Brownout Approach for Adaptive Management of Resources and Applications in Cloud Computing Systems: A Taxonomy and Future Directions

MINXIAN XU and RAJKUMAR BUYYA, The University of Melbourne, Australia

Cloud computing has been regarded as an emerging approach to provisioning resources and managing applications. It provides attractive features, such as an on-demand model, scalability enhancement, and management cost reduction. However, cloud computing systems continue to face problems such as hardware failures, overloads caused by unexpected workloads, or the waste of energy due to inefficient resource utilization, which all result in resource shortages and application issues such as delays or saturation. A paradigm, the brownout, has been applied to handle these issues by adaptively activating or deactivating optional parts of applications or services to manage resource usage in cloud computing system. Brownout has successfully shown that it can avoid overloads due to changes in workload and achieve better load balancing and energy saving effects. This article proposes a taxonomy of the brownout approach for managing resources and applications adaptively in cloud computing systems and carries out a comprehensive survey. It identifies open challenges and offers future research directions.

CCS Concepts: • **General and reference** → **General literature**; • **Computer systems organization** → **Cloud computing**; *Software and its engineering*;

Additional Key Words and Phrases: Cloud computing, adaptive management, brownout, quality of service, optional services

ACM Reference format:
Minxian Xu and Rajkumar Buyya. 2019. Brownout Approach for Adaptive Management of Resources and Applications in Cloud Computing Systems: A Taxonomy and Future Directions. *ACM Comput. Surv.* 52, 1, Article 8 (January 2019), 27 pages.
https://doi.org/10.1145/3234151

1 INTRODUCTION

Cloud computing has been regarded as one of the most dominant technologies in promoting the future economy (Irwin 2013). Traditionally, service providers used to establish their own data centers, making a huge investment to maintain applications and provide services to users. With cloud computing, resources can be leased by application providers and the applications can be deployed

This work is supported by China Scholarship Council, and Australia Research Council (ARC).
Authors' address: M. Xu and R. Buyya, Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Parkville, VIC, 3010, Australia; emails: minxianx@student.unimelb.edu.au, rbuyya@unimleb.edu.au.

0360-0300/2019/01-ART8 $15.00
https://doi.org/10.1145/3234151

without any upfront costs. Nowadays, many applications are developed for cloud computing systems, and clouds also provide elastic resources for applications (Son et al. 2017). This feature encourages enterprises to migrate their local applications to clouds (Aslanpour et al. 2017).

In addition to traditional requirements, cloud applications are experiencing unpredictable workloads because of the dynamic amount of requests and users (Cheng et al. 2016). Thus, cloud computing systems are also required to be designed as robust to handle unexpected events: request bursts, commonly called *flash crowds,* can increase the size of requests significantly. For example, the servers of Weibo, a Chinese web social network website owned by Sina, broke down after a Chinese celebrity announced his new relationship, the result of the celebrity's fans flooding into the website (Koetse 2017).

Similarly, in cloud data centers, unexpected hardware failures are also common issues. In 2016, severe weather led to the outage of Amazon data centers in Sydney, knocking services of many companies offline (Patrick Hatch 2016). Moreover, performance interference resulting from co-located applications and workload consolidations may lead to unexpected performance degradations (Zhang et al. 2014).

1.1 Need for Adaptive Management in Cloud Computing Systems

To handle these phenomena, applications must be carefully designed and deployed. For instance, techniques such as autoscaling (Lorido-Botran et al. 2014), data replication (Milani and Navimipour 2016b), workload consolidation (Homsi et al. 2017), and dynamic load balancing (Panwar and Mallick 2015) are applied to overcome unexpected events only if the available resources are adequate. However, these unexpected events are generally only of relatively short duration, and it is not economical to provision sufficient capacity at all times. But, without sufficient resource provisioning, applications can be saturated and cause users to experience longer response times or even no response at all. As a result, service providers may lose customers and revenues. Therefore, we argue that adaptive management of resources and applications is needed for cloud computing systems.

With adaptive management of resources and applications, different benefits can be achieved. Adaptive management can improve the Quality of Service (QoS) guarantee of cloud services. The QoS guarantee plays a crucial role in system performance for cloud environments (Li et al. 2017), and cloud computing systems are required to offer QoS guaranteed services (Singh and Chana 2016a; Singh and Chana 2016b). It is a challenging issue for clouds to support various co-located applications with different QoS constraints and to provision resources efficiently to be adaptive to users' dynamic behaviors (Sampaio et al. 2015; Dou et al. 2017). It is reported that 53% of mobile users abandon pages that have no response within three seconds. Therefore, Google firmly recommends a one-second web page loading time to improve users' experience (Everts 2016). QoS of applications can be improved by dynamically adding or removing hosts via adaptive management.

Energy efficiency of cloud computing systems can be improved by adaptive management. It has become a major problem in the IT industry that huge amounts of energy are consumed by cloud data centers (Mastelic et al. 2015). The rise and evolution of complicated computation-intensive applications have promoted the establishment of large cloud data centers that boost the total amount of power usage (Beloglazov and Buyya 2013). The physical servers deployed in clouds generate massive amounts of heat and require an environment equipped with powerful air conditioners. One of the key reasons for this huge energy usage is the inefficient utilization of resources (Beloglazov et al. 2012). Adaptive management is able to improve resource usage so that energy consumption can be reduced. For example, when there are fewer requests, adaptive management can reduce energy consumption by consolidating workloads onto fewer active physical machines; thus idle physical machines can be placed in low-power mode or fully turned off.

Balancing the loads in cloud computing systems is another objective that can be fulfilled by adaptive management. Load balancers are regular components of web applications that allow the system to be scalable and resilient (Randles et al. 2010). Numerous load balancing algorithms have been introduced by researchers, focusing on different optimization targets ranging from balancing virtual machine's load to physical machines, with specific optimizations by both heuristic and meta-heuristic algorithms (Xu et al. 2017). The purposes of load balancing can be varied, including geographical balancing (Liu et al. 2015), electricity costs reduction (Rahman et al. 2014), and application load balancing in cloud computing systems (Milani and Navimipour 2016a). Adaptive management can avoid overloads to achieve load balancing.

A promising approach for adaptive management of resources and applications in cloud computing systems is **brownout** (Klein et al. 2014a). In the field of brownout, applications/services are extended to two parts: mandatory and optional. The mandatory parts must be kept running all the time, such as the critical services in the system, including data-relevant services. The optional parts, on the other hand, need not be active all the time and can be deactivated temporarily to ensure system performance in the case of flash crowds.

A motivational example of a brownout-enabled application is the E-Commerce system, where product descriptions are shown along with related products suggested to end users. These related products are managed by a recommendation engine in the system. The recommendation engine can be identified as an optional part because it is not strictly necessary for the core function to work. Indeed, when the system is overloaded, even if the recommendation engine improves the user experience, it is preferable to deactivate the engine temporarily to obtain a more responsive website for more users.

1.2 Motivation of Research

Currently, brownout approaches have been applied in cloud computing systems for different optimization objectives, including system robustness improvement, overbooking, load balancing, and energy efficiency. Therefore, we investigate them in depth as noted here:

- The brownout approach has shown promise in managing applications and resources in cloud computing systems. Therefore, this article discusses the development and application of brownout approaches in the cloud computing area.
- We identify the necessity of a literature review to summarize progress in the brownout approach using adaptive management of resources and applications for cloud computing systems. Consequently, we have surveyed existing articles relevant to this topic, and we aim to draw more attention to and encourage efforts in advanced research with brownout approaches.

1.3 Our Contributions

The major contributions of our work are summarized as follows:

- We propose a taxonomy of brownout-based adaptive management of resources and applications in cloud computing systems.
- We investigate a comprehensive review of brownout approaches in cloud computing systems for adaptive management of applications and resources.
- We categorize the studied brownout approaches according to their common features and properties, and we compare the advantages and limitations of each approach.
- We identify the research gaps and future research directions in brownout-enabled cloud computing systems.

1.4 Related Surveys

A few articles have conducted surveys or taxonomies on resource management in cloud computing. Kaur and Chana (2015) conducted a comprehensive taxonomy and survey for energy-efficient scheduling approaches in clouds. Weerasiri et al. (2017)) introduced a survey and taxonomy for resource orchestration in clouds while not focusing on adaptive resource management. Zhan et al. (2015) investigated the cloud computing resource scheduling approaches and summarized their evolution. Mansouri et al. (2017) presented a survey and taxonomy on resource management in cloud environments, with a focus on the management of storage resources. Singh and Chana (2016b) proposed a systematic review of QoS-aware automatic resource scheduling approaches in cloud scenarios.

However, there is no existing survey and taxonomy focusing on the brownout approach. Thus, our article enhances previous surveys and focuses on the brownout-based approach. It also identifies the open challenges and future research directions in applying brownout in cloud computing systems for adaptive management of resources and applications.

1.5 Article Structure

The rest of the article is organized as follows: A brief background on cloud computing and adaptive management is introduced in Section 2. Then, in Section 3, a discussion on the evolution of brownout in cloud computing is presented. Section 4 depicts the methodology we applied to look for suitable related articles. Section 5 discusses the phases and taxonomy of brownout approaches in cloud computing systems. A review of brownout approaches and their mappings to the phases is presented in Section 6. Section 7 describes the brownout approach using a perspective model. Future directions and open challenges are discussed in Section 8. Finally, the conclusions of this work are given in Section 9.

2 BACKGROUND

In this section, we briefly introduce the background of cloud computing and adaptive management.

2.1 Cloud Computing

The appearance of cloud computing is regarded as a novel paradigm in the information technology industry (Buyya et al. 2014). The aim of cloud computing is to provide resources in the form of utilities like water, gas, and electricity for daily use. Some attractive characteristics including an on-demand resource provisioning model, scalability enhancement, operational cost reduction, and convenient access are offered by clouds. All these features enable cloud computing to be appealing to businesses by eliminating the complexity of service provider provisioning plans and by allowing companies to begin with the minimum resources required. Cloud platforms like EC2, Google Cloud, and Azure have been built by large infrastructure providers to support applications around the world with the purpose of assuring that these applications are scalable and available for the demands of the users (Mell et al. 2011). The cloud customers are allowed to dynamically lease and unlease on-demand resources based on their requirements.

2.2 Adaptive Management

In the past, a considerable amount of research in adaptive techniques to manage system resources has been conducted. Adaptive techniques are applied to management issues including protection, optimization, and recovery. These properties are often featured as self-* characteristics (Moreno 2017).

