Received: 00 Month 0000 | Revised: 00 Month 0000 | Accepted: 00 Month 0000

DOI: xxx/xxxx

ARTICLE TYPE

Evaluation of cloud autoscaling strategies under different incoming workload patterns

Maria Carla Calzarossa[1] | **Luisa Massari**[1] | **Daniele Tessera**[2]

[1]Department of Electrical, Computer and Biomedical Engineering, Università di Pavia, Pavia, Italy

[2]Department of Mathematics and Physics, Università Cattolica del Sacro Cuore, Brescia, Italy

Correspondence

*Luisa Massari, Email: luisa.massari@unipv.it

Present Address

Department of Electrical, Computer and Biomedical Engineering, Università di Pavia, Via Ferrata 5 - I-27100 Pavia, Italy

Summary

Cloud computing provides cost effective solutions for deploying services and applications. Even though resources can be provisioned on demand, they need to adapt quickly and in a seamless way to the workload intensity and characteristics and satisfy at the same time the desired performance levels. In this paper, we evaluate the effects exercised by different incoming workload patterns on cloud autoscaling strategies. More specifically, we focus on workloads characterized by periodic, continuously growing, diurnal and unpredictable arrival patterns. To test these workloads, we simulate a realistic cloud infrastructure using customized extensions of the CloudSim simulation toolkit. The simulation experiments allow us to evaluate the cloud performance under different workload conditions and assess the benefits of autoscaling policies as well as the effects of their configuration settings.

KEYWORDS:

Cloud computing, resource management, autoscaling policies, workload characterization, workload patterns, CloudSim.

1 | INTRODUCTION

Cloud technologies are being used nowadays in many scientific and business domains because of their ability to offer cost effective scalable solutions based on a pay-per-use model. This model – coupled with the native features of the cloud computing paradigm, such as virtualization, rapid elasticity, resource pooling – gives customers the possibility to efficiently deploy their applications and reduce the corresponding monetary cost.

A key factor of these technologies is the elasticity that allows the adaptation of the resources being provisioned to time varying workloads. Resources can be provisioned (i.e., allocated and deallocated) on demand [1], that is, when they are actually needed. Although very flexible, dynamic resource provisioning is quite challenging. In fact, to avoid under-provisioning and over-provisioning, it is necessary to accurately take account of the fluctuations of the workloads being processed.

The workloads deployed in cloud environments consist of many different types of services and applications with their own resource demands and performance requirements [2,3]. The intensity of these workloads and their arrival patterns often change rapidly and even in unpredictable manners as a consequence of the user interactions. Therefore, to cope with this variability and avoid at the same time performance degradations and service level violations, the decisions about provisioning of cloud resources must be taken automatically and in a timely manner.

In this article we address these issues by extending in various directions our previous work on cloud autoscaling [4]. More precisely, we define a modeling framework built around three main components, namely:

- workloads;
- cloud resources;

- workload and resource management.

In particular, among the various aspects entailed in workload and resource management, our modeling framework addresses the distribution of workloads to cloud resources (i.e., scheduling) and the dynamic provision of the "right" amount of resources at the "right" time according to the actual load conditions (i.e., autoscaling). Moreover, to evaluate cloud performance in different scenarios (e.g., workload characteristics, resource settings) we implement the framework in a simulation environment based on the CloudSim simulation toolkit [5] because of its ability to reproduce realistic cloud environments. Our experiments focus on workloads characterized by periodic, continuously growing, diurnal and unpredictable arrival patterns simulated under different autoscaling policies.

The paper is organized as follows. In Section 2 we present a review of the state of the art. The modeling framework for evaluating cloud performance under different workload conditions is defined in Section 3. The simulation environment that implements this framework is described in Section 4, whereas in Section 5 we present the setup of the simulation experiments and their results. Finally, we conclude the paper with some remarks in Section 6.

2 | RELATED WORK

In this section we review the state of the art in the areas of workload and resource management and simulation environments developed for evaluating the performance of workloads and cloud infrastructures.

2.1 | Workload and autoscaling

The issues related to workload and resource management in cloud environments have been extensively studied with the main objective of efficiently provisioning the resources and satisfying at the same time the service levels associated with the workloads being processed. In this framework, the availability of accurate workload models is fundamental. Some papers [6,7,8] focus on specific aspects of the workload arrival process, while others [9,10] address the problem of workload generation from the perspective of the cloud users. In particular, Magalhães et al. [9] propose web workload models that capture different user patterns. Solis Moreno et al [10] model the user behavior in terms of task submission rate and CPU and memory requirements. This study shows that submission rates are highly variable, thus highlighting the heterogeneity of cloud workloads.

In the framework of resource management, comprehensive surveys and taxonomies of cloud autoscaling policies have been published in the literature [1,11,12]. In general, these policies focus on various objectives, such as improving performance, increasing resource utilization, saving energy, reducing monetary cost and ensuring availability. The decisions about the allocation or deallocation of cloud resources often rely on some monitored or predicted low-level performance indicators (e.g., CPU, memory, network utilization) or high-level indicators (e.g., response time) based on the QoS perceived by users [13,14]. Ilyushkin et al. [15] propose a method that enables cloud users and providers to compare the performance of different autoscalers and exercise a simple control over elasticity when running workflow-based workloads.

Autoscaling policies are usually classified in two categories, namely, reactive and proactive, depending on the approach used to trigger scaling actions. On the one hand, reactive policies [16,17] are simpler because resource provisioning responds to workload changes. Nevertheless, these policies may be slow in detecting the changes and their performance is affected by their configuration settings.

On the other hand, proactive policies implement forecasting techniques that try to anticipate the future needs and trigger scaling actions based on these predictions. To predict workload changes, techniques, such as machine learning and neural networks, are usually applied [18,19,20,21]. Moreover, to manage unpredicted changes, some works [22,23,24,25] combine proactive autoscaling approaches – driven by workload models – with reactive approaches.

