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 on-demand model, scalability enhancement, and management costs 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 to resource shortages and application issues such as delays or saturated eventually. A paradigm named 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 it can avoid overloads due to changes in the workload and achieve better load balancing and energy saving effects. This paper proposes a taxonomy of 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 Literature**; • **Distributed Parallel and Cluster Computing**; • **System and Software**; • **Management of Cloud Computing Systems**;

Additional Key Words and Phrases: Cloud Computing, Adaptive Management, Brownout, Quality of Service, Optional Services

ACM Reference Format:
Minxian Xu and Rajkumar Buyya. 2018. Brownout Approach for Adaptive Management of Resources and Applications in Cloud Computing Systems: A Taxonomy and Future Directions. *ACM Comput. Surv.* 1, 1, Article 1 (April 2018), 27 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Cloud computing has been regarded as one of the most dominant technologies that promote the future economy [36]. Traditionally, the service providers used to establish their own data centers with a huge investment to maintain the applications and provide services to users. With cloud computing, resources can be leased by application providers and the applications can be deployed without any upfront costs. Nowadays, many applications are developed for the cloud computing systems, and Clouds also provide elastic resources for applications [61]. This feature attracts enterprises to migrate their local applications to Clouds [16].

In addition to the traditional requirements, the cloud applications are experiencing unpredictable workloads because of the dynamic amount of requests and users [21]. Thus, cloud computing systems are also required to be designed as robust to handle unexpected events: request bursts, which is commonly named as flash crowds that can increase the size of requests significantly. For example, the servers of Weibo, a Chinese web social network website owned by Sina, got a breakdown after a Chinese celebrity announced his new relationship, which resulted from the celebrity's fans flooding into the website [7].

Author's address: Minxian Xu and Rajkumar Buyya, Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Parkville, VIC, 3010, Australia.

0360-0300/2018/4-ART1 $15.00
https://doi.org/0000001.0000001

Similarly, in cloud data centers, unexpected hardware failures are also common issues. In 2016, the severe weather led to the outage of Amazon data centers in Sydney, knocking the services of many companies to be offline [4]. Moreover, performance inference resulted from co-located applications and workload consolidations may lead to unexpected performance degradations [70].

1.1 Need for Adaptive Management in Cloud Computing Systems

To handle the aforementioned phenomena, applications are needed to be carefully designed and deployed. For instance, techniques such as auto-scaling [42], data replication [48], workload consolidation [34] and dynamic load balancing [54] are applied to overcome unexpected events only if the available resources are adequate. However, these unexpected events are generally only lasting a relatively short duration, it is not economical to provision sufficient capacity as much as possible. Without sufficient resource provisioning, the applications can be saturated, and cause the users to experience longer response time or even no response at all. As a result, the service providers may lose customers and revenues. Therefore, we argue that the 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. QoS guarantee plays a crucial role in system performance for Clouds environment [40], and cloud computing systems are required to be designed to offer QoS guaranteed services [59] [60]. It is a challenging issue for Clouds to support various co-located applications with different QoS constraints, and provision resources efficiently to be adaptive to the users' dynamic behaviors. These aspects make it difficult to provision resources efficiently [58] [25]. It is reported that 53% mobile users abandon pages that have no response within three seconds. Therefore, Google firmly recommends a one-second web page loading time to improve user's experience [29]. 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 IT industry that huge energy is consumed by cloud data centers [45]. 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 [18]. The physical servers deployed in Clouds generate massive heat and require to be managed in an environment equipped with powerful air conditioners. One of the key reasons for huge energy usage is due to the inefficient utilization of resources [17]. Adaptive management is able to improve the resource usage so that the 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 the idle physical machines can be turned into the 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 the regular components of web applications, which allow the system to be scalable and resilience [57]. Numerous load balancing algorithms have been introduced by researchers, focusing on different optimization targets, ranging from balancing virtual machines load to physical machines , with specific optimizations by both heuristic and meta-heuristic algorithms [68]. The purposes of load balancing can be various, including geographical balancing [41], electricity costs reduction [56] and applications load balancing in cloud computing systems [47]. 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** [38]. In the field of brownout, the applications/services are extended to have two parts: mandatory and optional. The mandatory parts are desired to keep running all the time, such as the critical services in the systems including data-relevant services. The optional parts, on

the other hand, are not necessary to be active all the time and can be deactivated temporarily to ensure the system performance in the case of flash crowds.

A motivational example of brownout-enabled application is the E-Commerce system, where product descriptions are shown along with related products suggested to the 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 system for different optimization objectives, including system robustness improvement, overbooking, load balancing and energy efficiency. Therefore, we like to investigate them in depth as noted below:

- Brownout approach has shown a promising direction to manage 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 the progress in brownout approach with adaptive management of resources and applications for cloud computing systems. Consequently, we have surveyed the existing articles relevant to this topic and aimed to draw more attention and efforts to advance 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 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 et al.[37] conducted a comprehensive taxonomy and survey for energy efficient scheduling approaches in Clouds. Weerasiri et al. [64] introduced a survey and taxonomy for resource orchestration in Clouds while not focusing on adaptive resource management. Zhan et al. [69] investigated the cloud computing resource scheduling approaches and summarized their evolution. Mansouri et al. [44] presented a survey and taxonomy on resource management in Cloud environment, with the focus on management of storage resources. Singh et al. [60] proposed a systematic review of Quality of Service aware automatic resource scheduling approaches in Clouds scenario.

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

1.5 Article Structure

The rest of the paper is organized as follows: a brief background on cloud computing and adaptive management is introduced in Section 2. Then in Section 3, the discussion on evolution of brownout in cloud computing is presented. Section 4 depicts the methodology we applied to look for the 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 are presented in Section 6. Section 7 describes the brownout approach with 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 on cloud computing and adaptive management.

2.1 Cloud Computing

The appearance of cloud computing is regarded as a novel paradigm in information technology industry [19]. The aim of cloud computing is providing resources in the form of utility like water, gas, and electricity for daily use. Some attractive characteristics including 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 business runners, which gets rid of the complexity for service providers to do provisioning plan and allows companies to begin with the minimum resources as 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 these applications to be scalable and available for the demands of the users [46]. The cloud customers are allowed to dynamically lease and unlease the on-demand resources based on their requirements.

2.2 Adaptive Management

In the past years, a considerable amount of research in adaptive techniques to manage resources in the system has been conducted. The properties of adaptive techniques are achieving management in a self-adaptation manner, including protection, optimization, and recovery. These properties are often featured as self-* characteristics [49].

The feedback loop is the essential concept used to develop an adaptive system, which monitors the status and the environment of the system, and adapts as desired to obtain the required self-* characteristics. It is viewed as an important factor to enable adaptive management of resources and applications by taking advantage of feedback loops for applications [35]. In adaptive systems, one favorite representation of the feedback loop is the Monitor, Analyze, Plan, Execute and Knowledge loop, which is abbreviated as MAPE-K [15]. In the MAPE-K loop, there are several phases to be accomplished in the loop as below:

(1) Monitoring the system status and the environment situation by Monitor phase;

(2) Analyzing the collected data and determining whether the adaptation is required or not by Analyze phase;

(3) Planning the approach to adapt the system by Plan phase;

(4) Executing the plan by Execute phase,

where the knowledge pool is shared by these 4 phases and acts as integration role [22].

Table 1. The earliest work 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

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 are broadly searched in main-stream academic databases, including IEEEXplore, Springer, Elsevier, ACM Digital Library, ScienceDirect, Wiley Interscience and Google Scholar.

