Mazin Yousif
Lutz Schubert (Eds.)

LNICST 112

Cloud Computing

Third International Conference, CloudComp 2012
Vienna, Austria, September 2012
Revised Selected Papers

ICST

Springer

Lecture Notes of the Institute
for Computer Sciences, Social Informatics
and Telecommunications Engineering 112

Mazin Yousif   Lutz Schubert (Eds.)

Cloud Computing

Third International Conference, CloudComp 2012
Vienna, Austria, September 24-26, 2012
Revised Selected Papers

Springer

Volume Editors

Mazin Yousif
T-Systems International
Portland, OR, USA
E-mail: mazin.yousif@t-systems.com

Lutz Schubert
Universität Ulm
Institut für Organisation und Management
von Informationssystemen, Ulm, Germany
E-mail: lutz.schubert@uni-ulm.de

ISSN 1867-8211 e-ISSN 1867-822X
ISBN 978-3-319-03873-5 e-ISBN 978-3-319-03874-2
DOI 10.1007/978-3-319-03874-2
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013956154

CR Subject Classification (1998): D.2, C.2, C.4, H.2, I.2, H.4

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

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

Preface

It is a great pleasure to welcome all attendees to CLOUDComp2012. CLOUD Computing is surrounded by advertising hype. One might think it is the solution to all known problems in ICT. The reality is different. Cloud Computing has attracted the attention of businesses wishing to reduce IT costs (either internally or by outsourcing) and increase both flexibility of ICT delivery and accountability to business units for the ICT utilized. It has attracted researchers because of the host of ICT challenges brought into the spotlight by CLOUD Computing.

The advantages of CLOUD computing are attractive. However the challenges – ranging from technical difficulties e.g. in interoperation through to legalistic difficulties concerning the geolocation of personal data – raise interesting and complex research questions requiring solutions.

CLOUDComp2012 brings together research papers which have been peer reviewed by an excellent and representative Programme Committee, panels to discuss the pressing issues in CLOUD computing, and some inspiring invited talks. Located in the charming and cultural city of Vienna at an enchanting time of the year CLOUDComp2012 promises to be an exciting and stimulating event. It will surely advance our understanding of CLOUD Computing and doubtless open up new directions for research and development.

September 2013

Keith Jeffery
Lutz Schubert
Mazin Yousif

Organization

General Chair

Keith Jeffery	Science and Technology Facilities Council, Rutherford Appleton, UK

TPC Chairs

Mazin Yousif	High performance computing centre Stuttgart, Germany
Lutz Schubert	Head of Architecture (RDS), T-Systems, Germany

Local Chair

Ivona Brandić	Distributed Systems Group Information Systems Institute, Vienna University of Technology, Austria

Conference Manager

Erica Polini	EAI, Italy

Technical Program Committee

Albert Zomaya	University of Sydney, Australia
Javid Taheri	University of Sydney, Australia
Josep Manuel Bernabeu	Microsoft
Michael Behrendt	IBM, Germany
Mikhail Smirnov	Fraunhofer, Germany
Rajkumar Buyya	University of Melbourne, Australia
Simon Dobson	St. Andrews University, UK
Young Choon LEE	University of Sydney, Australia
Dr. Jürgen Falkner	Fraunhofer, Germany
Prof. Beniamino di Martino	Seconda Università degli studi di Napoli, Italy
Dr. Dimosthenis Kyriazis	NTUA, Greece
Dr. Ake Edlund	KTH, Sweden

Prof. Yi-Ke Guo	Imperial College London, UK
Dr. Karsten Oberle	Alcatel-Lucent Bell Labs, Germany
Ivan Breskovic	Vienna University of Technology, Austria
Dražen Lučanin	Vienna University of Technology, Austria
Toni Mastelic	Vienna University of Technology, Austria
Jia Liu	University of Stuttgart, Germany
Daniel Rubio Bonilla	University of Stuttgart, Germany
Eugen Volk	University of Stuttgart, Germany
Dr. Peter Kunszt	SyBIT, Germany

Table of Contents

The Need to Comprehend Clouds: Why We Still Can't Use Clouds Properly

Daniel Rubio Bonilla[1], Lutz Schubert[2], and Stefan Wesner[3]

[1] HLRS, University of Stuttgart,
Nobelstr. 19, 70569 Stuttgart, Germany
rubio@hlrs.de
[2] IOMI, University of Ulm,
89069 Ulm, Germany
lutz.schubert@uni-ulm.de
[3] KIZ, University of Ulm,
89069 Ulm, Germany
stefan.wesner@uni-ulm.de

Abstract. Clouds have become the modern concept of utility computing – not only over the web, but in general. As such, they are the seeming solution for all kind of computing and storage problems, ranging from simple database servers to high performance computing. However, clouds have specific characteristics and hence design specifics which impact on the capability scope of the use cases. This paper shows which subset of computing cases actually meet the cloud paradigm and what is needed to move further applications into the cloud.

Keywords: Cloud, Use Cases, Cloud Dwarves, Cloud Performance Criteria.

1 Introduction

The cloud concept allows reacting to system load dynamically to distribute the services according to actual usage, thus reducing the cost of ownership and leading to better resource utilisation. Clouds have become the modern paradigm of utility computing. At the same time, with the rise of GPU computing and multicore processor architecture, there is a growing belief that performance is proportional to the number of resources. It is thus frequently assumed that clouds can implicitly increase the performance of applications.

This assumption is however wrong for two major reasons: (a) performance is not generally proportional to number of resources and (b) applications do not simply change their behaviour (and thus quality criteria), just by being deployed in the cloud. There has been an abundant discussion on scalability and performance limitations, which shall not be repeated here (see e.g. [1]). This paper will elaborate why these limitations apply and which effect they have on the usability of the cloud for different application scenarios (section 2). It will give an assessment of the difficulty and expected value of migrating use cases to the cloud and will provide a first approach for classifying them regarding their benefits from clouds (section 3). We will show in particular that many factors regarding the relationship between code and cloud behaviour are effectively still unknown and outline the necessary work that needs to be done in order to improve future exploitation of cloud systems (section 4).

2 Cloud Delimiting Factors

To really exploit the full potential of cloud environments, it is absolutely necessary to first understand what clouds are and thereby which capabilities they really offer. Even though the concepts are widely known, the principles behind their realisation, and thus their limitations are less well documented. This is due to the quick uptake on the market, as well as because there is no reference technology for realising clouds, though Amazon EC2 and Google Docs are the de facto reference infrastructures.

According to the cloud report published by the European Commission, the primary cloud characteristics are specifically [3]:

- Utility Computing
- Elasticity
- Availability & Reliability
- Ease of Use

2.1 Size and Interconnect

A cloud environment must thus consist of multiple computing systems that can dynamically host multiple instances of the same service / application. In other words, that can replicate the functionality offered according to the current demand, and also reduce it in a similar fashion. Typically, this is realised by exploiting virtualisation technologies that host the respective logic, but can be easily encapsulated and therefore moved between instances, respectively replicated as a full image. The main point is that this behaviour is transparent to the user (i.e. does not require reconfiguration of their systems) and that it is steered according to the load, respectively availability requirements.

Due to technical constraints, elasticity is considerably slow, as distribution of the image typically involves communication of a large quantity of data, in particular if the image packs the whole operating system and execution environment of the service in question. With increasing complexity of the service and its runtime environment, the according delay implicitly increases and thus makes reaction to availability requirements slower. This is however a technical constraint posed by the typical implementation approach, and not by the cloud concepts as such.

As more and more applications move to the cloud, more and more users access internet-based services and the scale of individual applications increases to compensate the performance needs, the technical constraints become the major impeding factors for the fulfilment of the main cloud characteristics. This means that the number of resources available in a cloud environment may quickly become insufficient for the needs of the services, respectively users – in particular at times of peak demand. This leads to the same resource utilization problem again that kicked off the cloud concepts in the first instance.

Communication limitations become serious impeding factors for performance of web-based services. Not only the degree of sharing between connections and users, but also distance between server and client play a major role for this factor. Whilst downloading a file within your own country may reach a bandwidth of multiple MBps and a latency of less than 10ms; for foreign countries, depending on distance and connectivity, this may decrease to kBps and latency of a few hundred milliseconds. For a Gigabyte file this can make the difference between minutes and hours.

Latencies of milliseconds sound comparatively little considering the acceptable delays of about 1 second for loading and displaying a website without interrupting the user flow [4]. However, this latency is a constant, adding to any transaction between server and client. Implicitly, whilst it has hardly any impact on large, it creates massive delays for any interaction that bases on multiple communication exchanges with comparatively short messages. In particular for real-time interactive applications, such as MMO games, this built up in latency leads to massive problems.

Table 1. Network performance in different environments

	Latency	Bandwidth
Internet	150 ms	10 Mbps
Server Farms	10 ms	1 Gbs
High Performance Computing	<1 ms	up to 100 Gbs

Notably, latency and bandwidth are subject to physical constraints (cables, speed of light, routing etc.) that start to catch up with the requirements. It is of interest to compare the situation for average internet access with what is currently possible on the high end of the scale. Table 1 summarises the typical network performance parameters in three typical domains – obviously the figures may differ according to the server's capabilities, its load etc.

What is more important than the concrete figures is however that these domains differ by 1-2 orders of magnitude in performance, i.e. internet is 10 times "slower" (to react) than intranet, which in turn is (more than) 10 times slower than high performance computing. Similarly for the bandwidth, where internet's capabilities are 1/100s of the intranet, which in turn is roughly 1/100s of High Performance Computing (HPC). We must in this context distinguish between the cloud resources (typically server farms), and connection with the end-user. So that cloud access to the cloud, and potentially between clouds is internet-based, whereas resources within the cloud can reach an interconnect of the speed of server farms.

Notably cloud systems try to reduce the communication limitations further by compensating for distance and concurrent usage through replication and relocation of the service instances. Even so, the inherent (physical) restrictions cannot be overcome, but are the core constraining factors. What is more, the level of freedom, i.e. the degree of influence a management system can take, is higher with "higher-level" domains, than with low-level ones. This is partially due to the fact that the performance is actually achieved by maximum alignment of the system layout with the application cases – factors such as uncontrolled concurrency, or even shared resources are thus not supported in these domains in the first instance.

2.2 Execution System / Middleware

In order to allow for the dynamic distribution and instantiation of services, respectively applications, it is necessary to be able to package them and re-instantiate them with their full context. Virtualisation allows not only to host the full service environment, but also pausing, replicating and moving it with little additional overhead and with little impact caused by the underlying hardware.

However, virtualisation technologies limit the performance of the actual hardware, restrict the scalability and cannot easily share data and / or code between instances. As such, the most straight-forward usage for virtualisation consists in providing completely isolated images, where every user accesses his own instance. In cases where applications share data (e.g. Wikipedia like social environments) or even parts of the logic (such as in stock market analysis), solutions become more complicated: In these cases, it is more advisable to exploit an execution framework of its own, dedicated to the respective use case and on which specific data and configurations, rather than code is enacted. This means that every user is effectively using the same logic with different distributions and instances of data and shared algorithms.

A similar approach can be used to expose a dedicated application programming interface for the respective usage domain that allows the user to develop his / her own logic on top of a (cloud) managed infrastructure. This allows best adjustment to the underlying infrastructure and management of the enactment according to the specific domain requirements, but it at the same time limits the application scope.

The essence in these approaches is similar: to completely retain control over the systems and in particular the execution of the hosted logic – only in this fashion is it possible to realise the essential cloud capability, namely the dynamic adaptation to load criteria. The elasticity focuses specifically on the number of instances to be replicated in order to fulfil the respective quality of service criteria. The management and adaptation framework must thereby be well adjusted to the actual application case, in order to enact the required consistency mechanisms for shared data, to reroute messaging according to the instance relationships etc.

Management and adaptation create additional overhead that reduces execution performance, thus restricting dynamicity considerably. Most cloud environments take therefore generally a pro-active and cautious approach towards elasticity, i.e. create instances ahead of time (i.e. before the availability criteria is threatened to be violated) and keep instances alive even after need, to reduce re-instantiation time.

Again we can make a comparison between different means of instantiating and relocating an application / service / image, though comparing mechanisms rather than domains (see Table 2). These figures thereby completely neglect additional overhead for communicating the associated data over the network as described in the preceding section. Essentially, with the complexity of the mechanism (e.g. virtual machines over processes) the amount of data that needs to be shifted with the new instance increases, too. The effective speed in the according domain is therefore reduced by the factor produced by the typical interconnect setup (see above).

Table 2. Instantiation / replication handling performance

	Delay
Virtual Machines	Minutes
Managed Processes / Services(PaaS)	Seconds
Threads (OS)	Milliseconds

Similar to the interconnect performance, we can note that there are multiple orders of magnitude difference between the individual mechanisms, which impact on the instance handling speed accordingly.

2.3 Programmability

Programmability is a major issue for full usage of cloud systems. Keeping the wide customer base in mind, the programming language should adhere to well-established models, such as Java, C#, PHP etc. Notably, the language of the application itself is secondary, if full-fledged virtual images are provided (IaaS), but is of major concern in PaaS environments, where the extensions typically adhere to a specific language.

Stand-alone, non-adapted code versions work fine in IaaS cases where each instance can be treated completely isolated. Once dependencies, i.e. shared data is introduced, the application logic needs to be altered accordingly. If the developer wants to control specific features of the cloud behaviour (such as scaling behaviour), the according knowledge needs to be encoded right into the logic. It is worth noting in this context that not all cloud providers automate the elastic behaviour, but expose an according programming interface to the developer to trigger instantiation himself.

The main task therefore consists in rewriting the logic to externalise content and part of the logic. One subsequently introduces delays for data-exchange, which is proportional to the factors discussed above. A straight-forward approach may consist in externalising the database system and sharing it between instances, but this will create a bottleneck, which may counter all benefits from migrating to the cloud in the first instance. Introducing dedicated synchronisation points similarly leads to communication delays that delimit the execution performance.

Modern programming models allow for easy integration of web interactions and cloud features, but the relationship to the actual performance and behaviour is not clear. In other words, the available languages are not able to compensate for the deficiencies introduced by the algorithm itself. For example it makes a major difference whether the developer intends to share a large database that is updated multiple times, or whether the application effectively just exchanges data at session begin or end time. The language allows for either way without giving indicators of the performance impact, let alone controlling this impact. Most languages completely rely on the framework, respectively middleware to execute the transactions. This means that the user cannot estimate the timing impact correctly.

Lower level domains, such as HPC, therefore do not rely on managed communication frameworks, but essentially leave full control with the developer. Extensions provided through MPI or OpenMP only expose standardised mechanisms for common procedures, rather than taking responsibility away from the programmer. Thus, the program has to be instance aware to deal with the dynamic instantiation and relocation – typically it is therefore only exploited under very controlled conditions.

To optimally exploit the cloud characteristics, partial and conditional sharing of both code and data would have to be supported and ideally widely automated. No current programming model allows for such support though and the manual approach currently undertaken by e.g. HPC is highly complex, leaving only very few people ready to deal with it in the first instance.

2.4 Consequences

Considering the primary characteristics of clouds, it is obvious that full exploitation of the capabilities is tied to the use cases, even though the principle allows for a broader scope with the implicit loss of performance, respectively quality of service. Clouds are often treated as the panacea of IT, but they are effectively specialised domains, so that outside of this domain other environments show better performance.

These domains, however, cannot be easily specified along the line of "eScience", "business applications" or similar. Rather, they deal with specific immediate requirements (availability, elasticity etc.), thus allowing for a wide scope of use cases all following under "utility computing". But similarly, "eScience" or "business applications" cannot easily be translated into one specific IT domain either.

We can note, which conditions the application should fulfil in order to be able to exploit the cloud, respectively to benefit from it in the first instance – these are:

- Dynamic number of users, respectively requests
- Small and infrequently shared data between a limited number of instances
- Communication between instances and data sharing is primarily asynchronous
- Comparatively low scale of the application itself (i.e. degree of parallelism)

As noted, many of these conditions can be seen as comparatively "lax" boundaries, i.e. they allow for a certain degree of freedom – e.g. synchronous messages are certainly possible, if the delay is not crucial for the execution of the application. We can however also denote hard boundaries that cloud systems cannot fulfil and therefore constrain the scope of usage:

- Single instance applications simply do not benefit from cloud management
- Data intensive applications where performance is crucial
- Large scale applications which require a large amount of resources for fulfilling their work. Notably, they might run on the cloud, depending on their communication dependency, but they do not exploit the essential cloud capabilities
- Applications where execution performance is crucial and where performance is influenced by any communication related overhead, including instantiation

3 Classifying Your Application

As elaborated in the preceding chapter, clouds are constrained in their applicability scope and many use cases either do not benefit from the additional capabilities offered by the cloud, or even suffer from its limitations. It has also become obvious though, that it is not easy to classify an existing application or use case for its potential benefits from cloud environments. Most of the core characteristics identified in the preceding section may be hidden within the application, i.e. it requires real in-depth expertise of the application to identify it. What is more, it is not clear whether the according constraints could not be overcome by changing the code (see next section).

Here we provide a set of criteria that may help in classifying the use case / application and to assess the potential benefit to be gained from the cloud:

3.1 Classification Criteria

With respect to the preceding assessments, three major criteria stick out:

1. Scope (degree of sequentiality, respectively parallelism)
2. Strength (or "tightness" of the communication between instances)
3. Density (or amount and size of exchange / communication between instances)

These three criteria can more or less be directly mapped to instantiation speed, communication latency and bandwidth as elaborated in the first section:

Scope. The number of instances required and the number of instances maximally possible (i.e. scalability) define the resource "hunger" of an application and therefore its need for a larger infrastructure at all. We thereby need to distinguish between the resource need of a single instance (parallelism) and the amount of instances required due to the amount of requests / users (concurrency). The effective need is therefore the product of parallel scope time concurrency scope.

Strength. The acceptable communication delay for any shared data access or information exchange provides an indicator for the acceptable latency and therefore for the type of infrastructure required. We can most of all distinguish thereby whether the exchange is synchronous and therefore directly impacting on execution performance, or whether the communication can be executed asynchronously, in which case the impact on performance is considerably less.

Density. The communication delay in itself may have little impact if the amount of messages, respectively data accesses is comparatively low and if the messages in themselves are comparatively small. For example, even a synchronous, blocking request of multiple seconds duration may be ignored, if it only occurs a few times per hour, so that delay << execution time.

3.2 Analysing the Use Case

The benefit of the criteria provided above is that they can be extracted from a given application in a fairly easy way, though their interpretation may not be as straight-forward as the conditional cloud characteristics listed in the preceding section. In the following we assume that the application has not yet been migrated to the cloud, though the same principles would apply:

As has been noted multiple times, the cloud is an infrastructure consisting of multiple dynamically allocated servers that can host an elastic number of application and data instances, according to requirements. In order to exploit this infrastructure, the following migration scenarios are possible:

1. Keep the application standalone, sharing no data or code
2. Share data between instances to allow for collaboration, networking etc.
3. Distribute the code to allow partial sharing of functional logic
4. Distribute code and data over the infrastructure

The general idea is to make use of multiple resources thereby creating a better user experience. The simplest case is obviously 1), where no further actions have to be taken and each instance is simply hosted in its full environment (image) - this provides the service with considerable enhancements of availability through the cloud infrastructure, but provides no further benefits. For example, this does not allow users to share data with each other, nor does it allow to make efficient reuse of intermediary processing results, such as in live rendering - in short it does not allow for any collaborative enhancements. It also does not allow for exploitation of concurrency between individual functions, so that generally this creates no performance benefit.

Most enterprises and individuals consider transition to the cloud to improve service quality however and want particularly to improve execution performance, reduce resource costs to their minimum, and offer collaborative features. In such a case, any of the options 2.-4. may apply, whereas complexity increases with the higher options.

To identify which options are possible, it is necessary to analyse the use case in question. To this end, the dependencies between functional and data units need to be analysed with respect to their potential for being distributed. Whilst model driven architectures do provide some insight into these dependencies, the actual impact on the criteria listed above can only be estimated and may vary during execution.

To gather meaningful data, the best approach consists in instrumenting the code or by monitoring the memory behaviour as recommended and elaborated by the S(o)OS project. The S(o)OS project furthermore indicates how this information can be interpreted to the purpose of code segmentation, distribution and parallelisation: by weighing access frequency and read / write patterns the degree of dependencies (strength and density) can be qualified. Using this weight, segmentation cuts may be performed in the code according to the communication (and exeuction) capabilities of the destination platform [5].

We need to extend this model to investigate the impact of more or less arbitrary access attempts via the externalised interface. This behaviour can either be simulated and evaluated through actual memory usage, or propagated along the weighted graph in the form of heuristic representations of the user behaviour.

The analysis of the respective application should in all cases be able to indicate access frequency and size (density), as well as the dependency between request and actual usage (strength). By furthermore annotating the data with respect to whether it is expected to be shared between users, the analysis thus provides a fairly accurate distribution architecture, including communication requirements and instance control indicators. The according mechanisms will be investigated in the upcoming EC funded research project PaaSage, and published shortly.

3.3 Interpreting the Results

Basing on the analysis, the use case's "cloud potential" can be assessed (respectively the most suitable platform can be identified):

The code can principally be segmented where the expected control behaviour changes substantially (i.e. a boundary between shared and non-shared functions / data). For example, the graphical user interface of an application can typically be easily segmented from the actual logic. In the application analysis graph this is denoted through individual memory spaces per each external instantiation request, rather than a shared memory region. These separate regions are an indicator that either code and data, or just data can be maintained per individual instance, whereby the trigger in this case may be the external request, i.e. user connection.

The impact from segmentation and clustering can be assessed by measuring / calculating the implicit changes in strength and density of the connection and countering it with the scope of the instance, and therefore resource need. There is accordingly no strict boundary between use cases that are suitable and those that are not suitable for cloud environments. Just like in any IT case, execution can be enforced even in "foreign" environments, but at the cost of performance. Accordingly, the boundary is implicitly given by when the loss (performance, migration cost etc.) outweighs the gain (elasticity, availability etc.)

The table below may serve as an indicator for which freedom in the terms of the three main criteria can be considered acceptable for a cloud environment, before the loss will start to outweigh any potential benefit. Note that table 3 is not providing clear boundaries either - instead with lower rows in the table, the likelihood of benefitting from cloud environments diminishes.

Table 3. Suitable infrastructures for specific criteria combinations. Note that the "environment" transitions smoothly from Clouds via Server Farms / Grids to High Performance Computing.

Scope	Strength	Density	Suitable Environment
small scale, many instances	low	low	Clouds
small scale, many instances	medium / streaming	low	...
medium scale, few instances	low	medium	...
few instances, medium scale	medium	medium	...
...			...
large scale, few instanced	high	High	HPC

There is a high risk with this classification to neglect data dependent behaviour, i.e. the same code may exhibit different strength, density and scope, depending on the data it is processing. For example, the input parameter may directly specify the number of instances to create. If this relationship is not known, the impact of this factor should be evaluated through analysis of common data structures.

4 Making Your Application Cloud-Efficient

Whilst the analysis provides some insight into the potential of a use case to be migrated to the cloud, it does not help to assess alternatives, i.e. whether the code could principally be restructured to exploit the cloud features more effectively. For example a mathematical algorithm with high data dependencies will create a high requirement for density and will be fully impacted by the effective communication strength. But a large iteration over individual (i.e. data-independent) actions, such as extensive independent calculations on a parameter range, may be easily distributed, as the requirements for communication are comparatively low.

By choosing another implementation approach, the same application may expose characteristics befit better the cloud. We can consider this a functional transformation from logic A to B in a way that B exhibits a different set of criteria, more suitable for the specific destination platform. Transformations such as this are very typical for parallelisation efforts, where the developer attempts to make his code suitable for HPC clusters in order to exploit the additional performance of supercomputers.

As has been shown, the key driver for cloud based applications is not performance, but concurrency, availability and elasticity. The terminology of cloud providers is often highly confusing with this respect: when talking about scalability, performance and availability, they generally refer to the level of services typical for the web domain. A user may however easily expect scalability and performance to the degree of HPC. It must be stressed though that within the web service domain, clouds offer a major improvement over alternative approaches. To effectively migrate applications to the cloud, it is of utmost importance to be aware of this difference and to build the application around these primary constraints.

On this basis, core functional building blocks can be identified that are most suited for the cloud environment, respectively that relate better to other domains. Conversion means can be identified that help identification and conversion of specific routines to meet the environment's conditions best. Such information would allow the developer to choose the right algorithms at design time; at the same time it would provide indicators as to whether a certain application is principally suited for the designated environment. Such efforts have been undertaken in the domain of High Performance Computing for considerable time now, such as Berkley’s report on key algorithmic cores [2]. A similar attempt for clouds is now being initiated by the EC funded project PaaSage, but essentially the necessary expertise is still lacking as of today.

Some indicators can, however, already be identified and may serve as a basis for further elaboration. These indicators obviously relate strongly to the key criteria identified within this paper, namely: scope, strength and density:

Scope (and implicitly scale) is impacted by the degree of sharing within the application. One way of improving scope therefore consists in reducing strength and density, as discussed below. Another way of improving scope consists in clustering the logic, in alignment with the data dependencies, thus generating a modular software structure that shows different scale within the different modules.

Strength is a major hemming factor in efficient usage of clouds, as the communication delays must implicitly degrade performance. Programming models generally do not allow differentiating between time for fetching and using data, thus leading to the assumption that effectively data is available immediately. The effort to compensate for strength must instead be taken by the developer.

The best way to approach this consists in using asynchronous messaging right from design time. This automatically forces the developer to organise the code in a fashion that caters for weak strength communication, and to exploit idle time for performing other tasks. Asynchronicity also means that eventual consistency should be considered for shared data, rather than immediate consistency. Notably, switching to asynchronicity is not always possible, which could mean that the delay is secondary, or that the application is simply not suited for cloud infrastructures.

Density plays a particular role in combination with strength: multiple messages or huge messages can sum up in the performance degradation. Even if coupling is weak, the mass of communication will create an impact. The key point for the developer is to identify the right mix of number and size of messages – e.g. by packaging multiple small messages into one bigger to reduce the impact of latency, or by splitting up bigger messages to deal with bandwidth limitations.

Developers should carefully evaluate the impact of multitenant behaviour upon the application: generally not all data needs to be fully shared, but only within groups of instances and here only parts of the data at different times. Keeping an eye on the specific intention is a good way to reduce density concerns. For example, Google docs share documents between groups of people, yet the only data that needs to be communicated is the changes that are actually taking place in the document.

5 Conclusions

Clouds have been around for a considerable time now, but there still exists little knowledge about their actual capabilities and limitations, let alone about how to address them and which use cases are most suitable for it. More expertise needs to be gathered about the essential core application logics that are most suited for clouds. These core elements can be used as a basis to build up cloud applications, but also as a means to quantify existing applications to assess their "cloud-suitability". The paper presented provides an initial outline for identifying these cores by classifying the main criteria constituting cloud application performance and behaviour.

Acknowledgments. The work presented in this paper was partially funded by the EC research projects S(o)OS and PaaSage.

References

1. Huonder, F.: Parallelization for Multi-Core. Program Analysis and Transformation (2009)
2. Asanovic, K., Bodik, R., Catanzaro, B.C., Gebis, J.J., Husbands, P., Keutzer, K., Patterson, D.A., Plishker, W.L., Shalf, J., Williams, S.W., Yelick, K.A.: The Landscape of Parallel Computing Research: A View from Berkeley. EECS Department. University of California, Berkeley (2006)
3. Schubert, L., Jeffery, K.: Advances in Clouds - Report from the second Cloud Computing Expert Meeting. European Commission, Brussels (2012)
4. Miller, R.: Response time in man-computer conversational transactions. In: Proceedings AFIPS Fall Joint Computer Conference, vol. 33, pp. 267–277 (1968)
5. Schubert, L., Kipp, A.: Principles of Service Oriented Operating Systems. In: Vicat-Blanc Primet, P., Kudoh, T., Mambretti, J. (eds.) GridNets 2008. LNICST, vol. 2, pp. 56–69. Springer, Heidelberg (2009)

Assessing the Readiness to Move into the Cloud

Leire Orue-Echevarria, Juncal Alonso, Marisa Escalante, and Stefan Schuster

TECNALIA. ICT / European Software Institute Divison
Parque Tecnológico Ed. #202. E-48170 Zamudio. Spain
{leire.orue-echevarria,juncal.alonso,marisa.escalante,
stefan.schuster}@tecnalia.com

Abstract. The race to keep software compatible and optimal with respect to the latest trends is hard. 90% of software cost can be due to maintenance, and 75% on developing new features to stay competitive and relevant. The industry progresses through periods of incremental development interspersed with true paradigm shifts. Legacy software must keep up the pace.

At present we are experiencing one of these paradigm shifts, as remarked by the EC [1] "The speed of change in Internet technologies continues to be impressive. Software is becoming more and more pervasive: it runs on the devices that we use every day ... [opening] a new world of possible applications". Today, technological and business model innovation generates large demand for the transition of legacy software towards modernization. However, software modernization is not a trivial issue and if improperly done, it dangers the business continuity and sustainability.

This means that for any company meditating about the transition to the new paradigm of cloud computing, there is a need to have at its disposal an innovative and combined technical and business analysis on the maturity and prospect of the legacy application. The major target of this process is to identify in advance the perspectives of the migration and pre-evaluate the performance and business benefits with relation to the cost of the process. For the first time, the business value will be directly attached to the technical performance.

This paper presents this aforementioned approach, being currently developed and tested, in order to assess the maturity of an application and the convenience of migrating to the new cloud computing paradigm or not, based on quantitative indicators while always ensuring the company's business continuity. Following this approach, questions such as cost and effort of the migration, impact of new business models in the company or return of the investment will be provided in advance of tackling the actual modernization.

1 Introduction

New developments in the way services and applications can be delivered over the Internet have opened up huge opportunities to 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, i.e. providers 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 the last five years and how the way we work has changed thanks to the breakthrough of cloud computing, smartphones or social networks. 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 with regard to these new circumstances but without disrupting the business continuity of existing customers.

However, managing 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 lifecycle of software, from requirements to run-time and delivery has to be re-adapted 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 challenge that all companies face is **the decision whether to migrate their existing products or to start from scratch**. Current open [6] [7] [8] [9] and proprietary migration methodologies [10] [11] [12] [13] [14] 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 of the investment need to be answered **before** tackling the actual modernization. If the estimates they obtain suit their expectations and they finally decide on the migration to a service-based software, reusing as much as possible from the old one, they will face further challenges and difficulties, not only with respect to the usage of new technologies, or architecture but also with respect to assumptions that companies usually take for granted and then are no longer valid.

2 Approach

The main objective of the approach presented in this paper, currently being developed and tested, is to provide a set of methods and techniques that will support companies on the assessment for the modernization of their software towards a Cloud delivery model, sustaining them on the migration strategy and providing the required tools to analyse the impact of the potential transformation of the software in the company.

The modernization of the software and its delivery will be analyzed under two different, but intertwined dimensions: one focusing on Technology (architecture, performance, reliability, data schema, and so on) and another one on Organisational & Business aspects (pricing model, market addressed, organisational processes, etc). This is rather significant since the business model offered by the organization (based on the delivery of software artefacts) will change from a product to a service. In the cloud world, decisions taken at business level constraint the technology and vice versa, for instance, a billing component (business related) needs a monitoring component of application use (technology related).

After the assessment, the assessed organizations will be able to visualize a maturity map where the position of their current business service is shown, as well as the potential position (in terms of technology modernization and business model changes) once the migration takes place.

In addition, the 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 figures will be presented in measurement units and concepts easily shared, recognised and acknowledged by stakeholders.

Summarizing, the main outcomes of this approach are:

- a **method for characterising the technical and business dimensions** of the current legacy application, in particular those concerns related with its modernisation towards a selected target,
- a set of common **metrics and indicators** that characterise 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, operational risks,
- a **modernization strategy** with the activities to carry out in case the organisation decides to continue with the modernisation process after the figures are analysed.

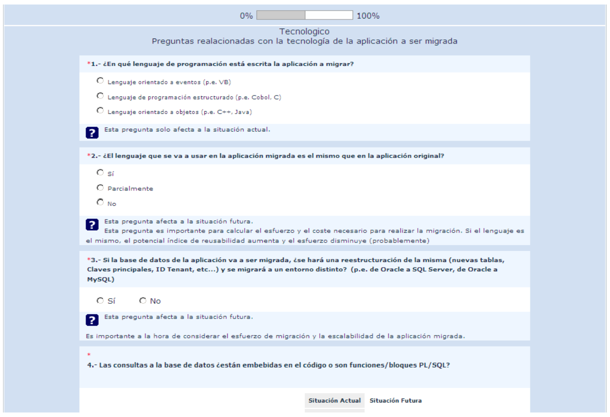
A. Business and Technical Modernization Assessment

This step focuses on the characterization of the metrics and indicators (metrics weighed and combined) of the business and technical dimension of the legacy application and the company, such as the pricing model, the targeted market, the product sustainability, SLAs, legal issue, metrics that describe the legacy application source code and data schema complexity, compliance with baseline legacy technologies, gap estimation between legacy and target baseline, etc. Authors of this approach have not yet found a similar Business and Technical Modernization Assessment procedure in literature, neither a classification of applications from an architectural point of view, nor a business model and process one.

In order to perform this assessment several issues and knowledge are pre-required. Among them: artefacts and knowledge related to source code and architecture, development process, GUI, source environment and desired environment, source and target infrastructure, covered and uncovered non-functional requirements.

This assessment is to be executed in several steps:

Step 1: Fill in on-line questionnaires. Examples in Spanish can be found here [2]. 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 fulfilment, helpdesk, maintenance, privacy and trust practices, marketing, business model, pricing model, target platform model (private, public or hybrid cloud) and performance and reliability. The questions are 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.



0% 100%

Tecnologico
Preguntas realacionadas con la tecnología de la aplicación a ser migrada

*1.- ¿En qué lenguaje de programación está escrita la aplicación a migrar?

- Lenguaje orientado a eventos (p.e. VB)
- Lenguaje de programación estructurado (p.e. Cobol, C)
- Lenguaje orientado a objetos (p.e. C++, Java)

Esta pregunta solo afecta a la situación actual.

*2.- ¿El lenguaje que se va a usar en la aplicación migrada es el mismo que en la aplicación original?

- Sí
- Parcialmente
- No

Esta pregunta afecta a la situación futura.
Esta pregunta es importante para calcular el esfuerzo y el coste necesario para realizar la migración. Si el lenguaje es el mismo, el potencial índice de reusabilidad aumenta y el esfuerzo disminuye (probablemente)

*3.- Si la base de datos de la aplicación va a ser migrada, ¿se hará una reestructuración de la misma (nuevas tablas, Claves principales, ID Tenant, etc...) y se migrará a un entorno distinto? (p.e. de Oracle a SQL Server, de Oracle a MySQL)

Sí No

Esta pregunta afecta a la situación futura.

Es importante a la hora de considerar el esfuerzo de migración y la escalabilidad de la aplicación migrada.

*
4.- Las consultas a la base de datos ¿están embebidas en el código o son funciones/bloques PL/SQL?

Situación Actual Situación Futura

Fig. 1. Assessment Questionnaires

Step 2: Based on the results attained in the questionnaires, an analysis is executed. 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" [3] this approach 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 takes place.

Step 3: Presentation of the results in a graphical manner. The authors found that an adequate and visible way to do it is by means of a quadrant. An example of such a quadrant is shown in the next figure:

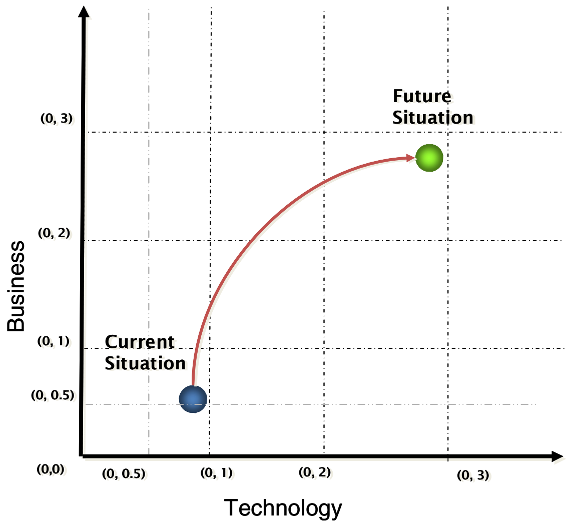


Fig. 2. Position of an application in the quadrant

The current maturity levels of the Technology and Business axis, as shown in the figure above, have been established based on the professional experience and State-of-the-Art studies [15] [16] [17] [18] from the authors. Nevertheless, the maturity levels will be accordingly updated with new achievements.

These maturity values represented in each axis have the following meanings:

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 (instalment), support, updates, upgrades, maintenance are paid under a different fee model than the license. No helpdesk. 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 10-20% 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. Helpdesk is institutionalized but not 24x7 and only in certain languages. Existence of SLA, upgrade protocol but upgrades are still seldom, legal department.
- (3,0) 100% of the sales are from the product as a service. Existence of a 24x7 helpdesk, multilingual, Marketing mostly done through the Internet (social media), SLA, upgrade protocol and procedures, Long Tail business model.

B. Technical Feasibility Analysis

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, centres 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 next:

- **Code analysis:** The goal of such an analysis is twofold. On one hand, 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 Cyclometric Complexity, CBO (coupling between objects), RFC (response for class) [19] RFC∞ [20] MPC (message passing coupling) [19] DAC (data abstraction coupling) and DAC1 [21] ICP (information-flow-based coupling) [22] 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 3rd party COTS and/or with other applications internal to the company.
- **High Level Modelling:** 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 modelling the application with UML in its different views, seizing the power of Reverse Engineering Techniques. Also important at this stage is also to analyse the database schema, data and transaction model in order to select the best database architecture (RDBMS, NoSQL, a hybrid solution) and migration strategy towards 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.

The main outcomes of this analysis are:

- a set of metrics and indicators that show how complex the code is and thus how much effort will be needed to transform the legacy application to a cloud oriented environment. These metrics will be used in the Business Feasibility Phase to extract the costs of the migration strategy,
- a classification of legacy artefacts to be considered in a posterior Reverse Engineering process, e.g.; source code, configuration files, data files, documentation, existing models, etc.
- based on the previous results, a taxonomy highlighting the main different types of legacy artefacts according to their corresponding characteristic and properties.
- effort calculation on the migration.

C. Business Feasibility Analysis

The goal of this business feasibility analysis is to provide guidelines that will aid the management level take the decision whether to tackle the migration to SOA and/or Cloud based on objective economic parameters. In order to do so, a cost-benefit analysis is being developed (including ROI and Payback) that will cover the specific issues related to this shift of business model, as well as means to calculate the impact and implications of changing business models in a company that is already sustainable.

Although a cost-benefit analysis has to be customized for each migration project, there are several common concepts that need to be analysed:

- Costs, divided in development costs and operational costs. Whilst the first kind focuses on how much it will cost to migrate an application compared to developing it from scratch, the second kind focuses on the costs related to training to employees with the new roles they are expected to have, the costs of the cloud provider, costs of regular updates and continuous maintenance, new marketing activities, as well as other structural costs.
- Revenues, considering not only the revenues from the business itself, that is, number of customers using the service, but also considering other issues that at first sight are not seen as direct revenues but that will eventually lead to that, such as:
    - (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 travelling for maintenance and installation.
    - (Reduced) Costs in marketing.
    - (Greater) margin by targeting new markets non reachable beforehand.

This analysis provides the management level with at least the following data:

- Net Present Value for the next five years.
- Return on Investment for the next five years.
- Payback in years.

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 SLA's, customer care, and other support processes but also those related to the business core, the software development, design and testing.

D. Modernization Strategy

Once the metrics and indicators have been analyzed, the organization will decide on the convenience 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 will be defined. This strategy will provide the organization with the needed roadmap in order to achieve the desired maturity expressed in the questionnaires.

3 Conclusion

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. 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.
- Future research with the technical and business feasibility analysis. Currently, these are rather manual and thus, time costly. Supporting tools will be developed in the near future.

Acknowledgement. This work has been partly funded by the Basque Government project SMARTWORK in the Future Internet and by the Spanish Ministry of Science and Innovation through the project named mCloud: Advance migration to the cloud [5].

Further acknowledgment goes to partners of the ARTIST FP7 proposal.

