Performance-Aware Management of Cloud Resources: A Taxonomy and Future Directions

SARA KARDANI MOGHADDAM, RAJKUMAR BUYYA, and
KOTAGIRI RAMAMOHANARAO, The University of Melbourne, Australia

The dynamic nature of the cloud environment has made the distributed resource management process a challenge for cloud service providers. The importance of maintaining quality of service in accordance with customer expectations and the highly dynamic nature of cloud-hosted applications add new levels of complexity to the process. Advances in big-data learning approaches have shifted conventional static capacity planning solutions to complex performance-aware resource management methods. It is shown that the process of decision-making for resource adjustment is closely related to the behavior of the system, including the utilization of resources and application components. Therefore, a continuous monitoring of system attributes and performance metrics provides the raw data for the analysis of problems affecting the performance of the application. Data analytic methods, such as statistical and machine-learning approaches, offer the required concepts, models, and tools to dig into the data and find general rules, patterns, and characteristics that define the functionality of the system. Obtained knowledge from the data analysis process helps to determine the changes in the workloads, faulty components, or problems that can cause system performance to degrade. A timely reaction to performance degradation can avoid violations of service level agreements, including performing proper corrective actions such as auto-scaling or other resource adjustment solutions. In this article, we investigate the main requirements and limitations of cloud resource management, including a study of the approaches to workload and anomaly analysis in the context of performance management in the cloud. A taxonomy of the works on this problem is presented that identifies main approaches in existing research from the data analysis side to resource adjustment techniques. Finally, considering the observed gaps in the general direction of the reviewed works, a list of these gaps is proposed for future researchers to pursue.

CCS Concepts: • **Software and its engineering → Cloud computing**;

Additional Key Words and Phrases: Anomaly detection, performance management, resource management, big-data analytics

ACM Reference format:
Sara Kardani Moghaddam, Rajkumar Buyya, and Kotagiri Ramamohanarao. 2019. Performance-Aware Management of Cloud Resources: A Taxonomy and Future Directions. *ACM Comput. Surv.* 52, 4, Article 84 (August 2019), 37 pages.
https://doi.org/10.1145/3337956

1 INTRODUCTION

Cloud computing as an on-demand, pay-as-you-go environment has been modelled based on two main concepts of elasticity and virtualization. The inherent flexibility brought by these techniques

Authors' addresses: S. Kardani Moghaddam, R. Buyya and R. Kotagiri, The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, VIC 3010, Australia.

0360-0300/2019/08-ART84 $15.00
https://doi.org/10.1145/3337956

in the area of high-performance computing is accompanied by the complexity of managing distributed resources while meeting the expectations of the users. The emergence of the public Cloud Service Providers (CSPs) — such as Amazon and Google, which are extending the scientific limited applications of the cloud environment to industrial, academic, and personal use cases — makse the need for more advanced and complex resource management solutions highly important.

The main goal for CSPs is to find better ways of using resources while maintaining the stipulations of the service level agreements (SLAs) as expected. SLAs are contracts among CSPs and customers to maintain the minimum Quality of Service (QoS) delivered by the offered applications. Breach of the SLAs costs the CSPs both money and their reputation. Considering dynamic characteristics of the cloud, including unreliability and heterogeneity in resources and workloads, simple static resource planning solutions do not work. Therefore, traditional resource management infrastructure is extended with monitoring modules that can provide timely information on the performance of the application along with the resource utilization of system components. The collected data from monitoring the system and application provide a source of highly valuable information about the health of the system. On the other hand, advances in data-learning methods have provided missing parts of data-aware performance management, offering all the concepts and tools for analyzing data to find patterns, trends, and interesting changes in the behavior of monitored components. The integration of two parts of performance data analytics and automated resource management brings new challenges and opportunities in both areas of theoretical concepts and practical implementation. In this article, we aim to identify the major challenges and corresponding solutions to the problem of dataaware performance analysis and resource management in the cloud. We present a taxonomy to depict various perspectives of performance management in the cloud, covering all aspects of data collection, analytics, and resource adjustment solutions.

1.1 Related Surveys and Our Contribution

Although there are a number of survey and review articles identifying various aspects of data analysis or cloud resource management, they are more focused on one side of the problem without considering the requirements of other parts of data-oriented performance management frameworks. For example, Chandola et al. [19] present a survey discussing the general concept of abnormality in data, including various types of anomalies and applications of anomaly detection in the context of different problems. The article presents a high-level review of specific data requirements as well as mathematical models and algorithms such as classification and clustering methods to extract hidden information on existing patterns or features of data. Ibidunmoye et al. [51] investigate the anomaly problems in specific areas of performance analysis and bottleneck identification in computing systems and applications. They present various factors contributing to performance anomaly problems, including the types of bottlenecks, the granularity of knowledge expected from data analysis, and possible algorithms to solve these problems. On the other hand, Qu et al. [91] present a taxonomy on the resource scaling problem, focusing on the challenges of distributed resource management in the context of large web applications hosted on cloud platforms. They identify the challenges of dynamic resource management to meet specific requirements of web applications and categorize various scaling solutions to manage resource requirements of the application.

In contrast to these works, our work has a more integrated view of the problem of performance-aware resource management in the cloud. It covers both areas of application-dependent workload analysis and anomaly detection techniques and their contribution to resource management, particularly auto-scaling methods as the main resource level solutions for cloud-hosted applications. We have also tried to specifically cover the works that integrate both sides of performance data analysis and corresponding resource adjustment techniques, implementing the complete circle of performance monitoring and data collection, data analysis, planning, and decision-making, and

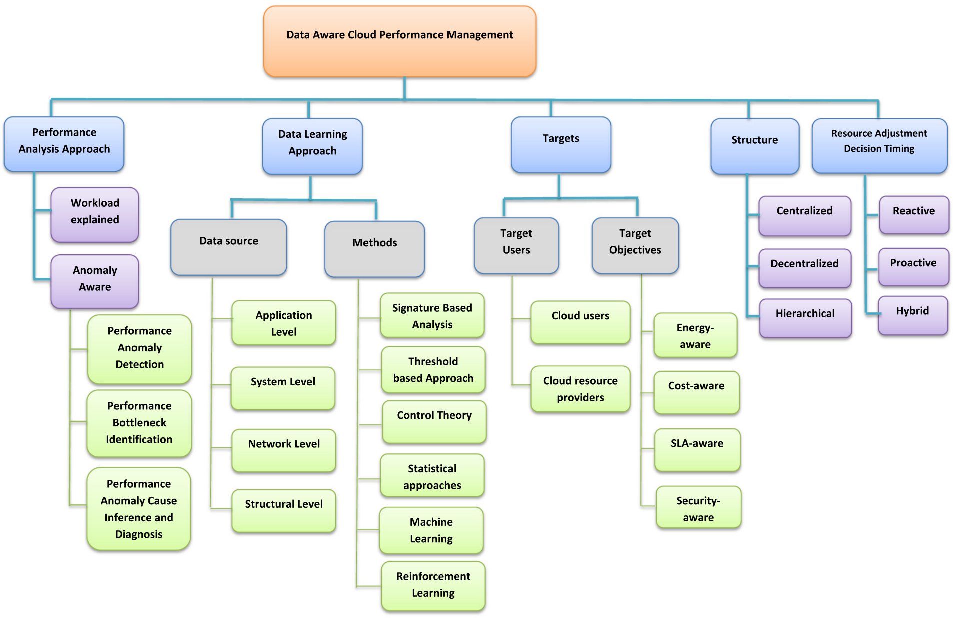

Fig. 1. The taxonomy of data-aware performance management in clouds.

the execution of selected actions. Moreover, we have a more updated review of the recent works in the area as well as new discussions on the research gaps and directions for future researchers. We also present a taxonomy of the source of performance-related problems, data analysis methods, and strategies to detect and handle anomalies, including scaling techniques.

1.2 Article Organization

The rest of the article is organized as follows: Section 2 describes the main blocks of data-aware resource management and existing challenges, followed by listing the most influential factors in this area. Section 3 introduces two main approaches in using data as a source of extra knowledge for resource management. Sections 4 to 8 review different characteristics of data analysis and resource management modules based on the categories identified in the taxonomy. Section 9 discusses the main gaps and directions for future researchers and Section 10 presents our conclusions.

2 BACKGROUND

The concept of resource management in the cloud environment encompasses all the techniques and procedures that help to adjust the configuration of resources according to the demands of the users and applications in the system. For example, auto-scaling solutions are based on a characteristic that allows system resources to expand or shrink at different levels of granularity (virtual machines [VMs], CPUs, RAM, etc.) automatically according to the perceived state of the system. To be clear about these concepts, we pursue the following definitions in the rest of the article:

***Performance Indicators*:** All of the measurable attributes from the resources and applications that demonstrate the degree of functionality of the corresponding unit in the system. These indicators continuously change over time and are initial sources of information for the health of the system. For example, the time it takes to load a page, known as *response time* (RT) from a web-based application, is the most perceptible sign of whether the system is performing at the expected level. Longer than usual RTs trigger the warning of having some sort of the problem, requiring technical considerations from the system administrators.

***System State*:** State or behavior of the system is an abstract representation of all of the operational attributes and performance indicators of the system that can be recognized in normal or abnormal/anomalous conditions.

The main indicators of an abnormal state are the presence of unexpected patterns or values in the performance indicators of the system.

Performance Degradations are caused by the abnormal behaviors when they affect the performance indicators adversary. For example, in the case of the increase in the number of requests (increased demand from customers) to a web server, if current resources cannot handle the newly received requests, the RT observed by users will increase. The unacceptable increases in RT are considered as performance degradation that should be avoided. One solution can be to add new resources corresponding to the overloaded component of the application so that the amount of resources is in accordance with the incoming load to the system.

Considering these definitions, any automated Resource Management Module (RMM) is dealing with two main challenges:

When is performance degradation happening in the system? In an ideal, highly reliable environment where no abnormal behavior is expected and applications show consistent behavior with a stable performance, traditional static scheduling solutions will work and dynamic scaling of resources is not required. However, in a real environment with a wide range of internal and external factors that can affect the behavior of the system, performance degradation has become an important challenge to be dealt with accurately. There is a wide range of causes identified for these problems, from fluctuations in the incoming workload to malfunctioning hardware or buggy software that can affect the performance of the application or VMs. Therefore, the onset time of the degradation should be known so that a proper and timely corrective action can be initiated. Monitoring sensors that follow the performance of each component generate vast amount of data, which include hidden patterns and signs of the health of the system. Previously, we had to rely on the expertise of human operators to skim data and find suspicious behavior. However, considering the scale of the data generated from thousands of machines located in different geographical locations, the manual approach is no longer feasible. Therefore, researchers have started to take advantage of advanced data analytics methods and more powerful and cost-effective computing hardware to automate and accelerate the process. This results in better-quality knowledge of the performance of the target systems.

What type of corrective action should be performed? In order to alleviate the performance problems of the system, the RMM should start a corrective action in the form of load redistribution, resource provisioning, migrations, and the like. Current resource providers, such as Amazon or Azure, offer migrations or simple threshold-based scaling services that change the number of VMs in the system. There are also more customized resource management policies, such as on-the-fly changes in the resource configuration of one VM, which is offered by some CSPs, such as [89]. The selection of proper action can be dependent on many factors, including technical or business limitations and type of problem. We have identified some of the most important factors, as follows:

- **Technical limitations:** Virtualization is the key concept for cloud models. It enables hosting different applications or the components of one application independently on one physical machine (PM) with a migration option available to move them to other PMs without significant downtimes in the system. Currently, many public resource providers such as Amazon and Microsoft Azure offer the required environment for CSPs to host their application on VMs and dynamically add/remove VMs in the system. There are also more fine-grained controls available to configure resources at the VM level, defined as *vertical* *scaling*. In this process, the size of the VM can change on-the-fly without any rebooting of the VMs. However, the functionality needs support from both hypervisor and the kernel of the VM. Currently, providers such as Amazon [5, 12] and Microsoft Azure do not support this functionality.

- **Business considerations:** There are a vast amount of the resources offered by cloud resource providers with various pricing strategies. For example, Amazon offers on-demand instances with hour/seconds-based pricing or much cheaper reserved instances with long-term contracts [5]. There are different pricing rules for vertical scaling of VMs, such as the offered rules by [89]. CSPs should consider these options when deciding on the configuration of their system and scaling policies. As a result, selecting the best action will be limited by the available budget predefined by the application owners. For example, in the case of a budget shortage, some levels of performance degradations may be acceptable from the owner's perspective.
- **Root cause of the problem:** In traditional threshold-based scaling, changes in the number of VMs is the most common response to performance problems in the system. However, there are a wide variety of reasons — from hardware faults to local software bugs in the application or security issues, such as Distributed Denial of Service (DDoS) attacks — that can create the signs of performance degradation. In cases in which resource shortage is not the main reason for the problem, adding new instances to the system may temporally alleviate the problem, but it is not optimal as a long-run solution. Moreover, as vertical scaling is becoming more prominent as a scaling option, having the knowledge of the underlying reason has become more important to formulating more cost- or resource-effective solutions. For example, in the case of a local memory shortage in one VM, a VM-level increase of available memory may be more effective than adding new VMs. A more detailed explanation of the pros and cons of these types of decisions are presented in Section 8.
- **SLA agreements:** SLA agreements are contracts between users and CSPs that identify the expected QoS received by customers. These expectations are usually based on the outputs of the system perceivable by customers, such as the availability of the service or the delays in response. Having specific requirements for the output of the system may limit available choices of the RMM. For example, CSPs may consider overprovisioning as a better option than dynamic scaling to manage high loads in the system when having a stable response time is highly important for the customers.

2.1 Data-Aware Resource Management

Motivated by the aforementioned challenges and requirements, researchers are leveraging various tools and concepts to offer more mature solutions for cloud resource management. An area that has been vastly investigated is data analytic techniques, which are bringing new opportunities and challenges in the area of distributed performance management. In order to apply these techniques, researchers are focusing on the obtainable knowledge from the data collected from performance indicators of the system and applications. It has been shown that these data are a valuable source of information on the health of the system and a starting point for detecting initial symptoms of abnormal behaviors. Based on the selected performance data to be monitored, the approach to the abstraction and modeling of the system, and the actions that are performed to mitigate the performance problems, different types of resource management strategies are proposed. In order to better understand the building blocks of these solutions, we first briefly review four main components of the data-aware performance management framework in the following paragraphs.

