytime, anything), the appropriate resource sharing and recovery dissemination strategy could be crucial for the offered QoS to the end-users. Infrastructureless MANets and MP2P networks due to the frequent and unexpected changes in their topology have many limitations and restrictions for any resource-sharing facility. In many cases, for facing problems of flooding-based approaches (like 'blind' diffusion), epidemic-directed resource migration solutions [51] are chosen as a 'gap fillers' to these issues. Therefore, a new stream in the cloud resource management policies will be the reliable resource sharing in opportunistic mobile peer-to-peer (MP2P) systems considering the group mobility in contrast to the resources and users availability and demands and the cloud recoverability. A very promising research area is the MP2P-like formation of cloud services, so that each mobile peer can act as a Migration Memory for any other connected peer who demands a resource—based on mobility profile [52] of the users—or in a location-based resource-sharing manner. Parameterised model can then be enriched in order to host temporal social metrics, location-aware social parameters and social interaction metrics [53] in order

to provide better resources utilisation and resources demand forecasting by mobile peers, from any cloud-oriented paradigm.

6.4.1 Open Research Issues

There are many research directions that need to be explored in order to enable consistent cloud resources with end-to-end reliability and availability. Therefore, new resource manipulation schemes are needed in which the mobile users can utilise multiple clouds in a unified fashion and using a common application interface. Resource manipulation schemes should be able to automatically discover and compose services for users using the so-called sky computing. Sky computing is an integrated model for cloud computing infrastructure where resources from multiple cloud providers are leveraged to create a large-scale distributed infrastructure. This infrastructure should have the support of an adaptive SLA between providers and small latency access to resources stored on the multiple cloud-based infrastructures. In a similar manner, the mobile sky computing, will enable providers to support a cross-cloud communication and enable users to access mobile services and applications running remotely gaining access to the end-recipient's outcome. However, to offer a service to both static and mobile user in a unified way, the service and context awareness and integration (i.e. convergence) would need to be explored further in terms of availability of resources (selection of high ranked available replicas and supported APIs). Requested resources outsourcing strategies where users can have any time access upon request should be dynamically encompassed in the architectural model of the cloud paradigm. In addition, API integration should also be established and host common mechanisms for controlling and managing the scalability of the potential scenarios, whereas it will enable dynamic access to resources regardless of their physical location.

6.5 Conclusion

Notwithstanding the cloud computing paradigm promises to be the ultimate outsourcing resources' and applications' solution starting from the individual users in small businesses scaling up to large enterprises and even governments, there are many resource consideration and constraints that should be addressed prior to the promulgation of services to the end-users. These resource considerations and constraints deal with the manipulation of resources requested by users and enable new mechanisms that face the QoS constraints and enable reliability in the offered services.

Efficient resource management in the cloud under the QoS constraints will be required for the efficient and profitable deployment of cloud services. As the cloud paradigm is based on the concept of virtualisation, the management of its resources

will have to start at the virtual level and descent to the real-world tangible level, where power consumption, CPU, memory and bandwidth allocation take place. The use of efficient algorithms for load balancing and scheduling will be required to manage the complex environment, and although such mechanisms exist, their application to cloud level is yet to be achieved.

API interoperability is hugely important, whereas cloud technology has not yet reached a level of maturity where academia and industry can through a convergence roadmap and standards develop their own infrastructure based upon a particular cloud API. As wide standards for manipulating the resources are not present, there will be still many distinctive APIs that may have a common target but through different pathways. Therefore, a common ground using an integrated API for a cloud infrastructure, and through it for an entire sky, will also give birth to new mechanisms for a more distributed resource management process where the users' workflow requests hosted in the cloud will be fully automated.

References

1. Yeo, C.S., Buyya, R., Dias de Assuncao, M., Yu, J., Sulistio, A., Venugopal, S., Placek, M.: Utility computing on global grids. In: Bidgoli, H. (ed.) Handbook of Computer Networks, p. 2008. Wiley, Hoboken (2008)
2. Mell, P., Grance, T.: The NIST definition of cloud computing. Technical Report Version 15, Information Technology Laboratory Newsletter, NIST (2009)
3. VMWare Inc.: VMware vSphere. In: The First Cloud Operating, White Paper (2009)
4. Sotomayor, B., Montero, R.S., Llorente, I.M., Foster, I.: Virtual infrastructure management in private and hybrid clouds. IEEE Internet Comput. **13**(5), 1422 (2009)
5. De Assuncao, M.D., Di Costanzo, A., Buyya, R.: Evaluating the cost benefit of using cloud computing to extend the capacity of clusters. In: Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing (HPDC 2009), Munich, 2009, pp. 141–150 (2009)
6. Rimal, B.P., Eunmi, C., Lumb, I.: A taxonomy and survey of cloud computing systems. Proceedings of Fifth International IEEE INC, IMS and IDC, Joint Conferences 25–27 Aug 2009 (NCM 2009), pp. 44–51. Seoul Korea (2009)
7. Google: Google Chrome OS: http://www.google.com/chrome (2012). Accessed 19 Apr 2012
8. Bolchini, C., Curino, C.A., Quintarelli, E., Schreiber, F., Tanca, L.: A dataoriented survey of context models. SIGMOD Rec. **36**(4), 1926 (2007)
9. Salehi, A.M., Javadi, B., Buyya, R.: QoS and preemption aware scheduling in federated and virtualized grid computing environments. J. Parallel Distrib. Comput. **72**(2), 231–245 (2012). ISSN: 0743–7315
10. Redhat http://www.redhat.com/. Accessed 27 Apr 2012
11. Salesforce.com: http://www.salesforce.com, Accessed 27 Apr 2012
12. Warrior, P.: CISCO CIO on cloud computing. http://blogs.cisco.com/news/comments/cisco_cto_on_cloud_computing/. Accessed 7 May 2012
13. Drepper, U.: Cost of the virtualization. ACM Queue, ACM, Feb (2008)
14. Kotsovinos, E.: Virtualization: Blessing or Curse? ACM Queue, ACM, Jan (2011)
15. OpenNebula: OpenNebula Project Leads: http://www.opennebula.org/. Accessed Feb 2012
16. OpenStack LLC: http://www.openstack.org. Accessed Feb 2012
17. Eucalyptus Systems, Inc.: http://www.eucalyptus.com/. Accessed Feb 2012
18. UC Santa Barbara: http://appscale.cs.ucsb.edu/. Accessed Feb 2012

19. IBM: IBM WebSphere Application Server: http://www.ibm.com/software/webservers/appserv/extend/virtualenterprise/. Accessed 17 Feb 2012
20. VMWare: http://www.cloudfoundry.com/. Accessed 17 Feb 2012
21. Varia, J.: Best practices in architecting cloud applications in the AWS cloud (ch. 18). In: Buyya, R., Broberg, J., Gościński, A. (eds.) Cloud Computing: Principles and Paradigms, pp. 459–490. Wiley, Hoboken (2011)
22. Gulati, A., Shanmuganathan, G., Ahmad, A., Holler, A.: Cloud-scale resource management: challenges and techniques. In: HotCloud, 2011, Portland, 14–15 June 2011
23. Epping, D., Denneman, F.: VMware vSphere HA and DRS technical deepdive. CreateSpace, Seattle (2010)
24. Saripalli, P., Kiran, G.V.R., Shankar, R.R., Narware, H., Bindal, N.: Load prediction and hot spot detection models for autonomic cloud computing. Proceedings of 2011 Fourth IEEE International Conference on Utility and Cloud Computing (UCC), 5–8 Dec, 2011, pp. 397–402 (2011)
25. Wang, G., Wang, K.: An efficient hybrid P2P MMOG cloud architecture for dynamic load management. Proceedings of 2012 International Conference on Information Networking (ICOIN), 1–3 Feb 2012, pp. 199–204 (2012)
26. Doyle, J., Shorten, R., O'Mahony, D.: "Fair-Share" for fair bandwidth allocation in cloud computing. Commun. Lett. IEEE **16**(4), 550–553 (2012)
27. Barroso, L.A., Dean, J., Hlzle, U.: Web search for a planet: the Google cluster architecture. IEEE Micro **23**(2), 22–28 (2003)
28. Platform Computing: LSF Version 4.1 Administrator's Guide. http://www.platform.com/services/support/ (2003). Accessed Mar 2012
29. Tannenbaum, T., Wright, D., Miller, K., Livny, M.: Condor: a distributed job scheduler. In: Gropp, W., Lusk, E., Sterling, T.L. (eds.) Beowulf Cluster Computing with Linux, pp. 307–350. MIT Press, Cambridge, MA (2001)
30. Kannan, S., Roberts, M., Mayes, P., Brelsford, D., Skovira, J.F.: Workload Management with LoadLeveler. IBM Redbooks, Poughkeepsie (2001). 2001
31. Yuan, D., Yang, Y., Liu, Y., Chen, J.: A data placement strategy in scientific cloud workflows. Futur. Gener. Comput. Syst. **26**(8), 1200–1214 (2010)
32. Liu, X., Heo, J., Sha, L.: Modeling 3-tiered web applications. Proceedings of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 27–29 Sept 2005, pp. 307–310 (2005)
33. Urgaonkar, B., Paci, G., Shenoy, P., Spreitzer, M., Tantawi, A.: An analytical model for multi-tier internet services and its applications. Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modelling of Computer Systems, 2005, pp. 291–302 (2005)
34. Zhang, Q., Cherkasova, L., Smirni, E.: A regression-based analytical model for dynamic resource provisioning of multi-tier applications. In: Proceedings of the 4th International Conference on Autonomic Computing, Jacksonville, 11–15 June 2007
35. Patterson, D.A.: A simple way to estimate the cost of downtime. Proceedings of the 16th USENIX Systems Administration Conference, Nov 2002, pp. 185–188 (2002)
36. Vishwanath, K.V., Nagappan, N.: Characterizing cloud computing hardware reliability. Proceedings of the 1st ACM Symposium on Cloud Computing, June 2010, pp. 193–204 (2010)
37. Chester, A.P., Leeke, M., Al-Ghamdi, M., Jarvis, S.A., Jhumka, A.: A modular failure-aware resource allocation architecture for cloud computing. In: Proceedings of the UK Performance Engineering Workshop (UKPEW'11), Bradford, 7–8 July 2011
38. Semeraro, G., Magklis, G., Balasubramonan, R., Albonesi, D.H., Dwarkadas, S., Scott, M.L.: Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. Proceedings of the 8th International Symposium on High-Performance Computer Architecture, 2002, pp. 29–42 (2002)
39. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. Proceedings of the 19th ACM Symposium on Operating Systems Principles, 2003, p. 177 (2003)

40. Kusic, D., Kephart, J.O., Hanson, J.E., Kandasamy, N., Jiang, G.: Power and performance management of virtualized computing environments via lookahead control. Clust. Comput. **12**(1), 1–15 (2009)
41. Srikantaiah, S., Kansal, A., Zhao, F.: Energy aware consolidation for cloud computing. Clust. Comput. **12**, 1–15 (2009)
42. Song, Y., Wang, H., Li, Y., Feng, B., Sun, Y.: Multi-tiered n-demand resource scheduling for VM-based data center. Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid-Volume 00, 2009, pp. 148–155 (2009)
43. Cardosa, M., Korupolu, M., Singh, A.: Shares and utilities based power consolidation in virtualized server environments. In: Proceedings of IFIP/IEEE Integrated Network Management (IM), Hofstra University, Long Island, 1–5 June 2009
44. Beloglazov, A., Buyya, R. (2010) Energy efficient resource management in virtualized cloud data centers. Proceedings of CCGRID 2010, pp. 826–831 (2010)
45. Sotomayor, R., Montero, S., Llorente, I.M., Foster, I.: Capacity leasing in cloud systems using the OpenNebula engine. In: Workshop on Cloud Computing and its Applications 2008 (CCA08), Chicago, Oct 2008 [Online publication]
46. Enormaly Inc.: http://www.enomaly.com. Accessed 14 May 2012
47. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The Eucalyptus open-source cloud-computing system. Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2009), May 2009, pp. 124–131 (2009)
48. Keahey, K., Foster, I.T., Freeman, T., Zhang, X.: Virtual workspaces: achieving quality of service and quality of life in the grid. Sci. Program. **4**(13), 265–275 (2005)
49. Keahey, K., Freeman, T.: Contextualization: providing one-click virtual clusters. In: eScience 2008, Indianapolis, Dec 2008
50. Mavromoustakis, C.X., Karatza, H.D. Embedded socio-oriented model for end-to-end reliable stream schedules by using collaborative outsourcing in MP2P systems. Comput. J. **54**(4), 19pp (2011)
51. Mavromoustakis, C.X.: Synchronized cooperative schedules for collaborative resource availability using population-based algorithm. Simul. Pract. Theory J. **19**(2), 762–776 (2011)
52. Mavromoustakis, C.X., Karatza, H.D.: Performance evaluation of opportunistic resource sharing scheme using socially-oriented outsourcing in wireless devices. Comput. J. **56**(2), 184–197 (2013)
53. Mavromoustakis, C.X., Dimitriou, C.D.: Using social interactions for opportunistic resource sharing using mobility-enabled contact-oriented replication. In: International Conference on Collaboration Technologies and Systems (CTS 2012). In Cooperation with ACM/IEEE, Internet of Things, Machine to Machine and Smart Services Applications (IoT 2012), Denver, 21–25 May 2012

Chapter 7
Management Infrastructures for Power-Efficient Cloud Computing Architectures

Antonio Corradi, Mario Fanelli, and Luca Foschini

Abstract The surging demand for inexpensive and scalable IT infrastructures has led to the widespread adoption of Cloud computing architectures. These architectures have therefore reached their momentum due to inherent capacity of simplification in IT infrastructure building and maintenance, by making related costs easily accountable and paid on a pay-per-use basis. Cloud providers strive to host as many service providers as possible to increase their economical income and, toward that goal, exploit virtualization techniques to enable the provisioning of multiple virtual machines (VMs), possibly belonging to different service providers, on the same host. At the same time, virtualization technologies enable runtime VM migration that is very useful to dynamically manage Cloud resources. Leveraging these features, data center management infrastructures can allocate running VMs on as few hosts as possible, so to reduce total power consumption by switching off not required servers. This chapter presents and discusses management infrastructures for power-efficient Cloud architectures. Power efficiency relates to the amount of power required to run a particular workload on the Cloud and pushes toward greedy consolidation of VMs. However, because Cloud providers offer Service-Level Agreements (SLAs) that need to be enforced to prevent unacceptable runtime performance, the design and the implementation of a management infrastructure for power-efficient Cloud architectures are extremely complex tasks and have to deal with heterogeneous aspects, e.g., SLA representation and enforcement, runtime reconfigurations, and workload prediction. This chapter aims at presenting the current state of the art of power-efficient management infrastructure for Cloud, by carefully considering main realization issues, design guidelines, and design choices. In addition, after an in-depth presentation of related works in this area, it presents some novel experimental results to better stress the complexities introduced by power-efficient management infrastructure for Cloud.

A. Corradi • M. Fanelli • L. Foschini (✉)
DEIS, University of Bologna, 40136 Bologna, Italy
e-mail: antonio.corradi@unibo.it; mario.fanelli@unibo.it; luca.foschini@unibo.it

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_7,

Keywords Cloud computing • Management infrastructure • Power consumption • Power efficiency • VM consolidation • SLA

7.1 Introduction

In the last few years, we witnessed the widespread and general adoption of Cloud architectures [1–3]. Cloud solutions introduce new interesting and compelling opportunities, by favoring the birth of new start-ups that do not need to invest up-front money to build their own IT infrastructures. In addition, Cloud architectures rely upon and melt together different preexisting technologies, such as resource virtualization, service-oriented architectures, and grid computing, to enable the scalable and elastic provisioning of computational resources.

With a closer business perspective, Cloud solutions usually distinguish three main types of actors: *service users*, *service providers*, and *Cloud providers*. Service users are the final clients that require access to particular online services. Service providers seize the opportunity to build new services and tend to externalize the execution of their own services to avoid the deployment of costly private IT infrastructures. Finally, Cloud providers are usually big players, such as Amazon, Google, and IBM, that offer to service providers all the system resources needed to execute their services on a pay-per-use basis.

To increase their economical revenue, Cloud providers tend to host as many service providers as possible; toward that goal, modern virtualization technologies enable the provisioning of multiple VMs on the same host and allow reducing unemployed computational resources. At the same time, such techniques enable a brand new set of reconfigurations actions, including dynamic VM resizing and migration, extremely useful to perform runtime management of the physical infrastructure. VM migration is a fundamental means to dynamically consolidate VMs on as few hosts as possible, so to possibly switch off unneeded hosts and reduce the power consumption of the data center (DC) [4]. Even if Cloud providers exploit big DCs where hundreds of powerful hosts communicate by cabled networks with gigabits of bandwidth, the runtime management of such resources is becoming a fundamental issue due to the high and increasing number of service providers, the time-varying nature of service workloads, DC power consumption limitations, etc.

In particular, it is worth noting that the energy required by both computational resources and cooling system represents a significant economical expenditure for the Cloud provider; for instance, in [5], the authors note that, depending on utilization, the energy costs of a single data center may exceed $15 M per year. From an environmental viewpoint, instead, Gartner estimates that the whole IT sector is responsible for the 2 % of global carbon dioxide emissions [6]. Hence, also fuelled by the emerging Green Computing research area [7, 8], *management infrastructures for power-efficient computing architectures* are receiving increasing attention in the literature. More formally, *power efficiency* relates to the amount of power required to process a workload in the Cloud: we define DC_A more power efficient than DC_B if and only if

the amount of power required by DC_A to perform a particular workload is lower than the one required by DC_B to perform the same workload. However, while power efficiency pushes toward the greedy consolidation of VMs, Cloud providers must comply also with Service-Level Agreements (SLAs) agreed with service providers and enforce them at runtime to prevent unacceptable performance degradations.

In conclusion, power-efficient management infrastructures for Cloud are extremely complex software stack solutions, and their design has to take into account different heterogeneous aspects, including SLA representation and enforcement, runtime reconfigurations due to VMs migration, workload prediction, and dynamic adaptation of DC cooling systems. A large number of research proposals and solutions recently emerged address power efficiency in Cloud settings, each one with its specific goals, advantages, and limitations. However, although different research efforts share the goal of reducing the Cloud power consumption while ensuring service providers' SLAs, to the best of our knowledge, only a few of them have started to explore opportunities and problems connected to the usage of service profiles, SLAs, and DC status to guide the consolidation process.

This chapter addresses the above issues by highlighting main technical challenges and by proposing practical design guidelines and models for the creation of energy-friendly VM allocation solutions. We claim that only full awareness of VMs execution environment can enable more effective reconfiguration decisions. In other words, novel approaches capable of exploiting full visibility of all those characteristics that describe the current execution environments, namely, *service awareness*, *SLA awareness*, and *DC awareness*, are fundamental to enable smarter and effective management operations. First, by exploiting information on service type, the management infrastructure can exploit increased visibility to mitigate VM migration side effects by deciding a proper mix of VMs per hosts and by prioritizing different Cloud reconfigurations so to avoid configurations that would likely lead to SLA violation. Second, by exploiting negotiated SLAs, the management infrastructure can prevent hosts overload and validate possible reconfigurations for the sake of placement stability. Finally, by exploiting additional information about DC status, the management infrastructure can better tailor reconfiguration decisions: the number of powered-on hosts, the current VM allocation status, and the workload introduced by each VM are all fundamental information to guide dynamic DC reconfigurations and avoid useless and costly VM migrations.

The chapter is organized as follows. Sections 7.2 and 7.3 introduce main requirements, sketch our design model – by using service, SLAs, and DC awareness as three main classification criteria – and clarify main design guidelines. Section 7.4 analyzes important research works available in literature by comparing them against the main directions considered for VM placement computation, so to better assess the technical soundness of our design guidelines. Finally, in Sect. 7.5, we detail our novel reference model for power-efficient Cloud management infrastructures: we introduce all main involved runtime phases (system monitoring, VM placement computation, and runtime system reconfigurations), and we present some experimental results, thus offering a complete view on the addressed topic. Conclusions and directions for future work are presented at the end of the chapter.

7.2 VM Placement in Modern Cloud

As stated before, Cloud architectures exploit virtualization technologies to provide multiple VMs, possibly belonging to different service providers, on the same host. Power-efficient management infrastructures need to carefully address the placement problem since the overall DC power consumption is strictly related with the number of powered-on hosts, networking elements state, and so on [9]. VM placement has to consider many heterogeneous constraints, spanning from service SLAs to limited computational resources, and to enforce anti-co-location VM constraints aimed to avoid that the failure of a single host deeply impacts service reliability (for instance, when all the Web servers giving access to the same service are on one host).

From a more formal perspective, given a set of hosts with limited resources and a set of VMs with well-defined resource requirements, the Cloud management infrastructure has to decide both final VM-to-host mappings and alternative VM relocation plan. In fact, VM placement is a continuous process where the management infrastructure periodically checks if a better placement exists and, if that is the case, reconfigures the current configuration through the definition of a new relocation plan. Modifying a preexisting VM configuration may involve several VM migrations and network reconfigurations and that, in its turn, introduces an important problem, namely, the design of a VM relocation plan useful to bring the DC from an initial state to a new desired one. For the sake of clarity, and in order to highlight main optimization goals and constraints, in the remainder we specifically focus on the placement function only.

Starting with the main optimization goal, a management infrastructure for power-efficient Cloud usually strives to increase host resource utilization, typically by switching off not required hosts. In this case, a very simple goal would be to consolidate the DC workload on as few hosts as possible and then to switch off not required hosts. However, that represents a very simplistic assumption since host power consumption depends on current computational load and heterogeneous hosts can have different power efficiency, thus requiring different amounts of energy to process the same workload [8]. In addition, scaling up to large DCs, an important fraction of the DC power consumption is actually related to cooling systems and networking [5, 10]. Hence, the objective function should consider and weight different sources of power utilization, including hosts, network, and cooling systems. Apart from that, other additional optimization goals, somewhat opposed to power efficiency, have to be considered. For instance, load balancing between hosts is important to prevent resource shortages and frequent VM relocations; similarly, reliability considerations push toward the placement of VMs belonging to the same service on different hosts, thus leaning against greedy consolidation. Hence, the placement function has to trade off management decisions about different aspects, and that complicates both the design and the validation of such solutions.

Moreover, the VM placement function has to model and consider all resource constraints because each host has limited resources in terms of CPU, memory, and I/O operations, and the placement process has to consider them to prevent solutions

that would require more resources than available ones. In addition, more complex constraints associated with the whole DC have to be considered for the sake of placement feasibility. On the one hand, it could be impossible to keep powered-on all the hosts due to limitations on aggregated power consumption and cooling systems. On the other hand, the DC network infrastructure can introduce tight constraints on the placement due to limited networking capacities; in particular, different aspects, including time-varying traffic demands, network topology, and dynamic multipath routing schema, make the definition and the enforcement of such constraints very difficult in the general case [11, 12].

7.3 Taxonomy of Power-Efficient Management Infrastructures

In the last few years, many research works addressed the problem of power-aware VM placement in Cloud systems and several industrial products are available for energy-efficient dynamic VM provisioning. At the same time, to the best of our knowledge, these solutions do not exploit full awareness of (i) running services, (ii) service providers' SLAs, and (iii) DC status to select the optimal power-efficient placement and to guide the VM relocation process.

This section details the three awareness dimensions, namely, service, SLA, and DC awareness, that a power-efficient Cloud management infrastructure should use to optimize the placement process. In particular, for each dimension, we introduce possible taxonomy directions, so to better clarify how each dimension affects the design and the decisions of the management infrastructure.

7.3.1 Service Awareness

In the public Cloud vision, service providers deploy their own services on a public Cloud provider that does not usually have any specific knowledge about hosted services. Similarly, in the private Cloud vision, a private DC executes highly heterogeneous services that may result in completely different workloads, ranging from CPU-bound to I/O-bound requirements. Broadly speaking, since the Cloud provider does not exactly know the services that will be deployed in the future, foreseeing possible side effects of specific reconfiguration decisions is one of the biggest challenges for Cloud management infrastructures. Hence, to ease that task, we claim that the Cloud management infrastructure should let service providers categorize services along two main directions, as shown in Fig. 7.1. The first one considers the prevailing type of resource needed and used at runtime, while the second one takes care of possible runtime interactions with users.

Along the first direction, we distinguish *CPU-bound services*, *memory-bound services*, and *I/O-bound services*. *CPU-bound services* introduce high CPU

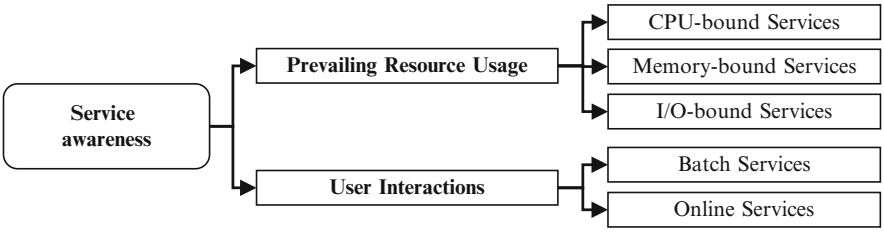


Fig. 7.1 Taxonomy used to categorize power-efficient Cloud management infrastructures according to service awareness

utilizations to perform intensive computations, such as data crunching algorithms. Services belonging to this category typically exhibit a small memory footprint and have a lightweight I/O workload, both toward local disks and network interfaces. *Memory-bound services*, instead, usually result in big memory footprints, such as memcached that needs huge amounts of memory to keep an in-memory cache of active sessions or data objects [13]. In this case, if the service offers simple query mechanisms for data retrieval, it will introduce limited workloads along other resource dimensions. *I/O-bound services* have both simple computation and limited memory footprints but introduce heavy I/O workload, such as big data exchanges with other hosts. For these services, the efficiency of I/O operations is fundamental to grant adequate service provisioning quality levels. Finally, let us remark that, since different services introduce heterogeneous workloads, VM consolidation can affect the final performance in different ways. For instance, it is well known that the overhead introduced by virtualization depends on both the number of VMs that access shared resources and the type of required operations. Hence, the Cloud management infrastructure has to reach a good mix of VMs placed on the same host, so to balance resulting workloads on each resource dimension.

Along the second direction, we consider *batch services* and *online services*. We define *batch services* as those ones that do not involve user interactions; as a consequence, the management infrastructure has more freedom in choosing potential reconfigurations since they will not have a direct impact on the quality perceived by service users. For instance, since raw CPU computation services are interested in receiving a specific amount of CPU time in a particular time frame, the management infrastructure has to guarantee a CPU budget only, but the constraint on network responsiveness is relaxed. *Interactive online services*, instead, are the ones that present tighter constraints on average response times; in this case, the management infrastructure should monitor incoming traffic and prevent dangerous network saturations that could result in dropped requests.

Therefore, one of our major claims is the need of service-aware approaches that do not merely treat VMs as black boxes, but rather are able to exploit increased knowledge on running services to better guide the VM placement process and better shape reconfiguration actions. As stated before, information on prevailing resource usage can be used to mix the VMs co-located on the same host, so to prevent biased

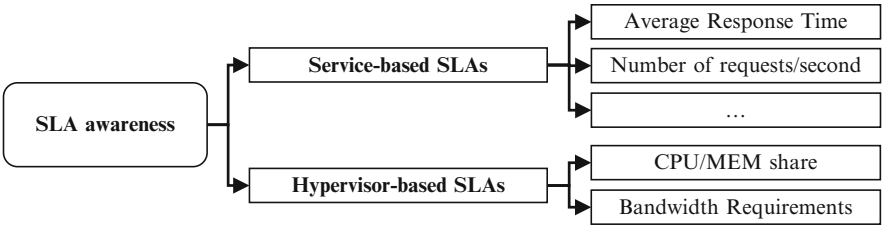


Fig. 7.2 Taxonomy used to categorize power-efficient Cloud management infrastructures according to SLA awareness

resource usages that increase the probability of performance degradation. Similarly, information on user interactions defines additional constraints on the VM placement process and can be used to prioritize specific reconfiguration actions.

7.3.2 SLA Awareness

A power-efficient Cloud management infrastructure has to properly handle all the main phases involved in SLA lifetime, spanning from initial definition to runtime monitoring and enforcement of SLAs. On the one hand, service providers require the definition and the enforcement of proper SLAs to ensure the correct provisioning of their services. On the other hand, Cloud providers have to offer SLA monitoring and enforcement mechanisms, by exploiting the full awareness of different service types to carefully tune SLA enforcement in the DC.

In finer details, service providers' SLAs can comprehend very heterogeneous metrics, including high-level service-based ones (average response times, number of queries for each second, etc.) and low-level hypervisor-based ones (CPU/MEM share, ensured network bandwidth, etc.). Along these main directions, shown in Fig. 7.2, we categorize preexisting works in two main categories, namely, *service-based SLAs* and *hypervisor-based SLAs*.

Expressive *service-based SLAs* are more difficult to ensure and need the introduction of complex monitoring infrastructures to effectively distribute and process performance indicators injected by services; at the same time, they should simplify the deployment of new services from the service providers' viewpoint. *Hypervisor-based SLAs*, instead, are simpler to enforce and usually require the Cloud management infrastructure to issue simple reconfiguration commands to the host hypervisors. Unfortunately, hypervisor-based SLAs complicate the deployment of new services from the service providers' viewpoint, since they will have to properly size the required deployment in terms of CPUs and MEM.

To be more concrete, let us introduce a very simple example in which a service provider deploys a new e-commerce Web site on the Cloud. This service presents highly time-varying workloads and can exhibit flash crowd phenomena [14]. In this

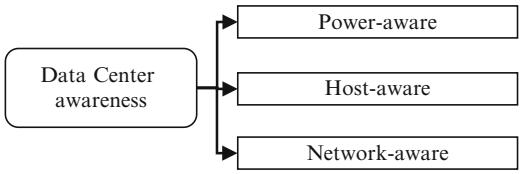


Fig. 7.3 Taxonomy used to categorize power-efficient Cloud management infrastructures according to DC awareness

case, an auto-scaling facility is fundamental to accommodate incoming requests, without having to pay for unnecessary computational resources. If the SLA offered by the Cloud provider is expressed in terms of average retrieval time (i.e., service-based SLAs), the service provider can deploy the new service and declare a target response time value; the Cloud management infrastructure, in its turn, will then take care of scaling the number of service instances when the average retrieval time becomes too high. When SLAs offered by the Cloud provider allow for the definition of raw CPU and memory resources (i.e. hypervisor-based SLAs), instead, the service provider will be in charge of initially conducting a thorough performance assessment to dimension the required deployment configuration. Let us note that, although auto-scaling is feasible also in this second case by triggering adaptations according to CPU load indicators, it will be not directly related with the performance experienced by service users.

7.3.3 Data Center Awareness

Finally, a management infrastructure for power-efficient Cloud should consider the state of the DC to better shape VM placement and reconfiguration decisions. In particular, VM live migration introduces a high management overhead, both at the sender and at the destination node and in terms of both CPU and network traffic, that can lead to runtime violations of service providers' SLAs [15, 16]. Of course, that complicates the design of the management infrastructure because it has to take care of multiple criteria during VM placement and relocation computation, including service providers' SLAs, hosts resources, and network status.

Along this direction, as shown in Fig. 7.3, we categorize management infrastructures for power-efficient Cloud in (1) *power-aware*, (2) *host-aware*, and (3) *network-aware*. In the remainder, we analyze one by one these directions to better highlight why they are important for the design of power-efficient management infrastructures.

First of all, power-efficient management infrastructures should be aware of the *power consumption* introduced by the current DC configuration to understand whether and to what extend the power efficiency can be improved. In this case, the power consumption of the DC includes hosts and networking elements, as well as power required by cooling systems [5]. Of course, DC power consumption is intrinsically related with its own computational power and increases when additional

hosts and network elements are powered on. However, to further increase energy savings, Cloud management infrastructures should always optimize power efficiency, namely, as the ratio between processed workload and current power consumption. In addition, they should consider additional and more sophisticated mechanisms available in modern CPUs, such as Dynamic Voltage Frequency Scaling (DVFS), to better trade off the number of powered-on hosts, the computational power of each host, and the aggregated power consumption.

Power-efficient management infrastructures should also be aware of *host resources*, because all VMs placed on the same host share the same local resources, mainly CPU, memory, and I/O. Although power efficiency pushes toward the greedy consolidation of VMs in as few hosts as possible, the management infrastructure has to enforce service providers' SLAs. In other words, it has to consider available resources and deployed VMs by taking care of hypervisor reconfigurations to ensure that each VM will receive only the amount of resources it is entitled for. In addition, since CPU and I/O workload measurements are highly time varying, filtering and forecasting techniques are useful to avoid erroneous management decisions based on temporary load spikes [17]. Interferences between co-located VMs could also hinder the precise estimation of introduced load indicators and that, in its turn, affects the evaluation of the final VM placement [18].

Finally, power-efficient management infrastructures should be aware of *DC network infrastructure*. DC networks usually exploit complex network topologies for the sake of scalability and reliability: at the current stage, several solutions, such as Fat-tree [19], VL2 [20] and BCube [21], have been proposed with the main goal of increasing inter-host available bandwidth. Although Virtual LANs (VLANs) can be used for security reasons, we have to consider that the network is intrinsically shared; hence, traffic demands coming from different Cloud customers interfere among themselves [12]. The monitoring of inter-VM traffic demands is not easy to address due to several issues, such as its time-varying characteristics and potential runtime reconfigurations automatically triggered by running services. Hence, the management infrastructure should carefully monitor and control traffic demands to prevent resource shortage at runtime and to perform VM consolidation while preventing dangerous link saturations. Finally, we remark that VM placements that are unfeasible from the network perspective are difficult to recover even via VM live migration. In fact, if network links are close to congestion, the introduced overhead can definitely choke close-to-congestion links and network elements; therefore, the management infrastructure has to consider the overall DC network status before making any VM relocation decision.

7.4 Related Works

In this section, we exploit the taxonomy introduced to survey the main state-of-the-art research works in this area, by classifying them according to the main attributes used for VM placement computation. Let us remark that, for the sake of clarity,

we voluntarily focused more on power-aware research works, namely, VM placement problems that explicitly aim to reduce overall DC power consumption. At the same time, we decided to cite also a few works not explicitly dealing with power awareness, since they are useful to highlight other interesting research directions ahead.

In particular, Sect. 7.4.1 categorizes the different research works according to whether they consider information on deployed services. Then, Sect. 7.4.2 focuses on the used SLA definitions; finally, Sect. 7.4.3 presents solutions that exploit a specific awareness about the current DC status in VM placement computation, so to account for VM relocation costs and to selectively choose suitable configuration actions.

7.4.1 Service Awareness

As explained in Sect. 7.3.1, we categorize services executed over the Cloud along two main directions: first, depending on the type of prevailing resource type (*CPU-bound*, *memory-bound*, and *I/O-bound*) and, second, according to the presence of user interactions (*batch* and *online*). Below, in our presentation, we will follow a chronological order; we will supply a brief introduction to the major research works and then we will categorize considered solutions according to our service-awareness taxonomy.

pMapper is a power and migration cost-aware placement framework presented in [22]. Authors consider the placement of applications in Cloud environments, while accounting for SLA violations and migration costs; they propose several strategies, based on host power efficiency and First Fit Decreasing heuristics, to reach proper trade-offs between performance and power consumption. A large set of shown experimental results confirms that there are interferences between VM live migrations and supports the technical validity of pMapper. However, to the best of our knowledge, authors do not focus on a specific service type, but rather assume the existence of a generic function to evaluate that the computed placement is feasible and complies with agreed SLAs.

In [23], authors focus on the Cloud resource provisioning problem for real-time services, with the main goal of reducing total power consumption required to process them. A real-time service is made by a set of tasks, each one described in terms of release time, relative deadline, and required execution time; if necessary, it supports also periodic tasks. In the adopted model, each task is described in terms of the worst-case execution time, and each host is associated with a maximum capacity in million instructions per second (MIPS). Because host power consumption depends on the current CPU frequency, it can be dynamically adjusted through traditional frequency-scaling techniques. Hence, according to our taxonomy, authors focus on CPU-bound batch services: VM placement considers computation only and exploits CPU requirements, while it does not consider any constraint on user interactions.

In [24], authors focus on CPU-bound online services by presenting a new optimization model to find the minimum number of physical hosts required to execute

a specific workload. The workload consists of different jobs, each one characterized by a number of requests/second to be processed. Authors assume that it is possible to relate each request with a number of instructions to be executed and use that indicator to handle the placement phase. It is worth noting that, apart from retrieving the minimum number of powered-on hosts, the model proposed by authors also finds the optimal value of the host CPU frequencies. As a possible drawback, authors make the assumption that Cloud jobs can be arbitrarily split between hosts to meet the required number of requests/second; although that can be viable in some scenarios, it does not fit well in the general VM allocation problem.

In [25], authors present different techniques to reduce runtime DC power consumption. They exploit host CPU load to automatically trigger VM migrations: if host CPU load goes beyond a specific upper bound, some VMs are migrated to prevent CPU saturation; when host CPU load becomes lower than a specific lower bound, they migrate all the VMs so to empty the host and switch it off. To the best of our knowledge, in this work authors do not consider any specific service type but exploit the CPU load indicator only to trigger migrations and VM consolidation operations aimed to use as fewer hosts as possible and to reduce total power consumption. Presented simulation results, obtained by the CloudSim simulator [26], demonstrate that all the proposed policies can largely reduce the DC power consumption.

In [27], authors focus on power-efficient VM placement problem by proposing a two-phase approach. The first phase finds the optimal number of VMs to meet service providers' SLAs, while the second one details VM-to-host mappings with the main goal of reducing powered-on hosts. The management infrastructure exploits service-supplied utility functions to evaluate the proposed VM placement; hence, different types of services can be supported by properly adjusting supplied utility functions.

Mistral is a novel solution to optimize VM placement while reducing power consumption [28]. Mistral considers transient costs associated with runtime reconfigurations and exploits dynamically updated indicators on placement stability to carefully tailor DC reconfiguration; toward that goal, authors introduce filtering and forecasting techniques to better weight management decisions. In addition, Mistral exploits A*-search techniques to find suitable plans of reconfiguration actions needed to reach the new placement. Considering the notion of SLA adopted by Mistral, i.e., a target mean response time for each service, we can safely assume that authors target online services.

In [29], authors present a new management infrastructure for multi-tier online applications deployed over the Cloud. The proposed solution does not merely treat single VMs but is able to manage complex applications composed by multiple VMs. Moreover, it exploits feedbacks, coming from the application controller, to both dynamically size VMs and reallocate them so to increase power efficiency. It is worth noting that authors exploit Multiple Input-Multiple Output (MIMO) control theory to grant the average response time in the SLA contract. In addition, they consider both CPU constraints and the amount of CPU resources allocated to each VM. In brief, this chapter focuses on online CPU-bound services.

Finally, in [30], authors propose a new energy-aware framework to dynamically consolidate VMs in Cloud DCs. The framework targets online iterative placement through successive adjustments of a preexisting configuration. At each step, the management infrastructure considers possible VM movements and estimates expected energy savings. At the same time, authors account for the energy consumed by VM migrations, by performing VM relocation only if the difference between saved energy and energy spent in the migration is positive. To the best of our knowledge, due to the very broad definition of adopted service SLAs, authors do not try to target a specific type of services.

7.4.2 SLA Awareness

Service providers' SLAs are fundamental inputs to be considered by the management infrastructure because they are crucial to guide VM placement and prevent unfeasible solutions. Along that direction, as detailed in Sect. 7.3.2, we consider *service-based SLAs* and *hypervisor-based SLAs* depending on used performance indicators. Let us also briefly note that some research works do not consider service providers' SLAs at all; for instance, in [25], authors dynamically reconfigure the DC to reduce the total power consumption, but the lack of SLAs leads to completely blind decisions.

Moving to SLA-based approaches, we preliminarily note that most considered related works adopt a service-based SLA approach. The work presented in [23] considers jobs that have to finish before a deadline supplied by service providers. The SLA is service based because it expresses service-related temporal deadlines: in this case, SLAs prevent aggressive consolidations that would result in jobs not meeting agreed deadlines, a dangerous situation if we target real-time scenarios. Similarly, in [24], authors consider a service-based SLA scenario where each service provider supplies an SLA that details the number of requests/second to serve with response time constraints. The usage of such performance metrics introduces novel challenges during VM placement: in fact, authors assume that, monitoring past executions, the management infrastructure can associate each request with an exact number of instructions to execute. However, the validity of such assumption in a real deployment is difficult to assess, mainly because the management infrastructure should dynamically inspect each VM. In [29], authors adopt a service-based SLA approach to give the chance of declaring average response times to the Cloud management infrastructure that, in its turn, dynamically resizes VMs by changing CPU shares. While previous approaches consider SLAs as strict constraints on final placement solutions, other approaches exploit SLAs only to weight the overall utility of the VM placement. For instance, both pMapper [22] and the work presented in [27] define specific utility functions and they use them to increase the cumulative service utility; of course, utility functions can capture SLAs even though they do not introduce real constraints on the placement problem. Similarly, in [28], authors use a reward-/penalty-based schema that considers SLA average

response times; however, also in this case, SLAs do not represent constraints on placement computation.

In conclusion, from above-related works, we can conclude that service-based SLAs are preferred to hypervisor-based ones. At some stage, the Cloud management infrastructure will map service-based SLAs to hypervisor-based ones to properly reconfigure host hypervisors; however, in real-world deployments, it would be better to supply more expressive service-based SLAs to service providers, so to simplify the deployment phase. In fact, as briefly mentioned before, SLAs can be use in VM placement computation either as part of the objective function or as constraints to be enforced. On the one hand, although the first approach ensures higher flexibility to the management infrastructure, it does not ensure strict performance requirements and is not suitable for critical services. On the other hand, the second approach could lead to failures of placement computation more likely. As a possible trade-off solution, the management infrastructure could consider different levels of service criticality and enforce strict SLAs only if needed.

7.4.3 Data Center Awareness

The management infrastructure has to be aware of computational resources available into the DC to detail final VM-to-host mappings. Although modern DCs can feature heterogeneous hosts (different types of CPUs/GPUs can be more suitable for specific types of workloads), DCs still tend to adopt homogeneous solutions because they simplify the placement problem. Similarly, from the network perspective, state-of-the-art network architectures, e.g., Fat-tree and VL2 [19, 20], exploit symmetric solutions where each host has the same nominal bandwidth, so to avoid more complex placement problems. Therefore, in the following, we explicitly assume homogeneous DCs in terms of both hosts and network facilities. Apart from that, as clarified in Sect. 7.3.3, we categorize power-efficient placement problems in *power-aware*, *host-aware*, and *network-aware*.

Given the specific focus of this chapter, we restricted our search only to the set of related works that exploit DC awareness in the VM placement process to reduce power consumption. In pMapper [22], authors empirically evaluate the power consumption introduced by different types of workload and exploit it to estimate host power efficiency and to guide the placement process. In both [23] and [25], authors directly relate host power consumption with CPU frequency according to the formula $C*f^3$, where C is a constant and f is the CPU frequency, and it is configured through DVFS according to the MIPS required by placed workloads. Similar assumptions are exploited in [24], where authors consider that host power consumption includes also a fixed idle power, not related with current workload. Differently, in [27], authors exploit the ratio between the aggregated CPU demand of placed VMs and the total host CPU capacity to account for power cost. In [28], each host is associated with an empirical nonlinear model, and the placement computation step exploits a utilization-based model to predict final DC power consumption.

Finally, for the sake of completeness, we remark that, apart from these more traditional approaches, new works are explicitly dealing with reducing the power consumption of the network elements. In [31], authors present VMFlow, a placement framework that maps VM-to-host while reducing network power consumption; unfortunately, authors consider only one VM for each host and do not treat the general VM placement problem. With a slightly different perspective, ElasticTree dynamically and continuously adjusts DC networking according to current traffic demands [10]; at each step, it identifies the minimum number of links and switching elements required to route current traffic and switches off not required ones. As showed by authors in [10], the dynamic adaptation of the DC network infrastructure through software-defined techniques can result in high power savings and can be exploited also by Cloud management infrastructures.

At the same time, most of the presented solutions are host-aware, namely they consider limited host resources to elaborate the final VM placement plan. Host resource capacities are usually treated as hard constraints to prevent unfeasible solutions. For instance, models in [23] and [25] host CPU capacity with a maximum number of MIPS. In [27], hosts are modeled with maximum CPU and memory capacities, while VMs have maximum CPU/memory requirements; then, the placement process ensures that the aggregated CPU/memory requirements introduced by placed VMs are lower than available host capacities. In general, any placement problem has to consider host resource constraints to prevent the greedy consolidation of all the VMs on one host; hence, even if not explicitly stated, all the previous works exploit host-awareness to control VM consolidation and to model both host capacities and VM requirements. In addition, to deal with power efficiency in the objective function, it is important to turn the host load into introduced power consumption; that can be addressed by using both mathematical and empirical approaches with different levels of precision that trade off accuracy and computational complexity.

Finally, all the considered related works do not explicitly consider network awareness: VM traffic demands are not included into placement computation and the network topology does not introduce additional constraints in terms of maximum bandwidth between different hosts. Although not strictly related with the focus of this chapter, it is important to note that network awareness is important and novel placement problems are emerging in this area. In [12], authors study the problem of VM placement with the goal of reducing the aggregate traffic into the DC. In [32], authors consider applications made by a computation and a storage part and propose a new optimization problem to place them while reducing runtime DC network traffic. In [11], we propose a new network-aware placement problem that places VMs so to reduce the worst-case load ratio over all the network cuts with the main goal of increasing placement stability with time-varying traffic demands. Finally, while network-aware research works usually assume well-defined traffic demands, some seminal works are starting to use equivalent capacity notions to co-locate VMs with uncorrelated traffic demands [33]; in particular, these approaches use stochastic variables to represent traffic demands and place VMs by ensuring that host network capacity is violated with a probability lower than a predefined threshold.

In conclusion, although related works considered in this chapter exploit DC awareness in terms of power consumption and host resources, we think that a lot of work is still required toward placement problems that consider both host-level and network-level aspects during power-efficient VM placement computation. In fact, our analysis of state-of-the-art literature confirmed that different previous works dealt with one specific aspect, either hosts or network, but none tried to address the whole complexity introduced by considering both hosts and network elements within the same placement problem.

7.5 Power-Efficient Management Infrastructure

The full awareness of VM execution context can be effectively used to take more effective DC reconfiguration decisions. The management infrastructure should exploit full visibility of any available information, namely, service/SLA/DC awareness, to guide VM placement decisions and runtime reconfigurations. First, by using service types, the management infrastructure can better estimate side effects of VMs migration to schedule needed reconfigurations and to decide the best mix of VMs for each host, in order to avoid placement solutions stressing too much a single-resource dimension only. Second, by using service providers' SLAs, the management infrastructure can preventively validate Cloud reconfigurations to increase placement stability; also, it can issue reconfiguration commands to host hypervisors so to enforce resource reservation mechanisms. Finally, by using information on DC status, the management infrastructure can reduce the space of possible reconfiguration decisions; for instance, number of powered hosts, current VM placement, and characterizations of VMs workloads are useful elements to guide DC reconfigurations and to prevent expensive and useless migrations.

Putting together all these requirements, the management infrastructure of a power-efficient Cloud has to address several heterogeneous aspects, including monitoring data gathering and processing, VM placement elaboration, and actuation of a suitable reconfiguration plans. To deal with such complexity, we can decompose the management infrastructure in three main stages as shown in Fig. 7.4: the first stage is in charge of DC monitoring, the second one exploits collected data to understand if a more suitable VM placement exists, and, finally, if that is the case, the third stage computes and applies a VM relocation plan. Hence, the proposed management infrastructure consists of a *Monitoring Layer*, a *Power-Efficient Placement Computation Layer*, and a *DC Reconfiguration Layer* that work together as a pipeline; in the remainder, we better detail and clarify associated management duties of each layer.

The Monitoring Layer collects monitoring information associated with current DC configuration. It gathers both host-level and DC-level information: host CPU, memory, and I/O load indicators are examples of host-level information; aggregated power consumption, network elements status, and cooling system state are, instead, examples of DC-level monitoring information. At the same time, this layer interacts

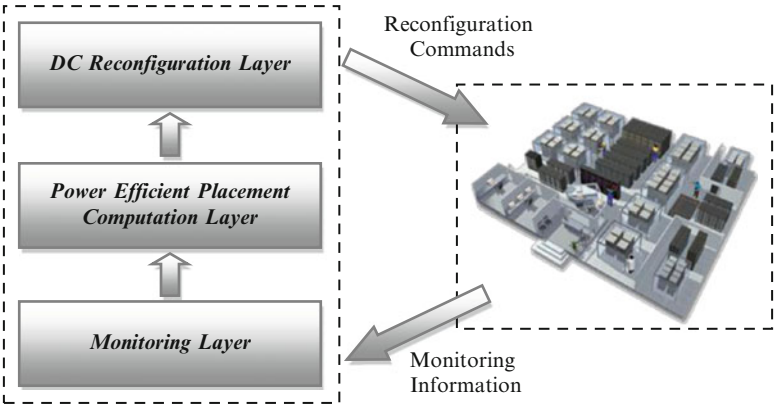


Fig. 7.4 Logical architecture of a power-efficient Cloud management infrastructure

with deployed services to collect high-level monitoring data, such as average response times, depending on adopted service providers' SLAs.

The Power-Efficient Placement Computation Layer exploits information on service types, service providers' SLAs, and DC status to assess if a better VM placement exists. Apart from monitoring information coming from the Monitoring Layer, it also queries external components of the management infrastructure to retrieve service profiles and service providers' SLAs. Service profiles can be either supplied by service providers' themselves or automatically detected by runtime profiling and VM inspection. The former approach is simpler since service providers will manually specify the attributes needed by the management infrastructure while the latter one requires active monitoring and introduces additional overhead.

When a better placement exists, the DC Reconfiguration Layer takes care of elaborating a suitable VM migration plan. Given both initial and target DC configurations, this layer is in charge of finding the minimum number of VM migrations to bring the DC to the target configuration. All these VM migrations have to be arranged in a proper plan that will detail temporal dependencies between them; in brief, whenever possible this layer should try to execute more VM migrations in parallel to reduce the total reconfiguration time. Moreover, it is worth recalling that VM live migrations introduce both high computational overhead and high bandwidth consumption [4]. Hence, this layer has to exploit DC status to understand which migrations are possible and their possible consequences on running services. For instance, some migrations might be given higher priorities than others to free host resources required to accommodate incoming VMs; at the same time, multiple migrations can be executed in parallel when they do not lead to SLA violations.

To ground our design guidelines to a real Cloud management system implementation, let us conclude this section by presenting some technical insights and seminal experimental results assessing the VM consolidation process in different service scenarios. In our previous work [34], we focused on the evaluation of

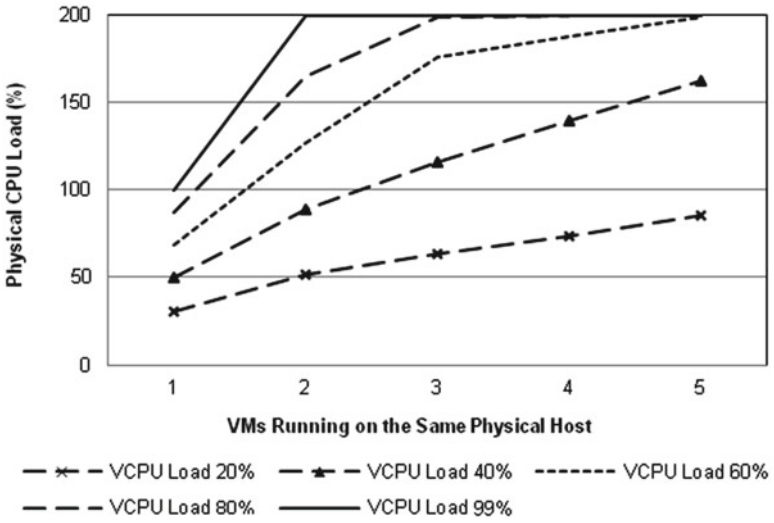


Fig. 7.5 Interferences between co-located VMs under CPU-bound services

interferences between co-located VMs under different types of workloads; in particular, according to the taxonomy of Fig. 7.1, we considered *CPU-bound* and *network-bound services*. Here, we want to remark that VM consolidation is a very complex task since co-located VMs interfere among them in many different, and hard to predict, ways; that justifies the adoption of service awareness to better drive the reconfiguration process. For the sake of brevity, we anticipate that all the results showed in this section are collected in a test bed made by two physical servers, each one with CPU Intel Core Duo E7600 @ 3.06 Ghz, 4 GB RAM, and 250 GB HD; also, we employed KVM as hypervisor, and all VMs have 1 VCPU and 512 MB RAM.

In the first set of experiments, we considered VMs executing CPU-bound services that introduce a VCPU load in {20, 40, 60, 80, and 99 %}, as shown in Fig. 7.5. We considered different consolidation ratios, spanning from 1 to 5 VMs, on the same host; since the physical host of these experiments has a dual-core CPU, the aggregate CPU consumption can reach 200 %. Moving to higher consolidation ratios, we experience performance degradation when the aggregated VCPU consumption is higher than available CPU; however, as reported in Fig. 7.5, we are always able to exploit all available resources, by reaching 200 % CPU utilization, meaning that the two CPU cores are fully loaded. Moreover, our experiments demonstrate (not reported here for the sake of brevity) that VMs get a fair share of available resources, and the performance degradation is easily predictable by considering the overcommit ratio of CPU resources.

In our second set of experiments, instead, we considered VMs running an Apache Web (AB) server that supplies static HTML pages as an example of network-bound service. In this case, we exploited the standard AB tool to load the Web servers [35];

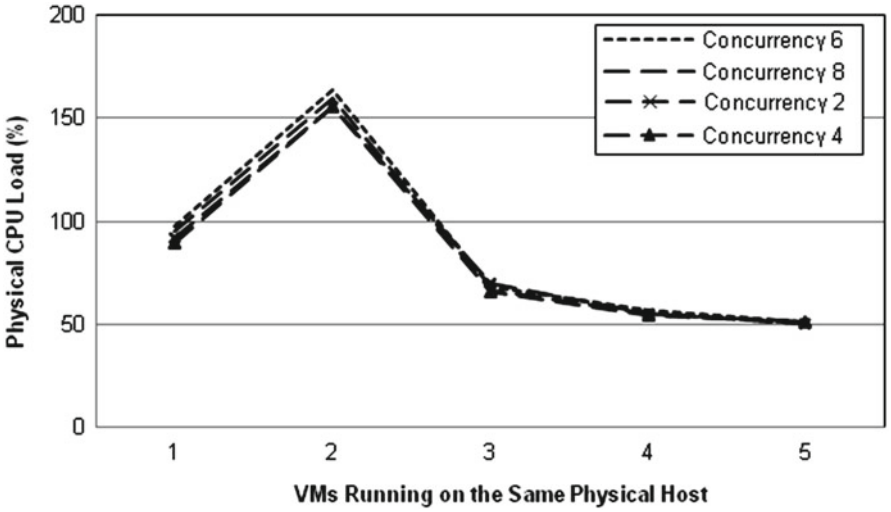


Fig. 7.6 Interferences between co-located VMs under network-bound services

different AB clients, one for each VM, emit 100,000 requests with a number of concurrent requests in {2, 4, 6, 8}. By analyzing the experimental results reported in Fig. 7.6, we can see that, with consolidation ratios higher than 2, the hypervisor is not able to grant all the available resources. We think that behavior is probably due to the fact that different VMs try to concurrently access the same network interface, thus leading to additional synchronization bottlenecks that prevent the full exploitation of all the available CPU resources.

In conclusion, as stated before, the Power-Efficient Placement Computation Layer should exploit service awareness to better evaluate possible side effects and interferences between co-located VMs. In addition, SLA awareness is important to prevent service violations, but it must be considered that SLA degradation for network-bound services is not always easily predictable as in the case of CPU-bound services.

7.6 Conclusion

Power-efficient management infrastructures for Cloud computing are gaining increased attention in the last years due to both environmental and economical issues. Modern DCs can introduce power consumption in the order of MWatts; hence, they require intelligent mechanisms to selectively switch off not required hosts and network elements, so to increase final power efficiency. Power-efficient Cloud infrastructure management has two positive effects: first, it reduces the environmental impact of the DC; second, it increases the economical revenue of the Cloud provider due to lower DC operational costs. At the same time, by using two

significant case studies, i.e., CPU-bound and network-bound services, we have shown that VM performance degradation due to VM consolidation cannot be easily predicted and depends on many factors, including virtualization support and type of services executed in the VM. Although different works are currently available in this research area, we think that additional research is required to handle both host and networking aspects at the same time.