In this work, we do not propose any new autoscaling policy, instead we apply state of the art reactive autoscaling policies to investigate the impact on the performance of different workload patterns and of the configuration settings of the policies.

2.2 | Simulation environments

The performance of the resource management and scheduling solutions is often evaluated with simulation approaches, since evaluations carried out in real cloud environments can be expensive and often unfeasible. In fact, it is difficult to perform repeatable experiments especially in case of resources offered by public cloud providers.

Most simulation frameworks – see [26] for a survey – rely on the CloudSim simulation toolkit [5] and its extensions. In particular, Vondra et al. [27] extend the CloudSim toolkit for adding and removing VMs at runtime and for accurately simulating interactive workloads. CloudAnalyst [28] supports visual modeling and simulation of large-scale cloud applications described in terms of number and location of users generating the traffic, number and location of data centers and number of resources in each data center. NetworkCloudSim [29] provides extensions to specify the network topology.

Piraghaj et al.[30] propose ContainerCloudSim, a tool that supports the evaluation of scheduling and allocation policies in containerized cloud data centers.

Similarly to these works, we develop a simulation environment implemented as customized extensions of the CloudSim toolkit. Our environment enables the performance evaluation of autoscaling policies under workloads with different characteristics and arrival patterns.

3 | MODELING FRAMEWORK

To assess cloud performance under different working scenarios, we propose a modeling framework consisting of three main components – namely, workloads, cloud resources, workload and resource management – interacting together (see Figure 1). In particular, the workload is the set of requests submitted by the users to the cloud infrastructure. In turn, the infrastructure consists of the resources, e.g., VMs, that can be provisioned. In the figure we can identify some resources already provisioned and some others available for provisioning. Finally, workload and resource management deals with scheduling the user requests into the cloud resources and dynamically adapting the resources according to the load conditions, i.e., autoscaling.

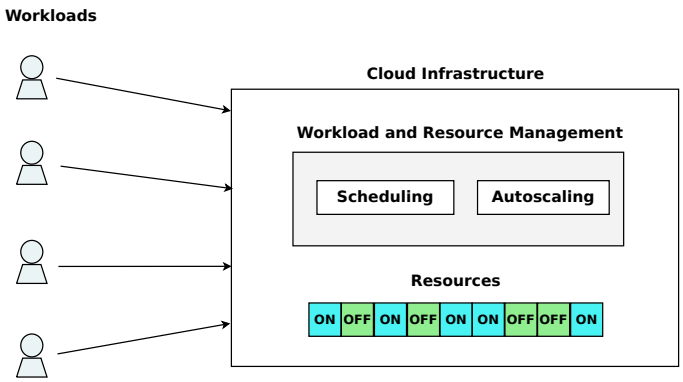


FIGURE 1 Modeling framework.

In what follows, we explore the overall properties of cloud workloads and the role of the management components in relation to the characteristics of the workloads being processed and of the cloud resources.

3.1 | Cloud workloads

As already pointed out, the amount of provisioned resources depends on the characteristics of the incoming workloads. In particular, different workload types and arrival patterns have different impacts on the usage of cloud resources. Therefore, it is important to have a clear understanding of the behaviors of the workloads.

Three main dimensions are typically considered for describing and classifying the workloads[3], namely:

- processing model;
- architectural structure;
- resource requirements.

In particular, two broad workload categories, i.e., batch and interactive, are easily identified according to their processing model. More precisely, batch workloads usually refer to long lived computation-intensive applications processed without any user intervention, while interactive workloads refer to short lived request/response applications submitted by a variable number of concurrent users.

Moreover, since the deployment of many applications and services often relies on the instantiation of multiple tasks (or services), their architectural structure explains how the components are organized and interact together to cope with the control and data flows of these applications. In general, these composite applications can consist of loosely coupled tasks – to be executed sequentially or in parallel – or of tightly coupled tasks

– whose execution is driven by precedence constraints among tasks. A Direct Acyclic Graph – whose nodes correspond to tasks and whose edges describe their constraints – is typically used to represent a task-oriented application, such as a workflow.

The third key dimension for describing the workloads is represented by their resource requirements, i.e., their demands. These quantitative attributes can refer to an entire application or to individual tasks of composite applications. Depending on the resources being exploited, the demands are defined as follows:

- computation demand, that is, amount of processing;
- memory demand, that is, amount of RAM;
- communication demand, that is, data volume to be exchanged with other tasks;
- transfer demand, that is, data volume to be transferred to/from (local or remote) I/O devices.

Note that these attributes are the basis for identifying – by applying multivariate analysis techniques – classes of workloads with different behaviors in terms of resource usage. This identification is an important step to accurately model and represent the peculiarities of the workloads being processed.

As for the workload demands, the frequency at which batch or interactive applications are submitted by the users, that is, their arrival rate, is another key aspect that affects the usage of cloud resources and has to be considered for effective resource provisioning. Workload streams are seldom deterministic, instead they are characterized by patterns that reflect the user behaviors. In detail, arrival patterns experience different effects, e.g., business hours, week vs weekend days, seasonality, flash crowd[31]. For example, diurnal patterns are characterized by higher rates during business hours and weekdays than over night and weekend days. Similarly, seasonality leads to patterns that periodically repeat in time, while continuously changing workloads often experience a steady growth. On the contrary, unpredictable events, such as flash crowds, cause sudden increases of the workload intensity, thus leading to unexpected peaks of load.

3.2 | Cloud resources

The characteristics of the resources offered by cloud providers mainly refer to the performance of the individual VMs, namely:

- processing capacity;
- memory size;
- network bandwidth;
- transfer rate to/from I/O devices.