References

[1] http://cordis.europa.eu/fp7/ict/ssai/docs/call8objective1-2-brochure-web.pdf

[2] http://sg1.esilab.org/limesurvey/index.php?sid=29249&newtest=Y&lang=es

[3] Simon, F., Seng, O., Mohaut, T.: Code-Quality-Management. Dpunkt.Verlag, Heidelberg (2006)

[4] Orue-Echevarria, L., Alonso, J., Gottschick, J., Restel, H.: FROM SOFTWARE-AS-A-GOOD TO SAAS: CHALLENGES AND NEEDS. Developing a tool supported methodology for the migration of non-SaaS applications to SaaS. In: ICSOFT 2011 (2011)

[5] http://innovation.logica.com.es/web/mcloud

[6] 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. Paper presented at the Joint 1997 Asia Pacific Software Engineering Conference and International Computer Science Conference (1997)

[7] Khusidman, V., Ulrich, W.: Architecture-Driven Modernization: Transforming the Enterprise. OMG (2007)

[8] Warren, I., Ransom, J.: Renaissance: A Method to Support Software System Evolution Paper presented at the 26th Annual International Computer Software and Applications Conference (2002)

[9] http://www.sei.cmu.edu/reports/08tn008.pdf

[10] ftp://service.boulder.ibm.com/s390/audio/pdfs/G224-7298-00_FinalMigratetoSOA.pdf
[11] http://www.oracle.com/technetwork/middleware/soasuite/overview/wp-soa-suite-11gr1-2-129551.pdf
[12] http://download.microsoft.com/download/d/d/e/ddeb427d-dc05-4ab0-b47e-74f0a936d892/Real-World-SOA-At-The-Edge.pdf
[13] http://www.sap.com/platform/soa/adoptionprogram.epx
[14] http://media.amazonwebservices.com/CloudMigration-main.pdf
[15] Moyer, C.: Building Applications in the Cloud. Ed. Addison Wesley (2011)
[16] Reese, G.: Cloud Application Architectures. Ed. O'Reilly Media (2009)
[17] Sosinsky, B.: Cloud Computing Bible. Ed.Wiley (2011)
[18] Chate, S.: Convert your web application to a multi-tenant SaaS solution. IBM developerWorks (2010)
[19] Chidamber, S.R., Kemerer, C.F.: Towards a Metrics Suite for Object Oriented Design. In: Proceedings of OOPSLA 1991, pp. 197–211 (1991)
[20] Chidamber, S.R., Kemerer, C.F.: A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering 20(6), 476–493 (1994)
[21] Li, W., Henry, S.: Object-oriented metrics that predict maintainability. Journal of Systems and Software 23(2), 111–122 (1993)
[22] 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, Slovenia (1995)

Design and Implementation of a Multi-objective Optimization Mechanism for Virtual Machine Placement in Cloud Computing Data Center

Soichi Shigeta, Hiroyuki Yamashima, Tsunehisa Doi,
Tsutomu Kawai, and Keisuke Fukui

Cloud Computing Research Center, Fujitsu Laboratories Ltd.
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki 211-8588, Japan
{shigets,yama,micky,tkawai,kfukui}@labs.fujitsu.com

Abstract. Cloud computing is becoming a popular way of supplying and using computing resources. A cloud-computing data center is equipped with a large number of physical resources and must manage an even larger number of virtual machines (VMs). The center's VM placement strategy affects the utilization of physical resources, and consequently, it influences operational costs. Our goal is to develop a multi-objective optimization mechanism for VM placement that satisfies various constraints and results in the lowest operational cost. The number of possible combinations of VMs and hosts can be extremely large. For the mechanism to be practical, the number of possible combinations must be reduced. We reduced computational overheads by classifying VM hosts into a relatively small number of equivalent sets. Simulation results show that expected operational costs can be significantly reduced by applying the proposed mechanism.

Keywords: cloud computing, VM placement, multi-objective optimization.

1 Introduction

Cloud computing is becoming an increasingly popular way of supplying and using computing resources. A number of commercial cloud services, such as Amazon EC2 [1] and S3 [2], Google AppEngine [3], Salesforce CRM [4], and Fujitsu Global Cloud Platform [5] are presently being used to run business systems. Cloud services can be classified into three types: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). For example, Fujitsu Global Cloud Platform and Amazon EC2 are examples of IaaS. Google AppEngine is an example of PaaS, and Salesforce CRM is an example of SaaS. In this paper, our main focus is on an IaaS data center with particular emphasis on reducing operational costs.

Reducing operational costs is a key to achieve high cost-benefit performance in cloud computing data centers. Lower operational costs are also important for price competitiveness because they will be reflected in the price of a service.

Accordingly, the aim of this research is to develop a multi-objective mechanism to determine the lowest cost for virtual machine (VM) placement that considers various components of cost efficiency (such as electrical power consumption, availability, and load balancing).

Traditional systems have been designed and built to satisfy the peak load that was estimated in advance. However, although periodic fluctuations for business workloads may be predictable, it is difficult to predict fluctuation of demand for a public cloud. Load balancing among physical servers is a typical criterion to increase the efficiency in cloud computing data centers. Therefore, many existing resource schedulers determine VM placement on the basis of utilization of physical servers (typically CPU and memory usage) or VM hosts. However, efficient and effective VM placement involves many other factors, and existing resource schedulers do not give these additional factors adequate consideration because multi-objective optimization is an extremely complex problem. If this problem is addressed in a straightforward way, an astronomical number of possible combinations will be generated because of the large number of physical servers and VMs in a cloud computing data center; i.e., a huge number of time-consuming computations are required to obtain an optimum result.

Consequently, timely optimization is a challenge. A VM deployment request from a user should be completed within several minutes (including time to boot up VM). Therefore, a very limited amount of time is available to determine an optimum VM placement.

2 Various Constraints to Virtual Machine Placement

In addition to the physical capacity of each host (i.e., CPU and memory), resource scheduling often has to consider various other constraints. These constraints are related to policies established by the operator of the cloud. Moreover, it is common for the constraints to conflict with each other. Examples of constraint policies are given below:

- Efficient use of electricity (towards a green environment): This policy allocates as many VMs as possible to a host. Electricity can be saved by powering-off idling hosts.
- Availability of a virtual system: This policy distributes VMs to different hosts (redundancy) to guard against VMs going down if a single host fails. This policy may conflict with the goal of saving electricity.
- Affinity: This policy allocates compatible or similar VMs to the same host. For example, network traffic via switches and routers can be reduced by allocating VMs that routinely communicate with each other to the same host. This is a typical situation for multiple VMs owned by a single tenant in a multi-tenant data center.
- Repulsion: VMs that compete for resources should not be placed on the same host. For example, VMs requiring higher network bandwidth should be placed repulsively to avoid network congestion. Firewalls are a typical example.

- Minimum migration cost: This policy determines VMs that should be migrated from one host to another host. Migration costs can be a function of factors memory size, I/O rate, and the number of hops between hosts.

3 Challenges

Realization of Practical Response Time. It is not practical to use a brute-force method to find optimum VM placement because of the large numbers of hosts and VMs in a cloud computing data center. To realize practical computation time, the number of possible combinations must be limited.

Arbitration. As mentioned in the previous section, the various policies established by the cloud computing data center to determine VM placement may conflict with each other; for example, reducing power consumption conflicts with high availability, and conversely, ensuring high availability may increase power consumption. One of the challenges for a cloud administrator is determining how to arbitrate such conflicting constraints. Prioritization could be a solution, but arbitrating the conflict between availability and power consumption is not axiomatic.

Flexibility. Since the requirements and prioritization of policies can differ among data centers, a multi-objective optimization system should allow a data center operator to configure (add and remove) policies to satisfy particular requirements. Moreover, any system for determining VM placement must be sufficiently flexible to respond to changes in economic, political, and social conditions, such as the rising cost of energy, preferential taxation systems for ecological initiatives, regulations requiring the restriction of CO_2 emissions, and other responses to global warming and climate change. For example, data centers in Japan had to respond to restricted electricity supply after the earthquake and tsunami, which occurred in March, 2011.

4 Solutions in the Design

Avoiding Excessive Numbers of Possible Placement Combinations. This is essential for the realization of a practical multi-objective optimization mechanism. We have solved this difficulty by defining and introducing equivalent sets of hosts.

Although a large number of hosts exist in a cloud computing data center, they can be classified into a relatively small number of equivalent sets on the basis of the status of each host. One host from a set can be used to evaluate the cost, thereby significantly reducing the required computations. For example, all hosts can be classified into two equivalent sets (powered on and off) to determine how to apply the electricity-saving policy mentioned in section 2.

Here, we assume that n is the number of hosts and m is the number of VMs to be deployed. When a brute-force method is used, there are n choices for placement of each VM. Therefore, the order of required computation is $O(n^m)$.

We are proposing an algorithm, which will be described in detail in a following section, which utilizes a representative VM from an equivalent set to minimize the number of computations. First, because we need to check the status of each host, for each constraint, the order of computations to classify n hosts into equivalent sets is $O(n)$. It is also $O(n)$ for k constraints. Note that it is reasonable to assume $k << n$. Second, $O(n)$ computations are required to calculate a comprehensive cost for n hosts. Additionally, if all the hosts are neighboring, a maximum of $O(n)$ computations are needed to conduct a neighborhood search. All these computations of $O(n)$ are necessary for m VMs. Therefore, the order of computation is $O(n*m)$.

For example, for 120 hosts and 8 VMs:

$$\begin{cases} O(n^m) = 120\text{^}8\ (\sim 10\text{^}16)\text{: brute-force method} \\ O(n*m) = 120*8\ (\sim 10\text{^}3)\text{: proposed mechanism.} \end{cases}$$

Flexibility and Arbitration. Multi-objective optimization must be flexible and capable of arbitrating conflicting constraints. For example, electricity saving and availability of a virtual system are compatible policies in a multi-tenant environment. Although the latter policy acts to distribute one tenant's VMs on different hosts, the former policy works to cluster several tenants' VMs on one of host. As a result, the number of powered-on hosts can be decreased by sharing hosts among tenants. Considering other optimization objectives, the proposed mechanism programmatically finds an optimum VM placement from a large number of possible combinations.

The structure of the proposed mechanism is illustrated in Figure 1. It consists of following three modules:

1. Cost Evaluation Plug-in Module
   The proposed mechanism has been designed with a plug-in structure to enable data center operators to implement various operational policies. The cost evaluation plug-in module evaluates the cost on the basis of a specific optimization objective function. The exceptional value of this design feature is that is allows a comparison of the impact of various constrains on the basis of the cost.

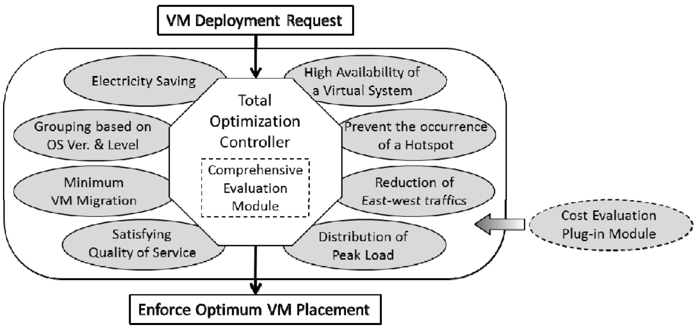

VM Deployment Request
Electricity Saving
High Availability of a Virtual System
Total Optimization Controller
Grouping based on OS Ver. & Level
Prevent the occurrence of a Hotspot
Comprehensive Evaluation Module
Minimum VM Migration
Reduction of East-west traffics
Cost Evaluation Plug-in Module
Satisfying Quality of Service
Distribution of Peak Load
Enforce Optimum VM Placement

Fig. 1. Structure of the proposed mechanism

2. Comprehensive-Evaluation Module
   This module gathers the evaluated cost from all plug-in modules. Subsequently, the comprehensive-evaluation module calculates a comprehensive cost considering the weight rating of each policy. The formula is as follows:

$$C_i = \sum_{j=1}^{k} w_j \cdot e_{i,j}$$

   where C_i is the comprehensive cost for host i, w_j is the weight of the policy j, and $e_{i,j}$ is an evaluated cost for host i by applying policy j.
3. Total Optimization Controller
   The total optimization controller allows the API to accept optimization requests. Interacting with other modules, this controller determines and enforces an optimum VM placement.

An overview of the algorithm is shown in pseudo code in Figure 2. The algorithm works in two phases. In the first phase, a temporal VM placement is determined as an initial state. Subsequently, in the second phase, a neighborhood search is performed.

```
Phase 1: Determine a temporal VM placement
for each VM to be deployed do
  for each Cost Evaluation Plug-in do
    Classify all hosts into equivalent sets based on
    status of each host;
    for each equivalent set of hosts do
      Calculate required additional cost if the VM is
      placed on a host;
    end
  end
  for each host do
    Calculate a comprehensive cost;
  end
  Sort the list of hosts based on comprehensive cost;
  for the head of list to the end of list do
    next unless the host has capacity to host the VM;
    Select the host as the temporal place for the VM;
    break;
  end
end.
```

```
Phase 2: Neighborhood search
for each VM do
  Find neighborhood hosts of the temporal host;
  for each neighborhood host do
    Calculate a comprehensive cost if the VM is placed
    on the host alternative on the temporal host;
  end
  Select the lowest cost host among the temporal and
  neighborhood hosts as an optimum place for the VM;
end.
```

Fig. 2. Overview of the algorithm

5 A Prototype Implementation

Total Optimization Controller / Comprehensive-Evaluation Module. A prototype of the Total Optimization Controller has been developed as a web application. We used Ruby on Rails [6], which is a framework for developing web applications, to realize fast implementation. The controller provides a REST API to accept requests from the Resource Orchestrator, which will be described in greater detail in Section 6. The comprehensive-evaluation module has been implemented as a component module of the total optimization controller.

Cost Evaluation Plug-in Modules. We have implemented plug-in modules for four typical VM placement policies. The plug-in modules are written in Ruby 1.8.

1. Electricity saving
   This plug-in module evaluates the cost of electricity. For simplicity, we focus only on whether a physical server is powered on or off. The actual electricity consumption depends on the load, but there is a significant difference between powered on and off.
   Evaluated cost will be:
   $$\begin{cases} E, & \text{if a VM is placed on a host which needs to be powered on} \\ 0, & \text{otherwise} \end{cases}$$

2. Availability of a virtual system
   This plug-in module takes particular note of redundancy of VMs in the same tier. An example of a commonly used three-tiered web system is shown in Figure 3. In the example, each tier (web, application, and database) has redundant VMs. However, the whole tier will be downed by the failure of a single host if redundant VMs are deployed on the same host. To prevent such a situation, this module considers the loss of redundancy as a cost.
   Evaluated cost will be:
   $$\begin{cases} F, & \text{if a VM is placed with another VM that belongs to the same tier} \\ 0, & \text{otherwise} \end{cases}$$

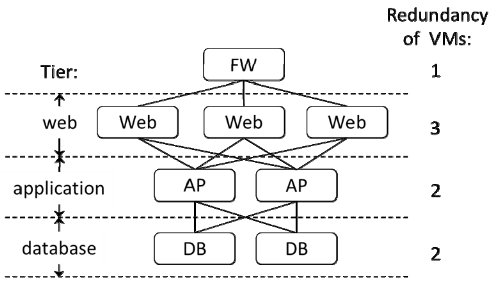


Fig. 3. Example of a 3-tiered web system

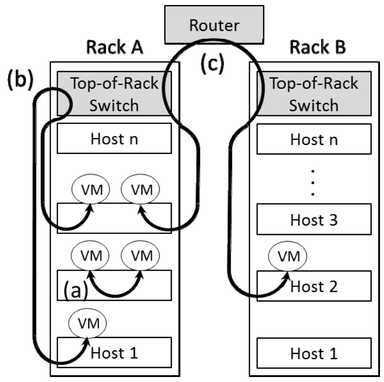


Fig. 4. Example of 3 types of communication situations

3. Affinity (reduction of network traffic beyond a top-of-rack switch)
   This plug-in module evaluates the impact of network traffic. As shown in Figure 4, communication between VMs is classified into three types:

   a. *S* (small): No network traffic is required outside of a host. VMs are placed on the same host.

   b. *M* (medium): Network traffic between hosts via a top-of-rack switch is required when VMs are placed on different hosts exiting in the same rack.

   c. *L* (large): Network traffics with inter-rack routing are required when VMs are placed on the hosts in different racks.

   Additionally, we introduced a *filling rate* to represent rack occupancy. For each rack, the *filling rate* is given as the number of deployed VMs divided by the total capacity (number of possible VMs) of the hosts. Note that the number of VMs is determined by the smallest VM equivalent. When the *filling rate* exceeds the predefined threshold, this plug-in module charges additional cost *A*. Therefore, placement of VMs on racks that are at or close to capacity is discouraged.

4. Repulsion
   We assume that a virtual system contains at least one VM that acts as a firewall. Typically, a firewall requires high bandwidth because all traffic to and from associated networks pass through it. Therefore, placing multiple VM firewalls on the same host is generally undesirable. This plug-in module considers competition for network bandwidth as a cost.
   Evaluated cost will be:
   $$\begin{cases} B^2\text{, if a firewall VM is placed with other firewall VMs} \\ 0\text{, otherwise} \end{cases}$$
   Note: The value of B is proportional to the number of VM firewalls on the host. We use B^2 as an analogy of the charge repulsion.

6 Preliminary Evaluation

We conducted simulations to evaluate the proposed mechanism. Fujitsu ServerView Resource Orchestrator [7] and a hardware simulator were used to construct a mock cloud computing data center environment. The Resource Orchestrator manages all pseudo physical resources (servers, network switches and storage) and VMs. In addition, it manages and allocates addressing resources (MAC addresses, IP addresses, and VLAN IDs). We have made a small modification to the Resource Orchestrator to invoke the proposed mechanism when it receives a request to deploy a virtual system from a user.

Conditions of Simulations

- Physical Servers: 120 homogeneous physical servers; each capable of hosting 20 "economy" VMs (see Table 1).
- Virtual Machines: As shown in Table 1, VMs has been classified into economy, standard, advanced, and high performance. These types were determined by reference to the Amazon EC2 and the Fujitsu Global Cloud Platform.
- Virtual Systems: A virtual system consists of 2-12 VMs that cab be comprised of various types. All VMs in a virtual system will be deployed or deleted synchronously on the basis of a request from a user.
- Duration of a simulation: 1 year.
- Overall CPU utilization: The average overall CPU utilization in a data center starts from 0% (no VM deployed) and grows up to 60% by the end of one year. Note that some deployed virtual systems are deleted during the simulation. In this preliminary evaluation, VM placement that resulted in an over-committed state was not allowed.
- Electricity costs: We assume that a physical server will consume 300 W of electricity (e.g., Fujitsu PRIMERGY RX200S5 equipped with two Intel Xeon 2.53GHz processors and 24GB memory). Based on the cost of the special high-tension voltage power in Tokyo, Japan, the cost is approximately 3.4 yen per hour.

Table 1. Types of VMs

Type	CPU	Memory (GB)
Economy	1	1.7
Standard	2	3.4
Advanced	4	7.5
High Performance	8	15.0

Note: "CPU=1" is equivalent CPU performance of Intel Xeon 1.0 GHz.

Simulation Results. Configurations of the weighted values for the applied plug-in modules are summarized in Table 2. "A" represents the lowest boundary of electricity cost. "F" distributes VMs considering both availability and repulsion (i.e., no electricity saving). The weight of availability and repulsion are gradually increased from "C" to "E."

Three patterns of request sequences, p1, p2, and p3, were assessed. Under conditions described in above, each virtual system was given randomly generated parameters: type and number of VMs, date and time of deployment, and deletion.

Figure 5 shows the average calculation time to find an optimum VM placement by the proposed mechanism and a brute-force method (simulated on a PC equipped with Intel Core i5-2520M 2.50GHz CPU and 4GB memory). The x-axis represents the number of VMs included in a virtual system. We can see that the proposed mechanism realized a practical calculation time even the number of VMs was increased. For example, for 8 VMs, the average calculation time by the proposed mechanism and a brute-force method were 0.097sec and 160sec, respectively.

Figure 6 shows the accumulated costs for one year of simulated operation. Figure 6(a) indicates the real cost of electricity. Figures 6(b) and (c) represent the assigned costs of availability and repulsion, respectively. As mentioned in Section 5, assigned costs are charged when constraints are not satisfied; i.e., smaller value is preferable.

Table 2. Configurations of weight for the applied plug-in modules

config.	electricity saving	availability	repulsion	affinity
A	1.0	-	-	-
B	1.0	2.0	-	-
C	1.0	0.5	0.5	1.0
D	1.0	1.0	1.0	1.0
E	1.0	2.0	2.0	1.0
F	-	1.0	1.0	-

(- : not applied)

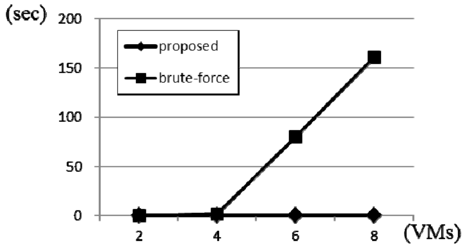
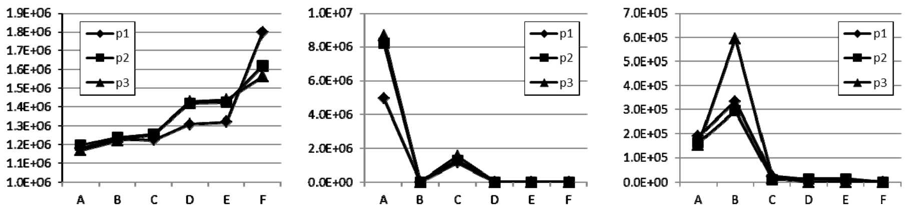


Fig. 5. Average calculation time to find an optimum VM placement



(a) electricity (b) availability (c) repulsion

Fig. 6. Accumulated costs (duration: 1 year)

The simulation results show that the electricity cost increases as other constraints are satisfied. In this simulation, however, both availability and repulsion were well satisfied for configurations "D" and "E." Compared with "F," 8.4% to 27.3% of electricity cost was saved.

7 Related Work

Ni et al. [8] implemented a probabilistic scheme to determine VM placement. A roulette wheel is used in their scheme. A sector on the roulette wheel corresponds to a host. To create larger selection probability, a larger central angle was assigned to a sector associated with a host that has larger amount of available resources. In the proposed VM mapping policy, multi-dimensional resource usage (e.g., CPU and memory) was considered. However, other constraints, such as electricity saving and high availability of a virtual system, were not considered.

Xu et al. [9] and Garces et al. [10] implemented multi-objective optimization mechanisms for VM placement and migration. Their common approach applies a genetic algorithm to solve a multi-objective optimization problem. Our approach does not use a genetic algorithm. As mentioned in Section 4, we introduced equivalent sets of hosts to avoid extremely large numbers of possible placement combinations.

Tsakalozos et al. [11] proposed an approach similar to ours; i.e., identifying potentially compatible groups of physical servers to reduce the search space. Moreover, a few constraints such as power saving and minimizing network traffic by co-deploying a set of VMs on the same physical server were considered. In this research, however, physical servers were classified into groups based solely on VM migration ability because the goal was load balancing through migration. In contrast, our mechanism generates equivalent sets of physical servers (hosts) for each constraint or policy. Note that the VM migration ability of a host can be added as a constraint by implementing a plug-in module. Subsequently, our mechanism places a VM on a host that was evaluated as having the lowest comprehensive cost because our goal is the reduction of operational cost and not load balancing.

8 Conclusion

We have designed and implemented a multi-objective optimization mechanism for VM placement. The proposed mechanism is flexible and allows data center operators to add their own desired optimization objectives or evaluate specific policies by implementing plug-in modules.

The unique value of our system is that a constraint is translated into an estimated cost (real or assigned). Each plug-in module evaluates the additional cost that would accrue if a VM is placed on a host. Subsequently, the Comprehensive-Evaluation Module gathers the results and calculates the total cost considering the weight of each policy. The Total optimization controller conducts a neighborhood search to find the lowest cost VM placement, and ultimately, it enforces the optimum VM placement.

A practical calculation time to find an optimum VM placement was realized by introducing the equivalent sets of hosts. The simulation results showed that both availability and repulsion could be satisfied with a cost saving of 8.4%-27.3% for electricity.

Acknowledgments. We would especially like to express our heart-felt gratitude to Messrs. Toru Sakiyama, Noriyuki Murata, and Makoto Nasuno who are systems software engineers at ASCADE Inc.

References

1. Amazon Elastic Compute Cloud (EC2), `http://aws.amazon.com/ec2/`
2. Amazon Simple Storage Service (S3), `http://aws.amazon.com/s3/`
3. Google App Engine, `http://code.google.com/appengine/`
4. Saleseforce CRM, `http://www.salesforce.com/crm/`
5. Fujitsu Global Cloud Platform, `http://www.fujitsu.com/global/solutions/cloud/solutions/global-cloud-platform/`
6. Ruby on Rails, `http://rubyonrails.org/`
7. Fujitsu ServerView Resource Orchestrator Cloud Edition, `http://www.fujitsu.com/fts/products/computing/servers/primergy/management/dynamize/ror-ce/ror-ce.html`
8. Ni, J., Huang, Y., Luan, Z., Zhang, J., Qian, D.: Virtual Machine Mapping Policy Based on Load Balancing in Private Cloud Environment. In: Proc. of 2011 Int'l Conf. on Cloud and Service Computing, Hong Kong, China (2011)
9. Xu, J., Fortes, J.A.B.: Multi-objective Virtual Machine Placement in Virtualized Data Center Environments. In: Proc. of 2010 IEEE/ACM Int'l Conf. on Green Computing and Communications, Hangzhou, China (2010)
10. Garces, N., Ortiz, N., Mendez, D., Donoso, Y.: Multi-Objective Optimization for Virtual Machine Migration on LANs for Opportunistic Grid Infrastructure. In: Proc. of the 3rd Int'l Conf. on Emerging Network Intelligence, Lisbon, Portugal (2011)
11. Tsakalozos, K., Roussopoulos, M., Delis, A.: VM Placement in non-Homogeneous IaaS-Clouds. In: Proc. of the 9th Int'l Conf. on Service Oriented Computing, Paphos, Cyprus (2012)

Agent Based Application Tools for Cloud Provisioning and Management

Luca Tasquier, Salvatore Venticinque, Rocco Aversa, and Beniamino Di Martino

Department of Information Engineering, Second University of Naples, Aversa, Italy
luca.tasquier@gmail.com, {salvatore.venticinque,rocco.aversa}@unina2.it, beniamino.dimartino@unina.it

Abstract. When service providers move to IAAS Clouds, whether their service is delivered by a legacy application, either it has been developed by a deployable open platform, the provisioning and the management workflow of the computing infrastructure really changes. Here we describe a set of tools which can be used to orchestrate agents' based services, which provide facilities for provisioning, management and monitoring of Clouds. The user is able to discover, allocate, configure and monitor Cloud services at infrastructure level through an approach that is agnostic respect to the specific Cloud vendor or to the Cloud technology.

1 Introduction

When service providers move to IAAS Clouds, whether their service is delivered by a legacy application, either it has been developed by a deployable open platform, the provisioning and the management workflow of the computing infrastructure really changes. First of all the Cloud elasticity supports the reconfiguration of the computing resources when application requirements change dynamically and the *pay-per-use* business model allows for the possibility to change the Resource Providers when a more convenient offer is found. Of course a number of issues arise in the current Cloud scenario due to the lack of interoperability among different technological Cloud solutions and because of the limited portability of Cloud applications. However, even when the service developer is able to overcome these difficulties, by making technical choices that are independent respect to the Cloud provider, it is not easy to discover and retrieve the available Cloud proposals, to check if they can accomplish the service requirements, and also to compare each other. Currently there is not a common ontology for describing service terms and service levels, neither in a formal way nor through natural language. Other issues regard the management of the acquired resources. Also in this case the lack of a wide adopted standard for service at Cloud infrastructure level (IAAS) affects the chance of opting for a different commercial or technological solution. In fact the use will have to change both management tool and methodology. Finally, the last motivation we are addressing deals with monitoring of resource utilization. This problem has been extensively investigated with the perspective of the resource provider, which aims at optimizing the utilization of its physical resources in order to improve its own service level and to increase its profit. However monitoring needs to be addressed with a different perspective in the case of a service provider that stocks computing resources through the Cloud market. Cloud customers cannot check the compliance of the Service Level Agreement (SLA) trusting the monitoring service of the same provider, who has a conflicting interest ensuring the guarantees on service levels it provides. Besides Cloud customers needs to detect under-utilization and overload conditions. In both the cases it is necessary to dimension the Cloud resource to avoid useless expenses and to not fail to satisfy the service requirements when workloads change dynamically. In this paper we present a set of tools which allows the user for orchestrating agents based services, which support discovery, brokering, management and monitoring of Cloud resources. We describe how these services can be used to execute a workflow for Cloud governance that allows for vendor agnostic provisioning, deployment, management and monitoring Cloud services at Infrastructure level. Related work is presented in Section 2. In Section 3 we introduce the available services and a workflow for Cloud governance. Section 5 describes application tools and how they can be used. Finally conclusions are due.

2 Related Work

The design and development of solutions for governance of multiple heterogeneous cloud is an issue addressed both in research activities and commercial domains [4]. Here we briefly provide an overview about related work and the technological assessment for Cloud provisioning, management and monitoring. The brokering of Cloud providers, whose offers can meet the requirements of a particular application, is a complex issue due to the different business models that are associated with such computing systems. The current Cloud computing technologies offer a limited support for dynamic negotiation of SLAs among participants. The work presented in [10] represents a first proposal to combine SLA-based resource negotiations with virtualized resources in terms of on-demand service provision. The architecture description focuses on three topics: agreement negotiation, service brokering and deployment using virtualization. It involves multi- ple brokers. A Cloud multi-agent management architecture is proposed in [5]. A simpler agents based architecture has been proposed in [13]. Preliminary investigations by the authors on related topics have been presented in [12]. SLA@SOI is the main project that aims (together with other relevant goals) at offering an open source based SLA management framework. It will provide benefits of predictability, transparency and automation in an arbitrary service-oriented infrastructure. About management of heterogeneous Clouds, interoperability is the main issue. By the research community there are many standardization efforts. Some examples are OCCI (Open Cloud Computing Interface), by the Open Grid Forum, and SOCCI (Service-Oriented Cloud-Computing Infrastructure) by the ISO Study Group on Cloud Computing (SGCC). In particular OCCI (Open Cloud Computing Interface) is a proposal of standard for

IAAS Cloud. It defines entities, relationships API and protocols for all kinds of management tasks. This solutions is aimed at the fulfillment of three requirements: integration, portability and interoperability for common tasks including deployment, autonomic scaling and monitoring still offering an high degree of extensibility. The OpenNebula solution already implements a RESTFull OCCI compliant interface, and other technologies like Eucalyptus and Openstack are working to be themselves compliant with it. Commercial providers are going to support themselves management facilities which enable the integration of third party clouds. For example Amazon can use its elasticity capability also exploiting computing resources shared by OpenNebula. Rather than adopting a standard or developing a new service interface, some efforts have been spent to develop technologies for integration of existing IAAS Clouds. DeltaCloud and JClouds are two different solutions. The first one provides a service with an uniform interface, but uses different drivers to redirect requests to the supported heterogeneous commercial Cloud providers and to private Clouds developed by open technologies. JCLouds, instead, offers a uniform and extensible API to develop applications that can directly interoperate with multiple IAAS Clouds. Another example of free and open source technology that aims at supporting the integrated management of heterogeneous Cloud is provided by My Cloud Portal[1]. It allows for setting up and managing of hybrid cloud, private and public, by the integration of Eucalyptus and Azure Cloud infrastructures. It provides a web interface by which it is possible to define workflows, perform monitoring activities and reconfigure settings. Infrastructure- level resource monitoring [6] [3] aims at the measurement and reporting of system parameters related to real or virtual infrastructure services offered to the user (e.g. CPU, RAM, or data storage parameters). Traditional monitoring technologies for single machines or Clusters are restricted to locality and homogeneity of monitored objects and, therefore, cannot be applied in the Cloud in an appropriate manner [8]. At the state of the art there are many tools which provide Cloud monitoring facilities, like Cloudkick, Nimsoft Monitor[2], Monitis[3], Opnet, RevealCloud. All of them are proprietary solutions and do not aim at defining a standard for monitoring interoperability. Some technologies for monitoring network and host like sFlow have been extended in order to support the transport of monitoring information of virtualized resources. For example host-sflow [1] exports physical and virtual server performance metrics by using the sFlow [2] protocol. Ganglia and other collectors of performance measures are already compliant with its specification. In [9] authors claims that an approach based on software agents is a natural way to tackle the monitoring tasks in the aforementioned distributed environments. Agents move and distribute themselves to perform their monitoring tasks. In [11] an optimal control of mobile monitoring agents in artificial-immune-system-based (AIS-based) monitoring networks has been studied.

[1] `http://www.mycloudportal.in/`
[2] `http://www.nimsoft.com/solutions/nimsoft-monitor`
[3] `http://portal.monitis.com`

3 Deployment and Execution Using the IAAS Cloud

As shown in Figure 1 the life cycle of a Cloud application that is running by using a Cloud infrastructure is divided into three phases:

1. *Cloud Provisioning.* The user has to choose the best Cloud resources for his/her application (e.g. VMs, storages, etc.). After that he/she has to select the best IaaS provider basing his/her thinking on a lot of parameters (cost per use, amount of VM memory, storage's size, bandwidth, etc.). Many times this reasoning is too difficult because each provider offers its resources highlighting different characteristics and parameters. This happens because the Cloud vendors haven not a common and standardized interface to describe the resource parameters, making the comparison among same resources an hard job.
2. *Cloud Configuration.* After selecting the best resources for his/her application, the Cloud customer needs to sign an SLA with a Cloud vendor. Once this has been done some management activities are carried before deploying applications. For instance OS images have to be attached and the purchased VMs have to be started. For this reason, the cloud user/developer has to know the allowed actions for that resource and the service interface for that Cloud provider. In fact the same resource purchased from a different provider have different interface and different supported functionalities. At this point the Cloud application can be deployed and executed.
3. *Cloud Monitoring.* Here Cloud users configure a network of probes that collects measures about the performance parameters of the Cloud resources. To get an up to date knowledge of Cloud performance and an history of the Cloud behavior it needs to periodically compute a set of performance indexes and to set up some triggers which notify critical conditions. In fact it would be useful to know if the workload of the infrastructure is different from the one foreseen, in order to avoid saturation or under-utilization of Cloud resources. This information is necessary to design effective reconfigurations of the infrastructure, in order to better adapt it to the current application requirements and to optimize performances and costs.

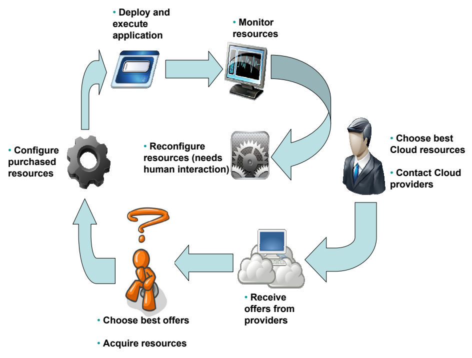

• Deploy and execute application
• Monitor resources
• Choose best Cloud resources
• Contact Cloud providers
• Reconfigure resources (needs human interaction)
• Configure purchased resources
• Receive offers from providers
• Choose best offers
• Acquire resources

Fig. 1. Deployment and execution workflow by using IAAS Cloud

4 The mOSAIC Solution: Cloud Agency

The EC-FP7-ICT mOSAIC project [7] intends to improve the state-of-the-art in Cloud computing by creating, promoting and exploiting an open-source Cloud application programming interface and a platform targeted for developing multi-Cloud oriented applications. Cloud Agency represents the mOSAIC solution for provisioning and management of Cloud resources that allows for the deployment and execution of the mOSAIC platform and applications.

Cloud Agency [12] is a multi agent system that accesses, on behalf of the user, the utility market of Cloud computing to maintain always the best configuration of resources, to satisfy the application requirements. It supplements the common management functionalities, which are currently provided by Cloud infrastructures, with new advanced services, by implementing transparent layer to IAAS Private and Public Cloud services.

Agents can be orchestrated by invoking those services, which are offered by Cloud Agency through an OCCI compliant RESTFull interface, according to the workflow defined in Section 3.

One of the leading agents that composes Cloud Agency is the *Broker Agent* that receives the list of those resources that the mOSAIC application needs for its deployment and execution, asks to providers for available offers, brokers the best one and allows for closing the transaction. *Vendor Agents* implement a wrapper for a specific Cloud: they are used to get an offer for resource provisioning, to accept or refuse that offer, to get informations about a resource or to perform an action on it (start, stop, resume). About Cloud monitoring, *Meter Agents* perform measures of performance indexes at IAAS level and return their values to the *Archiver Agent.* The Archiver collects the measures and maintains statistics about the mOSAIC system. For the reconfiguration service the *Tier Agent* has been developed: it is triggered by the Archiver and uses policies defined by the user to apply the necessary reactions. The *Mediator Agent* receives requests directly from the user: it is in charge of starting new transactions for provisioning by creating Broker agents; it also starts new Tiers and Meters, and returns information about Cloud Agency configuration and services.

Being implemented as a MAS (Multi Agent System), the internal work-flow of the agency is intrinsically asynchronous because agents react on the occurrence of an incoming message. In the same way RESTFull APIs of Cloud Agency have been conceived for an event-driven mOSAIC programming model. Asynchronous Service Requests (ASR) are used to ask Cloud Agency for something to be executed. For example to start a negotiation, to accept or to refuse the SLA, to start a VM, etc. Requests, as any other action, are not-blocking. It means that execution is started remotely, but the client can continue to run. On the other hand, requests will generate future events, which have to be handled by the requester. By the same way, asynchronous events from Cloud Agency are notified to the Cloud user. Synchronous Service Requests (SSR) are used to get informations. For example clients can ask for reading an SLA or the status of a negotiation, to get the list of vendors or the list of the resources. Queries are synchronous, they return immediately the response if it is available, and an exception otherwise.

5 Application Tools

A set of tools have been designed and developed to support the orchestration of Cloud Agency services by different type of Cloud users. The orchestrator can be actually personified by both a human user and by an autonomic application. In the first case the user takes the hat of a Cloud administrator, in the second case it is a developer that design the application. The list of tools includes some APIs for development of legacy applications and mOSAIC Cloud applications. For sake of space we skip the description of mOSAIC APIs development here. Instead we present two implementations of a Cloud Agency console and a Monitoring tool.

5.1 Cloud Agency Client API

A set of JAVA APIs support the development of event based client application of Cloud Agency. According to the event-driven interaction model the API allows for sending asynchronous and synchronous requests and for handling asynchronous notifications by implementing callbacks. The class diagram of a typical client application is shown in Fig. 2.

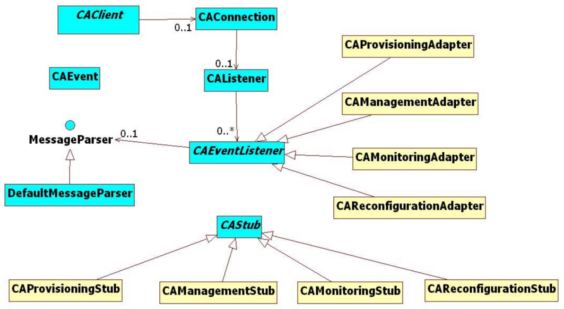


Fig. 2. Class diagram of Cloud Agency Client API

The blue colored classes represent the *core* API. The main goals of each class are described in Table 1.

In addition to the core API, several implementations of core abstract classes are provided. In particular:

– *stubs*: a stub offers a set of methods to invoke the Cloud Agency services. These methods wrap the REST requests in order to query the Cloud Agency RESTFull interface and to perform actions on Cloud resources or to obtain information about the state of the Cloud infrastructure. At the state-of-the-art of the development API client there are four classes that implement the CAStub class and that provide methods to invoke provisioning, management, monitoring and reconfiguration services;

Table 1. Core classes of Cloud Agency Client API

Class	**Description**
CAClient	This abstract class represents an instance of the Cloud Agency Client. Who wants to develop a new client for the Cloud Agency has to extend it
CAConnection	This class represents the established connection between the client and the Cloud Agency. It is in charge of authenticating the user against the Cloud Agency and of initializing instances of stubs and adapters
CAEventListener	This abstract class implements a generic event listener. It handles the generic asynchronous messages coming from the Cloud Agency
CAEvent	This class abstracts an asynchronous event received from a listener
MessageParser	This interface represents the message parser used by an event listener in order to process an asynchronous message coming from the Cloud Agency and to generate a CAEvent
DefaultMessageParser	This is the default implementation of a MessageParser
CAStub	This abstract class implements the generic stub

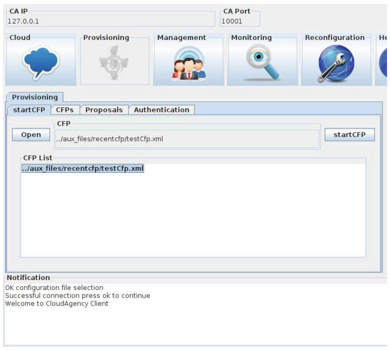
– *adapters*: an adapter is an handler for the asynchronous messages coming from the Cloud Agency, according to its event-driven architecture. When a new message arrives, it is forwarded to the adapter that implements the particular service listener. Each user can customize the client behavior on the arrival of particular events by simply extending the *CAEventListener* class or overriding an existing adapter. So he/she can implement autonomic reactions by adding his/her own new adapter by the core API. At the state-of-the-art of the development of the API client there are four classes that implement the *CAEventListener* class and that handle (without performing any action outside of the displaying the received message) provisioning, management, monitoring and reconfiguration events.

These APIs are also used to implement a command line and a graphical user interface of a Cloud Agency console.

5.2 Command Line and Graphical User Interface

Cloud Agency **Command Line Interface (CA-CLI)** offers a variety of commands to invoke the Cloud Agency's services. It follows the user in all the phases of the deployment and of the execution of his/her own application by giving him/her the possibility to book and manage the Cloud resources in a flexible and simple way. The CA-CLI helps the mOSAIC developer in the deployment process, beginning from the brokering of the best resources for his/her application. The application opens a console that starts a listener to handle the notifications sent by Cloud Agency and allows for the execution of a number of commands. The management is vendor agnostic, in the sense that the user asks for performing a specific operation (start, stop, restart, etc.) on any given resource.

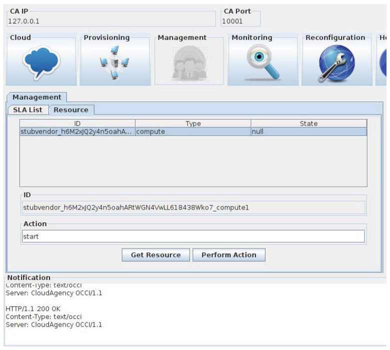
Cloud Agency **Graphical User Interface (CA-GUI)** is another tool that helps the user during the deployment and execution of his/her mOSAIC application. The functionalities offered by the CA-GUI are basically the same of the CA-CLI ones. Of course, the graphical interface is more powerful than the command line to take under control all the stuff during the provisioning phase: the editing and the listing of the *Call For Proposals (CFPs)*, the providers' proposals or SLAs, the acquired resources and their state. It also provides some additional functionalities in order to simplify the Cloud management, such as the listing of the available vendors, the start/stop of an available VM, the attach and detach of an available storage and so on.