First of all, since hosts represent the most important source of DC power consumption, it is fundamental to efficiently place the current workload; however, according to some studies [36], networking can account for the 5–10 % of the DC power consumption, and that justifies the usage of software-defined network techniques to further reduce total DC power consumption. At the same time, since both service and SLA awareness can be very useful to quickly identify and prioritize feasible reconfiguration decisions (as shown by our preliminarily experimental results), we think that additional work is needed also to better exploit them in order to dominate the complexity of such placement problems. Finally, it is important to devise new power optimization formulations to consider in the decision process not only single VMs but also groups of multiple VMs with high and correlated traffic, such as in the case of VMs collaborating to the same data processing task.

References

1. Lenk, A.: What's inside the Cloud? An architectural map of the Cloud landscape. In: Proceedings of ICSE Workshop on Software Engineering Challenges of Cloud Computing. IEEE Computer Society, pp. 23–31 (2009)
2. Armbrust, M. et al.: Above the Clouds: A Berkeley view of Cloud Computing. EECS Department, University of California, Berkeley (2009)
3. Buyya, R., et al.: Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Futur. Gener. Comput. Syst. **25**, 599–616 (2009)
4. Clark, C. et al.: Live migration of virtual machines. In: Proceedings of the 2nd Symposium on Networked Systems Design & Implementation, 2005
5. Le, K. et al.: Reducing electricity cost through virtual machine placement in high performance computing clouds. In: Proceedings of the Conference on High Performance Computing, Networking, Storage and Analysis (SC '11), 2011
6. Pettey, C.: Gartner estimates ICT industry accounts for 2 percent of global CO_2 emissions. http://www.gartner.com/it/page.jsp?id=503867 (2007). Accessed 6 Oct 2012
7. Murugesan, S.: Harnessing green IT: principles and practices. IEEE IT Prof. **10**, 24–33 (2008)
8. Baliga, J.: Green Cloud Computing: balancing energy in processing, storage, and transport. IEEE J. **99**, 149–167 (2011)
9. Beloglazov, A. et al.: A taxonomy and survey of energy-efficient data centers and Cloud Computing systems. In: Proceedings of CoRR, 2010
10. Heller, B. et al.: ElasticTree: saving energy in data center networks. In: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI'10), 2010
11. Biran, O. et al.: A stable network-aware VM placement for Cloud Systems. In: Proceedings of the IEEE CCGrid'12, Ottawa, 2012
12. Meng, X. et al.: Improving the scalability of data center networks with traffic-aware virtual machine placement. In: Proceedings of the 29th Conference on Information Communications (INFOCOM'10), 2010
13. Dormando: Memcached. http://memcached.org/ (2012). Accessed 6 Oct 2012

14. Elson, J., Howell, J.: Handling flash crowds from your garage. In: Proceedings of the USENIX 2008 Annual Technical Conference on Annual Technical Conference (ATC'08). USENIX, Berkeley (2008)
15. Voorsluys, W. et al.: Cost of virtual machine live migration in Clouds: a performance evaluation. Proceedings of the 1st International Conference on Cloud Computing (CloudCom'09), 2009
16. Breitgand, D. et al.: Cost-aware live migration of services in the Cloud. In: Proceedings of the 3rd Annual Haifa Experimental Systems Conference (SYSTOR '10), 2010
17. Kayacan, E.: Grey system theory-based models in time series prediction. Expert Syst. Appl. **37**, 1784–1789 (2010)
18. Isci, C. et al.: Runtime demand estimation for effective dynamic resource management. In: Proceedings of the IEEE Network Operations and Management Symposium (NOMS), 2010
19. Al-Fares, M. et al.: A scalable, commodity data center network architecture. In: Proceedings of the ACM SIGCOMM 2008 Conference on Data communication (SIGCOMM '08), 2008
20. Greenberg, A. et al.: VL2: a scalable and flexible data center network. In: Proceedings of the ACM SIGCOMM 2009 Conference on Data communication (SIGCOMM '09), 2009
21. Guo, C. et al.: BCube: a high performance, server-centric network architecture for modular data centers. In: Proceedings of SIGCOMM Computer Communication. ACM, pp. 63–74 (2009)
22. Verma, A. et al.: pMapper: power and migration cost aware application placement in virtualized systems. In: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware (Middleware '08), 2008
23. Kim, K.H. et al.: Power-aware provisioning of Cloud resources for real-time services, Proc. 7th International Workshop on Middleware for Grids, Clouds and e-Science (MGC 2009), 2009
24. Abdelsalam, H.S. et al.: Analysis of energy efficiency in clouds. In: Proceedings, Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009
25. Beloglazov, A., Buyya, R.: Energy efficient resource management in virtualized Cloud Data Centers. In: Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010
26. Buyya, R. et al.: Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: challenges and opportunities. In: Proceedings of the 7th High Performance Computing and Simulation Conference (HPCS'09), 2009
27. Van, H.N. et al.: Performance and power management for Cloud infrastructures. In: Proceedings of the IEEE 3rd International Conference on Cloud Computing (CLOUD'10), 2010
28. Jung, G. et al.: Mistral: dynamically managing power, performance, and adaptation cost in Cloud infrastructures. In: Proceedings of the IEEE 30th International Conference on Distributed Computing Systems (ICDCS'10), 2010
29. Wang, Y., Wang, X.: Power optimization with performance assurance for multi-tier applications in virtualized data centers. In: Proceedings of the 39th International Conference on Parallel Processing Workshops (ICPPW '10), 2010
30. Dupont, C. et al.: An energy aware framework for virtual machine placement in cloud federated data centres. In: Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy '12), 2012
31. Mann, V. et al.: VMFlow: leveraging VM mobility to reduce network power costs in data centers. In: Proceedings of the IFIP Networking, 2011
32. Korupolu, M. et al.: Coupled placement in modern data centers. In: Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2009
33. Wang, M. et al.: Consolidating virtual machines with dynamic bandwidth demand in data centers. In: Proceedings of the IEEE INFOCOM 2011 Mini-Conference, 2011
34. Corradi, A. et al.: VM consolidation: a real case based on OpenStack Cloud. Future Gener. Comput. Syst., Elsevier Science, SI on Management of Cloud Systems, Available online Jun. 2012, doi:10.1016/j.future.2012.05.012. pp. 1–10 (2012)
35. Kvitka, C.: ab – Apache HTTP server benchmarking tool. http://httpd.apache.org/docs/2.0/programs/ab.html. Accessed 8 Apr 2013
36. Hamilton, J.: Energy proportional datacenter networks. http://perspectives.mvdirona.com/2010/08/01/EnergyProportionalDatacenterNetworks.aspx (2010). Accessed 6 Oct 2012

Part III
Cloud Perspectives and Patterns

Chapter 8
Cloud Computing: A Mobile Context-Awareness Perspective

Nayyab Zia Naqvi, Davy Preuveneers, and Yolande Berbers

Abstract Mobile devices and their applications take advantage of cloud computing not only to overcome their limitations in computing, memory and storage resources but also to easily scale up to a large number of users. Many common mobile applications we use every day, such as e-mail, news, social networking and games, already rely on the cloud paradigm due to these reasons. Another trend of the last decade is the evolution towards smart applications that take into account the user's context, such as their whereabouts and ongoing activities, to adapt the behaviour of the applications according to the current situations. With context management and inferences becoming fairly complex processes, cloud services can offer an edge by taking over the most computationally intensive context management tasks and have these tasks carried out only once for multiple users. However, as a lot of this contextual information is being sensed and captured by the mobile devices themselves, it is not always clear whether it makes sense to upload all the raw sensed data to the cloud and have it completely processed there. These trade-offs depend on the amount of data being communicated and the kind of processing that is required. In this chapter, we analyse the challenges to develop context-aware cloud services for mobile applications and present a conceptual architecture for context processing and service provisioning in the Platform-as-a-Service layer. We highlight various architectural viewpoints of the presented federated context-processing middleware and round up this chapter with a research road map for the future.

Keywords Context-aware perspective • Mobile computing • Social networking • Context management

N.Z. Naqvi (✉) • D. Preuveneers • Y. Berbers
iMinds-DistriNet, Department of Computer Science, KU Leuven, 3001 Leuven, Belgium
e-mail: nayyab.naqvi@cs.kuleuven.be; davy.preuveneers@cs.kuleuven.be;
yolande.berbers@cs.kuleuven.be

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_8,

8.1 Introduction

The proliferation of smart applications is giving rise to a rapidly evolving subject area known as mobile cloud computing. Advances in mobility and wireless communications have increased the use of smart mobile applications. Many of the smart applications are now looking at the cloud-computing paradigm as an enabler to exploit its raw computing power, memory and storage resources to overcome the resource limitations of mobile devices [1]. As such, cloud services have not only changed the way in which we run these applications but also how we develop them. Single monolithic applications for the mobile have now evolved towards a distributed loosely coupled composition of services where some of the service components are still running on the mobile but where others are running in the cloud. Such an approach requires an upfront investment in the methodology of how to develop mobile cloud applications as it imposes new integration and interoperability requirements.

Furthermore, as many of the applications try to become smarter, they aim to understand and anticipate the situation of their users by alerting them in time, or adapting their behaviour according to the new context, and this without any intervention of their users [2]. As such, context management becomes an integral part of the mobile cloud applications, with some of these tasks running on the mobile (e.g. context sensing and simple processing) and other computationally more expensive context management tasks being outsourced and taking place remotely on the cloud infrastructure (e.g. pattern detection, reasoning and learning). For some tasks, the decision of what to do where is not always so clear-cut. This decision depends on various overhead trade-offs, including the quality of service (QoS) and the quality of experience (QoE). These trade-offs involve computation, communication, storage, energy consumption, availability, connectivity, as well as user preferences with respect to privacy and security for sensitive data.

The mobile cloud-computing paradigm from a context-aware perspective can be regarded as a research track that aims to find effective ways to make cloud services, being utilized in mobile applications, aware of the context of their customers and applications. To ease the development of context-aware applications, it is paramount to decouple the application logic and its adaptation logic from context acquisition and inference, and at the same time, it is equally important to provide efficient and effective ways to utilize the acquired context. That is why writing scalable and robust smart applications for mobile devices in the cloud environment not only involves being mindful of the architecture of both the cloud services and the management of the context information [3] but also about the communication and coordination between all the parties involved that drive the smart adaptive behaviour of the application.

In this chapter, we provide an overview of how context awareness aims to make mobile applications smart and discuss challenges of how to effectively utilize the cloud for this particular type of applications. We also discuss architectural blueprints for the design of a cloud-based context-processing middleware that can

adapt dynamically and off-load its components to the cloud for smart mobile applications to save resources of the mobile devices. The chapter has been organized into seven sections. In Sect. 8.2, we introduce the notion of context and some definitions and the role of context awareness in mobile applications. We depict various major building blocks for context-processing frameworks and discuss various well-known systems found in the literature. In Sect. 8.3, we further explore the role of the cloud and discuss the main challenges of outsourcing context management to the cloud in an efficient and effective manner. Architectural considerations for the context-aware applications are investigated in Sect. 8.4. The next section discusses the implementation flow of the identified services and an evaluation strategy, while in Sect. 8.6 we present some guidelines and a research road map for the future ending with a conclusion focusing on the significance of leveraging the cloud-computing paradigm for context-aware smart applications in Sect. 8.7.

8.2 Overview of Context-Aware Mobile Applications and Architectural Trends

In this section, we discuss the role of context awareness for intelligent behaviour of mobile applications and highlight the existing context-processing frameworks and their architectures.

8.2.1 Defining Context and the Role of Context Awareness

Context awareness is an area of research where applications aim to take into account information about their user and his situation and change the behaviour of the applications accordingly. Context awareness is an intrinsic characteristic of intelligent environments [4–6]. Context was first introduced by Schilit and Theimer [7] in 1994, and its definition has evolved ever since. It was initially referred to as location, identities of nearby people, objects and changes to these objects. Chen et al. [8] proposed a definition of the context as

> Context is the set of environmental states and settings that either determines an application's behaviour or in which an application event occurs and is interesting to the user.

Dey [4] elaborated in his most cited definition for context on the interaction of an application with its user, his environment and situation of the resources involved:

> Context is any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects.

Chaari [9] defined context in terms of dynamic application behaviour on the basis of user's varying context as follows:

> Context is a set of the external parameters that can influence the behaviour of an application by defining new views on its data and its available services. These parameters may be dynamic and may change during the execution.

There are many more definitions for context, and it is impossible to list them all. However, it is clear that context information is dependent on individual users, systems and circumstances, as one piece of information might be considered as context information in one scenario but not in another one. It all depends on whether the information is relevant for the application to make well-informed autonomous decisions or manifest autonomous behaviour. In this chapter, we will mainly consider all context information that can be used to improve the behaviour of the mobile application services in a particular situation while at the same time optimize the context management itself by leveraging cloud-computing capabilities.

8.2.2 Context Awareness in Mobile Applications

In mobile applications, context awareness can be regarded as the dynamic adaptation of these applications according to the continuously changing context of the user, the communication network, the physical environment and the device itself [10, 11]. For instance, if a user wants to download and play a video in his mobile device, a context-aware video playing application on the mobile device may automatically select the appropriate variant according to its available bandwidth, memory and battery power and display size of the device and if necessary off-load the transcoding of the original video to this variant to the cloud. This way, the mobile device gets a suitable video stream with respect to its own capabilities and resource availability.

Context awareness requires an effective approach to cater for getting the right information at the right time to make the correct decisions for changing or optimizing its behaviour. Context awareness usually involves three basic steps [7]:

- *Discovery*: This involves the identification of entities that are relevant for the application's tasks. In the first step, a context-aware application has to discover and explore its environment in order to get information to work with. In the above example, it not only means contextual information about resources on its own device but also about other networks, computational facilities and displays in the neighbourhood.
- *Selection*: A context-aware application has to filter the information that was discovered according to its specific needs. Information about displays that are located on the floor below is irrelevant. This selection process can be fairly complex as it may require complex filtering techniques to decide which sensor or device is offering relevant information.

- *Use*: The application uses the discovered and selected information to make the right decision or change its configuration or behaviour. We have identified several high-level ways of how context can be utilized:
  - Contextual information display: The context can be presented to the user. A simple example is finding the closest Mexican restaurant to your current location.
  - Contextual augmentation: Augmentation [10] will associate the metadata about the captured context, for example, showing on a map all the shops that sell a particular item, with nearby parking facilities and that will still be open for at least half an hour by the time you reach the shop.
  - Contextual mediation: This use case focuses on getting relevant context information to the entities that need it (but they might not be able to sense it themselves) [12, 13]. For example, remote applications might optimize their services or functionality to best meet the needs and limits arising from the local and evolving context of interaction.
  - Context-aware configuration: This use of context defines the rules for the actions to be performed in a given context, e.g. choosing a nearby printer or relocating the display to another screen.
  - Context-triggered actions: This context-aware behaviour will initiate certain steps when certain contextual conditions are met, e.g. turning on the silent mode of the mobile and hiding the notifications if the user is in a meeting.

In the following section, we discuss various frameworks and systems that have been used for the development prototype context-aware applications.

8.2.3 *Frameworks and Architectures for Context-Aware Applications*

Baldauf [14] discusses in his survey a number of context-processing systems and their architectures, including the Context-Awareness Sub-Structure (CASS) [15], Service-Oriented Context-Aware Middleware (SOCAM) [16] based on RMI, Context Broker Architecture (CoBrA) [17], Hydrogen [18] and the Context Toolkit [19], with the latter probably being the most famous one. Every system has its own processing approach, ways to describe the relevant context information in a specific scenario, as well as its own context communication and sharing protocols.

Some of the above context-processing systems are network based, while others have their components tightly coupled. However, we can identify the basic context-processing tasks to achieve context awareness, as follows:

- Context sensing and fetching
- Context filtering and aggregation
- Context storage
- Context reasoning
- Context use

In general, the architectural design for context-aware applications took off with a completely centralized approach dealing with the context of a single user and a single application, with the limited context management capabilities completely embedded in the application. Later on, context management was abstracted from the application logic with a dedicated context-aware processing server, allowing better reuse when running multiple local applications atop. Gradually, context management evolved into a distributed architecture, where context-aware applications leveraged the same remote centralized context-processing server, allowing for multiple remote users to consume personalized services.

However, there has been relatively little advancement in context-aware computing over the past 5 years on how to make context management itself more effective and efficient through distribution of the tasks involved and the overall workload. The work presented in this chapter attempts to enable a new phase of context-aware mobile application development using the cloud-computing paradigm for better scalability and reusability of context management by implementing loosely coupled context-supporting components to achieve the aforementioned context-processing tasks.

8.2.4 A Service-Oriented Approach for Context-Aware Mobile Applications

As discussed in the previous section, a number of architectures supporting context-aware applications have been developed, but little attention is paid to the special requirements and constraints of mobile devices. Many of the state-of-the-art centralized deployment models, application-specific communication protocols and context data representations are not easily applicable in a cloud environment, where sharing of the context between interlinked services raises new challenges of context data representation, storage, processing and distribution in a service-oriented way. Hong et al. [20] have highlighted the advantages and the need of a service-oriented infrastructure abstraction for robust context-aware systems.

A cloud-computing paradigm is an ecosystem built on top of web services [21]. In order to manage its processing and composition in a flexible manner, the supporting components of context processing need to be provisioned as loosely coupled web services. This realization in the form of web services requires representation and aggregation of context information in a structured and standard format. In current context-aware systems, XML-based languages are widely used for modelling and representing the context information. The Pervasive Profile Description Language (PPDL) is an XML-based language to represent situational context [22]. There are XML schemas to define contextual information of mobile networks, i.e. device status and reachability. Resource Description Framework (RDF) and Web Ontology Language (OWL) are also widely accepted to represent the context. Composite Capability/Preferences Profile (CC/PP) can be used for capabilities description and user preferences. CC/PP [23] is based on RDF. CC/PP is a widely

accepted W3C standard, but it does not specify how contextual information can be stored. Strang and Linnhoff-Popien [24] present a survey of six context modelling approaches: key-value modelling, markup scheme modelling, graphical modelling (UML, ER, etc.), object-oriented modelling, logic-based modelling and ontology-based modelling. According to their analysis that is based on the appropriateness requirements presented in the same chapter, they found that ontology-based modelling is the most promising approach for context modelling in pervasive computing environments.

Troung and Dustdar [25] shed light on context-processing systems designed especially for web services. The Akogrimo project [26] aims at supporting mobile users to access context-aware data on the grid. It focuses on the user's presence, his location and environmental information. It has a context manager that collects contextual information and delivers it to the application. Han et al. [27] present the Anyserver platform, which supports context awareness in mobile web services. This platform utilizes device information, networks and application type as context information, but it does not support context sharing. CoWSAMI [28] also provides a context manager to manage context sources. The ESCAPE [29] framework is a web-based context management system for teamwork and disaster management. It provides web services-based context sensing and sharing front-end as mobile device interface and provides a back-end service for storing and sharing context information. Omnipresent [30] is a location-based service system.

There are currently no context-processing systems aimed at context awareness specifically for mobile applications based on cloud services [25]. These systems aim at few attributes of the context and supporting components of the context processing. Mobility, context storage and context reasoning are not catered in most of the aforementioned systems.

Context awareness is an enabler for service adaptation. Service adaptation is also a complex process, which involves the rapid analysis of the varying context data and the customization of the appropriate services. There are many researchers who presented service adaptation approaches on the basis of the context. Maamar et al. [31] present an approach for service selection and task adaptation for context-based web service composition. They present a CC/PP-aware client proxy, context manager and policies for the personalization of web services dynamically on the basis of the resource context. The policies are used for consistency, feasibility and inspection. In mobile web services, content adaptation is also a common practice. Context information is used to adapt the content for a client request.

Seyler and Taconet [32] present a service orchestration adaptation technique by taking into account the service composition meta-model and context meta-model. Based on these meta-models, they present both deployment time and runtime adaptations. Soukkarieh and his colleague [33] present an architecture aiming at adapting content and presentation of services to the user's context. Their context management module is not asynchronous. User updates its context requirements and then context management updates itself.

Badidi and Esmahi [34] discuss a Software-as-a-Service (SaaS) approach for context information provisioning in the cloud-computing paradigm. They

propose a framework for context information provisioning, which relies on deploying context-supporting components in the form of services on the cloud and using context brokers to mediate between context consumers and context services using a publish/subscribe (pub/sub) model. This work is aligned with our approach in a way that we also focus on loosely coupled context-supporting components hosted in the cloud for general-purpose context-aware mobile applications.

8.3 Challenges of Shifting Context Awareness to the Cloud

This section highlights the challenges of developing and running context-aware mobile applications in the cloud, as well as challenges to utilize context awareness in a federated cloud environment. We have distilled these challenges from analysing various context-aware mobile applications [35–37] and what the effects would be for a (partial) deployment in the cloud.

8.3.1 Resource Limitations and Trade-Offs for Advanced Context Processing on Mobile Devices

Mobile cloud computing [38] is a model for transparent elastic augmentation of mobile device capabilities via ubiquitous wireless access to the cloud storage and computing resources. The adjustment of context-aware dynamic off-loading is done with respect to changing operating conditions while preserving the available sensing and interactivity capabilities of the mobile devices. Mobile devices have built-in sensors to sense the real-time situation of the users. As such, they are important for fetching the context data, but the size of the acquired context data varies, depending on the application's objectives. This amount of data can be significant, e.g. the triaxial accelerometer of smartphone used for activity recognition such as step counting or fall detection can generate 100 samples per second, which would produce about 200 kB per minute (one value of type double for each x, y, z axis and a time stamp, producing 32 bytes per sample). Sending all this data to the cloud for statistical feature extraction and selection may be too inconvenient.

Baldauf [14] stressed the importance of the method of context data acquisition when designing context-aware applications because it predefines the architectural style of the application. Most research into context-aware mobile applications has considered the use of various sensors, including position sensors, accelerometers, gyroscopes and cameras. Often, it is not the position itself that constitutes immediately useful context but additional information that can be inferred from the location, for instance, a user in a meeting room implies that he is in a meeting. Accelerometers and gyroscopes provide movement and orientation information. Cameras similarly provide access to potentially rich information that can be derived by means of feature extraction and video analysis for surveillance and assisted living scenarios.

While positional context, motion context and visual context are powerful for augmentation of devices with some awareness of their situation, they also have distinct shortcomings. Position is a static description of an environment and does not capture dynamic aspects of a situation. Its usefulness as context also largely depends on pre-captured knowledge about locations. Motion and vision, on the other hand, can be employed to capture activity and other dynamic aspects, but extraction of a specific context is computationally expensive and problematic in mobile and uncontrolled environments due to the shortage of resources for computation, data storage, network bandwidth and battery capacity. In these cases, context storage and context reasoning are often too impractical to be put on a mobile device. Applications like face detection in media or live gaming [39] can be impossible to run efficiently on a mobile device as it is extremely difficult to manage high-volume data or advanced computational capabilities due to limited resources. Some of the algorithms used for context-aware applications, such as speech recognition or image processing, also require intensive computation and produce large amount of data. Further, it is unreasonable to keep specific types of data on personalized devices, e.g. book ISBN numbers or postcodes of a county. This data is too large and storing it on portable devices is not feasible.

Contrary to mobile devices, cloud computing provides plentiful storage and processing capabilities. Mobile applications are being built on web standards where computation is offloaded to the powerful cloud resources. However, sometimes (such as in the step counting or fall detection cases using the accelerometer on the smartphone) processing mobile data in place on the device would be more efficient and less susceptible to network limitations, compared to off-loading data and processing to the remote cloud.

8.3.2 Connectivity, Latency and Bandwidth

When consuming context-sensitive cloud services on the mobile device, a continuous data communication is required. In a mobile environment, the network connectivity varies from high bandwidth to low bandwidth according to the available network connection. Furthermore, data integration with remote processing requires the local data to be moved to a remote location with the upload speed being highly connection oriented. Hence, even if the processing in the cloud is far more efficient compared to the mobile, the latency triggered by uploading the data can be a challenge to the context-aware cloud services running in the mobile [40], especially if this latency is subject to variations in bandwidth.

Furthermore, what if the connection is lost? A context-aware mobile application completely running in the cloud would stop working if the mobile device loses its connection. Fortunately, new programming languages such as HTML 5 enable data caching locally on the mobile device [41], so the interruptions in connectivity will not affect the ability to continue its operations. However, if the application relies on the cloud for all the context management tasks, the support for its smart behaviour is no longer available.

8.3.3 Security and Privacy

Storing and using the context data, for adapting cloud services, provided by third-party service providers, is a frequently discussed security issue in cloud computing [42]. Clearly, the infrastructure needs to be secure against unauthorized access, but it also needs ways to let people introspect, so that they can understand how their context data is being used. Furthermore, the infrastructure needs to be designed such that privacy concerns are legitimately and adequately addressed. Storing the sensitive information on the client device but allowing preprocessed and encrypted context to be sent to the cloud can address these challenges. But this procedure can be expensive for the mobile device in terms of computation and communication. Imagine a user aims to off-load the entire context processing to the cloud to reduce the computational overhead on his device. In such a scenario, enforcing encryption could really sacrifice any resource gains on the mobile.

8.3.4 Apportioning Decision for Federated Deployments

The biggest challenge for federated deployment of distributed context-aware mobile applications in the cloud is to decide when and what to off-load in an autonomic manner [40]. The context-aware middleware has to take this decision of how to split responsibilities between devices, applications and the middleware itself. It should be able to decide for which use cases the cloud is better and for which ones a cloud-based deployment does not bring any added value. For instance, Hinckley et al. [43] modified a PDA to have tilt, touch and proximity sensors. The screen would rotate simply by rotating the PDA. Also, the voice recorder would automatically activate if the PDA was tilted upwards, was being touched and was near something (ideally the user's face). All of this sensor data was processed locally on the device itself. However, if it were processed in a remote context middleware running in the cloud [42], it is likely that the interactivity would be affected due to network latencies. The essence of the problem is finding the middle ground, determining what the mobile devices and applications should handle and what the cloud should handle.

8.3.5 Managing the Context-Aware Service Life Cycle

Platform-as-a-Service (PaaS) aims to abstract the complexity of managing the life cycle of the cloud services by providing targeted tools and services. The state-of-the-art PaaS offers common services like authentication, authorization, billing and provisioning for multiple Software-as-a-Service (SaaS) applications. Another major challenge for context-aware cloud service providers is to exploit the benefits of cloud computing to manage quality of service (QoS) commitments to customers throughout the life cycle of a service. Context plays a vital role in a service life cycle

from service request to service invocation [44], its deployment or customization according to the user's situation and ongoing activities. Context-aware service provisioning and adaptation is technically a challenging job due to continuous varying context and heterogeneous kind of data.

8.4 Architectural Aspects for Context-Aware Mobile Applications in the Cloud

This section presents the architectural considerations and requirements to develop context-aware cloud services. A blueprint of a federated middleware in Platform-as-a-Service layer has been presented with a service-provisioning module to better manage the federation of mobile and cloud platform.

8.4.1 Requirements for Context Processing Within PaaS

The architecture for context-aware mobile applications is evolving to an enormous scale. It is evolving to a many-to-many fashion, where a large number of users can utilize the distributed context-processing middleware asynchronously and this for a great number of personalized services hosted by third-party service providers. Let us exemplify the use of context-based mobile cloud services by a case study of Dr. Peeters and his interaction with the technology:

> Dr. Peeters utilizes his mobile device to watch the morning news redirected on a nearby big screen or only to listen to it by text to speech conversion service, according to the available bandwidth of wireless connection. As he gets into his car to go to work, his smart device is re-configured to low bandwidth wireless connectivity and his desired information is pre-fetched or cached with the help of a context service in the cloud. While travelling, he is interested to see the information of patients who are in his appointment list today. If he has lectures or meetings, he would like to get notified. His device can offer him a text to speech conversion for his agenda. He can stream his online music repository to his mobile device as well. He is not worried on being late as he knows traffic is smooth as otherwise his intelligent mobile device would have notified him with an alternative route. During the day, Dr. Peeters manages multiple patients' cases simultaneously on the basis of their emergency of which he gets informed through push notifications. Other cloud services enable him to collaborate with his peers effectively and to participate in discussions remotely.

In the above-mentioned use case scenario, we cannot neglect the role of context information, as it is vital to make all the services work in harmony and adapt dynamically to achieve runtime intelligent behaviour of the services in a scalable way. The enabling factors for these interconnected mobile services are context-aware intelligence, interoperability, standardized communication protocols and platforms to deploy and efficiently adapt these services. Any software architecture designed for context-aware mobile applications, federated with the cloud services, will need to foresee the increasing heterogeneity of context-capturing devices. It will also have to provide service provisioning for varying context.

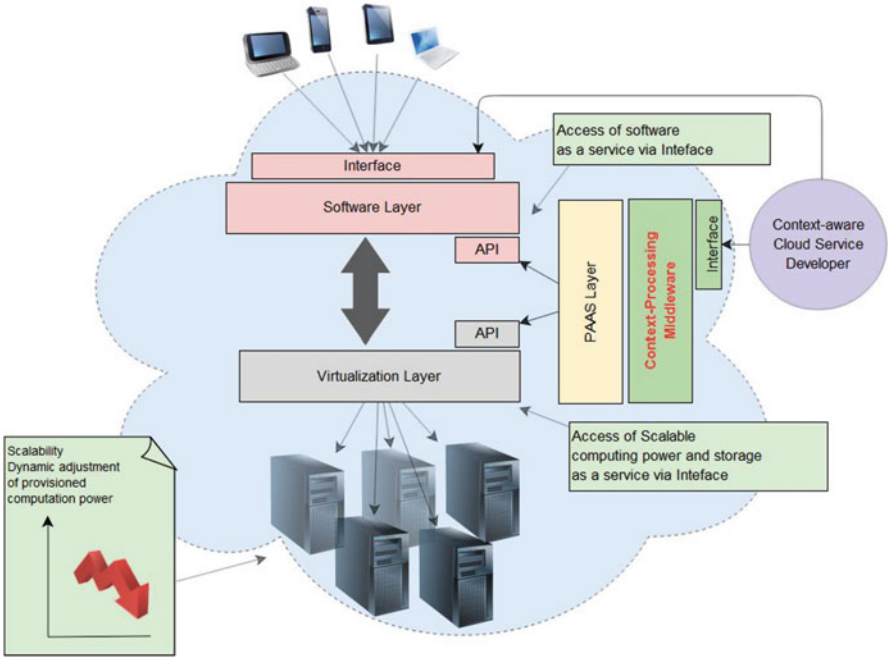


Fig. 8.1 Context processing in the cloud-computing model

In our opinion, there is a strong need of a web service-based context-awareness abstraction as a part of the PaaS layer in the cloud. Figure 8.1 shows how context awareness affects the cloud-computing model and interaction of a context-aware cloud service developer with the layers of cloud-computing paradigm. A context-processing middleware is shown as an abstracted layer within Platform-as-a-Service to develop context-aware cloud services and their dynamic adaptation. Platform-as-a-Service is already providing service provision for scalability of cloud services. The context-processing middleware can utilize the same service-provisioning infrastructure with the addition of a few more services to manage and adapt services on the basis of the varying context.

It would ease the context-aware application development, the context processing and service provisioning at runtime. The biggest benefit of such a type of middleware is that the sensors, services and devices can both be changed independently, even dynamically when other sensors, services, devices and applications are running. The encapsulated context processing increases the reusability and eases the development of large-scale, smart mobile applications.

We summarize below the architectural requirements for a context-processing middleware, as a part of the Platform-as-a-Service layer:

- General-purpose cloud services must not be tightly coupled to the context-processing middleware. It allows the context management to evolve as new sources of information become available or existing sources disappear.

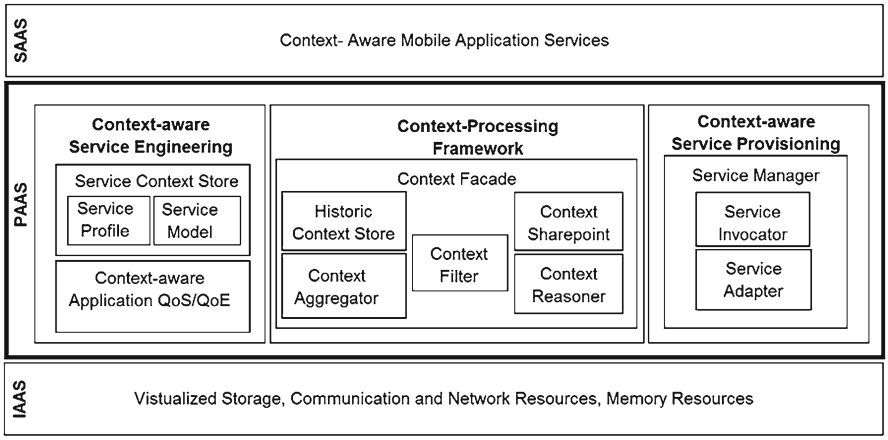


Fig. 8.2 Architecture of the context-processing middleware and context-aware service provisioning in the PaaS

- A separate abstraction layer of a service manager is required to match the context of the user and the services that the user has subscribe to.
- The service manager requires advanced adaptation algorithms to adapt cloud services according to the evolving context of the user and his device.
- A trade-off analysis technique is required for the apportioning decision of how to dynamically infer what and when to off-load to the cloud without affecting the QoS and QoE.
- Users should be able to add their own preferences and requirements about which context information and applications should remain local on the mobile.
- The architecture needs to be open and extensible to integrate new context management components, while at the same time remain scalable and elastic to handle a growing number of users.

Figure 8.2 depicts the three major building blocks of the context-aware cloud services in the PaaS

8.4.2 Information Viewpoint of the Context-Processing Architecture

In context-aware applications, the context data is highly distributed and possibly coming from and being used anywhere, anytime. Smart applications based on context-aware cloud services differ from traditional applications because the information processing life cycle is highly distributed. It is important that the interpretation of the context information is uniform for every participating cloud service. We leveraged our context ontology [45] to describe the contextual concepts as well as

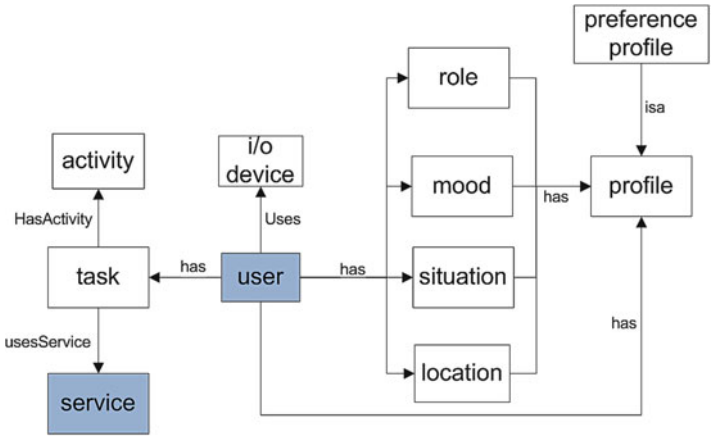


Fig. 8.3 User context ontology [46]

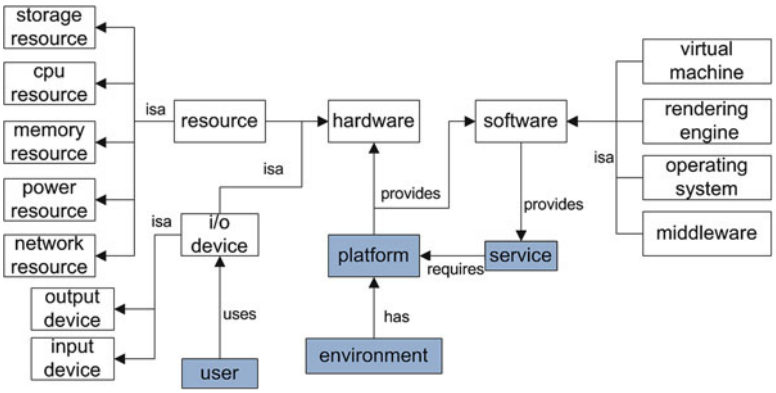


Fig. 8.4 Platform context ontology [46]

their interrelationships. The three major context concepts in our ontology [46] that we use in our architectural information viewpoint are the following.

1. *User context*: Any information related to user of the cloud service is regarded as user context. It can be location, activities, habits, preferences, noise level, light, temperature, etc. Figure 8.3 shows the ontology design for user's context.
2. *Platform context*: The information related to the mobile device that is significant for the execution of a service is classified as device context, e.g. battery level, available bandwidth and display resolution (Fig. 8.4).
3. *Service context*: Any information related to the service during its life cycle relevant for service adaptation, e.g. the functionality description, the interfaces and required resources. This type of information is essential for the service manager (see Fig. 8.2) to take the apportioning decision. Figure 8.5 shows the context of a service.

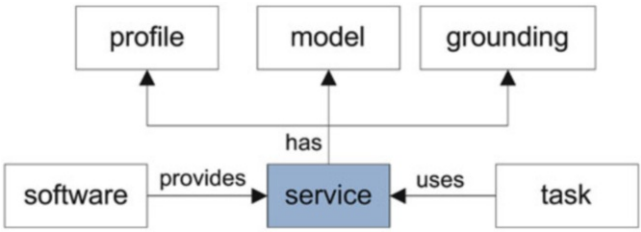


Fig. 8.5 Service context ontology [46]

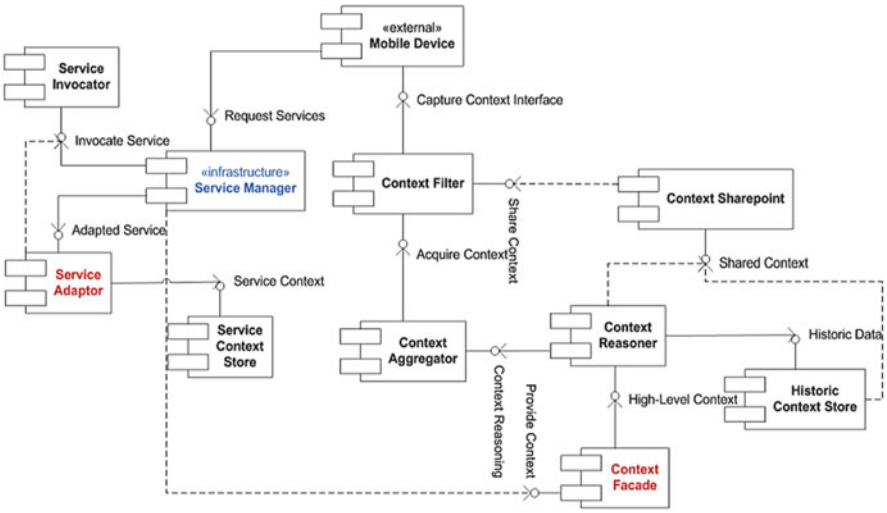


Fig. 8.6 Functional viewpoint of the architecture of context-processing middleware

8.4.3 Functional Viewpoint of the Context-Processing Architecture

Context awareness enables automatic service adaptation. It helps the system to adapt its behaviour according to desired objectives while offering interoperable and scalable personalized services. Figure 8.6 presents the functional viewpoint of the architectural design presented in Fig. 8.2 in the PaaS layer. This functional view of the architecture is designed to cater the functional requirements of a context-processing middleware. It defines the key functional elements, their responsibilities, the interfaces they expose and the interaction between them. In PaaS, there is already a mechanism to provision web services and on the fly creation of new services that do not disturb the existing fabric. However, we need a service manager for context-aware adaptation of the services. The service manager will take into account the context of users, devices and resources available to adapt to the best matching service as discussed in Sect. 8.4.2.

A mobile device requests a service to the *Service Manager* in the cloud. If the service is not context aware, the *Service Manager* invokes the service with the coordination of the *Service Invocator*. If the service is context aware, the *Service Manager* contacts the context-processing middleware component called *Context Façade*. *Context Façade* acts as an abstraction to other context-processing components. This service pushes the request of context information to the *Context Reasoner*. The context reasoning service asks the *Context Aggregator* and *Historic Context Store* for preprocessed and historic context data. It infers the requested context and provides it to the *Context Façade*. The *Context Sharepoint* service, to acquire shared context, is shown in this diagram as an optional service. The *Context Sharepoint* service contacts the *Context Filter* service to provide context of multiple users. Once the context of the user is determined by the *Context Façade*, the *Service Manager* accesses the context and contacts the *Service Adapter* to invoke the required service compatible to the current context. The *Service Adapter* is dependent on the *Service Context Store. Service Context Store* is a place where semantically defined services are published and each service has a valid context profile. The *Service Adapter* matches the service context profile with the user context and customizes the service accordingly. The *Service Manager* calls the *Service Adapter* service whenever the context changes.

8.5 Implementation and Evaluation

We have recognized a few specialized services on the basis of the use case scenario from Sect. 8.4.1. There are interlinked services running on a mobile device. We have distributed the context-awareness process in nine autonomous services. Figure 8.7 shows the high-level composition and interaction of these services.

We are using WSO2 Stratos [47], an open PaaS solution, for implementing our context-aware components as loosely coupled services and our empirical evaluation. We are in the process of evaluating the scalability of the context-processing middleware for a context-aware mobile healthcare application [48] using human activity recognition as a composition of several context-processing services. Although we have preliminary results of small-scale experiments, they do not provide any statistical significance for large-scale deployments with many users. For that, we need large context datasets and business-level ecosystems to evaluate different provisioning strategies in order to provide evidence regarding optimal strategies that improve customer QoE. Once our context-processing middleware is fully operational, we will also cross-validate the performance of our adaptation strategies by running the same instrumented applications on existing public cloud infrastructures (e.g. Amazon or Google App Engine).

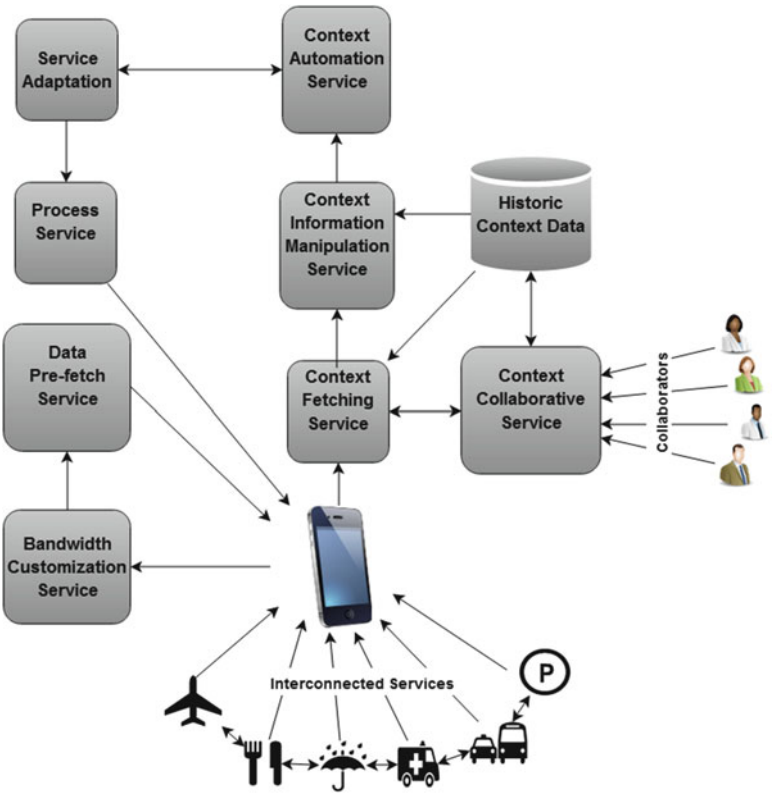


Fig. 8.7 Composition of the context-aware mobile services

8.6 A Research Road Map for Future Context-Aware Mobile Cloud Services

Cloud computing is a building block of the Future Internet, and context awareness is an integrated part of the Future Internet. This enables a common vision for the deployment of independent federated services and applications, characterized by a high degree of autonomous data capture, event transfer, network connectivity and interoperability. The added value is in context-aware mobile applications, such as environmental monitoring and healthcare monitoring [48], where the high volumes and rates of data need rapid processing to translate raw data into contextualized information. The context-aware self-adapting mobile applications will support the interaction between heterogeneous context-aware cloud services.

The context-processing and context-aware service provisioning in PaaS will give rise to the interactions among context-aware cloud services in order to be able to support heterogeneous sources of context data and many heterogeneous mobile devices through the use of standard interfaces and data models to ensure a high degree of interoperability among diverse context-aware applications.

From an architectural perspective, we need adoption of standards to promote interoperability and to allow each context-aware service to benefit from a competitive marketplace of interoperable technology solutions from multiple service providers. As greater trust is placed on the context-processing and service-provisioning middleware in PaaS, it will be essential to ensure that international quality and integrity standards are deployed and further developed. Beyond the need for standardization to ensure interoperability, this requires further research in the following areas: (1) ontology-based semantic standards for context-aware cloud services, (2) protocols for communication within and outside cloud environments, (3) efficient and advanced service management and adaptation algorithms and (4) ways to effectively apportion the context-processing services within mobile and cloud platforms. We strongly believe that these areas are part of the road map for future research towards mobile cloud computing with a context-awareness perspective.

8.7 Conclusion

Context-based cloud services and mobile connectivity will give rise to the new interaction with the cloud in which information and communication is embedded in the environment around us in the form of wireless networks. As a result of the extraordinary increase of modern mobile Internet-enabled devices and the pervasive wireless and cellular data networks, a large number of mobile users are requiring personalization services customized to their context. The next wave in the era of cloud computing will be outside the realm of the traditional cloud services and infrastructure. Most of the cloud services will be context aware.

In this chapter, we focused on the need of loosely coupled context-supporting components that work with a context-aware service-provisioning infrastructure to adapt services to the context of the user and his mobile device. We believe that the adoption of our PaaS-based context-processing middleware will allow future mobile cloud solutions to be developed quicker by abstracting all context management into reusable services, as well as accelerate research in the area of context-aware mobile applications for the cloud.

Acknowledgments This research is partially funded by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy and by the Research Fund KU Leuven.

References

1. Buyya, R., Chee Shin, Y., Venugopal, S.: Market-oriented cloud computing: vision, hype, and reality for delivering IT services as computing utilities. In: 10th IEEE International Conference on High Performance Computing and Communications, 2008, 25–27 Sept 2008, pp. 5–13. Dalian, China, (2008)
2. Christensen, J.H.: Using RESTful web-services and cloud computing to create next generation mobile applications. In: OOPSLA '09, 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications, 2009, pp 627–634. ACM, New York. doi:10.1145/1639950.1639958
3. Papakos, P., Capra, L., Rosenblum, D.S.: VOLARE: context-aware adaptive cloud service discovery for mobile systems. Paper presented at the Proceedings of the 9th international workshop on adaptive and reflective middleware, Bangalore, 2010
4. Dey, A.K.: Understanding and using context. Pers. Ubiquit. Comput. **5**(1), 125–134 (2001)
5. Saha, D., Mukherjee, A.: Pervasive computing: a paradigm for the 21st century. Computer **36**(3), 25–31 (2003). doi:10.1109/mc.2003.1185214
6. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: International Symposium on Handheld and Ubiquitous Computing, Karlsruhe, Germany, pp. 304–307. Springer, London (1999)
7. Schilit, B.N., Theimer, M.M.: Disseminating active map information to mobile hosts. IEEE Netw. **8**(5), 22–32 (1994). doi:10.1109/65.313011
8. Chen, G., Kotz, D.: A survey of context-aware mobile computing research. Dartmouth college, Technical report. Dartmouth college, Hanover (2000)
9. Chaari, T., Ejigu, D., Laforest, F., Scuturici, V.-M.: A comprehensive approach to model and use context for adapting applications in pervasive environments. J. Syst. Softw. **80**(12), 1973–1992 (2007). doi:10.1016/j.jss.2007.03.010
10. Gellersen, H.W., Schmidt, A., Beigl, M.: Multi-sensor context-awareness in mobile devices and smart artifacts. Mob. Netw. Appl. **7**(5), 341–351 (2002)
11. Dey, A.K., Häkkilä, J.: Context-awareness and mobile devices. In: Handbook of Research on User Interface Design and Evaluation for Mobile Technology, vol. 2, pp. 205–217. Information Science Reference/IGI Global, New York/Hershey (2008)
12. Chalmers, D., Dulay, N., Sloman, M.: A framework for contextual mediation in mobile and ubiquitous computing applied to the context-aware adaptation of maps. Pers. Ubiquit. Comput. **8**(1), 1–18 (2004)
13. Kim, Y.S., Lee, K.H.: A lightweight framework for mobile web services. Comput. Sci. Res. Dev. **24**(4), 199–209 (2009)
14. Baldauf, M.: A survey on context-aware systems. Ad Hoc Ubiq. Comput. **2**(4), 263–277 (2007)
15. JoonSeok, P., Mikyeong, M., Seongjin, H., Keunhyuk, Y.: CASS: a context-aware simulation system for smart home. In: 5th ACIS International Conference on Software Engineering Research, Management & Applications, 2007, 20–22 Aug. 2007. pp. 461–467. Busan, South Korea (2007). doi:10.1109/sera.2007.60
16. Tao, G., Pung, H.K., Zhang, D.Q.: A middleware for building context-aware mobile services. In: IEEE 59th, Vehicular Technology Conference, 2004, 17–19 May 2004, vol. 2655, pp. 2656–2660. Milan, Italy (2004). doi:10.1109/vetecs.2004.1391402
17. Chen, H., Finin, T., Joshi, A.: Semantic Web in the context broker architecture. In: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications, 2004, 14–17 March 2004, pp. 277–286. Orlando, Florida (2004). doi:10.1109/percom.2004.1276865
18. Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., Altmann, J., Retschitzegger, W.: Context-awareness on mobile devices – the hydrogen approach. In: Paper Presented at the

Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03), Hawaii, 10 pp. (2003)
19. Salber, D., Dey, A.K., Abowd, G.D.: The context toolkit: aiding the development of context-enabled applications. Paper presented at the Proceedings of the SIGCHI conference on Human Factors in Computing Systems: The CHI Is the Limit, Pittsburgh (1999)
20. Hong, J.I., Landay, J.A.: An infrastructure approach to context-aware computing. Hum. Comput. Interact. **16**(2), 287–303 (2001)
21. Mell, P., Granc, T.: The NIST Definition of Cloud Computing [NIST special publication]. National Institute of Standards and Technology, Gaithersburg, Maryland, USA (2011)
22. Ferschas, A., Hechinger, M., Riener, A., Schmitzberger, H., Franz, M., Rocha, M.S., Zeidler, A.: Context-aware profiles. In: International Conference on Autonomic and Autonomous Systems, Silicon Valley, California, USA. 2006, pp. 48–48. IEEE (2006)
23. Butler, M.H.: Current Technologies for Device Independence. HP Laboratories Bristol, UK (technical report) 83 (2001)
24. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive Mob. Comput. **6**(2), 161–180 (2010). doi:10.1016/j.pmcj.2009.06.002
25. Truong, H.L., Dustdar, S.: A survey on context-aware web service systems. Int. J. Web Inf. Syst. **5**(1), 5–31 (2009)
26. Jähnert, J., Villagrá, V., Wesner, S.: The Akogrimo Mobile Grid Reference Architecture. Akogrimo Whitepaper. http://www.mobilegrids.org (2006)
27. Han, B., Jia, W., Shen, J., Yuen, M.C.: Context-awareness in mobile web services. In: Parallel and Distributed Processing and Applications, pp. 519–528. Springer Berlin Heidelberg (Book Chapter) (2005)
28. Athanasopoulos, D., Zarras, A.V., Issarny, V., Pitoura, E., Vassiliadis, P.: CoWSAMI: interface-aware context gathering in ambient intelligence environments. Pervasive Mob. Comput. **4**(3), 360–389 (2008)
29. Truong, H.L., Juszczyk, L., Manzoor, A., Dustdar, S.: ESCAPE: an adaptive framework for managing and providing context information in emergency situations. In: EuroSSC'07 Proceedings of the 2nd European Conference on Smart Sensing and Context, 2007, pp 207–222. Springer, Berlin/Heidelberg (2007)
30. De Almeida, D.R., de Souza Baptista, C., Da Silva, E.R., Campelo, C.E.C., De Figueiredo, H.F., Lacerda, Y.A.: A context-aware system based on service-oriented architecture. In: 20th International Conference on Advanced Information Networking and Applications. Vienna, Austria 2006
31. Maamar, Z., AlKhatib, G., Mostefaoui, S.K.: Context-based personalization of web services composition and provisioning. In: IEEE 30th Euromicro Conference, 2004, pp. 396–403. Rennes, France (2004)
32. Seyler, F., Taconet, C., Bernard, G.: Context adaptation of web service orchestrations. In: 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2007, pp. 351–356. Paris, France (2007)
33. Soukkarieh, B., Sedes, F.: Integrating a context model in web services. In: IEEE International Conference on Web Services, 2007, pp. 1195–1196. Salt lake city, UTAH (2007)
34. Badidi, E., Esmahi, L.: A cloud-based approach for context information provisioning. World Comput. Sci. Inf. Technol. J. **1**(3), 63–70 (2011)
35. McGraw, I., Lee, C., Hetherington, L., Seneff, S., Glass, J.: Collecting voices from the cloud. In: Proceedings of LREC, Malta, May 2010
36. Angin, P., Bhargava, B., Helal, S.: A mobile-cloud collaborative traffic lights detector for blind navigation. In: Eleventh IEEE International Conference on Mobile Data Management, 2010, pp. 396–401. Kanas City, Missouri, USA (2010)
37. Abowd, G.D., Atkeson, C.G., Hong, J., Long, S., Kooper, R., Pinkerton, M.: Cyberguide: a mobile context-aware tour guide. Wirel. Netw. **3**(5), 421–433 (1997)

38. Kovachev, D., Cao, Y., Klamma, R.: Mobile Cloud Computing: A Comparison of Application Models. Arxiv preprint arXiv:11074940 (2011)
39. Chun, B-G., Maniatis, P.: Augmented smartphone applications through clone cloud execution. Paper presented at the Proceedings of the 12th conference on Hot Topics in Operating Systems, Monte Verit. Springer-Verlag Berlin, Heidelberg 2009
40. Kim, J.: Design and evaluation of mobile applications with full and partial offloadings. In: Advances in Grid and Pervasive Computing, pp. 172–182 (2012)
41. Pilgrim, M. (ed.): HTML5: Up and Running, 1st edn. O'Reilly Media, Sebastopol (2010)
42. Kumar, K., Yung-Hsiang, L.: Cloud computing for mobile users: can offloading computation save energy? Computer **43**(4), 51–56 (2010)
43. Hinckley, K., Pierce, J., Sinclair, M., Horvitz, E.: Sensing techniques for mobile interaction. Paper presented at the Proceedings of the 13th Annual ACM Symposium on User Interface software and Technology, San Diego, 2000
44. Sivashanmugam, K., Verma, K., Sheth, A., Miller, J.: Adding semantics to web services standards. In: International Conference on Semantic Web and Web Services, 2003, pp. 395–401. Erfurt, Germany (2003)
45. Preuveneers, D., Berbers, Y.: Adaptive context management using a component-based approach. In: Distributed Applications and Interoperable Systems, Athens, 2005, pp 1080–1082. Springer, Berlin/Heidelberg (2005)
46. Preuveneers, D., Van den Bergh, J., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V., De Bosschere, K.: Towards an extensible context ontology for ambient intelligence. In: Ambient Intelligence: Second European Symposium, EUSAI, Eindhoven, 2004, pp. 148–159. ACM, New York (2004)
47. Azeez, A., Perera, S., Weerawarana, S., Fremantle, P., Uthaiyashankar, S., Abesinghe, S.: WSO2 Stratos: an application stack to support cloud computing. it-Inf. Technol. **53**(4), 180–187 (2011)
48. Preuveneers, D., Berbers, Y.: Mobile phones assisting with health self-care: a diabetes case study. In: Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services, 2008, pp 177–186. ACM, New York (2008)

Chapter 9
Potential Concerns and Common Benefits of Cloud-Based Enterprise Resource Planning (ERP)

S. Parthasarathy

Abstract Enterprise resource planning (ERP) is an industry-driven concept and system that is universally accepted by the industry as a practical solution to achieve integrated enterprise information systems. Enterprises are analyzing the use of cloud provision to carry out their business processes. The shift to the cloud-based business management technology has already started as it leverages the use of abstracted resources. This chapter strives to compare and contrast conventional ERP and cloud ERP systems with respect to three perspectives: organizational, business, and technological. In this chapter, the architecture of conventional ERP and cloud ERP are discussed as well as the business drivers and the deployment options during the implementation of cloud-based ERP system. A case study of a cloud ERP system is carried out, and the findings are presented for the benefit of the ERP implementation teams. The objective of the research study is to help the top management to determine the best deployment options for their organizations. This has been done by means of highlighting the potential benefits an enterprise can reap if they choose to stay with cloud ERP and the concerns they should address to minimize the risks leading to a low-cost and effortless implementation of a complete cloud-based ERP software solution.