Table 1. The Earliest Works in Brownout Approach for Cloud Computing Sytems in 2014 and Their Citations

Title	Citations
"Brownout: Building more robust cloud applications"	84
"The straw that broke the camel's back: safe cloud overbooking with application brownout"	22
"Improving cloud service resilience using brownout-aware load-balancing"	23
"Control-theoretical load-balancing for cloud applications with brownout"	14

The feedback loop is the essential concept used to develop an adaptive system, one that monitors its status and its environment and adapts as desired to obtain the required self-* characteristics. An application's feedback loop is viewed as an important factor to enable the adaptive management of resources (Iglesia and Weyns 2015). In adaptive systems, one favorite representation of the feedback loop is the Monitor, Analyze, Plan, Execute, and Knowledge (MAPE-K) loop (Arcaini et al. 2015). In the MAPE-K loop, there are several phases to be accomplished:

1. Monitoring system status and environmental situation in the Monitor phase;
2. Analyzing the collected data and determining whether adaptation is required or not in the Analyze phase;
3. Planning the approach to adapt the system in the Plan phase;
4. Executing the plan in the Execute phase, where the Knowledge Pool is shared by these four phases and acts in an integration role (De Lemos et al. 2013).

3 ARTICLE SELECTION METHODOLOGY

In this section, we introduce the approach we followed to find our surveyed articles as well as the outcome.

3.1 Source of Articles

Related articles were broadly searched in mainstream academic databases, including IEEEXplore, Springer, Elsevier, ACM Digital Library, ScienceDirect, Wiley Interscience, and Google Scholar.

3.2 Search Method

Our search involved two phases. In the first phase, we used the keywords "Brownout" and "Cloud Computing" to search the titles and abstracts of research articles. Several results were found, but the numbers of these articles are quite limited, possibly because although some articles may be motivated by the mechanism of brownout, they do not use the term in their titles or abstracts. Thus, in the second phase, inspired by initial brownout research conducted in 2014, we planned to find other articles. Pioneering works in the brownout approach appearing in 2014 and their citation numbers[1] are presented in Table 1. By investigating articles that cite these four papers, we found more articles that addressed brownout.

3.3 Outcome

We found 18 research articles focusing on the brownout approach in cloud computing systems for adaptive management of resources and applications: 77.8% of these research papers were presented in conferences, 16.6% were published in journals, and 5.6% were presented in symposiums. Moreover, one master's thesis (Desmeurs 2015) and one PhD thesis (Moreno 2017) have explored this

[1]This search was conducted on February 5, 2018.

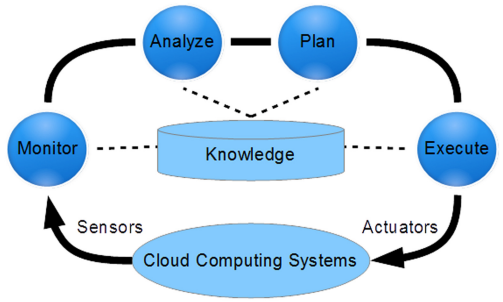

Fig. 1. Cloud computing systems with MAPE-K adaption loop.

topic. The contents of these two theses are derived from research articles we found and reviewed, therefore, they are not included in the following taxonomy and review.

4 BROWNOUT APPROACH

Brownout is inspired by the blackout approach used in emergency cases to cope with loss of or over-demand for electricity (Tomás et al. 2014). In this section, we introduce the background and evolution of brownout approach.

4.1 Overview of Brownout Approach

4.1.1 Definition. Brownout is a self-adaptive paradigm that enables or disables optional parts (application components or services) in the system to handle unpredictable workloads (Klein et al. 2014a). The idea behind the brownout paradigm is to be adaptive: Optional parts might temporarily be deactivated so that the essential functions of the system are ensured and applications avoid saturation. Deactivating certain optional functions can contribute to increasing the request acceptance rate by utilizing those resources for core functions.

Therefore, the brownout paradigm is a method to make the cloud computing system adaptive to changing workloads. Brownout also allows improved resource utilization while keeping applications responsive to avoid overloads. Moreover, brownout can be regarded as a special type of per request admission control, where some requests are fully admitted, while others may only be admissible without optional services. In the brownout paradigm, the *dimmer* is a control knob that takes a value from [0, 1] to represent the probability of running the optional parts and to manage brownout activities in the system (Klein et al. 2014a).

4.1.2 Architecture Model. The brownout-based scheduling of resources and applications in cloud computing systems complies with conventional types of adaptive architectures. It is derived from MAPE-K (Arcaini et al. 2015) feedback loop control, using phases, like the Monitor, Analyze, Plan, Execute, and Knowledge phases illustrated in Figure 1. Cloud computing systems are the target system of MAPE-K loops and are combined with MAPE-K loops through sensors and actuators. The responsibilities of each module in MAPE-K are:

- **Knowledge module:** Describes the whole system at the abstract level by capturing the major system features and status. The captured information is applied to trigger desirable adaptations;
- **Monitor module:** Collects data from sensors and monitors system status continuously. The collected data are transferred to the Analyze module for further analysis;
- **Analyze module:** Analyzes the data obtained from the Monitor module and provides references for the Plan module. Different analysis methods can be applied in this module;

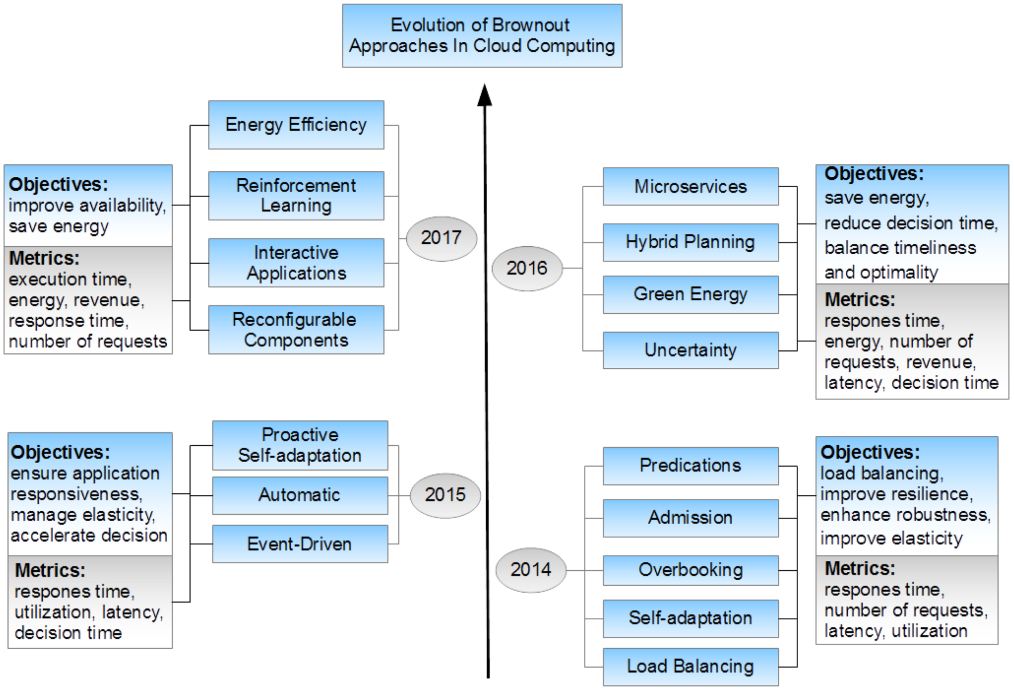

Fig. 2. Evolution of brownout approach in cloud computing systems.

- **Planning module:** Makes plans to change system states to be adaptive to workload fluctuations;
- **Execution module:** Implements the plans. Its main role is ensuring the specified system requirements. In addition, through actuators, it also traces new changes and makes other plans based on the predefined rules in the knowledge pool.

4.1.3 Brownout Management. The challenges for brownout management include:

- When should optional services be deactivated?
  This is relevant to system status. Optional services would be deactivated when some system indicators show the system is not running as expected (e.g., the system is becoming overloaded when requests are bursting).
- How to deactivate optional services?
  Services can be processed in different ways. For example, stateless services can be deactivated without any extra efforts. However, for the stateful services, the states should be recorded and reloaded when they are activated again.
- Which optional services should be deactivated?
  This depends on the optional service selection algorithm. The deactivated optional services can be selected based on the status of services, such as utilization.

4.2 Evolution of Brownout Approaches in Cloud Computing

The optimization of the objectives and metrics of brownout approaches in cloud computing systems has been investigated and several methods have been proposed over the years. As shown in Figure 2, we aim to show the evolution and development of brownout approaches in recent years.

In 2014, the application of brownout in cloud computing systems was proposed by Klein et al. (2014a), who introduced brownout as a self-adaptation programming paradigm. They also proposed that brownout can enhance system robustness when unpredictable requests are coming into the system. Two web application prototypes were presented, RUBiS (RUBiS 2009) and RuBBoS (RUBBoS 2005), which have been widely used in follow-up research. This work showed the effectiveness of the brownout approach, although the experiments were only conducted on a single machine. Dürango et al. (2014) presented load balancing strategies for application replicas based on brownout and admission control to maximize application performance and improve system resilience. However, the limitation is that the modeled application is not generalizable.

To find failures earlier and avoid the limitations of periodic, event-driven monitoring, Klein et al. (2014b) proposed two load balancing algorithms for brownout-aware services. The results showed that the proposed algorithm has better performance in fault-tolerance. Maggio et al. (2014) proposed and analyzed several different control strategies for brownout-aware applications to avoid overloads. Predictions for incoming load and service time are also applied in the proposed policies. To avoid Service-Level Objective (SLO) violations and unresponsive requests, Nikolov et al. (2014) presented a platform to manage resources adaptively for elastic clouds based on SLOs, which can overcome short-time request bursts. The proposed platform can adapt application execution to corresponding Service-Level Agreements (SLAs). However, faults handling is not considered in this work. Tomás et al. (2014) combined overbooking and brownout together to improve resource utilization without application degradation. Like Dürango et al. (2014), this approach is also based on admission control that gradually adapts applications according to requests.

In 2015, automatic algorithms based on brownout drew more attention. Desmeurs et al. (2015) presented an event-driven brownout technique to investigate the tradeoffs between utilization and response time for web applications. Several automatic policies based on machine learning and admission control were also introduced in this work. This work opens a direction to establish more energy-efficient cloud data centers, but the results were not evaluated under a real trace. Dupont et al. (2015) proposed another automatic approach to manage cloud elasticity in both infrastructure and software. The proposed method takes advantage of the dynamic selection of different strategies. This approach considers infrastructure level and extends the application of brownout by applying it to a broader range. Moreno et al. (2015) presented a proactive approach for latency-aware scheduling under uncertainty to accelerate decision time. The motivation is to apply a formal model to solve the nondeterministic choices of adaptation tactics (disabling different optional contents). The limitations of this work are (i) this work does not support concurrent tactics, and (ii) the uncertainty of environment predictions is not considered.