CA IP: 127.0.0.1
CA Port: 10001
CFP: ../aux_files/recentcfp/testCfp.xml
CFP List: ../aux_files/recentcfp/testCfp.xml

```
Notification
OK configuration file selection
Successful connection press ok to continue
Welcome to CloudAgency Client
```

(a) GUI tab for provisioning



CA IP: 127.0.0.1
CA Port: 10001

ID	Type	State
stubvendor_h6M2xJQ2y4n5oahA...	compute	null

ID: stubvendor_h6M2xJQ2y4n5oahARtWGN4VwLL618438Wko7_compute1
Action: start

```
Notification
Content-Type: text/occi
Server: CloudAgency OCCI/1.1

HTTP/1.1 200 OK
Content-Type: text/occi
Server: CloudAgency OCCI/1.1
```

(b) GUI tab for management

Fig. 3. Graphical User Interface for Cloud Agency Client

In Fig. 3 (a) it is shown how the CA-GUI appears. On top of the interface there is the location of the connected Cloud Agency instance. Just below this one there are some buttons, each one representing a Cloud Agency service and allowing the user for easily performing provisioning, management, monitoring and reconfiguration operations. Moreover, the *Cloud* button provides several functionalities to manage the CFPs and to get informations about Cloud Agency status. By clicking a button, a new panel appears, which is customized by the particular operations that the selected service allows. The CA-GUI handle both ASR and SSR in order to get the requested informations and/or to perform an action. On the bottom of the window the raw notification messages are displayed.

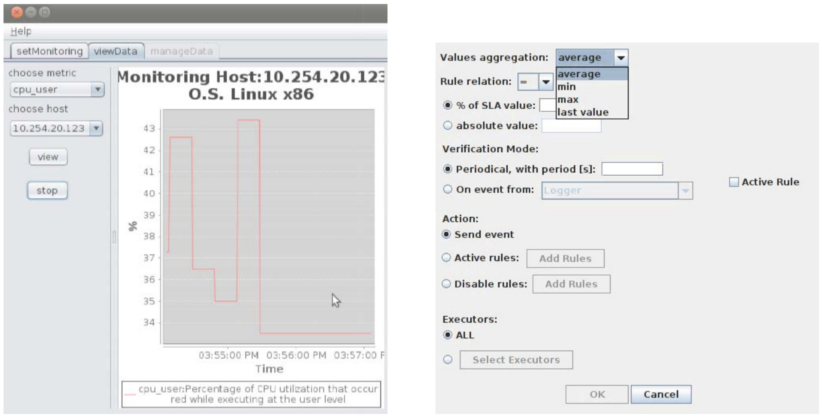
The management console, shown in Fig. 3 (b) allows the user for starting/stopping VMs, for loading and attaching VM images, for deploying and executing applications and so on.

The monitoring console allows for the configuration of the monitoring infrastructure on the acquired resources. For each resource is possible to select a set of measures, each one supported by a specific by Meter Agent. When a set of measures has been selected and the monitoring configuration is finished, the Cloud Agency creates a new Meter Agent sends it to the target resource. At this point the Meter Agent get the measures and sends them to the Archiver Agent, that stores them and is able to compute metrics on performace information on user's demand. Currently is possible to choose Host sFlow based measures. *Host sFlow* [1] agent measures and communicates physical and virtual server performance parameters using the *sFlow* [2] protocol. The agent provides scalable, multi-vendor, multi-OS performance monitoring with minimal impact on the systems being monitored.

5.3 Cloud Performance Monitor

After that the application has been deployed. As regards the mOSAIC developer, he/she can take under control the infrastructure performances by using the monitoring tool that can be started by the CA-GUI. It allows for the visualization of a list of available measures, as it is shown in Fig. 4 (a), or for setting up the computation of some metrics about performance indexes. When a new metric is created, the developer can read synchronously the last value of that index or can create a trigger to be notified asynchronously according to a specific time period or when a critical condition is verified as it is shown in Fig. 4 (b). An example of available metric is the average value of a measure that is periodically computed and that is notified when it is out of a certain range.

When a trigger is activated by the verification of a critical condition on a resource's parameter, the user can decide to be notified about the verified event or to activate/deactivate other rules previously defined and related to other


Help
setMonitoring
viewData
manageData
view
stop
Monitoring Host:10.254.20.123
O.S. Linux x86
43
42
41
40
39
38
37
36
35
34
%
03:55:00 PM
03:56:00 PM
03:57:00 P
Time
cpu_user:Percentage of CPU utilization that occurred while executing at the user level

choose metric: cpu_user
choose host: 10.254.20.123

Values aggregation: average (average, min, max, last value)
Rule relation: =
[✓ % of SLA value:]
absolute value:
Verification Mode:
[✓ Periodical, with period [s]:]
On event from: Logger
Active Rule
Action:
[✓ Send event]
Active rules: Add Rules
Disable rules: Add Rules
Executors:
[✓ ALL]
Select Executors
OK Cancel

(a) Visualization of performance's indexes

(b) Creating a trigger on a resource's parameter

Fig. 4. Monitoring Tool for Cloud Agency

resource's parameters and/or other resources. So the developer can set up a complex trigger by composing some simple ones.

6 Conclusion

A set of tools for provisioning and management of Cloud resources can leverage the burden of a user that wants to deploy and execute his/her application by using services at infrastructure level. In the framework of the FP7 mOSAIC project we have designed and developed agents based services for provisioning, management and monitoring of Cloud infrastructure. We presented here an API client and some application tools that allow for the orchestration of such agents based services according to a workflow to be adopted for the governance of Cloud resources. The human intervention is still required for taking necessary decisions about if and how reconfigure the infrastructure by using the Cloud elasticity or looking for new proposals, eventually from different providers. Future work will focus on the design and developing of autonomic applications to be automatically triggered when critical conditions have been detected in order to provide an autonomous and configurable closed-loop implementation of the proposed workflow.

Acknowledgements. This research is partially supported by FP7-ICT-2009-5-256910 (mOSAIC) and by Cloud@Home PRIN project by MIUR.

References

1. Host sflow, `http://host-sflow.sourceforge.net/`
2. sflow, `http://www.sflow.org`
3. Aversa, R., Di Martino, B., Venticinque, S.: Distributed agents network for ubiquitous monitoring and services exploitation, pp. 197–204 (2009) Cited By (since 1996)
4. Aversa, R., Di Martino, B., Venticinque, S.: Integration of mobile agents technology and globus for assisted design and automated development of grid services, pp. 118–125 (2009); Cited By (since 1996)
5. Cao, B.Q., Li, B., Xia, Q.M.: A service-oriented qos-assured and multi-agent cloud computing architecture. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) Cloud Computing. LNCS, vol. 5931, pp. 644–649. Springer, Heidelberg (2009)
6. Clayman, S., Galis, A., Chapman, C., Toffetti, G.: Monitoring service clouds in the future internet. Framework, 115–126 (2010), `http://www.future-internet.eu/fileadmin/documents/valencia_documents/plenary/Monitoring_service_clouds_in_the_Future_Internet.pdf`
7. Di Martino, B., Petcu, D., Cossu, R., Goncalves, P., Máhr, T., Loichate, M.: Building a mosaic of clouds. In: Guarracino, M.R., Vivien, F., Träff, J.L., Cannatoro, M., Danelutto, M., Hast, A., Perla, F., Knüpfer, A., Di Martino, B., Alexander, M. (eds.) Euro-Par-Workshop 2010. LNCS, vol. 6586, pp. 571–578. Springer, Heidelberg (2011)

8. Emeakaroha, V.C., Brandic, I., Maurer, M., Dustdar, S.: Low level metrics to high level slas - lom2his framework: Bridging the gap between monitored metrics and sla parameters in cloud environments. In: Smari, W.W., McIntire, J.P. (eds.) HPCS, pp. 48–54. IEEE (2010), `http://dblp.uni-trier.de/db/conf/ieeehpcs/ieeehpcs2010.html#EmeakarohaBMD10`
9. Ilarri, S., Mena, E., Illarramendi, A.: Using cooperative mobile agents to monitor distributed and dynamic environments. Inf. Sci. 178(9), 2105–2127 (2008)
10. Kertesz, A., Kecskemeti, G., Brandic, I.: An sla-based resource virtualization approach for on-demand service provision. In: Proceedings of the 3rd International Workshop on Virtualization Technologies in Distributed Computing, VTDC 2009, pp. 27–34. ACM, New York (2009)
11. Liu, W., Chen, B.: Optimal control of mobile monitoring agents in immune-inspired wireless monitoring networks. Journal of Network and Computer Applications 34(6), 1818–1826 (2011), doi:10.1016/j.jnca.2010.12.004
12. Venticinque, S.: Agent Based Services for Negotiation, Monitoring and Reconfiguration of Cloud Resources, pp. 178–202 (2012)
13. You, X., Wan, J., Xu, X., Jiang, C., Zhang, W., Zhang, J.: Aras-m: Automatic resource allocation strategy based on market mechanism in cloud computing. Journal of Computers 6(7) (2011), `http://ojs.academypublisher.com/index.php/jcp/article/view/jcp060712871296`

Performance Evaluation of Embedded Processor in MapReduce Cloud Computing Applications

Christoforos Kachris[1,*], Georgios Sirakoulis[1], and Dimitrios Soudris[2]

[1] Department of Electrical and Computer Engineering,
Democritus University of Thrace, Xanthi, Greece
{ckachris,gsirak}@ee.duth.gr
[2] Department of Electrical and Computer Engineering,
National Technical University of Athens, Athens, Greece
dsoudris@microlab.ntua.gr

Abstract. Current data centers consume huge amount of power to face the increasing network traffic. Therefore energy efficient processors are required that can process the cloud applications efficiently without consuming excessive power. This paper presents a performance evaluation of the processors that are mainly used in high performance embedded systems in the domain of cloud computing. Several representative applications based on the widely used MapReduce framework are mapped in the embedded processor and are evaluated in terms of performance and energy efficiency. The results shows that high performance embedded processors can achieve up to 7.8x better energy efficiency than the current general purpose processors in typical MapReduce applications.

Keywords: cloud computing, green computing, embedded processors, mapreduce, data centers.

1 Introduction

Over the last few years, the exponential increase of the Internet traffic, mainly driven from emerging applications like streaming video, social networking and cloud computing has created the need for more powerful warehouse data centers. These data centers are based on thousands of high performance servers interconnected with high performance switches. Most of the applications that are hosted in the data center servers (e.g. cloud computing applications, search engines, etc.) are extremely data-intensive and require high interaction between the servers in the data center.

A main concern in the design and deployment of a data centers is the power consumption. Many data consume a tremendous amount of electricity; some consume the equivalent of nearly 180,000 homes [1]. Greenpeace's Make IT Green report [2],

[*] The research project is implemented within the framework of the Action «Supporting Postdoctoral Researchers» of the Operational Program "Education and Lifelong Learning" (Action's Beneficiary: General Secretariat for Research and Technology), and is co-financed by the European Social Fund (ESF) and the Greek State.

estimates that the global demand for electricity from data centers was around 330bn kWh in 2007 (almost the same amount of electricity consumed by UK [3]). This demand in power consumption demand is projected to more than triple by 2020 (more than 1000bn kWh). According to some estimates [3],[4], the power consumption of the data centers in the US in 2006 was 1.5% of the total energy consumed at a cost of more than $4.5B.

	Derive electricity consumption	**Forecast electricity consumption**
	Billion kWh 2007	**Billion kWh 2020**
Data Centers	330	1012
Telecoms	293	951
Total Cloud	**623**	**1963**

Fig. 1. Power consumption of data centers, Source: Greenpeace, [2]

The power consumption inside the data center is distributed in the following way: the servers consume around 40% of the total IT power, storage up to 37% and the network devices consume around 23% of the total IT power [5]. And as the total power consumption of IT devices in the data centers continues to increase rapidly, so does the power consumption of the HVAC equipment (Heating-Ventilation and Air-Conditioning) to keep steady the temperature of the data center site. Therefore, the reduction in the power consumption of the network devices has a significant impact on the overall power consumption of the data center site. According to a study from Berk-Tek, saving 1W from the IT equipment results in cumulative saving of about 2.84W in total power consumption due to the reduced power consumption of the cooling systems [6]. Therefore, a reduction on the power consumption of the interconnection network will have a major impact on the overall power consumption of the data center.

The power consumption of the data centers has also a major impact on the environment. In 2007, data centers accounted for 14% of the total ICT greenhouse gases (GHG) emissions (ICT sector is responsible for 2% of global GHG emissions), and it is expected to grow up to 18% by 2020 [2]. The global data center footprint in greenhouse gases emissions was 116 Metric Tonne Carbon Dioxide (MtCO2e) in 2007 and this is expected to more than double by 2020 to 257 MtCO2e, making it the fastest-growing contributor to the ICT sector's carbon footprint.

Therefore, more energy efficient servers are required for the emerging cloud computing applications. In [7],[8], a performance evaluation study has been presented between high performance server cores (e.g. Intel Xeon processors) with low power general purpose cores (e.g. Intel Atom processors). The comparison has shown that low power general purpose cores can achieve better energy efficiency in the domain of web search applications. One of the first companies that adopted the used of low power general purpose processors was SeaMicro [9]. SeaMicro introduced in 2011 a new version of servers that packed 768 Intel Atom cores into a 10U chassis. According to the company the Atom-based data center could achieve ¼ the power and 1/6 the space of the commodity volume servers. Another company, Calxeda Inc. has recently presented a server based on the ARM cores called Server-on-a-Chip (SoC).

According to the company the EnergyCore ECX-1000 is the most energy-efficient processor for data centers [10]. However, until now there is not any qualitative comparison between the embedded processors and the general purpose processors in the domain of cloud computing applications.

In this paper we present a performance evaluation between the general purpose processors and the embedded processors (that are inherently designed for low power applications) for cloud computing applications that are based on the MapReduce framework. Section 2 presents the most common cloud computing applications under the MapReduce framework. Section 3 presents the architectural details of the processors and shows the comparison of these processors in terms of performance and energy consumption. Finally, Section 4 presents the conclusions of this paper.

2 Cloud Computing Applications

One of the most widely used frameworks that are hosted in the data centers is the MapReduce framework. **MapReduce** is a programming model and an associated implementation for processing and generating large data sets [11]. Users specify a map function that processes a *key/value* pair to generate a set of intermediate *key/value* pairs, and a reduce function that merges all intermediate values associated with the same intermediate key.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

The **Map function** takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key *I* and passes them to the Reduce function.

The **Reduce function** accepts an intermediate key *I* and a set of values for that key. It merges together these values to form a possibly smaller set of values. The intermediate values are supplied to the user's reduce function via an iterator. This allows us to handle lists of values that are too large to fit in memory.

2.1 The Phoenix MapReduce Framework

Phoenix is a programming API and runtime system based on Google's MapReduce model developed by Stanford University [12],[13]. The main implementation of the Phoenix framework is based on the same notions of the MapReduce framework as it is depicted in Figure 2. The **Map** function processes the input data and generates a set of intermediate key/value pairs. The **Reduce** function properly merges the intermediate pairs which have the same key. Given such a functional specification, the MapReduce runtime automatically parallelizes the computation by running multiple **map** and/or **reduce** tasks in parallel over disjoined portions of the input or intermediate data. Google's MapReduce implementation facilitates processing of terabytes on clusters with thousands of nodes. The Phoenix implementation is based on the same principles but targets shared-memory systems such as multi-core chips and symmetric multiprocessors.

Phoenix uses threads to spawn parallel Map or Reduce tasks. It also uses **shared-memory** buffers to facilitate communication without excessive data copying. The runtime schedules tasks dynamically across the available processors in order to achieve load balance and maximize task throughput. Locality is managed by adjusting the granularity and assignment of parallel tasks.

In this paper we evaluate 5 applications (4 of them commonly used in cloud application and on general application) that have been implemented using the Phoenix MapReduce framework [12]:

- **Word Count:** This application is commonly used in search engines for the indexing of the web pages based on the words. It counts the frequency of occurrence for each word in a set of files. The Map tasks process different sections of the input files and return intermediate data that consist of a word (key) and a value of 1 to indicate that the word was found. The Reduce tasks add up the values for each word (key).
- **String Match:** It processes two files: the “encrypt” file contains a set of encrypted words and a “keys” file contains a list of non-encrypted words. The goal is to encrypt the words in the “keys” file to determine which words were originally encrypted to generate the “encrypt file”.
- **Histogram:** It analyzes a given bitmap image to compute the frequency of occurrence of a value in the 0-255 range for the RGB components of the pixels. It can be used in image indexing and image search engines.
- **Linear Regression:** It computes the line that best fits a given set of coordinates in an input file. The algorithm assigns different portions of the file to different map tasks, which compute certain summary statistics like the sum of squares.
- **Matrix Multiply:** Each Map task computes the results for a set of rows of the output matrix and returns the (x,y) location of each element as the key and the result of the computation as the value. This application is a mainly computational intensive application and has been added to show the differences between typical mathematic benchmarks with the applications that are used in cloud computing applications.

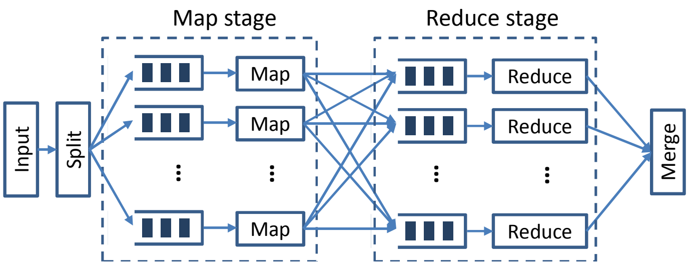


Fig. 2. The Phoenix MapReduce framework

3 Performance Evaluation

In this section we evaluate the Phoenix MapReduce framework in terms of performance and energy efficiency. The Phoenix MapReduce framework has been mapped to three different processors. The first processor is based on a high performance general purpose processor (HP-GPP: Intel i7-2600). This processor has 4 cores and the clock speed is 3.4GHz. The second processor is based on a low power general purpose processor (LP-GPP: Intel U5400 processor) with 2 cores and maximum clock frequency 1.2GHz. The third processor is an embedded system processor that is based on the OMAP4430 SoC with 2 ARM Cortex A9 cores [14]. The detailed characteristics of the processors are shown in the following table.

Table 1. Processor architecture characteristics

	HP-GPP	**LP-GPP**	**EP**
Processor	i7-2600	U5400	OMAP4430
# of Cores	4	2	2
Cores	Intel i7	Intel Pentium	ARM Cortex A9
Process	32nm	32nm	45nm
Frequency	3.4GHz	1.2GHz	1GHz
ISA	CISC	CISC	RISC
L1 Cache	64KB (I),64KB (D)	64KB (I),64KB (D)	32KB (I),32KB (D)
L2 Cache	256KB per core	256KB per core	1MB (shared)
L3 Cache	8MB	3MB	-
Instruction Set	64-bits	64-bits	32-bits
Integrated Graphics	YES	YES	YES

As it is shown in the table the high performance processor has larger cache size and higher clock frequency while the other two processors that are optimized for low power consumption have much smaller caches and lower clock frequencies. The main difference is that the first two processors are based on the Pentium x86 CISC instruction sets while the OMAP4430 processor is based on the ARMv7 RISC instruction set and is optimized for embedded systems applications. Furthermore, the Intel processors are 64-bits wide while the ARM cores are based on 32-bits. In both cases the Phoenix MapReduce framework was hosted on the same operating systems (Ubuntu Linux 11.10).

3.1 Performance Evaluation

Figure 3 depicts the performance evaluation of 4 different common cloud tasks using the Phoenix MapReduce framework in terms of execution time. Besides the 4 common cloud tasks it shows also the execution time of a typical benchmark (matrix multiplication) in order to show the differences between the cloud tasks and other common tasks used in benchmarking. The figure shows the normalized execution time compared to the HP-GPP (Intel i7-2600). As it is shown in this figure the execution time of the embedded processor is 3x to 8.4x higher than the execution time of the HP-GPP while it is 3x to 5x higher than the low power GPP. The lower execution time of the GPPs can be justified by the higher clock frequency and the more advanced instruction set (e.g. deeper pipeline scheme, larger L3 cache and more advanced branch prediction schemes). The highest difference in the execution time is however noticed on the *matrix multiplication* which is not used in the cloud computing applications. In the commonly used cloud computing tasks such as the *word count*, *histogram, linear regression* and *string match* the execution time of the embedded processor range from 3x to 5x higher than the HP-GPP.

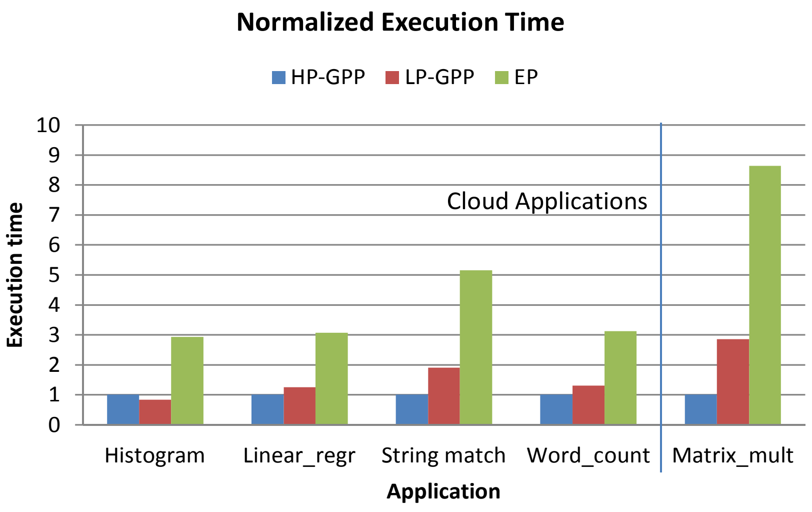


Fig. 3. Normalized Execution time for difference applications

The higher speedup of the *matrix multiplication* can be also justified by Figure 4. This figure shows the average miss rate of the branch predictions for the Intel low power processor. In this figure it is clear that the typical benchmark applications such as the matrix multiplication are much more predictable due to the control structure and therefore the branch miss rate is much smaller than the common cloud application tasks.

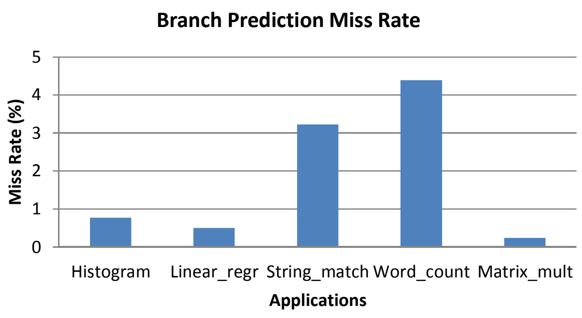


Fig. 4. Branch prediction miss rate

3.2 Energy Efficiency

In this section we evaluate the energy efficiency of the embedded processors compared to the general purpose processors. The energy efficiency is measured by the product of the power consumption by the total execution time of a specific task [15].

$$Energy = ExecTime \cdot Power$$

For the ARM processor we measure the power consumption of the processors using the *Pandaboard* [16] which integrates the OMAP4430 chip with the ARM processors and the DRAM memory. The power measurements are based on the current that is drawn by the ARM processors [17]. On the Intel processors we measure the power consumption using the *powerstat* application [18]. In all cases the CPU utilization is above 80% for the cloud computing applications which means that all processors consume almost the maximum power consumption (Figure 5). Again this figure shows the difference between the matrix-multiplication applications with the typical cloud computing applications. In the case of *matrix multiplication* the processor is fast enough to perform the tasks and the lower utilization is due to the system calls.

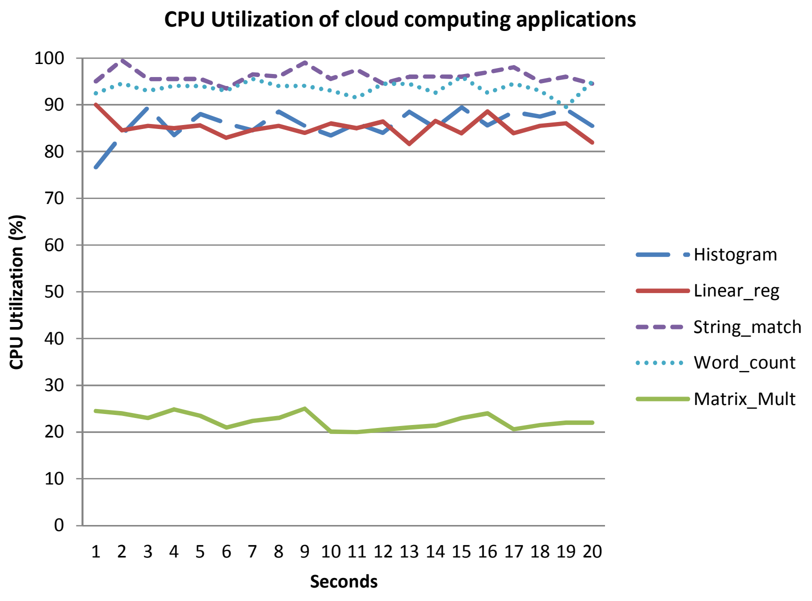


Fig. 5. CPU Utilization of the applications in time

Figure 6 depicts the normalized energy consumption of the HP-GPP, the LP-GPP and the embedded processor for different applications of the Phoenix MapReduce framework. The figure shows the normalized energy based on the energy consumption of the embedded processor. As is it shown in this figure the embedded processors can achieve up to 7.8x lower **energy** consumption compared with the HP-GPP. This is due to the fact that the **power** consumption of the embedded processor is much lower than the power of the GPP. The high power consumption of the GPP is due to the complex instruction set, the advanced branch prediction schemes and the larger caches of the processors. Therefore, even in the case that the embedded processor has longer execution time than the GPPs the overall energy that it consumes is much lower than the GPP. Therefore, in data centers which require energy efficient servers such as the microservers [9], embedded systems could be utilized efficiently reducing the overall power consumption. Furthermore, as the most cloud applications than are based on MapReduce framework are designed to run in parallel systems, the servers could even achieve the same performance in terms of throughput by replicating more embedded system cores but consuming much lower energy.

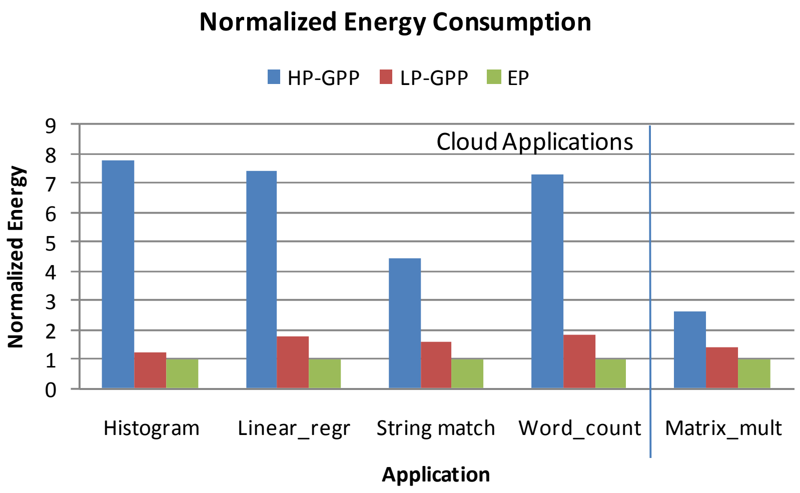


Fig. 6. Normalized Energy Consumption for different applications

4 Conclusions

In this paper we evaluate high performance embedded processors in the domain of cloud computing. We map typical cloud computing application in the ARM Cortex A9-MPCore cores and we compare it with high performance and low power general purpose processors. The performance evaluation shows that the execution time of the embedded processors is up 5x higher than the general purpose processors in tasks common in the cloud applications (word count, string match, etc.). However, the power consumption of the embedded processors is significantly lower the general purpose processors. Therefore high performance embedded processors can achieve up to 7.8x better energy efficiency in cloud computing applications, compared with the general purpose processors and they could be a viable alternative in data centers with lower energy consumption requirements such as microservers. These embedded processors could also be a promising alternative to any other cloud computing applications that can tolerate a small increase in the overall execution time but consuming much lower energy and thus reducing the operating cost of these data centers.

References

1. How Clean is Your Cloud. Greenpeace Technical Report (2012)
2. Make IT Green: Cloud Computing and its Contribution to Climate Change. Greenpeace International Technical Report (2010)
3. Report to Congress on Server and Data Center Energy Efficiency, U.S. Environmental Protection Agency, ENERGY STAR Program (2007)
4. SMART 2020: Enabling the low carbon economy in the information age, A report by The Climate Group on behalf of the Global eSustainability Initiative (GeSI) (2008)
5. Where does power go? GreenDataProject (2008), `http://www.greendataproject.org`
6. Huff, L.: Berk-Tek: The Choise for Data Center Cabling (2008)
7. Reddi, V.J., Lee, B.C., Chilimbi, T., Vaid, K.: Mobile processors for energy-efficient web search. ACM Trans. Comput. Syst. 29(3) (2011)
8. Reddi, V.J., Lee, B.C., Chilimbi, T., Vaid, K.: Web search using mobile cores: quantifying and mitigating the price of efficiency. In: Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA 2010) (2010)
9. The SeaMicro SM10000 Family System Overview, Datasheet, SeaMicro Inc. (2011)
10. Calxeda EnergyCore: ECX-1000 Series, Datasheet, Calxeda, Inc. (2012)
11. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: 6th Symposium on Operating Systems Design and Implementation (OSDI 2004) (2004)
12. Ranger, C., Raghuraman, R., Penmetsa, A., Bradski, G., Kozyrakis, C.: Evaluating MapReduce for Multi-core and Multiprocessor Systems. In: Proceedings of the 13th Intl. Symposium on High-Performance Computer Architecture (HPCA), Phoenix, AZ (February 2008)
13. Yoo, R.M., Romano, A., Kozyrakis, C.: Phoenix Rebirth: Scalable MapReduce on a Large-Scale Shared-Memory System. In: Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC), Austin, TX, pp. 198–207 (October 2009)
14. OMAP4430 Device Silicon Revision, Technical Reference Manual, Texas Instrument
15. Kaxiras, S., Martonosi, M.: Computer Architecture Techniques for Power-Efficiency. Morgan & Claypool Publishers (2008) 1598292080
16. Pandaboard, `http://www.pandaboard.org`
17. van Eijndhoven, J.: Measuring Power Consumption of the OMAP4430 using the PandaBoard. Vectofabrics Inc. (November 2011)
18. Powerstat: Power Consumption Calculator for Ubuntu Linux
19. Bohra, N., Eylon, E.: Micro Servers:An Emerging Category For Data Centers. Server Design Summit (November 2011)

Power-Aware Autonomous Distributed Storage Systems for Internet Hosting Service Platforms

Jumpei Okoshi, Koji Hasebe, and Kazuhiko Kato

Department of Computer Science, University of Tsukuba, Japan
oks@osss.cs.tsukuba.ac.jp, {hasebe,kato}@cs.tsukuba.ac.jp

Abstract. We present a power-saving method for large-scale distributed storage systems of Internet hosting services, whose prime example is a video/photo sharing service. The idea behind our method is to periodically exchange stored data among disks in an autonomous way so as to skew the workload towards a small number of disks while not overloading them. The objective of this paper is to explore a power-saving method that is adaptable to both constant massive influx of data and changes in data popularity. The performance is measured both in simulation and prototype implementation using a real access pattern of 20,000 public photos on Flickr. In the experiments, we observed that our method saved 14.5–39.7% of energy, while the overall average response time was 133 ms, where 6.8–19.1% of total accesses were of disks in low-power mode.

Keywords: power-saving, distributed storage systems, autonomous control.

1 Introduction

Energy efficiency has become a central issue in today's cloud computing. In particular, as a high percentage of the total computing system's energy is used by the data storage systems, various attempts at reducing power use in cloud storage systems have been proposed; e.g., [2,3,4,9,10]. These techniques are essentially based on an idea commonly considered in studies on power-saving in storage systems such as MAID [1] and PDC [5]. That is, they skew the workload towards a small number of disks and thereby enable other disks to be in low-power mode. However, when applying this idea to recent Internet hosting service platforms whose prime examples are YouTube[1] and Flickr[2], several important issues that have not been thoroughly investigated may arise owing to the following dynamic aspects of the stored data.

First, most previous studies explicitly or implicitly assumed that the number of stored data is fixed, but in a real situation, enormous data quantities are continuously uploaded. In addition, their popularity (i.e., frequency of access) may vary at any moment. Second, previous studies often assumed that there is

[1] http://www.youtube.com
[2] http://www.flickr.com/

a specific type of central controller for data allocation to effectively skew the workload, but this technique cannot be directly applied to large-scale storage systems owing to a scalability issue. The objective of this paper is to address these issues and explore a power-saving method that is adaptable to a typical environment of Internet hosting services, i.e., constant massive influx of data and changes in data popularity.

Our method is based on the idea behind MAID and PDC systems, which is the migration of frequently accessed data to a subset of the disks. However, to enhance scalability, our method periodically exchanges data among disks in an autonomous way such that frequently accessed disks tend to gather frequently accessed data from neighboring disks up to their capacity, and the opposite occurs for rarely accessed disks so as to extend their time in low-power mode. In this paper, we also consider several types of restrictions on the exchange of data and evaluate their effect on performance in terms of power consumption, response time, and data migration cost.

To evaluate the effectiveness of our method in a more realistic situation, we measured the performance both in simulation and prototype implementation using a real access pattern of 20,000 public photos uploaded to Flickr, which are observable outside the website. In the experiments, we observed that our method resulted in energy savings of 14.5–39.7%, while the overall average response time was 133 ms. According to our experimental results, our method effectively skewed the workload even if the data migration was conducted autonomously. On the other hand, accesses of data stored on disks in low-power mode totaled 6.8–19.1% of all accesses, and these accesses required extra time for the spinning-up of disks. As a major factor of this problem, we observed that the number of accesses rapidly decreased after one week from the upload for most of the files, and such infrequent accesses were evenly distributed. Thus, in our method, it was difficult to gather such unpopular data on some specific disks completely. This results in a trade-off between the idleness threshold (i.e., the period for never-accessed disks to spin down) and response time.

This paper is organized as follows. Section 2 presents related work. Section 3 describes the design of the proposed storage system. Section 4 gives the results of preliminary investigations of access patterns of public uploaded photos on Flickr, which are used in both simulations and in experiments with a prototype. Sections 5 and 6 present the simulation results and the evaluation of the prototype implementation. Finally, Section 7 concludes the paper and presents future work.

2 Related Work

There have been a number of studies on reducing storage power consumption. A common feature of many of the techniques proposed in the literature is that they skew the workload, and they can be classified into the following categories according to variations in their approach.

The first category, which includes MAID [1] and PDC [5], focuses on the popularity and concentrates popular data on specific disks. The second category, as typified by Pergamum [8], uses NVRAM to extend the low-power mode period by caching data to a write store. The final category considers redundancy (i.e., data replication). In DIV [6], original and redundant data are separated onto different disks, thereby allowing read/write requests to be concentrated on the disks with the original data. In Hibernator [13] and PARAID [11], data are collected or spread to adapt to changes in operational loads.

Although the above studies restricted their scope to storage systems consisting of a relatively small number of disks (typically, up to several dozen), recent works such as those of Harnik et al. [2], Kaushik et al. [4], Verma et al. [9], Vrbsky et al. [10], and our previous work [3] address power-saving in large-scale distributed storage on the basis of the existing skewing technique. Our research can be thought of as a direct successor to these studies based on the approach taken in the first category, but the main motivation is to explore power-saving in an environment where a huge number of data are continuously uploaded and the data access frequency varies at any moment, whose prime example is Internet hosting services.

3 System Design

Our proposed storage system is composed of several thousand (possibly heterogeneous) disks, each of which is classified into one of three groups: Group A, Group B, and the Empty disk pool. (See also Fig. 1 for the graphical presentation.) Each disk autonomously travels among these groups (depicted by the thick arrows in the figure) depending on its capacity in the following way.

The system is assumed to have many empty disks (with unique IDs), which are stored in the Empty disk pool for future increases in stored data. Initially, some of the empty disks in the pool are moved to Group A, and the data uploaded by the clients are always written to these disks. If a disk in Group A becomes full,

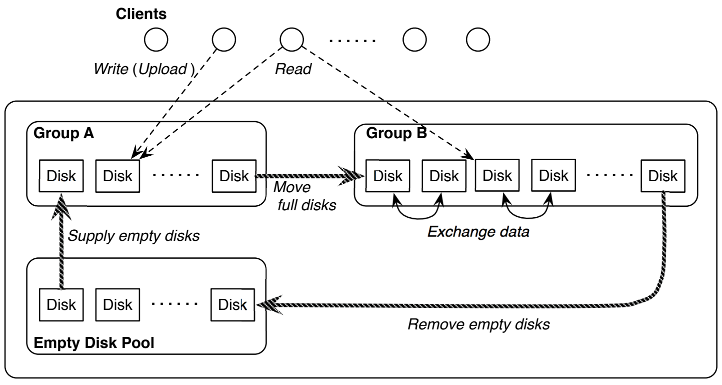


Fig. 1. Architecture of the System

then it moves to Group B and a new empty disk is supplied to Group A from the disk pool.

In Group B, to skew the workload, each disk periodically (e.g., once every hour) exchanges part of its stored data one-for-one following four steps.

Step 1: The disk (say, D_1) with the smallest ID randomly chooses a disk (say, D_2).
Step 2: D_1 and D_2 exchange information of their current workloads.
Step 3: If D_1 is more popular than D_2, the least frequently accessed file (say, f_L) on D_1 is exchanged for the most frequently accessed file (say, f_H) on D_2, and this process is repeated while the frequency of access of f_L is less than that of f_H.
Step 4: D_1 issues an instruction to the next disk.

When implementing this process, the access frequency of popular disks gradually increases, until the capacity of the disks is exceeded, by gathering popular data from the neighboring disks, and the opposite process is carried out for unpopular disks so as to extend their time spent in low-power mode. During this exchange process, if a disk eventually becomes empty, it is removed from Group B and queued in the Empty disk pool for the supply of disks to Group A.

Here, we may consider restrictions on the exchange to reduce migration cost, such as limits on the number of exchangeable files during any one exchange, the number of migrations for each file, and the scope of exchangeable disks. In this study, we evaluated the effects of restrictions on the energy performance of disks, which are described in later sections.

The underlying lookup service used by clients to access data is managed by a distributed hash table, such as Chord [7]. However, for adaption to our system in which data are frequently moved among disks, we use pointers in a list of key-value pairs to indicate the current disk on which data are stored. That is, if a file f_1 stored on disk D_1 migrates to D_2, the key of f_1 is associated with the destination of migration (i.e., D_2) in D_1's list of key-value pairs. In addition, to avoid multiple hops from pointer to pointer, each file is assumed to have information of the ID of the disk originally storing the file as metadata, and if f_1 originally stored on D_1 and moved to D_2 is moved further to D_3, the pointer of D_1 is rewritten so as to directly indicate the current position.

4 Data Access Tracing in Flickr

To evaluate the effectiveness of our method in a realistic situation, as a preliminary study, we traced access patterns of photo data uploaded to Flickr, which is one of the largest photo sharing services in the world.

In this preliminary study, we randomly selected 20,000 photos and traced the cumulative number of accesses for each file every hour over two weeks with APIs provided by the website. Owing to limitations of accessible observable data, all the selected photos are public, although the website has supposedly around four

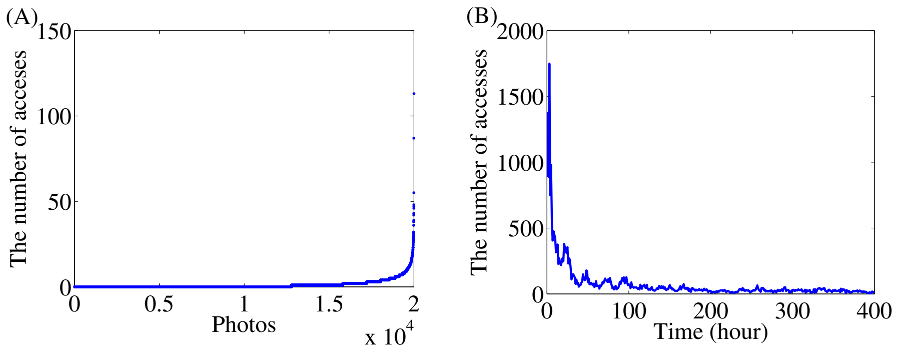


Fig. 2. (A) Distribution of popularity and (B) cumulative number of total accesses of 20,000 files on Flickr

times as many private photos as public photos (according to a Flickr report and our observations).

Figure 2-(A) shows the distribution of popularity for all photos at a lapse of 200 hours, while (B) shows the change in the cumulative number of total accesses of all the files over 400 hours. Figure 2-(A) shows that the file access frequencies were highly skewed, with 69.3% of files never being accessed after upload, while the number of accesses of the most popular file is 133. Figure 2-(B) shows strong negative correlation between the frequency of access and elapsed time. More precisely, the result shows that the frequency of access became highest (1748 accesses per hour) at a lapse of 3 hours and then rapidly decreased, eventually reaching 219 accesses per hour at a lapse of 17 hours. However, the frequency of access did not change after 200 hours, with most files being accessed once per hour or not at all. We here note that it is difficult to forecast which files will be accessed in a short time from the past access pattern. This makes it difficult to completely gather the accesses to some specific disks in an effort to avoid access of disks in low-power mode.

5 Simulation Results

To understand the effectiveness of our method for storage systems consisting of several thousands of disks, we first evaluated the running time of disks and the frequency of access of disks in low-power mode.

Parameters and Settings. In the evaluation presented in this section, we considered the following system. Group A consisted of a single disk and Group B consisted of up to 1500 disks whose number increases depending on the upload. Each disk in Group B required 5 seconds for spin-up where it had been in low-power mode. We set the idleness threshold as 30, 60, 90, and 120 seconds.

The workload in the simulations was based on the access traces obtained by the observation of Flickr described in Section 3. In our simulation, to determine the trace of each file, we chose at random from the set of 20,000 real traces when uploaded. However, as our traces were observed for 400 hours, we expanded the real traces to 2400 hours by repeating the access pattern from 200 hours to 400 hours for the period after 400 hours.

