QoS-Aware Autonomic Resource Management in Cloud Computing: A Systematic Review

Article *in* ACM Computing Surveys · December 2015

DOI: 10.1145/2843889

CITATIONS
188

READS
13,070

2 authors:

Sukhpal Singh Gill
Queen Mary, University of London
183 PUBLICATIONS **5,670** CITATIONS

SEE PROFILE

Inderveer Chana
Thapar University
117 PUBLICATIONS **4,352** CITATIONS

SEE PROFILE

QoS-Aware Autonomic Resource Management in Cloud Computing: A Systematic Review

SUKHPAL SINGH and INDERVEER CHANA, Thapar University, Patiala

As computing infrastructure expands, resource management in a large, heterogeneous, and distributed environment becomes a challenging task. In a cloud environment, with uncertainty and dispersion of resources, one encounters problems of allocation of resources, which is caused by things such as heterogeneity, dynamism, and failures. Unfortunately, existing resource management techniques, frameworks, and mechanisms are insufficient to handle these environments, applications, and resource behaviors. To provide efficient performance of workloads and applications, the aforementioned characteristics should be addressed effectively. This research depicts a broad methodical literature analysis of autonomic resource management in the area of the cloud in general and QoS (Quality of Service)-aware autonomic resource management specifically. The current status of autonomic resource management in cloud computing is distributed into various categories. Methodical analysis of autonomic resource management in cloud computing and its techniques are described as developed by various industry and academic groups. Further, taxonomy of autonomic resource management in the cloud has been presented. This research work will help researchers find the important characteristics of autonomic resource management and will also help to select the most suitable technique for autonomic resource management in a specific application along with significant future research directions.

Categories and Subject Descriptors: A.1 [**General Literature**]: Introductory and Survey; C.0 [**General**]: Systems Architectures; C.2.4 [**Computer-Communication Networks**]: Distributed Systems; D.4.1 [**Process Management**]: Scheduling; H.3.4 [**Systems and Software**]: Distributed Systems; J.7 [**Distributed Parallel and Cluster Computing**]; K.6.2 [**Management of Computing and Information Systems**]: Installation Management

General Terms: Documentation, Cloud Computing, Methodical Analysis, Theory, Management

Additional Key Words and Phrases: Resource provisioning, cloud computing, autonomic management, service-level agreement, quality of service, grid computing, resource scheduling, autonomic cloud computing, autonomic computing, self-management, self-optimizing, self-protecting, self-healing, self-configuring, resource management

---

Sukhpal Singh gratefully acknowledges the Department of Science and Technology (DST), Government of India, for awarding him the INSPIRE (Innovation in Science Pursuit for Inspired Research) Fellowship (Registration/IVR Number: 201400000761 [DST/INSPIRE/03/2014/000359]) to carry out this research work. Mr. Singh received the Gold Medal in Master of Engineering in Software Engineering. Mr. Singh is on the Roll-of-honor being the DST Inspire Fellow as an SRF Professional under the INSPIRE Fellowship. We would like to thank all the anonymous reviewers for their valuable comments and suggestions for improving the article. We would like to thank Dr. Maninder Singh [EC-Council's Certified Ethical Hacker (C-EH)] for useful suggestions.

Authors' addresses: S. Singh, Computer Science and Engineering Department, Thapar University, Patiala, Punjab, India-147004; email: ssgill@thapar.edu; I. Chana, Computer Science and Engineering Department, Thapar University, Patiala, Punjab, India-147004; email: inderveer@thapar.edu.

DOI: http://dx.doi.org/10.1145/2843889

ACM Reference Format:
Sukhpal Singh and Inderveer Chana. 2015. QoS-aware autonomic resource management in cloud computing: A systematic review. ACM Comput. Surv. 48, 3, Article 42 (December 2015), 46 pages.
DOI: http://dx.doi.org/10.1145/2843889

1. INTRODUCTION AND MOTIVATION

Cloud computing offers pay-per-use-based services such as infrastructure, platform, and software through different cloud providers [Caton et al. 2013]. As the cloud offers these three types of services, it requires Quality of Service (QoS) to efficiently monitor and measure the delivered services and further needs to follow Service-Level Agreements (SLAs) to ensure their efficient delivery. However, providing dedicated cloud services that ensure users' dynamic QoS requirements and avoid SLA violations is a big challenge in cloud computing. Currently, cloud services are provisioned and scheduled according to resources' availability without ensuring the expected performances. The cloud provider should evolve its ecosystem in order to fulfill QoS-aware requirements of each component of the cloud. To realize this, there is a need to consider two important aspects that reflect the complexity introduced by the cloud management: QoS-aware and self- or autonomic management of cloud services. The QoS-aware aspect involves the capacity of a service to be aware of its behavior to ensure the elasticity, high availability, reliability of service, cost, time, and so forth [Lango 2014]. Self- or autonomic management implies the fact that the service is able to self-manage itself as per its environment's needs. Therefore, significant challenging tradeoffs exist to ensure that the QoS guarantees and performance are met by improving cost-effectiveness and resource utilization. Based on human guidance, autonomic systems keep the system stable in unpredictable conditions and adapt quickly in new environmental conditions like software, hardware failures, and so forth. Autonomic systems are working based on QoS parameters and are inspired by biological systems that can easily handle problems like uncertainty, heterogeneity, and dynamism. Based on QoS requirements, autonomic systems provide self-optimization and manage the complexity of a system in a proactive way to reduce cost. The main issues in this context are as follows: (1) there is no single provider that provides autonomic services and (2) only AWS (Amazon Web Service) currently delivers integrated autonomic services with a very low degree of customization. In existing autonomic systems, only a few QoS parameters are considered [Yeo et al. 2010]. There is a need for an autonomic resource management system that considers all the important QoS parameters like availability, security, execution time, SLA violation rate, and so forth for better resource management [Huebscher et al. 2008; Simonin et al. 2013].

In the cloud environment, uncertainty and dispersion of resources cause problems in allocation of resources [Salehie et al. 2005], which result for many reasons, such as the following:

a. Heterogeneity (due to different types of resources and scheduling techniques)
b. Dynamism (detect and fulfill the requirements of the application at runtime)
c. Failures (failure of system or resources, which leads to performance degradation)

Unfortunately, present cloud computing systems and management techniques are unable to handle the aforementioned problems efficiently at runtime. To overcome these problems, cloud systems should contain self-management characteristics like self-optimizing, self-healing, self-protecting, and self-configuring. Thus, there is a need to automatically manage QoS requirements of cloud users, thus helping the cloud providers achieve the SLAs and avoid SLA violations. Autonomic cloud computing systems can provide the environment in which applications can be managed efficiently

by fulfilling QoS requirements of applications without human involvement [Kim et al. 2009].

1.1. Need for Autonomic Cloud Computing

The cloud services delivered by the heterogeneous and dynamic nature of the cloud resources depend on the QoS. Fulfilling QoS requirements and maximizing the efficiency and dispersion, heterogeneity, and uncertainty of resources bring challenges to cloud computing systems, which cannot be efficiently satisfied with traditional resource allocation policies in the cloud environment [Erdil 2013]. Autonomic cloud computing can provide one of the solutions for effective allocation of resources by fulfilling the QoS requirements of users and healing unexpected failures at runtime automatically, thus optimizing QoS parameters [Mohamed et al. 2014]. The first objective of autonomic cloud computing is to identify and schedule the suitable resources for the appropriate workloads on time to increase the effectiveness of resource utilization. In other words, the amount of resources should be minimum for a workload to maintain a required level of service quality or to minimize the workload completion time (or maximize throughput) of a workload automatically [Zhan et al. 2015]. For better resource scheduling, best-resource workload mapping is required. The second objective of autonomic cloud computing is to identify the adequate and suitable workload that supports the scheduling of multiple workloads, to be capable to fulfill numerous QoS requirements such as CPU utilization, availability, reliability, security, and so forth for the cloud workload. Therefore, resource scheduling considers the execution time of every distinct workload but also, most importantly, the overall performance based on the type of workload, that is, with different QoS requirements (heterogeneous workloads) and with similar QoS requirements (homogenous workloads) [Singh et al. 2015].

1.2. Motivation for Research

- Resource scheduling in the cloud consists of dynamic allocation of provisioned resources to cloud workloads automatically. Consequently, this study emphasizes autonomic resource management techniques based on different scheduling criteria.
- We recognized the necessity of a methodical literature survey after considering progressive research in autonomic resource management techniques in cloud computing. Therefore, we have summarized the available research based on a broad and methodical search in the existing database and present the research challenges for advanced research.

1.3. Our Contributions

- A comprehensive investigation has been conducted to study various existing autonomic resource management techniques in cloud computing accomplished by in-depth learning of autonomic resource provisioning and scheduling techniques.
- The aforementioned autonomic resource management techniques have been compared and categorized based on the common characteristics and properties of self-management (self-healing, self-configuring, self-optimizing, and self-protecting).
- Future research directions in the area of QoS-aware autonomic resource management in the cloud are presented.

1.4. Related Surveys

Earlier surveys by Buyya et al. [2012] and Rahman et al. [2011] have been very innovative, but as the research has persistently grown in the field of autonomic cloud computing, there is a necessity for a methodical literature survey to evaluate, upgrade, and integrate the existing research presented in this field. This research augments the previous surveys and presents a fresh methodical literature survey to evaluate and

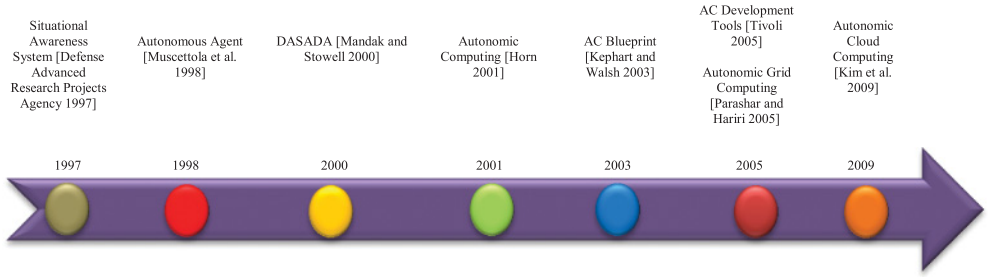

Fig. 1. Evolution of autonomic computing.

discover the research challenges based on available existing research in the field of autonomic resource management in cloud computing.

1.5. Article Organization

The organization of the rest of this article is as follows: The history of autonomic computing is presented in Section 2. Section 3 presents the background of autonomic cloud computing. Section 4 describes the review technique used to find and analyze the available existing research, research questions, and search criteria. Section 5 presents the extraction outcomes of the methodical literature survey. Section 6 describes the phases of autonomic resource management in the cloud. Section 7 presents QoS-aware autonomic resource management techniques and their comparisons and taxonomy of autonomic resource management. Section 8 describes the perspective model of autonomic resource management. Future research directions are described in Section 9, and discussions are presented in Section 10. Section 11 concludes this research work.

2. HISTORY OF AUTONOMIC COMPUTING

Autonomic computing is a self-manageable computing system, and the word "autonomic" originated from the biological area. The human body works in a self-regulating manner without human intervention. Autonomic systems are inspired by the biological system (autonomic nervous system) that can easily handle problems like uncertainty, heterogeneity, dynamism, faults, and so forth. Based on human guidance, autonomic systems keep the computing system stable in unpredictable conditions and adapt quickly in new environmental conditions like software and hardware failures. Due to the self-management property of computing systems, the complexity of the system is also invisible to the user. Just as the Autonomic Nervous System (ANS) controls the human body's functions (breathing, digestion, etc.), autonomic systems control the working of computing systems and applications without the involvement of humans. In this section, the evolution of autonomic computing is discussed briefly in the field of computer science along the path of 21st century.

2.1. Evolution of Autonomic Computing

In the early 1990s, researchers of different prestigious organizations started development of self-managing or autonomic systems because of the complexity and heterogeneity of computing systems, and research on autonomic systems continued for a whole decade [Rahman at al. 2011]. The evolution of autonomic computing is shown in Figure 1. For military purposes, DARPA (the Defense Advanced Research Projects Agency) started a preliminary autonomic project in 1997, called the Situational Awareness System (SAS) [Defense Advanced Research Projects Agency 1997]. The purpose of SAS was to get information about enemies (to detect enemy tanks in the battlefield)

on their personal devices using personal communications. Decentralized P2P (Peer-to-Peer) mobile adaptive routing was used to spread information to all the soldiers. For space projects (MarsPathfinder and Deep Space-1), NASA used autonomic systems in 1998 to make a space probe quickly by adapting extraordinary situations automatically [Muscettola et al. 1998]. Under these projects, NASA designed temporal planning based on Autonomous Agents (AAs). Therefore, autonomous operations were performed by spacecraft for long durations. In 2000, an autonomic-system-based project was launched by DARPA, called DASADA (Dynamic Assembly for Systems Adaptability, Dependability, and Assurance), to reduce the complexity of distributed systems, and it fulfilled requirements such as adaptability, dependability, and high assurance [Mandak and Stowell 2000].

In 2001, the journey of autonomic computing was started by IBM. IBM developed an autonomic computing system to reduce the increasingly rising complexity of managing computing systems, which further helped to reduce the workload of administrators [Horn 2001]. An architectural blueprint of an autonomic computing system was introduced in 2003 [Kephart and Walsh 2003], in which IBM proposed four properties of self-management (self-optimizing, self-healing, self-protecting, and self-configuring), discussed in Section 6.5. In 2005, IBM became the leader of autonomic computing and started developing autonomic-computing-based toolkits like Tivoli [2005]. Different autonomic computing development tools were offered by IBM to increase the adoption of autonomic computing; one famous tool was developed for making autonomic decisions based on rules embedded in a software application called PMAC (Policy Management Autonomic Computing). In 2005, Parashar and Hariri proposed an autonomic system in the context of grid computing to reduce the complexity of heterogeneous and dynamic components. This autonomic system also provides features like management, deployment, and composition of complex applications in the grid [Parashar and Hariri 2005]. Other existing autonomic systems in the grid are Aneka, Condor-G, Nimrod-G, and Pegasus [Rahman et al. 2011]. In 2009, the concept of autonomic computing in the cloud was introduced to deliver autonomic cloud services to users by considering reliability and scalability as important QoS parameters [Kim et al. 2009]. Fulfilling QoS requirements and maximizing the efficiency of cloud systems by using the concept of autonomic computing cannot be efficiently satisfied with traditional resource allocation policies in the cloud environment. The next section explains autonomic cloud computing in detail.

3. AUTONOMIC CLOUD COMPUTING: THE BACKGROUND

Autonomic systems based on QoS parameters are inspired by biological systems that can easily handle problems like uncertainty, heterogeneity, dynamism, faults, and so forth. The goal of autonomic systems is to execute an application within a deadline by fulfilling QoS requirements as described by users with minimum complexity. Autonomic computing systems are basically inspired from the ANS of humans. The ANS has the capability to deal with all situations dynamically and manage these situations in an unpredictable environment. As the ANS controls the human body's functions (breathing, digestion, etc.), Autonomic Cloud Computing Systems (ACCSs) control the working of cloud-based systems and applications without the involvement of humans. Similar to the ANS, ACCSs check, monitor, and respond according to the situation, such as self-healing, self-protecting, self-configuring, and self-optimizing.

3.1. Overview of Autonomic Cloud Computing

ACCSs are based on IBM's autonomic model [Kephart and Walsh 2003], which considers four steps of the autonomic system (Monitor, Analyze, Plan, and Execute) in a

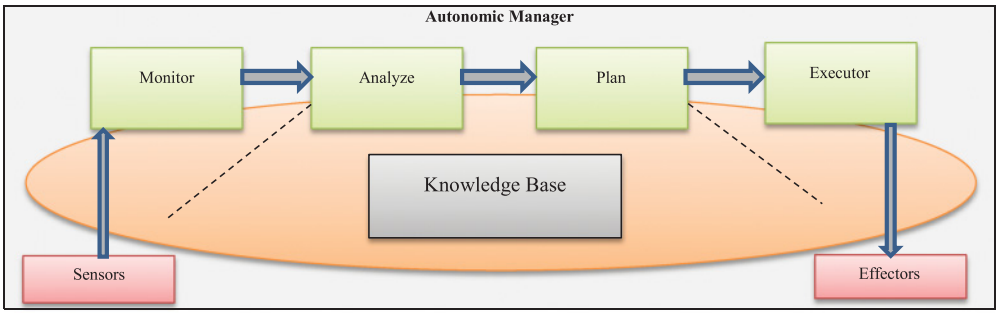

Fig. 2. Architecture of an autonomic system.

control loop, two interfaces (sensors and effectors) for environmental interaction, and one database (knowledge base) to store rules, as shown in Figure 2.

ACCSs are composed of Autonomic Elements (AEs); the Autonomic Manager (AM) is an intelligent agent that interacts with the environment through manageability interfaces (*Sensors* and *Effectors*) and takes actions according to the input received from sensors and rules defined in a knowledge base in low-level language. The AM is configured by the administrator based on alerts and actions in high-level language. Figure 19 shows the interaction of IBM's autonomic model with cloud computing as described in Section 8.

Initially, *Monitors* are used to collect the information from sensors for monitoring continuously the value of QoS parameters while interacting with outside interfaces and transfer this information to the next module for further analysis. The *Analyze and Plan* modules start analyzing the information received from the monitoring module and make a plan for adequate actions for corresponding alerts generated by the system. Once data has been analyzed, this autonomic system executes the actions corresponding to the alerts automatically [Rimal et al. 2011]. *Executor* implements the plan after complete analysis. Maintaining the value of QoS parameters is the main objective of *Executor*. Based on the output given by analysis, *Executor* tracks the new changes and takes action according to the rules described in the knowledge base. *Effector* is used to transfer the new policies, rules, and alerts to other nodes of the autonomic system with updated information.

3.2. Self-Management Properties of Autonomic Cloud Computing

Self-management in cloud computing has four generic properties: (1) self-healing, (2) self-protecting, (3) self-optimizing, and (4) self-configuring, as shown in Figure 3. A description and example of self-management properties are presented in Table I.

3.3. Evolution of Autonomic Cloud Computing

This section describes the QoS parameters in which the resource management is proposed across the backstory of the cloud. Further remarkable *Quality of Service* (*QoS)* parameters and *Focus of Study (FoS)* of autonomic cloud computing along with evolution of the cloud throughout the years are described in autonomic cloud computing evolution, as shown in Figure 4. In 2009, workload-provisioning-based [Quiroz et al. 2009] autonomic techniques and self-optimization-based [Nallur et al. 2009] autonomic techniques have been proposed. Quiroz et al. [2009] proposed a clustering-based decentralized approach to detect patterns in virtual environments, and a model-based approach had been designed to monitor the performance and approximate service time of the application to maintain the SLA.

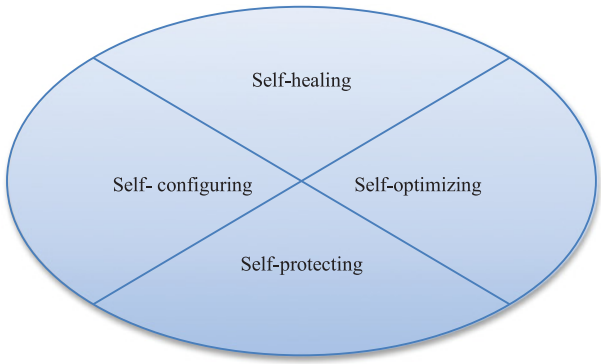

Fig. 3. Self-management in the cloud.