These characteristics affect the overall performance and the monetary cost associated with the workloads being processed. Therefore, to reduce monetary cost, cloud resources have to be dynamically allocated/deallocated according to the actual workload requirements.

3.3 | Workload and resource management

The workload and resources management is a fundamental component of any cloud infrastructure in that it affects the overall efficiency of the infrastructure and in particular the exploitation of the resources as well as the performance experienced by the users. In our modeling framework we consider two critical aspects of the management dealing with workload scheduling and resource autoscaling.

3.3.1 | Workload scheduling

As already pointed out, scheduling deals with distributing the incoming workloads to cloud resources. In particular, scheduling has to handle the workloads in an efficient manner and ensure at the same time their requirements (e.g., number of cores, memory size). Moreover, scheduling can be performed at the level of entire applications or at the level of individual tasks of composite applications.

Scheduling decisions are driven by different objectives (e.g., load balancing, VM consolidation). Hence, the scheduling policies have to be chosen accordingly. For example, for balancing the load across a pool of VMs, scheduling policies, such as Round Robin or Weighted Round Robin, are applied, while for VM consolidation the scheduling decisions need to be coupled with autoscaling strategies.

3.3.2 | Resource autoscaling

Resource autoscaling focuses on dynamically provisioning and releasing resources. In our framework, autoscaling decisions are based on estimations of some scaling indicators that quantify the actual performance (e.g., response time, throughput, VM utilization) of the cloud infrastructure. Static or dynamic thresholds associated with these indicators trigger scaling actions. These actions may also be triggered by predictions of the future incoming workloads and therefore of the future resource needs.

In general, the effectiveness of autoscaling mechanisms depends on many factors, including – among the others – the characteristics of the workloads and in particular of their incoming arrival patterns.

4 | SIMULATION ENVIRONMENT

As already stated, the modeling framework presented in Section 3 has been implemented in a simulation environment based on the CloudSim 4.0 simulation toolkit. Figure 2 depicts the overall architecture of our environment We can identify various types of components, namely, components

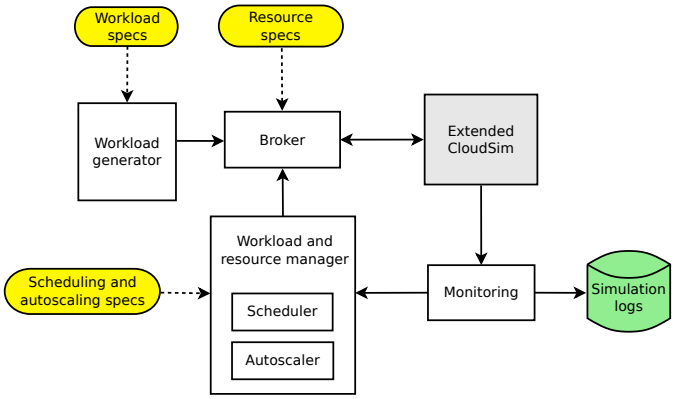


FIGURE 2 Architecture of the simulation environment.

that define the input specifications (i.e., Workload, Resource, Scheduling and autoscaling specs) and components that implement the modeling framework (i.e., *Workload generator*, *Broker*, *Workload and resource manager*). In addition, we identify a service component (i.e., *Monitoring*) that collects performance data into Simulation logs. The architecture includes the CloudSim simulation engine (i.e., *Extended CloudSim*) customized for integrating the various components of our simulation environment. In what follow, we describe the individual components of our simulation environment starting from the specifications to be provided.

The input specifications define the characteristics of the workload (e.g., incoming arrival patterns and resource demands), of the resources (e.g., VM processing capacity, startup time) and of the scheduling and autoscaling policies (e.g., policy, thresholds, cooling time). These specifications drive at simulation time the components responsible for generating the workload, allocating the resources as well as for mapping workload and resources and for their dynamic provisioning .

In particular, the *Workload generator* component generates workloads in terms of their arrival patterns and demands. In detail, arrivals are generated according to some basic patterns, such as growing, periodic and unpredictable, or their combinations. Moreover, thanks to the Stochastic Simulation in Java (SSJ) software library [32] workload demands can be described in terms of probability distributions.

The *Workload and resource manager* is the component responsible of scheduling the workload to the most suitable VMs (*Scheduler*) and of dynamically adapting (i.e., provisioning, releasing) the number of allocated VMs according to the time varying workload behavior (*Autoscaler*). Note that to decide about scheduling and autoscaling activities, this component relies on performance data (e.g., VM utilization) provided by the *Monitoring* component.

In detail, the *Scheduler* tries to balance the load among the available VMs by cooperating with the *Autoscaler*. For example, VMs that are about to be released by the *Autoscaler* will not be considered for scheduling new incoming workload requests. On the contrary, requests will be scheduled to newly allocated VMs as soon as they become ready. In fact, our simulation environment implements the concept of VM startup time [33], that is, the delay between the time a VM is provisioned and it is ready. The *Scheduler* implements well known scheduling policies [34], such as Least Recently Used, Round Robin, Weighted Round Robin and Constrained Scheduling.

The *Autoscaler* implements two reactive policies, namely, *Average Load* and *Discrete Sampling*. These policies aim at keeping the utilization of allocated VMs, that is, the fraction of time they are busy processing, within predefined thresholds. For this purpose, the policies compute scaling indicators based on some recent historical data of the VMs. More precisely, the *Average Load* policy focuses on the behavior of the individual VMs and computes as indicator the utilization over time of each VM. On the contrary, the *Discrete Sampling* policy considers the behavior of all VMs and computes as indicator the fraction of busy VMs. Scaling actions are then triggered according to these indicators.