Keywords Cloud • Enterprise resource planning • ERP • Software • Risks

9.1 Introduction to ERP and Cloud

Enterprise resource planning (ERP) is a generic term for integrated systems. ERP projects are a subclass of software projects often accompanied by large-scale organizational changes. ERP attempts to integrate all departments and functions across

S. Parthasarathy (✉)
Department of Computer Applications, Thiagarajar College of Engineering,
Madurai 625 015, Tamil Nadu, India
e-mail: parthatce@gmail.com

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_9,

a company onto a single computer system that can serve all the particular needs of those different departments. That is a tall order, building a single software program that serves the needs of people in finance and answers the needs of the people in human resources and in the warehouse. Each of those departments typically has its own computer system optimized for the particular ways that the department does its work. ERP combines them all together into a single integrated software program that runs off a single database so that the various departments can more easily share information and communicate with each other. That integrated approach can have a tremendous payback if companies install the correct software.

ERP covers the techniques and concepts employed for the integrated management of businesses as a whole, from the viewpoint of the effective use of management resources, to improve the efficiency of an enterprise. Enterprise resource planning is a term originally derived from manufacturing resource planning (MRP II) that followed material requirements planning (MRP). MRP evolved into ERP when "routings" became a major part of the software architecture, and a company's capacity planning activity also became a part of the standard software activity. ERP applications have been around for over 60 years, making them one of the oldest and most well-known business applications utilizing modern computer technology. From the punched card era with mainframe technology through the introduction of minicomputers, the advent of personal computers, the creation of the Internet, and the new world of cloud computing, ERP software vendors have traditionally kept up with advances in technology. ERP software standardizes an enterprise's business processes and data. The software converts transactional data into useful information and collates the data so that they can be analyzed. In this way, all of the collected transactional data become information that companies can use to support business decisions. The perspective that ERP software is simply a means of cutting cost is still prevalent. As a result, organizational resistance to ERP implementation has often been high, and not all ERP implementation programs have delivered the promised enterprise improvements.

Integration is the first and most important advantage of implementing ERP. The reason why ERP packages are considered for integrated is the automatic data updating that is possible among the related business components. Since conventional company information systems were aimed at the optimization of independent business functions in business units, almost all were weak in terms of the communication and integration of information that transcended the different business functions. In the case of large companies in particular, the timing of system construction and directives differs for each product and department/function, and sometimes it is disconnected. For this reason, it has become an obstacle in the shift to new product and business classification. In the case of ERP packages, the data of related business functions is also automatically updated at the time a transaction occurs. For this reason, one is able to grasp business details in real time and carry out various types of management decisions in a timely manner, based on that information.

The second advantage of an ERP system is their flexibility within the business functions. Different languages, currencies, accounting standards, and so on can be covered in one system, and functions that comprehensively manage multiple locations

of a company can be packaged and implemented automatically. To cope with company globalization and system unification, this flexibility is essential, and one can say that it has major advantages, not simply for development and maintenance but also in terms of management. Another advantage of an ERP package is its better analysis capability. By enabling the comprehensive and unified management of related business and its data, it becomes possible to fully utilize many types of decision support systems and simulation functions. Furthermore, since it becomes possible to carry out, flexibly and in real time, the filing and analysis of data from a variety of dimensions, one is able to give the decision makers the information they want, thus enabling them to make better and more informed decisions. The fourth advantage is the utilization of the latest developments in information technology (IT).

The ERP vendors were very quick to realize that in order to grow and to sustain that growth, they had to embrace the latest developments in the field of information technology. Therefore, they quickly adapted their systems to take advantage of the latest technologies like open systems, client/server technology, the Internet/intranet, and electronic commerce. It is this quick adaptation to the latest changes in information technology that makes the flexible adaptation to changes in future business environments possible. It is this flexibility that makes the incorporation of the latest technology possible during system customization, maintenance, and expansion phases. ERP includes many of the functions that will be necessary for future systems. However, undertaking reforms in company structures and business processes, so as to enable the full use of these major features, is the greatest task for companies that will use them. It is necessary to take note that casually proceeding with the implementation of ERP merely for reasons of system reconstruction is likely to result in turning the advantages into disadvantages.

The most important aspect to consider when installing an ERP system in a company is the attitude of the employees at all levels of the company to reform. How managers use information systems and information technology as strategic management methods is likely to become the turning point of the change. Other benefits of an ERP system include reduction in lead time and cycle time, on-time shipment, better customer satisfaction, improved supplier performance, increased flexibility, reduction in quality costs, improved resource utility, improved information accuracy, and enhanced decision-making capability. The elapsed time between placing an order and receiving it is known as lead time. It plays a significant role in purchasing and inventory control. In order to reduce the lead time, the organization should have efficient inventory management systems, integrated with the purchasing, production planning, and production departments. In this era of just-in-time manufacturing, the knowledge of the exact lead time for each and every item is of great importance for uninterrupted production.

For a company dealing with hundreds and thousands of raw materials and components, keeping track manually of the lead time for each and every individual item is practically an impossible task. Companies must be able to deliver customer-specific products (made-to-order). ERP systems provide them the freedom to change manufacturing and planning methods as needs change, without modifying or reconfiguring the workplace or plant layouts. With ERP systems, businesses are not limited to a

single manufacturing method, such as made-to-stock or made-to-order. Cycle time is the time between the receipt of the order and the delivery of the product. It can be reduced by the ERP systems, but the reduction will be more in the case of made-to-order systems. In the case of made-to-stock, the items are already manufactured and kept in warehouses or with distributors for sale. Here, the cycle time is reduced not on shop floor but during order fulfillment.

ERP systems provide vendor management and procurement support tools designed to coordinate all aspects of the procurement process. They support the organization in its efforts to effectively negotiate, monitor, and control procurement costs and schedules while assuring superior product quality. The supplier management and control processes comprise of features that will help the organization in managing the supplier relations and the supplier quality and monitoring the vendor activities. Flexibility is a key issue in the formulation of strategic plans in companies. Sometimes, flexibility means quickly changing something that is in practice, or completely changing to adjust to new product designs. In other words, flexibility is the ability to produce in small quantities, in order to produce a product mix that may better approximate actual demands and reduce work-in-progress inventories.

The term ERP originally implied systems designed to plan the use of enterprise-wide resources. Although the acronym ERP originated in the manufacturing environment, today's use of the term ERP systems has much broader scope. The best practices were also a benefit of implementing an ERP system. When implementing an ERP system, organizations essentially had to choose between customizing the software or modifying their business processes to the "best practice" functionality delivered in the ERP software. Because of their wide scope of application within a business, ERP software systems are typically complex and usually impose significant changes on staff work practices. Implementing ERP software is typically not an "in-house" skill, so even smaller projects are more cost effective if ERP implementation consultants are employed. The length of time to implement an ERP system depends on the size of the business, the scope of the change, and the willingness of the customer to take ownership for the project. A small project (e.g., a company of less than 100 staff) may be planned and delivered within 3 months; however, a large, multi-site, or multi-country implementation may take years.

To implement ERP, companies often seek the help of an ERP vendor or of third-party consulting companies. These firms typically provide three areas of professional services: consulting, customization, and support. The consulting team is typically responsible for the initial ERP implementation and subsequent delivery of work to tailor the system beyond "go live." Typically such tailoring includes additional product training, creation of process triggers and workflow, system optimization, and assistance in writing reports, complex data extracts, or implementing business intelligence. The consulting team is also responsible for planning and jointly testing the implementation. This is a critical part of the project and one that is often overlooked. Consulting for a large ERP project involves three levels: systems architecture, business process consulting (primarily re-engineering), and technical consulting (primarily programming and tool configuration activity). A systems architect designs the overall dataflow for the enterprise including the future dataflow plan. A business consultant studies an organization's current business processes and

matches them to the corresponding processes in the ERP system, thus "configuring" the ERP system to the organization's needs.

Technical consulting often involves programming. Most ERP vendors allow modification of their software to suit the business needs of their customer. Customization is the process of extending or changing how the system works by writing new user interfaces and the underlying application code. Such customizations typically reflect local work practices that are not currently in the core routines of the ERP system software. Smaller enterprises ducked the ERP wave in the mid-1990s and did not succumb to the cost structures or the risks that their larger brethren undertook. Now, these firms often have outdated or overloaded applications and need to invest in new technology to achieve or retain competitiveness in their markets. Today, many small and midsize businesses are in need of upgrading, different software applications, current or robust technology, and often more sophisticated business solutions like the ERP software solution. Due to increasing interorganizational production and control, a new management dimension occurs that focuses on both manufacturing activities and the supply chain. Its demands are cost control, needs to analyze costs/revenues on a product or customer basis, flexibility to respond to changing business requirements, more informed management decision making, and changes in the way of doing business. The enterprise information systems and the business software applications have continuously followed the new managerial concepts, evolving from systems without decision support toward complex applications.

The unified nature of an ERP system can lead to significant benefits, including fewer errors, improved speed and efficiency, and more complete access to information. With better access to information, employees and managers can gain a better understanding of what is going on in the enterprise so that they make better business decisions. Implementing ERP on platforms is not always easy because of the massive re-engineering process that involves security, quality assurance, and training for members of the organization entrusted to use the ERP systems. In addition to maximizing the operational effectiveness of the ERP, IT managers, system security officers, and system developers will find themselves challenged on security and control issues. Enterprise systems are complex and expensive and create dramatic organizational changes. ERP systems have the Herculean task of seamlessly supporting and integrating a full range of business processes, uniting functional islands, and making their data visible across the organization in real time [22]. ERP software is one of the fastest-growing segments of business computing today. The motivation for the enterprises to implement ERP systems are the integration, the standardization capabilities, the flexibility to customize the software, and their ability to drive effective business process reengineering and management of core and support processes [11, 26].

9.1.1 Features of Cloud Computing

Cloud computing is a new paradigm that is transforming the information technology (IT) industry and reshaping the way enterprise services are developed, deployed, sold, delivered, and consumed [13]. Instead of managing complex IT

Table 9.1 Features of cloud computing

Parameters	Cloud computing
Virtualization	Essential
Type of application	Interactive
Application development	In the cloud
Access	Via standard web protocols
Organization	Physical
Business models	Pricing (utility model, pay per use)
Service-level agreements	Essential
Control	Centralized (data center)
Openness	Low
Ease of use	Easy
Switching cost	High, due to incompatibilities

systems, customers can focus on the core competence of their enterprise while obtaining all required IT functions as a service. From the perspective of a cloud provider, remaining competitive and realizing the full potential of economies of scale that the cloud paradigm promises require extreme levels of standardization, automation, and optimization. Table 9.1 shows the features of cloud computing [2].

Figure 9.1 shows the basic architecture of cloud computing [13]. Cloud computing is staking a growing presence on the market with innovative, feasible business models and solutions that pose a threat to the business model of established IT vendors. Leading market analysts such as Forrester or Gartner estimate that by 2013 approximately 25 % of all IT services will be delivered in the form of cloud computing solutions. The traditional view of current cloud computing is broken into three separate layers:

- SaaS (Software as a Service)
- PaaS (Platform as a Service)
- IaaS (Infrastructure as a Service)

Figure 9.2 shows the layers of information technology (IT) as a service in cloud environment. At the basic level, "the cloud" or "cloud computing" refers to the method of provisioning software and hardware (from e-mail to accounting to customer relationship management to hardware platforms and infrastructures) to users via the Internet. Public cloud refers to infrastructure comprised of multi-tenant servers. With private cloud services, servers are dedicated to one organization. The term hybrid cloud refers to the combination of public cloud, private cloud, or on-premises infrastructure services. Cloud computing is an opportunity to completely transform how your business and its people work. Saving the organizations from capital expenditures that last longer on the balance sheet than in the server room may be the most often cited benefit, but there are some that may not have been considered.

The potential of emerging cloud computing business models and their competitive advantage over licensed software have been debated quite widely in the recent past [17, 28]. Most of these studies agree that the most appropriate business model for cloud computing in user-driven environment of today's markets is the SaaS model. Software as a Service (SaaS) architecture has been regarded as the

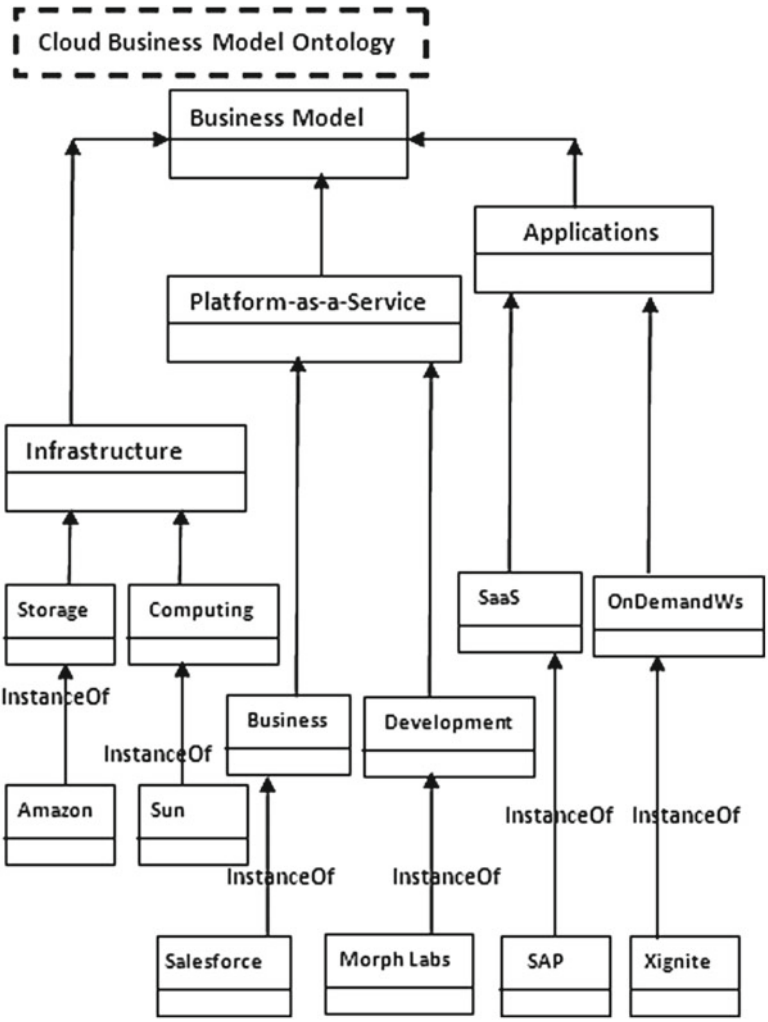


Fig. 9.1 Basic cloud computing architecture

leading business model for cloud computing since the dawn of the twenty-first century [8, 9, 19]. The general consensus [14, 23] is that the infrastructure outsourcing business model should follow the utility model. But what makes a particular service a utility is shaped by a combination of requirements, which Rappa [15] lists as: necessity, reliability, usability, utilization rates, scalability, and service exclusivity. Users run all their own applications, including operating systems, storage, and backup on a cloud provider's infrastructure.

The cloud vendors do not supply any operating system license or support but only the hardware and connectivity. The users' access to the purchased or internally developed applications are housed on the cloud provider's infrastructure with provider

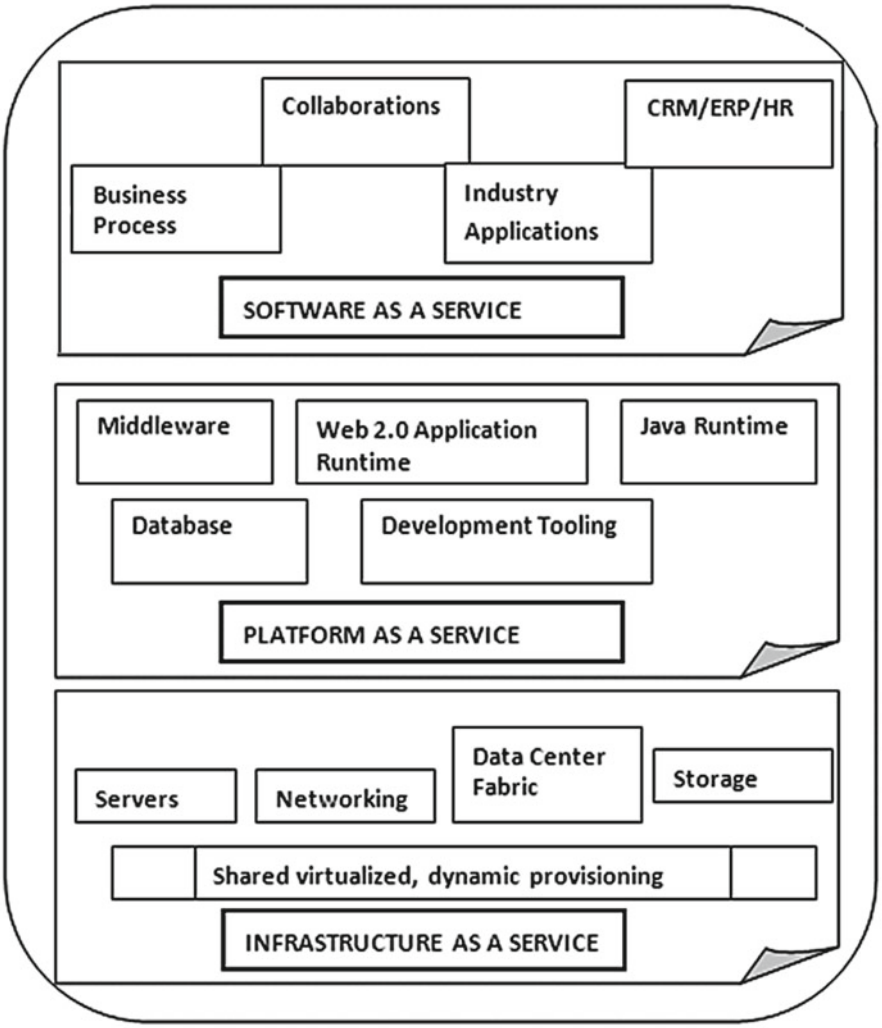


Fig. 9.2 Layers of IT as a service in cloud

managing the operating system, storage, hardware, and networking. The cloud vendor provides the additional layer of the operating system and supporting services to allow the users to focus on supporting their custom or purchased business operating applications. The term "hosting" or "hosted" is commonly associated with ERP software. In addition to the hardware and operating systems, the cloud provider houses and manages the installation, upgrades the user configurations for the ERP application. The ERP application licenses may either be subscribed to or owned. Users subscribe or rent access to the software application functionality over the Internet. The software publisher delivers the application to many customers, storing customer application-related data on public cloud servers [5].

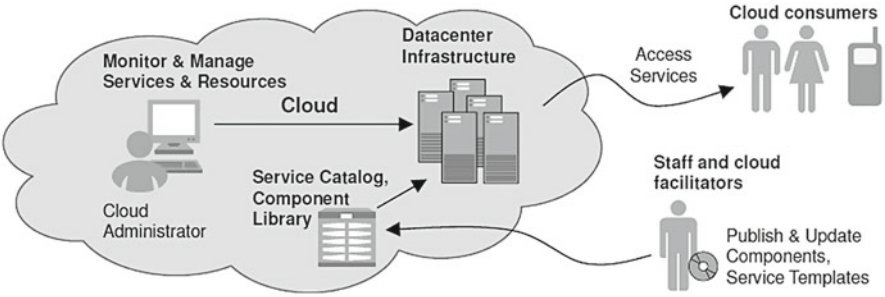


Fig. 9.3 Infrastructures and service management in cloud

9.2 Architectures of Conventional and Cloud ERP Systems

Cloud computing is changing the way industries and enterprises do their businesses in that dynamically scalable and virtualized resources are provided as a service over the Internet. Cloud computing is emerging as one of the major enablers for the manufacturing industry; it can transform the traditional manufacturing business model, help it to align product innovation with business strategy, and create intelligent factory networks that encourage effective collaboration [27]. Cloud consulting combines open source grid computing for distributed cloud computing and enterprise resource modeling (ERP) to provide Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) through a simple, unified API. Cloud communication refers to using Internet-based or cloud-based voice and data communications services, where telecommunications applications, switching, and storage are managed generally by third parties. These services can include capabilities that range from Voice over IP (VoIP) communications to hosted PBX and unified communications delivering voice, fax, video, and data requirements. Provisioning for these services is known as Communication as a Service (CaaS) [20]. Cloud computing is an emerging style of IT delivery in which applications, data, and IT resources are rapidly provisioned and provided as standardized offerings to users over the web in a flexible pricing model. It is a way of managing large numbers of highly virtualized resources so that, from a management perspective, they resemble a single large resource. This can then be used to deliver services with elastic scaling as shown in Fig. 9.3 [6].

The RESERVOIR (Resources and Services Virtualization without Barriers) Framework as shown in Fig. 9.4 [16] is a blueprint for companies to build a cloud with all the codes and architecture specifications needed to do so. Using the RESERVOIR open source technologies, cloud providers can now build an advanced cloud with the balancing of workloads, lowering costs, and moving workloads across geographic locations through a federation of clouds. RESERVOIR components are power tools enabling the on-demand delivery of more cost-effective services and more moderate operating costs, tested and deployed by four industry success stories. RESERVOIR enables the delivery of better services for businesses

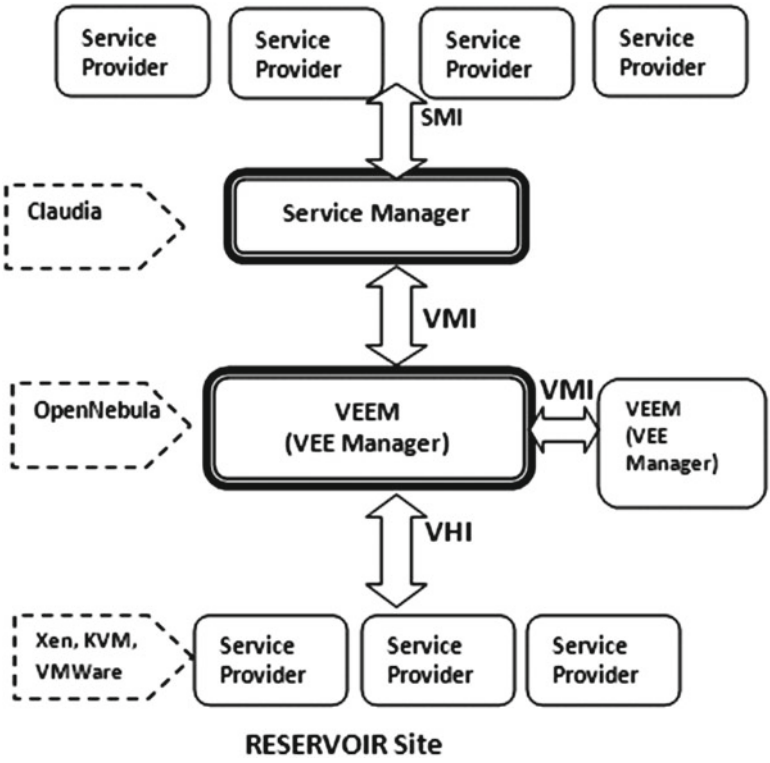


Fig. 9.4 RESERVOIR cloud infrastructures [16]

and eGovernment with energy efficiency and elasticity by increasing or lowering it based on demand. Leading ERP vendors such as the SAP success stories demonstrate the strengths of the RESERVOIR cloud infrastructure. Virtual Machine (VM), Service Provider (SP), Virtual Execution Environment (VEE), Open Virtualization Format (OVF), Service Manager (SM), Virtual Execution Environment Manager (VEEM), Service Management Interface (SMI), VEE Management Interface, VEE Host Interface (VHI), and Virtual Area Network (VAN) are indicated in Fig. 9.4 as open source cloud infrastructure.

SAP ERP 6.0 is a part of the SAP Business Suite. The ERP facilitates the flow of information among all business functions within the company as well as with other stakeholders. The suite runs over the SAP NetWeaver Application Server (NWAS), a three-tier architecture shown in Fig. 9.5 [4]. In this architecture, the presentation layer deals with the SAP graphical user interface (GUI) and the browser. The application layer consists of two elements, namely, the Dialog Instance (DI) and the Central Instance (CI). The central instance implements many services such as dialog, update, batch, spool, gateway, message server, and the enqueue server. The dialog instance processes user requests or Remote Function Call (RFC) from remote systems.

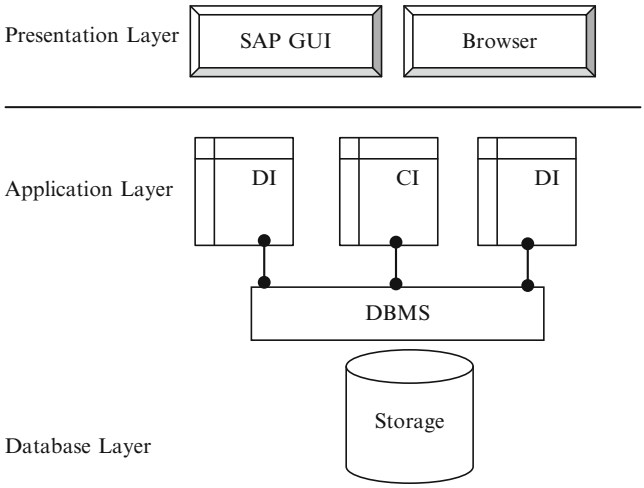


Fig. 9.5 SAP NETWeaver three-tier architecture [4]

9.3 Common Concerns

Enterprises that move their IT to the cloud are likely to encounter challenges such as security, interoperability, and limits on their ability to tailor their ERP to their business processes. The cloud can be a revolutionary technology, especially for small start-ups, but the benefits wane for larger enterprises with more complex IT needs [10]. The cloud model can be truly disruptive if it can reduce the IT operational expenses of enterprises. Traditional utility services provide the same resource to all consumers. Perhaps the biggest difference between the cloud computing service and the traditional utility service models lies in the degree to which the cloud services are uniquely and dynamically configured for the needs of each application and class of users [12]. Cloud computing services are built from a common set of building blocks, equivalent to electricity provider turbines, transformers, and distribution cables. Cloud computing does, however, differ from traditional utilities in several critical respects. Cloud providers compete aggressively with differentiated service offerings, service levels, and technologies. Because traditional ERP is installed on your servers and you actually own the software, you can do with it as you please. You may decide to customize it, integrate it to other software, etc. Although any ERP software will allow you to configure and set up the software the way you would like, "Software as a Service" or "SaaS" is generally less flexible than the traditional ERP in that you can't completely customize or rewrite the software. Conversely, since SaaS can't be customized, it reduces some of the technical difficulties associated with changing the software. Cloud services can be completely customized to the needs of the largest commercial users. Consequently, we have often referred to cloud computing as an "enhanced utility" [12]. Table 9.2 [5] shows the E-skills study for information and communications technology (ICT) practitioners conducted by the Danish Technology Institute [5] that describes the

Table 9.2 E-skills for cloud computing

Parameters	Cloud computing
Strategy and innovation	Understand the possibilities in cloud computing and how this translates into the business strategy
Enterprise architecture	Skills around the cloud such as built-in browsers and knowledge about the users. The architects do not need to build everything as before
Demand management	Capturing and prioritizing demand, assigning resources based on business objectives, and doing projects that deliver business value
Global sourcing	Negotiating skills. Insights into legal issues of privacy
Management	Transportation and storage of data
Project delivery	Skills regarding specification and deployment of ICT solution. Knowledge of standards and web service design
IT support and execution	Maintenance of hardware. Brokers and translators between business and vendors
Quality, risk, and compliance	Risk and compliance

character of skills needed. In other words, it highlights the technical requirements and other skill set required for the deployment of the cloud computing-based information system in enterprises.

Some of the common concerns of the users of cloud ERP are lack of control, vendor dependency, lack of interoperability with existing IT systems, legal and compliance issues, reliability, and data security. The main concern with cloud computing is the risk of having our data accessible to others on the web. Another concern is that a reputable cloud provider is required to clearly articulate specific governance, operational and regulatory guidelines that they follow to assure security of data. A complete and standard set of policies has to be framed for users' management, authorization, authentication, and access control. A cloud ERP vendor has to be chosen in such a way that their service-level agreement (SLA) is reviewed. Also, to verify whether the vendor has redundant infrastructure facilities in the case of any natural calamities and disaster at the primary facility center, a standby or redundant Internet service is required to ensure that an uninterrupted Internet connection is in place for the users when the primary Internet service connection goes down. Not all ERP solutions have the flexibility to run either in the cloud or on premises. Some vendors only support cloud deployment, while others allow you to change your deployment method without forcing you to go through a new implementation. If your business needs change in the future, having this option available could mean huge financial savings.

9.4 Benefits of Cloud ERP Systems

Cloud computing is rapidly emerging as the new platform for computing. It is, however, much more than simply a new set of technologies and business models. Cloud computing is transforming the way consumers, companies, and governments store

Table 9.3 Costs on conventional ERP vs. cloud ERP

No. of users	On-premise ERP (Lac)	Cloud ERP (Lac)
5 users, 2 remote users	45.29	24
10 users, 3 remote users	91.59	39
25 users, 3 remote users	126.79	66.5
50 users, 5 remote users	194.91	104

information, how they process that information, and how they utilize computing power. It can be an engine of innovation, a platform for entrepreneurship, and driver of corporate efficiency. However, while the term is increasingly commonly used, confusion remains over what exactly constitutes cloud computing, how the markets are unfolding, and what forces will drive their evolution and diffusion [12].

Cloud computing has enabled organizations to shift their information and communications technology (ICT) infrastructures to third parties, thereby optimizing their business options and reducing operational scope. Cloud computing has enhanced the operational efficiency of the organizations; also several new businesses and service opportunities have opened to cater to the growing demand [1, 3, 7, 25]. Current customers in India vouch for the benefits that cloud computing offers. From reduced capital expenditure to more efficient operations in the IT administration department, there is more and more business that is coming in from existing customers. Birla Tyres, a current Ramco customer, initially moved their sales, distribution, and dealer management operations to Ramco OnDemand (an ERP product from Ramco Systems) [17]. It also wanted to integrate analytics and business intelligence tools by gathering data from other systems. It is believed that the speed, flexibility, and cost savings from Ramco's ERP solution are now prompting them to move the whole system into the cloud.

Table 9.3 shows the costs involved in an ERP implementation on the conventional platform versus the cloud platform [24]. After referring to the research works [12, 16, 18, 21, 22, 28] and white papers on ERP products of leading ERP vendors, namely, the SAP, the Microsoft, and Ramco, Table 9.3 has been prepared to compare the features and potential benefits of conventional ERP system and cloud ERP system. The traditional ERP system runs on a local server, and users are connected to the system through intranet, extranet, and the Internet. The technology, architecture, and business models are also vendor specific and customer centric. The customization too becomes expensive and complex to fit the traditional ERP system into the enterprise. These issues are overcome in cloud ERP system, as shown in Table 9.4.

9.5 Case Study

In this section, we present the implementation of a cloud ERP system in a leading bank in India, by one of the leading ERP vendors, the Ramco System. The architecture of Ramco's cloud ERP system, the business challenges and the enterprise

Table 9.4 Conventional ERP vs. cloud ERP

Enterprise drivers	Conventional ERP	Cloud ERP
Cost/resources		
Operational cost	High	Low
Fixed cost	High	Relatively low
Ongoing cost	High	Relatively low
Maintenance	Complex	Easier
Man power reduction	20 %	30 %
Deployment/implementation		
Server	Local server	Cloud server
Implementation duration	24–36 months	9–18 months
Integration to Extension Development Kit (EDK) &Portal Development Kit (PDK)	Can be done	Readily available
Usability		
Access to users	On-premise users	On-premise and remote users
Software licensing	Company owns it	Subscription model and rental model
Response to customers	Depends on query	Faster
User friendliness	Depends on business alignment	High
Flexibility to increase or decrease users based on market dynamics	Requires vendor support	Dynamic
Functionality		
Streamlining of business processes	Standard procedure	Dynamism in defining core business processes
Process customization	Complex	Easier
System customization	Complex	Easier
IT alignment	Static business model	Optimized business model
Automation	Possible	Possible with less efforts
Security/safety measures		
Data security	Varies from one ERP to another	Inbuilt
Ease of disaster mitigation	Varies from vendor to vendor	Finest management
Secure data center	Not available by default	Powerful data storage
Automatic backups	Possible with external devices	Readily available
Architecture/platform		
Virtualization	Not addressed	Leverages virtualization
Software architecture	Static	Dynamic
Business models	Static	Dynamic to meet current business needs
Upgradation/new release	Needs vendor support	Can be automated
Operating system compatibility	Less	More

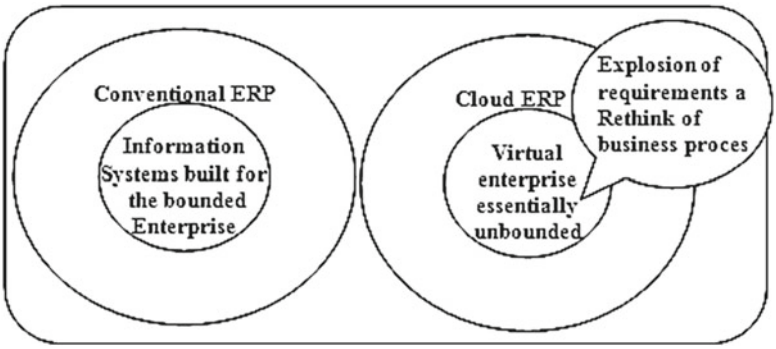


Fig. 9.6 ERP for enterprises

challenges they had before implementation, and the customer satisfaction at the end of the day of implementation of cloud ERP are discussed in this section.

Figure 9.6 shows the two options on developing an ERP system for enterprises. The conventional ERP system's limitations and the cloud ERP system's ability to scale up to the large enterprises are highlighted in Fig. 9.6 for the benefit of the readers. The case study discussed here considers the cloud ERP system designed and developed by a leading ERP vendor and successfully deployed in a leading bank. The following lists present business and enterprise challenges.

Business Challenges

- Need to leverage scope and scale of IT investments
- Need to be responsive
- Constant need for IT to change – needs expensive manpower, difficult to control outcome and budgets, and tough to satisfy business line managers

Enterprise Challenges

- Is the cloud geared for implementation in large and complex business environments?
- On the cloud, do we have the flexibility to create and maintain customizations?
- Can cloud ERP coexist with, and complement, conventional ERP?

Figure 9.7 shows Ramco's cloud computing architecture [17]. Ramco Virtual-Works is a tested and proven enterprise cloud computing platform that enables the development of multi-tenant, robust, and scalable web-enabled solutions. The Ramco VirtualWorks platform is based on a technology-independent SOA architecture; it protects your past investments and future-proofs your current infrastructure. It enables flexible, integrated business processes and focused analytics. It is a model-driven development methodology working on enterprise cloud computing platform and designed for the three business realities –uniqueness, change, and diversity. Ramco's cloud ERP system was implemented in a leading bank in India. This bank has nearly 3,500 users across 3,000 branches with 4 million loan accounts, 40 million deposit accounts, and 43 million customers.

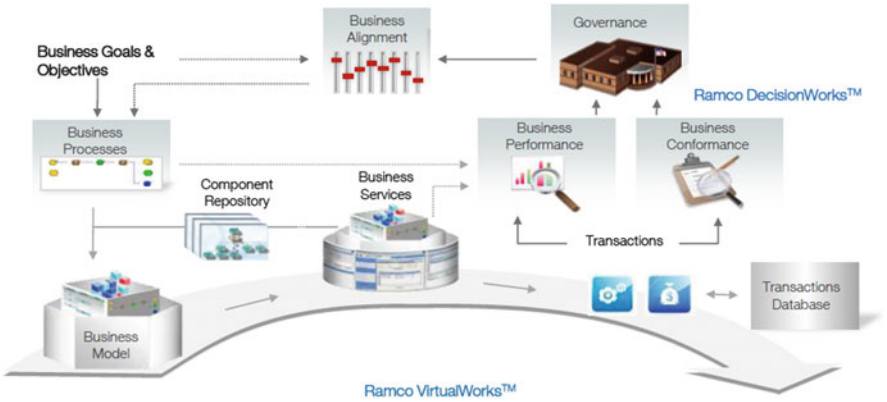


Fig. 9.7 Ramco VirtualWorks: cloud architecture

This bank, after the implementation of the cloud ERP, noticed the following business benefits: effective interface among core business areas, namely, loans, deposits, customers, trade finance, and financial profitability. The product user friendliness was found excellent. The extremely rapid implementation of huge data warehouse (7 TB Data) was carried out successfully. The pre-built analytics resulting in minimal engineering for DW design and building content is provided, and the OnDemand MIS is possible without involving IT. The conventional ERP implementation takes 2 years of duration, whereas, in this case, the implementation was done in 9 months. This project includes a single operational data source (ODS); portfolio analysis covering business growth, risk, and operations; and decision making from the corporate office to the branch offices. These are some salient features of this cloud ERP system for the bank.

Figure 9.8 [17] shows the benefits of Ramco's cloud ERP system when it was successfully launched in a leading bank in India. The customers' feedback were found satisfactory, and the product was recognized for its product functionality, tools and services for end user's direct usage, training and support capability, local statutory requirements, and change management.

9.6 Conclusions

The exquisiteness of modern ERP packages is that you have options to choose from during your ERP software selection. You can either deploy a solution by hosting it internally on your own servers (traditional ERP) or perhaps you would rather not deal with the software and have it hosted somewhere else as Software as a Service (SaaS). As part of the ERP selection process, however, you should be aware of five key variables that will ultimately factor into the decision that is right for you. It has been reported in cloud computing research studies that it promises to change information handling practices with widespread effects on

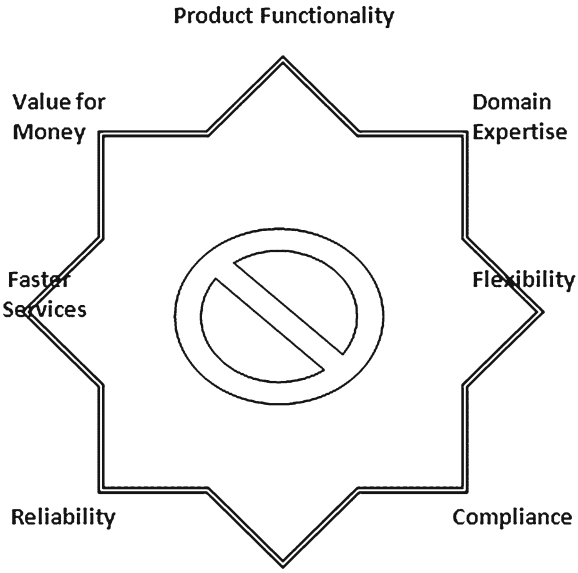


Fig. 9.8 Benefits of cloud ERP as reported by customers [17]

organizational structures and strategies, on small firms' access to advanced information technologies, on the costs of certain kinds of experimentation and innovation, and on the skills base and deployment of labor. These effects will differ according to numerous criteria, including the capabilities of particular firms and their current level of capital investment in IT, the characteristics of specific sectors, and the national context. Small- and medium-size companies cannot afford the cost of purchasing, implementing, and hiring ERP consultant and domain experts and cannot provide the required resources and infrastructure to support ERP system. Until recently, it had been assumed that only big businesses can afford such systems. For such small- and medium-size companies, cloud-based ERP may be the best option to avail the capabilities of ERP system without investing huge amount on various resources used and infrastructure to set up.

Cloud services are already transforming the business models of companies around the world – from small start-ups and medium-size firms to large multinational enterprises. Cloud computing offers new capabilities for innovation and entrepreneurship, lowering the bar for new entrants and facilitating experimentation. At the same time, it raises the scale required to be an effective computing infrastructure provider, and customer expectations of service flexibility and costs are undergoing a sea change. The basic architecture and business processes behind the conventional ERP and the cloud ERP, the cloud computing architecture, the business drivers of cloud ERP from an enterprise perspective, the features of conventional ERP and cloud ERP, the cost factors influencing cloud ERP, the deployment options for enterprises when they choose to go with cloud, and the potential benefits and common concerns on cloud ERP implementation have been discussed in this chapter. In order to allow the easy use of cloud systems and to enable the migration of applications between the clouds of different providers, there will be the need for a standardized cloud API. It is

observed that the clouds require new business models, especially with respect to the licensing of software. This has already become an issue in grids. Current licenses are bound to a single user or physical machine. However, in order to allow the efficient sharing of resources and software licenses, there has to be a move toward usage-based licenses, penalties and pricing, very much in the spirit of utility computing. Software vendors have to realize that clouds will especially enable and drive the adoption of Software as a Service models.

A case study is also presented to touch upon the responses received by the customers of the ERP vendors on cloud ERP implementation. A cloud-based solution that just works expands easily into new markets and rapidly scales your global ERP system with minimal IT effort; provides real-time visibility across your operations for financial reporting and faster operational decision making; reduces complexity and simplifies compliance with multisite, multi-language, multi-company, and multi-currency capabilities; improves your business processes, including intercompany trade, multi-site planning, shared services, and book closing; optimizes local operations while meeting enterprise requirements to consolidate financial and operational data; and implements a two-tier approach by synchronizing financial data between the divisional and parent organizations.

ERP applications have been around for over 60 years, making them one of the oldest and best business applications utilizing modern computer technology. From the punched card era with mainframe technology through the introduction of minicomputers, the advent of personal computers, the creation of the Internet, and the new world of cloud computing, ERP software vendors have traditionally kept up with advances in technology. The cost of ownership and the implementation costs for new releases of ERP are found to be the key factors for enterprises migrating from the standard ERP system to the cloud ERP system. Nowadays, the management of custom applications has become difficult to maintain. In a cloud ERP implementation, custom applications can live synergistically with the "standard" system. The solutions and services will help the cloud ERP customers to reduce the total cost of ownership (TCO) of their ERP systems, improve their scalability and business agility, and hide complexity by getting a coordinated set of solutions and services simplifying and optimizing the effort required to provision, deploy, monitor, and manage their ERP systems in virtual infrastructures and cloud environments and with the input of customers. A future research work shall consider the cloud ERP system deployed in multiple enterprises by different ERP vendors and prepare guidelines for the top management for decision making during the ERP software selection process.

References

1. Boss, G. et al.: Cloud Computing: IBM White Paper, Version 1.0. Available via DIALOG: http://www.ibm.com (2007). Accessed 20 Apr 2011
2. Weinhardt, C., et al.: Business models in the service world. IT Prof. **1**(March/April), 36–41 (2009)
3. Chang, V. et al.: A categorization of cloud business models. In: 10th International Symposium on Cluster, Cloud and Grid Computing, Melbourne, May 2010

4. Ragusa, C., Puliafito, A.: Running business applications in the Cloud: a use case perspective. In: Guarracino, M.R. (ed.) Euro-Par 2010 Parallel Processing Workshops. Lecture Notes in Computer Science, vol. 6586, pp. 595–602. Springer, Berlin/Heidelberg (2011)
5. Laugesen, N.S.: E-Skills for ICT practitioners and entrepreneurs. Danish Technical Institute-First Interim Report. Available Via DIALOG: http://europeaneskills.eu (2011). Accessed 25 June 2012
6. Greggo, A. Cloud computing in the enterprise: an overview. In: Enterprise Computing Conference, New York, 21–23 June 2009
7. Haynie, M.: Enterprise cloud services: deriving business value from cloud computing. Micro Focus Technical Report. Available Via DIALOG: http://www.techrepublic.com (2009). Accessed 20 Apr 2010
8. Huang, K.W., Sundararajan, A.: Pricing models for on-demand computing. Working Paper CeDER-05-26, Center for Digital Economy Research, Stern School of Business, New York University (2005)
9. Konary, A. et al.: The future of software licensing under siege. IDC White paper. Available Via DIALOG: http://www.softsummit.com (2004). Accessed 10 Jan 2011
10. Hofmann, P., Woods, D.: Cloud Computing: the limits of public Clouds for business applications. Internet Comput. IEEE **14**, 90–93 (2010)
11. Kamhawi, E.M., Gunasekaran, A.: ERP systems implementation success factors: IS and non-IS managers' perceptions. Int. J. Bus. Inf. Syst. **4**, 688–704 (2009)
12. Kenji, E., Kushida, J.M., Zysman, J.: Diffusing the Cloud: Cloud Computing and Implications for Public Policy. J. Ind. Compet. Trade **11**, 209–237 (2011)
13. Kochut, A.: Evolution of the IBM Cloud: enabling an enterprise cloud services ecosystem. IBM J. Res. Dev. **55**, 7–13 (2011)
14. Nurmi, D. et al.: Eucalyptus: a technical report on an elastic utility computing architecture linking your programs to useful systems: UCSB Technical Report. Available Via DIALOG: http://cs.ucsb.edu/research (2008). Accessed 11 Feb 2012
15. Rappa, M.A.: The utility business model and the future of computing services. IBM Syst. J. **43**, 32–42 (2004)
16. Rochwerger, B., et al.: The reservoir model and architecture for open federated cloud computing. IBM J. Res. Dev. **53**, 11–19 (2009)
17. Shankar, R.: Ramco Cloud ERP Gartner Symposium. Available via DIALOG: http://www.ramco.com (2011). Accessed 25 April 2012
18. Wrycza, S.: Research in Systems Analysis and Design: Models and Methods. Lecture Notes in Business Information Processing, vol. 93, pp. 54–64. Springer, Berlin/Heidelberg (2011)
19. Sundararajan, A.: Nonlinear pricing of information goods. Manag. Sci. **50**, 12 (2004)
20. Suciu, G.: Cloud consulting: ERP and communication application integration in open source cloud systems. Proc. TELFOR Conf. **1**, 578–581 (2011)
21. Strong, D.M., Volkoff, O.: A roadmap for enterprise system implementation. Computer **1**, 22–28 (2004)
22. Theilmann, W., et al.: The ERP hosting use case scenario. In: Wieder, P., et al. (eds.) Service Level Agreements for Cloud Computing, pp. 295–309. Springer, New York (2011)
23. Vazquez, T., et al.: Evaluation of a utility computing model based on the federation of grid infrastructures. Euro-Par 2007 Parallel Process. **1**, 372–381 (2007)
24. Venketrama Raja, P.R.: Ramco's cloud ERP Product, Available via DIALOG http://www.thesmartceo.in/on-a-cloud-erp.html (2011). Accessed 18 April 2012
25. Wang, L., Kunze, M. et al.: Cloud Computing: a perspective study. In: Grid Computing Environments Workshop (GCE'08), Austin, 16 November 2008
26. Wang, Z.J., Xu, X.-F.: Business process integration point classification and the priority evaluation method. Int. J. Bus. Inf. Syst. **4**, 210–232 (2009)
27. Xu, X.: From cloud computing to cloud manufacturing. Robot. Comput. Integr. Manuf.. **28**, 75–86 (February 2012)
28. Wan, Y., Clegg, B.: Enterprise management and ERP development: case study of Zoomlion using the dynamic enterprise reference grid. In: Proceedings of International Conference-CENTERIS, Viana do Castelo, 20–22 October 2010

Chapter 10
Cloud Computing Solution Patterns: Infrastructural Solutions

Shyam Kumar Doddavula, Ira Agrawal, and Vikas Saxena

Abstract Cloud computing is an important emerging paradigm that is affecting the way enterprise IT is being managed. Cloud computing offers several useful features like high scalability, agility through elasticity, on-demand self-service, and pay-per-use models when consuming and delivering IT capabilities. There are several scenarios in enterprise IT context where solution architecture approaches leveraging cloud computing technologies offer a better solution than established traditional options. However, the architectural design and deployment approaches for the emerging solutions based on cloud computing paradigm are different from traditional approaches, so there is a need for a new set of solution architectural patterns and best practices. There are infrastructural layer solution patterns and application layer solution patterns addressing concerns at the corresponding layers. The infrastructure layer solution patterns deal with concerns like how to architect compute infrastructure that deals with unpredictable workloads while keeping the costs down and how to architect storage infrastructure that handles storage of large volumes of data. This chapter describes a number of common infrastructure layer scenarios and use cases, the limitations of traditional solutions, and the cloud computing-based infrastructural solution patterns that a software architect can leverage.

Keywords Cloud computing • Cloud computing patterns • Cloud infrastructure • IaaS • Cloud architecture • Cloud building blocks

S.K. Doddavula (✉) • I. Agrawal • V. Saxena
Cloud Computing CoE, Infosys Labs, Infosys Ltd., Bangalore, Karnataka, India
e-mail: shyamkumar.d@gmail.com; Ira_Agrawal@infosys.com;
vikas.saxena.2006@gmail.com

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_10,

10.1 Introduction

The National Institute of Standards and Technology (NIST) defines cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [1]. NIST further defines that the cloud model is composed of the following [1]:

- Five essential characteristics, namely, "on-demand self-service," "broad network access," "resource pooling," "rapid elasticity," and "measured service"
- Three cloud service models, namely, "Software as a Service (SaaS)," "Platform as a Service (PaaS)," and "Infrastructure as a Service (IaaS)"
- Four deployment models, namely, "private cloud," "community cloud," "public cloud," and "hybrid cloud"

Cloud computing technologies are causing disruptive changes in how IT resources and services across the infrastructure, platform, and application layers are getting provisioned and in the way they are getting consumed. IT consumers are looking toward a pay-per-use model in procuring computing resources to align the IT costs to consumption, thereby reducing inefficiencies. IT providers are expected to provide computing resources as services, on demand accessible from anywhere over a network. Further, these resources are expected to be scaled up and down with rapid elasticity to meet the changing needs. IT solution architects are also increasingly looking at leveraging these technologies and models while architecting solutions. This chapter also provides an overview of the key architectural building blocks of a typical cloud solution and focuses on the infrastructure layer concerns and cloud computing solutions relevant to this layer. A companion chapter titled "Cloud Computing Solution Patterns: Application and Platform Solutions" that also appears in this book focuses on the platform and application layer concerns and solutions.

Infrastructure solution architects typically face same problem scenarios [3] again and again in the context of several solutions like how to optimize server infrastructure, how to design a cost-effective and highly scalable storage layer, and how to address business continuity and disaster recovery. So, there is a need for reusable solution patterns [2, 11, 12] that describe the architectural solutions for these scenarios so that each solution does not need to reinvent the solution strategy. Patterns [13] can be defined at several levels, from business to architecture, design, and programming to deployment and runtime administration. They assist in the creation of artifacts according to best practices through all stages of the software development life cycle. Patterns with their specification and implementation help the creation of various artifacts needed for a software solution. Patterns provide a powerful way to improve IT solution development by identifying best practices and design expertise that can be captured and then made available for reuse. Some of the advantages of implementing the patterns are improved productivity, reduced development time,

minimized complexity, increased quality, improved governance, improved cost, etc. Solution patterns capture standardized, reusable solution architectures for various commonly occurring problem scenarios. This chapter describes a few commonly required infrastructural solution patterns. The chapter also provides a comparison of traditional approaches with the cloud-based solutions for the scenarios and highlights the benefits and limitations of approaches.

10.2 Cloud Computing Solution Patterns

10.2.1 Cloud Solution Architectural Building Blocks

Figure 10.1 gives a high-level view of some of the key architectural building blocks of cloud solutions.

It consists of the following layers of capabilities:

- Cloud infrastructure layer which provides the basic infrastructural capabilities
- Cloud platform layer that provides the next level specialized platforms and frameworks

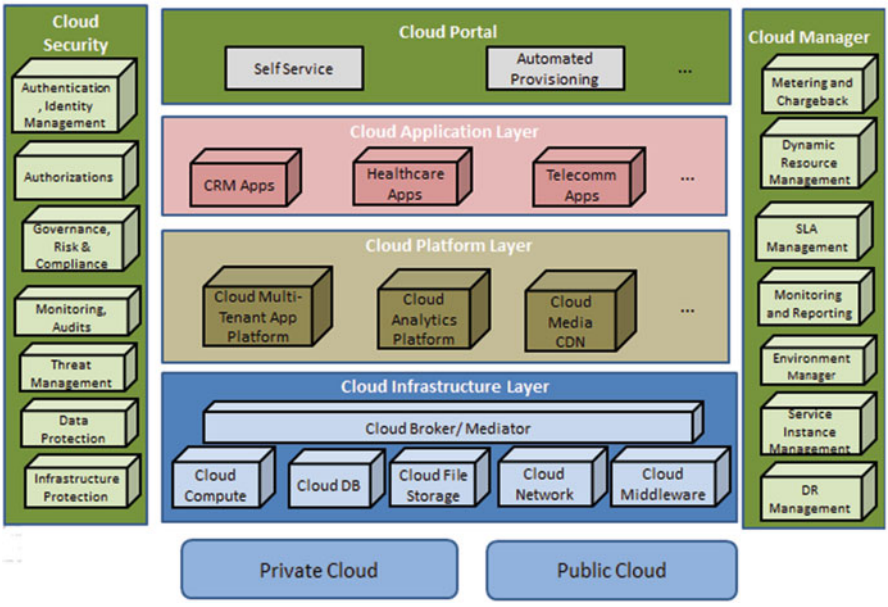


Fig. 10.1 Cloud solution architectural building blocks: high-level view

- Cloud portal, cloud manager, and cloud security layers which provide the crosscutting management and security components

A more detailed description of the various layers and building blocks is provided in the following subsections.

10.2.1.1 Cloud Infrastructure

This comprises the following infrastructural building blocks:

- *Cloud compute*: This provides a scalable and elastic compute infrastructure so that the processing layers of application can be developed over this component.
- *Cloud DB*: This provides a scalable and elastic data stores that can scale up and down to meet the application's storage needs. There are several implementation options for cloud data stores like document stores, key-value data stores, relational data stores, in-memory data grids, and graph data stores.
- *Cloud storage*: This provides a scalable file storage component that provides the foundation for several higher-level storage components. This is generally implemented as a distributed file system.
- *Cloud network*: This enables scalable network communication supporting different scenarios like Enterprise Application networks, Machine-to-Machine networks, and Compute grid networks.
- *Cloud middleware*: This provides messaging and other such integration enabler components which the layers above use for inter-component and inter-application collaborations.
- *Cloud broker/mediator*: This provides abstractions and inter-cloud integration components that enable easy migration of next set of layers across the various cloud implementations.

10.2.1.2 Cloud Platform

This comprises the platform building blocks like:

- *Cloud multi-tenant application platform*: This provides a scalable and elastic application development platform with features like dynamic auto-scaling application runtime environment and inbuilt multi-tenancy support that enables building scalable cloud applications without having to deal with such common concerns.
- *Cloud analytics platform*: This provides a scalable and elastic analytics platform that enables handle "Big data" addressing volume, velocity, and variety-related complexities.
- *Cloud media content delivery network* (*Cloud CDN*): This provides a scalable media content distribution platform leveraging the cloud storage and network infrastructure components.

10.2.1.3 Cloud Portal

This provides self-service and automated provisioning capabilities. This component enables cloud consumers to request and get the various cloud-based infrastructure, platform, and application components provisioned on demand. It also enables administration-, monitoring-, and management-related interactions.

10.2.1.4 Cloud Manager

This provides the various management capabilities addressing operational and support concerns like dynamic resource management for auto-scaling, service SLA management, metering and chargeback, monitoring and reports, instance management, and availability management.

10.2.1.5 Cloud Security

This provides the various capabilities that address security concerns like authentication and identity management, authorization, access controls and privilege management, governance and risk management, data protection, security of data in transit and at rest, threat and vulnerability management, and infrastructure protection.

10.2.2 Benefits and Limitations of Cloud Solutions

Some of the common benefits of using the cloud solutions include:

- Lower overall costs of infrastructure
  - Higher infrastructure utilization with resource pooling and sharing
  - Lower TCO for applications with variable workloads with dynamic scaling and pay-per-use models
  - Lower costs of business continuity and disaster recovery of IT infrastructure
- Improved service levels with self-service, elasticity, and automation
  - Better provisioning experience with self-service and automatic provisioning of applications, platforms, and infrastructure
  - Better application SLAs with dynamic resource management and auto-scaling
  - Faster time to market for business applications with faster provisioning of infrastructure and platforms
- Reduced maintenance and management concerns

- Higher scalability, elasticity, and lower administration overheads with abstraction of infrastructure and platform concerns

Some of the common limitations include:

- Security and compliance issues in deploying applications and data over infrastructure provided by third party.
- Lack of maturity, thereby making it difficult to migrate existing complex applications to cloud.
  - Applications may need re-engineering to be ported over to some cloud platforms.
  - Lack of advanced SQL support and ACID properties in NoSQL-based cloud data store options makes it difficult to migrate existing applications to cloud platforms.
  - It may be difficult to migrate specialized applications and platforms like grid computing applications.
- Scalability of private cloud deployment options may be limited.
- Use of virtualization may add performance overheads and also introduce limitations in supporting applications that cannot use virtualized infrastructure like real-time applications.

10.2.3 Pattern Documentation Structure

A solution architecture pattern captures standardized, reusable solution architectures for various commonly occurring problem scenarios. There are many different formats used for describing architecture patterns, but none of them has achieved widespread acceptance. The elements described below are some of the common aspects that can be found in most pattern definitions. So, these are the elements that are captured for the cloud solution architecture patterns in the subsequent sections.