In 2016, several articles were devoted to improving the adaption of decision time. Pandey et al. (2016) proposed a hybrid selection methodology that investigates multiple optimization objectives to balance the tradeoffs between different metrics, such as decision time and optimized results. A Markov decision process is applied to find the best choice among candidates, which represents the combination of different disabled optional contents. To overcome the limitations in Moreno et al. (2015), Moreno et al. (2016) presented an approach aiming to eliminate runtime costs when constructing the Markov Decision Process (MDP). The MDP is solved via stochastic dynamic programming. The results show that the decision time is reduced significantly. However, the requests are processed in an offline manner.

Some other researchers have focused their attention on energy saving for clouds. Hasan et al. (2016) introduced a green energy-aware scheduling approach for interactive cloud applications that allows dynamic adaptation. Each application has three different modes, and each mode has a different percentage of optional contents. Xu et al. (2016) applied brownout to reduce power consumption by dynamically and selectively disabling the optional components of applications.

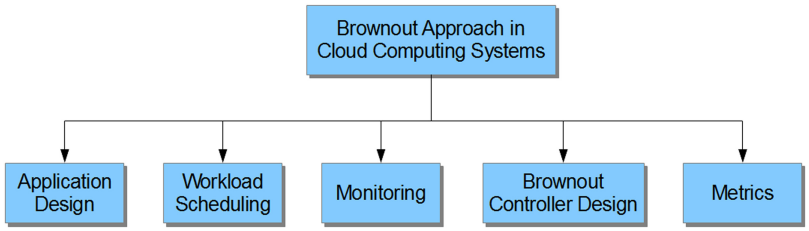

Fig. 3. Phases of applying the brownout approach in cloud computing systems.

The tradeoffs between energy and discount were investigated, and several heuristic component selection policies were also proposed.

In 2017, as an extension work of Hasan et al. (2016), Hasan et al. (2017b) investigated multiple metrics other than energy, including both user experience and performance of the interactive cloud application. An adaptive application management approach based on dynamically switching application modes was proposed to improve the tradeoffs among multiple optimization objectives. In order to improve the elasticity of the infrastructure by adding or removing resources, Hasan et al. (2017a) proposed a platform that applies a green energy-aware approach to schedule interactive applications in clouds. The proposed platform aims to utilize both infrastructure and application resources to adapt to changing workloads.

5 PHASES AND TAXONOMY OF BROWNOUT-BASED ADAPTIVE MANAGEMENT

In this section, we present the phases and a review of the brownout approach for the adaptive management of resources and applications in cloud computing systems. According to our surveyed articles, we have classified the adaptive management of resources and applications using brownout into five phases: application design, workload scheduling, monitoring, brownout controller design, and metrics, as demonstrated in Figure 3. These phases can be mapped to the MAPE-K modules as shown in Figure 1. Application design and workload scheduling correspond to the Knowledge module to describe the system; monitoring is mapped to the Monitor module to monitor system status; brownout controller design corresponds to the Analyze, Plan, and Execute modules; and metrics are mapped to the information obtained from the Sensors. Here, we explain the details of each phase.

5.1 Application Design

Applications are run in cloud computing systems to provide services for users. To enhance the system performance in clouds, applications are designed by referring to system configuration and users requirements. In Figure 4, the categories we used for the application design are (i) application type, (ii) application domain, (iii) optional parts, and (iv) application deployment.

5.1.1 Application Type. The application can be run locally or remotely while enabled by brownout. Brownout-enabled applications can be classified into two types according to application type: (i) desktop application and (ii) web application. The desktop application represents the brownout-enabled applications run locally on machines (e.g., a word processing application) (Alvares et al. 2017). The web application is implemented in the client-server model to provide web services that the users interact with through the Internet. The typical brownout-enabled web application is an online shopping system, an application that has been examined in many existing brownout-related articles (Dürango et al. 2014; Klein et al. 2014a; Maggio et al. 2014).

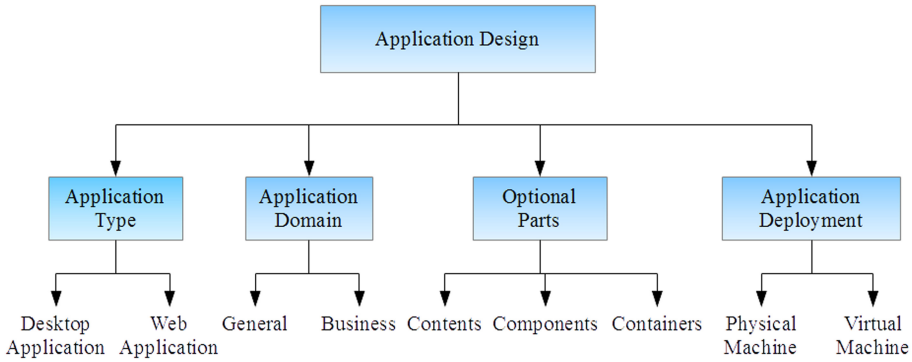

Fig. 4. Taxonomy based on application design.

5.1.2 Application Domain. Applications are implemented to provide functionalities for users. The developer's aim is to develop applications efficiently, even as the complexity of applications is increasing. Adaptive application management in cloud computing systems provides an approach to manage these complex applications. To manage these applications, the application's domain should be identified as applications in different domains have different management requirements. Applications in the general domain provide functions for general purposes (Xu et al. 2016), such as scientific calculation applications. Applications in the business domain, which focus on maximizing the profits of service providers, are more sensitive to certain specific performance metrics. For example, in an online shopping system, which belongs to the business domain, response time as a QoS requirement is one of the most critical metrics (Hasan et al. 2017b), since long response times or unresponsiveness leads to the loss of users.

5.1.3 Optional Parts. In brownout-enabled applications, the optional parts in the applications are temporarily deactivated to manage resources and applications. The optional parts represent the scheduling units in the applications. In existing articles, the optional parts are identified as (i) contents, (ii) components, and (iii) containers. Optional web contents on servers are to be showed selectively to users to save resource usage (Klein et al. 2014a). Components-based applications deactivate optional components to manage resource utilization (Alvares et al. 2017). In containerized clouds, each service is implemented as a container, and the optional containers can be activated/deactivated based on system status (Xu et al. 2016).

5.1.4 Application Deployment. In cloud computing systems, applications can be deployed on physical machines (PMs) or virtual machines (VMs). Deploying them directly on PMs enables applications to have almost the same performance as native systems, especially containerized applications. One PM can host multiple VMs, and multiple applications can be deployed on the same VM to improve the usage of shared resources. When applications are deployed on VMs, VM migrations and VM consolidation can be applied together with brownout to optimize resource utilization (Xu et al. 2016).

5.2 Workload Scheduling

Workload scheduling aims to schedule and allocate appropriate resources to suitable workloads according to the SLA defined by end-users and service providers so that the workloads can be executed efficiently and applications utilize resources efficiently. In Figure 5, the categories we use for workload scheduling are (i) workload type, (ii) resource type, (iii) dynamicity, (iv) workload trace, and (v) experiment platform.

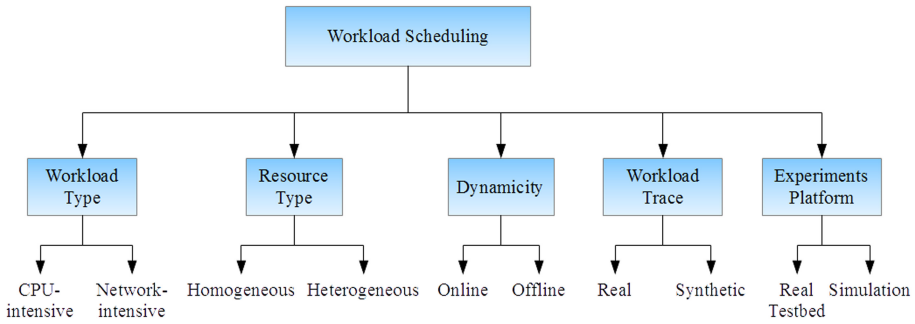

Fig. 5. Taxonomy based on workload scheduling.

5.2.1 Workload Type. The workload type represents the resource requirement of workloads. In current brownout-relevant articles, most works are focused on scheduling CPU-intensive workloads (Dürango et al. 2014; Desmeurs et al. 2015; Dupont et al. 2015) as computation resources are regarded as the main resource allocated to workloads. Some articles also consider network-intensive workloads to reduce network latency (Nikolov et al. 2014).

5.2.2 Resource Type. For homogeneous resource types, the resources offered by service providers are limited to a single type. This configuration simplifies workload scheduling and overlooks the various characteristics of a workload. The homogeneous configuration is mostly used in small-scale tests where resource diversity is limited or for the initial research of a novel approach (Dürango et al. 2014; Klein et al. 2014a). Cloud computing systems are heterogeneous; to take advantage of this feature, mature service providers offer heterogeneous resources for workload scheduling. For example, Amazon EC2 has provided more than 50 types of VMs, and these VMs are categorized into various classifications for different workload types, such as general purpose, computation-intensive, or memory-intensive (EC2 2018).

5.2.3 Dynamicity. The dynamicity of workload scheduling represents the time when the workload information is obtained. The dynamicity of workload scheduling is considered *online* if the workload information is only available when the workloads are coming into the system. If all the workload information is known in advance, and workloads can be rescheduled based on system resources, the scheduling process is identified as *offline*. One example of offline scheduling is the batch job (Hasan et al. 2016), in which the deadlines of jobs are known and jobs can be executed with delays based on resource availability. Compared with online workload scheduling, offline workload scheduling usually achieves more optimized results; however, it is not available for some scheduling scenarios, such as real-time applications.

5.2.4 Workload Trace. Different workload traces are used to evaluate the performance of brownout approaches. When brownout was initially proposed, synthetic traces, such as workloads generated based on Poisson distribution, were applied. Later, workloads derived from real traces were also applied. Presently, the popular real traces are from FIFA (FIFA 2014), Wikipedia (Wikibench 2017), and Planet lab trace (Park and Pai 2006).

