Computing
DOI 10.1007/s00607-014-0398-5

An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art

Khalid Alhamazani · Rajiv Ranjan · Karan Mitra · Fethi Rabhi · Prem Prakash Jayaraman · Samee Ullah Khan · Adnene Guabtni · Vasudha Bhatnagar

Received: 29 June 2013 / Accepted: 20 March 2014

Abstract Cloud monitoring activity involves dynamically tracking the Quality of Service (QoS) parameters related to virtualized resources (e.g., VM, storage, network, appliances, etc.), the physical resources they share, the applications running on them and data hosted on them. Applications and resources configuration in cloud computing environment is quite challenging considering a large number of heterogeneous cloud resources. Further, considering the fact that at given point of time, there may be need to change cloud resource configuration (number of VMs, types of VMs, number of appliance instances, etc.) for meet application QoS requirements under uncertainties (resource failure, resource overload, workload spike, etc.). Hence, cloud monitoring tools can assist a cloud providers or application developers in: (i) keeping their resources and applications operating at peak efficiency, (ii) detecting variations in resource and application performance, (iii) accounting the service level agreement violations of certain QoS parameters, and (iv) tracking the leave and join operations

K. Alhamazani · F. Rabhi
University of New South Wales, Sydney, Australia

R. Ranjan (✉) · P. P. Jayaraman
CSIRO, Canberra, Australia
e-mail: rajiv.ranjan@csiro.au

K. Mitra
Luleå University of Technology, Luleå, Sweden

S. U. Khan
North Dakota State University, Fargo, USA

A. Guabtni
NICTA, Sydney, Australia

V. Bhatnagar
University of Delhi, New Delhi, India

Published online: 16 April 2014

of cloud resources due to failures and other dynamic configuration changes. In this paper, we identify and discuss the major research dimensions and design issues related to engineering cloud monitoring tools. We further discuss how the aforementioned research dimensions and design issues are handled by current academic research as well as by commercial monitoring tools.

Keywords Cloud monitoring · Cloud application monitoring · Cloud resource monitoring · Cloud application provisioning · Cloud monitoring metrics · Quality of service parameters · Service level agreement

Mathematics Subject Classification 68U01

1 Introduction

According to National Institute of Standards and Technology NIST,[1] cloud computing is a " Model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (network, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [1]. *Service models*, *hosting*, *deployment models*, and *roles* are some of the important concepts and essential characteristics related to cloud computing technologies defined by NIST and elaborated in [1–6], Commercial cloud providers including Amazon Web Services (AWS), Microsoft Azure, Salesforce.com, Google App Engine and others offer the cloud consumers options to deploy their applications over a network of infinite resource pool with practically no capital investment and with modest operating cost proportional to the actual use. For example, Amazon EC2 cloud runs around half million physical hosts, each of them hosting multiple virtual machines that can be dynamically invoked or removed [7].

Several papers in the literature discuss, explore and propose surveys of cloud monitoring in different aspects [1–6, 8–13]. To the best of our knowledge, no specific survey considers monitoring applications across the different cloud layers namely Infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS). Further, none of the papers have focused on predictive cloud monitoring. In addition to that, none of the paper discusses the possibility of utilizing machine learning techniques with monitored data. In addition to the above factors, one arising aspect with the cloud computing is managing huge volume of data (Big Data). In the present environment, the term "Big Data" is described as a phenomenon that refers to the practice of collection and processing of very large datasets and the associated systems and algorithms used to analyze these enormous data sets [14]. Three well recognized characteristics of Big Data are Variety, Volume and Velocity (3 V's) of data generation [15, 16]. The steady growth of social media and smart mobile devices has led to an increase in the sources of outbound traffic, initiating "data tsunami phenomenon". For example, in [17–19], Wang et al. present a high-performance data-intensive computing solution for is massive remote sensing data processing. This poses significant challenges in cloud computing.

[1] http://www.nist.gov/itl/cloud/.

In [20], studies show that as more people join social media sites hosted on clouds, analysis of the data become more difficult and almost impossible to be analyzed. Another aspect of big data events can occur when e.g. services or processes of the infrastructure itself cause high load [21]. Other study [22,23] shows that VMs migrating, copy and saving current state can affect the performance of data transfer within the cloud. Moreover, different types of data originating from mobile devices makes understanding of composite data a challenging problem due to multi-modality, huge volume, dynamic nature, multiple sources, and unpredictable quality. Continuously monitoring of multi-modal data streams collected from heterogeneous data sources require monitoring tools that can cope with managing big data floods (data tsunami phenomenon).

In this paper, we identify and discuss three challenges of cloud monitoring: (1) How to determine layer specific application monitoring requirements i.e., how cloud consumer can stipulate at what cloud layer his/her running application should be monitored. (2) How cloud consumer can express what information he/she is interesting in to gain knowledge while his/her application is being monitored. (3) How cloud consumer can predict the application behavior in terms of performance.

1.1 Our contributions

The concrete contributions made by this paper are: (a) advancing the fundamental understanding of cloud resource and application provisioning and monitoring concepts, (b) identification of the main research dimensions and software engineering issues based on cloud resource and application types, and (c) presents future research directions for novel cloud monitoring techniques.

The remainder of the paper is organized as follows: Discussion on key cloud resource provisioning is presented in Sect. 2. Section 3 discusses the cloud application life cycle and in detail discusses the components of cloud monitoring. Section 4 present details on major research dimensions and software engineering issues related to developing cloud monitoring tools are. Section 5, discusses mapping of research dimensions to existing cloud monitoring tools. The paper ends with brief conclusion and overview of future work.

2 Cloud resource provisioning

Cloud resource provisioning is a complex task [24] and is referred to as the process of application deployment and management on cloud infrastructure. Current cloud providers such as Amazon EC2, ElasticHosts, GoGrid, TerraMark, and Flexian, do not completely offer support for automatic deployment and configuration of software resources [25]. Therefore, several companies, e.g. RightScale and Scalr provide scalable managed services on top of cloud infrastructures, to cloud consumers for supporting automatic application deployment and configuration control [25]. The three main steps for cloud provisioning are [24,26]:

Virtual machine provisioning where suitable VMs are instantiated to match the required hardware and configuration of an application. To illustrate, Bitnami[2] supports con-

[2] http://bitnami.org/faq/cloud_amazon_ec2.