Name
It represents a meaningful way to refer to a pattern. It is generally a single word or short phrase.

Intent
This section contains description of the problem indicating the intent in applying the pattern – the intended goals and objectives to be reached within the category and motivations described below.

Category
This section describes the context for which the solution is applicable.

Motivations
This section defines the various motivations and factors.

Solution
This section contains a description, using text and/or graphics for describing how to achieve the intended goals and objectives. The different approaches or implementation methods and their variants should be described in this section. The section also describes the key challenges associated with traditional solutions for the problem and how this solution meets up those challenges. The section also describes various components required to implement the solution.

Applicability
This section provides a description of scenarios and situations where this pattern can be applied.

Consequences
This section describes the implications, benefits, and limitations of applying the pattern.

Known Uses
This section describes known applications of the pattern. Known Uses can also serve as examples.

Related Patterns
This section describes the relationships between this pattern and others. The pattern may be inspired from other patterns. These may be predecessor patterns, whose resulting contexts correspond to the initial context of this one.

10.3 Common Use Cases and Cloud Solution Patterns

This section describes some of the common use cases, challenges with traditional solution options, cloud solution architecture, as well as benefits and limitations.

10.3.1 Server Infrastructure Optimization Through Enterprise Private Clouds

Name
Private cloud

Intent
The intent of the pattern is to optimize [4] server infrastructure used among the pool of applications with different workload peaks at different times.

Category
Deployment

Motivations

There are several scenarios in enterprises where applications require most of the resources only for a short duration of time, but the infrastructure is over provisioned to meet the peaks of such application. For example, a month-end processing application requires 50 CPU for the last 5 days of the month and requires 5 CPU for the rest of the month, but the 50 CPU infrastructure is provisioned and so the overall utilization of the resources is low.

Also, with traditional approach, in order to make sure that failure of such applications does not hamper other running applications, they are provisioned in separate set of physical servers; therefore, reallocation of unutilized resources to other applications when not needed is more difficult.

Cloud Solution

The solution is to create a cloud computing platform in the enterprise private internal network that enables pooling of available infrastructure resources and helps organization to leverage the existing infrastructure setup to provision the applications (along with all required underlying infrastructure, software platforms, etc.) on demand with reduced time and intelligent allocation of resources. This overcomes several challenges like security and compliance issues faced with public cloud adoption while leveraging existing investments. Enterprise private clouds [5] are seen as a natural progression of initiatives like virtualization of resources already taken up by several organizations. Enterprise private cloud solutions add capabilities like self-service and automation and charge back over the virtualized infrastructure. There are several implementation options like those based on VMware ESX, Citrix Xen, Citrix XenServer, Microsoft Hyper-V, Solaris zones, and IBM AIX WPARs and LPARs.

The conceptual view of the solution shown in Fig. 10.2 provides an overview of the service centric provisioning model with cloud computing.

The IT administrator uses a cloud portal created over a private cloud platform to populate a service catalog with the application(s) service templates capturing the commonly used and applicable deployment patterns in the enterprise. The solution architect then creates a service request using the service catalog based on a selected template in the cloud portal and deploys the application on the cloud infrastructure and makes it available for the end users to access them. Application developers use application development and testing tools to develop and test the applications. The operational support personnel use various tools to manage the applications over the cloud infrastructure. The cloud manager monitors the application usage and dynamically provisions additional infrastructure and scales the application when there is more usage and similarly de-provisions infrastructure when there is lower usage. It also meters the usage and charges the users based on it.

A key strategy is to dynamically reallocate the resources to different applications during their workload peak so that the infrastructure can be used efficiently. In traditional solution, resources are allocated to application with static binding which results in silos of infrastructure of each application. It also results in upfront estimation of the resources needed, and the consequent long hardware procurement cycles make deployment architecture changes difficult. The strategy with the cloud solution is to use virtualized infrastructure so that resources can be pooled and dynamically

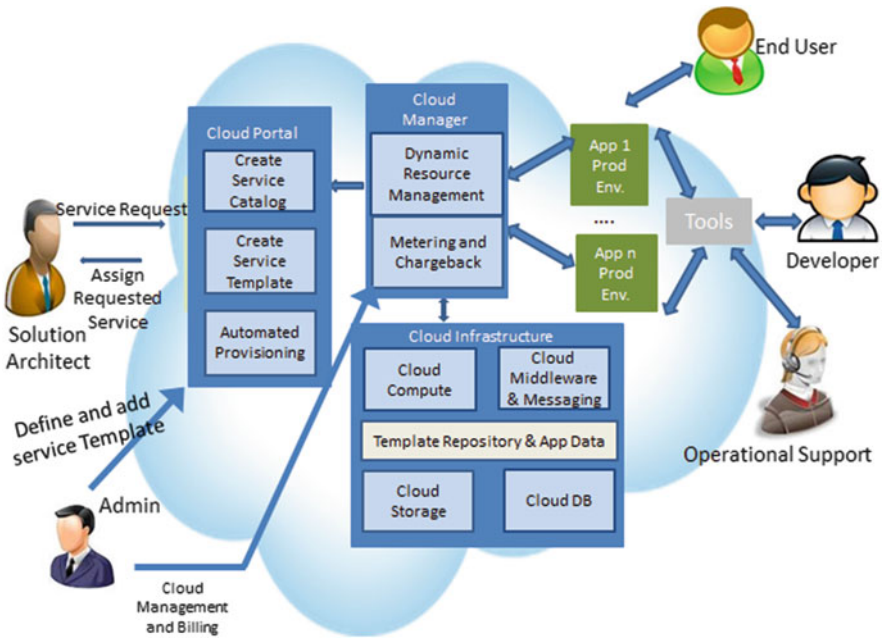


Fig. 10.2 Enterprise private cloud platform

reallocated based on current workloads and priorities of the various applications and also it enables on-demand self-service provisioning of infrastructure.

Some of the architectural building blocks needed to implement this pattern are:

- Cloud infrastructure:
  - Cloud DB: Elastic data stores that can scale up and down to meet the application's storage needs.
  - Cloud storage: Elastic storage for storing application-specific files including the application-specific template repository.
  - Compute cloud: The processing layer of the application is developed over this component.
  - Cloud middleware: Integration across application is achieved using the cloud middleware.
- Cloud manager:
  - Automated provisioning manager: For automating the provisioning of the cloud infrastructure and the application components
  - Cloud manager: For monitoring the application usage and dynamically scaling the infrastructure up and down and also to meter the usage and charge the users based on it
  - Cloud portal: For self-service provisioning of the cloud infrastructure and to define the service catalog and template specific to the application(s)

Applicability
This pattern is applicable to the scenarios in large enterprises with a big pool of applications with varying workloads deployed on existing infrastructure. In such scenarios, there will be existing investments that the enterprises will want to leverage while improving the overall utilization of the infrastructure using private clouds. Enterprises operating in domains with constant change, which need agile infrastructure to be able to reallocate them as per changing business needs, will also find this solution architecture relevant.

Consequences
With a private cloud solution, economies of scale possible within an enterprise across the various lines of business are utilized by pooling the infrastructure in the enterprise, so the consequences are that it works better in organizations which have sufficient scale, and it also means that the scalability of the cloud is limited to the scale of the single enterprise. So, the scalability of private cloud solutions is limited to the resources pooled unlike in public cloud which can have higher scalability. Private clouds are successful only when the infrastructure across the various lines of business is pooled; this requires proper governance and sharing policies so that the priorities of the various lines of business are balanced with that of savings costs across the organization. Most of the private cloud solutions available to implement need virtualization, so the applications that cannot use virtualized infrastructure like real-time applications and those that need specialized hardware cannot be provisioned over cloud infrastructure. The virtualized infrastructure may introduce performance overheads.

Known Uses
Private cloud infrastructures are majorly built for enhancing the in-house capabilities of the organizations. They are mainly customized implementations on virtualization software like VMware ESX, Citrix XenServer, and Microsoft Hyper-V. Terremark also offers private cloud infrastructure for the enterprises.

Related Pattern
Cloud burst

10.3.2 Scalable and Low-Cost Storage, Backup, and Archival: Internet Scale Content Distribution Solutions

Name
Cloud storage

Intent
The intent of the pattern is to provide a scalable and low-cost storage solution for the increasing unstructured data generated from different sources like social media and rich media data like video, from the sensor devices. The solution should also provide the efficient backup and archival mechanism for such huge data.

Category
Storage

Motivation
With increasing adoption of social media by consumers, several enterprises have been creating their own social media platforms that enable their customers form online communities, use online discussion forums, and provide reviews and ratings of the services and solutions. This is resulting in large amounts of data that needs to be stored.

Use of rich media for better user experience is common these days, so there are scenarios where an enterprise wants a web-based portal for its applications. This portal is content heavy as it uses a lot of multimedia content like a rich promotion campaign with a lot of video and image files. So, this huge amount of data that is generated needs to be stored and its backup and archival also needs to be maintained. As the amount of information collected is growing, the backup of such information is also growing. Using traditional storage solutions involve a lot of CAPEX and a lot of administration overheads. Traditional solutions are better suited for scenarios where the value per byte of information is high like transactional data, but for scenarios like social media data where the value per byte is not well defined but can be a potential source of insights and competitive differentiation, there is a need for alternative solutions. Another aspect is lack of well-defined structure for these new kinds of information requiring solutions based on alternative architectures that are better suited for these.

Cloud Solution
The solution is to leverage several cloud storage [6] and implementation options based on private and public cloud deployment models which will provide a scalable and low-cost storage. There are public cloud storage solutions like Amazon S3 and Azure Blob Storage that provide a scalable solution for storing files through a public API (REST and WebServices). These solutions provide highly scalable storage with a pay-per-use model for storage and retrieval.

There are several content delivery network solutions available, e.g., CloudFront from Amazon, Windows Azure CDN, and CloudFlare that provide the ability to syndicate content to multiple servers across the globe to enable faster response times in content downloads. Some of these are integrated with the cloud storage solutions, thus making it easier for developing content-centric web applications.

There are several public cloud database solutions, like Amazon Simple DB, SQL Azure, and Google Big Table, which provide scalable and on-demand object storage solutions with public API. There are also NoSQL solutions like HBase, MongoDB, Cassandra, and CouchDB that enable implementing scalable in-house data stores leveraging commodity hardware.

There are also open-source solutions like Hadoop Distributed File System and Commercial Off-The-Shelf (COTS) storage solutions from vendors like Nirvanix and Parascale that enable creating scalable file storage based on commodity hardware within the enterprise network. These use technologies similar to those used by the public cloud storage solutions to provide a scalable solution.

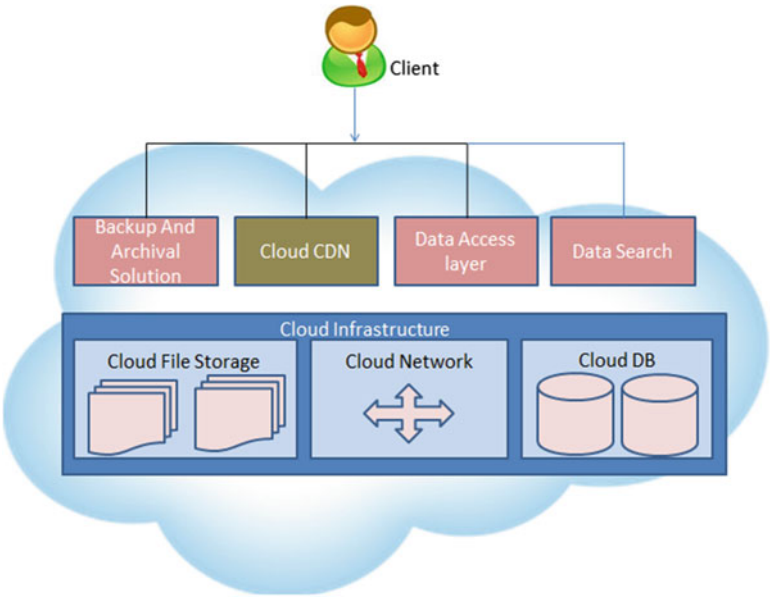


Fig. 10.3 Cloud storage, archival, content delivery solutions

With disk slowly becoming the next tape and memory becoming the next disk due to advances in technologies and falling prices, there are several in-memory storage solutions emerging. There are open-source distributed cache solutions like memcached and COTS from vendors like Oracle (Coherence), GemFire, Terracotta, GigaSpaces, and SAP (HANA) that provide scalable in-memory data stores.

The conceptual view of solution to address storage needs based on these technologies is shown in Fig. 10.3.

A scalable storage infrastructure layer is created leveraging public cloud file storage, public cloud data stores, in-house NoSQL data stores, in-memory data stores, and distributed file system-based solutions. A data access and search layer is created which provides abstractions for the various logical categories of data stores (like File Store, Document Store, Key-Value Store, Column Store, Graph Store) hiding the implementation and deployment details of the underlying infrastructure layer. The data access layer is then leveraged by the platform and application layers. Specialized frameworks like "backup and archival," "content delivery network (CDN)" based on the underlying cloud storage infrastructure layer, addressing the specific needs of a particular use case, are then developed or bought/leveraged if COTS solutions are available.

Here, a key challenge is that huge amount of data needs to be handled, and existing traditional storage solutions are too expensive and cannot scale. The cost of maintenance of such high-end storage and network infrastructure is also very high. So, the strategy with the cloud solution is to provide a horizontally scalable solution

with the commodity hardware. Public cloud deployment models can help if there is a need to convert from CAPEX- to OPEX-based financial model.

Some of the architectural building blocks needed to implement this solution are:

- Cloud infrastructure:
  - Cloud DB: Horizontally scalable and elastic data stores based on commodity hardware that can scale up and down to meet the application's storage needs
  - Cloud storage: Elastic storage for storing application-specific files
- Cloud CDN: These help address media content distribution use cases.

Applicability
The pattern is applicable for the scenarios where huge volumes of data where the cost per unit of data is low. Such scenarios need a storage solution that should be scalable with low cost while providing acceptable response time.

Consequences
Data leaves the organization boundary for public cloud storage options, and consequently data protection and regulatory compliance issues need to be dealt with though there will be benefits of higher scalability and probably lower costs too.

Several of these cloud storage and database solutions are still evolving so they may lack features like strict ACID properties and extensive support of SQL. Therefore, migration of existing complex application over to cloud infrastructure may not be straightforward.

As per the CAP (aka Brewer's) theorem, it is not possible to provide consistency, availability, and partition tolerance simultaneously. Several of these NoSQL storage solutions may have made a choice of guaranteeing one or two of these architectural concerns and so may suit only those use cases where the choice made is appropriate.

Known Uses
Google Drive, Dropbox, and Microsoft's SkyDrive are a few popular cloud-based storage solutions.

Related Pattern
Big data analytics

10.3.3 Handling Workload Spikes with Cloud Burst

Name
Cloud burst

Intent
The intent of the pattern is to provide a cost-effective solution that enables an application to meet minimum service levels during unexpected workload spikes.

Category
Deployment, application

Motivations

There are scenarios in enterprises where there are existing applications deployed over existing infrastructure for which investments have already been done. Existing infrastructure is able to handle normal workloads but is unable to meet the occasional spikes in workloads with acceptable service levels, for example, an e-commerce portal that receives a lot of requests after a marketing campaign.

In the traditional solution, there will always be a trade-off between meeting the service levels and keeping a lot of infrastructure idle with over provisioning. Like if we have the infrastructure for normal workload, the SLAs will be met at normal times only, but during the peak times the SLAs will not be met. If we design the infrastructure to handle peak workload, then most of the time the resources will lie idle.

Cloud Solution

The solution strategy is to make use of public cloud platforms to handle the workload bursts while leveraging in-house private clouds for the normal workloads. Cloud bursting [8] refers to the ability to get infrastructure at runtime from a cloud provider and enabling the applications to use that to meet the service levels during a temporary spike in the workload.

There are solutions that allow such seamless migration from private cloud setup to a public cloud during temporary workload spikes to maintain the SLAs like vCloud Express which is based on VMware technologies that are designed to enable seamless migration of workloads from internal VMware cloud to a public cloud provider supporting vCloud API. Similarly, there are other hybrid cloud solutions from vendors like CloudSwitch and Elastra that are designed for these use cases. Hybrid cloud solutions enable organizations to leverage their existing investments in the application infrastructure for normal workloads while at the same time meeting the service levels by paying for additional infrastructure only when needed.

Figure 10.4 provides a conceptual view of the cloud burst solution pattern.

In Fig. 10.4, the application seamlessly bursts over onto a public cloud using a cloud mediator. The cloud manager will take care of auto-provisioning of application through cloud portal, meters the usage, and bills the customers accordingly.

One of the key challenges is to maintain the SLAs in case of temporary spikes in the workload. So, the cloud solution strategy here will be to have a hybrid cloud setup with application on private cloud and burst to public cloud to handle spikes. It is basically about getting the extra infrastructure from the public cloud provider to handle the workload thereby maintaining the SLAs and reducing the costs.

Some of the architectural building blocks needed to implement this are:

- Cloud infrastructure:
  - Cloud compute and cloud DB: Elastic compute platform and the data stores that can scale up and down to meet the application's processing and storage needs
- Cloud manager
  - Automated provisioning manager: For automating the provisioning of the cloud infrastructure and the application components
  - Cloud manager: For monitoring the application usage and provisioning infrastructure from the cloud to handle the workload burst

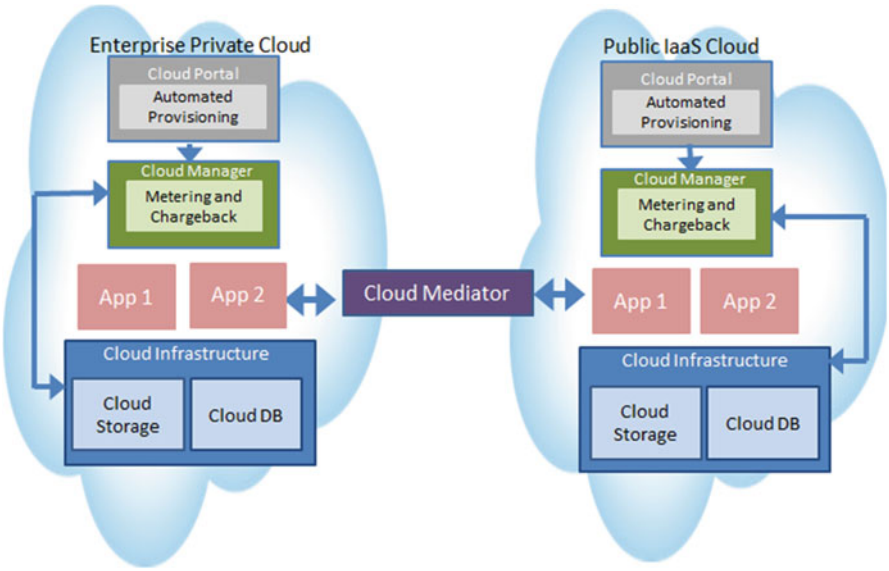


Fig. 10.4 Hybrid cloud solutions for "cloud burst"

- Cloud mediator: This component enables seamless migration of applications from private to public cloud setup.

Applicability
The pattern is applicable for the scenarios that involve an application that experiences occasional workload bursts.

Consequences
This solution architecture involves use of public cloud infrastructure so there will be associated benefits and issues. These solutions typically work only if bottleneck is compute capacity, but in several scenarios the bottleneck may not just be the compute platform and can be other components of the application architecture like the database. It is harder to make a typical commercial database leverage additional infrastructure from a public cloud and scale as they are typically designed to scale up vertically than scale up horizontally. Also, migration of applications over public cloud infrastructure may result in security and compliance issue. Many of these solutions are still not mature enough to handle complex application.

Known Uses
WebSphere CloudBurst and IBM CloudBurst, are the known implementations for handling cloud bursts.

Related Pattern
Public cloud applications, private cloud

10.3.4 Cloud-Based On-Demand Development and Testing

Name
Test and development clouds

Intent
The intent of the pattern is to enable on-demand development and test platforms.

Category
Structural

Motivations
During the software/application development process, creation of separate development (Dev) and test environments is a very common practice. Nearly 30–50 % servers in a typical large enterprise are dedicated for such development and test environment, and these environments are idle for most of the time as they are used only during the application software releases. Each application may have its own test and development environment. Such silos of development and test environment for each application may result in underutilization of hardware resources.

It may also be possible that an application may require multiple test environments. In such cases if the infrastructure sizing is done for the peak, there will be a significant underutilization of the infrastructure. Moreover, the setting up and maintenance of these environments is not an easy task. It requires a lot of effort and time to set up these environments. Because of the complexities involved, setting up test environments results in long provisioning times, thereby delaying projects.

Organizations are also facing challenges in creating test environments for Internet applications at the scale that matches production setup, so usually the test environment may not be at the same scale as the production environment because of which the testing may not reflect a real production scenario and so may not be adequate.

Cloud Solution
The solution is to leverage an on-demand development and test platforms based on cloud infrastructure [7] on pay-per-use model that can be provisioned as needed and decommissioned when they are not needed. These solutions enable creation of test and development environments on demand through automation and self-service. Multiple deployment models based on combinations of public and private clouds are possible for the testing tools and environments.

The conceptual view of the solution is shown in Fig. 10.5.

Templates of the test and development environments of the applications are created using virtual machine snapshots and are then used to provision the development and test environments on demand and made available to the developers and testers. Testing tools can be provisioned on demand on the cloud infrastructure itself which are used for manual or automated testing of the applications. The cloud portal provides the self-service portal and the catalog with the environment templates.

A key challenge is to provision a complete environment with a single click. The challenge here is to provision all the elements of the environment such as

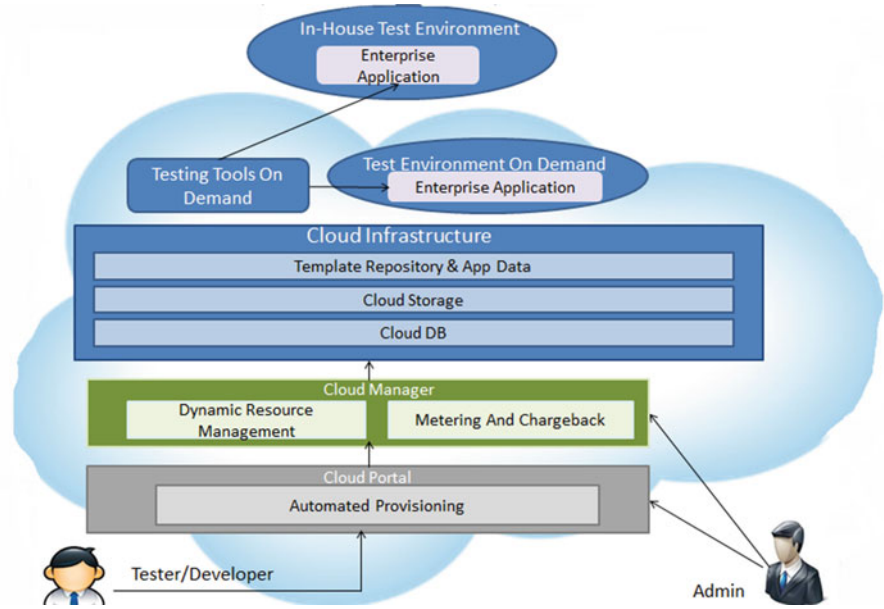


Fig. 10.5 Testing on cloud

web servers, application servers, database servers, development tools, and testing tools with one click. This abstraction of the complete environment as a single service instance enables easier management as the system can track the various infrastructure and software components provisioned as part of an environment instance that has been provisioned.

The various architectural components of this solution are:

- Cloud infrastructure:
  - Cloud storage and DB: To provide the storage needs of the test and development environments including test data and virtual machine snapshots
- Cloud manager
  - Automated provisioning manager: For automating the provisioning of the cloud infrastructure, the application components, testing, and development tools
  - Cloud manager: For monitoring and management, metering and billing, etc.
  - Cloud portal: For self-service provisioning of the cloud infrastructure

Applicability

The pattern is applicable for large enterprises with a lot of software application development and releases. Such enterprises can benefit from being able to provision and de-provision test and development environments on demand.

Consequences
Provisioning virtualized test environments can cause a problem if the applications use non-virtualized infrastructure in production environment as the use of virtualized environment for testing may not uncover all potential problems in production.
Also, security and compliance issues for public cloud-based deployment models still remain open for this solution as well.

Known Uses
PushToTest is an example of a public cloud-based testing tool. Test and development environment provisioning is one of the most common use cases for public cloud usage.

Related Pattern
High-performance computing cloud, cloud desktops

10.3.5 Business Continuity with Cloud Infrastructure

Name
Disaster recovery in cloud

Intent
The intent of the pattern is to provide a cost-effective disaster recovery solution leveraging cloud infrastructure.

Category
Behavioral

Motivations
Enterprises which adhere to regulatory compliances such as PCI DSS, HIPPA, and SOX generally have a DR site which is an exact replication of the production site. The traditional method of setting up and maintaining a DR site involves maintaining a replica of data center with the servers, storage, and networking in a different place. This redundancy incurs high costs.

Also, testing the DR process each time for any change in the production environment is a complex and costly task.

Cloud Solution
An alternative to the traditional approach is to leverage cloud infrastructure for business continuity/disaster recovery [9]. The production environment is virtualized; after that the servers are backed up as VM templates on cloud storage and provisioned on demand during disaster recovery from these templates, thus reducing dependency on the physical servers and enabling automation in provisioning the servers and the applications making the DR process simpler than it is otherwise with traditional solution using physical servers and applications. The pay-per-use model reduces capital expenditure in maintaining the infrastructure. The cloud infrastructure makes the solution more scalable.

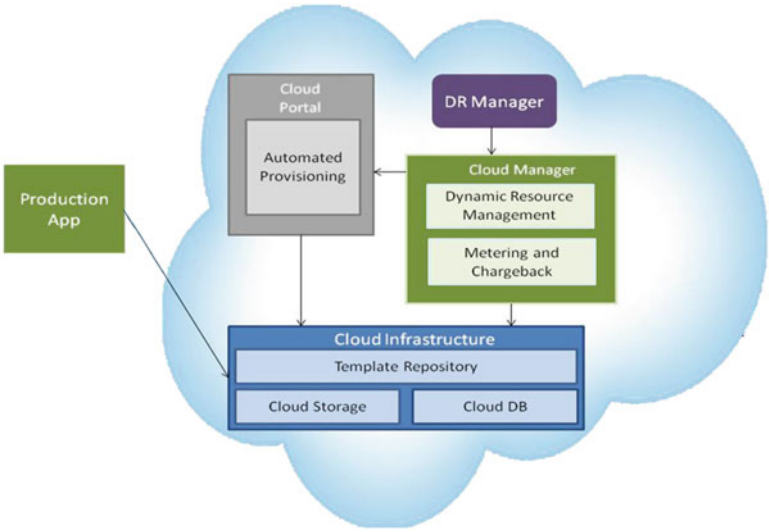


Fig. 10.6 Business continuity using cloud infrastructure

The conceptual view of the solution pattern is shown in Fig. 10.6.

The production servers are virtualized, and periodically virtual machine snapshots are taken of the production servers. These snapshots are transferred and stored on a cloud storage, and when needed the servers are recreated over the cloud using those snapshots. Similarly replicas of data stores and file systems are also created and stored on cloud storage. They are then used during disaster scenarios to restore the data. Cloud providers generally provide multiple data centers spread geographically across multiple locations, thus providing high availability. By leveraging such infrastructure, the disaster recovery costs are reduced.

Cloud providers offer only standardized infrastructure so there will be limitations. Security and regulatory concerns with public cloud need to be addressed to be able to use this effectively. There can also be issues with transferring large amounts of data which may affect the recovery point objective (RPO) and recovery time objectives (RTO) of the DR solution using this approach.

Some of the architectural building blocks needed to implement this are:

- Cloud infrastructure:
  - Cloud storage and DB: For taking the backups of the virtual machines and data
- Cloud manager
  - Automated provisioning manager: For automating the provisioning of the applications during recovery
  - Cloud manager: For monitoring and managing the cloud infrastructure and also for metering and chargebacks
  - Cloud portal: For self-service provisioning of the DR site and applications

Applicability
The pattern is applicable for scenarios where DR is needed for simple applications, especially for small and medium businesses (SMBs) who cannot afford the costs of a redundant DR site.

Consequences
There can be performance issues with replication and backups over a WAN which may make it technically difficult to meet some of the expected RPO and RTO service levels in some cases.

Known Uses
Dell AppAssure's Cloud Backup and Disaster Recovery and Citrix Cloud Solution for Disaster Recovery are examples of such cloud-based business continuity solutions.

Related Pattern
Private cloud, public cloud applications

10.3.6 Desktop as a Service Using Cloud Infrastructure

Name
Cloud desktops

Intent
The intent of the pattern is to provide low-cost solution providing desktops on demand with appropriate work environments.

Category
Deployment

Motivations
A large portion of IT resources is spent in managing and operating desktop environments for employees. There are several complexities involved in maintaining such an environment like periodic technology refresh, providing patches and updates, providing disaster recovery and backups, and support for employee mobility. Traditional approach of procuring physical desktops and using them involves several complexities because of the physical distribution of the desktops across various locations and incurs high costs.

Another common scenario is that of a large enterprise starting a new branch in a different country. The complexities of managing physical desktops and local compliance requirements make it highly cost prohibitive to manage employee desktops at remote locations.

Cloud Solution
The solution is to set up a desktop cloud to provision remote desktops on cloud infrastructure [10]. A desktop cloud solution helps to provide "anytime, anywhere" access to applications, information, and resources, thereby improving

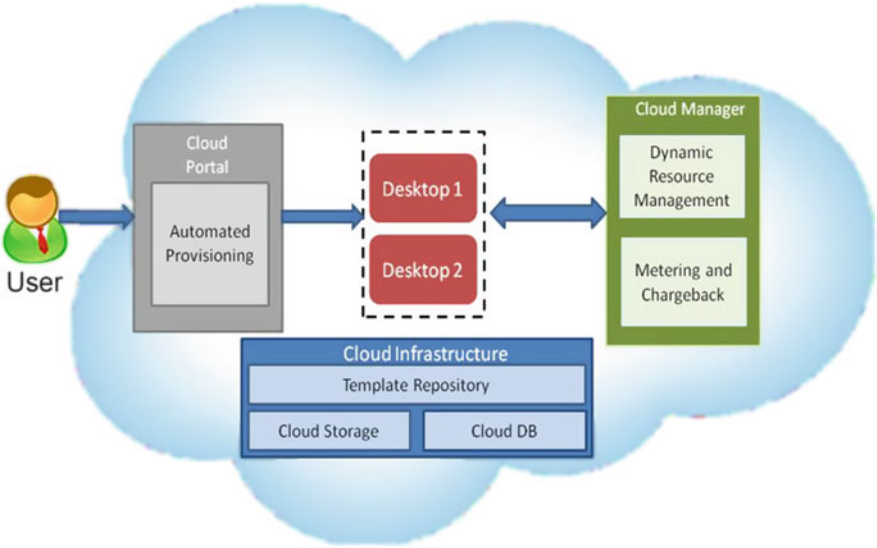


Fig. 10.7 Desktop cloud solution

employee productivity. These solutions generally enable thin clients or any other Internet-connected device to access virtualized desktops with hosted applications. Organizations are looking to use these solutions to avoid the costs associated with constant refresh of desktop hardware. There are also multiple deployment models with private cloud being a popular choice. Refer to Fig. 10.7.

Cloud infrastructure is used to provision the remote desktops for users/employees. The cloud manager monitors the application usage and dynamically provisions additional infrastructure and scales the number of desktops required based on the usage and similarly de-provisions infrastructure when there is lower usage. This significantly cuts the expenditure on infrastructure. The cloud portal can be used by the user to provision desktops and storage on demand.

A key challenge here is to secure the data and desktops on cloud. There are several cloud security solutions like virtual private clouds (e.g., Amazon AWS VPC), disk encryption software, and single sign-on and identity management solutions that can be leveraged to address several security concerns.

Another key challenge is addressing performance and latency issues, especially if the users have to access the resources over a WAN.

Some of the architectural building blocks needed to implement this solution are:

- Cloud infrastructure:
  - Compute cloud: For the desktop computing needs
  - Cloud storage: For storing application-specific files and repository of VM templates

- Cloud manager
  - Auto-provisioning manager: For automating the provisioning of remote desktops and applications
  - Cloud manager: For monitoring the individual desktop resource usage and dynamically scaling the infrastructure up and down
  - Cloud portal: For self-service provisioning of the desktops

Applicability
This pattern is applicable for large organizations having a large number of employees. The solution not only results in significant cost cutting but also provides scalability of infrastructure.

Consequences
This solution results in centralization of the computing resources of all desktops across the organization, which results in reduced complexity in managing them ensuring timely updates, security, easier employee mobility, etc. The flip side is that there can be loss of employee productivity if there are latency and performance issues.

Known Uses

IBM Smart Business Desktop Cloud is an example of an implementation of this pattern.

Related Pattern
High-performance computing cloud

10.4 Conclusion

This chapter has provided an overview of the architectural building blocks of a typical cloud solution and then described a few solution patterns elaborating on common problem scenarios that enterprise solution architects face at the infrastructure layer while developing enterprise solutions, the traditional solution approaches and their limitations, and cloud computing solution approaches and their benefits and limitations. Enterprise infrastructure solution architects can benefit from reusing these solution patterns. As the technologies evolve, the solution strategies also need to evolve. So, these should be treated as starting points for capturing these evolving solutions.

References

1. Mell, P., Grance, T.: The NSIT definition of cloud computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf (2011)
2. Guest, S.: 5 cloud computing patterns along with examples and Azure solutions. http://www.infoq.com/presentations/Patterns-for-Cloud-Computing (2010)

3. Amrhein, D., Anderson, P.: Cloud computing use cases. http://www.opencloudmanifesto.org/Cloud_Computing_Use_Cases_Whitepaper-2_0.pdf (2009)
4. CA Technologies: IT optimization through workload automation. http://whitepapers.businessweek.com/detail/RES/1335968686_109.html (2012)
5. Doddavula, S., Wasudeo, A.: Adopting cloud computing – enterprise private clouds: http://www.infosys.com/infosys-labs/publications/infosyslabs-briefings/documents/cloud-computing-enterprise-private-clouds.pdf (2009)
6. Kim, S.-G., Han, H., Eom, H., Yeom, H.Y.: Toward a cost-effective cloud storage service. In: Advanced Communication Technology (ICACT), The 12th International Conference. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5440497&isnumber=5440112
7. Tilley, S., Parveen, T.: Migrating software testing to the cloud. In: IEEE International Conference on Software Maintenance (ICSM). http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5610422&isnumber=5609528 (2010)
8. Bicer, T., Chiu, D., Agrawal, G.: A framework for data-intensive computing with cloud bursting. In: IEEE International Conference on Cluster Computing (CLUSTER). http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5610422&isnumber=5609528 (2011)
9. Chris, P.: Business continuity with cloud. http://www.eweek.com/c/a/Cloud-Computing/How-to-Ensure-Business-Continuity-with-Cloud-Computing/ (2010)
10. Carta, D.P.: Your office in the clouds: the best online virtual desktops. http://gigaom.com/collaboration/your-office-in-the-clouds-the-best-online-virtual-desktops/ (2009)
11. Fehling, C., Retter, R.: Cloud computing patterns. http://cloudcomputingpatterns.org/ (2011)
12. Ho, R.: Cloud computing patterns. http://architects.dzone.com/news/cloud-computing-patterns (2009)
13. IBM: Pattern solutions. http://www.ibm.com/developerworks/rational/products/patternsolutions/ (2011)

Chapter 11
Cloud Computing Solution Patterns: Application and Platform Solutions

Shyam Kumar Doddavula, Ira Agrawal, and Vikas Saxena

Abstract With the convergence of several trends such as digitization of the value chains of several industries, increasing adoption of social media by Internet users, and pervasiveness of computing with processing in systems like automobiles and household appliances, there are new business and IT models emerging, including application marketplaces, industry business clouds, and Software as a Service. Also, there are new opportunities and technology challenges emerging, e.g., data deluge caused by data doubling every 2 years, with much more unstructured data being generated than structured data. With this background, the traditional architectural solutions and technologies are unable to meet the requirements of scenarios like application marketplaces involving hundreds of thousands of applications, e.g., in a mobile application store with millions of low-value transactions that are highly unpredictable. To this end, architectural solutions leveraging cloud platforms [1] are being evaluated to meet such requirements. Also, traditional data storage, search, and processing technologies (e.g., relational databases) are unable to scale economically to meet these increasing data volumes and varieties at the expected velocity of response times. Internet companies like Google, Yahoo, and Facebook that have been dealing with such extreme data requirements have developed Big data solutions and technologies like NoSQL, Distributed File Systems, and MapReduce frameworks that can leverage commodity hardware and provide the needed scalability and fault tolerance. These architectural solutions and technologies are now being made available for solution architects through open source, commercial technology, and cloud providers. There are reusable architectural solution patterns at infrastructure and software platform/application layers. This chapter describes some of the common emerging business scenarios and architectural solution patterns at platform and application layers leveraging the cloud [1] and Big data technologies.

S.K. Doddavula (✉) • I. Agrawal • V. Saxena
Cloud Computing CoE, Infosys Labs, Infosys Ltd., Bangalore, Karnataka, India
e-mail: shyamkumar.d@gmail.com; Ira_Agrawal@infosys.com;
vikas.saxena.2006@gmail.com

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_11,

Keywords Cloud solution patterns • Cloud computing patterns • Cloud infrastructure • SaaS • PaaS • Cloud architecture • Cloud building blocks

11.1 Introduction

With the increasing adoption of social media by Internet users, there is a lot more user-generated content. More and more devices are getting instrumented and getting connected to the Internet. These devices are generating huge amounts of data. As a result, there is a data deluge and the data that enterprise IT systems need to deal with is increasingly becoming more and more semi-structured and unstructured. Traditional enterprise information and data management solutions and technologies are not cost effective and not scalable to deal with this deluge of unstructured "Big data."

The value chains of several industries are getting digitized. For example, e-books and online publishing are dramatically changing the business models of the publishing and media industry. Online commerce and social recommendations driven buying behavior and digitized and instrumented supply chain systems are similarly disrupting retail business models, bringing in new players. Telecom industry is similarly being disrupted with increasing adoption of smartphones, the smartphone platform (e.g., iPhone and Android) ecosystems, and application marketplaces. Enterprise software platforms and applications are also seeing dramatic changes to accommodate these changing business needs.

Another companion chapter in this book (Sect. 10.1.1) provides an overview of the architectural building blocks of a typical cloud solution and describes the *infrastructure* layer solutions. The current chapter discusses a number of commonly occurring architectural problem scenarios [3] faces by enterprise solution architects in the *platform* and *application* layers and describes how cloud computing technologies can be leveraged to build cost-effective and scalable solutions that meet the next-generation IT requirements. Readers are also advised to refer to Sect. 10.2 to further understand some of the common infrastructure-level scenarios and solution patterns while referring to this chapter for the application layer scenarios and solutions.

11.2 Common Use Cases and Cloud Solution Patterns

This section describes some of the common use cases [3], their challenges with traditional solution options, cloud solution architecture, and its benefits and limitations, but first we provide the structure of the solution patterns presented in the chapter.

11.2.1 Pattern Documentation Structure

A solution pattern captures standardized, reusable solution architectures for various commonly occurring problem scenarios. There are several different formats for describing, and the elements described below are some of the common aspects that can be found in most pattern definitions. So, these are the elements that are captured for the cloud solution patterns [2, 11, 12] in the subsequent sections. Although the structure is fully described in another chapter in this book (Sects. 10.1.2 and 10.1.3), we briefly mention it here, for the sake of completeness.

Name
Represents a meaningful way to refer to a pattern.

Intent
This section contains description of the problem indicating the intent, the intended goals, and objectives.

Category
Describes the context for which the solution is applicable.

Motivations
Defines the underlying motivations and factors.

Solution
This section contains a description of how to achieve the intended goals and objectives. The different approaches or implementation methods and their variants as well as key challenges are also mentioned.

Applicability
Provides a description of scenarios where the pattern can be applied.

Consequences
This section describes the implications, benefits, and limitations.

Known Uses
Describes known applications using examples.

Related Patterns
This section names other related patterns.

11.2.2 Applications with Highly Variable Workloads in Public Clouds

Name
Public cloud applications

Intent
The intent of the pattern is optimizing [4] costs for applications with one-off and highly variable usage.

Category
Deployment and application

Motivations
There are several scenarios in large enterprises where there are applications which are used only occasionally like, say, an application that is meant for a particular marketing event or an application that generates a quarter-end report. Procuring infrastructure and maintaining it for the period when the application does not get used can lead to inefficiencies and unnecessary expenses.

Also, there are some applications whose usage varies a lot, so sizing the infrastructure and provisioning right infrastructure for these applications is a challenge. If the sizing is done for their peak usage to be able to meet the performance requirements, then there will be significant underutilization of the resources. There are also scenarios where enterprises are looking to try out and launch an experimental innovative solution. There is not much visibility into how many users are likely to sign up for the service. Using traditional models will result in significant capital expense (CAPEX), and that limits the number of such experiments that can be taken up.

Cloud Solution
The solution is to leverage a public cloud platform to deploy such applications with highly variable usage instead of procuring the hardware and deploying it in enterprise data centers. There are several public cloud platforms that offer pay-per-use model with on-demand provisioning and auto-scaling of infrastructure based on usage. These solutions enable optimizing infrastructure usage to reduce costs and also enable scaling up easily to meet any surges in application usage without affecting response times. The conceptual view of the solution is shown in Fig. 11.1.

An application architect designs and develops the application leveraging a public cloud platform. The IT administrator uses a Cloud Portal provided by a public cloud provider to provision the infrastructure needed for the application. The administrator then deploys the application on the cloud infrastructure and makes it available for the end users to access it. The Cloud Manager monitors the application usage, dynamically provisions additional infrastructure, and scales the application when there is more usage and similarly de-provisions infrastructure when there is lower usage.

A key challenge is meeting the application's SLAs even though the workload is highly variable. In traditional approach, the infrastructure provisioned is static and so is either over provisioned or cannot meet the varying workloads requirements. The strategy with the cloud solution is to use "elastic infrastructure" which can be provisioned on demand and implement dynamic resource scaling to change infrastructure provisioned based on current workload [5].

Some of the architectural building blocks that provide the core capabilities needed for this are:

- Cloud Infrastructure:
  - Cloud DB: Elastic data stores that can scale up and down to meet the application's storage needs.
  - Cloud Storage: Elastic storage for storing application-specific files.

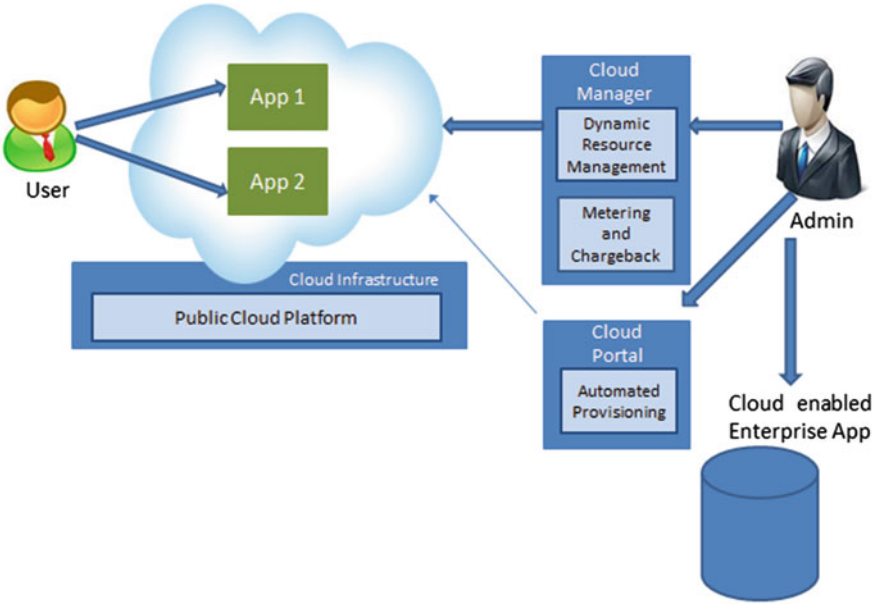


Fig. 11.1 Applications on cloud platforms

- Compute Cloud: The processing layer of the application is developed over this component.
- Cloud Middleware: Integrations across application are achieved using the Cloud Middleware.

- Cloud Manager and Portal:
  - Automated Provisioning Manager: For automating the provisioning of the cloud infrastructure and the application components
  - Cloud Manager: For monitoring the application usage and dynamically scaling the infrastructure up and down
  - Cloud Portal: For self-service provisioning of the cloud infrastructure

- Cloud Security:
  - Cloud Security: All the cloud security components like authentication, authorization, and data protection

There are multiple cloud technology options for this, e.g., (1) Public Infrastructure as a Service (IaaS) clouds and (2) Public Platform as a Service (PaaS) clouds.

Implementation Option 1: Public IaaS Cloud
One of the options is IaaS public clouds like Amazon AWS, Rackspace, Terremark, and Savvis that offer infrastructure on pay per use. These solutions provide the infrastructure, and migration to this model for an existing application involves deploying the application on the cloud infrastructure. This generally may not need lot of changes to the application components if it is an existing application. New

applications can be designed to take advantage of the scalable cloud storage, compute, and middleware infrastructure components.

Implementation Option 2: Public PaaS Cloud
Another option is PaaS public clouds like Microsoft Azure, Google App Engine, and Force.com. These offer a highly scalable application runtime platform like a .NET technologies-based platform in the case of Azure and a Python or Java-based platform in the case of Google App Engine.

Applicability
The pattern is applicable for the scenarios where dynamic scaling of applications is required based on the load experienced by the application or the CPU/memory utilization of the instances. Additionally, the pattern can be used in scenarios where a new innovation is to be tested but the traffic to be experience by the web application is not known. In such a case, dynamic scaling can help a lot in terms of capital expenditure involved.

Consequences
One of the consequences of this solution architecture is the attackable surface of the application increases when it is in a public cloud instead of the traditional model of deploying the application in an enterprise data center as the infrastructure is shared by several unknown and potentially malicious users. Another consequence is some of the aspects of infrastructure management are now controlled by the cloud provider so there are benefits of not having to worry about them, but there will be limitations of not being able to control them.

Using this solution architecture means that the IT model is operating expense (OPEX) driven instead of traditional capital expense (CAPEX) model. This can lead to potential benefits of having to pay only for what is used but may result in changes to the financial models. Using this solution means use of standardized infrastructure provided by the cloud provider, so it generally results in the benefits of economies of scale, but there may be limitations if specialized hardware is needed.

Depending upon the implementation option involved, there are different benefits and limitations for the same.

Implementation Option 1: Public IaaS Cloud
Benefits:

Some of the key benefits include:

- Faster time to market with self-service and automation
- Lower costs because of pay-per-use models for variable workloads
- Flexibility to support different application infrastructure needs

Limitations:

There are several limitations including:

- Security and compliance-related challenges with shared infrastructure provided by third party
- Lack of maturity of features and limitation thereof

Implementation Option 2: Public PaaS Cloud
Benefits:
Some of the key benefits include:

- Lower costs because of pay-per-use models for variable workloads
- Higher scalability, elasticity, and lower administration overheads with abstraction of infrastructure and platform concerns

Limitations:
There are several limitations including:

- May need re-engineering of the application to use the PaaS provider API
- May result in lock-in to the API as well as data lock-in

Known Uses
Most cloud service providers offer load balancing features without affecting the performance. Amazon EC2, for example, offers load balancing features.

Terremark vCloud Express maps each port of a public IP address to a group of virtual machines. This allows maximum use of a public IP address.

Related Pattern
Cloud burst

11.2.3 Consuming SaaS for Non-differentiating Business Needs

Name
SaaS consumer

Intent
The intent of the pattern is to reduce the Total Cost of Ownership (TCO) of applications that do not provide any competitive differentiation.

Category
Deployment and application

Motivations
The traditional model for making business enabler applications like e-mail, calendar, meeting organizer software, messenger, and collaboration software available to the users is to procure and deploy them in-house within the enterprise network. These applications are critical for performing various business functions, but the functionality they provide is commoditized, and so they are generally available as standardized products and there is no use of custom developing them within the organization. So, Commercial Off-The-Shelf (COTS) products are generally bought, customized if needed, and used. This model of buy and use involves considerable overheads in maintaining them and upgrading them and also consumes valuable IT resources while not yielding any competitive differentiators to the business.

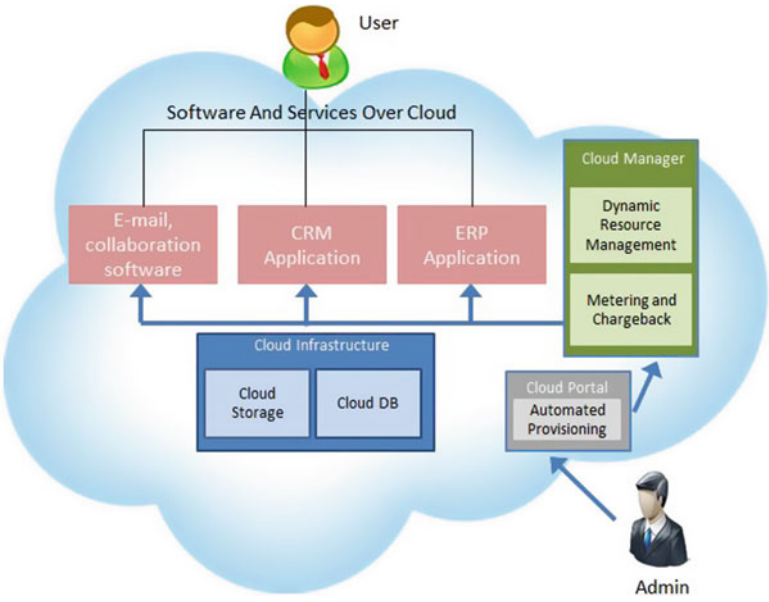


Fig. 11.2 SaaS solutions for non-differentiators

The already existing traditional solutions involve procuring the software licenses and receiving support services separately and upgrading the software when there are newer releases. This involves lot of overhead in maintaining such kind of applications and also incurs huge cost.

Cloud Solution

An alternative approach to procuring and using the non-differentiating applications is getting them on a subscription basis in Software-as-a-Service (SaaS) model so that these applications can be used as a service on pay per use. For example, there are Business Productivity Online Suite (BPOS) offerings from vendors like Microsoft and Google that provide office productivity suite including e-mail and messaging, calendaring, and collaboration solution online on pay-per-use model. Similarly there are other applications like CRM applications (from Salesforce.com) and ERP applications (NetSuite) that are available in SaaS model. There are also entire business processes like Hire-To-Retire (HRO Business Platform from Infosys) that handle all human resource management processes and Source-To-Pay which handle all processes from sending RFPs to making the payments that are available on pay per use. This reduces the overheads of upgrades and constant technology refreshes for those applications while getting the best of breed solution. The conceptual view of the solution is shown in Fig. 11.2.

The IT administrator uses a Cloud Portal to provision and get an instance of an application as a service and makes it available for the users. The applications and the related data reside in a public cloud. The Cloud Manager enables metering the usage and charges the users based on it.

One of the key challenges here is consuming the applications as services and integrating them with the other existing in-house applications. Some of the architectural building blocks needed to implement this are:

- Cloud Management
  - Cloud Portal: For self-service provisioning of the cloud infrastructure and to define the service catalog and template specific to the application(s)
  - Automated Provisioning Manager: For automating the provisioning of the application and its infrastructure
  - Cloud Manager: For monitoring the application usage and dynamically scaling the infrastructure up and down and also to meter the usage and charge the users based on it

Applicability
This solution strategy can be used for applications whose features are identified as non-differentiating. In such cases, if there is corresponding SaaS provider offering the application as a service with acceptable quality of service including security and performance, then this approach can be used. There is increasing use of SaaS, and so software vendors are increasingly supporting this model.

Consequences
Consuming application as a service in SaaS model implies that some of the proprietary data will now be in a public cloud. Use of applications over public cloud infrastructure can lead to security and compliance issue. Also, there can be issues with locking as the data may now be in the SaaS provider's proprietary format, making it harder to migrate to another provider if there are issues.

Known Uses
Salesforce.com, Microsoft BPOS, and Google Apps are examples of a few popular SaaS solutions.

Related Pattern
SaaS provider

11.2.4 High-Performance Computing Using Cloud Infrastructure

Name
High-Performance Computing Cloud

Intent
The intent of the pattern is to enable high-performance compute grid over a cloud infrastructure [7–9].

Category
Structural

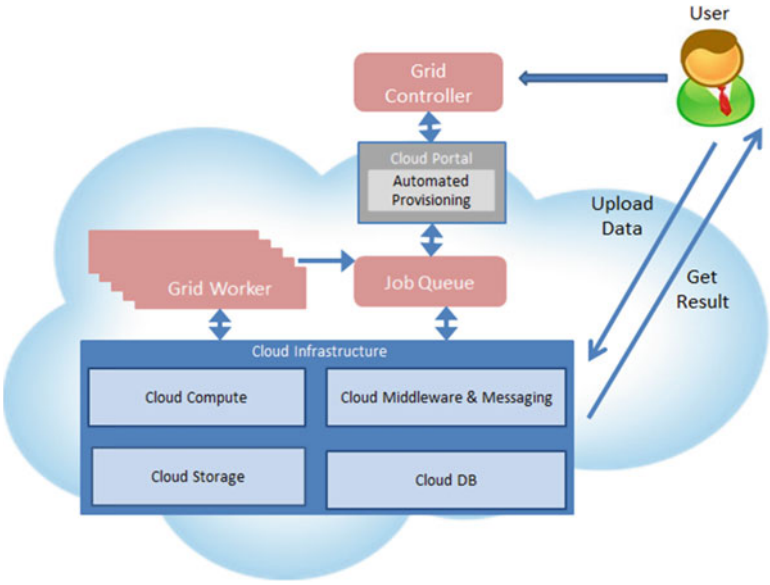


Fig. 11.3 HPC grid on cloud

Motivations

There are several scenarios in enterprises that need high-performance compute grid infrastructure for performing the complex computations like scientific computations and financial computations over huge amounts of data like, say, in case of drug discovery in life sciences, risk analysis in financial services has high computation needs on large amounts of data.

To cater these high-performance computing scenarios, enterprises have been creating single-purpose in-house compute grids. Though these dedicated single-purpose high computing grids can efficiently handle the grid computing tasks, the utilization of the resources in these grids is generally low, and the cost involved in creating and maintaining such grids over high-end infrastructure is high.

Cloud Solution

An alternative to traditional approaches is creating such specialized grids on the cloud infrastructure with a pay-per-use model. The cloud-based grid would be elastic in nature, and the associated cost for maintenance and creating such grids can be lower than an in-house grid. The conceptual view of the solution is given in Fig. 11.3.

The grid controller integrates with Cloud Portal to manage the resources over grid. The incoming job requests are captured in job queues implemented using Cloud Middleware and Messaging components. Grid workers implemented over Cloud Compute read the job request from the queue and performs the actual complex computational processing over the huge amount of data that resides over the scalable Cloud Storage.

One of the key challenges here is ensuring that existing grid applications with complex algorithms work in new architecture with minimal changes.

The architectural blocks we need for this solution are:

- Cloud Infrastructure:
  - Cloud Storage and Cloud DB: To store the data to be used for the complex computations
  - Compute Cloud: The processing layer of grid
  - Cloud Middleware: To enable messaging and asynchronous processing
- Cloud Manager:
  - Automated Provisioning Manager: For automating the provisioning of the grid application components
  - Cloud Manager: For monitoring the grid usage and accordingly auto-scaling
  - Cloud Portal: For self-service provisioning of the grid environments

Applicability
The pattern is applicable for the scenarios like drug discovery and complex financial calculations where a high computing environment is needed for short durations.

Consequences
Grid computing needs specialized platforms like high-performance file systems which may not be available in several public clouds. There are challenges in data transfer to and from the public clouds because of bandwidth limitations to the extent that options like shipping disks with data physically are being resorted to in order to get around these limitations.

Known Uses
Life sciences organizations are experimenting with grids on public clouds.

Related Pattern
Test and Dev Clouds and Cloud Desktops

11.2.5 Low-Cost and Large-Scale Analytics Using "Big Data" Technologies

Name
Big data analytics

Intent
The intent of the pattern is providing a low-cost, large-scale analytics solutions on Big data for scenarios like fraud detection, product recommendations, and EDW.