Autoscaling specifications define the thresholds that trigger scaling actions, the number of VMs to be allocated/deallocated at each action as well as other specific configuration parameters (e.g., time interval used for computing the individual VM utilization, number of samples and sampling interval for computing the fraction of busy VMs).

To avoid oscillations in scaling actions and take account of the cause-effect principle, autoscaling policies adopt a grace time period, namely, a cooling time, between two consecutive actions. This time – defined in the autoscaling specifications – allows the *Scheduler* to adapt to the available number of VMs before any further scaling action.

The interactions among the various components of our simulation environment and the *Extended CloudSim* are orchestrated by the *Broker* component. More precisely, this component collects the workload arrivals from the *Workload generator* and forwards them to the CloudSim simulation engine according to the resource management directives. Similarly, the *Broker* forwards the specifications of the VMs available for provisioning to the simulation engine.

5 | PERFORMANCE EVALUATION

The simulation environment presented in Section 4 is the basis for evaluating the performance of the workloads and the benefits of the proposed autoscaling policies and for assessing the effects exercised by their configuration parameters. In fact, simulation is widely used in cloud environments because of its ability to run experiments in repeatable and controlled conditions. For this purpose, we simulate the deployment of workloads – characterized by different arrival patterns and resource demands – on a cloud infrastructure consisting of multiple instances of a single VM type. In what follows, we present the details of the setup and the results of the simulation experiments.

5.1 | Experiment setup

The setup of the simulation experiments refers to the specifications of the characteristics of the workloads and of their arrival process as well as of the configuration parameters associated with the resource management policies.

5.1.1 | Workload specifications

In our comprehensive evaluation we simulate the static and dynamic behaviors of the workloads by varying the following characteristics:

1. Workload composition:
    - stand alone applications statistically identical in terms of resource demands;
    - composite applications grouped in different classes according to their resource demands;
    - web requests drawn from a real workload.
2. Workload arrival process:
    - arrival rate;
    - arrival patterns (i.e., growing, periodic, unpredictable, diurnal).
3. Resource requirements:
    - constant;
    - probability distributions that account for the variable nature of the workloads.

Note that in our experiments the resource requirements of the applications refer to their computation demands. Moreover, to take fully account of the user behaviors, we consider a simulated time of at least an hour, while for reproducing the diurnal patterns we consider a 24 hours simulated time.

5.1.2 | Management specifications

Specifications about the management of the cloud resources refer to a large number of critical parameters that control the behavior of the *Scheduler* and *Autoscaler*. These specifications include the choice of the autoscaling policy and the parameters that drive the autoscaling decisions (e.g., number of samples, sampling interval). Other important parameters that affect the behavior and performance of autoscalers refer to the lower and upper thresholds that trigger scaling actions as well as how often the indicator is checked (in our simulations every second), the cooling time between two consecutive actions, the time resolution of the *Monitoring* and the number of VMs to be provisioned or released at each scaling action – referred to as *DeltaVM* in what follows. In addition, the specifications define the properties of the VMs, i.e., their processing capacity and startup time. Finally, for distributing the incoming requests across the allocated VMs, the specifications define the scheduling policy – that is, Weighted Round Robin in our experiments.

Table 1 summarizes the main management parameters that characterize all simulation experiments.

VM startup time	Monitoring resolution	Thresholds		Cooling time
[s]	[s]	Lower	Upper	[s]
30	0.1	0.4	0.8	35

TABLE 1 Management specifications.

5.2 | Experimental results

To assess the behavior and performance of the workload and of the autoscaling policies, we perform three sets of experiments by varying the workload characteristics and the configuration parameters of the resource management policies. For these evaluations, we consider various metrics, e.g,. completion time, number of scaling actions, VM utilization. Moreover, in order to assess how fast a policy reacts to over/underload conditions, we compute how often the VM utilization is above or below the predefined thresholds – in what follows referred to as *PctOverLoad* and *PctUnderLoad*, respectively.

5.2.1 | Static and dynamic provisioning

The first set of experiments is aimed at assessing the benefits of a dynamic resource provisioning with respect to a static one. The workload considered in these experiments consists of simple standalone applications composed of a single task whose processing demand is equal to 522 ms. These applications are submitted by the users according to two distinct arrival patterns, namely, continuously growing and unpredictable (see Fig. 3). The growing pattern is characterized by a rate linearly increasing from 100 up to 6,000 requests/min in one hour simulated time, that is, on average 3,050 requests/min. On the contrary, for the unpredictable pattern, the behavior of the arrivals suddenly changes and their rate increases of two orders of magnitude in about 20 minutes (from 30 up to 2,400 requests/min). The average arrival rate over two hours simulated time is equal to 1,123 requests/min.

Resources are provisioned according to the *Discrete Sampling* policy – with scaling indicator computed at one-second resolution over the three latest samples of the VM utilization. Let us recall that the thresholds associated with scaling actions are equal to 0.4 and 0.8. In addition, the number of VMs to be provisioned/released at each scaling action varies over the simulated time and is set to 10% of the number of allocated VMs. Figure 3 suggests that the number of allocated VMs nicely follows the growing arrival pattern, whereas for the unpredictable pattern the policy reacts more slowly and VMs are not released as soon as the arrival rate decreases.

To assess the benefits of dynamic provisioning, we compare its performance with the performance of static provisioning. For this purpose, we allocate 36 and 74 VMs in the simulation experiments with the growing arrival pattern and 16 and 31 VMs in the simulation experiments with the unpredictable pattern. These values correspond to the average (Avg) and maximum (Max) numbers of VMs provisioned by the *Discrete Sampling* policy. As can be seen in Fig. 3, since static provisioning does not take account of the workload patterns, it often leads to either under-provisioning or over-provisioning.

