Journal of Network and Systems Management (2020) 28:197–246
https://doi.org/10.1007/s10922-019-09504-0

Resource Management in a Containerized Cloud: Status and Challenges

Pieter-Jan Maenhaut[1] · Bruno Volckaert[1] · Veerle Ongenae[1] · Filip De Turck[1]

Received: 11 February 2019 / Revised: 9 September 2019 / Accepted: 11 November 2019 /
Published online: 22 November 2019

Abstract

Cloud computing heavily relies on virtualization, as with cloud computing virtual resources are typically leased to the consumer, for example as virtual machines. Efficient management of these virtual resources is of great importance, as it has a direct impact on both the scalability and the operational costs of the cloud environment. Recently, containers are gaining popularity as virtualization technology, due to the minimal overhead compared to traditional virtual machines and the offered portability. Traditional resource management strategies however are typically designed for the allocation and migration of virtual machines, so the question arises how these strategies can be adapted for the management of a containerized cloud. Apart from this, the cloud is also no longer limited to the centrally hosted data center infrastructure. New deployment models have gained maturity, such as fog and mobile edge computing, bringing the cloud closer to the end user. These models could also benefit from container technology, as the newly introduced devices often have limited hardware resources. In this survey, we provide an overview of the current state of the art regarding resource management within the broad sense of cloud computing, complementary to existing surveys in literature. We investigate how research is adapting to the recent evolutions within the cloud, being the adoption of container technology and the introduction of the fog computing conceptual model. Furthermore, we identify several challenges and possible opportunities for future research.

Keywords Resource management · Containers · Virtual machines · Cloud computing · Fog computing · Edge computing · Survey

✉ Pieter-Jan Maenhaut
pieterjan.maenhaut@ugent.be

[1] Ghent University-imec, iGent, Technologiepark-Zwijnaarde 126, 9052 Gent, Belgium

1 Introduction

Over recent years, cloud computing has become an important aspect of our daily life, and many novel applications have been developed on top of the cloud. These applications are often available as online web services, which can be accessed through a custom app or directly through the web browser. The term cloud computing has a broad meaning: it not only refers to the online applications and services hosted in the cloud, but also to the underlying frameworks and technologies that enable them.

One of the key enablers of cloud computing is the so-called elasticity, which allows cloud applications to dynamically adjust the amount of provisioned resources based on the current and/or expected future demand. Given the increasing popularity and amount of cloud applications, efficient resource management is of great importance, as it can not only result in higher scalability of the cloud environment, but also in lower operational costs. Efficient resource management can be beneficial for multiple actors. For the cloud infrastructure provider, it aids to minimize the power consumption, as unprovisioned hardware can be put in standby or even turned off. This also helps to reduce the energy footprint of the data center, which is one of the main goals of green cloud computing. For the consumer, efficient resource management helps to achieve high scalability and high availability while minimizing the rental costs. And when multiple consumers share the same physical hardware, the provider can offer its instances at a lower price.

As a result, resource management within cloud environments has been a major research topic since the introduction of cloud computing. A typical research objective is to minimize the amount of provisioned computational resources, in order to lower the operational costs, without violating the objectives described in so-called Service Level Agreements (SLAs). An example of this is Virtual Machine (VM) packing, which aims to consolidate virtual servers onto a minimal number of physical machines. Multiple resource allocation strategies have been developed by both academics and industry, often resulting in open source and/or commercial products. A popular example is Swift [1], a highly scalable cloud storage system, which is integrated into the OpenStack cloud stack, and OpenStack [2] itself, an open-source framework for building a private cloud environment, which has multiple resource management functions built in.

A recent trend within cloud computing is the uprise of new types of clouds, such as mobile edge and fog computing [3, 4]. The cloud is no longer limited to the centrally hosted data center, accessible from a laptop or desktop computer with a broadband internet connection, but lightweight devices such as mobile phones and Internet of Things (IoT) devices can also benefit from the near infinite amount of resources offered by the cloud. These devices can offload computational intensive tasks to a more powerful cloud environment, and by installing dedicated hardware at the edge of the network, close to the end user devices, the latency can be reduced, as well as the consumed network bandwidth towards the public cloud.

When it comes to virtualization, a key enabler for cloud computing, container technology has recently gained popularity, thanks to the minimal overhead compared to traditional VMs, and the great portability it offers [5, 6]. These benefits could facilitate the migration of containers between different cloud environments, and the deployment of services at the edge of the cloud, for example onto less powerful ARM hardware located within IoT devices. Furthermore, the offered portability provides an interesting opportunity for offloading within a fog-cloud environment, allowing developers to reconfigure which services are running locally or in the cloud, without paying the heavy penalty of traditional VM migrations.

In this survey, we investigate how recent research related to cloud resource management is adapting to support these new technologies. This survey is complementary to existing surveys in literature, as most previously published surveys only handle resource management within traditional cloud environments [5–15] or only consider virtual machines as virtualization technology [4, 9–12, 14, 15]. Furthermore, as illustrated in Sect. 3.1, a majority of surveys focus on a specific aspect of resource management such as resource scheduling or dynamic spot pricing. This survey covers the broad range of resource management, and is not limited to a single cloud type or virtualization technology. The remainder of this article is structured as follows. In the next section, we introduce all relevant concepts and technologies related to resource management in containerized cloud environments. In Sect. 3 we provide an overview of recent research related to resource management, and identify several challenges and opportunities in Sect. 4. We finish this article by presenting our conclusions in Sect. 5.

2 Related Concepts and Technologies

In this section, we provide an overview of all relevant concepts and technologies. First, we start with a brief summary of cloud computing, and introduce the main concepts behind edge/fog computing. Next, we elaborate on virtualization, as this is one of the key enablers for cloud computing, and introduce containerization (OS-level virtualization) as an alternative for VMs. Finally, we describe all main functions related to cloud resource management.

2.1 Cloud, Edge and Fog Computing

2.1.1 Traditional Cloud Computing

With cloud computing, different deployment models can be distinguished. The National Institute of Standards and Technology (NIST) defined four main deployment models [16]:

- In a **Private Cloud**, the cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers.
- A **Community Cloud** is similar to a private cloud, but the infrastructure is provisioned for exclusive use by a specific community of consumers.

- A **Public Cloud** is provisioned for open use by the general public, and is usually fully accessible over the public internet.
- A **Hybrid Cloud** is a composition of two or more distinct cloud infrastructures.

Applications can either be deployed within a single cloud, or using multiple clouds. To avoid vendor lock-in, one can for example choose to deploy its application using different public cloud platforms offered by different providers. Another example is a hybrid cloud which consists of a private cloud and a public cloud. In this model, the main application is typically deployed on the private cloud, and the public cloud is used for executing computational intensive tasks, or to support the private cloud when the demand for computing capacity spikes. The latter case is often referred to as **Cloud Bursting**.

Within the context of public cloud computing, three main service models can be distinguished, as defined by the NIST [16]:

- **Infrastructure as a Service (IaaS)**: in this model, the provider offers (typically virtual) computational resources to the consumer, for example as VMs. The consumer does not manage or control the underlying cloud infrastructure, but does have control over operating systems, storage, deployed applications and possibly limited control over the network (e.g. for defining firewall rules).
- **Platform as a Service (PaaS)**: in this model, the provider offers a set of languages, libraries, services, and tools to the consumer for deploying its applications. In contrast to IaaS, the consumer typically has no control over the operating system and storage, but can control the deployed applications and applicable configuration settings for the hosting environment.
- **Software as a Service (SaaS)**: in this model, applications running on a cloud infrastructure are offered to the consumer. These applications are typically deployed on top of an IaaS or PaaS environment. The consumer has no control over the underlying infrastructure and software, except for limited application specific customization.

In the above definitions, a provider offers services to a consumer. The term provider however has a broad sense, and Armbrust et al. defined three main actors within Cloud Computing [17]:

- The **Cloud Provider** or infrastructure provider manages a physical data center, and offers (virtualized) resources to the cloud users, either as IaaS or PaaS instances.
- The **Cloud User** rents virtual resources (e.g. a VM) from the cloud provider to deploy its cloud applications, which he provides (typically as SaaS) to the end users.
- The **End User** uses the SaaS applications provided by the cloud user. The end user generates workloads that are processed using cloud resources.

The end user typically does not play a direct role in resource management, but the behavior of the end users can influence, and be influenced by the resource management decisions of the cloud user and the cloud provider [9]. Cloud users manage cloud resources from the perspective of the deployed applications, whereas for cloud providers the main focus is the management of the underlying physical resources.

2.1.2 Fog and Edge Computing

Access to the cloud is no longer limited to traditional devices such as servers, desktops and laptops. With mobile edge computing (also referred to as mobile cloud computing) for example, mobile devices collaborate with a cloud environment. As these mobile devices are usually connected using a less reliable connection with limited bandwidth, and are often battery powered, some tasks will be executed directly on the device, whereas other tasks will be transferred to the cloud. Executing tasks on the device can reduce the network congestion and lower the latency, but will increase the energy consumption of the device. Offloading tasks to the cloud on the contrary can decrease the energy consumption, and can also decrease the execution time for computational intensive tasks.

Mobile edge computing is in fact a special case of **Edge Computing**, which in general aims to provide context aware storage and distributed computing at the edge of the network [18, 19]. Another term that is often used is **Fog Computing**, originally coined by Cisco to extend the cloud computing paradigm to the edge of the network [20]. As of today, there is no clear distinction between both terms, and they are often used in literature as interchangeable terms. However, in March 2018, the NIST published a conceptual model for fog computing, which adopts many of the terms introduced by Cisco [21]. Therefore, in the remainder of this survey, we will mainly use the term fog computing.

Fog computing can be implemented in different ways, depending on the used architecture, the function and location of the intermediate fog nodes, the offered services, and the target applications. In general, a distinction can be made between three main implementations [22]:

- In a general **Fog Computing** implementation, dedicated fog nodes (e.g. gateways, devices, computers or micro data centers) are deployed at any point of the architecture between the edge devices and the cloud. These heterogeneous nodes can for example gather data from the edge devices and perform some (pre-)processing of the gathered data. Doing so can help to reduce the network congestion towards the central cloud, and can also help to reduce the response time. The heterogeneity of the fog nodes is often hidden from the end devices, by exposing a uniform fog abstraction layer which offers a set of functions for resource allocation, monitoring, security and device management together with storage and computing services.
- With **Mobile Edge Computing**, computational and storage capacities are available at the edge of the network, in the radio access network, mainly to reduce latency and to improve context awareness. Mobile edge computing aims to reduce the network congestion and is often implemented at the cellular base stations.

- With **Cloudlet Computing**, trusted clusters of computers are connected to the Internet, offering resources to nearby mobile devices. A cloudlet is a small-scale cloud datacenter, located at the edge of the network, and is mainly used to support resource-intensive and interactive mobile applications with low latency.

An example topology for fog computing is illustrated in Fig. 1. Fog computing typically aims to reduce the latency and the load on the cloud, and is often used in the context of IoT, in which large amounts of data are collected for analysis and processing [19, 23–26].

2.2 Virtualization

2.2.1 VMs and Containers

Cloud computing is mainly built on top of virtualization, as cloud users typically rent virtual resources from the cloud providers. A typical form of virtualization is the use of VMs, in which multiple VMs are emulated on top of a so-called hypervisor. This hypervisor creates and runs the virtual machines, and runs on a host machine (typically a physical server), whereas the VMs are called guest machines. There are two main types of hypervisors:

- A type-1 or **native/bare-metal hypervisor** runs directly on the host's hardware. A popular example of a type-1 hypervisor is VMWare ESX/ESXi.

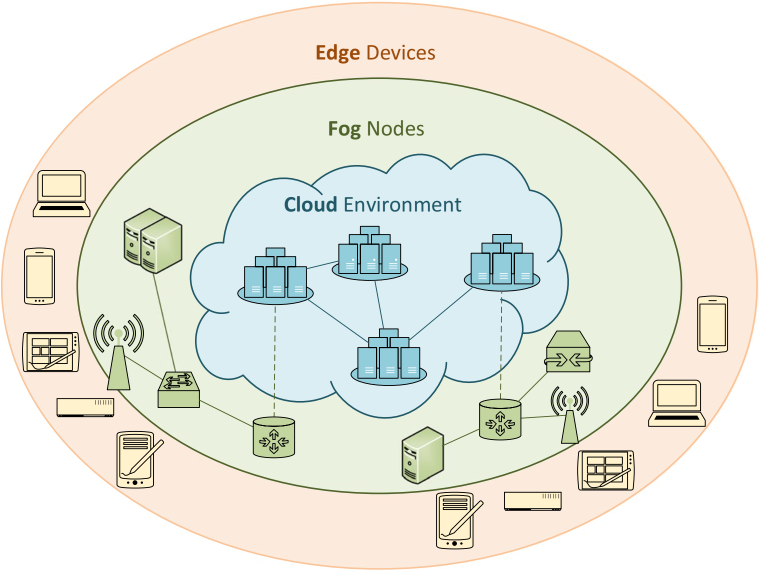

Fig. 1 Example topology for fog computing. Fog nodes bring the cloud closer to the end user, and the edge devices can offload computational intensive tasks to the central cloud

- A Type-2 or **hosted hypervisor** runs on top of a conventional Operating System (OS), possibly together with other computer programs. Popular examples of hosted hypervisors include VMWare Player and VirtualBox.

In general, type-1 hypervisors are more efficient than type-2 hypervisors, and most cloud environments are built using type-1 hypervisors.

A VM that is emulated on top of a hypervisor runs the full software stack, meaning that an OS is deployed on the virtual disk of the VM, and the required software is installed on top. When deploying a VM, the user can either start from scratch and create a new virtual machine with an empty virtual disk, install the preferred OS and all required binaries and libraries, or a pre-configured template can be used for deploying a new VM which already contains the operating system and a typical software stack (e.g. a web server). In the latter case, the cloud user only needs to customize the packages, and deploy its application on top. Because the full OS is installed on the virtual disk of the VM, this virtual disk is easily a few gigabytes in size.

Recently, container technologies have emerged as a more lightweight alternative for VMs [27–30]. The major difference with VMs is that a container typically has no operating system installed, but instead all containers deployed on a single machine are running directly on the operating system kernel (OS-level virtualization). As a result, containers are much smaller in size. A typical container image is a few hundreds megabytes, whereas a similar virtual disk for a VM with the same applications installed will typically be a few gigabytes. To launch a new container, the user can either start from a base image (e.g. an Ubuntu-flavored base image or an official NodeJS base image) and install and configure all required software packages, or he can create a new container based on a pre-configured image that is pulled from a central repository, with most of the required software already installed and configured.

OS-level virtualization (or containerization) has existed for some time, with LXC [31] being one of the first popular container engines. LXC was initially released in 2008, but in 2013 Docker [32] was released as a successor for LXC, and quickly became one of the most popular container engines. Initial releases of Docker were still using LXC as default execution environment, but in later releases Docker replaced LXC with its own library. To facilitate the deployment, Docker containers can be published to Docker Hub [33], a publicly available, centrally hosted repository for storing fully configured container images, or organizations can configure their own private Docker image repository. Docker however only offers tools for deploying and managing containers on top of a single physical machine.

For the management and deployment of containerized applications over a cluster of Docker servers, a container orchestration system such as Kubernetes [34] is required. Docker initially offered its own orchestration tools, called Docker Swarm mode, providing limited functionality for managing container clusters [35]. In 2017, the team behind Docker however announced native Kubernetes support, and recommended Kubernetes as orchestration tool for enterprise environments [36].

As containers are lightweight, they are often used for deploying applications that are designed using a Service-Oriented Architecture (SOA). With SOA, an application is decomposed into several collaborating services, and every service can be deployed into a separate container. This allows for fine-grained scalability, as each service can be scaled up or down individually, instead of scaling the whole application as a whole. In multi-cloud environments, the use of lightweight containers also offers multiple opportunities for achieving high scalability and cost-efficient deployments, thanks to the offered portability [37].

2.2.2 Live Migration

As the demand for resources changes over time, it might be required to migrate some VMs or containers to a different physical host in order to prevent over-utilization of the available physical hardware resources. VMs are relatively large in size, making migration an expensive operation, especially when moving the VM to a different physical location, as the whole virtual disk needs to transferred [38]. When migrating a VM, the machine can first be turned off, which facilitates the migration process as there are almost no risks such as losing state or consistency, but there will be a noticeable downtime. Most hypervisors however also support the migration of running virtual machines between different physical machines, without disconnecting the client or application, referred to as live migration. Live migration will also include some downtime of the VM, but when this is not noticeable by the end users, the migration is called a seamless live migration.

Despite the aforementioned advantages of containers, live migration of containers still remains an important research challenge [39]. Containers are a hierarchy of processes, and existing methods for process migration are often applied, for example using the Checkpoint-restore in Userspace (CRIU) tool [40]. However, such methods could cause significant delays [39, 41], resulting in a relatively high downtime, e.g. when the application running inside a container modifies large amounts of memory faster than the container can be transferred over the network to a remote host. The feasibility of live container migrations in this scenario will therefore be mainly dependent on the network bandwidth between the source and destination location and the characteristics of the running container(s).

Furthermore, the migration of containers could introduce some additional problems, as they not only share the underlying OS but also some libraries [38]. During migration, the destination host must support these libraries, together with the libraries required by other containers. The selection of a feasible destination host is therefore an important issue. In contrast, a VM can be migrated to any destination host that can accommodate the VM and is managed by the same type of hypervisor.

2.2.3 Advantages and Risks

The main benefit of containers is that they introduce less virtualization overhead than VMs, because there is no additional layer of virtualization. Instead, they are executed directly on the kernel of the host OS. Containers are therefore considered more efficient and allow for greater scalability. However, the lack of a virtualization layer introduces new security risks due to the lower level of isolation; containers were not designed as a security mechanism to isolate between untrusted and potentially malicious containers [42, 43]. Because containers deployed onto the same host share a common OS, they allow for attacks on shared resources such as the file system, network and the kernel. Kernel bugs can be exploited through a large attack surface, or an attack could target the shared host resources to enable misconfiguration, side channels or data leakage [44]. As a result, container security is considered an obstacle for the wide adoption of containerization technologies. To increase the security of containers, some protection mechanisms can be applied such as security hardening mechanisms and host based intrusion detection systems [42]. However, adding such mechanisms will introduce an additional overhead which could negatively impact the scalability and performance of container environments [43].