The main parts of automated resource management in the cloud can be explained based on a classic control loop known as MAPE (Monitor, Analyze, Plan, Execute) [53], which is shown in

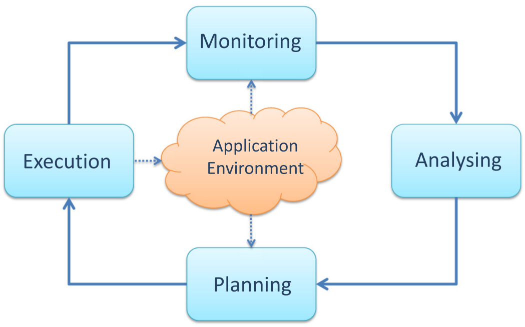

Fig. 2. General phases of a data-aware performance manager in the cloud.

Figure 2. The performance of the system is continuously monitored and a range of attributes from resources and applications is collected. The collected data are cleaned, modelled, and analyzed to identify any symptom of changes in the normal behavior of the system. Finally, based on the output of the analysis phase, a proper action is selected and the target components are informed to start the execution of the action. We briefly explain each phase and list the related categories of the taxonomy to each part below.

Monitoring: The performance of the system can be tracked by collecting the values of the attributes from the components of the system. These attributes include all of the workload metrics, system traces, network features, or performance indicators of the system, such as CPU and memory utilization, number of incoming requests, number of threads, or response time of the application. The **data level** part of the taxonomy indicates different levels of the data collected during the monitoring phase.

There are a variety of tools to help monitor and collect data from system components, including the *Top* and *Iostat* packages or *Ganglia* framework [37]. One can select a proper tool based on factors such as the granularity of data to be collected, level of access to the system components, scalability, and characteristics of the system.

One point worth mentioning is how to select a proper monitoring interval time. The interval can be as short as 1 second or as long as 1 hour. Smaller intervals make it possible to capture the fast-changing patterns or fluctuations with higher accuracy. However, the amount of the storage required for keeping all of the recorded data and overhead of processing and cleaning the data significantly increases. Selecting larger intervals reduces the overhead and required storage, but the possibility of missing or delayed detection of changes in the pattern of the performance data increases, which can cause delayed triggering of the corrective actions and more SLA violations. One should select a proper interval considering the trade-off between accuracy and computation complexity, reliability of the environment, and type of the application [6, 104]. For example, one approach is to define the sampling interval as a function of dynamicity of application by following the pattern of changes in the workload or performance indicators and adjusting the sampling interval accordingly. Another approach follows a fine-grained dynamicity analysis that considers the behavior of the metrics separately. In this approach, the monitoring intervals can be tuned at the metric level by having larger intervals for the metrics with no change points in past data. Smaller intervals are selected for highly dynamic metrics whose changing behavior is directly impacting the performance indicators of the application.

Analyzing: Two main blocks of the data analyzer module are data preparation and performance modeling/analyzing. All steps required for cleaning and filtering, dimensionality reduction, building the models, analyzing new observations, and deciding on the model updates when the state of the system changes are parts of this phase. A wide range of techniques and algorithms can be used to learn a model based on the historical behavior of the system. The **Data Learning Approach** and **Performance Analysis Approach** parts of the taxonomy present different categorization of existing methods for this phase.

Planning: The inputs for the planning module are the information about the current or future state of the system from the analyzer, current configuration of the resources from the application environment, and the objectives and constraints from customers or resource providers. Depending on the obtained knowledge, the module can select from a range of possible actions, such as adding/removing VMs, changing the configuration/placement of multiple VMs, or inbound traffic balancing. Decisions can be formulated based on past experiments and knowledge about possible causes of changes in the system. Therefore, the process can be implemented as a simple sequence of if-else rules or, at a larger scale, as a database that can map a combination of influential parameters to their corresponding mitigation action. The subcategories presented in Figure 5 focuses on this phase. A detailed explanation of possible actions can be found in Section 8.

Execution: This is where the final execution of planned actions in the system is performed. The module uses existing libraries and APIs to communicate with application components or deployed VMs to add new resources, remove the idle ones, change the configurations of existing VMs, or update load balancer configuration files. This phase concerns the development strategies and techniques that are out of the scope of the current research.

In the following sections, we present different aspects of a data-aware resource management solution based on the categories shown in Figure 1 and subcategories presented in Figures 4 and 5. Based on the categories and identified approaches, we map each work to corresponding features in Table 2 to give the readers a quick view of the main contributions of each work.

3 PERFORMANCE MANAGEMENT IN THE CLOUD

Monitoring tools collect a valuable source of the data to be analyzed and provide a timely update on the performance state of the application and resources. Data-learning approaches offer researchers all of the necessary concepts and tools to sift through the collected data and predict the future behavior or find interesting patterns of unexpected behaviors or anomalies with their possible causes. In this section, two main approaches for analyzing the performance of the system are presented.

3.1 Workload-Driven Performance Management

Performance of the system can be modelled and predicted based on workload-related features such as the number of requests received or amount of processing required at each time interval. Di et al. [30] propose a method for long-term load prediction in Google data centers. They consider load in the system as the main factor affecting the performance of the system and ignore other sources of data. In order to have a better representation of the statistical properties of the load, including trends and seasonality, different metrics based on load measurement values are derived. The prediction is done by training a Bayes classifier and exploiting a time window approach, which is a suitable way to smooth high fluctuations in the load. However, other types of anomalies that can be directly related to specific performance metrics cannot be detected in this approach, meaning that unexpected behavior can occur in the system, possibly causing negative impacts on the user experience. Work presented by Cetinski and Juric [18] considers a single attribute, number of required processors at a certain time, to estimate the utilization of resources. They expand the training dataset by introducing new attributes based on similar patterns in historical data. The results show that these new attributes improve the prediction accuracy of the Random Forest algorithm compared with the K-Nearest Neighbor algorithm. However, their prediction does not include the concept of unexpected behaviors resulting from various anomalous sources. VScaler, proposed by

Yazdanov and Fetzer [122], leverages a combination of workload prediction and reinforcement learning (RL) to automatically scale VM resources considering the user-provided SLAs. The RL approach in this framework helps to automate the learning process, considering the uncertainty of the environment in the form of changes in the workload model of the application. Another work, by Yang et al. [121], presents a cost-aware resource auto-scaling mechanism that considers both costs of adding new VMs and business software licenses during a scaling up procedure. A combination of linear ion-based workload prediction, integer programming–based prescaling, and threshold-based real-time scaling is introduced for capacity planning and resource management. Real-time scaling can be considered a reactive step to compensate for prediction errors, but the simulation-based validation of this approach ignores many complexities and time requirements of mentioned methods; thus, these assumptions must be carefully verified.

In the workload-explained performance management approach, the changes in the pattern of the workload are the primary influential factor that can affect performance and, hence, the resource decision-making of the system. The definition of the workload is dependent on the application and can be demonstrated as the number of requests sent to an interactive application such as web-based systems, number of tasks/jobs running in the system, and so on. One can also consider the resource demands of the jobs to be processed at each time as a representation of the existing load of the system. However, this assumption should be verified, whether resource consumption is a sole function of the load of the target application or the dynamic factors, such as the effect of background applications and sharing of resources, are considered in the process.

3.2 Anomaly-Aware Performance Management

A different approach to addressing the problem of performance management targets the abnormality in system behavior as a starting point for possible problems to be addressed by the RMM. In the context of big data–enhanced solutions, these performance problems are considered as outliers and anomalies in the data that can be identified by using a variety of methods, such as statistical or machine-learning algorithms. Therefore, we first define *anomaly* in a general context. Then, the problem of identifying anomalies in the context of the cloud is explained and existing works that follow this approach are discussed in more detail.

3.2.1 What are Anomalies? Anomalies are the patterns in the data that do not conform to the usual behavior of the observed data. The concept of anomalies and anomaly detection are very general and presented under different names, including *outliers and novelty detection*, *finding surprising patterns in data*, and *fault or abnormal behavior detection in systems* [19]. These areas have been investigated over a long period of time as part of medical and clinical data clustering, image processing and surveillance cameras, financial fraud detection, and several other applications.

3.2.2 Performance Anomaly Identification in the Cloud Environment. In the general definition of anomaly detection, the goal is to model the normal behavior of the system so that any unexpected change in patterns can be seen as an anomaly. However, considering the user-centric approach in resource management decisions in the cloud, application owners are more interested in the events that can affect the performance of the system and degrade the QoS experienced by the user. We refer to all of these events as *performance anomalies.* Considering that performance degradations can cause resource wastage, loss of reputation, and cost penalties for cloud service and resource providers, many researchers have investigated the relation between measurements from the system and application-dependent performance indicators to have a better understanding of different causes of performance problems. We identify three levels of knowledge obtained from the process of performance anomaly analysis as shown in Figure 3, which are detailed below.

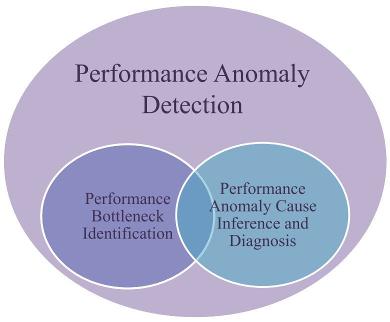
Performance Anomaly Detection
Performance Bottleneck Identification
Performance Anomaly Cause Inference and Diagnosis

Fig. 3. Different levels of knowledge from performance anomaly analysis.

Performance Anomaly Detection: The goal of a data analyzer module at the performance anomaly detection level is to find any abnormal pattern in the behavior of the system that can be a symptom of performance problems. Therefore, the input for these frameworks is usually the system and application performance indicators while the output is a performance alert when an anomaly is detected in the system. Having this goal and considering the fact that a correlation of different metrics can be related to various types of anomalies, Guan and Fu [45] present an automatic anomaly identification technique for adaptive detection of performance anomalies such as disk- and memory-related failures. Their proposed method investigates the idea that a subset of principal components of metrics can be highly correlated to specific failures in the system. A combination of the neural network method and adaptive Kalman filter is used in a procedure of learning from historical data, updating the prediction models based on the current prediction errors and adapting to the newly detected anomalies to improve detection performance. The work presented in [25] focuses on two general categories of anomaly sources, workload-related and performance-related data in streaming servers. They justify this separation as a requirement to select the best repair action in response to degradations caused by targeted faults. A feature selection procedure based on naïve Bayes is employed and the most relevant features are reported. Ashfaq et al. [10] target the problem of anomaly detection from a new perspective, highlighting the scalability problem of data analytic solutions for resource management issues in the cloud environment. They propose a general framework for anomaly detection based on splitting feature space into multiple disjoint subspaces and applying anomaly detection methods on each subspace separately. The idea behind using feature space slicing is to decrease the likelihood that a high number of normal instances can average out the effect of a few dispersed numbers of malicious instances during anomaly identification. Since this approach requires higher computation resources, as it needs to run multiple simultaneous instances of the algorithm on different subspaces, it is more suited for high-performance computing platforms.

Behavior Identification Architecture (BARCA) [23] is a framework for online identification of anomalies in distributed applications. It divides the anomaly detection process into two steps. First, a one-class classifier distinguishes normal behavior from unexpected behavior. Then, a multiclass classifier is used to separate different types of abnormal behaviors. The framework generates time series of different collected performance data and extracts new features, such as skewness and mean of data, which better represent the characteristics of the time series and help to reduce the dimensionality of feature space.

The abovementioned works target the reactive anomaly detection problem in the cloud environment. In order to be able to move the system back from the abnormal to normal state with minimum negative impact, we need to know about the probability of having abnormal values in the future. In proactive approaches, systems are able to exhibit goal-directed behavior by anticipating possible future abnormalities and taking action [50]. Gu and Wang [43] investigate proactive anomaly detection in data stream processing systems. Their proposed solution includes a phase of predicting resource utilization and then applying an anomaly identification algorithm on predicted data. Considering time sensitiveness of streamed data, the proposed procedure is online and the classifier will be updated periodically based on the new data. To address the prediction problem, they apply a Markov chain to capture changing patterns of different metrics to predict future resource utilization. Markov chains are based on the idea that the future state depends only on the current state and not past values. This assumption can be problematic, especially for data with recurrent patterns and events. Tan et al. [105] address this problem by integrating a 2-dependent Markov model as the predictor with tree-augmented naïve (TAN) Bayesian networks for anomaly detection. Another study by [26] investigates unsupervised behavior-learning problems for proactive anomaly detection. The proposed framework uses self-organizing maps (SOMs) to map a high-dimensional input space (performance metrics) to a lower-dimensional map without losing the structural information of original instances.

Performance Bottleneck Identification: Performance bottleneck identification goes one level deeper in the process of finding anomaly events in the data, trying to find possible bottleneck metrics that are closely related to the observed performance degradations as well. This approach is closely related to the problem of resource management, as it targets finding possible system resources that need to undergo a reconfiguration so that the provided resources meet the requirements of the application. Tan et al. [105] leverage TAN to distinguish the normal state from abnormal ones and to report the most related metrics to each type of anomaly. Canonical correlation analysis and Support Vector Machine (SVM)–based feature selection are used by the FD4C framework [116] to diagnose faults in web applications. They use a recursive approach based on feature elimination to rank the most important metrics for each type of the anomaly. Xiong et al. [118] have a different approach for detecting performance bottlenecks. They try to find the most relevant metrics in the performance of the application and follow the changes in these metrics as a sign of performance problems. However, they show that the predicted metrics are also good indicators of the source of analyzed performance problems, pointing to the source host and type of bottleneck resource. UBL presented in [26] uses the topological properties of SOMs to compare the anomaly and normal states and identify the metrics that are different between these states as faulty metrics.

Performance Anomaly Cause Inference and Diagnosis: The aforementioned anomaly detection approaches mostly focus on detection of abnormal symptoms and a coarse-grained identification of the possible *resource level metrics* that contribute to performance degradation. However, none of them digs deep into the data obtained from the application to find the underlying reasons for the observed problems. Indeed, identified bottleneck metrics can be indicators of having an application or VM level fault or inconsistency in the system: for example, high incoming load to the application or a faulty loop in the software code that saturates the CPU of the VM or the problem of VM/application contentions, which may cause degradations in memory utilization.

We can identify different directions in the fine-grained analysis of the source of the faults. First, we identify works that aim to localize the source of the fault to one *component*, such as nodes, VMs, or application components. For example, the works done in [82, 83] address the fault localization problem in distributed applications. The proposed frameworks combine the knowledge of inter-component dependencies with change point selection methods, taking into account that abnormal changes usually start from the source and propagate to other nonfaulty parts based on component interactions.