3.2 Search Method

Two phases of our search are involved. In the first phase, we use the keywords "Brownout" and "Cloud Computing" to search the title and abstract of research articles. Several results are found, however, the number of these articles are quite limited. We think that some articles may be motivated by the mechanism of brownout while they do not use the keywords in the title or abstracts. Thus, in the second phase, based on the initial brownout researches conducted in 2014, we plan to find other articles using brownout inspired by these work. We show the pioneering work of brownout approach in 2014 and their number of citations[1] in Table. 1. We investigated the articles that cite these four papers and found more articles that are using brownout.

3.3 Outcome

We have found 18 research articles focusing on 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% in symposiums. Moreover, one master thesis [23] and one PhD thesis [49] have been explored for this topic. The contents of these two theses are derived from the research articles we have found and reviewed, therefore, they are not included in the following taxonomy and review.

4 BROWNOUT APPROACH

Brownout is inspired by the blackout in emergency cases that to cope with electricity. Such as light bulbs emit fewer lights to save energy usage to handle emergency situations [63]. 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 concerning how to handle unpredictable workloads [38]. The idea behind the brownout paradigm is as follows: to be adaptive, optional parts might temporarily be

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

deactivated so that the essential functions of the system is ensured as well as avoiding saturated applications. Deactivating certain optional functions can contribute to increasing request acceptance rate by utilizing the optional resources for the core functions.

Therefore, the brownout paradigm is a method to make cloud computing system to be adaptive the changing workloads. Brownout also enables to improve 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, **dimmer** is a control knob that takes a value in [0, 1] to represent the probability to run the optional parts and manages the brownout activities in the system. [38].

4.1.2 Architecture Model

The brownout-based scheduling of resources and applications in cloud computing systems complies the conventional type of adaptive architectures. It is derived from MAPE-K [15] feedback loop control including phases, like the monitor, analyze, plan, execute, and knowledge illustrated in Fig. 1. Cloud computing system is the target system of MAPE-K loop and combined with MAPE-K loop through sensors and actuators. The responsibilities of each module in MAPE-K are as below:

- **Knowledge module:** It is a module that describes the whole system at abstract level by capturing the major system features and status. The captured information is applied to trigger the desirable adaptations;
- **Monitor module:** It is used to collect the data from sensors and monitor the system status continuously. The collected data is transferred to Analyze module for further analysis;
- **Analyze module:** It analyzes the data obtained from Monitor module and provides references for Plan module. Different analysis methods can be applied in this module;
- **Planning module:** It makes plans to change system states to be adaptive the fluctuations of workloads;
- **Execution module:** It implements the plans. Its main role is ensuring the specified system requirement. In addition, through actuators, it also traces the new changes and makes other plans based on the predefined rules in the knowledge pool.

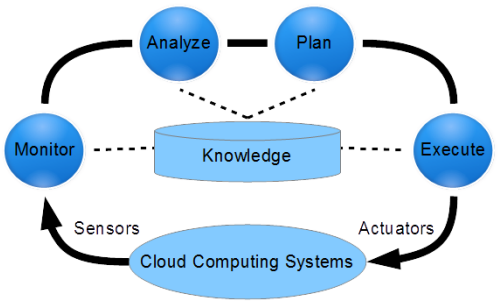

Fig. 1. Cloud computing systems with MAPE-K adaption loop

4.1.3 Brownout Management

The challenges for brownout management includes:

- When the optional services should be deactivated?
  It is relevant to system status. The optional services would be deactivated when some system indicators show the system is not running as expected. Such as, the system is becoming overloaded when requests are bursting.

- How to deactivate optional services?
  Services can be processed in different ways. For example, for the stateless services, they 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?
  It 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 objectives and metrics of brownout approaches in cloud computing system have been investigated and proposed throughout the years. As shown in Fig. 2, we aim to show the evolution and development of brownout approaches in recent years.

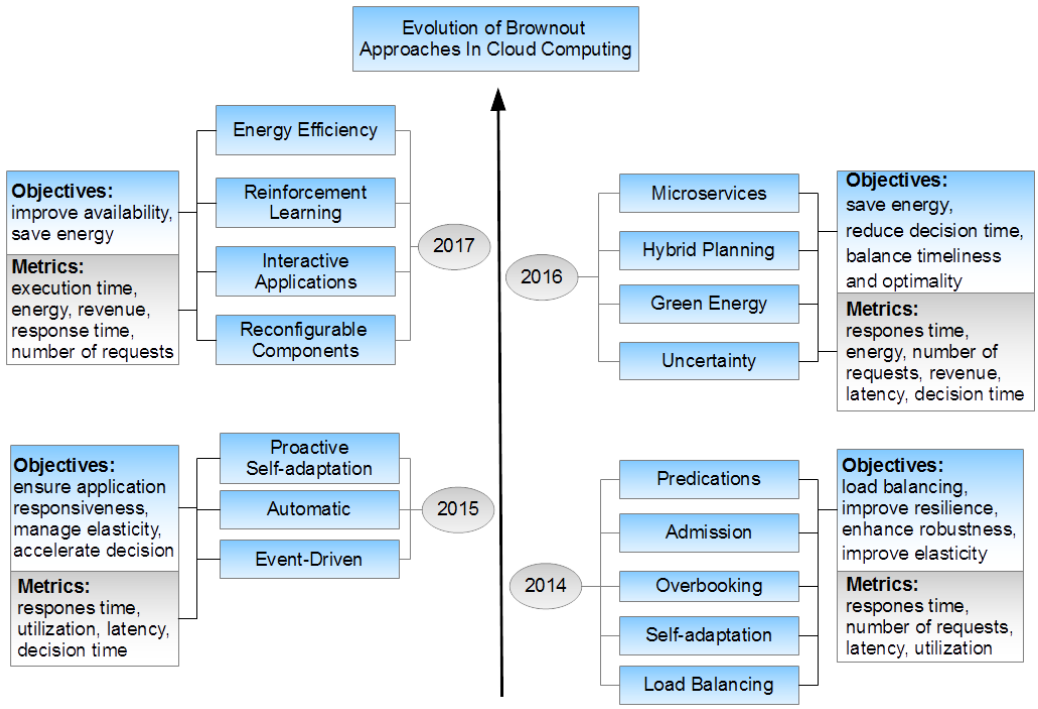

Fig. 2. Evolution of brownout approach in cloud computing systems

In 2014, the application of brownout in cloud computing systems was proposed by Klein et al. [38], 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 have been presented: RUBiS [2] and RuBBoS [1], which have been widely used in the follow-up research. This work showed the effectiveness of brownout approach, while the experiments were only conducted on a single machine. Durango et al. [27] 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 general.

To find failures earlier and avoid the limitation of periodically monitoring, based on event-driven, Klein et al. [39] 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. [43] 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, Nokolov et al. [52] presented a platform to manage resource adaptively for elastic Cloud based on SLOs, which can overcome short-time request bursts. The proposed platform can adapt application execution to corresponding service level agreement (SLA). However, faults handling is not considered in this work. Tomas et al. [63] combined overbooking and brownout together to improve resource utilization without application degradation. Like [27], this approach is also based on admission control that gradually adapts application according to requests.

In 2015, automatic algorithms based on brownout had drawn more attention. Desmeurs et al. [24] presented an event-driven brownout technique to investigate the trade-offs 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, while the results were not evaluated under real trace. Dupont et al. [26] proposed another automatic approach to manage cloud elasticity in both infrastructure elasticity and software elasticity. 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. [50] presented a proactive approach for latency-aware scheduling under uncertainty to accelerate the decision time. The motivation is applying formal model to solve the nondeterministic choices of adaptation tactics (disabling different optional contents). The limitations of this work are: 1) this work does not support concurrent tactics, and 2) the uncertainty of environment predictions is not considered.