In the simulations, we first measured the running time to evaluate the impact of both the configurations of idleness threshold and restrictions of data exchange on the power consumption then measured the number of accesses of disks in low-power mode in some configurations.

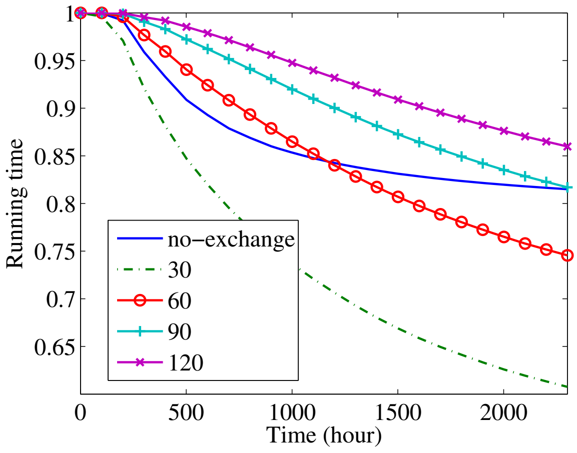


Fig. 3. Running time in some configurations of idleness threshold

Impact of Exchange and Idleness Threshold on Power Consumption. Figure 3 shows the change in running time for different configurations of idleness threshold. The figure shows the relative time when the running time in the case that all the disks are always active is equal to one. In this figure, the "no-exchange" configuration means that the disks spin down after an idle time (30 seconds) without data exchange and, in other configurations, disks spin down after an idle time (30, 60, 90, and 120 seconds) with data exchange. We observed that power consumption after 2400 hours was reduced by 14.5–39.7% in each configuration. From this result, we observed that the data exchange was effective in reducing power consumption as demonstrated by comparing the "no-exchange" configuration and all the configurations with data exchange, among which the "30" configuration was the most effective.

Impact of the Restrictions of Data Exchange on Power Consumption. Figure 4 shows the change in running time for four configurations, which differed in terms of the application of two restrictions. To denote these configurations, we use two digits b_1 and b_2, where b_1 indicates application of a limit on the exchange range (in the case that $b_1 = 1$, one disk can change files with only 20

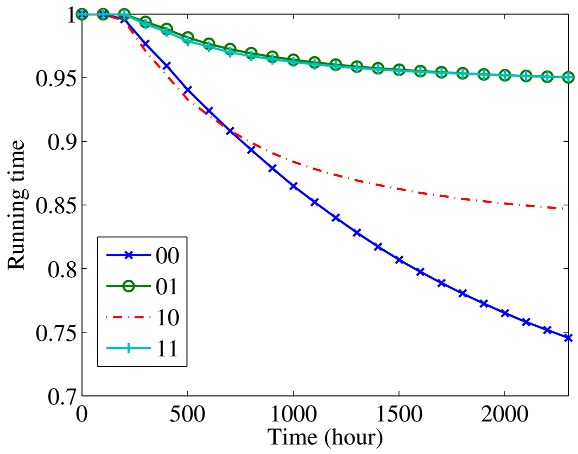


Fig. 4. Running time for some restrictions of data exchange

specific disks), while b_2 indicates application of a limit on data migration (in the case that $b_2 = 1$, the migration capacity is limited to 10% of each disk space). The figure shows the relative time when the running time in the case that all the disks are active is equal to one. Here, the idleness threshold is fixed as 60 seconds in each configuration. In this simulation, we observed that our method still reduced power even if one of the restrictions was introduced, which may reduce the migration cost. In addition, under these parameter settings, the limit on the exchange range (i.e., b_1) was stronger than the limit on data migration (i.e., b_2), which would change depending on a given trace.

Accesses of Disks in Low-Power Mode. Figure 5 shows the change in ratio of the access of disks in low-power mode among all accesses in the same configuration as in the case of Figure 3. We observed that 6.8–28.7% of accesses were of disks in low-power mode. This result means that 6.8–28.7% of accesses need extra time for spinning-up. However, it should be emphasized that the data exchange realized a 9.5% reduction in accesses of disks in low-power mode, as seen by comparing the "no-exchange" configuration and "30" configuration, which had the same idle time but differed in terms of there being data exchange.

6 Experiments on an Implementation

We conducted an experiment on the current prototype implementation of our proposed system to evaluate the applicability of our method to a real system. We measured the response time and the cost of data migration in an environment where the system workload was the same as that in the simulation.

Our prototype consisted of 50 PC servers (where one server was used for the client and the other servers were used for Group B), each of which was equipped

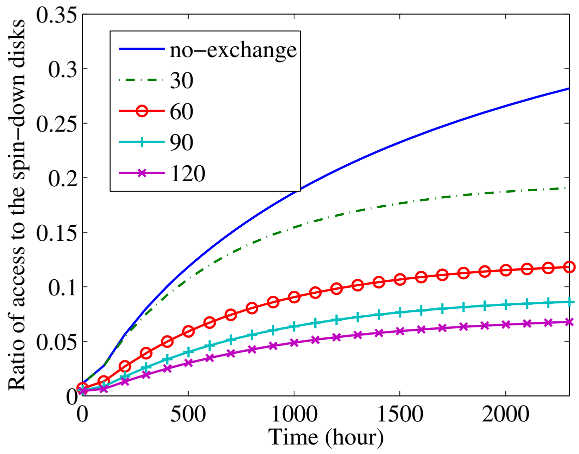


Fig. 5. Ratio of access of disks in low-power mode

with a Dual Xeon 3.60 GHz CPU, 2 GB memory, and a single 36 GB SCSI disk. For our prototype, owing to the limitation of our experimental environment (i.e., the bandwidth of different servers), we evaluated the response time of data access by measuring the time from sending a request until the data were loaded into the memory of the server. In addition, no underlying lookup service to access data was implemented in our prototype. Thus, in the experiments, the data were accessed by their storing server.

Parameters and Settings. In our experimental environment, although the real capacity of a disk was 36 GB, we assumed that the capacity was 500 GB, which was emulated by only accessing one file. In addition, because it was difficult to spin up or spin down disks in the current system configuration, we realized these actions by letting the server wait before accessing the disk. The two parameters were spin-up time of 5 seconds and idle time before spin-down of 60 seconds. (According to simulation analysis, this configuration was the best in terms of response time and power consumption.)

Response Time. Figure 6 shows the change in response time. The figure shows that the overall average response time is 133 ms. However, we observed that some responses were delayed and required more than 5 seconds after 15 hours, among which the worst took 18,307 ms. This result means that some accesses were of the spin-down disk and had to wait for the disk to spin up or for the completion of another file access.

Migration Cost. Table 1 shows the migration cost evaluated by measuring the number of files that migrated per data exchange and the elapsed time. Note that this result was for 48 hours and the migration cost would decrease gradually after 48 hours. In addition, owing to the limitation of our experimental environment, times in Table 1 are predicted values. This result shows that the required time for most of the migration was short relative to the frequency of exchange (i.e., once every hour) and would have no adverse impact on the performance.

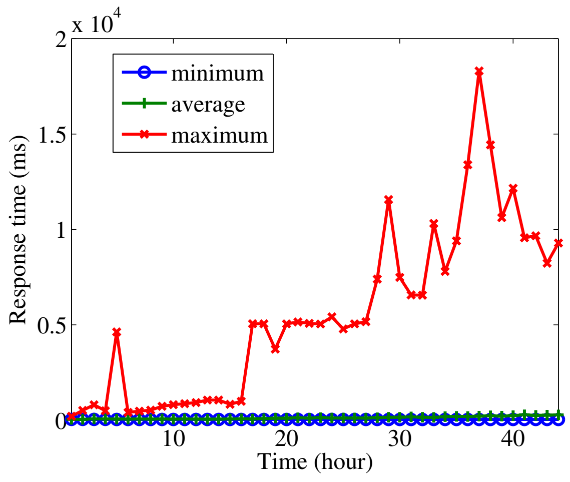


Fig. 6. Response time

Table 1. Migration cost

	minimum	median	average	maximum
The number of exchanged files	2	1691	13583	163079
Time (s)	0.2	170	1426	17123

7 Conclusions and Future Work

We presented a power-saving method for large-scale distributed storage systems, especially those on Internet hosting service platforms. The idea to skew the workload is similar to that of MAID and PDC, but to adapt to the constant massive influx of data and changes in data popularity, our approach periodically exchanges data among disks autonomously, instead of introducing a central controller to manage the relocations. We also evaluated the performance both in simulations and prototype implementation for a real access pattern of 20,000 public photos on Flickr. We observed that our method saved 14.5–39.7% energy, even if we introduced several restrictions on data migration. The ratio of accesses of disks in low-power mode was 6.8–19.1%, and the overall average response time was 133 ms.

The results of this research suggest that our autonomous approach can effectively skew the workload, but some accesses require extra time to wait for the spinning-up of disks in low-power mode. This results in a trade-off between the idleness threshold of disks, which may increase the power consumption, and the performance of the response time. To more effectively skew the workload, we should investigate a radical solution that possibly considers effective allocations of replicas. In addition, we should thoroughly estimate the cost of underlying lookup services. These topics shall be investigated in future work.

References

1. Colarelli, D., Grunwald, D.: Massive arrays of idle disks for storage archives. In: ACM/IEEE Conference on Supercomputing, pp. 1–11 (2002)
2. Harnik, D., Naor, D., Segall, I.: Low power mode in cloud storage systems. In: Parallel and Distributed Processing Symposium, International, pp. 1–8 (2009)
3. Hasebe, K., Niwa, T., Sugiki, A., Kato, K.: Power-Saving in Large-Scale Storage Systems with Data Migration. In: IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2010), pp. 266–273 (2010)
4. Kaushik, R.T., Bhandarkar, M.: GreenHDFS: towards an energy-conserving, storage-efficient, hybrid Hadoop compute cluster. In: 2010 International Conference on Power Aware Computing and Systems (HotPower 2010), pp. 1–9 (2010)
5. Pinheiro, E., Bianchini, R.: Energy conservation techniques for disk array-based servers. In: International Conference on Supercomputing, pp. 68–78 (2004)
6. Pinheiro, E., Bianchini, R., Dubnicki, C.: Exploiting redundancy to conserve energy in storage systems. In: ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, pp. 15–26 (2006)
7. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: ACM SIGCOMM, pp. 149–160 (2001)
8. Storer, M., Greenan, K., Miller, E., Voruganti, K.: Pergamum: Replacing tape with energy efficient reliable, disk-based archival storage. In: USENIX Conference on File and Storage Technologies (FAST 2008), pp. 1–16 (2008)
9. Verma, A., Koller, R., Useche, L., Rangaswami, R.: SRCMap: energy proportional storage using dynamic consolidation. In: 8th USENIX Conference on File and Storage Technologies (FAST 2010), pp. 154–168 (2010)
10. Vrbsky, S.V., Lei, M., Smith, K., Byrd, J.: Data Replication and Power Consumption in Data Grids. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom 2010), pp. 288–295 (2010)
11. Weddle, C., Oldham, M., Qian, J., Wang, A., Reiher, P., Kuenning, G.: PARAID: A gear-shifting power-aware RAID. In: USENIX Conference on File and Storage Technologies (FAST 2007), pp. 245–260 (2007)
12. Yao, X., Wang, J.: RIMAC: a novel redundancy-based hierarchical cache architecture for energy efficient, high performance storage systems. In: ACM SIGOPS/EuroSys European Conference on Computer Systems, pp. 249–262 (2006)
13. Zhu, Q., Chen, Z., Tan, L., Zhou, Y., Keeton, K., Wilkes, J.: Hibernator: helping disk arrays sleep through the winter. In: ACM Symposium on Operating Systems Principles, pp. 177–190 (2005)

A Digital Infrastructure for Green Utility Computing: The Preliminary Holistic Research Agenda

Elena Irina Neaga[*]

School of Business and Economics,
Loughborough University, Loughborough, LE11 3TU, United Kingdom
e.i.neaga@lboro.ac.uk

Abstract. The paper presents ICT environmental issues and discusses the transition towards green utility computing. Cloud services mainly based on virtualisation have a huge undiscovered yet potential to adequately address the issues of ICT to become greener alongside other economics (labour related costs such as people managing the ICT infrastructure). In order to outline the research agenda the paper includes a systemic / holistic analysis of the ICT development and usage as a pervasive/ubiquitous enabler in the daily life activities as well as different economic sectors (e.g. transportation, manufacturing, commerce, tourism etc.) and the increased environmental impacts. The main aim of this approach is the definition of a multi perspective research agenda for the development of an environmentally aware digital infrastructure to deploy green utility computing in the cloud. The increasing complexity of managing the whole infrastructure of platforms, people and services that transparently support ICT as a utility is also discussed. Some boundaries conditions are imposed.

Keywords: utility and cloud computing, virtualization, green/environmentally friendly ICT, energy efficiency, complex system and complexity, system dynamics.

1 Introductory Background

Environmental issues are emerging as important topics in the agenda of almost any technology advancement and associated debates. Several ICT vendors are becoming increasingly aware of the importance of producing environmentally friendly products. This is not surprising as the Global Action Plan (2009) supported by Logicalis, and the Environmental ICT showed that 2% of global carbon emissions are due to ICT [1]. An interesting aspect is that 98% of global carbon emissions require solutions [1] where ICT is expected to play an important role to support comprehensive environmentally aware approaches as described by Ghose et al. (2008) [2]. Indeed, ICT could support the energy efficiency and environmental awareness through intelligent systems enabling to reduce resource consumption, and power management. Xiao (2011) provided an approach for modeling and managing efficient power management of mobile technologies [3]. Other approaches suggested how new computing technologies

[*] Corresponding author.

including virtualisation and the cloud can be used for the realisation of the full spectrum of benefits of a distributed intelligent power management systems [4], [5]. Alternative solutions might apply the bio-inspiration alongside new technique such as cuckoo search. Generally, the application of ICT in intelligent systems for production and consumption of energy has the following impacts and directions that have provided the basis of the definition of key research questions:

Direct impact: Consumption of energy and non-renewable resources in the ICT sector and related products. How can a digital infrastructure materialised through software and hardware system become greener through the application of a wide range of methods as well as the new cloud computing paradigm? How could be minimised the negative environmental impacts of obsolete equipment?
Enabling solution: How can ICT reduce energy consumption in other sectors: using video conferencing instead of travelling, home work place, intelligent traffic systems, smart buildings, intelligent transportation management systems etc.? (as depicted in figure 2)?
Systemic Approach: Green ICT which generally, refers to the use of ICT resources in an energy-efficient and cost-effective manner particularly applying cloud computing and holistically analysed the resulted complex system. How can green ICT change the way we live and work and resulting in substantially reducing the production and consumption of energy? Is the social awareness through policies and regulations enough explained and directed to easing environmental issue and urge conservation?

How is it possible to develop systems metrics and use them?

At the same time more often than before moving from pervasive ICT office/home based applications to ICT as a utility is an obvious step forward. ICT as a utility or utility computing should be simple, accessible and reliable appearing invisible and providing easy and flexible access to computing resources that should be metered. There are similarities with other utilities, but in order to achieve the foreseen vision of IT becoming a utility (alongside water, gas, electricity, and telephone) different computing platforms (e.g. grids) have been proposed and analysed particularly, by Buyya et al. (2009) [6]. The full spectrum of consequences of achieving the vision of ICT as a utility should require an analysis and definition of a research agenda, short and long term challenges and a roadmap directed at providing recommendations for future research, development, implementation, deployment and adoption. For example, it should be considered that the use of ICT equipment especially obsolete is energy consuming and an increasing share of the energy consumption is desirable. Also recent studies demonstrated that carbon dioxide emissions from some data centres are increasing.

This paper discusses the systemic relationships between ICT development, pervasive support and the environmental issues with the main aim of defining a realistic research agenda for the identification of the design solutions of an environmentally friendly digital infrastructure for cloud or utility computing. The application of the cloud computing model that uses the virtualisation feature is at the core of this approach. Generally, virtualization is a technique for hiding the physical characteristics of computing resources to simplify the way in which other systems, applications, or end users interact with those resources. Virtualization, in essence is the ability to emulate hardware via software and it is used in some computing paradigms including cloud computing. Virtualisation is only one technology behind cloud computing development and new technologies might be even more beneficial for energy savings, reducing greenhouse emissions and related issues.

2 The Landscape of Green Utility ICT

2.1 Assessment of the Impacts of ICT on the Environment

A comprehensive review of the state-of-the art until 2007 of the impacts of electronic business and generally ICT on the environment has been provided by Yi and Thomas (2007) [7]. This review article presents the contents of various journal papers and thesis as well as other resources such as projects and associated reports, conference and symposia, and websites with the main aim of examining and capturing the most important approaches to date, either for a general knowledge of the area of the environmental impact and issue of extensive use of ICT. This background study could be used by experts carrying out future research and defining the directions of ICT as a utility matching the green computing requirements. This review highlighted that the dominant approach is either a micro-level case study or a macro-level statistical method and it is concluded that a more predictive and empirical model, which can be applied within different sectors should be more beneficial in the long term. Evaluations of the impacts of ICT on the environment have been made by different organisations including European Commission (EC) together with Joint Research Centre (JRC) [8] that have reported the future impacts of ICT on environmental sustainability. A complex system representing a set of implications of the application/use of ICT has been identified and it is presented in figure 1 that also has identified the systems boundaries.

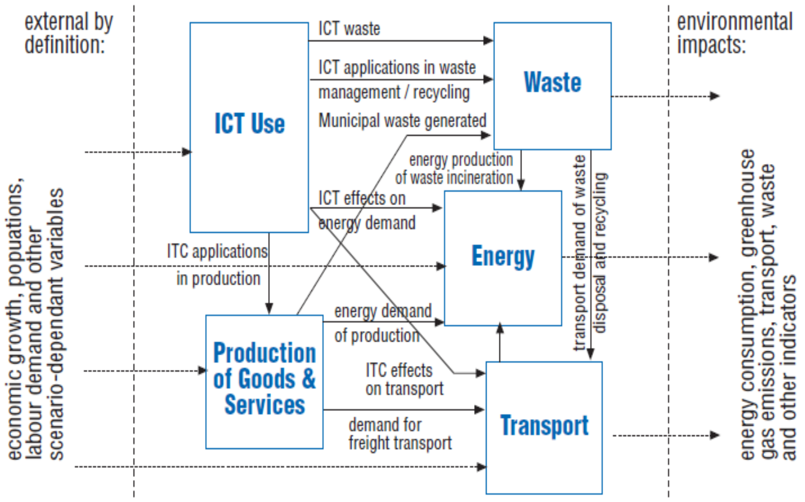

external by definition:
economic growth, popuations, labour demand and other scenario-dependant variables
ICT Use
ICT waste
ICT applications in waste management / recycling
Municipal waste generated
Waste
energy production of waste incineration
ICT effects on energy demand
Energy
ITC applications in production
Production of Goods & Services
energy demand of production
ITC effects on transport
demand for freight transport
Transport
transport demand of waste disposal and recycling
environmental impacts:
energy consumption, greenhouse gas emissions, transport, waste and other indicators

Fig. 1. Complexity and relationships of the Environmental Impacts of using ICT to different sectors [8]

The main outcome of this report has identified that the direct impact created by ICTs is negative, the overall impact on environmental sustainability may vary, depending on the applications and the aggregated effects of large numbers of people and organisations using ICTs. The UK's target is to reduce emissions by 80% by 2050 and by at least 34% by 2020. Large organisations such as public sector bodies or universities, where IT usage is extremely high find very difficult to reach these targets without adopting a fundamentally different approach in their IT energy strategy. If ICTs are to enable a decrease in absolute energy consumption, policy must also be designed in order to promote the environmentally positive impacts of ICTs, whilst inhibiting the negative ones [8].

Hilty et al. (2006) have developed a simulation study and a system dynamics model based on systems representation, relationships, and decomposition shown in figure 2. The simulation study combined scenario based demonstrations and expert consultations for analysing the positive and negative impacts of ICT on environmental sustainability [9]. The basic idea behind a systemic modeling approach has been the development of conceptual bridges from the use of ICT in different economic sectors to the environmental impact indicators (e.g. greenhouse emission, accounting for the following three types of ICT impacts or effects [9], [10], [11], [12]:

1. Primary impacts due to the effects of the physical existence and running of ICT (environmental impacts of the production, use, recycling and disposal of the hardware).
2. Secondary impacts to the indirect environmental effects of ICT due to its power to change processes (such as production or transport processes), resulting in a modification (decrease or increase) of their environmental impacts.
3. Low level impacts due to the environmental effects of the medium- or long-term adaptation of behaviour (e.g. consumption patterns) or economic structures due to the stable availability of ICT and the provided services.

However, due to some limitations of the system dynamics the results should not be interpreted as forecasting the development of the environmental indicators, because their absolute values in 2020 will greatly depend on the selected scenarios and on different parameters used for modeling that could not accurately capture the uncertainities. On the one hand, significant opportunities for improving environmental sustainability are in the potential impact of ICTs on the rational use of heating energy, and the support of decentralized electricity production from renewable sources and its important role in the product-to-service paradigm shift. On the other hand, ICT applications that make freight and passenger transport more time efficient (cheaper or faster) will immediately create more traffic and possibly more energy consumption. There is no empirical evidence for assuming anything other than a strong price rebound effect here, which could have severe environmental consequences in terms of energy use and greenhouse gas emissions (GHG).

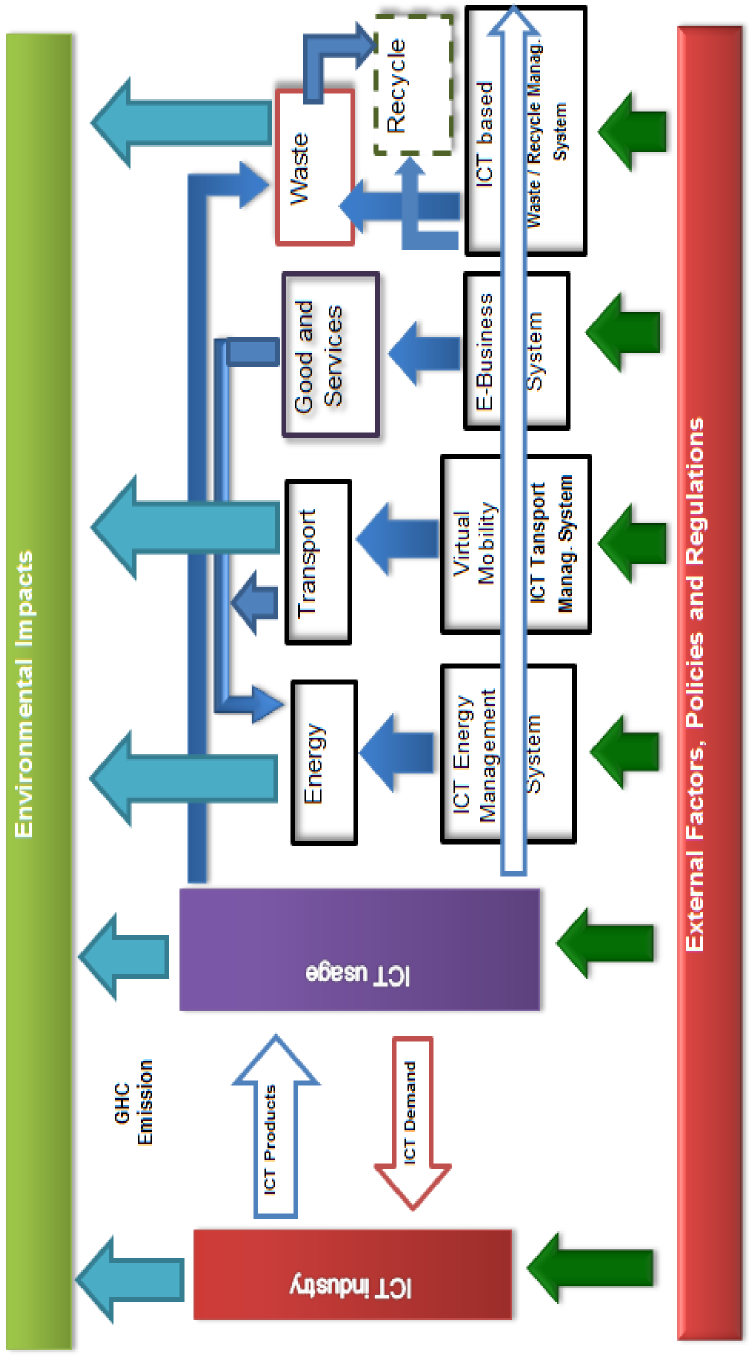


Fig. 2. Environmental Awareness System Representation based on [9]

2.2 Exploring the Vision of Green Utility Computing Model

The challenge of any research is not to just recognise the issues, but to know what can be done, how it can be done, and to identify certain solutions or at least directional recommendations towards solutions.

It is not enough to know that ICT being pervasive has been changing our lives, and different sectors of the economy such as manufacturing, business and transportation sectors. Implementing and operating utility computing might change even more, and ultimately an integrated approach which can influence the practices through environmental policies of green ICT of an organisation is needed. Bose and Luo (2011) have defined from the theoretical perspective a framework for the assessment of the potential of the organisation to undertake green ICT initiative especially using virtualization and cloud computing. This framework considers the organisational, technological and environmental perspectives attempting to create synergies that will increase the likelihood that companies will successfully implement green utility computing initiatives [13].

3 ICT as a Utility in the Cloud

3.1 Why Cloud Computing for Utility Computing?

Cloud computing has the foundation in distributed computing implemented usually using service oriented architecture and grid computing [14], [15]. Buyya et al. (2009) have provided the following description [6] “A cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers." The distributed computing and mobile technologies have been already experimented for utility computing and these cover the essential requirements of utility computing. Moreover the National Institute of Standards and Technologies (NIST) has emphasized the elasticity feature of computing resources in their definition of cloud computing [16] that is largely accepted and frequently cited. This definition is as follows: “Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services)”.

Cloud Computing has resulted from the convergence of Grid Computing, and computing services, and represents the contemporary trend towards the external deployment of ICT resources, such as computational power, storage or business applications, and obtaining them as services. These resources can be dynamically re-configured to adjust to a variable load (scale), allowing also for an optimum resource utilization that are needed for utility computing. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the infrastructure provider by means of customized Service Level Agreements [17]. Grace (2010) has made explicit the distinctions between clouds, service oriented architecture and grid computing [18]. However the delimitation line is very narrow and particularly, with grid computing. Also a cloud infrastructure can be defined on service orientated architectures by adding a layer of virtualization and self- provision. On the other hand a service architecture could be built on clouds by adding a service layer. In fact the service concept is at the core of both cloud computing and service oriented architecture. Cloud computing is a broader concept than utility computing and relates to the underlying architecture in which the services are designed [6]. It may be applied equally to utility computing services and data centres and both could contribute to energy saving of ICT usage.

3.2 The complexity of Operating Utility Computing

As a great deal of uncertainty still exists and also aiming to the deployment of utility computing further research is necessary for a deeper understanding and assessment of the environmental impacts considering systems boundaries conditions without neglecting essential implications. At the same time meeting environmental policy goals is required; and as such more accurate capturing interactions within such a complex system is needed. As operating a utility is also a complex issue, this holistic approach will encompass the several interrelated key aspects:

1. Intelligent transportation systems' impact on increasing transport performance and promoting a shift from the use of the private car to public transport
2. ICT based equipment's electricity monitoring and consumption in different sectors including the domestic and tertiary sector
3. Efficiency in electricity generation, distribution, monitoring and decrease the consumption using intelligent energy savings systems
4. ICT-supported the management of energy savings and allocation of incentives
5. Searching for new solutions based on advanced mathematics and/or biologically inspired methods (e.g. cuckoo search algorithm)
6. Using of a virtual utility to promote renewable energy and combined heat and power
7. ICT-supported systems for recovery and recycling of the waste in general and waste from electrical and electronic equipment in particular
8. Moving/Adopting IT utility services in the cloud and/or using only virtualisation technologies
9. Designing green data centres (servers, storage drivers, any telecoms equipment housed within the data centre) that should provide an alternative solution to reduce the power consumption
10. Efficient design and optimisation for green IT solutions including communication, networks and data storage
11. Including an impact assessment in early design stage of ICT products

4 Cloud Driven Environmentally Aware Digital Infrastructure

Several authors have analysed the environmental impact of running an ICT digital infrastructure due to electricity consumption (needed to run hardware e.g., workstations, servers, switches, backup drives, etc.) and cooling system (needed to reduce the heating generated by the hardware). ICT uses a great deal of energy and it is rising fast as follows [1]:

1. ICT equipment accounts for 10% of the UK's electricity consumption.
2. Non-domestic energy consumption from ICT equipment rose by 70% from 2000–2006, and is predicted to further grow 40% by 2020.
3. Data centres account for about a quarter of the ICT sector's emissions.

A cloud computing infrastructure for ICT as a utility is likely to substantially reduce all these costs, and the environmental impact alongside also reducing labour-related costs, as less people (e.g., technicians) than existing ICT infrastructures will be required to run a cloud-based ICT infrastructure.

In cloud computing, everything is defined and treated as a service such as SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service) [15], [18]. These services define a multi-layered infrastructure as shown in figure 3 and the related main services are defined as follows:

1. *Software as a Service* (SaaS), where applications are hosted and delivered online via a web browser offering traditional desktop functionality
2. *Platform as a Service* (PaaS), where the cloud provides the software platform for systems (as opposed to just software)
3. *Infrastructure as a Service* (IaaS), where a set of virtual computing resources, such as storage and computing capacity, are hosted in the cloud; customers deploy and run only their own applications for obtaining the needed services.

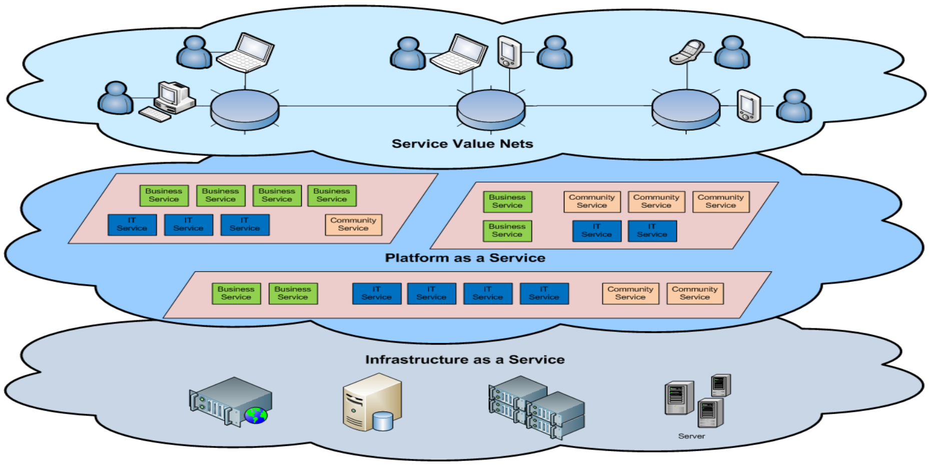


Fig. 3. The Cloud based Digital Infrastructure

At the infrastructure level, processing, storage, networks, and other fundamental computing resources are defined as standardized services over the network. Cloud providers' users can deploy and run operating systems and software for their underlying infrastructures. The middle layer, i.e. PaaS provides abstractions and services for developing, testing, deploying, hosting, and maintaining applications in an integrated development environment. The application layer provides a complete application set of SaaS. The advantage of the cloud based digital infrastructure is not just related to how much resources users can save by not buying and installing hardware and software and therefore using less power. Users of cloud computing are more likely to significantly reduce their carbon footprint. In the United Kingdom, for example, increasingly stringent regulations (such as the Carbon Reduction commitment and EU Energy Using Products Directive) are likely to put pressure on organisations such as the educational institutions to make ICT more environmentally sustainable [19]. In an environment where there is an increasing concern about organisations' carbon footprint and energy costs, virtualized services (such as those provided in a cloud based environment) are of very much interest [20]. Also data centers in the cloud could be environmental friendly storage, but concerns of data protection and privacy are also important.

5 Concluding Remarks

This approach is the basis for the definition of a systemic vision of green cloud based utility computing model. Despite the lack of quantitative evidence of the advantages of the systems approaches a balanced complete research agenda should include:

1. An holistic in-depth analysis of the impacts of ICT usage on the environmental sustainability and the definition of a roadmap through public consultation;
2. Effective inter-disciplinary systems research developing green sustainable ICT
3. Mechanisms for dealing with the complexity of operating utility computing, enhancing major benefits and minimising potential negative consequences particularly on the environment
4. Implementation solutions for utility computing using virtualisation for cloud based services and an elaborated digital infrastructure
5. Other key issues such as data privacy, and security and the need of a legal and regulatory standardised framework.

Acknowledgment. This paper has received financial support from the Network+ focused on IT as a Utility funded by RCUK under Digital Economy Programme. We particularly, thank to the ITaaU Network+ champion, Professor Jeremy Frey.

References

1. Global Action Plan: Green ICT Handbook A guide to Green ICT, @Global Action Plan (2009)
2. Ghose, A., Hasan, H., Spedding, T.: Carbon-centric Computing: IT Solutions for Climate Change, A report prepared by the University Working Group on the Carbon-centric Computing Initiative, University of Wollongong, p. 17 (2008)
3. Xiao, Y.: Modeling and Managing Energy Consumption of Mobile Devices. PhD thesis, Aalto University (2011)

4. Xiao, Y., Savolainen, P., Karpanen, A., Siekkinen, M.: Practical power modeling of data transmission over 802.11g for wireless applications. In: E-Energy 2010, pp. 75–84. ACM (2010)
5. Xiao, Y., Hui, P., Savolainen, P.: CasCap: Cloud-assisted Context-aware Power Management for Mobile Devices. In: Proceedings of MCS 2011, pp. 13–17. ACM (2011)
6. Buyya, R., Chee Shin Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. In: Future Generation Computer Systems, vol. 25, pp. 599–616 (2009)
7. Yi, L., Thomas, H.R.: A review of research on the environmental impact of e-business and ICT. Environment International 33, 841–849 (2007)
8. Erdmann, L., Hilty, L., Goodman, J., Arnfalk, P.: European Commision and Joint Research Centre: The Future Impact of ICT on Environmental Sustainability. In: Casal, C.R., Wunnik, C.V., Sancho, L.D., Burgelman, J.C., Desruelle, P. (eds.) Technical Report Series (2004)
9. Hilty, L.M., Arnfalk, P., Erdmann, L., Goodman, J., Lehmann, M., Wager, P.: The relevance of information and communication technologies for environmental sustainability — a prospective simulation study. Environmental Modelling and Software 21(11), 1618–1629 (2006)
10. EITO (European Information Technology Observatory): European Economic Interest Grouping, Frankfurt am Main (2002)
11. Kohler, A., Erdmann, L.: Expected environmental impacts of pervasive computing. Human and Ecological Risk Assessment 10(5), 831–852 (2004)
12. Hilty, L.M.: From Environmental Informatics to Sustainability Informatics? Invited Lecture. In: EnviroInfo 2010 (24th International Conference in Informatics for Environmental Protection) and InterGEO 2010 (Kongress und Fachmesse für Geodäsie, Geoinformation und Landmanagement), Köln (2010)
13. Bose, R., Luo, X.: Integrative framework for assessing firms' potential to undertake Green IT initiatives via virtualization – A theoretical perspective. Journal of Strategic Information Systems 20, 38–54 (2011)
14. European Commission: The Future of Cloud Computing - Opportunities for European Cloud Beyond 2010, European Commission Public Report (2010)
15. Rittinghouse, J.W., Ransome, J.F.: Cloud Computing Implementation, Management and Security. CRC Press Taylor and Francis (2010)
16. Mell, P., Grance, T.: The NIST definition of cloud computing v15. Version 15 (2009), `http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc`
17. Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M.: A break in the clouds: towards a cloud definition. SIGCOMM Comput. Commun. Rev. 39(1), 50–55 (2009)
18. Grace, L.: Basics about Cloud Computing, Software Engineering Insititute, Carnegie Mellon University, USA (2010), `http://www.sei.cmu.edu/library/assets/whitepapers/Cloudcomputingbasics.pdf`
19. James, P., Hopkinsons, L.: Sustainable ICT in further and higher education: Final report. A Report for Joint Information Services Committee (JISC) (2009)
20. Katz, R.N.: The gathering cloud: Is this the end of the middle? In: Katz, R.N. (ed.) The Tower and the Cloud: Higher Education in the Age of Cloud Computing (2008)

Energy-Aware Multi-Agent Server Consolidation in Federated Clouds

Alessandro Ferreira Leite and Alba Cristina Magalhaes Alves de Melo

Department of Computer Science
University of Brasilia, Brasilia, Brazil

Abstract. In this paper, we propose and evaluate a server consolidation approach for efficient power management in virtualized federated Data Centers. The main goal of our approach is to reduce power consumption, trying to meet QoS requirements with limited energy defined by a third party agent. In our model, we address application workload considering the costs due to turning servers on/off and Virtual Machine migrations in same Data Center and between different Data Centers. Our simulation results with 2 data centers and 400 simultaneous Virtual Machines show that our approach is able to reduce more than 50% of energy consumption, while still meeting the QoS requirements.

1 Introduction

Cloud Computing is a recent paradigm for provision of computing infrastructure, platform and/or software. This paradigm shifts the location of these components to the Internet in order to reduce costs associated with resource management (hardware and software) [10].

Cloud Computing is gaining popularity since it helps companies to reduce costs and carbon footprint. Usually, services are executed in big Data Centers containing a large number of computing nodes. The energy requirements of the whole Data Center have a high impact on the total operation costs [11], which can be over 60% of the peak load [8, 15]. Therefore, reducing the energy consumption without sacrificing Quality-of-Service (QoS) requirements is an important issue.

In Cloud Computing, Data Centers usually employ virtualization techniques to provide computing resources as utilities and Virtual Machine (VM) technologies for server consolidation. Server consolidation is the process of gathering several virtual machines into a single physical server. It aims at minimizing the number of physical servers required to host a group of Virtual Machines.

Many studies have been conducted to provide power reduction and some of them are based in server consolidation [1]. However, server consolidation in Cloud Computing can introduce some difficulties such as: (i) the Cloud environment must usually provide Quality of Service (QoS) guarantees, normally defined in terms of Service Level Agreements (SLA); (ii) it is common to occur dynamic changes of the incoming requests rate; (iii) the usage pattern of the resources is often unpredictable and (iv) different users have distinct preferences.

Currently, with the energy costs increasing, the focus shifts from optimizing Data Center resource management for pure performance to optimize it for energy efficiency while maintaining the level of services [2].

In a Cloud Computing environment, there are distinct participants with distinct objectives, preferences and disposition to pay for services. In this scenario, a Multi-agent System (MAS) can be used where each participant is an autonomous agent that incorporates market and negotiation capabilities.

In this work, we propose a Federated Application Provision (FAP) strategy which uses multiple agents and server consolidation techniques to achieve power-aware resource allocation, by taking into account SLAs, energy consumption and carbon footprint. In our approach, the user should pay according to the efficiency of his/her applications in terms of resource utilization and power consumption. Therefore, we propose that the price paid by the users should increase according to the whole energy consumption of the Data Center, especially when the user does not accept to negotiate QoS requirements.

Experimental results for our FAP consolidation strategy were obtained in the CloudSim [3] simulator, with 2 Data Centers, each one belonging to a different Cloud, and 400 simultaneous virtual machines show that our approach is able to reduce an average of 53.57% of energy consumption, while meeting the SLA requirements.

The remainder of this paper is organized as follows. Section 2 presents concepts of Cloud Computing. Section 3 discusses energy green performance indicators. The proposed strategy for Federated Cloud server consolidation is presented in Section 4. In Section 5, experimental results are discussed. Section 6 presents related work. Finally, Section 7 presents the conclusion and future work.

2 Cloud Computing

There are many definitions of cloud computing in literature. Most of these definitions state that a Cloud Computing system should have (i) pay-per-use capability, (ii) elastic capacity and the illusion of infinite resources, (iii) self-service, (iv) virtualized resources and (v) QoS enhancement functionality. The cloud service models are divided in three classes, according to the abstraction level and the service model of the providers: Infrastructure-as-a-Service (IaaS), Plataform-as-a-Service (PaaS), and Software-as-a-Services (SaaS) [17].

In the Infrastructure-as-a-Service (IaaS) model, the user can request processing power, storage, network and other fundamental computing resources such as the operating system, for a period of time and pay only what he/she uses. Plataform-as-a-Service (PaaS) are development platforms that allow the creation of applications with supported programming languages and tools hosted in the cloud and accessed through a browser. This model can slash development time, offering readily available tools and services. In the Software-as-a-Service (SaaS) model, applications run on the Cloud infrastructure and are accessible from various client devices. From the user view, the SaaS model allows him/her to save money in servers and software licenses.

Cloud Computing can be viewed as a combination of many preexisting technologies such as virtualization and server consolidation, among others.

The term virtualization refers to the abstraction of compute resources (CPU, memory, I/O) from the applications, aiming to improve sharing and utilization of computer systems. One immediate benefit of virtualization is the option to run multiple operating systems and software stacks on a single physical platform.

Server consolidation is the process of gathering several Virtual Machines (VMs) into a single physical server. It is often used by Data Centers to increase resource utilization and reduce electric power consumption costs [11]. For example, consider a set of VMs $\{vm_1, vm_2, vm_3, vm_4\}$ and a set of hosts $\{h_1, h_2, h_3\}$, each of them with a quad-core processor, where each processor is capable of executing one VM. A power efficient allocation schedule could initially assign all the VMs to the same host in such a way that the other hosts could be put in the power-saving state or turned off. A possible solution to this problem would be, therefore, to pack the maximum workload in the smallest number of servers, keeping each resource (CPU, disk, network, among others) on every server at 100% utilization and put the idle servers in power-saving state.

