The Journey of QoS-Aware Autonomic Cloud Computing

Sukhpal Singh, Inderveer Chana, and Maninder Singh, *Thapar University, India*

Resource allocation in the cloud can be challenging due to heterogeneity, dynamism, and failures. The authors offer a broad literature analysis of resource management in cloud computing, including resource provisioning and scheduling, and autonomic resource provisioning and scheduling.

The cloud offers three main service types: infrastructure, platform, and software. Thus, quality of service (QoS) in the cloud means efficiently monitoring and measuring services and following service-level agreements (SLAs) to ensure their efficient delivery.[1] However, providing dedicated cloud services that ensure users' dynamic QoS requirements and avoid SLA violations is a big challenge. Currently, cloud services are provisioned and scheduled according to resource availability but fail to ensure expected performance.

To address these issues, cloud providers should evolve their ecosystems to fulfill the QoS-aware requirements of each cloud component. To this end, two important aspects must be considered that reflect the complexity cloud management introduces: QoS-awareness and self-management. The QoS-aware aspect involves a service's capacity to be aware of its behavior, thus ensuring its elasticity, high availability, and reliability, along with important QoS parameters (cost, time, energy, and so on).[2] Self-management implies that the service can self-manage as its environment requires.

Based on QoS requirements, autonomic systems provide self-optimization and manage complexity proactively to reduce costs. Existing autonomic systems encompass only a few QoS parameters; thus, an autonomic resource management system is necessary that considers all the important QoS parameters—including availability, security, execution time, SLA violation rate,

 Published by the IEEE Computer Society

and energy consumption—for better resource management.[3] Unfortunately, present cloud computing systems and management techniques cannot handle these problems efficiently at runtime. To overcome such issues, cloud systems should have self-management characteristics such as self-optimization, self-healing, self-protection, and self-configuration. Thus, there is a need to automatically manage cloud users' QoS requirements to help providers meet SLAs and avoid violations.

Here, we highlight the prior research, current status, and future directions of cloud resource management.

The Need for Autonomic Cloud Computing

Autonomic cloud computing (ACC) can provide one solution for effective resource allocation by fulfilling users' QoS requirements and healing unexpected failures at runtime automatically, thus optimizing QoS parameters.[4] ACC's first objective is to identify and schedule those resources suitable to the appropriate workloads on time, thus increasing the effectiveness of resource utilization. In other words, the resource amount should be the minimum needed for a workload to maintain a required level of service quality or automatically minimize the workload completion time (maximize throughput).[5] Achieving better resource scheduling requires the best resource workload mapping. ACC's second objective is to identify an adequate and suitable workload that supports the scheduling of multiple workloads; this lets the cloud service fulfill numerous QoS requirements—including resource utilization, availability, reliability, security, and energy—for the cloud workload. Resource scheduling thus considers the execution time of each distinct workload; most importantly, overall performance is based on the workload type (with different QoS requirements).

We conducted a comprehensive investigation of resource management to study various resource management techniques—resource provisioning, resource scheduling, and autonomic resource provisioning and scheduling—in cloud computing. We compare and categorize these techniques[1–14] based on QoS and the focus of study (FoS) (see Table 1). We also present future research directions for autonomic resource management in the cloud.

Autonomic Cloud Computing Evolution

Autonomic computing systems were basically inspired by the human autonomic nervous system (ANS).[6] The ANS is able to deal with situations in the human body dynamically and manage them in unpredictable environments. ACC systems control the workings of cloud-based systems and applications without involving humans, similar to the ANS. Such systems perform checks, monitor data, and respond accordingly. Figure 1a illustrates the evolution of ACC across the years. As research in cloud computing progressed, ACC techniques and QoS parameters (Figure 1b) also evolved.

Resource Management in the Cloud

The goal of autonomic resource provisioning and scheduling is to execute applications within a deadline and fulfill users' QoS requirements with minimum complexity. Let's first examine the current status of resource management.

Cloud systems should have self-management characteristics such as self-optimization, self-healing, self-protection, and self-configuration.

Resource management is an umbrella activity that describes all the characteristics of resources. It comprises resource provisioning and resource scheduling, which can be further extended to autonomic resource provisioning and scheduling.