Table I. Self-Management Properties

Self-Management Property	Description	Example
Self-healing	Capability of an autonomic system to identify, analyze, and recover from unfortunate faults automatically	Improves performance through fault tolerance by reducing or avoiding the impact of failures on execution
Self-configuring	Capability of an autonomic system to adapt to the changes in the environment	Installation of missed or outdated components based on the alert generated by the system without human intervention
Self-optimizing	Capability of an autonomic system to improve the performance	To complete the execution of current workload and reduce overloading and underloading of resources
Self-protecting	Capability of an autonomic system to protect against intrusions and threats	To detect and protect the autonomic system from malicious attacks

Nallur et al. [2009] presented a self-optimization-based autonomic technique using utility theory to maintain the SLA and considered QoS requirements such as scalability and availability. This theoretical technique is difficult to identify the SLA deviation and SLA violation in case of performance degradation.

In 2010, autonomic techniques based on anomaly detection [Smith et al. 2010], SLA awareness [Bouchenak 2010], and deadlines and budgets [Mao et al. 2010] were suggested. Smith et al. [2010] presented an autonomic anomaly detection technique in which data was analyzed and stored in a uniform format after data collection to detect the faulty nodes. In this technique, Bayesian Network-based Dimensionality Reduction is used to extract the relevant data to reduce the computation overhead and increase accuracy; thus, complexity to handle the anomalies is increased due to the use of an unsupervised learning technique. Bouchenak [2010] presented an SLA-based technique for elastic clouds to meet the QoS requirements of cost and availability and also discussed the impact of SLA violations on performance. To deliver cloud services with an SLA guarantee, there is a need to specify the value of SLA deviation along with a penalty or compensation in case of SLA violations. Mao et al. [2010] proposed an autoscaling technique (Cloud Auto Scaling (CAS)) in which computing resources are scaled based on performance requirements and workload information in virtual environments. CAS schedules activities of VM instance startup and shutdown automatically to improve the performance. CAS enables users to finish the execution of

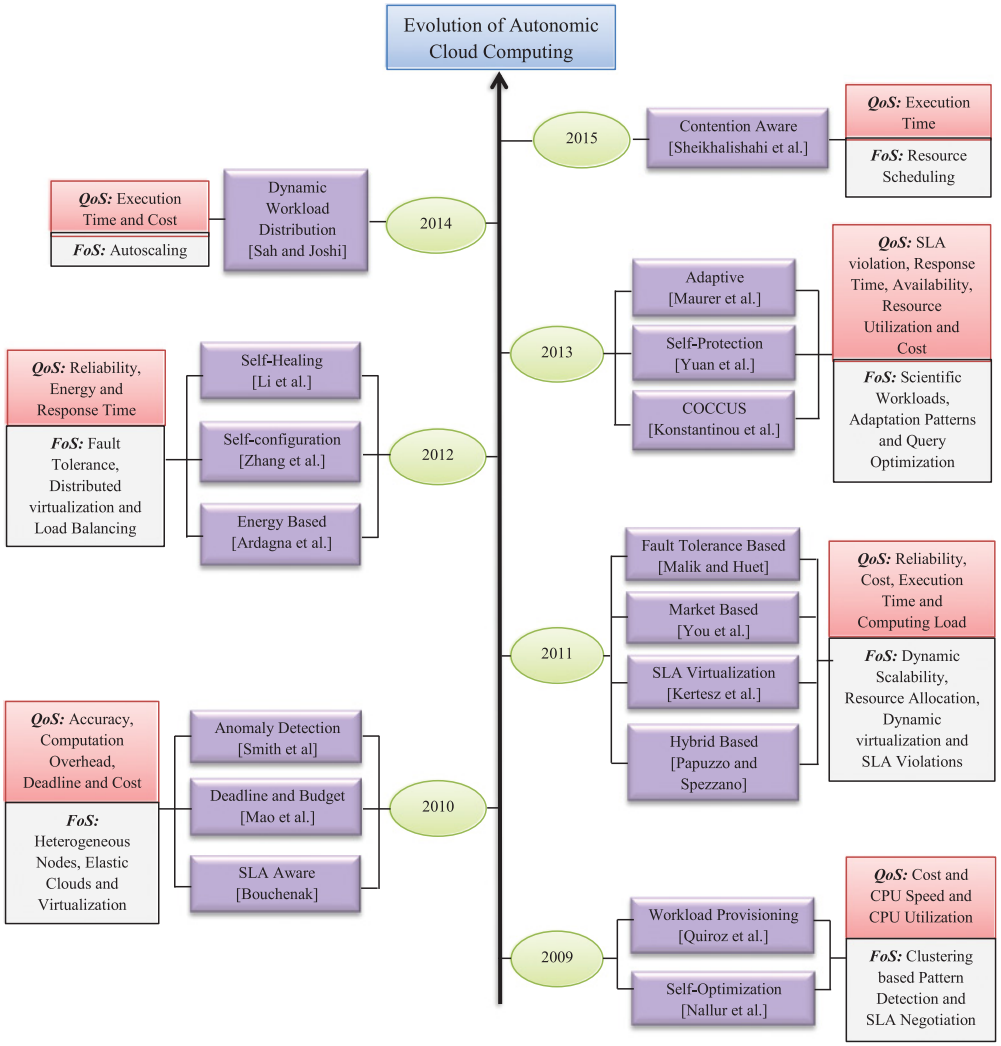

Fig. 4. Evolution of autonomic cloud computing.

workloads or tasks within their deadline with minimum cost. The VM manager maintains a relation between cloud providers and the auto-scaling mechanism and executes a plan of the auto-scaling decision maker to finish the execution of workloads.

In 2011, autonomic techniques based on fault tolerance [Malik and Huet 2011], markets [You et al. 2011], hybrids [Papuzzo and Spezzano 2011], and SLA aware virtualization [Kertesz et al. 2011] were proposed. To adapt, manage, enact, and configure the workflows, this technique used a P2P agent-based framework, but this mechanism is not able to share information among different nodes because of decentralization of data.

Malik and Huet [2011] proposed a forward- and backward-mechanism-based autonomic technique (Adaptive Fault Tolerance in Real-Time Cloud (AFRTC)) in a virtual environment to identify the faults through a checkpoint protocol. The AFRTC system is used to detect the faults, provide fault tolerance, and calculate the reliability of

nodes to make decisions. The reliability of nodes in virtual environments is changing adaptively. A node is reliable if a virtual node produces results within the specified deadline only. A node will be removed if it fails continually, and a new node will be added. AFRTC uses backward recovery if any node does not achieve the required level of reliability. You et al. [2011] proposed a market-based autonomic technique (Autonomic Allocation of Resources technique (ARAS-M)) by considering QoS requirements using a genetic algorithm to maintain an equilibrium state between service and request. ARAS-M uses a market-based mechanism to allocate the resources to workloads based on QoS requirements specified by the user. In this autonomic system, a Genetic Algorithm (GA) is used to attain an equilibrium state by adjusting price automatically. This system fulfills the demand of every workload along with its QoS requirements. Papuzzo and Spezzano [2011] proposed a bio-inspired-mechanism-based hybrid autonomic technique to handle peak load situations to reduce workload variations. Kertesz et al. [2011] proposed an SLA-aware-virtualization-based autonomic technique that focused on demand deployment, service brokering, and agreement negotiation to reduce SLA violations. This technique executes only homogenous workloads on homogenous resources, and due to dependency among different modules, SLA violations cannot be identified until the execution is completed.

In 2012, self-healing [Li et al. 2012], self-configuration [Zhang et al. 2012], and energy-based [Ardagna et al. 2012] autonomic techniques were proposed. Li et al. [2012] proposed a monitoring- and recovery-based self-healing technique to fulfill and verify the QoS requirements (reliability to identify faults at runtime) described by users. Zhang et al. [2012] proposed a power-aware adaptive autonomic technique that configures virtual resources based on power usage and SLA requirements. In this approach, the relation between resource utilization and energy consumption is not clearly mentioned, and it considers power consumption of the CPU directly without considering the power consumption of every independent component to get an accurate value. Ardagna et al. [2012] presented an autonomic technique based on dynamic voltage frequency scaling using a local search procedure to execute the realistic and synthetic workloads in an effective manner to reduce energy cost and SLA violations. This technique does not sufficiently explain the different variations among realistic and synthetic workloads.

In 2013, adaptive-based [Maurer et al. 2013], self-protecting [Yuan et al. 2013], and cost-based [Konstantinou et al. 2013] autonomic techniques were proposed. Maurer et al. [2013] proposed an SLA-based autonomic technique (Case Base Reasoning (CBR)) based on an autonomic control loop to execute both synthetically generated workloads after classification using a rule-based and case-based reasoning approach. CBR uses human-based interaction to make an agreement between the user and provider called SLA for successful execution of workloads by considering resource utilization and scalability as QoS requirements. In this system, various elastic levels are defined and a control loop is used to enable the autonomic computing in a virtual environment. SLA violations and resource utilization are improved in this autonomic system. Yuan et al. [2013] presented a rainbow-architecture-based ABSP (Architecture-Based Self-Protection) technique in which security threats are detected at runtime through the use of patterns. ABSP reduces security breaches and improves the depth of defense. The Self-COngured, Cost-based Cloud qUery Services (COCCUS) technique [Konstantinou et al. 2013] uses centralized architecture to provide the query-based facility in which users can ask queries regarding scheduling policies, priorities, and budget information. CloudDBMS is used to store information about the scheduling policies and user queries for further use. The main objectives of COCCUS are to (1) get and execute the user queries, (2) store the queries in the data structure, and (3) minimize the maintenance cost of query execution.

In 2014, a dynamic workload distribution [Sah and Joshi 2014] autonomic technique was proposed. Sah and Joshi [2014] proposed a dynamic workload-distribution-based autonomic technique (Autonomic Workload Manager (AWM)) using dynamic scalability and a modified auto-scaling algorithm to provide the solution for flat separable queuing network models. AWM uses DPSMS (Distributed Provisioning and Scaling Decision Making System) to distribute the workloads on resources based on their common QoS characteristics. AWM divides resources into two categories: coarse-grained and fine-grained resources. AVM allocates the resources based on minimum response time and high throughput. This autonomic system executes in three steps: (1) allocate resources to workloads, (2) minimize execution time, and (3) check execution status (if it executes within time and budget then it continues execution; otherwise, provide more resources).

In 2015, an autonomic technique based on resource contention-aware scheduling [Sheikhalishahi et al. 2015] was proposed. Sheikhalishahi et al. [2015] proposed the ARCS (Autonomic Resource Contention Scheduling) technique for a distributed system to reduce resource contention in which more than one job shares the same resource simultaneously. ARCS has four main components: (1) front-end policies (it performs admission control and queuing of jobs), (2) scheduler (it contains a backfilling scheduling algorithm), (3) information service (information about scheduler), and (4) back-end policies (mapping of resources with jobs). ARCS established a relationship among layers of distributed resource management [Maurer et al. 2011].

3.4. QoS-Aware Cloud

The settlement between consumers and cloud service providers fundamentally consists of parameters like price, time, and other QoS parameters. There are presently numerous methods that resolve the issue of expense and time slot settlement mechanisms without taking into account the significant characteristics of QoS [Hashem et al. 2015]. Ferretti et al. [2010] proposed a QoS-aware cloud architecture that aims to satisfy QoS requirements of the application. The principal function of this architecture was the efficient resource management of the virtual execution environment associated with the application [Zhou and Jiang 2014]. This architecture includes features that eliminate resource over provisioning by changing and configuring the amount of resources dynamically. But it does not describe the behavior of an application [Garg et al. 2011].

Nathuji et al. [2010] presented "Q-Cloud," a QoS-aware control framework that manages resource allocation in order to alleviate the consolidated workload interference problem. The principal aim of this framework was resource allocation at runtime of cohosting applications based on QoS requirements. The Q-States (QoS states) notion was proposed in order to assign additional QoS levels to the application. The goal of these states was to offer additional flexibility to users in order to easily improve their application-specific QoS level. This is why the QoS monitoring feature presented according to the "as a service" paradigm is interesting. This facility ensured continuous control of QoS attributes in order to avoid SLA violations [Singh et al. 2010]. Van et al. [2010] attempted to manage autonomic virtual resources for hosting cloud services. In this approach, a two-level architecture was proposed to separate applications' QoS specifications from the allocation and provision of resources. To analyze the QoS requirements of the hosted service, an application-oriented local decision model was presented for every application. This module determines a high-level performance goal in order to make the best decision in the allocation and provision phases. While the cited solutions aim to satisfy QoS requirements of applications, the management is still resource based by adapting resource reservations to QoS requirements [Zissis et al. 2012].

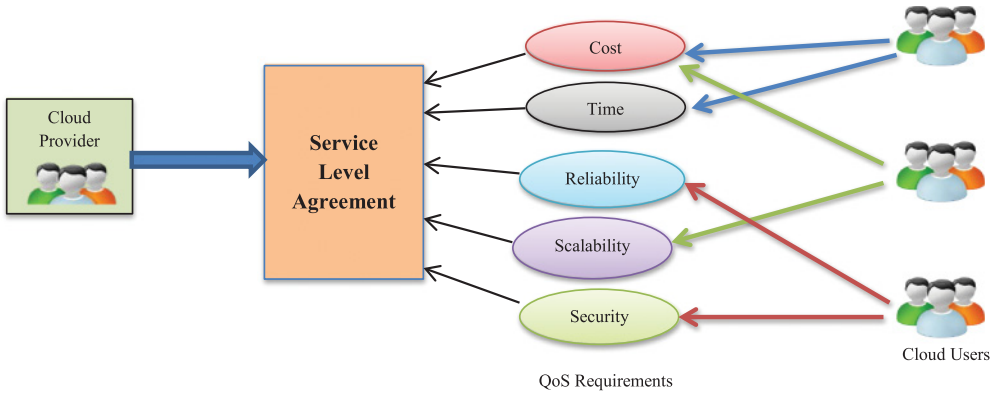

Fig. 5. Process of SLA negotiation.

3.5. SLA and QoS: Intertwined in the Autonomic Cloud

The cloud services offered to users consist of a set of components, which may be offered by different providers [Chana and Singh 2014]. To satisfy the requests of customers, the service must be provided in accordance with the required level of QoS, and QoS management must be considered to provide the End-to-End (E2E) QoS level. In current solutions, a degradation of a component can produce degradation of the global service [Weng and Bauer 2010]. Thus, one of the major issues in the current solutions of the cloud is to deliver the required services according to the QoS level expected by the user [Toosi et al. 2014]. The interaction of the cloud user and cloud provider to negotiate SLA is shown in Figure 5.

As more and more users give their applications to cloud providers, SLAs between clients and providers appear key [Kritikos et al. 2013]. To impose SLAs, endless supervision on QoS parameters is required because of the cloud's dynamic nature. In the current scenario of the cloud, it is a challenge to ensure QoS for cloud services. QoS is increasingly significant when composing services, because a debasing QoS in one of the services can deleteriously disturb the QoS of the complete composition [Assuncao et al. 2015; Ranjan et al. 2015]. To fulfill the QoS requirements of a cloud service, the required amount of resources is provisioned by the service provider [Martin et al. 2011]. Present technology of the cloud is not able to fulfill the SLA completely; however, research groups (industrial and academic) are doing research on QoS assurance problems in the cloud environment [Bonvin et al. 2011; Rodero et al. 2012].

Cloud users expect the system to guarantee the required QoS services regardless of any unexpected events. Consumer needs vary unpredictably depending on their types (developer, service provider, end-user) and their strategies (QoS requirement, cost effective, optimization, etc.). These unstable demands can result in SLA violations due to QoS degradation of the cloud services [Buyya et al. 2011]. While cloud providers can ensure the elasticity, high availability, and reliability of services, the QoS expectations of users are not yet achieved. Moreover, QoS is a very important aspect that must be considered in the different phases of the life cycle of cloud solutions. Thus, the QoS aspects should be considered from the design phase of cloud components in order to achieve the expected QoS requirements [Breskovic et al. 2011; Beach et al. 2015].

Complementarily, a self-management approach must be introduced to guarantee the E2E behavior of the global service. Self-management implies the ability of each service component to manage its behavior itself [Toosi et al. 2014]. Several independent applications are hosted on the cloud platform using a shared pool of resources to execute applications based on their QoS requirements [Faruk and Sivakumar 2014].

Table II. Comparison of QoS-Based Cloud and QoS-Based Grid

Criteria	QoS-Based Cloud	QoS-Based Grid	Findings
Service management	On-demand installation (installed only if required)	Preinstalled services (ready to use)	QoS-based cloud service uses less storage space.
QoS management	Critical applications (urgent workloads)	Noncritical applications	QoS-based cloud service is able to execute urgent applications with short deadline.
SLA fulfillment approaches	Self-adaptation mechanisms	Request/response models	QoS-based cloud service uses autonomic systems to deliver service in an effective manner.

Table III. Research Issues in QoS-Based Grid Computing

Research Issues	Reasons
Variation of resource availability	Due to hardware or software failure, there is dynamic system configuration and resource contention.
Parallel processing	Due to unavailability of required resources at a particular point of time, some subtasks are stuck in deadlock condition and continue to struggle for resources, which further leads to customer dissatisfaction (increasing execution time and cost).
Decentralized control	Global scheduler does not have control over local workloads.

To provide more flexibility along with VM consolidation, server virtualization is used for such platforms, and VM is migrated based on the required size of the virtual machine. Automating the process of management of virtual machines and considering the QoS requirements of hosted applications are key challenges [Tchana et al. 2013; Rodero et al. 2010]. Autonomic systems (similar to biological systems) need assistance from humans; they learn to maintain the stability of the system by taking appropriate steps and adopt to the changing environmental conditions (software failures, hardware failures, external conditions, etc.) automatically [Mosallanejad et al. 2014]. Generally, QoS is considered an important element of autonomic systems, and Figure 2 shows the main properties of an autonomic system (*Monitor, Analyze, Plan,* and *Execute*) in a control loop [Singh and Chana 2013a; Fargo et al. 2013].

3.6. QoS-Based Grid and Cloud

Computing-resource-based services offered to users consist of a set of components, which may be offered by different providers. To satisfy the request of customers, service must be provided in accordance with the required level of QoS. QoS is the service guarantee based on the parameters described by consumers, and the SLA is an authorized agreement that exhibits it. One of the major challenges in the current computing solutions (grid and cloud) is providing the required services according to the QoS level expected by users [Brandic and Dustdar 2011]. We have found three different criteria: (1) service management (to create, provide, maintain, and ensure required services), (2) QoS management (to deliver service by optimizing QoS parameters), and (3) SLA fulfillment approaches (to provide service without violation of SLA); based on these criteria, QoS-based grid and QoS-based cloud services can be compared, as shown in Table II.

The QoS-based grid computing issues of variation of resource availability, parallel processing, and decentralized control are described in Table III.

Existing research reported that QoS parameters such as execution time, cost, latency, throughput, and availability need to be optimized in future QoS-based grid systems.

Table IV. Research Issues in QoS-Based Cloud Computing

Research Issues	Reasons
Increasing resource cost	Overloading and underloading of resources are due to fluctuation of workloads.
Increasing workload execution time	Uncertainty and dispersion of resources cause problems in allocation of resources, resulting from factors such as heterogeneity, dynamism, and failures.
Increasing power consumption	A tremendous amount of energy is consumed by these data centers, which leads to high operational costs and contributes toward carbon footprints to the environment.

The main QoS constraints that QoS-based cloud computing faces need to be considered for efficient utilization of resources and are described in Table IV.

Existing research reported that QoS parameters such as execution time, cost, security, reliability, resource contention, resource utilization, energy consumption, SLA violation rate, and availability need to be optimized in future QoS-based cloud systems.

4. REVIEW TECHNIQUE

The methodical survey technique described in this research article has been taken from Kitchenham and Charters [2007]. The stages of this literature review include creating a review framework, executing the survey, investigating the results of review, recording the review results, and exploring research challenges. Table V describes the list of research questions required to plan the survey in autonomic cloud computing. Table VI describes the 380 research papers retrieved in a manual search and electronic database search. Figure 6 describes the review technique used in this methodical analysis.