Category
Structure and application

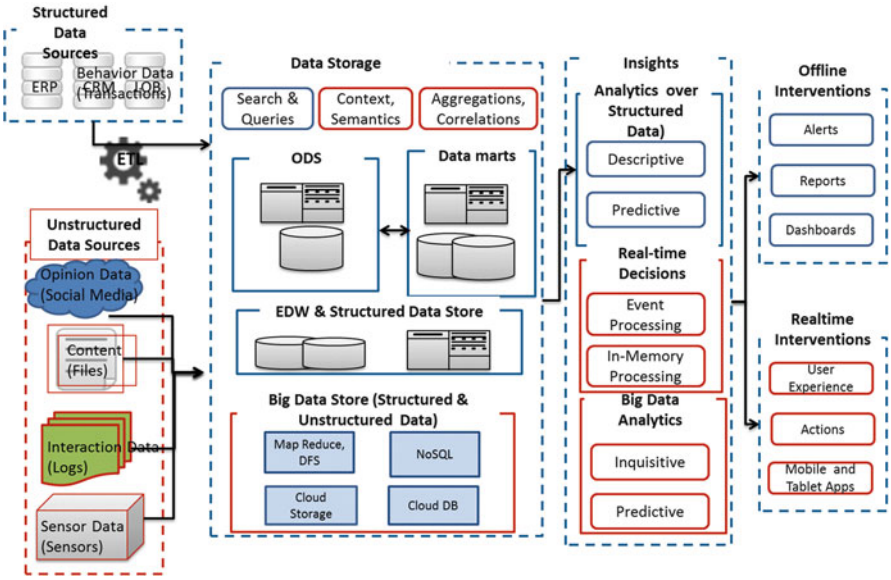


Fig. 11.4 Cloud analytics solution

Motivations

With increasing use of self-service channels and automation of business processes, there is increasing amount of information being captured each day. Data volumes are growing exponentially in most organizations. To give an example: In supply chain management kind of scenario, with the increasing use of solutions like RFID, there is lot more information being captured about the products, inventory, etc. Valuable insights can be obtained by mining this data. Therefore, organizations are looking at various methods to capture and store information and analyze it so that they can use those insights for improving the business efficiency and effectiveness. With more and more data being collected each day, there is a need for a low-cost, highly scalable data analytics solution as traditional storage, and analytics solutions involve high costs and can't scale to meet the growing data needs.

Cloud Solution

The solution strategy is to use highly scalable cloud storage along with parallel processing frameworks like MapReduce that can scale horizontally to peta-bytes of data using commodity hardware. Cloud computing solutions like Elastic MapReduce by Amazon for a public cloud deployment model and Apache Hadoop for in-house deployment enable storing and processing large datasets. These solutions enable leveraging commodity hardware to scale horizontally to perform analytics and business intelligence computations on large datasets. Organizations are looking to leverage these solutions to meet their data warehousing and advanced analytics needs. The conceptual view of the solution is provided in Fig. 11.4.

Data from structured data sources like OLTP, CRM, and ERP application databases and from unstructured data sources like web and application logs, sensors, and social

media is captured and stored in a Big data store. The Big data store is implemented using a combination of components that can be deployed in-house like Distributed File Systems (Hadoop Distributed File System), NoSQL data stores (Cassandra, HBase, MongoDB, CouchDB), public cloud file storage (Azure Blob Storage, Amazon S3, etc.), and cloud database components (SQL Azure, Amazon SimpleDB, etc.). The data is then processed in batch using frameworks like MapReduce and in real time using frameworks like stream-processing frameworks to identify patterns and derive insights. These are then used to apply real-time interventions.

A key architectural challenge is storing and processing large amounts of data cost effectively. For this, the strategy is to use techniques like Distributed File System, MapReduce, NoSQL, and frameworks that can scale horizontally using commodity hardware. Some of the architectural building blocks required to implement this solution are:

- Cloud Infrastructure:
  - Cloud DB and Cloud Storage: To meet the storage needs
  - Compute Cloud: For the analytics processing needs

Applicability
The pattern is applicable for the scenarios with large data volumes and different varieties of data without proper structure that cannot be handled by traditional solutions like RDBMS and traditional analytics products.

Consequences
Use of cloud and NoSQL storage technologies implies ability to scale to peta-bytes of data and low costs, but there may not be rich analytics packages with the various algorithms readily available that are based on these storage technologies.

Known Uses
Facebook has developed a peta-byte scale enterprise data warehouse based on Hadoop and has even open sourced the framework (Hive) that they developed in the process.

Related Pattern
Cloud Storage

11.2.6 Delivering Applications to Consumers Using SaaS Model

Name
SaaS provider

Intent
The intent of the pattern is to provide a solution to offer products and services in pay-per-use model.

Category
Deployment

Motivations
Independent Software Vendors (ISVs) are looking at new business models to increase their customer base and extend the reach of their products and service offerings to the "long tail." Offering the products and services on pay per use enables increase in the reach of the product to those who could not previously afford them. Even large organizations are looking for pay-per-use models instead of the traditional model of procuring the software, deploying it in-house, and then using it, in order to avoid the hassles of keeping them up to date and managing the software in-house.

Another key driver is adoption of similar models by competitors which is forcing the other ISVs to also provide their products and services as SaaS offerings in subscription-based models.

Cloud Solution
The solution is to deliver products and services in Software-as-a-Service (SaaS) model [6]. The most common scenario is an ISV has an existing software application that they would now like to offer in SaaS model. There are different approaches available to convert an existing ISV offering into SaaS with different maturity models: The lowest maturity model is where there are several code bases of the application with tenant-specific application code. This is the most complex to manage from the ISV perspective as any bug fixes of new features will require changes across multiple code bases. The next best model is where there is a common code base and tenant-specific customized code, but there are still multiple independent instances of the application for each tenant. The next best model is where the application is highly configurable, so there is a common code base for all tenants and there are tenant-specific configurations, but the application still doesn't support the concept of multiple tenants natively because of which there are still independent instances per tenant. The highest maturity level is where there is single code base and also the application coded to be aware of the multiple tenants, and so there is only one instance of the application which supports all the different tenants, thereby utilizing a common infrastructure to support all tenants efficiently. Refer to Fig. 11.5.

Option 1: Using Shared Cloud Infrastructure
One of the solution approaches is to create a shared infrastructure across multiple tenants and use intelligent and dynamic resource management to optimize allocation of available infrastructure across multiple instances of an application that are isolated through virtualization or similar techniques. In this model, efficiencies are brought about by sharing the infrastructure across multiple tenants. There are variations in this like one in which the application is capable of being configured to multiple customers with same code base and another in which the customizations need code changes and so independent customized instances. This enables ISVs to get started with minor changes to existing application architecture.

Option 2: Using Shared Cloud Platform and Multi-tenant App
Another solution approach is to make the application multi-tenant and deploy it on a scalable cloud platform like Force.com, Google App Engine, LongJump, or

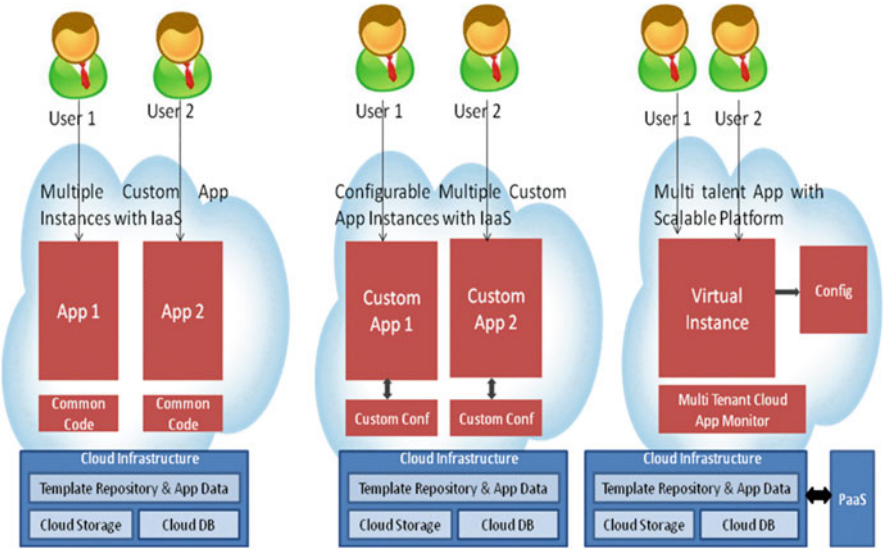


Fig. 11.5 SaaS solution architectures

Microsoft Azure that takes care of application scalability. This architecture is highly optimized but may need many changes if it is an existing product that needs to be offered as SaaS.

The cloud infrastructure provides scalable hosting. The various architecture components of the cloud infrastructure solution are:

- Cloud Infrastructure:
  - Compute Cloud: For the processing layer of the ISV application
  - Cloud DB: Elastic data stores that can scale up and down to meet the application's storage needs
  - Cloud Storage: Elastic storage for storing application-specific files and template repository

Applicability

The pattern is applicable for scenarios for Independent Software Vendors who offer a software product and support services. This is especially applicable for ISVs targeting small and medium businesses and end users directly.

Consequences

There can be security and regulatory concerns with sharing common infrastructure and common instances across multiple tenants. There can also be issues with one of the tenants taking away too many resources, thereby making the application unavailable for the remaining tenants. There can also be concerns with eavesdropping and sharing of sensitive client information

from the SaaS consumers which may need to be addressed through not just technology solutions but through contractual means.

Known Uses
Force.com is an example of a platform specifically designed to support developing applications to be delivered using multi-tenant SaaS model.

Related Pattern
SaaS consumer

11.2.7 Industry-Specific Business Clouds and Marketplaces

Name
Business Cloud Co-creation platforms

Intent
The intent of the pattern is to provide a solution to support ecosystem-driven and open innovation models and marketplaces.

Category
Deployment

Motivations
Traditional model of innovating within enterprise with closed models limits the potential service offerings, doesn't meet the pace of innovation and time-to-market needs of this generation of online services. So, several organizations are looking to adopt open innovation models and are creating vertical cloud platforms and marketplaces for their partners and other innovators to build applications over their core products and services. This requires solutions that enable partners to co-create and consumers to discover and use the applications developed.

Cloud Solution
The solution is to create a "Business Cloud Co-creation" platform [10] providing the core business capabilities as application services with API that partners can use to build interesting applications and host them in application container platforms over a cloud infrastructure. The solution also provides appropriate development tools and marketing capabilities needed for others to build and market their applications.

Organizations can leverage these "Business Cloud Cocreation" to create marketplaces that bring the consumers and application developers together so that there is an ecosystem of partners innovating constantly over the organization's core products and services. Customized app SDKs can be developed to further ease and speed up the application development process. Some examples of innovative business models are given in Fig. 11.6.

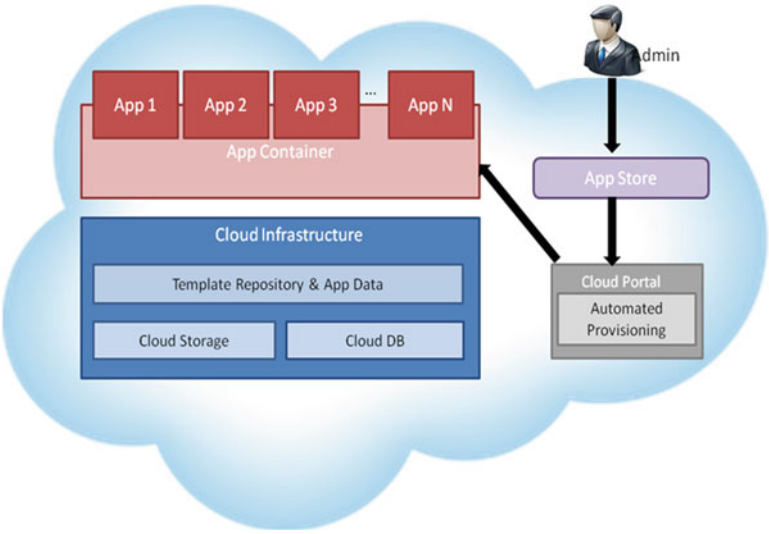


Fig. 11.6 Business Cloud Cocreation platform

Example 1

Telcos are creating their domain-specific platforms offering their core capabilities as services that others can use to build interesting applications. Telcos are also providing marketplaces that enable these to be commercialized.

Example 2

Facebook created a social networking platform that enables others to build interesting applications and monetize them.

Example 3

Amazon created an e-commerce platform that others can use to build their e-commerce portals and sell their products.

A key technical challenge involved here is creating a scalable sandbox with appropriate security and collaboration enablers and a scalable application container platform. Security of data and applications over a cloud infrastructure is a key concern. The conceptual view of the solution is shown in Fig. 11.6.

A key challenge while implementing such a solution using traditional methods is the lack of customized app SDKs to build applications using the core services and offerings. Since many of cloud service providers such as Microsoft Azure provide SDKs specifically for applications development on cloud, organizations can further

customize their services to use these SDKs or integrate their services with these SDKs to provide a business cloud platform for others to innovate and build their solutions using these customized services and offerings.

The various architectural components of the solution are:

- Cloud Infrastructure:
  - Compute Cloud, Cloud DB, and Storage: Provides a scalable processing and storage layers for the application container platform
  - Cloud Portal: Provides the infrastructure for the developer portal and marketplace portal
  - Cloud Manager: Takes care of the resource management

Applicability
The pattern is applicable for large organizations who are adopting an open innovation or an ecosystem centric innovation model. This is applicable for large organizations who intend to provide a core business platform with an industry-specific cloud platform that their partners can use to innovate and build applications. This is more applicable for organizations dealing with end consumers directly there by requiring multiple partners to offer interesting mash-up applications to the end users.

Consequences
Adoption of such cocreation platforms may require a lot of organization changes in the business models, business processes, etc. There can be potential issues with applications being developed by third parties with not much control so there may be a need for application certification mechanisms, secure integration of third party application, etc.

Known Uses
AT&T Foundry, Nokia Store, Apple iPhone App Store, and Facebook Apps are examples of such emerging "Business Cloud Cocreation" platforms.

Related Pattern
Public cloud applications

11.3 Conclusion

This chapter described a number of solution patterns with details of common problem scenarios that enterprise solution architects face in the platform and application layers while developing enterprise solutions, the traditional solution approaches and their limitations, and the cloud computing solution, their benefits, and limitations. Enterprise solution architects can benefit from reusing and continuously improving on these solution patterns.

References

1. Mell, P., Grance, T.: The NSIT definition of cloud computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf (2011)
2. Guest, S.: 5 cloud computing patterns along with examples and Azure solutions. http://www.infoq.com/presentations/Patterns-for-Cloud-Computing (2010)
3. Amrhein, D., Anderson, P.: Cloud computing use cases. http://www.opencloudmanifesto.org/Cloud_Computing_Use_Cases_Whitepaper-2_0.pdf (2009)
4. CA Technologies: IT optimization through workload automation. http://whitepapers.businessweek.com/detail/RES/1335968686_109.html (2012)
5. Subramanian, K.: Hybrid clouds: a whitepaper sponsored by Trend Micro Inc. http://www.trendmicro.com/cloud-content/us/pdfs/business/white-papers/wp_hybrid-clouds-analyst-ksubramanian.pdf (2011)
6. Hong Cai.: A transparent approach of enabling SaaS multi-tenancy in the cloud. In: 6th World Conference on Services. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5575773&isnumber=5575460 (2010)
7. Ekanayake, J., Fox, G.: High Performance Parallel Computing with Cloud and Cloud Infrastructure. Department of Computer Science, Indiana University, Bloomington. http://grids.ucs.indiana.edu/ptliupages/publications/cloudcomp_submission.pdf (2010)
8. Papuzzo, G., Spezzano, G.: Autonomic management of workflows on hybrid Grid-Cloud Infrastructure. In: 7th International Conference on Network and Service Management (CNSM). http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6104004&isnumber=6103944 (2011)
9. Miceli, C., Miceli, M., Jha, S., Kaiser, H., Merzky, A.: Programming abstractions for data intensive computing on clouds and grids. In: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID. http://dx.doi.org/10.1109/CCGRID.2009.87 (2009)
10. Doddavula, S., Subramanian, R., Deb, B.: What is beyond operational efficiency? http://www.infosys.com/cloud/resource-center/documents/beyond-operational-efficiency.pdf (2009)
11. Fehling, C., Retter, R.: Cloud Computing Patterns: http://cloudcomputingpatterns.org/ (2011)
12. Ricky Ho.: Cloud computing patterns. http://architects.dzone.com/news/cloud-computing-patterns (2009)

Part IV
Cloud Modernization and Migration

Chapter 12
Cloud Application Modernization and Migration Methodology

Rajaraajeswari Settu and Pethuru Raj

Abstract In the midst of all the noise, hype, and hope for the cloud-inspired business augmentation and acceleration, there is a big market opening up for application modernization and migration engagements. That is, worldwide service providers and IT divisions are readying themselves with procedures, key guidelines, metrics, tools, and practices to systematically accomplish application renovation to be deposited and delivered from optimal and operational cloud infrastructures and platforms. In this chapter, we share some of the insights and information on where to initiate and how to accomplish the cloud movement successfully and without much pain. We present and discuss the method to be followed in order to simplify and streamline the process of assessment, fit-for-purpose analysis, renovation, and movement toward competent and connected clouds. We inscribe the key technical as well as business reasons for such a strategic exercise.

Keywords Application modernization and migration • Cloud onboarding • Toolset and processes

12.1 Introduction

The key driver for *cloudification* is that expensive IT resources are lying idle. That is, their utilization rate is abysmally low. Thus, the first and foremost for cloud building is to do initial assessment of all the resources for the proposed cloud infrastructure.

R. Settu
Department of Master of Computer Applications, Raja Rajeswari College of Engineering, Bangalore, Karnataka, India
e-mail: s.raja.raajeswari@gmail.com

P. Raj (✉)
Wipro Technologies, Bangalore, Karnataka, India
e-mail: pethuru.raj@wipro.com

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_12,

Clouds are being positioned as the consolidated, virtualized, automated, and shared IT infrastructure for efficiently hosting, managing, and delivering scores of service-centric and enterprise-scale IT resources (applications, platforms, and even infrastructural modules) as services to worldwide users via the public Internet, which is being touted and trumpeted as the cheapest communication infrastructure. In other words, as the knowledge-driven service era sets in and stabilizes steadily, the utility and ubiquity of clouds as the path-breaking and promising service deployment and delivery container is set to climb further. Cloud started its innings as a centralized and converged infrastructure, and now with the maturity of a series of enabling techniques and standards, clouds are being empowered significantly in order to solve high-end problems through a seamless federation of multiple clouds.

The main turnarounds and transformations accrued out of the pioneering cloud idea is that clouds enable IT agility, affordability, and autonomy, which in turn provides business automation, acceleration, and augmentation. The goals of IT simplicity and sensitivity are being readied and realized with the greater understanding of novelty-packed cloud concepts. IT resource utilization goes up remarkably; IT resource elasticity, self-servicing, and application scalability are getting automated using competent software solutions. Cloud infrastructures are extremely lean, green, and programmable. A bunch of delectable enhancements such as the extensibility, malleability, usability, and consumability of IT modules at different layers and levels of the enterprise IT stack are portending good days for cloud computing. The increased IT efficiency easily gets translated into overall business efficiency, and this has the immense and immeasurable potential to unleash scores of fresh IT capabilities and business opportunities. On the operational side, there are several solid and sustainable improvements on the aspects of manageability, performance, maintainability, and simplicity. All these clearly drive business executives and entrepreneurs to refocus on the viability, vitality, and value additions of the innovation-breeding cloud idea. By delegating the management and governance of infrastructure and software platforms to a team of highly skilled professionals being employed by cloud service providers (CSPs), customers could off-load operational responsibilities and risks to CSPs.

A cloud environment predominantly gives an illusion of infinite processing and storage capabilities and capacities. This radical and radiant empowerment does a lot of good for some specific enterprise applications. Applications that are designed from the ground up to spread their workloads across multiple servers will be able to immensely benefit from the provision of automated scaling of resources to match varying demands. User and workloads are bound to change very frequently. This is quite appealing for applications with unpredictable or cyclical usage patterns, because a cloud orchestrator/broker/service bus can monitor usage and can dynamically scale resources up or down on need basis. This behavior, in sync up with the pay-by-usage characteristic of clouds, could lead to significant cost savings.

With the greater and deeper understanding of the business, technical, and use cases of clouds, organizations across the world are preparing sanguine strategies and roadmaps to plunge into the cloud space, which is by the way steadily expanding through a host of innovations and improvizations. However, the cloud onboarding has to be very carefully and calculatedly articulated and accomplished as there are several

constrictions and challenges associated with that. This chapter gives an overview of why, when, how, and what for cloud onboarding and a lean migration methodology for smartly and successfully embarking on the long and arduous cloud journey.

12.2 Cloud-Inspired Enterprise Transformation

Enterprise transformation becomes a neat and nice reality via a wide array of actions such as process excellence, infrastructure consolidation, and intrinsic assimilation of proven and potential architectural styles and technologies. If there is a glut in IT infrastructure, then infrastructure rationalization, consolidation, and optimization procedures are widely and wisely being recommended. That is, a lean infrastructure is the most desired state of any growing and glowing business, and this significantly reduces the tightly aligned infrastructural complexity and cost (capital as well as operational) in due course of time. With the adoption of autonomic computing concepts, infrastructures are turning out to be the prime target for innately incorporating self-diagnosing, self-healing, self-configuring, self-defending, and self-managing capabilities in order to sharply decrement human intervention, instruction, and interpretation.

Process innovation is the next thing to be contemplated very seriously for enterprise automation, acceleration, and augmentation. Process reengineering, integration, orchestration, management, and governance techniques need to be leveraged in order to bring in lean processes that in turn would have a cascading effect on shifting enterprise operations, outputs, and offerings. Thirdly, newer architectural styles are showing a lot of promises and potentials in refining enterprise outlooks. The recently erupted enterprise-scale and business-critical service-oriented architecture (SOA), event-driven architecture (EDA), model-driven architecture (MDA), Web-oriented architecture (WOA), resource-oriented architecture (ROA), etc., are becoming very popular in designing, developing, deploying, delivering, and sustaining on-demand, adaptive, real-time, dynamic, flexible, and modular applications. Finally, disruptive, transformative, and inventive technologies are emerging and evolving fast in the IT landscape. Verifying and validating these technologies to be fit for the purpose are very vital for the delectable success. There are path-breaking technologies that can be generically or specifically utilized for different IT functions. Virtualization, mobility, cloud-based big data, and business analytics are some of the shining and sustainable trends in the technology space.

With the overwhelming acceptance of cloud concepts, there is a big turnaround waiting to happen in IT. Especially on the infrastructure front, there are possibilities for radical optimization. The cloud paradigm has also brought in newer delivery, pricing, and consumption models. Applications are bound to be hosted, managed, and maintained in centralized and converged infrastructures and platforms in order to be discovered, accessed, and used over the Internet, which is being positioned as the common, cheap, and competent communication infrastructure for universal connectivity. In short, the incoming trends and technologies will require organizations to seriously look at consolidation, standardization, and rationalization.

Before plunging into the cloud landscape, there has to be a deeper analysis and greater understanding of its intrinsic capabilities and constraints. There are some scenarios wherein cloud is the best bet whereas there are some other areas wherein cloud may be a laggard. There are a few well-known shortcomings such as security, availability, and lack of controllability and visibility. There are distinct views being expressed based on different viewpoints around cloud computing. As such cloud is not a specific software or hardware product. The cloud is not simply virtualizing existing IT infrastructure using VMware vSphere, Citrix XenApp, Oracle VM, and another server-based virtualization (SBC) product but making it accessible over the Web. It is much more than that. It is an amalgamation of a number of established technologies. Professors, pundits, and professionals have come out with a variety of enabling tools, best practices, and key guidelines.

The cloud is on-demand, dynamically provisioned, and shared IT resources discoverable and accessible on the Internet. However, cloud computing is more than just putting your hardware infrastructure on the Internet. What really defines a cloud as opposed to running a consolidated system in a private or publically accessible network environment are these four characteristics. Tom Laszewski [1] explains these characteristics in his article, *A Brief Introduction on Migrating to an Oracle-Based Cloud Environment*, in detail. A brief discussion of the core characteristics and cloud mechanisms now follows.

12.2.1 Provisioning

Dynamic and self-service provisioning of applications involves deploying applications to your cloud infrastructure with a few mouse clicks. So, when moving to the cloud, application provisioning will be made easier since there are not hundreds or thousands of client machines to deploy too. Self-service provisioning of computing infrastructure in a cloud infrastructure is also very desirable as it can cut down the time it takes to deploy new infrastructure for a new application or scale-up/down infrastructure for an existing application. Public cloud service providers like Savvis, Terremark, and AT&T all have self-service portals where users can sign up for cloud services and their compute infrastructure is ready within hours for usage. For example, Oracle Enterprise Manager provides many application programming interfaces (APIs) that can be incorporated into the self-service portals of cloud providers to automate the provisioning of compute infrastructure for Oracle products such as Oracle Fusion Middleware and Oracle database.

12.2.2 Metering and Chargeback

Metering and chargeback are mechanisms for gathering compute resource usage (metering) and then charging back the user, department, or company for the computing cycle that they consume. In the client/server model, metering and chargeback were

not necessary because each user had their own personal computer and typically each department or customer had their own database server. Today, most cloud services are offered on flat pricing model structured upon usage of computing resources for fixed unit of time (month or days). Although this model reduces the cost of using IT infrastructure, end users and developers pay a price regardless of actual usage. If a user subscribes for a compute infrastructure for a month and even if they actively use it for only 10 or 15 days, they still have to pay for entire month. This pricing model is typically derived by calculating the overall cost of the infrastructure (acquisition cost + operational cost + margin) and then dividing the amount by a fixed unit of time such as months, days, or years. Using sophisticated metering and chargeback tools provides both public and private cloud providers the ability to monitor actual usage of resources by the users and then charge them accordingly. Accurate resource usage information (metering) up to the minute or second in time can lead to a true pay as you go model which is a key premise of cloud computing. Products such as Oracle Enterprise Manager capture very detailed information about the resource usage and store it in its repository that can be later used for chargeback purposes.

12.2.3 Multi-tenancy

Multi-tenancy, or running multiple customers or departments on the same central hardware and software infrastructure, is not an issue for client/server applications since each user has her own instance of the application, and most likely each department or customer has its own dedicated database server. In cloud environments, the application, database, and hardware infrastructure are shared among departments or even companies. There are methods being published in order to smoothly transition from single-tenant application to multi-tenant application.

Multi-tenancy for most companies and cloud hosting providers is achieved by using Server-Based Computing or Virtual Desktop Infrastructure. Server-Based Computing (SBC) is most commonly associated with VMware vSphere. Virtual Desktop Infrastructure (VDI) has become the latest buzzword in the cloud computing arena. Both offer virtualization and both hold the virtual environment on a centralized server. This is where the similarities end. VDI provides a full desktop experience, while SBC offers terminal service-based access to a centralized/shared operating environment.

Multi-tenancy is one of the most important characteristics of cloud environment. Multi-tenancy, in an Oracle database, can be achieved in a number of ways without using hardware or OS-based virtualization using database features and options. Oracle database schemas, one database instance per customer, and Oracle Virtual Private Database (VPD) can all be used to implement database-level virtualization. It is also important to manage Oracle database resources such as session time limits, number of concurrent sessions allowed, degree of parallelism, and undo segment storage use based on the prioritization policies defined for each consumer. To facilitate the effective resource management (Oracle Database's Instance caging feature),

Oracle DB Resource Manager and Oracle DB Quality of Service (QoS) Management can all be used to ensure database resources are not oversubscribed by a particular customer. Multi-tenancy can be achieved at the application tier without VDI or SBC products using Oracle Solaris Zones and WebLogic domains.

12.2.4 Elasticity

Elasticity refers to the ability to dynamically provision, migrate, and allocate computing resources to users, departments, or customers. It includes the infrastructure to easily set up and "tear down" applications. One of the first areas that customers focus on when migrating to the cloud is developer test, system testing, and User Acceptance Testing (UAT) environments. The ability to quickly set up a new test environment and then delete it once the testing is done is a cost-effective and time-saving exercise when done using cloud. Elasticity also implies the ability to provide more computing resources dynamically upon consumer request.

Migrating to the cloud does not need to be a gut-wrenching and expensive endeavor. However, we must consider how we implement metering and chargeback, provisioning, elasticity, and multi-tenancy before planning and working on the migration process [2] to the cloud.

Elasticity is one of the most important characteristics of cloud environment. It has been the hallmark of Oracle platform due to its support for grid computing which allows Oracle to utilize all the computing resources distributed across many servers large or small as if it were a massive server. Key element in the grid architecture is the support for clustering in the Oracle database and the Oracle Fusion Middleware. Clustering allows Oracle platform products to overcome the physical limits imposed by servers. Clustering combined with virtualization support in Oracle platform products makes it a highly elastic cloud platform. However, virtualization can also be done with Oracle at the database, application, and operating system layers without using SBC or VDI products. Oracle does offer both VDI (Oracle Virtual Desktop) and SBC (Oracle Virtual Server).

12.3 Moving to the Cloud

Business organizations and IT service organizations are establishing pragmatic plans and readying themselves for embarking on the strategic cloud journey. The ultimate cloud can be private or public or even the hybrid version. Deepak Jain [3] proposes the following activities for successfully moving to a private cloud environment:

- Conduct a thorough analysis of process and applications for security, latency, load variability, availability, and interdependencies.
- Map the interdependencies to the infrastructure layer.

- Build a heat map where they map the process and application to as-is, internal private cloud, hosted private cloud, and public cloud.

The next key step to building a private cloud is to look at return on investment (RoI) or the total cost of ownership (TCO). Then, it comes down to implementation, which must have a robust design that includes:

- Standardization
- Consolidation and virtualization
- Automation
- Self-service and governance
- Metering and chargeback
- Management

A key aspect is to take care of integration challenges and leverage the existing investments in the design stage. The following are the recommended steps that will ensure that customers have a smooth sailing into the cloud. With respect to the migration activities, enterprises need to look at IT infrastructure transformation through the following four broad areas.

12.3.1 Consolidation

Customers are increasingly looking to achieve business and IT alignment, more meaningful reporting, better coordination between various departments, and a reduction of the overheads associated with managing multiple vendors. It is a well-accepted fact that consolidation is a big lever for cost savings. As IT assets increase, so does IT infrastructure complexity, creating significant management problems. In addition, data center energy consumption skyrockets, especially when energy prices rise. Therefore, the consolidation approach should focus on consolidation of:

- IT operations, such as a central monitoring and command center, IT service desk, and knowledge database.
- IT Infrastructure, like reducing the number of data centers and computer rooms.
- IT procurement: With a vast experience on procurement for global organizations, service providers gain the scale to negotiate and manage hardware and software spend better services through a shared model for service desks and a factory model for application packaging.

12.3.2 Standardization

Simplification is an important lever for controlling costs. Over time, many customers have added complexity to their IT environments like disparate applications, operating systems, technologies, processes, or tools used. Standardization is an

IT complexity mitigation technique. Also, heterogeneous and multiple IT resources need to interact, cooperate, and share their special competencies. For such uninterrupted interactions, seamless and spontaneous interoperability is the need of the hour. Industry strength and open standards are the mandatory things for enabling such kinds of integration, orchestration, and collaboration. It is a logical and correct approach that outsourcing buyers make one-time investments to standardize the IT estate and look at RoI over 5–7 years.

12.3.3 Rationalization

Application and infrastructure portfolio rationalization is an enterprise-wide activity in order to bring down cost and a critical component in business transformation. There are tools that facilitate enterprise portfolio management (EPM). Architecture revisit and remedy are being recommended for eliminating the waste and to reach the goal of lean establishment. All these directly and indirectly contribute toward cost savings. There are several parameters being considered for IT rationalization and simplification. The major parameters to rank each application are first identified and used for verifying and validating each and every tangible resource in the IT landscape. The prominent parameters included business value, IT architecture fit, application efficiency and agility, risk of managing application, complexity, and resistance to change in the application. This enabled the customer to reduce the number of applications. The result, significantly reduced maintenance costs and increased uptime, according to Jain.

12.3.4 Virtualization

This is a breakthrough technology [4] that has brought in a series of paradigm shifts in the IT infrastructure side. All kinds of physical servers get transitioned into virtual entities that can be effectively managed through IT resource management solutions. Data centers are being comprehensively virtualized and subsequently automated in order to reduce their operational costs. On-demand provisioning of IT resources is being automated; thereby the goal of elasticity is being accomplished easily. Virtualization has permeated into server, storage, network, database, application, services, etc.

On the desktop side, virtualization (VDI) leads to better control, lower costs, and increased ease of desktop management and refresh. VDI provides centralized management of all an enterprise's desktops, thereby controlling the applications installed and used. While infrastructure migration to cloud platforms provides benefits, it can also hamper system performance if not addressed meticulously. In this context, a physical to virtual migration of infrastructure results in better resource utilization. It also eases the time and managerial efforts that go into infrastructure migration

to cloud platforms. Take this physical to virtual to cloud (P2V2C) migration in a step-by-step manner in order to ease any hidden hassles.

12.4 A Perspective on Cloud Migration

There are several kinds of cloud delivery and deployment models. Moving applications, services, and data stores to cloud infrastructures and platforms is beset with a number of critical challenges and concerns. This movement involves a lot of upfront planning, facilitating migration tools, best practices, and pragmatic experiences. Many factors must be considered when moving an application to the cloud: applications component, network stack, management, security, dependency, and orchestration.

It is the deployment or migration of data, applications, or integrated solutions of compute, storage, and network resources to a public, private, or hybrid cloud. Onboarding addresses business needs such as a spike in demand, business continuity, backup and archival, and capacity optimization. Enterprises can use onboarding to address capacity demands without the need to deploy additional infrastructure. Cloud onboarding should be considered in the design of overarching and enterprise-wide cloud infrastructure that supports internal, external, and federated clouds. It provides a very compelling usage for enterprises who want to maximize the elastic capabilities of cloud computing.

The great premise of cloud onboarding is to allow the cloud to act as an additional resource or extension of the data center for the following reasons:

- For occasions when the data center becomes overloaded by demand spikes
- For cost-effective capacity management and seamless load balancing
- For disaster recovery and failure mitigation

12.4.1 Application Migration: A Case Study

In this scenario [5], Company C is a small oil and gas company which owns some offshore assets in the North Sea oilfields.

Company C needed a data acquisition system to allow them to manage their offshore operations by monitoring data from their assets on a minute by minute basis. Company C's assets rely on the production facilities of Company A (a major oil company); therefore, the data comes onshore via Company A's communication links. Company C does not have the capabilities to develop their own IT systems; hence they outsourced the development and management of the system to Company B, which is an IT solutions company with a small data center. Figure 12.1 provides an overview of the system, which consists of two servers, namely,

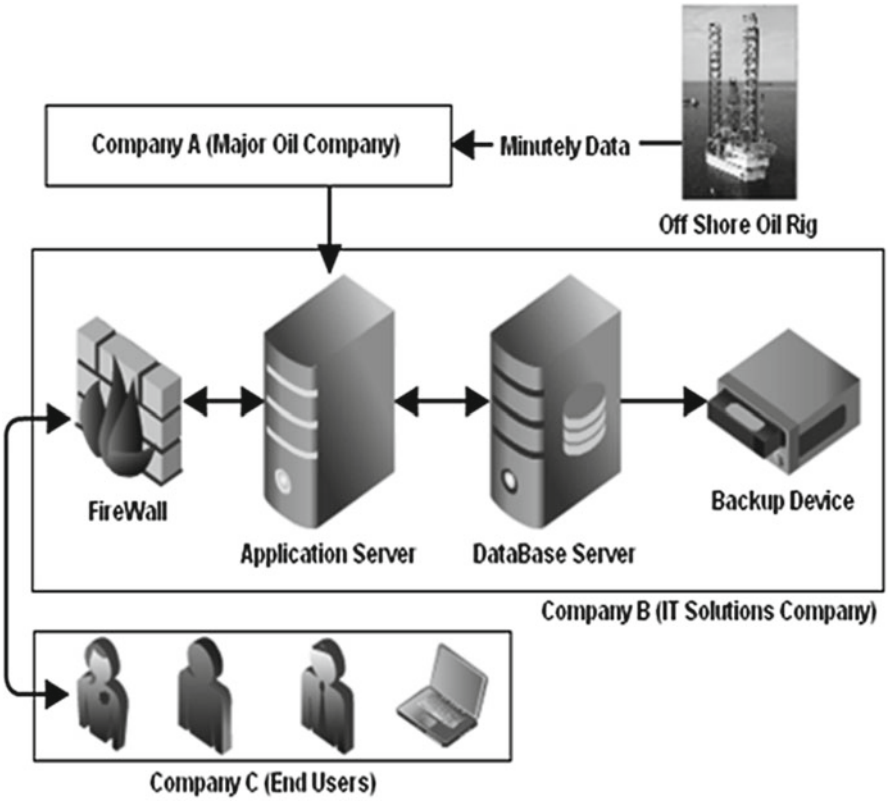


Fig. 12.1 System overview

- A database server that logs and archives the data coming in from offshore into a database. A tape drive is used to take daily backups of the database, and the tapes are stored off-site.
- An application server that hosts a number of data reporting and monitoring applications. The end users at Company C access these applications using a remote desktop client over the Internet.

The system infrastructure was deployed in Company B's data center some years back. Since then, Company B's support department has been maintaining the system and solving any problems that have risen. This case study investigated how the same system could be deployed using the cloud offerings of Amazon Web Services. Figure 12.2 provides an overview of this scenario, where Company B deploys and maintains the same system in the cloud.

Application modernization and migration to cloud bring in a number of advantages. There are cost comparisons between on-premise and off-premise (cloud) deployment, management, enhancement, and delivery of applications, and there are definite cost advantages if application is placed in cloud environments.

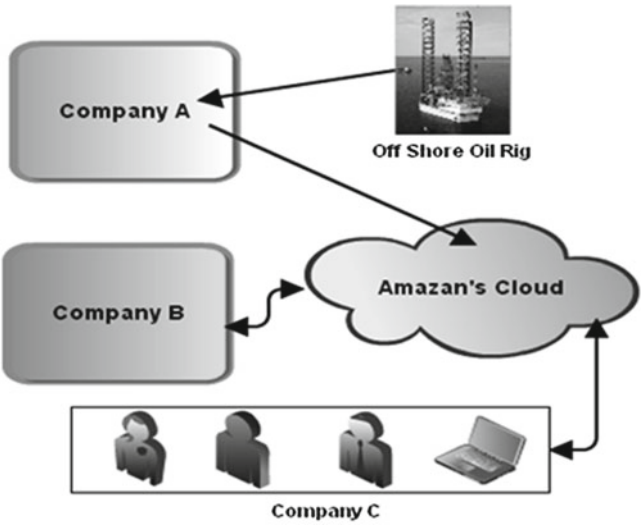


Fig. 12.2 System deployed in cloud

12.4.1.1 Cloud Migration Use Cases

Intel Cloud Builders Guide has come out with three types of use cases for cloud onboarding:

- *Use case 1* – Onboarding simple, self-contained applications or virtual machines. The migration and instantiation of a single, integrated, virtualized application, such as a static Web application, and simple, self-contained applications, into an external or public cloud.
- *Use case 2* – Onboarding multiple-VM workloads with storage and services residing in the enterprise: the instantiation of a multi-VM workload with interconnections to storage and other enterprise applications. One or more VMs of the workload are migrated to the cloud, but storage and other services (infrastructure, database, and so on) still reside in the enterprise. Network transparency between the two clouds enables seamless execution.
- *Use case 3* – Onboarding multiple-VM workloads with storage: the migration and instantiation of an integrated number of applications and a virtualized workload that interconnect with storage are also located in the cloud. This involves the migration of both application workloads and storage to the cloud.

12.4.1.2 Considerations for Migration

Cloud onboarding can be complicated; it can require the redesign or re-architecture of application components, network stack, management, security, or orchestration, all dependent upon the application and infrastructure intricacies and complexities.

Before an application can be successfully on-boarded to the cloud, here are some key considerations.

The application will likely consist of more components and infrastructure dependencies than will initially appear in the cloud environment. Some of these components, like directory services, domain name system (DNS), or dynamic host configuration protocol (DHCP), that are shared by other applications simply cannot be moved to the cloud, and cloud applications that require them must integrate back into the enterprise implementation.

Enterprises want to run applications across multiple cloud service providers in an integrated, secure manner with an integrated management solution. Enterprises must also consider the way the applications run in the enterprise versus the way they will run within the cloud. Cloud onboarding may involve a conversion from P2V or V2V environments. Corporates still want to keep up their customer, corporate, and confidential data locally in their own enterprise servers or private cloud. Even mission-critical applications are still kept in local and on-premise servers. Customer-facing applications can be moved to public clouds. This sort of distributed and decentralized application landscape is bound to create a lot of management complexity. Integration is an issue, controllability and visibility will go down, security and privacy are other problems, etc.

12.5 Core Considerations for Application Migration

Applications need to go through a series of optimizations before getting exported to cloud environments. Sometimes existing applications may need to be subjected to re-architecting to facilitate the migration. To exploit the flexibility of a cloud environment, we need to understand which application architectures are properly structured to operate in a cloud, the kinds of applications and data that run well in cloud environments, data backup needs, and system workloads. The key is trying to architect applications that reduce or eliminate the number of difficult-to-resolve dependencies between the application stack and the capabilities provided by the CSP. There are at least three cloud application architectures in play today.

- *Traditional Application Architectures* such as three-tier architectures are designed for stable demand rather than large variations in load. They do not require an architecture that can scale up or down.
- *Synchronous Application Architectures* are those wherein end-user interaction is the primary focus. Typically, large numbers of users may be pounding on a Web application in a short time period and could overwhelm the application and system.
- *Asynchronous Application Architectures* are essentially all batch applications that do not support end-user interaction. They work on sets of data, extracting and inserting data into databases. Cloud computing offers scalability of server resources, allowing an otherwise long-running asynchronous job to be dispersed over several servers to share the processing load.

There are legacy (siloed, massive, and monolithic), Web, enterprise, and embedded applications that are the prime candidates to be carefully considered for cloud-enablement so that they can be ported to cloud environments without any compatibility issue and provided as a publicly discoverable, accessed, and leveraged as a service over the Web for subscription and usage-based fee.

12.6 A Cloud Onboarding Process

This section lists out a five-step process that helps rediscover the optimal balance of performance, agility, sustainability, and cost. The steps being:

- Evaluate
- Select
- Migrate
- Optimize
- Operate

12.6.1 Evaluate

Every cloud migration has to begin with due diligence of the existing infrastructure. The need is to identify the technical, business, legal, dependency, and usability constraints of the applications that are designated for migration. The nonfunctional aspects (quality of service (QoS) attributes such as performance, scalability, mission-criticality, availability, and security) ought to be evaluated. Finally, the business-centric evaluation according to the business architecture has to happen. The major focuses would be on the business advantages, the challenges, the revenue part, and so on.

12.6.2 Select

Once the applications are evaluated for their suitability to cloud environments, the cloud deployment and delivery models and platforms need to be selected to get as much flexibility as possible through this migration. Each application has to be subjected to consideration according to costs, burstiness and proximity to end users, security, and many other factors. There is a broad range of deployment options available today.

- *Bare Metal* – Traditional "down to the concrete" machines, where you control every aspect of the environment and there is no overhead for virtualization or management. While this requires up-front investment, it is the most powerful option for applications with known and relatively consistent resource profiles.

- *Virtual Machines* – Virtualization can turn a physical server into many virtual servers. It is a great option for workloads that play well with others but still need custom environments and nonstandard configurations.
- *Private Cloud* – Consolidation of underutilized and unutilized compute machines, creating virtual machines out of them via hypervisors, incorporating a number of automation software for self-servicing, workload management, job scheduling, system management, capacity prediction and service governance, etc. Private clouds are very safe providing high controllability and visibility.
- *Virtual Private Cloud* – This is not a private cloud but being siphoned out of public cloud with enhanced and foolproof security via a dedicated IP address.
- *Infrastructure Cloud* – This, being the common and widely talked about public cloud, provides virtual infrastructure boxes such as servers, load balancers, and firewalls on demand but charged on hourly basis or usage. It is a dynamic pool of commodity servers for creating and shutting down virtual machines as per the changing needs. This is by far the cheapest option with resource elasticity and application scalability (horizontal).
- *Platform Cloud* – There are a stream of platforms for application design, development, debugging, deployment, and delivery on cloud infrastructures. Cloud infrastructure management solutions take care of the platform management requirements. But this has lock-in risks and limits the choice of languages and tools.
- *Mashups, SaaS, and RESTful APIs* – Due to the extremely simple nature, cloud service providers (CSPs) are giving out RESTful APIs for their resources to application developers and user agents. There are mashup editors for composing business-aligned mashups getting crafted quickly and delivered to business users. Service composition gets sharply simplified with the usage of RESTful APIs.

12.6.3 Migrate

Once we understand what belongs where, it is time to migrate the applications and data. This requires careful planning to avoid any unanticipated surprises and downtime. For systems that are migrating from on-premise to on-demand environments, this often begins with virtualization and making the applications portable. On the other hand, for cloud-based systems that will benefit from dedicated hardware, this may involve architectural changes and server farm configuration. There are myriad supporting services such as DNS, authentication, monitoring, monitoring and billing, and backup, which must also be migrated in order to streamline cloud-based application delivery.

The candidate applications need to be further evaluated to make sure that it is feasible to migrate and also prepare for the migration process itself.

- *Application Architectures* – The application architecture will affect how an application can be migrated to cloud environments and sometimes whether an application is suitable for migration.

- *Multi-tiered Applications* – The majority of enterprise applications are built using multiple tiers to decouple the major functions and modules in the system. One such approach is to organize the application using three tiers as follows:
  - A data management tier, which consists of relational or other database components
  - A business logic tier, which uses application platform or containers, such as Java EE or Microsoft's .NET
  - A presentation tier, which is responsible for interfacing with the user interfaces or other external systems, including managing state and data for presentation to these external systems

Applications that use a layered architecture approach have well-defined interfaces between these layers. Based on application usage patterns, it might be possible to have migrating application tiers or modules within a tier separately. For example, in a Web application, static content can be migrated to a content delivery network provider to allow parts of a website to load more quickly. In other cases, WAN bandwidth restrictions might prevent tiers from being separated, and all layers of the application will be migrated to a cloud. In either case, each layer and its major distributed components and modules should be evaluated separately for how it should be sized and migrated to a cloud. Application tiers might also have varying security and zoning requirements. For example, some application data might have to be secured behind a firewall.

- *Scale-Up and Scale-Out Architectures* – A "scale-up" architecture is one where the application can benefit from more resources, such as CPU and memory, being added to a single server or node. In contrast, "scale-out" architecture is one where an application scales by additional nodes being made available for the workload; that is, it scales horizontally. Scale-out applications can take advantage of the pay-by-usage cost model of the cloud. When there are increased requests for an application, more nodes can be deployed to handle the increased load. When the requests slow down, the additional nodes can be powered off to reduce costs. Today, it is not possible to dynamically scale up an application running on a single machine instance. This might change in the future, because virtualization systems are starting to support hot-plug features, where more memory and CPU can be added dynamically.
- *Multi-tenancy* – This is a vital factor to be given utmost consideration when transitioning applications from on-premise to cloud environment. Sharing is a foremost feature of clouds and there would be several concurrent users of the application. Multi-tenancy is all about leveraging just one instance of the application for meeting up the requirements of multiple users. There are different mechanisms for enabling multi-tenancy in cloud applications at different levels (infrastructures, applications, databases, etc.).
- *Application Dependency Mapping* – This activity identifies dependencies among applications on a shared data center infrastructure. The data collection can be done in multiple passes or phases: initially to identify all of the application's

immediate dependencies and then to identify what other applications are dependent on the applications' dependencies. For example, if both application A and application B are using the same database server, this needs to be identified so that the migration plan can include a combined move or can include steps to split the dependencies.

- *Application profiling* is used to measure and collect real usage data of an application before it is migrated. This data can help size the application deployment in a cloud. Ideally, application data should be collected for at least 10–15 days to allow capture of variances in daily and weekly usage patterns. For each node on which the application runs, the following data should be collected:
  - CPU usage.
  - Memory usage.
  - Storage data such as throughput, latency, and input/output operations per second (IOPS).
  - Network data such as throughput, connections per second, and dropped connections.
  - The node-level data can be used to estimate how many and what type of machines will be necessary when the application is migrated.

In addition to node-level statistics, it is also important to profile the user activity, such as the total number of connected users, request and transaction rates, and request latencies. The usage data can also be used to build automated tests for the application to make sure of the same or an improved level of service after the application is migrated.

The node data, along with application usage data, can also provide an initial estimate of the costs of the cloud resources.

12.6.4 Optimize

Once the applications find a new residence, they need fine-tuning. Cloud-based systems need to grow and shrink automatically. That is, auto-scaling to meet demand and then shrinking when not needed to save money. Systems running on dedicated infrastructure may need tuning to squeeze the most out of the platforms they run on and can often benefit from acceleration and optimization techniques that simply are not available in a cloud environment. The optimization activities ultimately result in high-performance, flexibility, dependability, lower costs, etc.

12.6.5 Operate

To simplify and streamline cloud IT operations, there are several software-based automated solutions. All kinds of virtual resources getting created are smartly closed once their role and responsibility comes to an end. There are cloud service

brokers, cloud service governance engines, service gateways, system management solutions, etc. All these assist administrators to guarantee high performance and assurance.

12.7 Importance of Policy in Cloud Migration

Andrew Hillier, CTO and co-founder of CiRBA Inc., in [6], insists on the importance of policies for pain-free and riskless migration to clouds. Many organizations are aggressive and articulate in their plans to leverage the potentially sound cloud technologies for sustainable and strategic business acceleration and automation. The fundamental benefits of cloud technologies fervently call for structured and substantial actions. But taking action before properly thinking through when, how, and why these technologies should be used can be dangerous, and this ad hoc and thoughtless actions can lead to a "ready, shoot, and aim" approach to cloud migration. The key to avoiding this is to thoroughly and thoughtfully engage in cloud migrations in terms of well-intended policies.

IT infrastructure is generally complex, and in most cases, the rules of what can and cannot be done might not always be well documented. Combine this with the ever-changing needs of applications and their data, and it becomes very difficult to properly assess [7] and match application requirements with their underlying infrastructure capabilities. To make things worse, one wrong move can cause irreparable damage both to the application and the migration project. A loss of confidence can spell disaster, and, hence, policy [6] establishment and stringent enforcement go a long way in avoiding all the bad and sad migration-induced consequences.

The notion of policy is always daunting for many, and people do not want to get unnecessarily embroiled in a lengthy and up-front analysis of a migration task. Fortunately there are migration-specific patterns, best practices, facilitating procedures, key guidelines, and enabling toolsets. There are several major areas of policy, and if these can be appropriately embedded in the tools and processes used to perform transformations, then cloud migrations can happen satisfactorily, quickly, and safely.

- *Planning Policy* – This defines the order, priority, and grouping of systems that are to be migrated to the cloud. This enables to plan effectively to move these chunks of infrastructure and/or applications. The policies in this area include whether to migrate by physical criteria (e.g., data centers running out of power) or logical criteria (e.g., app-by-app) and also how to deal with application interconnect. Latency can be a serious problem when moving to external clouds, making interconnect analysis critical.
- *Candidate Qualification Policy* – This covers the qualitative and quantitative criteria that indicate whether a specific server workload is suitable for the target cloud environment. On the qualitative front, considerations like the treatment of sensitive data and required SLA levels will dictate whether a specific application or specific components of an application can be placed in a certain type of cloud

capacity. On the quantitative front, high levels of activity like context switching or I/O can also rule out migration to certain environments, which may not be designed to support high throughput or performance or where these capabilities may cost more money.

- *Sizing Policy* – This governs how existing workloads should be sized into the cloud capacity. Many clouds are packages into fixed instance sizes (e.g., small, medium, large), which are modeled in the cloud catalog. The policy on how to map the existing workloads into these containers is critical as any misallocation of resources will either cause performance issues or will waste money by over-provisioning the applications. Policies in this area, for example, may dictate that decisions are based on the peak activity of the last year, trended forward, and normalized to the cloud capacity using SPEC benchmarks, targeting to use 75 % of the cloud container. Load balancers and other structures that provide an opportunity to use the more "elastic" features of clouds must also be reflected in the overarching policy.
- *Placement Policy* – This governs how workloads are placed in the actual back-end infrastructure and dictates everything from business constraints (e.g., these apps cannot go together) to technical constraints (e.g., CPU and memory over commit ratios) to performance targets to operational policies. In many cases, these policies may be owned by the infrastructure groups managing the servers, but they must also be reflected in the migration planning to ensure that any subtleties are accounted for early in the process.
- *Exception Policy* – This deals with what to do with the workloads that are not suitable candidates for the cloud being targeted. Common examples include large database servers, which may be better placed on physical servers or in shared database environments. This aspect of policy ideally defines an order of precedence, such as trying custom cloud containers, then VMs, and then blades and rack mount servers.

Thus, well-defined policies can help in preparing, policing, and polishing the cloud migration. Giving some thought to these areas ahead of time and using them to reach a common ground with application owners and other stakeholders are extremely important to the success of cloud migration initiatives. Modeling these policies in order to enable the automation of cloud migration decisions can greatly accelerate migration projects and prevent costly mistakes along the way.

12.8 Cloud Migration Services

The recent recessionary trend in the world economy has made company executives and decision-makers across the world to rethink and refine their priorities and business directions. The number one priority is to derive maximum value out of all the IT investments made so far. As there is a growing disconnect between business and IT, the envisioned advantages of IT could not be fully leveraged by businesses, and, hence, the expectations from IT is consistently on the rise. As IT is being

overwhelmingly recognized as the best business enabler, there will be constant insistence on unearthing ways and means of simplifying and streamlining IT-sponsored business acceleration, automation, and augmentation. There are a few pioneering business-impacting technologies such as service-oriented architecture (SOA) and cloud computing. The seamless convergence of these two aspects in association with the enterprise architecture (EA) methods results in service-driven and cloud-centric enterprises.

12.8.1 Emergence of Cloud Enterprises

The cloud rage is definitely on. Global corporates are seriously strategizing to join in the well-entrenched cloud journey. The ultimate goal for global organizations is to reposition and rebrand them as cloud enterprises. A dazzling array of cloud concepts, technologies, platforms, products, processes, and practices are being smartly recognized and utilized overwhelmingly by companies of all sizes across the globe in order to be immensely benefited out of all the remarkable achievements and accomplishments in the cloud arena. Though appropriating the path-breaking cloud idea presents new opportunities and fresh possibilities, there are some noteworthy pain points in the long and arduous journey. Here our cloud technology consultants can chip in their cloud experience, education, and expertise-spurred insights to articulate workable and specific schemes to facilitate business transformation.

12.8.2 Cloud Advisory and Advocacy Services

The Cloud Advisory Services are for organizations to arrive at the right cloud architecture to fit their current and future compute, communication, storage, and analytic needs. Through a comprehensive and closer engagement, it is possible to come out with a pragmatic cloud strategy that helps to craft a specific cloud adoption roadmap with timelines and a migration plan. The engagement comprises several phases utilizing a structured delivery approach. The major activities in this engagement are listed below.

12.8.3 Cloud Enterprise Architecture (CEA) Services

Cloud, being the most pioneering and penetrating idea on the enterprise space, is to seamlessly link up with enterprise architecture to lead the emergence and establishment of cloud enterprise architecture. The prominent architectural modules of CEA are:

- Cloud Business Architecture
- Cloud Application Architecture
- Cloud Information Architecture

- Cloud Technology Architecture
- Cloud Integration Architecture/Cloud Brokerage Services
- Cloud Security Architecture
- Cloud Management Architecture
- Cloud Governance Architecture

The other noteworthy developments in the cloud space include the following:

- Cloud Databases (The Hybrid of Transactional and Analytical Databases)
- Big Data Computing (Hadoop, HDFS, and NoSQL Databases)
- Cloud Backup and Archival
- Cloud Analytics
- Next-Generation Clouds (Mobile, Device, Sensor, High-Performance, Service, Social, Science, Data, and Knowledge Clouds)

12.8.4 Cloud Adoption Assessment Services

This is the foremost step toward the cloud embarkation journey. Assessing, analyzing, and articulating the actionable insights about the IT landscape go a long way in facilitating the cloud adoption. That is, analyzing the currently running IT infrastructures, platforms, databases, applications, and support systems very diligently in order to come out with a comprehensive report is the key and details which systems are good for cloud and which are not. Consultants can also identify and analyze workloads to get a deeper understanding of the IT landscape. They can provide the right and the relevant details about how cloud-enabling of some systems creates fresh opportunities and new possibilities. They can identify the positive as well as the negative implications through this transition and provide the expected cost benefit insights when embracing the cloud model. The challenges, risks, and security threats in adopting the identified and agreed cloud model will be expressed in detail. The best practices, tools, and frameworks with a detailed map of migration opportunities can be also shared. The predominant questions include the following while plunging into the cloud readiness evaluation phase:

- How far is the new cloud model helpful for your organization?
- What systems and subsystems are suitable for cloud-enablement?
- What kind of cloud delivery/deployment models are suitable for the organization?
- How is the modernization/migration to cloud going to happen?
- Whether to go for public cloud provider or to set up a private cloud or to transition the existing data center into a private cloud?
- What are the business, technical, and user benefits?
- What would be the TCO and RoI of cloud embarkation?
- How to enable tighter cloud security?