VMs and containers thus both have their advantages and disadvantages, and a combination of both virtualization technologies is also possible, for example when deploying a container engine on top of VMs [29, 45]. In this scenario, the application is deployed inside a container, and the container runtime is running on top of the guest OS of the VM. Such hybrid model could potentially combine the advantages of both technologies.

To summarize, Fig. 2 provides an overview of the typical models for deployment of an application or service within a virtualized environment. When deploying in a public cloud environment, the question arises who should be responsible for which environment, especially for the hybrid model. A cloud provider typically manages resources at the infrastructure level, for example by offering VMs to the cloud users. A cloud user could thus rent several VMs and deploy a container system on top, or the cloud provider itself could offer a containerized environment that is deployed on top of virtual machines.

2.3 Resource Management

Resource management is a broad term, which refers to all required functionalities related to the allocation, provisioning and pricing of (virtual) resources. For

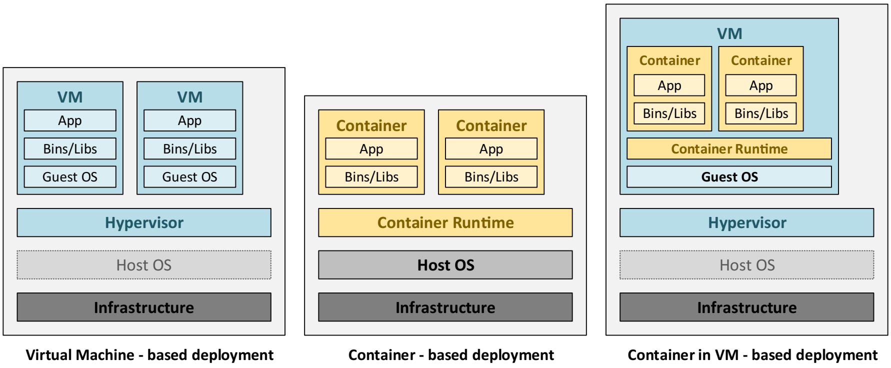

Fig. 2 Comparison between the different models for deployment within a virtualized environment. The application or service can be either deployed inside a VM, a container, or a container hosted in a VM

the deployment of cloud applications, the minimal required amount of resources needs to be determined, and in an elastic cloud environment the allocated amount of resources can change dynamically based on the current demand. Furthermore, by monitoring and profiling the applications or the resources, an estimate can be made regarding the future demand. In a public cloud environment, the cloud provider needs to determine the price billed to the cloud users based on the actual resource usage, and the cloud user can charge the end users for using the SaaS applications.

2.3.1 Management Objectives

With public cloud computing, cloud providers need to satisfy the SLAs agreed upon with the cloud users regarding the provisioning of virtual infrastructure. Such an SLA can consist of multiple constraints which must always be satisfied, and Service-Level Objectives (SLOs) which should be satisfied. A typical management objective is a specified monthly uptime percentage for the virtual instances, or a maximum allowed response time for the cloud environment. The provider can choose to offer its infrastructure to all cloud users using a single SLA, or can pursue service differentiation by offering different service levels to the customers. The provider could also choose to apply different objectives during different operational conditions, for example by guaranteeing different objectives during low load or overload.

The cloud user can also have an SLA with the end-users, consisting of objectives regarding the offered services (typically as SaaS). To comply with these objectives, the cloud user may seek to exploit the elasticity property of the cloud environment. The cloud user could for example over-provision resources in order to guarantee the objectives, or could try to minimize the operational costs but with the risk of violating the SLA with the end users.

2.3.2 Resource Elasticity

In an optimal scenario, every cloud application would be deployed in a location close to the end users in order to minimize latency, on hardware that is powerful enough to guarantee compliance with the selected SLAs, and on a dedicated server to maximize performance isolation. This scenario however would lead to high operational costs and energy consumption, and a waste of resources as most of the time the provisioned server instances would be in an idle state. Resource allocation strategies aim to solve this issue, by packing multiple applications belonging to different customers onto the same physical hardware, while guaranteeing performance and data isolation and compliance with SLA requirements. Resource management consists of multiple tasks, with the main tasks being the allocation, provisioning and scheduling of (virtual) resources.

- **Resource Allocation** refers to the allocation (reservation) of a pool of resources (e.g. computational resources, network bandwidth and storage) for a given consumer.
- **Resource Provisioning** on the other hand is the effective provisioning of (a part of) the allocated resources in order to execute a given task. A typical example

of resource provisioning is the deployment of a new virtual machine by the consumer, which uses a subset of the allocated CPU, network and storage resources.
- When executing a large batch of tasks in the cloud, **Resource Scheduling** aims to find a feasible execution order for these tasks, making optimal usage of the available resources while respecting the deadlines defined for each individual task.
- **Resource Orchestration** is a broad term, that includes both scheduling, management and provisioning of additional resources. Orchestrators typically manage complex cross-domain processes, and aim to meet the defined objectives, for example meeting the application performance goals while minimizing costs and maximizing performance.

Resource allocation, provisioning, scheduling and orchestration are closely related, and are the main building blocks for application elasticity within a cloud environment. When allocating resources, a further distinction can be made between static and dynamic allocation.

- With a **Static Resource Allocation** strategy, the required amount of resources is determined during deployment, and the allocation of resources does not change during the lifetime of the deployed applications. Static resource allocation however can lead to under- or over-provisioning, when the amount of allocated resources is not in line with the current demand.
- With **Dynamic Resource Allocation**, the amount of allocated resources can change during execution, in order to meet the current demand. Dynamic resource allocation can lead to a higher utilization of the physical resources, and allows for server consolidation in order to reduce the operating costs.

Dynamic resource allocation is often seen as the most efficient means to allocate hardware resources in a data center [46]. However, dynamic resource allocation typically involves migration of running applications, which leads to an overhead and possible service disruptions.

2.3.3 Resource Profiling

When allocating resources, a distinction can be made between reactivity and proactivity:

- With a **Reactive** control mechanism, the amount of allocated resources is adjusted over time in response to a detected change in demand.
- With a **Proactive** control mechanism, the amount of allocated resources is adjusted based on a predicted change in demand.

For proactive control mechanisms, a prediction of the demand is often made using historical measurements. This is typically done using **Demand Profiling**, and can happen either at the application level, when predicting the demand for individual applications, or at the infrastructure (data center) level, when predicting the global demand within the cloud environment. Apart from estimating the demand, an estimation can also be made regarding the state of the physical and virtualized resources, often referred to as **Resource Utilization Estimation**. An estimation can be made for the different types of resources, such as compute, network, storage and power resources, and these estimations serve as input for both the monitoring and scheduling processes.

By **Monitoring** the actual resource utilization, the provider can detect if the current allocation scheme fits the current demand. In an elastic cloud environment, additional resources can be provisioned on the fly if there is an over-utilization of the provisioned resources (under-provisioning), and when more resources are allocated than required (over-provisioning), a certain amount of resources can be deallocated to decrease the operational costs. Monitoring processes can also be used to determine failure of certain components. Furthermore, monitoring information can provide useful input for both demand profiling and resource utilization estimation.

2.3.4 Resource Pricing

Especially with public cloud computing, the cloud user or end user will be charged based on its usage of the cloud resources or cloud services. In this context, a distinction can be made between application pricing and infrastructure pricing [9].

- With **Application Pricing**, the cloud user determines the price for the services (typically offered as SaaS applications) provided to the end users.
- With **(Virtual) Infrastructure Pricing**, the cloud provider determines the price charged for the virtual resources rented to the cloud users.

For application pricing, the cloud user could either provide its application for free, at a fixed price (e.g. a monthly incurring bill, with the price based on the number of active users) or he could charge the cloud user based on the actual usage (e.g. the total amount of bandwidth or data storage used by the consumer).

For infrastructure pricing, cloud providers traditionally use a **Static Pricing** scheme to cover the infrastructure and operational costs of the data center, especially within the context of public cloud computing. With static pricing, a price point is established and maintained for an extended period of time. The cloud provider can choose to offer its services using flat rate pricing, usage based pricing or a tiered pricing strategy. With flat rate pricing, cloud users are charged a fixed price for a package which could consist of the required services and a given number of users. As long as the package doesn't change, the price remains constant and is therefore predictable. Tiered pricing is similar, but in this case the provider offers multiple packages, with different combinations of features offered at different price points. Because cloud users can select the package that best fits their needs, tiered pricing allows for a broader market. With usage based pricing, often referred to as the 'Pay As You Go' model, cloud users are charged based on the actual resource usage. A combination of different pricing models is also possible. The cloud provider can for example charge a fixed price based on the number of instantiated VMs, together with a variable price based on the amount of consumed network bandwidth and/or additional storage. Other pricing models also exist, but these are often derived from one of the three previously mentioned models. For example, price per request is commonly found within cloud computing pricing schemes, which is a form of usage based pricing. It is also worth noting that flat rate pricing, usage based pricing and tiered pricing strategies are also often applied at SaaS level (application pricing).

Recently, **Dynamic Pricing** schemes are gaining popularity as an alternative to static pricing, mainly to increase the utilization of the data center [10, 47–51]. With dynamic pricing, the price of a product or service can change over time. The cloud provider can for example lease its resources at a lower price when the demand is low, and increase the prices as the demand increases. Another example of dynamic pricing is spot pricing, in which the cloud provider offers dynamically priced resources at a lower price, but with less guarantee of availability [10]. Dynamic pricing can also be based on an auction-based pricing model, in which multiple cloud users bid for a bundle of virtual cloud resources [48, 49]. The cloud provider will then select a set of cloud users, the winners, and needs to determine a feasible allocation over its physical hardware.

3 Cloud Resource Management: State of the Art

This section provides an overview of recent research (published between 2015 and 2018) focusing on resource management within cloud environments. We selected this time period as this chapter extends the survey previously published by Jennings and Stadler [9], which already provides an extensive overview of research related to resource management published before 2015. We reviewed over 150 research papers from five main publishers, namely ACM, Elsevier, IEEE, Springer and Wiley. A majority of the reviewed articles were published in either ACM Transactions on Internet Technology (TOIT) [52], IEEE Transactions on Cloud Computing (TCC) [53], IEEE Transactions on Parallel and Distributed Systems (TPDS) [54], IEEE Transactions on Network and Service Management (TNSM) [55], Springer Journal of Network and Systems Management (JNSM) [56] or Wiley Journal of Software: Practice and Experience (SPE) [57].

In the remainder of this section, a brief summary of previous surveys focusing on resource management is first provided. We then categorize the research items within three main areas, as illustrated in Fig. 3. For each category, an overview of all relevant research items is provided, and in this overview, we added attributes to denote the used cloud type (traditional or fog), the scope (single cloud or multi-cloud) and the virtual allocation entity (VM or container). Furthermore, a summary of the most relevant research is provided for each resource management functional element, and we especially investigate the impact of containers and new cloud deployment models.

As a reference, Table 1 provides a mapping from all research items (excluding surveys) to the covered resource management functional elements of Fig. 3. As can be seen from this table, some publications can be attributed to multiple categories and/or functional elements. For these items, we selected the most relevant category

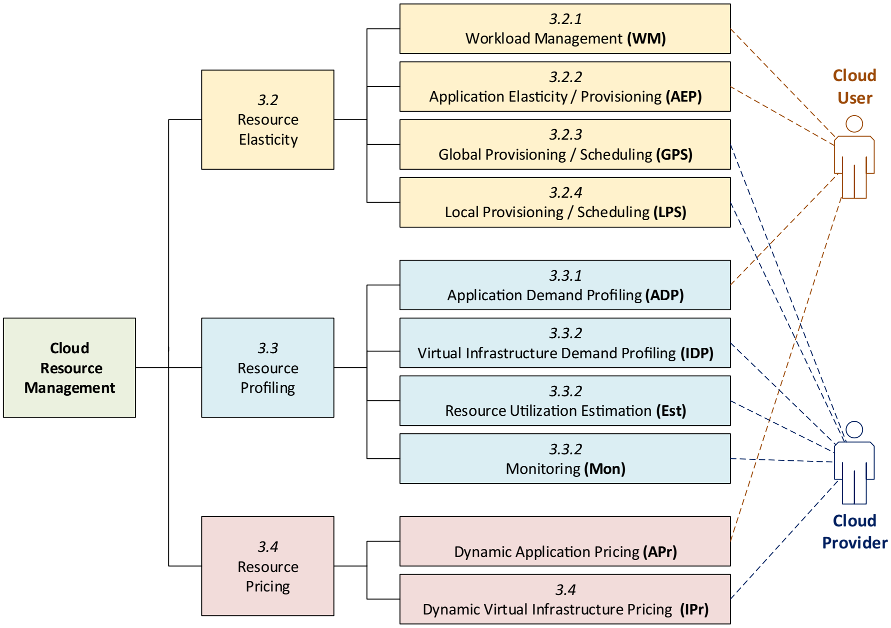
Cloud Resource Management
3.2 Resource Elasticity
3.2.1 Workload Management (WM)
3.2.2 Application Elasticity / Provisioning (AEP)
3.2.3 Global Provisioning / Scheduling (GPS)
3.2.4 Local Provisioning / Scheduling (LPS)
3.3 Resource Profiling
3.3.1 Application Demand Profiling (ADP)
3.3.2 Virtual Infrastructure Demand Profiling (IDP)
3.3.2 Resource Utilization Estimation (Est)
3.3.2 Monitoring (Mon)
3.4 Resource Pricing
Dynamic Application Pricing (APr)
3.4 Dynamic Virtual Infrastructure Pricing (IPr)
Cloud User
Cloud Provider

Fig. 3 Cloud resource management taxonomy used in this article, based on the conceptual framework introduced by Jennings and Stadler [9]. For each functional element, the corresponding subsection is denoted in the figure

and/or element, and in the remainder of this section these items are included in the corresponding subsection.

3.1 Previous Surveys

Table 2 provides an overview of previous surveys related to resource management within cloud environments. In 2015, Jennings and Stadler published an extensive overview of resource management within the public cloud [9]. In their survey, the authors introduced a conceptual framework for cloud resource management consisting of multiple functional elements, as illustrated in Fig. 4. In this figure, we added a mapping from the different resource management functional elements to the categories used in this article, namely Elasticity, Profiling and Pricing. Furthermore, Jennings and Stadler characterized cloud provisioning schemes based on the placement approach (static, dynamic, network aware and/or energy aware) and the control architecture (centralized, hierarchical or distributed). The authors did briefly mention mobile edge computing as one of the challenges, but the main focus of their survey is the management of VMs in traditional cloud environments.

Yousafzai et al. extended the research of Jennings and Stadler by introducing a taxonomy for categorizing cloud resource allocation schemes [14]. The introduced taxonomy is based on multiple attributes, being the optimization objective, the design approach, the target resource allocation type, the applied optimization

Table 1 Mapping from all research items (excluding surveys) to the resource management functional elements of Fig. 3

Publication	Year	Elasticity				Profiling				Pricing	
		WM	AEP	GPS	LPS	ADP	IDP	Est	Mon	APr	IPr
Aazam and Huh [58]	2015			✓			✓	✓			✓
AbdelBaky and Unuvar [59]	2015			✓							
Amannejad et al. [60]	2015		✓		✓						
Chiang et al. [61]	2015			✓							
Dabbagh et al. [62]	2015			✓			✓	✓	✓		
Dhakate and Godbole [63]	2015								✓		
Huang et al. [64]	2015			✓							✓
Jin et al. [65]	2015										✓
Katsalis et al. [66]	2015				✓						
Kumbhare et al. [67]	2015	✓	✓								
Lee et al. [68]	2015			✓							✓
Li and Kanso [69]	2015		✓	✓	✓				✓		
Liu et al. [70]	2015			✓					✓		
Mashayekhy et al. [48]	2015			✓							✓
Moens et al. [71]	2015		✓								
Mukherjee et al. [72]	2015				✓						
Petri et al. [73]	2015			✓							✓
Sharma et al. [74]	2015										✓
Stankovski et al. [75]	2015			✓					✓		
Wang et al. [76]	2015			✓							✓
Wuhib et al. [77]	2015			✓			✓	✓	✓		
Zhang et al. [78]	2015		✓								
Aazam et al. [79]	2016			✓							✓
Ayoubi et al. [80]	2016		✓	✓							
Choi et al. [81]	2016			✓					✓		
D.C. Rodrigues et al. [82]	2016								✓		
Dai et al. [83]	2016			✓							
Elgazzar et al. [84]	2016		✓								
Espling et al. [85]	2016			✓							
Goudarzi et al. [86]	2016			✓			✓	✓			
Huang and Tsang [87]	2016			✓							
Kang et al. [88]	2016	✓									
Khatua et al. [89]	2016		✓								
Mashayekhy et al. [49]	2016			✓							✓
Mishra and Bellur [90]	2016			✓							
Nakagawa and Oikawa [91]	2016				✓			✓	✓		
Pantazoglou et al. [92]	2016			✓							
Righi et al. [93]	2016		✓								
Salah et al. [94]	2016		✓								
Sharma et al. [29]	2016				✓						

Table 1 (continued)