4.1. Sources of Information

We searched broadly in electronic database sources as recommended by Kitchenham and Charters [2007]; the following electronic databases were used for searching:

- Springer (www.springerlink.com)
- ScienceDirect (www.sciencedirect.com)
- Google Scholar (www.scholar.google.co.in)
- IEEE eXplore (www.ieeexplore.ieee.org)
- ACM Digital Library (www.acm.org/dl)
- Wiley Interscience (www.Interscience.wiley.com)
- HPC (www.hpcsage.com)
- Taylor & Francis Online (www.tandfonline.com)

4.2. Search Criteria

The keyword “autonomic cloud computing” is involved in the abstract of each research paper in every search. It is a time-consuming and general method for review. The various search strings used in this review are described in Table VI. This methodical literature survey included both quantitative and qualitative research articles written in English from 2009 to 2015.

The basic research in this area was commenced in 2009, but rigorous development took place after 2010. We included research papers from journals, conferences, symposiums, workshops, and white papers from the industry along with technical reports. Exclusion criteria used at different stages are described in Figure 6. We applied individual searches on some journals of Springer, Wiley, Taylor and Francis, Science Direct, and so forth to cross check the e-search. Our search retrieved over 380 research articles, as shown in Figure 6, which were reduced to 298 research articles based on their titles, 217 research articles based on their abstracts and conclusions, and 171 research articles based on the full text. Then, these 171 research articles were investigated completely to find a final collection of 110 research articles through reference investigation and eliminating common challenges based on the criterion of inclusion and exclusion.

Table V. Research Questions and Motivation

Review Questions	Motivation
1. What is the current status of autonomic cloud computing? 2. How to clearly recognize the present and prospective desires of cloud consumers? 3. How to cut down the transfer and data cost and improve the cost-based transparency? 4. How to minimize the execution time of workloads to improve resource utilization? 5. How to optimize the resource utilization and minimize the cost simultaneously? 6. How to reduce the uptime of resources? How to reduce the execution cost and meet the deadline at same time? 7. What are the criteria for negotiation between consumer and provider? 8. How to reduce energy consumption and its impact on environment? 9. What other optimization techniques should be considered for efficient autonomic resource scheduling? 10. How to schedule the resources automatically to avoid overloading and underloading of resources? 11. What new rules should be required for effective autonomic resource scheduling? 12. How to design the autonomic resource scheduling algorithm to provide dynamic scalability at CPU, network, and application levels? 13. How to understand the cloud workloads for better autonomic resource scheduling? How to allocate the resources to cloud workloads for efficient utilization of resources? 14. How to identify and classify the various cloud workloads to design autonomic cloud computing system?	Various autonomic cloud computing techniques used in cloud computing are reported. Various scheduling criteria and QoS parameters for cloud autonomic resource scheduling considered so far are stated according to their level of importance. The research challenge in terms of the research question is discovering the existing research that assessed and compared the distinct autonomic cloud computing techniques. This study compared the different types of autonomic cloud computing techniques. For every type and subtype of autonomic cloud computing technique, various types of existing research have been presented. It is hard to detect the actual cost for autonomic resource scheduling. It will support in planning enhanced and extremely accessible approaches. The main aim of this review is to make an autonomic cloud computing database for future research through standardization and benchmarking of relative investigation of existing research.
1. How to develop an autonomic resource scheduling technique for cloud resources based on users' QoS requirements? 2. How to design an autonomic architecture that can fulfill QoS requirements of cloud service? 3. What are the QoS requirements of application and service the user plans to utilize from the cloud? 4. How to enable SLA by searching the suitable service based on QoS requirements and schedule the resources to every type of service? 5. How to understand and fulfill the QoS requirements of a particular service as described by users? 6. What are the criteria to modify the SLA with respect to time? What are the penalty and compensation criteria if the resource provider violates the SLA? 7. How to develop an autonomic resource allocation mechanism through effective utilization of resources and maintained SLA?	It provides the knowledge about the review done in this research article. It is mandatory to find out the number of research papers in each type of autonomic cloud computing technique, which helps to find the key research areas. A time-based count describes how autonomic cloud computing terms like SLA and QoS have progressed over time. Autonomic cloud computing has become the hotspot area in the cloud. The latest research in cloud is going toward effective autonomic cloud computing. The research challenges in terms of research questions emphasize identifying the present prominence of research in autonomic cloud computing. Different research questions are used to identify the key research areas for future investigation in the field of autonomic cloud computing.
1. What simulation tools are used for autonomic cloud computing and what parameters they are considering? 2. How to validate the autonomic resource scheduling technique through tools?	It is significant to recognize distinct autonomic cloud computing simulation tools.

Table VI. Search String

Serial No.	Keywords	Synonyms	Dates	Content Type
1	Autonomic Scheduling	Autonomic Resource Scheduling Algorithms	2009–2015	Journal, conference, workshop, magazine, white paper, and transactions
2	Autonomic Provisioning	Autonomic Resource Provisioning Algorithms		
3	Self-Healing	Fault tolerance in ACCS		
4	Self-Protecting	Security and attack detection in ACCS		
5	Self-Optimization	Minimization of cost, time, energy, etc., in ACCS		
8	Self-Configuration	Automatic confirmation of resources		
9	Autonomic QoS	Quality-of-Service parameters in ACCS		
10	Autonomic SLA	Service-Level Agreement in ACCS		
11	SLA and QoS	SLA and QoS in ACCS		
12	Self-Management	Resource management in ACCS		

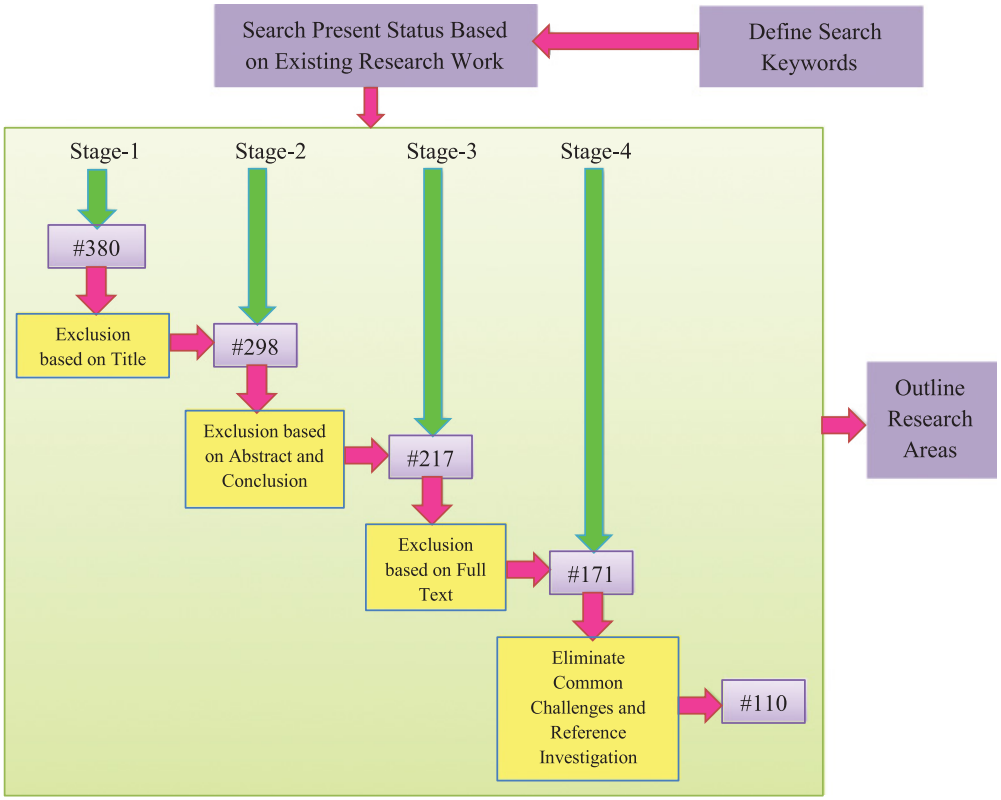

Fig. 6. Review technique used in this systematic review.

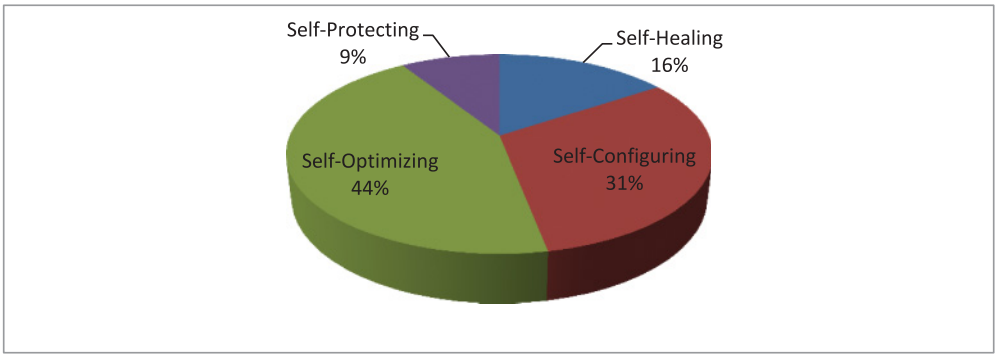

Fig. 7. Self-management in the cloud.

4.3. Quality Assessment

A quality assessment was implemented on the outstanding research articles subsequently using the criterion of inclusion and exclusion to find suitable research articles. Autonomic-cloud-computing-related research articles are included in various distinct conferences and journals. Every research article was explored for unfairness and external and internal validation of results according to CRD guidelines given by Kitchenham and Charters [2007] to provide high-quality research articles on QoS-aware autonomic resource management techniques.

4.4. Data Extraction

We faced a lot of problems in extracting suitable data when the methodical literature survey started. We contacted numerous authors to find the in-depth knowledge of research if required. The following procedure for data extraction was used in our review:

- One author extracted data from 110 research articles after in-depth review.
- Review results were cross-checked by other author on random samples.
- During cross-checking, if there were any conflicts, then a compromise meeting was called to resolve the conflict.

5. EXTRACTION OUTCOMES

The motive of this research work is to find the available research in autonomic cloud computing and is stated in Table VI in the form of research questions. Most of the research articles on autonomic cloud computing are published in a comprehensive variety of conference proceedings and journals. Figure 7 shows the percentage of research papers discussing autonomic cloud computing techniques (self-protecting, self-optimizing, self-configuring, and self-healing) from 2009 to 2015.

Thirty-five percent of the research articles were published in conferences, 48% in journals, 5% in workshops, and 12% in symposiums. The greatest percentage of research publications came from journals, followed by conferences. Figure 7 depicts the most research in the field of self-optimizing (44%), followed by self-configuring (31%), self-healing (16%), and self-protecting (9%). Figure 8 describes the percentage of research papers that consider different QoS parameters (execution time, scalability, cost, response time, energy, deadline, throughput, availability, security, and resource utilization) from 2009 to 2015. Figure 8 depicts that cost is used as a QoS parameter in most research papers (20%), while only 2% of research papers used scalability and 2% used security as a QoS parameter. Mostly resource utilization (17%), execution time

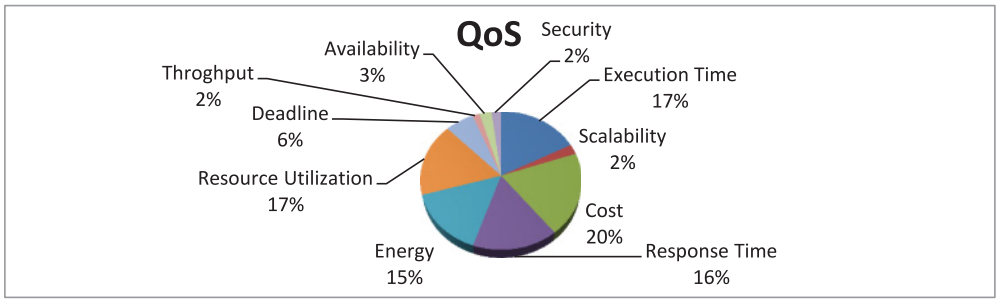

Fig. 8. QoS parameters considered in autonomic cloud computing.

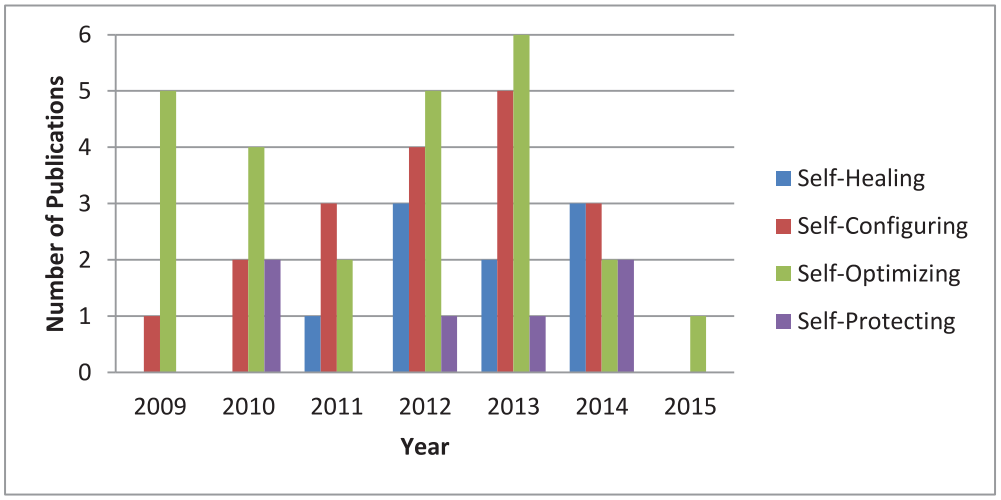

Fig. 9. Time count of self-management in the cloud.

(17%), response time (16%), and energy (15%) were used as QoS parameters in research papers.

The systematic map in Figure 9 helps in recognizing the important areas of self-management that were highlighted from 2009 to 2015.

6. PHASES OF AUTONOMIC RESOURCE MANAGEMENT IN CLOUD

Categorization of autonomic cloud computing systems along with their characteristics is presented in this section. Based on the literature, we have categorized autonomic resource management of the cloud into six phases: (1) design of application, (2) workload scheduling, (3) allocation, (4) monitoring, (5) self-management, and (6) QoS parameters, as shown in Figure 10. Further, we explain the characteristics of every component in detail.

6.1. Design of Application

Data-intensive or computation-intensive applications are executed in the cloud environment, but the system performs better if applications run in parallel. To improve the performance of autonomic cloud computing systems, there is a need for dynamic composition of applications based on the configuration of the system and requirements of users [Singh and Chana 2015b]. As shown in Figure 11, the components used in

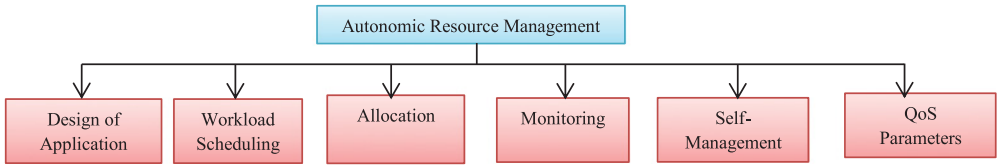

Fig. 10. Phases of autonomic resource management.

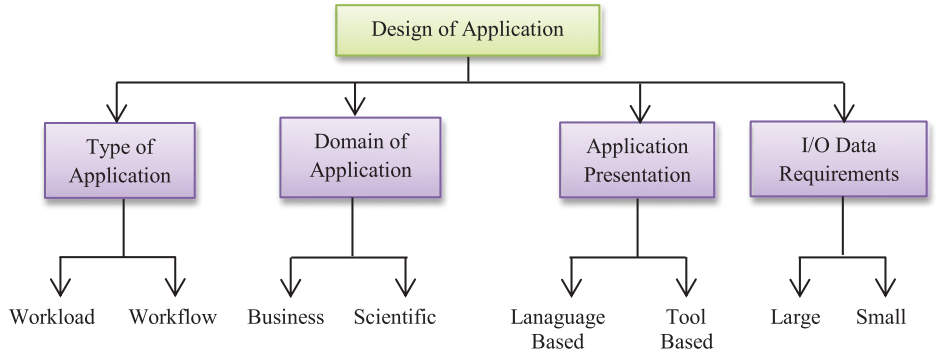

Fig. 11. Taxonomy based on design of application.

the design of the application are (1) type of application, (2) domain of application, (3) application presentation, and (4) I/O data requirements.

6.1.1. Type of Application. Application in the cloud consists of several tasks (set of instructions executed on computing resource). Cloud applications have been divided into two types based on relations among tasks: (1) workflow and (2) workload (homogenous or heterogeneous). *Workload* is an abstraction of work of that instance or set of instances going to be performed—for example, running a web service or being a Hadoop data node is a valid workload. *Workflow* is a term used to describe the set of interrelated tasks and their distribution among different available resources for better resource provisioning. Directed Acyclic Graphs are used to present the workflow in terms of nodes (tasks) and vertices (relationship among tasks) [Xu et al. 2012].

6.1.2. Domain of Application. Presently, engineers are developing scientific applications to manage resources and user requests in an efficient manner, but the complexity of applications is increasing. The cloud is an emerging platform to manage complex *scientific* applications. In the scientific domain, the computing system fulfills the requirements of various applications like *business* applications, HTC (High-Throughput Computing) applications, and HPC (High-Performance Computing) applications. Cloud technology is also being used in the domain of healthcare to support business functions and provide more effective diagnostic processes. QoS requirements for healthcare applications are high throughput, availability, scalability, and so forth. In biological applications, it is very difficult to maintain large datasets because it requires extensive I/O operations [Moreno-Vozmediano et al. 2013]. To resolve this issue, there is a need for computing infrastructure that helps to maintain the datasets in an efficient manner. In geoscience applications, both geospatial and nonspatial data are collected and analyzed [Solomon et al. 2013]. Due to an increase in the volume of data, there is a need for cloud technology to process data and produce output within a certain deadline. Specifically, the GIS (Geographic Information System) is an important component of geoscience

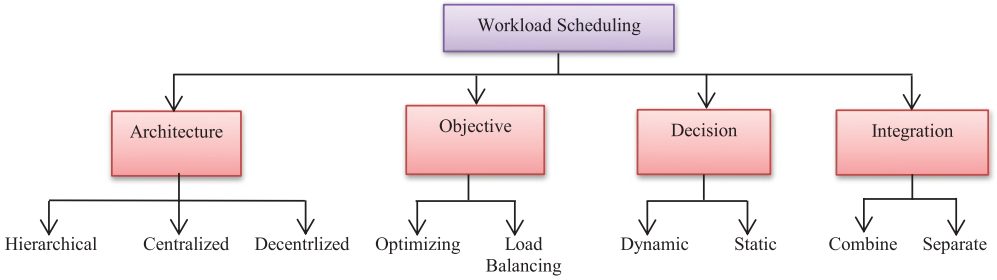

Fig. 12. Taxonomy based on workload scheduling.

applications. QoS requirements for GIS applications are security, processing speed, larger storage space, and so forth. Other applications of cloud computing include Google Docs, Dropbox, Video Encoding, Multiplayer Online Gaming, Tower Planning, and Data Analytics.

6.1.3. Application Presentation. In the cloud, basically applications are designed using data definition languages or application generation tools. XML (eXtensible Markup Language) is used to create applications in data *language*. Applications in XML can be presented in an easy manner, but this requires a proper nested structure, so it is difficult to remember the syntax. To avoid this, most cloud users use GUI (Graphical User Interface)-based *tools* for better visualization (PNs (Petri Nets)) [Amoretti et al. 2013].