Resource Provisioning

Resource provisioning is the process of identifying adequate resources for a given workload based on QoS requirements as described by the cloud consumer. Resource management maps user workloads to resources based on QoS requirements. These resources are then said to be provisioned as per user requests. A resource provisioning process (Q-aware) is shown in Figure 2a. The challenges of resource management range from managing resource heterogeneity to efficiently matching available resources to workloads (known as matchmaking) through the workload analyzer (broker).[1] The broker

Table 1. Comparison of resource management techniques from 2009–2015.

Category	Technique	Focus of study	QoS service parameters addressed
Resource provisioning	Amazon Elastic Compute cloud performance analysis[2]	Server-oriented applications	Reduced response time
	Virtual machine multiplexing-based,[4] reputation-based resource provisioning mechanism (RPM)[3]	Virtualization and SLA	Improved resource utilization and response time
	SLA-based,[2] adaptive,[6] optimal[7]	Autonomic resources, multitier and Map Reduce applications	Improved execution time, resource utilization, and cost
	Deadline-driven,[1] dynamic,[8] energy-aware,[9] QoS-based[10]	Workloads, scientific and elastic applications	Improved execution time, response time, deadlines, and cost, but increased energy consumption
	Rule-based[11]	Hybrid clouds	Improved scalability but increased cost
	Adaptive,[10] dynamic[8]	Fault tolerance, high-level applications	Reduced execution time, but increased response time
	Q-aware[6]	Workload clustering	Improved execution time, cost, network bandwidth, resource utilization, availability, serviceability, and customer confidence levels
Resource scheduling	Hybrid,[14] bargaining-based[13]	Virtual infrastructure, multiple distributed resources	Reduced response and execution times
	Compromised cost time,[4] time,[3] profit,[6] VM-based,[7] nature- and bio-inspired[6]	Clustering, Map Reduce, processor sharing	Improved execution time, profit rate, and response time, but poorer resource utilization
	Cost- and SLA-based[10]	VM heterogeneity, user satisfaction	Improved response time, cost, and resource availability, but increased SLA violations
	Energy-,[11] QoS-,[10] and optimization-based[12]	Network performance, SLA, workload consolidation	Improved execution time and resource utilization, but increased power consumption
	Priority-based[13]	Parallel workloads	Reduced response time and number of migrations
	Dynamic,[10] bandwidth-based[14]	Divisible task scheduling, resource consumption patterns, SLA	Improved execution time, resource cost, and transition cost
	QoS-based Resource Scheduling Framework (QRSF)[5]	Cost, time, bargaining-based scheduling	Reduced execution time and energy consumption but increased cost
Autonomic resource provisioning and scheduling	Workload provisioning and self-optimization[1]	Clustering-based pattern detection, SLA negotiation	Improved cost and resource utilization
	SLA-aware,[5] deadline-driven,[1] budget and anomaly detection[6]	Heterogeneous nodes, elastic clouds, virtualization	Improved accuracy, computation overhead, deadlines, and cost
	Fault-tolerance,[9] market-based[10]	Dynamic scalability, resource allocation, dynamic virtualization, SLA violations	Increased reliability, cost, and execution time, but increased computing load
	Self-healing,[10] self-configuration,[11] energy-based[8]	Fault tolerance, distributed virtualization, load balancing	Reduced energy consumption but reduced reliability and increased response time
	Adaptive,[1] self-protection[1]	Scientific workloads, adaptation patterns, query optimization	Improved SLA violations, response time, and cost but reduced availability and resource utilization
	Dynamic workload distribution[6]	Autoscaling	Reduced execution time and cost
	EARTH[2]	Fuzzy logic, virtualization	Improved energy consumption, latency, and resource utilization, and improved computing capacity and energy efficiency