Publication	Year	Elasticity				Profiling				Pricing	
		WM	AEP	GPS	LPS	ADP	IDP	Est	Mon	APr	IPr
Wajid et al. [95]	2016			✓					✓		
Wan et al. [96]	2016			✓							✓
Wanis et al. [97]	2016									✓	✓
Wolke et al. [46]	2016			✓				✓	✓		
Wu et al. [98]	2016			✓							
Xu et al. [99]	2016	✓	✓								
Zhou et al. [100]	2016								✓		
Awada and Barker [101]	2017			✓							
Awada and Barker [102]	2017			✓							
Babaioff et al. [103]	2017			✓			✓				✓
Chard et al. [104]	2017		✓			✓					
Chi et al. [47]	2017			✓							✓
Dalmazo et al. [105]	2017							✓			
Hai and Nguyen [106]	2017			✓						✓	✓
Hoque et al. [107]	2017			✓							
Jin et al. [108]	2017			✓							
Khasnabish et al. [109]	2017			✓					✓		
Li et al. [110]	2017				✓						
Li et al. [111]	2017			✓							
Lloyd et al. [112]	2017					✓					
Maenhaut et al. [113]	2017		✓	✓							
Mebrek et al. [114]	2017		✓								
Mechtri et al. [115]	2017			✓							
Merzoug et al. [116]	2017			✓							
Mireslami et al. [117]	2017		✓								
Nardelli et al. [118]	2017			✓							
Nitu et al. [119]	2017			✓							
Paya and Marinescu [120]	2017		✓								
Rankothge et al. [121]	2017		✓	✓							
Tang et al. [122]	2017			✓							✓
Xu et al. [123]	2017	✓	✓								
Yang et al. [124]	2017			✓							
Yi et al. [125]	2017			✓							✓
Yu and Pan [126]	2017			✓							
Zhang et al. [127]	2017			✓	✓				✓		
Alam et al. [128]	2018		✓								
Aral and Ovatman [129]	2018		✓	✓							
Atrey et al. [130]	2018	✓	✓			✓					
Barkat et al. [131]	2018			✓							
Balos et al. [132]	2018		✓			✓	✓	✓			
Barrameda and Samaan [133]	2018	✓	✓			✓					

Table 1 (continued)

Publication	Year	Elasticity				Profiling				Pricing	
		WM	AEP	GPS	LPS	ADP	IDP	Est	Mon	APr	IPr
Borjigin et al. [134]	2018			✓							✓
Bouet and Conan [135]	2018	✓	✓								
Cheng et al. [136]	2018	✓	✓	✓							
Diaz-Montes et al. [137]	2018	✓	✓	✓							
Gill et al. [138]	2018	✓									
Govindaraj and Artemenko [41]	2018			✓	✓						
Guo and Shenoy [139]	2018	✓	✓			✓					
Guo et al. [140]	2018	✓	✓								
Guo et al. [141]	2018			✓							
Hauser and Wesner [142]	2018			✓			✓	✓	✓		
Heidari and Buyya [143]	2018	✓									
Jia et al. [144]	2018			✓							
Jia et al. [145]	2018			✓			✓	✓			
Khabbaz and Assi [146]	2018	✓		✓							
Lahmann et al. [147]	2018				✓						
Lin et al. [148]	2018		✓	✓							
Mikavica et al. [50]	2018										✓
Nawrocki and Sniezynski [149]	2018	✓	✓								
Prakash et al. [45]	2018				✓			✓	✓		
Prats et al. [150]	2018	✓				✓	✓		✓		
Rahimi et al. [151]	2018	✓	✓			✓					
Sahni and Vidyarthi [152]	2018	✓	✓								
Santos et al. [24]	2018		✓	✓							
Scheuner and Leitner [153]	2018		✓			✓					
Simonis [154]	2018	✓	✓								
Sofia and GaneshKumar [155]	2018	✓	✓	✓							
Stoyanov and Kollingbaum [39]	2018			✓	✓						
Takahashi et al. [156]	2018	✓	✓								
Tesfatsion et al. [30]	2018				✓				✓		
Trihinas et al. [157]	2018								✓		
Wang and Gelenbe [158]	2018			✓							
Wei et al. [159]	2018			✓				✓			
Xie and Jia [160]	2018	✓		✓							
Yao and Ansari [26]	2018	✓	✓								
Zhang and Wen [161]	2018	✓	✓								
Zhang et al. [162]	2018			✓							✓

WM workload management, *AEP* application elasticity and provisioning, *GPS* global provisioning and scheduling, *LPS* local provisioning and scheduling, *ADP* application demand profiling, *IDP* (Virtual) infrastructure demand profiling, *Est* resource utilization estimation, *Mon* monitoring, *APr* dynamic application pricing, *IPr* dynamic virtual infrastructure Pricing

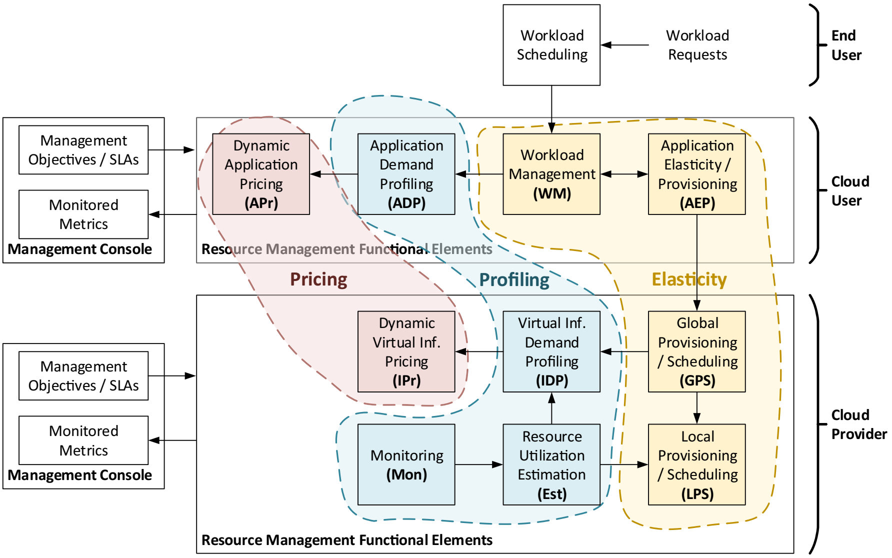
Workload Scheduling
Workload Requests
End User
Management Objectives / SLAs
Monitored Metrics
Management Console
Dynamic Application Pricing (APr)
Application Demand Profiling (ADP)
Workload Management (WM)
Application Elasticity / Provisioning (AEP)
Resource Management Functional Elements
Cloud User
Pricing
Profiling
Elasticity
Dynamic Virtual Inf. Pricing (IPr)
Virtual Inf. Demand Profiling (IDP)
Global Provisioning / Scheduling (GPS)
Management Objectives / SLAs
Monitored Metrics
Management Console
Monitoring (Mon)
Resource Utilization Estimation (Est)
Local Provisioning / Scheduling (LPS)
Cloud Provider
Resource Management Functional Elements

Fig. 4 Conceptual framework for resource management in a cloud environment, as introduced by Jennings and Stadler [9]. In this figure, we added a mapping from the functional elements of the framework to the categories used in this article (Elasticity, Profiling and Pricing)

Table 2 Overview of previous surveys focusing on resource management within cloud environments

Publication	Year	Elasticity	Profiling	Pricing	Traditional	Fog	VM	Container
Jennings and Stadler [9]	2015	✓	✓	✓	✓	✓	✓	
Mann [11]	2015	✓	✓		✓		✓	
Yi et al. [4]	2015	✓		✓		✓	✓	
Zhan et al. [15]	2015	✓	✓		✓		✓	
Herrera and Botero [8]	2016	✓			✓		✓	✓
Masdari et al. [12]	2017	✓			✓		✓	
Yousafzai et al. [14]	2017	✓			✓		✓	
Bittencourt et al. [7]	2018	✓			✓		✓	✓
Kumar et al. [10]	2018	✓		✓	✓		✓	
Mouradian et al. [3]	2018	✓		✓		✓	✓	✓
Pahl et al. [5]	2018	✓			✓			✓
Poullie et al. [13]	2018	✓			✓		✓	✓
Rodriguez and Buyya [6]	2018	✓	✓		✓			✓
Zhang et al. [38]	2018	✓			✓	✓	✓	✓

method, the utility function, the processing mode, and the target instances. Poullie et al. also focused on the allocation of resources, and presented an overview of multi-resource allocation schemes for data centers [13]. Both surveys also mainly focus on the allocation of VMs in traditional cloud environments.

Other surveys are mainly focusing on scheduling and orchestration [5–8, 12, 15]. Bittencourt et al. for example introduced a taxonomy for scheduling in traditional cloud environments [7]. Masdari et al. also investigated the topic of scheduling, but their main focus is on scheduling schemes based on particle swarm optimization [12]. Herrera and Botero focus on Network Functions Virtualization (NFV), and presented an overview of allocation and scheduling schemes for virtual network functions [8]. Rodriguez et al. recently published an extensive overview of orchestration systems specific for container-based clusters [6]. Similarly, Pahl et al. provide an overview of recent research focusing on the orchestration of containers [5]. Zhang et al. recently published a survey on the migration of virtual instances in cloud environments [38]. The authors briefly mention containers and fog computing, but the main focus is the migration of VMs in traditional cloud environments.

When it comes to resource pricing, Kumar et al. provided an overview of dynamic (spot) pricing within traditional clouds [10]. The authors categorized different spot pricing models in three main categories, namely economics based models (auction-based or game theory based), statistics based models and optimization based models.

Recently, Mouradian et al. published an extensive survey on fog computing [3]. In their survey, the authors provided some comments regarding resource allocation, scheduling and pricing in the context of fog computing. Their survey however is not limited to resource management, but instead aims to provide a general overview of all aspects of fog computing. The authors for example also discussed several possible architectures within fog computing.

As can be seen from this overview, previously published surveys either focus on a specific aspect of resource management, or a specific cloud type. Most surveys cover resource management within traditional cloud environments, and do not yet consider containers as an alternative for VMs. In this article however, our goal is to cover the broad range of resource management, and we also do not limit ourselves to a single cloud type or virtualization technology.

3.2 Resource Elasticity

3.2.1 Workload Management

Table 3 provides an overview of recent work with the main focus on the management of user workloads. The scheduling of workloads within a cloud environment differs from scheduling on traditional distributed systems, due to the on-demand resource provisioning and the pay-as-you-go pricing model which is often used by infrastructure providers [152]. A special type of workload is a workflow, which consist of multiple individual tasks that can have several relationships between them. The scheduling of such workflows is often bound by **multiple constraints**, such as strict deadlines for individual tasks [138, 140, 146, 152] and task dependencies [136].

Multiple solutions have been proposed for the scheduling of workflows in a **VM-based** environment [99, 136, 138, 140, 146, 152, 155]. Sahni and Vidyarthi

Table 3 Overview of recent research with the main focus on workload management

Publication	Year	Function				Cloud		Scope		Entity	
		WM	AEP	GPS	LPS	TC	FC	SC	MC	VM	CT
Kumbhare et al. [67]	2015	✓	✓			✓		✓		✓	
Kang et al. [88]	2016	✓				✓		✓			✓
Xu et al. [99]	2016	✓	✓			✓			✓	✓	
Xu et al. [123]	2017	✓	✓			✓		✓			
Atrey et al. [130]	2018	✓	✓			✓		✓		✓	
Cheng et al. [136]	2018	✓	✓	✓		✓		✓		✓	
Diaz-Montes et al. [137]	2018	✓	✓	✓		✓			✓	✓	
Gill et al. [138]	2018	✓				✓		✓		✓	
Guo et al. [140]	2018	✓	✓			✓			✓	✓	
Heidari and Buyya [143]	2018	✓				✓		✓		✓	
Khabbaz and Assi [146]	2018	✓		✓		✓		✓		✓	
Sahni and Vidyarthi [152]	2018	✓	✓			✓		✓		✓	
Sofia et al. [155]	2018	✓	✓	✓		✓		✓		✓	
Simonis [154]	2018	✓	✓			✓			✓		✓
Takahashi et al. [156]	2018	✓	✓			✓			✓	✓	✓
Xie and Jia [160]	2018	✓		✓		✓		✓			

WM workload management, *AEP* application elasticity and provisioning, *GPS* global provisioning and scheduling, *LPS* local provisioning and scheduling, *TC* traditional cloud, *FC* fog computing, *SC* single cloud, *MC* multi-cloud, *VM* virtual machine, *CT* container

for example proposed a dynamic cost-effective deadline-constrained heuristic algorithm for scheduling of scientific workflows using VMs in a public cloud environment [152]. The proposed algorithm aims to minimize the costs, while taking into account the VM performance variability and instance acquisition delay to identify a just-in-time schedule for a deadline-constrained workflow. Guo et al. also introduced a strategy for scheduling of deadline-constrained scientific workflows, but within multi-cloud environments [140]. Their strategy aims to minimize the execution cost of the workflow, while meeting the defined deadline. Similarly, Xu et al. proposed a strategy for the scheduling of scientific workflows in a multi-cloud environment, but their focus is on reducing the energy consumption [99]. Khabbaz et al. proposed a deadline-aware scheduling scheme [146], and focus on improving the data center's Quality of Service (QoS) performance, by considering the request blocking probability and the data center's response time. Sathya Sofia and GaneshKumar on the other hand introduced a multi-objective task scheduling strategy based on a non-dominated sorting genetic algorithm [155]. The proposed algorithm uses a neural network for predicting the required amount of VM resources, based on the characteristics of the tasks and the resource features. Cheng et al. presented a system for resource provisioning and scheduling with task dependencies, based on deep reinforcement learning [136]. The proposed solution also invokes a deep Q-learning-based two-stage resource provisioning and task scheduling processor, for the automatic generation of long-term decisions. Gill et al. argue that few existing resource scheduling algorithms consider cost and execution time constraints [138]. As a result, the authors present a novel strategy for the scheduling of workloads on the available cloud resources, based on Particle Swarm Optimization.

Xu et al. note that inside data centers, there exist a vast amount of delay-tolerant jobs, such as background and maintenance jobs [123]. As a result, the authors proposed a scheme for the provisioning of both delay sensitive and delay-tolerant jobs, that aims to minimize the total operational costs, while still guaranteeing the required QoS for the delay sensitive jobs, and achieving a desirable delay performance for the delay-tolerant jobs.

Big-data computing applications can also benefit from the elasticity of cloud environments [143, 154, 160]. Such applications typically demand concurrent data transfers among the computing nodes, and it is important to determine an optimal transfer schedule in order to achieve a maximum throughput. Xie and Jia however claim that some existing methods cannot achieve this, as they often ignore link bandwidths and the diversity of data replicas and paths [160]. As a result, the authors proposed a max-throughput data transfer scheduling approach that aims to minimize the data retrieval time. Large amounts of data generated by internet and enterprise applications are often stored in the form of graphs. To process such data, graph processing systems are typically used. In this context, Heidari and Buyya proposed two dynamic repartitioning-based algorithms for scheduling of large-scale graphs in a cloud environment [143]. The proposed algorithms consider network factors in order to reduce the costs. The authors also introduced a novel classification for graph algorithms and graph processing systems, which can aid to select the best strategy for processing a given input graph. For real-time big-data applications, stream processing systems are often used instead of batch processing systems as these allow for processing of data upon arrival. However, according to Kumbhare et al., traditional stream processing systems often use simple scaling techniques with elastic cloud resources to handle variable data rates, which can have a significant impact on the application QoS [67]. To tackle this issue, the authors introduced the concept of dynamic dataflows for the scheduling of high-velocity data streams with low latency in the cloud. These dataflows use alternate tasks as additional control over the dataflow's cost and QoS.

In a **federated multi-cloud environment**, different types of resources that may be geographically distributed can be collectively exposed as a single elastic infrastructure. By doing so, the execution of application workflows with heterogeneous and dynamic requirements can be optimized, and the federated multi-cloud can tackle larger scale problems. Diaz-Montes et al. introduced a framework for managing the end-to-end execution of data-intensive application workflows within a federated cloud [137]. The proposed framework also supports dynamic federation, in which computational sites can join or leave on the fly, and the framework can recover from failures happening within a site.

For scheduling of workloads that are executed inside **containers**, Kang et al. proposed a brokering system that aims to minimize the energy consumption, while guaranteeing an acceptable performance level [88]. The authors also proposed a new metric, called Power consumption Per Application (ppA), and the proposed system applies workload clustering using the k-medoids algorithm. Simonis on the other hand presented a container-based architecture for big-data applications, that allows for interoperability across data providers, integrators and users [154]. By using self-contained containers, the presented architecture allows for horizontal scale-out, high reliability and maintainability. Takahashi et al. introduced a portable load balancer for Kubernetes clusters, which is usable in any environment, and hence facilitates the integration of web services [156].

Highlights for workload management: Workloads that are being executed in a cloud environment are often bound by multiple constraints, which should be taken into account by the scheduling strategy to guarantee the required QoS. In recent years, several strategies have been proposed, but most of them focus on the execution inside VM instances, for example by predicting the minimal amount of VM resources required for a given set of tasks.

In a federated multi-cloud environment, geographically distributed resources can be exposed as a single elastic infrastructure, to optimize the execution of application workflows and to tackle large scale problems. An important challenge in this context is support for dynamic federation, meaning that computational sites should be able to join or leave on the fly, and the used framework should be able to cope with such changes.

Container technology can be beneficial for the execution of workloads, especially when using a service oriented architecture, as self-contained containers allow for transparent microservices, horizontal scale-out and high reliability and maintainability.

3.2.2 Application Elasticity and Provisioning

Table 4 provides an overview of recent work with the main focus on application elasticity and provisioning. Applications deployed in a cloud environment can benefit from the offered elasticity by adjusting the provisioned amount of resources based on the current demand. Additional instances can be deployed on the fly, and a load balancer will typically be used to distribute the load over the available instances. Cloud applications however are often stringent to given SLOs, agreed upon between the cloud user and the application end user. In order to satisfy a given service level objective, the minimal amount of cloud resources required for the given task needs to be determined.