More details about these experiments are presented in Table 2. In particular, the table summarizes the application completion time and the utilization of the provisioned VMs together with the *PctOverLoad* and *PctUnderLoad* that quantify how often the VM utilization is above/below the upper/lower thresholds over the simulated time. As expected, the applications are completed very quickly in the case of over-provisioning (i.e., Max) – although VMs are poorly utilized. On the contrary, even though the experiments with the autoscaling policy and Avg provisioning lead

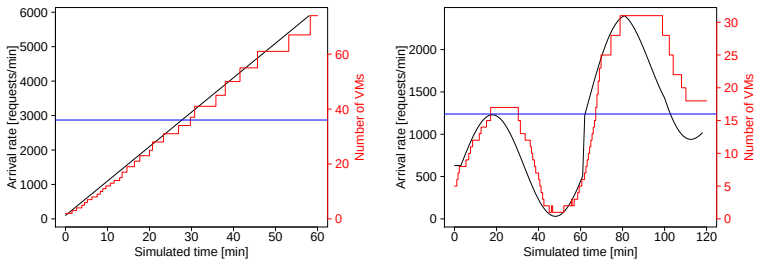


FIGURE 3 Continuously growing (a) and unpredictable (b) workload patterns (black lines) and number of VMs (red step functions) allocated by the *Discrete Sampling* autoscaling policy. Blue lines represent the static Avg provisioning.

	Growing pattern				Unpredictable pattern			
	Completion time [s]	VM utilization	*PctOverLoad* ≥ 0.8	*PctUnderLoad* ≤ 0.4	Completion time [s]	VM utilization	*PctOverLoad* ≥ 0.8	*PctUnderLoad* ≤ 0.4
Autoscaling	2.03	0.72	31.27%	1.56%	4.49	0.58	19.14%	19.25%
Avg	51.6	0.66	46.78%	29.08%	159.40	0.60	38.96%	32.89%
Max	1.35	0.35	2.02%	60.22%	0.73	0.31	0.49%	72.63%

TABLE 2 Summary of the performance of the *Discrete Sampling* autoscaling policy and of static provisioning (i.e., Avg, Max) for growing and unpredictable workload patterns.

to similar VM utilizations, the application completion times are very different. In fact, the number of allocated VMs does not take account of the actual workload increase and becomes insufficient. We remark that these behaviors do not depend on the arrival patterns. Moreover, because of the peculiarity of the growing pattern, the VM utilization obtained with autoscaling is higher and in particular the *PctUnderLoad* is only 1.56%. Hence, autoscaling provides a good compromise between cloud resource usage and workload performance.

5.2.2 | Autoscaling parameter configuration

The second set of experiments is aimed at studying the impact of the configuration parameters associated with the resource management policies under different workloads. Both *Discrete Sampling* and *Average Load* policies are evaluated.

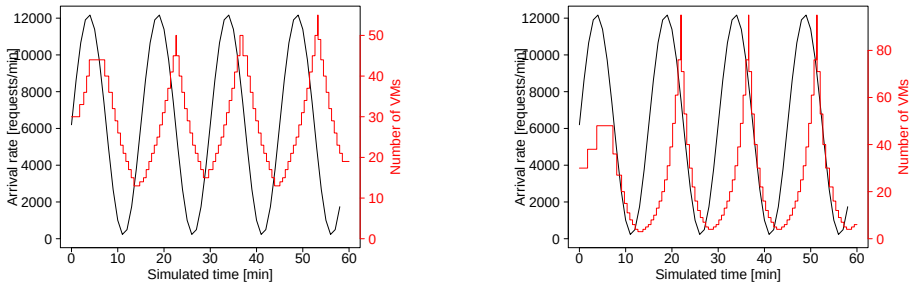
Discrete Sampling policy

We test the *Discrete Sampling* policy under different configuration parameters, by varying *DeltaVM* – that is the number of VMs allocated/deallocated at each scaling action – and the number of samples for computing the scaling indicator.

The workload consists of composite applications subdivided into three groups according to their processing demands, namely, short applications – whose demands are uniformly distributed in the range $[10, 14]$ ms – medium applications – whose demands are uniformly distributed in the range $[200, 240]$ ms – and large applications – whose demands are uniformly distributed in the range $[400, 522]$ ms. The arrival process follows a periodic pattern, characterized by a rapid increase followed by a rapid decrease with a rate ranging from 200 to 12,000 requests/min.

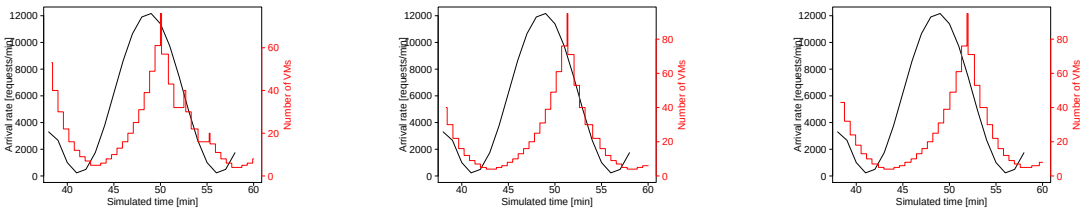
For this type of pattern, the choice of the value of *DeltaVM* is very important. Indeed, to quickly react to the high variability of the workload arrivals and to their sudden changes, the processing capacity of the VMs to be allocated/deallocated has to match the workload characteristics and intensity. Hence, instead of provisioning/releasing a fixed number of VMs at a time – that may be too conservative or aggressive – it is appropriate to choose a variable number proportional to the number of allocated VMs (by specifying the corresponding percentage). As shown in Figure 4, the autoscaling policy better adapts the resource provisioning to the workload arrival pattern when a larger number of VMs is allocated/deallocated at each scaling action (i.e., *DeltaVM* equal to 25%).