Another direction is to distinguish among different *types* of faults. Dean et al. [28] propose PerfCompass, which analyzes the generated system calls to distinguish between internal and external faults. They focus on software-related bugs, such as endless loops, as the target internal

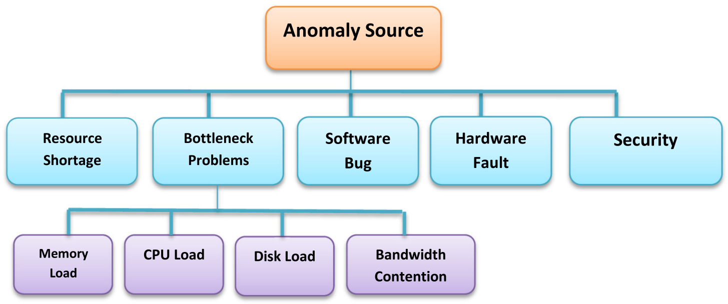

Fig. 4. Source of performance problems.

faults. Cid-Fuentes et al. [23] apply a set of the SVM-based binary classifiers to distinguish among livelock, deadlock, and starvation faults.

To achieve a more fine-grained identification of causes, Dean et al. [27] propose PerfScope to analyze the anomalies occurring due to the software bugs of the application. The framework studies the patterns in *system calls* and tries to find anomalous interactions between user and kernel. Triage [109] is another failure diagnosis online software package that identifies the conditions as well as the *code* and variables involved in the failure state. TaskInsight, presented in [124], focuses on thread and process-level performance information, which helps to localize the problem to the target anomalous task.

3.2.3 Cloud Performance Anomaly Root Causes. Cloud application owners typically start to allocate resources based on the recommended application requirements and then change resource configurations by continuously monitoring performance indicators to find performance violations. The root cause of these performance problems can vary widely, as shown in Figure 4.

Hardware faults include the problems that originate from corrupted or performance-degraded hardware that host target applications [31, 113]. Software-related problems can be caused by a buggy code in the application or misconfiguration that causes inconsistency in the functionality of software or interactions among components. This type of problem also can be caused by network-related bugs and misconfiguration that happen at the application level, including reported bugs in Skype, MySQL, or IPv6 compatibility issues [125].

Security problems, including attacks, are another source for unexpected behaviors caused by unusual pattern of requests, such as successful port-scans and attacks on the application server. A wide variety of literature targets this area, proposing various types of intrusion detection systems based on the concepts of statistical feature analyzing, classification, and clustering [10, 47, 65]. In order to identify these types of the problems, one needs to collect network layer datasets, including packet header information or the frequency of sender IP addresses to detect unusual patterns in the requests [10, 96]. Other sources of data to help recognize access patterns to the application are server log files, which record the history of authentication and user access requests over time.

Resource shortage issues are another reason for performance problems when the lack of enough resources to satisfy existing requests causes service interruptions and degradation in performance. The limitation can be owing to budget constraints or business policies that do not allow adding extra resources to the system or that reduce the amount of existing resources. Unexpected termination of services performed by resource providers is one example of problems that can expose the system to degradation of performance and throughput [90].

Bottleneck issues are another reason for the performance problems that are caused by insufficient resources in one or more components of the application. The problem can be due to the specific requirements of offered services, such as working with CPU-intensive software, and the lack of consistency among application demands with provided resources. Another example is the effect of background processes, which can temporarily saturate the resources of the machine, ignoring the requirements of other installed applications. It is worth noting that if the components are dependent and have interactions, the bottleneck problem in one part of the system can quickly affect other dependent components. As a result, performance issues will propagate in the system, causing application performance degradation [43].

4 TARGET

The body of literature regarding performance-aware resource management addresses the effectiveness of triggered actions from different perspectives in terms of the target area of the final solution. We distinguish two main factors that can affect this decision and, accordingly, design of the proposed frameworks. These factors are explained in this section.

4.1 Users

To establish a distributed, shared, pay-as-you-go environment, different players with heterogeneous and even conflicting objectives should be able to cooperate. Current literature mostly recognizes 3 main players/layers—cloud resource provider (CRPs), CSPs, and final users—as contributing roles in the design of resource management frameworks [87]. CRPs, also known as infrastructure providers, provide access to the pool of resources in the form of physical machines, storage, networks, and other types of resources that are necessary for creating a distributed computing environment. This can be provided as direct access to the physical resources or through virtualization technologies, usually via VMs. CSPs perform as the interface between CRPs and the final users, offering a range of services hosted on computing resources leased from CRPs. The final users are the customers that demand the cloud-hosted services by sending requests and data through predesignated interfaces. The separation of the layers is not always clear; sometimes, more than one role can be performed by one entity. For example, some CRPs also offer customized software packages as a service without the intervention of third-party providers [40]. Another approach distinguishes a broker layer that acts as a mediator between CRPs and CSPs. The broker has the information from both parties, including the user SLA and resource prices, and usually performs negotiations with multiple resource providers to find the offer that best meets the SLA requirements [15]. Although these layers each have their own responsibilities, from our perspective in the study of performance management, we identify two main groups as the target users:

- *Cloud users/application owners* who have access to the application-dependent information, including the code-level data, components design, workload patterns, and QoS requirements at the VM level
- *Resource providers/owners* who have information on the hardware characteristics of PMs and make decisions on VM allocation/placements

Depending on the target users, resource management can be adapted to address the improvements of measurable metrics to favor one or both groups [36, 123]. The selection of the target users can affect the selection of the objective as well as the source of the data to be processed in the learning procedure. The objective of the RMM determines the direction of decision-making in terms of the measurable metrics to be improved. Moreover, each group of users has access to different sources of data and can help the RMM decision-making through managing various parts of application, network, and hardware configurations. We have explained these concepts in Sections 4.2 and 6. It is worth noting that, in all cases, while one user group may be mentioned as the main executer of a proposed solution, other groups can also be involved when the final decisions indirectly affect their respective goal attainment. For example, CSPs may need to be considered when the final decisions (such as VM migrations/scalings) can violate budget constraints or predefined security concerns.

4.2 Objectives

Depending on the beneficiary of the proposed solutions, a variety of metrics are selected as the objective of improvements during resource management decision-making. As shown in Figure 1, we identify 4 main categories with regard to target objective metrics. It is worth noting that a combination of these objectives is usually considered in an optimization problem based on trade-off or through a list of the constraints provided by users.

- *Energy:* Energy-aware solutions try to minimize the power consumption of a system through a variety of mechanisms including application optimizations, dynamic scaling, VM configuration, and allocation and consolidation techniques [74]. Energy consumption of a system is usually calculated with regard to the resource utilization of PMs. Consolidation techniques concern the minimization of the active PMs by placing VMs so that the VMs can be used effectively while reducing the number of underutilized PMs as much as possible [59, 95]. VM placement [69, 95] and VM reconfiguration, including hardware tuning techniques [107], are other solutions that try to save energy by improving the performance of machines and reducing the power usage of hardware components. Load distribution is also among traditional approaches of SLA-aware resource management, which can also be used for the purpose of energy saving by prioritizing low-energy or renewable powered data centers during the resource allocation process [85]. These solutions are mainly focused on resource provider objectives and require direct access to the resources and information that may not be available for the service providers. In contrast, solutions that address the energy problem from service providers' perspective focus on component-level optimization of applications and green software designs [88, 119]. For example, [119] proposes an algorithm to dynamically select application components to be deactivated as a response to performance degradation in an overloaded machine. This helps the cloud service to be responsive to the users during high loads on the system by keeping the nonessential components of the applications in suspended mode.
- *Cost:* Considering the model of pay-as-go as the basis of cloud systems, market-based models focus on the monetary value of offered resources and services for cloud providers and consumers [32, 63, 123]. These models provide solutions to optimize the total cost of executing tasks considering a variety of pricing models. Resource utilization metrics — such as memory and CPU consumption, storage, and bandwidth — are among measurable metrics to be identified as cost indicators in proposed solutions. The cost-effectiveness of a final solution usually inversely impacts the QoS values (higher delays or runtime). Therefore, it is a common approach to consider a trade-off between the cost and target performance metrics as the objective function to reduce or penalize the adverse impacts on QoS [63, 101].
- *SLA:* Alternatively, a large body of literature addresses the problem of resource management with more customized approaches to target specific application requirements. These requirements are usually noted in the SLA contracts and their violations incur penalties. A variety of indicators—such as response time, service up-time, and scalability—are included in this category [58, 100, 123]. For example, [123] addresses the availability-aware resource provisioning by considering service agreements on the minimum accepted up-times. The violation of these SLAs causes penalties that are taken into account in addressing optimization problems.
- *Security:* Security-aware solutions offer mechanisms to protect the safety and integrity of users, data, applications, and underlying infrastructure [20]. Alongside intrusion-aware

frameworks, which make use of traffic- or VM-related data to detect possible attacks on the application, this category also deals with policies to protect data, including user data provided to the application or user behavior patterns. A variety of solutions are combined to secure the integrity, availability, and confidentiality of data during phases of the data life cycle in the system [20]. These solutions can be coarse-grained approaches, including VM migration/placements and workload isolation or fine-grained solutions that directly target data security by encryption, isolation, and cleansing techniques [29].

5 ARCHITECTURE

In an environment with geographically distributed resources, decisions regarding placing and interaction among components can highly affect the performance of the system and corresponding corrective actions. Various factors — such as the amount of available storage, resource demands, or the speed of information dissemination — can contribute to these decisions. In the following, we briefly discuss three main approaches for the placement of different components of a resource management framework in the environment.

5.1 Centralized

Traditional frameworks to analyze the health state of the system typically follow a centralized approach. In a centralized structure, all data from local components, including performance indicators and resource configurations, are sent to a master node. The master node is responsible for maintaining a continuously updated model of the whole system and triggers alarms when a performance problem is detected. All of the tasks regarding workload prediction, resource utilization estimation and performance problem analysis is done at this module [24, 56, 93].

An advantage of a master node approach is to have all system-related performance information in one place; in this manner, one can analyze the interactions and relation among the metrics at different layers and track the fault propagation among connected components. However, as the scale of the system increases, there will be more components and resources to be monitored, usually in geographically distributed regions. The process generates a huge volume of collected data to be transferred and analyzed in one place. This makes system modeling and abstraction as well as triggering a proper resource management action to be traversed across all involved resources super complex, computationally intensive, and time-consuming.

5.2 Distributed

In a dynamic and scalable system, administrators are more inclined to deploy computing modules as decentralized components [43, 92, 105]. Therefore, each VM/PM or small clusters of machines in the system will have a local analyzer dedicated to processing locally collected monitored data to model the behavior of the system and locally decide the resource configurations. This approach can be easily deployed and has higher scalability and manageable computation time. Moreover, failures in one node do not affect the functionality of the remaining processing modules. This approach is also used in the fog-computing–based Internet of Things (IoT) infrastructure to model a fine-grained connectivity that extensively uses the advantages of having multiple layers of distributed computing for a highly scalable environment to connect people and devices [72]. Distributed computing helps to significantly decrease decision-making time, as it deals with smaller environments (monitoring one VM or host compared to the whole environment) with a reduced amount of data or problem-causing factors.

5.3 Hierarchical

While distributed architecture solves the scalability issue of centralized approaches, the lack of a central manager makes it hard to include the interaction and dependency among components during analysis. Each module has an abstract performance model of its local environment, completely ignoring the effect of any external factors such as the dependency among different layers of multi-tier applications. Moreover, distributed modules do not have a big picture of the system to decide on resource adjustments at higher levels of granularity, such as adding new VMs. As a solution for this problem, one can consider a combination of centralized and distributed architecture in a hierarchal model, where each monitored component has a local analyzer module to get up-to-date information on local behavior and trigger local corrective actions [68, 79]. A summary of the local state of the component can also be shared to a central module [2, 82]. The central module uses a combination of knowledge from local states, the dependency among components and high-level information of the functionality of local components to create a general model of behavior for the whole system. This approach combines the scalability and flexibility of distributed methods with systemwide knowledge of the centralized master node, which helps the system to respond to local problems quickly while having enough knowledge to plan for global problems.

6 DATA-LEARNING APPROACHES

The data-learning module of the RMM receives a set of the raw data as input and tries to extract up-to-date information on the performance state of the system to be sent to the decision-making module. Depending on the objective of the solution, as described in Section 4.2, the required metrics are collected from the system and processed with a variety of data analysis techniques, including statistical and machine-learning approaches. A discussion of the sources of data to be monitored and techniques for analyzing data is presented in following sections.

6.1 Data Sources

Monitored metrics are the measurable features that provide a basis to describe and model the state of the system. Depending on the target users and objective of the proposed solutions, a variety of metrics from system resources and application components can be collected. In general, we have identified four primary sources of data that give information on the system from different perspectives to be processed by the analyzer.

System-Level: System-level metrics refer to the collection of attributes that can describe or predict the behavior of the running environment, including VMs or PMs. One category of these attributes is resource-level metrics, which act as the performance indicators of the running system at different levels of granularity from VMs to specific processes and threads. For example, one can present the number of assigned CPU cores or the percentage of used CPU, memory or disk I/O at different time intervals as indicators of the functionality of the system during runtime of the applications. Another source of data that can be categorized as part of the system metrics are generated system calls that show the pattern of interactions with operating system services. It is shown that these patterns can be affected by different types of internal and external faults that help to detect and localize the source of the faults [27, 28].

Many cloud resource providers offer monitoring services to collect data from PMs and VMs. Amazon CloudWatch is an example of these services. There is also a range of system monitoring and application debugging tools such as htop, Iostat and strace, each offering a level of information about utilization of resources or pattern of interactions among processes in the system. While these tools give valuable information about the functionality of one machine, their use for a cluster of machines needs more scripting and data management. For a more flexible monitoring of distributed systems, advanced frameworks such as Ganglia are introduced [37]. Ganglia is a distributed framework based on the hierarchical design that uses technologies such as XML and RRDtools for monitoring different components of the system. It is accompanied by a dashboard to view live statistics of the monitored system while the recorded data is also available for deeper analysis. Aceto et al. [1] provide a detailed analysis of various monitoring tools.

One point worth mentioning here is that some of the abovementioned tools require direct access to the monitored VMs or some components of the monitoring modules should be installed on the machines beforehand. Therefore, the outputs of these monitoring components are accessible by cloud application owners (*cloud users*) who have access to the VMs. Moreover, cloud users have direct access to the components of the application. Therefore, QoS-aware solutions that require knowledge of system performance are designed with the assumption of having an access level that is the same as the application owners. Alternatively, there are works that follow the blackbox rules, trying to avoid or decrease the dependency on the application or guest VMs by using hypervisor capabilities to collect data from outside of the VMs [18, 56, 105]. These solutions can be used by cloud resource providers. A combination of having the knowledge at the VM level accompanied by the capability to access the underlying hardware (including hardware tuning and VM/storage server placement) gives the resource providers a great deal of power to manage and adjust the utilization as well as SLA, such as privacy and security requirements of application owners [104].