6.1.4. I/O Data Requirements. Different types of data, input, intermediate, and output data, are used to manage the cloud-based application. In geoscience applications, both geospatial and nonspatial data are collected and analyzed. Due to an increase in the volume of data, there is a need for cloud technology to process data and produce output within a certain deadline. Every type of geographically referenced data is collected, stored, manipulated, and managed. Some applications need data input in sequence mode, but others need data input in parallel mode [Pandey et al. 2012]. We consider data as *large* if an application needs a massive amount of input data and data as *small* if applications need a lesser amount of data for execution [Rak et al. 2011].

6.2. Workload Scheduling

Workload scheduling is composed of two functions: resource allocation and resource mapping. The objective of resource allocation is to assign appropriate resources to the suitable workloads on time, so that applications can utilize the resources effectively [Kailasam et al. 2010]. Resource mapping is a process of workloads with appropriate resources based on the QoS requirements specified by users in terms of SLAs to minimize the cost and execution time and maximize the profit. As shown in Figure 12, the components of workload scheduling are (1) architecture, (2) objective, (3) decision, and (4) integration.

6.2.1. Architecture. For performance, autonomy, and scalability of the autonomic system, scheduling architecture is an important component [Buyya et al. 2012]. Basically, there are three types of scheduling architecture in the cloud: hierarchical, centralized, and decentralized [Mastelic et al. 2014]. In the *centralized* architecture of scheduling, the central controller makes all the decisions for all the tasks and subtasks. The scheduler processes the user request and maps appropriate resource(s) with the workload for execution, but scalability is not provided in this architecture to both resources and workloads. In the *hierarchical* architecture, there are different levels of schedulers,

that is, higher- and lower-level schedulers. The higher-level scheduler is a central controller that controls all the lower-level schedulers and reduces the complexity of execution by assigning lower-level schedulers to every small task. This type of architecture schedules resources based on different scheduling techniques, but in the case of a failure in central control, the whole computing system will fail [Abdul-Rahman et al. 2011]. In case of *decentralized* architecture of scheduling, resources are assigned to the workloads based on their individual requirements through the property of scalability. But this approach is only suitable for homogenous workloads and resources [Gergin et al. 2014].

6.2.2. Objective. The scheduler performs matchmaking (mapping of workloads to available resources) after submission of workloads by users and determines its possibility (whether the task can be provisioned on resources based on QoS requirements or not). The scheduler sends requests to the resource provider for scheduling. The scheduler releases extra resources from the resource pool based on the performance required [Saripalli et al. 2011]. The scheduler also contains information about the resources for submitting workloads and monitors the desired performance that will cause the system to either acquire or release resources (*load balancing*). To map the user workload to a corresponding cloud resource is a challenging task based on QoS requirements. Considering many QoS requirements is a necessary task for efficient resource scheduling in the cloud. The main objective of the scheduler is to schedule the workloads or tasks on available resources after mapping with minimum cost and time and maximum resource utilization (*optimizing*).

6.2.3. Decision. Different scheduling algorithms are used in the cloud to map the tasks to resources based on the QoS requirements specified by the user. Scheduling in cloud computing is of two types: static and dynamic. Matchmaking of the user workload to a particular resource based on user requirements is called *static* resource scheduling, and based on provisioning of resources, the mapping and execution of user workloads can be done, which is called *dynamic* resource scheduling. Dynamic scheduling maps the resources at runtime, which provides scalability and improves the performance by reducing wastage of resources [Mazzucco 2010].

6.2.4. Integration. The Scheduler provides the function of integration of different execution units to give the final result of execution [Mehrotra et al. 2014]. The broker is used to track the status of execution to check whether the number of resources is sufficient for the task or more are required. Based on the principles of SOA (Service-Oriented Architecture), different numbers of schedulers are used to execute different individual tasks. After successful execution of all the tasks, the broker combines the result of every scheduler to generate the final output of execution. Integration can be *separate* or *combined*.

6.3. Allocation

The process of resource allocation is controlled by a centralized agent called the Cloud Resource Manager (CRM). The CRM manages all the cloud workloads and resources and maps the resources and workloads efficiently. There are different entities and interfaces associated with the CRM. The scaling listener is used to map the workloads with appropriate resources based on QoS requirements described by users [Singh and Chana 2015d]. Generally, in resource allocation, cloud consumers submit workloads along with their QoS requirements to the cloud provider for execution. After submission, the cloud provider wants to execute workloads with minimum time, while cloud consumers want to execute with minimum execution cost. Based on QoS requirements and these constraints, the resources are provisioned from a set of resources $\{r_1, r_2, r_3, \ldots, r_n\}$

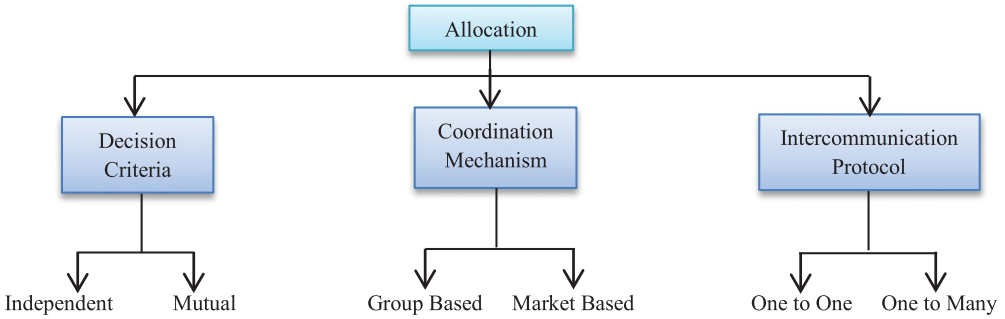

Fig. 13. Taxonomy based on allocation.

for users' workloads $\{w_1, w_2, w_3, \ldots, w_m\}$ with maximum resource utilization and customer satisfaction. Resource allocation based on workflow consists of mapping of every cloud workload to a suitable resource and permitting workloads to fulfill a particular benchmark or performance standard. Workflow-based resource scheduling determines resources and executes workloads on appropriate resources [Khargharia et al. 2008]. Efficient scheduling of workflow can improve the performance by allocation of appropriate resources. To maximize the revenue and improve user satisfaction, an effective allocation of resources is desired in the cloud environment. As shown in Figure 13, components of resource allocation are (1) decision criteria, (2) coordination mechanism, and (3) Intercommunication Protocol.

6.3.1. Decision Criteria. Decisions are made in computing systems through the interaction of autonomic components for scheduling and mapping allocation of resources [Nallur et al. 2009]. There are two types of decisions in autonomic cloud computing: mutual and independent. In the *independent* decisions scheme, every scheduler schedules the resources and executes them independently after submission of tasks in an autonomic computing system without taking care of the resource utilization status. In the *mutual* decisions scheme, all the tasks are executed based on the mutual coordination among different types of schedulers (low-level scheduler and high-level scheduler). This type of decision-making scheme reduces resource contention [Caton and Rana 2012].

6.3.2. Coordination Mechanism. For successful execution of resources, there is a need for robust coordination among distributed and dynamic entities. In the cloud, there are two types of coordination: group based and market based. In *group*-based coordination, groups are formed based on the similar requirements of different cloud users and resources are shared among those groups. Cloud providers also provide resources at the group level to fulfill user requirements, and the group-based approach provides better performance and customer satisfaction along with fault tolerance. SLA is used to make a written agreement between different groups. In the *market*-based mechanism, the concept of negotiation is used to provide resources to different cloud users through negotiated SLAs [Maggio et al. 2012]. Interested economic entities interact with each other through a virtual environment for selling and buying computing resources [Caton et al. 2013; Wu et al. 2013].

6.3.3. Intercommunication Protocol. In the cloud, two types of protocol are used for coordination among autonomic components: *one to many* and *one to one*. In *one to one*, one consumer is interacting with one provider based on a negotiated SLA, but in *one to many*, one cloud provider provides service to more than one cloud user and this consumes large network bandwidth [Casalicchio and Silvestri 2013].

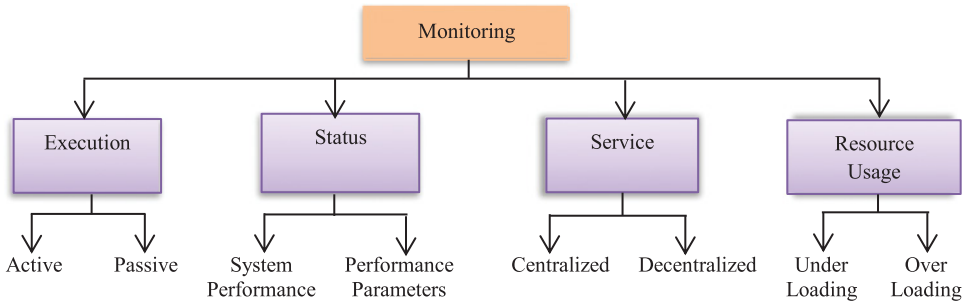

Fig. 14. Taxonomy based on monitoring.

6.4. Monitoring

Performance optimization can be best achieved by efficiently monitoring the utilization of computing resources. So, we need a comprehensive intelligent monitoring agent to analyze the performances of computing resources. A monitoring system is used to depict resource and memory utilization. Monitoring collects information about the cloud environment (load on processor, resource utilization, task execution, and status of active resources) [Sanaei et al. 2014]. Sensor-gathered information and required changes will be implemented through effectors after analysis. As shown in Figure 14, monitoring in the cloud consists of four levels: (1) execution, (2) status, (3) service, and (4) resource usage.

6.4.1. Execution. To avoid unexpected failures, the scheduler monitors the status of execution (started, queued, completed, resumed, failed) after submission of the task or workload in the cloud system [Jamshidi et al. 2014]. In the cloud, execution monitoring is of two types: passive and active. In *active* monitoring, the monitoring agent continually checks the execution of current workloads or tasks and modifies the execution procedure based on information coming from other schedulers [Paton et al. 2009]. Schedulers in this type of monitoring will send a beat automatically after a fixed time to the client about the status of execution and also inform when resources are free for other execution after successful completion of the current execution. In *passive* monitoring of execution, the local monitoring agent is used to monitor the status of execution to check whether the workload or application is executing according to QoS requirements specified by users in SLAs [Anithakumari and Sekaran 2014].

6.4.2. Status. An Autonomic Element (AE) is used to monitor the performance of the system to check the resource utilization, memory utilization, and network usage through the use of monitoring tools. AEs also monitor the *parameters* (SLA deviation, resource uptime) specified to check the *performance* of the system in order to avoid SLA violations. In case of SLA violations, the AE takes the required steps for the prevention of SLA violations [Singh and Chana 2012].

6.4.3. Service. Service monitoring collects the information about free resources and their status of resource utilization and processor load [Singh and Chana 2015a]. A service monitoring agent is used to monitor the status of execution to check whether the workload or application is executing according to QoS requirements specified by users in SLAs. Cloud providers' SLAs will give an indication of how much actual SLA deviation of service the provider views as feasible, and to what amount it is agreeable to require its own financial resources to compensate for unexpected outages. In the cloud, there are two types of service: centralized and decentralized [Mayer et al. 2015]. A *centralized* repository is used to store the monitored data in centralized service, but

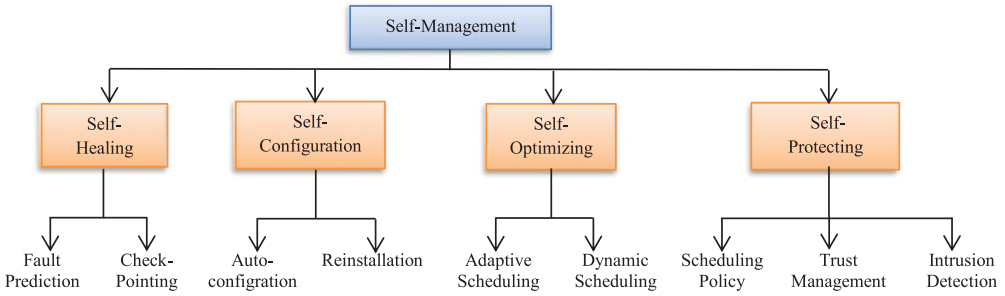

Fig. 15. Taxonomy based on self-management.

this type of service is not scaling with an increase in the number of resource providers and users. But in the case of *decentralized* service, load is balanced and fault tolerance is provided efficiently [Etchevers et al. 2011].

6.4.4. Resource Usage. The resource monitoring system collects the resource usage by measuring through-performance metrics such as resource and memory utilization to avoid *underloading* and *overloading* of resources. The cloud provider needs to retain an adequate number of resources to deliver continuous service to cloud consumers during peak load [Gill 2015]. Monitoring of resource usage is used to take care of important QoS requirements like security, availability, performance, and so forth during workload execution. There are two main aspects of monitoring of resource usage: (1) consumers want to execute their workload at minimum cost and minimum time without violation of SLAs, and (2) providers want to execute the workload with a minimum number of resources. For this, monitoring of resource usage is a vital part of resource management to measure the SLA deviation, QoS requirements, and resource usage. Monitoring of resource usage can be referred to as monitoring the performances of both physical and virtual infrastructure [Makris et al. 2013].

6.5. Self-Management

Self-management in cloud computing has four properties: (1) self-healing, (2) self-configuring, (3) self-protecting, and (4) self-optimization, as shown in Figure 15. To implement all the properties of self-management together is a difficult task, and based on the requirements and goals of an autonomic system, some of the properties can be considered.

6.5.1. Self-Healing. In cloud computing, self-healing is the capability of a system to identify, analyze, and recover from unfortunate faults automatically. This property of self-management improves performance through *fault tolerance* by reducing or avoiding the impact of failures on execution [Didona et al. 2014]. Failures can occur in the cloud for the following reasons: (1) unexpected changes in the configuration of the execution environment, (2) unavailability of resources, (3) overloading of resources, (4) shortage of memory, and (5) network failures. Autonomic systems use techniques (check-pointing, failure forecasting, and replication) to handle these failures. The *check-pointing* technique is used to transfer the failed workload or tasks to the other available resources to start the execution from the point of failure. The *failure forecasting* technique can be used to predict the requirement of resources in the future in order to avoid failure of execution. In the *replication* technique, workload is executed on more than one resource to increase the chances of successful execution [Maurer et al. 2012].

6.5.2. Self-Configuring. In cloud computing, self-configuring is the capability of a system to adapt to changes in the environment [Niehörster and Brinkmann 2011].

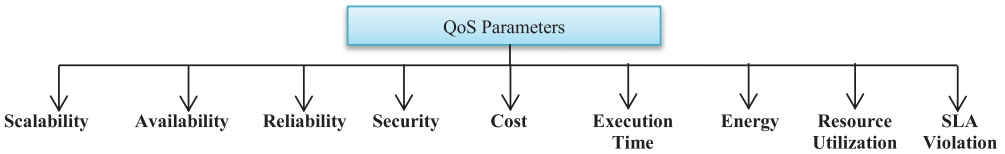

Fig. 16. Taxonomy based on QoS parameters.

Self-configuring in autonomic systems consists of the *installation* of missed or outdated components based on an alert generated by the system without human intervention. Some components may be *reinstalled* in changing conditions [Etchevers et al. 2011].

6.5.3. Self-optimizing. In cloud computing, self-optimizing is the capability of a system to improve the performance. Dynamic scheduling techniques are using in the cloud to map the tasks or workloads on appropriate resources [Beloglazov et al. 2012]. *Dynamic* scheduling continually checks the status of execution and improves the system performance based on the feedback given by the autonomic element. For data-intensive applications, *adaptive* scheduling is used, which can be easily adapted in changed environments [Grozev and Buyya 2014; Xiao and Boutaba 2005].

6.5.4. Self-protecting. In cloud computing, self-protecting is the capability of a system to protect against intrusions and threats. To maintain the security and integrity of a system, it is required to detect and protect the autonomic system from malicious attack [Mencagli et al. 2014]. To achieve this property of self-management, secure *scheduling policies* should be provided on both sides (provider side and user side). Security policies should be required in which the system should be shut down before a strong attack happens [Qu et al. 2010]. In the *trust management* system approach, malicious attackers can be detected through behavioral auditing. In the *intrusion detection* technique, attacks are continually monitored and analyzed by the system to avoid future attacks [Vieira et al. 2014; Maximilien and Singh 2004].

6.6. QoS Parameters

Cloud-based systems consider different QoS parameters to design a successful system [Casalicchio et al. 2013]. From the literature, we identified eight types of QoS parameters (scalability, availability, reliability, security, cost, time, energy, SLA violation and resource utilization) used in autonomic cloud computing [Simonin et al. 2013; Chihi et al. 2013; Pietro et al. 2013; Addis et al. 2010; Ayadi et al. 2013] systems, as shown in Figure 16.

Scalability is the capability of a computing system to maintain the performance while increasing the number of users or resource usage in order to fulfill the requirements of users. The system should be able to produce the correct results when the load is increased. *Availability* is the ability of a system to ensure the data is available with the desired level of performance in normal as well as in fatal situations excluding scheduled downtime. *Reliability* is the capability of a system to perform consistently according to its predefined objectives. *Security* is the ability to protect the data stored on the cloud by using data encryption and passwords. *Energy* is the amount of energy consumed by a resource to finish the execution of the workload. *Execution time* is the time required to execute the workload completely. *Cost* is the amount of cost that can be spent in one hour for the execution of workload. *Resource utilization* is a ratio of actual time spent by the resource to execute the workload to the total uptime of the resource for a single resource. *SLA violation* is the possibility of defilement of the Service-Level Agreement.

Table VII. Summary of QoS-Aware Autonomic Resource Management Techniques

Technique	Description	Author	Organization
CBR	Case-Based Reasoning	Maurer et al. [2013]	University of Manchester, UK
ASP	Application Service Provider	Cardellini et al. [2011]	University of Roma, Italy
COCCUS	self-COngured, Cost-based Cloud qUery Services	Konstantinou et al. [2013]	University in Athens, Greece
CAS	Cloud Auto Scaling	Mao et al. [2010]	University of Virginia, USA
AWM	Autonomic Workload Manager	Sah et al. [2014]	Pokhra University, Nepal
AMF	Autonomic Management Framework	Khargharia et al. [2008]	University of Arizona, USA
SH-SLA	Self-Healing SLA	Mosallanejad et al. [2014]	Universiti Putra Malaysia, Malaysia
ARAS-M	Autonomic Resource Allocation Strategy based on Market Mechanism	You et al. [2011]	Hangzhou Dianzi University, China
BN-DSS	Bayesian Network-based Decision Support System	Bashar [2013]	Prince Mohammad Bin Fahd University, Saudi Arabia
SNOOZE	Self-Organizing and Healing	Feller et al. [2012]	INRIA Rennes - Bretagne Atlantique, France
DeSVi	Detecting SLA Violation infrastructure	Emeakaroha et al. [2012]	Vienna University of Technology, Vienna, Austria
AFTRC	Adaptive Fault Tolerance in Real-time Cloud	Malik et al. [2011]	INRIA - Sophia Antipolis, France
CoTuner	Coordinated Self-Configuration of Virtual Machines	Bu et al. [2013]	Wayne State University, USA
AROMA	Automated Resource allocation and cOnfiguration of MApReduce	Lama and Zhou [2012]	University of Colorado, USA
HTC	High-Throughput Cluster computing system	Kijsipongse and Vannarat [2010]	National Electronics and Computer Technology Center, Bangkok, Thailand
SHAPE	Self-Healing And self-Protection Environment	Singh and Singh [2014]	Thapar University, India

7. QoS-AWARE AUTONOMIC RESOURCE MANAGEMENT TECHNIQUES IN THE CLOUD

In this section, a survey on selected QoS-aware autonomic resource management techniques is conducted and taxonomy (*as discussed in the previous section in Figure 10*) is mapped to the key characteristics of autonomic resource management techniques to find the gaps in existing research. A summary of selected QoS-aware autonomic resource management techniques is presented in Table VII. A comparison of techniques and their classification based on taxonomy is described in Tables VIII through XIII.