This workload has also been used to evaluate the sensitivity of the autoscaling policy varying the number of samples used to compute the scaling indicator. Figure 5 shows a snapshot of these behaviors with one, three and ten samples. We notice that a small number of samples leads to oscillations in the number of allocated VMs. On the contrary, a large number of samples makes the policy slow in reacting to workload changes. Simulation experiments with more complex workloads – characterized by different arrival patterns – confirm these results.



DeltaVM equal to 10% *DeltaVM* equal to 25%

FIGURE 4 Behavior of the *Discrete Sampling* autoscaling policy with *DeltaVM* equal to 10% (a) and 25% (b). Black curves correspond to the arrival patterns, red step functions to the allocated VMs.



One sample Three samples Ten samples

FIGURE 5 Behavior of the *Discrete Sampling* autoscaling policy varying the number of samples for the computation of the scaling indicator. Black curves correspond to the arrival patterns, red step functions to the allocated VMs.

Average Load policy

To evaluate the performance of the *Average Load* policy under different flavors, we use as scaling indicator the utilization over time of each VM and we vary the condition that triggers the autoscaling action as follows:

(a) at least one indicator above/below thresholds;

(b) mean of all indicators above/below thresholds;

(c) all indicators above/below thresholds.

As configuration parameters, we set a time interval equal to three seconds for computing the indicator and *DeltaVM* equal to 10% for the scaling actions.

The workload considered in these experiments – characterized by a growing arrival pattern – consists of composite applications with a core component that – upon completion – instantiates 40 components at 0.1 seconds apart from each other. These components – subdivided into three groups according to their processing demands– are scheduled on the same VM and processed sequentially or in parallel according to their arrival time and processing demand.

Figure 6 depicts the behavior of the three flavors of the policy as a function of simulated time. The step function represents the number of VMs allocated and the light blue area the unused processing capacity. Figure 6(c) suggests that, whenever all indicators are required to be above/below thresholds, the cost of the autoscaling – in terms of number of scaling actions – is minimized, even though the policy is quite conservative. In addition, a closer look of the VM utilizations (see Fig. 7) shows that the *PctOverLoad* is about 40%.

In summary, as Table 3 suggests, autoscaling decisions based on the mean of all indicators result in a good compromise between resource provisioning and performance.

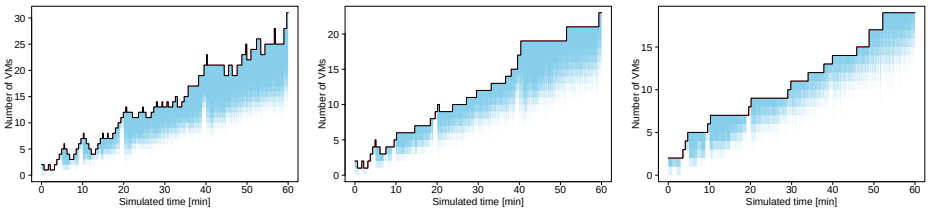


FIGURE 6 Number of allocated VMs for the *Average Load* policy whose scaling actions are triggered by at least one indicator (a), mean of all indicators (b) and all indicators (c). The light blue area represents the unused processing capacity.

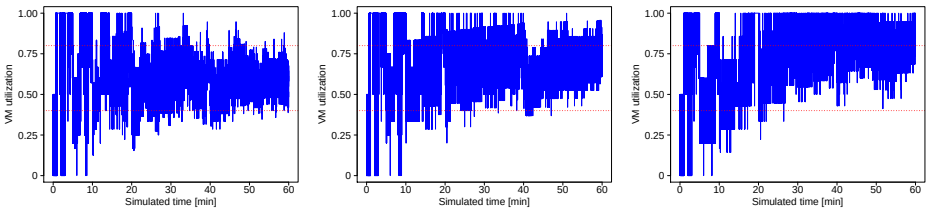


FIGURE 7 VM utilization as a function of simulated time for *Average Load* policy whose scaling actions are triggered by at least one indicator (a), mean of all indicators (b) and all indicators (c).

	At least one	Mean	All
Number of scaling actions	72	27	15
VM utilization	0.58	0.68	0.76
Completion time [s]	4.1	4.5	5.6

TABLE 3 Performance of *Average Load* policy whose scaling actions are triggered by at least one indicator, mean of all indicators and all indicators.

5.3 | Web workload

The last set of experiments is aimed at assessing the performance of the autoscaling policies when processing web requests drawn from a real workload. In particular, the experiments are driven by a workload model derived from measurements collected on the web server of the University of Pavia during a 24 hours period. Some 330,000 HTTP requests for web pages were received and processed by the web server. The composition of the web pages, that is, number of embedded objects and their size, is obtained by analyzing the structural properties of the website. A page consists of an HTML file that on average references 40 embedded objects. The processing demand of each object is set proportional to its size and is described in terms of two equally probable uniform distributions, one in the range $[0.2, 2]$ μs and the other in the range $[110, 118]$ ms.

Another important characteristic of the workload model is the interarrival time between two consecutive requests to the embedded objects of a given page. Since these times are usually quite small, that is, one second or less, we model them using two uniform distributions in the range $[0, 1]$ s and $[1, 2]$ s, respectively. The probabilities associated with these distributions are set to 0.97 and 0.03.

In these experiments, we consider the *Discrete Sampling* autoscaling policy – with scaling indicator computed over four samples and *DeltaVM* equal to one. Figure 8 shows the daily arrival pattern as a function of time and the behavior of the autoscaling policy in terms of the number of allocated VMs. The workload intensity is characterized by a clear diurnal pattern, with much fewer requests – as little as 42 requests/min – during the night and early morning hours, and a peak of about 400 requests/min around noon. To cope with this variability, up to 23 VMs are allocated during the simulated time, with a mean of 14 VMs. In addition, as expected, the average VM utilization is equal to 0.61. This is in line with the upper and lower thresholds (i.e., 0.8 and 0.4) used in these experiments.