5.2.5 Experiments Platform. Both real testbeds and simulations have been conducted to test the performance of brownout approaches. Experiments under real testbed are more persuasive. The Grid'5000 platform (Grid5000 2017) has been adopted in several articles; this platform provides APIs to collect PM utilization and energy consumption data. However, some uncontrolled factors exist in real testbeds, such as network traffic and unpredictable loads. Therefore, simulation

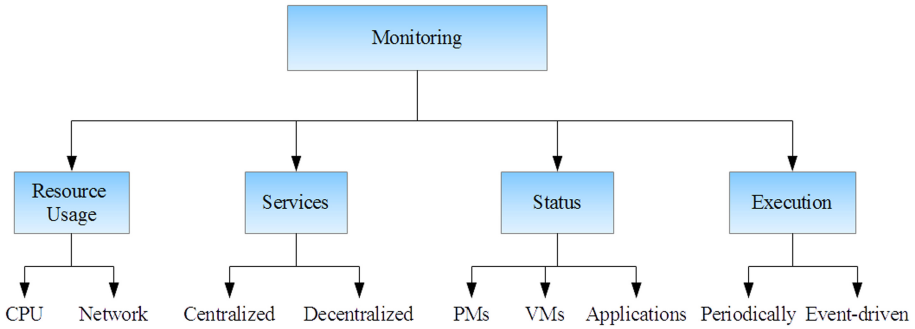

Fig. 6. Taxonomy based on monitoring.

tools provide more feasible options. In addition, with simulations, it is easier to conduct experiments with heterogeneous resources as well as at large-scale sizes. The cloud simulation toolkit, CloudSim (Buyya et al. 2009), has been used for simulating brownout-enabled workload scheduling.

5.3 Monitoring

The objective of monitoring is to achieve performance optimization by monitoring resource usage in a cloud computing system. Therefore, a monitoring component is required to collect system data and analyze resource utilization information. After analysis, decisions to change the system status and resource usage are made to ensure that the system satisfies a specific SLA. As shown in Figure 6, we categorize the components of monitoring as (i) resource usage, (ii) services, (iii) status, and (iv) execution.

5.3.1 Resource Usage. The monitor in a cloud environment is used to track resource usage, including CPU, memory, and network usage via monitoring tools. As mentioned in Section 5.2.1, current brownout-related workload scheduling is focused on handling CPU-intensive and network-intensive workloads (Klein et al. 2014b; Nikolov et al. 2014). So, monitors in brownout-enabled systems are mainly monitoring CPU and network resource usage. Two objectives of monitoring resource usage can be achieved from both users' and service providers' perspectives: users expect their requests to be processed within QoS parameters, and service providers aim to execute the workloads with minimum resource usage.

5.3.2 Services. Service monitoring gathers information about resource status to check whether the workload is being executed by applications as desired. Two types of service monitoring exist in cloud computing systems: One is centralized and the other is distributed. In a centralized moniter, a central repository is applied to store the collected data, which are not scalable when the number of monitored targets is increased (Xu et al. 2016). For distributed service monitoring, the monitored data are stored distributedly to achieve better fault tolerance and load balancing (Alvares et al. 2017).

5.3.3 Status. To be more specific, monitoring resource utilization is regarded as monitoring the status of different levels, including PMs (Moreno et al. 2015), VMs (Dupont et al. 2015), and applications (Tomás et al. 2014). Based on various optimization goals, data are collected from different levels. For example, to reduce the energy use of cloud computing systems, the power consumption of PMs should be collected. To improve the response time of requests, it is necessary to obtain the resource usage of applications.

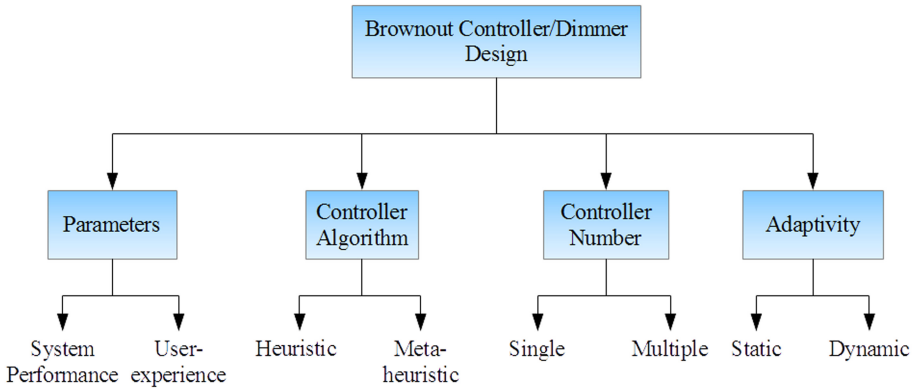

Fig. 7. Taxonomy based on brownout controller design.

5.3.4 Execution. To avoid unexpected failures and execute workloads promptly, execution monitoring has two types: periodic and event-driven. In periodic monitoring, the monitor periodically checks the resource usage and makes decisions on the execution process for the next time period (Dürango et al. 2014). In an event-driven monitor, changes in the execution process are triggered once a specific event is detected, as when resource usage is above the predefined overloaded threshold (Klein et al. 2014b). The motivation of an event-driven monitor is its real-time requirement, which is suitable for latency-aware applications.

5.4 Brownout Controller/Dimmer Design

In brownout-enabled systems, the deactivation/activation operations on optional parts are managed by a brownout controller. The brownout controller also has a control knob called a *dimmer*, which represents the probability of optional parts being deactivated. Thus, the design of the brownout controller is the most important part of a brownout approach. In Figure 7, we have categorized the classification of brownout controller/dimmer design as (i) parameters, (ii) controller algorithm, (iii) controller number, and (iv) adaptivity.

5.4.1 Parameters. A brownout controller can be designed based on different parameters. These parameters can be classified based on system and user perspectives as system performance and user experience, respectively. If system performance is addressed, the brownout controller is configured with system parameters, such as resource utilization (Dürango et al. 2014). If the brownout controller aims to optimize user experience, parameters like response time can be designed into a brownout controller (Klein et al. 2014a).

5.4.2 Controller Algorithm. Similar to the resource scheduling problem, finding the optimal solutions of which optional parts are to be deactivated/activated by controller algorithms is computationally expensive. Thus, finding approximate solutions is an alternative to most proposed approaches. We have classified the surveyed controller algorithms as heuristic and meta-heuristic. The heuristic algorithms comply with predefined constraints and try to find an acceptable solution for a particular problem (Talbi 2009). Usually, these constraints are varied for different problems, and the solutions can be obtained within a limited time. One type of heuristic algorithms is the greedy algorithm, and it has been adopted in several works (Desmeurs et al. 2015; Dupont et al. 2015; Moreno et al. 2015). Meta-heuristic algorithms are generally applied to general-purpose problems (Talbi 2009) and have standard procedures for problem construction

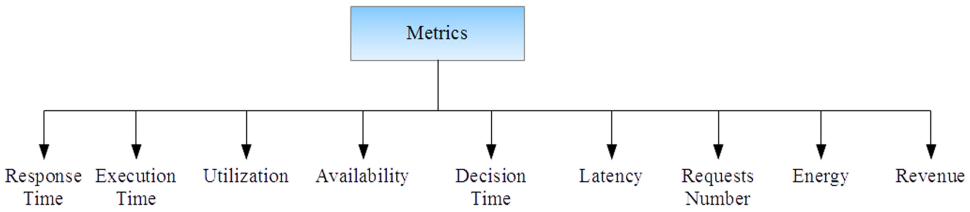

Fig. 8. Taxonomy based on metrics.

and solving. One example of a meta-heuristic algorithm is the approach based on a Markov decision process that has been applied in two works (Xu and Buyya 2017; Moreno et al. 2016).

5.4.3 Controller Number. The brownout controller may have single or multiple controllers to manage the optional parts in cloud computing systems. In Tomás et al. (2014), multiple controllers are applied in each application; there is a controller, and its dimmer value is calculated based on its status. Thus, the dimmer values of different applications in Tomás et al. (2014) can be varied. For overall management, a single controller is applied. For example, in Hasan et al. (2017a), the dimmer value is calculated as the severity of overloads in the whole data center.

5.4.4 Adaptivity. Adaptivity represents whether the brownout controller is adaptive to changes in workloads. It is categorized as static and dynamic. In our surveyed approaches, most controllers are dynamic (Dürango et al. 2014; Maggio et al. 2014; Tomás et al. 2014) and can be dynamically adapted to changes in the system. Only limited approaches are static (i.e., apply static parameters). The static brownout controllers easily violate specific SLAs (Hasan et al. 2016; Hasan et al. 2017a).

5.5 Metrics

Different metrics are considered in cloud computing systems to evaluate the performance of different brownout-based approaches. As shown in Figure 8, from our surveyed literature, we have identified nine metrics: response time, execution time, utilization, availability, decision time, latency, request number, energy, and revenue.

Response time (Dürango et al. 2014; Klein et al. 2014a; Maggio et al. 2014) is the total time elapsed from when users send requests until they receive a response. The response time can be influenced by system processing time, which is also relevant to hardware resource utilization. This metric should be reduced to improve the user experience. *Execution time* (Alvares et al. 2017; Zhao et al. 2017) is the time required to complete the workload's execution. Minimizing the execution time can improve system QoS. *Utilization* (Nikolov et al. 2014; Tomás et al. 2014) is the actual resource percentage used to run workloads compared to the total resources provided by the service provider. Available utilization should be improved to maximize resource usage. *Availability* (Alvares et al. 2017; Zhao et al. 2017) is the capability of the cloud computing system to guarantee that services are available with expected performance as well as its ability to handle fatal situations. This metric should be ensured in cloud computing systems (e.g., for 99.9% time, services should run with expected performance). *Decision time* (Moreno et al. 2015; Moreno et al. 2016) is the time that a scheduling algorithm takes to find the optional parts to be deactivated/activated, which is associated with algorithm design. To find solutions faster, in the online scheduling scenario, the decision time of an algorithm should be accelerated. *Latency* (Klein et al. 2014a; Moreno et al. 2015) is the time delay spent on message communication across the network, which is relevant to hardware resources and utilization. In business applications, latency is a crucial metric to indicate system performance. *Request number* (Dupont et al. 2015; Hasan et al. 2016) is the number of requests received by the system. The scheduling algorithm has better performance if more requests can be