Application-Level: Application-level metrics are collected from application components deployed in the system. Since these metrics are directly related to the runtime state of the application, they can be very informative and good indicators of the health of the application or environment. For example, in a web-based application, response time, which is the delay from initiating the request until the user receives the results, is considered as an indicator of the quality of the offered web service. Longer response times can be warning signs for a high number of requests or some problem inside the systems [105, 121]. There are also more fine-grained works that study the software code and debug the flow of data and compare the effect of various input variables or environment configurations [109]. These works are more related to the field of software debugging or runtime diagnosis. However, we have included them in our survey as they can help to distinguish internal application faults from external ones, which consequently affects the type of the corrective actions from the service provider side.

Network-Level: There is a body of work that studies knowledge obtained from analyzing network-level data, such as packet headers or the frequency of received packets from specific users, which makes them suitable for identifying network-related issues, particularly security threats such as DoS attacks [10, 96]. The source of the problem in these cases is usually associated with external factors; therefore, these frameworks are complementary to the ones that directly target the internal state of the system and resources.

Structural-Level: While a per-component monitoring gives valuable information on the functionality of individual parts of the system, these components are in continuous interaction in a distributed system. In other words, the functionality of one part may be dependent on the correct execution of another part. Therefore, the fault in one component can be quickly spread based on the application architecture and path of flow of the data/commands among components. Having the information on the execution order of application components along with the timestamped data of the performance metrics can give new insights into localizing actual sources of the faults that are propagated from different layers of the application [82, 83].

6.2 Methods

The core part of data-aware resource management is the data analysis module, which obtains knowledge on the current or future state of the system to select the best action and keep the system compliant with the SLA requirements. There are different approaches to learning and analyzing the health of the system from collected measurements. It is also a common approach to combine two or more of these techniques for different parts of data analysis and decision-making modules. We categorize and summarize the characteristics of the identified approaches.

6.2.1 Signature-Based Analysis. The state of a system can be characterized by the values of the attributes of its components at different levels of granularity. Considering that various types of faults or performance problems leave distinctive signs on the attributes, one can capture a snapshot of all values during abnormal/normal behavior and represent it as the fingerprint of this state. The works presented in [78] and [22] distinguish performance problems caused by anomalies from those that are the result of an application update or changes in resource consumption models. They create a profile of the application performance based on the concept of transaction processing times and corresponding resource utilizations. The profiles are used for the comparison between old and new application performance to detect the changes. While these works leverage the application-related measurements to create profiles, Brunnert and Krcmar [14] use resource profiles to detect performance changes in enterprise applications (EAs). The resource profiles are defined as the amount of required resources, including CPU and memory for each transaction of an EA version. Resource profiles are not dependent on hardware characteristics or workloads; therefore, they are more robust solutions for areas such as capacity planning or energy estimation. Another approach is to use the profiles of events for diagnosing types of anomalies. For example, Sharma et al. [97] propose a fault management framework that first detects an anomalous behavior by statistical analyzers. Then, detected deviations are matched with the predefined signatures of the faults to identify the cause of the problem.

While signature-based approaches usually show low false-positive rates, a key challenge is the creation of baseline profiles capturing the states of the system. For anomaly detection, creating signatures requires domain knowledge of the problem and there is always a high chance of missing unknown anomalies, which increases the false-negative rates in the results.

6.2.2 Threshold-Based Approach. The threshold-based approach is a simple yet popular way among cloud providers to define a set of rules to manage the resources in the system [5]. The idea behind this approach is that anomalies can cause an unusual increase or decrease in the utilization of the resources, affecting the values of attributes of the system or application. One can define the scaling up/down rules by identifying a threshold for the acceptable utilization in the system. If the target utilizations exceed the threshold, scaling actions are triggered. Therefore, two main parts of each threshold-based rule are the condition and the action. Regarding the condition part of the rule, an attribute of the system that can be a resource-level metric or application-level performance indicator is selected. Then, proper lower/upper thresholds are identified. Whenever a threshold value is exceeded, the conditions are met and the action is started. The second part of the rule is defining appropriate actions, such as deciding on the number of VMs to be added or the VMs that can be shut down in the system. These actions, also known as *horizontal scaling policies*, are offered by most cloud resource providers. Gmach et al. [38] investigate the reactive threshold-based approach to detect overutilized or underutilized servers. Yang et al. [121] extend this approach with a linear regression-based prediction phase and apply one of the vertical or horizontal policies when a violation of the threshold is met. There are also works that implement the threshold-based policies for baseline comparison with their proposed frameworks [46, 49]. For example, Hong et al. [49] compare a Markov-based anomaly detection scheme with a simple threshold-based monitoring that triggers anomaly alerts when the resource utilization thresholds are violated. While this approach is very common and easy in terms of implementation, it requires a deep understanding of workload patterns and trends.

Considering the dynamic nature of today's applications, threshold-based solutions cannot fulfill the complex—sometimes conflicting—expectations of application and resource owners.

6.2.3 Control Theory. To improve the degree of adaptation, the mathematical concepts of control loops are investigated to create responsive strategies for dynamicity of the environment. Control loops help to automate the resource scaling decisions by creating a systematic way of adapting to changes in the system. The controller should trigger proper corrective actions by adjusting the values of input variables to maintain the output or controlled variables close to a baseline. The process usually is designed as a loop, with a variety of metrics from the system as input. The outputs are translated to some type of the action in terms of the system configuration adjustments. Regarding *open-loop controllers*, the corrective action is selected solely based on the inputs. In the *closed-loop*, also known as *feedback controllers*, the changes in the controlled variable are received as feedback to be considered by the controller for the next action. While the latter is the most common architecture for implementing the adaptability in the dynamic environments, two other extensions of this architecture, known as *Observe-Decide-Act (ODA)* and *Monitor-Analysis-Plan-Execute (MAPE)*, are also exploited in the area of autonomic cloud resource management [21]. The ODA frameworks typically cover 3 main roles of observer, decider, and executer with the main goal of decoupling the responsibility at different steps among respective players involved during the system development process [48]. The separation of responsibilities allows application and system developers to deeply focus on knowledge from their part of the problem and also makes the final system more adaptable to different applications and systems. The MAPE framework is another extension that breaks the action part of the general loop into two subproblems of analysis and execution. Following this architecture, Aslanpour et al. [11] propose a cost-aware autoscaling framework with the focus on possible improvements at the execution level. Nevertheless, in general, the feedback controllers are the most common architecture to be investigated in this article. Integration of the Kalman filter and feedback controllers are studied in [57] to manage the allocation of resources based on the CPU utilization of VMs. Al-Shishtawy and Vlassov [4] combine feedback and feedforward controllers, harnessing the power of both approaches for multitier applications. The Feedforward part is acting as a predictive controller to proactively avoid SLA violations caused by unexpected increases in workload. In the case of violations, a feedback controller reacts to compensate the deviations in the performance.

Lyapunov control is another approach for solving optimization problems, especially for online decision-making where detailed information on system behavior is not available. Lyapunov-based systems are also applied in optimization problems when the stability of the system (e.g., the queuing delays) is a point of interest [71, 111]. For example, Lu et al. [71] use this approach to provide a cost minimization model with controllable provisioning delays in an auction-based resource management. This approach has been shown to perform better compared with traditional methods in terms of dealing with the dimensionality problems in large systems without the preknowledge of statistical features of controlled components [35].

Proportional-Integral-Derivative (PID)–based controllers are another technique, exploited in [41], for managing the number of VMs in the system, aiming at keeping service quality in accordance with agreement levels. Alternatively, considering the need for more flexible systems with the capability of regulating more than one metric, Persico et al. [87] combine a fuzzy-based scheduling algorithm with a PID controller for horizontal scaling of resources. In contrast to the model-based controllers, which need some knowledge of the dynamics of the environment [39, 57], a PID-based controller makes the system more flexible in a model-free adaptation, usually through the iterative tunings of the parameters. However, despite the inherent simplicity of PID controllers to cover a broad range of applications, they may suffer from oscillation or delayed convergence, especially in highly unstable systems.

Predictive models are another approach commonly applied for complex, highly dynamic environments. Model Predictive Controllers (MPCs) consider dynamic models of the system over consecutive time slots (time-horizons). The current state as well as future predictions are taken into account to repetitively optimize the controller model. APPLEware is a distributed MPC-based middleware that optimizes both energy use and performance for co-located VMs [62]. The control actions are in the form of CPU and memory changes and are computed based on the predictions of energy and performance over prediction horizons to optimize the cost function at corresponding time intervals. Another instance of the predictive models is Receding Horizon Controllers (RHCs) [61], which constantly solve optimization problems over a moving time horizon [77]. The iterative optimization of RHC is used in [8, 9] to minimize the cost for reserved and on-demand VMs while meeting constraints on the response times of applications. Similarly, Roy et al. [94] apply this model for minimizing resource allocation costs considering both costs of leasing resources and the penalty costs of SLA violations. Incerto et al. [54] exploit the iterative optimization of RHCs for fine-grained adaptivity at the application level. The control model is used to automate parameter tuning of software components that are modeled based on queuing networks. While the iterative optimization of these models helps to achieve better adaptability to changing environments, it also increases the runtime complexity for solving optimization models at each step.

6.2.4 *Statistical Approaches.* This approach usually assumes that the key attributes of the system follow a known or inferable behavior. Therefore, observing and collecting the data on the system attributes provides a baseline from which any deviation is identified as an anomaly. The definition of baseline behavior is usually based on some statistical characteristics of the data, such as mean and standard deviations. For example, many works on anomaly detection are based on the assumption that values of target attributes follow a normal distribution. Multivariate adaptive statistical siltering (MASF) is a common method in this group that tries to find multiple sets of control limits based on statistical analysis of the previous measurements of the features during normal system operations [16]. The observations outside of the control limits are considered as possible anomalies in the system. Wang et al. [115] generalize this concept to more flexible thresholds, being more adaptable to dynamics of the workloads in data centers.

While these solutions are simple and lightweight, the highly dynamic nature of the cloud requires more flexible solutions that can capture the relation among features. A body of work [52, 75, 86] attempts to show some type of correlation among resource metrics and QoS indicators, using this information for better understanding of the nature of the anomalies and further performance analysis, such as cause identification. Another approach that addresses the problem of proactive resource scaling leverages regression-based methods to find the relation among metrics and performance indicators [56, 114, 118, 121]. The prediction of future workloads or resource utilization gives insight into the possible changes in the system that require a reconfiguration of resources. For example, MLscale is an auto-scaler that uses the regression method to predict the values of metrics and, consequently, the performance indicators of the system through a hypothetical scaling and decides on the best action based on the results [114].

Another area within the domain of distributed resource management, where statistical techniques have been commonly used, is analyzing the network-level state of the system, which helps to distinguish between normal traffic and network-related security issues, such as attacks. Gu et al. [44] employ relative entropy to compare the new traffic data with the baseline distribution and identify anomalous traffic. Cao et al. [17] target DoS attacks launched by malicious tenants of the VMs in cloud data centers. Entropy is calculated based on resource and network utilization information. They show that the entropy of a VM's status drops when the attack starts. Ashfaq et al. [10] divide the feature space of the problem based on the information content concept, putting statistically similar instances in the same subspaces. The idea behind this approach is to avoid the effect of averaging out of anomaly points and also localizing noise artifacts in separate subspaces.

6.2.5 Machine Learning. Machine-learning concepts include techniques that enable a system to learn from experiences over time without being explicitly programmed. The massive amount of collectible data from a system is a valuable source of the information to be used by these techniques to learn from the environment. Each technique tries to structure data in a different way to generate an abstract model relating the input to output variables. Generally, we can divide these techniques into two main categories: supervised and unsupervised. In the following, we explain each category in more detail.

- *Supervised Learning*: Supervised algorithms require the dataset to be labelled, meaning that the desired output should also be clear during the training phase. This approach is more suitable for problems in which the goal is to find a mapping between the input and output variables; thus, having the new input observation, one can find the possible output. A common case of using this technique is the classification of a set of records when each input should be assigned to one of the predefined classes. In the area of anomaly detection techniques, the classifiers can be used to categorize different types of anomalies or more generally distinguish between the normal and anomalous state of the system. Following this approach, Gu and Wang [43] try to detect the type of future anomalies by using a naïve Bayesian classifier. A set of binary classifiers is trained to distinguish among different types of bottleneck problems. The authors show that the proposed classifiers can achieve high accuracy, detecting the anomalous symptoms caused by some of the common bottleneck issues at the application and resource level. Similarly, Tan et al. [105] exploit TAN to predict the anomalous state of the system. Cunha and Moura e Silva [25] apply two classification algorithms, J48 trees and naïve Bayes, on the historical data through tenfold cross-validation. The goal is to differentiate between workload-related anomalies that are caused by higher request rates and other types of performance anomalies. Decision trees are leveraged in [29] for classification of traffic data by supervised learning of attack types. The authors suggest that the simplicity and interpretability of trees can help humans to better understand the nature of problems and their related features. Supervised learning is also applicable for learning the features. Accordingly, Shi et al. [99] exploit the knowledge from principle component structures and data labels to learn a more robust set of features for traffic classification.
  Neural networks are commonly applied for the prediction of future utilization, performance indicators, or workload metrics to model the performance of the application [3, 56, 114]. Rather than a direct identification of anomalous events (considering only the context of the data and patterns), these works usually focus on finding the symptoms of performance degradation in the application. In contrast, Guan and Fu [45] identify anomalous events by combining neural networks and Kalman filters to adaptively calculate the principal components of data. The idea behind their approach is that a subset of principal components is more related to specific types of failure in the system.
  The aforementioned models are trained to find a relation between input and output variables to predict output values for test instances. The outputs can be related performance metrics, such as response time or a category of anomalous states assigned to the input instance. A major limitation of these works is the requirement to have a labelled dataset for training. The process of labelling a dataset is time-consuming and needs adequate knowledge of the domain problem. Moreover, in a dynamic environment, there is always a chance that the

underlying mapping of the variables changes, which requires continuous reconfiguration and regeneration of the models for the new states of the system.

