Cluster Comput (2015) 18:919–932
DOI 10.1007/s10586-015-0449-5

Multi-agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure

Mahmoud Al-Ayyoub[1] · **Yaser Jararweh**[1] · **Mustafa Daraghmeh**[1] · **Qutaibah Althebyan**[1]

Received: 18 July 2014 / Revised: 10 December 2014 / Accepted: 13 March 2015 / Published online: 24 March 2015

Abstract The cloud computing paradigm provides a shared pool of resources and services with different models delivered to the customers through the Internet via an on-demand dynamically-scalable form charged using a pay-per-use model. The main problem we tackle in this paper is to optimize the resource provisioning task by shortening the completion time for the customers' tasks while minimizing the associated cost. This study presents the dynamic resources provisioning and monitoring (DRPM) system, a multi-agent system to manage the cloud provider's resources while taking into account the customers' quality of service requirements as determined by the service-level agreement (SLA). Moreover, DRPM includes a new virtual machine selection algorithm called the host fault detection algorithm. The proposed DRPM system is evaluated using the CloudSim tool. The results show that using the DRPM system increases resource utilization and decreases power consumption while avoiding SLA violations.

Keywords Cloud computing · Multi-agent system · Resource provisioning · Regression analysis

✉ Mahmoud Al-Ayyoub
maalshbool@just.edu.jo

Yaser Jararweh
yijararweh@just.edu.jo

Mustafa Daraghmeh
mustafa.daraghmeh@gmail.com

Qutaibah Althebyan
qaalthebyan@just.edu.jo

1 Jordan University of Science and Technology, Irbid, Jordan

1 Introduction

The world witnesses rapid advancements in the software used daily. With the emergence of new online services, the users have a continuously increasing need for more resources (such as computational power, memory, permanent storage, network bandwidth, etc.). Thus, the challenge of maintaining a high quality of service lies at the heart of the competition between companies. New technologies especially cloud computing offer companies a chance to provide high-quality online services with minimal cost.

The customers under the cloud computing paradigm are relieved from the burden of owning and operating the physical infrastructure required for their businesses. This means that they do not have to worry about developers and programming teams. Moreover, the customers need not worry about how or where the required tasks are performed; they only care about the cost of using the resources, the services they can get from the cloud providers and the quality of services guarantees. The cloud providers are the ones worrying about achieving efficient utilization and provisioning of resources (be it hardware or software resources), using intelligent ways to monitor and manage the resources, and applying different methods to get the best quality of service guarantees without causing any violation in the service-level agreement (SLA). These are some of the important challenges facing the cloud providers.

1.1 Dynamic resource provisioning

In cloud computing, *elasticity* refers to the degree of automatic adaptation in resource provisioning in response to the continuous changes in the customer's workload and demands. This is basically achieved by automatically, *scaling up* or *scaling down* the resources assigned to a certain

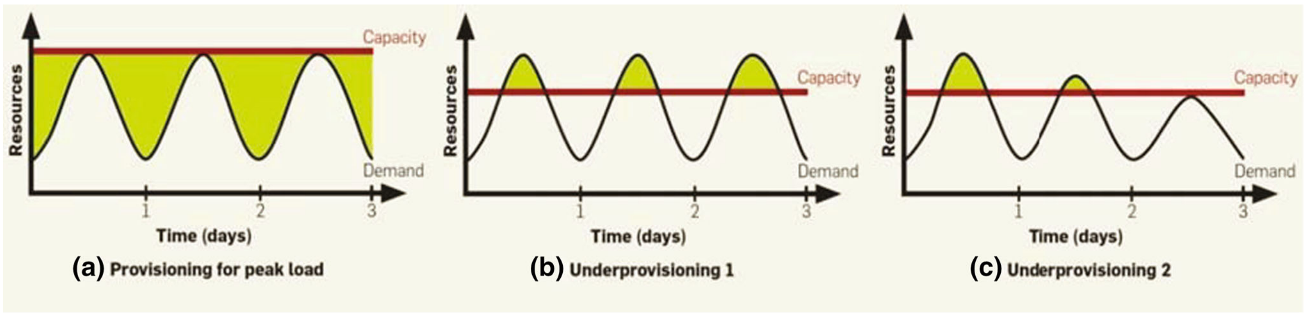


Fig. 1 The overprovisioning and underprovisioning problems [1]

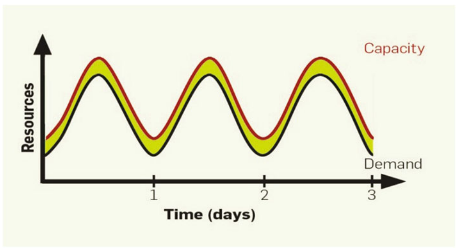


Fig. 2 Dynamic resource provisioning for the current customers demands over time [1]

customer. Such mechanism matches as closely as possible the available resources with the current demands of the customers [8]. So, elasticity can be simply put as avoiding both the overprovisioning and the underprovisioning problems leading to good dynamic resource provisioning [1].