In 2016, several articles were devoted to improving the adaption of decision time. Pandey et al. [53] proposed a hybrid selection methodology that investigates multiple optimization objectives to balance the trade-offs between different metrics, such as decision time and optimized results. Markov decision process is applied to find the best choice among candidates, which represent the combination of different disabled optional contents. To overcome the limitations in [50], Moreno et al. [51] presented an approach aiming to eliminate the run-time costs when constructing 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 paid their attention to energy saving for Clouds. Hasan et al. [33] introduced a green energy-aware scheduling approach for interactive cloud application that allows being dynamically adapted. Each application has three different modes, and each mode has a different percentage of optional contents. Xu et al. [67] applied brownout to reduce power consumption by dynamically and selectively disabling the optional components of applications. The trade-offs between energy and discount were investigated, and several heuristic components selection policies were also proposed.

In 2017, as an extension work of [33], Hasan et al. [32] investigated multiple metrics other than energy, including both user experience and performance for the interactive cloud application. An adaptive application management approach based on dynamically switching application modes was proposed to improve the trade-offs among multiple optimization objectives. In order to improve the elasticity of infrastructure by adding or removing resources, Hasan et al. [31] proposed a platform that applies 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, the phases and review of brownout approach for adaptive management of resources and applications in cloud computing systems are presented. According to our surveyed articles, we have classified adaptive management of the resources and applications with brownout into five phases: application design, workload scheduling, monitoring, brownout controller design, and metrics, as demonstrated in Fig. 3. These phases can be mapped to the MAPE-K modules in as shown in Fig. 1. Application design and workload scheduling correspond to the Knowledge module to describe 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. Now, we explain the details of each phase.

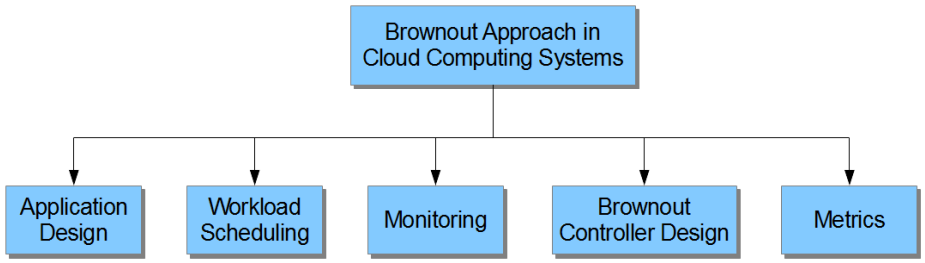

Fig. 3. Phases of applying brownout approach in cloud computing systems

5.1 Application Design

Applications are running in cloud computing systems and providing services for users. To enhance the system performance in Clouds, applications are required to be designed by referring to system configuration and users requirements. In Fig. 4, the categories we used for the application design are: 1) application type, 2) application domain, 3) optional parts, and 4) application deployment.

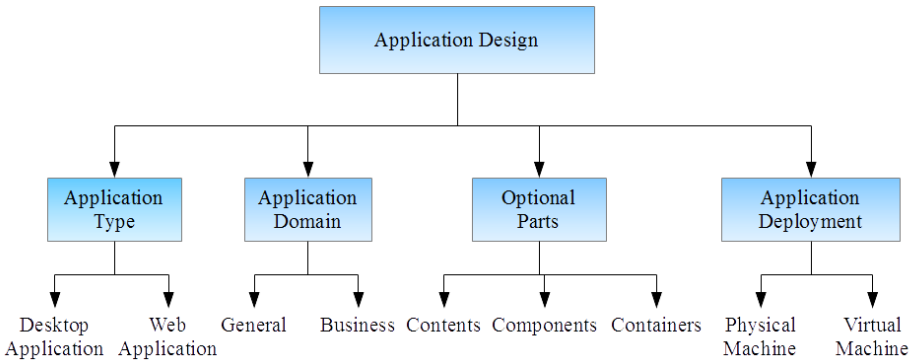

Fig. 4. Taxonomy based on application design

5.1.1 Application Type. The application can be running locally or in the remote manner enabled by brownout. Brownout-enabled applications can be classified into two types according to application type: (1) desktop application, and (2) web application. The desktop application represents the brownout-enabled applications runs locally on machines, for example, the texts processing application [14]. The web application is implemented in the client-server model to provide web services that the users interact through the Internet. The typical brownout-enabled web application is online shopping system, which has been applied in many existing brownout-related articles [27][38][43].

5.1.2 Application Domain. Applications are implemented to provide functionalities for users. The developer is aiming to develop applications in a more efficient manner, while the complexity of applications is growing. Adaptive application management in cloud computing systems provides an approach to manage the complex applications. To manage these applications, the domain of applications should be identified, as applications in different domains have different management requirements. For the applications in the general domain, applications are providing functions for general purposes [67], such as scientific calculation applications. While for the applications in the business domain, which focuses on maximizing the profits of service providers, they are more sensitive to some specific performance metrics. For example, in the online shopping system that belongs to the business domain, the response time as QoS requirement is one of the most critical metrics [32], since long response time 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 (1) contents, (2) components and (3) containers. Optional web contents on servers are to be showed selectively to users to save resource usage [38]. Components-based applications deactivate optional components to manage resource utilization [14]. In containerized clouds, each service is implemented as containers, and the optional containers can be activated/deactivated based on system status. [67]

5.1.4 Application Deployment. In cloud computing systems, applications can be deployed on physical machines (PMs) or virtual machines (VMs). Deploying applications directly on PMs enables applications to have almost the same performance as native systems, such as 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 with brownout together to optimize resource utilization [67].

5.2 Workload Scheduling

Workload scheduling aims at scheduling and allocating appropriate resources to suitable workloads according to SLA defined by end-users and service providers so that the workloads can be executed efficiently and the applications utilize resources efficiently. In Fig. 5, the categories we used for workload scheduling are: 1) workload type, 2) resource type, 3) dynamicity, 4) workload trace and 5) experiment platform.

5.2.1 Workload Type. The workload type represents the resource requirement of workloads. In current brownout-relevant articles, most works are focusing on scheduling the CPU-intensive workloads [27][24][26], as computation resources are regarded as the main resource allocated to workloads. Some articles also consider network-intensive workloads to reduce the network latency [52].

5.2.2 Resource Type. For homogeneous resource type, the resources offered by service providers are limited to a single type. This configuration simplifies the workload scheduling and overlooks the

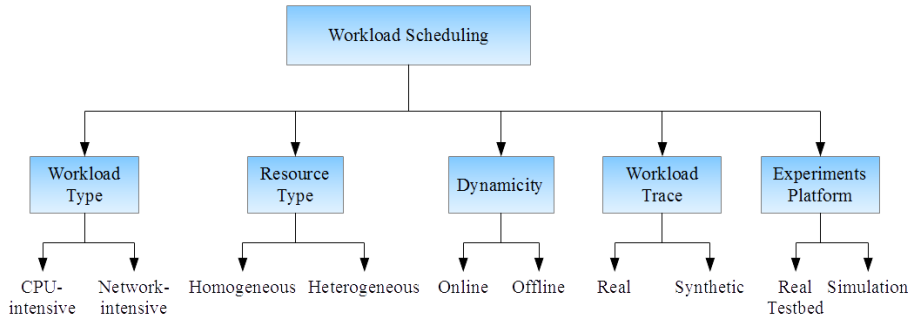

Fig. 5. Taxonomy based on workload scheduling

various characteristic of workload. The homogeneous configuration is mostly used in small-scale tests where resource diversity is limited or for the initial research of novel approach [27][38]. Cloud computing systems have the nature of heterogeneity, to take advantage of this feature, mature service providers are offering heterogeneous resources for workload scheduling. For example, Amazon EC2 has provided more than 50 types of VMs, and these VMs are categorized as various classifications for different workload types, such as general purpose, computation intensive purpose or memory intensive purpose [28].