This workload has also been used to test the new HTTP/2 protocol [35] and in particular the so called "server push" optimization mechanism that allows web servers to speculatively send resources without waiting for explicit client requests. Therefore, for improving the time required to load

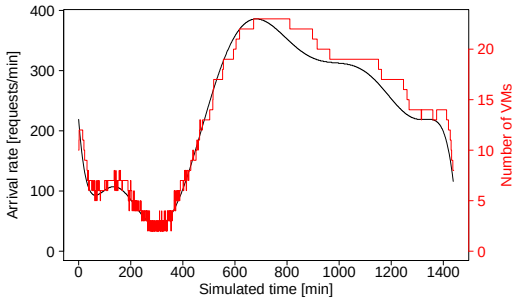


FIGURE 8 Daily request arrival pattern (black curve) and number of allocated VMs (red step function).

the entire web page, a single HTTP request can trigger multiple HTTP responses. In this experiment, we assume that half of the embedded objects is pushed by the web server, while the other half is explicitly requested by the client. Table 4 provides basic statistics of the time required to process a web page without and with pushing mechanism. The cumulative distribution functions of the time spent to process a page are shown in Fig. 9.

	Minimum	Mean	Maximum
No push	12.3	21.2	38.4
Push	5.1	11.0	22.1

TABLE 4 Time – expressed in seconds – to process a web page without and with the push mechanism.

In details, the red curve refers to the experiment without any pushing, whereas the blue curve refers to the experiment with pushing. We notice that pushing mechanism allows for a reduction of this time of about 50%.

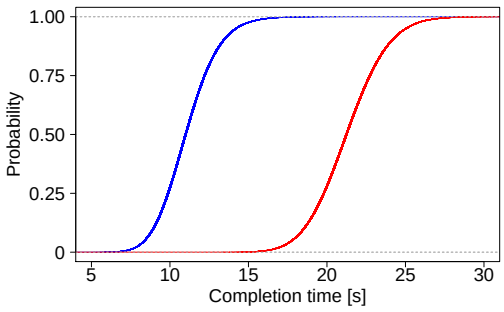


FIGURE 9 Cumulative distribution functions of the time required to process a Web page without (red curve) and with (blue curve) pushing.

6 | CONCLUSIONS

Cloud technologies offer cost effective scalable solutions for deploying applications and services. The evaluation of the behaviors and performance of these technologies is very challenging and has to take account of the workload and infrastructure characteristics as well as of the complex strategies adopted to efficiently provision and manage cloud resources.

In this article we defined a modeling framework that copes with these issues. In particular, our framework covers the various aspects entailed in cloud management by focusing in particular on workload scheduling and resource autoscaling. To assess the effects exercised by different incoming workload patterns on the behavior and performance of autoscaling strategies, we implemented the framework as customized extensions of the CloudSim simulation toolkit because of its capability to reproduce realistic cloud environments.

In our experiments we simulated various types of workloads, namely, simple standalone applications and composite applications – consisting of single and multiple components – as well as web workloads. Moreover, we considered arrival patterns typical of cloud workloads, namely, periodic, growing, unpredictable and diurnal. As expected, the simulation experiments have shown that autoscaling policies improve VM utilization with respect to a static resource provisioning independently of the workload characteristics and patterns. However, the performance of these policies is strongly affected by the choice of their configuration parameters which in turn depend on the workloads being processed.

ACKNOWLEDGEMENTS

Authors would like to thank Dr. Momin I.M. Tabash for his important contribution to the workload analysis and the anonymous referees for their valuable comments and suggestions that improved the overall quality and clarity of the manuscript.

References

1. Al-Dhuraibi Y., Paraiso F., Djarallah N., Merle P.. Elasticity in Cloud Computing: State of the Art and Research Challenges. *IEEE Transactions on Services Computing.* 2018;11(2):430–447.

2. Calzarossa M. C., Massari L., Tessera D.. Workload Characterization: A Survey Revisited. *ACM Computing Surveys.* 2016;48(3):48:1–48:43.

3. Calzarossa M. C., Della Vedova M. L., Massari L., Petcu D., Tabash M. I. M., Tessera D.. Workloads in the Clouds. In: Fiondella L., Puliafito A., eds. *Principles of Performance and Reliability Modeling and Evaluation*, Springer Series in Reliability Engineering. Springer 2016 (pp. 525–550).

4. Calzarossa M. C., Massari L., Tabash M. I. M., Tessera D.. Cloud autoscaling for HTTP/2 workloads. In: Proc. of the 3rd Int. Conf. of Cloud Computing Technologies and Applications - CloudTech'17:167-171; 2017.

5. Calheiros R., Ranjan R., Beloglazov A., De Rose C. A. F., Buyya R.. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience.* 2011;41(1):23–50.

6. Iqbal W., Erradi A., Mahmood A.. Dynamic workload patterns prediction for proactive auto-scaling of web applications. *Journal of Network and Computer Applications.* 2018;124:94-107.

7. Pacheco-Sanchez S., Casale G., Scotney B., McClean S., Parr G., Dawson S.. Markovian Workload Characterization for QoS Prediction in the Cloud. In: Proc. of the 2011 IEEE 4th Int. Conf. on Cloud Computing - CLOUD'11:147–154; 2011.

8. Panneerselvam J., Liu L., Antonopoulos N., Bo Y.. Workload Analysis for the Scope of User Demand Prediction Model Evaluations in Cloud Environments. In: Proc. of the 2014 IEEE/ACM 7th Int. Conf. on Utility and Cloud Computing - UCC'14:883–889; 2014.

9. Magalhães D., Calheiros R. N., Buyya R., Gomes D. G.. Workload modeling for resource usage analysis and simulation in cloud computing. *Computers & Electrical Engineering.* 2015;47:69–81.