Several models have been proposed for the cost-efficient SLA-aware allocation of VM resources in a traditional cloud environment [71, 93, 94, 117]. Salah et al. for example presented an analytical model based on Markov chains, to predict the minimal number of VMs required for satisfying a given SLO performance requirement [94]. Their model takes the offered workload and number of VM instances as input, together with the capacity of each VM instance. The model not only returns the minimal number of VMs required for the workload, but also the required number of load balancers needed for achieving proper elasticity. Mireslami et al. presented a multi-objective cost-effective algorithm for minimizing the deployment cost while meeting the QoS performance requirements [117]. The proposed algorithm offers the cloud user an optimal choice when deploying a web application in a traditional cloud environment. Righi et al. introduced

Table 4 Overview of recent research with the main focus on application elasticity and provisioning

Publication	Year	Function				Cloud		Scope		Entity	
		WM	AEP	GPS	LPS	TC	FC	SC	MC	VM	CT
Moens et al. [71]	2015		✓			✓		✓		✓	
Zhang et al. [78]	2015		✓			✓			✓		
Elgazzar et al. [84]	2016		✓			✓	✓	✓		✓	
Khatua et al. [89]	2016		✓			✓		✓		✓	
Righi et al. [93]	2016		✓			✓		✓		✓	
Salah et al. [94]	2016		✓			✓		✓		✓	
Mebrek et al. [114]	2017		✓			✓	✓		✓		
Mireslami et al. [117]	2017		✓			✓		✓		✓	
Paya and Marinescu [120]	2017		✓			✓		✓		✓	
Alam et al. [128]	2018		✓			✓	✓		✓		✓
Barrameda and Samaan [133]	2018	✓	✓			✓	✓	✓		✓	
Bouet and Conan [135]	2018	✓	✓			✓	✓		✓		
Guo and Shenoy [139]	2018	✓	✓			✓			✓	✓	
Nawrocki et al. [149]	2018	✓	✓			✓	✓	✓			
Rahimi et al. [151]	2018	✓	✓			✓	✓		✓		
Santos et al. [24]	2018		✓	✓		✓	✓		✓		
Yao and Ansari [26]	2018	✓	✓				✓	✓		✓	
Zhang and Wen [161]	2018	✓	✓			✓	✓	✓			

WM workload management, *AEP* application elasticity and provisioning, *GPS* global provisioning and scheduling, *LPS* local provisioning and scheduling, *TC* traditional cloud, *FC* fog computing, *SC* single cloud, *MC* multi-cloud, *VM* virtual machine, *CT* container

a fully-organizing PaaS-level elasticity model, designed specifically for running High-Performance Computing (HPC) applications in the cloud [93]. Their model does not require any user intervention or modifications to the application's source code, but (de-)allocates VMs using an aging-based approach to avoid unnecessary VM re-configurations. The model also uses asynchronism for creating and terminating VMs in order to minimize the execution time of the HPC applications.

In **multi-cloud environments**, applications or individual components should be deployed in the environment that is best suited. Cloud providers may offer their services using different pricing models, and some models may be more suitable for either short term or long term tasks. For the storage of data in heterogeneous multi-cloud environments, Zhang et al. introduced a data hosting scheme which aims to help the cloud user by selecting the most suitable cloud environment, together with an appropriate redundancy strategy for achieving high availability [78]. The proposed solution considers the used pricing strategy, the availability requirements and the data access patterns. For deploying applications in a multi-cloud environment, Khatua et al. introduced several algorithms which aim to determine the optimal amount of resources to be reserved, while minimizing the total cost by selecting the most appropriate pricing model [89].

In a **mobile edge environment**, mobile devices can transfer resource-intensive computations to a more resourceful computing infrastructure, such as a public cloud environment. Multiple offloading approaches exist, often focusing on different objectives or following a different approach [26, 84, 133, 149, 161]. Nawrocki and Sniezynsky for example proposed an agent-based architecture with learning possibilities, based on supervised and reinforcement learning, to optimally schedule services and tasks between the mobile device and the cloud [149]. Elgazzar et al. introduced a framework for cloud-assisted mobile service provisioning, which aims to assist mobile devices in delivering reliable services [84]. The presented framework supports dynamic offloading, based on the current resource utilization and network conditions, while satisfying the user-defined energy constraints. Barrameda and Samaan focus on the costs, and presented a statistical cost model for offloading in a mobile edge environment [133]. In this cost model, the application is modeled as a tree structure for representing dependencies and relations among the application modules. The cost for each module is then modeled as a cumulative distribution function that is statistically estimated through profiling. Zhang et al. on the other hand investigate the topic of energy-efficient task offloading, and proposed an algorithm that aims to minimize the energy consumption on the mobile devices while still guaranteeing deadlines [161]. Somehow related, Mebrek et al. also focus on the energy efficiency, but in the context of a multi-tier IoT-fog-cloud environment, and the authors presented a model for the power consumption and delay for IoT applications within both fog and traditional cloud environments [114]. Similarly, Yao and Ansari presented an approach for offloading and resource provisioning in an IoT-fog environment, but the authors aim to minimize the VM rental cost for the fog environment while still guaranteeing QoS requirements.

For applications running in a **multi-tiered (layered) cloud environment**, which for example could consist of edge devices, a fog and a central cloud layer, Alam et al. presented a layered modular and scalable architecture that aims to increase the efficiency of the applications [128]. The proposed architecture collects and analyzes data at the most efficient and logical place, balances the load, and pushes computation and intelligence to the appropriate layers. Furthermore, the proposed architecture uses Docker containers, which simplifies the management and enables distributed deployments. Similarly, Santos et al. proposed a framework for the autonomous management and orchestration of IoT applications in an edge-fog-cloud environment [24]. The authors introduced a Peer-to-Peer fog protocol for the exchange of application service provisioning information between fog nodes. Rahimi et al. focus on multi-tiered mobile edge environments, and presented a framework for modeling mobile applications as location-time workflows, in which user mobility patterns are translated to mobile service usage patterns [151]. These workflows are then mapped to the appropriate cloud resources using an efficient heuristic algorithm. Bouet and Conan also focus on multi-tiered mobile edge environments, and proposed a geo-clustering approach for optimizing the edge computing resources [135]. The authors introduced an algorithm that provides a partition of mobile edge computing clusters, which consolidates as many communications as possible at the edge.

Highlights for application elasticity and provisioning: Applications deployed in a cloud environment can be stringent to given SLOs. To satisfy these objectives, the required amount of resources needs to be determined. Multiple prediction models have been presented, but most of them focus on the deployment of applications inside VMs. However, VM re-configurations are typically costly and should hence be avoided.

With fog computing, and especially mobile cloud computing, less powerful devices can transfer computational intensive tasks to another environment. This requires an offloading approach, that could for example focus on energy efficiency or minimizing the operational costs. For these environments, containers offer clear benefits, as they facilitate the management and allow for distributed deployments. In multi-cloud environments, the application or individual components should be deployed in the optimal environment, for example to balance the load or to minimize the operational costs.

3.2.3 Local Provisioning and Scheduling

Table 5 provides an overview of recent work with the main focus on local provisioning and scheduling. In **VM-based cloud environments**, multiple VMs are deployed onto a single server, and a hypervisor is used for allocating the virtual resources on top of the physical hardware. Zhang et al. argue that when VMs deployed onto the same physical server compete for memory, the performance of the applications deteriorates, especially for memory-intensive applications [127]. To tackle this issue, the authors proposed an approach for optimizing the memory control using a balloon driver for server consolidation. Li et al. on the other hand argue that the accuracy of CPU proportional sharing and the responsiveness of I/O processing are heavily dependent on the proportion of the allocated CPU resources [110]. The authors illustrate that an inaccurate CPU share ratio, together with CPU proportion dependent I/O responsiveness, can affect the performance of the hypervisor. This could lead

Table 5 Overview of recent research with the main focus on local provisioning and scheduling

Publication	Year	Function				Cloud		Scope		Entity	
		WM	AEP	GPS	LPS	TC	FC	SC	MC	VM	CT
Amannejad et al. [60]	2015		✓		✓	✓	✓	✓		✓	
Katsalis et al. [66]	2015				✓	✓		✓		✓	
Mukherjee et al. [72]	2015				✓	✓		✓		✓	
Nakagawa and Oikawa [91]	2016				✓	✓		✓			✓
Sharma et al. [29]	2016				✓	✓		✓		✓	✓
Li et al. [110]	2017				✓	✓		✓		✓	
Zhang et al. [127]	2017			✓	✓	✓		✓		✓	
Lahmann et al. [147]	2018				✓	✓		✓		✓	✓
Prakash et al. [45]	2018				✓	✓		✓		✓	✓
Tesfatsion et al. [30]	2018				✓	✓		✓		✓	✓

WM workload management, *AEP* application elasticity and provisioning, *GPS* global provisioning and scheduling, *LPS* local provisioning and scheduling, *TC* traditional cloud, *FC* fog computing, *SC* single cloud, *MC* multi-cloud, *VM* virtual machine, *CT* container to unstable performance and therefore could violate SLA requirements. As a result, the authors proposed a novel scheduling scheme that achieves accurate CPU proportional sharing and predictable I/O responsiveness. Katsalis et al. also focus on CPU sharing, and presented several CPU provisioning algorithms for service differentiation in cloud environments [66]. The algorithms are based on dynamic weighted round robin, and guarantee CPU service shares in clusters of servers. Mukherjee et al. argue that, while resource management methods may manage application performance by controlling the sharing of processing time and input-output rates, there is generally no management of contention for virtualization kernel resources or for the memory hierarchy and subsystems [72]. Such contention however can have a significant impact on the application performance. As a result, the authors presented an approach for detecting contention for shared platform resources in virtualized environments. Amennejad et al. illustrate that when VMs compete for shared physical machine resources, the web services deployed on these VMs could suffer performance issues [60]. Cloud users however typically have only access to VM-level metrics and application-level metrics, but these metrics are often not useful for detecting inter-VM contention. To tackle this issue, the authors proposed a machine-learning based interference detection technique to predict whether a given transaction being processed by a web service is suffering from interference. The proposed technique only relies on web transaction response times, and does not require any access to performance metrics of the physical resources.

For **container-based deployments**, Nakagawa and Oikawa argue that deployed containers often consume much more memory than expected [91]. Although there are several methods to prevent such memory overuse, most existing methods have their shortcomings such as an increase in operational costs, or the detection of false-positives. In their paper, the authors proposed a new memory management method for container-based virtualization environments. The proposed method detects containers that have a sign of memory overuse, and puts a limitation on the allowed memory consumption for these containers. Lahmann et al. investigated if VM resource allocation schemes are appropriate for container deployments [147]. Specifically, they focus on the gaps between memory allocation and memory utilization for application deployments in container clusters. Their main conclusion is that VM resource allocation schemes should not simply be used for the allocation of containers, but a fine-grained allocation scheme should be used instead. Sharma et al. studied the differences between hardware virtualization (VMs) and OS virtualization (containers) regarding performance, manageability and software development [29]. According to their findings, containers promise bare metal performance, but they may suffer from performance interference as they share the underlying OS kernel. Unlike VMs which typically have strict resource limits, containers also allow for soft limits, which can be helpful in over-commitment scenarios as they can make use of underutilized resources allocated to other containers. Tesfatsion et al. also studied the differences between VMs and containers, but with a focus on the virtualization overhead [30]. According to the presented results, no single virtualization technology is a clear winner, but each platform has its advantages and shortcomings. Containers for example offer a lower virtualization overhead, but can raise security issues due to the lower level of isolation. Both Tesfatsion and Sharma however note that a hybrid form, in which containers are deployed on top of VMs, could offer promising solutions that combine the advantages of both virtualization technologies.

However, when containers are provisioned inside VMs, the guest OS manages virtual resources inside a VM, whereas the hypervisor manages the physical resources distributed among the VMs. As a result, two control centers are managing the set of resources used by the containers. The hypervisor typically takes control actions such as memory ballooning, which allows a host system to artificially enlarge its memory pool by reclaiming unused memory allocated to other virtual machines, or withdrawal of a virtual CPU to manage over-provisioning, without being aware of the effects of those actions on individual containers deployed inside the VM. Prakash et al. illustrated that such actions can have unpredictable and non-deterministic effects on the nested containers [45]. To tackle this issue, the authors proposed a policy driven controller that smooths over the effects of hypervisor actions on the nested containers.

Highlights for local provisioning and scheduling: In VM-based environments, a hypervisor will strictly allocate resources to the deployed VMs. The deployed VMs however can compete for the shared physical resources, but the hypervisor should detect and prevent this to not violate SLA requirements. With OS-level virtualization, the underlying OS kernel is shared, and containers can use unutilized resources allocated to other containers. These soft limits should be taken into account, as they can have unpredictable effects on other unrelated containers deployed on the same physical hardware. Each virtualization technology clearly has its advantages and limitations, and deploying containers inside VMs could combine the advantages of both technologies, but this introduces challenges for resource management as two control centers are managing the set of resources used by the containers.

3.2.4 Global Provisioning and Scheduling

Table 6 provides an overview of recent work with the main focus on global provisioning and scheduling. As can be seen from this table, a majority of research is focusing on this resource management functional element. When it comes to resource allocation, the used scheme can be either static or dynamic, with the latter indicating that the amount of resources allocated for a specific task can change over time.

For the allocation of resources in a **VM-based environment**, Wolke et al. did an experimental study on the benefits of dynamic resource allocation [46]. According to their findings, reactive or proactive control mechanisms do not always decrease the average server demand, but instead can lead to a high number of migrations, which negatively impacts the response times and could even lead to network congestion. The authors note that in general, live VM migrations should be exceptional, and capacity planning via optimization should be used instead, especially in environments with long-running and predictable application workloads. Somewhat related, Wu et al. studied the overhead introduced by launching new VMs in the context of Cloud bursting [98]. According to their findings, this overhead is not constant, but instead depends on the physical resource utilization (e.g. CPU and I/O device utilization) at the time when the VM is launched. This variation in overhead can have a

Table 6 Overview of recent research with the main focus on global provisioning and scheduling

Publication	Year	Function				Cloud		Scope		Entity	
		WM	AEP	GPS	LPS	TC	FC	SC	MC	VM	CT
AbdelBaky and Unuvar [59]	2015			✓		✓			✓		
Chiang et al. [61]	2015			✓		✓		✓		✓	
Li and Kanso [69]	2015		✓	✓	✓	✓		✓		✓	✓
Liu et al. [70]	2015			✓		✓		✓		✓	
Stankovski et al. [75]	2015			✓		✓			✓		✓
Wuhib et al. [77]	2015			✓		✓		✓			
Ayoubi et al. [80]	2016		✓	✓		✓		✓		✓	
Choi et al. [81]	2016			✓		✓		✓			✓
Dai et al. [83]	2016			✓		✓		✓		✓	
Espling et al. [85]	2016			✓		✓		✓		✓	
Goudarzi et al. [86]	2016			✓		✓		✓		✓	
Huang and Tsang [87]	2016			✓		✓		✓		✓	
Mishra and Bellur [90]	2016			✓		✓		✓		✓	
Pantazoglou et al. [92]	2016			✓		✓		✓		✓	
Wajid et al. [95]	2016			✓		✓			✓	✓	
Wolke et al. [46]	2016			✓		✓		✓		✓	
Wu et al. [98]	2016			✓		✓		✓		✓	
Awada and Barker [101]	2017			✓		✓			✓		✓
Awada and Barker [102]	2017			✓		✓		✓			✓
Hoque et al. [107]	2017			✓			✓	✓			✓
Jin et al. [108]	2017			✓		✓		✓		✓	
Khasnabish et al. [109]	2017			✓		✓		✓			
Li et al. [111]	2017			✓		✓		✓		✓	
Maenhaut et al. [113]	2017		✓	✓		✓		✓			
Mechtri et al. [115]	2017			✓		✓			✓	✓	
Merzoug et al. [116]	2017			✓		✓		✓		✓	
Nardelli et al. [118]	2017			✓		✓		✓		✓	✓
Nitu et al. [119]	2017			✓		✓		✓		✓	✓
Rankothge et al. [121]	2017		✓	✓		✓		✓		✓	
Yang et al. [124]	2017			✓		✓		✓		✓	
Yu and Pan [126]	2017			✓		✓		✓			
Aral and Ovatman [129]	2018		✓	✓		✓	✓		✓		
Barkat et al. [131]	2018			✓		✓			✓	✓	
Govindaraj and Artemenko [41]	2018			✓	✓	✓	✓	✓			✓
Guo et al. [141]	2018			✓		✓			✓	✓	
Jia et al. [144]	2018			✓			✓		✓		
Jia et al. [145]	2018			✓		✓		✓		✓	
Lin et al. [148]	2018		✓	✓		✓	✓	✓			
Stoyanov and Kollingbaum [39]	2018			✓	✓	✓		✓			✓
Wang and Gelenbe [158]	2018			✓		✓		✓			
Wei et al. [159]	2018			✓		✓		✓		✓	

Table 6 (continued)

WM workload management, *AEP* application elasticity and provisioning, *GPS* global provisioning and scheduling, *LPS* local provisioning and scheduling, *TC* traditional cloud, *FC* fog computing, *SC* single cloud, *MC* multi-cloud, *VM* virtual machine, *CT* container

significant impact on cloud bursting strategies. As a result, the authors introduced a VM launching overhead reference model based on operational data, which could help to decide when and where a new VM should be launched.

Global provisioning and scheduling often includes **VM consolidation** [85, 87, 90, 108, 119, 131, 141], which typically aims to pack the virtual machines onto few physical servers in order to reduce the operational costs. Huang et al. for example presented a framework for VM consolidation that aims to achieve a balance among multiple objectives [87], which can also be used in a context that requires minimal system re-configurations. Similarly, Guo et al. presented an approach for the real-time adaptive placement of VMs in large data centers [141]. The authors use a shadow routing based approach, which allows for a large variety of objectives and constraints to be treated within a common framework. When consolidating VMs, both the relationships and possible interference between collocated VMs, as well as the tightness of packing should be taken into account. Espling et al. for example introduced an approach for the placement of VMs with an internal service structure, component relationships and placement constraints between them [85]. Jin et al. presented an approach that takes into account the possible interference between collocated VMs, as this interference can have a negative impact on the performance of the deployed applications [108]. Mishra et al. on the other hand presented a study on the tightness of VM packing [90]. A tight packing approach can lead to future issues as there is no room to expand, whereas provisioning VMs for their peak usage can result in wasted resources as peaks occur infrequently and typically for a short time. Liu et al. however prefer an aggressive resource provisioning approach [70], by initially over-provisioning resources and later reducing the amount of resources if needed. Doing so can increase the performance by reducing the adaption time, while limiting SLO violations when dealing with rapidly increasing workloads. On the physical servers hosting the VMs, some resources could be left unused and therefore wasted when they are insufficient for hosting a new VM. In this context, Nitu et al. proposed a consolidation strategy that dynamically divides a VM into smaller ‘pieces’, so that each piece fits into the available ‘holes’ on the servers [119].