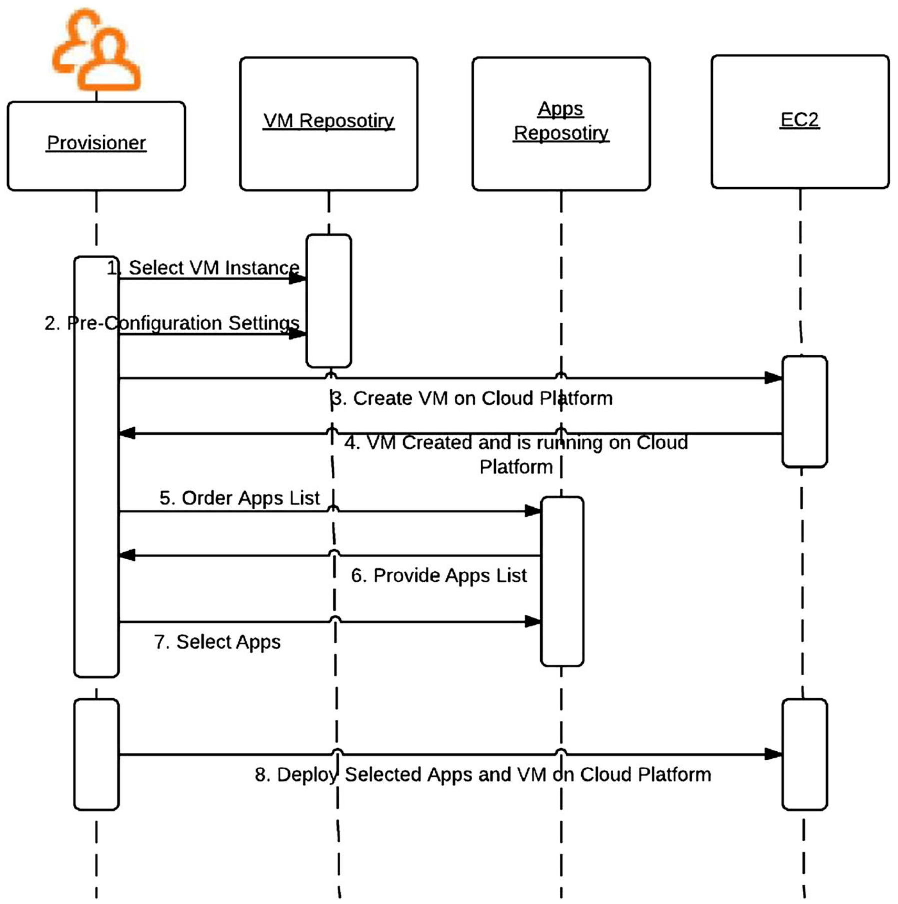

Provisioner
VM Reposotiry
Apps Reposotiry
EC2
1. Select VM Instance
2. Pre-Configuration Settings
3. Create VM on Cloud Platform
4. VM Created and is running on Cloud Platform
5. Order Apps List
6. Provide Apps List
7. Select Apps
8. Deploy Selected Apps and VM on Cloud Platform

Fig. 1 Provisioning and deployment sequence diagram

sumers to provision a Bitnami stack that consists of VM and appliances. On the other hand, Amazon EC2 consumers may firstly provision a VM on the cloud then choose the appliances to provision on that VM.

Resource provisioning it is the process of mapping and scheduling the instantiated VMs onto the cloud's physical servers. This is handled by cloud-based hypervisors. For example, public clouds expose APIs to start/stop a resource but not to control which physical server within that region/datacenter will host the VM. Figure 1, illustrates the steps where a cloud consumer attempts to provision cloud resources on Amazon EC2 platform. In step 1, from the VM repository, a consumer views the available VMs provided by the cloud platform and selects the preferable VM instance type. In step 2, the consumer sets up his/her preferences/configurations on this VM. In steps 3 and 4, the user deploys this VM on the cloud platform successfully. Subsequently, in steps 5 and 6, the consumer retrieves back a list of available applications from the applications repository. In step 7, the consumer simply opts for his/her desired applications that he/she would like to provision. Finally, in step 8, the cloud consumer deploys the applications and the VM on the cloud platform.

Application provisioning is the process of application deployment on VMs on the cloud infrastructure. For example, deploying a Tomcat server as an application on a VM hosted on the Amazon EC2 cloud. Applications provisioning can be done in two ways. The first method consists of deploying the applications together while hosting a VM. In the second method, the consumer may want to first deploy the VM, and then as a separate step, he/she may deploy the applications.

After the provisioning stage, a cloud workflow instance might be composed of multiple cloud services, and in some cases services from a number of different service providers. Therefore, monitoring the quality of cloud instances across cloud providers become much more complex [27]. Further, at run time, QoS of the running instance needs to be consistently monitored to guarantee SLA and avoid/handle abnormal system behaviour. Monitoring is the process of observing and tracking applications/resources at run time. It is the basis of control operations and corrective actions for running systems on clouds. Despite the existence of many commercial monitoring tools in the market, managing service level agreements (SLAs) between multiple cloud providers still pose a major issue in clouds.

In some way, cloud monitoring, SLA and dynamic configuration are correlated in the sense that one has an impact on another. In other words, enhancing monitoring functionalities will in turn assist meeting SLAs as well as improving dynamic configuration operations at run time. Moreover, SLA has to be met by the cloud providers in order to reach the required reliability level required by consumers. Also, auto-scaling and dynamic configurations are required for optimal use of cloud technology. This all-together leads us to conclude that cloud monitoring is a key element that has to be further studied and enhanced.

3 Cloud monitoring

Under this section, we present the basic components, phases and layers of application architecture on clouds. Also, this section will present the state of the art in cloud monitoring as well as how it is conceptually correlated to QoS and SLA.

3.1 Application life cycle

The application architecture determines how, when, and which provisioning operations should be processed and applied on cloud resources. The high level application (e.g., multimedia applications) architecture is multi-layered [28]. These layers may consist of clients/application consumers, load balancers, web servers, streaming servers, application servers, and a database system. Notably, each layer may instantiate multiple software resources as needed and when required. Such multiple instantiations can be allocated to one or more hardware resources. Technically, across those aforementioned system layers, a number of provisioning operations take place at design time as well as run-time. These provisioning operations should ensure SLA compliance by achieving the QoS targets.

Resource selection It is the process where the system developer selects software (web server, multimedia server, database server, etc.) and hardware resources (CPU, storage, and network). This process encapsulates the allocation of hardware resources to those selected software resources.

Resource deployment During this process, system administrator instantiates the selected software resources on the hardware resources, as well as configuring these resources for successful communication and inter-operation with the other software resources already running in the system.

Resource monitoring In order to ensure that the deployed software and hardware resources run at the required level to satisfy the SLA, a continuous resource monitoring process is desirable. This process involves detecting and gathering information about the running resources. In case of the detection of any abnormal system behavior, the system orchestrator is notified for policy-based corrective actions to be undertaken as a system remedy.

Resource control Is the process to ensure meeting the QoS terms stated in the SLA. This process is responsible for handling system uncertainties at run time e.g. upgrade or downgrade a resource type, capacity or functionality.