- *Unsupervised Learning*: In contrast to supervised learning, the unsupervised approach sifts through data trying to find hidden structures and patterns. Therefore, it does not need any prior information about the labels of training data. The objective is to cluster input data based on their features without any assumption about their distribution [124]. Unsupervised learning is particularly suited for the cloud environment, where the system administrators may not have access to detailed VM utilization states or may be unaware of the internal performance of the application. Following this approach, Dean et al. [26] propose unsupervised behavior learning (UBL) and investigate the unsupervised learning problem for proactive anomaly detection. UBL is a framework that applies an SOM to identify the anomalous states in cloud systems. An SOM is an unsupervised type of artificial neural network that projects data instances from a high-dimensional space to a lower space (usually two) while keeping the topological structure of the data. Comparing the neurons of the generated map distinguishes normal and anomalous states while a list of ranked metrics can also be inferred as a springboard to finding the cause of the problems. Hidden Markov Models (HMMs) are used in [49] to model system as a Markov process. In a Markov process, it is assumed that the state of the system at each time is dependent only on the previous state. Two hidden states, normal and anomalous, are determined and the probability matrices are initialized through an unsupervised training process. A two-phase clustering approach is proposed in [64], which tries to find a VM placement solution to minimize the number of PMs and reduce the performance degradation caused by the contention between co-located VMs. Hierarchical and K-means clusterings are applied to cluster VMs based on the peak of utilization metrics and the correlation among them. Alternatively, Ashfaq et al. [10] apply clustering as a preprocessing phase to divide feature instances into distinctive categories based on their statistical attributes. These clusters form the basic blocks of data to be analyzed separately by anomaly detection modules.
  While the traditional methods of behavior learning can help to identify anomalous events, the detection is usually a by-product of other purposes, such as clustering/classification. Isolation-based technique is another unsupervised approach that addresses this problem by directly targeting the characteristics of anomalous instances, which are few and different from normal instances [66, 67]. This method is leveraged in [84] to design a sequential unsupervised learning of the features where the calculated scores from isolation-based trees are used as a signal for the selection of a subset of features for the next iterations.
  Unsupervised learning helps the system to detect both known and unknown anomalies. The process does not assume any prior knowledge about the statistical features or patterns in the data and tries to find the common characteristics observed among different sets of instances. However, depending on the level of details provided in data, the accuracy of unsupervised learning to recognize the exact categories of anomalies may be affected.

6.2.6 Reinforcement Learning. Reinforcement learning (RL) focuses on the gradual learning through sequential interactions of the agents with the environment. The target goal of the agent is to maximize a reward function by selecting the best possible action based on the state of the system. The important feature of this approach is learning by experience from the environment, which helps to start the process without a prior knowledge of the system. In the area of cloud resource management, an auto-scaler can act as an agent that interacts with system components, including VMs and PMs. The state of the system is represented by the system attributes and performance indicators, while the reward is shown by the degree of QoS (such as response time or throughput) achieved by the application. The set of actions includes all possible corrective actions, such as resource- and application-level reconfigurations to avoid performance degradation. Dutreilh et al. [34] compare threshold-based and Q-learning approaches for the problem of horizontal auto-scaling in the cloud. They investigate the functionality of each method in the presence of the main instability sources in the control systems, listing the observed potentials and weak points for each case. Duggan et al. [33] target the problem of VM live migration, considering the available bandwidth and network congestion problem. They formulate the problem as an autonomous control system through using RL and creating a multidimensional state/action space based on VM utilization and available bandwidth. VScaler, proposed in [122], is another framework for fine-grained resource management in the cloud that uses RL to decide on the times to scale up/down in the system. To speed up the process of learning and exploration of the controller, the authors introduce the parallel technique, which enables multiple agents to collaborate in different parts of the state space.

As we can see, the gradual learning concept provided by RL fits the nature of the problems in cloud performance management very well by involving dynamism and uncertainty factors during the learning procedure. However, a main challenge that RL-based solutions face is the size of the possible states and actions for the system. Considering the continuous nature of time series measurements and the scale of the target machines to be handled in distributed environments, the problem of high dimensionality is becoming more important. To overcome this limitation, different approaches, such as fuzzification of the table or using more abstract representation of data to limit the possible states or actions, are proposed in the literature [7, 68]. Arabnejad et al. [7] extend a rule-based fuzzy controller with 2 different RL approaches, Q-learning and SARSA. The fuzzy concept helps to reduce the dimensionality of the state/action table, which is an important issue affecting the complexity of the RL algorithms. Alternatively, a combination of dimensionality projections and sparsity-based data structures are used in [13] to overcome the dimensionality problem. The proposed approach is used to manage efficient migration of VMs in real time with respect to the energy and performance of the system. Adaptive partitioning of state space is another approach that initiates the model with one or few states and gradually decides on the partitioning of states during the interactions with the environment [70].

7 ACTION TRIGGER TIMING: FROM REACTIVE TO PROACTIVE

When a corrective action should be triggered is a challenging question, as it is highly dependent on the nature of the application and SLAs. Traditional approaches to this problem are mostly reactive. In *reactive methods*, any decision about changes in the number of VMs, configuration of resources, or VM replacements is a response to the abnormal behavior of the system indicators that identify changes in performance or QoS. As the degradation already has occurred and considering the delays before corrective actions take effect, a certain amount of SLA violations should be allowed in SLA contracts. Moreover, in an unreliable environment involving various factors that affect the stability of the system, fluctuations in performance are a common observation, which can increase the number of SLA violations. Usually, these approaches follow a threshold-based strategy where the scaling starts when the measured metrics exceed the accepted values [5, 55]. For example, when CPU utilization of the server exceeds the threshold, new resources are added to the system. Therefore, the reactive approach does not consider the performance anomaly as a gradually happening event with detectable presigns.

A step further in adaptive design of the resource management system is to include periodic actions to resolve possible performance problems at each time interval. While the reactive approach performs resource adjustment action as a response to performance degradations, periodic solutions are triggered by time. For example, software rejuvenation is a preventive technique that tries to clear the state of the software. This process can be repeated at regular intervals to resolve software performance degradations or avoid possible problems caused by the software aging phenomenon [76, 81]. These time-triggered techniques can also be categorized as proactive approaches when regular updates of the system help to alleviate performance problems before they cause violations of SLAs. Alternatively, *event-driven* proactive approaches perform corrective actions when an event is detected that is suspected to affect the behavior of the system in an undesired way. Event-driven proactive methods attempt to find the warning signs before they can cause unacceptable levels of performance degradation so that they can start preventive actions, such as migrating VMs or adding new resources [79, 105, 121]. They focus on future events and are mainly based on the prediction of future values and states. If proactive analysis of data can give an early enough alert of a possible performance problem, it helps the RMM to plan and quickly initiate a proper action before the system goes into an anomalous state. Accordingly, the system returns or continues the normal condition, reducing the number of violations. One point worth mentioning here is how to decide on a proper value for the period of prediction. Short-term predictions are more accurate in the case of workload-related metrics because the measurements made during short periods of time show higher correlations compared with the observations made during longer periods. Longer time predictions are more challenging and better fit the data with regular patterns or seasonality. Di et al. [30] investigate the long-term prediction problem for workload data in cloud data centers. They propose a Baysian-based method to predict the average load in the system based on the derived features that capture different aspects of the statistical characteristics of data. Another strategy to decide on prediction interval is considering the required time for the corrective action to be effective in the system. If the workload is predicted for an interval shorter than the action time, the value of proactive resource management diminishes. Islam et al. follow this approach to determine a prediction interval based on the time it takes to launch a new VM. Therefore, upon receiving an alert of a possible load problem, the system has enough time to start the scaling action. Finally, *hybrid solutions* are based on a combination of reactive and proactive approaches. These solutions are more realistic for the applications in the real environment where there is always the possibility of new and unknown events happening that are not detected by proactive mechanisms [38, 121]. In these cases, the reactive component helps the system to decrease the adversarial effects of undetected anomalies in the system while the proactive module can learn the new undesired behavior by updating prediction models with recent observations.

8 PERFORMANCE ADJUSTMENT METHODS

Upon receiving an alert of an ongoing or possible performance problem in the system, the RMM should start an adjustment process so that the active resources meet the new requirements of workloads and applications. There are different solutions to alleviate the performance problems in the system, including changing resource configurations, adding new computing units, or replacing the VMs. In this section, we explain these techniques in detail, focusing on coarse-grained methods at the resource and VM level, as shown in Figure 5. Table 1 presents an overview of some of the advantages and limitations of these methods. As explained in Section 4.1, CSPs are included as contributing users for all types of decision-making to highlight the importance of considering the application side of SLAs regarding related objectives, such as budget and security constraints, during the decision-making process.

8.1 Application-Level Methods

These actions are directly applied to the application and its environment; therefore, they are not specific to cloud systems. This approach targets degradation in performance of the application or operating system caused by internal problems, such as data corruption, numerical error

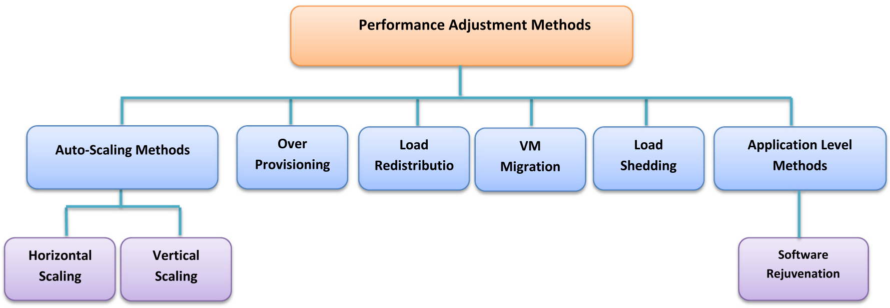

Fig. 5. Cloud performance adjustment techniques.

Table 1. An Overview of Performance Adjustment Methods

Method	Contributing users	Advantage	Limitations
Application Level	CSP	No dependency on CRPs, No cost from newly added resources.	This approach may not solve nonapplication-originated problems and is not applicable for all types of applications. It may cause short-term performance degeneration caused by nonresponsive components during deactivation time.
Overprovisioning	CRP, CSP	Increases reliability for highly dynamic workloads.	Resource wastages, Higher costs.
Horizontal Scaling	CRP, CSP	No need for special hardware support.	Increases all resource types without considering the source of the problem (homogeneous VM types). Start-up delays.
Vertical Scaling	CRP, CSP	Custom resource adjustment (cost-effectiveness), No need for new software instances (license problems)	Requires special hardware supports.
Load Distribution	CRP, CSP	No extra resources (cost-effectiveness)	Limited applicability (especially for nonstateless systems).
Load Shedding	CSP	No extra resources (cost-effectiveness)	Performance degradation by ignoring some requests.
VM Migration	CRP, CSP	No extra resources (cost-effectiveness), Reducing cold spots (energy saving)	Short-term performance degeneration during movement, security/privacy issues.

accumulation, or exhaustion of operating system resources [112]. Software rejuvenation is a possible corrective action in these types of problems, which tries to identify the problematic application or system component, clean its internal state, and restarts the components. Indeed, simple application or VM restarts, which are used by system administrators as the first reaction to many performance degradations, are preliminary cases of applying this approach. Dynamic component activation can also be triggered at this level, when the deactivation of optional components helps the system to manage higher loads or save energy according to the SLA requirements [119]. Since the application-level methods directly affect the functionality of the application, CSPs or application owners are actively involved in making these decisions.

8.2 Overprovisioning

In order to efficiently use the capacity of cloud-offered resources, it is vital to have a proper estimation of required resources that keep the performance and QoS of applications at an acceptable level. However, resource estimation is a complex problem, especially for dynamic workloads with time-dependent fluctuations. A traditional solution to handling the uncertainty in resource demands of applications is to provide enough resources to process the maximum expected workload in the system. This solution guarantees a stable application performance in the presence of workload fluctuations. However, peaks in workloads are usually transient events that do not last long and happen very rarely. This means that, most of the time, resources are underutilized and incur extra costs for CSPs. Considering the dynamic nature of applications, such as web workloads, overprovisioning is not avoidable in fault-tolerant resource management solutions. However, researchers are trying to find a proper trade-off between the level of fault tolerance of the system and overprovisioned resources to have more control over the functionality of the system [90].

8.3 Auto-Scaling Methods

Scaling is defined as increasing or decreasing the amount of resources (e.g., CPU, RAM, disk) for meeting SLA and performance standards. The scaling of resources can be done by changing the number of machines in the system or at the VM level by changing the configuration of one VM. In this section, we explain each of these approaches in more detail.

8.3.1 Horizontal Scaling. Horizontal scaling, which is the primary block of every RMM framework, helps to provide more resources by adding new VMs to the system [55, 120, 121]. The unit of changes in horizontal scaling is one VM. However, the newly added VMs can have a customized resource configuration (CPU, RAM, I/O) or preconfigured VMs can be launched by selecting one of the instance types offered by the provider. For example, Amazon offers a range of VM types with different levels of configurations from small to very large instances [5]. As a complementary option, Google also offers users the chance to define their requirements with more details by launching customized instances.

The time it takes for a new VM to be launched, also known as VM startup time, is highly important, especially for proactive RMMs. If startup times are longer than the predicted point for a possible breach of the SLA, the effect of the scaling process will be same as the reactive strategies with the added overhead of the data analysis module. Considering this, Mao and Humphrey [73] investigate possible influential factors, such as the type of instance, OS image size, or VM location for different providers. Their study provides researchers a basic understanding of the contributing factors that should be constant during experiments and also estimated average times to be taken into account in the simulation of the cloud environment [42].

8.3.2 Vertical Scaling. While horizontal scaling is a conventional strategy in the management of resources offered by providers, the advent of virtualization techniques has introduced new resource-level scaling opportunities, including elastic VMs. In vertical scaling, the VM elasticity feature is used to change existing VM configurations, virtual cores or RAM, on the fly to adapt to new requirements of the system [80, 121]. The online reconfiguration without turning the VM off is getting more attention, especially when time and cost factors are considered in RMM decisions. First, maintaining SLA objectives becomes challenging when there are sudden changes and spikes in the workload, and the RMM needs a quick solution to increase the resources in the system. In horizontal scaling, the time it takes to launch new VMs can be a bottleneck when a fast, effective solution is needed. On the other hand, new VMs require new software to be installed, which can lead to additional license costs. Elastic VMs can offer the required concepts and technologies to implement practicable strategies for these problems. Indeed, depending on the application, resource level scalings might be the best idea in the case of resource shortages. For example, a CPU-saturated system hosting a CPU-intensive application can be scaled by adding a new core to the VM without wasting memory, bandwidth, and other resources. Moreover, the same VM can continue the execution of existing requests without the need for interruption or transferring their execution profiles to a separate VM.