Table 2. Summary of Brownout Approaches

Approach	Year	Author	Description	Orgranization
COBLB (Dürango et al. 2014)	2014	Durango et al.	Load Balancing with Brownout and Control Theory	Umea University, Sweden
SAB (Klein et al. 2014a)	2014	Klein et al.	Robust Cloud Applications with Brownout	Umea University, Sweden
EPBH (Klein et al. 2014b)	2014	Klein et al.	Resilience and Load Balancing with Brownout	Umea University, Sweden
FPF (Maggio et al. 2014)	2014	Maggio et al.	Brownout Prediction	Lund University, Sweden
CLOUDFARM (Nikolov et al. 2014)	2014	Nikolov et al.	Adaptive Resource Management	University of Ulm, Germany
BOB (Tomás et al. 2014)	2014	Tomas et al.	Overbooking with Brownout	Umea University, Sweden
EDB (Desmeurs et al. 2015)	2015	Desmeurs et al.	Event-Driven Application with Brownout	Umea University, Sweden
CAA (Dupont et al. 2015)	2015	Dupont et al.	Cloud Elasticity with Brownout	INRIA, France
PLA (Moreno et al. 2015)	2015	Moreno et al.	Proactive Self-Adaptation for Uncertainty Environment	Carnegie Mellon University, USA
HYB-Q (Hasan et al. 2016)	2016	Hasan et al.	Green Energy for Cloud Application	INRIA, France
PLA-SDP (Moreno et al. 2016)	2016	Moreno et al.	Decision-Making for Proactive Self-Adaptation	Carnegie Mellon University, USA
HYBP (Pandey et al. 2016)	2016	Pandey et al.	Hybrid Planning in Self-Adaptation	Carnegie Mellon University, USA
LUFCS (Xu et al. 2016)	2016	Xu et al.	Energy Efficiency in Clouds with Brownout	University of Melbourne, Australia
MODULAR (Alvares et al. 2017)	2017	Alvares et al.	Modular Autonomic Manager in Software Components	INRIA, France
SaaScaler (Hasan et al. 2017b)	2017	Hasan et al.	Power and Performance Scaler for Applications	INRIA, France
GPaaScaler (Hasan et al. 2017a)	2017	Hasan et al.	Green Energy Aware Scaler for Interactive Applications	INRIA, France
RLBF (Zhao et al. 2017)	2017	Zhao et al.	Reinforcement Learning for Software Adaptation	Peking University, China
BMDP (Xu and Buyya 2017)	2017	Xu et al.	Energy Efficient Clouds with Markov Decision Process and Brownout	University of Melbourne, Australia

served with the same amount of resources. *Energy* (Xu et al. 2016; Xu and Buyya 2017) is the total power used for the cloud computing system to provide its services to users. The system should reduce energy consumption while ensuring QoS. *Revenue* (Hasan et al. 2017a; Hasan et al. 2016) is the profit obtained from users when service providers are offering services. The service provider aims to maximize its revenue while lowering costs.

6 REVIEW OF BROWNOUT APPROACHES

In this section, we conduct a review of brownout-based approaches for adaptive management of resources and applications in a cloud computing system. To identify the differences in surveyed articles, we use the taxonomy in Section 5 to map the key features of these approaches. Table 2 shows a summary of selected brownout approaches, and Tables 3 to 7 summarize the comparison of selected brownout approaches and their categorized classification according to our taxonomy. For

Table 3. Taxonomy Based on Application Design

Approach	Application Type	Application Domain	Optional Parts	Application Deployment
COBLB	Web application	Business	Contents	Virtual Machine
SAB	Web application	Business	Contents	Virtual Machine
EPBH	Web application	Business	Contents	Virtual Machine
FPF	Web application	Business	Components	Physical Machine
CLOUDFARM	Web application	Business	Components	Physical Machine
BOB	Web application	Business	Contents	Virtual Machine
EDB	Web application	Business	Contents	Virtual Machine
CAA	Web application	Business	Contents	Virtual Machine
PLA	Web application	Business	Contents	Virtual Machine
HYB-Q	Web application	Business	Contents	Virtual Machine
PLA-SDP	Web application	Business	Contents	Virtual Machine
HYBP	Web Application	Business	Contents	Virtual Machine
LUFCS	Web Application	General	Components	Virtual Machine
MODULAR	Desktop Application	General	Components	Physical Machine
SaaScaler	Web application	Business	Contents	Virtual Machine
GPaaScaler	Web application	Business	Contents	Virtual Machine
RLBF	Web application	Business	Components	Physical Machine
BMDP	Web Application	General	Containers	Virtual Machine

Table 4. Taxonomy Based on Workload Scheduling

Approach	Workload Type	Resource Type	Dynamicity	Trace	Experiments Platform
COBLB	CPU-intensive	Homogeneous	Online	Synthetic	Simulation
SAB	CPU-intensive	Homogeneous	Online	Synthetic	Real testbed
EPBH	CPU-intensive	Homogeneous	Online	Synthetic	Real testbed
FPF	CPU-intensive	Homogeneous	Online	Synthetic	Simulation
CLOUDFARM	Network-intensive	Heterogeneous	Online	Synthetic	Real testbed
BOB	CPU-intensive	Homogeneous	Online/Offline	Real	Real testbed
EDB	CPU-intensive	Homogeneous	Online	Synthetic	Real testbed
CAA	CPU-intensive	Homogeneous	Offline	Synthetic	Real testbed
PLA	CPU-intensive	Homogeneous	Online	Real	Real testbed
HYB-Q	CPU-intensive	Homogeneous	Online	Real	Real testbed
PLA-SDP	CPU-intensive	Homogeneous	Offline	Real	Real testbed
HYBP	CPU-intensive	Heterogeneous	Online	Real	Simulation
LUFCS	CPU-intensive	Heterogeneous	Online	Real	Simulation
MODULAR	CPU-intensive	Homogeneous	Online	Synthetic	Real testbed
SaaScaler	CPU-intensive	Homogeneous	Online	Real	Real testbed
GPaaScaler	CPU-intensive	Homogeneous	Online	Real	Real testbed
RLBF	CPU-intensive	Homogeneous	Online/Offline	Synthetic	Real testbed
BMDP	CPU-intensive	Heterogeneous	Online	Real	Simulation

instance, Table 3 shows the comparison based on the taxonomy of application design as presented in Section 5.1, and Table 7 shows the metrics comparison based on the discussion in Section 5.5.

The Convex Optimization-Based Load Balancing (COBLB) (Dürango et al. 2014) technique extends the brownout approach for services with multiple replicas, which are copies of applications

Table 5. Taxonomy Based on Monitoring

Approach	Resource Usage	Services	Status	Execution
COBLB	CPU	Centralized	Applications	Periodically
SAB	CPU	Centralized	Applications	Periodically
EPBH	CPU	Centralized	Applications	Event-Driven
FPF	CPU	Centralized	Applications	Periodically
CLOUDFARM	Network	Centralized	PMs, Applications	Periodically
BOB	CPU	Centralized	Applications	Periodically
EDB	CPU	Centralized	Application	Event-Driven
CAA	CPU	Centralized	VMs	Periodically
PLA	CPU	Centralized	PMs	Periodically
HYB-Q	CPU	Centralized	Applications	Periodically/Event-Driven
PLA-SDP	CPU	Centralized	PMs	Periodically
HYBP	CPU	Centralized	PMs	Periodically
LUFCS	CPU	Centralized	PMs	Periodically
MODULAR	CPU	Decentralized	PMs	Periodically/Event-Driven
SaaScaler	CPU	Centralized	VMs, Applications	Periodically/Event-Driven
GPaaScaler	CPU	Centralized	VMs, Applications	Periodically
RLBF	CPU	Centralized	PMs	Periodically
BMDP	CPU	Centralized	PMs	Periodically

Table 6. Taxonomy Based on Brownout Controller/Dimmer Design

Approach	Parameters	Controller Algorithm	Controller Number	Adaptivity
COBLB	System Performance	Heuristic	Multiple	Dynamic
SAB	User-experience	Heuristic	Multiple	Dynamic
EPBH	System Performance	Heuristic	Single	Dynamic
FPF	User-experience	Heuristic	Single	Dynamic
CLOUDFARM	System Performance	Heuristic	Multiple	Dynamic
BOB	System Performance	Heuristic	Multiple	Static/Dynamic
EDB	System Performance	Heuristic	Multiple	Dynamic
CAA	System Performance	Heuristic	Multiple	Dynamic
PLA	System Performance	Meta-heuristic	Single	Dynamic
HYB-Q	User-experience	Heuristic	Single	Static
PLA-SDP	System Performance	Meta-heuristic	Single	Static
HYBP	System Performance	Meta-heuristic	Single	Dynamic
LUFCS	System Performance	Heuristic	Multiple	Dynamic
MODULAR	System Performance	Heuristic	Multiple	Dynamic
SaaScaler	User-experience	Heuristic	Single	Static
GPaaScaler	User-experience	Heuristic	Single	Static
RLBF	System Performance	Meta-heuristic	Multiple	Static/Dynamic
BMDP	System Performance	Meta-heuristic	Multiple	Dynamic

providing the same functionalities. These replicas contain optional contents and can be served selectively according to system status. To enhance system load balancing performance, the technique also collects information about adaptation in replicas. In this technique, all replicas are managed in queuing systems adopting resource sharing. Response time and the number of requests are improved by this technique.

Table 7. Taxonomy Based on Metrics

Approach	Response Time	Execution Time	Utilization	Availability	Decision Time	Latency	Requests Number	Energy	Revenue
COBLB	✓						✓		
SAB						✓	✓		
EPBH	✓						✓		
FPF	✓						✓		
CLOUDFARM			✓						
BOB	✓		✓						
EDB	✓		✓						
CAA	✓						✓		
PLA					✓	✓			
HYB-Q	✓						✓	✓	✓
PLA-SDP					✓	✓			
HYBP	✓				✓				
LUFCS								✓	✓
MODULAR		✓		✓					
SaaScaler	✓						✓	✓	✓
GPaaScaler	✓							✓	✓
RLBF		✓		✓					
BMDP								✓	✓

The Self-Adaptive Brownout (SAB) (Klein et al. 2014a) approach adds a dynamic parameter that affects user experience and the amount of resources required by the application. This parameter is adapted based on both workload and available resources to enhance the robustness of applications. The proposed approach is synthesized with a control theory approach to adaptively determine when to activate/deactivate optional features in the applications. With control theory, application behaviors can be predicted, and some system constraints can be guaranteed. In this approach, the maximum latency is controlled so that latency is reduced. Also, SAB can serve more users with fewer resources.

Equality Principle-Based Heuristic (EPBH) (Klein et al. 2014b) is focused on enhancing the resilience of cloud services when a system faces a resource shortage. Similar to COBLB (Dürango et al. 2014), this approach is also applied to application replicas. This event-driven approach is based on heuristics and requires all replicas to have a control parameter (dimmer) value. According to the parameter value, EPBH decides which replicas to receive more load. EPBH has been proved to improve resilience when a resource shortage is triggered by failures.

The objective of the Feedforward Plus Feedback (FPF) (Maggio et al. 2014) approach is to keep the average response time below a certain threshold. FPF works by configuring the probability to serve requests with optional computations. FPF is able to handle requests bursts for cloud applications and reacts to changes in the resource amount allocated to cloud applications. The goal of FPF is to obtain the number of currently queued requests from applications and predict the latency experienced by future requests. The results demonstrate that although FPF spends effort on obtaining more information, the overloads are mitigated.

CLOUDFARM (Nikolov et al. 2014) is an elastic cloud platform to manage resource reservations in flexible and adaptive ways based on actual resource demands and predefined SLAs. It provides flexible SLAs that can be configured dynamically to fulfill the elasticity of cloud computing systems. Based on SLAs and costs produced by users, the services can be dynamically downgraded

to avoid bursts so that system utilization and revenue can be improved. CLOUDFARM also introduces an abstract level by using a virtual framework for all applications, which benefits from its lightweight and dynamic degradation from full mode to degraded mode.