Some provisioning and scheduling schemes have been proposed that focus on the **deployment of containers** in a cloud environment [59, 81, 101, 102, 107, 118]. Awada and Barker for example presented a cloud-based container management service framework, that offers the required functionalities for orchestrating containerized applications [102]. Their framework takes into account the heterogeneous requirements of the applications, and jointly optimizes sets of containerized applications and resource pools within a cloud environment. The authors also presented an extension of their framework for use in multi-region cloud container-instance clusters [101]. Abdelbaky et al. also focus on a multi-cloud environment, and introduced a framework that enables the deployment and management of containers across multiple hybrid clouds and clusters

[59]. Their framework takes into account the objectives and constraints of both the cloud provider and cloud user, and uses a constraint-programming model for selecting the required resources. For the deployment of containers within VMs, Nardelli et al. introduced a strategy for the elastic provisioning of VMs required for deploying the containers [118]. Hoque et al. analyzed different container orchestration tools, and presented a framework for the orchestration of containers within a fog cloud environment [107].

Although containers have distinctive advantages over VMs, the live migration of containers could still introduce a comparatively high overhead and downtime [39, 41]. Stoyanov and Kollingbaum investigated live migration of containers using the popular Checkpoint-Restore in Userspace (CRIU) tool [39]. The authors proposed a novel approach for the live migration that utilizes a recently published CRIU feature called image cache/proxy. Similarly, Govindaraj and Artemenko also proposed a new live migration scheme for containers that aims to reduce the downtime of the migrated container [41].

Live migration is often used for achieving high availability, together with other technologies such as failure detection and checkpoint/restore mechanisms. In this context, Li and Kanso presented a general comparison between VMs and containers from a high availability perspective [69]. According to their findings, there are many solutions available for achieving high availability in a VM environment, typically implemented by the hypervisor as failover clustering. However, current container platforms still lack many of these features. There is some initial work available on container clustering, but the authors note that there are no mature features yet for monitoring or failure detection and recovery, and therefore additional extensions are required on top of container technologies to support high availability in a container-based environment.

Highlights for global provisioning and scheduling: The allocation of resources can be either static or dynamic. A dynamic allocation strategy can lead to a higher efficiency, but the introduced reconfiguration overhead should not be neglected. Therefore, using a dynamic allocation strategy will not always be beneficial, especially when provisioning VMs. The (re)allocation of VMs often includes VM consolidation, which aims to pack the VMs onto few physical servers. During the VM consolidation process, the tightness of packing plays an important role, and possible relationships between VMs should be taken into account.

When deploying containers, an orchestrator is typically used to optimize the allocation scheme over the available resources. Existing container orchestration tools exist for the deployment and management of containers, but these are still relatively young and still lack some important features that are offered in VM environments, for example for achieving high availability which includes live migration of running applications.

3.3 Resource Profiling

Table 7 provides an overview of recent work related to resource profiling, which includes application and infrastructure demand profiling, resource utilization estimation and monitoring.

3.3.1 Application Demand Profiling

When deploying applications in an IaaS cloud environment, both the quantity and type of VM resources need to be determined. Application demand profiling can be used for assessing demand patterns for individual applications, which can be used as input for workload management and application pricing. In this context, Lloyd et al. introduced a workload cost prediction methodology which harnesses operating system time accounting principles to support equivalent workload performance using alternate virtual machine types [112]. By using resource utilization checkpoints, the total resource utilization profile is captured for service oriented application workloads executed across a pool of VM. Based on the obtained workload profiles, the estimated cost is calculated, which could help cloud users for finding alternate infrastructures that afford lower hosting costs while offering equal or better performance. Somewhat related, Prats et al. introduced an approach for the automatic generation of workload profiles [150]. The authors examined and modeled application behavior by finding phases of similar behavior in the workloads. In the presented approach, resource monitoring data is first passed through conditional restricted Boltzmann machines to generate a low-dimensional and time-aware vector. This vector is then passed through clustering methods such as k-means and hidden Markov models to detect the similar behavior phases.

Chard et al. introduced a middleware for the profiling, prediction and provisioning of applications in a cloud environment [104]. The authors have developed an automated profiling service that is able to derive approximate profiles for applications executed on different environments. Based on these profiles, the expected cost is calculated for executing a particular workload in a dynamic cloud market, with the

Table 7 Overview of recent research with the main focus on resource profiling

Publication	Year	Function				Cloud		Scope		Entity	
		ADP	IDP	Est	Mon	TC	FC	SC	MC	VM	CT
Dabbagh et al. [62]	2015		✓	✓	✓	✓		✓		✓	✓
Dhakate and Godbole [63]	2015				✓	✓			✓	✓	✓
D.C. Rodrigues et al. [82]	2016				✓	✓		✓		✓	
Zhou et al. [100]	2016				✓	✓		✓		✓	
Chard et al. [104]	2017	✓				✓			✓	✓	
Dalmazo et al. [105]	2017			✓		✓		✓		✓	
Lloyd et al. [112]	2017	✓				✓		✓		✓	
Balos et al. [132]	2018	✓	✓	✓		✓			✓	✓	
Hauser and Wesner [142]	2018		✓	✓	✓	✓		✓		✓	✓
Prats et al. [150]	2018	✓	✓		✓	✓		✓			
Scheuner and Leitner [153]	2018	✓				✓		✓		✓	
Trihinas et al. [157]	2018				✓	✓		✓	✓	✓	

ADP application demand profiling, *IDP* (Virtual) infrastructure demand profiling, *Est* resource utilization estimation, *Mon* monitoring, *TC* traditional cloud, *FC* fog computing, *SC* single cloud, *MC* multi-cloud, *VM* virtual machine, *CT* container aim of computing bids that are based on probabilistic-durability guarantees. Once the results from profiling and market prediction are obtained, the middleware provisions infrastructure and manages it throughout the course of the workload execution.

Due to the immense growth in the cloud computing market and the resulting wide diversity of cloud services, micro-benchmarks could be used for identifying the best performing cloud services. As a result, Scheuner and Leitner have developed a cloud benchmarking methodology that uses micro-benchmarks to profile applications, in order to predict how an application performs on a wide range of cloud services [153]. The authors validated their approach using several metrics and micro-benchmarks with two applications from different domain. Although micro-benchmarking is a useful approach, the results illustrate that only few selected micro-benchmarks are relevant when estimating the performance of a particular application.

Within the context of scientific computing, Balos et al. present an analytical model that matches scientific applications to effective cloud instances for achieving high application performance [132]. The model constructs two vectors, an application vector consisting of application performance components and a cloud vector comprising cloud-instance performance components. By profiling both the application and cloud instances, an inner product of both vectors is calculated to produce an application-to-cloud score, which represents the application's execution time on the selected cloud instance.

Highlights for application demand profiling: Application demand profiling can be useful for estimating the required amount of resources, as well as the expected operational costs. In a public cloud market, profiling applications can also be used to determine the best suited environment. Applications can either be profiled as a whole, or micro-benchmarks can be used to predict how an application would perform.

3.3.2 Monitoring, Infrastructure Demand Profiling and Resource Utilization Estimation

Cloud monitoring systems play a crucial role for supporting scalability, elasticity, and migrations within a cloud environment. Da Cunha Rodriguez et al. presented a general overview of cloud monitoring [82]. The authors also provided a comparison among relevant cloud monitoring solutions, focusing on abilities such as the accuracy, autonomy and comprehensiveness.

For automatic resource provisioning, the deployed applications, services and the underlying platforms need to be continuously monitored at multiple levels and time intervals. Trihinas et al. however argue that current cloud monitoring tools are either bound to specific cloud platforms, or have limited portability to provide elasticity support [157]. The authors described several challenges for monitoring elastically adaptive multi-cloud services, and introduced an automated, modular, multi-layer and portable cloud monitoring framework. The presented framework can automatically adapt when elasticity actions are enforced to either the cloud service or to the monitoring topology, and can recover from faults introduced in the monitoring configuration.

Hauser and Wesner presented an approach for monitoring resource statistics on the physical infrastructure level [142], to provide the required information for profiling of the physical resources. Based on the monitoring information, a resource utilization profile is provided to the cloud middleware and customer. Such a profile consists of both a static (e.g. number of CPU cores) and dynamic part (e.g. current utilization), and is generated using statistical computations like histograms and Markov chains.

Dabbagh et al. proposed an energy-aware resource provisioning framework that predicts future workloads [62]. Based on monitoring information, the proposed framework predicts the number of future VM requests, along with the amount of CPU and memory resources associated with each of these requests, and provides accurate estimations of the number of physical machines required. Although the proposed solution is based on the provisioning of VMs, the authors note that their framework could easily be adapted for estimating the number of physical machines required for the provisioning of containers.

Monitoring can also play an important role for achieving high availability and reliability. As the public cloud is a multi-tenant environment, failure of a single physical component can have a significant impact on a large number of tenants. To increase cloud reliability, Zhou et al. presented a recovery approach based on checkpoint images, which consist of service checkpoint images and delta checkpoint images [100].

Dhakate and Godbole proposed an architecture for monitoring, testing, reporting and alerting of an entire cloud environment [63]. The required monitoring software is packed inside Docker containers, which can be deployed directly from the Docker Hub repository. The authors also developed a dashboard that provides a general overview of the health status of the whole cloud environment.

Highlights for monitoring, infrastructure demand profiling and resource utilization estimation: Monitoring systems play a crucial role for supporting scalability, elasticity, and migrations within a cloud environment. Together with resource utilization estimation, a resource utilization profile can be generated. Monitoring can also aid in achieving high availability and reliability. When the monitoring system detects a failure, it can initiate a recovery approach, or alert the cloud provider.

3.4 Resource Pricing

Table 8 provides an overview of recent research focusing on resource pricing. As most items focus on (virtual) infrastructure pricing, in the remainder of this section, we will only discuss this functional element. We will first provide a brief overview of research built on top of static pricing models, followed by research focusing on dynamic pricing models.

3.4.1 Static Pricing

In the IaaS market, virtual resources are typically priced using a pay-per-use pricing model, and the granularity of usage for such pricing is often at VM level. However,

Table 8 Overview of recent research with the main focus on resource pricing

Publication	Year	Function		Cloud		Scope		Entity	
		APr	IPr	TC	FC	SC	MC	VM	CT
Aazam and Huh [58]	2015		✓	✓	✓		✓		
Huang et al. [64]	2015		✓	✓	✓		✓		
Jin et al. [65]	2015		✓	✓		✓		✓	
Lee et al. [68]	2015		✓	✓	✓	✓		✓	✓
Mashayekhy et al. [48]	2015		✓	✓		✓		✓	
Petri et al. [73]	2015		✓	✓			✓	✓	
Sharma et al. [74]	2015		✓	✓		✓			
Wang et al. [76]	2015		✓	✓			✓		
Aazam et al. [79]	2016		✓	✓			✓	✓	
Mashayekhy et al. [49]	2016		✓	✓		✓		✓	
Wan et al. [96]	2016		✓	✓		✓			
Wanis et al. [97]	2016	✓	✓	✓		✓		✓	
Babaioff et al. [103]	2017		✓	✓		✓		✓	✓
Chi et al. [47]	2017		✓	✓		✓		✓	
Hai and Nguyen [106]	2017	✓	✓	✓	✓	✓			
Tang et al. [122]	2017		✓	✓		✓	✓	✓	
Yi et al. [125]	2017		✓	✓		✓		✓	✓
Borjigin et al. [134]	2018		✓	✓			✓		
Mikavica et al. [50]	2018		✓	✓		✓		✓	
Zhang et al. [162]	2018		✓	✓		✓			

APr dynamic application pricing, *IPr* dynamic virtual infrastructure Pricing, *TC* traditional cloud, *FC* fog computing, *SC* single cloud, *MC* multi-cloud, *VM* virtual machine, *CT* container

a majority of applications running on top of VMs struggle to fully utilize the allocated amount of resources, leading to a waste of unused resources and are therefore not cost-efficient due to these coarse-grained pricing schemes [65, 68].

Jin et al. investigated an optimized fine-grained and fair pricing scheme [65]. The authors address two main issues: the profits of resource providers and customers often contradict mutually, and the VM maintenance overhead like startup costs are often too huge to be neglected. The presented solution not only derives an optimal price in the acceptable price range, that satisfies both customers and providers, but also finds a best-fill billing cycle to maximize social welfare. Lee et al. also proposed a resource management mechanism for fine-grained resource sharing, which allows for real pay-per-use pricing [68]. Their mechanism consists of a container-based resource allocator, and a real-usage based pricing scheme. By using containers instead of virtual machines, a higher resource utilization can be achieved and the authors also illustrate that the proposed mechanism can achieve a near-optimal cost efficiency.

Tang et al. investigated the problem of joint pricing and capacity planning in the IaaS provider market [122]. The authors studied two models, in the first model there is a single IaaS provider (monopoly market), whereas the second model considers multiple IaaS providers. For the monopoly market model, the authors proposed a method for determining the optimal amount of end-user requests to admit and number of VMs to lease for SaaS providers, based on the current resource price charged by the IaaS provider. For the model with multiple IaaS providers, the authors proposed an iterative game-theory based algorithm for finding the so-called Nash equilibrium. Borjigin et al. also presented an approach for finding the Nash equilibrium, but within NFV markets [134]. The presented double-auction approach aims to maximize the profits for all participants, being the brokers, the cloud users and the cloud providers.

Yi et al. argue that cloud users with small and short demands, typically cannot find an instance type offered by a cloud provider that fits their needs or fully utilizes the purchased instance-hours [125]. On the other hand, cloud providers are faced with the challenge of consolidating small, short jobs, which exhibit strong dynamics, to effectively improve resource utilization. To address these issues, the authors proposed a novel group buying mechanism that organizes jobs with complementary resource demands into groups, and allocates them to container group buying deals predefined by cloud providers. Each group buying deal offers a resource pool for all the jobs in the deal, which can be implemented as a virtual machine or a physical server. By running each job inside a container, the proposed solution allows for flexible resource sharing among the different users in the same group buying deal, while improving resource utilization for the cloud providers.

Highlights for static virtual infrastructure pricing: Static pricing models are often based on the number of provisioned VMs. A majority of applications however struggle to fully utilize the allocated amount of resources, leading to a waste of unused resources. A fine-grained pricing model could tackle this issue, presenting an interesting opportunity for the deployment of applications inside containers. Small and short tasks can be executed in containers, which can then be grouped and allocated to VMs. A group buying approach can be used for acquiring the required set of VMs.

3.4.2 Dynamic Pricing

When using a dynamic, **auction-based pricing model**, multiple cloud users bid for a bundle of typically heterogeneous cloud instances. The cloud provider will then select a set of cloud users, and needs to determine a feasible allocation over its set of physical machines. A major issue with dynamic auction-based pricing is that cloud users are typically self-interested, meaning that they want to maximize their own utility. The cloud users could untruthfully alter their requests, for example by requesting several sets of resources different from their actual need, in order to manipulate the outcomes of the bidding and to gain an unfair advantage [48, 76, 79].

To tackle this issue, Mashayekhy et al. [48] proposed a resource management mechanism that consists of three phases: winner determination, provisioning and allocation, and pricing. In the winner determination phase, the cloud provider decides which users receive the requested bundles. In the provisioning and allocation phase, VM instances are provisioned to the winning users. In the pricing phase, the cloud provider dynamically determines the price that the winning users should pay for their requests. The authors claim that their solution is strategy-proof, meaning that cloud users have no incentives to lie about their requested bundles and their valuations. In [49], the authors proposed an auction-based online mechanism for VM provisioning, allocation and pricing in clouds that considers several types of resources. The proposed mechanism allocates VM instances to selected users for the period they are requested for, and ensures that users will continue using their VM instances for the requested period. In addition, the mechanism determines the price users have to pay for using the allocated resources. The authors proved that the mechanism is incentive-compatible, meaning that it gives incentives to users to reveal their actual requests.

Cloud data centers often consist of heterogeneous infrastructure, and the cloud provider could adapt the offered prices based on the used hardware. Zhang et al. for example presented an approach for the pricing of cloud storage for data centers consisting of multiple storage tiers that offer distinct characteristics [162]. The approach is based on a two-stage auction process for requesting storage capacity and accesses with given latency requirements. The presented solution provides a hybrid storage and access optimization framework, which aims to maximize the cloud provider's net profit over multiple dimensions.

When the current demand is low, cloud providers can offer their services at a lower price, e.g. Amazon's **spot instances**. Recently, Amazon introduced a new variety of spot instances, namely spot block instances [163]. These instances run continuously for a finite duration (1 to 6 hours). Pricing is based on the requested duration and the available resources, and spot block prices are typically 30 to 45% less than on-demand prices. Mikavica et al. analyzed two auction-based pricing mechanisms, namely uniform price auction and generalized second-price auction, for pricing the cloud provider's idle resources in the form of spot block instances [50]. Furthermore, the authors proposed a model for spot block price determination under these pricing mechanisms. The presented results show that, regardless of the chosen auction mechanism and bidding strategy, spot block instances are a cost-effective solution that embodies advantages of both on-demand instances and spot instances. Wan et al. on the other hand present a reactive pricing algorithm, allowing the cloud provider to determine the server price based on the actual resource demand [96]. The presented approach takes into account the renewable energy, spot power price and the battery level, and dynamically tunes the server price in response to state changes. The authors focus on pricing of physical servers, but the presented approach can easily be extended for pricing of VMs.