The Case-Based Reasoning (CBR) technique [Maurer et al. 2013] uses human-based interaction to make an agreement between users and providers called an SLA for successful execution of workloads by considering resource utilization and scalability as QoS requirements. In this system, various elastic levels are defined, and a control loop is used to enable autonomic computing in the virtual environment. A knowledge base is used to store the rules used in decision making after monitoring of data (real and synthetic workloads) for resource configuration. This system executes in four steps:

Table VIII. Taxonomy Based on Design of Application

Technique	Type of Application	Domain of Application	Application Presentation	I/O Requirement
CBR	Workload	Scientific	Language Based	Large
ASP	Workflow	Business	Tool Based	Small
COCCUS	Workload	Scientific	Language Based	Small
CAS	Workload	Scientific	Tool Based	Small
AWM	Workload	Business	Tool Based	Small
AMF	Workload	Business	Language Based	Large
SH-SLA	Workflow	Scientific	Language Based	Small
ARAS-M	Workload	Scientific	Tool Based	Large
BN-DSS	Workload	Business	Tool Based	Small
SNOOZE	Workload	Scientific	Tool Based	Small
DeSVi	Workflow	Scientific/ Business	Language Based	Large
AFTRC	Workflow	Scientific	Language Based	Small
CoTuner	Workload	Scientific/ Business	Tool Based	Large
AROMA	Workflow	Scientific	Tool Based	Small
HTC	Workload	Business	Language Based	Small
SHAPE	Workload	Scientific	Tool Based	Large

Table IX. Taxonomy Based on Workload Scheduling

Technique	Architecture	Objective	Decision	Integration
CBR	Hierarchical	Optimizing	Dynamic	Combine
ASP	Centralized	Optimizing	Static	Separate
COCCUS	Centralized	Load Balancing	Static	Separate
CAS	Decentralized	Optimizing/Load Balancing	Dynamic	Combine
AWM	Centralized	Load Balancing	Dynamic	Combine
AMF	Hierarchical	Optimizing	Static	Combine
SH-SLA	Hierarchical	Load Balancing	Dynamic/Static	Separate
ARAS-M	Centralized	Optimizing	Static	Separate
BN-DSS	Decentralized	Optimizing	Dynamic	Combine
SNOOZE	Hierarchical	Optimizing/Load Balancing	Dynamic	Combine
DeSVi	Decentralized	Optimizing/Load Balancing	Static	Separate
AFTRC	Centralized	Load Balancing	Static	Combine
CoTuner	Centralized	Optimizing	Dynamic/Static	Combine
AROMA	Decentralized	Load Balancing	Dynamic	Separate
HTC	Centralized	Optimizing	Dynamic	Combine
SHAPE	Centralized	Optimizing	Dynamic	Combine

(1) retrieve the most similar case, (2) solve the problem through a similar case, (3) revise the solution, and (4) store the key features of the solution in knowledge database for future use. SLA violations and resource utilization are improved in this autonomic system. Application Service Provider (ASP) [Cardellini et al. 2011] uses WSDL (Web Service Description Language) and Web Interface (HTTP) to design proactive and reactive heuristic policies to get an optimal solution. All the important QoS parameters are mentioned in the SLA document. In this autonomic system, performance history

Table X. Taxonomy Based on Allocation

Technique	Agreement	Allocation Mechanism	Intercommunication Protocol
CBR	Independent	Group Based	One to One/One to Many
ASP	Independent	Group Based	One to One
COCCUS	Independent	Group Based	One to Many
CAS	Mutual	Market Based	One to Many
AWM	Independent	Market Based	One to One
AMF	Mutual	Market Based	One to One/One to Many
SH-SLA	Mutual	Group Based	One to One
ARAS-M	Independent	Market Based	One to Many
BN-DSS	Independent	Group Based	One to One
SNOOZE	Independent	Group Based	One to One/One to Many
DeSVi	Mutual	Group Based	One to Many
AFTRC	Mutual	Group Based	One to One
CoTuner	Independent	Group Based	One to Many
AROMA	Mutual	Group Based	One to One
HTC	Independent	Market Based	One to One
SHAPE	Independent	Group Based	One to Many

Table XI. Taxonomy Based on Monitoring

Technique	Execution	Status	Service	Resource Usage
CBR	Active	System Performance/ Performance Parameters	Centralized	Underloading/ Overloading
ASP	Passive	System Performance	Centralized	Underloading
COCCUS	Passive	System Performance/ Performance Parameters	Centralized	Underloading
CAS	Active	System Performance	Decentralized	Underloading/ Overloading
AWM	Passive	System Performance/ Performance Parameters	Centralized	Overloading
AMF	Passive	System Performance	Decentralized	Overloading
SH-SLA	Active	System Performance	Centralized	Underloading/ Overloading
ARAS-M	Passive	Performance Parameters	Centralized	Underloading
BN-DSS	Active	System Performance/ Performance Parameters	Decentralized	Underloading
SNOOZE	Passive	Performance Parameters	Centralized	Underloading/ Overloading
DeSVi	Passive	Performance Parameters	Decentralized	Overloading
AFTRC	Active	Performance Parameters	Centralized	Overloading
CoTuner	Active	System Performance/ Performance Parameters	Centralized	Overloading
AROMA	Active	Performance Parameters	Decentralized	Underloading
HTC	Passive	System Performance	Centralized	Underloading
SHAPE	Active	Performance Parameters	Centralized	Underloading/ Overloading

is used to resolve the alerts generated at runtime due to some QoS parameters. ASP provides the feature of load balancing and VM allocation at runtime through the use of a fully controlled autonomic loop. In this system, lease cost and SLA violations are reduced.

The Self-COngured, Cost-based Cloud qUery Services (COCCUS) technique [Konstantinou et al. 2013] uses a centralized architecture to provide the query-based facility in which users can ask queries regarding scheduling policies, priorities, and

Table XII. Taxonomy Based on Self-Management

Technique	Self-Healing	Self-Configuration	Self-Optimization	Self-Protecting
CBR	Fault Prediction	Auto-configuration/ Reinstallation	Adaptive Scheduling	Scheduling Policy
ASP	NA	Auto-configuration	Adaptive Scheduling	NA
COCCUS	Check-Pointing	Auto-configuration/ Reinstallation	Dynamic Scheduling	NA
CAS	NA	Auto-configuration	Dynamic Scheduling	Trust Management
AWM	Fault Prediction	Auto-configuration	Adaptive Scheduling	Scheduling Policy
AMF	NA	Auto-configuration/ Reinstallation	Adaptive Scheduling/ Dynamic Scheduling	NA
SH-SLA	Fault Prediction	NA	Dynamic Scheduling	NA
ARAS-M	NA	NA	Adaptive Scheduling/ Dynamic Scheduling	NA
BN-DSS	Fault Prediction	Auto-configuration	Adaptive Scheduling	NA
SNOOZE	Fault Prediction/ Check-Pointing	Auto-configuration/ Reinstallation	Dynamic Scheduling	Trust Management
DeSVi	NA	Auto-configuration/ Reinstallation	Adaptive Scheduling	NA
AFTRC	Fault Prediction/ Check-Pointing	NA	Dynamic Scheduling	NA
CoTuner	NA	Auto-configuration	Dynamic Scheduling	NA
AROMA	NA	Auto-configuration	Adaptive Scheduling	Scheduling Policy
HTC	Fault Prediction	Auto-configuration/ Reinstallation	Adaptive Scheduling/ Dynamic Scheduling	Scheduling Policy
SHAPE	Fault Prediction	NA	Dynamic Scheduling	Intrusion Detection

budget information. Cloud DBMS is used to store the information about the scheduling policies and user queries for further use. The main objectives of COCCUS are to (1) get and execute the user queries, (2) store the queries in the data structure, and (3) minimize the maintenance cost of query execution. The Cloud Auto Scaling (CAS) technique [Mao et al. 2010] schedules activities of VM instance startup and shutdown automatically to improve the performance. CAS enables users to finish the execution of workloads or tasks within their deadline with minimum cost. The Windows Azure Platform is used to implement this autonomic system. CAS contains four components: history repository, performance monitor, VM manager, and auto-scaling decider. The performance monitor checks the processing time, execution time, and arrival time of the workload. The history repository is used to store particular information about the workload. The VM manager maintains a relation between cloud providers and the auto-scaling mechanism and executes a plan of the auto-scaling decider to finish the execution of workloads [Singh and Chana 2013c]. The Autonomic Workload Manager (AWM) technique [Sah et al. 2014] uses DPSMS (Distributed Provisioning and

Table XIII. Taxonomy Based on QoS Parameters

Technique	Scalability	Availability	Reliability	Security	Cost	Time	Energy	Resource Utilization	SLA Violation
CBR	√			√	√	√		√	√
ASP					√				√
COCCUS					√	√			
CAS	√			√	√	√			
AWM	√			√		√		√	
AMF						√	√		
SH-SLA		√	√					√	√
ARAS-M					√				
BN-DSS	√					√			
SNOOZE	√			√	√	√		√	
DeSVi					√	√			√
AFTRC		√	√						
CoTuner								√	
AROMA				√	√	√		√	
HTC				√		√			
SHAPE		√	√	√	√	√		√	

Scaling Decision Making System) to distribute the workloads on resources based on their common QoS characteristics. AWM divides resources into two categories: coarse-grained and fine-grained resources. AVM allocates the resources based on minimum response time and high throughput. This autonomic system executes in three steps: (1) allocate resources to workloads, (2) minimize execution time, and (3) check execution status (if it executes within time and budget, then it continues execution; otherwise, provide more resources). The Autonomic Management Framework (AMF) [Khargharia et al. 2008] uses an autonomic mechanism of performance and power management theoretically. This system consists of three modules: managed system, component interface, and autonomic manager. In the managed system module, AMF continually monitors, analyzes, and executes the adequate actions to maintain the appropriate level of performance. This component interface defines procedures to measure performance at runtime and includes the operation port, control port, function port, and configuration port. The autonomic manager executes all the workloads on adequate resources with minimum execution time and cost.

The Self-Healing SLA (SH-SLA) technique [Mosallanejad et al. 2014] is an autonomic system designed to enable hierarchical self-healing that monitors the SLA SLA violations and takes necessary steps to prevent SLA violations. SLAs with similar agreements will interact with each other to notify the status of execution [Singh and Chana 2013c]. SLA-SH consists of four steps: (1) specify SLA, (2) negotiate based on SLA, (3) monitor the SLA, and 4) react according to SLA deviation. The Autonomic Resource Allocation Strategy based on Market Mechanism (ARAS-M) technique [You et al. 2011] uses a market-based mechanism to allocate the resources to workloads based on QoS requirements specified by users. In this autonomic system, the Genetic Algorithm (GA) is used to attain an equilibrium state by adjusting the price automatically. This system fulfills the demand of every workload along with QoS requirements.

The Bayesian Network-based Decision Support System (BN-DSS) [Bashar 2013] provides autonomic scaling of utility computing resources. The BN-DSS system studies the historical behavior of the autonomic system and predicts the performance, applicability, and feasibility based on this historical data and negotiates the SLA [Leite et al. 2014]. Structural learning is used to study the behavior of the system.

The Self-Organizing and Healing (SNOOZE) technique [Feller et al. 2012] uses hierarchical architecture to allocate the resources for the workloads in the virtual environment. SNOOZE works in three layers: physical layer, hierarchical layer, and client layer. The local controller is used to control the nodes and machines are structured in clusters in the physical layer, group leaders and group managers are used in the hierarchical layer for clustering of fault-tolerant components, and the user interface is provided through the client layer. The design of SNOOZE is very simple and scalability is achieved.

The Detecting SLA Violation infrastructure (DeSVi) technique [Emeakaroha et al. 2012] uses a resource monitoring mechanism to prevent the violation of SLAs. DeSVi allocates the resources of the workloads in the virtual environment, and resources are monitored by mapping user-defined SLAs with low-level resource metrics. Service-level objectives have been defined to detect the violation in SLAs and resource utilization. DeSVi executes transactional web applications and image rendering applications that contain heterogeneous workloads and monitors consumption of resources. The Adaptive Fault Tolerance in Real time Cloud (AFRTC) [Malik et al. 2011] system is used to detect the faults, provide fault tolerance, and calculate the reliability of nodes to make decisions. Reliability of nodes in the virtual environment is changing adaptively [Singh and Chana 2013b]. A node is reliable if a virtual node produces results within a specified deadline; otherwise, the node is not reliable. The node will be removed if it fails continually and a new node will be added. AFRTC uses backward recovery if any node does not achieve the required level of reliability [Carpen-Amarie 2011]. The Coordinated Self-Configuration of Virtual Machines (CoTuner) technique [Bu et al. 2013] uses a model-free hybrid reinforcement learning technique to enable coordination among applications and virtual resources. CoTuner works based on knowledge-guided exploration policies to design a methodology for autonomic configuration of resources in case of fluctuation of workloads. Coordinated auto-configuration is achieved by using a simplex-based space reduction technique.

The Automated Resource allocation and cOnfiguration of MApReduce (AROMA) technique [Lama and Zhou 2012] allocates resources to workloads based on QoS requirements specified by cloud users. AROMA enables configuration of jobs of Hadoop automatically to compare the value of resource utilization with already executed workloads. Machine-learning techniques are used to make the provisioning decisions to reduce execution time. The High-Throughput Cluster (HTC) computing system [Kijsipongse and Vannarat 2010] is an extension of rocks clusters to extend the local cluster to remote resources of the cloud transparently and securely. HTC works based on dynamic provisioning mechanisms, that is, the job scheduling policy in which the database is updated regularly when a new node is added or removed. Updated information will be distributed to all other nodes for the purpose of synchronization. The Self-Healing And self-Protection Environment (SHAPE) technique [Singh and Singh 2014] is an autonomic system to recover from various faults (hardware, software, and network faults) and protect from security attacks (DDoS, R2L, U2L, and probing attacks). SHAPE is based on component-based architecture, in which new components can be added or removed easily. Open-source technologies are used to implement this autonomic system. A comparison of QoS-aware autonomic resource management techniques is presented in Table XIV.

8. QOS-AWARE AUTONOMIC RESOURCE MANAGEMENT IN THE CLOUD: A PERSPECTIVE MODEL

The ability to improve allocation of resources and utilization of resources to fulfill the QoS requirements of cloud users is called self-management. Important aspects like workload management and resource utilization are mandatory to consider for such a

Table XIV. Comparison of QoS-Aware Autonomic Resource Management Techniques

Technique	Objective Function	Merits	Demerits
CBR	Reduce execution time and cost	CPU time and SLA violations are reduced	Unable to handle large problems
ASP	Improves negotiation time for SLAs	Reduce SLA violations and lease cost	Decision delay
COCCUS	Reduce maintenance cost	Cost reduced	Unable to handle multiple workflows and problem of starvation
CAS	Optimize resource utilization and cost	Average CPU time and cost reduced	Not suitable for network-intensive applications
AWM	Reduce monetary cost	Makespan reduced	Does not considered penalty cost and compensation
AMF	Reduce job's execution time	Execution time and network traffic reduced	Lack of stabilized API
SH-SLA	Reduce SLA deviation	SLA violations reduced	Does not consider heterogeneous workloads
ARAS-M	Reduce energy consumption	Resource uptime reduced	Failure prediction not measured accurately
BN-DSS	Improves resource utilization	Resource contention reduced	Starvation
SNOOZE	Improves computational capacity	Time, scalability, and cost improved	Only single provider considered
DeSVi	Reduce resource consumption	Resource contention reduced	SLA violations, does not consider penalty cost and compensation
AFTRC	Improve resource utilization	Energy consumption reduced	Avg. decision time larger
CoTuner	Reduce power consumption	No. of deadlines missed, execution time, and cost reduced	For workloads, sensitivity of weight calculation not considered
AROMA	Improve resource utilization and execution time	Execution time reduced	SLA violation
HTC	Reduce power consumption	Energy consumption and execution time reduced	Cost is larger
SHAPE	Improve security, execution, time and cost	Time and cost reduced and availability, reliability, and security improved	Self-optimization and self-configuration not considered

characteristic. The utility function is mostly used for self-management of resources [Melendez et al. 2013; You et al. 2013; Xu et al. 2014]. Automatic management of the performance of a system is a complex issue due to unpredictability of resources. The main research issues in this context are as follows [Singh and Chana 2015c, 2015d]:

a) No service provider provides all the cloud services required to implement autonomic service management.
b) Except AWS (Amazon Web Services), no service provider provides integrated autonomic services, but AWS provides service with a low degree of customization.

IaaS (Infrastructure as a Service)-based AWS (Amazon Web Service) currently delivers cloud-based integrated autonomic services to cloud users [Amazon Web Services 2013; Jackson et al. 2010; Madduri et al. 2013; Iqbal et al. 2014]. Users interact with the Amazon cloud through the Internet from any geographical location, as shown in Figure 17. On the provider side, Amazon uses Web Server, Elastic Compute Cloud (EC2) Instance Web Server, Amazon Simple Queue Service (SQS), Simple Database,

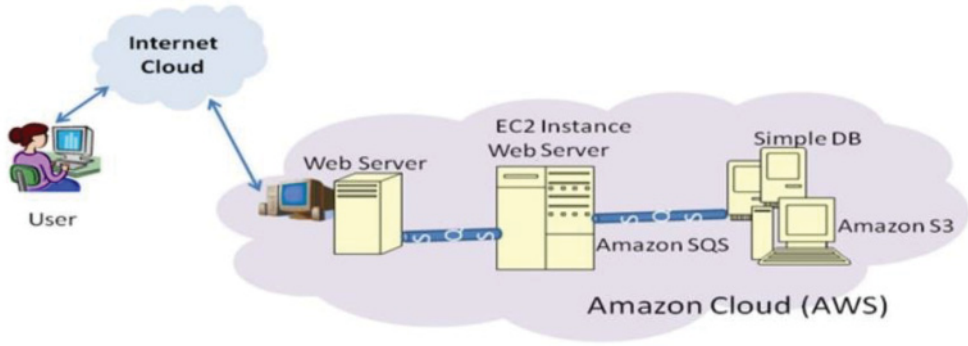

Fig. 17. Cloud-based integrated autonomic AWS.

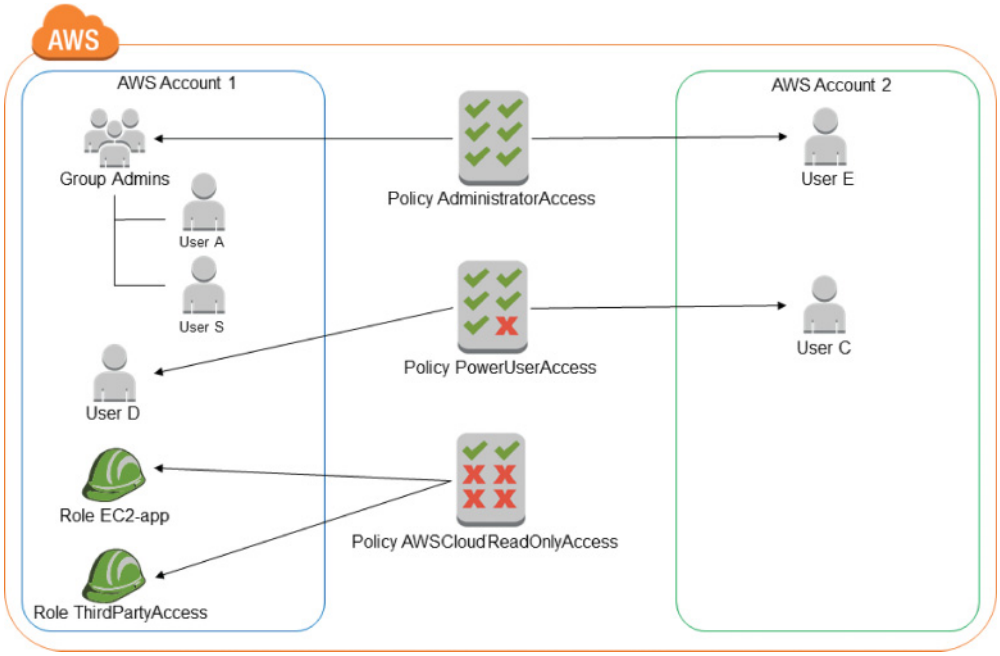

Fig. 18. Use case of autonomic AWS.

and Amazon S3 (Simple Storage Service) to provide service in a managed and effective manner to cloud users.