The *overprovisioning* problem can occur when the reserved resources for a certain customer exceed its demands as shown in the left part of Fig. 1. In the figure, the red line represents the allocated resources based on the peak load, which is correctly estimated leading to zero SLA violations. However, without elasticity through non-peak times, resources are wasted (the shaded area represents the wasted resources). On the other hand, the *underprovisioning* problem can occur when the reserved resources are not suitable to the current customer's demands. This problem causes SLA violations leading to losing revenue and customers. The underprovisioning problem can have multiple cases as shown in the middle and right parts of Fig. 1, respectively. In these figures, the shaded areas represent the SLA violations, which can vary over time depending the customer's demands [1].

To address the resource provisioning problems discussed in the previous paragraph, dynamic resource provisioning is utilized [1]. The resource capacities are scaled up or down over time depending on the customer's demands as shown in Fig. 2. Such policies are already employed in several systems [11,13]. One famous example is Amazon's Web Services (AWS) used on Amazon's EC2 system.[1]

1.2 Cloud resource monitoring

Resource and application monitoring in cloud computing is very important for both the cloud customers and providers. The cloud customers need to monitor their applications and the reserved resources for them to make sure their tasks are working without any problem and to be able to make more accurate future requests for resources and thus avoid paying for resources that will not be used. On the other hand, the cloud providers need to monitor the resource utilization and collect information about it to make better provisioning decisions and prevent any violation in the SLA. Moreover, the resource monitoring encompasses collecting information about resources and using this information to make decisions related to other components in the cloud environment [9].

Amazon's AWS provides to its customers a feature called CloudWatch.[2] This feature offers the customers the ability to monitor their applications on AWS and the resources assigned to them. Also, this feature provides the system administrators and developers the ability to track and collect different performance metrics, react immediately, and gain insights to maintain their applications and businesses seamlessly. CloudWatch can monitor custom metrics produced by cloud customers for their applications and services and can also monitor AWS resources such as Amazon Relational Database Service (RDS) and Amazon EC2.

In this study, we focus on the dynamic resource monitoring and provisioning problems in Infrastructure as a Service (IaaS) cloud computing. The objective is to get the best resource utilization and the shortest completion time of the customers' tasks while minimizing the cost such as reducing energy consumption. So, we need some mechanisms to reduce the number of virtual machines (VMs) migrations and reduce the number of active physical machines to reduce the energy consumption. Moreover, we need some mechanisms to manage and formulate cloud customers' requests to increase resource utilization.

The rest of this paper is organized as follows. Section 2 presents a literature review. In Sect. 3, we present our system

[1] http://aws.amazon.com/autoscaling.

[2] https://aws.amazon.com/cloudwatch.

model and evaluate it in Sect. 4. Finally, we conclude in Sect. 4.1.

2 Related works

As mentioned the previous section, the dynamic resource provisioning mechanism has the ability to adapt in an automatic manner to the changes occurring on the customers' workload and demands at any time. This is achieved by scaling up or down the resources through a set of algorithms, rules and policies to match as closely as possible the available resources with the current workload and demands, thus, avoiding both the overprovisioning and the under-provisioning problems [1,8,11,13].

The resource and application monitoring in the cloud computing is very important for both the cloud provider and the customers. Several mechanisms are suggested for the customers to monitor their applications and reserved resources to make sure their tasks work according to the SLA that has been agreed upon in advance between the cloud provider and the customers in addition to being able to make more accurate future requests for resources. As for the cloud providers, the monitoring mechanisms allow them to monitor their resource utilization in order to maintain their customers' applications running seamlessly and avoid any SLA violation [9].

There are several cloud computing frameworks that have specific mechanisms to monitor and provision their resources. In this section, we will describe some of the most popular frameworks in the field of cloud computing and their mechanisms.