10. Solis Moreno I., Garraghan P., Townend P., Xu J.. Analysis, Modeling and Simulation of Workload Patterns in a Large-Scale Utility Cloud. *IEEE Transactions on Cloud Computing.* 2014;2(2):130–142.

11. Lorido-Botran T., Miguel-Alonso J., Lozano J. A.. A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments. *Journal of Grid Computing.* 2014;12(4):559–592.

12. Qu C., Calheiros R. N., Buyya R.. Auto-scaling Web Applications in Clouds: A Taxonomy and Survey. *ACM Computing Surveys.* 2018;51(4):73:1–73:33.

13. Assuncao M. D., Cardonha C. H., Netto M. A. S., Cunha R. L. F.. Impact of user patience on auto-scaling resource capacity for cloud services. *Future Generation Computer Systems.* 2016;55:41–50.

14. Liu X. L., Yuan S. M., Luo G. H., Huang H. Y., Bellavista P.. Cloud Resource Management with Turnaround Time Driven Auto-Scaling. *IEEE Access.* 2017;5:9831-9841.

15. Ilyushkin A., Ali-Eldin A., Herbst N., et al. An Experimental Performance Evaluation of Autoscalers for Complex Workflows. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems.* 2018;3(2):8:1–8:32.

16. Netto M. A. S., Cardonha C. H., Cunha R. L. F., Assuncao M. D.. Evaluating Auto-scaling Strategies for Cloud Computing Environments. In: Proc. of the 22nd Int. Symp. on Modeling, Analysis & Simulation of Computer and Telecommunications Systems - MASCOTS'2014:187–196; 2014.

17. Augustyn D. A.. Improvements of the Reactive Auto Scaling Method for Cloud Computing. In: Gaj P., Kwiecien A., Sawicki M., eds. *Computer Networks*, Communications in Computer and Information Science, vol. 718: Springer 2017 (pp. 422–431).

18. Roy N., Dubey A., Gokhale A.. Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting. In: Proc. of the 2011 IEEE 4th Int. Conf. on Cloud Computing - CLOUD'11:500-507; 2011.

19. Weingärtner R., Bräscher G.B., Westphall C.B.. Cloud resource management: A survey on forecasting and profiling models. *Journal of Network and Computer Applications.* 2015;47:99–106.

20. Nikravesh A. Y., Ajila S. A., Lung C-H.. An autonomic prediction suite for cloud resource provisioning. *Journal of Cloud Computing: Advances, Systems and Applications.* 2017;6(1).

21. Kumar J., Singh A. K.. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems.* 2018;81:41–52.

22. Ali-Eldin A., Tordsson J., Elmroth E.. An adaptive hybrid elasticity controller for cloud infrastructures. In: Proc. of the IEEE Network Operations and Management Symposium:204–212; 2012.

23. Gandhi A., Harchol-Balter M., Raghunathan R., Kozuch A.. AutoScale: Dynamic, Robust Capacity Management for Multi-Tier Data Centers. *ACM Transactions on Computer Systems.* 2012;30(4):14:1–14:26.

24. Fernandez H., Pierre G., Kielmann T.. Autoscaling Web Applications in Heterogeneous Cloud Infrastructures. In: Proc. of the IEEE Int. Conf. on Cloud Engineering - IC2E:195–204; 2014.

25. Gandhi A., Dube P., Karve A., Kochut A., Zhang L.. Adaptive, Model-driven Autoscaling for Cloud Applications. In: Proc. of the 11th Int. Conf. on Autonomic Computing - ICAC'14:57–64; 2014.

26. Fakhfakh F., Kacem H.H., Kacem A.H.. Simulation Tools for Cloud Computing: a Survey and Comparative Study. In: Proc. of the IEEE 16th Int. Conf. on Computer and Information Science - ICIS'17:221–226; 2017.

27. Vondra T., Šedivý J., Castro J. M.. Modifying CloudSim to accurately simulate interactive services for cloud autoscaling. *Concurrency and Computation: Practice and Experience.* 2017;29(10).

28. Wickremasinghe B., Calheiros R. N., Buyya R.. CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications. In: Proc. of the IEEE 24th Int. Conf. on Advanced Information Networking and Applications - AINA'10:446-452; 2010.

29. Garg S. K., Buyya R.. NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations. In: Proc. of the 2011 IEEE 4th Int. Conf. on Utility and Cloud Computing - UCC'11:105–113; 2011.

30. Piraghaj S. F., Dastjerdi A. V., Calheiros R. N., Buyya R.. ContainerCloudSim: An environment for modeling and simulation of containers in cloud data centers. *Software: Practice and Experience.* 2017;47:505–521.

31. Calzarossa M., Della Vedova M.L., Massari L., Nebbione G., Tessera D.. A methodological approach for time series analysis and forecasting of web dynamics. In: N. Nguyen F. Xhafa, ed. *Transactions on Computational Collective Intelligence XXXIII*, Lecture Notes in Computer Science, vol. 11610: Springer 2019 (pp. 128–143).

32. L'Ecuyer P., Meliani L., Vaucher J.. SSJ: A Framework for Stochastic Simulation in Java. In: Proc. of the 2002 Winter Simulation Conference:234–242; 2002.

33. Mao M., Humphrey M.. A Performance Study on the VM Startup Time in the Cloud. In: Proc. of the 2012 IEEE 5th Int. Conf. on Cloud Computing - CLOUD'12:423–430; 2012.

34. Pinedo M.L. *Scheduling: Theory, Algorithms, and Systems*. Springer; 2016.

35. Belshe M., Peon R., Thomson M.. *Hypertext Transfer Protocol Version 2 (HTTP/2).* RFC 7540: IETF; 2015.