The consolidation process can be performed in a single step using the peak load demands, known as static consolidation, or in a dynamic manner, by reevaluating periodically the workload demand in each VM. In static consolidation, VMs stay in the same physical server during their whole lifetime. The utilization of the peak load demand should ensure that the VM does not overload. However, in a dynamic environment with different access patterns, one or more VMes can become idle, resulting in an inefficient power allocation.Dynamic consolidation aims to tackle this problem by taking into account the current workload demands. Dynamic consolidation may require migrating VMs between physical servers in order to [7]: (i) pull out physical servers from an overload state or (ii) turn off a physical server when it is idle or when the VMs mapped to it can be moved to another physical server.

Server consolidation in a Cloud Computing environment presents some additional difficulties since the Cloud must also provide reliable QoS, normally defined in terms of Service Level Agreements (SLA). An SLA describes characteristics such as maximum throughput and minimum response time, that must be delivered by the deployed systems. If an SLA is violated, economical penalties usually apply.

3 Energy-Aware Computing

Cloud Computing solutions may have a potential impact on green house gas (GHC), which include CO_2 emissions. Saving energy of a Data Center with acceptable QoS requirements is an economical incentive for data center operators, as well as a significant contribution to the environment. This requires the design of energy-aware solutions. Energy-awareness can be characterized by taking into account the amount of resources and QoS requirements required by the applications and also the energy requirements along their life cycle [9].

Several indicators have been introduced to measure Data Center efficiency under the vision of achieving economical, environmental and technological sustainability [13]. In this context, Green Performance Indicators (GPI) are defined as the driving policies for data collection and analysis related do energy consumption. The idea of GPIs is interesting because it can be adapted as part of Service Level Agreements (SLA), where requirements about energy efficiency versus the expected quality of services are specified and need to be satisfied. The GPIs are classified in four clusters (IT Resource Usage, Application Lifecycle, Energy Impact and Organizational). In this work, we consider only the IT Resource Usage and the Energy Impact GPI clusters.

The IT Resource Usage GPIs characterize the energy consumption of an application as a function of the energy consumed by its resources. Examples of metrics are *CPU usage*, *Memory usage* and *I/O activity*.

The Energy Impact GPIs describe the impact of Data Centers and applications on the environment, considering power supply, consumed materials, emissions, and other energy related factors. The most important Energy Impact GPI metrics are: a) application performance indicators, which measure the *energy consumption per computing unit*, using typically FLOPS/kWh or Number of Transactions/kWh; b) *Data Center Infrastructure Efficiency (DCiE)*, which is used to determine the energy efficiency of a Data Center as a whole; and c) *Compute Power Efficiency (CPE)*, which computes the data center power. In this metric, the power consumed by idle servers is computed as overhead.

4 Design of the Multi-Agent Consolidation Mechanism

The main goal of our approach is to meet the QoS requirements of the applications, while keeping the power consumption of the Data Centers below a given energy threshold defined by a third party agent. To achieve this goal, we propose a Multi-Agent strategy to negotiate the resource allocations among Clouds.

We consider a federated Cloud environment with four distinct agents: Cloud Service Provider (CSP), Cloud User (CLU), Energy Power Provider (EPP) and Carbon Emissions Agency Regulator (CEAR) as shown in Figure 4(a). In our design, the CEAR determines the amount of carbon emissions that both the CSP and the EPP can emit in a period of time.

In each Data Center, there is one coordinator responsible for monitoring data center metrics, negotiating with the other agents. There are also sensors to monitor energy consumption, resource usage and SLA violation as shown in Figure 4(b). The scenario proposed is a set of Data Centers composed of a set of Virtual Machines, which are mapped to a set of physical servers that are interconnected and deployed in a hybrid cloud model.

Let R $\{r_1, r_2, \cdots, r_n\}$ be the set of resources in Data Center i with a capacity c_i^k, where $k \in R$. The Energy Consumption for the Data Center (E_i) is defined as $E_i = (p_{max} - p_{min}) * U_i + p_{min}$ [14], where p_{max} is the power consumption at the peak load, p_{min} is the minimum power consumption in active mode, and U is the resource utilization of Data Center i as defined in $U_i = \sum_{j=1}^{n} u_{i,j}$ [14], where $u_{i,j}$ is the resource usage of resource j in Data Center i.

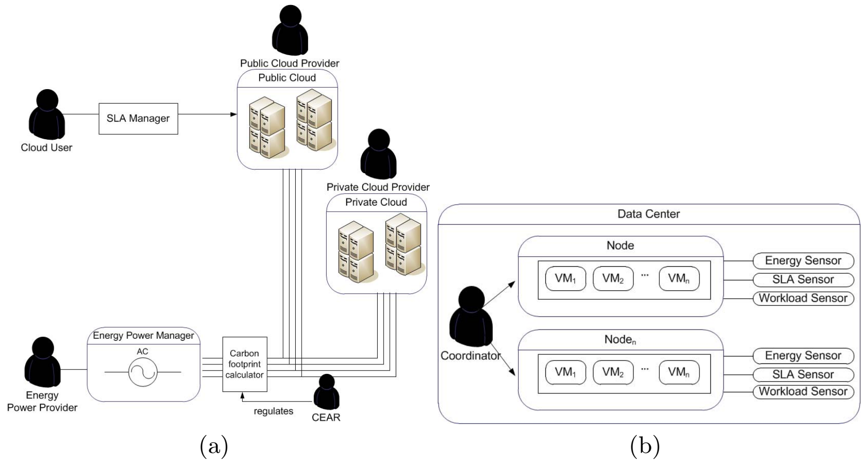


Fig. 1. (a) Agents of the cloud market (b) Detail view of a Data Center

Resources R are managed by the Cloud Service Providers (CSP), which are used by a set of Cloud Users (CLU). The energy power is provided by an Energy Power Provider (EPP). The relation between the CSPs and a CLU is determined by a set of QoS metrics described in SLAs. The Data Center is subject to an energy consumption threshold agreed among the CSP, the EPP and CEAR. When the energy consumption threshold is violated, this implies in additional costs. To calculate the carbon footprint of the CSP and the EPP, the CEAR uses the following metrics: *CPU usage, Memory usage, I/O activity* and *CPE* (Section 3).

Let T represent the set of tasks to be executed in a resource r_i which is subject to a set of QoS constraints. The following steps are executed:

1. When a task t_i is submitted, the Cloud Provider calculates the price of t_i's execution.
2. The Cloud Provider tries to place t_i in an appropriate resource, using consolidation techniques to reduce the number of physical servers.
3. If the Cloud Provider does not have enough available resources or the energy threshold will be violated, the Cloud Provider first contacts another Cloud Provider and negotiates with it the execution of this task. In this case, the price of this execution (P_t) is defined as shown in Equation (1).

$$P_t = E_t + \epsilon_t + \lambda_t \tag{1}$$

   where ϵ_t is the cost Energy Impact of task t on the environment, and λ_t is the cost to transfer task t to another Cloud Provider.
4. If it does not succeed, the Cloud Provider tries to consolidate its VMs considering the tasks SLAs.
5. If not possible, it tries to negotiate the energy threshold with the CEAR and with the EPP agents.

6. If all negotiations fail, the Cloud Provider finds the SLA whose violation implies in lower cost and execute the task. In this case, the price to execute the tasks is defined as shown in Equation (2).

$$V_t = P_t + \gamma + \delta \tag{2}$$

where γ is the cost of violate the QoS requirements of other tasks and δ is the cost associated with energy consumption violation.

To control task allocation, each Cloud Provider has a matrix representing tasks $t_i \in T$, virtual machines $vm_j \in VM$ and physical servers $r_z \in R$, where: 1 represents that t_i is allocated at vm_j in resource r_z; 0 indicates that t_i can be allocated at vm_j; and -1 represents that this allocation is impossible.

In order to illustrate our strategy, consider a federated Cloud with 2 Data Centers (DC1 and DC2) and a user that contracts DC1 to execute his applications. Consider that DC1 is overloaded and that the QoS requirement described in the SLAs is response time. In this scenario, when the user submits tasks to execute, the DC1 Cloud Provider first tries to execute them locally, considering energy consumption and the available resources. Since DC1 is overloaded, its Cloud Provider contacts DC2 and negotiates with it the execution of the tasks. If DC2 accepts it, the cost of the tasks execution is calculated with Equation (1). If DC2 does not accept, then DC1 tries to consolidate its virtual machines and, if not possible, it tries to negotiate the energy threshold with the CEAR and the EPP agents. If all negotiations fail, then DC1 finds the SLA whose violations implies in lower cost and terminates the execution of its associated task. In this case, the cost to execute the tasks is calculated with Equation (2).

5 Experimental Results

In this section, we present the evaluation of the strategy proposed in Section 4. We used CloudSim [3], which is a well-established Cloud simulator that has been used in many previous works [16], [18], among others. It is a simulation toolkit that enables modeling and simulation of Clouds and application provisioning environments, with support to Data Centers, Virtual Machines and resource provisioning policies.

Since we are dealing with federated environments, we extended CloudSim by adding four classes (*CloudEnergyReg, DCEnergySensor, FedPowerVMAllocPolicy, CustomerDCBroker*) to it, as well as isolation of queue events and support for concurrent execution.

The *CloudEnergyReg* class represents the behavior of the CEAR agent. This agent communicates with the Data Center cloud coordinator to inform the energy consumption threshold. The *DCEnergySensor* class implements the Sensor interface that monitors the energy consumption of the Data Center and informs the coordinator. When the energy consumption is close to the limit, this sensor creates an event and notifies the coordinator, that can take actions. The *FedPowerVmAllocPolicy* class extends the *VmAllocPolicy* class to implement the proposed federated server consolidation mechanism Virtual Machine allocation to hosts that can belong to different Data Centers. Finally, the *CustomerDCBroker* class models the QoS requirements customer behavior, negotiates with the cloud coordinator and requests computations.

5.1 Evaluation in Two Scenarios

In order to evaluate the effectiveness of our Federated Application Provisioning strategy (FAP), we used a simulation setup that is similar to the one used in [3]. The simulation environment included 2 Data Centers (DC1 and DC2), with 100 hosts each. These hosts had one CPU core with 1000 MIPS, 2GB of RAM and 1TB of storage. The workload model included provisioning for 400 VMs, where each VM requested one CPU core, 256 MB of RAM and 1GB of storage. The CPU utilization distribution was set to the Poisson distribution, where task required 150 MIPS or 10 minutes to complete execution. We assumed CPU utilization of 20, 40, 60, 80 and 100% and a global energy consumption threshold of 3 kWh of energy per data center. Initially, the provisioner allocates as many as possible virtual machines on a single host, without violating any constraint of the host. The SLA was defined in terms of response time (10 minutes).

In the first evaluation scenario, there are two Data Centers (DC1 and DC2) and tasks are always submitted to DC1. If DC1 becomes overloaded, VMs are migrated from DC1 to DC2. The simulation was repeated 10 times and the mean values for energy consumption without our mechanism using only DC1 (trivial), and with our Federated Application Provision strategy (FAP) mechanism are presented in Figures 2 (a), (b) and (c).

Figure 2(a) shows that the proposed provision technique is able to reduce the total power consumption of the Data Centers, without SLA violation. In this case, an average reduction of 53.37% in power consumption was achieved since DC1 consumed more than 9kWh with the trivial approach and no more than 4.9 kWh was consumed by both Data Centers with our approach (2.92 kWh for DC1 and 1.98 kWh for DC2). In order to achieve this, DC1 tried first to maximize the usage of its resources and to consume the limit of energy power without violating the SLAs. DC2 was used only when DC1 was overloaded, if DC1 was in the imminence of SLA violation or when the energy consumption was close to the limit.

Figure 2(b) presents the number of VM migrations when our mechanism is used. It can be seen that the number of migrations decreases as the threshold of CPU usage increases. This result was expected since with more CPU capacity, the allocation policy tends to use it and allocate more VMs in the same physical machine. In Figure 2(c), we measured the wallclock time needed to execute 400 tasks, with our mechanism (FAP) and without our mechanism (trivial). It can be seen that FAP increases the whole execution time. This occurs because of the overhead caused by the VMs migrations between data centers, and the negotiations between the CLU and the CSP agents. Nevertheless, this increase in less than 22%, since the wallclock execution time without and with the mechanism is 21.5 min and 27.4 min, respectively, for 100% CPU utilization. We consider

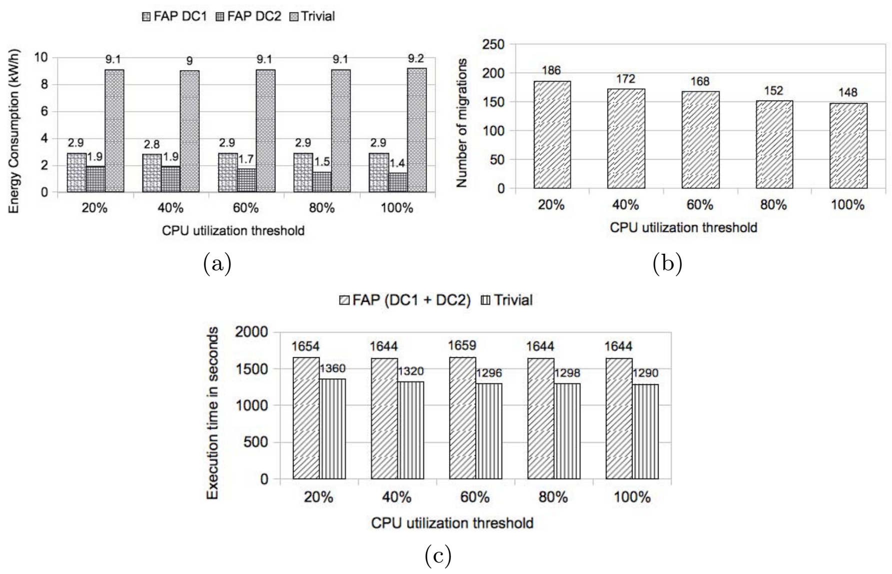


Fig. 2. Case Study 1:(a) Total energy consumption by data centers (b) Number of VM migrations from DC1 to DC2 for the FAP mechanism (c) Execution time of tasks

that this increase in the execution time is very low and it is compensated by the reduction in the power consumption (Figure 2).

In the second scenario, we consider two users, with distinct SLAs and each user makes 400 task execution requests to a different data center (DC1 and DC2). Our goal is to observe the rate of SLA violation when the workload of both Data Centers is high (Figures 3 (a), (b), (c) and (d)).

In Figure 3(a), we can see that, even in a scenario with overloaded Data Centers, our mechanism is able to maintain the power consumption below the threshold (3 kWh) for each Data Center. With the CPU utilization threshold of 80%, the power consumption decreased from 9.13 kWh to 5.65 kWh (DC1 + DC2), reaching 38.2% of reduction in power consumption.

The number of SLA violations with two overloaded Data Centers was lower than the one obtained with one overloaded Data Center (DC1) (Figure 3(d)). With the CPU utilization threshold of 80%, the SLA violation decreased from 43.94% (DC1) to 31.48% (DC1 + DC2), reaching 12.46% of reduction in SLA violations. This shows the appropriateness of VM migrations between different Data Centers in an overloaded scenario.

6 Related Work

Table 1 summarizes the main characteristics of 6 papers that propose server consolidation strategies for distributed environments.

As can be seen in this Table, three approaches [4, 6, 5] use multi-agent systems to reduce power consumption and costs. One of them is targeted to Clouds and

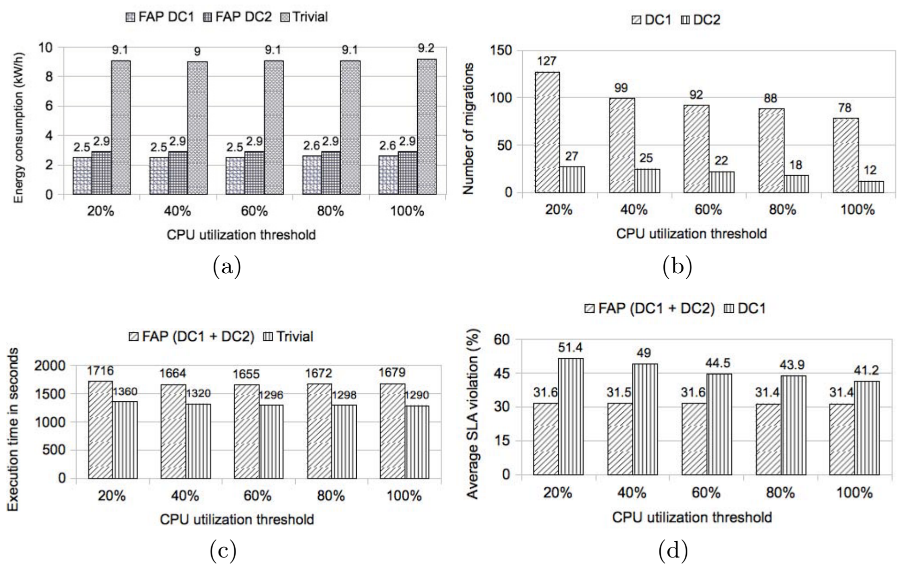


Fig. 3. Case Study 2: (a) Total energy consumption by data centers (b) Number of VM migrations (c) Execution time of tasks with 2 overloaded Data Centers (d) Average SLA Violation for the federated approach

Table 1. Comparative summary of Cloud server consolidation strategies

Paper	Target	Power-Aware	Federated	Multi-agent	Migration	SLA
[4]	Cloud	No	No	Yes	No	No
[6]	Cluster	No	No	Yes	No	No
[5]	Cluster	Yes	No	Yes	No	Yes
[12]	Cluster	Yes	No	No	No	Yes
[14]	Cloud	Yes	No	No	No	No
[3]	Cloud	Yes	No	No	Same DC	Yes
This work	Cloud	Yes	Yes	Yes	Among DCs	Yes

two execute in cluster computing environments. Three approaches [12, 14, 3] reduce power consumption in cloud computing considering SLAs. None of the analyzed proposals consider federation cloud environment nor VM migration among data centers.

7 Final Consideration and Future Work

In this paper, we proposed and evaluated a server consolidation approach for efficient power management in virtualized data centers taking into account energy consumption, and QoS requirements. The results obtained in the CloudSim [3] simulator, with 2 data centers and 400 simultaneous virtual machines show that very good energy consumption savings are obtained with our approach, while meeting the QoS requirements. The best gain (53.57%) was obtained when we have one overloaded data center. In this case, we were able to reduce the energy consumption from 9.13 kW/h with the trivial approach to 4.9 kW/h with an increase of less than 22% in the execution time.

As future work, we intend to investigate formal models for power-aware resource allocation in Cloud Computing Systems and propose extensions that take more parameters into consideration.

Acknowledgment. This work is partially supported by FAPDF/Brazil and CNPq/Brazil.

References

[1] Beloglazov, A., et al.: A taxonomy and survey of energy-efficient data centers and cloud computing systems. Advances in Computers 82, 49 (2010)
[2] Buyya, R., et al.: Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. 25, 599–616 (2009)
[3] Calheiros, R.N., et al.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience 41(1), 23–50 (2011)
[4] Chen, Y., Yeh, H.: An implementation of the multiagent system for market-based cloud resource allocation. J. Computing 2(11), 27–33 (2010)
[5] Das, R., et al.: Autonomic multi-agent management of power and performance in data centers. In: AAMAS, pp. 107–114 (2008)
[6] Ejarque, J., Sirvent, R., Badia, R.M.: A multi-agent approach for semantic resource allocation. In: CloudCom, pp. 335–342 (December 2010)
[7] Ferreto, T.C., et al.: Server consolidation with migration control for virtualized data centers. Future Gener. Comput. Syst. 27(8), 1027–1034 (2011)
[8] Fan, X., Weber, W., Barroso, L.A.: Power provisioning for a warehouse-sized computer. In: ISCA, pp. 13–23. ACM (2007)
[9] Ferreira, A., et al.: Energy-aware design of service-based applications. In: ISOCC, pp. 99–114 (2009)
[10] Hayes, B.: Cloud computing. Commun. ACM 51, 9–11 (2008)
[11] Hoelzle, U., Barroso, L.A.: The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. M. C. Pub. (2009)
[12] Kim, K.H., et al.: Sla-based scheduling of bag-of-tasks applications on power-aware cluster systems. IEICE Trans. on Inf. Sys. E93-D(12), 3194–3201 (2010)
[13] Kipp, A., Jiang, T., Fugini, M., Salomie, I.: Layered green performance indicators. Future Gener. Comput. Syst. 28(2), 478–489 (2012)
[14] Lee, Y.C., Zomaya, A.Y.: Energy efficient utilization of resources in cloud computing systems. The Journal of Super Computing, 1–13 (2010)
[15] Lefurgy, C., Wang, X., Ware, M.: Server-level power control. In: Fourth Autonomic Computing. IEEE Computer Society (2007)
[16] Shi, Y., Jiang, X., Ye, K.: An energy-efficient scheme for cloud resource provisioning based on cloudsim. In: IEEE ICCC, pp. 595–599 (2011)
[17] Vaquero, L.M., et al.: A break in the clouds: towards a cloud definition. SIGCOMM Comput. Commun. Rev. 39, 50–55 (2008)
[18] Zhang, Q., Gürses, E., Boutaba, R., Xiao, J.: Dynamic resource allocation for spot markets in clouds. In: Hot-ICE 2011, pp. 1–6 (2011)

Specifying Cloud Application Requirements: An Ontological Approach

Yih Leong Sun, Terence Harmer, and Alan Stewart

Queen's University of Belfast,
University Road, Belfast, BT7 1NN, UK
{ysun05,t.harmer,a.stewart}@qub.ac.uk
http://www.qub.ac.uk

Abstract. Increasingly business organisations are deploying service applications onto cloud infrastructures. Given the available range of infrastructure providers and products, it is a challenging task to select the most appropriate set of cloud resources for a given application. Cloud providers offer resources in various formats using different pricing structures. There is a mismatch between the terminology used to specify an application's requirements and that used to describe provider resources. In this paper, a resource allocation approach based on mapping application requirements onto cloud infrastructure products is proposed. Two domain-specific ontologies for media transcoding and financial services are used to illustrate how application requirements can be modelled. It is then shown how requirements can be mapped onto a general ontological description of cloud resources. The resource ontology is provider-agnostic and provides a framework for searching the cloud market for a set of products that meet an application's requirements.

Keywords: cloud computing, cloud programming model, ontology.

1 Introduction

Many organisations are utilising cloud infrastructures as a flexible and cost-effective platform on which to execute business applications. Increasingly providers offer infrastructure resources or services in the cloud market for hosting cloud-based applications. Different providers offer different ways of leasing their cloud resources using different pricing structures. In order to achieve a high degree of business continuity, it is important that cloud-based applications can operate even under adverse conditions. In the event of service interruptions caused by a provider's resources malfunctioning, the organisation should have the option to migrate their applications elsewhere and avoid a vendor lock-in situation. From the application developer's point of view, finding an appropriate set of infrastructure resources that meet an application's requirements in a multi-provider cloud environment is a challenging task because of the range of products available as well as the dynamic nature of the market.

Typically developers analyse an application's requirements and then select a suitable set of infrastructure resources on which to execute the application.

Consequently, there is a need for a programming model which facilitates the specification and construction of cloud-based infrastructures. Such a programming model should enable application developers to build cloud-based infrastructures easily and rapidly. In order to develop such a programming model, it is desirable to have a means for mapping application requirements onto infrastructure products (offered by multiple providers). Many different frameworks have been proposed for discovering and utilising cloud resources. Typically cloud requirements are analysed from the provider's point of view, based on the resource capabilities offered by the provider. However, there is a mismatch between the perspective of application developers and the low-level details supplied in a typical resource specification. Currently, there are no suitable mechanisms for describing requirements from an application's viewpoint. This paper proposes an application-centric, multi-layer ontology as a way of describing application requirements in the cloud context. The ontology allows application developers to formulate high-level domain-specific application requirements and subsequently to use these descriptions to search for the most appropriate set of resources in a multi-provider cloud environment. There is potential for application developers to utilise the proposed ontology so as to automate the resource discovery process.

This paper is organised as follow. Section 2 provides a brief summary of a provider-agnostic programming model. Section 3 discusses related research and defines a multi-layer model. Section 4 describes ontology translation using two examples while conclusion are drawn in Section 5.

2 A Provider-Agnostic Programming Model

An overview of a programming model for selecting and utilising cloud-based infrastructures in a multi-provider cloud environment is given below.

The model is wrapped in a provider-agnostic API [7] (see Figure 1) and incorporates a set of cloud providers that make up the cloud market. The selection of a particular provider depends on user preferences and provider's financial models. This set of providers is associated with a pool of available resources. In practice, an application is mapped onto a set of resources from the resource pool that meets the application's requirements. Multiple types of suitable resources may be discovered. An initial result set can be filtered using heuristics in order to find the most suitable set of resources for a particular application. After a best fit resource is identified, it can be reserved for future use. When resources are needed, they are instantiated and user's application is deployed. After the application finishes, the underlying resources can be discarded.

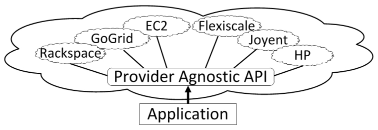


Fig. 1. A provider-agnostic programming model

This model allows developers to acquire infrastructure resources without knowledge of the internal implementation details of the underlying providers. This provides an abstract view of infrastructure resources and insulates the application from API changes arises from the underlying providers. Applications can be deployed and scaled according to system constraints, within a given budget, and have portability across multiple providers. This paper proposes the use of ontologies to formulate application requirements. These requirements are subsequently used in the resource mapping process of the proposed model.

3 A Multi-layer Ontological Model

An *ontology* is a formal description of the entities and the relationships between them within a particular knowledge domain. Ontologies have been used to describe cloud resources elsewhere. Most cloud-related ontologies are resource-centric and give definitions from the perspective of the capabilities of a particular resource provider.

Bernstein et al [2] proposes an ontology-based catalog which describes the resource capabilities offered by cloud providers, such as CPU, storage and compliance capabilities. Reservoir [4] proposes a service specification mechanism which includes VM details, application and deployment setings. Mosaic [8] introduces an ontology to describe cloud resources with a set of functional and non-functional properties. LoM2HiS [3] proposes a framework for mapping low-level resource metrics to high-level SLA parameters which focuses on hardware or network attributes. This paper proposes an application-centric multi-layer ontology that focuses on user requirements rather than just the cloud resources.

In this paper, a three-layer model is used to describe the process of mapping application requirements onto cloud resources (see Figure 2). The top layer uses a *domain-specific ontology* to express high-level application requirements semantically using application specific terminology. Two examples of such domain-specific requirements are: (i) application data must be processed within UK in order to be compliant with the UK Data Protection Act; (ii) media file must be transcoded into *wmv* format and played on *windows phone* device.

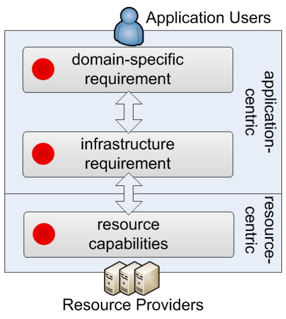


Fig. 2. A multi-layer ontology model

The middle layer of the proposed model is an *infrastructure requirement ontology*. This layer describes provider-agnostic infrastructure constraints that are needed to deliver the application requirements. In going from the top to the middle layer, high-level *domain-specific* requirements are mapped to *infrastructure level* requirements. The bottom layer in the model is the *resource* layer which specifies the resource capabilities offered by various cloud providers. This paper focuses on the *domain-specific* and *infrastructure requirement* layer. The *resource* layer has been widely investigated elsewhere [2,8].

3.1 Domain-Specific Ontology

A domain-specific ontology can be used to capture high-level application constraints. The ontological layer is application-centric, focused on user needs and expressed using domain specific terminology. Two examples of domain-specific ontologies are given in order to illustrate the model.

Media transcoding is the process of converting media files (video or audio) from one format to another. Transcoding is computational intensive and requires high storage and fast bandwidth [6,10]. Often users impose a budget for the provisioning of transcoding infrastructure. Consider a media company that broadcasts a series of animation videos. The video sources use *avi* format and are made available 5 hours before the broadcast schedule. For certain applications, these need to be transcoded into *windows media* format at a frame rate of *30 frame per second* and delivered to *Windows Phone* devices via *http streaming*. The company has a budget of *£100* for the transcoding operations. The application's requirements may be specified in a high-level notation as:

Video conversion : AVI to Windows Media
Mobile encoding : Windows Phone
Delivery deadline : 9am next morning
Encoding features : frame-rate conversion; http adaptive streaming
Budget : £100

More generally, the following domain-specific ontology is used for specifying media transcoding requirements:

Budget requirements specify monetary constraints for running a transcoding task. These can be specified as the maximum amount that a user is prepared to spend per day or per hour.

Format requirements specify the container format of the source and transcoded media; for example, transcoding a video from *avi* format to *flv* format.

Codec requirements are the audio or video codecs of the media; for example, *mpeg4*, *h264*, *mp3*, *aac.*

Device requirements refer to the destination devices that the transcoded media will be played on; for example, *iPhone*, *Windows Phone*, *PC.*

Processing Filter requirements are advanced video and audio filters, including both pre-processing and post-processing filters; for example, frame rate conversion, de-interlacing, watermarking, audio resampling, etc.

Content Sensitivity requirements refer to the sensitivity of the media content. A sensitive media content is associated with high security capabilities; for example, secure transfer channel or encryption.

Output Storage requirements refer to the storage destination of transcoded media; for example, cloud storage, such as Amazon S3.

Delivery Time requirements refer to the time when the transcoded media will be ready for delivery (i.e. when the transcoded media is needed for use).

Delivery Channel requirements refer to how the transcoded media is delivered; for example, to be delivered as a downloaded file, HTTP live streaming over the internet, or mobile streaming over 3G network.

The proposed transcoding ontology adopts the terminology used by the media industry. Many media people would be familiar with such high-level specifications rather than the details of hardware or software resources that need to be deployed to perform transcoding tasks. Moreover, there are many different solutions which can be used to implement the same transcoding task: developers can use a *basic resource* facility and additionally install the transcoding software themselves; alternatively, developers could use a *packaged resource* with pre-installed software, such as the AWS resource pre-built with Wowza Transcoder [1]. By using the proposed model, media users can specify an application's requirements using appropriate terminology; this in turn allow developers to search for the most suitable set of resources. It is up to developers to define a mapping algorithm which relates the domain-specific ontology to the infrastructure constraints. Historical database can be used to facilitate the mapping process by providing actual performance data. For example, in [6] several jobs are run concurrently on a multi-core resource in order to meet a delivery deadline.

Financial service sector is a challenging domain for infrastructure deployment [9]. It is highly regulated and demands high availability of resources. One way to increase infrastructure resilience is to deploy mirror infrastructures located in different geographical locations. Consider a financial company which needs to revalue their customers' portfolios on a daily basis [9], the company must comply with the UK Data Protection Act. The application service must be made available 12 hours a day. Extra application services are required if several large portfolios are to be revalued at the same time. These requirements can be specified as:

Compliance: UK Data Protection Act 1998
Service hours required: 8am to 8pm, weekdays
Service performance: High response time; high availability;
high scalability; high disaster recovery
Portfolio valuation software: Supplied by users

A generic domain-specific ontology for financial services application is proposed as follows:

Budget requirements are the monetary constraints for running the financial application, typically specified as a maximum spend per month.

Compliance requirements are the rules, regulations, legislation or laws that need to be comformed with in the financial service domain, such as UK Data Protection Act 1998.

Security requirements refer to how financial data is accessed and transferred.

Data requirements refer to the quality and integrity of the financial data. For example, financial data must be verified and audited by a third-party data verification service.

Performance requirements indicates the uptime requirements or guaranteed response time for a certain time range in a day, for example, 99.99% uptime and 100ms response time between 8am to 8pm, Monday to Friday.

Availability requirements refer to the capabilities of a financial application to continue operate without service interruption in the event of component failures, for example, availability level can be categorised as high, medium or low; a high availability requirements indicates that a mirror infrastructure must be provisioned in different geographical locations.

Scalability requirements refer to how flexible the infrastructure can grow or shrink when demands fluctuate.

Financial companies require quick turn-around time to deploy applications in order to remain competitive in the fast-paced financial market. The proposed ontologies provide an easy and quick mechanism for financial users to specify high-level requirements using appropriate terminology. Developers can translate these high-level specifications to lower level infrastructure constraints. For example, *low availability* means that infrastructure replication is not required, whereas, *high availability* means that an application must be deployed on a mirror infrastructure located in different geographical locations.

3.2 Infrastructure Requirement Ontology

In the infrastructure requirement ontology, a **requirement** specifies the capabilities or qualities that are necessary (or desired) for an infrastructure. **Infrastructure requirements** are divided into different categories (see Figure 3):

Cost requirement is the budget for deploying cloud infrastructure.

Performance requirement refers to effectiveness and quality of the infrastructure. **Network latency performance** is the delay incurred in the processing of data across the network; **bandwidth performance** is the speed of the network including **incoming** and **outgoing bandwidths**.

Resource requirement refers to the specification of individual resources (hardware, software or operating system). Four categories are identified: **hosting environment** defines the operating system requirement of the host, such as Windows 7; **hardware capability** refers to the hardware components, such as CPU, RAM, storage space; **software stack** indicates the list of software or services that need to be installed on a resource.

Geographical requirement refers to location of resources (including data).

Compliance code requirement refers to regulatory, industry or security standard that the infrastructure needs to comply with, such as ISO27002.

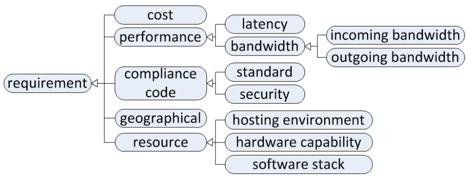


Fig. 3. Hierarchy of infrastructure requirements

A **requirement** can be either **hard requirement** or **soft requirement**. A **hard requirement** is a compulsory requirement which remains invariant over the application's lifecyle – an example might be legislation regulation; a **soft requirement** is a desired requirement which can change or be re-prioritised – for example, it might be budget or performance related. This concept is represented using the **hasRequirementType** property and the **requirementType** emumeration property. A **priority level** data property is defined to indicate the importancy of a **requirement**. This property can be used to measure and calculate weightings during requirement prioritisation and resource filtering process. **Requirements** may depend on each other. For example, UK Data Protection Act (**compliance requirement**) indicates that no data can be processed or stored outside the UK boundary. This translates to a dependency relationship on **geographical requirement**. Figure 4 illustrates the **requirement** ontology.

Infrastructure and **requirement** are core entities in the infrastructure requirement ontology. Every **infrastructure** has at least one **site**. A **site** has one or more **resource groups**. A **resource group** is a set of **resources**. **Requirements** can be applied at different levels of the infrastructure layout: **infrastructure**, **site**, **resource group** or **resource** level. The relationship between **infrastructure** and **requirement** is expressed using the **hasRequirement** property (see Figure 5).

A **restriction** class is defined to identify the conditions or constraints associated with a **requirement**. Each **requirement** has at least one **restriction** which is expressed using **isConstrainedBy** property (see Figure 4).

Cost requirement is constrained by **cost restriction**. A **cost restriction** can be a **total cost** or it can be divided into **compute cost**, **software cost**, **storage cost**, or **bandwidth cost**. Each **cost restriction** is associated with **cost frequency** (per hour, per day) and **cost money** (amount, currency).

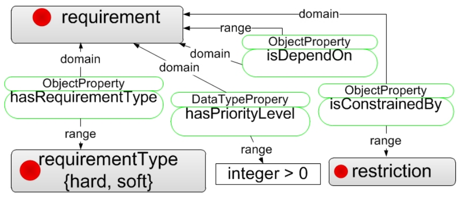


Fig. 4. Requirement

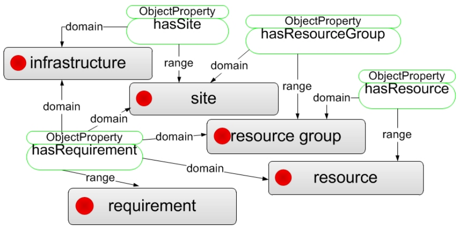


Fig. 5. Infrastructure and requirement

Performance requirement is constrained by performance-related restriction. **Latency performance** is constrained by **latency restriction** (expressed via **hasLatency** property). **Bandwidth performance** is constrained by **bandwidth restriction**. **Bandwidth restriction** indicates minimum amount of bandwidth required using the **hasMinBandwidth** property.

Resource requirement is constrained by resource-related restriction. **Hosting environment** is constrained by **operating system restriction** which indicates the operating system types (via **hasOS** property). **Hardware capability** is constrained by **hardware restrictions**, such as **cpu core restriction** (via **hasMinCPUCore** property), **cpu speed restriction** (via **hasMinCPUSpeed** property), **cpu architecture restrictioin** (via **hasCPUArchitecture** property), **RAM restriction** (via **hasMinMemory** property), and **storage space restriction** (via **hasMinStorageSpace** property). **Software stack** is constrained by **software restriction** which indicates the list of softwares or services that need to be installed on the resource (via **hasSoftware** property).

Geographical requirement is constrained by **location restriction**. **Location restriction** is associated with **hasLocation** property that indicates the location (country or data center location).

Compliance code requirement is constrained by **compliance restriction**, which can be **standard code restriction** – contains the standard's code format or **regulatory restriction** – contains the name of regulation.

3.3 Resource Ontology

The resource ontology, the bottom layer of the proposed model, defines the properties of the resources offered by cloud providers. This layer has been widely investigated elsewhere – see [2] and [8]. These existing ontologies can be applied as the resource ontology in the proposed model. The mapping of infrastructure requirements to the resource ontology can be achieved by using query language [5] – this topic is outside the scope of this paper.

4 Translation from Domain-Specific Ontology

Here consideration is given as to how a domain-specific ontology can be translated to the infrastructure requirements ontology.

For the media transcoding example, application developers need to provision an infrastructure which fulfils the transcoding requirements as well as balances the budget and delivery time constraints. Transcoding requirements, such as video format conversion, frame rate conversion or http streaming, indicate the features or capabilities of a transcoding software that need to provide. Particular software, such as FFmpeg or Rhozet, can be used to perform the transcoding task. However, each software has different system requirements. For example, Rhozet must be run on Windows operating system, whereas FFmpeg can be run on Linux. Using the proposed multi-layer ontology model, domain-specific transcoding requirements can be translated into software requirements, where each software has associated operating system or hardware dependancy requirements. Moreover, tight delivery deadline requirement may necessitated the use of high-cpu resources. Figure 6 illustrates how the transcoding requirements are translated into infrastructure requirements.

Examples of the ontology relationships between domain-specific and infrastructure layer for the media transcoding application are given as follow: ***Format, codec, device, processing filters*** and ***delivery channel*** *isDependOn* **software stack** which indicates the transcoding software's capabilities or features; ***delivery time*** *isDependOn* **network latency**, **bandwidth** and **hardware capability** as it requires fast bandwidth and high cpu for fast processing.

For the financial services example, the UK Data Protection Act regulatory requirement indicates that the infrastructure resources being provisioned must be located within UK. Two identical mirror infrastructure must be provisioned at different geographical location in order to meet the high availability and high disaster recovery requirements. High bandwidth usage is required as the application service needs to utilise stock market values. The demand of high response time requires high-cpu and high-memory resources for faster computation. The formulation of such requirements are illustrated in Figure 7.

Examples of the ontology relationships for the financial services application are given below: ***Compliance*** *isDependOn* **geographical** because a 'UK Data Protection Act' indicates that no data can be processed outside the UK boundary; ***data*** *isDependOn* **software stack** as it requires third party verification services; ***high availability*** and ***high scalability*** are complex requirements and depend on how infrastructure resources are provisioned. The concepts of **infrastructure**, **site**, **resource group** and **resource** are used to indicate that different geographical sites must be provisioned, and each site *hasRequirement* **latency**, **bandwidth** and **geographical**.

The middle layer of the proposed model serves as an agent that translate the domain-specific ontology onto related infrastructure requirements ontology. This provides an abstract view of high-level requirements from the application's perspective. Infrastructure requirements ontology can then be mapped to resource ontology using ontology query language [5].

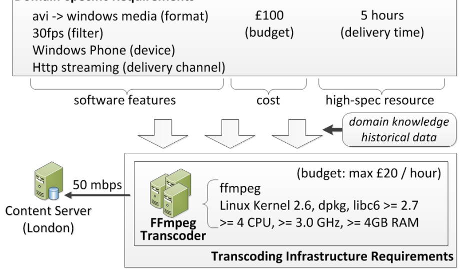

Domain-specific Requirements
avi -> windows media (format)
30fps (filter)
Windows Phone (device)
Http streaming (delivery channel)
£100 (budget)
5 hours (delivery time)
software features
cost
high-spec resource
domain knowledge historical data
50 mbps
Content Server (London)
FFmpeg Transcoder
(budget: max £20 / hour)
ffmpeg
Linux Kernel 2.6, dpkg, libc6 >= 2.7
>= 4 CPU, >= 3.0 GHz, >= 4GB RAM
Transcoding Infrastructure Requirements

Fig. 6. Media transcoding application

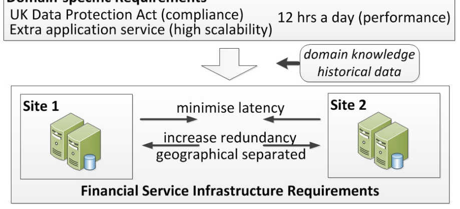

Domain-specific Requirements
UK Data Protection Act (compliance)
Extra application service (high scalability)
12 hrs a day (performance)
domain knowledge historical data
Site 1
Site 2
minimise latency
increase redundancy
geographical separated
Financial Service Infrastructure Requirements

Fig. 7. Financial services application

5 Conclusion and Future Work

The cloud market is developing rapidly with a dynamic environment of providers and products. Searching for suitable resources in such a dynamic environment is challenging. Little attention has been paid to describe a cloud application's requirements at an appropriate level of abstraction. In this paper, an application-centric multi-layer ontology for describing cloud application requirements is proposed. This ontology provides a semantic mechanism for capturing application needs in a language familiar from users' application domains. Two examples are used to illustrate the formulation of application requirements. We hope to enhance the ontology by studying other application domains. We also hope to develop techniques for mapping user requirements into infrastructure constraints. We believe that our approach offers an effective mechanism to compare and select resources from a multi-provider cloud market.