3.2 Cloud monitoring

In clouds, monitoring is essential to maintain high system availability and performance of the system and is important for both providers and consumers [8–10]. Primarily, monitoring is a key tool for (i) managing software and hardware resources, and (ii) providing continuous information for those resources as well as for consumer hosted applications on the cloud. Cloud activities like resource planning, resource management, data center management, SLA management, billing, troubleshooting, performance management, and security management essentially need monitoring for effective and smooth operations of the system [29]. Consequently, there is a strong need for monitoring looking at the elastic nature of cloud computing [30].

In cloud computing, monitoring can be of two types: high-level and low-level. High-level monitoring is related to the virtual platform status [31]. The low-level monitoring is related to information collected about the status of the physical infrastructure [31, 32]. Cloud monitoring system is a self-adjusting and typically multi-threaded system that is able to support monitoring functionalities [33]. It comprehensively monitors pre-identified instances/resources on the cloud for abnormalities. On detecting an abnormal behavior, the monitoring system attempts to auto-repair this instance/resource if the corresponding monitor has a tagged auto-heal action [33]. In case of auto-repair failure or an absence of an auto-heal action, a support team is notified. Technically, notifications can be sent by different means such as email, or SMS [33].

3.2.1 Monitoring, QoS, and SLA

As mentioned earlier, cloud monitoring is needed for continuous assessment of resources or applications on cloud platform in terms of performance, reliability, power usage, ability to meet SLA, security, etc [34]. Fundamentally, monitoring tests can be computation based and/or network based. Computation based tests are concerned about the status of the real or virtualized platforms running cloud applications. Data metrics considered in such tests include CPU speed, CPU utilization, disk throughput, VM acquisition/release time and system up-time. Network based tests focus on network layer data related metrics like jitter, round-trip time RTT, packets loss, traffic volume etc [31,32,35].

At run-time, a set of operations take place in order to meet the QoS parameters specified in SLA document that guarantees the required performance objectives of the cloud consumers. The availability, load, and throughput of hardware resources can vary in unpredictable ways, so ensuring that applications achieve QoS targets is not trivial. Being aware of the system's current software and hardware service status is imperative for handling such uncertainties to ensure the fulfillment of QoS targets [8]. In addition, detecting exceptions and malfunctions while deploying software services on hardware resources is essential e.g., showing QoS delivered by each application component (software service such as web server or database server) hosted on each hardware resource. Uncertainties can be tackled through the development of efficient, scalable, interoperable monitoring tools with easy-to-use interfaces.

3.2.2 Monitoring across different applications and layers

As mentioned previously, application components (e.g., streaming server, web server, indexing server, compute service, storage service, and network) are distributed across cloud layers including PaaS and IaaS. Thus, in order to guarantee the achievement of QoS targets for the application as a whole, monitoring QoS parameters should be performed across all the layers of cloud stack including Platform-as-a-Service (PaaS) (e.g., web server, streaming server, indexing server, etc.) and Infrastructure-as-a-Service (IaaS) (e.g., compute services, storage services, and network). Figure 2 illustrates how different components in a cloud platform are distributed across the cloud platform layers. Table 1 shows the QoS parameters that a monitoring system should consider at each cloud layer.

Typically, QoS targets vary across application types. For example, QoS targets for eResearch applications are different from static, single-tier web applications (e.g., web site serving static contents) or multi-tier applications (e.g., on demand audio/video streaming). Based on application types, there is always a need to negotiate different SLAs. Hence, SLA document includes conditions and constraints that match the nature of QoS requirements with each application type. For example, a genome analysis experiment on cloud services will only care about data transfer (upload and download) network latency and processing latency. On the other hand, for multimedia applications, the quality of the transferred data over network is more important. Hence, other parameters gain priority in this case. Failing to track QoS parameters will eventually lead to SLA violations. Consequently, monitoring is fundamental and responsible for SLAs compliance certification [36]. Moreover, a multi-layer application monitoring approach can provide significant insights into the application performance and system performance to both the consumer and cloud administrator. This is essential for consumers as they can identify and isolate application performance bottlenecks to

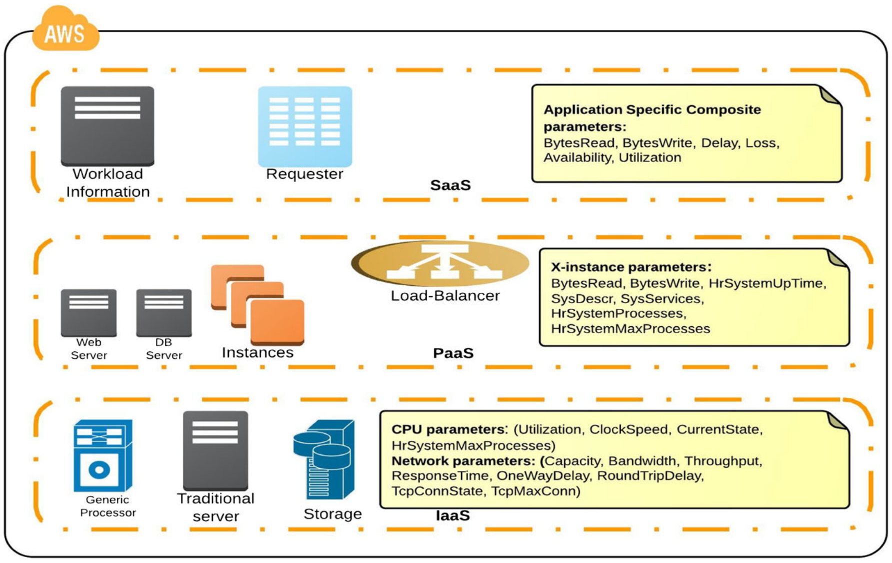

AWS
Workload Information
Requester
SaaS
Application Specific Composite parameters:
BytesRead, BytesWrite, Delay, Loss, Availability, Utilization
Web Server
DB Server
Instances
Load-Balancer
PaaS
X-instance parameters:
BytesRead, BytesWrite, HrSystemUpTime, SysDescr, SysServices, HrSystemProcesses, HrSystemMaxProcesses
Generic Processor
Traditional server
Storage
IaaS
CPU parameters: (Utilization, ClockSpeed, CurrentState, HrSystemMaxProcesses)
Network parameters: (Capacity, Bandwidth, Throughput, ResponseTime, OneWayDelay, RoundTripDelay, TcpConnState, TcpMaxConn)

Fig. 2 Components across cloud platform layers

Table 1 QoS parameters at each cloud platform layer

Cloud layer	Layer components	Targeted QoS parameters
SaaS	Appliances x,y,z, etc	BytesRead, BytesWrite, Delay, Loss, Availability, Utilization
PaaS	Web Server, Streaming Server, Index Server, Apps Server, etc	BytesRead, BytesWrite, SysUpTime, SysDesc, HrSystemMaxProcesses, HrSystemProcesses, SysServices
IaaS	Compute Service, Storage Service, Network, etc	CPU parameters: Utilization, ClockSpeed, CurrentState; network parameters: Capacity, Bandwidth, Throughput, ResponseTime, OneWayDelay, RoundTripDelay, TcpConnState, TcpMaxConn

specific layers. From a cloud administrator point-of-view, the QoS statistics on application performance across layer can help them maintain their SLAs delivering better performance and higher consumer satisfaction.