QoS: quality of service; FoS: focus of study; SLA: service-level agreement

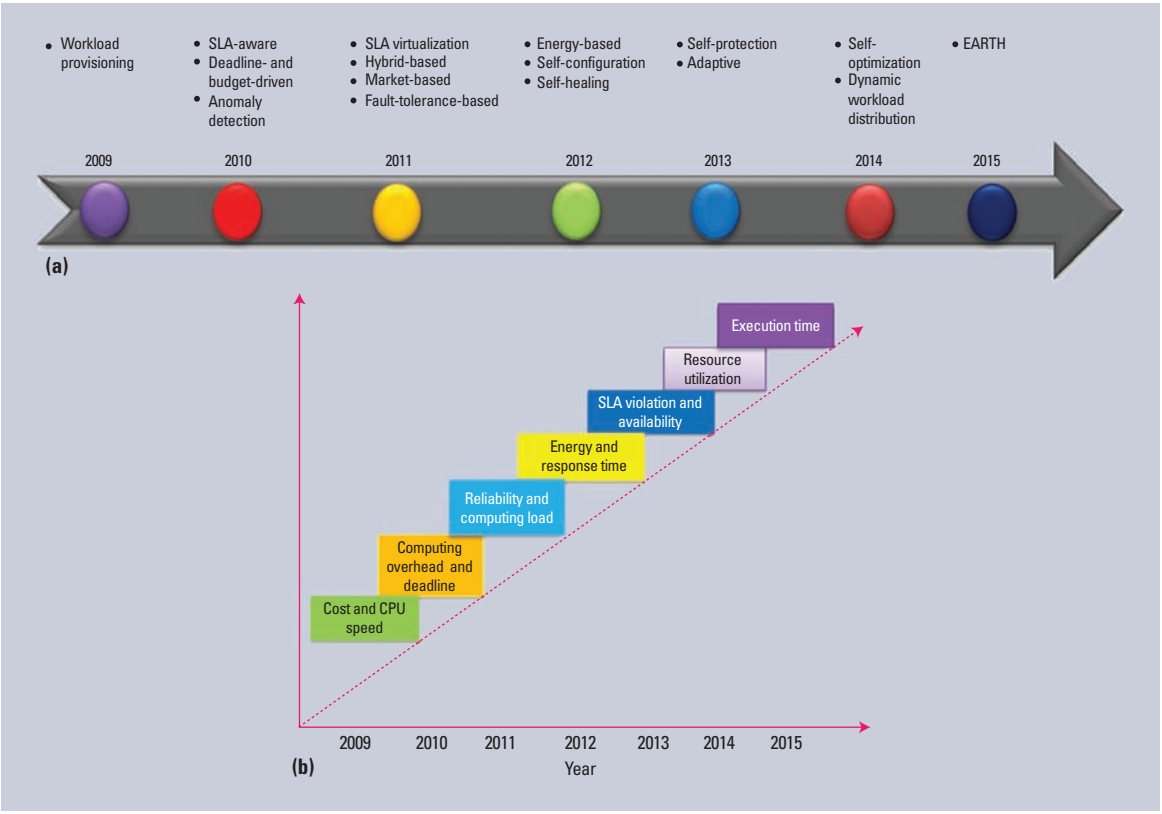

Figure 1. The journey of quality-of-service (QoS)-aware autonomic cloud computing (ACC). As research in cloud computing progressed, (a) ACC techniques and (b) QoS parameters also evolved.

performs matchmaking after the user submits the workloads and determines whether the workload can be provisioned on resources based on QoS requirements.[6] The broker also sends requests to the resource scheduler for scheduling after successful resource provisioning. The broker releases extra resources from the resource pool based on the required performance. The broker stores information about resources for submitting workloads and monitors the desired performance that will cause the system to either acquire or release resources.[4]

In Q-aware, a *bulk of workloads* comes for execution and is processed and stored in the workload queue. The *workload analyzer* (WA) contains information about resources, QoS metrics, and the SLA to provision the resources for workload execution based on QoS requirements. In *SLA measure*, the WA receives information from the suitable SLA. After studying and confirming the various QoS constraints that the workload requires, the WA checks resource availability. *QoS metric data* contains information regarding QoS metrics used to calculate weight for workload clustering. All the workloads are submitted and analyzed based on their QoS requirements. Different workloads are clustered efficiently for execution on different sets of resources. Resource details include the number of CPUs, memory size, cost, type, and amount.[1] The *resource provisioner* provides the demanded resources to the workload for execution in the cloud environment only if the required resources are available in the resource pool. If the required resources are not available according to the QoS requirement, then the *workload resource manager* (WRM) asks to resubmit the workload with modified QoS requirements in the form of an SLA, based on the information of existing resources. After resource provisioning, workloads are submitted to the resource scheduler. Then, the resource scheduler requests permission to submit the workload for resources.