References

1. Wowza Transcoders, `http://www.wowza.com/forums/content.php?23-pre-built-amis-amazon-machine-images` (last accessed: June 30, 2012)
2. Bernstein, D., Vij, D.: Using Semantic Web Ontology for Intercloud Directories and Exchanges. In: International Conference on Internet Computing (2010)
3. Emeakaroha, V.C., Brandic, I., Maurer, M., Dustdar, S.: Low level Metrics to High level SLAs - LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments. In: 2010 International Conference on High Performance Computing and Simulation, HPCS (July 2010)
4. Galán, F., Sampaio, A., Rodero-Merino, L., Loy, I., Gil, V., Vaquero, L.M.: Service specification in cloud environments based on extensions to open standards. In: Proceedings of the Fourth International ICST Conference on Communication System Software and Middleware, COMSWARE 2009. ACM (2009)
5. Haase, P., Motik, B.: A mapping system for the integration of OWL-DL ontologies. In: Proceedings of the First International Workshop on Interoperability of Heterogeneous Information Systems, IHIS 2005, pp. 9–16. ACM (2005)
6. Harmer, T., Wright, P., Cunningham, C., Hawkins, J., Perrott, R.: An application-centric model for cloud management. In: Proceedings of the 2010 6th World Congress on Services, SERVICES 2010. IEEE Computer Society (2010)
7. Harmer, T., Wright, P., Cunningham, C., Perrott, R.: Provider-independent use of the cloud. In: Sips, H., Epema, D., Lin, H.-X. (eds.) Euro-Par 2009. LNCS, vol. 5704, pp. 454–465. Springer, Heidelberg (2009)
8. Moscato, F., Aversa, R., Di Martino, B., Fortis, T., Munteanu, V.: An analysis of mOSAIC ontology for Cloud resources annotation. In: 2011 Federated Conference on Computer Science and Information Systems, FedCSIS (September 2011)
9. Sun, Y.L., Perrott, R., Harmer, T., Cunningham, C., Wright, P.: An SLA Focused Financial Services Infrastructure. In: Proceedings of the 1st International Conference on Cloud Computing Virtualization, Singapore (2010)
10. Wright, P., Harmer, T., Hawkins, J., Sun, Y.L.: A Commodity-Focused Multi-cloud Marketplace Exemplar Application. In: 2011 IEEE International Conference on Cloud Computing (CLOUD) (July 2011)

A Conceptual Framework for Simulating Autonomic Cloud Markets

Simon Caton[1], Ivan Breskovic[2], and Ivona Brandic[2]

[1] Karlsruhe Institute of Technology, Germany
simon.caton@kit.edu
[2] Vienna University of Technology, Austria
{breskovic,ivona}@infosys.tuwien.ac.at

Abstract. One of the major challenges facing the Cloud paradigm is the emergence of suitable economic platforms for the trading of Cloud services. Today, many researchers investigate how specific Cloud market platforms can be conceived and in some cases implemented. However, such endeavours consider only specific types of actors, business models, or Cloud abstractions. We argue that market platforms for the Cloud paradigm cannot (yet) be rigidly defined, and require the ability to progress and evolve with the paradigm. In this paper, we discuss an alternative approach: autonomic markets. Autonomic markets automatically adapt to changed environmental conditions based upon a given concept of "performance". We describe the autonomic MAPE loop in the context of electronic markets and consider the types of a knowledge produced and required for decision making. Finally, we present a conceptual framework for a market simulator, a critical tool for autonomic markets, based upon experiences using the GridSim simulation tool.

Keywords: Cloud Computing, Electronic Markets, Autonomic Computing, Computational Economics, Market Simulation.

1 Introduction

Today electronic marketplaces are challenged by a highly dynamic context: high product variability, unpredictable participant behaviour, and the emergence of new actors as well as actor types. Consequently, market situations that have previously been unimaginable will arise and novel theories and paradigms are needed to facilitate and control them. Examples of new market contexts are in the domains of Smart Grids, the Internet of Services and Cloud Computing as well as Social and Collaborative Environments. Many of these domains are already or will become inherently reliant upon the economic systems represented by electronic markets that can address their allocation problems.

However, a key challenge is that we do not know the most fitting anatomy of an appropriate market platform. Even assuming the existence of an "adequate" platform, a subtle or disruptive change within the domain can mean that the platform no longer satisfies its domain requirements. Today, electronic market platforms are static and not conceived to handle changes in their domain or elements of uncertainty in their architecture. Therefore, we argue that the market engineering process cannot simply extend traditional approaches, but requires a new methodology. In [6], we introduced an alternative: Autonomic Markets; a goal orientated approach for market engineering to enable autonomic adaption.

To evaluate the vision of an autonomic market, we need an experimental platform and it is not possible to simply map existing markets. Therefore, an appropriate research methodology for their study is simulation, as it enables the creation of what if scenarios and the ability to observe how autonomic adaption evolves a market over time. Through simulation, we can implement economic and management models of autonomic markets to access their performance (with respect to goal fulfilment), tractability and feasibility for different adaption strategies. Although real-life markets cannot be mapped directly for an autonomic market, their traces as well as event and trading catalogues can act as input data as a means to drive specific what-if scenarios. In this paper, we reflect upon the lessons learned in [6], in order to create a set of requirements and conceptual architecture for an autonomic market simulation tool.

This paper is structured as follows: Section 2, presents an overview of the autonomic market vision; Section 3 discusses related work; Section 4 presents a case study of GridSim in simulating markets; Section 5 draws upon this case study articulate a conceptual framework for a autonomic market simulation tool; finally Section 6 summarises the paper and discusses future work.

2 Autonomic Markets: An Overview

Our vision of an autonomic market platform is that *institutional forms* and underlying *infrastructures* can be *adapted* at runtime with the goal of improving a given concept of *"market performance"*. Infrastructure adaption refers to modifying the computational infrastucture of the market platform that enables its core functionality. This, for example, includes computational resources, delivery mechanisms, communication channels, security procedures, etc. Institutional adaption relates to modifying market components such as rules of participation, allocation and pricing mechanisms, and tradable artefacts. Market performance is characterised through specific goals that can include market liquidity, immediacy, stability, security, participant welfare, energy efficiency, allocation efficiency, etc. An institutional market form, i.e. an instantiated parameterisation of market components, is what we refer to as a market configuration. Therefore, infrastructure adaption in our context is what we commonly understand as elastic infrastructures in the Cloud paradigm and institutional adaption is a change in one or more parameter settings and hence a change in market configuration.

By making market platforms autonomic, we hope to enable evolution beyond initial design principles by "learning" or adapting towards ideal configurations, possibly with certain levels of oscillation. Through this ability we can begin to explore, analyse and evaluate autonomic market platforms as well as the impact of different market configurations and goals. Autonomic adaption will enable different capabilities in economics such as autonomic (economic) mechanisms, self-regulation, fault tolerance, as well as autonomic market (re)engineering. To make a market autonomic, we propose applying the extended autonomic control loop, i.e. the MAPE-K cycle, to a complex array of parameterisable (hence adaptable) economic components, where each component can be imagined as a traditional managed element within the Autonomic Computing paradigm. We specifically focus on market platforms for the domain of Cloud computing, as it is well defined domain with respect to its requirements on a market platform. A successful implementation of the Cloud computing methodology (i.e. fulfilling its promise of computing as a commodity) is only possible with appropriate methodologies and techniques for the definition and management of Cloud market platforms. We believe that the application of our autonomic market concept can tackle the challenges of the paradigm. In the remainder of this section, we first provide a motivating example for autonomic markets, before briefly describing the application of the MAPE-K loop to an economic system.

2.1 Motivating Example

Consider that a market provider decides that a market goal is the completion of a certain number of trading transactions per unit of time. Observing the market's adherence to this goal is trivial. However, many different events can cause deviations from this goal, some of which may be exogenous (e.g. external outages) and others observable within the platform (e.g. infrastructure bottlenecks, a reduction in the number of active participants, etc.). To remedy deviations with respect to this goal, several different options can be considered depending on the cause of the deviation. Examples are: (1) scaling computing nodes up or the infrastructure as a whole out to increase the number of concurrent trades per unit time, or reduce the time needed to process individual trades; (2) tuning the matching algorithm to reduce the compute time (e.g. applying a heuristic instead of an optimal algorithm); (3) purging the order book(s) to remove redundant data; and (4) tuning allocation mechanism properties (e.g. the maximum number of entries in the order book, the clearing or pricing functions). Moreover, combinations of these options are also valid, as well as more aggressive adaptions such as changing the allocation mechanism for another. Although in this example a simple market goal has been chosen, there are many more complicated goals as well as goal combinations that can have large impacts upon the stability of a market (e.g. goals concerning market liquidity, revenue, efficiency, etc.).

2.2 Applying the Autonomic Loop to a Market Platform

The MAPE-K loop contains five elements: monitoring, adaptation, planning, execution, and a knowledge management components, which we explain below.

Monitoring data is critical for the instrumentation of any form of adaption. In [6], we defined a monitoring methodology for an autonomic marketplace and demonstrated how the performance of a market platform can be measured with respect to a specific set of market performance indicators. This task is performed by monitoring sensors, which gather low-level monitoring data from the market middleware and implementation of the market model. Using the predefined mappings, the measured monitoring data is mapped to the higher level performance indicators, which is then used to assess the market performance.

The main purpose of a **knowledge** component is to store, manage and analyse real-time monitoring data and experiences from previous adaptations. The knowledge gathered in this process can be (1) empirical, i.e. derived from the observations on the market (e.g. infrastructure status, payoffs from previous adaptations, etc.), (2) contextual, i.e. instance-specific (e.g. initial/desired configurations and business models), and (3) institutional, i.e. concerning the economic anatomy of the marketplace (e.g. valid alternative configurations and market rules, constraints and regulations). While empirical knowledge is gathered through monitoring and logging techniques, contextual knowledge is (initially) set by the market administrator. Institutional knowledge is defined partially by the market administrator and partially established based upon contextual knowledge and changes (i.e. evolutions) within the marketplace.

The **analysis** phase is used to analyse mapped data from the monitoring sensors to derive possible actions for market adaptation in order to improve market performance with respect to a set of goals. As already mentioned, there are two main adaption options: the market's infrastructure and its configuration. Finding which of these options is the most fitting is, however, not trivial. Autonomic adaptation of infrastructure properties has already been discussed in a large body of literature (e.g. [1,8,10]). This, however, is not the case for institutional adaption. To facilitate institutional adaption, we need to understand what different market configurations mean for the fulfilment of a given set of goals, which can be achieved through simulation to enable the analysis of what-if scenarios to determine and assess adaption options.

The **planning** phase of the autonomic adaptation cycle includes two important steps. Firstly, it identifies the most suitable adaptation path(s) for the execution of the infrastructure and/or institutional changes by leveraging contextual knowledge. Secondly, as an adaptation path may include more than one market component or steps, it determines the order and timing of the adaptations to be instrumented. This may result in multiple rounds of the adaptation cycle with the goal of observing how single changes impact the market performance and ultimately lead to an iterative adaption process.

The **execution** phase is the execution of an adaption path. In the case of an infrastructure adaption, this relates to an interaction with the resource fabrics through the platform middleware. For institutional adaption, it refers to a check point of the current market status, a new parameterisation of the market configuration, and redeployment (if necessary) of effected market components.

3 Related Work

For positioning our work within the state-of-the-art, we briefly describe existing research on electronic markets and classify it into two categories: (1) applying foundations of autonomic computing to the implementation of electronic markets, and (2) simulating electronic markets for Grid and Cloud environments.

3.1 Autonomic Markets

To enable the flexibility promised by the Grid and Cloud paradigms, market platforms have to be adaptive and sustainable. We argue that this can be achieved with autonomic (self-* [13]) capabilities. Several early works served as groundwork for prototypical implementation of autonomic aspects in complex systems. For example, [16] discuss the need for autonomic capabilities of distributed service systems and briefly outline the application of the self-* capabilities in this context. Today, autonomic computing is used primarily to address technical issues to make systems autonomic, e.g. negotiation protocols to make Grid or Cloud services self-adaptive [5] or consider autonomic service management frameworks without explicitly considering economic methods (e.g. [12,15]).

The idea of applying economic methods and considerations to autonomic systems was initially proposed by [11]. However, current research focuses on specific economic issues and only partially considers the aspects needed to make marketplaces autonomic. For example, [19] proposed a self-organising resource allocation mechanism in dynamic Application Layer Networks (ALNs). They do not, however, consider issues such as the adaptation of the market itself, but rather the optimisation of a small piece: the allocation mechanism. Similarly, [17] propose mechanisms that can adaptively adjust parameters based on past participant behaviour. An example of economically-inspired market infrastructures is provided by [9] who present a self-optimising infrastructure platform for service delivery using economic (congestion-based) pricing. Yet they, consider only the infrastructure, and not the market itself. [4] study the mapping of high-level business objectives to lower level objectives to enable autonomic access optimisation for databases via an economic scheduler.

3.2 Simulating Electronic Marketplaces

Simulation of electronic markets for Grid and Cloud computing has been discussed in several large research projects, including SORMA [19], GridEcon [18] and 4CaaSt [14]. [19] developed a market simulator to compare centralised and decentralised service allocation mechanisms in market scenarios according to a defined set of metrics. In their work, they considered complex interdependencies that are broken down into two interrelated markets, namely a service market, which involves trading of application services, and a resource market, which involves trading of computational and infrastructure resources such as processors, memory, etc. [18] present the GridEcon platform - a testbed for designing and evaluating economics-aware services in a commercial Cloud computing setting. The authors assume the difficulties in predicting the context of a service market and motivate development of an environment for evaluating its behaviour in an emulated market platform. The platform is composed of the Marketplace, which allows trading goods using different market mechanisms, and the Workflow Engine, which enables a simple composition of a market environment by describing the service interactions between economics-aware services. [14] discuss a mechanism for the resolution of the customers' requirements that enhances the process of selecting Cloud services from the business point of view. The work is related to the 4CaaSt project and aims to create a PaaS Cloud platform that supports the optimised and elastic hosting of Internet-scale multi-tier applications.

[2] discuss a framework for modelling and simulating service-oriented applications and autonomic policies for service provisioning and resource orchestration for Application Layer Networks in utility computing environments. The approach is evaluated within CATNETS project and investigates the use of an economic model (Catallaxy) in distributed environments like Grids and P2P networks. [20] discuss the design of a simulator with a set of features for simulation of Grid testbeds as an extension to GridSim. They model heterogeneous computational resources of variable performance, scheduling of jobs based on various policies, differentiated network service, and workload trace-based simulation.

Although many of these market simulators successfully address some of the main challenges of electronic markets in distributed environments, they are fairly static and do not have any autonomic capabilities. Therefore to orchestrate and evaluate autonomic markets, a more flexible simulation approach is necessary.

4 A Case Study for Market Simulation Using GridSim

In [6], we used GridSim as a means to explore how a market could be monitored as first steps towards adding autonomic capabilities to an electronic market. We selected GridSim for a variety of reasons: (1) it implements numerous mechanisms for resource allocations [3] as well as interfaces for implementing additional mechanisms; (2) it is designed as an extensible multi-layer architecture which allows new (technical) components or layers to be added and integrated [7]; (3) it allows different classes of heterogeneous resources; and (4) as an open-source toolkit it has already been used widely. Although GridSim simulates Grid resource and networks and not the Cloud computing paradigm directly, it is important to note that these two contexts do not differ significantly, as the core techniques for matching buyers to sellers are equal and independent of technical paradigm. Using the layered architecture of GridSim, we implemented three monitoring sensors for: market mechanisms, the market in general, and computational infrastructure of the platform as extensions to the existing GridSim layers to monitor the infrastructure and institutional performance of the market.

The **Mechanism sensor** monitors the performance of a market mechanism, which includes revenue, the number of resource allocations, and average price for a single unit of resource. The actual allocation is handled by GridSim. The mechanism sensor uses the GridSim interfaces to receive a notification of an allocation, i.e. a match of a bid to an ask. Once a resource is allocated, the sensor receives and stores information about the allocation in the knowledge component. Using the same GridSim interfaces, the sensor also gathers mechanism-specific information like number of bids and asks awaiting allocation.

The **Market sensor** gathers market information, for example data concerning the past and current number of sellers and buyers on the market and information concerning the resources traded. This is achieved by using GridSim interfaces of the architecture layers responsible for resource and job management.

The **Infrastructure sensor** monitors the usage of computational resources. In particular, it monitors the utilisation and performance of the underlying operating system and hardware infrastructure. For example, processor utilisation and speed, number of threads, memory usage, hard-disk usage, etc. The infrastructure monitoring is based on the interfaces defined by the *java.lang.Management* package, which is a management interface for monitoring and management of the Java virtual machine as well as the host operating system.

Despite the large flexibility of GridSim, its numerous interfaces and multi-layered architecture, creating a simulation scenario is not a trivial task, as many market actions and the creation of communication objects between the layers of GridSim are left to the user. However, GridSim does provide a small set of examples that illustrate the implementation of simple trading scenarios. In our feasibility study, we applied one of the example scenarios. This example allowed us to control basic trading properties, i.e. the number of buyers and sellers in the market and the number of requests per buyer, etc., which for our purposes was adequate. It also enabled the construction of a market, establishment of participants and resources, and provided an easy platform upon which to implement a monitoring framework. It was also straightforward to implement a basic benchmark scenario to test the monitoring framework.

However, we encountered difficulties when we created more realistic and elaborate scenarios, for example: different participant types (e.g. malicious users, market makers, speculators, monopolists, and other strategic behaviours); more complicated trades, i.e. multiple resource entities; dynamic context: adding or removing participants or resource types at runtime; and engineering aspects like market growth or contraction. When trying to create such scenarios, we encountered runtime exceptions for the following reasons: (1) GridSim expects the number of users to remain fixed; (2) it is not possible to change the quantity of resources that sellers offer and buyers request, i.e. total supply and demand is predefined; (3) new resource types cannot be added at runtime; (4) it is not possible to manage the timings of the bid/ask submissions, this is controlled by GridSim's event handlers, which makes it difficult to implement users with specific participation strategies. Through our efforts to counter these as well as other challenges, we were moving away from GridSim's initial use case, eventually making it impossible to control and extend further. Consequently, we were no longer confident that changes in the market were engineered by us as opposed to errors in the GridSim runtime. It would be easy to say that this is a failing of GridSim, but our scenarios were straying outside of GridSim's scope.

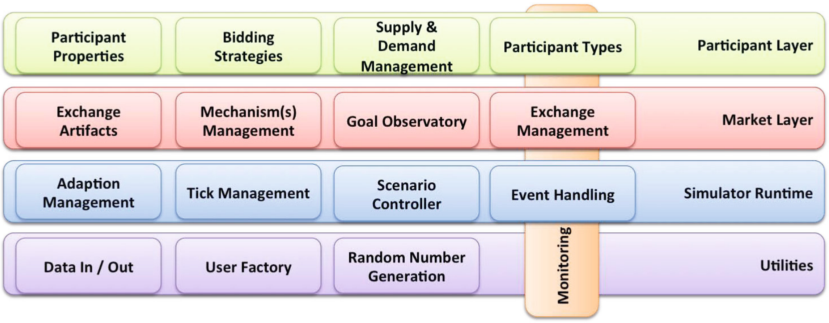


Fig. 1. Conceptual Framework of a Simulation Environment for Autonomic Markets

5 Conceptualisation of a Simulator for Autonomic Markets

Based upon our experiences with GridSim, we realised that trying to simulate different aspects for the study of autonomic markets needs a more flexible simulation approach. In Section 3, we discussed some alternatives to GridSim, but failed to see the ability to capture all aspects that we feel are necessary without significant effort in the extension of an approach. In this section, we propose a conceptual architecture for an autonomic market simulator that will act as a testing environment for the future studies. Fig. 1 illustrates the layers and components of our proposed simulation architecture, which are as follows:

The **Monitoring Framework** captures key information on the market platform through links to the Participant, Market and Simulator layers, and makes this information available to the components that require it (e.g. the Goal Observatory). Monitoring information here captures the state of: mechanisms, the market in general and the computational infrastructure, as described in [6].

The **Participant Layer** captures the aspects necessary to represent market participants as well as their various nuances and differentiating factors. The key component is *participant type*, which identifies whether a participant is a consumer, provider, prosumer, or broker. It also enables different participant flavours like market makers, speculators, monopolists, aggressive and passive participants. In accordance to the typical market simulators, we define *bidding strategies*, as well as the management of *supply and demand*. We use the word "management" to illustrate that this is not a statically defined process, but entails stochastic and dynamic behaviours such as participants joining or leaving the market, as well changes in their individual properties and requirements over time. *Participant properties* capture additional information needed for each participant type, e.g. range of wealth, resource types offered/desired, etc.

The **Market Layer** defines the components to implement an electronic market. This includes: the *artefacts* to be traded, including their type, quantity and period of availability or desirability; different allocation *mechanisms* like the English or Continuous Double Auction, but also the means to create custom mechanisms and have multiple active mechanisms. Mechanism management here refers to the programming constructs to transparently include any arbitrary mechanism by exposing a standard interface. A mechanism manager controls how bids and asks are passed to a mechanism and when instances are created and destroyed; a *Goal Observatory* for defining goals and keeping track of their adherence via the monitoring framework; a *exchange management* that keeps track of all incoming asks and bids, matches, as well as one or more active mechanisms; and finally, *adaption management* as an instantiation of a market adaption component.

The **Simulator Layer** is the basis for the simulator. It includes a singleton *event handler*, as this enables a simple programming model without the need for complex thread or concurrency management, and a *tick manager* to control "time" in the simulator as a sequence of discrete periods. In each tick, we invoke participants in a renewed random order, and give them the option to "act", i.e. do something in the market. We also define a *scenario controller*, which through the event handler can instigate new scenarios for observation, based upon the current time. The scenario controller permits us to create issues of instability or change specific settings in the market to study how the market changes, and later how adaption actions have improved or worsened the situation. We can also layer (simple) scenarios to create more complex compound scenarios.

We also define key **utilities** to assist in market simulation. These include: readers for trace data from existing markets to "stimulate" market events or scenarios as well as writers to store monitoring data; a *participant factory* to facilitate the generation of multiple participant types based on a set of input parameters; and as a key premise for all simulators, a *random number generator* which can simply be the inclusion of the Colt Library[1] or similar.

6 Summary

In this paper, we have argued that existing electronic market platforms are insufficient for immature domains like Cloud computing. Therefore, we proposed the concept of an autonomic marketplace platform: the automatic adaption of the economic models of the platform and its underpinning infrastructure based upon a given concept of "market performance". We described how the autonomic MAPE-K loop can be applied to an electronic market platform. Finally, we presented our experiences in trying to build a simulation tool as a premise for the study and evaluation of an autonomic market using GridSim as a case study. However, we encountered too many scenario specific obstacles that merited a bespoke simulation framework. Building on top of the lessons learned from GridSim, we defined a conceptual framework for a market simulator that can facilitate different aspects of study for an autonomic market. These include the definition of destructive scenarios, stochastic events and extended user types. Our future work is the continued investigation of our simulation tool, its on-going development as well as the development of scenarios for its calibration.

[1] `http://acs.lbl.gov/software/colt/`

References

1. Abdelwahed, S., Bai, J., Su, R., Kandasamy, N.: On the application of predictive control techniques for adaptive performance management of computing systems. IEEE Transactions on Network and Service Management 6(4), 212–225 (2009)
2. de Assunção, M., Streitberger, W., Eymann, T., Buyya, R.: Enabling the simulation of service-oriented computing and provisioning policies for autonomic utility grids. In: Veit, D.J., Altmann, J. (eds.) GECON 2007. LNCS, vol. 4685, pp. 136–149. Springer, Heidelberg (2007)
3. Assunçao, M.D.D., Buyya, R.: An evaluation of communication demand of auction protocols in grid environments. Tech. rep. (2006)
4. Boughton, H., Martin, P., Powley, W., Horman, R.: Workload Class Importance Policy in Autonomic Database Management Systems. In: Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks, pp. 13–22. IEEE Computer Society (2006)
5. Brandic, I., Music, D., Dustdar, S.: Service mediation and negotiation bootstrapping as first achievements towards self-adaptable grid and cloud services. In: Grids meet Autonomic Computing Workshop, in Conjunction with the 6th International Conference on Autonomic Computing and Communications, pp. 1–8. ACM (2009)
6. Breskovic, I., Haas, C., Caton, S., Brandic, I.: Towards Self-Awareness in Cloud Markets: A Monitoring Methodology. In: Proceedings of the 9th IEEE International Conference on Dependable, Autonomic and Secure Computing (2011)
7. Buyya, R., Sulistio, A.: Service and utility oriented computing systems: Challenges and opportunities for modeling and simulation communities. In: Annual Simulation Symposium, pp. 68–81 (2008)
8. Calheiros, R.N., Ranjan, R., Beloglazov, A., Rose, C.A.F.D., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software - Practice and Experience (2010)
9. Callaway, R., Devetsikiotis, M., Viniotis, Y., Rodriguez, A.: An autonomic service delivery platform for service-oriented network environments. IEEE Transactions on Services Computing 3(2), 327–331 (2010)
10. Caton, S., Rana, O.: Towards Autonomic Management for Cloud Services based upon Volunteered Resources. Concurrency and Computation: Practice and Experience 23 (2011)
11. Cheliotis, G., Kenyon, C.: Autonomic economics. In: IEEE International Conference on Electronic Commerce, pp. 120–127 (June 2003)
12. Cheng, Y., Leon-Garcia, A., Foster, I.: Toward an autonomic service management framework: A holistic vision of SOA, AON, and autonomic computing. IEEE Communications 46(5), 138–146 (2008)
13. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer 36, 41–50 (2003)
14. Menychtas, A., Gatzioura, A., Varvarigou, T.: A business resolution engine for cloud marketplaces. In: Proceedings of the Third International Conference on Cloud Computing Technology and Science, pp. 462–469 (2011)
15. Oyenan, W.H., Deloach, S.A.: Towards a systematic approach for designing autonomic systems. Web Intelligence and Agent Systems 8, 79–97 (2010)
16. Papazoglou, M., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: State of the art and research challenges. Computer 40(11), 38–45 (2007)

17. Pardoe, D., Stone, P., Saar-Tsechansky, M., Tomak, K.: Adaptive mechanism design: a metalearning approach. In: 8th International Conference on Electronic Commerce, pp. 92–102. ACM (2006)
18. Risch, M., Altmann, J., Guo, L., Fleming, A., Courcoubetis, C.: The gridecon platform: A business scenario testbed for commercial cloud services. In: Altmann, J., Buyya, R., Rana, O.F. (eds.) GECON 2009. LNCS, vol. 5745, pp. 46–59. Springer, Heidelberg (2009)
19. Streitberger, W., Eymann, T.: A simulation of an economic, self-organising resource allocation approach for application layer networks. Computer Networks 53, 1760–1770 (2009)
20. Sulistio, A., Cibej, U., Venugopal, S., Robic, B., Buyya, R.: A toolkit for modelling and simulating data grids: an extension to gridsim. Concurr. Comput.: Pract. Exper. 20, 1591–1609 (2008)

Opinion Model Based Security Reputation Enabling Cloud Broker Architecture

Pramod S. Pawar[1,2], Srijith K. Nair[2], Fadi El-Moussa[2], Theo Dimitrakos[2], Muttukrishnan Rajarajan[1], and Andrea Zisman[1]

[1] City University London, London EC1V 0HB, United Kingdom
r.muttukrishnan@city.ac.uk, a.zisman@soi.city.ac.uk
[2] British Telecommunications, Security Practice, Adastral Park, Ipswich IP5 3RE, UK
{pramod.s.pawar,srijith.nair, fadiali.el-moussa,theo.dimitrakos}@bt.com

Abstract. Security and trust in service providers is a major concern in the use of cloud services and the associated process of selecting a cloud service provider that meets the expectations and needs of one's security requirements is not easy. As a solution, we propose a broker architecture model that enables us to build a security reputation framework for cloud service providers, capturing comprehensive evidence of security information to build its trust and security reputation

Keywords: broker, reputation, subjective logic, security.

1 Introduction

Cloud computing has become one of the fastest growing segments of the IT industry. Cloud computing involves a provider delivering a variety of IT enabled resources to consumers as a service over the Internet. Cloud computing services are offered as Software as a Service (SaaS), Platform as a Service (PasS) or Infrastructure as a Service (IaaS) [22]. Virtualization is a core enabling technology for cloud IaaS architectures. Even though several advantages of the use of cloud based services have been identified, in particular the pay-as-you-consume costing model and the minimization of capex costs, the inherent loss of control of data and process to external parties (cloud service providers) have the customers worried.

Since security remains a major concern in the use of cloud services, an individual or an enterprise expects a high level of confidence and trust in the cloud service provider it would like to use. The enterprise needs a process to identify and decide on the most suitable service provider to fulfill its security requirements for its service to be deployed. Reputation systems have been effectively used in making such decisions, however it is highly challenging to apply the concept to the cloud ecosystem, with a security context. This is challenging mainly due to the reluctance of the cloud service providers to publicize their security related information to the internet community or even to a selected group of customers. Relevant information may include events or incidence recorded due to security activities like firewall filtering, intrusion detection/prevention systems, security policies, authentication/authorization, identity management and key management.

However one also need to keep in mind the fact that IT service providers have been providing details of their security systems and associated processes to third party (security) auditors for obtaining security certifications and legal compliance status. These certifications are often essential requirements of the service provider to gain confidence of their customers and the industry as a whole. In order to obtain security certification the service provider needs to share, among other details, the security event related information to the third party auditors. The higher the level of security certification required, the more critical security events information and process details are expected by the auditors. In order to avoid security leakage it is a common practice to obtain non-disclosure agreements with auditors before this critical security information are shared. An enterprise needing cloud services have to rely on the security certifications of the cloud service providers to establish trust in the providers. This approach however constraint the enterprise to match their security requirements based only on the certification information published by the service providers and the associated minimum requirements that needs to be met by the service provider for obtaining the certification, due to unavailability of other detailed information.

As a way of breaking this impasse we propose the use of a Cloud Broker (CB) that inherits and expands on the role of the security auditor, enabling the broker to obtain access to the security events due to the high trust placed by the service providers, which may not be possible with the wider community. The CB provisions the enterprises with security reputation of the cloud service providers based on their security requirements as specified to the CB. The registration with the broker allows the cloud service providers to highlight their security strengths without exposing their internal security details like event information to the wider customer base and at the same time also benefited by CB's potentially wider customer base. The cloud service consumers benefit from the service that provides a closest match between their security requirements and the security reputation of the cloud service providers.

The remaining of the paper is structured as follows: Section 2 provides the background and related work. Section 3 describes the cloud broker architecture and its components. Section 4 describes our approach of the reputation modeling to build the security reputation of the cloud service provider. Section 5 provides applicability of this work in an existing project OPTIMIS – Optimized Infrastructure Services. Section 6 provides concluding remarks and future work.

2 Related Work

Reputation system based trust model have been adopted in several open systems such as internet websites, e-commerce, P2P Systems and mobile adhoc networks [7][15][16][6][12][17][9][18]. Resnick et. al. [15][16] discusses the importance of reputation system to decide whom to trust in the Internet where large number of producers or consumers may not know each other. Epinion [17], eBay [15][16] are some of the very popular electronic markets using reputation systems. Trust management systems help reduce free riding of the nodes in the P2P systems where each entity can act as client and server, expecting to contribute in the systems.

The trust model for P2P systems in [21] considers transactions and shared experiences as recommendations and uses Bayesian estimation methods to compute trust values. The Beta reputation model in [8] is based on beta distribution that considers the direct experience as well as feedback from other agents to model the behavior of a system. Both models [8][21] are based on the belief theory, but in [21] the use of Bayesian estimation expects probabilities for each question of interest.

The study of trust is closely related to *uncertainty* and we observe that many of the reputation system proposed have given either no importance or a very low importance to uncertainty. Exceptions are found in the works described in [7][14][10][13][20]. The belief model in [7] uses metric called *opinion* to describe *belief* and *disbelief* about a proposition as well as the degree of *uncertainty* regarding probability of an event. The work on [13][20] proposes opinion metric as in [7] but giving importance to uncertainty due to the evidence that impacts the belief and disbelief about a proposition. In [7] the uncertainty is modeled only based on the amount of total evidence i.e. as the total evidence increases, the uncertainty decreases, while in [13][20] the uncertainty also takes into account the amount of positive and negative evidence contained in total evidence. The work in [13] shows that it provides low prediction errors compared to [7][20]. Opinion models have been extensively used for estimating the quality by combining multiple factors. The opinion model proposed in [13] uses the subjective logic to combine evidences and due to its low prediction errors forms the best choice for building reputation of the cloud service providers.

In the recent years reputation systems have also been used in the cloud computing paradigm [1][3][5][13]. In [3], trust is one of the core component used by software as a service provider, along with risk, eco-efficiency and cost for evaluating the cloud infrastructure provider, for their service. The trust of the cloud infrastructure provider in [3] is evaluated by the model proposed in [13]. The work in [5] identifies several vulnerabilities in cloud services provided by Google, IBM, Amazon and proposes an architecture to reinforce the security and privacy by suggesting a hierarchy of P2P reputation system to protect cloud resources. The focus in [13] and [5] has been on use of conventional trust within a cloud service ecosystem and no specific context of security to build reputation of the cloud service providers is considered.

The concept of a broker as intermediaries between the service providers and service consumers with the aim of relieving the customer from evaluating trust and risk of the service provider has been used in the grid and cloud environments before [11][4][19][2]. The work in [4] proposes broker architecture in grids with the focuses on evaluating the reliability of the risk information from the resource providers. Within the context of cloud computing environment [11], cloud broker can be used as *i) cloud service intermediation:* intermediation for multiple services to add value-additions like identity management or access control *ii) cloud service aggregation:* bringing together two or more fixed cloud based service *iii) cloud service arbitrage:* similar to cloud service aggregation, but more dynamic aggregation to provide flexibility. The work in [11][4] have been focusing in identifying trust and risk of the service providers without any security context.

This paper proposes a broker architecture that enables the gathering of security related events of the cloud service providers, which otherwise is difficult to be shared with the end users, and uses the reputation model proposed in [13] to build the security reputation of the cloud service providers.

3 Cloud Broker Architecture

We introduce a Cloud Broker architecture that enables building of security reputation of individual service provider and sharing the same with its customers. The proposed broker architecture is shown in Figure 1 that includes various components namely: i) *Cloud Service Provider Interface (CSPI)* ii) *Enterprise users Interface (EUI)* iii) *Monitors (M)* and iv) *Trust Engine (TE).* The entities involved in the architecture are Cloud Service Providers (CSP) and Enterprise Users (EU). The CSP and the EU register with broker. The registration of the CSP at the broker includes the agreement with the broker to share security related information with the broker and in turn the broker has a non-disclosure agreement with the service provider.

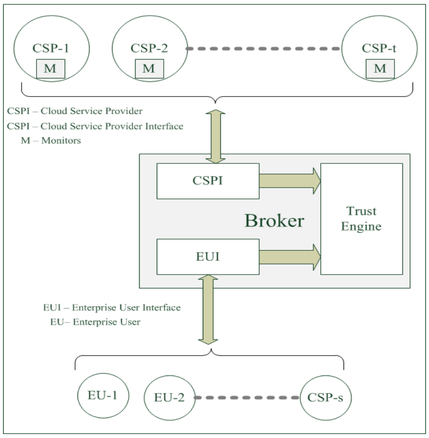

CSP-1
M
CSP-2
M
CSP-t
M
CSPI – Cloud Service Provider
CSPI – Cloud Service Provider Interface
M – Monitors
CSPI
Broker
Trust Engine
EUI
EUI – Enterprise User Interface
EU– Enterprise User
EU-1
EU-2
CSP-s

Fig. 1. Cloud Broker Architecture

3.1 Cloud Service Provider Interface (CSPI)

This interface enables the service provider to provide details of its security practices and security measures in place, allowing advertising its security strengths. In our experience, we find cloud service providers try to provide the following security measures as a basic step towards securing their customers environment: *i) Protecting individual virtual environment ii) Filter traffic between each virtual instances iii) Hardening the hypervisor iv) Protecting the network infrastructure v) Protecting the data stored at each individual virtual instance vi) Policy enforcement for authentication and access management to individual virtual instances vii) Patch management*

3.2 Enterprise User Interface (EUI)

This interface allows the enterprise users to input their security requirements, select most appropriate cloud service provider for their security needs, provide feedback on the services and also register complaints. The requirements associated with a service and the security features expected, are encoded in the *service manifest* as discussed in [3].The feedback and the complaints form a vital piece of evidence to model the cloud service providers reputation based on its security strength.

3.3 Monitors

The broker receives security violation events of the service provider by registering to the pub-sub [18] monitors in the service provider's infrastructure. The threats that prevent organizations from adoption of the cloud services and the areas for gathering metrics are identified as follows: *i) Insecure Authentication or Authorization:* Interface allowing customers to manage cloud services in order to perform provisioning, management, orchestration, and monitoring their virtual instances *ii) Insider Attack:* An insider from cloud service provider could have privileged access to confidential data or gain control over the cloud service with no or little risk of detection *iii) Multitenant Attack:* Cloud environment is meant to allow multiple users share resources (CPU, network, memory, storage, etc.) and an improper isolation of the multi-tenant architecture may lead to have access to any other tenant's data *iv) Data Leakage:* Customers data on the cloud could be compromised, deleted or modified *v) Malware Propagation:* Any malware that infects a virtual instance could propagate over the shared host or to hypervisor, spreading rapidly, giving ability to eavesdrop on customer's transactions.

3.4 Trust Engine

The trust engine contained in the cloud broker is the core part of the architecture that performs the *trustworthiness* calculation for the cloud service providers. Figure 2 shows the internal work flow used for computing the reputation of cloud service provider based on the inputs received from the interfaces of the broker.

i. *Evidence:* The evidences provided to the opinion model are gathered from monitors, cloud service provider interface and enterprise user interface.
ii. *Opinion Model :* The evidences received from different monitors are used to form an opinion about a cloud service provider based on the opinion model proposed in [13]. The opinion of a proposition x, represented as $w(x)$ or w_x is defined in terms of *belief* $b(x)$ or b_x, *disbelief* $d(x)$ or d_x and *uncertainty* $u(x)$ or u_x where $b(x)+d(x)+u(x)=1$. The opinion model in [13] is given as follows:

$$W_x = (b_x, d_x, u_x, a_x) \tag{1}$$

$$b_x = c\, r / t \tag{2}$$

$$d_x = c\, s\ / t \tag{3}$$

$$u_x = t / (r\, s + f^2 + 1) \tag{4}$$

$$c = 1 - u_x \tag{5}$$

where: *r* is amount of positive evidence; *s* is amount of negative evidence; *t* is total evidence given as *t=r+s*; *c* or *c(t) or c(r,s)* is certainty as a function of total evidence; and *f* is distance of focus to the centre of an ellipse formed by mapping the positive and negative evidence to major and minor semi-axes of an ellipse.

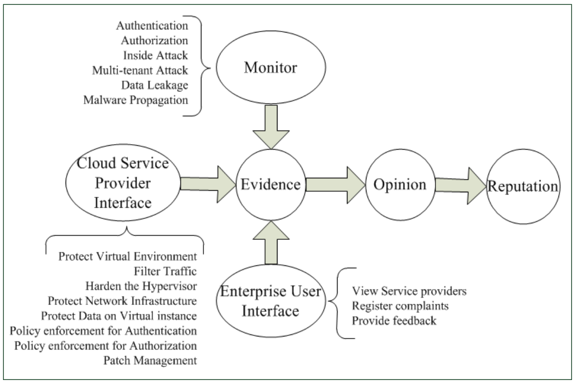


Fig. 2. Trust Engine

The opinion formed by the monitors is combined with the opinion formed based on the enterprise user's feedback and complaints. The subjective logic by Josang [7] is used to combine multiple opinions to form a single opinion using the operators such as conjunction, consensus that allows performing logical operations on opinions. This paper uses the opinion model proposed in [13] and the subjective logic operators [7]. The conjunction operator is standard logic "AND" operating on the opinions. The consensus operator enables combining the opinions of entity A and entity B representing an imaginary entity [A,B]'s opinion about proposition *x*.

iii. *Reputation:* The probability expectation of an opinion is used to provide the reputation rating. The expectation of an opinion is given as *E(w(x))=b+au* where *E(w(x)) Є [0,1]* and *a(x)* is base rate that provides the weight of uncertainty that contributes to the probability expectation.

Figure 2 shows process of modeling the security reputation by broker. The first step is the broker getting evidential information from two sources a) Monitor and b) Customer interface. The second step is to convert the evidence obtained to compute an opinion. The third step is to calculate the reputation of a service provider based on the opinion formed. The details of reputation calculation are given in section 4.

4 Reputation System

The reputation of a cloud service provider is calculated in terms of its *trustworthiness(T)* using opinion obtained from computations, namely i) *Incidence Monitoring(M)*: Security incedence events received from monitoring ii) *Enterprise User Rating(EUR)*: Ratings provided by the enterprise user for satisfaction of the security features provided by CSP. The *trustworthiness(T)* is given by applying the conjunction operator of subjective logic on the opinions obtained from each of these computation and then calculating the expectation of the combined opinion.

$$T=\text{Expectation}\ (W_{M\ \wedge\ EUR}) \tag{6}$$

Where W_M is the opinion obtained from the monitoring(M) as well as the W_{EUR} is the opinion obtained from the enterprise user ratings(EUR). The symbol Λ is the *conjunction operator* used to combine the two opinions.

4.1 Incidence Monitoring

The incidence monitoring records evidence about the incidences related to parameters such as authentication, authorization, inside attacks, multi-tenent attack, data leakage and malware propogation. These incidences can either be identified by the cloud service provider and sent to the broker or the broker after receiving the security events carries further analysis to identify the incidences from the data received. Both approaches have their own advantages and disadvantages.