4 Evaluation dimensions

Under this section, we present the basic components that can be considered as evaluation dimensions in order to evaluate a monitoring tool in cloud computing.

4.1 Monitoring architectures

In cloud monitoring, the network and system related information is collected by the systems. For example, CPU utilization, network delay and packet losses. This informa-

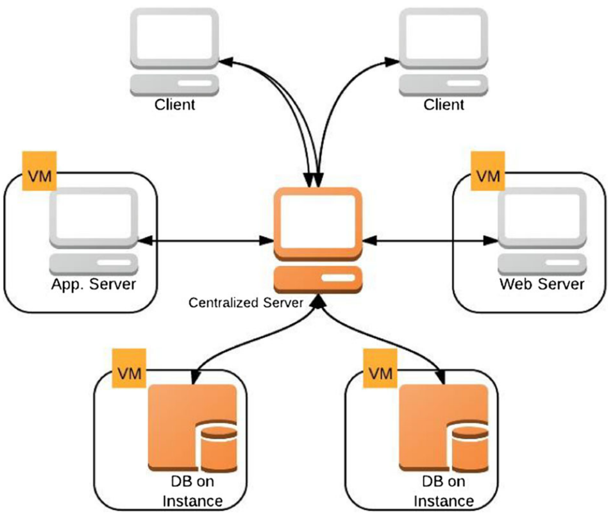


Fig. 3 Centralized monitoring architecture

tion is then used by the applications to determine actions such as data migration to the server closest to the user to ensure that SLA requirements are met. Typically, network monitoring can be performed on centralized and de-centralized network architectures.

4.1.1 Centralized

In centralized architecture shown in Fig. 3, the PaaS and IaaS resources send QoS status update queries to the centralized monitoring server. In this scheme, the monitoring techniques continuously pull the information from the components via periodic probing messages. In [33], the authors show that a centralized cloud monitoring architecture allows better management for cloud applications. Nevertheless, centralized approach has several design issues, including:

- Prone to a single point of failure;
- Lack of scalability;
- High network communication cost at links leading to the information server (i.e., network bottleneck, congestion); and
- Possible lack of the required computational power to serve a large number of monitoring requests.

4.1.2 Decentralized

Recently, proposals for decentralized cloud monitoring tools have gained momentum. Figure 4 shows the broad schematic design of decentralized cloud monitoring system. The decentralization of monitoring tools can overcome the issues related to current centralized systems. A monitoring tool configuration is considered as decentralized if none of the components in the system is more important than others. In case one of

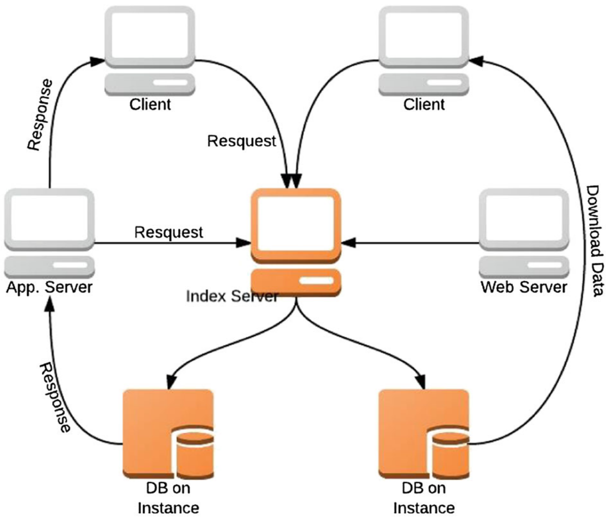


Fig. 4 Decentralized monitoring architecture

the components fails, it does not influence the operations of any other component in the system.

Structured peer-to-peer Looking forward to have a network layout where a central authority is defused has lead to the development of the structured peer-to-peer networks. In such a network overlay, central point of failure is eliminated. Napster is a popular structured peer-to-peer system [37].

Unstructured peer-to-peer Unstructured peer-to-peer networks overlay is meant to be a distributed overlay but the difference is that the search directory is not centralized unlike structured peer-to-peer networks overlay which, leads to absolute single point failure in such network overlay. Gnutella is one of the well-known unstructured peer-to-peer systems [37].

Hybrid peer-to-peer Is a combination of structured and unstructured peer-to-peer networks systems. Super peers can act as local search hubs in small portions of the network whereas the general scope of the network behaves as unstructured peer-to-peer system. Kazaa is a hybrid of centralized Napster and decentralized Gnutella network systems.

4.2 Interoperability

The interoperability perspective in technology focuses on the system's technical capabilities to interface between organizations and systems. It also focuses on the resulting mission of compatibility or incompatibility between systems and data collation partners. Modern business applications developed on cloud are often complicated and require interoperability. For example, an application owner can deploy a web server on Amazon Cloud while the database server may be hosted in Azure Cloud. Unless data and applications are not integrated across clouds properly, the results and benefits

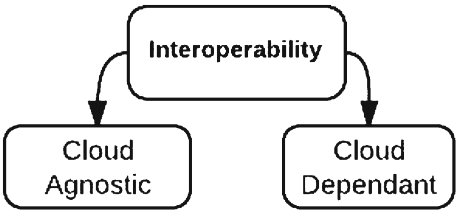


Fig. 5 Interoperability classification

Table 2 Monitoring tools and interoperability

Platform	Interoperability, Cloud-Agnostic (Multi-Clouds)
Monitis [38]	Yes
RevealCloud [39,40]	Yes
LogicMonitor [41]	Yes
Nimsoft [42]	Yes
Nagios [31,43]	Yes
SPAE [44,45]	Yes
CloudWatch [46]	No
OpenNebula [47]	No
CloudHarmony [48]	Yes
Azure FC [49,50]	No

of cloud adoption cannot be achieved. Interoperability is also necessary to avoid cloud provider lock-in.