Resource Scheduling

Resource scheduling is the process of mapping and executing cloud consumer workloads based on resources selected through resource provisioning. A resource scheduling model, the QoS-based Resource Scheduling Framework (QRSF), is shown in Figure 2b. First, the cloud consumer

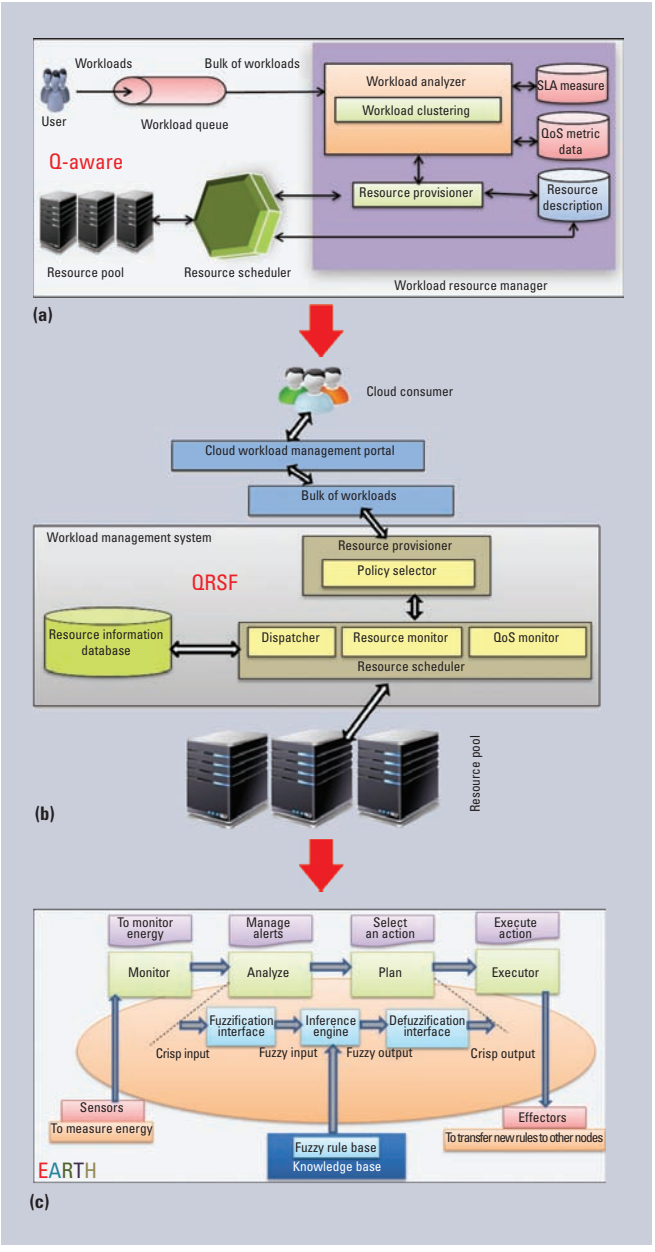

Figure 2. Resource management in the cloud. (a) The Q-aware technique demonstrates resource provisioning, whereas (b) the QoS-based Resource Scheduling Framework (QRSF) illustrates resource scheduling. (c) EARTH, an energy-aware autonomic resource management model, was developed for the effective, autonomic scheduling of resources.

submits a request for workload execution in the form of workload details through the *cloud workload management portal*. Based on these details, the broker (resource provisioner) finds the suitable resources for a given workload and determines the feasibility of provisioning resources based on QoS requirements.[1] All the provisioned resources are kept in the resource queue, whereas the remaining resources are in the resource pool.[5] Submitted workloads are processed in the workload queue. At this stage, the scheduling agent maps the provisioned resources to given workloads, executes the workloads, and releases the resources back to the resource pool after successful workload completion. Based on QoS requirements, the resources are assigned to the cloud workloads for execution. The *policy selector* is used to select the appropriate scheduling policy based on workload details described by the cloud consumer.[2] Based on the scheduling policy, resources are allocated to the cloud workloads.