For each monitoring parameter, the number of incidents occuring within a time window *w* are observed. Every incident identified, adds to the negative evidence and absence of incidents increases the positive evidence. Based on the positive and negative evidences, opinions are formed for each of the parameters. Let $W_{AT,}$ $W_{AR,}$ $W_{IA,}$ $W_{MT,}$ $W_{DL,}$ and W_{MP} be opinions formed for CSP based on the monitoring parameter of authentication, authorization, inside attacks, multi-tenent attack, data leakage and malware propogation respectively. Consider for example that there are *n* monitors associated with monitoring of authentication incidence at CSP-1. Then the opinion W_{AT} for CSP-1 is given as the *consensus* of all *n* monitors. Considering all monitoring parameters, the overall opinion W_M for CSP-1 is given by applying *conjunction* operator over the *consensus* opinion, which is as follows:

$$W_M = W_{AT}^{M1,\ldots,Mn} \Lambda\, W_{AR}^{M1,\ldots,Mn} \Lambda\, W_{IA}^{M1,\ldots,Mn} \Lambda\, W_{MT}^{M1,\ldots,Mn} \Lambda\, W_{DL}^{M1,\ldots,Mn} \Lambda W_{MP}^{M1,\ldots,Mn} \tag{7}$$

Where $W_{AT}^{M1,..,Mn}$ is consensus opinion by monitors M1 to Mn regarding authentication. Similarly consensus opinions for other parameters are obtained.

4.2 Enterprise User Rating

For every usage of the services from the CSP, the enterprise user rates the satisfaction of security features and capabilities provided by the CSP corresponding to the requirements set forward initially by the user. Consider *q* enterprise users registered with the broker and provide ratings to the CSP for each of the monitoring parameters. The overall opinon W_{EUR} for CSP-1 based on the enterprise user rating is given by applying the *conjunction* operator over the consensus opinion, as follows:

$$W_{EUR} = W_{AT}^{EU1,EU2\ldots,EUq} \Lambda\, W_{AR}^{EU1,EU2\ldots,EUq} \Lambda\, W_{IA}^{EU1,EU2\ldots,EUq} \Lambda\, W_{MT}^{EU1,EU2\ldots,EUq} \Lambda\, W_{DL}^{EU1,EU2\ldots,EUq} \Lambda\, W_{MP}^{EU1,EU2\ldots,EUq} \tag{8}$$

Where $W_{AT}^{EU1,EU2...,EUq}$ is consensus opinion for CSP-1 given by enterprise user EU1 to EUq based on the authentication. Similarly $W_{AR}^{EU1,EU2...,EUq}$, $W_{IA}^{EU1,EU2...,EUq}$, $W_{MT}^{EU1,EU2...,EUq}$, $W_{DL}^{EU1,EU2...,EUq}$ and $W_{MP}^{EU1,EU2...,EUq}$ are the consensus opinion for CSP-1 by EU1 to EUq based on authorization, inside attacks, multi-tenent attack, data leakage and malware propogation respectively.

4.3 Trust of Cloud Service Provider

The *trustworthiness(T)* of the cloud service provider is given by calculating the expectation of the opinions W_M and W_{EUR} given by Incidence *monitoring* and the *Enterprise User* respectively. The *trustworthiness(T)* can be represented as:

$$T = \text{Expectation}\ (W_M \wedge W_{EUR}) = \text{Expectation}\ (W_{M \wedge EUR}) \tag{9}$$

Where $W_{M \wedge EUR} = (b_{M \wedge EUR}, d_{M \wedge EUR}, u_{M \wedge EUR}, a_{M \wedge EUR})$ and the expectation of the opinion $W_{M \wedge EUR}$ is given as :

$$E(W_{M \wedge EUR}) = b_{M \wedge EUR} + (a_{M \wedge EUR})(u_{M \wedge EUR}) \tag{10}$$

5 Applicability of This Architecture

The cloud broker architecture proposed in this paper is a very generic and not limited to any specific environment. However, a practical, environment specific implementation of the proposed architecture is being used in the OPTIMIS [3][11] project. OPTIMIS toolkit is a set of software components for simplified management of cloud services and infrastructures that assists the cloud service providers to provide optimized services based on the TREC (Trust, Risk, Eco-efficiency and Cost).

TREC components are part of the basic toolkit. The trustworthiness of an IP (Infrastructure Provider) enables the SPs (Service Provider) to identify and select the IP having proven capabilities to provide the required service. The risk assessment performed provides the SP with the risk involved in the construction, deployment and operation of a service. The eco-efficiency aids in selecting a cloud service provider based on the energy consumption. Along with the trust, risk and eco-efficiency factor, cost forms the trade-off factor in providing of the optimized service.

The broker architecture [11] in the OPTIMIS project already have a support of the TREC toolkit, SLA agreement and the monitoring infrastructure which can be enabled to build the security reputation of the IP using the proposed reputation model [13] described in section 4 and the security related events captured in section 3. Figure 3 shows the high level sequence diagram for broker implementation in OPTIMIS project. Following are the sequence of steps: *a)* The SP uses the *IDE (Integrated Development Environment)* to create a service which is described in a *service manifest b)* The *IDE* passes the service manifest and the optimization objective to the *SD (Service Deployer)* for deployment of the service *c)* The *SD* uses the cloud broker interface to submit the *service manifest* and the optimization objective *d)* The cloud broker has *Registry* where all SPs and IPs register before using the cloud broker services *e)* The broker after receiving a request for deployment of a service gets the list of IPs from the *Registry f)* The *TREC* component of the broker contains the historical assessments of all SPs and IPs stored in the *DB (Database) g)* Based on the TREC assessments, the broker filters the IPs and the *DO (Deployment Optimizer)* initiates SLA negotiations with the filtered IPs *h)* In the process of negotiation, the broker interacts with the AC(Admission Control) which checks its current infrastructure status and provides offers based on the request made *i)* Once all the offers for all the components of the service is received the broker applies the optimization algorithm to provide the SP with the ranked list of IPs for each of its service components based on the TREC *j)* The SP deploys all its components considering the ranked list *k)* The service is deployed using the CO (*Cloud Optimizer*) at the IP side. The CO provides all VM(*Virtual Machines*) related information to the SP, which in turn is forward to the Broker *l)* The broker passes the VM information, to the TREC components to receive monitoring events for these service components

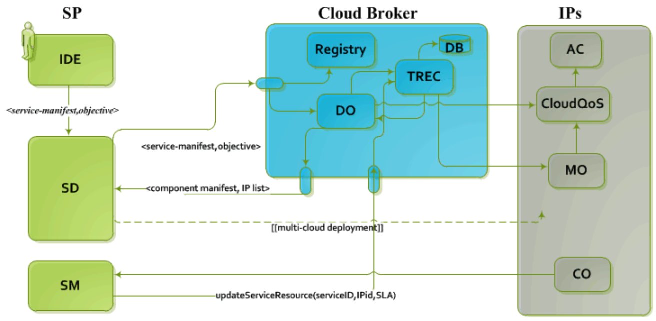

SP
Cloud Broker
IPs
IDE
Registry
DB
TREC
DO
AC
CloudQoS
MO
<service-manifest,objective>
<service-manifest,objective>
SD
<component manifest, IP list>
[[multi-cloud deployment]]
SM
CO
updateServiceResource(serviceID,IPid,SLA)

Fig. 3. High level sequence diagram for broker in OPTIMIS

6 Conclusion and Future Work

In this paper we propose security reputation systems using broker architecture for cloud service providers, allowing customers to achieve a level of expectation from cloud service providers about their deployed security systems. By having a broker and using security reputation based on evidence helps customers build confidence in using a specific service provider and also gives incentive to cloud providers to demonstrate their security capabilities to the customers. As future work we intend to identify a comprehensive security requirements that map to the monitoring infrastructure which will enable the broker to provide the cloud service provider and the enterprise user with a generic interface to specify its capabilities and requirements. We also aim to perform a rigorous evaluation of the proposed architecture by using the simulated as well as real data of the cloud service providers using the OPTIMIS infrastructure.

Acknowledgment. This work has been partially supported by the EU within the 7th Framework Programme under contract ICT-257115-Optimized Infrastructure Services (OPTIMIS).

References

1. Alhamad, M., Dillon, T., Chang, E.: SLA-Based Trust Model for Cloud Computing 13th International Conference on Network-Based Information Systems (2010)
2. Dumitrescu, C., Raicu, I., Foster, I.: DI-GRUBER: A Distributed Approach to Grid Resource Brokering. In: Proceedings of the ACM/IEEE Conference on Supercomputing (2005)
3. Ferrer, A.J., Hernández, F., Tordsson, J., Elmroth, E., Ali-Eldin, A., Zsigri, C., Sirvent, R., Guitart, J., Badia, R.M., Djemame, K., Ziegler, W., Dimitrakos, T., Nair, S.K., Kousiouris, G., Konstanteli, K., Varvarigou, T., Hudzia, B., Kipp, A., Wesner, S., Corrales, M., Forgó, N., Sharif, T., Sheridan, C.: OPTIMIS: a Holistic Approach to Cloud Service Provisioning. Future Generation Computer Systems 28(1), 66–77 (2012)
4. Gourlay, I., Djemame, K., Padgett, J.: Reliability and Risk in Grid Resource Brokering. In: IEEE International Conference on Digital Ecosystems and Technologies (DEST) (2008)
5. Hwang, K., Kulkarni, S., Hu, Y.: Cloud Security with Virtualized Defense and Reputation-based Trust Management. In: Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing (2009)
6. Jin, Y., Bloch, P., Cameron, G.: A Comparative Study: Does the Word-of-mouth Communications and Opinion Leadership Model Fit Epinions on the Internet? In: Proceedings of the Hawaii International Conference on Social Sciences (2002)
7. Jøsang, A.: A Logic for Uncertain Probabilities. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 9(3), 279–311 (2001)
8. Jøsang, A., Ismail, R.: The Beta Reputation System. In: Proceedings of the 15th Bled Electronic Commerce Conference e-Reality: Constructing the e-Economy (2002)
9. Kerr, R., Cohen, R.: Modeling trust using transactional, numerical units. In: PST 2006: Proceedings of the Conference on Privacy, Security and Trust, Markham, Canada (2006)
10. Li, F., Wu, J.: Uncertainty Modeling and Reduction in MANETs. IEEE Transactions on Mobile Computing 9(7) (2010)
11. Nair, S.K., Porwal, S., Dimitrakos, T., Ferrer, A.J., Tordsson, J., Sharif, T., Sheridan, C., Rajarajan, M., Khan, A.U.: Towards Secure Cloud Bursting, Brokerage and Aggregation. In: IEEE 8th European Conference Web Services (ECOWS) (2010)
12. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web, Technical report, Stanford Digital Library Technologies Project (1998)
13. Pawar, P.S., Rajarajan, M., Nair, S.K., Zisman, A.: Trust model for optimized cloud services. In: Dimitrakos, T., Moona, R., Patel, D., McKnight, D.H. (eds.) IFIPTM 2012. IFIP AICT, vol. 374, pp. 97–112. Springer, Heidelberg (2012)
14. Ray, I., Poolsappasit, N., Dewri, R.: An Opinion Model for Evaluating Malicious Activities in Pervasive Computing Systems
15. Resnick, P., Zeckhauser, R.: Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System. In: Baye, M.R. (ed.) The Economics of the Internet and E-Commerce. Advances in Applied Microeconomics, vol. 11, pp. 127–157. Elsevier Science, Amsterdam
16. Resnick, P., Zeckhauser, R., Swanson, J., Lockwood, K.: The Value of Reputation on Ebay: A Controlled Experiment. Experimental Economics 9(2), 79-101(23) (2006)
17. Schneider, J., Kortuem, G., Jager, J., Fickas, S., Segall, Z.: Disseminating trust information in wearable communities. Personal and Ubiquitous Computing 4(4), 245–248, doi:10.1007/BF02391568
18. Srivatsa, M., Liu, L.: Secure Event Dissemination in Publish-Subscribe Networks. In: 27th International Conference on Distributed Computing Systems (ICDS 2007) (2007)

19. Venugopal, S., Buyya, R., Winton, L.: A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids. In: Concurrency and Computation: Practice and Experience, vol. 18(6), pp. 685–699. Wiley Press, New York (2006)
20. Wang, Y., Singh, M.P.: Evidence-Based Trust: A Mathematical Model Geared for Multiagent Systems. ACM Transactions on Autonomous and Adaptive Systems, Vol 5(4), Article 14 (2010)
21. Wu, P., Wu, G.: A Reputation-Based Trust Model for P2P Systems. In: International Conference on Computational Intelligence and Security (2009)
22. http://csrc.nist.gov/publications/PubsSPs.html#800-145. The NIST Definition of Cloud Computing. Special Publication 800-145

Cloud Security and Privacy in the Light of the 2012 EU Data Protection Regulation

Andreas Kronabeter and Stefan Fenz

Vienna University of Technology,
Institute of Software Technology and Interactive Systems,
Favoritenstraße 9-11, 1040 Wien, Austria

Abstract. The essential characteristics of cloud computing such as elasticity or broad network access provide many economic benefits for their users, but with these benefits also many security and privacy risks come along. These risks can be generally classified into legal and technical risks. The upcoming general data protection regulation by the European Commission (COM (2012) 11) strengthens the consumer's rights with changes like a single set of European rules and more data protection obligations for organizations. Once the general data protection regulation becomes effective, organizations will have to fulfill more requirements to comply with the law, especially in situations of security breaches or issues about the life cycle and the processing of data. In this paper we describe a framework for the evaluation of cloud service providers in regard to the upcoming EU data protection regulation. The framework shall help service providers to comply with the new regulation, and shall enable consumers to evaluate the security and privacy competencies of cloud service providers.

Keywords: cloud computing, European Union data protection regulation, security, data protection, privacy, evaluation framework.

1 Introduction

Security and privacy issues which come along with cloud computing have grown in significance. The rapidly technological progress makes it difficult for legal regulations, laws and security provisions to be up to date. Virtualization, multitenancy, and outsourcing raise many questions according to how a provider runs his security policy and how he is handling security issues as well as the responsibilities of the user. Relevant work about cloud security risks and recommendations was published by Gartner [5], the National Institute of Technology (NIST) [6], the Cloud Security Alliance (CSA) [7] and the European Network and Information Security Agency (ENISA) [8]. According to Gartner the seven cloud computing security risks users have to face are: (i) privileged user access, (ii) regulatory compliance, (iii) data location, (iv) data segregation, (v) recovery, (vi) investigative support, and (vii) long-term viability. NIST defines trust, multitenancy, encryption and compliance as the key issues of cloud computing [9].

In this paper we present an evaluation framework which should help future as well as current users/providers of cloud computing services to comply with the upcoming EU data protection regulation (COM (2012) 11) [2]. In the following, we (i) introduce the upcoming European data protection regulation (COM (2012) 11) and the legal key changes for data protection in Europe (Section 2), and (ii) present the actual evaluation framework and describe how it supports user/providers at identifying/providing secure cloud services (Section 3).

2 EU Data Protection Regulation - COM (2012) 11

At the beginning of 2012 the European Commission presented their proposal for a comprehensive reform of the EU‘s 1995 data protection rules. The key changes of the *”Proposal for a Regulation of the European Parliament and of the Council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation)”* are [10]:

– Single set of rules applicable across the EU.
– ”Right to be forgotten”: If the user no longer wants his data to be processed and the provider has no legitimate reason to keep it, the data shall be deleted.
– ”Right to data portability”: The user can transfer, without any problems, personal data from one service provider to another one. This is important to avoid vendor and data lock-in.
– Easier access to personal data.
– Clear rules on when the EU law applies to data controllers outside the EU.
– European Data Protection Board as a new supervisory body.
– Obligatory notification of data breaches within 24 hours
– Increased responsibility and accountability for those processing personal data
– More transparency about data handling with a better information policy.
– The right for an individual to refer all cases to their home national data protection authority is claimed.
– The rules of the general data protection regulation will also apply to organizations not established in the EU, if their services are offered in the EU.

2.1 Definitions in the Context of the EU Data Protection Regulation

”Controller” means the natural or legal person, public authority, agency or any other body which alone or jointly with others determines the purposes, conditions and means of the processing of personal data; where the purposes, conditions and means of processing are determined by EU law or Member State law, the controller or the specific criteria for his nomination may be designated by European Union law or by Member State law.

"Representative" means any natural or legal person established in the European Union who, explicitly designated by the controller, acts and may be addressed by any supervisory authority and other bodies in the EU instead of the controller, with regard to the obligations of the controller under this regulation.

"Processor" means a natural or legal person, public authority, agency or any other body which processes personal data on behalf of the controller.

"Main establishment" means the controller's place of establishment in the European Union where the main decisions as to the purposes, conditions and means of the processing of personal data are taken; if no decisions as to the purposes, conditions and means of the processing of personal data are taken in the European Union, the main establishment is the place where the main processing activities in the context of the activities of an controller's establishment in the EU take place. The processor's 'main establishment' means the place of its central administration in the EU.

"Processing" means any operation or set of operations which is performed upon personal data or sets of personal data, whether or not by automated means, such as collection, recording, organization, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, erasure or destruction.

2.2 Territorial Scope

The EU regulation will apply on the processing of personal data in the context of activities of an establishment of a controller or a processor in the EU. It also applies on the processing of personal data of data subjects residing in the EU by controllers not established in the EU, where the processing activities are related to:

– The offering of goods or services to such data subjects in the EU, or
– the monitoring of their behavior.

3 Evaluation Framework

This section presents an evaluation framework for organizations which decide to outsource part of their IT to a cloud service provider. The framework should help to decide if a cloud provider can be assumed as reliable. The areas of relevance are based on the provided information from widely accepted institutions such as NIST or the Cloud Security Alliance. The concerns and risks of these areas are linked with the upcoming EU data protection regulation to understand what a company and provider has to mind and implement to comply with the proposed regulation. The framework highlights the responsibilities for both provider and user.

The different areas of relevance have been already analyzed in the literature. NIST summarized security and privacy issues and recommendations an organization should follow in their "Guidelines on Security and Privacy in Public

Cloud Computing" [9]. The different areas are Governance, Compliance, Trust, Architecture, Identity and Management, Software Isolation, Data Protection, Availability, and Incident Response.

The Cloud Security Alliance published their security guidance for critical areas regarding cloud computing with the focus on governing and operating issues [3]. The governing part includes Governance and Enterprise Risk Management, Legal Issues, Compliance and Audit, Information Management and Data Security, Interoperability and Portability. The operating part includes Traditional Security, Business Continuity, Disaster Recovery, Data Center Operation, Incident Response, Application Security, Encryption and Key Management, Identity, Entitlement, Access Management, Virtualization, and Security as a Service.

The Australian Government provides with their Cloud Computing Security Considerations [11] a checklist of questions, according to security issues an organization has to deal with when using cloud computing.

The described approaches enumerate what an organization has to consider in regard to security and privacy. With our evaluation framework we combine these approaches and further consider the upcoming EU data protection regulation. We provide a checklist for general security and privacy considerations as well as for legal and organizational requirements according to the upcoming EU data protection regulation.

3.1 Legal and Organizational Requirements

Legal and organizational requirements cover governance, service level agreements, support and information, and compliance.

Governance includes the accountability, responsibility and transparency of an organization. To fulfill these requirements certifications and audits are used. Certifications and audits on which users can rely on are important since users are not able to get a complete insight of all security relevant issues. Hence, the provider should provide information about certification such as PCI DSS, ISO / IEC 27001, etc. and audit standards like SAS70 Type II. Third party audits should be a vital part of any assurance program.

Service Level Agreements are a contract between a provider and a user on the level of the provided service. SLAs and Terms of Service are essential to a reliable cloud provider. Service Level Agreements should contain:

– Adequate system availability (uptime, response time)
– Credits in case of outages
– Adequate compensation for a breach
– Notification in cases of failure or critical situations

Support and Information should be made available in a transparent and easily accessible way by the provider. The user should get as much information as possible. Therefore support and documentation by the provider is necessary. The following points should be made available:

- Frequently Ask Questions (FAQ)
- Help Lines and Wikis
- Reaction time on requests
- An extensive documentation about security
- Information about the billing system and the business continuity strategy

Compliance to laws and regulations is the base of every service provider to become reliable. It refers to the organization's responsibility to comply with regulations, laws and standards to assure secure services. With Audits it can be shown that a standard of security is reached but contractual obligations to protect personal information are essential for security and privacy issues. Laws and regulations can change depending on where the data is stored and processed. Legislative obligations (excerpt):

- Health Insurance Portability and Accountability Act (HIPPA)
- Gramm-Leach-Bliley Act (GLBA)
- Federal Information Security Management Act (FISMA)
- Sarbanes Oxley Act (SOX)
- Safe Harbor
- EU Data Protection Directive 95/46/EC

3.2 Legal and Organizational Requirements According to the Upcoming EU Data Protection Regulation

For a controller to comply with the EU data protection regulation in the matter of legal and organizational requirements it is important to consider the following points:

- The Controller needs to designate a representative, which can be any natural or legal person established in the EU. The representative can be addressed by a supervisory authority instead of the controller.
- Article 22 "Responsibility of the controller" contains the implementation of appropriate measures and strategies as well as the adoption of policies so that the processing of personal data is in compliance with this regulation. The measure shall include:
  - According to Article 28 "Documentation"; the controller and processor and, if any, the controller‘s representative, shall maintain documentation of all processing operations. The documentation should be available, on request, to the supervisory authority.
  - Implementation of data security requirements according to Article 30 (described in the data protection section).
  - According to Article 33 "Data protection impact assessment"; the controller or the processor acting on the controller's behalf has to perform an assessment of the impact of the envisaged processing operations, in case that the processing operations present specific risks.

  - According to Article 34 (1) and (2) "Prior authorization and prior consultation"; the controller or the processor has to obtain an authorization from the supervisory authority prior to the processing of personal data.
  - According to Article 35 (1); the controller and processor shall designate a data protection officer, if the processing is carried out by a public authority or the processing is carried out by an enterprise with 250 employees or more.

  To ensure the effectiveness of these measures the controller has to implement mechanism for the verification. The verification shall be carried out by independent internal or external auditors.
- According to Article 24 "Joint Controller"; if a controller decides to determine the purpose, conditions and means of processing personal data jointly with others, the joint controllers have to determine the respective responsibilities to comply with the regulation.
- According to Article 26 "Processor"; a controller shall choose a processor providing sufficient guarantees to implement appropriate technical and organizational measures as well as procedures in such a way that the processing will comply with the regulation. The processing shall be governed by a contract for binding the processor to the controller, in particular the processor shall:
  - act only on instructions from the controller;
  - employ only reliable staff;
  - implement all required measures according to security of processing;
  - support the controller in complying to the data security obligations of the regulation;
  - hand over all results after the end of the processing;
  - make available all information necessary to control compliance.

The controller and the processor have to document the controllers instructions and the processor's obligations listed above. Important to mention is that if a processor processes the data different than instructed by the controller, the processor will be considered as controller according to that processing and has to be applied to Article 24 "Joint Controllers". Moreover, the controller and the processor and, if any, the representative of the controller, shall co-operate, on request, with the supervisory authority in the performance of its duties.

3.3 Data Protection

The protection of data is a vital issue to make a cloud environment secure. A service provider should possess the following points to fulfill data and information protection requirements:

Data Center: A high standard of protection requires the access to information about data centers and the mechanism that are used to secure a data center. The following points about data centers should be considered:

– Quantity. Organizations should provide information about how many data centers are used to store and process data.
– Physical Security. Information about the physical provisions to secure the data centers should exist.
– Data Backup and Data Redundancy. It should be possible to backup and store data in several locations. The user should get information regarding backup procedures.
– Information about the location of the data centers should be provided. In the best case the user can choose where the data will be stored and processed.
– Data loss. The case of data loss should be stated in a contract, SLA or terms of service.
– Data isolation. Due to multi-tenancy and his complexity it is important how data will be isolated.

Data Security

– Data sanitization techniques should be implemented.
– Auditing and Certifications should be verifiable.
– Data Encryption, Key Management. Techniques like PKI, PKCS, KEYPROV (CT-KIP, DSKPP) or EKMI should be implemented.
– Data/Vendor Lock-in. Exit strategies and other options should be stated in a contract.
– Data ownership. It should be clear who possesses the data and who is responsible for it.
– Identity and Key Management. Evidence for the access and authentication is necessary.
– Implementation of incident response strategies.
– Monitoring of data security.
– Implementation of network security strategies.

3.4 Data Protection According to the Upcoming EU Data Protection Regulation

Important to mention for the security of data is again Article 26 which states that a controller has to choose a processor providing sufficient guarantees about the implementation of all technical measures so that the processing will comply with the EU data protection regulation. The processing shall be governed by a contract. In other words the controller has to protect himself legally with a contract otherwise he may be responsible for data breaches.

Data Loss / Data Breach: According to Article 30 "Security of processing"; controller and processor have to ensure with appropriate technical measures an adequate level of security. Both shall take these measures to protect personal data against unlawful or accidental destruction or accidental loss and have to prevent unlawful forms of processing. In particular any unauthorized disclosure, dissemination, access or alteration of personal data.

– Incident Response / Notification: According to Article 31 "Notification of a personal data breach to the supervisory authority"; the controller has to notify the personal data breach to the supervisory authority without undue delay and where feasible within 24 hours after getting aware of it. The processor has to alert and inform the controller immediately after the establishment of a personal data breach. According to Article 32 "Communication of a personal data breach to the data subject"; the controller has to notify the data subjects after informing the supervisory authority without undue delay.
– Sanctions: A breach could result in a fine up to 1.000.000 EUR or in case of an enterprise up to 2% of its annual worldwide turnover. The fines will be imposed by the supervisory authority.

Data / Vendor-Lock in: According to Article 18 "Right to data portability"; a data subject has the right to obtain from the controller a copy of data that is undergoing processing in an electronic and structured format which is commonly used. That means if a controller is choosing a provider the controller is responsible for the provision of those data, this should be stated within a contract.

Data Lifecycle: According to Article 17 "Right to be forgotten and to erasure"; a data subject has the right to obtain from the controller the erasure of personal data relating to them. Further the controller has to implement mechanisms to ensure that the time limits established for the erasure of personal data or for a periodic review of the need for the storage of the data are observed.

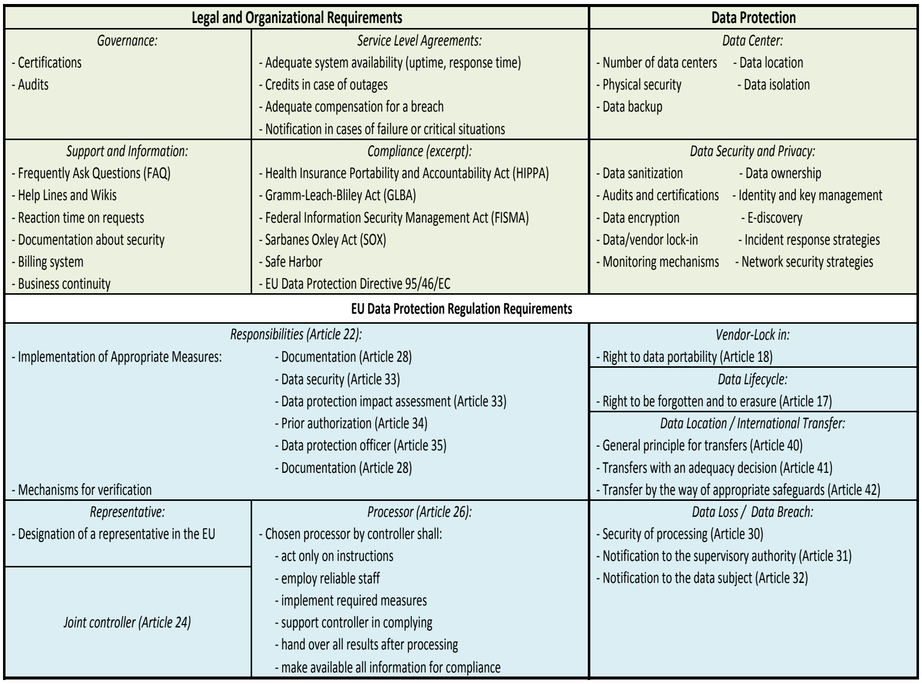
Data Location / International Transfer: The transfer of personal data to third countries or international organization is stated within chapter five of the EU data protection regulation. A controller has to consider the following points:

– According to Article 40 "General principle for transfers"; any processing of personal data to a third country or to an international organization is just permitted if the controller and the processor comply with the conditions of the proposed regulation.
– According to Article 41 "Transfers with an adequacy decision"; if the commission states that the third country, territory or the international organization has an adequate level of protection the transfer may take place. Therefore, the commission publishes in the "Official Journal of the European Union" a list of those countries, territories and international organizations with an adequate level of security and a list of those which don't have an adequate level of security.
– Article 42 "Transfer by way of appropriate safeguards"; discusses the scenario if the commission has taken no decision. In that case the controller or processor has to adduce appropriate safeguards in a legally binding instrument. These safeguards can be provided by

- binding corporate rules which shall specify according to Article 43 "Transfer by way of binding corporate rules"; their legally binding nature; the structure and contact details of the group of undertakings; the data transfer and the type of processing as well as purpose; the general data protection principles; the acceptance by the controller or processor established on the territory; the mechanisms for verification of compliance with the rules; or
- standard data protection clauses adopted by the commission and by a supervisory authority; or
- contractual clauses between the controller or processor and the recipient of the data.

Some exceptions for the transfer of personal data, if the above described points do not exist are stated in Article 44 "Derogations".

Figure 1 summarizes the described evaluation framework. Providers/consumers can use it to review if the legal and technical requirements are given and fulfilled by the provider and consumer. The framework is applicable on all service models and all deployment models of cloud computing. It shall be used by screening the provider and the contractual relationship according to the listed points, and further to check if the own organizational provisions comply with the upcoming EU data protection regulation.



Legal and Organizational Requirements		Data Protection
Governance: - Certifications - Audits	*Service Level Agreements:* - Adequate system availability (uptime, response time) - Credits in case of outages - Adequate compensation for a breach - Notification in cases of failure or critical situations	*Data Center:* - Number of data centers - Physical security - Data backup - Data location - Data isolation
Support and Information: - Frequently Ask Questions (FAQ) - Help Lines and Wikis - Reaction time on requests - Documentation about security - Billing system - Business continuity	*Compliance (excerpt):* - Health Insurance Portability and Accountability Act (HIPPA) - Gramm-Leach-Bliley Act (GLBA) - Federal Information Security Management Act (FISMA) - Sarbanes Oxley Act (SOX) - Safe Harbor - EU Data Protection Directive 95/46/EC	*Data Security and Privacy:* - Data sanitization - Audits and certifications - Data encryption - Data/vendor lock-in - Monitoring mechanisms - Data ownership - Identity and key management - E-discovery - Incident response strategies - Network security strategies

EU Data Protection Regulation Requirements		
Responsibilities (Article 22): - Implementation of Appropriate Measures: - Mechanisms for verification	- Documentation (Article 28) - Data security (Article 33) - Data protection impact assessment (Article 33) - Prior authorization (Article 34) - Data protection officer (Article 35) - Documentation (Article 28)	*Vendor-Lock in:* - Right to data portability (Article 18) *Data Lifecycle:* - Right to be forgotten and to erasure (Article 17) *Data Location / International Transfer:* - General principle for transfers (Article 40) - Transfers with an adequacy decision (Article 41) - Transfer by the way of appropriate safeguards (Article 42)
Representative: - Designation of a representative in the EU *Joint controller (Article 24)*	*Processor (Article 26):* - Chosen processor by controller shall: - act only on instructions - employ reliable staff - implement required measures - support controller in complying - hand over all results after processing - make available all information for compliance	*Data Loss / Data Breach:* - Security of processing (Article 30) - Notification to the supervisory authority (Article 31) - Notification to the data subject (Article 32)

Fig. 1. Cloud Security and Privacy Evaluation Framework

4 Conclusion

In this paper we analyzed the existing work and conditions for an evaluation framework to secure cloud computing in accordance to the upcoming data protection regulation by the European Commission. It is concluded that security and privacy are the major challenge customers and providers have to deal with when using and offering cloud computing services. Due to the proposed data protection regulation an organization deciding to use cloud computing will have to deal with new significant and onerous obligations. Further, also the providers have to upgrade their policies and security implementations. The described framework will help organization as well as providers to comply with the obligations of the upcoming EU data protection regulation. As cloud computing will win on importance in the future, the proposal for a major reform of the European Union legal framework on the protection of personal data is an important step towards securing sensitive data in the cloud.

References

1. The NIST Definition of Cloud Computing - SP 800-145, National Institute of Standards and Technology (2011), `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`
2. Proposal for a Regulation of the European Parliament and of the Council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation), European Commission, `http://ec.europa.eu/justice/data-protection/document/review2012/com_2012_11_en.pdf`
3. Security Guidance for Critical Areas of Focus in Cloud Computing V3.0, Cloud Security Alliance (2011), `https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf`
4. Nimis, J., Tai, S., Baun, C., Kunzem, M.: Cloud Computing: Web-basierte dynamische ITServices. Springer, Heidelberg (2011)
5. Technology Research, Gartner Inc., `http://www.gartner.com/technology/home.jsp`
6. National Institute of Standards and Technology (NIST), `http://www.nist.gov/index.html`
7. Cloud Security Alliance (CSA), `https://cloudsecurityalliance.org/`
8. Securing Europe‘s Information Society (ENISA), `http://www.enisa.europa.eu/`
9. Guidelines on Security and Privacy in Public Cloud Computing - SP 800-144, National Institute of Standards and Technology (2011), `http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf`
10. Commission proposes a comprehensive reform of the data protection rules, European Commission, `http://ec.europa.eu/justice/newsroom/data-protection/news/120125_en.htm`
11. Australian Government (Department of Defense), Cloud Computing Security Considerations, `http://www.dsd.gov.au/infosec/cloudsecurity.htm`

Biometric Identity Trust: Toward Secure Biometric Enrollment in Web Environments

Florian Obergrusberger, Baris Baloglu, Johannes Sänger, and Christian Senk

University of Regensburg
Department of Management Information Systems
93053 Regensburg, Germany

Abstract. The nonrepudiation of a biometric authentication depends on the authenticity of the corresponding biometric profile. If the enrollment process is not controlled by some trusted entity, a user's biometric data might be misleadingly linked to another person's digital identity. To secure the biometric enrollment in open Web-based environments, we propose the biometric observer principle: An arbitrary trustworthy person observes an individual's enrollment at a biometric identity provider and confirms this to the system. The concept rests on a specified trust model, which assesses the trustworthiness of both the observer and the authenticity of an observed biometric profile. Trust relations between observer and observed persons are managed by the authentication system. We implemented a cloud-based biometric identity provider to validate and demonstrate the proposed concept.

Keywords: Authentication, Biometrics, Identity Management, Trust.

1 Introduction

Effective access control to cloud resources requires a high quality of user authentication [18]. A possible way to achieve strong authentication in a very flexible way is the employment of cloud-based biometric authentication services [20]. Before a biometric authentication is possible, an enrollment process has to be passed in order to register a biometric template with the biometric system [9,15]. Therefore it might be necessary to secure the enrollment by restricting access to legitimate persons only. Additionally, this persons have to accomplish the process correctly. To achieve such a secure enrollment, we propose the biometric oberver principle which applies basic ideas from the Web of Trust concept.

The remainder of this paper is structured as follows: Section 2 defines biometrics and secure biometric enrollment. Section 3 refers to the relevant basics of trust and trust models. In Section 4, the conceptual basics for the convergence of trust models with a secure enrollment and a prototype implementation are provided. Section 5 discusses the presented approach and Section 6 summarizes the results and directs future research.

2 Securing the Biometric Enrollment

Biometric authentication is defined as the automated identification or verification of a person using behavioral or physiological characteristics such as fingerprint, palmprint or keystroke dynamics [15]. Basic requisition for an effective biometric authentication is (besides the security and performance of the biometric authentication) a secure prior enrollment [8,9]. Enrollment describes the process where an individual's biometric feature is registered in form of a digital template with the biometric system [15]. After the enrollment is successfully completed the biometric system can be run in two different modes, verification or identification, to authenticate an enrolled user [9]. In verification mode, a user provides his claimed identity and a biometric sample, which is then checked against the corresponding biometric profile stored at the system (1:1 comparison). When operating in identification mode, a user only provides a biometric sample and the biometric system determines the corresponding digital identity based on all available templates (1:n comparison). Compared to traditional authentication techniques based on knowledge (passwords) or tokens, biometric features are inherently and naturally bound to a person. This implicates potential increases regarding both the practicability and nonrepudiation of an authentication [15]. Especially in cases where a person's digital identity is involved in legally binding transactions, a proofable binding between digital identity and the corresponding natural person reduces the risk of fraudulent behavior such as identity theft. To ensure the authenticity of a biometric profile, a trusted entity verifies a natural person's identity by specified means (e.g. identification document) and supervises this person's enrollment process afterward. The observer confirms the enrollment's correct (and secure) accomplishment by authenticating to the biometric system with his own biometric sample.

3 Trust and Trust Models

At first, this section introduces the notion and characteristics of trust. Then some trust models, especially the Web of Trust, are introduced.

3.1 Defining Trust

The notion of trust is a topic that has been discussed in research for years. Although trust has already been analyzed in detail in various disciplines there is no generally accepted definition [13]. This is on the one hand due to the fact that trust is often associated with terms like credibility, reliability or confidence [21]. On the other hand, trust can be contemplated in a cognitive, emotional and behavioral dimension [21]. Oxford Dictionary defines trust as "firm belief in the reliability, truth, or ability of someone or something" [19]. This definition is very close to the definition of "reliability trust", which can be found in literature regarding online trust and reputation systems (e.g. eBay) [17]. Moreover, trust has several characteristics. The following list shows some properties that are important in respect of this work [6,14]:

- *Subjectivity*: Trust is always perceived individually;
- *Fuzziness*: There is a smooth transition between trust and distrust;
- *Direction*: Trust is unidirectionally bound to an entity;
- *Conditional transitivity*: Trust can be transitive. With transitivity, the level of trust decreases.

In order to establish trust toward an entity, different trust models have emerged.

3.2 Trust Models

In literature, various types of trust models can be found. An accepted classification differentiates between policy-based trust and reputation-based trust [2,22]. Policy-based systems mostly address the problem of authorization and access-control [2]. To establish trust, credentials are exchanged [22]. An example for the usage of credentials is the login on a computer, where username and password have to be provided. The possession of these credentials proof the administrator's trust toward the user [2]. In a reputation-based model in contrast, trustworthiness is measured by means of collective referrals or ratings [2,17]. Oxford Dictionary defines reputation as "the beliefs or opinions that are generally held about someone or something" [19]. Hereby the subjective trust is deduced from a combination of personal experience and referrals obtained over social networks or across trust paths [2,22]. For trust paths, transitivity is an important characteristic. Two parties don't need to have direct information about each other, they can rely on the information of a trusted third [2]. A trust model that takes advantage of this property is the Web of Trust. The following example is commonly used to describe this coherence.

Alice, a friend of Carol's knows that Bob's public-key certificate is authentic. Therefore she signs it. Carol however doesn't know Bob. If they want to communicate in private, Bob hands over his public-key certificate. Carol doesn't know if it is authentic by herself. But she sees that Alice signed and trusts it. Hence Carol can trust Bob's certificate in a transitive way [1].

4 Concept and Implementation

Subject of this section is the design and implementation of a system which ensures the authenticity of a biometric profile in open environments. Authenticity refers to the profile's genuineness and trustworthiness by means of a definite identity [10]. For this purpose, we introduce the role of the *observer*, which is a trusted person that supervises the enrollment process.

4.1 Biometric Observer or Four-Eyes Principle

The authenticity of the biometric data captured during the enrollment process should be verified by a trusted instance to prevent fraudulent use. Especially when the enrollment is conducted at home or at an open registry point, this is difficult to implement. For that reason we developed the biometric observer or four-eyes principle, which shall enable a flexible and efficient protection. With this principle, an arbitrary user which is already enrolled, the so-called observer, vouches for the authenticity of the enrollment process and can guarantee for the originality of the biometric profile. The validation of the user identity can be tied to different guidelines. A schematic flow is shown in Figure 1.

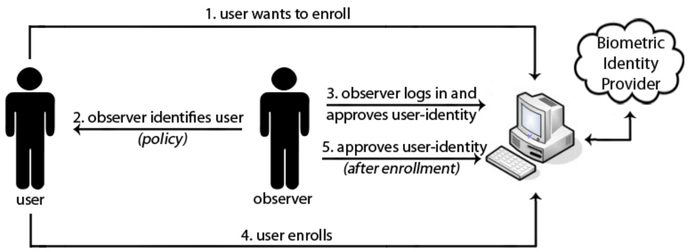

1. user wants to enroll
Biometric Identity Provider
2. observer identifies user
(policy)
3. observer logs in and approves user-identity
5. approves user-identity
(after enrollment)
user
observer
4. user enrolls

Fig. 1. Schematic Flow (Observation)

1. A user wants to create a biometric profile. Therefore he starts the enrollment process, where the name and, if necessary, various identity-related attributes are handed over.
2. To ensure the authenticity of the profile, an already enrolled user, the observer, acts as trusted instance and checks the identity of the user.
3. The observer logs in with his biometric profile, verifies the identity of the enrolling user and, if required, specifies by which means this verification was conducted.
4. The user starts providing his biometric data (enrollment).
5. When the enrollment process is completed, the observer approves the accuracy of the process.

By means of this method, trust can be established across several steps. If Alice observed Bob for example, she can trust Carol's and Dave's profiles transitively, whose enrollment processes were observed by Bob. The level of trust however decreases in this coherence. These trust relations can be described within a directed graph. Every profile is represented through a node in the graph and the relations are directed edges. In this scenario, the distance of two nodes is crucial for the level of reliability.