This dimension refers to the ability of a cloud monitoring framework to monitor applications and its components that may be deployed across multiple cloud providers. While it is not difficult to implement a cloud-specific monitoring framework, to design generic cloud monitoring framework that can work with multiple cloud providers remain a challenging problem. Next, we classify the interoperability (Fig. 5) of monitoring frameworks into the following categories:

Cloud dependent Currently many public cloud providers provide their consumers monitoring tools to monitor their application's CPU, storage and network usage. Often these tools are tightly integrated with the cloud providers existing tools. For example, CloudWatch, offered by Amazon is a monitoring tool that enables consumers to manage and monitors their applications residing on AWS EC2 (CPU) services. But, this monitoring tool does not have the ability to monitor an application component that may reside on other cloud provider's infrastructure such as GoGrid and Azure. Table 2 illustrates some examples of cloud monitoring tools that are specific to a cloud provider as well as Cloud Agnostic.

Cloud Agnostic In contrast to single cloud monitoring, engineering cloud agnostic monitoring tools is challenging. This is primarily due to fact that there is not a common unified application programming interface (API) for calling on cloud computing services' runtime QoS statistics. Though recent developments in cloud programming API including Simple Cloud, Delta Cloud, JCloud, and Dasein Cloud simplify interaction of services (CPU, storage, and network) that may belong to multiple clouds, they have limited or no ability to monitor their run-time QoS statistics and application behaviors. In this scenario, monitoring tools are expected to be able to retrieve QoS data of services and applications that may be part of multiple clouds. Cloud agnostic monitoring tools are also required if one wants to realize a hybrid cloud architecture involving services from private and public clouds. Monitis monitoring tool provides the ability of accessing different clouds e.g. Amazon EC2, Rackspace and GoGrid. It utilizes the concept of widgets where consumers can view more than one widget in a page. In Monitis [38], consumers need to provide only cloud account credentials to access monitoring data of their cloud applications running on different cloud provider infrastructure. They can also specify which instance to monitor. Hence, a consumer can view two different cloud instances using two different widgets in one single page.

4.3 Quality of service (QoS) matrix

It is non-trivial for application developers to understand what QoS parameters and targets he/she needs to specify and monitor across each layer of a cloud stack including PaaS (e.g., web server, streaming server, indexing server, etc.) and IaaS (e.g., compute services, storage services, and network). As shown in Fig. 6, this can be by one parameter or a group of parameters.

4.3.1 Single parameter

In this scenario, a single parameter refers to a specific system QoS target. In each system, there are major atomic/single values that have to be tracked closely and continuously. For example, CPU utilization is basically expressed by only one single parameter in the SLA. Such parameters can affect the whole system and a violation in SLA can lead to a serious system failure. Unlike composite parameters where a single parameter might not be of priority to the system administrator, single parameters in most cases gain high priority when monitoring SLA violations and QoS targets.

4.3.2 Composite parameters

In a composite parameter scenario, a group of different parameters are taken into consideration. In the cloud, software application is composed of many cloud software services. Thus, the performance quality can be determined by collective behaviors of those software services [27]. After observing multiple parameters for estimating a functionality of one or more concerned processes, one result could be obtained to

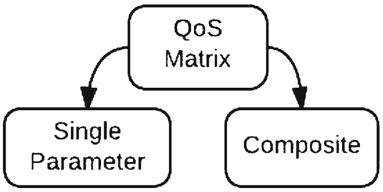


Fig. 6 QoS matrix classification

Table 3 Monitoring tools and layers' visibility

Platform	Visibility multi-layers (composite QoS matrix)
Monitis [38]	Yes
RevealCloud [39,40]	Yes
LogicMonitor [41]	Yes
Nimsoft [42]	Yes
Nagios [31,43]	Yes
SPAE [44,45]	No
CloudWatch [46]	Yes
OpenNebula [47]	No
CloudHarmony [48]	No
Azure FC [49,50]	Yes

evaluate the QoS. To illustrate, "loss" can be considered as a composite parameter of two single parameters "one way loss" and "round trip loss". Similarly, "delay" can be considered as composite parameters of three single parameters "one way delay", "RTT delay", and "delay variance". Table 3 shows a list of some commercial tools for cloud monitoring and it illustrates which of them support or do not support monitoring multiple QoS parameters.

4.4 Cross-layer monitoring

As shown in Fig. 7, application components (streaming server, web server, indexing server, compute service, storage service, and network) related to a multimedia streaming application are distributed across cloud layers including PaaS and IaaS. In order to guarantee the achievement of QoS targets for the multimedia application as a whole, it is critical to monitor QoS parameters across multiple layers [51]. Hence, the challenge here is to develop monitoring tools that can capture and reason about the QoS parameters of application components across IaaS and PaaS layers. As demonstrated in Fig. 8, we categorize the visibility of commercial monitoring tools into following categories:

Layer specific Cloud services are distributed among three layers namely, SaaS, PaaS, and IaaS. Monitoring tools originally are oriented to perform monitoring tasks over services only in one of the aforementioned layers. Most of present day commercial tools are designed to keep track of the performance of resources provisioned at the IaaS layer. For example, CloudWatch is not capable of monitoring information related to load, availability, and throughout of each core of CPU services and its effect on the QoS (e.g., latency, availability, etc.) delivered by the hosted PaaS services (e.g., J2EE application server). Hence, there exists a considerable gap and research challenges in developing a monitoring tool that can monitor QoS statistics across multiple layers of the cloud stack.

Layer Agnostic In contrast to the previous scenario, monitoring at multiple layers enables the consumers to gain insights to applications' performance across multiple layers. E.g., consumers can retrieve data at the same time from PaaS and IaaS for the

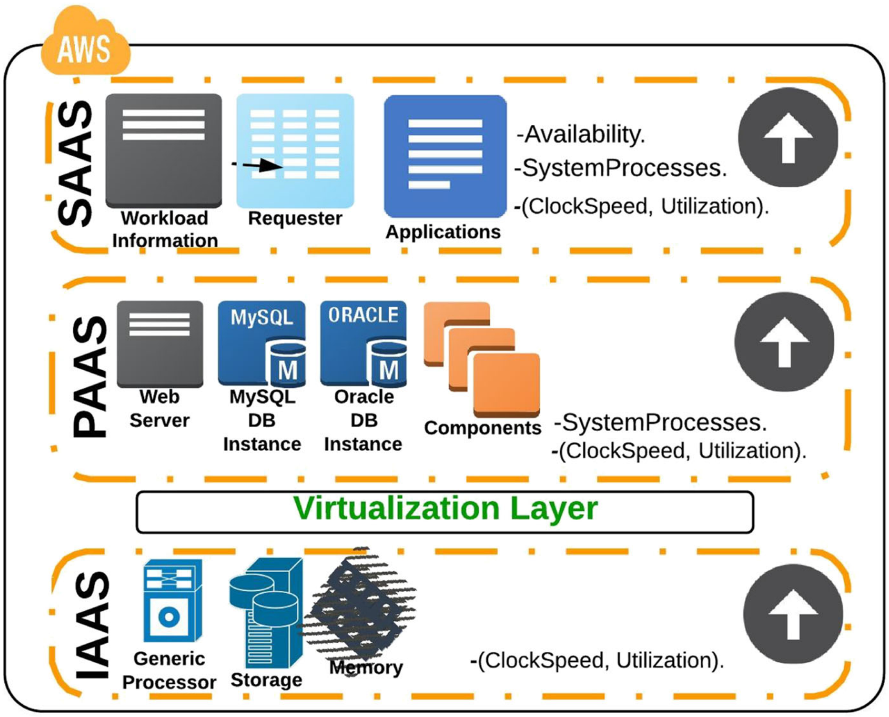
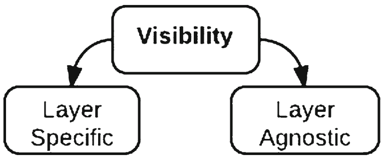


Fig. 7 Components across cloud layers and QoS propagating

Fig. 8 Visibility categorization



same application (Table 3). This type of cloud monitoring is essential in all cases but obviously it is more effective for consumers requiring complete awareness about their cloud applications.