The Brownout OverBooking (BOB) (Tomás et al. 2014) technique is designed to ensure graceful degradation when loads spike and thus avoid overloads in a resource overbooking scenario. The motivation of BOB is to combine overbooking and brownout together, where the overbooking system takes advantage of application information from brownout and applies deactivation operations to relieve overloads. In BOB, an overbooking controller is responsible for admitting or rejecting new requests, and a brownout controller is responsible for adapting the resources allocated to accepted requests. Higher utilization is achieved while response time is ensured by BOB.

To ensure application responsiveness, an Event-Driven Brownout (EDB) (Desmeurs 2015) technique at the application level has been proposed. In this approach, the application is allowed to run optional codes that are not necessary for key functionalities but that provide extra user experience. EDB combines machine learning techniques and control theory to dynamically configure a given threshold that represents the number of pending requests. The configuration operation is based on application response time. An advance in ensuring response time is achieved without sacrificing utilization.

The Cloud Automatic Approach (CAA) (Dupont et al. 2015) aims at managing cloud elasticity in a cross-layered manner. The motivation is to combine infrastructure elasticity and software elasticity to overcome the conceptual limitations of the IaaS and make software at the SaaS level involved in the elasticity process. Software at SaaS is running at different levels and can be degraded to lower levels when resources are limited. And for resources at the IaaS level, physical machines can be scaled in and out according to system loads. An advantage in response time is obtained when the cross-layered method is working in a coordinated way.

The Proactive Latency-Aware (PLA) (Moreno et al. 2015) method addresses the inefficient reactive adaption problem when adaptation has latency. Thus, PLA considers adaptation latency and applies the probabilistic model to decide adaptations. The main motivation is to use a formal model to find adaptation decisions which are underspecified through nondeterminism and then resolve the nondeterministic choices to maximize optimization goals. PLA takes the uncertainty of the environment into account and predicts needed resources by looking ahead. The effectiveness of adaptation decisions is significantly improved.

The QoS-aware Hybrid Controller (HYB-Q) (Hasan et al. 2016) technique aims at achieving green energy-aware scheduling by using both green and brown energy. The target applications are focused on interactive cloud applications that can be dynamically adapted to available green energy based on changing conditions. HYB-Q obtains information in a feedback loop about the number of requests executed under different modes in the previous time period and computes the current adaptation value. In the controller, the response time and workload changes are periodically checked. If the response time rises above the predefined threshold, the user experience is downgraded to a lower mode to avoid overloads by the controller. It is observed that providers' revenue is improved and the usage of brown energy is reduced.

As an extension work of PLA (Moreno et al. 2015), the Proactive Latency-Aware with Markov Decision Process (PLA-SDP) (Moreno et al. 2016) takes advantage of the probabilistic property of the Markov decision process to adapt application functionalities. To eliminate the runtime overhead of constructing MDP, stochastic dynamic programming is applied to solve MDP. The decision time is vitally reduced while the same results are obtained.

The objective of Hybrid Planning (HYBP) (Pandey et al. 2016) is to find tradeoffs between timeliness and optimality of solutions to adapt application modes. HYBP combines two approaches

together to achieve the best balance between conflicts. One is deterministic planning and the other is Markov decision process planning, which can be fitted in different cases. In case a fast response is required, deterministic planning is applied. When the time is adequate, MDP planning is applied to generate an optimal solution. Simulated experiments have shown that HYBP can improve system performance by balancing these tradeoffs.

The Lowest Utilization First Component Selection (LUFCS) policy (Xu et al. 2016) is a heuristic technique to save data center power usage via dynamically deactivating application components. Those components with lower utilization are selected with higher priority until the sum of these components' utilization equals the expected utilization reduction. This technique is combined with VM consolidation to decrease the active number of hosts. The discount is also considered for users if some services are not provided. Therefore, there is a tradeoff between saved energy and revenue, which is investigated by LUFCS.

The MODULAR (Alvares et al. 2017) approach leverages the modularity feature of the component-based system to strengthen domain-specific language applications. Domain-specific language has high-level constructs to describe the configurations and executed policies of the target system. With the brownout mechanism, the components can be transferred from a nominal configuration to a degraded one, which incurs different loads for the system. Thus, through dynamic reconfiguration, the system becomes self-adaptive.

Compared with HYB-Q (Hasan et al. 2016) that focuses on green energy usage, the SaaScaler (Hasan et al. 2017b) approach analyzes the tradeoff between power and performance by considering green energy usage for interactive cloud applications. SaaScaler can satisfy different metrics. Considering these metrics, three levels of user experience are defined, and capacity requirements are dynamically adjusted among these levels to serve more users. Experiments conducted in real testbed show that energy consumption is reduced while performance and revenues are improved by carefully tuning applications.

GPaaScaler (Hasan et al. 2017a) considers adaptation both at the infrastructure and application levels by using a green energy source. At the infrastructure level, resources are added or removed according to applications' resource demand. While the applications are dynamically changing their service levels according to performance and green energy availability, these adaptations are implemented separately and can be coordinated. Lower costs and less usage of brown energy are achieved.

The objective of the Reinforcement Learning-Based Framework (RLBF) (Zhao et al. 2017) approach is to overcome the limitations of rule-based adaptation in which decisions are only made based on static rules. Based on reinforcement learning, RLBF enables automatic learning rules with various optimization objectives in an offline manner. Additionally, the rules can also evolve according to real-time information for online adaptation in selecting optional services to run. The efficiency and effectiveness of the adaptation process are improved by this approach.

Aiming at improving the tradeoffs mentioned in Xu et al. (2016), another brownout-based approach was proposed using the Markov Decision Process (Xu and Buyya 2017). Called BMDP, it aims to select better combinations of deactivated application components. With MDP, the solution space is increased and more possible solutions are found. To reduce the solution space and avoid the curse of dimensioning, key states that can reduce energy consumption are identified. Taking advantage of MDP, a better tradeoff between energy and revenue is fulfilled.

To summarize the merits and demerits of the reviewed brownout approaches, a comparison of the advantages and limitations of each brownout approach is presented in Table 8. For example, the COBLB (Dürango et al. 2014) approach achieves better resilience but the adopted application model is not generalizable.

Table 8. Comparison of Brownout Approaches

Approach	Objectives	Advantages	Limitations
COBLB	Load balancing for Cloud applications	Resilience is improved	Application model is not general
SAB	Enhance robustness of Cloud applications	Support more users with fewer resources	Only tested on a single machine
EPBH	Improve resilience	Response time is reduced	Parameters are chosen empirically
FPF	Mitigate overloads	Prediction is applied	Only validated with simulations
CLOUDFARM	Improve elasticity	SLA is ensured when there are bursts	Faults handling is not considered
BOB	Resource overbooking	Improve resource utilization without application degradation	Scalability is not discusses
EDB	Balance utilization and response time	Utilization is improved and response time is ensured	Not evaluated with real workloads
CAA	Manage elasticity	Elasticity is improved at both infrastructure and software levels	Limited number of tactics
PLA	Accelerate decision time	Latency is reduced	Concurrent tactics are not supported
HYB-Q	Green energy provisioning for interactive cloud application	Brown energy is saved	Non-interactive cloud applications are not investigated
PLA-SDP	Accelerate decision time	Decision time is reduced	Not available for online requests
HYBP	Balance decision time and optimality	Decision time is reduced	Not validated in real testbed
LUFCS	Energy efficient management of application at component level	Energy consumption is reduced	Not all combinations of deactivated components are accessed
MODULAR	Manage modular components	System availability is improved	The availability for other applications is not discussed
SaaScaler	Investigate tradeoffs between energy and performance	Energy is reduced and QoS is improved	Resources addition or removal are not flexible
GPaaScaler	Reduce energy consumption	Energy is saved	Not available for real-time requests
RLBF	Enhance flexibility of approach	Better adaptation effectiveness is achieved	Convergence time is not analyzed
BMDP	Improve trade-off between energy and discount	Trade-off between energy and discount is improved	Not validated in real testbed

7 A PERSPECTIVE MODEL

The brownout approach has shown its ability to improve applications and resources management to handle changes in workloads. However, the tradeoffs between workload handling and performance degradation must be considered. Several optimization objectives are usually considered together to investigate these tradeoffs. The primary research problems are:

1. How do we enable a brownout approach in a cloud computing system?
2. How do we enable brownout to manage resources and applications adaptively?
3. How do we balance the tradeoffs between different metrics when applying brownout?

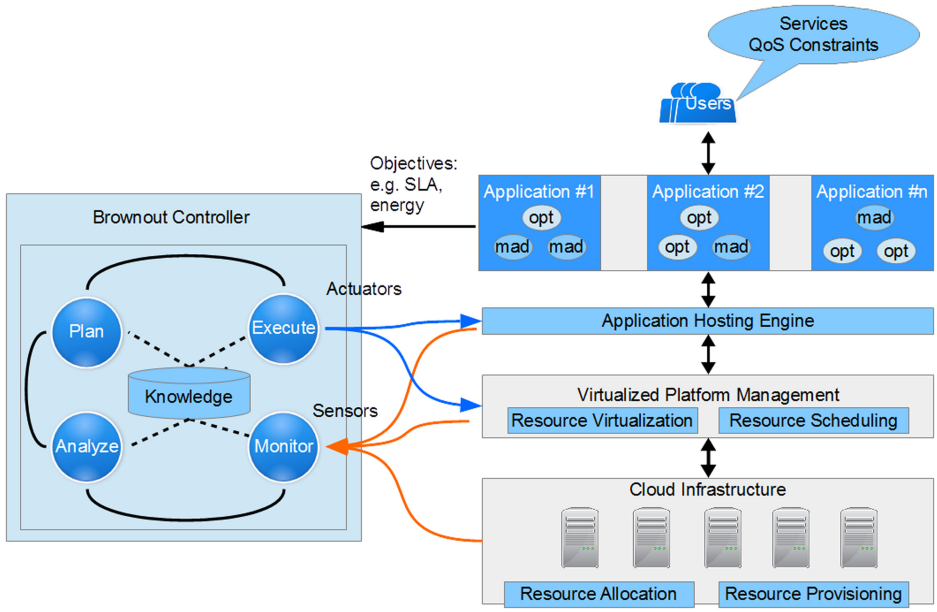

Fig. 9. Perspective model of applying brownout in cloud computing systems.

To deal with these issues, we need a brownout-enabled mechanism for adaptive management. We propose a perspective model of a brownout-enabled cloud computing system for adaptive resource scheduling as shown in Figure 9.