In a **multi-cloud environment**, service and resource providers can co-exist in a market where the relationship between clients and services depends on the nature of the application and can be subject to a variety of different QoS constraints. Deciding whether a cloud provider should host a service in the long-term would be influenced by parameters such as the service price, the QoS guarantees required by the customers, the deployment costs and the constraints. In this context, Petri et al. introduced a market model to support federated clouds and investigate its efficiency using two real application scenarios [73]. The authors also identified a cost-decision based mechanism to determine when tasks should be outsourced to external sites in the federation. Wang et al. also focused on multi-cloud environments, by introducing an intelligent economic approach for dynamic resource allocation, which can be used for the trading of various kinds of resources among multiple consumers and providers [76]. The presented approach is based on intelligent combinatorial double auction, and includes a price formation mechanism, consisting of price prediction and matching. The authors also proposed a reputation system to exclude dishonest participants, as well as a paddy field algorithm for selecting the winners.

In a **federated cloud environment**, services can be provided through two or more clouds, which is often done using a middleware entity, called a cloud broker. Such cloud broker is responsible for reserving and managing the resources, discovering services according to the customer's demands, SLA negotiation and match-making between the involved service provider and the customer. Aazam et al. presented a holistic brokerage model to manage on-demand and advanced service reservation, pricing and reimbursement [79]. The authors consider dynamic management of customer's characteristics as well as taking into account historical records when evaluating the economics related factors. Futhermore, they introduced a mechanism of incentives and penalties, which helps to establish trust between the cloud users and service providers.

Highlights for dynamic virtual infrastructure pricing: With a dynamic, auction-based pricing model, multiple cloud users bid for a bundle of cloud resources. A major issue with this is that the cloud users can alter their requests in order to manipulate the bidding outcomes. To tackle this issue, the cloud provider could give incentives to the users to reveal their actual requests. In federated and other multi-cloud environments, a broker is typically used for reserving and managing resources. When allocating resources, this broker could take into account the actual prices offered by the different environments, in order to minimize the costs.

4 Challenges and Opportunities

Virtualization is the fundamental technology that powers cloud computing, and the majority of cloud providers are still providing virtual resources in the form of VMs to the cloud users. As a result, most research related to resource management in cloud environments is focusing on the different aspects related to the provisioning, profiling and pricing of such VMs. Container technology however is gaining popularity, as it offers a more lightweight alternative to traditional VMs. Apart from this new virtualization technology, new cloud models are emerging, bringing the cloud closer to the end user, which is especially useful for devices with a limited network connection, or for low-latency applications. In this section, we identify several challenges and opportunities for resource management in cloud environments, mainly related to these recent trends.

4.1 Dynamic Resource Allocation for Containerized Applications

Dynamic resource allocation for VMs will not always be beneficial for the cloud environment, due to the costly nature of VM migrations [46]. Existing dynamic resource allocation approaches therefore often put a heavy penalty on such migrations to avoid unnecessary VM re-configurations. Containers on the other hand are more lightweight and portable, but the live migration of containers still remains an important research challenge [39]. Existing methods can cause significant delays [39, 41], resulting in a relatively high downtime. Furthermore, when migrating containers, a feasible destination host must be selected, which for example supports the libraries required by the migrated container.

In this context, future research could investigate how existing dynamic allocation strategies designed for the allocation of VMs perform when handling containers. Lahmann et al. already did some initial research [147], but with the main focus on memory allocation and memory utilization. Existing methods should also be extended to support automated destination host selection, which requires sufficient knowledge about the deployed containers. Furthermore, when containers are deployed inside VMs, it could be interesting to study the effects of using a static resource allocation strategy for the provisioning of the virtual machines, combined with a dynamic strategy for the deployment of containers inside the provisioned VMs.

4.2 Cloud Management Systems for Bare-Metal Containers

When containers are deployed inside VMs, actions taken by the hypervisor can have unpredictable and non-deterministic effects on the nested containers [45]. Virtual machines also introduce a noticeable overhead, as they typically run a full software stack. When running containers directly on the OS of the physical machine, this overhead could be eliminated, which could lead to a higher scalability, efficiency and a higher resource utilization. This however introduces the need for a cloud management system that manages the allocation of containers on the physical hardware.

There are already some valuable tools available to implement such management system. Juju for example is an open source tool developed by Canonical, to facilitate the deployment and scaling in cloud environments, and can also be used for the management of containers [164]. Kubernetes on the other hand is a container orchestration system, designed for the deployment, scaling and management of containerized applications, and can be deployed using juju. However, a bare-metal cloud container management system should not only provide the required functionality for allocating and provisioning containers, but should also guarantee sufficient security and isolation between the different tenants. Achieving a clear isolation is challenging, as containers share the underlying OS kernel [29]. The use of containers could introduce some security risks, and a container management system should implement some protection mechanisms [42–44]. Furthermore, the system should also monitor the actual amount of resources used over time by the deployed containers, in order to charge the customers based on the actual resource usage. Unlike VMs, containers often have soft limits, meaning that the actual usage can be different from the allocated amount of resources [29, 91]. This presents opportunities for achieving a higher overall resource utilization, but the management system should also have built-in functionalities for preventing starvation when highly demanding containers clog up all available resources.

4.3 Management of a Hybrid Edge/Fog/Cloud Environment

In a hybrid edge-fog-cloud environment, resources that may be geographically distributed can be collectively exposed as a single elastic infrastructure. This however introduces the need for a framework that coordinates the management of resources among the different environments. While there is already some initial research available [165], many research challenges are still remaining.

To achieve an efficient deployment of applications in such an environment, a feasible location for each component needs to be determined [128], ideally in an autonomous way. Computation offloading can be used for transferring resource-intensive computations from less powerful devices to a more powerful cloud environment [84], but tasks and applications can also be offloaded to the fog environment to reduce latency and preserve bandwidth [135]. The management framework should support dynamic offloading based on the resource status of the mobile systems and the current network conditions, but should also satisfy the user-defined objectives and constraints.

The use of portable containers presents interesting opportunities to facilitate the management and migration of the components. For components deployed in a public cloud environment, security challenges introduced by the multi-tenant cloud environment should also be addressed [166, 167]. A hybrid environment should also allow for auditing in order to create reliable and secure cloud services [168].

4.4 Experimental Validation of Resource Management Strategies

Resource management strategies are often only validated by means of simulations [169], for example by using CloudSim [170], in which the whole cloud computing environment is modeled and simulated in software. This is mainly because of the nature of the research, as resource allocation strategies for example are often designed for managing large sets of applications within large cloud environments. Experimental validation using real cloud hardware would not only be costly as it would require multiple cloud instances for a relative long time period, the validation process would also be time-consuming. Some large-scale academic testbed environments have been developed to support experimentation in a wide variety of research domains and with increased realism compared to simulations, such as the Fed4Fire [171] and the FUTEBOL [172] projects. Although these environments allow for large-scale system validation and offer valuable toolsets for experimentation, they have limited infrastructure resource availability as they are heavily used by researchers worldwide, as well as considerable software and hardware maintenance costs. Typically, these testbeds are used for large and mature validation tests and are less suited for small repetitive tests with highly frequent updates.

The rise of new cloud types such as fog cloud environments, as well as the adoption of container technology however can facilitate the validation of resource management strategies. Using low-cost hardware, a small-scale test bed could be built for the initial validation. A Raspberry Pi for example is already powerful enough to host several containers. By combining experiments on a small-scale test bed with simulations using large-scale scenarios, the research would not only gain credibility, but an implementation of the proposed solution on real hardware would also illustrate that the resource management strategy works in practice.

4.5 Towards Serverless Cloud Computing

Although container technology is gaining popularity, cloud computing is still mainly built around the provisioning of VMs, and cloud users are typically charged based on the number of provisioned VMs. A majority of applications however struggle to fully utilize the allocated amount of resources, leading to a waste of unused resources [65]. A fine-grained pricing model could tackle this issue, presenting an interesting opportunity for the deployment of applications inside containers. This is also one of the main ideas behind serverless computing [173]. Serverless computing is an event-driven cloud execution model, in which the cloud user provides the code and the cloud provider manages the life-cycle of the execution environment of that code. Cloud users are then charged based on the actual amount of resources consumed by an application, rather than on pre-purchased units of capacity. Serverless computing could facilitate cloud deployments, as the cloud user no longer needs to deploy and manage several cloud instances, and could also offer economic advantages especially for the execution of small, short jobs. Furthermore, containers could play an important role in the evolution of serverless computing, as they can be deployed easily and fast and introduce minimal overhead. Therefore, serverless computing could become more adopted in the near future, and it could also facilitate the step towards cloud computing for a broader audience.

5 Conclusions

In this article, we presented an overview of recent research, published between 2015 and 2018, with the main focus on resource management within cloud environments. We especially investigated how cloud resource management is adapting to support newly introduced trends, such as containers as the virtualization technology and the rise of fog/edge computing. We categorized the research items based on the main resource management functional element, and provided a brief summary for each element. While the majority of recent research is still focusing on the management of virtual machines in a traditional single cloud environment, we identified several interesting opportunities for resource management in a future fully containerized multi-tiered edge-fog-cloud, which could overcome many shortcomings of today's cloud environments.

References

1. Swift—openstack. https://wiki.openstack.org/wiki/Swift. Accessed 9 Sept 2019
2. Openstack—build the future of open infrastructure. http://openstack.org. Accessed 9 Sept 2019
3. Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R.H., Morrow, M.J., Polakos, P.A.: A comprehensive survey on fog computing: state-of-the-art and research challenges. IEEE Commun. Surv. Tutor. **20**(1), 416–464 (2018). https://doi.org/10.1109/COMST.2017.2771153
4. Yi, S., Li, C., Li, Q.: A survey of fog computing: concepts, applications and issues. In: Proceedings of the 2015 Workshop on Mobile Big Data, Mobidata '15, pp. 37–42. ACM, New York (2015). https://doi.org/10.1145/2757384.2757397
5. Pahl, C., Brogi, A., Soldani, J., Jamshidi, P.: Cloud container technologies: a state-of-the-art review. In: IEEE Transactions on Cloud Computing, pp. 1–1 (2018). https://doi.org/10.1109/TCC.2017.2702586
6. Rodriguez, M.A., Buyya, R.: Container-based cluster orchestration systems: a taxonomy and future directions. Softw. Pract. Exp. (2018). https://doi.org/10.1002/spe.2660
7. Bittencourt, L.F., Goldman, A., Madeira, E.R., da Fonseca, N.L., Sakellariou, R.: Scheduling in distributed systems: a cloud computing perspective. Comput. Sci. Rev. **30**, 31–54 (2018). https://doi.org/10.1016/j.cosrev.2018.08.002
8. Herrera, J.G., Botero, J.F.: Resource allocation in NFV: a comprehensive survey. IEEE Trans. Netw. Serv. Manag. **13**(3), 518–532 (2016). https://doi.org/10.1109/TNSM.2016.2598420
9. Jennings, B., Stadler, R.: Resource management in clouds: survey and research challenges. IEEE Trans. Netw. Serv. Manag. **23**(3), 567–619 (2015). https://doi.org/10.1007/s10922-014-9307-7
10. Kumar, D., Baranwal, G., Raza, Z., Vidyarthi, D.P.: A survey on spot pricing in cloud computing. J. Netw. Syst. Manag. **26**(4), 809–856 (2018). https://doi.org/10.1007/s10922-017-9444-x
11. Mann, Z.A.: Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. ACM Comput. Surv. **48**(1), 11:1–11:34 (2015). https://doi.org/10.1145/2797211
12. Masdari, M., Salehi, F., Jalali, M., Bidaki, M.: A survey of PSO-based scheduling algorithms in cloud computing. J. Netw. Syst. Manag. **25**(1), 122–158 (2017). https://doi.org/10.1007/s10922-016-9385-9
13. Poullie, P., Bocek, T., Stiller, B.: A survey of the state-of-the-art in fair multi-resource allocations for data centers. IEEE Trans. Netw. Serv. Manag. **15**(1), 169–183 (2018). https://doi.org/10.1109/TNSM.2017.2743066
14. Yousafzai, A., Gani, A., Noor, R.M., Sookhak, M., Talebian, H., Shiraz, M., Khan, M.K.: Cloud resource allocation schemes: review, taxonomy, and opportunities. Knowl. Inf. Syst. **50**(2), 347–381 (2017). https://doi.org/10.1007/s10115-016-0951-y
15. Zhan, Z.H., Liu, X.F., Gong, Y.J., Zhang, J., Chung, H.S.H., Li, Y.: Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Comput. Surv. **47**(4), 63:1–63:33 (2015). https://doi.org/10.1145/2788397
16. Mell, P., Grance, T.: Sp 800-145. The NIST definition of cloud computing. Technical report (2011)
17. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010). https://doi.org/10.1145/1721654.1721672
18. Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P., Riviere, E.: Edge-centric computing: vision and challenges. SIGCOMM Comput. Commun. Rev. **45**(5), 37–42 (2015). https://doi.org/10.1145/2831347.2831354
19. Hong, H.: From cloud computing to FOG computing: unleash the power of edge and end devices. In: 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 331–334 (2017). https://doi.org/10.1109/CloudCom.2017.53
20. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC '12, pp. 13–16. ACM, New York (2012). https://doi.org/10.1145/2342509.2342513
21. Iorga, M., Feldman, L.B., Barton, R., Martin, M., Goren, N.S., Mahmoudi, C.: Sp 500-325. Fog computing conceptual model. Technical report (2018)
22. Dolui, K., Datta, S.K.: Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In: 2017 Global Internet of Things Summit (GIoTS), pp. 1–6 (2017). https://doi.org/10.1109/GIOTS.2017.8016213

23. Haouari, F., Faraj, R., AlJa'am, J.M.: Fog computing potentials, applications, and challenges. In: 2018 International Conference on Computer and Applications (ICCA), pp. 399–406 (2018). https://doi.org/10.1109/COMAPP.2018.8460182
24. Santos, J., Wauters, T., Volckaert, B., De Turck, F.: Fog computing: enabling the management and orchestration of smart city applications in 5G networks. Entropy (2018). https://doi.org/10.3390/e20010004
25. Sarkar, S., Chatterjee, S., Misra, S.: Assessment of the suitability of fog computing in the context of internet of things. IEEE Trans. Cloud Comput. **6**(1), 46–59 (2018). https://doi.org/10.1109/TCC.2015.2485206
26. Yao, J., Ansari, N.: Qos-aware fog resource provisioning and mobile device power control in IOT networks. IEEE Trans. Netw. Serv. Manag. **16**(1), 1 (2018). https://doi.org/10.1109/TNSM.2018.2888481
27. Adufu, T., Choi, J., Kim, Y.: Is container-based technology a winner for high performance scientific applications? In: 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 507–510 (2015). https://doi.org/10.1109/APNOMS.2015.7275379
28. Eberbach, E., Reuter, A.: Toward El Dorado for cloud computing: lightweight VMs, containers, meta-containers and oracles. In: Proceedings of the 2015 European Conference on Software Architecture Workshops, ECSAW '15, pp. 13:1–13:7. ACM, New York (2015). https://doi.org/10.1145/2797433.2797446
29. Sharma, P., Chaufournier, L., Shenoy, P., Tay, Y.C.: Containers and virtual machines at scale: A comparative study. In: Proceedings of the 17th International Middleware Conference, Middleware '16, pp. 1:1–1:13. ACM, New York (2016). https://doi.org/10.1145/2988336.2988337
30. Tesfatsion, S.K., Klein, C., Tordsson, J.: Virtualization techniques compared: performance, resource, and power usage overheads in clouds. In: Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering, ICPE '18, pp. 145–156. ACM, New York (2018). https://doi.org/10.1145/3184407.3184414
31. Linux containers. https://linuxcontainers.org. Accessed 9 Sept 2019
32. Docker—enterprise container platform. https://www.docker.com. Accessed 9 Sept 2019
33. Docker–docker hub. https://www.docker.com/products/docker-hub. Accessed 9 Sept 2019
34. Kubernetes—production-grade container orchestration. https://kubernetes.io. Accessed 9 Sept 2019
35. Docker—swarm mode overview. https://docs.docker.com/engine/swarm/. Accessed 9 Sept 2019
36. Docker blog—extending docker enterprise edition to support kubernetes. https://blog.docker.com/2017/10/docker-enterprise-edition-kubernetes/. Accessed 9 Sept 2019
37. Reniers, V.: The prospects for multi-cloud deployment of SaaS applications with container orchestration platforms. In: Proceedings of the Doctoral Symposium of the 17th International Middleware Conference, Middleware Doctoral Symposium'16, pp. 5:1–5:2. ACM, New York (2016). https://doi.org/10.1145/3009925.3009930
38. Zhang, F., Liu, G., Fu, X., Yahyapour, R.: A survey on virtual machine migration: challenges, techniques, and open issues. IEEE Commun. Surv. Tutor. **20**(2), 1206–1243 (2018). https://doi.org/10.1109/COMST.2018.2794881
39. Stoyanov, R., Kollingbaum, M.J.: Efficient live migration of Linux containers. In: Yokota, R., Weiland, M., Shalf, J., Alam, S. (eds.) High Performance Computing, pp. 184–193. Springer International Publishing, Cham (2018)
40. CRIU—checkpoint/restore in userspace. https://criu.org. Accessed 9 Sept 2019
41. Govindaraj, K., Artemenko, A.: Container live migration for latency critical industrial applications on edge computing. In: 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), vol. 1, pp. 83–90. (2018). https://doi.org/10.1109/ETFA.2018.8502659
42. Mattetti, M., Shulman-Peleg, A., Allouche, Y., Corradi, A., Dolev, S., Foschini, L.: Securing the infrastructure and the workloads of Linux containers. In: 2015 IEEE Conference on Communications and Network Security (CNS), pp. 559–567 (2015). https://doi.org/10.1109/CNS.2015.7346869
43. Young, E.G., Zhu, P., Caraza-Harter, T., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H.: The true cost of containing: a gvisor case study. In: 11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19). USENIX Association, Renton (2019). https://www.usenix.org/conference/hotcloud19/presentation/young
44. Bui, T.: Analysis of Docker Security. arXiv e-prints (2015)