The workload is dispatched by the *dispatcher* only if the workloads can be executed according to the QoS parameters mentioned in the SLA. The *resource monitor* checks the status of resource scheduling—for example, whether the required number of resources is provided. The *QoS monitor* contains the information regarding QoS parameters to check whether all the workloads are executing in their specified range. *Suppose deadline* is a QoS parameter, so the QoS monitor's responsibility is to check whether workloads are executed before the desired deadline. There is an SLA violation if the workload executes after the desired deadline.

Autonomic Resource Provisioning and Scheduling

Autonomic resource provisioning and scheduling improves resource utilization and user satisfaction.[2] Self-management is a service's ability to efficiently maximize resource allocation and utilization to satisfy the requirements (QoS) of different users. EARTH, an energy-aware autonomic resource management model was developed for the effective scheduling of resources; it considers energy consumption to be an important QoS parameter and maintains SLA. Figure 2c presents EARTH's architecture. EARTH is based on IBM's four-step autonomic model: *monitor*, *analyze*, *plan*, and *execute* (see Figure 3).

Initially, monitors are used to collect information from sensors (which get information about energy consumption and resource utilization for all the systems working under the cloud and update the information from time to time). They continuously monitor the value of energy consumption and resource utilization 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 to take adequate action for corresponding alerts. Once the data has been analyzed, EARTH executes the actions corresponding to the alerts automatically. EARTH also compares the actual energy consumption with a threshold value of energy consumption. If energy consumption is less than this threshold value, then workload execution continues; otherwise, an alert is generated. The input stage consists of three linguistic variables: *workload waiting time*, *workload execution time*, and *resource energy consumption*. The output stage first executes the workload with the highest priority (the workload processing priority) with minimum energy consumption and maximum resource utilization. We classified workloads into two categories based on the workload processing priority: urgent workloads and nonurgent workloads. EARTH calculates a workload's final priority.

Workloads with the highest priority are put into the urgent workload category, whereas the remaining workloads are considered nonurgent. Fuzzy rules are used to schedule the workloads according to their priorities. EARTH automatically checks the total number of workloads in the workload queue after each new workload is added. Workload priority changes adaptively—for example, when the priority of a newly added workload is higher. In such cases, the workload deadline is a mandatory consideration. Otherwise, workloads with higher priority wait for a long time, leading to starvation and reducing user satisfaction. Furthermore, EARTH checks the status of resource requirements. If sufficient resources are provided, then it starts workload execution; otherwise, it adds new resources. The executor implements the plan after analyzing alerts completely. The executor's main objective is to reduce latency and energy consumption and improve resource utilization and energy efficiency. Based on the output of the analysis (*crisp output*), the executor tracks new workload submissions and resource additions and takes action according to rules. The *effector* is used to transfer the new policies, rules, and alerts to other nodes with updated information. EARTH always executes those workloads with the highest priority (that is, those with the earliest deadline).

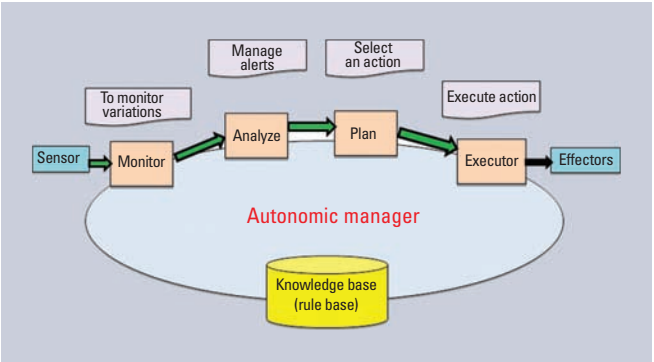

Figure 3. Architecture of an autonomic manager.