The open cloud computing interface (OCCI) Framework by Venticinque et al. [15] supports provisioning, monitoring and auto configuration for the cloud resources to satisfy the application requirements at infrastructure level (IaaS cloud computing service model). The OCCI includes a set of protocols and API with independent vendor and neutral platform, which solve different problems in management of common tasks with satisfaction of integration, portability and interoperability requirements including autonomic scaling, deployment and monitoring.

The authors in [15] proposed a new architecture by re-implementing the OCCI depending on the Cloud Agency with JADE agent platform as an execution environment which supports agents' communication using agent communication channel (ACC) and a set of implementation protocols. The agents communicate with each other through the standard agent communication language (ACL) over HTTP. The OCCI have four abstract interfaces including resource management, monitoring, reconfiguration and brokering which can be implemented by developers to extend Cloud Agency with new algorithms for each abstract interface as plug-ins of Cloud Agency to be used automatically by agents over their general interfaces. The extended OCCI uses Message Transfer Protocol (MTP) to translate OCCI HTTP, although the OCCI-MTP permits the communication among the Cloud Agency. The clients manage two types of requests: synchronous and asynchronous. The main tasks of the OCCI-MTP is verifying the request correctness, translating the REST request to ACL-Message and forwarding the request from clients to the comprehend agent. For the experiments part, the authors developed and designed a set of API that used to communicate and request handler between clients and Cloud Agency services (provisioning, monitoring, management and auto configuration), and the result of experiments is hopeful to continue with suggested improvements and developments for the current implementation.

The CREATE framework architecture by Duong et al. [6], is an extensible and reusable web service based framework with a set of resource provisioning algorithms for dynamic resource provisioning and adaptation on-demand under the IaaS cloud computing service model. The framework was built to interact with different local resource managers, and to manage several sets of resources assimilated from different cloud providers. The design of their framework is flexible and easily extensible.

CREATE has three main components including resource sets (RSS) monitoring and managements service (RMM), RRSs adaptation service (RA) and Cloud clustering service (CC). The first component RMM service is designed to manage the resources configuration and to collect information about the resources and status of RSS workload, although the RA service is designed to manage at the same time many of RRSs, the last component CC service is designed to obtain or release resources from different cloud providers by flexible REST-based interface. The authors in CREATE framework present six resource adaptation algorithms: Job-Number-Parallel (JNP), Job-Number-Sequential (JNS), Job-Time-Parallel (JTP), Job-Time-Sequential (JTS), Hybrid-Parallel (HP) and Hybrid-Sequential (HS).

The main objectives of resource adaptation algorithms in CREATE framework is to minimize the jobs' completion time and at the same time reduce the cost and overhead of resources. The authors in evaluation methodology that used many different realistic workload scenarios to evaluate and assess the resource adaptations algorithms, the result of their evaluation proves the robustness and effectiveness of some algorithms.

The Aneka system proposed by Vecchiola et al. [14] is a .NET-based application platform-as-a-Service (PaaS) for cloud computing, which offers a set of APIs and runtime environment applications through multiple programming models, and deploying them on private and public cloud computing platforms like Amazon EC2 and GoGrid. The Aneka system offers a resource provisioning mechanism based on SLA orientation. This mechanism works as if the system receives new tasks from cloud customers. It estimates the time needed to complete these tasks in order to finish with current available resource and compare it with SLA deadline time. If the estimation time to complete new tasks is smaller than the SLA deadline time, then the system continues working, otherwise, the system scales up the cloud resources and continues work to avoid any SLA violation.

The Elastic-JADE system, proposed by Siddiqui at al. [12], has three parts (user, local machine and Amazon EC2 cloud), which allows it to automatically scale (up or down) Amazon EC2's resources through JADE platform when heavily loads occur in the local platform.

When the resources in local machine exceed a certain threshold (heavily loads occur), the local resources (processing and memory) expend through management agent component in both platform (local machine and Amazon EC2 cloud). The management agent in local machine is responsible for monitoring the complete system in both platforms, communicate with the management agent in Amazon EC2 cloud and send command messages for scale up or down of resources depending on system load.

The Amazon Web Services (AWS) provide customers with an auto-scaling mechanism to scale up or down the Amazon EC2 virtual machines automatically according to the workload and demand changes. This mechanism can save the customers money and improve the performance of Amazon EC2 virtual machines.