From the users' perspective, users interact with the system and submit their requests for services to the system. The users may have constraints for the submitted requests, such as QoS constraints or budget. From the service providers' perspective, these workloads generated by users are executed by applications.

Applications are provided by service providers to offer services to users, and these applications are managed by the application hosting engine (e.g., Docker (Docker 2017b) and Apache Tomcat (Tomcat 2018)). Applications can be deployed on either a virtualized platform (virtual resources) or a cloud infrastructure (physical resources). The host application engine can be a container-based management platform (e.g., Docker Swarm (Docker 2018), Kubernetes (2018), or Mesos (2018)), which provides management of container-based applications. A virtualized platform manages virtualized resources (e.g., the VMs managed by VMware (2018)). As for resource allocation and provisioning in a cloud infrastructure, these can be managed by infrastructure management platforms, such as OpenStack (2018).

To deal with research problem (1), the applications can be composed of optional and mandatory components/services. The optional components/services should be identified by service providers and can be activated/deactivated by the brownout controller. For instance, in Docker Swarm, the services can be deployed via a configuration file (Docker 2017a) and the file can set the service with the feature "optional."

To resolve research problem (2), a brownout controller is required based on the MAPE-K architecture model, and it fits into the feedback loop of the MAPE-K model, as introduced in Figure 1. As described in Section 4.1.2, it has modules (Monitor, Analyze, Plan, and Execute) to achieve adaptation with the cloud computing system. Sensors and Actuators are used to interact with the

cloud computing system. Sensors collect information from different levels in the cloud computing system, including the application hosting engine, virtualized platform, and cloud infrastructure. The Sensors devices can be attached to hardware (e.g., power meter). The collected information is provided to the Monitor module.

After analyzing the information in Analyze module, the Plan module makes decisions for applying brownout control operations in which the brownout-based scheduling policies are implemented. Based on these decisions, the Execute module applies brownout via actuators to the application hosting engine and the virtualized platform to enable/disable optional components/services. These operations can be fulfilled via the APIs provided by the application hosting engine or virtualized platform.

To handle research problem (3), the Knowledge pool in MAPE-K model is applied to store the predefined objectives (e.g., load balancing, energy efficiency, or SLA constraints) and tradeoffs (e.g., between energy use and SLA). The rules in the Knowledge pool, such as SLA rules, can be updated according to brownout-based policies.

8 FUTURE RESEARCH DIRECTIONS

Although significant progress has been achieved in applying brownout to cloud computing systems and the adaptive management of resources and applications is improved, there are still some research gaps and challenges in this area to be further explored. They are discussed here.

- For resource management using brownout, current works mostly focus on management of computation resources. More resource types, like memory, network, and storage, need to be considered as parameters to form more comprehensive resource management. For example, the brownout approach can be applied to manage data-intensive applications.
- Brownout has been applied to optimization goals including load balancing and energy efficiency. In addition to these optimization goals, other goals such as more complicated cost-aware resource scheduling scenarios and more QoS metrics can be applied with brownout.
- Presently, there is a lack of standard benchmarks for performance evaluation of brownout-based approaches. A benchmark is needed to test the performance of new algorithms and compare these with other approaches having the same optimization objective.
- The scalability of the brownout controller can be improved by using distributed controllers. The centralized controller can be a bottleneck for the whole system. Therefore, distributed controllers design can be investigated.
- The cost model could be improved by considering more parameters. An integrated model considering more cost-aware parameters will be attractive for service providers.
- Most experiments are tested on RUBiS, but more prototype systems based on target systems other than RUBis should be introduced to show the broader applicability of brownout.
- Rather than considering a single data center, the brownout approach for cloud computing systems can be extended to multiple data centers to deal with multicloud challenges, such as geographical load balancing. The knowledge pool in the perspective model can be used to track the availability of the system when the system is scaled.
- The decision and execution times of the brownout-based algorithm to find optional parts for deactivation can be reduced by using machine learning methods. For example, based on response time constraints, requests can be clustered by machine learning algorithms (K-means) for further execution on the same type of machines to improve resource usage.
- The overhead and runtime complexity of heuristic and meta-heuristic approaches is not discussed in our surveyed works, but should be investigated.

We summarize the future research directions identified as follows:

- **Integration with containers:** Containers have the flexible ability to provide services with isolated functions, and their quick start and stop operations enable the brownout approach to work efficiently. Therefore, integrating brownout with containers is a promising direction to improve scheduling performance.
- **Exploring more scenarios:** Brownout has been used for load balancing, energy efficiency, latency-awareness, and cost-awareness in cloud computing systems. Additionally, more scenarios, such as sustainable cloud computing systems with brownout, can be investigated to reduce carbon emissions.
- **Combined with existing approaches:** Brownout can be combined with VM consolidation to achieve better resource management effects (Xu et al. 2016; Xu and Buyya 2017). It is reasonable to expect better results when combining brownout with other validated resource management techniques, such as DVFS for energy efficiency.
- **Apply brownout with other application models:** As current works primarily focus on web applications, it is important to explore how brownout approaches can be applied in other application composition models such as Map-Reduce, bag of tasks application, and stream processing.
- **Implementing the perspective model:** Figure 9 shows a perspective model to resolve the research problems when applying brownout in cloud computing systems. Implementing a prototype system based on this model significantly advances research in the brownout arena. To implement the perspective model, some open source software (e.g., Docker, OpenStack) can be applied.

9 SUMMARY AND CONCLUSION

Brownout is a novel paradigm for self-adaptation that enables optional parts in the system to be temporarily disabled in order to deal with changing situations. In addition, it has been shown to be a promising approach to improve system performance and user experience. This article introduces a review and taxonomy of brownout-based adaptive resource and application management for cloud computing systems. A taxonomy is presented according to five phases: (i) application design, (ii) workload scheduling, (iii) monitoring, (iv) a brownout controller/dimmer design, and (v) metrics. The taxonomy of each phase is summarized based on the different classifications of brownout approaches. Then, a comprehensive review of existing brownout approaches is presented. Furthermore, a comparison of the advantages and limitations of the surveyed brownout approaches is also made. Finally, future directions for and open challenges to brownout-based approaches to managing resources and applications in cloud computing systems are given.

Our analysis shows that brownout can be applied in cloud computing systems to meet different optimization objectives. The brownout approach also provides a new option for adaptive management of resources and applications. This article shows the progress of brownout since it was first applied to clouds, and it also helps readers to understand the research gap existing in this area. One promising direction is to combine brownout with other existing techniques to achieve better system performance.

ACKNOWLEDGMENTS

We thank anonymous reviewers for their excellent comments on improving the paper. We also thank Dr. Sukhpal Singh Gill, Jungmin Jay Son, and Shashikant Ilager for their suggestions on improving this paper.

REFERENCES

Frederico Alvares, Gwenaël Delaval, Eric Rutten, and Lionel Seinturier. 2017. Language support for modular autonomic managers in reconfigurable software components. In *Proceedings of the 2017 IEEE International Conference on Autonomic Computing*. IEEE, 271–278.

Paolo Arcaini, Elvinia Riccobene, and Patrizia Scandurra. 2015. Modeling and analyzing MAPE-K feedback loops for self-adaptation. In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 13–23.

Mohammad Sadegh Aslanpour, Mostafa Ghobaei-Arani, and Adel Nadjaran Toosi. 2017. Auto-scaling web applications in clouds. *Journal of Network Computer Applications* 95 (2017), 26–41.

Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. 2012. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems* 28, 5 (2012), 755–768.

Anton Beloglazov and Rajkumar Buyya. 2013. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Transactions on Parallel and Distributed Systems* 24, 7 (2013), 1366–1379.

Rajkumar Buyya, Rodrigo N Calheiros, Jungmin Son, Amir Vahid Dastjerdi, and Young Yoon. 2014. Software-defined cloud computing: Architectural elements and open challenges. In *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics*. 1–12.

Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N. Calheiros. 2009. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In *Proceedings of the International Conference on High Performance Computing and Simulation*. 1–11.

Dazhao Cheng, Jia Rao, Changjun Jiang, and Xiaobo Zhou. 2016. Elastic power-aware resource provisioning of heterogeneous workloads in self-sustainable datacenters. *IEEE Transactions on Computer* 65, 2 (2016), 508–521.

Rogério De Lemos, Holger Giese, Hausi A. Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M Villegas, Thomas Vogel, et al. 2013. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*, Rogério De Lemos, Holger Giese, Hausi A. Müller, and Mary Shaw (Eds.). 1–32.

David Desmeurs. 2015. *Algorithms for Event-Driven Application Brownout*. Master Thesis, Umea University.

David Desmeurs, Cristian Klein, Alessandro Vittorio Papadopoulos, and Johan Tordsson. 2015. Event-driven application brownout: Reconciling high utilization and low tail response times. In *Proceedings of the 2015 International Conference on Cloud and Autonomic Computing*. 1–12.

Docker. 2017a. Docker Compose file version 3 reference. Retrieved March 27, 2018 from https://docs.docker.com/compose/compose-file/.

Docker. 2017b. Docker Documentation | Docker Documentation. Retrieved March 27, 2018 from https://docs.docker.com/.

Docker. 2018. Swarm mode overview | Docker Documentation. Retrieved March 27, 2018 from https://docs.docker.com/engine/swarm/.

Wanchun Dou, Xiaolong Xu, Shunmei Meng, Xuyun Zhang, Chunhua Hu, Shui Yu, and Jian Yang. 2017. An energy-aware virtual machine scheduling method for service QoS enhancement in clouds over big data. *Concurrency and Computation: Practice and Experience* 29, 14 (2017), 1–20.

Simon Dupont, Jonathan Lejeune, Frederico Alvares, and Thomas Ledoux. 2015. Experimental analysis on autonomic strategies for cloud elasticity. In *Proceedings of the 2015 IEEE International Conference on Cloud and Autonomic Computing*. 81–92.

Jonas Dürango, Manfred Dellkrantz, Martina Maggio, Cristian Klein, Alessandro Vittorio Papadopoulos, Francisco Hernández-Rodriguez, Erik Elmroth, and Karl-Erik Årzén. 2014. Control-theoretical load-balancing for cloud applications with brownout. In *Proceedings of the 2014 IEEE 53rd Annual Conference on Decision and Control*. 5320–5327.

Amazon EC2. 2018. Amazon Web Services. Retrieved March 27, 2018 from https://aws.amazon.com/ec2/.

Tammy Everts. 2016. Google: 53 longer than 3 seconds to load. Retrieved March 27, 2018 from https://www.soasta.com/blog/google-mobile-web-performance-study/.

FIFA. 2014. 1998 World Cup Web Site Access Logs - The Internet Traffic Archive. Retrieved March 27, 2018 from http://ita.ee.lbl.gov/html/contrib/WorldCup.html.