Future Research Perspectives

Considerable progress has been achieved in autonomic cloud computing, and scalable computing infrastructures are easily implemented by cloud computing on a pay-per-use basis. However, we identified the following research directions from the existing resource management literature:[1–14]

- Presently, autonomic computing systems work as single-tier architectures. To achieve better performance in terms of deadlines, job distribution, availability, and other important QoS parameters (cost, time, energy, and so on), cloud autoscaling architectures must shift to a multitier application environment.
- Datacenters consume a tremendous amount of energy, which leads to high operational costs and contributes heavily toward carbon footprints. So, cloud-based autonomic systems can be further extended by adding energy efficiency as a QoS parameter, which can further reduce environmental impacts.
- Immediate decisions in autonomic systems can be improved using optimized data mining or machine learning techniques.
- Autonomic systems can be extended by adding heterogeneous resources in the resource allocation mechanism through more infrastructure providers, which can more effectively serve a higher number of user requests and fulfill user requirements (QoS).
- Due to the large number of user requests in multiple service queues, autonomic systems lack user interaction. More QoS parameters, such as bandwidth, storage, energy, time, and cost, can be added to efficiently test system performance.
- Autonomic systems consider only functional requirements (input and output) in SLAs. Nonfunctional requirements (QoS) should also be considered for better agreement and performance.

Important QoS requirements in autonomic resource management are scalability, reliability, security, execution time, cost, availability, energy, and so on.
- Autonomic system performance can be improved by adding time-, energy-, and cost-based resource scheduling policies for self-optimization, thus improving resource utilization. Reducing costs will add to the provider's profits and reduce user costs.
- The scalability, storage management, and load balancing of autonomic systems can be improved using distributed NoSQL databases, which have high recall and precision rates.
- The reliability of autonomic systems can be improved by adding proactive schedulers that make scheduling decisions based on infrastructure and network characteristics using rule-based policies in which systems make decisions automatically.

To overcome these challenges, a resource management technique is required that provisions and schedules cloud resources as per user requirements (QoS). This technique should be self-managing (autonomic) so as to adapt itself at runtime and would help mitigate SLA violations and reduce costs. For example, if the SLA requires autoscaling and high performance, urgent cloud workloads could be placed in the priority queue for earlier execution through automatic allocation of reserved resources. Further possible future research directions for QoS are as follows:

- 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. Efficient resource management in autonomic cloud computing requires finding the main reasons for workload detection and resources for better mappings in the future.
- Different QoS parameters have to be reassessed to implement the autonomic resource management mechanisms for given QoS parameters.
- Workloads need to be executed automatically so as to be scalable, flexible, and avoid resource overloading and underloading.
- Due to the difficulty of predicting the behavior (in terms of QoS requirements) and demand (in terms of resources) of workloads and applications, there is a need for effective QoS-aware autonomic resource management techniques that can easily make the right decisions regarding the dynamic scaling of resources, workloads, and applications.
- At present, it is difficult to make autonomic resource management techniques cost-effective and energy-efficient due to increasing energy requirements and operating costs. To resolve this problem, there is need for an autonomic system that uses the best hardware and software configurations to improve resource utilization and fulfill important QoS parameters.
- There is a need to develop autonomic resource management techniques that can optimize both user-centric (budget and execution time) and resource-centric (reliability, availability, energy, and utilization) QoS targets.
- Most cloud companies have a large number of resources for serving different business applications and handling large amounts of data. However, these resources are difficult to migrate and manage due to security and privacy issues related to interoperability and integration.

Our survey has helped determine research gaps in autonomic cloud computing as well as still-existing research issues. We summarized the existing literature in terms of the systematic evolution of autonomic cloud computing. To know the impact of QoS requirements on self-management, we must understand the evolution of autonomic cloud computing to know whether the provisioned resources are scheduled efficiently. We can further conclude the following:

- Contrast and assessment of ACC in the cloud can aid in selecting an autonomic scheduling algorithm based on a workload's QoS requirements.
- QoS parameters (energy, cost, time, and so on) can be improved in the delivered cloud service if resources are reserved in advance.
- Proper mapping of workloads and resources can improve performance significantly.
- Resource allocation based on workload type (homogenous and heterogeneous) can improve resource utilization.

We anticipate that our research will help others determine the important characteristics of autonomic resource management.

Acknowledgments

Author Sukhpal Singh acknowledges the Department of Science and Technology (DST), government of India, for awarding him the Innovation in Science Pursuit for Inspired Research (INSPIRE) Fellowship (registration/IVR number 201400000761 [DST/INSPIRE/03/2014/000359]) to carry out this research.