- How to manage and govern cloud service portfolio?
- How to explore new avenues for fresh revenues?
- What are the pros and cons of the proposed cloud model?

12.8.5 Cloud Design Services

Consultants can design the futuristic cloud infrastructure by leveraging the open and industry-strength standards, design patterns, and best practices gained through the empirical evidence and practical experience. They can develop private clouds quickly through cloud appliance/cloud-in-a-box and prefabricated hardware, software, and service modules.

There are some real concerns for organizations about moving their IT assets and artifacts to public cloud [8] due to the persisting security and privacy challenges. There are other worries such as the lack of accountability, auditability, controllability, and visibility. Hence, corporates are also working toward building a private cloud for the above-mentioned reasons. The primary factors behind developing private clouds are that they are completely secure and accessible to only your employees and have heightened flexibility, controllability, and visibility. All the underutilized and unutilized compute machines get consolidated, centralized, virtualized, and automated to establish private clouds. The availability, scalability, and performance of IT resources can be fine-tuned quickly by your own administrators, whereas the utilization through resource consolidation, management, and sharing can go up significantly. The regulatory compliance can be achieved without much sweat. The existing data center can be fully analyzed in order to recommend a series of plausible and pragmatic measures toward operational and cost optimizations. The prime focus will be on defining the maturity and gaps in the following:

- Virtualization – server, storage, network, desktop, application, and service
- Capacity planning, workload distribution, and management
- Management and monitoring of virtual resources
- Infrastructure elasticity and application scalability
- Auditing and security compliance

As a part of this service, consultants can do a kind of a pilot application migration to the cloud. The consulting team would want to undertake a pilot exercise with a handful of applications to uncover any hidden risks and process changes and also help justify the business case for migration to the cloud. It is beneficial to perform the first migration properly, therefore forming a baseline process to evolve before further application migrations to the cloud. They can provide experienced cloud consulting services to plan, oversee, and implement your pilot or phased migration. The purpose of any pilot is to ensure a seamless shift of operations to cloud services.

12.8.6 Cloud Development Services

This is about designing, developing, debugging, deploying, and delivering enterprise-class, mission-critical, service-oriented, and cloud-based applications for both public and private consumption.

12.8.7 Cloud Management and Operations

On one hand, there is a growing portfolio of services and applications while, on the other, virtualization has facilitated the dynamic formation of a number of virtual machines out of physical machines. Thus, besides these virtual resources, there would be a need to effectively manage the physical resources too. Thus, cloud service and infrastructure management becomes hotter as days go by. There are cloud management solutions, cloud service brokers, cloud governance engines, service delivery platforms (SDPs), etc. fast emerging for sustaining the cloud adoption.

This is all about understanding the challenges of managing business applications and infrastructure over cloud environment and pondering about viable and value-added solutions. Consultants can empower companies to reap maximum returns on their investments, ensuring the quality of service (QoS) requirements as per the agreed SLA, guaranteed uptime and zero disruption. System integrators and independent service vendors are investing heavily in building intellectual capital in this potential space. And through purpose-specific partnerships with different cloud product vendors, they can fulfill the industry expectations of setting up and sustaining cloud application and infrastructure management solutions.

12.8.8 Cloud Implementation and Migration Services

This is all about leveraging the hard-earned expertise and experience in areas like application rationalization and reengineering, infrastructure virtualization, data center consolidation and optimization, and migration services to provide quicker and quality implementation and migration of customers from traditional data centers to flexible and futuristic compute environments such as public, private, or even hybrid cloud.

For building private clouds, one has to leverage the standards-compliant infrastructure cloud solutions such as Eucalyptus, Nebula, OpenStack, and CloudStack for private cloud realization. Then, the process of application modernization to be cloud-ready has to be initiated to deploy the SaaS-enabled IT solutions and products in the clouds.

12.8.9 Cloud Protection Services

This is all about understanding, analyzing, and approaching all sorts of security needs at different levels and layers: physical security, network security, and application and data security.

12.8.10 Cloud Support Services

This is a support service for continual service improvement and application life cycle management (ALM) services.

12.8.11 Cloud Center Transformation Implementation Services

There are several newer services emerging and evolving in the expanding cloud landscape, namely,

- Cloud Infrastructure Design, Integration, and Planning
- Cloud Center Network Architecture and Engineering
- Cloud Center Storage Architecture and Engineering
- Cloud Center Optimization for Lean and Green Centers
- Enabling Hybrid and Connected Cloud Centers

12.9 Key Points to Consider When Migrating

Look for an established vendor with a track record [9]. A cloud vendor that is well established will have a wider breadth of knowledge and deeper insights into potential pitfalls than a smaller less-established vendor. They are also more likely to have higher security standards, a better range of services, more resources available to meet peak demand, and a better quality of support and training available for their users. The support and training provisions alone are likely to make the biggest difference to a customer that is new to the cloud.

Does the project really need to be migrated? If management and customers are happy with the current hosting arrangements, and particularly so if they are cheaper than the cloud option, then there is no reason to move. Cloud migrations are usually only necessary when considering large-scale hardware purchases in order to sustain or scale existing in-house projects or enable new projects to take place. Such migrations are only value for money if the perceived benefits of the migration outweigh the costs of performing it.

12.9.1 Consider Data Security

Security check is an important pre-check before embarking on the journey to cloud. With any outsourced service, security is paramount. If security testing has been done properly at the application level, then the difference in the chance of it being hacked as a result of bad programming whether it be hosted on the cloud or in-house is negligible. Application security aside, there are three key areas which require attention when deploying a service on any outsourced provider's hardware, cloud, or otherwise.

1. *Encryption and Authentication of Communication* – The only way to avoid data being snooped on and intercepted while traveling between the corporate network and the external application is to encrypt it. Various well-established and easy-to-implement technologies exist to do this, such as the HTTPS protocol for encrypting websites. At the same time, steps must be taken to ensure that only authorized users are able to access the external service, which by default would otherwise be visible to the whole Internet. Again many established technologies already exist to do this, such as OpenAM.
2. *Virtual Firewalls, Virtual Separation, and Encryption at Rest* – Steps must be taken to ensure that the virtual server is only accepting connections to the application of interest. Allowing access to unrelated software increases the chances of hackers gaining access through routes other than the application and increases the maintenance overhead of ensuring these other access points are all suitably patched with the latest security fixes. In a virtualized environment, true separation of virtual machines within a physical server must be guaranteed so that two virtual machines on the same physical server cannot accidentally access each other's resources. In the unlikely event of a successful hacking attempt, make it a point to encrypt any sensitive data while it is stored on disk so that intruders are unable to read it.
3. *Physical Firewalls* – These are also known as doors, locks, keys, walls, and security guards. The easiest way to take over any machine is by walking up to it and taking physical control of it, so the most effective defense against the most determined intruder is a good physical security system at the data center. Any data center worth its salt will be at the very least certified to international standards of physical security.

12.9.2 Data Transfer

IaaS clouds are Internet based; therefore there is usually only one method of getting data into them: uploading files across the Internet. Many Internet connections used by smaller businesses offer far slower upload speeds than download speeds, and it can take an eternity to upload even a gigabyte of data. Even the fastest corporate networks struggle to upload a few tens of gigabytes within a reasonable timeframe,

e.g., a dataset from a DNA-sequencing machine. If the transfer of large datasets to and from the migrated software is a requirement, then careful consideration needs to be made as to how this could be achieved, reduced, or avoided. Some IaaS providers offer the option of shipping hard drives of data to them to avoid these upload bottlenecks, but be aware that shipping delays and workload at the vendor's data center may mean the process is far from instant. The best approach is to take a very close look at the data transfer requirements and see if they can be minimized, e.g., by preprocessing large datasets locally to produce a smaller summary dataset for upload. Conversely, the cloud can be a good way of improving download speeds for customers using an application. Cloud vendors can deploy an application on whichever of their physical servers are closest to a customer's location. This can make a big difference to the response times customers get from the application.

12.9.3 Data Storage and Location

Cloud data storage can be expensive, particularly for very large quantities of data, so consideration should be given to data retention policies. Should old data be archived off cloud to a tape library or other external resource? Is raw data needed at all or are summaries sufficient? While not hugely expensive, movement of data within the cloud does cost money in terms of internal bandwidth charging, depending on the vendor, so applications should avoid moving it around unnecessarily. Shared data resources such as shared drives or central relational databases can be more effective than directly copying data between virtual machines, particularly for data sources that are usually accessed randomly or partially rather than sequentially or in their entirety.

12.9.4 Scaling

The scalability of cloud applications is not something that magically happens upon deployment (at least, not in IaaS – although PaaS deployments of single applications do inherit a certain amount of scalability from the host environment). Applications have to be placed behind load balancers/auto-scalers within the cloud in order to be scaled up on demand. Some cloud vendors offer these as part of the service, others require the installation of third-party tools, and, however, most scaling and balancing solutions incur some additional expense. Once the application is behind a load balancer or auto-scaler, the application itself needs to be aware that it could be scaled. If we are migrating an in-house application that already sits behind an in-house load balancer, then chances are that very little will have to be changed to support scaling in the cloud. For applications that have not yet been load balanced in house, developers will need to assess the code to ensure that it can cope with a changing environment. How will user sessions be persisted? How will they coordinate access to any central data resources in order to avoid conflict?

12.9.5 Service Level Guarantees

The first question to ask any cloud vendor is what their availability guarantee is and what recompense there might be if they fail to live up to their claims. A cloud vendor failing to provide the agreed service is the worst possible situation for any cloud application to be in. Particular attention should be paid to the processes in place in case of vendor collapse or takeover. Once confident of the vendor's service guarantees, the next check is to look at vendor backup plans. Do they take on- or off-site backups? What is their disaster recovery plan in case of loss of a data center? Do they guarantee to recover the contents of the virtual servers, i.e., the applications and data, or will they only recover the base operating systems? Independent off-site backup plans should be built with these answers in mind.

12.9.6 Upgrade and Maintenance Schedules

Cloud vendors will need to update their systems from time to time to install the latest security patches and new features. Applications built on top of the cloud will need to be aware that these patches take place and have plans in place to ensure that they will not be adversely affected after the upgrade. Vendors often give an option to decide when the upgrade will take place, subject to a fixed final deadline when it will happen anyway, so application developers should carry out testing well in advance to ensure service continuity. Likewise, if a vendor schedules planned downtime of cloud services to perform maintenance, try to schedule application maintenance windows to coincide with this in order to prevent excessive numbers of outages for customers using the application.

12.9.7 Software Architecture

Traditional applications are designed for traditional hardware configurations. Cloud applications are designed for cloud infrastructure and features. The two are not necessarily equivalent. While it is entirely feasible to take a traditional application and simply copy it to a cloud-based replica of its original environment, this is not always the most effective use of cloud functionality. Questions need to be asked regarding the choice of infrastructure: Does it need a grid/cluster equivalent or can cloud alternatives such as Hadoop or self-instantiating instances provide a better service? Does it need an integrated load balancer or can the cloud's default load balancer suffice? Does it need database replication to distribute requests or can a single larger virtual database server handle all the traffic on its own?

12.10 Technical and Business Cases of Cloud Embarkation

Businesses are under tremendous pressure for IT-enabled transformation as per the soaring expectations of customers, clients, and consumers. The buzzwords such as transformational projects and technologies are in the limelight these days. There are next-generation concepts such as smarter enterprises (IBM), smarter organizations (Infosys), instant-on enterprises (HP), and twenty-first-century virtual corporation (Wipro), being pronounced, proclaimed across the globe very often. Essentially it means organizations globally should have:

- Deeply and extremely connected, automated, and shared (collaborative enterprises)
- Real-time and real-world understanding and fulfillment of customers' needs (context-aware and customer-centric enterprises)
- Insights-driven and instant-on enterprises (cognitive enterprises)
- Micro and high-resolution management of enterprises (compact enterprises)

12.10.1 Roles of Cloud in Structuring/Sustaining Next-Generation Business Enterprises

Cloud primarily represents infrastructure transformation. As widely and overwhelmingly accepted, infrastructure optimization plays a very important role in realizing enterprise/business transformation. Thus, the cloud technology is being received as a transformative and augmentative method by business executives as well as infrastructure architects. Apart from a lot of cost savings especially capital cost, the much discussed, dissected, and discoursed cloud paradigm brings to table a litany of benefits as enumerated below for the benefit of readers. Corporates mandate their own IT divisions to spur the cloud journey whereas others tie up with IT service organizations. Jain [3] says companies benefit by using an outsourcing service provider for their IT transformation because suppliers can:

1. *Better Manage Cost and Risks* – Outsourcing allows buyers to do the job with the least possible risks and at the most optimized cost through their service provider's global delivery model. IT service organizations and system integrators through their long years of service delivery experience are capable of overcoming all kinds of impediments toward bringing the designated success for their esteemed customers.
2. *Support Business Innovation* – Due to the repeatable and reusable expertise gained over the years, service providers could understand their customers' current and future needs precisely. Accordingly right innovation can be quickly incorporated into the solution being supplied. All kinds of architectural, technical, and infrastructure complexities can be taken care of by service providers by simplifying

IT through application rationalization and infrastructure consolidation. The IT agility being derived out of this transition leads to business agility.
3. *Increase Market and Mind Shares* – Service providers help their clients increase their touch points with their customers, reduce their time to market, and help them deliver more efficient and premium services to their customers. The brand value and competency of business are bound to reach greater heights instantly.
4. *Enhance End-User Experience* – End-user experience of business services is essential for determining the business success and survival. There are several nonfunctional (quality of service (QoS)) attributes such as performance, scalability, availability, security, simplicity, accessibility, and usability. How easy and fast a business transaction is being accomplished by business users matter most in this competitive environment.

12.10.2 Other Increments and Improvizations

Real-time and real-world analytics for deeper visibility, tighter controllability, and micro management that in turn provide the following:

- Speed and streamline the productization, process improvization, better planning, and execution activities
- Lower TCO and higher RoI
- Enhanced customer loyalty and delight for repeat business
- Synchronized and single customer views and voices
- Visualizing, planning, and implementing people-centric and instantly integrated and insightful products and services
- The realization of smart, sophisticated, and situation-aware applications
- Facilitating personal and preferred service delivery and consumption
- The neat and nice reality of anytime, anywhere, any device, any network, any media, any service, etc.

12.11 Conclusion

It is a well-known and widely accepted fact that cloud computing has the innate potential and long-term promise to penetrate deep down into every industry segment to be extremely pervasive and persuasive. The raging cloud idea is being massively adopted and adapted accordingly across the globe. It is being made fit for purpose to suit the evolving requirements by individuals, innovators, and institutions. Besides a variety of generic as well as specific cloud infrastructures and software applications, a cornucopia of cloud-centric processes, platforms, patterns, and practices are being produced by product vendors, value-added resellers, system integrators, universities, public sector organizations, and private research labs, to support and sustain the cloud paradigm to make it more relevant, robust, resilient, and responsive.

With clouds, we have a stream of promising and novel business models that include next-generation delivery, pricing, subscription, deployment, and consumption models, and, hence, there is an unprecedented rush for cloud adoption. Businesses are seriously evaluating the irresistible cloud option. Professionals are working overtime to identify the constraints (internal as well as external) and collaborating with one another toward unearthing competent technology solutions to overcome the identified limitations. It is made clear by leading market watchers and analysts that every single company and corporate will have substantial and strategic presence in clouds in the days to unfold. Cloud migration strategy is one cool and core requirement for efficiently utilizing the emerging and enlarging cloud landscape for the betterment of tomorrow's enterprises.

References

1. Laszewski, T.: A brief introduction on migrating to an oracle-based cloud environment. http://syngress.com/phishwrap/phishwrap-102011/a-brief-introduction-on-migrating-to-an-oracle-based-cloud-environment/ (2011)
2. Cisco: Planning the migration of enterprise applications to the cloud. http://www.cisco.com/en/US/services/ps2961/ps10364/ps10370/ps11104/Migration_of_Enterprise_Apps_to_Cloud_White_Paper.pdf (2010)
3. Rosenthal, B.E.: How outsourcing service providers enable business growth through IT infrastructure transformation. http://www.outsourcing-center.com/2010-09-how-outsourcing-service-providers-enable-business-growth-through-it-infrastructure-transformation-article-39870.html (2010)
4. VMWare: Virtualizing business-critical applications with confidence. http://www.vmware.com/files/pdf/techpaper/vsp4-virtualize-biz-critical-apps.pdf (2011)
5. Khajeh-Hosseini, A., Greenwood, D., Sommerville, I.: Cloud migration: a case study of migrating an enterprise IT system to IaaS. http://arxiv.org/ftp/arxiv/papers/1002/1002.3492.pdf (2010)
6. Hillier, A.: The importance of policy in cloud migration. http://www.datacenterknowledge.com/archives/2011/09/13/the-importance-of-policy-in-cloud-migration/ (2011)
7. Deb, B.: Assess enterprise applications for cloud migration. http://www.ibm.com/developerworks/cloud/library/cl-assessport/ (2010)
8. Varia, J.: Migrating your existing applications to the AWS cloud: a phase-driven approach to cloud migration. http://media.amazonwebservices.com/CloudMigration-main.pdf (2010)
9. Holland, R.: Ten steps to successful cloud migration. http://www.eaglegenomics.com/download-files/whitepaper/CloudWhitePaper.pdf (2011)

Chapter 13
An Assessment Tool to Prepare the Leap to the Cloud

Leire Orue-Echevarria, Marisa Escalante, and Juncal Alonso

Abstract To keep software compatible and optimal with respect to the latest trends is not easy. Nearly 90 % of the software cost can be due to maintenance, and approximately 75 % of it is spent on developing new features to stay functionally competitive and relevant. The industry progresses through periods of incremental development interspersed with true paradigm shifts, and, therefore, legacy software must keep up the pace. However, software modernization is not a trivial issue, and, if improperly done, it dangers the business continuity and sustainability. Companies considering transition to the new paradigm of cloud computing need to have available an innovative and combined technical and business analysis on the maturity and prospect of the legacy application. This chapter presents an approach for assessing the maturity of these applications and the convenience of migrating to the new cloud paradigm. Two issues are taken into account: on one hand, the perspectives in terms of technical challenges, maturity, and effort of the migration are identified; on the other, the performance and business benefits in relation to the cost of the process are pre-evaluated before tackling the actual migration process. Perhaps, for the first time, the business value is directly attached to the technical performance.

Keywords Migration to cloud • Maturity assessment • Feasibility analysis • Cloud modernization • Legacy software • Multitenancy • Scalability

L. Orue-Echevarria (✉) • M. Escalante • J. Alonso
TECNALIA ICT – European Software Institute Division, Parque Tecnológico
Ed. #202, E-48170 Zamudio-Bizkaia, Spain
e-mail: leire.orue-echevarria@tecnalia.com; marisa.escalante@tecnalia.com;
juncal.alonso@tecnalia.com

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_13,

13.1 Introduction

New developments in the way services and applications can be delivered over the Internet have opened up huge opportunities for software vendors. The Internet is not only getting faster and thus data is transferred in a quicker manner but it is also becoming more reliable in what concerns transactions among customers and providers. This is making possible the offerings of basic IT appliances such as servers for storage or computing clusters as a service, that is, vendors provide the hardware and infrastructure and clients provide the data. The decoupling of responsibilities accelerates the development of new service platforms and software products.

Due to the fact that the innovation rate is accelerating, software products in the Internet era need also to constantly evolve. Take a look, for instance, at the last 5 years and how the way we work has changed. This is due to the breakthrough of cloud computing, smartphones, or social networks, among many other technologies and new approaches. Innovations in the technological space affect the systems that the software has to support or needs to adapt to. Innovations in the business space also affect the licensing usage and delivery model. Software products have to be improved and adapted with regard to these new circumstances but without disrupting the business continuity of existing customers.

Accordingly, more and more traditional software vendors notice the need to transform their current business and technology model in order to remain competitive and address new markets. Software-as-a-Service (SaaS) and cloud computing are seen as the most promising way to achieve this change:

- Availability of legacy applications through usage of service-oriented implementations may open new channels for these applications to be offered as services over the Internet, combined with third-party services, thus adding more value for the end customer. Furthermore, through the multitenancy usage of such applications, costs will be minimized, the client base will become wider, and the concept of separate implementations for each customer will disappear.
- Usage of cloud platforms (IaaS, PaaS, and SaaS) in order to reduce costs, both in terms of software produced and resources utilized. Besides, due to the increased scalability, dynamic adaptation to the demand and produced revenue maximization may be achieved.

However, this transition from software-off-the-shelf (often residing as legacy applications) to SaaS implies a tremendous endeavor comprising business, application, and technical issues.

In this chapter, the authors suggest that software modernization is a particular case of software migration. Here, a target software framework has been specified whereby the target specification can be considered much more modern than source specification. Managing such a software modernization is still a significant challenge in today's software life cycle. This challenge is usually considered as inevitable, unpredictable, costly, technically difficult, time and resource consuming, and poorly supported by tools and techniques or formalisms. The complete life cycle of

software, from requirements to run-time and delivery, has to be readapted to the new technological and business conditions, requirements, and challenges, since there is an increasing need for tools/means to support software evolution and adaptation as a key value for next generation service-based software modernization.

The first issue that all companies face is the decision whether to migrate their existing products or to start developing from scratch. Existing open methodologies [2–5] and proprietary migration methodologies [6–10] to service-based software mostly begin with a big bang approach or perform a feasibility analysis well advanced the migration process. Questions such as cost and effort of the migration impact of new business models in the company or return on investment (ROI) need to be answered before tackling the actual modernization. For doing so, an extensive qualitative and quantitative analysis must be made, taking the previous factors under consideration and deciding whether the newly transformed service will have a successful ROI. This cost-benefit analysis should consider technical and business aspects and evaluate in business terms the technical improvements.

If the estimates obtained with the previous analysis suit the expectations of the company and finally decide on the migration to a service-based software, reusing as much as possible from the old one, further challenges and difficulties will surely be faced, not only with respect to the usage of new technologies or architecture but also with respect to assumptions that companies usually took for granted and that are no longer valid.

This chapter presents an approach where assessing the maturity of an application provides information about the convenience or not of migrating to the new cloud computing paradigm, based on quantitative indicators, while always ensuring the company's business continuity.

13.2 Background

Transforming legacy applications into the services paradigm, both at business and technical levels, allows systems to remain mostly invariable while exposing functionalities to a large number of customers through well-defined service interfaces. However, not all organizations start from the same initial situation in terms of the maturity of an application or the organization at both the technical side of the migration as well as the business aspects. Likewise, the desired situation of each company or application after a migration has taken place can differ depending on the market to address, resources available to commit, time requested for the new product, etc. Based on all these aspects, specific and personalized modernization strategies should be applied for each specific application or organization. Furthermore, implementing this pre-migration analysis would provide the organizations with a powerful tool to know, before undertaking the actual modernization process, the estimated effort, time, or resources required, thus facilitating the decision-making process.

Khadka et al. [11] identify, in their systematic literature review (SLR), some common phases out of three evaluation frameworks: Butterfly [2], ADM [3], and

Renaissance Method [4]. These phases called Evolution Planning and Evolution Implementation and Management include tasks such as understanding the legacy and target system, feasibility analysis, and migration implementation. In this SLR, they have analyzed 121 distinct approaches from 2000 to 2010, but none of them include all phases and tasks necessary for a correct migration, and of course, none of them include issues related to business aspects.

The Service-Oriented Migration and Reuse Technique (SMART) [5] is one of the most widely used strategies to migrate to SOA. SMART process, developed by SEI at Carnegie Mellon University, helps organizations to make initial decisions about the feasibility of reusing legacy components as services within a SOA environment. SMART considers the specific interactions that will be required by the target SOA environment and any changes that must be made to the legacy components. To achieve this, SMART gathers information about legacy components, the target SOA environment, and candidate services to produce (1) a preliminary analysis of the viability of migrating legacy components to services, (2) an analysis of the migration strategies available, and (3) preliminary estimates of the costs and risks involved in the migration. SMART is vendor and tool independent and provides a clear roadmap of the steps to consider in the migration. Furthermore, this approach does not address important issues on a formal basis like QoS, security, scalability, and multitenancy, and it does not provide a complete solution execution for migrating legacy software.

The European FP7 project, REMICS, is also developing a methodology for the reuse and migration of legacy software to cloud, centered mainly on the technical aspect and nothing at all on the business aspect. REMICS migration methodology also incorporates a "requirements and feasibility" [12] activity area but with the focus on the technical requirements. REMICS does not provide estimates on how much effort or time is required to perform the migration, as the main purpose of this activity area is to gather the additional requirements for the migrated system, and to identify the main components of the solution and their implementation strategy.

13.3 Challenges

Previously mentioned open and proprietary migration methodologies to service-based software mostly begin with a big bang approach or perform a feasibility analysis well advanced the migration process. These methodologies and approaches leave aside the pre-migration phase or estimate a very high-level initial analysis not raising important aspects such as business model issues or organizational changes to be considered.

Hence, none of the current solutions provide an appropriate migration preparation strategy focusing on the information required to support the decision-making process when considering a software modernization process into loosely coupled systems provisioned as a service. Furthermore, they do not provide tools to automate this analysis. Additionally, none of the methods described above treat the business

model requirements organizational changes or legal and trust aspects, as well as their implications in the modernization process, essential to the cloud computing delivery model.

After the analysis of the current state-of-the-art and existing solutions, the authors have identified that it is needed a solution to assess the feasibility to migrate the current system so that decision makers in the companies can decide, based on objective indicators, the suitability and feasibility of the migration. This solution includes the following:

1. A benchmarking tool to analyze where the application to be migrated stands currently in terms of architecture and business models and where the application is aimed to be.
2. Based on this leap (the current situation as is and the desired future situation to be), a gap analysis is performed and indicators will be obtained. These indicators will include a cost-benefit analysis (including ROI and payback), impact of what this shift of business model implies in the company (in terms of processes, additional costs, personnel, new aspects that need to be considered, etc.), and an estimation of the effort needed to change the new delivery model (based on some indicators obtained from high-level reverse and forward engineering activities). These data will analyze the feasibility of the migration and management can take their decisions based on objective indicators.
3. Based on this approach, it is created a customized roadmap on how to embark on the migration for that specific application and desired target platform, as well as the degree of achievement available at all times for the developers that are executing the migration activities. This specific roadmap allows firstly evaluating and secondly determining how to efficiently implement and run the actual process.

13.4 Approach

In order to face up to the aforementioned challenges, the approach presented in this chapter, currently being developed and tested, provides a set of methods and techniques that will support companies in the assessment for the modernization of their software toward a cloud delivery model, sustaining them on the migration strategy and providing the required tools to analyze the impact of the potential transformation of the software in the company.

The modernization of the software and its delivery will be analyzed following two different, but intertwined, dimensions: one focusing on technology (architecture, performance, reliability, data schema, and so on) and another one on organizational and business aspects (pricing model, market addressed, organizational processes, etc.). This is rather significant since the business model offered by the organization (based on the delivery of software artifacts) will change from a product to a service.

After the assessment, the assessed organizations will be able to visualize a maturity map showing where the current business and service stands, as well as the

potential position (in terms of technology modernization and business model changes), once the migration has taken place.

In addition, this modernization assessment will support the analysis of such initial and desired situations through a set of impact assessment tools. The main purpose of these tools is to establish a collection of objective and measurable metrics and indicators on which to estimate the feasibility of the migration. Furthermore, the numbers will be presented in measurement units and concepts easily shared, recognized, and acknowledged by stakeholders. Summarizing, the main outcomes of this approach are:

- A method for characterizing the technical and business dimensions of the current legacy application, in particular those concerns related with its modernization toward a selected target
- A set of common metrics and indicators that characterize relevant technical aspects of the legacy application and the business model before and after the migration takes place
- A set of tools that will automatically evaluate the figures related to the modernization processes such as resources and effort required, impact in the company processes, estimated ROI and payback, and operational risks
- A modernization strategy with the activities to carry out in case the organization decides to continue with the modernization process after the figures are analyzed

In order to obtain the aforementioned outcomes, this approach is composed of four main phases that are detailed in the next sections:

- Business and technical modernization assessment
- Technical feasibility analysis
- Business feasibility analysis
- Modernization strategy definition

13.4.1 Business and Technical Modernization Assessment: Characterizing SaaS Applications and Providers

As mentioned before, this approach starts with an assessment which focuses on the characterization of the metrics and indicators (metrics weighed and combined) of the business and technical dimensions of the legacy application and the company, such as the pricing model, the targeted market, the product sustainability, SLAs, legal issues, metrics that describe the legacy application source code and data schema complexity, compliance with baseline legacy technologies, and gap estimation between legacy and target baseline. This assessment requires artifacts and knowledge related to source code and architecture, development process, GUI, source environment and desired environment, source and target infrastructure, and covered and uncovered nonfunctional requirements.

The execution of this assessment has been performed in several steps:

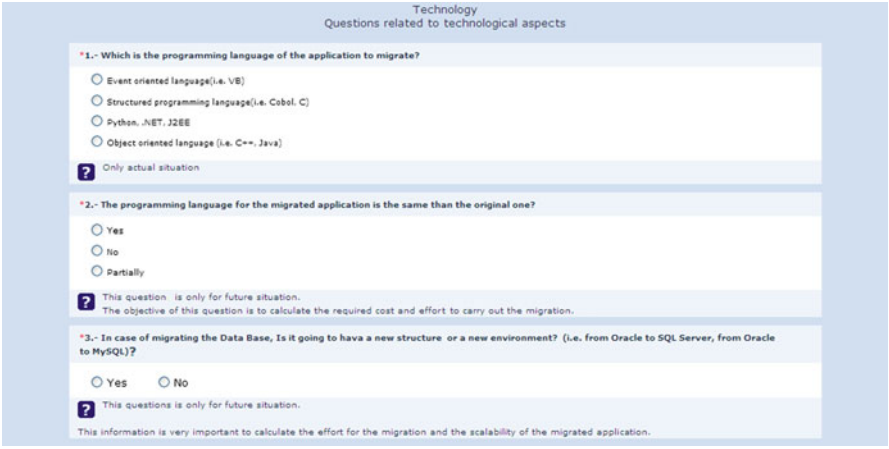


Fig. 13.1 Example of the assessment questionnaires

13.4.1.1 Step 1: Questionnaire

Through a set of questionnaires (Fig. 13.1), the modernization of the software and its delivery will be analyzed under two different, but related, dimensions: one focusing on technology (architecture, performance, reliability, data schema, and so on) and another one on organizational and business aspects (pricing model, market addressed, organizational processes, etc.).

The questionnaires are composed of a set of questions related both to the current situation and the desired situation, that is, how the application and business model shall behave once the migration takes place. The questions have been designed based on study of different states of the art and best practices, tools and techniques that analyze the different aspects that characterize the maturity of an application, be it cloud or not. Questions are classified by importance. Three groups have been created to analyze the current situation questions and four groups for the desired situation:

- Type 1: Questions related to relevant aspects for being a cloud compliant solution
- Type 2: Questions related to important aspects for being a cloud compliant solution
- Type 3: Questions related to not so important aspects for being a cloud compliant solution
- Key Question (only for the desired situation): These few questions are related to key aspects for being totally cloud compliant (multitenancy, SLA agreement, licensing model, etc.)

These user-friendly questionnaires can be answered by a person with a technical role, a person with a more business-oriented role, or a person covering both roles. The main requirement is to have a good knowledge of the application in terms of architectural issues, programming language, security, SLA fulfillment, help desk,

maintenance, privacy and trust practices, marketing, business model, pricing model, and performance and reliability.

In order to facilitate the completion of the questionnaires, the questions related to technological aspects are classified in the following groups:

- Architecture and programming: that is, programming language, database type
- Operating systems and interoperability such as how the application is installed in different OS, specific hardware needs, and how the application interacts with others
- Service level agreements (SLA), that is, application monitoring and performance indicators
- User management and security such as user profile management
- IaaS (i.e., migrated application deployment and hybrid, public, or private cloud)

And the classification for the questions related to business aspects is:

- Monitoring: type of elements to monitor the usage, interactions with other applications
- Billing: automatic payment, type of elements for billing
- Provision: licensing model, new version provision, more than one business model
- Users interaction such as help desk type and feedback mechanism

13.4.1.2 Step 2: Analysis

Based on the data collected via the questionnaires, an analysis is performed. Similar to the evaluation of quality criteria as motivated by the ISO 9126 quality model standard and used in the methodology of the "bidirectional quality model" [13], this approach, the Quadrant, measures several metrics by evaluating questions and checklists. The measured metrics are weighted following a certain criteria and aggregated into indicators of interest, which define a characteristic used to calculate the maturity of the business model and the maturity of the technology model, both before the migration and after the migration take place.

13.4.1.3 Step 3: Presentation

The next step is to present the results in a graphical manner. After several pilots, the authors found that an adequate way to present the questionnaires' result is by means of a quadrant. An example is shown in Fig. 13.2. In order to automate this process, as much as possible, for each question, the answers are positioned in the quadrant. The quadrant has three main sub-quadrants. A weight is assigned to each answer for positioning it in one of these sub-quadrants, following this schema:

- 1: sub-quadrant 3
- 0.66: sub-quadrant 2

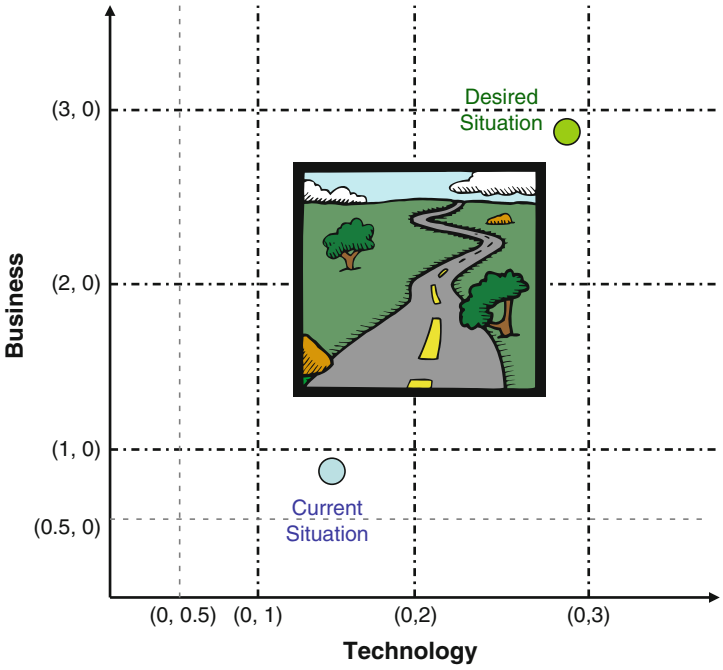


Fig. 13.2 Example of how the position of an application is shown in the quadrant

- 0.33: sub-quadrant 1 (next to sub-quadrant 2)
- 0.15: sub-quadrant 1 (next to axis)

Once all the answers are weighted and positioned, the overall position for establishing the maturity of the application is performed. The process followed is explained next:

1. For each type of question, count the answers in each sub-quadrant.
2. The sub-quadrant is determined by the number of answers of type 1.
3. The position in the sub-quadrant is determined by answers of types 2 and 3.
4. In the case of the desired situation, it is important to have in mind the so-defined key questions. If one of the answers of these questions is weighted as 1, the position is moved one tranche in the sub-quadrant.

Figure 13.2 shows different maturity levels which can be matched totally to the different positions in the quadrant. Currently, these maturity levels of the technology and business are categorized as follows:

Technology Axis

- (0,0) Monolithic: Interface logic, business logic, and data logic are in the same machine.

- (0,0.5) Client–server with a thick client (i.e., VB application), event driven; code tightly coupled to the interface. DB is in the local network or on a server outside but all the logic remains in the same machine.
- (0,1) Client–server with a thin client (i.e., J2EE application, 2-n tier), with no usage of web services; multiple DB instances.
- (0,2) Client–server with a thin client such as Mozilla, Opera, Chrome, or Internet Explorer (i.e., J2EE application, 2-n tier), with usage of web services; a unique instance of the DB; multiple instances of the application.
- (0,3) Client–server with a thin client; 1 DB instance; 1 DB application; n appearance customizations.

Business Axis

- (0,0) License (installment), support, updates, upgrades, and maintenance are paid under a different fee model than the license; no help desk; no SLA; no upgrade protocol and procedures.
- (0,0.5) Most revenues are obtained from sales of licenses. Even though, there exist some sales (less than 10 % of the total amount) that are made in a service form with a flat rate model.
- (1,0) Most revenues are obtained from sales of licenses. Between 10 and 20 % are from the product sale as service with pay per use, flat rate, hybrid pricing models. SLA is being defined. Upgrade protocol and procedures are being defined.
- (2,0) More than 60 % of the sales are from the product as a service. Help desk is institutionalized but not 24×7 and only in certain languages; existence of SLAs, upgrade protocol, but upgrades are still seldom, legal department.
- (3,0) One hundred percent of the sales are from the product as a service; existence of a 24×7 help desk, multilingual, marketing mostly done through the Internet (social media), SLA, upgrade protocol and procedures, and long tail.

Once the company is able to visualize both its current and desired situation, it is time to analyze what this gap means in terms of impact in the company. The purpose of the following two phases is to assess and help companies decide whether the migration is feasible for them based on objective numbers and indicators and not on subjective considerations both from technical and business point of view.

13.4.2 Technical Feasibility Analysis: Understanding the Application and Complexity

Technical constraints may impede or limit the migration of an application to a cloud-based environment, exposing it as a service. Some of these constraints derive from the legacy system itself while others come from the technology in which the legacy system was implemented. The result of this may be that migrating a legacy system is more expensive than building it from scratch.

A possible way to discern whether the migration is possible or not is a feasibility analysis. Existing migration methods to SOA like SEI's SMART [5] propose doing it mainly by means of questionnaires or interviews to key people leaving aside the use of supporting tools. The feasibility analysis performed in this approach, however, centers it on several tools with the main purpose of providing numbers and graphical images that will ease the decision of whether to tackle the migration or not. These tools and their purpose are described below:

- *Code analysis*: The goal of such an analysis is twofold: On one hand, the first goal of this code analysis is to represent the coupling of methods, types, classes, namespaces, or assemblies in varying sizes according to common metrics like lines of code (LOC), McCabe's cyclomatic complexity, CBO (coupling between objects), RFC (response for class) [14], RFC∞ [15], MPC (message passing coupling) [16], DAC (data abstraction coupling) and DAC1 [16], ICP (information-flow-based coupling) [17], and COF (coupling factor) in order to obtain other useful views of code cohesiveness and complexity. These views can be presented in a variety of ways such as a matrix, a dependency graph, or a tree. On the other hand, the second goal of this code analysis is to discern the dependency of the legacy software in third-party COTS and/or with other applications internal to the company.
- *High-level modeling*: This activity has as main goal to obtain the understanding of candidate functions or modules that might be exposed as services in an easy manner. The best way to obtain this knowledge is by modeling the application with UML in its different views, seizing the power of reverse engineering techniques. Also important at this stage is to analyze the database schema, data, and transaction model in order to select the best database architecture (RDBMS, NoSQL, or a hybrid solution) and migration strategy toward the chosen one.
- *Effort estimation tool*: Based on the desired target cloud platform, this tool provides an estimation of the work (effort) that will be needed to transform it to that target platform.

13.4.3 Business Feasibility Analysis: Understanding the Impact of Changing Business Models, Costs, and ROI

The goal of the business feasibility analysis is to provide the management with guidelines to decide to move forward with the migration based on economic factors. The approach followed for such an analysis has been by means of a cost-benefit analysis. Although a cost-benefit analysis has to be customized for each migration project, there are several assumptions that will always occur or at least will have a great probability. These are:

- (Reduced) Costs of no quality: More quality in the application as a service since upgrades are done more frequently and every time this happens, regression tests are performed. This is measured in terms of less rework (and human resources dedicated to it) and less time dedicated to solve customer complaints.

- (Reduced) Costs in traveling for maintenance and installation.
- (Reduced) Costs in marketing.
- Costs of cloud provider: Three options.
- (Greater) Margin by targeting new markets nonreachable beforehand.
- (Reduced) Labor costs, thanks to the use of (semi)automated tools that support the migration vs. programming the application from scratch vs. executing the migration by hand, with no tool support.

This analysis provides the management level with at least the following data:

- Net present value for the next 5 years
- Return on investment for the next 5 years
- Payback

However, not only economic factors are studied in this business analysis. The business processes within the company are also analyzed in order to determine the impact of the business model transition at process level. Cloud computing business model implies the redefinition of old processes and the creation of new ones, such as how to control and maintain the SLAs, customer care, and other support processes but also those related to the business core, the software development, design, and testing.

13.5 Modernization Strategy and Improvement of Life Cycle: Tackling the Migration

Once the metrics and indicators from the previous activities have been analyzed, the organization will decide whether to continue with the modernization process or not. If the organization decides to continue with the modernization process, a strategy indicating the activities to carry out needs to be defined. This strategy will provide the company with the needed roadmap in order to achieve the desired maturity expressed in the questionnaires.

The modernization strategy will be personalized for each migration considering the result of the pre-migration phase. Even though, there exist key issues which share some common aspects even if they will be considered and adapted to each specific situation. The performance guidelines of these key strategic areas are now presented.

- *Multitenancy*: It is a key aspect in all cloud compliant applications. The applications offer as SaaS, in addition to multiple users, are usually offered to different tenants. For this reason, it is relevant that the application could be multitenant. The ability to deliver software to multiple client organizations or tenants from a single share instance of the software application without implicating data integrity is multitenancy. There are several levels of multitenancy:
  - Cloud virtualization: only hardware is shared.
  - Simple application with sharing databases.
  - Simple application sharing databases.

- *Scalability*: The most important issue to obtain the scalability of the application is to define and separate the layers with static and dynamic data. The static data scale easily; otherwise, dynamic data require some specific mechanisms. If the stateless nodes are separated from the transactional ones, the problem is simpler. The best manner to scale an application is to maintain it stateless; in this way, several instances from the database can be created without synchronizing the state among them. The scalability could be dealt from two different points of view: application layer scalability and storage layer scalability.
- *Monitoring*: When an application is offered as SaaS, other relevant issue is to monitor the application usage and all the used resources. This monitoring is used to establish, for example, the fee policy. Several aspects could be monitored. These aspects could be classified as physical resources (CPU, storage) where the monitoring is useful for monitoring the use of the infrastructure by each tenant and application usage (access), concurrent users in order to be able to bill according to the real usage of the application.
- *Business Model*: Software-off-the-shelf business model cannot be applied directly without adjustments to the SaaS business model. Changes are required in the business model in order to ensure some degree of success in the service provision. SaaS business model may involve the automation of some processes in order to meet the customer expectations over some de facto standard in the interaction with SaaS; for example, online subscriptions, message-based support, or configurable mail notifications.

13.6 Use Cases: Practical Implementations

The presented work is currently being tested in eight different companies in Spain. Even though the sample is not big enough to ensure complete correctness, the approach has proven to be valid. Main future research topics and developments with respect to the presented work here have to do mainly with the technical and business feasibility analysis. In the following sections, two of these use cases are presented.

13.6.1 Use Case 1: Focusing on the Business Axis

This use case is from one small IT company which provides solutions for several sectors like banking, e-commerce, or aerospace industry. The idea of this company is to migrate a solution of a virtual help assistant in order to align it to the cloud business paradigm. This migration implies not only changes in its business model but also changes in some technological aspects in order to be able to apply it to the new business model. The first step performed was to describe the application as it is now and what are the main ideas for the migrated application. Table 13.1 reflects the main facts of this description.

Table 13.1 Use case 1. Main characteristics of the application

	Current situation	Desired situation
Technological aspects	SaaS oriented	Same architecture
	Three layers of architectural style	New module for managing users
	Two types of services:	Monitoring and control of the new methods for billing
	Client and statistics management	User authentication
	XML/JSON responses processing	
Business aspects	Initial fee for installation and a small use	More or less the same business model but monitoring some aspects in order to change and adjust the fees
	Fee for daily maintenance and support	Time
		Number of connections/conversations
		Definition of SLA agreements

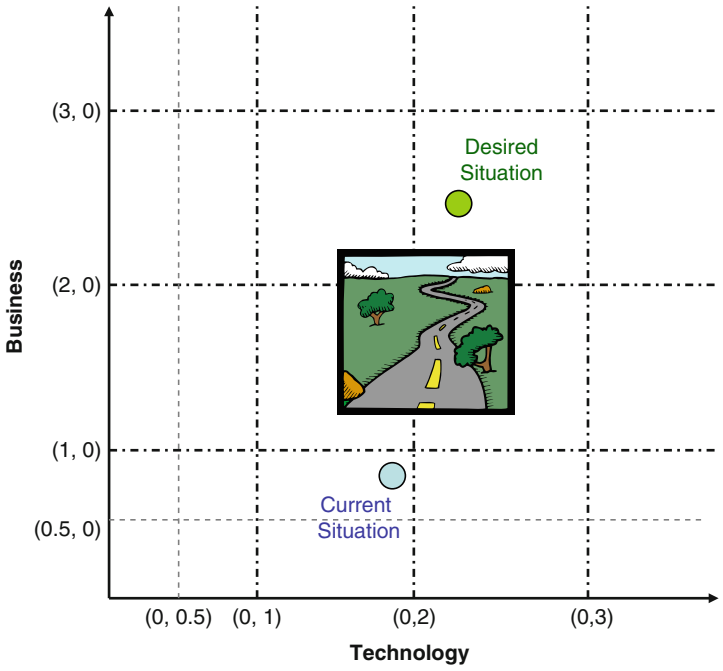


Fig. 13.3 Position of use case 1 application in the quadrant

Once the new scenario for the migrated application was defined, the technological and business experts within the company filled the questionnaires for characterizing both current and the desired application. Figure 13.3 shows the graphical representation of the maturity level for the application as it is now and as it is wished to be.

Table 13.2 Use case 1. ROI analysis

ROI measures		Y1	Y2	Y3	Y4	Y5
Cost of capital	10 %					
Net present value	€165,011					
Return on investment (%)		71,58	94,21	123,9	151,83	174,50
Payback (in years)	1,28					

This graphical representation means the following:

- Current position: [2 (technological), 1 (business)]
  - Technological – 2: Client–server with a thin client such as Mozilla, Opera, Chrome, or Internet Explorer (i.e., J2EE application, 2-n tier), with usage of web services; a unique instance of the DB; multiple instances of the application.
  - Business – 1: Most revenues are obtained from sales of licenses. Between 10 and 20 % are from the product sale as service with pay per use, flat rate, and hybrid pricing models. SLA is being defined. Upgrade protocol and procedures are being defined.
- Desired position: [3 (technological), 3 (business)]
  - Technological – 3: Client–server with a thin client; 1 DB instance; 1 DB application; and n appearance customizations.
  - Business – 3: 100 % of the sales are from the product as a service; existence of a 24×7 help desk, multilingual, marketing mostly done through the Internet (social media), SLA, upgrade protocol and procedures, and long tail.

Following the approach presented before, the next step to perform is the cost-benefit analysis (Table 13.2) in order to gather some data that will determine the feasibility or not to tackle the migration project. For this economic analysis, the following assumptions are made:

- Cost of capital (the required return necessary to make a capital budgeting project, such as building a new factory, worthwhile; cost of capital includes the cost of debt and the cost of equity) is 10 %.
- An implementation filter of 75 % in the first year increased by 5 % every year has been applied as correction factor.

From the point of view of business processes, this organization should take into consideration the following processes:

- Billing process: Improve the billing process in order to automate it based on the data obtained in the monitoring activities. This implies the integration or creation of a billing module in the application.
- Improve the processes related to SLA: Automated negotiation of SLA, monitoring SLA, and so on. Up until now, the SLA monitoring activities were manual and controlled by paper.

Based on the presented data, obtained after the business and technical feasibility analysis, this company decided to continue with the modernization process. This company implemented a modernization strategy where the most relevant points are:

- They have integrated an element for monitoring the aspects required for the new billing process, creating it from scratch. The main features monitored at application level have been the number of concurrent users and number of queries performed by any given user.
- They have inserted module for automatic billing based on the data the monitoring module provides. This billing module is able to also create and dispatch the bills based on the customers' preferences (once a month, once a week, whenever they made use of the solution, etc.).
- They have created a mechanism for user authentication, needed also for the two activities explained above.
- They have defined clear process and mechanism for SLAs.

13.6.2 Use Case 2: Migration at Technical and Business Level

This use case comes from an IT SME, who has developed and commercializes a product for instant messaging and chatting inside a company domain. In order to obtain a high-level knowledge of the application to be migrated, a short description of its architecture and business processes and model was requested. Using this material and the responses to the questionnaires, the snapshot of this application is depicted in Table 13.3. The migration this company wants to achieve includes not only business aspects as in the case presented above but also technological ones such as scalability, multitenancy, or inclusion of billing. This last issue, billing, has very strong implications in the business decisions and business model since one leads to the other and the other depends on the first. In this particular case, the real challenges of legacy migration to SaaS clearly appear.

The graphical representation of these situations is presented in Fig. 13.4.

Theoretically, this is what these positions mean:

- Current position: [2 (technological), 1 (business)]
  - Technological – 2: Client–server with a thin client such as Mozilla, Opera, Chrome, or Internet Explorer (i.e., J2EE application, 2-n tier), with usage of web services; a unique instance of the DB; multiple instances of the application.
  - Business – 1: Most revenues are obtained from sales of licenses. Between 10 and 20 % are from the product sale as service with pay per use, flat rate, and hybrid pricing models. SLA is being defined. Upgrade protocol and procedures are being defined.
- Desired position: [3 (technological), 3 (business)]
  - Technological – 3: client–server with a thin client; 1 DB instance; 1 DB application; n appearance customizations.

Table 13.3 Use case 2. Main characteristics of the application

	Current	Desired
Technological aspects	Three-tier architecture based on J2EE: Web Server (http) + Application Server + Database Server	Pure multitenant application but no restructured database (i.e., multitenancy will not be applied at database level)
	Business layer separated from the presentation layer	Architecture: composed of web services
	Not purely multitenant	Infrastructure: vCloud
		User resources monitoring: CPU cycles, RAM, processor, and number of simultaneous users
		Scalability requirements: application with high usage peaks at certain points of the day
		Automatic billing
Business aspects	Licenses by user + consultancy fees to install, customize, and deploy + maintenance fees	Pay per use: Billings based on the usage of the application (the price plan must be a combination of the number of concurrent users and infrastructure use)
	Help desk implemented	SLAs have to be defined and differentiated between which SLAs depend on the application and which depend on the infrastructure
	SLAs are not defined	

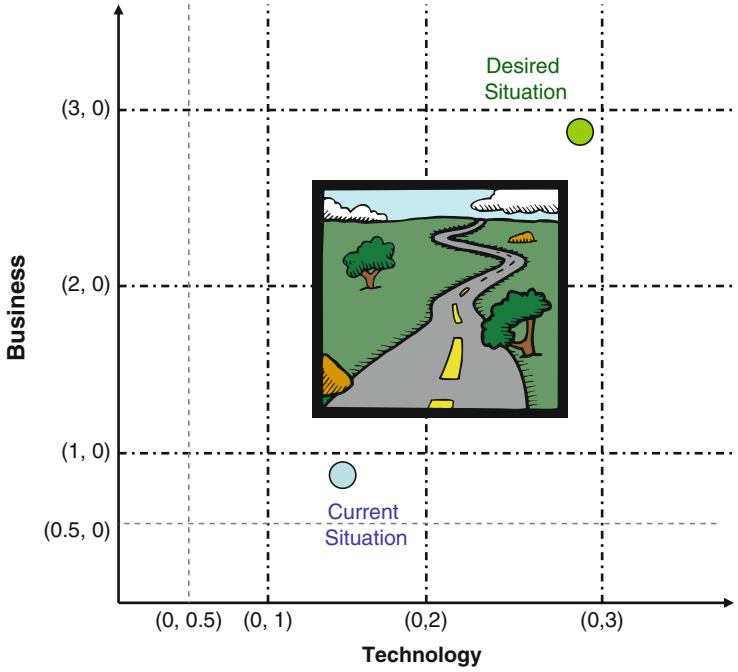


Fig. 13.4 Position of use case 2 application in the quadrant

Table 13.4 Use case 2. ROI analysis

ROI measures		Y1	Y2	Y3	Y4	Y5
Cost of capital	10 %					
Net present value	€18.189					
Return on investment (%)		41,78	59,00	81,29	101,75	120,85
Payback (in years)	2,66					

– Business – 3: 100 % of the sales are from the product as a service; existence of a 24×7 help desk, multilingual, marketing mostly done through the Internet (social media), SLA, upgrade protocol and procedures, and long tail.

Following the approach presented along this chapter, the next step to perform is the cost-benefit analysis (Table 13.4) in order to gather some data that will determine the feasibility or not to tackle the migration project. For this economic analysis, as in the previous use case, the following assumptions were made:

- Cost of capital (the required return necessary to make a capital budgeting project, such as building a new factory, worthwhile; cost of capital includes the cost of debt and the cost of equity) is 10 %.
- An implementation filter of 75 % in the first year increased by 5 % every year has been applied as correction factor.
- The migration is to be performed (semi)automatically, profiting from the use of reverse and forward engineering tools. This automation degree, based on previous experiences in other environments, is between 20 and 40 %.

The last step is the analysis of current business processes that need to be redesigned because of the migration. In this case, these are the ones that need to be modified:

- Upgrades and software updates: Currently, they do not have a mechanism for such defined. Updates are done "on the fly."
- Billing: For the time being, the bills are generated manually. With the new paying model of "pay per use," the billing activities need to be automated. A new billing component has been integrated in the application. In this case, contrary to the above use case, they integrated an open source one that they customized to their needs.
- Help desk. They now have the option to manage incidents also by e-mail or in any other participative way.
- Marketing strategy: The company mentions in their analysis that they want to open up new markets and reach a broader audience. Their current marketing strategy, based on traditional marketing techniques, is not useful any more. They have interest in viral campaigns and are setting up the mechanisms for it.

The processes that need to be defined from scratch are:

- SLA management: establishes the SLA parameters such as response time and QoS that until now were neither defined nor controlled.

- Privacy and security: mainly role-based access and location of data. The type of cloud they selected ensures them that it will be private, with access only to people inside the company.

13.7 Conclusions and Future Research

Currently, a large number of companies have identified the SaaS business model as the one to be adopted in the near future for their software products. Nevertheless, the performance of this migration, from a software-off-the-shelf-based company to a SaaS-oriented company, is quite an important challenge to face. Due to the complexity and the implications (at technical, business, and organizational levels), an incorrect transition of business models or technology migration may lead to a fatal situation, even bankruptcy or disappearance of the company.

The theoretical and practical solution presented in this chapter, as well as the envisioned future research directions, will bridge the transition toward the new era of Internet services providing the software industry methods and criteria that will enable important breakthroughs in software engineering methods and architectures, allowing flexible, dynamic, dependable, and scalable development; provision; and consumption of advanced service-based software.

This complete solution includes key features such as the following:

- A benchmarking tool to assess the current and desired company situation around SaaS, including both the technological and business dimension
- A set of tools to calculate the impacts of core issues involved in the migration process
- A risk minimized, stepwise procedure with tailoring strategies to migrate legacy software to a service-oriented paradigm
- A catalogue of essential components needed to deploy and provide successful and sustainable service-based software

As already mentioned, this approach is currently being validated in eight companies in Spain, with different maturity levels (current and desired), market spectrum, and technology. As it is still early to say, for the time being, no major shortcomings were encountered. This indicates that the overall solution is adequate and addresses software evolution problems. However, there is still a lot of work to do and improvements to make. These include:

- The set of questions have proved to be valid for the eight cases in which the approach was tested and piloted. However, as the environment was quite controlled in terms of current and target business models and migrated products and technologies, it is clear that if the scope is widened, a new set of questions may arise.
- Also, due to different time constraints, the analysis of the position of the products in the quadrant was performed manually. The idea is to automate this analysis as much as possible to deliver completely as a service over the Internet.
- The technical and business feasibility analysis need to be further developed and automated. The results given at the moment are disperse and the decision-making

process may be cumbersome depending on the maturity of the organization and how used they are to manage these data. The authors are focusing on a more integrated and dynamic approach where it can be seen what would happen if a variable changes its values.

Acknowledgments This work has been partly funded by the Basque Government project SMARTWORK in the Future Internet, by the Spanish Ministry of Science and Innovation through the project named mCloud: Advance migration to the cloud [18] and the FP7-ICT project ARTIST (Advanced software-based seRvice provisioning and migraTIon of legacy SofTware) under the contract number 317859 [1, 19].