5.2.3 Dynamicity. The dynamicity of workload scheduling represents the time when the workload information is obtained. The dynamicity of workload scheduling is noted as online if the information of workloads is only available when the workloads are coming into systems. If all the information of workloads are known in advance and workloads can be rescheduled based on system resource, this scheduling process is identified as offline. One example of the offline scheduling is the batch job [33], in which the deadlines of jobs are known and jobs can be executed with delay based on resource availability. Compared with online workload scheduling, offline workload scheduling is prone to achieve 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 on, workloads derived from real traces are also applied. Presently, the popular real traces are from FIFA [3], Wikipedia [65], and Planet lab trace [55].

5.2.5 Experiments Platform. Both real testbed and simulations have been conducted to test the performance of brownout approaches. Experiments under real testbed are more persuasive. Grid'5000 platform [30] has been adopted in several articles, which provides APIs to collect PM utilization and energy consumption. However, some uncontrolled factors exist in the real testbed, such as network traffics and unpredictable loads. Therefore, simulation tools provide more feasible options. Besides, with simulations, it is easier to conduct experiments with heterogeneous resources as well as large-scale size. The cloud simulation toolkit, CloudSim [20], has been used for simulating brownout-enabled workload scheduling.

5.3 Monitoring

The objective of monitoring is achieving performance optimization by monitoring the resource usage in cloud computing systems. Therefore, a monitoring component is required to collect system and analyze the resource utilization information. After analysis, decisions to change the system

status and resource usage are made to ensure system to satisfy the specified SLA. As shown in Fig. 6, we categorize the components of monitoring as 1). resource usage, 2) services, 3) status, and 4) execution.

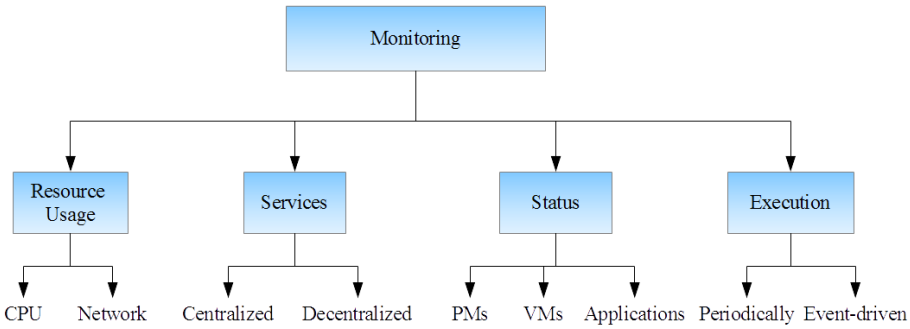

Fig. 6. Taxonomy based on monitoring

5.3.1 Resource Usage. The monitor in Clouds environment is used to monitor the resource usage, including CPU, memory and network usage via the monitoring tools. As mentioned in section 5.2.1, the current brownout related workload scheduling are focusing on handling CPU-intensive and network-intensive workloads [39][52]. So, the monitors in brownout-enabled systems are mainly monitoring the 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, and service providers aim to execute the workloads with minimum resource usage.

5.3.2 Services. Service monitoring gathers the information about resource statuses to check whether the workload is 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 manner, a central repository is applied to store the collected data, which is not scalable when the number of monitored targets is increased [67]. For the distributed service monitoring, the monitored data is stored distributedly to achieve better fault tolerance and load balancing [14].

5.3.3 Status. To be more specific, monitoring resource utilization is regarded as monitoring the status of different levels, including PMs [50], VMs [26], and applications [63]. Based on various optimization goals, data are collected from different levels. For example, to reduce the energy of cloud computing systems, power consumption of PMs should be collected. To improve the response time of requests, it is required to obtain the resource usage of applications.

5.3.4 Execution. To avoid unexpected failures and execute workloads promptly, the execution monitoring has two types: periodically and event-driven. In the periodical manner, the monitor periodically checks the resource usage and makes decisions on execution process for the next time period [27]. In an event-driven manner, changes in execution process are triggered once a specific event is detected, such as resource usage is above the predefined overloaded threshold [39]. The motivation of event-driven 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 brownout controller. The brownout controller also has a control knob called dimmer, which

represents the probability of the optional parts to be deactivated. Therefore, we can notice that the brownout controller design is the most important part for brownout approach. In Fig. 7, we have categorized the classification of brownout controller/dimmer design as 1) parameters, 2) controller algorithm, 3) controller number and 4) adaptivity.

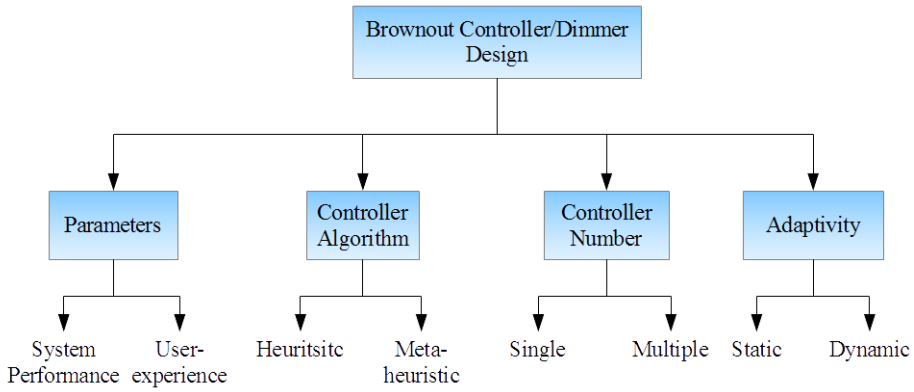

Fig. 7. Taxonomy based on brownout controller design

5.4.1 Parameters. 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. If system performance is addressed, the brownout controller is configured with system parameters, such as resource utilization [27]. If the brownout controller aims to optimize user-experience, parameters like response time can be designed into brownout controller [38].

5.4.2 Controller Algorithm. Similar to the resource scheduling problem, finding the optimal solutions of suitable optional parts to be deactivated/activated by controller algorithms is computationally expensive. Thus, finding the approximate solutions is an alternative to the most of proposed approaches. We have classified the surveyed controller algorithms as heuristic and meta-heuristic. The heuristic ones comply with the predefined constraints and try to find an acceptable solution for particular problem [62]. Usually, the constraints are varied for different problems, and the solutions can be obtained within a limited time. One type of heuristic algorithms is greedy algorithm and has been adopted in [24][26][50]. As for meta-heuristic algorithms, they are generally applied to general purpose problems [62], and have standard procedures for problem construction and solving. One example of the meta-heuristic algorithm is the approach based on Markov decision process that has been applied in [66][51].

5.4.3 Controller Number. The brownout controller may have single or multiple controllers to manage the optional parts in cloud computing systems. In [63], 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 [63] can be varied. For overall management, a single controller is applied. For example, in [31], the dimmer value is calculated as the severity of overloads in the whole data center.