AWS provides integrated autonomic services to the different cloud users but with a very low degree of customization, as shown in Figure 18. Figure 18 shows that there are two different AWS accounts in which five users are using services provided by AWS with different rights. Three types of user groups using the AWS are (1) the administrator (who is mainly responsible for creating, delivering, maintaining, ensuring, and removing service), (2) the power user (who has fewer rights as compared to the administrator and is mainly responsible for delivering, maintaining, and ensuring service), and (3) the end-user or third party (who can only use the service).

The degree of customization of AWS decreases from the administrator to the end-user. Due to less customization at the end-user level, AWS is not very effective and has difficulty fulfilling the QoS requirements as described by users in the SLA. Thus, there is a need to automatically manage QoS requirements of cloud users, thus helping the cloud providers in achieving the SLAs and avoiding SLA violations. To resolve this problem, there is need for a QoS-aware autonomic resource management technique that uses the best hardware and software configurations to optimize both QoS targets: user centric (budget and execution time) and resource centric (reliability, availability, and utilization).

The following advantages can be achieved in the future:

- By providing a GUI-based web service by AWS, users can easily specify their service requirements, so AWS will be easy to use.
- By delivering service with a high degree of customization, users can create their service specifications by selecting the database, web application platform, programming language, operating system, and other services according to their requirements, so AWS will more flexible.
- To make AWS much more effective, users have to pay only for using resources like storage, compute power, and so forth for a required time without a long-term contract.
- By incorporating a virtual backbone (global computing infrastructure), AWS will become reliable.
- Scalability of service can be improved by using AWS tools (elastic load balancing and auto-scaling) in which applications can be easily scaled down or up based on requirements.
- AWS will become secure by using an E2E approach including software, operational, and physical measures.

To resolve the aforementioned issues of autonomic cloud computing, there is a need for a QoS-aware autonomic resource management mechanism. QoS-aware autonomic management is done on two levels: global and local, as shown in Figure 19. On the global level, cloud consumers interact with the system to submit their workload or application for execution along with QoS requirements pertaining to the SLA. The task of execution of the workload or application is divided into subtasks or small levels called local levels. The actual execution of the workload or application is performed on the local level after verification of availability of resources. The knowledge pool is used to store the predefined rules defined by the system administrator, and the rules will be updated from time to time based on new polices of resource allocation.

QoS is the ability to assume a desired level of performance based on the QoS requirements described by consumers. The SLA is an authorized document that specifies QoS requirements stated by cloud consumers in written form [Singh and Chana 2015a]. QoS-aware autonomic resource management is based on the architecture of autonomic systems defined in Figure 2, in which the AE is an agent that accepts the input and produces final results based on QoS parameters defined in the SLA. It contains four phases (Monitor, Analyze, Plan, and Execute) of the autonomic system as discussed in *Section 3.1*. QoS-aware autonomic resource management systems are composed of AEs and an AM. The AM is an intelligent agent that interacts with the environment through manageability interfaces (*Sensors* and *Effectors*) and takes actions according to the input received from sensors and rules defined in the knowledge base in low-level language. Sensors perform two functions: resource discovery and resource monitoring [Singh and Chana 2015b]. The AM is configured by the administrator based on alerts and actions in high-level language.

Resource discovery is able to find adequate (with minimum cost and execution time) resources for the workload or application based on QoS requirements through

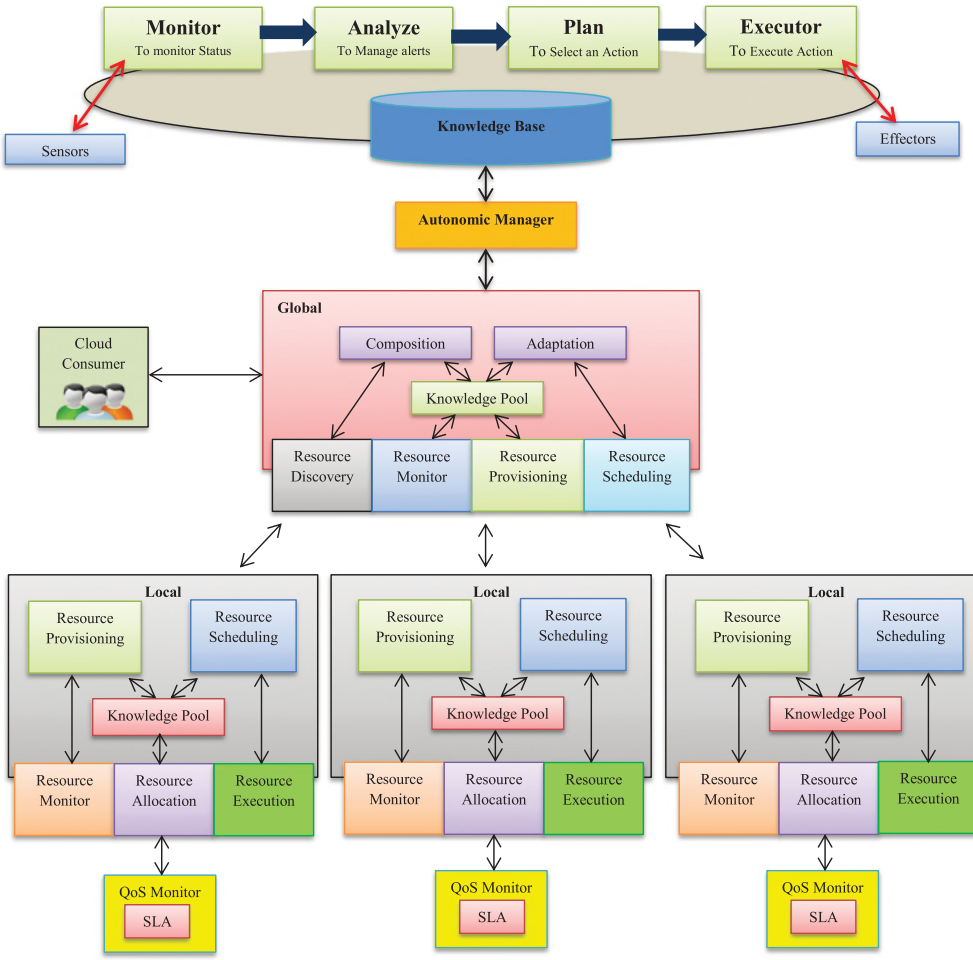

Fig. 19. QoS-aware autonomic resource management in the cloud.

Composition. Composition is able to determine the best resource/workload pair for execution. The resource monitor is able to monitor that the conditions are being fulfilled as specified in the policy and ensures that all the resources are provided. All the inputs received from monitors will be analyzed and action will be taken according to the alert generated by the QoS monitor.

The QoS monitor is used to verify whether all the QoS attributes defined in the SLA are fulfilled or not by using *Adaptation*. If they are not fulfilled, then the QoS monitor will generate an alert to provide more resources to fulfill the current demand of the application. The adaptation function is able to maintain effective execution in case of an abrupt change in QoS conditions. Based on QoS requirements and the policy of the system, resources are provisioned to the workload or application and *Resource Provisioning* information will be sent to users for verification [Singh and Chana 2015c]. After successful verification by users, the AE allocates resources to workload(s) for *Resource Scheduling* [Singh and Chana 2015d]. The executor performs the final step of resource execution and completes the execution within a specified deadline, and the system will be continuing for other workloads or applications.

9. FUTURE RESEARCH DIRECTIONS IN QOS-AWARE AUTONOMIC CLOUD COMPUTING

Though a lot of progress has been achieved and scalable computing infrastructures are easily implemented by cloud computing on pay-per-use basis, still there are many issues and challenges in this field that need to be addressed. The following research directions have been identified from the existing literature [Mosallanejad et al. 2014; Malik et al. 2011; Maurer et al. 2013; You et al. 2011; Lama and Zhou 2012; Khargharia et al. 2008; Kijsipongse and Vannarat 2010; Bashar 2013; Mao et al. 2010; Konstantinou et al. 2013; Bu et al. 2013; Sah et al. 2014; Cardellini et al. 2011; Singh and Singh 2014; Feller et al. 2012; Emeakaroha et al. 2012; Viswanathan et al. 2011; Hossny et al. 2012] of QoS-aware autonomic cloud computing:

a) *Service-Level Agreements*: There is a need for autonomic cloud infrastructures to fulfill the QoS requirements described by cloud users in terms of SLA and to reduce interactions between cloud consumers and the computing environment. Therefore, an effective strategy to detect SLA violations in advance is a research issue that can avoid performance degradation.
b) *Autonomic Resource Provisioning*: Self- or autonomic management implies the fact that the service is able to self-manage as per its environment. An autonomic management system is required for dynamic resource provisioning to fulfill the QoS requirements as described by cloud users and to reduce service costs and improve efficiency of the system. Cloud computing is an effective platform to execute web-based services on a pay-as-you-go basis, but due to larger variation in user demand, it is difficult to provision resources effectively.
c) *Quality of Service:* To fulfill the QoS requirements of cloud services, a required amount of resources are provisioned by service providers. Based on these QoS requirements, SLAs are designed and SLA violations are detected regularly, which further decides the penalty or compensation in case of SLA violations. Thus, there is need to provision an adequate amount of resources dynamically by service providers to reduce or avoid SLA violations.
d) *Resource Scheduling*: The challenges of resource scheduling include dispersion, uncertainty, and heterogeneity of resources that are not resolved with traditional resource management mechanisms in cloud environments. Thus, there is a need to make cloud services and cloud-oriented applications efficient by taking care of these properties in the cloud environment. The aim of resource scheduling is to allocate appropriate resources at the right time to the right workloads so those applications can utilize the resources effectively. In other words, the amount of resources should be minimum for a workload to maintain a desirable level of service quality or maximize throughput (or minimize workload completion time) of a workload. To address this problem, new solutions need to be developed.

Open research challenges existing in the field of QoS-aware autonomic cloud computing include the following:

- Due to the large number of user requests in multiple service queues, cloud-based autonomic systems are lacking in user interactions with the system, and more QoS parameters like bandwidth, storage, CPU power, and so forth can be added to test the performance of the system in a better way.
- Autonomic systems consider only functional requirements (input and output) in SLAs; nonfunctional requirements (QoS) should be written into SLAs for better agreement and performance. Important QoS requirements in autonomic research management include scalability, reliability, security, execution time, cost, and availability.

- Cloud-based autonomic systems can be extended by adding an optimized query plan (store queries in data structure and execute based on their priorities decided based on QoS requirements), which will improve user satisfaction.
- Presently, autonomic computing systems work as single-tier architecture. To achieve better performance in terms of deadlines, job distribution, availability, and other important QoS parameters, the cloud auto-scaling architecture should be shifted to a multitier application environment.
- A tremendous amount of energy is consumed by these data centers, which leads to high operational costs and contributes toward carbon footprints to the environment. So, a cloud-based autonomic system can be further extended by adding energy efficiency as a QoS parameter, which will further reduce the impact on the environment.
- The utility of stochastic, predictive, and heuristic approaches in terms of overhead and runtime complexity can be investigated, which is not discussed in existing cloud-based autonomic systems.
- Immediate decisions in autonomic systems can be improved by using optimized data mining or machine-learning techniques. As a result, autonomic systems keep the computing system stable in unpredictable conditions and adapt quickly in new environmental conditions like software and hardware failures.
- Cloud-based autonomic systems can be extended by adding heterogeneous resources in the mechanism of resource allocation by adding more infrastructure providers, which will be effective to serve more user requests and fulfill user requirements (QoS).
- Scalability, storage management, and load balancing of autonomic systems can be improved by using a distributed NoSQL database because these databases have a high recall rate and precision rate.
- Cloud-based autonomic systems can be extended by using multiple data centers instead of a single data center to improve scalability and availability, and the knowledge database can be used to track; further, availability of multiple data centers reduces SLA violations.
- The reliability of autonomic systems can be improved by adding proactive schedulers and using scheduling decisions based on infrastructure and network characteristics using rule-based policies in which systems make decision automatically.
- Search time to find an optimal configuration in cloud-based autonomic systems can be improved through machine-learning techniques. As a result, based on similar QoS requirements, workload can be clustered and executed on the same set of resources simultaneously.
- Performance of cloud-based autonomic systems can be improved by adding execution time and cost-based resource scheduling policies for self-optimization to improve resource utilization. Reducing cost will be profitable for users, and reducing the execution time will be beneficial for providers.
- Autonomic systems can be extended by adding component-level testing through replication and a check-pointing-based approach to improve the reliability of the system. The check-pointing technique is used to transfer the failed workload or tasks to the other available resources to start the execution from the point of failure. In the replication technique, workload is executed on more than one resource to increase the chances of successful execution.

To overcome these challenges, a resource management technique should be developed that provisions and schedules the cloud resources as per the user requirements (QoS). The resource management technique should be self-managing (autonomic) so as to adapt itself at runtime and would help in mitigating SLA violations and in reducing costs. For example, if the SLA requires auto-scaling and performance, based on

SLAs, urgent cloud workloads would be placed in priority queue for earlier execution by allocating the reserve resources automatically. Dynamic scalability needs to be considered in autonomic techniques to handle situations like overloading and underloading. Possible future research directions are as follows:

- Further research in the area of autonomic cloud computing based on various QoS parameters is an open issue.
- The real impact of SLA is still questionable. SLA violations need to be detected during autonomic resource provisioning and execution in cloud computing.
- It is very difficult to find the most suitable resource for specific workloads for effective autonomic resource management. For efficient resource management in autonomic cloud computing, there is a need to find the main reasons for detection of workloads and resources for better mappings in the future.
- Different QoS parameters have to be reassessed to implement the autonomic resource management mechanisms for the given QoS parameters.
- Workloads need to be executed automatically so as to be scalable and flexible, and to avoid overloading and underloading.
- There is also a need to test the QoS-aware resource management on a real cloud environment. Based on existing research, we found that autonomic management of resources is an open research issue.

10. DISCUSSION

We reviewed 110 research articles in this research work and presented them in a systematic and categorized manner. Existing surveys by Buyya et al. [2012] and Rahman et al. [2011] reflect various research issues. The initial void for future work has been filled by these surveys in the domain of autonomic computing. This survey article presents the most recent research work related to autonomic cloud computing and augments the previous surveys. A systematic technique has been used to develop an evolution of autonomic cloud computing that identifies *Quality of Service (QoS)* and *Focus of Study (FoS)* parameters in self-management. We explored autonomic cloud computing in detail and compared the QoS-aware autonomic techniques based on important aspects of self-management. We recognized the research issues addressed and open challenges still unresolved in autonomic cloud computing. Furthermore, we have found key discoveries in autonomic cloud computing that occurred after 2008. In this research article, the existing research is presented in chronological order in different sections, which makes it easy for perspective readers and authors to find the latest research done after 2009. Key outcomes of our methodical literature review are discussed in this section along with weakness and strengths of the evidence.

This methodical analysis has suggestions for prospective research scholars who are already working in this area and looking for new ideas, and for professional experts working in cloud-based services providing enterprises who want to use different QoS-aware autonomic resource management techniques for improving cloud services. Many open issues are presented for professional experts and prospective researchers. Self-management is an evolving field of research in the cloud. Existing research authenticates that there is inconsistency between the provider and user as to mapping the workload with appropriate resources without violating SLAs. There is a need for a certified autonomic QoS-based resource scheduling framework to overcome the time-consuming process of manual mapping of workloads with adequate resource; the research should be focused on every type of autonomic cloud computing technique based on scheduling criteria and objective functions. Then, this authenticated framework would help to build the foundation for further research in the area of autonomic cloud computing, which can be used in the industry and research.

For supporting autonomic computing in the cloud environment, the following major research challenges need to be solved:

- Due to difficulty in predicting the behavior (in terms of QoS requirements) and demand (in terms of resources) of the workload/application, there is a need for an effective QoS-aware autonomic resource management technique that can easily make the right decisions regarding dynamic scaling of resources and workloads/applications.
- In the present scenario, it is very difficult to make the autonomic resource management technique cost effective and energy efficient due to increasing energy requirements and operating costs. To resolve this problem, there is a need for an autonomic system that uses the best hardware and software configurations to improve resource utilizations and fulfill important QoS parameters.
- There is a need to develop an autonomic resource management technique that optimizes both QoS targets: user centric (budget and execution time) and resource centric (reliability, availability, and utilization).
- Most of the cloud companies have a large number of resources to serve the different business applications and a large amount of data, but it is difficult to migrate and manage due to security and privacy issues related to interoperability and integration.
- The QoS-aware autonomic resource management technique needs to be decentralized. In centralized distributed systems, it is very difficult to manage the large number of user requests in multiple service queues, which further leads to performance degradation (decreases reliability and scalability).

This research depicts a broad methodical literature analysis of autonomic cloud computing techniques to find research gaps for future research.

11. CONCLUSIONS

This research article presents a methodical survey on autonomic cloud computing (ACC). The taxonomy of ACC has been presented based on six different perceptions: (1) design of application, (2) workload scheduling, (3) allocation, (4) monitoring, (5) self-management, and (6) QoS requirements. After this, a taxonomy of every perspective is presented based on different characteristics of autonomic resource management. Further, a survey on QoS-aware autonomic resource management is conducted and a taxonomy mapped to the key characteristics of QoS-aware autonomic resource management to find the gaps in existing research. This survey helps to find the research gaps existing in autonomic cloud computing, and research issues still existing have been identified. We summarized the existing literature in the form of systematic evolution of autonomic cloud computing. To know the impact of QoS requirements on self-management, there is a need to understand the evolution of autonomic cloud computing to determine whether the provisioned resources are scheduled efficiently or not. The current research on autonomic cloud computing is more focused on self-optimizing and self-healing aspects. In order to provide protection and incorporate dynamic scalability in autonomic cloud computing, ACCSs are required to focus on self-configuring and self-protecting policies. The following facts can be further concluded:

- Contrast and assessment of autonomic cloud computing in the cloud can aid in selecting the autonomic scheduling algorithm based on a workload's QoS requirements.
- QoS parameters can be improved in the delivered cloud service if resources are reserved in advance.
- Proper matching of workload and resource can improve the performance significantly.
- Allocation of resources based on type of workload (homogenous and heterogeneous) can improve the resource utilization.

REFERENCES

Omar A. Rahman, Masaharu Munetomo, and Kiyoshi Akama. 2011. Multi-level autonomic architecture for the management of virtualized application environments in cloud platforms. In *Proceedings of the IEEE International Conference on Cloud Computing (CLOUD'11)*. IEEE, 754–755. DOI:http://dx.doi.org/10.1109/CLOUD.2011.58

Bernardetta Addis, Danilo Ardagna, Barbara Panicucci, and Li Zhang. 2010. Autonomic management of cloud service centers with availability guarantees. In *Proceedings of the IEEE 3rd International Conference on Cloud Computing (CLOUD'10)*. IEEE, 220–227. DOI:http://dx.doi.org/10.1109/CLOUD.2010.19

Amazon Web Services. 2013. Amazon EC2 instances. Retrieved from http://aws.amazon.com/ec2/instance-types/.