4.5 Programming interfaces

Programing interfaces allows the development of software systems to enable monitoring across different layers of the cloud stack. It involves several components such as APIs, widgets and the command line to enable a consumer to monitor several components of the complex cloud systems in a unified manner.

4.5.1 Application programming interface

An application programming interface (API) is a particular set of rules ('code') and specifications that software programs follow to communicate with each other (Fig. 9).

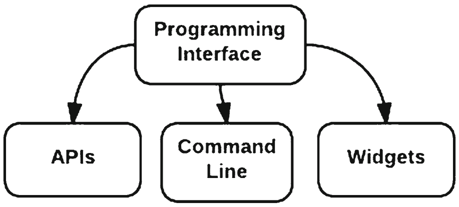


Fig. 9 Different types of programming interfaces

It serves as an interface between different software programs and facilitates their interaction; similar to the way the user interface facilitates interaction between humans and computers. In fact, most commercial monitoring tools such as Rackspace, Nimsoft, RevealCloud, and LogicMonitor provide their consumers with extensible open APIs enabling them tp specify their own required system functionalities.

4.5.2 Command-line

A command line provides a means of communication between a consumer and a computer that is based solely on textual input and output.

4.5.3 Widgets

In computer software, a widget is a software service available to consumers for running and displaying applets via a graphical interface on the desktop. Monitis and Reveal-Cloud are two popular commercial tools that provide performance data to consumers on multiple customizable widgets.

4.5.4 Communication protocols

All commercial tools adopt communication protocols for data transfer. Communication protocols vary and are different from one monitoring tool to another. For example, Monitis and Rackspace follow HTTPs and FTP protocols. Another example is Logic-Monitor, which adopts the encrypted Simple Network Management Protocol (SNMP).

5 Commercial monitoring tools

5.1 Monitis

Monitis [38] founded in 2005, has one unified dashboard where consumers can open multiple widgets for monitoring. A Monitis consumer needs to enter his/her credentials to access the hosting cloud account. In addition, a Monitis consumer can remotely monitor any website for uptime, in-house servers for CPU load, memory, or disk I/O, by installing Monitis agents to retrieve data about the devices. A Monitis agent can also be used to collect data of networked devices in an entire network (behind a firewall). This technique is used instead of installing a Monitis agent on each single device. Widgets can also be emailed as read only version to share the monitored information. Moreover, Monitis provides rich features for reporting the status of instances where consumers can specify the way a report should be viewed e.g. chart, or graph. It also enables its consumers to share the report publicly with others.

5.2 RevealCloud

CopperEgg [39,40] provides RevealCloud monitoring tool. It was founded in 2010 and Rackspace is a main partner. RevealCloud enables its consumers to monitor across cloud layers e.g. SaaS, PaaS, and IaaS. It is not dedicated to only one cloud resources provider, rather it is generic to allow a consumer to get its benefits within most popular cloud providers e.g. AWS EC2, Rackspace, etc. RevealCloud is one of the very few monitoring tools that supports maintaining monitored historical data.It can track up to 30 days of historical data, which is considered as a prime feature that most commercial monitoring tools lack.

5.3 LogicMonitor

LogicMonitor [41] was founded in 2008 and it is a partner with several third parties such as NetApp, VMWare, Dell, and HP. Similar to RevealCloud, LogicMonitor enables its consumers to monitor across cloud layers e.g. SaaS, PaaS, and IaaS. It also enables them to monitoring application operations on multi-cloud resources. Protocol used in communications is SSL. Moreover, LogicMonitor uses SNMP as a method of retrieving data about distributed virtual and physical resources.

5.4 Nimsoft

Nimsoft [42] was founded in 2011. Nimsoft supports multi-layers monitoring of both virtual and physical cloud resources. Moreover, Nimsoft enables its consumers to view and monitor their resources in case they are hosted on different cloud infrastructures e.g. a Nimsoft consumer can view resources on Google Apps, Rackspace, Amazon, Salesforce.com and others through a unified monitoring dashboard. Also, Nimsoft gives its consumers the ability to monitor both private and public clouds.

5.5 Nagios

Nagios [43] was founded in 2007, it supports multi-layer monitoring. It enables its consumers to monitor their resources on different cloud infrastructures as well as in-house infrastructure. Nagios utilizes SNMP for monitoring networked resources. Moreover, Nagios has been extended with monitoring functionalities for both virtual instances and storage services using a plugin-based architecture [31]. Typically, a Nagios server is required to collect the monitoring data, which would place it as a centralized solution. Moreover, Nagios is a cloud solution as a user would need to setup a Nagios server. However, many possible configurations can help create multiple hierarchical Nagios servers to reduce the disadvantages of a centralized server.

5.6 SPAE by SHALB

SHALB [44] was founded in 2002 and provides a monitoring solution called Security Performance Availability Engine (SPAE). SPAE is a typical network monitoring tool supporting a variety of network protocols such as HTTP, HTTPS, FTP, SSH, etc. It uses SNMP [45] to perform all of its monitoring processes and emphasizes security monitoring and vulnerability. However, SPAE does not support monitoring at different layers (IaaS, PaaS and SaaS). It enables its consumers to monitor networked resources including cloud infrastructure.

5.7 CloudWatch

CloudWatch [46] is one of the most popular commercial tools for monitoring the cloud. It is provided by Amazon to enable its consumers monitoring their resources residing on EC2. Hence, it does not support multi-cloud infrastructure monitoring. The technical approaches used in CloudWatch to collect data are implicit and not exposed to users. CloudWatch is limited in monitoring resources across cloud layers. However, an API is provided for users to collect metrics at any cloud layer but requires the users to write additional code.

5.8 OpenNebula

OpenNebula [47] is an open source monitoring system that provides management for data centers. It uses SSH as the protocol permitting consumers to gain access and gather information about their resources. Mainly, OpenNebula is concerned with monitoring physical infrastructures involved in data centers such as private clouds.