Moreover, AWS provides the CloudWatch feature, which offers the customers the ability to monitor their applications and assigned resources on AWS, in addition to providing the system administrators and developers the ability to track and collect performance metrics, react immediately, and gain insights to maintain their applications and businesses seamlessly. CloudWatch can monitor custom metrics produced by cloud customers for their applications and services and can also monitor other resources such as Amazon RDS DB and Amazon EC2.

The Coasters framework proposed by Hategan et al. in [7] offers both usability and performance goals by building a uniform framework that allows multiple services to access cloud and grid resources. In other word, it is a uniform access and resources provisioning for cloud and grid computing systems.

The Coasters system is coupled with the Swift scripting language to support parallel execution of different jobs. One main advantage of the Coasters system is usability as it does not require users to log into remote systems and prepare services or configure compute nodes. Moreover, since the coasters system defines a real path between the client and the service provider, it attains noticeable performance gains.

Bonvin et al. [3] proposed the scattered autonomic resources (Scarce) framework, a multi-agent platform to dynamically manage the resources using an economic-based approach. The agents run on the server side and are responsible for managing the resources and continually checking the systems health.

3 Proposed system

In this section, we present the details of our proposed dynamic resource provisioning and monitoring (DRPM) system. The DRPM system is a multi-agent system that takes into account several factors such as the limitation of cloud provider resources, customers satisfaction and power consumption.

The DRPM system has three main phases: monitoring, analysis and execution. The objective of the first phase is to monitor the cloud provider's resources such as computational power (henceforth referred to as CPU), memory (henceforth referred to as RAM), permanent storage (henceforth referred to as storage), network bandwidth (henceforth referred to as BW), etc., in addition to monitoring the customers' tasks and demands. The raw data from this phase are analyzed in the second phase to improve resource management by generating optimized decisions regarding the VM's specification to be allocated for each customer's request depending on the SLA. These decisions are sent to the third phase for execution.

3.1 System architecture

As shown in Fig. 3, the DRPM system consists of two main layers: the customers' applications layer and cloud provider's resources layer. The provider's resources include a set of datacenters each with a large number of physical machines (hosts). Each host has specific characteristics (in terms of CPU, RAM, storage and BW) to host multiple VMs. Moreover, it has monitoring sensors to measure the utilization of its resources and the execution of the customers' applications on it. The applications layer consists of a set of customers having several jobs possibly spanning multiple VMs. Each job is associated with specific performance goals specified in the SLA (e.g., time to complete their tasks, number of requests for specific time periods).

The multi-agent components include a global utility agent and a set of local utility agents. While the global utility agent has the classical role of the "central broker" allowing it to manage all of the system resources, the local agents are assigned to each customer with the objective of improving the resource utilization without causing SLA violations. One of the advantages of using multiple agents is to allow for certain optimizations that can conducted "locally" (i.e., at the customer level) without burdening the global agent with the low-level details of these optimizations. Here we discuss one such example where each