References

1. S. Singh and I. Chana, "QoS-Aware Autonomic Resource Management in Cloud Computing: A Systematic Review," *ACM Computing Surveys*, vol. 48, no. 3, 2015, pp. 1–46.
2. S. Singh and I. Chana, "EARTH: Energy-Aware Autonomic Resource Scheduling in Cloud Computing," *J. Intelligent and Fuzzy Systems*, vol. 30, no. 3, 2016, pp. 1581–1600.
3. H. Viswanathan et al., "Uncertainty-Aware Autonomic Resource Provisioning for Mobile Cloud Computing," *IEEE Trans. Parallel and Distributed Systems*, vol. 26, no. 8, 2015, pp. 2363–2372.
4. L. Mashayekhy, M.M. Nejad, and D. Grosu, "A PTAS Mechanism for Provisioning and Allocation of Heterogeneous Cloud Resources," *IEEE Trans. Parallel and Distributed Systems*, vol. 26, no. 9, 2015, pp. 2386–2399.
5. S. Singh and I. Chana, "A Survey on Resource Scheduling in Cloud Computing: Issues and Challenges," *J. Grid Computing*, vol. 14, no. 1, 2016, pp. 1–48; doi.org/10.1007/s10723-015-9359-2.
6. S. Singh and I. Chana, "Cloud Resource Provisioning: Survey, Status, and Future Research Directions," *Knowledge and Information Systems*, vol. 49, no. 3, 2016, pp. 1–65; dx.doi.org/10.1007/s10115-016-0922-3.
7. S. Singh et al., "SOCCER: Self-Optimization of Energy-efficient Cloud Resources," *Cluster Computing*, vol. 19, no. 4, 2016, pp. 1787–1800.
8. K.M. Khan and Q. Malluhi, "Establishing Trust in Cloud Computing," *IT Professional*, vol. 12, no. 5, 2010, pp. 20–27.
9. N.S. Chauhan and A. Saxena, "A Green Software Development Life Cycle for Cloud Computing," *IT Professional*, vol. 15, no. 1, 2013, pp. 28–34.
10. A. Hameed et al., "A Survey and Taxonomy on Energy Efficient Resource Allocation Techniques for Cloud Computing Systems," *J. Computing*, vol. 98, no. 7, 2014, pp. 751–774.
11. M.B. Qureshi et al., "Survey on Grid Resource Allocation Mechanisms," *J. Grid Computing*, vol. 12, no. 2, 2014, pp. 399–441.
12. S. Pandey et al., "An Autonomic Cloud Environment for Hosting ECG Data Analysis Services," *Future Generation Computer Systems*, vol. 28, no. 1, 2012, pp. 147–154.
13. C.S. Yeo et al., "Autonomic Metered Pricing for a Utility Computing Service," *Future Generation Computer Systems*, vol. 26, no. 8, 2010, pp. 1368–1380.
14. Z. Sanaei et al., "Heterogeneity in Mobile Cloud Computing: Taxonomy and Open Challenges," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, 2014, pp. 369–392.

Sukhpal Singh *is a lecturer in the Computer Science and Engineering Department at Thapar University, India. His research interests include software engineering, cloud computing, the Internet of Things, and fog computing. Singh has worked as an SRF-Professional on a Department of Science and Technology project, government of India, and has 30-plus research publications in reputed journals and conferences. He received a PhD in autonomic cloud computing from Thapar University, and the Gold Medal in MEng in software engineering. Contact him at ssgill@thapar.edu.*

Inderveer Chana *is a professor in the Computer Science and Engineering Department at Thapar University, India, and works on various research projects funded by the government of India. Her research interests include grid and cloud computing, software engineering, and software project management. Chana has more than 100 research publications in reputed journals and conferences, and has supervised more than 40 awarded ME theses and seven PhD theses. She received a PhD in computer science with a specialization in grid computing from Thapar University. Contact her at inderveer@thapar.edu.*

Maninder Singh *is an associate professor in the Computer Science and Engineering Department at Thapar University, India. His research interests are in network security and cloud computing. Singh is on the Roll of Honor at the EC-Council US, and is certified as an ethical hacker (C-EH), security analyst (ECSA), and Licensed Penetration Tester (LPT). He received his PhD in network security from Thapar University. Contact him at msingh@thapar.edu.*