5.4.4 Adaptivity. The adaptivity represents whether the brownout controller is adaptive to the change of workloads. It is categorized as static and dynamic. In our surveyed approaches, most of them are dynamic [27][43][63], which can be dynamically adapted to the change of system. Only limited approaches are static, which apply static parameters. The static brownout controllers are easy to violate the specific SLAs [33][31].

5.5 Metrics

Different metrics are considered in cloud computing systems to evaluate the performance of different brownout-based approaches. As shown in Fig. 8, from our surveyed literature, we have identified 9 metrics: response time, execution time, utilization, availability, decision time, latency, request number, energy and revenue.

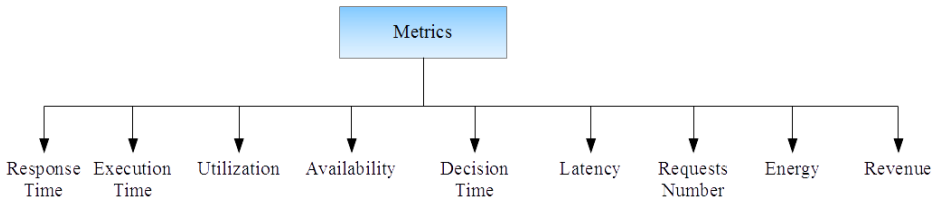

Fig. 8. Taxonomy based on metrics

Response Time [27][38][43] is the total time it costs 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* [14][71] is the time required to complete the workloads execution. Minimizing the execution time can improve system QoS. *Utilization* [52][63] is the actual resource percentage used to run workloads to the total resource provided by service provider. Available utilization should be improved to maximize the resource usage. *Availability* [14][71] is the capability of the cloud computing system to guarantee the services are available with expected performance as well as handle fatal situations. This metric should be ensured in cloud computing systems, e.g. during 99.9% time, the services are running with expected performance. *Decision Time* [50][51] is the time that scheduling algorithm takes to find the optional parts to be deactivated/activated, which is associated with algorithm design. To find the solutions faster, in online scheduling scenario, the decision time of algorithm should be accelerated. *Latency* [38][50] is the time delay that spends on message communication across the network, which is relevant to hardware resources and utilization. In business applications, latency is an crucial metric to indicate the system performance. *Requests Number* [26][33] is the number of requests received by the system. The scheduling algorithm has better performance if more requests can be served with the same amount of resources. *Energy* [67][66] is the total power used for the cloud computing system to provide the services to users. The system should reduce energy consumption while ensuring QoS. *Revenue* [31][33] is the profit obtained from users when service providers are offering services. The service provider aims to maximize their revenue as well as lowering costs.

6 REVIEW OF BROWNOUT APPROACHES

In this section, a review of brownout-based approaches for adaptive management of resources and applications in cloud computing system is conducted. 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 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) [27] technique extends brownout approach for services with multiple replicas, which are the copies of applications providing same

Table 2. Summary of brownout approaches

Approach	Year	Author	Description	Orgranization
COBLB [27]	2014	Durango et al.	Load Balancing with Brownout and Control Theory	Umea University, Sweden
SAB [38]	2014	Klein et al.	Robust Cloud Applications with Brownout	Umea University, Sweden
EPBH [39]	2014	Klein et al.	Resilience and Load Balancing with Brownout	Umea University, Sweden
FPF [43]	2014	Maggio et al.	Brownout Prediction	Lund University, Sweden
CLOUDFARM [52]	2014	Nikolov et al.	Adaptive Resource Management	University of Ulm, Germany
BOB [63]	2014	Tomas et al.	Overbooking with Brownout	Umea University, Sweden
EDB [24]	2015	Desmeurs et al.	Event-Driven Application with Brownout	Umea University, Sweden
CAA [26]	2015	Dupont et al.	Cloud Elasticity with Brownout	INRIA, France
PLA [50]	2015	Moreno et al.	Proactive Self-Adaptation for Uncertainty Environment	Carnegie Mellon University, USA
HYB-Q [33]	2016	Hasan et al.	Green Energy for Cloud Application	INRIA, France
PLA-SDP [51]	2016	Moreno et al.	Decision-Making for Proactive Self-Adaptation	Carnegie Mellon University, USA
HYBP [53]	2016	Pandey et al.	Hybrid Planning in Self-Adaptation	Carnegie Mellon University, USA
LUFCS [67]	2016	Xu et al.	Energy Efficiency in Clouds with Brownout	University of Melbourne, Australia
MODULAR [14]	2017	Alvares et al.	Modular Autonomic Manager in Software Components	INRIA, France
SaaScaler [32]	2017	Hasan et al.	Power and Performance Scaler for Applications	INRIA, France
GPaaScaler [31]	2017	Hasan et al.	Green Energy Aware Scaler for Interactive Applications	INRIA, France
RLBF [71]	2017	Zhao et al.	Reinforcement Learning for Software Adaptation	Peking University, China
BMDP [66]	2017	Xu et al.	Energy Efficient Clouds with Markov Decision Process and Brownout	University of Melbourne, Australia

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

Self-Adaptive Brownout (SAB) [38] approach adds a dynamic parameter that affects user experience and the amount of resource 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 control theory approach to adaptively determine when to activate/deactivate optional features in the applications. With control theory, the behaviors of applications can be predicted, and some system constraints can be guaranteed. In this approach,

Table 3. Taxonomy based on application design

Approach	Application Type	Application Domain	Optional Parts	Application Deployment
COBLB [27]	Web application	Business	Contents	Virtual Machine
SAB [38]	Web application	Business	Contents	Virtual Machine
EPBH [39]	Web application	Business	Contents	Virtual Machine
FPF [43]	Web application	Business	Components	Physical Machine
CLOUDFARM [52]	Web application	Business	Components	Physical Machine
BOB [63]	Web application	Business	Contents	Virtual Machine
EDB [24]	Web application	Business	Contents	Virtual Machine
CAA [26]	Web application	Business	Contents	Virtual Machine
PLA [50]	Web application	Business	Contents	Virtual Machine
HYB-Q [33]	Web application	Business	Contents	Virtual Machine
PLA-SDP [51]	Web application	Business	Contents	Virtual Machine
HYBP [53]	Web Application	Business	Contents	Virtual Machine
LUFCS [67]	Web Application	General	Components	Virtual Machine
MODULAR [14]	Desktop Application	General	Components	Physical Machine
SaaScaler [32]	Web application	Business	Contents	Virtual Machine
GPaaScaler [31]	Web application	Business	Contents	Virtual Machine
RLBF [71]	Web application	Business	Components	Physical Machine
BMDP [66]	Web Application	General	Containers	Virtual Machine

Table 4. Taxonomy based on workload scheduling

Approach	Workload Type	Resource Type	Dynamicity	Trace	Experiments Platform
COBLB [27]	CPU-intensive	Homogeneous	Online	Synthetic	Simulation
SAB [38]	CPU-intensive	Homogeneous	Online	Synthetic	Real testbed
EPBH [39]	CPU-intensive	Homogeneous	Online	Synthetic	Real testbed
FPF [43]	CPU-intensive	Homogeneous	Online	Synthetic	Simulation
CLOUDFARM [52]	Network-intensive	Heterogeneous	Online	Synthetic	Real testbed
BOB [63]	CPU-intensive	Homogeneous	Online/Offline	Real	Real testbed
EDB [24]	CPU-intensive	Homogeneous	Online	Synthetic	Real testbed
CAA [26]	CPU-intensive	Homogeneous	Offline	Synthetic	Real testbed
PLA [50]	CPU-intensive	Homogeneous	Online	Real	Real testbed
HYB-Q [33]	CPU-intensive	Homogeneous	Online	Real	Real testbed
PLA-SDP [51]	CPU-intensive	Homogeneous	Offline	Real	Real testbed
HYBP [53]	CPU-intensive	Heterogeneous	Online	Real	Simulation
LUFCS [67]	CPU-intensive	Heterogeneous	Online	Real	Simulation
MODULAR [14]	CPU-intensive	Homogeneous	Online	Synthetic	Real testbed
SaaScaler [32]	CPU-intensive	Homogeneous	Online	Real	Real testbed
GPaaScaler [31]	CPU-intensive	Homogeneous	Online	Real	Real testbed
RLBF [71]	CPU-intensive	Homogeneous	Online/Offline	Synthetic	Real testbed
BMDP [66]	CPU-intensive	Heterogeneous	Online	Real	Simulation

the maximum latency is controlled so that the latency is reduced. Also, SAB can serve more users with fewer resources.