Grid5000. 2017. Grid5000:Home. Retrieved Aril 10, 2018 from https://www.grid5000.fr/mediawiki/index.php/Grid5000:Home.

M. D. Sabbir Hasan, Frederico Alvares, and Thomas Ledoux. 2017a. GPaaScaler: Green energy aware platform scaler for interactive cloud application. In *Proceedings of the 10th ACM International Conference on Utility and Cloud Computing*. 79–89.

M. D. Sabbir Hasan, Frederico Alvares, Thomas Ledoux, and Jean-Louis Pazat. 2017b. Investigating energy consumption and performance trade-off for interactive cloud application. *IEEE Transactions on Sustainable Computing* 2, 2 (2017), 113–126.

M. D. Sabbir Hasan, Frederico Alvares de Oliveira, Thomas Ledoux, and Jean-Louis Pazat. 2016. Enabling green energy awareness in interactive cloud application. In *Proceedings of the 2016 IEEE International Conference on Cloud Computing Technology and Science*. 414–422.

Soamar Homsi, Shuo Liu, Gustavo A Chaparro-Baquero, Ou Bai, Shaolei Ren, and Gang Quan. 2017. Workload consolidation for cloud data centers with guaranteed QoS using request reneging. *IEEE Transactions on Parallel and Distributed Systems* 28, 7 (2017), 2103–2116.

Didac Gil De La Iglesia and Danny Weyns. 2015. MAPE-K formal templates to rigorously design behaviors for self-adaptive systems. *ACM Transactions on Autonomous and Adaptive Systems* 10, 3 (2015), 1–31.

Neil Irwin. 2013. These 12 technologies will drive our economic future. Retrieved March 27, 2018 from https://www.washingtonpost.com/news/wonk/wp/2013/05/24/these-12-technologies-will-drive-our-economic-future/?utm_term=.e5ad3815c7eb.

Tarandeep Kaur and Inderveer Chana. 2015. Energy efficiency techniques in cloud computing: A survey and taxonomy. *ACM Computing Surveys (CSUR)* 48, 2 (2015), 1–46.

Cristian Klein, Martina Maggio, Karl-Erik Årzén, and Francisco Hernández-Rodriguez. 2014a. Brownout: Building more robust cloud applications. In *Proceedings of the 36th International Conference on Software Engineering*. 700–711.

Cristian Klein, Alessandro Vittorio Papadopoulos, Manfred Dellkrantz, Jonas Dürango, Martina Maggio, Karl-Erik Årzén, Francisco Hernández-Rodriguez, and Erik Elmroth. 2014b. Improving cloud service resilience using brownout-aware load-balancing. In *Proceedings of the 2014 IEEE 33rd International Symposium on Reliable Distributed Systems*. 31–40.

Manya Koetse. 2017. Weibo servers down after Lu Han announces new relationship. Retrieved March 27, 2018 from https://www.whatsonweibo.com/weibo-servers-lu-han-announces-new-relationship/.

Kubernetes. 2018. Production-grade container orchestration - Kubernetes. Retrieved June 12, 2018 from https://kubernetes.io/.

Hongjian Li, Guofeng Zhu, Yuyan Zhao, Yu Dai, and Wenhong Tian. 2017. Energy-efficient and QoS-aware model based resource consolidation in cloud data centers. *Cluster Computing* (2017), 1–11.

Zhenhua Liu, Minghong Lin, Adam Wierman, Steven Low, and Lachlan LH Andrew. 2015. Greening geographical load balancing. *IEEE/ACM Transactions on Networking* 23, 2 (2015), 657–671.

Tania Lorido-Botran, Jose Miguel-Alonso, and Jose A. Lozano. 2014. A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing* 12, 4 (2014), 559–592.

Martina Maggio, Cristian Klein, and Karl-Erik Årzén. 2014. Control strategies for predictable brownouts in cloud computing. *IFAC Proceedings Volumes* 47, 3 (2014), 689–694.

Yaser Mansouri, Adel Nadjaran Toosi, and Rajkumar Buyya. 2017. Data storage management in cloud environments: Taxonomy, survey, and future directions. *ACM Computing Surveys (CSUR)* 50, 6 (2017), 1–51.

Toni Mastelic, Ariel Oleksiak, Holger Claussen, Ivona Brandic, Jean-Marc Pierson, and Athanasios V. Vasilakos. 2015. Cloud computing: Survey on energy efficiency. *ACM Computing Surveys (CSUR)* 47, 2 (2015), 1–36.

Peter Mell, Tim Grance, et al. 2011. The NIST definition of cloud computing. (2011).

Mesos. 2018. Apache Mesos. Retrieved June 12, 2018 from http://mesos.apache.org/.

Alireza Sadeghi Milani and Nima Jafari Navimipour. 2016a. Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends. *Journal of Network and Computer Applications* 71 (2016), 86–98.

Bahareh Alami Milani and Nima Jafari Navimipour. 2016b. A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions. *Journal of Network and Computer Applications* 64 (2016), 229–238.

Gabriel A. Moreno. 2017. *Adaptation Timing in Self-Adaptive Systems*. PhD Thesis, Carnegie Mellon University.

Gabriel A. Moreno, Javier Cámara, David Garlan, and Bradley Schmerl. 2015. Proactive self-adaptation under uncertainty: A probabilistic model checking approach. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. 1–12.

Gabriel A. Moreno, Javier Cámara, David Garlan, and Bradley Schmerl. 2016. Efficient decision-making under uncertainty for proactive self-adaptation. In *Proceedings of the 2016 IEEE International Conference on Autonomic Computing*. 147–156.

Vladimir Nikolov, Steffen Kächele, Franz J. Hauck, and Dieter Rautenbach. 2014. Cloudfarm: An elastic cloud platform with flexible and adaptive resource management. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. 547–553.

OpenStack. 2018. Open source software for creating private and public clouds. Retrieved March 27, 2018 from https://www.openstack.org/.

Ashutosh Pandey, Gabriel A. Moreno, Javier Cámara, and David Garlan. 2016. Hybrid planning for decision making in self-adaptive systems. In *Proceedings of the 2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems*. 130–139.

Reena Panwar and Bhawna Mallick. 2015. Load balancing in cloud computing using dynamic load management algorithm. In *Proceedings of the 2015 IEEE International Conference on Green Computing and Internet of Things*. 773–778.

KyoungSoo Park and Vivek S. Pai. 2006. CoMon: A mostly-scalable monitoring system for PlanetLab. *ACM SIGOPS Operating Systems Review* 40, 1 (2006), 65–74.

Tim Biggs and Patrick Hatch. 2016. Banks, websites down as wild weather knocks out Amazon Web Services. Retrieved March 27, 2018 from http://www.afr.com/technology/banks-websites-down-as-wild-weather-knocks-out-amazon-web-services-20160605-gpc8ob.

Ashikur Rahman, Xue Liu, and Fanxin Kong. 2014. A survey on geographic load balancing based data center power management in the smart grid environment. *IEEE Communications Surveys and Tutorials* 16, 1 (2014), 214–233.

Martin Randles, David Lamb, and A. Taleb-Bendiab. 2010. A comparative study into distributed load balancing algorithms for cloud computing. In *Proceedings of the 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*. 551–556.

RUBBoS. 2005. RUBBoS: Bulletin Board Benchmark. Retrieved March 27, 2018 from http://jmob.ow2.org/rubbos.html.

RUBiS. 2009. RUBiS. RUBiS: Rice University bidding system. Retrieved March 27, 2018 from http://rubis.ow2.org/.

Altino M. Sampaio, Jorge G. Barbosa, and Radu Prodan. 2015. PIASA: A power and interference aware resource management strategy for heterogeneous workloads in cloud data centers. *Simulation Modelling Practice and Theory* 57 (2015), 142–160.

Sukhpal Singh and Inderveer Chana. 2016a. EARTH: Energy-aware autonomic resource scheduling in cloud computing. *Journal of Intelligent and Fuzzy Systems* 30, 3 (2016), 1581–1600.

Sukhpal Singh and Inderveer Chana. 2016b. QoS-aware autonomic resource management in cloud computing: A systematic review. *ACM Computing Surveys (CSUR)* 48, 3 (2016), 1–46.

Jungmin Son, Amir Vahid Dastjerdi, Rodrigo N. Calheiros, and Rajkumar Buyya. 2017. SLA-aware and energy-efficient dynamic overbooking in sdn-based cloud data centers. *IEEE Transactions on Sustainable Computing* 2, 2 (2017), 76–89.

El-Ghazali Talbi. 2009. *Metaheuristics: From Design to Implementation*. Vol. 74. John Wiley and Sons.

Luis Tomás, Cristian Klein, Johan Tordsson, and Francisco Hernández-Rodríguez. 2014. The straw that broke the camel's back: Safe cloud overbooking with application brownout. In *Proceedings of the 2014 IEEE International Conference on Cloud and Autonomic Computing*. 151–160.

Tomcat. 2018. Apache Tomcat - Welcome! Retrieved March 13, 2018 from http://tomcat.apache.org/.

VMware. 2018. VMware - Official Site. Retrieved March 27, 2018 from https://www.vmware.com/.

Denis Weerasiri, Moshe Chai Barukh, Boualem Benatallah, Quan Z. Sheng, and Rajiv Ranjan. 2017. A taxonomy and survey of cloud resource orchestration techniques. *ACM Computing Surveys (CSUR)* 50, 2 (2017), 1–41.

Wikibench. 2017. Retrieved March 15, 2018 from http://www.wikibench.eu/wiki/2007-10/.

Minxian Xu and Rajkumar Buyya. 2017. Energy efficient scheduling of application components via brownout and approximate Markov decision process. In *Proceedings of the 15th International Conference on Service-Oriented Computing*. 206–220.

Minxian Xu, Amir Vahid Dastjerdi, and Rajkumar Buyya. 2016. Energy efficient scheduling of cloud application components with brownout. *IEEE Transactions on Sustainable Computing* 1, 2 (2016), 40–53.

Minxian Xu, Wenhong Tian, and Rajkumar Buyya. 2017. A survey on load balancing algorithms for virtual machines placement in cloud computing. *Concurrency and Computation: Practice and Experience* 29, 12 (2017), 4123–4138.

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 Computing Surveys (CSUR)* 47, 4 (2015), 1–33.

Yunqi Zhang, Michael A. Laurenzano, Jason Mars, and Lingjia Tang. 2014. Smite: Precise QoS prediction on real-system smt processors to improve utilization in warehouse scale computers. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*. 406–418.

Tianqi Zhao, Wei Zhang, Haiyan Zhao, and Zhi Jin. 2017. A reinforcement learning-based framework for the generation and evolution of adaptation rules. In *Proceedings of the 2017 IEEE International Conference on Autonomic Computing*. 103–112.

Received April 2018; revised June 2018; accepted June 2018