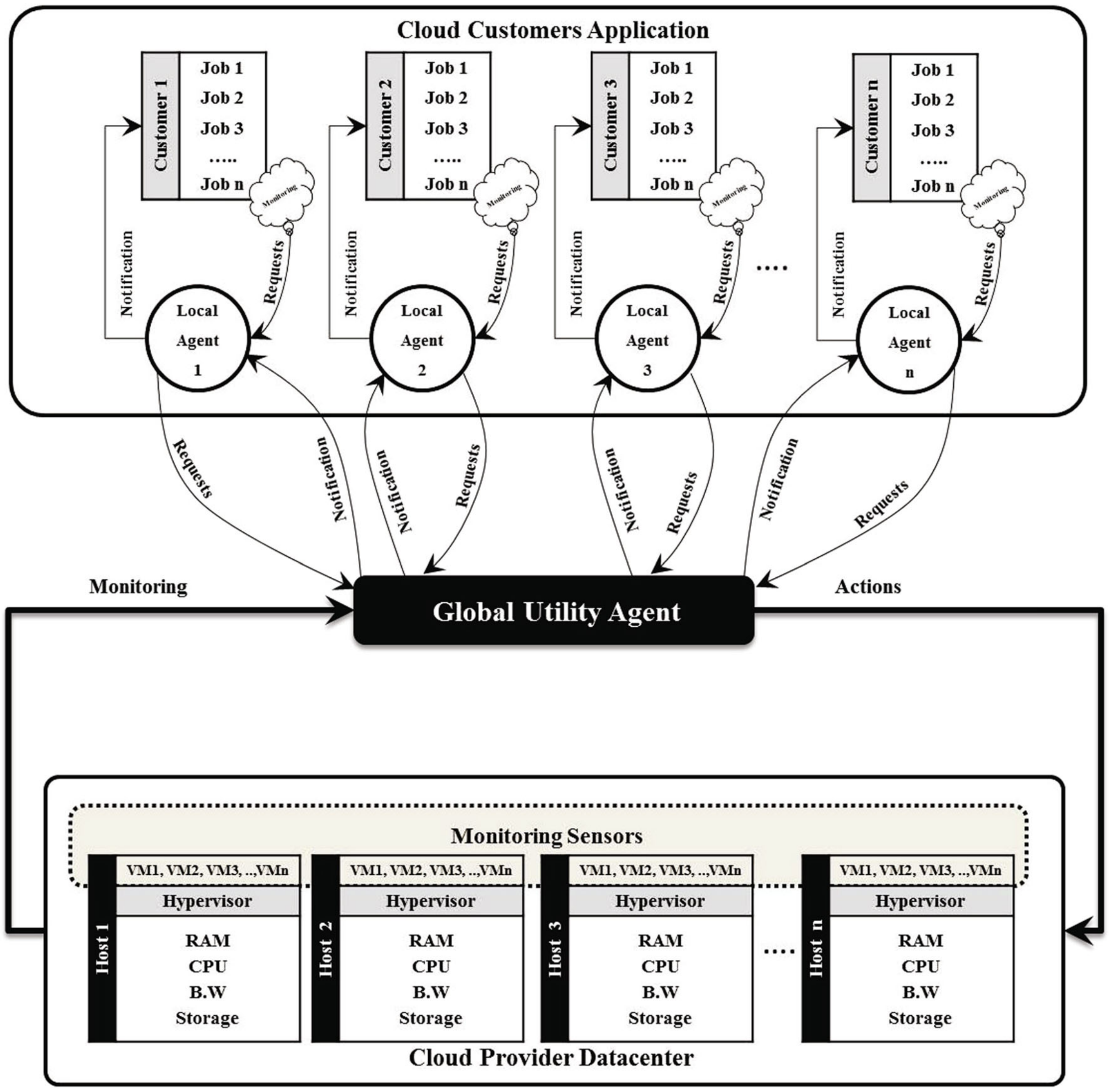


Fig. 3 Architecture of the multi-agent based dynamic resource provisioning and monitoring (DRPM) system

customer is assigned a local agent responsible for *reformulating* each of its requests in order to better match its actual usage of the assigned resources. This can be achieved using any prediction technique. Here, for simplicity purposes, we show that even the simplest of techniques can be very useful. Based on a regression analysis of its history, the local agent can estimate the amount of resources that will actually be used (thus, avoiding the over-provisioning problem) without causing an SLA violation (thus, avoiding the under-provisioning problem). The component responsible for this task is called the Request Reformulation (RR) algorithm. We show in the experiments in Sect. 4.2 that the introduction of the RR significantly enhance the system's performance.

After reformulating the requests, the local agent sends the new requests to the global agent, which is responsible for the actual provisioning of the resources by communicating with the hypervisors of the physical machines in the different datacenters under the cloud provider's control. The global utility agent should take into account the limitations on the provider's resources in addition to various performance metrics such as power consumption. In the following paragraphs, more detailed discussion of the local utility agents as well as the global utility agent is presented.

3.2 The local utility agents

The main responsibility for each local utility agent is to perform "localized" optimizations on the customer level without burdening the global agent with the details of such optimizations. In the example we present here, each local agent monitors the customer assigned to it and builds a history of its resource utilization, basically, by keeping track of how much of its requested resources it actually uses. Once a new