Michele Amoretti, Francesco Zanichelli, and Gianni Conte. 2013. Efficient autonomic cloud computing using online discrete event simulation. *J. Parallel Distrib. Comput.* 73, 6 (2013), 767–776. DOI:http://dx.doi.org/10.1016/j.jpdc.2013.02.008

S. Anithakumari and K. Chandra Sekaran. 2014. Autonomic SLA management in cloud computing services. In *Recent Trends in Computer Networks and Distributed Systems Security*. Springer, Berlin, 151–159. DOI:http://dx.doi.org/10.1007/978-3-642-54525-2_13

Danilo Ardagna, Barbara Panicucci, Marco Trubian, and Li Zhang. 2012. Energy-aware autonomic resource allocation in multitier virtualized environments. *IEEE Trans. Services Comput.* 5, 1 (2012), 2–19. DOI:http://dx.doi.org/10.1109/TSC.2010.42

Marcos D. Assunção, Rodrigo N. Calheiros, Silvia Bianchi, Marco A. S. Netto, and Rajkumar Buyya. 2015. Big Data computing and clouds: Trends and future directions. *J. Parallel Distrib. Comput.* 79 (2015), 3–15. DOI:http://dx.doi.org/10.1016/j.jpdc.2014.08.003

Ines Ayadi, Noemie Simoni, and Gladys Diaz. 2013. QoS-aware component for cloud computing. In *Proceedings of the 9th International Conference on Autonomic and Autonomous Systems (ICAS'13)*. 14–20. Retrieved from https://www.thinkmind.org/index.php?view=article&articleid=icas_2013_1_30_20051.

Abul Bashar. 2013. Autonomic scaling of cloud computing resources using BN-based prediction models. In *Proceedings of the IEEE 2nd International Conference on Cloud Networking (CloudNet'13)*. IEEE, 200–204. DOI:http://dx.doi.org/10.1109/CloudNet.2013.6710578

Thomas Beach, Omer Rana, Yacine Rezgui, and Manish Parashar. 2015. Governance model for cloud computing in building information management. *IEEE Trans. Services Comput.* 8, 2 (2015), 314–327. DOI:http://dx.doi.org/10.1109/TSC.2013.50

Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. 2012. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Comput. Syst.* 28, 5 (2012), 755–768. http://dx.doi.org/10.1016/j.future.2011.04.017

Nicolas Bonvin, Thanasis G. Papaioannou, and Karl Aberer. 2011. Autonomic SLA-driven provisioning for cloud applications. In *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 434–443. DOI:http://dx.doi.org/10.1109/CCGrid.2011.24

Sara Bouchenak. 2010. Automated control for SLA-aware elastic clouds. In *Proceedings of the 5th International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks*. ACM, 27–28. DOI:http://dx.doi.org/10.1145/1791204.1791210

Ivona Brandic and Scharam Dustdar. 2011. Grid vs cloud - A technology comparison. *Inf. Technol.: Methods Appl. Comput. Sci. Inf. Technol.* 53, 4 (2011), 173–179. DOI:http://dx.doi.org/10.1524/itit.2011.0640

Ivan Breskovic, Michael Maurer, Vincent C. Emeakaroha, Ivona Brandic, and Schahram Dustdar. 2011. Cost-efficient utilization of public SLA templates in autonomic cloud markets. In *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing (UCC'11)*. IEEE, 229–236. DOI:http://dx.doi.org/10.1109/UCC.2011.38

Xiangping Bu, Jia Rao, and Cheng-Zhong Xu. 2013. Coordinated self-configuration of virtual machines and appliances using a model-free learning approach. *IEEE Trans. Parallel Distrib. Syst.* 24, 4 (2013), 681–690. DOI:http://dx.doi.org/10.1109/TPDS.2012.174

Rajkumar Buyya, Rodrigo N. Calheiros, and Xiaorong Li. 2012. Autonomic cloud computing: Open challenges and architectural elements. In *Proceedings of the 3rd International Conference on Emerging Applications of Information Technology (EAIT'12)*. IEEE, 3–10. DOI:http://dx.doi.org/10.1109/EAIT.2012.6407847

Rajkumar Buyya, Saurabh K. Garg, and Rodrigo N. Calheiros. 2011. SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In *Proceedings of the International Conference on Cloud and Service Computing (CSC'11)*. IEEE, 1–10. DOI:http://dx.doi.org/10.1109/CSC.2011.6138522

Valeria Cardellini, Emiliano Casalicchio, Francesco Lo Presti, and Luca Silvestri. 2011. SLA-aware resource management for application service providers in the cloud. In *Proceedings of the 1st*

International Symposium on Network Cloud Computing and Applications (NCCA'11). IEEE, 20–27. DOI:http://dx.doi.org/10.1109/NCCA.2011.11

Alexandra Carpen-Amarie. 2011. Towards a self-adaptive data management system for cloud environments. In *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum (IPDPSW'11)*. IEEE, 2077–2080. DOI:http://dx.doi.org/10.1109/IPDPS.2011.381

Emiliano Casalicchio, Daniel A. Menascé, and Arwa Aldhalaan. 2013. Autonomic resource provisioning in cloud systems with availability goals. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*. ACM, 1–10. DOI:http://dx.doi.org/10.1145/2494621.2494623

Emiliano Casalicchio and Luca Silvestri. 2013. Mechanisms for SLA provisioning in cloud-based service providers. *Comput. Networks* 57, 3 (2013), 795–810. DOI:http://dx.doi.org/10.1016/j.comnet.2012.10.020

Simon Caton, Ivan Breskovic, and Ivona Brandic. 2013. A conceptual framework for simulating autonomic cloud markets. In *Cloud Computing*. Springer International Publishing, 92–102. DOI:http://dx.doi.org/10.1007/978-3-319-03874-2_10.

Simon Caton and Omer Rana. 2012. Towards autonomic management for cloud services based upon volunteered resources. *Concurrency and Computation: Practice and Experience* 24, 9 (2012), 992–1014. DOI:http://dx.doi.org/10.1002/cpe.1715

Inderveer Chana and Sukhpal Singh. 2014. Quality of service and service level agreements for cloud environments: Issues and challenges. In *Cloud Computing-Challenges, Limitations and R&D Solutions*. Springer International Publishing, 51–72. DOI:http://dx.doi.org/10.1007/978-3-319-10530-7_3

Hanen Chihi, Walid Chainbi, and Khaled Ghedira. 2013. An energy-efficient self-provisioning approach for cloud resources management. *ACM SIGOPS Operating Syst. Rev.* 47, 3 (2013), 2–9. DOI:http://dx.doi.org/10.1145/2553070.2553072

Inderpreet Chopra and Maninder Singh. 2014. SHAPE—An approach for self-healing and self-protection in complex distributed networks. *J. Supercomp.* 67, 2 (2014), 585–613. DOI:http://dx.doi.org/10.1007/s11227-013-1019-3

Defense Advanced Research Projects Agency. 1997. Retrieved from http://www.darpa.mil/sto/strategic/suosas.html.

Diego Didona, Paolo Romano, Sebastiano Peluso, and Francesco Quaglia. 2014. Transactional auto scaler: Elastic scaling of replicated in-memory transactional data grids. *ACM Trans. Auton. Adapt. Syst.* 9, 2 (2014), 1–32. DOI:http://dx.doi.org/10.1145/2620001

Vincent C. Emeakaroha, Marco A. S. Netto, Rodrigo N. Calheiros, Ivona Brandic, Rajkumar Buyya, and César A. F. De Rose. 2012. Towards autonomic detection of SLA violations in Cloud infrastructures. *Future Generation Comput. Syst.* 28, 7 (2012), 1017–1029. DOI:http://dx.doi.org/10.1016/j.future.2011.08.018

D. Cenk Erdil. 2013. Autonomic cloud resource sharing for intercloud federations. *Future Generation Comput. Syst.* 29, 7 (2013), 1700–1708. DOI:http://dx.doi.org/10.1016/j.future.2012.03.025

Xavier Etchevers, Thierry Coupaye, Fabienne Boyer, and Noel De Palma. 2011. Self-configuration of distributed applications in the cloud. In *Proceedings of the IEEE International Conference on Cloud Computing (CLOUD'11)*. IEEE, 668–675. DOI:http://dx.doi.org/10.1109/CLOUD.2011.65

Xavier Etchevers, Thierry Coupaye, Fabienne Boyer, Noel De Palma, and Gwen Salaun. 2011. Automated configuration of legacy applications in the cloud. In *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing (UCC'11)*. IEEE, 170–177. DOI:http://dx.doi.org/10.1109/UCC.2011.32

Farah Fargo, Cihan Tunc, Youssif Al-Nashif, and Salim Hariri. 2013. Autonomic performance-per-watt management (APM) of cloud resources and services. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*. ACM, 1–10. DOI:http://dx.doi.org/10.1145/2494621.2494624

M. N. Faruk and D. Sivakumar. 2014. Towards self-configured multi-agent resource allocation framework for cloud computing environments. *Int. J. Eng. Technol.* 6, 2 (2014), 1–10. Retrieved from http://www.enggjournals.com/ijet/docs/IJET14-06-02-201.pdf.

Eugen Feller, Louis Rilling, and Christine Morin. 2012. SNOOZE: A scalable and autonomic virtual machine management framework for private clouds. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID'12)*. IEEE, 482–489. DOI:http://dx.doi.org/10.1109/CCGrid.2012.71

Stefano Ferretti, Vittorio Ghini, Fabio Panzieri, Michele Pellegrini, and Elisa Turrini. 2010. QoS–aware clouds. In *Proceedings of the IEEE 3rd International Conference on Cloud Computing (CLOUD'10)*. 321–328. DOI:http://dx.doi.org/10.1109/CLOUD.2010.17

Saurabh K. Garg, Srinivasa K. Gopalaiyengar, and Rajkumar Buyya. 2011. SLA-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter. In *Algorithms and Architectures for Parallel Processing*. Springer, Berlin, 371–384. DOI:http://dx.doi.org/10.1007/978-3-642-24650-0_32

Ian Gergin, Bradley Simmons, and Marin Litoiu. 2014. A decentralized autonomic architecture for performance control in the cloud. In *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E'14)*. IEEE, 574–579. DOI:http://dx.doi.org/10.1109/IC2E.2014.75

Sukhpal S. Gill. 2015. Autonomic Cloud Computing: Research Perspective. 1–3. Retrieved from http://arxiv.org/ftp/arxiv/papers/1507/1507.01546.pdf

Nikolay Grozev and Rajkumar Buyya. 2014. Multi-cloud provisioning and load distribution for three-tier applications. *ACM Trans. Auton. Adapt. Syst.* 9, 3 (2014), 1–21. DOI:http://dx.doi.org/10.1145/2662112

Ibrahim A. T. Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. 2015. The rise of "big data" on cloud computing: Review and open research issues. *Inform. Syst.* 47 (2015), 98–115. DOI:http://dx.doi.org/10.1016/j.is.2014.07.006

Paul Horn. 2001. Autonomic computing: IBM's perspective on the state of information technology. *Technical Report, IBM Corporation.* IBM, 1–38. Retrieved from http://people.scs.carleton.ca/~soma/biosec/readings/autonomic_computing.pdf.

Eman Hossny, Sara Salem, and Sherif M. Khattab. 2012. Towards automated user-centric cloud provisioning: Job provisioning and scheduling on heterogeneous virtual machines. In *Proceedings of the 8th International Conference on Informatics and Systems (INFOS'12)*. IEEE, 18–24. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6236557.

Markus C. Huebscher and Julie A. McCann. 2008. A survey of autonomic computing—degrees, models, and applications. *ACM Comput. Surv. (CSUR)* 40, 3 (2008), 1–7. DOI:http://dx.doi.org/10.1145/1380584.1380585

Waheed Iqbal, Matthew N. Dailey, and Diego Carrera. 2014. Low cost quality aware multi-tier application hosting on the Amazon cloud. In *Proceedings of the International Conference on Future Internet of Things and Cloud (FiCloud'14)*. IEEE, 202–209. DOI:http://doi.ieeecomputersociety.org/10.1109/FiCloud.2014.40.

Keith R. Jackson, Lavanya Ramakrishnan, Krishna Muriki, Shane Canon, Shreyas Cholia, John Shalf, Harvey J. Wasserman, and Nicholas J. Wright. 2010. Performance analysis of high performance computing applications on the Amazon web services cloud. In *Proceedings of the IEEE 2nd International Conference on Cloud Computing Technology and Science (CloudCom'10)*. IEEE, 159–168. DOI:http://dx.doi.org/10.1109/CloudCom.2010.69

Pooyan Jamshidi, Aakash Ahmad, and Claus Pahl. 2014. Autonomic resource provisioning for cloud-based software. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 95–104. DOI:http://dx.doi.org/10.1145/2593929.2593940

Sriram Kailasam, Nathan Gnanasambandam, Dharanipragada Janakiram, and Naveen Sharma. 2010. Optimizing service level agreements for autonomic cloud bursting schedulers. In *Proceedings of the In ICPP Workshops*. 285–294. DOI:http://dx.doi.org/10.1109/ICPPW.2010.54

Jeffrey O. Kephart and William E. Walsh. 2003. An architectural blueprint for autonomic computing. *Technical Report, IBM Corporation* IBM, 1–29. Retrieved from http://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf.

Attila Kertesz, Gabor Kecskemeti, and Ivona Brandic. 2011. Autonomic SLA-aware service virtualization for distributed systems. In *Proceedings of the 19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP'11)*. IEEE, 503–510. DOI:http://dx.doi.org/10.1109/PDP.2011.17

Bithika Khargharia, Salim Hariri, and Mazin S. Yousif. 2008. Autonomic power and performance management for computing systems. *Cluster Comput.* 11, 2 (2008), 167–181. DOI:http://dx.doi.org/10.1109/ICAC.2006.1662393

Ekasit Kijsipongse and Sornthep Vannarat. 2010. Autonomic resource provisioning in rocks clusters using eucalyptus cloud computing. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*. ACM, 61–66. DOI:http://dx.doi.org/10.1145/1936254.1936265

Hyunjoo Kim, Manish Parashar, David J. Foran, and Lin Yang. 2009. Investigating the use of autonomic cloudbursts for high-throughput medical image registration. In *Proceedings of 10th IEEE/ACM International Conference on Grid Computing*. 34–41. DOI:http://dx.doi.org/10.1109/GRID.2009.5353065

Hyunjoo Kim, Yaakoub el-Khamra, Shantenu Jha, and Manish Parashar. 2009. An autonomic approach to integrated HPC grid and cloud usage. In *Proceedings of the 5th IEEE International Conference on e-Science (e-Science'09)*. 366–373. DOI:http://dx.doi.org/10.1109/e-Science.2009.58

Barbara Kitchenham and Stuart Charters. 2007. Guidelines for performing systematic literature reviews in software engineering. *Technical Report EBSE-2007-01.* 1–44. Retrieved from http://userpages.uni-koblenz.de/~laemmel/esecourse/slides/slr.pdf.

Ioannis Konstantinou, Verena Kantere, Dimitrios Tsoumakos, and Nectarios Koziris. 2013. COCCUS: Self-configured cost-based query services in the cloud. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 1041–1044. DOI:http://dx.doi.org/10.1145/2463676.2465233

DeKyriakos Kritikos, Barbara Pernici, Pierluigi Plebani, Cinzia Cappiello, Marco Comuzzi, Salima Benrernou, Ivona Brandic, Attila Kertész, Michael Parkin, and Manuel Carro. 2013. A survey on service quality description. *ACM Comput. Surv.* 46, 1 (2013), 1–58. DOI:http://dx.doi.org/10.1145/2522968.2522969

Palden Lama and Xiaobo Zhou. 2012. AROMA: Automated resource allocation and configuration of mapreduce environment in the cloud. In *Proceedings of the 9th International Conference on Autonomic Computing.* ACM, 63–72. DOI:http://dx.doi.org/10.1145/2371536.2371547

Jason Lango. 2014. Toward software-defined SLAs. *Commun. ACM* 57, 1 (2014), 54–60. DOI:http://dx.doi.org/10.1145/2541883.2541894

Alessandro F. Leite, Tainá Raiol, Claude Tadonki, Maria Emilia M. T. Walter, Christine Eisenbeis, and Alba Cristina Magalhães Alves de Melo. 2014. EXCALIBUR: An autonomic cloud architecture for executing parallel applications. In *Proceedings of the 4th International Workshop on Cloud Data and Platforms.* ACM, 1–6. DOI:http://dx.doi.org/10.1145/2592784.2592786

Wenrui Li, Pengcheng Zhang, and Zhongxue Yang. 2012. A framework for self-healing service compositions in cloud computing environments. In *Proceedings of the 19th IEEE International Conference on In Web Services (ICWS'12)*. IEEE, 690–691. DOI:http://dx.doi.org/10.1109/ICWS.2012.109

Ravi K. Madduri, Paul Dave, Dinanath Sulakhe, Lukasz Lacinski, Bo Liu, and Ian T. Foster. 2013. Experiences in building a next-generation sequencing analysis service using galaxy, globus online and Amazon web service. In *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery.* ACM, 1–3. DOI:http://dx.doi.org/10.1145/2484762.2484827

Martina Maggio, Henry Hoffmann, Alessandro V. Papadopoulos, Jacopo Panerati, Marco D. Santambrogio, Anant Agarwal, and Alberto Leva. 2012. Comparison of decision-making strategies for self-optimization in autonomic computing systems. *ACM Trans. Auton. Adapt. Syst.* 7, 4 (2012) 1–32. DOI:http://dx.doi.org/10.1145/2382570.2382572

Prodromos Makris, Dimitrios N. Skoutas, and Charalabos Skianis. 2013. A survey on context-aware mobile and wireless networking: On networking and computing environments' integration. *IEEE Commun. Surv. Tutorials* 15, 1 (2013), 362–386. DOI:http://dx.doi.org/10.1109/SURV.2012.040912.00180

Sheheryar Malik and Fabrice Huet. 2011. Adaptive fault tolerance in real time cloud computing. In *Proceedings of the IEEE World Congress on Services (SERVICES'11).* IEEE, 280–287. DOI:http://dx.doi.org/10.1109/SERVICES.2011.108

Wayne S. Mandak and Charles A. Stowell. 2000. Dynamic assembly for system adaptability, dependability and assurance (DASADA) project analysis. *PhD dissertation.* Naval Postgraduate School, Monterey, California. Retrieved from http://calhoun.nps.edu/bitstream/handle/10945/10926/ADA393486.pdf?sequencThis.