Equality Principle-Based Heuristic (EPBH) [39] is focusing on enhancing the resilience of cloud services when system faces resource shortage. Similar to COBLB [27], this approach is also applied to application replicas. This event-driven approach is based on heuristic and requires all the replicas to have a control parameter (dimmer) value. According to the parameter value, EPBH decides which replicas to receive more loads. EPBH has been proven to improve resilience when resource shortage is triggered by failures.

Table 5. Taxonomy based on monitoring

Approach	Resource Usage	Services	Status	Execution
COBLB [27]	CPU	Centralized	Applications	Periodically
SAB [38]	CPU	Centralized	Applications	Periodically
EPBH [39]	CPU	Centralized	Applications	Event-Driven
FPF [43]	CPU	Centralized	Applications	Periodically
CLOUDFARM [52]	Network	Centralized	PMs, Applications	Periodically
BOB [63]	CPU	Centralized	Applications	Periodically
EDB [24]	CPU	Centralized	Application	Event-Driven
CAA [26]	CPU	Centralized	VMs	Periodically
PLA [50]	CPU	Centralized	PMs	Periodically
HYB-Q [33]	CPU	Centralized	Applications	Periodically/Event-Driven
PLA-SDP [51]	CPU	Centralized	PMs	Periodically
HYBP [53]	CPU	Centralized	PMs	Periodically
LUFCS [67]	CPU	Centralized	PMs	Periodically
MODULAR [14]	CPU	Decentralized	PMs	Periodically/Event-Driven
SaaScaler [32]	CPU	Centralized	VMs, Applications	Periodically/Event-Driven
GPaaScaler [31]	CPU	Centralized	VMs, Applications	Periodically
RLBF [71]	CPU	Centralized	PMs	Periodically
BMDP [66]	CPU	Centralized	PMs	Periodically

Table 6. Taxonomy based on brownout controller/dimmer design

Approach	Parameters	Controller Algorithm	Controller Number	Adaptivity
COBLB [27]	System Performance	Heuristic	Multiple	Dynamic
SAB [38]	User-experience	Heuristic	Multiple	Dynamic
EPBH [39]	System Performance	Heuristic	Single	Dynamic
FPF [43]	User-experience	Heuristic	Single	Dynamic
CLOUDFARM [52]	System Performance	Heuristic	Multiple	Dynamic
BOB [63]	System Performance	Heuristic	Multiple	Static/Dynamic
EDB [24]	System Performance	Heuristic	Multiple	Dynamic
CAA [26]	System Performance	Heuristic	Multiple	Dynamic
PLA [50]	System Performance	Meta-heuristic	Single	Dynamic
HYB-Q [33]	User-experience	Heuristic	Single	Static
PLA-SDP [51]	System Performance	Meta-heuristic	Single	Static
HYBP [53]	System Performance	Meta-heuristic	Single	Dynamic
LUFCS [67]	System Performance	Heuristic	Multiple	Dynamic
MODULAR [14]	System Performance	Heuristic	Multiple	Dynamic
SaaScaler [32]	User-experience	Heuristic	Single	Static
GPaaScaler [31]	User-experience	Heuristic	Single	Static
RLBF [71]	System Performance	Meta-heuristic	Multiple	Static/Dynamic
BMDP [66]	System Performance	Meta-heuristic	Multiple	Dynamic

The objective of Feedforward Plus Feedback (FPF) [43] approach is keeping 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 the requests bursts for cloud applications and reacts to the changes of resource amount allocated to cloud applications. The idea of FPF is obtaining the number of currently queued requests from the applications and predicting the latency experienced by future requests. The results demonstrate that although FPF spends efforts on obtaining more information, the overloads are mitigated.

Table 7. Taxonomy based on metrics

Approach	Response Time	Execution Time	Utilization	Availability	Decision Time	Latency	Requests Number	Energy	Revenue
COBLB [27]	✓						✓		
SAB [38]						✓	✓		
EPBH [39]	✓						✓		
FPF [43]	✓						✓		
CLOUDFARM [52]			✓						
BOB [63]	✓		✓						
EDB [24]	✓		✓						
CAA [26]	✓						✓		
PLA [50]					✓	✓			
HYB-Q [33]	✓						✓	✓	✓
PLA-SDP [51]					✓	✓			
HYBP [53]	✓				✓				
LUFCS [67]								✓	✓
MODULAR [14]		✓		✓					
SaaScaler [32]	✓						✓	✓	✓
GPaaScaler [31]	✓							✓	✓
RLBF [71]		✓		✓					
BMDP [66]								✓	✓

CLOUDFARM [52] 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 virtual framework for all the applications, which benefits from its lightweight and dynamic degradation from full mode to degraded mode.

Brownout OverBooking (BOB) [63] technique is designed to ensure graceful degradation when loads spike and thus avoiding overloads in resource overbooking scenario. The motivation of BOB is combining overbooking and brownout together, where 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) [23] technique at application level has been proposed. In this approach, the application is allowed to run optional codes that are not necessary for key functionalities but for extra user experience. EDB combines machine learning techniques and control theory together 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.

Cloud Automatic Approach (CAA) [26] aims at managing cloud elasticity in a cross-layered manner. The motivation is combining the infrastructure elasticity and software elasticity to overcome the conceptual limitations of IaaS and make software at SaaS involved in the elasticity process. For the software at SaaS, they are running at different levels and can be degraded to lower levels when the resource is limited. And for the resources at 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 manner is working in a coordinated way.

Proactive Latency-Aware (PLA) [50] 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 using a formal model to find the adaptation decisions

Table 8. Comparison of brownout approaches

Approach	Objectives	Advantages	Limitations
COBLB [27]	Load balancing for Cloud applications	Resilience is improved	Application model is not general
SAB [38]	Enhance robustness of Cloud applications	Support more users with fewer resources	Only tested on a single machine
EPBH [39]	Improve resilience	Response time is reduced	Parameters are chosen empirically
FPF [43]	Mitigate overloads	Prediction is applied	Only validated with simulations
CLOUDFARM [52]	Improve elasticity	SLA is ensured when there are bursts	Faults handling is not considered
BOB [63]	Resource overbooking	Improve resource utilization without application degradation	Scalability is not discusses
EDB [24]	Balance utilization and response time	Utilization is improved and response time is ensured	Not evaluated with real workloads
CAA [26]	Manage elasticty	Elasticity is improved at both infrastructure and software levels	Limited number of tactics
PLA [50]	Accelerate decision time	Latency is reduced	Concurrent tactics are not supported
HYB-Q [33]	Green energy provisioning for interactive cloud application	Brown energy is saved	Non-interactive cloud applications are not investigated
PLA-SDP [51]	Accelerate decision time	Decision time is reduced	Not available for online requests
HYBP [53]	Balance decision time and optimality	Decision time is reduced	Not validated in real testbed
LUFCS [67]	Energy efficient management of application at component level	Energy consumption is reduced	Not all combinations of deactivated components are accessed
MODULAR [14]	Manage modular components	System availability is improved	The availability for other applications is not discussed
SaaScaler [32]	Investigate trade-offs between energy and performance	Energy is reduced and QoS is improved	Resources addition or removal are not flexible
GPaaScaler [31]	Reduce energy consumption	Energy is saved	Not available for real-time requests
RLBF [71]	Enhance flexibility of approach	Better adaptation effectiveness is achieved	Convergence time is not analyzed
BMDP [66]	Improve trade-off between energy and discount	Trade-off between energy and discount is improved	Not validated in real testbed