However, the elastic VMs need the support from both the guest VM OS and hypervisor. Therefore, owing to added complexity, many public providers do not offer this functionality. ProfitBricks [89] is one of the Infrastructure as a Service (IaaS) providers that supports the live vertical scaling of CPU cores, RAM, network interface card and hard disks. To have a better understanding of the extent of support for vertical scaling, one can refer to work done by [110], which studies this capability for some of the common hypervisors and guest OSs as well as OpenStack framework in the cloud environment.

8.4 Load Distribution

In a large scalable environment, we can find many replicas of one service, such as databases or application server components. The problem of distributing application requests among existing replicas of one component is a functionality provided by load balancers. A load balancer such as haProxy can be configured with a weight for each replica and distribute the loads according to this configuration among multiple VMs. Amazon Web Service Elastic Load Balancer (AWS ELB) [5] offers a simple round-robin strategy that assigns an equal weight to all active VMs and selects them from a list in the order of appearance. Therefore, in a round-robin–based balanced environment, the utilization of all VMs is affected similarly. Noticeably, this approach does not provide a cost- or energy-efficient solution, especially in an underutilized environment, where a small fraction of the provided computing and storage resources are enough to maintain the required SLA. A more efficient version of this approach assigns weights based on the specific server characteristics, such as the load of the server, which helps to send new requests to least used servers first [108]. Grozev and Buyya [42] follow this approach to consolidate web requests in a few servers without violating the QoS. The proposed method monitors CPU and RAM utilizations of the servers along with the availability of the network buffer capacity to assign new requests. Soni and Kalra [103] prioritize the servers based on their hardware characteristics, distributing loads based on the computation power and availability of the machines. These approaches help the system to balance the workload among existing resources to keep performance at an acceptable level. However, load distribution is task-level resource management, which does not control the amount of resources in the system. Therefore, they should be integrated with other scaling methods that help to maintain enough resources to handle the existing workload of the application.

8.5 Load Shedding

Load shedding is a type of self-healing approach primarily used in electrical power management to handle high loads in the system. The idea behind this approach is to maintain the availability of the system by sacrificing some QoS in the presence of faults. In datastream mining, load shedding refers to mechanisms that try to find when and how much data can be discarded so that the system can continue working with an acceptable degradation of performance [106]. This approach is not effective for the applications with highly dynamic workloads and strict QoS requirements [60]. However, one can consider request admission and resource reservation policies in the system in terms of the SLA agreements so that the extra requests that cannot be handled by available resources will be rejected, saving time and money for both users and service providers.

Table 2. Comparison of Data-Aware Performance Management Approaches in Large-Scale Systems

Work	Data Level	Learning Approach (ML, Machine Learning; RL, Reinforcement Learning)	Anomaly Aware	Anomaly Problem	Cause Inference Level	Proactive	Resource Adjustment Techniques (H, Horizontal; V, Vertical)	Module
[30]	System	ML	X	_	_	✓	Load balancing	Data
[18]	System	ML, Statistical	X	_	_	X	_	Data
[121]	System	Threshold, ML	X	_	_	✓	V, H	Data, plan
[55]	System, Application	Threshold, Statistical	X	_	_	✓	H	Data, plan
[25]	System, Application	ML	✓		_	X	_	Data
[10]	Network	Statistical	✓	Network	_	X	_	Data
[82]	System, Structure	ML	✓	Software bug, resource bottleneck	Component, Metrics	X	_	Data
[45]	System	ML	✓	Resource bottleneck	Type of anomaly	X	_	Data
[43]	System, Application	ML	✓	Resource bottleneck	Type of anomaly	✓	_	Data
[23]	System	ML	✓	Deadlock, starvation, livelock	Type of anomaly	✓	_	Data
[26]	System	ML	✓	CPU/Memory leak, network hog	Metrics	✓	_	Data
[105]	System	ML	✓	Resource bottleneck	Metrics	✓	V, Migration	Data, plan
[68]	System	RL, ML	X	_	_	✓	H	Data, plan
[47]	Network	Signature, ML	✓	Network	Type of anomaly	✓	_	Data
[116]	System, Application	ML, Statistical	✓	Resource bottleneck	Metrics	X	V, Migration	Data, plan
[118]	System, Application	ML, Statistical	✓	Resource bottleneck	Metrics	X	_	Data
[83]	System	Statistical	✓	Resource bottleneck, offload bug, load balancing bug	Component	X	_	Data
[28]	System	Statistical	✓	Resource bottleneck, software bugs, multi-tenancy problem, network packet loss, deadlock	Type of anomaly (external vs. internal)	X	_	Data
[27]	System	ML, Signature	✓	Software bugs	Code level	X	_	Data
[109]	Application	Statistical	✓	Software bugs	Code level	X	_	Data
[124]	System	ML		Resource bottleneck, database abuse	Target thread/ process	X	_	Data
[56]	System	ML, Statistical	X	_	_	✓	_	Data
[96]	Network	Statistical	✓	DDos attacks	_	X	_	Data

(Continued)

Table 2. Continued

Work	Data Level	Learning Approach (ML, Machine Learning; RL, Reinforcement Learning)	Anomaly Aware	Anomaly Problem	Cause Inference Level	Proactive	Resource Adjustment Techniques (H, Horizontal; V, Vertical)	Module
[24]	System, Application	Statistical	✓	Resource bottleneck	Metric	X	_	Data
[78]	System, Application	Signature	✓	_	_	X	_	Data
[22]	System, Application	Statistical, ML, Signature	✓	Resource bottleneck, application update	_	X	_	Data
[14]	System, Application, Structure	Signature, Statistical	X	_	_	X	_	Data
[97]	System, Application	Statistical, Signature, ML	✓	Load, software bugs	Metrics, Type of anomaly	X	V, H, Migration	Data, plan
[38]	System	Threshold	_	_	_	✓	H, Migrations	Data, plan
[46]	System, Application	Threshold	X	_	_	X	H, V	Data, plan
[8]	System, Application	Control Theory	X	_	_	P	H	Data, plan
[49]	System	ML	✓	Resource bottleneck	_	X	_	Data
[41]	System	Control Theory	✓	Load, hardware failure	_	X	H	Data, plan
[57]	System, Application	Control Theory	X	_	_	X	V	Data + plan
[4]	System, Application	Control Theory	X	_	_	X	H	Data + plan
[115]	System	Statistical	✓	Resource bottleneck, software bugs	_	X	_	Data
[52]	System	Statistical	✓	Resource bottleneck	_	X	_	Data
[86]	Application, System	Threshold, Statistical	✓	_	Metrics	X	_	Data
[75]	System, Application	Threshold, Statistical	✓	Resource bottleneck	Metrics	X	_	Data
[114]	System, Application	ML, Statistical	X	_	_	✓	H	Data, plan
[44]	Network	Statistical	✓	Port scan	Packet information	X	_	Data
[17]	System	Statistical	✓	DoS Attack	_	X	_	Data
[3]	System, Application	ML	X	_	_	✓	_	Data
[64]	System	ML	✓	Resource bottleneck	_	✓	VM Placement	Data, plan
[79]	System	ML, threshold	✓	Resource bottleneck	Component, metrics	✓	H, V	Data, plan

(Continued)

Table 2. Continued

Work	Data Level	Learning Approach (ML, Machine Learning; RL, Reinforcement Learning)	Anomaly Aware	Anomaly Problem	Cause Inference Level	Proactive	Resource Adjustment Techniques (H, Horizontal; V, Vertical)	Module
[34]	System, Application	Threshold, RL	X	_	_	X	H	Data, plan
[122]	System	RL, Statistical	X	_	_	✓	V	Data, plan
[33]	System	RL	X	_	_	X	Migration	Data, plan
[7]	System, Application	RL	X	_	_	X	H	Data, plan
[90]	System	Signature	✓	Resource shortage	_	X	H, Over-provisioning	Data, plan
[98]	System, Application	Signature, ML	✓	Resource bottleneck	_	✓	V, Migration	Data, plan
[80]	System	Threshold	✓	Resource bottleneck	_	X	V, Migration	Data, plan
[117]	System, Application	Threshold, Statistical	✓	Resource bottleneck	_	✓	Migration	Data, plan
[102]	System	Statistical	X	_	_	✓	Migration	Data, plan
[125]	System	Rule	✓	Network-related application bugs	Target system calls	X	_	Data
[5]	System	Threshold	X	_	_	X	H	Data, plan

8.6 VM Migration

The scalability feature of cloud systems is highly dependent on virtualization technology that enables multiple applications to reside on one PM by sharing available resources. VMs are preferably not dependent on one machine and can be moved among different hosts when needed. This capability brings new opportunities for improving the use of cloud applications by finding proper placement of VMs in the system [117]. Migration can be done to fulfill different objectives. Migrating VMs from underutilized hosts and consolidating them in few hosts enable the system to shut down unused hosts, saving costs and energy. Duggan et al. [33] investigate this problem while considering available bandwidth as a factor to determine the best time for migrations. An overutilized host, where the guest VMs are consuming all of the available resources offered by the host, is another target that can benefit from migration. In this case, the application on the overloaded host may experience performance degeneration, as there are requests that cannot be handled in time owing to the saturated resource and long waiting queues. Accordingly, Sommer et al. [102] propose a prediction-based migration strategy to find the overloaded hosts and trigger migration procedure to move some of the VMs to the existing underloaded hosts. A combination of multiple forecasting methods is employed to predict the future resource consumptions of the VMs and identify the possible overloaded hosts based on the predicted values.

Migration is also a complementary procedure when the host cannot fulfill requests to increase the resource capacity required for vertical scaling actions. PREPARE, proposed in [105], uses this strategy to correct the performance problems caused by internal faults or load anomalies. Live migration is implemented when vertical scaling action is ineffective or not possible owing to the lack of resources on the host.

9 GAP ANALYSIS AND FUTURE DIRECTIONS

Based on the analysis of different aspects of performance-aware resource management in the cloud, we propose a list of potential research areas and future directions.

9.1 VM Elasticity Analysis

Vertical elasticity is one of the rather new functionalities introduced for cloud resource scaling that is not yet prevalent compared with horizontal scaling. Technical limitations to support VM elasticity along the higher complexity of fine-grained scaling (CPU and RAM level compared to VM level) management in data centers makes this approach limited in practice; most known public providers do not offer this service. Therefore, a more detailed study of the characteristics of vertical elasticity, especially time sensitivity analysis for different resource types, OS, and hypervisors, is required to help in the design of accurate solutions in the area of proactive resource management.

9.2 Adaptable Learning in the Cloud

The effectiveness of many advanced machine-learning methods is highly dependent on preconfigurations that define a proper threshold or parameter values, usually based on the characteristics of data and applications. There are many works targeting the problem of parameter configuration of learning methods in the field of data prediction and anomaly detection. However, prediction and control of system behavior is becoming more complex with the growth of highly dynamic environments such as the cloud. Considering these challenges, robust statistics metrics and model/assumption-free frameworks are among techniques that require more attention to have more realistic solutions. These mechanisms help to increase the flexibility and robustness of models, which is highly important for improving the adaptation of systems to a variety of applications and datasets. Dynamic tuning of model parameters in accordance with recent changes and feedback of systems are interesting approaches for developing robust solutions to be applicable for different workload patterns. Self-adaptable systems, which are discussed in Section 6.2.3, and gradual learning frameworks, such as reinforcement learning from Section 6.2.6, are among approaches that can be effectively combined with data-learning techniques to enable a mechanism of live interaction with the environment to facilitate adaptable resource management decision-making. However, a common point of failure for these mechanisms is exploration capability in terms of dealing with large state/action tables that affect the accuracy and convergence rate of the solutions. While new approaches such as adaptive partitioning of state space [70] have been introduced that address these types of problems, the applicability and efficiency of proposed solutions should be investigated further.

9.3 Anomaly Cause Inference

Anomaly cause inference has been studied as part of the data analysis module to help the RMM better decide on selecting corrective actions. However, existing works in this area mostly are coarse-grained, giving suggestions about the bottleneck metrics based on resource-level information. Moreover, the contribution of knowledge from cause identification in the process of RMM planning and resource management has not been fully investigated. A more autonomic and integrated cause identification process that has continuous interaction with planning modules to get timely feedback on the quality of information is a challenge for future researchers.

9.4 Realistic View for Anomaly-Aware Auto-Scaling

In this article, we have tried to cover both areas of data analysis, including prediction and anomaly detection as well as resource adjustment planning for performance management. The significance of having performance alerts ahead of time for planning modules to have enough time for preparation and triggering actions makes real-time sensitivity analysis a challenge that can be investigated only in a real environment. There are several technical limitations that can affect the performance of both parts, which can be detected only during a real implementation of the complete framework, including all interactions among distributed components.

9.5 Application-Dependent Detection Accuracy Trade-offs

In data analysis, part of the investigated problem, Area Under the Curve (AUC), commonly is used to show the performance of anomaly detection algorithms in detecting anomaly points. However, in the area of cloud performance analysis, the number of normal instances in the collected data usually is much higher than anomaly points. The lack of balance in the number of instances for different classes raises the question of whether the AUC metric is biased by true negative points. We believe that presenting the performance results by comparing both metrics AUC and Precision-Recall Area Under the Curve (PRAUC), which demonstrate the functionality of the algorithms from different points of view, is an important part of the anomaly detection problems in this area. This is a point that can be very important for some applications, which require complex recovery points in the case of true anomaly events. For example, for prevention mechanisms that target disk-related problems with expensive mitigation actions, a solution with higher precision and a minimum of false alarms may be preferred. Referring to our survey, this is an interesting point that is highly neglected and requires a detailed analysis of the effectiveness of proposed anomaly detection methods considering service owner preferences.

10 CONCLUSIONS

With the emergence of cloud computing and the power of data analytic methods, new opportunities for improvement of performance-aware distributed resource management mechanisms has been introduced. The analysis of collected data from measurable system attributes gives valuable information about the health state of the system and performance of the hosted applications.

In this article, we investigate different approaches in the performance management of the cloud environment. Identifying the major limitations and considerations in the selection of the best strategy for proper resource configuration highlights the need for more complex and automated procedures to handle the dynamism of the environment. We have proposed a taxonomy of problems focusing on the value of the data as a source of knowledge for resource management decision-making and presented a survey of the existing works in the field of performance-aware cloud resource management accordingly. The listed categories in the taxonomy are defined based on the characteristics of the reviewed works, including presented architecture, the granularity of collected performance data, targeted performance problems and the types of resource management actions. Based on the reviewed works, a list of observed gaps and possible directions is suggested that can give new insights and starting points for future researchers.