5.9 CloudHarmony

CloudHarmony [48] started monitoring services in the beginning of 2010. It provides a set of performance benchmarks of public clouds. It is mostly concerned in monitoring the common operating system metrics that are related to (CPU, disk and memory). Moreover, cloud to cloud network performance in CloudHarmony is evaluated in terms of RTT and throughput.

5.10 Windows Azure FC

Azure Fabric Controller (Azure FC) [49,50] is adopting centralized network architecture. It is a multi-layer monitoring system but, it does not support monitoring across different cloud infrastructures. Moreover, Azure FC utilizes SNMP for performing monitoring.

6 Classification and analysis of cloud monitoring tools based on taxonomy

With increasing cloud complexity, efforts needed for management and monitoring of cloud infrastructures need to be multiplied. The size and scalability of clouds when compared to traditional infrastructure involves more complex monitoring systems that have to be more scalable, effective and fast. Technically, this would mean that there is a demand for real-time reporting of performance measurements while monitoring cloud resources and applications. Therefore, cloud monitoring systems need to be advanced and customized to the diversity, scalability, and high dynamic cloud environments.

In Sect. 4, we analyzed in detail the main evaluation dimensions of monitoring. As discussed, not all of those dimensions are adopted by monitoring systems in either open source or commercial domains. Though, most of these dimensions, which are basically related to performance, have been addressed by the research community and have received some, attention, more considerable effort to achieve higher level of maturity is essential for monitoring cloud systems.

Decentralized approaches are gaining more trust over centralized approaches. In contrast to unstructured P2P, structured P2P networks present a practical and more efficient approach in terms of network architecture. However, considerable study is needed on decentralized networks that are with various degrees of centralization. Considering interoperability, either cloud-dependent or cloud-agnostic, both of these monitoring approaches gain high importance. Currently, both approaches are supported by several monitoring systems. Through our study, we found that cloud-dependent monitoring systems are mostly commercial, whereas, cloud-agnostic monitoring systems are typically open source.

We observe that matrix of the quality of service is the most important dimension of a monitoring system and list the quality parameters that can be monitored along with the related criteria. We also elaborate on how those quality parameters should be monitored, detected and reported. At which cloud layer a monitoring system should operate the monitoring processes. Further, the aggregation of multiple parameters for a consumer application is a critical aspect of monitoring. This means that a monitoring system should not be cloud layer specific or layer agnostic. This will determine the visibility characteristic of a cloud monitoring system. All of these issues in monitoring need more study by the cloud community and is still in demand formore technical improvements. Table 4 summarizes our study of monitoring platforms against evaluation dimensions explored in Sect. 4.

7 Conclusion and future research directions

This paper presented and discussed the state-of-the-art research in the area of cloud monitoring. In doing so, it presented several design issues and research dimensions that could be considered to evaluate a cloud computing system. It also presented several cloud monitoring tools, their features and shortcomings. Finally, this paper presented a taxonomy of current cloud monitoring tools with focus on future research directions that should be considered in the development of efficient cloud monitoring systems.

Table 4 Monitoring platforms against evaluation dimensions

Platform	Network arch. (centralized)	Network arch. (decentralized)	Interoperability multi-cloud	Visibility multi-layers	SNMP	Extendable APIs
Monitis [38]	Not-stated (SaaS solution)	Not-stated (SaaS solution)	Yes	Yes	Yes	Yes
RevealCloud [39,40]	Not-stated (SaaS solution)	Not-stated (SaaS solution)	Yes	Yes	Not-stated	Yes
LogicMonitor [41]	Not-stated (SaaS solution)	Not-stated (SaaS solution)	Yes	Yes	Yes	Yes
Nimsoft [42]	Yes	Yes	Yes	Yes	Yes	Yes
Nagios [31,43]	Yes	Yes	Yes	Yes	Yes	Yes
SPAE [44,45]	Not-stated (SaaS solution)	Not-stated (SaaS solution)	Yes	No	Yes	No
CloudWatch [46]	Not-stated (SaaS solution)	Not-stated (SaaS solution)	No	Yes	Not-stated	Yes
OpenNebula [47]	Yes	No	No	No	Not-stated	No
CloudHarmony [48]	Not-stated (SaaS solution)	Not-stated (SaaS solution)	Yes	No	Not-stated	No
Azure FC [49,50]	Yes	Not-stated	No	Yes	Yes	Yes

Since monitoring becomes an essential component of the whole cloud infrastructure, its elasticity has to be given a high considerable priority. Based on this fact and on the aforementioned monitoring aspects and approaches, we believe that considerable effort is required to have more reliable cloud monitoring systems. Furthermore, we found there is a lack of reachable standards on procedure, format, and metrics to assess the development of cloud monitoring. Hence, we recommend having more collaborative use of research facilities in which tools, lessons learned and best practices can be shared among all interested researches and professions.

References

1. Mell P, Grance T (2011) The NIST definition of cloud computing (draft). NIST Spec Publ 800:145
2. Letaifa A, Haji A, Jebalia M, Tabbane S (2010) State of the art and research challenges of new services architecture technologies: virtualization, SOA and cloud computing. Int J Grid Distrib Comput 3
3. Cong C, Liu J, Zhang Q, Chen H, Cong Z (2010) The characteristics of cloud computing. In: 39th international conference on parallel processing workshops (ICPPW), pp 275–279
4. Zhang S, Zhang S, Chen X, Huo X (2010) Cloud computing research and development trend. In: 2nd international conference on future networks, ICFN'10, pp 93–97
5. Ahmed M, Chowdhury ASMR, Ahmed M, Rafee MMH (2012) An advanced survey on cloud computing and state-of-the-art research issues. Int J Comput Sci Issues (IJCSI) 9
6. Atzori L, Granelli F, Pescapè A (2011) A network-oriented survey and open issues in cloud computing
7. Shin S, Gu G (2012) CloudWatcher: network security monitoring using openflow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?). In: 2012 20th IEEE international conference on network protocols (ICNP), pp 1–6
8. De Chaves SA, Uriarte RB, Westphall CB (2011) Toward an architecture for monitoring private clouds. IEEE Commun Mag 49:130–137