which are underspecified through nondeterminism, and then resolve the nondeterministic choices to maximize optimization goals. PLA takes the uncertainty of 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) [33] technique is aiming 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 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 arises above the predefined threshold, the user experience is downgraded to 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 [50], Proactive Latency-Aware with Markov Decision Process (PLA-SDP) [51] takes advantage of the probabilistic property of Markov decision process to adapt application functionalities. To eliminate the run-time overhead of constructing MDP, stochastic dynamic programming is applied to solve MDP. The decision time is vitally reduced while same results are obtained.

The objective of Hybrid Planning (HYBP) [53] is finding the trade-offs between timeliness and optimality of solutions to adapt application modes. HYBP combines two approaches together to achieve the best balance between the conflicts. One is deterministic planning and the other is Markov decision process planning, which can be fitted in different cases. In the case of fast response is required, the deterministic planning is applied to give a fast response. When the time is adequate, the MDP planning is applied to generate an optimal solution. Simulated experiments have shown that HYBP can improve system performance by balancing the aforementioned trade-offs.

The "Lowest Utilization First Component Selection" policy (LUFCS) [67] is a heuristic technique to save data center power usage via dynamically deactivating application components. The components with lower utilization are selected with higher priority until the sum of these components' utilization equals to the expected utilization reduction. This technique is combined with VM consolidation to decrease the active number of hosts. The discount is also considered to be given to users if some services are not provided. Therefore, there is a trade-off between saved energy and revenue, which is investigated by LUFCS.

MODULAR [14] approach leverages modularity feature of the component-based system to strengthen the domain-specific language application. Domain-specific language has high-level constructs to describe the configurations and executed policies of the target system. With brownout mechanism, the components can be transferred from nominal configuration to degraded one, which incurs different loads for the system. Thus, the system is entitled the capability of dynamic reconfiguration to be self-adaptive.

Compared with HYB-Q [33] that focuses on green energy usage, SaaScaler [32] approach analyzes the trade-off between power and performance by considering green energy usage for the interactive cloud applications. SaaScaler can satisfy different metrics. Considering these metrics, three levels of user experience are defined. Capacity requirement is 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 [31] considers adaptation both at infrastructure and application levels by using green energy source. At the infrastructure level, resources are added or removed according to resource demand of applications. 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 together. Lower costs and less usage of brown energy are achieved.

The objective of Reinforcement Learning-Based Framework (RLBF) [71] approach is overcoming the limitations of rule-based adaptation that 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 be evolved with real-time information for online adaptation of selecting optional services to run. The efficiency and effectiveness of adaptation process are improved by this approach.

Aiming at improving the trade-offs mentioned in [67], another brownout-based approach combining with Markov Decision Process, called BMDP [66] aims to select the better combinations of deactivated application components was proposed. With MDP, solutions space is increased and more possible solutions are found. To reduce the solutions space and avoid the curse of dimension, key states that can reduce energy consumption are identified. Taking advantage of MDP, a better trade-off between energy and revenue is fulfilled.

To summarize the merits and demerits of reviewed brownout approaches, a comparison of the advantages and limitations of each brownout approach is presented in Table 8. For example, COBLB [27] approach achieves better resilience while the adopted application model is not general.

7 A PERSPECTIVE MODEL

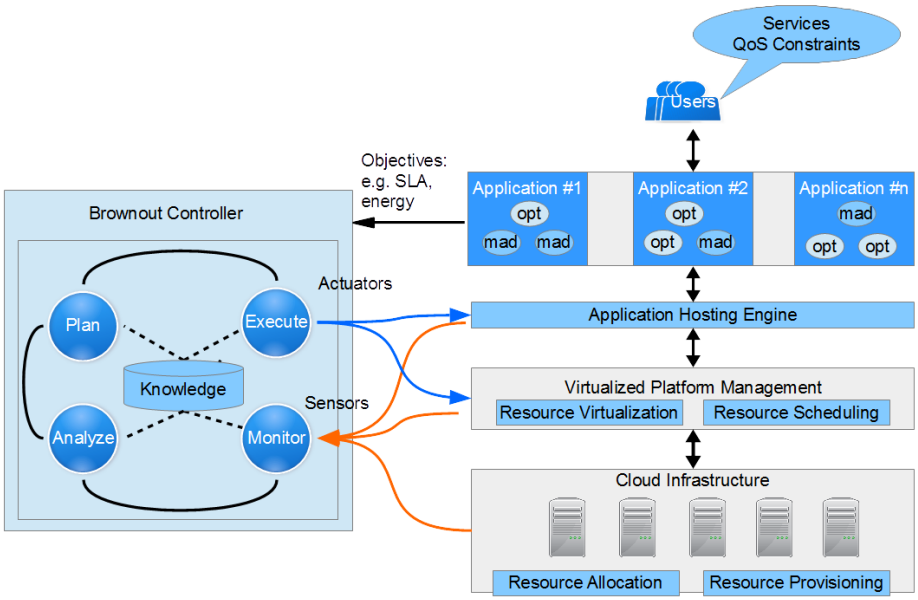

Fig. 9. Perspective model of applying brownout in cloud computing systems

Brownout approach has shown its ability to improve applications and resources management to handle changes in workloads. However, the trade-offs between workload handling and performance degradation are necessary for consideration. Several optimization objectives are usually considered together to investigate the trade-offs. The primary research problems in the context are as below:

- (1). How to enable brownout approach in cloud computing system?
- (2). How to enable brownout to manage resources and applications adaptively?
- (3). How to balance the trade-offs between different metrics when applying brownout?

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

From 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 service providers' perspective, these workloads generated by users are executed by applications.

Applications are provided by service providers to offer services for users, and these applications are managed by the application hosting engine, e.g. Docker [6] and Apache Tomcat [9]. Applications can be deployed on either virtualized platform (virtual resources) or cloud infrastructure (physical resources). The host application engine can be container-based management platforms, e.g. Docker Swarm [12], Kubernetes [11] or Mesos [8], which provide management of container-based applications. The virtualized platform manages the virtualized resources, for example, the Virtual Machines managed by VMware [13]. As for the resource allocation and provisioning in cloud infrastructure, they can be managed by infrastructure management platform, such as OpenStack [10].

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 brownout controller. For instance, in Docker Swarm, the services can be deployed via a configuration file [5] and the file can set the service with the feature "optional".

To resolve research problem (2), a brownout controller is required based on MAPE-K architecture model and fits into the feedback loop of MAPE-K model as introduced in Fig. 1. As described in Section 4.1.2, it has modules including Monitor, Analyze, Plan and Execute to fulfill adaptation with cloud computing system. Sensors and Actuators are used to interact with cloud computing system. Sensors collect the information from different levels in cloud computing systems, including application hosting engine, virtualized platform, and cloud infrastructure. The Sensors can be the devices attached to hardware, e.g. power meter. The collected information is provided to Monitor module.