REFERENCES

[1] Giuseppe Aceto, Alessio Botta, Walter de Donato, and Antonio Pescapè. 2013. Cloud monitoring: A survey. *Computer Networks* 57, 9, 2093–2115.

[2] B. Addis, D. Ardagna, B. Panicucci, M. S. Squillante, and L. Zhang. 2013. A hierarchical approach for the resource management of very large cloud platforms. *IEEE Transactions on Dependable and Secure Computing* 10, 5, 253–272.

[3] Samuel Ajila and Akindele Bankole. 2016. Using machine learning algorithms for cloud client prediction models in a web VM resource provisioning environment. *Transactions on Machine Learning and Artificial Intelligence* 4, 1, 28.

[4] Ahmad Al-Shishtawy and Vladimir Vlassov. 2013. ElastMan: Elasticity manager for elastic key-value stores in the cloud. In *Proceedings of the ACM Cloud and Autonomic Computing Conference (CAC'13)*. ACM, Article 7, 10 pages.

[5] Amazon. 2019. Amazon. Retrieved July 29, 2019 from https://aws.amazon.com/.

[6] Ali Anwar, Anca Sailer, Andrzej Kochut, and Ali R. Butt. 2015. Anatomy of cloud monitoring and metering: A case study and open problems. In *Proceedings of the 6th Asia-Pacific Workshop on Systems.* ACM, New York, NY, Article 6, 7 pages.

[7] Hamid Arabnejad, Claus Pahl, Pooyan Jamshidi, and Giovani Estrada. 2017. A comparison of reinforcement learning techniques for fuzzy cloud auto-scaling. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.* IEEE Press, Piscataway, NJ, 64–73.

[8] D. Ardagna, M. Ciavotta, and R. Lancellotti. 2014. A receding horizon approach for the runtime management of IaaS cloud systems. In *Proceedings of the 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing.* IEEE, 445–452.

[9] D. Ardagna, M. Ciavotta, R. Lancellotti, and M. Guerriero. 2018. A hierarchical receding horizon algorithm for QoS-driven control of multi-IaaS applications. *IEEE Transactions on Cloud Computing*, 1–1.

[10] Ayesha Binte Ashfaq, Sajjad Rizvi, Mobin Javed, Syed Ali Khayam, Muhammad Qasim Ali, and Ehab Al-Shaer. 2014. Information theoretic feature space slicing for statistical anomaly detection. *Journal of Network and Computer Applications* 41, 473–487.

[11] Mohammad Sadegh Aslanpour, Mostafa Ghobaei-Arani, and Adel Nadjaran Toosi. 2017. Auto-scaling web applications in clouds: A cost-aware approach. *Journal of Network and Computer Applications* 95, 26–41.

[12] Microsoft Azure. 2019. Azure. Retrieved July 29, 2019 from https://azure.microsoft.com.

[13] D. Basu, X. Wang, Y. Hong, H. Chen, and S. Bressan. 2017. Learn-as-you-go with Megh: Efficient live migration of virtual machines. In *IEEE 37th International Conference on Distributed Computing Systems (ICDCS'17).* IEEE, 2608–2609.

[14] Andreas Brunnert and Helmut Krcmar. 2017. Continuous performance evaluation and capacity planning using resource profiles for enterprise applications. *Journal of Systems and Software* 123, Supplement C, 239–262.

[15] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. 2009. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems* 25, 6, 599–616.

[16] Jeffrey P. Buzen and Annie W. Shum. 1995. MASF - multivariate adaptive statistical filtering. In *International CMG Conference.* Computer Measurement Group, 1–10.

[17] J. Cao, B. Yu, F. Dong, X. Zhu, and S. Xu. 2014. Entropy-based denial of service attack detection in cloud data center. In *Proceedings of the 2nd International Conference on Advanced Cloud and Big Data.* IEEE, 201–207.

[18] Katja Cetinski and Matjaz B Juric. 2015. AME-WPC: Advanced model for efficient workload prediction in the cloud. *Journal of Network and Computer Applications* 55, 191–201.

[19] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Computing Surveys* 41, 3, Article 15 (2009), 15 pages.

[20] Deyan Chen and Hong Zhao. 2012. Data security and privacy protection issues in cloud computing. In *Proceedings of the 2012 International Conference on Computer Science and Electronics Engineering - Volume 01 (ICCSEE'12).* IEEE, 647–651.

[21] Tao Chen, Rami Bahsoon, and Xin Yao. 2018. A survey and taxonomy of self-aware and self-adaptive cloud autoscaling systems. *Computing Surveys* 51, 3, 61:1–61:40.

[22] Ludmila Cherkasova, Kivanc Ozonat, Ningfang Mi, Julie Symons, and Evgenia Smirni. 2009. Automated anomaly detection and performance modeling of enterprise applications. *ACM Transactions on Computer Systems* 27, 3, Article 6, 32 pages.

[23] J. Á. Cid-Fuentes, C. Szabo, and K. Falkner. 2015. Online behavior identification in distributed systems. In *34th IEEE Symposium on Reliable Distributed Systems.* Montreal, Quebec, Canada, IEEE, 202–211.

[24] Ira Cohen, Moises Goldszmidt, Terence Kelly, Julie Symons, and Jeffrey S. Chase. 2004. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation (OSDI'04).* USENIX Association, 16–16.

[25] Carlos Augusto Cunha and Luis Moura e Silva. 2012. Separating performance anomalies from workload-explained failures in streaming servers. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID'12).* IEEE Computer Society, Washington, DC, 292–299.

[26] Daniel Joseph Dean, Hiep Nguyen, and Xiaohui Gu. 2012. UBL: Unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems. In *Proceedings of the 9th International Conference on Autonomic Computing.* ACM, New York, NY, 191–200.

[27] Daniel J. Dean, Hiep Nguyen, Xiaohui Gu, Hui Zhang, Junghwan Rhee, Nipun Arora, and Geoff Jiang. 2014. PerfScope: Practical online server performance bug inference in production cloud computing infrastructures. In *Proceedings of the ACM Symposium on Cloud Computing (SOCC'14).* ACM, New York, NY, Article 8, 13 pages.

[28] Daniel J. Dean, Hiep Nguyen, Peipei Wang, and Xiaohui Gu. 2014. PerfCompass: Toward runtime performance anomaly fault localization for infrastructure-as-a-service clouds. In *Proceedings of the 6th USENIX Conference on Hot Topics in Cloud Computing (HotCloud'14)*. USENIX Association, Berkeley, CA, 16–16.

[29] Henri Maxime Demoulin, Isaac Pedisich, Linh Thi Xuan Phan, and Boon Thau Loo. 2018. Automated detection and mitigation of application-level asymmetric DoS attacks. In *Proceedings of the Afternoon Workshop on Self-Driving Networks (SelfDN'18)*. ACM, New York, NY, 36–42.

[30] Sheng Di, Derrick Kondo, and Walfredo Cirne. 2014. Google hostload prediction based on Bayesian model with optimized feature combination. *Journal of Parallel and Distributed Computing* 74, 1, 1820–1832.

[31] Thanh Do, Mingzhe Hao, Tanakorn Leesatapornwongsa, Tiratat Patana-anake, and Haryadi S. Gunawi. 2013. Limplock: Understanding the impact of limpware on scale-out cloud systems. In *Proceedings of the 4th Annual Symposium on Cloud Computing (SOCC'13)*. ACM, New York, NY, Article 14, 14 pages.

[32] Shridhar G. Domanal and G. Ram Mohana Reddy. 2018. An efficient cost optimized scheduling for spot instances in heterogeneous cloud environment. *Future Generation Computer Systems* 84, 11–21.

[33] Martin Duggan, Jim Duggan, Enda Howley, and Enda Barrett. 2017. A reinforcement learning approach for the scheduling of live migration from under utilised hosts. *Memetic Computing* 9, 4 (2017), 283–293.

[34] Xavier Dutreilh, Aurélien Moreau, Jacques Malenfant, Nicolas Rivierre, and Isis Truck. 2010. From data center resource allocation to control theory and back. In *Proceedings of the IEEE 3rd International Conference on Cloud Computing (CLOUD'10)*. IEEE Computer Society, Washington, DC, 410–417.

[35] W. Fang, X. Yao, X. Zhao, J. Yin, and N. Xiong. 2018. A stochastic control approach to maximize profit on service provisioning for mobile cloudlet platforms. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48, 4, 522–534.

[36] Massimo Ficco, Christian Esposito, Francesco Palmieri, and Aniello Castiglione. 2018. A coral-reefs and Game Theory-based approach for optimizing elastic cloud resource allocation. *Future Generation Computer Systems* 78, 343–352.

[37] Ganglia. 2019. Ganglia. Retrieved July 29, 2019 from http://ganglia.sourceforge.net/.

[38] Daniel Gmach, Jerry Rolia, Ludmila Cherkasova, and Alfons Kemper. 2009. Resource pool management: Reactive versus proactive or let's be friends. *Computer Networks* 53, 17, 2905–2922.

[39] S. Gong, B. Yin, W. Zhu, and K. Cai. 2017. Adaptive resource allocation of multiple servers for service-based systems in cloud computing. In *IEEE 41st Annual Computer Software and Applications Conference*, Vol. 2. IEEE, 603–608.

[40] Google. 2019. Google Cloud. Retrieved July 29, 2019 from https://cloud.google.com.

[41] D. Grimaldi, V. Persico, A. Pescape, A. Salvi, and S. Santini. 2015. A feedback-control approach for resource management in public clouds. In *IEEE Global Communications Conference*. IEEE, 1–7.

[42] Nikolay Grozev and Rajkumar Buyya. 2014. Multi-cloud provisioning and load distribution for three-tier applications. *ACM Transactions on Autonomous and Adaptive Systems* 9, 3, Article 13 (Oct. 2014), 21 pages.

[43] Xiaohui Gu and Haixun Wang. 2009. Online anomaly prediction for robust cluster systems. In *Proceedings of the 25th IEEE International Conference on Data Engineering, 2009*. IEEE, Shanghai, China, 1000–1011.

[44] Yu Gu, Andrew McCallum, and Don Towsley. 2005. Detecting anomalies in network traffic using maximum entropy estimation. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC'05)*. USENIX Association, Berkeley, CA, 32–32.

[45] Qiang Guan and Song Fu. 2013. Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures. In *Proceedings of the 32nd IEEE International Symposium on Reliable Distributed Systems (SRDS'13)*. IEEE Computer Society, Braga, Portugal, 205–214.

[46] Rui Han, Li Guo, Moustafa M. Ghanem, and Yike Guo. 2012. Lightweight resource scaling for cloud applications. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID'12)*. IEEE Computer Society, Washington, DC, 644–651.

[47] Mohammad Amin Hatef, Vahid Shaker, Mohammad Reza Jabbarpour, Jason Jung, and Houman Zarrabi. 2018. HIDCC: A hybrid intrusion detection approach in cloud computing. *Concurrency and Computation: Practice and Experience* 30, 3, e4171.

[48] Henry Hoffman. 2013. *SEEC: A Framework for Self-aware Management of Goals and Constraints in Computing Systems (Power-aware Computing, Accuracy-aware Computing, Adaptive Computing, Autonomic Computing)*. Ph.D. Dissertation. Cambridge, MA.

[49] Bin Hong, Fuyang Peng, Bo Deng, Yazhou Hu, and Dongxia Wang. 2015. DAC-Hmm: Detecting anomaly in cloud systems with hidden Markov models. *Concurrency and Computation: Practice and Experience* 27, 18, 5749–5764.

[50] Markus C. Huebscher and Julie A. McCann. 2008. A survey of autonomic computing–degrees, models, and applications. *ACM Computing Surveys* 40, 3, 7:1–7:28.

[51] Olumuyiwa Ibidunmoye, Francisco Hernández-Rodriguez, and Erik Elmroth. 2015. Performance anomaly detection and bottleneck identification. *Computing Surveys* 48, 1, 4:1–4:35.

[52] O. Ibidunmoye, E. B. Lakew, and E. Elmroth. 2017. A black-box approach for detecting systems anomalies in virtualized environments. In *International Conference on Cloud and Autonomic Computing (ICCAC'17)*. IEEE, 22–33.

[53] IBM. 2006. An architectural blueprint for autonomic computing. *IBM White Paper* 31, 1–6.

[54] Emilio Incerto, Mirco Tribastone, and Catia Trubiani. 2017. Software performance self-adaptation through efficient model predictive control. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE'17)*. IEEE Press, 485–496.

[55] Waheed Iqbal, Matthew N. Dailey, David Carrera, and Paul Janecek. 2011. Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Future Generation Computer Systems* 27, 6, 871–879.

[56] Sadeka Islam, Jacky Keung, Kevin Lee, and Anna Liu. 2012. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems* 28, 1, 155–162.

[57] Evangelia Kalyvianaki, Themistoklis Charalambous, and Steven Hand. 2009. Self-adaptive and self-configured CPU resource provisioning for virtualized servers using Kalman filters. In *Proceedings of the 6th International Conference on Autonomic Computing (ICAC'09)*. ACM, New York, NY, 117–126.

[58] Pankaj Deep Kaur and Inderveer Chana. 2014. A resource elasticity framework for QoS-aware execution of cloud applications. *Future Generation Computer Systems* 37, 14–25.

[59] Md Anit Khan, Andrew Paplinski, Abdul Malik Khan, Manzur Murshed, and Rajkumar Buyya. 2018. *Dynamic Virtual Machine Consolidation Algorithms for Energy-Efficient Cloud Resource Management: A Review*. Springer International Publishing, 135–165.

[60] W. Kleiminger, E. Kalyvianaki, and P. Pietzuch. 2011. Balancing load in stream processing with the cloud. In *IEEE 27th International Conference on Data Engineering Workshops*. IEEE, 16–21.

[61] Wook Hyun Kwon and Soo Hee Han. 2006. *Receding Horizon Control: Model Predictive Control for State Models*. Springer Science & Business Media.

[62] P. Lama, Y. Guo, C. Jiang, and X. Zhou. 2016. Autonomic performance and power control for co-located web applications in virtualized datacenters. *IEEE Transactions on Parallel and Distributed Systems* 27, 5, 1289–1302.

[63] X. Li, M. A. Salehi, M. Bayoumi, and R. Buyya. 2016. CVSS: A cost-efficient and QoS-aware video streaming using cloud services. In *16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'16)*. IEEE, 106–115.