In a model where Alice observes Bob during the enrollment process (Figure 2), Bob in contrast just is observed and does not prove the identity of his observer (Alice), there is only a one-way relationship. Hereby every single user builds his own tree of trust with himself being the anchor. As a consequence, Alice will never be part of Bob's tree of trust, since there are only trust relations to one's followers. From a global perspective this leads to a hierarchy, a tree with the system administrator on top of it as global trust anchor that enrolled at the beginning without observation. To establish a Web of Trust, in which all nodes can potentially trust each other, a subsequent approval of a profile's

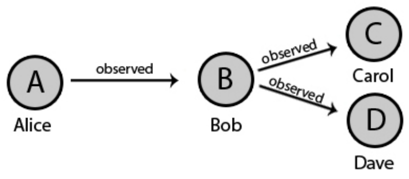


Fig. 2. Trust Relationship Tree

authenticity must be possible, to build a bilateral trust relationship. This takes us to the second method to proof the authenticity of a profile, the *confirmation.*

In contrast to the observation process, the confirmation is carried out between two already enrolled users. Analogous to the observer, the role of the confirmer is introduced. Figure 3 shows a generic confirmation process. With the confirmation, bilateral trust relations can be established. Moreover, the trustworthiness can be increased after the completion of the enrollment process. This leads to a Web of Trust.

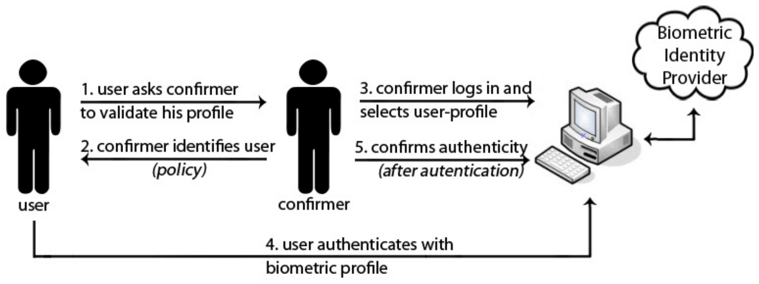

Biometric Identity Provider
1. user asks confirmer to validate his profile
2. confirmer identifies user (policy)
3. confirmer logs in and selects user-profile
5. confirms authenticity (after autentication)
user
confirmer
4. user authenticates with biometric profile

Fig. 3. Schematic Flow (Confirmation)

In Figure 4 Alice observed Bob's enrollment and Bob confirmed the authenticity of Alice's biometric profile afterward. Hence there is a bilateral trust relationship. The relation between Alice and Dave however is different. Alice can trust Dave's profile transitively. Dave confirmed the validity of Alice's profile and therefore has a direct trust relation toward Alice.

4.2 Trust Metric

To make the level of trust measurable, a trust metric is necessary. Since the literature concerning trust metrics has been growing rapidly during the last years, a lot of trust metrics exist [11]. Some of them could certainly be used to solve this problem. In this work we point out what requirements a trust metric has to meet and what it could look like. The described metric should be understood as an example. As mentioned in section 3.1, trust is subjective

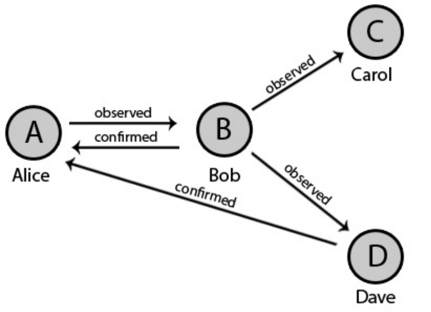


Fig. 4. Bilateral Trust Relationship (Web)

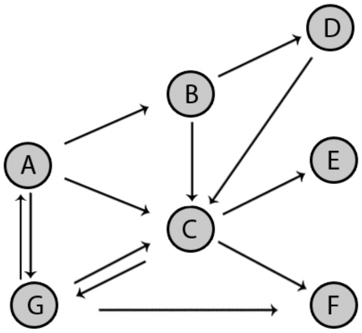


Fig. 5. Web of Trust (Global View)

and perceived individually. Thus the metric has to represent the level of trust customized from the perspective of each node. Hereby the distance of two nodes, by reason of the decreasing level of trust with transitivity, and the reputation of a node in the Web of Trust have to be included. The problem in general is to some extend related to the rating problem regarding websites in the Google search algorithm. The so-called PageRank calculates the reputation level of a website on base of the reputation of the linking pages [7]. In contrast to the PageRank, the trust level of a node in this metric is no global value. It has to be calculated individually from the perspective of each node. Hence, the following requirements were set up for the metric:

1. The node, from whose perspective the trust value of the other nodes is measured, is the "root" node. All edges to the root node are not considered. The root node has the trust value 1.
2. A trust value is calculated for all nodes of the web that can be reached over a trust path from the root node. The trust values of these nodes are within the interval $]1;\infty[$. The closer the trust value is to 1, the higher is the level of trust. For all nodes that can't be reached from the root over a trust path, the value 0 is assigned. The value 0 means that there is no trust relationship at all. Additionally the maximal length of a trust path can be defined in order to limit the size of the web.
3. The final trust value is calculated on base of two factors: (a) the direct trust factor, which is the distance between the root and a considered node. The distance is the length of the shortest path between two nodes. The length is the number of edges a path uses. The distance between any node and itself is 0. With every additional node on the trust path the distance is increased by 1. (b) the reputation factor, which includes the reputation derived from all trust paths that point to the node. A node must not appear twice in a trust path.

These requirements lead us to the following exemplary recursive function, inspired by the PageRank:

$$T_A(N_X) = \underbrace{d(N_A, N_X)}_{direct-trust-factor} + \underbrace{\left(e^{r\left(\left(\sum_{i=1}^{n} \frac{-1}{T_A(N_i))}\right)\right)}\right)}_{reputation-factor}$$

$T_A(N_X)$:	*Trust Value of a node X from node A's perspective*
$d(N_A, N_X)$:	*Distance between node A and node X*
r:	*Reputation weight parameter*

To calculate the trust value of a node X, $T_A(N_X)$, the length of the shortest path to node X is determined. Then the reputations factor corrects the value depending on its reputation. The parameter r can be chosen individual in order to weight the importance of the reputation factor. In our example r=0.75.

To demonstrate this function, an example is provided. Figure 5 shows an exemplary Web of Trust. The paths show directed trust relations, derived from observation or confirmation. In Figure 6 it is evident that the trust value increases (the trust level decreases), while moving away from the root node A. Node C has the highest level, apart from the root, because it is very near to A and has a high reputation in the web. Node B however has a considerably lower level, because there are no other trust paths but the one from Node A (Figure 7). Node F has a comparatively high trust level since there are trust paths from high level nodes (C and G) although it has no direct relation to the root. From node B's perspective, the trust values are different. Node A for example has a significant low level compared to the other nodes, because the transitive trust path has a length of 4.

Since observation and confirmation are rated equally in this metric, the obervation could be renounced during the enrollment process. In this case, a profile is untrusted at the beginning. The scope of an untrusted profile however must be restricted until the authenticity is proved by confirmation. This supports scenarios where a minimum level of enrollment security (and quality) is required.

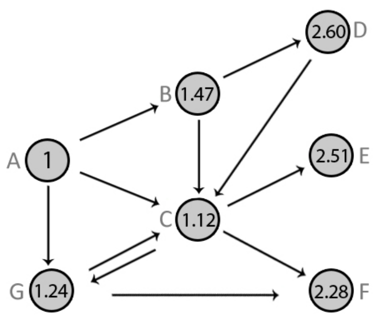


Fig. 6. Trust Values (Perspective A)

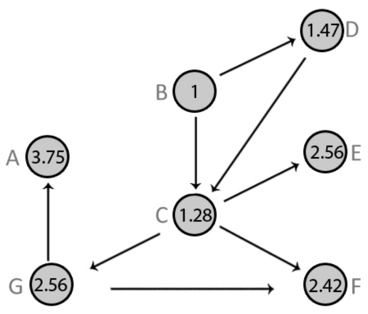


Fig. 7. Trust Values (Perspective B)

4.3 Prototype Implementation

To implement the described observer principle, a cloud-based biometric authentication system was developed. Biometric systems require suitable biometric reading devices (sensors) to collect and to digitize an individual's biometric raw data [15]. Here, it explicitly depends on the respective applied method, which kind of sensor is needed. For instance, whereas voice, face, or keystroke data can be acquired with common and standardized devices such as microphones, webcams and computer keyboards, procedures like iris or fingerprint recognition require specific dedicated sensors, thus restricting the applicability of such methods. Consequently, for the application in open environments (e.g. public cloud computing), the former methods are preferable. Below, we particularly apply keystroke dynamics. Keystroke dynamics is determined by unique characteristics such as speed, rhythm and the continuity and precision of typing [4,5]. These characteristics are represented by a combination of key events, that is, pressing and releasing of a key as well as hold and transition periods [3,16].

The current prototype implementation of the four-eyes principle allows the biometric system's administrator to enable an observed enrollment for new users. In this case the administrator is in charge of selecting observers to supervise new users' enrollment processes. The biometric system's administrative graphical user interface allows the assignment of a certain observer and invites the respective user to enroll. This invitation is sent via e-mail which also contains a one-time access token to the enrollment application. This collects typing samples from the user and generates the biometric template. After the user successfully finished the enrollment process the application demands for the observer to authenticate biometrically. Thus it is possible for new users to create a biometric profile and enroll all over the world, as long as an observer is available.

5 Discussion

This work aims for increasing the security of the biometric enrollment process by implementing the four-eyes principle. Here, the quality and security of the biometric authentication system is out of scope and not considered by the model developed. For a secured enrollment, an observer already known to the biometric system supervises the enrollment process of another person. The observer verifies this physical person's identity and then confirms the binding to the digital identity created. Therefore the observer's trust in observed persons' digital identities is strengthened.

Referring to the Web of Trust model, other individuals trusting the observer's digital identity also benefit from the observed enrollment. Because the newly enrolled user's digital identity is on their trust path, the conditional transitivity of trust allows them to calculate a trust value for it. Another positive effect of such an observed enrollment is the possibility to decrease the number of failed enrollments. Since the observer has to be enrolled to the biometric system, he is already familiar with the enrollment process and can help the enrolling person to avoid mistakes. The proposed four-eyes principle can be used for both operational modes of biometric systems, verification and identification, since both modes aim for authenticating a person and confirm the binding between digital identity and the natural person behind. Because biometric profiles and trust relations are maintained by the biometric system, it is responsible to ensure the authenticity of this data. If a person wants to prove his trust in other persons' digital identities, he cannot do this on his own, he has to rely on the information provided by the biometric system instead. A decentralized approach in which participants inform each other about their trust relations would release the biometric system from maintaining the trust relations, but ensuring the authenticity of the biometric profiles would still lie in the biometric system's area of responsibility.

Because a user to be observed and a potential observer do not initially know each other, the user has to discover a qualified one and physically meet him. This requires efforts regarding coordination and travelling and is not explicitly supported by the system proposed.

6 Conclusion

To secure the biometric enrollment in Web-based environments, we propose the biometric observer principle and provide a respective prototype implementation. The concept applies major ideas of the Web of Trust. The supervision of a user's enrollment by an observer increases the authenticity of the created biometric template. A comprehensive trust model enables the subjective formalization of the trustworthiness of the biometric identities of both observers and other entities. The relations between observer and observed persons are maintained in the system's database.

Future work should include the design of a user-based trust-metric configuration and the convergence of the four-eyes principle with a public key infrastructure to allow users to sign trust paths and biometric templates.

References

1. Abdul-Rahman, A.: The PGP Trust Model. EDI-Forum: The Journal of Electronic Commerce 10(3), 27–31 (1997)
2. Artz, D., Gil, Y.: A survey of trust in computer science and the semantic web. Web Semant. 5(2), 58–71 (2007)
3. Bakdi, I.: Benutzerauthentifizierung anhand des Tippverhaltens bei Verwendung fester Eingabetexte. Universitäts Verlag, Regensburg (2007)
4. Bartmann, D., Bakdi, I., Achatz, M.: On the Design of an Authentication System Based on Keystroke Dynamics Using a Predefined Input Text. International Journal of Information Security and Privacy 1(2), 1–12 (2007)
5. Bergando, F., Gunetti, D., Picardi, C.: User Authentication through Keystroke Dynamics. ACM TISSEC 5(4), 367–397 (2002)
6. Bless, R., Mink, S., Blaß, E.-O., Conrad, M., Hof, H.-J., Kutzner, K., Schöller, M.: Sichere Netzwerkkommunikation: Grundlagen, Protokolle und Architekturen. Springer, Berlin (2005)

7. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. 30(1-7), 107–117 (1998)
8. Dorfner, M.: Evaluation und Weiterentwicklung von Zertifizierungsverfahren für biometrische Systeme: Eine exemplarische Betrachtung von Zertifizierungsverfahren mit dem Schwerpunkt IT-Sicherheit. Schriftenreihe Studien zur Wirtschaftsinformatik, Kovač (2012)
9. Dotzler, F.: Datenschutzrechtliche Aspekte und der Einsatz biometrischer Systeme in Unternehmen: Eine exemplarische Betrachtung von Systemen auf der Grundlage des biometrischen Merkmals Tippverhalten. Kölner Wissenschaftsverlag, Köln (2010)
10. Eckert, C.: IT-Sicherheit: Konzepte, Verfahren, Protokolle, 6th edn. Oldenbourg, München (2009)
11. Gómez Mármol, F., Martínez Pérez, G.: State of the art in trust and reputation models in P2P networks. In: Handbook of Peer-to-Peer Networking, pp. 761–784. Springer (2010)
12. Grandison, T., Sloman, M.: A survey of trust in internet applications. IEEE Communications Surveys Tutorials 3(4), 2–16 (2000)
13. Herzig, A., Lorini, E., Huebner, J.F., Vercouter, L.: A logic of trust and reputation. Logic Journal of IGPL 18(1), 214–244 (2010)
14. Huang, J., Fox, M.S.: An ontology of trust: formal semantics and transitivity. In: Proceedings of the 8th International Conference on Electronic Commerce: The New e-commerce: Innovations for Conquering Current Barriers, Obstacles and Limitations to Conducting Successful Business on the Internet, ICEC 2006, pp. 259–270. ACM, New York (2006)
15. Jain, A., Flynn, P., Ross, A. (eds.): Handbook of Biometrics. Springer, New York (2007)
16. Janakiraman, R., Sim, T.: Keystroke Dynamics in a General Setting. In: Lee, S.-W., Li, S.Z. (eds.) ICB 2007. LNCS, vol. 4642, pp. 584–593. Springer, Heidelberg (2007)
17. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decis. Support Syst. 43(2), 618–644 (2007)
18. Mather, T., Kumaraswamy, S., Latif, S.: Cloud security and privacy (an enterprise perspective on risks and compliance), 1st edn. Theory in practice. O'Reilly, Sebastopol (2009)
19. Oxford Dictionaries (visited on April 15, 2012)
20. Senk, C., Dotzler, F.: Biometric Authentication as a Service for Enterprise Identity Management Deployment: A Data Protection Perspective. In: Sixth International Conference on Availability, Reliability and Security, ARES 2011, Vienna, Austria, pp. 43–50. IEEE (2011)
21. Wang, Y.D., Emurian, H.H.: An overview of online trust: Concepts, elements, and implications. Computers in Human Behavior 21, 105–125 (2005)
22. Work, F., Bonatti, P.A., Shahmehri, N., Duma, C., Olmedilla, D., Nejdl, W., Baldoni, M., Baroglio, C., Martelli, A., Coraggio, P., Antoniou, G., Peer, J., Fuchs, N.E.: Rule-based policy specification: State of the art and future work (2004)

Future of Cloud-Based Services for Multi-factor Authentication: Results of a Delphi Study

Christian Senk

University of Regensburg
Department of Management Information Systems
93053 Regensburg, Germany
christian.senk@uni-r.de

Abstract. The *Cloud Computing* model potentially leverages the diffusion of strong multi-factor authentication systems. In order to systematically evaluate the future of cloud-based services for multi-factor authentication, a 3-rounded *Delphi* survey with experts in the German-speaking area was conducted. Results indicate the substantially increasing importance of such services in both organizational and user-centric application fields. Furthermore, seven primary success factors have been identified. Most critical are factors regarding the ease of adoption as well as security- and compliance-related issues.

Keywords: Authentication, Delphi Study, Cloud Computing.

1 Introduction

While for many use cases basic password-based user authentication is considered to become too insecure, there are substantial barriers regarding the adoption of strong(er) multi-factor authentication systems. Here, on the one hand side, the *Cloud Computing* model opens up opportunities to lower related barriers and to drive the adoption; on the other hand side, inherent risks might significantly restrict the applicability of related systems. In this context, we investigate following research questions (RQ) to assess the future application of such systems:

- **RQ1:** How will the practical relevance of cloud-based services for multi-factor authentication develop and which authentication methods will prevail?
- **RQ2:** Which are relevant practical application fields for such systems?
- **RQ3:** Which requirements are critical for the diffusion of such systems (referred to as *success drivers*) and should thus be reflected by service providers?

Since for this purpose no comprehensive data is available, an expert survey is conducted applying the *Delphi* method. The remainder of this paper is structured as follows: Section 2 explains the theoretical background and related work. Section 3 lays out the research design including the applied method as well as the justification of its application. The findings are set forth and discussed in section 4. Section 5 finally summarizes this paper and directs future research.

2 Theoretical Background and Related Work

This section sets forth the paper's theoretical fundamentals as well as related work in the field of cloud-based authentication services.

2.1 Cloud Computing

According to the *National Institute of Standards and Technology* (NIST), *Cloud Computing* is defined as a "model for enabling convenient on-demand network access to a shared pool of configurable computing resources [...] that can be rapidly provisioned and released with minimal management effort or service provider interaction" [15]. Cloud services refer to resources at the infrastructure, platform or application layer and provide specific advantageous characteristics such as multi-tenancy, easy standardized access through thin clients, scalability of the underlying infrastructure, and automated self-service provisioning [11, 14, 15]. Hence, the most frequently mentioned obstacles are concerns regarding security and compliance, but also issues related to the ease of integration with existing systems and possible lock-in effects [11, 14].

2.2 Authentication

Users can generally be authenticated using knowledge-based, token-based or biometric methods [12]. Most systems implement basic PIN- or password-based mechanisms (knowledge) [4]. However, because of several inherent drawbacks, the strength of authentication of knowledge-based mechanisms is considered to be insufficient for many applications [5, 18]. A possible way to increase this strength is to replace or to supplement existing controls with token-based procedures (e.g. one-time password (OTP) generators) or biometric methods (e.g. face recognition, keystroke dynamics) [4, 8, 12]. The combination of different kinds of authentication methods is referred to as *multi-factor authentication* [4, 12].

2.3 Authentication as a Service

The application of security services according to the *Cloud Computing* model is referred to as *Security as a Service*, SECaaS) and, accordingly, promises additional specific benefits compared to on-premises solutions or traditional security service outsourcing [1, 9, 17]. A study conducted by the author in 2011[1] discovered that statistically, there are three drivers for the adoption of SECaaS:

– **Perceived Ease of Adoption:** Degree to which the adopter believes that the SECaaS adoption is effortless, both technically and organizationally speaking;

[1] Survey was conducted in 2011 in cooperation with the *German Federal Association for Information Technology, Telecommunications and New Media* (BITKOM e.V., see: `http://www.bitkom.de`); detailed data is not published, yet.

- **Perceived Usefulness:** Degree to which the adopter believes that the adoption increases its performance; this includes cost- and quality-related benefits;
- **Trust:** Degree to which the adopter believes that the adoption is free of risks, which includes mainly security-related but also social and strategic risks.

Below, cloud-based systems for (strong) user authentication are referred to as *Authentication as a Service* (AaaS). Such systems are operated and maintained by *Authentication Service Providers* (ASP) in order to determine a user's identity by specified means and to assert this to respective target systems. Here, it must be noted that AaaS regards user authentication *from* the cloud and not *within* existing cloud systems [e.g., 2]. The results of the aforementioned survey emphasizes the relevance of AaaS. Of 164 participating organizations, 12.8% plan to invest in cloud-based services for multi-factor authentication within the next three years. In the medium and long run further 7.9% intend to use such systems. Findings of Forrester Research support this. According to a survey among 324 IT security decision-makers conducted in 2008, 75% were planning or considering changes or upgrades to their customer authentication processes; 72% showed general interest in AaaS [7].

3 Research Design

In the first part of this section, the basic content-related concept of the study is laid out which includes a total of 50 hypotheses (H). The applied methodology is introduced and justified afterward.

3.1 Concept

RQ1: Development (H1-H4). We initially argue that the relevance of AaaS is induced by an increasing demand for strong (multi-factor) authentication and a hypothesized decreasing significance of inherently weak knowledge-based authentication methods (H2). Thus, we not only expect the increasing importance of such systems (H1) but also of strengthening biometric (H3) and token-based authentication methods (H4) required to implement AaaS systems. To investigate this development, we intend to evaluate the general relevance of AaaS as well as authentication approaches today, short-, medium- and long term.

RQ2: Application Fields (H5-H19). Since the respective type of an AaaS consumer implicates different individual requirements (e.g. regarding service level agreements (SLA), interface design), one must differentiate whether it is an organization that adopts such a service or a private person employing it autonomously. Based on related literature [e.g., 4, 12, 14], we identified possible networked application fields which were then hypothesized regarding their potential relevance for AaaS employments (see result tables 2 & 3).

RQ3: Success Factors (H20-H50). Since the success of AaaS solutions directly depends on its adoption, we systematize possible success factors according to the aforementioned adoption drivers. Furthermore, to enable deeper insights, we differentiate success factor candidates at the different levels of an AaaS solution. This includes the *system* implementation itself, one or more implemented authentication *methods*, and organizational attributes specific to a *provider* (ASP), offering at least one system. All hypothesized items are derived from related literature [e.g., 3, 6] (see result tables 4, 5 & 6).

3.2 Applying the Delphi Method

The *Delphi* method can be defined as a structured group communication process which allows individuals to deal with complex problems and has proven to be a popular instrument in IS research and technology forecast [13, 16]. Here, classical studies are characterized by the following attributes [10]: (1) Survey of selected experts; (2) use of standardized questionnaires; (3) anonymity of individual responses; (4) calculation of statistical group answers; (5) iteration of the survey; (6) provision of the group answers (controlled feedback) to the respondents.

The novelty, complexity and specificity of this paper's research object requires the involvement of declared experts in related fields (e.g. *Cloud Computing*). Compared to alternative approaches like group discussions or expert surveys, the *Delphi* method tends to reveal more reflected and thus better expert judgment [10]. Major drawback, on the other hand side, is a higher expenditure of time due to additional survey rounds conducted [10]. Essential for a high quality of a *Delphi* study's generated results is the selection of experts with an appropriately deep understanding of the research topic [10]. Related literature suggests a panel size of 10-18 individuals or more which are selected non-randomly by the *Delphi* monitoring team [10, 13, 16]. The panel should furthermore be composed interdisciplinary to cover a more faceted set of expert opinions [10].

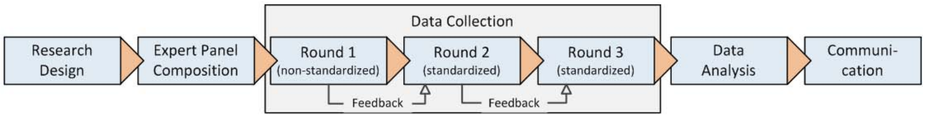


Fig. 1. Process Model of the Study

The study follows the process depicted by figure 1. In a first step, the research questions were specified, related contents systematized and a measurement model derived. Afterward, potential experts were identified and selected to join the expert panel. The expert panel was initially questioned in a non-standardized form (Round (R) 1), and then in two successive standardized survey rounds (R2 and R3) with controlled feedback. After completion, the data was analyzed and key findings were distributed to all active panel members.

4 Findings

Below, the outcome of the conducted survey is laid out.

4.1 Composition of Expert Panel

Potential experts were appealed, informed about this study and its objective, and invited to apply via e-mail reasoning why and how they could contribute to this topic[2]. Then, the panel was composed. Of 39 candidates 36 experts were selected. All experts provide at least 3 years of experience in related fields. R1 was completed by 34 and R2 by 32 persons. The last round revealed 24 responses. This corresponds to a total panel mortality rate equals 33.3%. The final panel was composed almost equally of experts of the fields *consumer* (34.5%), *provider or developer* (34.5%), and *research* (31.0%)[3]. Details about the panel composition and its development are depicted by figure 2.

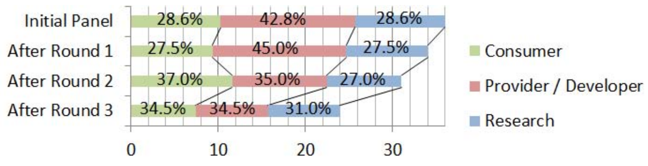


Fig. 2. Composition of the Expert Panel

4.2 Data Collection

The data collection was carried out from January to April 2012. The first round revealed 34 responses to two open questions regarding the most important (1) *application fields* and (2) *success factors* as perceived by the experts. The unstructured answers were mapped to the existing measurement model. This first (non-standardized) round was conducted via e-mail or telephone interview and was used both to double-check the completeness of the designed model and to determine the intuitively most named items. Afterward, the measurement model was translated into a standardized online questionnaire for R2 and pre-tested by 10 IT security master students and the research team of the partner project *SkIDentity*[4]. To provide for feedback in the 2nd (and 1st standardized) survey round the previously most named items were highlighted accordingly. R2 and R3 were conducted consecutively online including both open and closed questions. The survey of R3 contained the visualized statistical group answers of R2. Furthermore, after R2, we removed non-significant items.

[2] For this, IT professionals of the network of BITKOM e.V. were contacted. Additionally, declared experts were directly addressed via XING, see `http://www.xing.de`.
[3] Multiple answers were permitted.
[4] See, `http://www.skidentity.com/`

4.3 Results

Subject of this sub-section is the description and analysis of the gathered data.

RQ1: Development. The development of the significance of AaaS and (independently) general authentication approaches is illustrated by figure 3. Though the relevance of AaaS is evaluated to be rather low within the next three years, in the medium to long run the panel forecasts a significant increase and a respective high importance (Median=4)[5]. A congruent development is expected for token-based authentication methods, indicating the dependence of AaaS on such methods. This is supported by the evaluation of implementable authentication procedures. The panel was asked to rank the five most relevant methods regarding the implementation of AaaS in the medium and long term. Here, token-based methods performed clearly better than all other biometric or knowledge-based procedures, both for private and business user-centric applications. Table 1 summarizes the results ordered by average rank (business). The data also suggests a significant decrease of the relevance of knowledge-based methods from currently *very high* to *medium*. The importance of biometrics correlates negatively and increases from *very low* to *medium* and even subtends the curve for knowledge-based procedures. H1–H4 are supported.

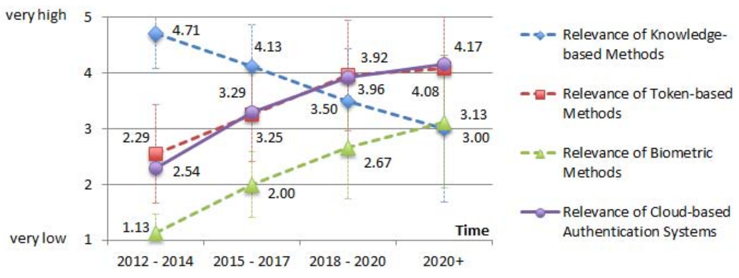


Fig. 3. Development of the Relevance of Authentication Methods and AaaS (n=24)

RQ2: Application Fields. Possible application fields were rated by the panel on a 5-point *Likert* scale with following semantics: [1] *absolutely not relevant*, [2] *rather not relevant*, [3] *neutral*, [4] *rather relevant*, [5] *absolutely relevant.* Regarding organizational application fields, the data indicates the significant relevance of AaaS for the authentication of partners or corporate customers in a federation, the enhancement of *Identity & Access Management* (IAM) systems, the protection of outsourced or cloud applications, and the authentication of private end users in the public sector. An item was rated to be relevant for a median greater than or equal to 4. Statistical details are summarized in table 2. Of 24 experts, 58% name legal or regulative requirements as primary reason for the adoption of AaaS. Business partner demands are secondary (25%). The evaluation of the extent of pain implicated by these drivers was approached looking

[5] For all tests regarding median values, in this and the following section, the (non-parametric) *One-Sample Wilcoxon Signed Rank Test* was applied with $\alpha = 5\%$.

Table 1. Ranking of Authentication Methods regarding AaaS Application

Authentication Method	Type	∅ Rank Business	∅ Rank Private
Hardware-based security token with dedicated reading device	Token	2.52	4.78
Hardware-based security token for OTP	Token	3.30	5.50
Password-protected private keys and certificates	Token	4.28	6.65
Software token (e.g. OTP via smartphone appl.)	Token	5.09	2.72
Text-based password or PIN	Knowledge	8.74	7.04
Fingerprint recognition	Biometrics	9.96	9.72
Keystroke dynamics (text-dependent)	Biometrics	10.07	10.72
Face recognition	Biometrics	10.43	10.02
Voice recognition (text-independent)	Biometrics	10.57	10.11
Keystroke dynamics (text-independent)	Biometrics	10.63	9.70
Hand vein structure recognition	Biometrics	10.65	16.00
Dynamic signature recognition	Biometrics	10.76	10.35
Social knowlegde-based procedures	Knowledge	10.89	10.20
Graphical passwords	Knowledge	10.63	9.50

at the relative value of strong authentication. In this regard, the panel was asked to estimate the average value of strong user authentication proportionally to the value of the respective transaction or business application to be protected. The result was 14.54%. Furthermore, the data indicates that AaaS is most relevant for web-enabled applications involving high protection needs. Regarding even higher security needs (*critical*), applications AaaS is not feasible due to inherent cloud challenges.

The user-centric adoption of AaaS shows promise for the protection of (semi-) critical processes both for private and public applications. Table 3 lists all rated items and the corresponding test results.

Table 2. Organizational Application Fields

Application	Mean	SD	Median	Min	Max	H
Authentication of partners or corporate customers	4.08	0.78	4	3	5	H10+
Authentication of private end users in the public sector	4.08	0.97	4	1	5	H12+
Protection of outsourced (cloud-) applications	4.00	0.83	4	2	5	H9+
Functional extension of IAM systems	3.75	0.74	4	2	5	H6+
Protection of network access points	3.75	0.99	4	1	5	H8-
Composition to more significant business service	3.67	1.05	4	1	5	H13-
Protection of infrastructure resources	3.58	1.25	4	1	5	H5-
Authentication of private customers for commercial use cases	3.58	0.93	4	1	5	H11-
Dedicated protection of internal applications	3.08	0.83	4	1	4	H7-

Table 3. User-centric Application Fields

Application	Mean	SD	Median	Min	Max	H
(Semi-) critical public applications (e.g. e-Government)	4.21	1.02	4	1	5	H15+
(Semi-) critical private applications (e.g. e-Banking)	4.13	1.03	4	2	5	H16+
Private cloud storages and synchronisation services	3.63	1.06	4	1	5	H18-
Innovative / future applications (e.g. e-car infrastructures)	3.46	1.06	3	2	5	H19-
Global user-centric web single sign-on	3.13	1.15	3	1	5	H14-
Less critical processes or applications (e.g. social networks)	2.68	1.14	3	1	4	H17-

RQ3: Success Factors. For the determination of the success factors, the panel had to rate each hypothesized item on a 5-point *Likert* scale with following semantics: [1] *absolutely not critical*, [2] *rather not critical*, [3] *neutral*, [4] *rather critical*, [5] *absolutely critical.* An item is considered to be a *weak* success factor [+] when its median is significantly equal to or greater than 4.0, a *moderate* success factor [++] when it is (additionally) equal to or greater than 4.5, or a *strong* success factor [+++] for a median equals 5.0. The remaining items were evaluated to be no success factor at all [o] causing the falsification of the corresponding hypotheses. The factors already eliminated after R2 are also enlisted (labelled [*]). The analysis of all success factor candidates is summarized by table 4 (*method*-related), table 5 (*system*-related) and table 6 (*provider*-related)

Table 4. Evaluation of Factors at the Method Level

Factor	Mean	SD	Median	Min	Max	Relevance	H
Ease of use and user acceptance	4.88	0.34	5	4	5	+++	H22+
Transparency & data protection performance	4.29	0.81	4,5	3	5	++	H23+
Independence from dedicated hardware or software	4.00	0.78	4	2	5	+	H20+
Security and strength of the authentication	3.96	0.75	4	2	5	+	H24+
Time-efficient usability	3.92	0.65	4	2	5	+	H21+
Reachability of confidentiality	3.42	1.06	3,5	2	5	o	H27-
Reachability of non-repudiation	3.33	1.01	3	2	5	o	H26-
Scalability of the strength of authentication	3.17	0.76	3	2	5	o	H25-

Table 5. Evaluation of Factors at the System Level

Factor	Mean	SD	Median	Min	Max	Relevance	H
Transparency and usability of the system	4.50	0.51	4,5	4	5	++	H31+
Data security from the consumers' point of view	4.42	0.72	5	3	5	++	H32+
Service access and use by any device	4.29	0.81	4,5	3	5	++	H37+
Ease of technical service integration	4.29	0.62	4	3	5	++	H29+
Comprehensibly secure system interfaces	4.00	0.78	4	2	5	+	H34+
High availability and immediate service recovery	3.96	0.69	4	3	5	+	H33+
Low total costs for service use	3.92	0.72	4	3	5	+	H28+
Reachability of a high strength of authentication	3.83	0.64	4	2	5	+	H36+
Ability to scale and to customize function range	3.50	0.83	3	2	5	o	H38-
Existing integration with relevant target systems	3.42	1.06	3,5	2	5	o	H30-
Management and provisioning of user attributes	3.25	0.94	3	2	5	o	H39-
Ability to (ex-)port user application data*	3.10	0.98	3	2	5	o	H35-
Usability in private and business environments*	2.87	1.18	3	1	5	o	H40-

Table 6. Evaluation of Factors at the Provider Level

Factor	Mean	SD	Median	Min	Max	Relevance	H
Market visibility and reputation of the ASP	4.42	0.65	4,5	3	5	++	H45+
(External) Auditability	3.92	0.78	4	3	5	+	H47+
Flexible and customer-oriented licensing models	3.88	0.85	4	2	5	+	H41+
Transparent spec. of legal consequences & effects	3.83	0.70	4	3	5	+	H44+
Location of the ASP and its infrastructure	3.79	1.06	4	1	5	o	H50-
Comprehensive certification	3.71	0.95	4	2	5	o	H46-
Differentiated & standardized SLA	3.67	0.64	4	3	5	o	H42-
Customer support	3.58	0.72	4	2	5	o	H48-
Ability to customize SLA*	3.10	0.72	4	1	5	o	H43-
Synergy effects with other services*	2.70	0.72	4	1	5	o	H49-

including descriptive statistics as well as the evaluation of the relevance of each item and of the corresponding hypothesis[6]. The most substantial and only *strong* success factor is *User acceptance and Ease of Use* of an implemented authentication method. Furthermore, six *moderate* success factors have been identified, four at the system level and each one at the method and provider level.

4.4 Discussion and Implications

According to the experts' judgement, AaaS is a significant future technology for both private users and organizations in order to increasingly replace or supplement existing password-based authentication with stronger methods. Primary authentication methods will be token-based; biometric ones are evaluated to be rather supplementary even in the medium to long run. Considering the determined success factors, possible reasons might, for instance, include an expected lower end user acceptance or data protection-related concerns [e.g., 4, 12]. However, actual reasons must be investigated in more detail and in regard to specific use cases.

Private user-centric applications include public fields such as e-Government and rather critical private web-based services such as e-Banking. Here, mainly soft tokens and device-dependent hard-tokens will be used for the implementation of AaaS. Expert feedback furthermore indicates that services for public applications will mainly be based on electronic identity cards (eID) while private scenarios will utilize more ubiquitous soft-token-based methods. For organizational and business user-centric applications, hardware tokens based on dedicated reading devices promise highest security [4] and are despite of involved costs clearly most important for the implementation of AaaS systems.

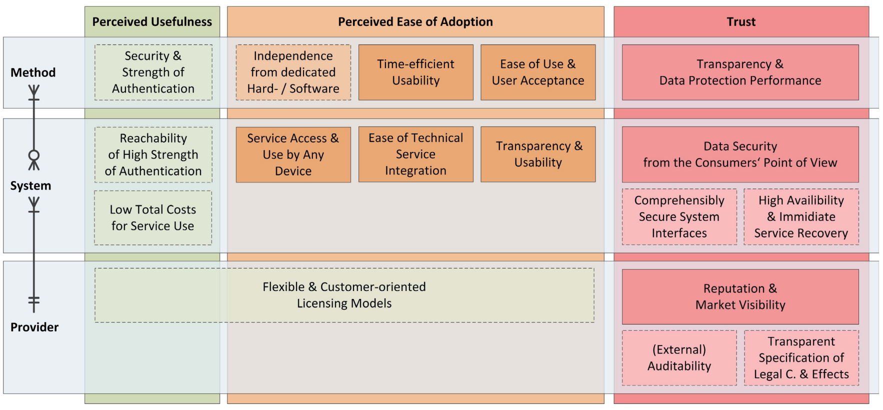

Perceived Usefulness
Perceived Ease of Adoption
Trust
Method
System
Provider
Security & Strength of Authentication
Independence from dedicated Hard- / Software
Time-efficient Usability
Ease of Use & User Acceptance
Transparency & Data Protection Performance
Reachability of High Strength of Authentication
Service Access & Use by Any Device
Ease of Technical Service Integration
Transparency & Usability
Data Security from the Consumers' Point of View
Low Total Costs for Service Use
Comprehensibly Secure System Interfaces
High Availibility & Immidiate Service Recovery
Flexible & Customer-oriented Licensing Models
Reputation & Market Visibility
(External) Auditability
Transparent Specification of Legal C. & Effects

Fig. 4. Systematization of determined Success Factors

[6] [+] support, [-] falsification of a hypothesis.

Figure 4 systematizes the determined success factors. While boxes for *strong* and *moderate* success factors feature solid lines, weak items are labelled with a broken line. The figure points out that, according to expert judgement, factors regarding the ease of adoption and the reduction of involved risks are more important than items related to the perceived usefulness of AaaS. These should be regarded by service providers in order to provide attractive authentication products to the market. Here, particularly the importance of security-related factors is supported by related literature and current research [e.g., 11, 14].

All in all, due to the size, quality and composition of the expert panel, we assume reliable results of this *Delphi* study for the German-speaking area.

5 Conclusion

This paper systematically investigates the development, relevant application fields and success drivers of cloud-based services for multi-factor authentication. For this purpose, a 3-rounded *Delphi* survey was conducted with 24 experts of the German-speaking area. The results indicate the significantly increasing importance of such services for both organizational and user-centric applications. Certain application fields were identified to be less or not relevant from a practical point of view. Moreover, seven success factors regarding applied authentication methods, the cloud service design and provider attributes have been identified. Authentication service providers might use these results to effectively direct development, certification or marketing programs. Future research should focus on security controls of such services and on system and interface design.

References

[1] Allen, J., Gabbard, D., May, C.: Outsourcing Managed Security Services. Security improvement module. Carnegie Mellon University, Software Engineering Institute (2003)

[2] amazon web services: AWS Multi-Factor Authentication (2012), `http://aws.amazon.com/de/mfa/`

[3] Braz, C., Robert, J.M.: Security and usability: the case of the user authentication methods. In: Proceedings of the 18th International Conferenceof the Association Francophone d'Interaction Homme-Machine, IHM 2006, pp. 199–203. ACM, New York (2006)

[4] Clarke, N.L.: Transparent User Authentication - Biometrics, RFID and Behavioural Profiling. Springer (2011)

[5] Cowan, N., Morey, C.C., Chen, Z., Gilchrist, A.L., Saults, J.S.: Theory and measurement of working memory capacity limits, pp. 49–104. AP (2008)

[6] Cranor, L., Garfinkel, S.: Security and Usability. O'Reilly Media, Inc. (2005)

[7] Forrester Research: Authentication-As–A-Service: A commissioned study conducted by Forrester Consulting on behalf of VeriSign (2009), `http://www.verisign.co.uk/static/auth-as-a-service.pdf`

[8] Gomi, H.: An authentication trust metric for federated identity management systems. In: Cuellar, J., Lopez, J., Barthe, G., Pretschner, A. (eds.) STM 2010. LNCS, vol. 6710, pp. 116–131. Springer, Heidelberg (2011)

[9] Gupta, A., Zhdanov, D.: Growth and sustainability of managed security services networks: an economic perspective. Economics of Information Security, 1–35 (2007)

[10] Häder, M.: Delphi-Befragungen: Ein Arbeitsbuch. VS Verlag für SW (2009)

[11] Höfer, C., Karagiannis, G.: Cloud computing services: taxonomy and comparison. Journal of Internet Services and Applications, 1–14 (2011)

[12] Jain, A., Flynn, P., Ross, A. (eds.): Handbook of Biometrics. Springer, New York (2007)

[13] Linstone, H., Turoff, L., Turoff, M.: The Delphi Method: Techniques and Applications (2002)

[14] Mather, T., Kumaraswamy, S., Latif, S.: Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance. O'Reilly Media, Inc. (2009)

[15] Mell, P., Grance, T.: The NIST Definition of Cloud Computing. National Institute of Standards and Technology 53(6), 50 (2009), `http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc`

[16] Okoli, C., Pawlowski, S.D.: The Delphi method as a research tool: an example, design considerations and applications. Information & Management 42(1), 15–29 (2004)

[17] Senk, C., Holzapfel, A.: Market overview of security as a service systems. In: Pohlmann, N., Reimer, H., Schneider, W. (eds.) ISSE 2011 Securing Electronic Business Processes (2011)

[18] Smith, R.E.: Authentication: From passwords to public keys. Addison-Wesley, Boston (2002)

Author Index