9. Grobauer B, Walloschek T, Stocker E (2011) Understanding cloud computing vulnerabilities. In: IEEE security and privacy, vol 9, pp 50–57
10. Moses J, Iyer R, Illikkal R, Srinivasan S, Aisopos K (2011) Shared resource monitoring and throughput optimization in cloud-computing datacenters. In: 2011 IEEE international parallel and distributed processing symposium (IPDPS), pp 1024–1033
11. Wang L, Kunze M, Tao J, von Laszewski G (2011) Towards building a cloud for scientific applications. Adv Eng Softw 42(9):714–722
12. Wang L, Chen D, Ma Y, Wang J (2013) Towards enabling cyberinfrastructure as a service in clouds. Comput Electr Eng 39(1):3–14
13. Wang L, von Laszewski G, Younge AJ, He X, Kunze M, Tao J (2010) Cloud computing: a perspective study. New Gener Comput 28(2):137–146
14. Begoli E, Horey J (2012) Design principles for effective knowledge discovery from big data. In: Joint working IEEE/IFIP conference on software architecture (WICSA) and European conference on software architecture (ECSA), pp 215–218
15. Bryant R, Katz RH, Lazowska ED (2008) Big-data computing: creating revolutionary breakthroughs in commerce, science and society
16. Labrinidis A, Jagadish H (2012) Challenges and opportunities with big data. In: Proceedings of the VLDB endowment, vol 5, pp 2032–2033
17. Ma Y, Wang L, Liu D, Yuan T, Liu P, Zhang W (2013) Distributed data structure templates for data-intensive remote sensing applications. Concurr Comput Pract Exp 25(12):1784–1797
18. Zhang W, Wang L, Liu D, Song W, Ma Y, Liu P, Chen Dan (2013) Towards building a multi-datacenter infrastructure for massive remote sensing image processing. Concurr Comput Pract Exp 25(12):1798–1812
19. Zhang W, Wang L, Ma Y, Liu D (2013) Design and implementation of task scheduling strategies for massive remote sensing data processing across multiple data centers. Pract Exp Softw. doi:10.1002/spe.2229
20. Twitter and Natural Disasters (2011) Crisis communication lessons from the Japan tsunami. http://www.sciencedaily.com/releases/2011/04/110415154734.htm. Accessed 22 Feb 2014
21. Nita M-C, Chilipirea C, Dobre C, Pop F (2013) A SLA-based method for big-data transfers with multi-criteria optimization constraints for IaaS. In: 2013 11th roedunet international conference (RoEduNet), pp 1, 6
22. Zhao M, Figueiredo RJ (2007) Experimental study of virtual machine migration in support of reservation of cluster resources. In: Proceedings of the 2nd international workshop on virtualization technology in distributed computing, p 5
23. Wang L, Chen D, Zhao J, Tao J (2012) Resource management of distributed virtual machines. IJAHUC 10(2):96–111
24. Calheiros RN, Ranjan R, Buyya R (2011) Virtual machine provisioning based on analytical performance and qos in cloud computing environments. In: International conference on parallel processing (ICPP), pp 295–304
25. Kirschnick J, Calero A, Edwards N (2010) Toward an architecture for the automated provisioning of cloud services. IEEE Commun Mag 48:124–131
26. Ranjan R, Zhao L, Wu X, Liu A, Quiroz A, Parashar M (2010) Peer-to-peer cloud provisioning: service discovery and load-balancing. In: Cloud computing, Springer, pp 195–217
27. Liu X, Yang Y, Yuan D, Zhang G, Li W, Cao D (2011) A generic QoS framework for cloud workflow systems. In: 2011 IEEE ninth international conference on dependable, autonomic and secure computing (DASC), pp 713–720
28. Ranjan R, Benatallah B Programming cloud resource orchestration framework: operations and research challenges. In: Technical report. http://arxiv.org/abs/1204.2204. Accessed 22 Feb 2014
29. Aceto G, Botta A, de Donato W, Pescapè A (2013) Cloud monitoring: a survey. Comput Netw 57:2093–2115
30. Shao J, Wei H, Wang Q, Mei H (2010) A runtime model based monitoring approach for cloud. In: 2010 IEEE 3rd international conference on cloud computing (CLOUD), pp 313–320
31. Caron E, Rodero-Merino L, Desprez F, Muresan A (2012) Auto-scaling, load balancing and monitoring in commercial and open-source clouds
32. Spring J (2011) Monitoring cloud computing by layer, part 1. IEEE Secur Priv 9:66–68

33. Anand M (2012) Cloud monitor: monitoring applications in cloud. In: Cloud computing in emerging markets (CCEM), 2012 IEEE international conference on communication, networking and broadcasting, pp 1–4
34. Kutare M, Eisenhauer G, Wang C, Schwan K, Talwar V, Wolf M (2010) Monalytics: online monitoring and analytics for managing large scale data centers. In: Proceedings of the 7th international conference on autonomic computing, pp 141–150
35. Sundaresan S, de Donato W, Feamster N, Teixeira R, Crawford S, Pescape A (2011) Broadband internet performance: a view from the gateway. In: ACM SIGCOMM computer communication review, pp 134–145
36. Massonet P, Naqvi S, Ponsard C, Latanicki J, Rochwerger B, Villari M (2011) A monitoring and audit logging architecture for data location compliance in federated cloud infrastructures. In: IEEE international symposium on parallel and distributed processing workshops and PhD forum (IPDPSW), pp 1510–1517
37. Davis C, Neville S, Fernandez J, Robert J-M, Mchugh J (2008) Structured peer-to-peer overlay networks: ideal botnets command and control infrastructures? In: Computer security—ESORICS 2008, pp 461–480
38. Monitis (2014) http://portal.monitis.com/. Accessed 22 Feb 2014
39. RevealCloud (2014) http://copperegg.com/. Accessed 22 Feb 2014
40. RevealCloud (2014) http://sandhill.com/article. Accessed 22 Feb 2014
41. LogicMonitor (2014) http://www.logicmonitor.com/why-logicmonitor. Accessed 22 Feb 2014
42. Nimsoft (2014) http://www.nimsoft.com/solutions/nimsoft-monitor/cloud. Accessed 22 Feb 2014
43. Nagios (2014) http://www.nagios.com. Accessed 22 Feb 2014
44. SPAE (2014) http://shalb.com/en/spae/spae_features/. Accessed 22 Feb 2014
45. SPAE (2014) http://www.rackaid.com/resources/server-monitoring-cloud. Accessed 22 Feb 2014
46. CloudWatch (2014) http://awsdocs.s3.amazonaws.com/AmazonCloudWatch/latest/acw-dg.pdf. Accessed 22 Feb 2014
47. OpenNebula (2014) http://opennebula.org/documentation:rel4.0. Accessed 22 Feb 2014
48. Cloudharmony (2014) http://cloudharmony.com/. Accessed 22 Feb 2014
49. Azure FC (2014) http://www.techopedia.com/definition/26433/azure-fabric-controller. Accessed 22 Feb 2014
50. Azure FC (2014) http://snarfed.org/windows_azure_details#Configuration_and_APIs. Accessed 22 Feb 2014
51. Nathuji R, Kansal A, Ghaffarkhah A (2010) Q-clouds: managing performance interference effects for QoS-aware clouds. In: Proceedings of the 5th European conference on computer systems, pp 237–250