[64] Xi Li, A. Ventresque, J. O. Iglesias, and J. Murphy. 2015. Scalable correlation-aware virtual machine consolidation using two-phase clustering. In *Proceedings of the International Conference on High Performance Computing Simulation (HPCS'15)*. IEEE, 237–245.

[65] Wei-Chao Lin, Shih-Wen Ke, and Chih-Fong Tsai. 2015. CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-Based Systems* 78, 13–21.

[66] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *Proceedings of the 8th IEEE International Conference on Data Mining*. IEEE, 413–422.

[67] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2012. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data* 6, 1, Article 3, 39 pages.

[68] N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, Q. Qiu, J. Tang, and Y. Wang. 2017. A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In *Proceedings of the 37th IEEE International Conference on Distributed Computing Systems (ICDCS'17)*. IEEE, 372–382.

[69] Y. Liu, X. Ruan, S. Cai, R. Li, and H. He. 2018. An optimized VM allocation strategy to make a secure and energy-efficient cloud against co-residence attack. In *International Conference on Computing, Networking and Communications (ICNC'18)*. IEEE, 349–353.

[70] Konstantinos Lolos, Ioannis Konstantinou, Verena Kantere, and Nectarios Koziris. 2017. Adaptive state space partitioning of Markov decision processes for elastic resource management. In *IEEE 33rd International Conference on Data Engineering*. IEEE, 191–194.

[71] L. Lu, J. Yu, Y. Zhu, and M. Li. 2018. A double auction mechanism to bridge users' task requirements and providers' resources in two-sided cloud markets. *IEEE Transactions on Parallel and Distributed Systems* 29, 4, 720–733.

[72] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. 2018. Fog computing: A taxonomy, survey and future directions. In *Internet of Everything*. Springer, 103–130.

[73] Ming Mao and Marty Humphrey. 2012. A performance study on the VM startup time in the cloud. In *Proceedings of the 5th IEEE International Conference on Cloud Computing (CLOUD'12)*. IEEE Computer Society, 423–430.

[74] Toni Mastelic, Ariel Oleksiak, Holger Claussen, Ivona Brandic, Jean-Marc Pierson, and Athanasios V. Vasilakos. 2014. Cloud computing: Survey on energy efficiency. *Computing Surveys* 47, 2, Article 33, 36 pages.

[75] Tatsuma Matsuki and Naoki Matsuoka. 2016. A resource contention analysis framework for diagnosis of application performance anomalies in consolidated cloud environments. In *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering (ICPE'16)*. ACM, New York, NY, 173–184.

[76] M. Melo, J. Araujo, R. Matos, J. Menezes, and P. Maciel. 2013. Comparative analysis of migration-based rejuvenation schedules on cloud availability. In *IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 4110–4115.

[77] Gabriele Mencagli, Marco Vanneschi, and Emanuele Vespa. 2013. Reconfiguration stability of adaptive distributed parallel applications through a cooperative predictive control approach. In *Euro-Par 2013 Parallel Processing*, Felix Wolf, Bernd Mohr, and Dieter an Mey (Eds.). Springer, Berlin, 329–340.

[78] N. Mi, L. Cherkasova, K. Ozonat, J. Symons, and E. Smirni. 2008. Analysis of application performance and its change via representative application signatures. In *Proceedings of the IEEE Network Operations and Management Symposium*. IEEE, 216–223.

[79] Sara Kardani Moghaddam, Rajkumar Buyya, and Kotagiri Ramamohanarao. 2019. ACAS: An anomaly-based cause aware auto-scaling framework for clouds. *Journal of Parallel and Distributed Computing* 126 (2019), 107–120.

[80] Germán Moltó, Miguel Caballer, and Carlos de Alfonso. 2016. Automatic memory-based vertical elasticity and oversubscription on cloud platforms. *Future Generation Computer Systems* 56, C, 1–10.

[81] M. A. Mukwevho and T. Celik. 2018. Toward a smart cloud: A review of fault-tolerance methods in cloud systems. *IEEE Transactions on Services Computing*.

[82] Hiep Nguyen, Zhiming Shen, Yongmin Tan, and Xiaohui Gu. 2013. FChain: Toward black-box online fault localization for cloud systems. In *Proceedings of the 33rd IEEE International Conference on Distributed Computing Systems*. IEEE, 21–30.

[83] Hiep Nguyen, Yongmin Tan, and Xiaohui Gu. 2011. PAL: Propagation-aware anomaly localization for cloud hosted distributed applications. In *Proceedings of the Managing Large-scale Systems via the Analysis of System Logs and the Application of Machine Learning Techniques (SLAML'11)*. ACM, New York, NY, Article 1, 8 pages.

[84] Guansong Pang, Longbing Cao, Ling Chen, Defu Lian, and Huan Liu. 2018. Sparse modeling-based sequential ensemble learning for effective outlier detection in high-dimensional numeric data. In *32nd AAAI Conference on Artificial Intelligence*. AAAI, 3892–3899.

[85] Debdeep Paul, Wen-De Zhong, and Sanjay K. Bose. 2016. Energy efficiency aware load distribution and electricity cost volatility control for cloud service providers. *Journal of Network and Computer Applications* 59, C, 185–197.

[86] M. Peiris, J. H. Hill, J. Thelin, S. Bykov, G. Kliot, and C. Konig. 2014. PAD: Performance anomaly detection in multiserver distributed systems. In *Proceedings of the 7th IEEE International Conference on Cloud Computing*. IEEE, 769–776.

[87] V. Persico, D. Grimaldi, A. Pescapè, A. Salvi, and S. Santini. 2017. A fuzzy approach based on heterogeneous metrics for scaling out public clouds. *IEEE Transactions on Parallel and Distributed Systems* 28, 8, 2117–2130.

[88] Giuseppe Procaccianti, Héctor Fernández, and Patricia Lago. 2016. Empirical evaluation of two best practices for energy-efficient software development. *Journal of Systems and Software* 117, C, 185–198.

[89] ProfitBricks. 2019. ProfitBricks. Retrieved July 29, 2019 from https://www.profitbricks.com/help/Live_Vertical_Scaling.

[90] Chenhao Qu, Rodrigo N. Calheiros, and Rajkumar Buyya. 2016. A reliable and cost-efficient auto-scaling system for web applications using heterogeneous spot instances. *Journal of Network and Computer Applications* 65, 167–180.

[91] Chenhao Qu, Rodrigo N. Calheiros, and Rajkumar Buyya. 2018. Auto-scaling web applications in clouds: A taxonomy and survey. *Computing Surveys* 51, 4, Article 73, 33 pages.

[92] J. Rao, X. Bu, C. Xu, and K. Wang. 2011. A distributed self-learning approach for elastic provisioning of virtualized cloud resources. In *IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*. IEEE, 45–54.

[93] Jia Rao, Xiangping Bu, Cheng-Zhong Xu, Leyi Wang, and George Yin. 2009. VCONF: A reinforcement learning approach to virtual machines auto-configuration. In *Proceedings of the 6th International Conference on Autonomic Computing*. ACM, 137–146.

[94] N. Roy, A. Dubey, and A. Gokhale. 2011. Efficient autoscaling in the cloud using predictive models for workload forecasting. In *IEEE 4th International Conference on Cloud Computing*. IEEE, 500–507.

[95] X. Ruan and H. Chen. 2015. Performance-to-power ratio aware virtual machine (VM) allocation in energy-efficient clouds. In *IEEE International Conference on Cluster Computing*. IEEE, 264–273.

[96] P. Shamsolmoali and M. Zareapoor. 2014. Statistical-based filtering system against DDOS attacks in cloud computing. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*. IEEE, 1234–1239.

[97] Bikash Sharma, Praveen Jayachandran, Akshat Verma, and Chita R. Das. 2013. CloudPD: Problem determination and diagnosis in shared dynamic clouds. In *Proceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'13)*. IEEE Computer Society, Washington, DC, 1–12.

[98] Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, and John Wilkes. 2011. CloudScale: Elastic resource scaling for multi-tenant cloud systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 5:1–5:14.

[99] Hongtao Shi, Hongping Li, Dan Zhang, Chaqiu Cheng, and Wei Wu. 2017. Efficient and robust feature extraction and selection for traffic classification. *Computer Networks* 119 (2017), 1–16.

[100] Sukhpal Singh and Inderveer Chana. 2015. QoS-aware autonomic resource management in cloud computing: A systematic review. *Computing Surveys* 48, 3, Article 42, 46 pages.

[101] Vishakha Singh, Indrajeet Gupta, and Prasanta K. Jana. 2018. A novel cost-efficient approach for deadline-constrained workflow scheduling by dynamic provisioning of resources. *Future Generation Computer Systems* 79, 95–110.

[102] M. Sommer, M. Klink, S. Tomforde, and J. Höhner. 2016. Predictive load balancing in cloud computing environments based on ensemble forecasting. In *Proceedings of the IEEE International Conference on Autonomic Computing*. IEEE, 300–307.

[103] G. Soni and M. Kalra. 2014. A novel approach for load balancing in cloud data center. In *Proceedings of the IEEE International Advance Computing Conference (IACC'14)*. IEEE, 807–812.

[104] Hassan Jamil Syed, Abdullah Gani, Raja Wasim Ahmad, Muhammad Khurram Khan, and Abdelmuttlib Ibrahim Abdalla Ahmed. 2017. Cloud monitoring: A review, taxonomy, and open research issues. *Journal of Network and Computer Applications* 98, 11–26.

[105] Y. Tan, H. Nguyen, Z. Shen, X. Gu, C. Venkatramani, and D. Rajan. 2012. PREPARE: Predictive performance anomaly prevention for virtualized cloud systems. In *Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems*. Macau, China. IEEE, 285–294.

[106] Nesime Tatbul, Uğur Çetintemel, Stan Zdonik, Mitch Cherniack, and Michael Stonebraker. 2003. Load shedding in a data stream manager. In *Proceedings of the 29th International Conference on Very Large Data Bases*. Morgan Kaufmann, 309–320.

[107] Fei Teng, Lei Yu, Tianrui Li, Danting Deng, and Frédéric Magoulès. 2017. Energy efficiency of VM consolidation in IaaS clouds. *Journal of Supercomputing* 73, 2, 782–809.

[108] Yong Meng Teo and Rassul Ayani. 2001. Comparison of load balancing strategies on cluster-based web servers. *SIMULATION* 77, 5-6, 185–195.

[109] Joseph Tucek, Shan Lu, Chengdu Huang, Spiros Xanthos, and Yuanyuan Zhou. 2007. Triage: Diagnosing production run failures at the user's site. In *Proceedings of 21st ACM Symposium on Operating Systems Principles*. ACM, 131–144.

[110] Marian Turowski and Alexander Lenk. 2015. *Vertical Scaling Capability of OpenStack*. Springer, 351–362.

[111] R. Urgaonkar, U. C. Kozat, K. Igarashi, and M. J. Neely. 2010. Dynamic resource allocation and power management in virtualized data centers. In *IEEE Network Operations and Management Symposium (NOMS'10)*. IEEE, 479–486.

[112] Kalyanaraman Vaidyanathan and Kishor S. Trivedi. 2005. A comprehensive model for software rejuvenation. *Proceedings of the IEEE Transactions on Dependable and Secure Computing* 2, 2, 124–137.

[113] Kashi Venkatesh Vishwanath and Nachiappan Nagappan. 2010. Characterizing cloud computing hardware reliability. In *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC'10)*. ACM, New York, NY, 193–204.

[114] M. Wajahat, A. Gandhi, A. Karve, and A. Kochut. 2016. Using machine learning for black-box autoscaling. In *7th International Green and Sustainable Computing Conference (IGSC'16)*. IEEE, 1–8.

[115] C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan. 2011. Statistical techniques for online anomaly detection in data centers. In *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management (IM'11) and Workshops*. IEEE, 385–392.

[116] T. Wang, W. Zhang, C. Ye, J. Wei, H. Zhong, and T. Huang. 2016. FD4C: Automatic fault diagnosis framework for web applications in cloud computing. *IEEE Transactions on Systems, Man, & Cybernetics: Systems* 46, 1, 61–75.

[117] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. 2007. Black-box and gray-box strategies for virtual machine migration. In *Proceedings of the 4th USENIX Conference on Networked Systems Design and Implementation*. USENIX Association, Berkeley, CA, 17–17.

[118] Pengcheng Xiong, Calton Pu, Xiaoyun Zhu, and Rean Griffith. 2013. vPerfGuard: An automated model-driven framework for application performance diagnosis in consolidated cloud environments. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering (ICPE'13)*. ACM, New York, NY, 271–282.

[119] M. Xu, A. V. Dastjerdi, and R. Buyya. 2016. Energy efficient scheduling of cloud application components with brownout. *IEEE Transactions on Sustainable Computing* 1, 2, 40–53.

[120] Jingqi Yang, Chuanchang Liu, Yanlei Shang, Bo Cheng, Zexiang Mao, Chunhong Liu, Lisha Niu, and Junliang Chen. 2014. A cost-aware auto-scaling approach using the workload prediction in service clouds. *Information Systems Frontiers* 16, 1, 7–18.

[121] Jingqi Yang, Chuanchang Liu, Yanlei Shang, Zexiang Mao, and Junliang Chen. 2013. Workload predicting-based automatic scaling in service clouds. In *Proceedings of the IEEE 6th International Conference on Cloud Computing (CLOUD'13)*. IEEE Computer Society, Washington, DC, 810–815.

[122] Lenar Yazdanov and Christof Fetzer. 2013. VScaler: Autonomic virtual machine scaling. In *Proceedings of the IEEE 6th International Conference on Cloud Computing (CLOUD'13)*. IEEE Computer Society, 212–219.

[123] Shuai Yuan, Sanjukta Das, R. Ramesh, and Chunming Qiao. 2018. Service agreement trifecta: Backup resources, price and penalty in the availability-aware cloud. *Information Systems Research* 29, 4 (2018), 947–964.

[124] X. Zhang, F. Meng, P. Chen, and J. Xu. 2016. TaskInsight: A fine-grained performance anomaly detection and problem locating system. In *Proceedings of the IEEE 9th International Conference on Cloud Computing*. IEEE, 917–920.

[125] Yanyan Zhuang, Eleni Gessiou, Steven Portzer, Fraida Fund, Monzur Muhammad, Ivan Beschastnikh, and Justin Cappos. 2014. NetCheck: Network diagnoses from blackbox traces. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation (NSDI'14)*. USENIX Association, Berkeley, CA, 115–128.

Received June 2018; revised February 2019; accepted May 2019