References

1. http://cordis.europa.eu/fp7/ict/ssai/docs/call8objective1-2-brochure-web.pdf. Retrieved in May 2012
2. Wu, B., Lawless, D., Bisbal, J., Grimson, J., Wade, V., O'Sullivan, D., et al.: Legacy systems migration – a method and its tool-kit framework. In: The Proceedings of the Joint 1997 IEEE Asian-Pacific Software Engineering Conference and International Computer Science Conference (APSEC'97/ICSC'97), Hong Kong, China. December, 1997
3. Khusidman, V., Ulrich, W.: Architecture-Driven Modernization: Transforming the Enterprise. OMG. http://www.omg.org/docs/admtf/07-12-01.pdf. Retrieved in October 2011 (2007)
4. Warren, I., Ransom, J.: Renaissance: a method to support software system evolution. Paper presented at the 26th Annual International Computer Software and Applications Conference, Oxford, UK, pp. 415–420 (2002)
5. http://www.sei.cmu.edu/reports/08tn008.pdf. Retrieved in October 2011
6. ftp://service.boulder.ibm.com/s390/audio/pdfs/G224-7298-00_FinalMigratetoSOA.pdf. Retrieved in October 2011
7. http://www.oracle.com/technetwork/middleware/soasuite/overview/wp-soa-suite11gr1-2-129551.pdf. Retrieved in October 2011
8. http://download.microsoft.com/download/d/d/e/ddeb427d-dc05-4ab0-b47e-74f0a936d892/Real-World-SOA-At-The-Edge.pdf. Retrieved in October 2011
9. http://www.sap.com/platform/soa/adoptionprogram.epx. Retrieved in October 2011
10. http://media.amazonwebservices.com/CloudMigration-main.pdf. Retrieved in October 2011
11. http://www.staff.science.uu.nl/~khadk101/publication/LegacyToSoaMigration.pdf. Retrieved in October 2011
12. REMICS consortium: D2.2-REMICS Methodology (2011)
13. Simon, F., Seng, O., Mohaut, T.: Code-quality-management. Dpunkt. Verlag, Heidelberg (2006)
14. Chidamber, S.R., Kemerer, C.F.: Towards a metrics suite for object oriented design. In: Proceedings of OOPSLA'91, pp. 197–211. ACM, New York (1991).
15. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Trans. Softw. Eng. **20**(6), 476–493 (1994)
16. Li, W., Henry, S.: Object-oriented metrics that predict maintainability. J. Syst. Softw. **23**(2), 111–122 (1993)
17. Lee, Y.S., Liang, B S., Wu, S.F., Wang, F.J.: Measuring the coupling and cohesion of an object-oriented program based on information flow. In: Proceedings of International Conference on Software Quality, Maribor (1995)
18. http://innovation.logica.com.es/web/mcloud. Project Number IPT-2011-1558-430000. Retrieved in October 2011
19. http://www.artist-project.eu. Retrieved in October 2012

Chapter 14
Leveraging Potential of Cloud for Software Performance Testing

Krishna Markande and Sridhar J. Murthy

Abstract Considerable amount of cloud adoption is proven in the area of software testing. There are several providers in the area of hosted testing tools and services like IBM and HP. The functional testing tools cover a variety of aspects in functionality testing for Web applications, Web services and mobile applications. Another set of tools in relation to cloud environments is in the area of performance testing from cloud providers such as BlazeMeter, SOASTA, LoadStorm and Keynote. This chapter discusses various options available for leveraging cloud for performance testing of software applications. Additionally, the chapter covers some of the key considerations when selecting the solution providers. The chapter also presents a conceptual reference framework for leveraging the readily available public cloud infrastructures for optimized cost and high-volume load simulations from different regions for efficient testing. Such a custom framework can help in making the enterprise system services more market ready, which in turn aids in improving the overall quality of the enterprise systems. Proposed reference framework can be used in performance testing, regression testing, benchmarking and product certification of any on-premise or cloud-deployed services. This helps in reducing the overall testing cost and test cycle duration, thereby achieving accurate capacity planning.

Keywords Software testing • Performance testing • Software life cycle • Regression testing • Cloud testing

K. Markande • S.J. Murthy (✉)
Department of Engineering Services, Infosys Ltd., Bangalore, Karnataka, India
e-mail: Krishna_Markande@infosys.com; Sridhar_Murthy@infosys.com

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_14,

14.1 Introduction

Testing an enterprise application is an essential part of software development life cycle and enterprise application deployment environment. Applications and systems need to be tested for correctness and quality. Every system must be tested for performance under varied load to check if the system fails so that the systems can be constantly improvized. In software engineering, especially performance testing of enterprise systems determines the behaviour in terms of responsiveness, reliability, scalability, stability and predictability of the resource usage under a particular test cycle [1].

When developing and testing these enterprise systems, we find enormous challenges making it critical to economize overall capital and operational costs and time to market the solutions, so as to beat the competition. Along with the cost factor, key criteria of testing depend on usage of network, compute, storage resources and security as they all should be considered with equal importance. Cloud computing with its infinite computational, network bandwidth, storage resources along with elasticity and scalability spread across in multiple data provides an ideal infrastructure for enterprise systems for hosting and testing. Enterprises always need additional resources on hosting to meet the growing demands of enterprise as well as just-in-time needs of additional resources for testing [2].

Performance failures result in poor productivity, lost revenue, cost overruns and missed market windows, thereby damaging the customer relationship [3]. For example, Aberdeen Research conducted a poll and its report speaks that nearly 60 % of the organizations were unhappy with the performance of enterprise critical systems; respondents also told that the performance issues were causing corporate revenue leak by 9 % [4]. To address this concern, it is important to ensure that enterprise systems being rolled out are ready to handle real-time load scenarios and scale to proportions. This scale ideally cannot be simulated accurately using existing methodologies in own data centre, lab or hosted data centres. By leveraging the cloud service provider capabilities and in conjunction with the cloud-based testing framework, the enterprise systems can be quickly and efficiently tested by deploying the load injection test server instances in the cloud, which can simulate close to near real-time production scenarios.

Apart from meeting performance requirements and quality of systems, enterprises are looking for optimal costs [5]. Infrastructure cost is one of the main factors in performance testing and this can be reduced, by leveraging cloud infrastructure instead of own data centre or test labs.

Architectural details on the implementation of the cloud-based testing framework and its interaction with service components are described in detail below. Finally, details are provided on what is required for the users who use such custom framework to gain real benefits of this solution.

14.2 Factors in Performance Testing

It is imperil in rolling out any enterprise systems, products and services to users without adequate end-to-end testing. Nowadays, in global markets, users of systems are located all around the globe accessing these systems from various geographical locations, with different access channels and with varying bandwidth. As the number of users increase, these systems need to cater for higher volumes of load. Although enterprise systems are well architected, designed and deployed, it is highly complex to test these systems effectively, simulating near real-time user scenarios and volumes. Building a test automation system to simulate near real-time scenarios is a challenge in itself which has to address geographic distribution of users, network bandwidth, transaction volumes at peak intervals and combination of user scenario and personas.

While testing these enterprise system and services, it is critical and also important to minimalize the overall testing cost by reducing the overall testing time, infrastructural cost, accommodating frequent rework to the system and making the testing cycle more efficient and flexible. Cloud computing with its unlimited computational, storage, elastic nature and data centres spread across geography provides an ideal infrastructure for enterprise systems in testing and making the testing closer to near real-time production scenario.

14.3 Traditional Approach to Software Performance Testing

Performance testing with high-volume user load demands a large amount of infrastructural resources, bandwidth and cost. Also, infrastructure resources are needed for a longer duration throughout the performance testing cycle and also for every release phase [6]. Infrastructure resources are provisioned and configured within the enterprise data centres. Considerable effort and time are spent in procurement of resources, deployment and configuration of the test servers. Data centres which host the test servers to test the enterprise application do not have complete governing mechanism to check the infrastructural resources utilized and the service opted for [7].

The advantage of a hosting test server and application in the same enterprise data centre is that complete infrastructural resources requested for testing the enterprise systems are allocated. Other applications in the data centre are trusted and not impacting the performance of the systems under test. Dedicated test servers have dedicated IP address, bandwidth, storage and its own server rack for overall performance. Enterprise system and test servers are completely trusted and usages of the infrastructure are limited to the test plans [8].

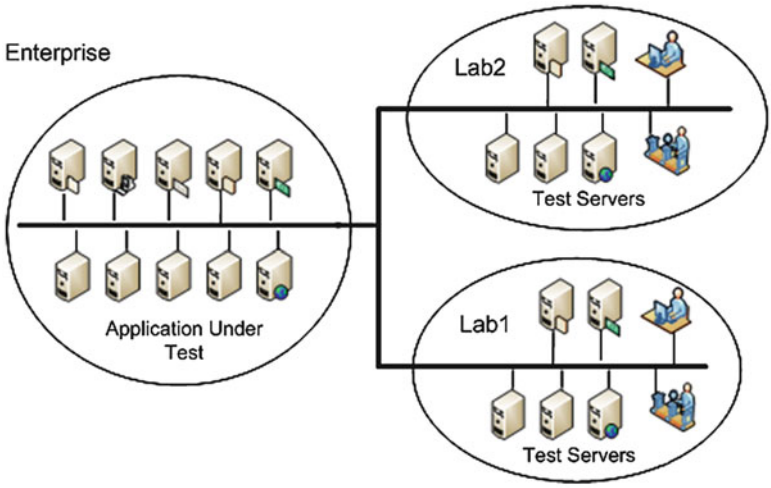


Fig. 14.1 Systems and test infrastructure are within the enterprise

To get the features mentioned above, dedicated hosting of servers in the data centre is the solution, but all these come at a price and the enterprise needs to pay for all the resources irrespective of the usage as servers are not completely utilized during lean test cycle days.

Most large enterprises procure the complete infrastructural needs of the application and allocate them to a specific application as depicted in Fig. 14.1 that shows an application under test. As the need of the application grows, additional servers are added to the existing resources. Rarely there will be any removal of the resources or consolidation of servers as most enterprises do not want to disturb already-working model. Over a period of time as additional features are added to the existing product, the deployment topology of the application under test grows and at the same time users of the application also grow, to meet the demand additional servers might have to be provisioned at appropriate time. Automated testing of the application and manual testing of the application happen at periodic release cycle.

Testing of the enterprise systems is a time-bound work: once started, a set of test cases and scripts are run and the reports are generated to know the health of the systems. The infrastructure resources needed for performance testing are always fluctuating, and enterprise data centre might lack the flexibility and elasticity to set up the test infrastructure quickly and optimally as per needs of the test plans [9].

The test infrastructure to test the application needs to be adequately provisioned to measure the quality of service (QoS) offered by the AUT. If either of the enterprise system or the test infrastructure is not provisioned, it might not serve to provide right results; extrapolation of reports might be one solution, but it does not address most of the issues faced by the application under real-time traffic conditions.

While most enterprises are recognized for the success of their applications, testing on the adequately provisioned test infrastructure is required; few have the required infrastructural resources to emulate peak load demands in their test lab.

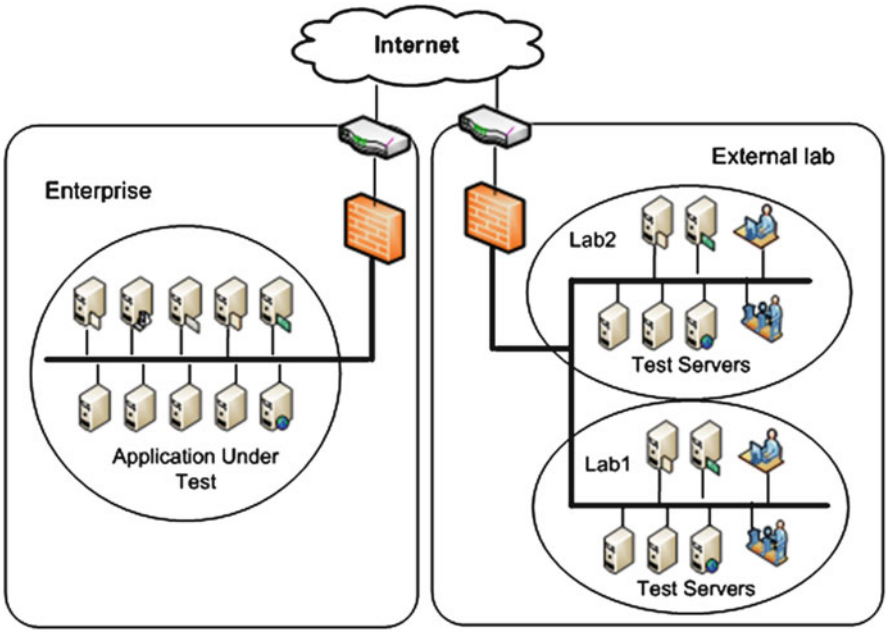


Fig. 14.2 Systems inside the enterprise and test infrastructure are hosted externally

To meet the demand of the growing needs of servers for simulating high load, external service providers are opted for. Service provider charges for test infrastructural resources allotted irrespective of their utilization [10].

Depending on the phase of the performance test cycle, external infrastructural resources can be provisioned to meet the testing needs of the application. During load testing of the application, a set of resources can be provisioned and the same set of infrastructural resources can be reused for stress, endurance and isolation testing. Figure 14.2 showcases two separate test infrastructure setups for testing of applications. Enterprise applications are hosted in the enterprises' data centre and the test infrastructures are hosted on external data centres. Lab 1 is used for performance testing on one of the application and Lab 2 is used for performance testing of another application belonging to the same enterprise; appropriate network and security constraints are adequately provisioned to meeting the needs.

An enterprise has complete control over the services and applications under test in this model. Test servers are utilized as needed by the test planning cycle. Test servers are configured only once with the same enterprise system, and configuration details are used in every test cycle.

System changes have to be manually updated and maintained at the test severs. Test servers at times might have inefficiency due to bandwidths, maintenance cycle at the hosted environment, greater security risk of exposing the enterprise servers to hosted environment, high costs and resources not being fully utilized during the lean test cycles.

14.4 Performance Testing in the Cloud

Load testing should guarantee the performance, availability, scalability and reliability of enterprise applications. An enterprise which uses this custom framework for load testing will be interested to understand how this framework addresses the response time for a given set of requests and the ever-increasing number of user requests and finally ensures the agreed value of the response time even with increased load. This section covers how this custom load testing framework can be leveraged by the enterprise to get best results with optimal infrastructure needs and lowered cost of ownership – as follows:

- Load testing: Load testing is one of the ways to govern the performance of the enterprise system. This will help understand the systems performance, when the system is impended with very high amount of load over duration of time. The results will showcase the performance of the systems and also highlight the weak link and bottleneck areas of the enterprise system.
- Stress testing: Enterprise system normally performs well with average load; once the load increases or reaches its max capacity, the performance of the system deteriorates. Every enterprise would like to understand the maximum capacity the system can address without any changes in the quality of response and time taken. This testing will determine the robustness of the system under max load and help in capacity planning.
- Endurance testing: Enterprise system might perform well during the above testing, but during continuous testing cycle with extended load, the performance of the system and memory and CPU utilization should be consistent and should not deteriorate and impact the performance of the system under test. Measuring the response time and throughput and monitoring the system characteristics periodically help detect memory leaks and performance issues. The goal is to ensure that the system behaviour is consistent while applying significant load over a period of time.

Testing web and enterprise application in the cloud using cloud computing infrastructural resources to simulate real-world user scenario.

14.4.1 Leveraging Different Types of Applications

We can consider leveraging cloud for testing various types of applications. It is critical to carry out thorough performance testing before the application goes live to avoid any issues due to improper capacity, data contentions, memory crunch and other resource availabilities.

First type of applications that we can consider is consumer or external user facing large ‘Web applications’. Web applications are usually targeted for large volume user base where the user spread is across different geographical regions. This ideally

becomes eligible for cloud-based performance testing since cloud infrastructure can be consumed from different geographies from the public cloud service providers. Similarly stress testing and endurance testing also can be carried out more efficiently and cost-effectively due to on-demand and pay-per-use cloud infrastructure features.

Similarly, this cloud-based performance testing can be leveraged for large user-based applications, business facing services and also for large enterprise applications which are targeted for internal users.

14.5 Cloud-Based Software Performance Testing

As the cloud computing evolves and enterprise adoption of cloud service becomes ever more pervasive, multiple cloud infrastructural resources can be utilized as a test infrastructure and can be connected to existing infrastructure. A cloud infrastructure resource creates substantial new opportunities in testing of applications when needed. Enterprise current model of separate testing infrastructure for each and every department can now be shared across common pool of resources.

Cloud-based performance testing enables easy, convenient, on-demand access to the shared pool of test infrastructural resources. Those test resources can be provisioned over networks for servers, automation software, storage of test scripts and reports, network bandwidth and security needs to test the application, thereby reducing management efforts and service provider interaction.

While there are several factors as mentioned above for leveraging cloud for performance testing, the following section depicts a simple illustration of cost comparison of public cloud approach against the virtual machine approach, as shown in Fig. 14.3.

To summarize the above results, cloud approach provides approximately 80–85 % cost reduction per year on infrastructure alone (considering the assumptions listed in Table 14.1). The above comparison is against the cost using VMs vs. public cloud infrastructure, while the cost saving will be much higher when we compare against on-premise dedicated test servers. Also the above calculation changes depending upon the actual duration of test cycle needed and the frequency of testing.

Apart from the infrastructure cost saving shown in Fig. 14.3, other primary contributions for cost saving by using public cloud provider VMs are software licensing cost, software maintenance cost and IT support cost. Depending upon the kind of software needed for testing and skill sets needed for maintenance and support, this varies and adds up to the cost saving as licensing, maintenance and support costs are minimal/nil compared to on-premise scenario. Table 14.1 captures some of the assumptions made and the cost of public cloud (using Azure) used for the above calculations.

In addition, leveraging cloud-based load simulation provides flexibility and capability for instantaneous higher scale, reduces setup time and enables near real-time testing. This helps to achieve accurate capacity planning, improved

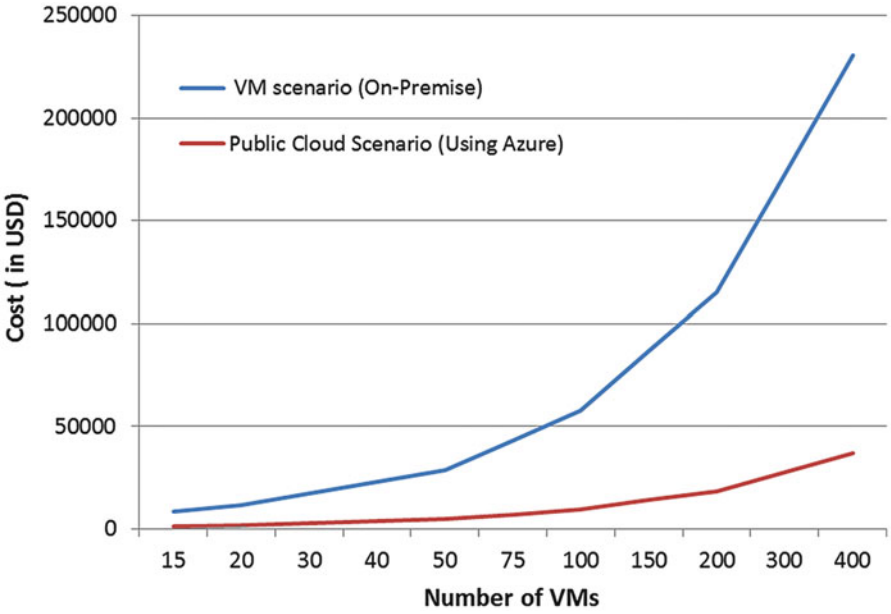


Fig. 14.3 Cost analysis of VMs vs. public cloud VM

Table 14.1 Key consideration for load simulation from cloud server providers

No.	Data points and assumptions
1	Azure compute rate = 0.48/h, added 10 % additional for storage, data transfer, etc. Total $0.53/h
2	Approximate monthly rent for VMs = $ 144 per VM (8 GB RAM, 4 Core CPU)
3	Assumed 1 week of test execution per month, 4 months in a year

user experience and cutting down beta test time due to closer near real-time test scenarios and workloads. There are several providers, e.g. BlazeMeter, SOASTA, LoadStorm and Keynote, that provide performance testing tools in a hosted model, as shown in Fig. 14.4.

One can benefit from the important characteristics of cloud such as scale, elasticity, self-provisioning and pay per use by adopting cloud infrastructure or cloud-based load testing tools. Following are the sample features, relevant in the context of performance testing of software solutions.

14.5.1 Scale and Elasticity

It is important to simulate high volume of load and to have flexibility for scaling up and scaling down as per the test cycles and test scenarios. To take an example,

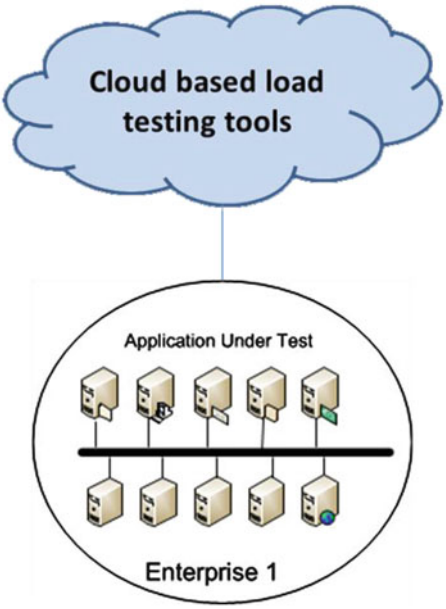


Fig. 14.4 Cloud-based load testing tools

performance testing may need a workload scenario where some of the heavy usage use cases will have to be run at particular intervals with a large number of test servers to simulate such a load. Leveraging such hosted cloud-based load test tools can be beneficial. It provides great flexibility to scale up and scale down resources as per the test scenario and is available on-demand or instantaneous which helps in optimizing the infrastructure cost.

14.5.2 Self-Provisioning

The cloud-based load testing tools provide management consoles where the test admins can provision the test infrastructure as per infrastructure plan. With user-friendly portals it abstracts the complexity of technical aspects of cloud usage and additional knowledge needed in provisioning. One can select the size of virtual machines (VMs), number of VMs needed and any other available configurations from such provisioning portal for setting up the test infrastructure in the cloud. An important aspect that needs consideration is the speed and time needed for provisioning the test infrastructure. Provisioning of such test infrastructures in hosted test tools environment is instantaneous and can take up several minutes depending upon the number of test servers needed. In the same way, the legacy systems can take up to a couple of weeks for procurement and internal processes.

14.5.3 Pay per Use

Provisioning test infrastructure as and when needed and paying only for the duration of the test execution is critical to keep the infrastructure cost optimal. Cloud-based load testing tools provide various charging and billing options depending upon the parameters such as size of VMs, number of VMs, geographic distribution of VMs, network bandwidth, security needs and storage used. In case of legacy test infrastructure setup, the physical test servers enabling such VMs will have to be retained throughout the test cycle and not just for the test execution. This difference can result in good amount of infrastructure cost savings.

Another approach followed by some of the enterprises is to have private cloud infrastructure for creating test environments for performance testing. Though this provides good cost advantage due to shared VMs and resources, scaling and elasticity can be the limitations of private cloud. Some of the large enterprises are providing API-based access to provision and de-provision resources such as VMs from the private cloud for their internal teams – this is still in early stages and will have to use only the internal portal for provisioning. Hence, cost of infrastructure for test servers could be a little higher compared to on-demand public cloud infrastructure.

While such cloud-based load testing tools could be a worthy option for small- or medium-size enterprises, they might not suit large-size enterprises where the test infrastructure needs are higher in size and are needed for testing several applications or services on a continuous basis. It can make sense to have own self-provisioning engine and to leverage public cloud infrastructure to provide scale and elasticity of the test infrastructure with optimized infrastructure cost.

In addition to reduced cost, one can leverage same set of load testing tools by having provisioning engine to set up the virtual machines with those testing tools installed in the VMs. This eliminates the need of migrating the existing test scripts or reskills the resources to accommodate the load testing tools provided by the cloud-based performance test providers.

Most enterprises look for majority of the below key features when using any cloud-based testing tools:

- Ease of integration of the existing infrastructure with cloud provider.
- Ease of provisioning and configuration any kind of test servers and test application. It is difficult to solve functional anomalies in the cloud.
- Data integrity, security and topology should be the same.
- End-to-end user interaction should be the same such that the experience does not differ whether the application is hosted in the enterprise or in the cloud.
- Applications' performance should be the same when hosted in the enterprise or in the cloud.
- Simulating user load across geographies.

- Hosting the application under test (AUT) as well as load testing application to test AUT in the cloud.

We can now look at the next level of details needed to build a custom load simulation framework.

14.6 Key Factors for Cloud-Based Testing Platform

Testing of application in the cloud has been there for a long time, and majority of the enterprises are using cloud for testing their application or hosting the application in the cloud. Most enterprises also have their own private clouds. Based on the industrial experience and the solutions built over the past few years, following is a summary of some of the major service providers in the cloud-based testing arena:

- *Keynote*: It is on-demand performance monitoring for enterprise and mobile. It is integrated with application life cycle management, testing and QA platform. It provides End-to-end quality of service and performance monitoring solutions [11].
- *BlazeMeter*: It has been built on top of open source testing tool JMeter. It is a self-service, load testing solution for high-volume load tests against websites, and it offers deployment of services to efficiently load test complex, rich applications [12].
- *SOASTA*: Available as on-demand service in the cloud through virtual resources, SOASTA has a unified integration platform from design, monitoring and reporting. It provides real-time business intelligence for performance management. It easily fits into iterative software development life cycle [13].
- *LoadStorm*: It allows tight integration with the software development life cycle (SDLC) and the applications by simulating heavy volumes of Web loads. It lets the developer create load plans and generate concurrent users to simulate the load [14].

There are multiple other cloud service providers, e.g. VMLogic, Skytap, HP LoadRunner, PushToTest, ITKO, Cloudtesting.com, Monitis, GFI, CloudSleuth and BrowserMob. The key considerations, which each of the load testing cloud-based service providers needs to address, are:

- On-demand test services
- Cross browser support
- Script-based test execution
- Integration with development cycle
- Real-time monitoring and reporting
- Support for functional testing
- Integration with other products or API
- Service cost

While such cloud-based load testing tools could be a worthy option for one segment of enterprise because of the quick turnaround and minimal cloud skill set needs, it does not suit other segments where the usage is high and very frequent and the testing cycle is continuous for every release. In this case, it can make sense to have self-provisioning engine and leverage public cloud infrastructure to provide scale and elasticity of the test infrastructure with optimized infrastructure cost and higher flexibility.

A reference framework and the approach for addressing those aspects are discussed in the following section using a custom load simulation framework.

14.7 Custom Load Simulation Framework

Load simulation plays a key role in end-to-end testing. It is an important component for stress testing, performance testing, benchmarking and product certification [15]. There are several implicit aspects to be taken care as part of such custom framework such as feasibility to generate high-volume user load, auto scale and provision for monitoring and reporting features.

Note that Fig. 14.5 depicts a sample test server and enterprise system/application/service setup. Real-time test environment depends on the selected cloud service providers.

Another aspect to be considered in such framework is to simulate user requests from different regions by leveraging cloud infrastructure from multiple data centres of cloud service providers, as shown in Fig. 14.5. Granularities of such locations are needed for specific performance tests as it can vary based on enterprise scenarios. Cloud-based performance simulation framework showcases real-time performance testing in a cost-effective way; it also meets the needs of load and stress testing of systems under test.

Framework can be leveraged for both types of deployment, i.e. target enterprise systems are deployed in hosted model or in the model where deployment is within enterprise data centres. Also, such frameworks can be leveraged in performance testing of target test applications which are consumer-centric Web-based application and services-based, intranet application for enterprise employee and business services. There will be specific approaches to be followed in handling these scenarios, but the results are for better quality of systems using minimal testing infrastructure and optimal costs.

Key benefits of the custom solution framework are:

- Savings in overall test cost and faster time to market: Using the cloud infrastructure for load generation results in significant savings:
  - Time savings – Leverage public cloud service providers' APIs for developing framework components such as provisioning engine, monitoring and storage. This results in development of framework faster. In addition, once the framework is in place, setting up the test infrastructure becomes much faster with

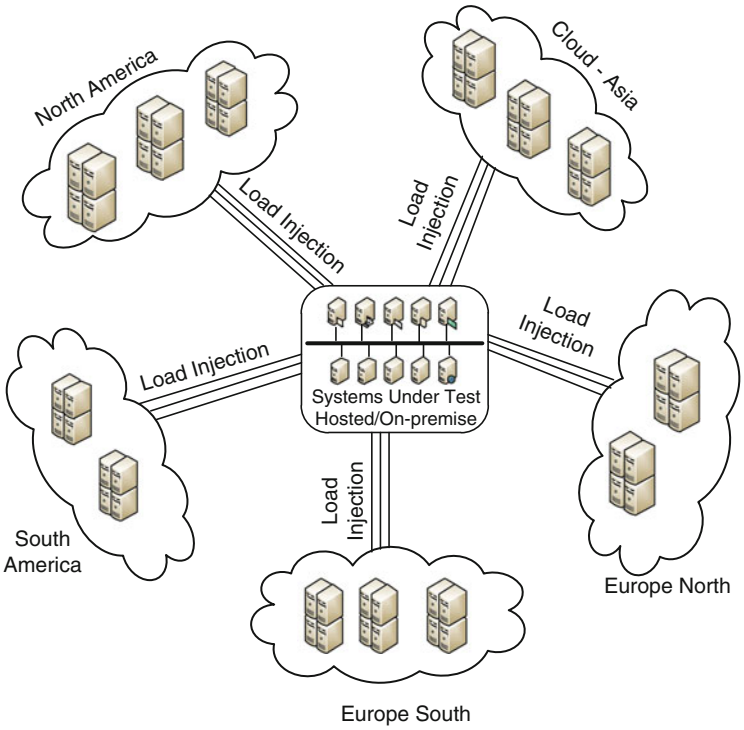


Fig. 14.5 Performance simulation on target test systems

such provisioning portals by reducing overall test life cycle. By bundling the require test software and tools as part of virtual machine images, one can cut down the overall test environment setup time drastically.

- Cost savings – Pay only for what is used – infrastructure cost during load testing period only. Provisioning portal of the framework can help in provisioning and de-provisioning of the test infrastructure on-demand basis, and the billing is only for the duration of the actual usage of those infrastructure components.
- Effort savings – Numerous person days are saved in procuring and setting up of the infrastructure with automated provisioning, configurable tools and utilities. In addition, the reporting functionalities such as consolidation of test result reports and monitoring of test infrastructure can eliminate numerous manual interventions.
- Figure 14.6 shows the component implementation of the custom cloud load simulator framework components. The framework has various engines from deployment of the test servers, queuing, execution of the test scripts, processing of events, orchestrating of the communication between the components and monitoring the systems under test. At the top framework is the adapter for integration with various load testing tools which support multiple cloud infrastructures, networks and information services.

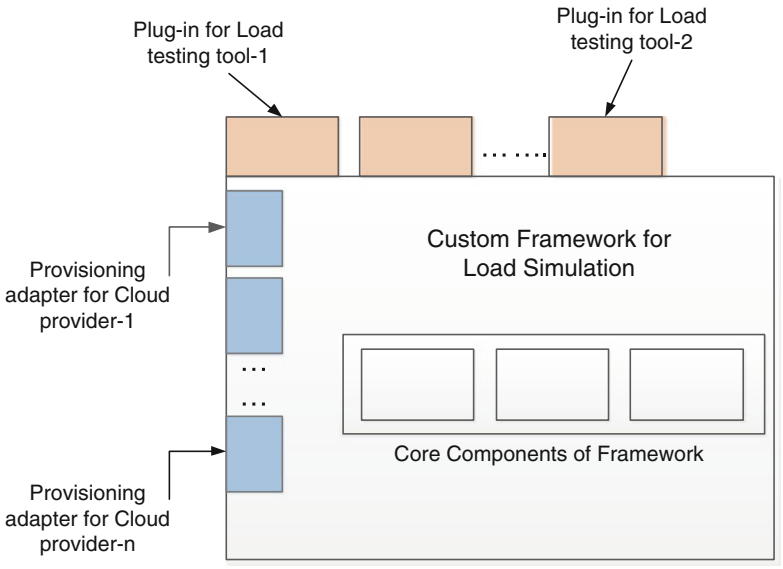


Fig. 14.6 Custom framework and load test tool plug-ins and cloud provider adapters

- Efficiency in testing: Leveraging the cloud infrastructure helps in simulating 'production identical' load generation across geographies. Leveraging right components identified as part of such custom framework and selecting right set of load testing tools can provide flexibility to control various test parameters such as test user profiles, network bandwidth and combination of test cases, leading to a 'foolproof' testing and ensuring high quality of the target enterprise systems.
- Leveraging existing investment and skill sets: While developing such a custom test framework, it is important to take the list of existing load testing tools, enterprise level road map for test infrastructure, etc. One can add specific plug-in for supporting those specific load testing tools such as Grinder, JMeter and other open source tools which is being used by the test team, as shown in Fig. 14.6, so that the effort needed for migrating those test scripts and cost involved in reskilling the testing team can be avoided. Similarly specific adapters can be developed to support specific public cloud infrastructure or platform, or leverage own private cloud, so that the overall test approach gets aligned with enterprise technology road map.

14.7.1 Cloud-Based Testing Usage Scenarios

A scalable enterprise application can be of multiple types, but typically the application with its underlying resources and components adapts to the dynamically changing situation and meets the needs of availability and reliability on the services offered.

With the growth of World Wide Web platform and diversified user base, the load and growth of the servers has become unpredictable. Enterprise application has to evolve and grow to meet the constantly changing needs of the market demand [16].

Enterprises usually overprovision to meet the demand. This may not be an effective way to provision the resources as the over-engineered resources might not be fully utilized. The enterprise also provisions the resources based on the usage pattern. This can help enterprise meet the demand at peak times; it is not a very cost-effective strategy as the load on the server depends on too many factors which are market driven compared to planned holiday session provisioning.

Neither of the above approaches is ideally suitable for an enterprise; enterprise has to determine the infrastructural resources needed to meet the demand based on multiple factors like user base, usage pattern, planned enhancements and spike in demand, which is why the cloud-based model is an excellent fit in determining the right infrastructural resource needed by the enterprise.

Reference architecture can be leveraged for different scenarios and for different types of AUT. Some of the examples for different scenarios are as follows:

- Scenario 1: Load testing, endurance testing and capacity planning with closer to real-time scenarios of load simulation from various geographies, location, network bandwidth, varying profiles, etc.
- Scenario 2: Benchmarking services/applications during each release and version upgrades
- Scenario 3: Product certification with required service level agreements and nonfunctional requirements as met under various test conditions

14.8 Solution: Platform for Near Real-Time Testing

Cloud-based custom framework showcases simulation of near real-time loads on the enterprise systems which is economical and quick to deploy and starts the test servers which satisfies the diverse needs of testing of small, medium and large systems. Platform should be leveraged irrespective of the hosting model of the application under test (AUT) that is systems hosted in on-premise or in the service provider environment, as shown in Fig. 14.7. Cloud-based performance simulation platform should adhere to certain broad functionalities; the following are the high-level features to be considered in an efficient custom load simulation framework [17]:

- Provision for setting up and configuring test infrastructure in multiple data centres across different regions (public, private, hybrid clouds) and testing the various performance metrics parameters such as response time, throughput, latency and failure rate
- Real-time testing of systems using the cloud infrastructure resources, configurable and elastic in nature to meet the demands of specific enterprise system
- Unified performance testing management platform which can provision, configure, execute, monitor, analyse and operate multiple cloud service provider

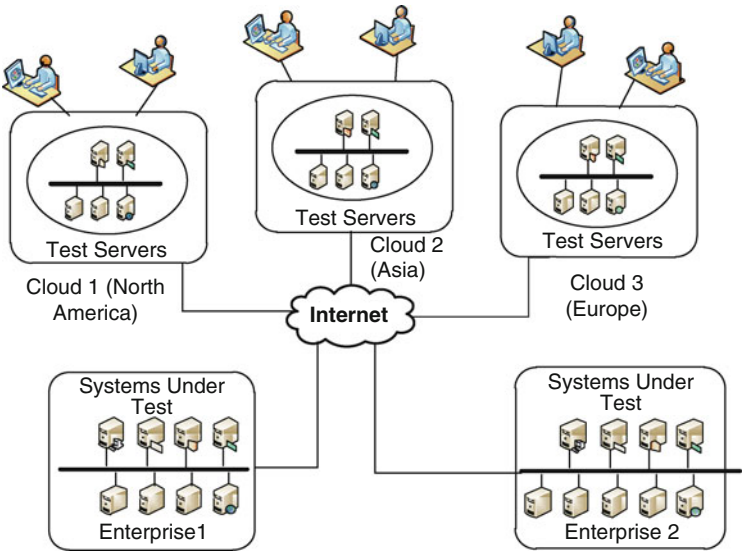


Fig. 14.7 Representative testing of enterprise system from multiple cloud providers

- Simulating the performance test cases for near real-time production scenarios across geographic locations, platforms, technologies, network bandwidth and Internet backbones
- Simulating multiple configurable usage pattern and use case combinations with high user volume
- Optimizing performance test infrastructure cost by leveraging on-demand multiple cloud infrastructures by providing a self-provisioning portal
- Built-in components for analysing, reporting and monitoring key performance parameters of the enterprise system

Note that Fig. 14.7 depicts an illustrative setup. Actual enterprise test infrastructure can be much more granular.

14.9 Core Components of the Custom Framework

The implementation of a custom cloud-based performance testing framework involves four major components: compute service, queuing service, storage system and a comprehensive platform to interconnect the above components and the enterprise system – thereby ensuring proper flow of message between them and the cloud service providers. To demonstrate this platform and its capabilities, cloud service provider's services are used; any private, public or hybrid cloud can be plugged with this platform. This depicted framework as shown in Fig. 14.8 uses compute service (Amazon EC2 services and Azure Compute), queuing service (Amazon SQS, Azure

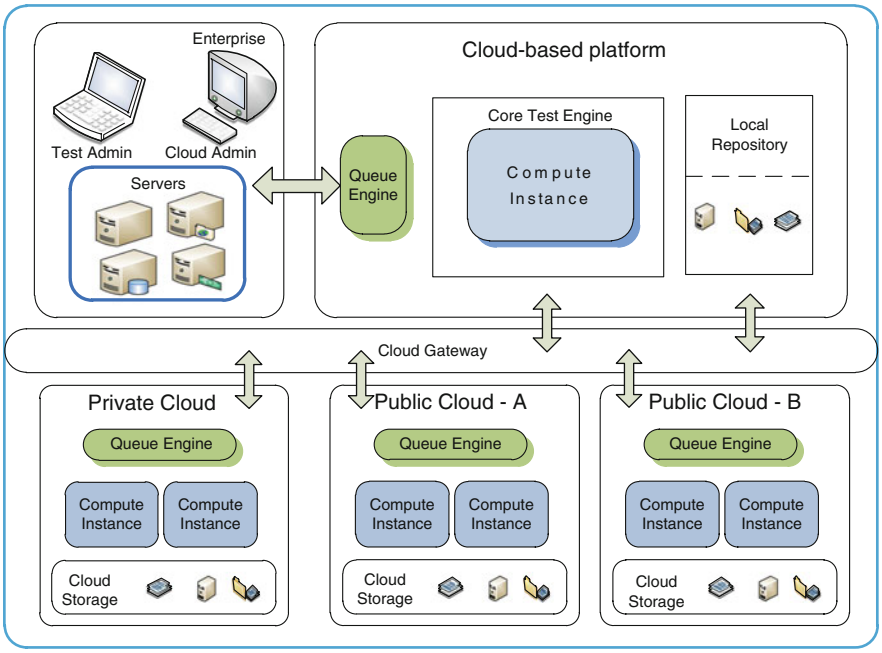


Fig. 14.8 Custom cloud-based platform and its components

Queues, etc.) and storage service (Amazon S3 Services and Azure Blobs) and provides the required interlinking components for successful implementation of performance testing on cloud-based platform.

Brief overview of each of the platform components is provided below. The interaction of these components with the cloud providers to implement cloud-based testing platform is also detailed in the next sections.

Compute Service: This service provides compute in the cloud environment, enabling users with the flexibility in processing computationally intensive applications. This service provides benefits in implementing scalable test environments, expanding and contracting the compute needs based on traffic patterns [18]. Compute provides access to computational resources CPUs. These low-level resources were not being utilized as individual commodity resources so far; they are now exposed to all through virtualized environment. Cloud service providers typically offer this capability of compute resources, i.e. raw access to resources as a service, typically to host services and applications [19].

Amazon Elastic Compute Cloud is a Web service; it provides compute capacity in the cloud. It allows scaling of application with ease by launching compute instances through Web services with an option to choose the operating system. Cloud platform is designed to scale up or down in terms of count of instances depending on the needs of the enterprise [20].

Microsoft's cloud computing platform Azure is used for building Web applications, hosted in Microsoft data centre. It can run various types of application in the data centre on Windows server or Linux compute images; the operating system can be tailored to meet the needs of the application. Running applications through Azure compute needs to be designed to implement Azure roles to utilize built-in load balancer to distribute load evenly [21].

Queuing Service: This service acts as a gateway to the custom cloud-based performance testing framework. It is reliable and scalable and provides easy retrieval of jobs as they pass test messages from one compute instance to another for completion of test cycle as a workflow. There is a limit on the message size, duration of jobs in the queues and type of message; it depends upon the cloud service providers. Input job and output result queues form the entry and exit of performance test message for the platform. Messages are queued and dequeued through simple API call. Additional queues can be created based on demand and load on the test environment. Framework has the option to dedicate queues for a specific application, department, type of testing, etc.; built-in macros as well as custom macros can be defined to assist in configuration of the queues for the enterprise testing needs.

Storage Service: This service provides storage mechanism for the test environment starting from test server image, test scripts and application configuration. Storage files are limited to few MBs to GBs; it depends on cloud service providers configuration limits and the storage bucket size. Framework has two separate storage systems, local storage and cloud storage area. Local storage area contains details of all the test suites and cloud storage area consists of test suites related to the jobs executed in that particular cloud infrastructure [22].

Platform: The platform orchestrates the flow of performance test events across the various cloud service provider components and the enterprise systems. The platform consists of core test engine to monitor the queue of the platform, workflow engine to govern the setup of virtual test server instances, deployment engine to interact with various cloud infrastructure provider and perform the provisioning and de-provisioning of compute instances, scheduler to perform predefine schedule test tasks and rules and policy engine to continuously monitor the test infrastructure for any variation in test instances characteristics and accordingly provision additional capacity to meet the enterprise demand. When the number of test job increases or decreases in the input queue, these test instances are added or terminated, respectively, taking full advantage of 'utility computing' model pay only for the resources used [23].

Core Test Engine: It orchestrates the request coming from various test users, underlying platform and monitoring system. The performance job requests are queued in the platform; each request has a weightage and priority for addressing it by the core engine. Along with that the workflow engine, deployment engine and monitoring engine send the request to the core engine for additional actions. Core engine depending on the type of request processes them by orchestrating through the various components in the core engine at the earliest with optimal compute, storage, network and security needs [24].

Cloud Gateway: It is a core component of this platform; it brings flexibility and control in terms of connecting various cloud service provider data centres with the enterprise data centre. It acts like a façade to the end users and the various departments of the enterprise from the complexity of the cloud deployment on the various clouds, platform and different network topology. It provides ease of use of services based on the range of service offered by the cloud providers.

As enterprises, internal users and consumers access the deployed cloud services through aggregated wide area networking (WAN) connections to and fro from private or public cloud. WAN has brought Ethernet to the forefront of transport mechanism connecting cloud services. Gateway provides the following:

- Connectivity between the enterprise and the cloud services
- Connects the enterprise network with the edge networks of the cloud providers
- On-demand network bandwidth allocation
- Connectivity between various public, private, hybrid and enterprise cloud service providers spread across different geographic locations
- Connects cloud to end users

14.10 Anatomy of Testing an Application in the Platform

End-to-end flow of a performance testing on an enterprise system through custom cloud-based testing framework and how it orchestrates the flow of events through the various components are shown in Fig. 14.9. Custom cloud-based performance testing framework includes managing the cloud service provider infrastructure resources, workflow engine and the test jobs. The target enterprise test system and the components along with the test scripts can be hosted in the enterprise data centres, or they can be provisioned in the cloud infrastructure.

Prior to creating a test server environment, the two queues, input and output queues, are created. This is performed by the test macro within the management platform. The platform also provides macro which assists in creating system configuration tasks.

The test script is submitted to the input job queue as test job; it contains the required input files to test the target test system. The core engine executes the script as a job on the test server node(s) dedicated to test the target systems, and it also performs any post-processing specific operations. The engine sends the test results to the output queue; it also sends the error reports generated while testing. The platform performs the following chain of events:

- Test job is submitted to the cloud management platform with the test script and system details.
- The required test server(s) image and configuration scripts are uploaded to the appropriate location in the cloud repository.
- After the test job is retrieved from the queue by the core engine, it checks the validity of the job.

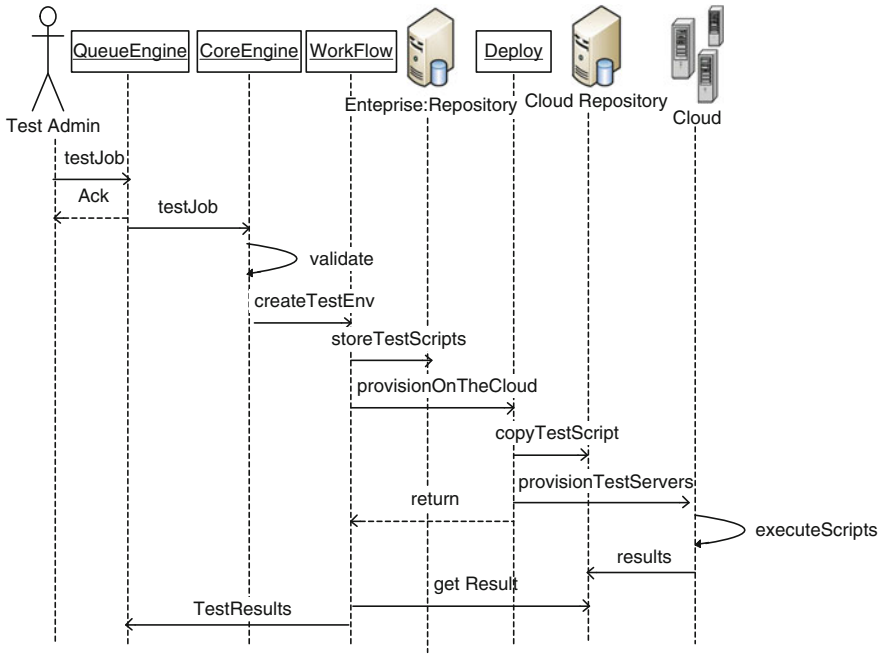


Fig. 14.9 Flow of events

- The workflow engine creates the necessary test server(s) environment and passes the configuration scripts to the deployment engine.
- Deployment engine moves the required test script from repository to a local directory on the test server instance.
- It executes the enterprise system configuration details on the test instances for setting up end-to-end test environment.
- The test server runs the test script on the enterprise systems running inside the enterprise or in the same cloud environment.
- The test results are placed in the output directory on the test server instance.
- Platform moves the result to the cloud repository.
- Workflow engine periodically checks for the results and on completion of the job successfully; test result from the cloud repository is sent to the output queue along with status.

14.11 Overall Architecture of Custom Load Simulation Framework

Proposed custom framework as shown in Fig. 14.10 consists of queue manager. Its main function is to check the input queue for any test jobs and process the message. The management dashboard in the platform can be configured to specify

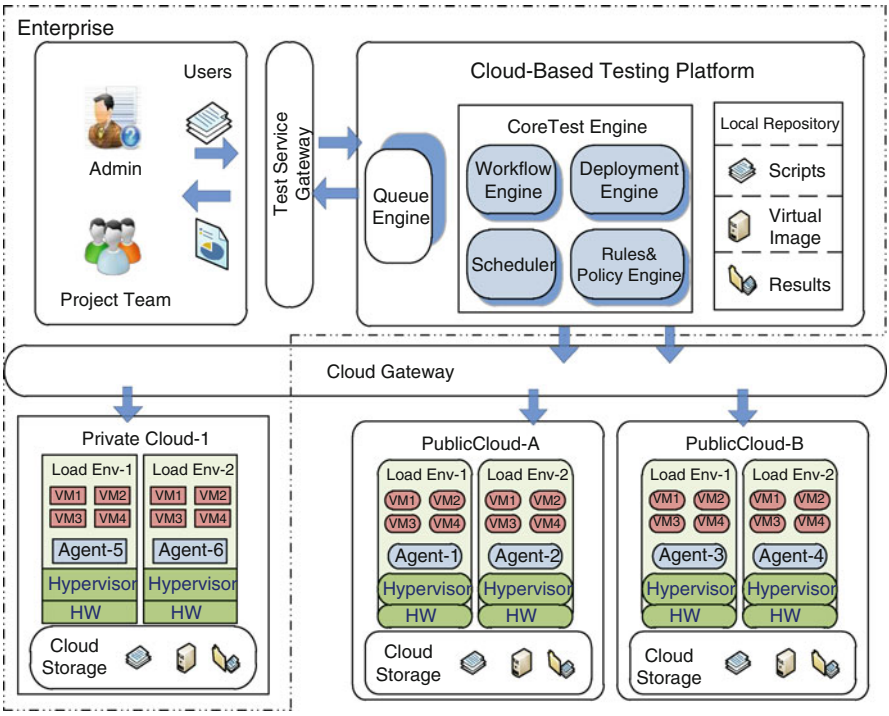


Fig. 14.10 Component architecture of custom framework

the number of queue manager needed to monitor and process the requests. The number of queue managers to monitor is based on the number of jobs in the queue or on specific enterprise system or on testing type.

After checking the validity of the test job, the core test engine copies the specific enterprise system test scripts, test server image and configuration parameters to the local repository. Next, call is made to workflow engine to set up the test environment in the cloud service provider infrastructure as specified by the test job request via the deployment engine. Priority of the message is also checked before calling the deployment engine. High-priority test jobs get preference in execution when compared to lower-priority performance test jobs. Deployment engine copies the request test scripts, test server instances images and other configuration details to the specified cloud storage repository.

Each test job in most cases is dependent on one or more applications that may be executed in a definitive order; the execution order varies and depends on the dependent application. Workflow engine provides an ability to define workflows to make decisions, to perform iterative operations, for forking multiple operations and waiting for an instruction or a set of instructions to complete thereby significantly improving the efficiency of executing test jobs in the cloud. The key features of Workflow Manager are:

- Ability to define loop, decisions, forks, joins and sub-flows
- Customized process

- Global and local flow parameters
- Orchestrating between the various input/output tasks
- Pre- and post-processing scripts
- Ability to run tasks with different credentials
- Monitoring flows and take corrective action as needed

Deployment engine provisions the test image in the test environment based on operating system, test application, chosen cloud environment, geographic region and number of test server instance needed in each geography. Once the test instances are created in the private/public/hybrid cloud, it will also provide additional system configuration details with the location of the enterprise systems, testing type, number of test runs needed, test result reports, error reports, etc. to the test instances provisioned.

The rules and policy engine defines a set of actions for each application to be executed when a condition is met or at a predefined trigger points. The data captured by the rules engine is evaluated based on a set of predefined business and operational rules. If any of the rules meet, then it triggers the associated action to increasing or decreasing infrastructural resources as needed for the test job. The heartbeat of test infrastructure is checked at regular intervals and rules engine is the key to guaranteeing the health of the system based on the heartbeat message [25].

A predefined test job increases the requirement of infrastructural resources, and time-bound events can be scheduled for execution at a latter point of time on the platform. Scheduler identifies all the requests and priorities them as and when it is requested, by checking the real-time infrastructural capacity and provisioning the request through the workflow engine.

Test instance checks for the appropriate test scripts in the cloud storage repository and runs network connectivity tests to check the enterprise systems availability for performance tests. Multiple test instances might be needed to process a single performance test job. Each test instance communicates with the platform via the agent which is bundled along with the test instance. Agent will periodically update the health and progress of the test instance to the platform, and it also performs the task of moving the test results and error reports to the specified cloud storage repository once the test scripts are run by the test instance.

Cloud storage is the repository for all the test scripts, application, agents and reports. Common storage pool is used and it is shared with the entire compute instances. Data among the various storage service providers are shared via framework, and direct sharing of storage across data centre is avoided. Cloud platform is deployed as a set of instances in the cloud to minimize the network latency. For those instances which are configured across cloud providers, the communications between the various providers are secure via the cloud gateway.

Cloud platform is deployed in the enterprise data centre and manages the various cloud platforms of the enterprise as shown in dotted lines; it also manages via secure network communication additional infrastructural instances of public clouds A and B. Test infrastructure instances and scripts are stored in the local repository, and depending on the deployment topology, these images are copied onto the cloud providers' storage repository.

Hypervisor is positioned on the bare metal or on the host operating system for hardware virtualization. The virtual machines (VM) run on the virtualization layer also known as compute instance; VMs are managed by the hypervisor. It can start, stop, suspend or resume a compute instance based on the input from the platform. Also hypervisor has a monitoring agent which provides alarms, logs and performance statistics to the platform [26].

All users of this platform access the services through the common dashboard. Based on the type of roles and privileges the user belongs to in the enterprise, corresponding operations are permitted on the cloud. Details on how a user can access multiple cloud service provider infrastructural resources are explained later at platform integration layer section.

The platform updates the dashboard about the health of the performance test based on the heartbeat sent from the agents. Test admin can suspend and resume any performance test from the dashboard. Platform can also decide to pause any performance test jobs due to resource constraints or due to high-priority job which needs immediate processing in the specified cloud service provider environment.

Features exposed by the cloud-based performance testing platform are very similar to any IT administrator provisioning the test environment in the enterprise data centre. The simulation platform handles the extensibility, scalability and reliability of the test environment by optimally provisioning and configuring the environment in the cloud. IT administrators have complete control on the test environment in the cloud service provider infrastructure from the platform dashboards.

14.12 Deploying Test Environment in the Cloud Using the Framework

The testing process and additional details on the flow of event between the enterprise systems, custom cloud-based performance testing platform and cloud service provider infrastructure are shown in Fig. 14.11.

This flow and sequence of events is rerun for all the test server instances that have to be deployed in the test environment by the platform. The test script can be created from the testing tools available in the market for performance testing such as Grinder, JMeter or LoadRunner. The platform can run multiple test scripts at the same time to test the enterprise systems.

The platform provides a scheduler to preschedule any test runs at predefined time interval as batch jobs. The scheduler also monitors the input queue and launches test manager to process the job from the input queue.

Once the scheduler has launched the test manager instances, the platform initiates allocation of server resources for testing. The platform sets up the required test server environment in the cloud. It then executes the test server installation and configuration scripts as defined by the test job on the test environment. The test environment at the end of setup will have a set of configured test server instance,

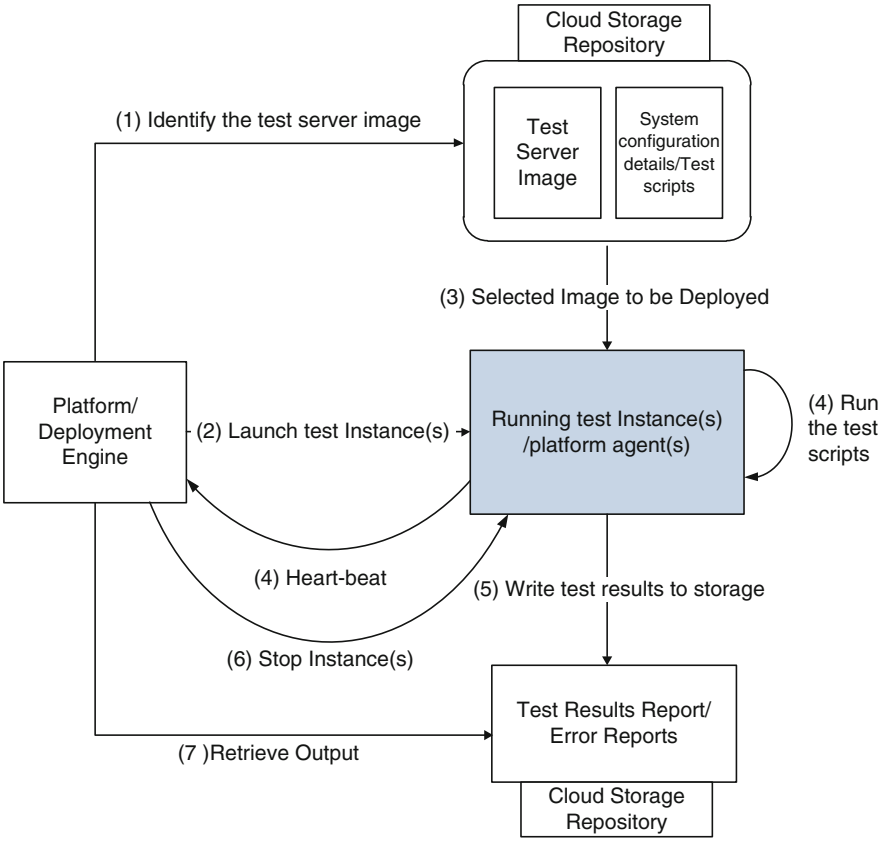


Fig. 14.11 Deploying the test server in the cloud

with the base operating system deployed in the test geographic region with the required configuration information. For every test cycle, test server template can be manually configured or by executing a prebuilt macro.