Ming Mao, Jie Li, and Marty Humphrey. 2010. Cloud auto-scaling with deadline and budget constraints. In *Proceedings of the 11th IEEE/ACM International Conference on Grid Computing (GRID'10).* IEEE, 41–48. DOI:http://dx.doi.org/10.1109/GRID.2010.5697966

Patrick Martin, Andrew Brown, Wendy Powley, and Jose Luis Vazquez-Poletti. 2011. Autonomic management of elastic services in the cloud. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC'11).* IEEE, 135–140. DOI:http://dx.doi.org/10.1109/ISCC.2011.5984006

Toni Mastelic, Ariel Oleksiak, Holger Claussen, Ivona Brandic, Jean-Marc Pierson, and Athanasios V. Vasilakos. 2014. Cloud computing: Survey on energy efficiency. *ACM Comput. Surv.* 47, 2 (2014), 1–36. DOI:http://dx.doi.org/10.1145/2656204

Michael Maurer, Ivona Brandic, and Rizos Sakellariou. 2011. Enacting SLAs in clouds using rules. In *Euro-Par 2011 Parallel Processing*. Springer, Berlin, 455–466. DOI:http://dx.doi.org/10.1007/978-3-642-23400-2_42

Michael Maurer, Ivona Brandic, and Rizos Sakellariou. 2012. Self-adaptive and resource-efficient SLA enactment for cloud computing infrastructures. In *Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD'12).* IEEE, 368–375. DOI:http://dx.doi.org/10.1109/CLOUD.2012.55

Michael Maurer, Ivona Brandic, and Rizos Sakellariou. 2013. Adaptive resource configuration for cloud infrastructure management. *Future Generation Comput Syst.* 29, 2 (2013), 472–487. DOI:http://dx.doi.org/10.1016/j.future.2012.07.004

E. Michael Maximilien and Munindar P. Singh. 2004. Toward autonomic web services trust and selection. In *Proceedings of the 2nd International Conference on Service Oriented Computing.* ACM, 212–221. DOI:http://dx.doi.org/10.1145/1035167.1035198

Philip Mayer, José Velasco, Annabelle Klarl, Rolf Hennicker, Mariachiara Puviani, Francesco Tiezzi, Rosario Pugliese, Jaroslav Keznikl, and Tomáš Bureš. 2015. The autonomic cloud. In *Software Engineering for Collective Autonomic Systems*. Springer International Publishing, 495–512. DOI: http://dx.doi.org/10.1007/978-3-319-16310-9_16

Michele Mazzucco. 2010. Towards autonomic service provisioning systems. In *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE, 273–282. DOI: http://dx.doi.org/10.1109/CCGRID.2010.125

Rajat Mehrotra, Srishti Srivastava, Ioana Banicescu, and Sherif Abdelwahed. 2014. An interaction balance based approach for autonomic performance management in a cloud computing environment. In *Adaptive Resource Management and Scheduling for Cloud Computing*. Springer International Publishing, 52–70. DOI: http://dx.doi.org/10.1007/978-3-319-13464-2_5

Jose O. Melendez, Anshuman Biswas, Shikharesh Majumdar, Biswajit Nandy, Marzia Zaman, Pradeep Srivastava, and Nishith Goel. 2013. A framework for automatic resource provisioning for private clouds. In *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'13)*. IEEE, 610–617. DOI: http://dx.doi.org/10.1109/CCGrid.2013.91

Gabriele Mencagli, Marco Vanneschi, and Emanuele Vespa. 2014. A cooperative predictive control approach to improve the reconfiguration stability of adaptive distributed parallel applications. *ACM Trans. Auton. Adapt. Syst.* 9, 1, (2014), 1–27. DOI: http://dx.doi.org/10.1145/2567929

Mohamed Mohamed, Mourad Amziani, Djamel Belaïd, Samir Tata, and Tarek Melliti. 2014. An autonomic approach to manage elasticity of business processes in the Cloud. *Future Generation Comput. Syst.* 50, (2014), 49–61. DOI: http://dx.doi.org/10.1016/j.future.2014.10.017

Rafael Moreno-Vozmediano, Rubén S. Montero, and Ignacio M. Llorente. 2013. Key challenges in cloud computing: Enabling the future internet of services. *IEEE Internet Comput.* 17, 4 (2013), 18–25. DOI: http://dx.doi.org/10.1109/MIC.2012.69

Ahmad Mosallanejad, Rodziah Atan, Masrah Azmi Murad, and Rusli Abdullah. 2014. A hierarchical self-healing SLA for cloud computing. *Int. J. Digital Information Wireless Commun. (IJDIWC)* 4, 1 (2014), 43–52. DOI: http://dx.doi.org/10.17781/P001082

Nicola Muscettola, P. Pandurang Nayak, Barney Pell, and Brian C. Williams. 1998. Remote agent: To boldly go where no AI system has gone before. *Artific. Intell.* 103, 1 (1998), 5–47. DOI: http://dx.doi.org/10.1016/S0004-3702(98)00068-X

Vivek Nallur, Rami Bahsoon, and Xin Yao. 2009. Self-optimizing architecture for ensuring quality attributes in the cloud. In *Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (WICSA/ECSA'09)*. IEEE, 281–284. DOI: http://dx.doi.org/10.1109/WICSA.2009.5290820

Ripal Nathuji, Aman Kansal, and Alireza Ghaffarkhah. 2010. Q-Clouds: Managing performance interference effects for QoS-aware clouds. In *Proceedings of the European Conference on Computer Systems (EUROSYS'10)*. 237–250. DOI: http://dx.doi.org/10.1145/1755913.1755938

Oliver Niehörster and André Brinkmann. 2011. Autonomic resource management handling delayed configuration effects. In *Proceedings of the IEEE 3rd International Conference on Cloud Computing Technology and Science (CloudCom'11)*. IEEE, 138–145. IEEE. DOI: http://dx.doi.org/10.1109/CloudCom.2011.28

Suraj Pandey, William Voorsluys, Sheng Niu, Ahsan Khandoker, and Rajkumar Buyya. 2012. An autonomic cloud environment for hosting ECG data analysis services. *Future Generation Comput. Syst.* 28, 1 (2012), 147–154. DOI: http://dx.doi.org/10.1016/j.future.2011.04.022

Giuseppe Papuzzo and Giandomenico Spezzano. 2011. Autonomic management of workflows on hybrid grid-cloud infrastructure. In *Proceedings of the 7th International Conference on Network and Services Management*. International Federation for Information Processing, 230–233. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6104004.

Manish Parashar and Salim Hariri. 2005. Autonomic grid computing. In *Proceedings of International Conference on Autonomic Computing*. 1–10. DOI: http://users.cs.cf.ac.uk/O.F.Rana/barcelona-ac-course/autonomic-computing-intro.pdf

Norman Paton, Marcelo A. T. De Aragão, Kevin Lee, Alvaro A. A. Fernandes, and Rizos Sakellariou. 2009. Optimizing utility in cloud computing through autonomic workload execution. *Bull. Technical Committee on Data Eng.* 32, 1 (2009), 51–58. Retrieved from http://www.cs.man.ac.uk/~rizos/papers/bde09.pdf.

Roberto D. Pietro, Flavio Lombardi, Fabio Martinelli, and Daniele Sgandurra. 2013. Anticheetah: An autonomic multi-round approach for reliable computing. In *Proceedings of the IEEE 10th International Conference on Ubiquitous Intelligence and Computing, 2013, and Autonomic and Trusted Computing (UIC/ATC'13)*. IEEE, 371–379. DOI: http://dx.doi.org/10.1109/UIC-ATC.2013.77

Guangzhi Qu, Osamah A. Rawashdeh, and Dunren Che. 2010. Self-protection against attacks in an autonomic computing environment. *IJ Comput. Appl.* 17, 4 (2010), 250–256. Retrieved from http://www2.cs.siu.edu/~dche/publications/ijca10_qu.pdf.

Andres Quiroz, Hyunjoo Kim, Manish Parashar, Nathan Gnanasambandam, and Naveen Sharma. 2009. Towards autonomic workload provisioning for enterprise grids and clouds. In *Proceedings of the 10th IEEE/ACM International Conference on Grid Computing*. IEEE, 50–57. DOI:http://dx.doi.org/10.1109/GRID.2009.5353066

Mustafizur R. Rahman, Rajiv Ranjan, Rajkumar Buyya, and Boualem Benatallah. 2011. A taxonomy and survey on autonomic management of applications in grid computing environments. *Concurrency and Computation: Practice and Experience* 23, 16 (2011), 1990–2019. DOI:http://dx.doi.org/10.1002/cpe.1734

Massimiliano Rak, Antonio Cuomo, and Umberto Villano. 2011. CHASE: An autonomic service engine for cloud environments. In *Proceedings of the 20th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'11)*. IEEE, 116–121. DOI:http://dx.doi.org/10.1109/WETICE.2011.21

Rajiv Ranjan, Lizhe Wang, Albert Y. Zomaya, Dimitrios Georgakopoulos, Xian-He Sun, and Guojun Wang. 2015. Recent advances in autonomic provisioning of big data applications on clouds. *IEEE Trans. Cloud Comput.* 3, 2 (2015), 101–104. DOI:http://dx.doi.org/10.1109/TCC.2015.2437231

Bhaskar P. Rimal, Admela Jukan, Dimitrios Katsaros, and Yves Goeleven. 2011. Architectural requirements for cloud computing systems: an enterprise cloud approach. *J. Grid Comput.* 9, 1 (2011), 3–26. DOI:http://dx.doi.org/10.1007/s10723-010-9171-y

Ivan Rodero, Hariharasudhan Viswanathan, Eun Kyung Lee, Marc Gamell, Dario Pompili, and Manish Parashar. 2012. Energy-efficient thermal-aware autonomic management of virtualized HPC cloud infrastructure. *J. Grid Comput.* 10, 3 (2012), 447–473. DOI:http://dx.doi.org/10.1007/s10723-012-9219-2

Ivan Rodero, Juan Jaramillo, Andres Quiroz, Manish Parashar, and Francesc Guim. 2010. Towards energy-aware autonomic provisioning for virtualized environments. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 320–323. DOI:http://dx.doi.org/10.1145/1851476.1851520

Sushil K. Sah and Shashidhar R. Joshi. 2014. Scalability of efficient and dynamic workload distribution in autonomic cloud computing. In *Proceedings of the International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT'14)*. IEEE, 12–18. DOI:http://dx.doi.org/10.1109/ICICICT.2014.6781244

Mazeiar Salehie and Ladan Tahvildari. 2005. Autonomic computing: Emerging trends and open problems. *ACM SIGSOFT Software Eng. Notes* 30, 4 (2005), 1–7. DOI:http://dx.doi.org/10.1145/1082983.1083082

Zohreh Sanaei, Saeid Abolfazli, Abdullah Gani, and Rajkumar Buyya. 2014. Heterogeneity in mobile cloud computing: Taxonomy and open challenges. *IEEE Commun. Surv. Tutorials* 16, 1 (2014), 369–392. DOI:http://dx.doi.org/10.1109/SURV.2013.050113.00090

Prasad Saripalli, G. V. R. Kiran, R. Ravi Shankar, Harish Narware, and Nitin Bindal. 2011. Load prediction and hot spot detection models for autonomic cloud computing. In *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing (UCC'11)*. IEEE, 397–402. DOI:http://dx.doi.org/10.1109/UCC.2011.66

Mina Sedaghat, Francisco Hernández-Rodriguez, and Erik Elmroth. 2014. Autonomic resource allocation for cloud data centers: A peer to peer approach. In *Proceedings of the ACM Cloud and Autonomic Computing Conference (CAC'14)*. 131–140. DOI:http://dx.doi.org/10.1145/2494621.2494623

Mehdi Sheikhalishahi, Lucio Grandinetti, Richard M. Wallace, and Jose Luis Vazquez Poletti. 2015. Autonomic resource contention aware scheduling. *Software: Practice and Experience* 45, 2 (2015), 161–175. DOI:http://dx.doi.org/10.1002/spe.2223

Matthieu Simonin, Eugen Feller, A. Orgerie, Yvon Jégou, and Christine Morin. 2013. An autonomic and scalable management system for private clouds. In *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'13)*. IEEE, 198–199. DOI:http://dx.doi.org/10.1109/CCGrid.2013.44

Sukhpal Singh and Inderveer Chana. 2012. Cloud based development issues: A methodical analysis. *Int. J. Cloud Comput. Services Sci. (IJ-CLOSER)* 2, 1 (2012), 73–84. Retrieved from http://iaesjournal.com/online/index.php/IJ-CLOSER/article/view/1704.

Sukhpal Singh and Inderveer Chana. 2013a. Consistency verification and quality assurance (CVQA) traceability framework for SaaS. In *Proceedings of the IEEE 3rd International on Advance Computing Conference (IACC'13)*. IEEE, 1–6. DOI:http://dx.doi.org/10.1109/IAdCC.2013.6506805

Sukhpal Singh and Inderveer Chana. 2013b. Introducing agility in cloud based software development through ASD. *Int. J. u-and e-Service, Sci. Technol.* 6, 5 (2013), 191–202. http://dx.doi.org/10.14257/ijunesst.2013.6.5.17

Sukhpal Singh and Inderveer Chana. 2013c. Advance billing and metering architecture for infrastructure as a service. *Int. J. Cloud Comput. Services Sci. (IJ-CLOSER)* 2, 2 (2013), 123–133. Retrieved from http://iaesjournal.com/online/index.php/IJ-CLOSER/article/view/1960/739.

Sukhpal Singh and Inderveer Chana. 2015a. EARTH: Energy-aware autonomic resource scheduling in cloud computing. *J. Intell. Fuzzy Syst.* IOS Press, Systems Preprint (2015), 1–20. DOI:http://dx.doi.org/10.3233/IFS-151866

Sukhpal Singh and Inderveer Chana. 2015b. QoS-aware autonomic cloud computing for ICT. In *Proceedings of the International Conference on Information and Communication Technology for Sustainable Development (ICT4SD'15).* Springer International Publishing. Retrieved from http://www.springer.com/in/book/9789811001277#aboutBook.

Sukhpal Singh and Inderveer Chana. 2015c. Q-aware: Quality of service based cloud resource provisioning. *Comput. Elect. Eng.* (2015). DOI:http://dx.doi.org/10.1016/j.compeleceng.2015.02.003

Sukhpal Singh and Inderveer Chana. 2015d. QRSF: QoS-aware resource scheduling framework in cloud computing. *J. Supercomput.* 71, 1 (2015), 241–292. DOI:http://dx.doi.org/10.1007/s11227-014-1295-6

Rahul Singh, Upendra Sharma, Emmanuel Cecchet, and Prashant Shenoy. 2010. Autonomic mix-aware provisioning for non-stationary data center workloads. In *Proceedings of the 7th International Conference on Autonomic Computing*. AVC, 21–30. DOI:http://dx.doi.org/10.1145/1809049.1809053

Derek Smith, Qiang Guan, and Song Fu. 2010. An anomaly detection framework for autonomic management of compute cloud systems. In *Proceedings of the 34th Annual Computer Software and Applications Conference and Workshops (COMPSACW'10).* IEEE, 376–381. DOI:http://dx.doi.org/10.1109/COMPSACW.2010.72

Bogdan Solomon, Dan Ionescu, Cristian Gadea, Stejarel Veres, and Marin Litoiu. 2013. Self-optimizing autonomic control of geographically distributed collaboration applications. In *Proceedings of the ACM Cloud and Autonomic Computing Conference.* ACM, 1–8. DOI:http://dx.doi.org/10.1145/2494621.2494650

Alain Tchana, Giang S. Tran, Laurent Broto, Noel DePalma, and Daniel Hagimont. 2013. Two levels autonomic resource management in virtualized IaaS. *Future Generation Comput. Syst.* 29, 6 (2013), 1319–1332. DOI:http://dx.doi.org/10.1016/j.future.2013.02.002

I. B. M. Tivoli. 2005. Autonomic computing policy language. *Tutorial, IBM Corp.* (2005). DOI:http://cs.nju.edu.cn/yangxc/autonomic-computing/ACwpFinal.pdf

Adel N. Toosi, Rodrigo N. Calheiros, and Rajkumar Buyya. 2014. Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Comput. Surv.* 47, 1 (2014), 1–47. DOI:http://dx.doi.org/10.1145/2593512

Hien N. Van, Frederic Dang Tran, and Jean-Marc Menaud. 2009. Autonomic virtual resource management for service hosting platforms. In *Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing.* IEEE, 1–8. DOI:http://dx.doi.org/10.1109/CLOUD.2009.5071526

Kleber M. M. Vieira, Fernando Schubert, Guilherme A. Geronimo, Rafael de Souza Mendes, and Carlos B. Westphall. 2014. Autonomic intrusion detection system in cloud computing with big data. In *Proceedings of the International Conference on Security and Management (SAM-14)*. 173–178. Retrieved from http://www.researchgate.net/publication/266080250.

Hariharasudhan Viswanathan, Eun Kyung Lee, and Dario Pompili. 2011. Self-organizing sensing infrastructure for autonomic management of green datacenters. *IEEE Network* 25, 4 (2011), 34–40. DOI:http://dx.doi.org/10.1109/MNET.2011.5958006

Delei Weng and Michael A. Bauer. 2010. Using policies to drive autonomic management of virtual systems. In *Proceedings of the International Conference on Network and Service Management (CNSM'10).* IEEE, 258–261. DOI:http://dx.doi.org/10.1109/CNSM.2010.5691193

Linlin Wu, Saurabh K. Garg, Steve Versteeg, and Rajkumar Buyya. 2013. SLA-based resource provisioning for software as a service applications in cloud computing environments. *IEEE Trans. Services Comput.* 7, 3 (2013), 465–485. DOI:http://dx.doi.org/10.1109/CCGrid.2011.51

Jin Xiao and Raouf Boutaba. 2005. QoS-aware service composition and adaptation in autonomic communication. *IEEE J. Selected Areas Commun.* 23, 12 (2005), 2344–2360. DOI:http://dx.doi.org/10.1109/JSAC.2005.857212

Cheng-Zhong Xu, Jia Rao, and Xiangping Bu. 2012. URL: A unified reinforcement learning approach for autonomic cloud management. *J. Parallel Distrib. Comput.* 72, 2 (2012), 95–105. DOI:http://dx.doi.org/10.1016/j.jpdc.2011.10.003

Li D. Xu, Wu He, and Shancang Li. 2014. Internet of things in industries: A survey. *IEEE Trans. Industrial Inform.* 10, 4 (2014), 2233–2243. DOI:http://dx.doi.org/10.1109/TII.2014.2300753

Chee S. Yeo, Srikumar Venugopal, Xingchen Chu, and Rajkumar Buyya. 2010. Autonomic metered pricing for a utility computing service. *Future Generation Comput. Syst.* 26, 8 (2010), 1368–1380. DOI:http://dx.doi.org/10.1016/j.future.2009.05.024

Gae-Won You, Seung-Won Hwang, and Navendu Jain. 2013. URSA: Scalable load and power management in cloud storage systems. *Trans. Storage* 9, 1 (2013), 1–29. DOI:http://dx.doi.org/10.1145/2435204.2435205

Xindong You, Jian Wan, Xianghua Xu, Congfeng Jiang, Wei Zhang, and Jilin Zhang. 2011. ARAS-M: Automatic resource allocation strategy based on market mechanism in cloud computing. *J. Comput.* 6, 7 (2011), 1287–1296. DOI: http://dx.doi.org/10.4304/jcp.6.7.1287-1296

Eric Yuan, Sam Malek, Bradley Schmerl, David Garlan, and Jeff Gennari. 2013. Architecture-based self-protecting software systems. In *Proceedings of the 9th International ACM SIGSOFT Conference on Quality of Software Architectures*. ACM, 33–42. DOI: http://dx.doi.org/10.1145/2465478.2465479

Zhi-Hui Zhan, Xiao-Fang Liu, Yue-Jiao Gong, Jun Zhang, Henry Shu-Hung Chung, and Yun Li. 2015. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Comput. Surv.* 47, 4 (2015), 1–33. DOI: http://dx.doi.org/10.1145/2788397

Ziming Zhang, Qiang Guan, and Song Fu. 2012. An adaptive power management framework for autonomic resource configuration in cloud computing infrastructures. In *Proceedings of the 31st IEEE International Performance Computing and Communications Conference (IPCCC'12)*. IEEE, 51–60. DOI: http://dx.doi.org/10.1109/PCCC.2012.6407738

Xiaobo Zhou and Chang-Jun Jiang. 2014. Autonomic performance and power control on virtualized servers: Survey, practices, and trends. *J. Comput. Sci. Technol.* 29, 4 (2014), 631–645. DOI: http://dx.doi.org/10.1007/s11390-014-1455-4

Dimitrios Zissis and Dimitrios Lekkas. 2012. Addressing cloud computing security issues. *Future Generation Comput. Syst.* 28, 3 (2012), 583–592. DOI: http://dx.doi.org/10.1016/j.future.2010.12.006

Received May 2015; revised September 2015; accepted October 2015