45. Prakash, C., Prashanth, P., Bellur, U., Kulkarni, P.: Deterministic container resource management in derivative clouds. In: 2018 IEEE International Conference on Cloud Engineering (IC2E), pp. 79–89 (2018). https://doi.org/10.1109/IC2E.2018.00030
46. Wolke, A., Bichler, M., Setzer, T.: Planning vs. dynamic control: resource allocation in corporate clouds. IEEE Trans. Cloud Comput. **4**(3), 322–335 (2016). https://doi.org/10.1109/TCC.2014.2360399
47. Chi, Y., Li, X., Wang, X., Leung, V.C.M., Shami, A.: A fairness-aware pricing methodology for revenue enhancement in service cloud infrastructure. IEEE Syst. J. **11**(2), 1006–1017 (2017). https://doi.org/10.1109/JSYST.2015.2448719
48. Mashayekhy, L., Nejad, M.M., Grosu, D.: Physical machine resource management in clouds: a mechanism design approach. IEEE Trans. Cloud Comput. **3**(3), 247–260 (2015). https://doi.org/10.1109/TCC.2014.2369419
49. Mashayekhy, L., Nejad, M.M., Grosu, D., Vasilakos, A.V.: An online mechanism for resource allocation and pricing in clouds. IEEE Trans. Comput. **65**(4), 1172–1184 (2016). https://doi.org/10.1109/TC.2015.2444843
50. Mikavica, B., Kostić-Ljubisavljević, A.: Pricing and bidding strategies for cloud spot block instances. In: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 0384–0389 (2018). https://doi.org/10.23919/MIPRO.2018.8400073
51. Weinman, J.: Cloud pricing and markets. IEEE Cloud Comput. **2**(1), 10–13 (2015). https://doi.org/10.1109/MCC.2015.3
52. ACM Transactions on Internet Technology (TOIT). https://dl.acm.org/citation.cfm?id=J780. Accessed 9 Sept 2019
53. IEEE transactions on cloud computing (tcc). https://www.computer.org/csdl/journal/cc. Accessed 9 Sept 2019
54. IEEE transactions on parallel and distributed systems (TPDS). https://www.computer.org/csdl/journal/td. Accessed 9 Sept 2019
55. IEEE transactions on network and service management (TNSM). https://www.comsoc.org/publications/journals/ieee-tnsm. Accessed 9 Sept 2019
56. Springer journal of network and systems management (jnsm). https://www.springer.com/computer/communication+networks/journal/10922. Accessed 9 Sept 2019
57. Wiley journal of software: practice and experience (spe). https://onlinelibrary.wiley.com/journal/1097024x. Accessed 9 Sept 2019
58. Aazam, M., Huh, E.: Fog computing micro datacenter based dynamic resource estimation and pricing model for iot. In: 2015 IEEE 29th International Conference on Advanced Information Networking and Applications, pp. 687–694 (2015). https://doi.org/10.1109/AINA.2015.254
59. Abdelbaky, M., Diaz-Montes, J., Parashar, M., Unuvar, M., Steinder, M.: Docker containers across multiple clouds and data centers. In: 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), pp. 368–371 (2015)
60. Amannejad, Y., Krishnamurthy, D., Far, B.: Managing performance interference in cloud-based web services. IEEE Trans. Netw. Serv. Manag. **12**(3), 320–333 (2015). https://doi.org/10.1109/TNSM.2015.2456172
61. Chiang, Y., Ouyang, Y., Hsu, C.: An efficient green control algorithm in cloud computing for cost optimization. IEEE Trans. Cloud Comput. **3**(2), 145–155 (2015). https://doi.org/10.1109/TCC.2014.2350492
62. Dabbagh, M., Hamdaoui, B., Guizani, M., Rayes, A.: Energy-efficient resource allocation and provisioning framework for cloud data centers. IEEE Trans. Netw. Serv. Manag. **12**(3), 377–391 (2015). https://doi.org/10.1109/TNSM.2015.2436408
63. Dhakate, S., Godbole, A.: Distributed cloud monitoring using docker as next generation container virtualization technology. In: 2015 Annual IEEE India Conference (INDICON), pp. 1–5 (2015). https://doi.org/10.1109/INDICON.2015.7443771
64. Huang, X., Yu, R., Kang, J., Ding, J., Maharjan, S., Gjessing, S., Zhang, Y.: Dynamic resource pricing and scalable cooperation for mobile cloud computing. In: 2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), pp. 786–792 (2015). https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.155

65. Jin, H., Wang, X., Wu, S., Di, S., Shi, X.: Towards optimized fine-grained pricing of IaaS cloud platform. IEEE Trans. Cloud Comput. **3**(4), 436–448 (2015). https://doi.org/10.1109/TCC.2014.2344680
66. Katsalis, K., Paschos, G.S., Viniotis, Y., Tassiulas, L.: Cpu provisioning algorithms for service differentiation in cloud-based environments. IEEE Trans. Netw. Serv. Manag. **12**(1), 61–74 (2015). https://doi.org/10.1109/TNSM.2015.2397345
67. Kumbhare, A.G., Simmhan, Y., Frincu, M., Prasanna, V.K.: Reactive resource provisioning heuristics for dynamic dataflows on cloud infrastructure. IEEE Trans. Cloud Comput. **3**(2), 105–118 (2015). https://doi.org/10.1109/TCC.2015.2394316
68. Lee, Y.C., Kim, Y., Han, H., Kang, S.: Fine-grained, adaptive resource sharing for real pay-per-use pricing in clouds. In: 2015 International Conference on Cloud and Autonomic Computing, pp. 236–243 (2015). https://doi.org/10.1109/ICCAC.2015.36
69. Li, W., Kanso, A.: Comparing containers versus virtual machines for achieving high availability. In: 2015 IEEE International Conference on Cloud Engineering, pp. 353–358 (2015). https://doi.org/10.1109/IC2E.2015.79
70. Liu, J., Zhang, Y., Zhou, Y., Zhang, D., Liu, H.: Aggressive resource provisioning for ensuring qos in virtualized environments. IEEE Trans. Cloud Comput. **3**(2), 119–131 (2015). https://doi.org/10.1109/TCC.2014.2353045
71. Moens, H., Dhoedt, B., Turck, F.D.: Allocating resources for customizable multi-tenant applications in clouds using dynamic feature placement. Future Gener. Comput. Syst. **53**, 63–76 (2015). https://doi.org/10.1016/j.future.2015.05.017
72. Mukherjee, J., Krishnamurthy, D., Rolia, J.: Resource contention detection in virtualized environments. IEEE Trans. Netw. Serv. Manag. **12**(2), 217–231 (2015). https://doi.org/10.1109/TNSM.2015.2407273
73. Petri, I., Diaz-Montes, J., Zou, M., Beach, T., Rana, O., Parashar, M.: Market models for federated clouds. IEEE Trans. Cloud Comput. **3**(3), 398–410 (2015). https://doi.org/10.1109/TCC.2015.2415792
74. Sharma, B., Thulasiram, R.K., Thulasiraman, P., Buyya, R.: Clabacus: a risk-adjusted cloud resources pricing model using financial option theory. IEEE Trans. Cloud Comput. **3**(3), 332–344 (2015). https://doi.org/10.1109/TCC.2014.2382099
75. Stankovski, V., Taherizadeh, S., Taylor, I., Jones, A., Mastroianni, C., Becker, B., Suhartanto, H.: Towards an environment supporting resilience, high-availability, reproducibility and reliability for cloud applications. In: 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), pp. 383–386 (2015). https://doi.org/10.1109/UCC.2015.61
76. Wang, X., Wang, X., Che, H., Li, K., Huang, M., Gao, C.: An intelligent economic approach for dynamic resource allocation in cloud services. IEEE Trans. Cloud Comput. **3**(3), 275–289 (2015). https://doi.org/10.1109/TCC.2015.2415776
77. Wuhib, F., Yanggratoke, R., Stadler, R.: Allocating compute and network resources under management objectives in large-scale clouds. J. Netw. Syst. Manag. **23**(1), 111–136 (2015). https://doi.org/10.1007/s10922-013-9280-6
78. Zhang, Q., Li, S., Li, Z., Xing, Y., Yang, Z., Dai, Y.: Charm: a cost-efficient multi-cloud data hosting scheme with high availability. IEEE Trans. Cloud Comput. **3**(3), 372–386 (2015). https://doi.org/10.1109/TCC.2015.2417534
79. Aazam, M., Huh, E., St-Hilaire, M., Lung, C., Lambadaris, I.: Cloud customer's historical record based resource pricing. IEEE Trans. Parallel Distrib. Syst. **27**(7), 1929–1940 (2016). https://doi.org/10.1109/TPDS.2015.2473850
80. Ayoubi, S., Zhang, Y., Assi, C.: A reliable embedding framework for elastic virtualized services in the cloud. IEEE Trans. Netw. Serv. Manag. **13**(3), 489–503 (2016). https://doi.org/10.1109/TNSM.2016.2581484
81. Choi, S., Myung, R., Choi, H., Chung, K., Gil, J., Yu, H.: GPSF: General-purpose scheduling framework for container based on cloud environment. In: 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 769–772 (2016). https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2016.162
82. Da Cunha Rodrigues, G., Calheiros, R.N., Guimaraes, V.T., Santos, G.L.d., de Carvalho, M.B., Granville, L.Z., Tarouco, L.M.R., Buyya, R.: Monitoring of cloud computing environments: concepts, solutions, trends, and future directions. In: Proceedings of the 31st Annual ACM

Symposium on Applied Computing, SAC ’16, pp. 378–383. ACM, New York (2016). https://doi.org/10.1145/2851613.2851619

83. Dai, X., Wang, J.M., Bensaou, B.: Energy-efficient virtual machines scheduling in multi-tenant data centers. IEEE Trans. Cloud Comput. **4**(2), 210–221 (2016). https://doi.org/10.1109/TCC.2015.2481401
84. Elgazzar, K., Martin, P., Hassanein, H.S.: Cloud-assisted computation offloading to support mobile services. IEEE Trans. Cloud Comput. **4**(3), 279–292 (2016). https://doi.org/10.1109/TCC.2014.2350471
85. Espling, D., Larsson, L., Li, W., Tordsson, J., Elmroth, E.: Modeling and placement of cloud services with internal structure. IEEE Trans. Cloud Comput. **4**(4), 429–439 (2016). https://doi.org/10.1109/TCC.2014.2362120
86. Goudarzi, H., Pedram, M.: Hierarchical sla-driven resource management for peak power-aware and energy-efficient operation of a cloud datacenter. IEEE Trans. Cloud Comput. **4**(2), 222–236 (2016). https://doi.org/10.1109/TCC.2015.2474369
87. Huang, Z., Tsang, D.H.K.: M-convex VM consolidation: towards a better VM workload consolidation. IEEE Trans. Cloud Comput. **4**(4), 415–428 (2016). https://doi.org/10.1109/TCC.2014.2369423
88. Kang, D., Choi, G., Kim, S., Hwang, I., Youn, C.: Workload-aware resource management for energy efficient heterogeneous docker containers. In: 2016 IEEE Region 10 Conference (TENCON), pp. 2428–2431 (2016). https://doi.org/10.1109/TENCON.2016.7848467
89. Khatua, S., Sur, P.K., Das, R.K., Mukherjee, N.: Heuristic-based resource reservation strategies for public cloud. IEEE Trans. Cloud Comput. **4**(4), 392–401 (2016). https://doi.org/10.1109/TCC.2014.2369434
90. Mishra, M., Bellur, U.: Whither tightness of packing? The case for stable VM placement. IEEE Trans. Cloud Comput. **4**(4), 481–494 (2016). https://doi.org/10.1109/TCC.2014.2378756
91. Nakagawa, G., Oikawa, S.: Behavior-based memory resource management for container-based virtualization. In: 2016 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science Engineering (ACIT-CSII-BCD), pp. 213–217 (2016). https://doi.org/10.1109/ACIT-CSII-BCD.2016.049
92. Pantazoglou, M., Tzortzakis, G., Delis, A.: Decentralized and energy-efficient workload management in enterprise clouds. IEEE Trans. Cloud Comput. **4**(2), 196–209 (2016). https://doi.org/10.1109/TCC.2015.2464817
93. d R Righi, R., Rodrigues, V.F., da Costa, C.A., Galante, G., de Bona, L.C.E., Ferreto, T.: Autoelastic: automatic resource elasticity for high performance applications in the cloud. IEEE Trans. Cloud Comput. **4**(1), 6–19 (2016). https://doi.org/10.1109/TCC.2015.2424876
94. Salah, K., Elbadawi, K., Boutaba, R.: An analytical model for estimating cloud resources of elastic services. J. Netw. Syst. Manag. **24**(2), 285–308 (2016). https://doi.org/10.1007/s10922-015-9352-x
95. Wajid, U., Cappiello, C., Plebani, P., Pernici, B., Mehandjiev, N., Vitali, M., Gienger, M., Kavoussanakis, K., Margery, D., Perez, D.G., Sampaio, P.: On achieving energy efficiency and reducing CO_2 footprint in cloud computing. IEEE Trans. Cloud Comput. **4**(2), 138–151 (2016). https://doi.org/10.1109/TCC.2015.2453988
96. Wan, J., Zhang, R., Gui, X., Xu, B.: Reactive pricing: an adaptive pricing policy for cloud providers to maximize profit. IEEE Trans. Netw. Serv. Manag. **13**(4), 941–953 (2016). https://doi.org/10.1109/TNSM.2016.2618394
97. Wanis, B., Samaan, N., Karmouch, A.: Efficient modeling and demand allocation for differentiated cloud virtual-network as-a service offerings. IEEE Trans. Cloud Comput. **4**(4), 376–391 (2016). https://doi.org/10.1109/TCC.2015.2389814
98. Wu, H., Ren, S., Garzoglio, G., Timm, S., Bernabeu, G., Chadwick, K., Noh, S.: A reference model for virtual machine launching overhead. IEEE Trans. Cloud Comput. **4**(3), 250–264 (2016). https://doi.org/10.1109/TCC.2014.2369439
99. Xu, X., Dou, W., Zhang, X., Chen, J.: Enreal: An energy-aware resource allocation method for scientific workflow executions in cloud environment. IEEE Trans. Cloud Comput. **4**(2), 166–179 (2016). https://doi.org/10.1109/TCC.2015.2453966
100. Zhou, A., Wang, S., Zheng, Z., Hsu, C., Lyu, M.R., Yang, F.: On cloud service reliability enhancement with optimal resource usage. IEEE Trans. Cloud Comput. **4**(4), 452–466 (2016). https://doi.org/10.1109/TCC.2014.2369421

101. Awada, U., Barker, A.: Improving resource efficiency of container-instance clusters on clouds. In: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), pp. 929–934 (2017). https://doi.org/10.1109/CCGRID.2017.113

102. Awada, U., Barker, A.: Resource efficiency in container-instance clusters. In: Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing, ICC '17, pp. 181:1–181:5. ACM, New York (2017). https://doi.org/10.1145/3018896.3056798

103. Babaioff, M., Mansour, Y., Nisan, N., Noti, G., Curino, C., Ganapathy, N., Menache, I., Reingold, O., Tennenholtz, M., Timnat, E.: Era: A framework for economic resource allocation for the cloud. In: Proceedings of the 26th International Conference on World Wide Web Companion, WWW '17 Companion, pp. 635–642. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2017). https://doi.org/10.1145/3041021.3054186

104. Chard, R., Chard, K., Wolski, R., Madduri, R., Ng, B., Bubendorfer, K., Foster, I.: Cost-aware cloud profiling, prediction, and provisioning as a service. IEEE Cloud Comput. **4**(4), 48–59 (2017). https://doi.org/10.1109/MCC.2017.3791025

105. Dalmazo, B.L., Vilela, J.P., Curado, M.: Performance analysis of network traffic predictors in the cloud. J. Netw. Syst. Manag. **25**(2), 290–320 (2017). https://doi.org/10.1007/s10922-016-9392-x

106. Hai, T.H., Nguyen, P.: A pricing model for sharing cloudlets in mobile cloud computing. In: 2017 International Conference on Advanced Computing and Applications (ACOMP), pp. 149–153 (2017). https://doi.org/10.1109/ACOMP.2017.13

107. Hoque, S., d. Brito, M.S., Willner, A., Keil, O., Magedanz, T.: Towards container orchestration in fog computing infrastructures. In: 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), vol. 2, pp. 294–299 (2017). https://doi.org/10.1109/COMPSAC.2017.248

108. Jin, X., Zhang, F., Wang, L., Hu, S., Zhou, B., Liu, Z.: Joint optimization of operational cost and performance interference in cloud data centers. IEEE Trans. Cloud Comput. **5**(4), 697–711 (2017). https://doi.org/10.1109/TCC.2015.2449839

109. Khasnabish, J.N., Mithani, M.F., Rao, S.: Tier-centric resource allocation in multi-tier cloud systems. IEEE Trans. Cloud Comput. **5**(3), 576–589 (2017). https://doi.org/10.1109/TCC.2015.2424888

110. Li, J., Ma, R., Guan, H., Wei, D.S.L.: Accurate cpu proportional share and predictable i/o responsiveness for virtual machine monitor: a case study in xen. IEEE Trans. Cloud Comput. **5**(4), 604–616 (2017). https://doi.org/10.1109/TCC.2015.2441705

111. Li, J.Z., Woodside, M., Chinneck, J., Litiou, M.: Adaptive cloud deployment using persistence strategies and application awareness. IEEE Trans. Cloud Comput. **5**(2), 277–290 (2017). https://doi.org/10.1109/TCC.2015.2409873

112. Lloyd, W.J., Pallickara, S., David, O., Arabi, M., Wible, T., Ditty, J., Rojas, K.: Demystifying the clouds: harnessing resource utilization models for cost effective infrastructure alternatives. IEEE Trans. Cloud Comput. **5**(4), 667–680 (2017). https://doi.org/10.1109/TCC.2015.2430339