The server template can be configured to deploy additional installation components needed to successfully run the performance test on the enterprise systems. Test-specific configuration scripts as well as deployable and additional files can be downloaded from the enterprise network using secure file share repository and installed on the test instance at the end of the instance's start-up cycle.

If the enterprise application under test (AUT) is in the cloud, enterprise system instances are created and deployed similar to the test instance. If the AUT is in the remote environment or in the enterprise data centre, then the test server instance is configured with the system details. Each test server instance is responsible for running a set of test suites on the enterprise system and uploading the test server results or error reports to the cloud storage repository.

The test server instances are deployed in the service provider infrastructure environment as a test service. The platform handles the elasticity by ensuring that the required numbers of test server instances are deployed and provisioned adequately in the enterprise system network topology. On-demand the platform can add additional test server or remove them during the course of execution of test scripts. Enterprise IT administrator, test admin, test team and the enterprise management team interact with the platform through unified dashboards for all actions and status.

14.13 Platform Integration Layer with Multiple Cloud Service Providers

Every enterprise has multiple systems, services and applications. In a typical enterprise organization, there will be multiple departments and each department will have multiple programs having simple or diversified infrastructure requirement needs. Each program has to meet a certain set of key infrastructural criteria such as compute, network, storage and security. This list increases day by day and hence the complexity of identifying, assigning appropriate values and monitoring has increased. The main concern from the IT administrator is for better utilization of the entire infrastructure.

There are multiple cloud service providers enterprise can choose to deploy applications and services on public, private or hybrid clouds; all of them have their own trade-offs. While public clouds are typically accessed via the Internet, the private clouds are located on premises.

Public clouds are run and managed by third parties, and different customer applications are likely to be inter-deployed together on the servers, storage and network infrastructures.

Private clouds are built for the usage of one enterprise, providing the complete control over data, quality of service and security. The enterprise owns the data centre infrastructure and has control on the systems, services and application deployment.

Hybrid clouds conglomerate both public and private cloud models. The ability to mix and match a private cloud with the certain resources of a third-party cloud can be used to provide better service levels in the face of rapid workload fluctuations. Cloud service providers have multiple service offerings in terms of the following:

- Operating system
  - Windows, Unix, Linux, Solaris
  - 34 bit/64 bit
- CPU/memory
  - Number of core CPU
  - Number of cluster cores
  - Number of GB of memory

- Storage
  - Number of GB of storage
- Network
  - Number of Gigabit of bandwidth
- Security/VPN
  - Security services
  - Firewall
  - Intrusion detection
- Pricing
  - Pay per usage
  - Monthly
  - Subscription/annual

Having flexibility of using multiple public providers along with own private cloud for the load simulation provides more efficiency to have more granular load simulation, cost-effectiveness and test efficiencies. Importantly, the enterprises adopting such multiple cloud provider services for different internal departments need to put an efficient and robust security, accounting and governance framework in place.

14.14 Applying Cloud Simulation Platform in Reality

While the custom performance simulation platform provides various components to speed up the actual testing time and improving the productivity and quality, there are various aspects which need to be analysed while using such framework for a specific performance testing scenario. Some of the important aspects to be analysed and for consideration are captured, as shown in Table 14.2. Detailed analysis on these helps in planning the overall performance testing.

The load simulation solution should support majority of the nonfunctional requirements like availability, scalability, maintainability and reusability, which are critical for testing a high-volume user-based scenario – as discussed below.

Availability: In the performance simulation requirement, very high availability is not mandated, but the requirements were to maintain constant load and stagger the step loads throughout the test period. The above solution approach proposes usage of public cloud such as Amazon EC2 and Microsoft Azure instances located in different locations. Required programs and tools were installed and configured to generate the extreme load. Multiple instances and fallback clusters and automated engine for adding new instances are the approaches provided in the solution to meet needed availability to handle availability requirements.

Table 14.2 Key consideration for load simulation from cloud server providers

No.	Considerations	Description
1	Target test application	Gather details on target test application type (Web application, Web service, etc.), type of access (HTTP, HTTPS, FTP, SMTP, etc.)
2	Cloud infrastructure for load injection	Gather details on cloud infrastructure to be leveraged in the solution. It is important to capture details on spear of user base (region wise) and proportions. This aids in deciding the distribution of cloud infrastructure for the provisioning engine
3	Load testing scenarios	High-level details on multiple load test scenarios, covering test duration, volume of users, proportions of multiple use cases, step load needed, volume of data to be loaded, number of test cycles, etc.
4	Mock test data	Gather requirements of test data creation and data file creation as per the test cases and functionalities of the target test application
5	Capacity planning inputs	Gather requirements of server-side monitoring tools and analyse on any specific custom plug-ins or any additional monitoring tools needed for server-side monitoring to aid in proper capacity planning inputs
6	Custom reports	Gather details on various reports needed during performance testing and identify needs of any additional custom reports apart from the built-in reports

Scalability: Solution uses one of the cloud instances as controller, which will be associated with multiple instances called load agents; the load agents will generate the required load. To scale up the number of users, we can increase the controller and load agent combination proportionately.

Maintainability: Custom test framework solution proposes usage of standard design patterns, protocols, coding guidelines and conventions to provide easier code maintainability. The solution proposes using REST APIs, JNI, Azure SDK and standard IDEs like Visual Studio and Eclipse which are used by enterprise to build a custom cloud-based performance simulation platform.

14.14.1 Business Value and Benefits

The following list presents the business values and benefits:

- *Quality of testing*: Leverage the cloud infrastructure for 'near real-time production scenario' by generating loads across geographies, combining user profiles and varying network bandwidth leads to a 'fool-proof' and quality testing.
- *Minimize reskilling and reuse investment*: Use of standard and tested tools without significant investments in the solution reduces the additional investment and minimizes the additional skill set development. Also user-friendly provisioning and monitoring features can minimize reskilling needed for using cloud infrastructure for IT admins and managers.

- *Business continuity*: Improves business continuity through identification of weak links and risks in the system at an early stage.
- *Accelerate testing*: Ready-made solution to simulate new application and load testing shrinks beta testing duration. Saves weeks/months of time in procuring and setting up infrastructure with on-demand provisioning and de-provisioning of cloud infrastructure.
- *Reduce cost*: Pay for use for actual loud infrastructure cost during load testing with on-demand provisioning and de-provisioning. This avoids any wastage of resources of test infrastructure and brings the cost to very optimal.

14.15 Conclusion and Future Work

Cloud testing is a hot research topic in the software engineering community. Although there are many players in the field with respect to testing as a service (TaaS) in the cloud, there are many issues and challenges in cloud-based testing. To provide on-demand testing services and assuring the QoS needs in a scalable cloud environment, innovative testing techniques and solutions are needed by every enterprise. Resources in the cloud are abundant and advances in technology are increasing the cloud capabilities; the opportunities are limitless on scaling of application horizontally and vertically. The usage of enterprise application keeps growing and planning of resources always keeps changing. Testing these resources and meeting the demands with ideal infrastructures is a daunting task.

Testing these applications is a challenge. At the same time, no two enterprise applications are alike but similarities on the resources needed, usage pattern and other common traits of the application can be utilized in testing the applications and getting them ready to face the real test while keeping in mind the cost constraints of the enterprise.

The suggested framework takes care of the overall concepts with regard to enterprise application testing, but every application in an enterprise is different and there could be multiple possibilities in testing each tier of an application. Best results and scalable solution can be achieved by constantly testing the application with the right set of load relevant to the architecture of the application and geographical user base.

To make it more beneficial to the enterprises, we can plan to add another core component in the 'custom framework' called 'test script migrator' that can facilitate migration of test scripts of different load testing tools to industry standard ones. This can be considered as a plug-in to support some of the recognized load test tools. By providing this, enterprise can reduce the effort needed for developing plug-ins for different load test tools; instead, they can use test script migrator. More research is needed to evaluate this in terms of cost savings, feasibility and quality of migration, and the option has to be chosen based on those results.

This chapter provides an overview of various components needed for cloud-based testing, mentions current major players and presents comprehensive overview of a proposed framework by discussing the concepts, issues and challenges. The major contribution includes insightful discussion about anatomy of cloud-based testing frameworks in terms of requirements, benefits and comparison with conventional testing.

References

1. Connie, U.S., Lloyd, G.W.: Software performance engineering. http://www.springerlink.com/content/g311888355nh7120 (2004)
2. Lazic, L., Mastorakis, N.: Cost effective software test metrics. http://www.jameslewiscoleman.info/jlc_stuff/project_research/CostEffectiveSoftwareTestMetrics_2008.pdf, June 2008
3. Connie U.S., Lloyd G.W.: An SPE approach. http://www.perfeng.com/papers/pdcp.pdf (September 15, 2000)
4. Aberdeen Research on performance of business critical applications. Application Performance Management: Getting IT on the C-Level's Agenda. http://www.aberdeen.com/Aberdeen-Library/5807/RA-application-performance-management.aspx, November 2008
5. Darrell M.W.: Saving Money Through Cloud Computing. http://www.brookings-tsinghua.cn/~/media/Files/rc/papers/2010/0407_cloud_computing_west/0407_cloud_computing_west.pdf (7 April 2010)
6. Qiyang Chen, Rubin Xin: Montclair State University, Montclair, NJ, USA. Optimizing Enterprise IT Infrastructure Through Virtual Server Consolidation. Proceedings of the 2005 Informing Science and IT Education Joint Conference. URL: http://informingscience.org/proceedings/InSITE2005/P07f19Chen.pdf (2005)
7. Filippos, I.V., Elaine, J.W., AT&T Labs: Performance testing of software systems. In: Proceedings of the 1st International Workshop on Software and Performance, pp. 80–87. http://dl.acm.org/citation.cfm?id=287337 (1998)
8. Scott Tilley Florida Institute of Technology (STITC 2011): WORKSHOP SESSION: Cloud computing and infrastructure. In: 3rd International Workshop Software Testing in the Cloud, pp. 377–379. URL:http://dl.acm.org/citation.cfm?id=2093959 (2011)
9. Carlstrom, J., Rom, R.: Application-aware admission control and scheduling in web servers. In: 21st IEEE Infocom, New York, Timeframe, pp. 824–831, 23–27 June 2002
10. Sidharth Subhash Ghag, Divya Sharma, Trupti Sarang, Infosys Limited: Software validation of application deployed on windows Azure. URL: http://www.infosys.com/cloud/resource-center/Documents/software-validation-applications.pdf (May 2011)
11. Keynote solution. http://www.keynote.com/ (2008)
12. Blazemeter The jMeter Cloud. http://blazemeter.com/ (2011)
13. Cloud Testing for Performance Intelligence. http://www.soasta.com/ (2010)
14. Cloud based load testing. http://loadstorm.com/ (2010)
15. Menasce, D.A.: Load testing of Web sites. Internet Comput. IEEE **6**(4), 70–74, New York, Timeframe (July/August 2002)
16. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: enabling scalable virtual organizations. Int. J. High Perform. Comput. Appl. **15**(3), 200–222 (Fall 2001)
17. Shyam Kumar Doddavula, Raghuvan Subramanian, Brijesh Deb, Infosys Limited.: Cloud computing, what beyond operational efficiency? URL: http://www.infosys.com/cloud/resource-center/Documents/beyond-operational-efficiency.pdf (May 2011)
18. Open Virtualization Format Specification: DMTF standard version 1.0.0, Doc. no.DSP0243. http://www.dmtf.org/standards/published_documents/DSP0243_1.0.0.pdf (February 2009). Accessed 16 April 2010.
19. Mensce, D., Almeida, V.: Capacity Planning for Web Performance Models and Methods. Prentice-Hall, Englewood Cliffs (1998)
20. Amazon: Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2 (2009)
21. Microsoft, Azure: http://www.windowsazure.com/en-us/ (2010)
22. Roop, R.: Deliver cloud network control to the user. URL: http://www.ibm.com/developerworks/cloud/library/cl-cloudvirtualnetwork/ (21 October 2010)
23. Bose, S., Pasala, A., Dheepak, R.A., Murthy, S., Malaiyandisamy, G., Infosys Limited.: SLA management in cloud computing. In: Cloud Computing Principles and Paradigms, Chapter 16, pp. 413–436. Wiley, Hoboken (2011)
24. Bose, S., Tiwari, N., Pasala, A., Padmanabhuni, S.: SLA aware "on-boarding" of applications on the cloud. Infosys Lab Brief. **7**(7), 27–32 (2009)

25. Van Halle, B.: Business Rules Applied – Building Better Systems Using Business Rules Approach. Wiley, Hoboken (2002)
26. Bose, S., Sudarrajan, S.: Optimizing migration of virtual machines across datacenters. In: Proceeding of the 38th International Conference on Parallel Processing Workshops (ICPP), pp. 306–313, Vienna, 22–25 September 2009

Chapter 15
Open-Source Cloudware Support for the Portability of Applications Using Cloud Infrastructure Services

Dana Petcu and Massimiliano Rak

Abstract Initially intended to support elastic and long-running Web applications, cloud computing paradigm is currently associated with a large variety of services, from infrastructure to software. In this context, the portability of the applications, based on cloud services, even only of infrastructure type, has become a challenge. To overcome this situation, several recent research and development efforts are targeting new methods, tools, and standards to avoid the problems of vendor lock-in. One outcome of such efforts resulted in a special category of open-source platform (as a) services that can be modified by application developers according to their specific needs. Looking deeper into this category, the present chapter provides an analysis of the emerging open-source platform (as a) service and points toward one particular new solution, targeting application developers and sustaining the concept of elastic Web applications as well as of reusable open-source components.

Keywords Platform as a service • Cloudware • Open source • Deployable service • Portability • Elasticity

15.1 Introduction

The trust in cloud computing paradigm is shadowed by the current problem of vendor lock-in: due to the fact that there are no standards widely accepted at this moment. Almost each cloud vendor has its own APIs, and the code that is prepared

D. Petcu (✉)
Institute e-Austria Timisoara, West University of Timisoara, Timisoara, Romania
e-mail: petcu@info.uvt.ro

M. Rak
Department of Industrial and Information Engineering,
Second University of Naples, Naples, Italy
e-mail: massimiliano.rak@unina2.it

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*,
Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4_15,

to use these services cannot be ported from one cloud to another. Therefore, the last 4 years were marked by several attempts to overcome this problem.

The first visible efforts were focused on proposing a thin layer of uniform APIs (e.g., Deltacloud, Libcloud, and jclouds) for certain groups of cloud providers, or on proposing standard interfaces (like OCCI for compute resources and CDMI for storage resources). While these solutions have been partially adapted to Infrastructure-as-a-Service level (IaaS), they are not completely solving the problem, since the current standard of uniform API compliant services is addressing only a certain category of developers, those with basic knowledge in system administration.

While at the IaaS level, the thin layer of uniform and open APIs has solved partially the problem of cloud application migration, at the PaaS level, the application portability is possible only between a small number of PaaSs (e.g., in the case of open-source clones of the proprietary ones). Moreover, the problem escalates with the increase of the number of PaaSs that have been born in the last 2 years. This increase is motivated by the fact that a large category of application developers still prefer to develop applications in a friendly IDE that simplify the task of application deployment, locally or remotely. Platform-as-a-Service products (PaaS) intend to answer to their needs hiding the complexity of application deployments on the cloud environments.

Due to the difficulty in designing the deployment, debugging, and monitoring procedures for multiple clouds (even for one in some cases), PaaS providers address only partially the current needs of the developers, and there is no general solution that is commonly adopted: each PaaS provider offers a special flavor in its design and not all the features that are expected (e.g., usually no debug facilities, no Security-as-a-Service offer, and so on).

Analyzing the PaaSs offers, we note that at least two categories exist: the *platform services* (also named hosting services or integrated solutions) and the *platform software* (also named software services or deployment-based solution). According to [18], a cloud platform service is a hardware and software combination, provided by organizations running Internet-scale services. A cloud platform software is a deployable middleware (referred in what follows as Cloudware) provided by software vendors and open-source projects; a platform hosting organization can use this deployable software to offer services in public or private clouds.

If supported by the first category, the applications are clearly tight to the particular infrastructure service provisioning. This is the case of the well-known Google's App Engine, Microsoft's Azure, Rackspace's Cloud Sites, Amazon's Elastic Beanstalk, Salesforce's Force.com, Joyent's Smart Platform, and many others.[1] Some of these offers were reshaped recently to allow the deployment using IaaS services from different providers (usually up to three, a minimal level for the portability of the applications).

[1]To name a few others with worldwide visibility: AT&T Synaptic, BitNami, Bungee Connect, CloudBees, dotCloud, Engine Yard AppCloud, Heroku, Jelastic, LongJump, mCloud, OrangeScape, Project Caroline, Stackato, StratosLive, Nodester, Nodejitsu, phpcloud.com.

Open source takes an important role to attract the application developers by allowing them to customize their interfaces, but the deployment and hosting remains on the provider sites (clear examples in this context are the open-source offers of Joyent or Nodester). The approaches undertaken in what concerns the application support are quite diverse:

- Deploy application code to specific instances of virtual machines, as in the case of Azure or Elastic Beanstalk.
- Develop application code according to the platform rules and let the platform deal with the deployment issues, as in the case of App Engine or Heroku.
- Create metadata for an application interpreted by the PaaS at the run-time, as in the case of Force.com or OrangeScape.

The third approach is the most appealing by allowing a high degree of abstraction and productivity, while the first approach ensures more control and programmability for the application developer. The second approach is the most common one.

The second category of PaaS (of software services) makes a bigger step in the direction of solving the portability problem, allowing the deployment of the support middleware in various data centers. Moreover, in this specific category, the open-source Cloudware has the highest potential to support the development of innovative and portable applications, if it is adopted on a large scale (as was the case of PVM and MPI in parallel computing or Globus in Grid computing). According to [16], the combination of cloud computing and open source increases the service delivery over the traditional forms of software consumption. Same author states that when cloud computing utilizes open-source software, another level of maturity and diversity of the open-source stack will be reached, and by this the cloud can expand.

Motivated by the above considerations, we present in this chapter an overview of the most known open-source platform software and discuss in some detail a new offer, namely, mOSAIC's open-source Cloudware that has taken the path of the third approach described above.

15.2 Overview of the Open-Source Cloudware

Looking at the current offers, we can state that, at this moment, there is no PaaS offering completely provided as open source. Therefore, the degree of openness of the PaaS is subject of various studies. Following are the criteria to measure the openness that can be used in terms of the degree of freedom for the application developers who are using the PaaS [3]:

- Freedom to choose the infrastructure service, the operating system, and the virtualization technologies
- Freedom to choose the language and the stack components (like Web framework, application server, database)
- Freedom to customize the platform or to add additional components

- Freedom to move data, scripts, and applications to other clouds
- Freedom to choose self-support by supporting third-party open components
- Freedom to reuse tools and components of the platform when moving to another platform

These six criteria are used in this chapter to define the characteristics of a particular open PaaS. Following these, we have considered that the cloud services that show a low degree in fulfilling the first criteria are not open-source Cloudware, despite their declared openness. We excluded by this rule, for example, dotCloud which is available only on Amazon services. Very few examples of comparison of cloud services exist. We offer here a brief summary of these which have inspired our work.

The authors of [4] compare three IaaS offers, XCP (Xen Cloud Platform), Eucalyptus, and OpenNebula, in terms of architecture, networking, virtual machine placement, and inter-host communication. The lesson learned from their comparison is that each offer is adequate for a certain type of interaction with the cloud services. Similarly, the authors of [19] are analyzing XCP, Nimbus, OpenNebula and Eucalyptus, TPlatform, ECP (Enomaly's Elastic Computing Platform), and Apache's VCL, looking only to the underlying infrastructure, addressed community, configuration flexibility, and closeness to the hardware level.

The list of the open-source cloud computing products analyzed in [20, 21] is going beyond OpenNebula and Eucalyptus to UEC (Ubuntu Enterprise Cloud), Abiquo, RHCF (Red Hat Cloud Foundations), OpenStack, Nimbus, and mOSAIC. As in the previous cases, the authors of these papers are particularly interested in comparing IaaS solutions, but the criteria that are proposed can be applied also to the PaaS comparisons. The almost 100 comparison criteria were grouped in six categories: virtualization, storage, management, network, security, and support.

Open-source-based services, belonging to a particular deployment category apart from IaaS, PaaS, and SaaS, are considered in [7]. The authors mention that the current open-source platforms are mainly allowing the monitoring, managing, and controlling of the virtual instances (IaaS features mainly). Eucalyptus, OpenNebula, and Nimbus are considered to fall in this category.

Several online documents of the research communities involved in cloud computing communities are attempting to provide comparisons of cloud computing offers. For the PaaS category, a good example is the one promoted by TheCloudTutorial [1] where the PaaSs are classified according to the languages and frameworks that are supported, development status, vendor, number of developers, number of applications, number of outages, maturity in terms of years, billing model, performance metrics (like tenant density, environment utilization, self-service efficiency), geographical location of physical infrastructure, and user communities. The list of open-source PaaS is including, beyond the offers discussed in this chapter, others, like GigaSpaces' Cloudify, Apigee's Usergrid, and WSO2's StratosLive (not discussed here, as they are not in the category of deployable PaaS).

We underline again that cloud computing paradigm adoption is hindered by the difficulties in realizing portability and interoperability between the current cloud services, and, in order to overcome this barrier, a key role is played by the availability

and adoption of open-source solutions. Born only recently, the open-source PaaS offer is already quite diverse. Moreover, it has already the potential for being agnostic to the infrastructure services. As proof, we gathered in Table 15.1 the data of the most relevant offers up to date. Several classification criteria (as the ones already mentioned above) can be considered in order to offer a clear survey of the state of the art. The criteria considered in Table 15.1 are mainly underlying the most common characteristics or features. Examples of such features are support for several languages, connectors to several open-source solutions for storage, usage of particular virtual machine images, preference for command-line or Web interfaces, and beta version or production status. Moreover, it is possible to consider if the PaaS is dedicated only to Web applications or not, if it is cloud agnostic or not, if it is allowing desktop simulations or not, if it is easy to integrate existing frameworks or not. As mentioned in [22], almost all open-source platforms are based on Linux, but the operating system can also be considered as a criterion.

Table 15.1 provides a high-level overview of the kind of functionalities which PaaS systems offer. In order to have a more detailed view of the features offered by different PaaS solution, we identified three main use cases or stages: development, deployment, and execution. Each of such use cases has different needs and involves different features. Therefore, Table 15.2 focuses on typical features of interest for each of such use case, like load balancing, auto-scaling, application monitoring, multitenancy for execution stage, or support for standard libraries, thread access, and network connectivity in the developing stage. Table 15.2 details these only criteria for three open-source platform software, Cloud Foundry, mOSAIC, and OpenShift.

An analysis of Tables 15.1 and 15.2 provides evidence for the complexity of a clear and complete comparison of different PaaS solutions. In order to help readers to build up such analysis, in Table 15.3, we have grouped the criteria that we consider relevant for the Cloudware comparisons in different categories. Summarizing the different proposals from the literature, we considered the criteria exposed in Table 15.3 as suited for the particular case of PaaS. These criteria include the ones mentioned in the previous studies, as well as other new ones that are revealing some special features of open-source Cloudwares. To give an example, we presented in Table 15.4 how these criteria are fulfilled by a particular solution, namely, mOSAIC, which is detailed in the next section.

15.3 Overview of mOSAIC's Cloudware

The analysis reported in the previous section has been initiated in order to identify the positioning of a recent open-source PaaS, mentioned earlier and named mOSAIC, designed specially to overcome the portability problem. mOSAIC is a project partially funded by the European Commission in the frame of the FP7-ICT program (details at www.mosaic-cloud.eu). It targets the design and development of an API and PaaS for developing applications for multiple clouds. We start this section with a presentation of the main reasons to build a new PaaS and an overview of

Table 15.1 Comparison of seven open-source platform software

Product	AppScale	Cloud Foundry	ConPaaS	mOSAIC	OpenShift	WaveMaker
Owner	Univ. of California	VMware	Contrail Consortia	mOSAIC Consortia	Red Hat	VMware
Site	appscale.cs.ucsb.edu	www.cloudfoundry.com	www.conpaas.eu	www.mosaic-cloud.eu	openshift.com	www.wavemaker.com
Repository	appscale.googlecode.com/svn/	github.com/cloudfoundry	www.conpaas.eu/download/	bitbucket.org/mosaic	github.com/openshift	dev.wavemaker.com/wiki/bin/
State	1.5/Jul 2011	0.x, Beta	0.1/Sep 2011	0.5/Aug 12, Alfa	Production	6.4.4/Dec 2011
Language	Python, Java, Go	Java, Ruby, Node.js, Groovy	PHP	Java, Node.js, Erlang, Python	Java, Python, Perl, PHP, Ruby	Java
Data support	HBase, Redis, Hypertable, MySQL Cluster, Cassandra, Voldemort, MongoDB, MemcacheDB	MongoDB, SQLFire, PostgreSQL, Redis	Scalaris, MySQL, XtreemFS	Riak, MemcacheDB, Redis, MySQL, CouchDB Amazon S3, HDFS	MySQL, MongoDB, Amazon RDS	Amazon S3, Rackspace
OS	Ubuntu, CentOS	VMware image	XtreemOS image	Linux	Red Hat Virtualization	VMware image
Messaging	Channel	RabbitMQ	Own design	RabbitMQ	Own design	Own design
Clouds tested	Amazon EC2, Eucalyptus	VMware	Own test bed	Amazon EC2, Eucalyptus, Flexiscale, Arctur, OpenNebula, Rackspace	RightScale, Rackspace, SmartCloud, Amazon EC2	Amazon EC2, Rackspace, OpSource, Eucalyptus
Interface	CLI, Web	CLI	Web	CLI, Web, REST	CLI, REST	Studio

Table 15.2 Detailed comparison of three open-source platform software

Product	Cloud Foundry	mOSAIC	OpenShift
Development support	1	2	3
Dedicated to Web apps or general	Web apps	General	Web apps
Desktop cloud simulator	Yes	Yes	No
API access	No	Yes	No
Support standard programming libs	Yes	Yes	Yes
Impact on Web app architecture	No	Yes	No
Friendly to migrate Web applications	Moderate	Moderate	High
Complexity of porting Web app	Medium	Low	Low
Standard support tools	Spring tools	No	JBoss, Zend
Thread access	Yes	No	Yes
MySQL	Yes	Yes	Yes
Allows to choose stack components	Yes	Yes	No
Allow to pull data out	Yes	Yes	Yes
Debugging mode	Yes	Yes	Yes
Deployment support	1	2	3
Lock-in when building own cloud	Yes (VMware)	No	Yes (RHE)
Web server (e.g., Tomcat)	Yes	Yes	Yes
Built-in balancer	No	Yes	Yes
Auto-scaling app server	No	Yes	Yes
Auto-scaling database	No	Yes	No
Performance analytics	Yes	No	Yes
Support multiple cloud providers	Yes	Yes	Yes
Agreements SLA	No	Yes	No
Deploy with a special tool	Yes	No	No
Support private cloud	Yes	Yes	No
Allows to add third-party components	Yes	Yes	Yes
Execution support	1	2	3
Command line (CLI)	Yes	Yes	Yes
Web console	No	Yes	Yes
Access to logs via Web	No	Yes	Yes
Web-based monitoring	No	Yes	Yes
Multitenant	Yes	Yes	Yes

the full solution. These general views are followed by a deeper discussion about the measures by which the new solution is fulfilling the current requirements for an open-source Cloudware.

15.3.1 Targets of mOSAIC

mOSAIC is designed to support the activities of the developers of software or service products relaying on top of cloud resources (infrastructure or software), namely, cloud applications. These applications are offered to the end-users and consume cloud resources.

Table 15.3 General criteria for comparing PaaS offer

Stage	Level	Criteria	Variants or sub-criteria
Development	General	Type of support	Service vs. software
		Source availability	Open vs. closed
		Cloud dependence	Agnostic vs. dependent; support single provider vs. multiple provider; type of cloud
		Interfaces	API access; command-line interface; Web console; drag and drop; others
	Particular	Programming style	Web apps vs. generic code; code generation; thread access; support migration of existing application; app portability; elasticity
		Standard support	REST; SOAP; JSON; CDMI; OCCI; others
		Framework	Eclipse plug-ins; IDE tools; others
		Language support	Java; JavaScript; PHP; Ruby; Python; Perl; Node.js; others
		Data management support	MySQL; NewSQL; key-value store; document store; Blobstore; in-memory data; database aaS; DFS; big data support; others
Deployment	General	Type of support	Service vs. software
		Source availability	Open vs. closed
		Cloud dependence	Agnostic vs. dependent; single provider vs. multiple providers; type of clouds
		Interfaces	API access; command-line interface; Web console; drag and drop; others
	Particular	Environment	Tomcat; .NET; others
		Tools	Deploy within or without a special framework; debugging mode; others
		Supported OS	Linux; Windows; others
		Integration with other services	Messaging services; data services; CRM services; interoperability between clouds
		Cloud enrollment	Support SLA negotiation; support brokering; support policy-based selection of services
Execution	General	Type of support	Service vs. software
		Source availability	Open vs. closed
		Cloud dependence	Agnostic vs. dependent; single provider vs. multiple providers; type of clouds
		Interfaces	API access; command-line interface; Web console; drag and drop; others
	Particular	Scaling	Dynamic scaling; auto-scaling VMs; auto-scaling storage; auto-scaling the connectivity
		Monitoring	App monitoring; Web-based and logs; alerting and resolution; policy-based auto-remediation
		Performance	Performance analytics; load balancing built-in or external; support for high availability
		Service level	Multitenancy; security; outbounded vs. bounded network connection
Model	Particular	ROI	Model of payment; licenses; openness basic
		Design	Corporation design vs. public funds
		Price	Min offer; max offer; free promotion offer
		Status	Production vs. development

Table 15.4 Criteria fulfilled by mOSAIC

Stage	Level	Criteria	Characteristic (only the one supported)
All stages	General	Type of support	Software
		Source availability	Open
		Cloud dependence	Agnostic
			Support multiple providers
			All clouds
		Interfaces	API access – open
			Command-line interface
			Web console
Development	Particular	Programming style	Generic code
			Code generation for interop
			No thread access
			Limited support for migration of existing apps
			Support for application portability
			Programmable support for elasticity
		Standard support	REST
			OCCI
		Framework	Eclipse plug-ins
		Language support	Java
			JavaScript
			Python
			Node.js
			Erlang
		Data management support	MySQL
			Key-value store
			In-memory data
			Distributed file systems
Deployment	Particular	Tools	Deploy within a special framework
			Debugging mode
			Others: mOS
		Supported OS	Linux category
		Integration with other services	Messaging services
			Data services
		Cloud enrollment	Support for interoperability between clouds
		Support cloud enrollment	Support SLA negotiation
			Support brokering
			Support policy-based selection of services
Execution	Particular	Monitoring	Application monitoring
			Web-based
			Access to logs via Web
		Performance	Performance analytics
			Load balancing external
		Service level	Multitenancy
Model	Particular	Return on investment	Free of payment
			Apache 2.0
			Open
		Design price	Public funds
		Status	Development

Note that the cloud computing is mainly addressing a new programming paradigm, of programmable resources [9], and the main target of cloud computing is therefore the new applications that are aware of the cloud resources. In this context, the mOSAIC API and PaaS are trying to support the development of such new applications. Moreover, the take-up of the cloud computing paradigm is hindered by the need to change the code of legacy applications to profit from cloud characteristics like elasticity. While a solution to this problem is beyond the scope of mOSAIC, it can help the code owner to find the cloud that is proper to be used, through the mOSAIC's Cloud Agency and Semantic Engine. Two questions of the developers addressed by mOSAIC are:

Q1. How can an application be ported from one cloud to another (relates to portability between clouds)?
Q2. Which cloud is appropriate for the application (relates to selection of cloud providers and technologies)?

Consequently, two primary types of customers can be interested in the proposed solution:

- Developers of new portable cloud applications (interested to have an answer to Q1)
- Developers of existing applications to be ported on the cloud (interested to have an answer to Q2)

In this context, three scenarios of using multiple clouds are supported by mOSAIC.

15.3.1.1 Scenario 1: Changing the Cloud

Application developers or their clients may wish to change providers to make an optimal choice regarding utilization, expenses, or earnings. Other reasons to port application or data from one provider to another can be provider out of business, better options in market than current ones, technology changes, contract termination, legal issues, etc. Unfortunately, the software stack that allows the usage of the cloud services is usually defined by the resource providers, and standards and protocols allowing code portability are not respected or not yet available.

The applications designed using mOSAIC's APIs can be ported from one IaaS provider to another. The software stack to support the cloud application is using open-source technologies.

15.3.1.2 Scenario 2: Service Brokerage

Finding the best fitted cloud services for a certain application (either in development phase or in deployment phase) in the conditions of the fast-growing market offer is a challenging problem for its developer. Unfortunately, the market of cloud services is characterized by a large variety and the lack of standards.

mOSAIC's Semantic Engine, based on a cloud ontology, supports the user in matching requests with offers (functionalities and resources). Moreover, the mOSAIC's Cloud Agency plays a role of a broker of cloud offers.

15.3.1.3 Scenario 3: Development of Cloud Applications

The development of cloud applications requires an expertise in preparing the execution environment (in the case of IaaS) or the programming style (in the case of PaaS focusing on paradigms serving mainly Web applications). Unfortunately, the costs of writing cloud applications are high. Moreover, intensive testing phases can be prohibited from a financial point of view.

The mOSAIC's light and deployable PaaS can be used on a desktop as well as on a local cluster or in a private cloud. After intensive testing, the full stack of software can be deployed without changes in a public cloud or can be used to build a hybrid cloud.

15.3.2 General Overview and Availability

Figure 15.1 provides an architectural view of architecture of the full software stack of mOSAIC. This stack includes:

- Application support (API implementations in Java, Python, Node.js and Erlang, application tools, semantic engine, and service discoverer)
- Software platform (deployable core components, deployable application components, and application service components)
- Infrastructure support (cloud agency, vendor agents, and broker)
- Cloud adapters (support for hosting and deployable services)
- Proof-of-the-concept applications (in five fields)

The open-source part of mOSAIC solution (referred here as Cloudware) includes part of the application support (the API implementations and application tools), part of the software platform support (platform core components and application support components), and part of the cloud adapters.

The mOSAIC's concepts were exposed earlier: the programming style promoted by mOSAIC was presented in [13, 15], the initial software platform architecture in [14], proof-of-the-concept applications in [5], and particular components in [12]. The codes of the Cloudware are available on bitbucket.org open-source codes' repository (http://bitbucket.org/mosaic/ from which it is possible to get the links to the documentation pages). While the previous papers about mOSAIC have been engaged in debates about the technical aspects of the implementation as well as the interpretation of the preliminary test results, in what follows we emphasize how mOSAIC responds to several requirements imposed by current practices or limits in cloud computing.

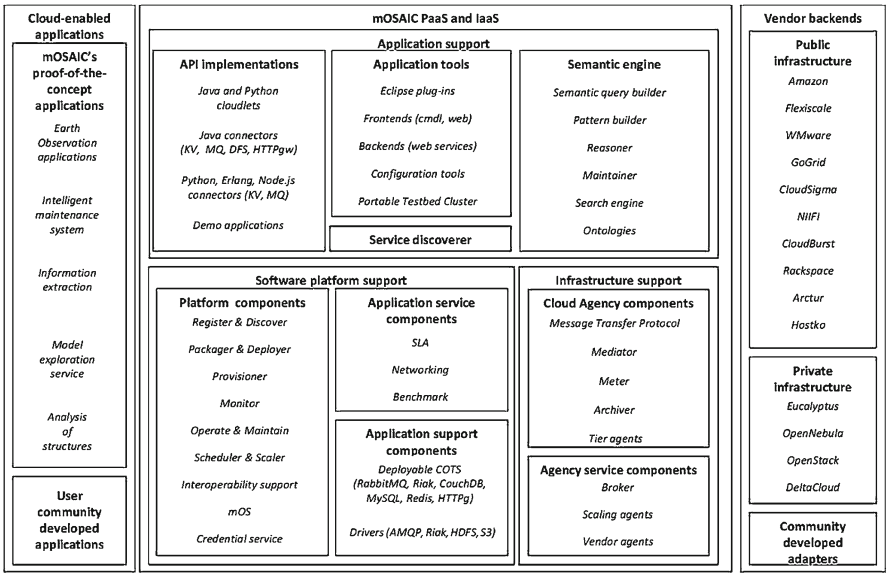


Fig. 15.1 Components of the mOSAIC's architecture

15.3.3 Cloudware Design Requirements and mOSAIC's Responses

15.3.3.1 Support for Programmability of e-Infrastructures

Beyond the lowering of the level of access to e-infrastructure, the cloud computing takes a step forward toward the implementation of the concept of programmable e-infrastructures. While in the case of infrastructure (as a) services, the setting of the resources (required on demand) imposes a certain interaction with the human user, in the case of platform (as a) services, the e-infrastructure acquirement and settings are automated and hidden from the application developer or user. The benefit of using a PaaS consists in the possibility to focus the innovative energies toward problem-solving instead of system fighting. The drawback is the lack of the control of the developer or the user that can be later on translated in an unpredicted performance, due to the platform overhead, or in a low quality of application developed on the top of the platform, especially in case of multitenancy or low security settings. The balance between the developer or user needs of control and the provider needs of automatization and elasticity should be carefully maintained and customized for the needs of the type of applications.

The offer of the available libraries for programming the e-infrastructures (analyzed, e.g., in [8]) is quite diverse and tends to be vendor dependent. Meta-APIs were proposed to overcome the diversity (to name a few: Libcloud, jclouds, Deltacloud, CloudSigma) by introducing a thin abstraction level allowing the access to the APIs of two or more vendor offers. Three characteristics of these step-forward solutions

are evident: (a) they maintain a close relationship with the programming style of the connected APIs; (b) they reduce the number of operations, methods, and objects to the common denominator of the linked solutions; and (c) the overhead that is introduced by the thin layer is negligible.

mOSAIC is proposing to make a step forward: to eliminate the need of keeping the programming style of the APIs offered by the cloud resource vendors. This is possible by introducing other levels of abstractions.

The above-mentioned meta-API level is maintained by what is called a driver, a piece of software that is written in a certain language and in a certain programming style. Drivers are needed for different types of resources, like message queues, key-value store, and distributed file systems. A simple parameter can transform the driver in a translator in the native API of a certain cloud resource (either hosted or deployable).

Another level of abstraction is proposed, namely, the *connector*, aiming at describing the functionality of a certain type of cloud resource in a particular language. Then a cloudlet, written in the same language as the connector, is a certain application component that calls the operations described in the connector and describes the application reaction to the results. For example, if the driver can be REST or OCCI compliant and written, for example, in Python and acting on bases of synchronous communications with the resource, the connector can be written, for example, in Java and can allow asynchronous communications with or between the cloud resources, while the cloudlet written also in Java describes the reaction to the events associated with the connector, if an event-driven architecture is requested for the application. In between the driver and connector, another level of abstraction, an interoperability API, is acting as a proxy generator.

The feasibility of this solution with several levels of abstractions is demonstrated by the current Java and Python implementations of mOSAIC's set of APIs and its preliminary efficiency tests (showing the low overhead and the benefits of decoupling the application programming style from the resource API style, e.g., in [15]). Moreover, a cloud ontology [11], a semantic engine, and a service discovery mechanism are helping the easy development of the applications.

A clear benefit of the platform services or software vs. infrastructure services or software is that it removes the need for preparing the running environment for the applications. The current programming style of most of the PaaSs requires the description of the specific needs in the form of parameters of certain methods or operations inside the program. In [17], another approach is considered: service manifests are accompanying the application. Following this latest described idea, in mOSAIC, such a complementary specification is decomposed in two: the application descriptor and the call for resource proposals.

The application needs in terms of resource types (abstractions of the cloud resources) and the communication needs are presented in the form of an application descriptor. Note that this descriptor is referring only to cloud storage and cloud connectivity, as well as number of instances of cloudlets and so-called containers of cloudlets (platform controllers of cloudlets at execution time). Then, the brokerage of the cloud services can start with this descriptor. If a connection to a cloud provider of hosting IaaS is already established, a trigger can be launched, even from an IDE

like Eclipse, to deploy the application components and the mOSAIC software platform (needed in this case to run and monitor the application) on the available resources. Otherwise, a call of proposals is generated and the Cloud Agency is entitled to come back with the links to the available resources (even on local premises in the case of deployable IaaS, and a certain guarantee from a cloud hosting provider related to a service-level agreement in the case of hosting IaaS).

Another tendency in the PaaS design is the reuse of other vendor or general-purpose services due to several reasons: (a) reuse of the vendor tools and (b) easy migration of the existing applications toward clouds. While most PaaSs are designed now to serve Web applications, a good example is the default deployment of a Tomcat servlet container on the virtual machines hosting the applications developed with PaaS. We see Tomcat as a particular case of a cloud resource that is deployable (denoted with COTS in Fig. 15.1). While Tomcat is not a cloud-native technology, there are several open-source and deployable software that are cloud native and which allow the elasticity in terms of resources – to name a few: RabbitMQ for message queuing, Riak for key-value store, Hadoop for distributed file systems, and Jetty for HTTP server and clients. In terms of mOSAIC, such COTS are cloud resources. Simple applications, like a Twitter watcher or producer-consumer communications, can be designed using only these COTS as cloud resources on a minimal configuration of virtual machines (as required, e.g., by Riak, at least two machines) – such simple examples are provided in the mOSAIC code repository.

Note that the use of COTS in common PaaS is restricted to a small number due to the need of their adaptations to the communications in the clouds, with the exception of the commercial offer of dotCloud, in which case the use of COTS is the main business idea.

Of particular interest for mOSAIC is the reuse of messaging queuing systems that are cloud native, as key components of the solution. The main mechanisms of interactions between application components and the elasticity at the application level are depending on the message queuing system. RabbitMQ was successfully used until now in the context of mOSAIC tests, but the replacement with another queuing system is a matter at the level of the application descriptor and availability of the software in the component repositories from where the installations at the application deployment are triggered.

15.3.3.2 Support for Elastic Applications

The cloud computing paradigm was used with success until now mainly for Web applications. Their main characteristics that are satisfied through the usage of the cloud computing are at least the following: (a) require a certain guarantee in what concern the application's response time and availability, independent from the location and number of users; (b) have long life cycle; and (c) use Web standards in what concerns the input and output data.

Intended to support mainly the Web applications, the most common offer in the PaaS category is restricting the application programming style to the specificities of

Web applications (note that simple mechanisms to port existing Web applications are also included in current PaaS, not necessarily allowing to benefit from cloud elasticity without reprogramming). Therefore, there is some interest from the scientific community to adopt current PaaS for their requirements that are different from the (a) to (c) items mentioned above. Most of the scientific applications are running for a limited period of time, requiring a large amount of resources, the response time is long, but estimable, and the usually huge data are in proprietary formats. IaaS services are therefore preferred by the scientific communities, and elasticity of the cloud is rarely a feature that is requested. However, the interest of the scientific community in IaaS is higher at least due to (a) the apparently unlimited computational power, (b) the apparent control of the running environment compared with the case of grid computing, and (c) the possibility to share large datasets of scientific results.

Aware of both community requirements, mOSAIC's solution is trying to address, beyond the classical Web application developer community, another niche segment of the developers of cloud-enabled applications: the ones who are interested to deploy applications (not necessarily Web applications) requiring dynamic adaptation of the resource capacities.

There are two ways of implementing elasticity in current PaaS: either programmatic, the changes in terms of resources are initiated by the applications following a certain performance measurement, to which the platform should conform, or automatic (namely, auto-scaling), the changes in terms of resources are initiated by the platform itself to ensure a certain level of service quality. Up to date, the dominant way in the case of the PaaS is the first one (see also the examples from Table 15.2). Auto-scaling is currently possible for most of the Web applications by scaling up or scaling down the number of servers that receive the stateless requests. To go beyond this state of the art and to widen the categories of applications that are supported, a new algorithm for auto-scaling was proposed in [2] for a continuously flow of batch applications (or components of a large application, like a media convertor serving unpredictable number of requests).

The elasticity of the applications is supported by mOSAIC in both ways: automated at the level of containers (number of cloudlets) and programmed at the level of application (number of containers). Note that a message received by an application component from another application component is expected to trigger a reaction treated by its cloudlet (therefore, an event-driven architecture with asynchronous communications was adopted). Since a container is managing several instances of cloudlets and watching the message queues, it increases or decreases the number of cloudlets according to the number of untreated messages in the queue and decides which cloudlet will treat the message. It emits an alert when a certain number of cloudlets are overpassed so that, programmatically, the number of containers should be increased.

Another aspect that is important in supporting elastic applications built from components and expected to run for long time (like in the case of Web applications) is the possibility to start, stop, and replace components of the application without the need of rebooting all other components of the applications. This was one of the

main requirements in the design of mOSAIC's Cloudware. The Web interface and the command-line interface provided by mOSAIC's Cloudware are allowing the control of the application components.

15.3.3.3 Support of Portability Between Clouds

There are several usage scenarios of services from multiple clouds that are nowadays pushing the adoption of interoperability and portability solutions. The NIST report [7], for example, mentions the cases of (1) the simultaneous use of services from different clouds or (2) the serial use when (2a) migration between clouds is requested, (2b) interfaces between clouds are used for bursting purposes, or (2c) a certain service is used after a selection. mOSAIC intends to offer a solution mainly for the cases (2a) through its deployable Cloudware (referred in Fig. 15.1 as application support, software platform, and cloud adaptors) and (2c) through its brokering mechanisms (referred to in Fig. 15.1 as infrastructure support and semantic engine). We focus in what follows on the case of (2a) that requires a high degree of portability of applications.

The portability of applications between clouds to support the case (2a) is expected to be achieved in different ways at different deployment levels. At IaaS level, it is expected that virtual machines and data are able to migrate to another cloud provider with a simple change of configuration. At PaaS level, as in mOSAIC case, a minimal application rewriting is expected.

Nowadays, the key topics of portability at IaaS and PaaS levels are the APIs, the support from programming languages, the data formats, the coupling degree of the services, as well as the abstract layers for queuing and messaging. More precise, the current approaches for the portability of applications applied to the development phase can be classified in different categories [15]:

- Use of open APIs, for example, jclouds, Libcloud, and Simple Cloud
- Use of open protocols: OCCI and Deltacloud
- Use of standards: OVF and CDMI
- Use of abstraction layers: mediators [17] or reference architectures [10]
- Use of semantic repositories, for example, UCI

mOSAIC has considered three of the above paths: (1) an open API that is vendor agnostic, (2) an abstraction layer in the form of a reference architecture for an open-source PaaS (described earlier), and (3) a semantic repository (based on a cloud ontology [11]) supported by a semantic engine.

The above-mentioned approaches for the development stage are ensuring that the functionality of the application is expressed in a vendor-agnostic manner. If this is the case, the IaaS or PaaS offer should include special services for adding on the fly, reconfigure or remove resources (expected to be manually at IaaS level and automated at PaaS level).

The design requirements imposed on a PaaS for portability reasons are quite complex and involve different levels, from market to monitoring. Following the list

of requirements identified in [15], we point here toward the ones that are supported by mOSAIC's Cloudware:

- Market level: service-level agreements
- Application level: scale-in and scale-out, data portability
- Programming: same tools for cloud-based or local applications
- Deployment: deploy in multiple clouds with single management tool, automated provisioning
- Monitoring: performance monitoring, sets of benchmarks

Adopting the mOSAIC approach to achieve portability, the application is expected to have a price implication: the mOSAIC developer should adopt the programming style of mOSAIC-compliant applications – for example, component based, following an event-driven architecture, use of message passing as communication style between the application components, and use of the drivers that are potentially restricting the access to the full functionality of the resource. Moreover, an overhead (low one according to [15], but not negligible) is introduced by the platform usage compared to the case of direct use of resources of certain cloud providers.

The adoption of mOSAIC of an event-driven architecture contrary to the most common case of request/response interaction model should be explained. Event-driven architecture is appropriate for Web applications offering near real-time data, like portals, dashboards (real-time reporting), social networks (automatically up to date), or business event applications. Moreover, the performance of event-based programs running under heavy load is observed to be stable in comparison with that of threaded programs; furthermore, event-based programs use a single thread, and, thus, the programmer does not need to handle data races. Another PaaS that adopted the event-driven architecture is the commercial Azure.

An important component of the mOSAIC solution, and a proof by itself of the degree of portability, is the Portable Testbed Cluster that allows the off-line development of the applications on the developer desktop for debugging and test purposes (personal cloud) and easy deployment of the ready-for-production application on on-premises resources (private cloud) or the e-infrastructure resources provided by commercial clouds (public, corporate, community, or hybrid clouds). Looking forward to the case of using multiple clouds, an important and complex issue to be solved is the scheduling in heterogeneous environments of different clouds. Preliminary studies for the particular cases of batch of components or applications are reported in [6].

15.4 Conclusions

The emergency of open-source platform as a service has a high potential to shake the way in which the cloud is used today. By adopting them, a step forward can be made in the direction of programming the e-infrastructure's behavior. Although the

portability of the application is hindered by the current practices in designing such platforms, solution to the issue can be the adoption of open-source deployable platform (as a) services. mOSAIC's open-source API and platform is built following this idea and currently offers a proof of the concept that portability between e-infrastructure providers is possible and a larger variety of applications can be served by the PaaS, beyond the classical Web applications.

Paying attention to mOSAIC solution, a considerable part of this chapter was dedicated to highlighting how mOSAIC is responding to current challenges of cloud computing adoption. Moreover, in order to assess mOSAIC's platform positioning in the cloud market, we have started a survey on the open-source deployable platforms. The potential criteria used to differentiate the offers are various, and there is a lack of a consensus on the most relevant ones. Therefore, we proposed in this chapter a set of criteria that can be relevant for the particular case of open-source platforms.

Acknowledgments This work was supported by the grant of the European Commission FP7-ICT-2009-5-256910 (mOSAIC) and, in the case of the first author, also by Romanian National Authority for Scientific Research, CNCS UEFISCDI, PN-II-ID-PCE-2011-3-0260 (AMICAS).

References

1. Astrea IT Services: The Cloud Tutorial – Cloud computing platforms. http://the.cloudtutorial.com/CloudComputingPlatforms.html (2012). Accessed 18 May 2012
2. Caprarescu, B.A., Petcu, D.: Decentralized probabilistic auto-scaling for heterogeneous systems. CoRR arXiv:1203.3885v1 [cs.DC]. http://arxiv.org/pdf/1203.3885v1 (2012). Accessed 18 May 2012
3. Cope, R.: How open is open? A PaaS scorecard. Wazi – Open Source Articles, Tutorials and Licensing Information. http://olex.openlogic.com/wazi/2011/how-open-is-open-a-paas-score-card/ (2011). Accessed 18 May 2012
4. Cordeiro, T., Damalio, D., Pereira, N., Endo, P., Palhares, A., Gonçalves, G., Sadok, D., Kelner, J., Melander, B., Souza, V., Mångs, J.E.: Open source cloud computing platforms. In: Proceedings: GCC 2010, pp. 366–371, Nanjing (2010). doi: 10.1109/GCC.2010.77
5. Di Martino, B., Petcu, D., Cossu, R., Goncalves, P., Mahr, T., Loichate, M.: Building a mosaic of clouds. EuroPAR 2010 workshops. LNCS **6586**, 529–536 (2011). doi:10.1109/HPCSim.2011.5999853
6. Frincu, M., Villegas, N., Petcu, D., Mueller, H.A., Rouvoy, R.: Self-healing distributed scheduling platform. In: Proceedings of the CCGrid'11, pp. 225–234, Newport Beach (2011)
7. Höfer, C.N., Karagiannis, G.: Cloud computing services: taxonomy and comparison. J. Internet Serv. Appl. **2**(2), 81–94 (2011). doi:10.1007/s13174-011-0027-x
8. Hogan, M., Liu, F., Sokol, A., Tong, J.: Nist Cloud computing standards roadmap-version 1.0, Special Publication 500–291, December 2011. http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/StandardsRoadmap/NIST_SP_500-291_Jul5A.pdf (2011). Accessed 18 May 2012
9. Larus, J.: Programming clouds. LNCS **6011**, 1–9 (2010). doi:10.1007/978-3-642-11970-5_1
10. Loutas, N., Peristeras, V., Bouras, T., Kamateri, E., Zeginis, D., Tarabanis, K.: Towards a reference architecture for semantically interoperable clouds. In: Proceedings of IEEE CloudCom 2010, pp. 143–150, Indianapolis (2010). doi: 10.1109/CloudCom.2010.38

11. Moscato, F., Aversa, R., Di Martino, B., Petcu, D., Rak, M., Venticinque, S.: An ontology for the cloud in mOSAIC. In: Wang, L., Ranjan, R., Chen, J., Benatallah, B. (eds.) Cloud Computing: Methodology, Systems, and Applications, pp. 467–486. CRC Press, Boca Raton (2011)
12. Panica, S., Neagul, M., Craciun, C., Petcu, D.: Serving legacy distributed applications by a self-configuring cloud processing platform. In: Proceedings of the IDAACS'2011, vol. I, pp. 139–145, Prague (2011). doi: 10.1109/IDAACS.2011.6072727
13. Petcu, D., Frincu, M., Craciun, C., Panica, S., Neagul, M., Macariu, G.: Towards open-source cloudware. In: Proceedings of the UCC 2011, pp. 330–331, Melbourne (2011). doi: 10.1109/UCC.2011.53
14. Petcu, D., Craciun, C., Neagul, M., Panica, S., Di Martino, B., Venticinque, S., Rak, M., Aversa, R.: Architecturing a sky computing platform. Servicewave 2010 workshops. LNCS **6569**, 1–13 (2011). doi:10.1007/978-3-642-22760-8_1
15. Petcu, D., Macariu, G., Panica, S., Craciun, C.: Portable cloud applications – from theory to practice. Future Gener. Comput. Syst. (2012). doi: 10.1016/j.future.2012.01.009
16. Prentice, B.: Cloud computing and open source: an industry-altering one-two punch. Gartner Res. G00159058. http://www.gartner.com/id=742523 (2008)
17. Rochwerger, B., Breitgand, D., Epstein, A., Hadas, D., Loy, I., Nagin, K., Tordsson, J., Ragusa, C., Villari, M., Clayman, S., Levy, E., Maraschini, A., Massonet, P., Munoz, H., Tofetti, G.: Reservoir—when one cloud is not enough. Computer **44**, 44–51 (2011)
18. Saleh, M.: Cloud platforms: a rational comparison. http://staff.ppu.edu/elearning/docs/UnderstandingCloudPlatformsARationalComparison.pdf (2011). Accessed 18 May 2012
19. Takako Endo, P., Estácio Gonçalves, G., Kelner, J., Sadok, D.: Survey on open-source cloud computing solutions. In: Proceedings of the SBRC 2010. http://sbrc2010.inf.ufrgs.br/anais/data/pdf/wcga/st01_01_wcga.pdf (2010). Accessed 18 May 2012
20. Voras, I., Mihaljevic, B., Orlic, M., Pletikosa, M., Zagar, M., Pavic, T., Zimmer, K., Cavrak, I., Paunovic, V., Bosnic, I., Tomic, S.: Evaluating open-source cloud computing solutions. In: Proceedings of the MIPRO 2011, pp. 209–214, Opatija. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5967051&isnumber=5967009 (2011)
21. Voras, I., Mihaljevic, B., Orlic, M.: Criteria for evaluation of open source cloud computing solutions. In: Proceedings of the ITI 2011, pp. 137–142, Cavtat. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5974009&isnumber=5973970 (2011)
22. Xia, T., Zheng, L., Nenghai, Y.: Research on cloud computing based on deep analysis to typical platforms. LNCS **5931**, 601–608 (2009). doi:10.1007/978-3-642-10665-1_59

Index

Z. Mahmood (ed.), *Cloud Computing: Methods and Practical Approaches*, Computer Communications and Networks, DOI 10.1007/978-1-4471-5107-4