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

To handle the 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 trade-offs (e.g. trade-offs between energy and SLA). The rules in Knowledge pool, such as SLA rules, can be updated according to brownout-based policies.

8 FUTURE RESEARCH DIRECTIONS

Although the significant progress of applying brownout to cloud computing systems has been achieved and 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 below:

- For resource management with 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, brownout approach can be applied to manage data-intensive applications.
- Brownout has been applied to optimization goals including load balancing and energy efficiency. Besides these optimization goals, other goals such as more complicated cost-aware resource scheduling scenario, and more QoS metrics can be applied with brownout.

- Presently, there is a lack of standard benchmark for performance evaluation of brownout-based approaches. It is required to have a benchmark to test the performance of new algorithm and compare it with other approaches having the same optimization objective.
- Scalability of brownout controller can be improved by using distributed controllers. The centralized controller can be bottleneck of the whole system. Therefore, distributed controllers design can be investigated.
- 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, more prototype systems based on the target systems other than RUBis should be introduced to show broader availability of brownout.
- Rather than considering a single data center, brownout approach for cloud computing systems can be extended to multiple data centers to deal with multi-cloud challenges, such as geographical load balancing. The knowledge pool in the perspective model can be used for tracking availability of the system when the system is scaled.
- The decision time and execution time of brownout-based algorithm to find the optional parts to be deactivated 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 are not discussed in our surveyed works, which should be investigated.

We summarize the future research directions identified as follows:

- **Integration with container:** Containers have provided a flexible ability to provide services with isolated function, and their quick start and stop operations enable brownout approach to work efficiently. Therefore, integrating brownout and containers is a promising direction to improve scheduling performance.
- **Exploring more scenarios:** Brownout has been used for load balancing, energy efficient, latency-aware and cost-aware 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:** It has been evaluated that brownout can be combined with VM consolidation to achieve better resource management effects [67][66] . 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 focused 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:** Fig. 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 brownout area. To implement the perspective model, some open source softwares, e.g. Docker, OpenStack, can be applied.

9 SUMMARY AND CONCLUSIONS

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 as a promising approach to improve system performance and user experience. This paper introduces

a review and taxonomy of brownout-based adaptive resources and applications management for cloud computing systems. A taxonomy is presented according to 5 phases: (1) application design, (2) workload scheduling, (3) monitoring, (4) brownout controller/dimmer design and (5) metrics. The taxonomy of each phase is summarized based on the different classification of brownout approaches. Then, a comprehensive review of existing brownout approaches is presented. Furthermore, the comparison of the advantages and limitations of the surveyed brownout approaches are also made. Finally, the future directions and open challenges of 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 for different optimization objectives. Brownout approach also provides a new option for adaptive management of resources and applications. This article shows the progress of brownout since it was firstly borrowed to Clouds, and it also helps readers to find the research gap existing in the discussed area. It is a promising direction to combine brownout with other existing techniques to achieve better system performance.

ACKNOWLEDGMENTS

This work is supported by China Scholarship Council, and Australia Research Council (ARC). We thank Editor-in-Chief (Prof. Sartaj Sahni), Associate Editor (Prof. Yeh-Ching Chung), and 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

[1] 2005. RUBBoS: Bulletin Board Benchmark. (2005). http://jmob.ow2.org/rubbos.html
[2] 2009. RUBiS. RUBiS: Rice University Bidding System. (2009). http://rubis.ow2.org/
[3] 2014. 1998 World Cup Web Site Access Logs - The Internet Traffic Archive. (2014). http://ita.ee.lbl.gov/html/contrib/WorldCup.html
[4] 2016. (2016). http://www.afr.com/technology/banks-websites-down-as-wild-weather-knocks-out-amazon-web-services-20160605-gpc8ob
[5] 2017. Docker Compose file version 3 reference. (2017). https://docs.docker.com/compose/compose-file/
[6] 2017. Docker Documentation | Docker Documentation. (2017). https://docs.docker.com/
[7] 2017. Weibo Servers Down After Lu Han Announces New Relationship. (2017). https://www.whatsonweibo.com/weibo-servers-lu-han-announces-new-relationship/
[8] 2018. Apache Mesos. (2018). http://mesos.apache.org/
[9] 2018. Apache Tomcat - Welcome! (2018). http://tomcat.apache.org/
[10] 2018. Open source software for creating private and public clouds. (2018). https://www.openstack.org/
[11] 2018. Production-Grade Container Orchestration - Kubernetes. (2018). https://kubernetes.io/
[12] 2018. Swarm mode overview | Docker Documentation. (2018). https://docs.docker.com/engine/swarm/
[13] 2018. VMware - Official Site. (2018). https://www.vmware.com/
[14] 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.
[15] 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.
[16] 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.
[17] 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.
[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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*. 1–32.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] Amazon EC2. 2018. Amazon Web Services. (2018). https://aws.amazon.com/ec2/

[29] Tammy Everts. 2016. Google: 53longer than 3 seconds to load. (2016). https://www.soasta.com/blog/google-mobile-web-performance-study/

[30] Grid5000. 2017. Grid5000:Home. (2017). https://www.grid5000.fr/mediawiki/index.php/Grid5000:Home

[31] MD Sabbir Hasan, Frederico Alvares, and Thomas Ledoux. 2017. 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.

[32] Md Sabbir Hasan, Frederico Alvares, Thomas Ledoux, and Jean-Louis Pazat. 2017. Investigating Energy Consumption and Performance Trade-Off for Interactive Cloud Application. *IEEE Transactions on Sustainable Computing* 2, 2 (2017), 113–126.

[33] Md 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.

[34] 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.

[35] 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.

[36] Neil Irwin. 2013. These 12 Technologies will Drive our Economic Future. (2013). https://www.washingtonpost.com/news/wonk/wp/2013/05/24/these-12-technologies-will-drive-our-economic-future/?utm$_t$erm=.e5ad3815c7eb

[37] 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.

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

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

[40] 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.

[41] 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.

[42] 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.
[43] 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.
[44] 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.
[45] 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.
[46] Peter Mell, Tim Grance, et al. 2011. The NIST definition of cloud computing. (2011).
[47] Alireza Sadeghi Milani and Nima Jafari Navimipour. 2016. 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.
[48] Bahareh Alami Milani and Nima Jafari Navimipour. 2016. 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.
[49] Gabriel A Moreno. 2017. *Adaptation Timing in Self-Adaptive Systems.* PhD Thesis, Carnegie Mellon University.
[50] 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.
[51] 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.
[52] 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.
[53] 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.
[54] 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.
[55] 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.
[56] 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.
[57] 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.
[58] 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.
[59] Sukhpal Singh and Inderveer Chana. 2016. EARTH: Energy-aware autonomic resource scheduling in cloud computing. *Journal of Intelligent and Fuzzy Systems* 30, 3 (2016), 1581–1600.
[60] Sukhpal Singh and Inderveer Chana. 2016. QoS-aware autonomic resource management in cloud computing: a systematic review. *ACM Computing Surveys (CSUR)* 48, 3 (2016), 1–46.
[61] 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.
[62] El-Ghazali Talbi. 2009. *Metaheuristics: from design to implementation.* Vol. 74. John Wiley and Sons.
[63] 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.
[64] 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.
[65] Wikibench. 2017. (2017). http://www.wikibench.eu/wiki/2007-10/
[66] 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.
[67] 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.

[68] 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.
[69] 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.
[70] 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.
[71] 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.