113. Maenhaut, P.J., Moens, H., Volckaert, B., Ongenae, V., Turck, F.D.: A dynamic tenant-defined storage system for efficient resource management in cloud applications. J. Netw. Comput. Appl. **93**, 182–196 (2017). https://doi.org/10.1016/j.jnca.2017.05.014

114. Mebrek, A., Merghem-Boulahia, L., Esseghir, M.: Efficient green solution for a balanced energy consumption and delay in the IOT-fog-cloud computing. In: 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA), pp. 1–4 (2017). https://doi.org/10.1109/NCA.2017.8171359

115. Mechtri, M., Hadji, M., Zeghlache, D.: Exact and heuristic resource mapping algorithms for distributed and hybrid clouds. IEEE Trans. Cloud Comput. **5**(4), 681–696 (2017). https://doi.org/10.1109/TCC.2015.2427192

116. Merzoug, S., Kazar, O., Derdour, M.: Intelligent strategy of allocation resource for cloud datacenter based on MAS CP approach. In: Proceedings of the International Conference on Computing for Engineering and Sciences, ICCES '17, pp. 50–55. ACM, New York (2017). https://doi.org/10.1145/3129186.3129197

117. Mireslami, S., Rakai, L., Far, B.H., Wang, M.: Simultaneous cost and qos optimization for cloud resource allocation. IEEE Trans. Netw. Serv. Manag. **14**(3), 676–689 (2017). https://doi.org/10.1109/TNSM.2017.2738026

118. Nardelli, M., Hochreiner, C., Schulte, S.: Elastic provisioning of virtual machines for container deployment. In: Proceedings of the 8th ACM/SPEC on International Conference on Performance

Engineering Companion, ICPE ’17 Companion, pp. 5–10. ACM, New York (2017). https://doi.org/10.1145/3053600.3053602

119. Nitu, V., Teabe, B., Fopa, L., Tchana, A., Hagimont, D.: Stopgap: Elastic VMS to enhance server consolidation. In: Proceedings of the Symposium on Applied Computing, SAC ’17, pp. 358–363. ACM, New York (2017). https://doi.org/10.1145/3019612.3019626
120. Paya, A., Marinescu, D.C.: Energy-aware load balancing and application scaling for the cloud ecosystem. IEEE Trans. Cloud Comput. **5**(1), 15–27 (2017). https://doi.org/10.1109/TCC.2015.2396059
121. Rankothge, W., Le, F., Russo, A., Lobo, J.: Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms. IEEE Trans. Netw. Serv. Manag. **14**(2), 343–356 (2017). https://doi.org/10.1109/TNSM.2017.2686979
122. Tang, L., Chen, H.: Joint pricing and capacity planning in the IaaS cloud market. IEEE Trans. Cloud Comput. **5**(1), 57–70 (2017). https://doi.org/10.1109/TCC.2014.2372811
123. Xu, D., Liu, X., Niu, Z.: Joint resource provisioning for internet datacenters with diverse and dynamic traffic. IEEE Trans. Cloud Comput. **5**(1), 71–84 (2017). https://doi.org/10.1109/TCC.2014.2382118
124. Yang, Y., Chang, X., Liu, J., Li, L.: Towards robust green virtual cloud data center provisioning. IEEE Trans. Cloud Comput. **5**(2), 168–181 (2017). https://doi.org/10.1109/TCC.2015.2459704
125. Yi, X., Liu, F., Niu, D., Jin, H., Lui, J.C.S.: Cocoa: dynamic container-based group buying strategies for cloud computing. ACM Trans. Model. Perform. Eval. Comput. Syst. **2**(2), 81–831 (2017). https://doi.org/10.1145/3022876
126. Yu, B., Pan, J.: Optimize the server provisioning and request dispatching in distributed memory cache services. IEEE Trans. Cloud Comput. **5**(2), 193–207 (2017). https://doi.org/10.1109/TCC.2015.2469663
127. Zhang, W., Xie, H., Hsu, C.: Automatic memory control of multiple virtual machines on a consolidated server. IEEE Trans. Cloud Comput. **5**(1), 2–14 (2017). https://doi.org/10.1109/TCC.2014.2378794
128. Alam, M., Rufino, J., Ferreira, J., Ahmed, S.H., Shah, N., Chen, Y.: Orchestration of microservices for IOT using docker and edge computing. IEEE Commun. Mag. **56**(9), 118–123 (2018). https://doi.org/10.1109/MCOM.2018.1701233
129. Aral, A., Ovatman, T.: A decentralized replica placement algorithm for edge computing. IEEE Trans. Netw. Serv. Manag. **15**(2), 516–529 (2018). https://doi.org/10.1109/TNSM.2017.2788945
130. Atrey, A., Seghbroeck, G.V., Volckaert, B., Turck, F.D.: Brahma+: a framework for resource scaling of streaming and asap time-varying workflows. IEEE Trans. Netw. Serv. Manag. **15**(3), 894–908 (2018). https://doi.org/10.1109/TNSM.2018.2830311
131. Barkat, A., Kechadi, M.T., Verticale, G., Filippini, I., Capone, A.: Green approach for joint management of geo-distributed data centers and interconnection networks. IEEE Trans. Netw. Serv. Manag. **26**(3), 723–754 (2018). https://doi.org/10.1007/s10922-017-9441-0
132. Balos, C., Vega, D.D.L., Abuelhaj, Z., Kari, C., Mueller, D., Pallipuram, V.K.: A2cloud: An analytical model for application-to-cloud matching to empower scientific computing. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), pp. 548–555 (2018). https://doi.org/10.1109/CLOUD.2018.00076
133. Barrameda, J., Samaan, N.: A novel statistical cost model and an algorithm for efficient application offloading to clouds. IEEE Trans. Cloud Comput. **6**(3), 598–611 (2018). https://doi.org/10.1109/TCC.2015.2513404
134. Borjigin, W., Ota, K., Dong, M.: In broker we trust: a double-auction approach for resource allocation in NFV markets. IEEE Trans. Netw. Serv. Manag. **15**(4), 1322–1333 (2018). https://doi.org/10.1109/TNSM.2018.2882535
135. Bouet, M., Conan, V.: Mobile edge computing resources optimization: a geo-clustering approach. IEEE Trans. Netw. Serv. Manag. **15**(2), 787–796 (2018). https://doi.org/10.1109/TNSM.2018.2816263
136. Cheng, M., Li, J., Nazarian, S.: Drl-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers. In: 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 129–134 (2018). https://doi.org/10.1109/ASPDAC.2018.8297294
137. Diaz-Montes, J., Diaz-Granados, M., Zou, M., Tao, S., Parashar, M.: Supporting data-intensive workflows in software-defined federated multi-clouds. IEEE Trans. Cloud Comput. **6**(1), 250–263 (2018). https://doi.org/10.1109/TCC.2015.2481410

138. Gill, S.S., Buyya, R., Chana, I., Singh, M., Abraham, A.: Bullet: particle swarm optimization based scheduling technique for provisioned cloud resources. J. Netw. Syst. Manag. **26**(2), 361–400 (2018). https://doi.org/10.1007/s10922-017-9419-y
139. Guo, T., Shenoy, P.: Providing geo-elasticity in geographically distributed clouds. ACM Trans. Internet Technol. **18**(3), 38:1–38:27 (2018). https://doi.org/10.1145/3169794
140. Guo, W., Lin, B., Chen, G., Chen, Y., Liang, F.: Cost-driven scheduling for deadline-based workflow across multiple clouds. IEEE Trans. Netw. Serv. Manag. **15**(4), 1571–1585 (2018). https://doi.org/10.1109/TNSM.2018.2872066
141. Guo, Y., Stolyar, A.L., Walid, A.: Shadow-routing based dynamic algorithms for virtual machine placement in a network cloud. IEEE Trans. Cloud Comput. **6**(1), 209–220 (2018). https://doi.org/10.1109/TCC.2015.2464795
142. Hauser, C.B., Wesner, S.: Reviewing cloud monitoring: towards cloud resource profiling. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), pp. 678–685 (2018). https://doi.org/10.1109/CLOUD.2018.00093
143. Heidari, S., Buyya, R.: Cost-efficient and network-aware dynamic repartitioning-based algorithms for scheduling large-scale graphs in cloud computing environments. Softw. Pract. Exp. **48**(12), 2174–2192 (2018). https://doi.org/10.1002/spe.2623
144. Jia, B., Hu, H., Zeng, Y., Xu, T., Yang, Y.: Double-matching resource allocation strategy in fog computing networks based on cost efficiency. J. Commun. Netw. **20**(3), 237–246 (2018). https://doi.org/10.1109/JCN.2018.000036
145. Jia, G., Han, G., Jiang, J., Chan, S., Liu, Y.: Dynamic cloud resource management for efficient media applications in mobile computing environments. Pers. Ubiquitous Comput. **22**(3), 561–573 (2018). https://doi.org/10.1007/s00779-018-1118-5
146. Khabbaz, M., Assi, C.M.: Modelling and analysis of a novel deadline-aware scheduling scheme for cloud computing data centers. IEEE Trans. Cloud Comput. **6**(1), 141–155 (2018). https://doi.org/10.1109/TCC.2015.2481429
147. Lahmann, G., McCann, T., Lloyd, W.: Container memory allocation discrepancies: an investigation on memory utilization gaps for container-based application deployments. In: 2018 IEEE International Conference on Cloud Engineering (IC2E), pp. 404–405 (2018). https://doi.org/10.1109/IC2E.2018.00076
148. Lin, Y., Lai, Y., Huang, J., Chien, H.: Three-tier capacity and traffic allocation for core, edges, and devices for mobile edge computing. IEEE Trans. Netw. Serv. Manag. **15**(3), 923–933 (2018). https://doi.org/10.1109/TNSM.2018.2852643
149. Nawrocki, P., Sniezynski, B.: Adaptive service management in mobile cloud computing by means of supervised and reinforcement learning. J. Netw. Syst. Manag. **26**(1), 1–22 (2018). https://doi.org/10.1007/s10922-017-9405-4
150. Prats, D.B., Berral, J.L., Carrera, D.: Automatic generation of workload profiles using unsupervised learning pipelines. IEEE Trans. Netw. Serv. Manag. **15**(1), 142–155 (2018). https://doi.org/10.1109/TNSM.2017.2786047
151. Rahimi, M.R., Venkatasubramanian, N., Mehrotra, S., Vasilakos, A.V.: On optimal and fair service allocation in mobile cloud computing. IEEE Trans. Cloud Comput. **6**(3), 815–828 (2018). https://doi.org/10.1109/TCC.2015.2511729
152. Sahni, J., Vidyarthi, D.P.: A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment. IEEE Trans. Cloud Comput. **6**(1), 2–18 (2018). https://doi.org/10.1109/TCC.2015.2451649
153. Scheuner, J., Leitner, P.: Estimating cloud application performance based on micro-benchmark profiling. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), pp. 90–97 (2018). https://doi.org/10.1109/CLOUD.2018.00019
154. Simonis, I.: Container-based architecture to optimize the integration of microservices into cloud-based data-intensive application scenarios. In: Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings, ECSA '18, pp. 34:1–34:3. ACM, New York (2018). https://doi.org/10.1145/3241403.3241439
155. Sathya Sofia, A., GaneshKumar, P.: Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II. J. Netw. Syst. Manag. **26**(2), 463–485 (2018). https://doi.org/10.1007/s10922-017-9425-0
156. Takahashi, K., Aida, K., Tanjo, T., Sun, J.: A portable load balancer for kubernetes cluster. In: Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region, HPC Asia 2018, pp. 222–231. ACM, New York (2018). https://doi.org/10.1145/3149457.3149473

157. Trihinas, D., Pallis, G., Dikaiakos, M.D.: Monitoring elastically adaptive multi-cloud services. IEEE Trans. Cloud Comput. **6**(3), 800–814 (2018). https://doi.org/10.1109/TCC.2015.2511760
158. Wang, L., Gelenbe, E.: Adaptive dispatching of tasks in the cloud. IEEE Trans. Cloud Comput. **6**(1), 33–45 (2018). https://doi.org/10.1109/TCC.2015.2474406
159. Wei, L., Foh, C.H., He, B., Cai, J.: Towards efficient resource allocation for heterogeneous workloads in iaas clouds. IEEE Trans. Cloud Comput. **6**(1), 264–275 (2018). https://doi.org/10.1109/TCC.2015.2481400
160. Xie, R., Jia, X.: Data transfer scheduling for maximizing throughput of big-data computing in cloud systems. IEEE Trans. Cloud Comput. **6**(1), 87–98 (2018). https://doi.org/10.1109/TCC.2015.2464808
161. Zhang, W., Wen, Y.: Energy-efficient task execution for application as a general topology in mobile cloud computing. IEEE Trans. Cloud Comput. **6**(3), 708–719 (2018). https://doi.org/10.1109/TCC.2015.2511727
162. Zhang, Y., Ghosh, A., Aggarwal, V., Lan, T.: Tiered cloud storage via two-stage, latency-aware bidding. IEEE Trans. Netw. Serv. Manag. (2018). https://doi.org/10.1109/TNSM.2018.2875475
163. Introducing Amazon EC2 spot instances for specific duration workloads. https://aws.amazon.com/about-aws/whats-new/2015/10/introducing-amazon-ec2-spot-instances-for-specific-duration-workloads/. Accessed 9 Sept 2019
164. Juju solutions for container management. https://jaas.ai/containers. Accessed 9 Sept 2019
165. Masip-Bruin, X., Marín-Tordera, E., Juan-Ferrer, A., Queralt, A., Jukan, A., Garcia, J., Lezzi, D., Jensen, J., Cordeiro, C., Leckey, A., Salis, A., Guilhot, D., Cankar, M.: mf2c: towards a coordinated management of the IOT-fog-cloud continuum. In: Proceedings of the 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects, SMARTOBJECTS ’18, pp. 8:1–8:8. ACM, New York (2018). https://doi.org/10.1145/3213299.3213307
166. Almutairi, A., Sarfraz, M.I., Ghafoor, A.: Risk-aware management of virtual resources in access controlled service-oriented cloud datacenters. IEEE Trans. Cloud Comput. **6**(1), 168–181 (2018). https://doi.org/10.1109/TCC.2015.2453981
167. Zhai, Y., Yin, L., Chase, J., Ristenpart, T., Swift, M.: CQSTR: Securing cross-tenant applications with cloud containers. In: Proceedings of the Seventh ACM Symposium on Cloud Computing, SoCC ’16, pp. 223–236. ACM, New York (2016). https://doi.org/10.1145/2987550.2987558
168. Lins, S., Schneider, S., Sunyaev, A.: Trust is good, control is better: creating secure clouds by continuous auditing. IEEE Trans. Cloud Comput. **6**(3), 890–903 (2018). https://doi.org/10.1109/TCC.2016.2522411
169. Maenhaut, P.J., Volckaert, B., Ongenae, V., De Turck, F.: Efficient resource management in the cloud: from simulation to experimental validation using a low-cost raspberry pi testbed. Softw. Pract. Exp. **49**(3), 449–477 (2019). https://doi.org/10.1002/spe.2669
170. Calheiros, R., Ranjan, R., Beloglazov, A., De Rose, C., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. **41**(1), 23–50 (2011). https://doi.org/10.1002/spe.995
171. Fed4fire+—federation for fire plus. https://www.fed4fire.eu/. Accessed 9 Sept 2019
172. FUTEBOL Brazil/UFRGS. http://futebol.inf.ufrgs.br/. Accessed 9 Sept 2019
173. Eivy, A.: Be wary of the economics of “serverless” cloud computing. IEEE Cloud Comput. **4**(2), 6–12 (2017). https://doi.org/10.1109/MCC.2017.32

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Pieter-Jan Maenhaut received his Master of Science degree in Information Engineering Technology from the University College Ghent, Belgium in June 2009, and his Master of Science degree in Computer Science Engineering from the University of Ghent in July 2014. In February 2012 he joined the Internet and Data Science Lab (IDLab, former IBCN), a research group in the Department of Information Technology (INTEC). In July 2019, he received his PhD on efficient resource management in a multi-tenant cloud environment. His research interests include efficient resource management in cloud environments, both from the point of view of applications deployed within a cloud environment and from the point of view of the underlying physical infrastructure.

Bruno Volckaert is professor advanced programming and software engineering within the department of Information Technology (INTEC) at Ghent University, and senior researcher at imec. He obtained his Master of Computer Science degree in 2001 from Ghent University, after which he worked on his PhD at Ghent University on resource management for grid computing, which he obtained in 2006. His current research deals with reliable and high performance distributed software systems for a.o. Smart Cities, distributed decision support systems, scalable cybersecurity detection and mitigation architectures and autonomous optimization of cloud-based applications. He has worked on over 45 national and international research projects and is author or co-author of more than 125 peer-reviewed papers published in international journals and conference proceedings.

Veerle Ongenae received a masters degree in Mathematics (1992) from Ghent University. Her research on left symmetric algebras resulted in a PhD in 1998. In 2003 she became a docent at the Ghent University College and in 2013 at the University Ghent. She is a member of the Broadband Communication Networks research group of the Department of Information Technology of the Faculty of Engineering. Her main research focus is distributed software. She is responsible for various courses on programming, software development, web technologies, data technologies and distributed applications.

Filip De Turck leads the network and service management research group at the Department of Information Technology of Ghent University, Belgium and imec. He (co-) authored over 500 peer-reviewed papers and his research interests include telecommunication network and service management, and design of efficient virtualized network systems. In this research area, he is involved in several research projects with industry and academia, serves as Chair of the IEEE Technical Committee on Network Operations and Management (CNOM), and is on the TPC of many network and service management conferences and workshops. He also serves as Editor in Chief of IEEE Transactions of Network and Service Management (TNSM), and is steering committee member of the IEEE Conference on Network Softwarization (IEEE NetSoft).