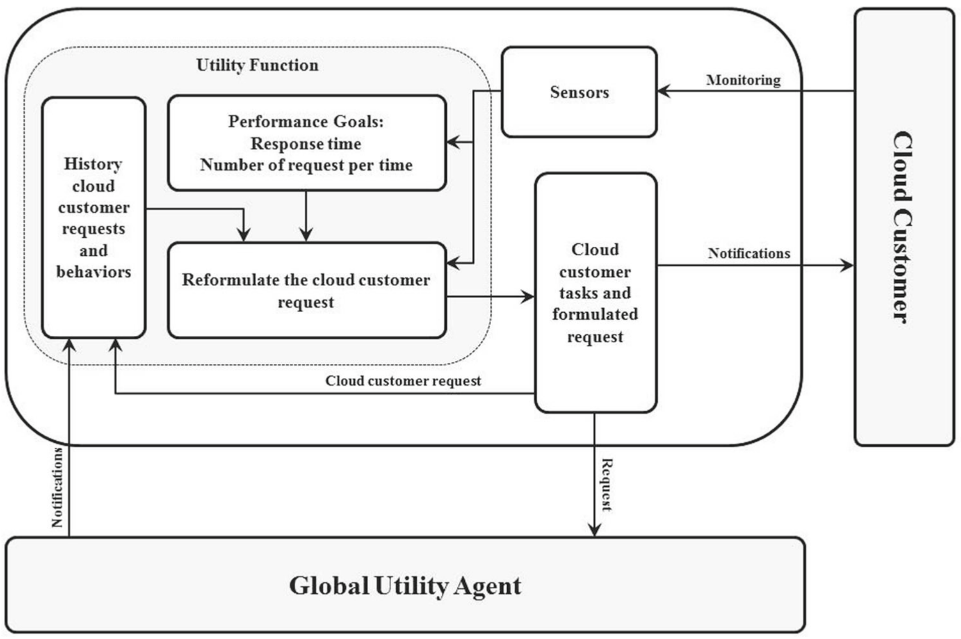

Utility Function
History cloud customer requests and behaviors
Performance Goals: Response time Number of request per time
Reformulate the cloud customer request
Sensors
Monitoring
Cloud customer tasks and formulated request
Notifications
Cloud Customer
Cloud customer request
Notifications
Request
Global Utility Agent

Fig. 4 Local utility agent architecture

request for a VM with specific characteristics is received from the customer, the local utility agent utilizes the history it has built in reformulating the customer's requests to better match its actual usage. The objective of such reformulation is to avoid both the overprovisioning and the underprovisioning problems while taking into account other issues such as the number of request per time unit, response time, etc. The output of the reformulating step is sent to the global utility agent which is responsible for the actual provisioning of the resources. The architecture of a local utility agent is depicted in Fig. 4 whereas Fig. 5 depicts the flowchart of a local utility agent.

The local utility agent use simple regression analysis to reformulate cloud customer requests by predicting the value of the wasted resources depending on customer request. Regression analysis [5] can be used to predict the value of a random variable Y whose value depends on the value of an independent variable X by using least squares linear regression equation $\widehat{y} = b_0 + b_1 x$ where b_0 is a constant which can be computed as $b_0 = \overline{y} - b_1 \times \overline{x}$, b_1 is the regression coefficient which can be computed as $b_1 = \sum_i [(x_i - \overline{x})(y_i - \overline{y})] / \sum [(x_i - \overline{x})^2]$, and $\widehat{y}$ is the *predicted* value of the dependent variable.[3] In the following, we show how can these equations be used to estimate the amount of resources that will actually be used based on the history of requests made by the same user. For the sake of simplicity, the discussion below pertains to one specific resource (CPU). Other resources such as RAM, storage and BW can be estimated similarly.

The following equation is used to estimate the value of the CPU that will actually be used by the customer as a function of the current as well as the previous requests made by the same customer.

$$EC = (RC - WC) + B,$$

where EC, RC and WC represent the estimated, currently requested and wasted CPU, respectively. As for B, it is a parameter whose value determines whether to act conservatively or aggressively based on the estimated values of the resources to be used. Note that, in this work, overestimating the amounts of resources to be used leads to wasted resources whereas underestimating them leads to SLA violations. In general, wasting some resources is a price most cloud providers are willing to pay to avoid SLA violations. However, this need not be the case all the time. By setting the value of B properly, the cloud providers are given control over the trade-off between saving resources and incurring SLA violations. Large values of B are suitable for conservative behavior since it inflates the estimated values and leaves little room for errors in the estimates (SLA violations). On the other hand, for the scenarios in which the cloud provider cares more about saving resources than SLA violations, small values of B are more suitable. Some of the experiments conducted is dedicated to the different potential values of B and their effect on the performance of the system (See Sect. 4).

Adapted from [5], the following equation is used to compute the wasted CPU.

$$WC = RC \times C_1 + C_0,$$

3 http://stattrek.com/regression/linear-regression.aspx?Tutorial=AP.

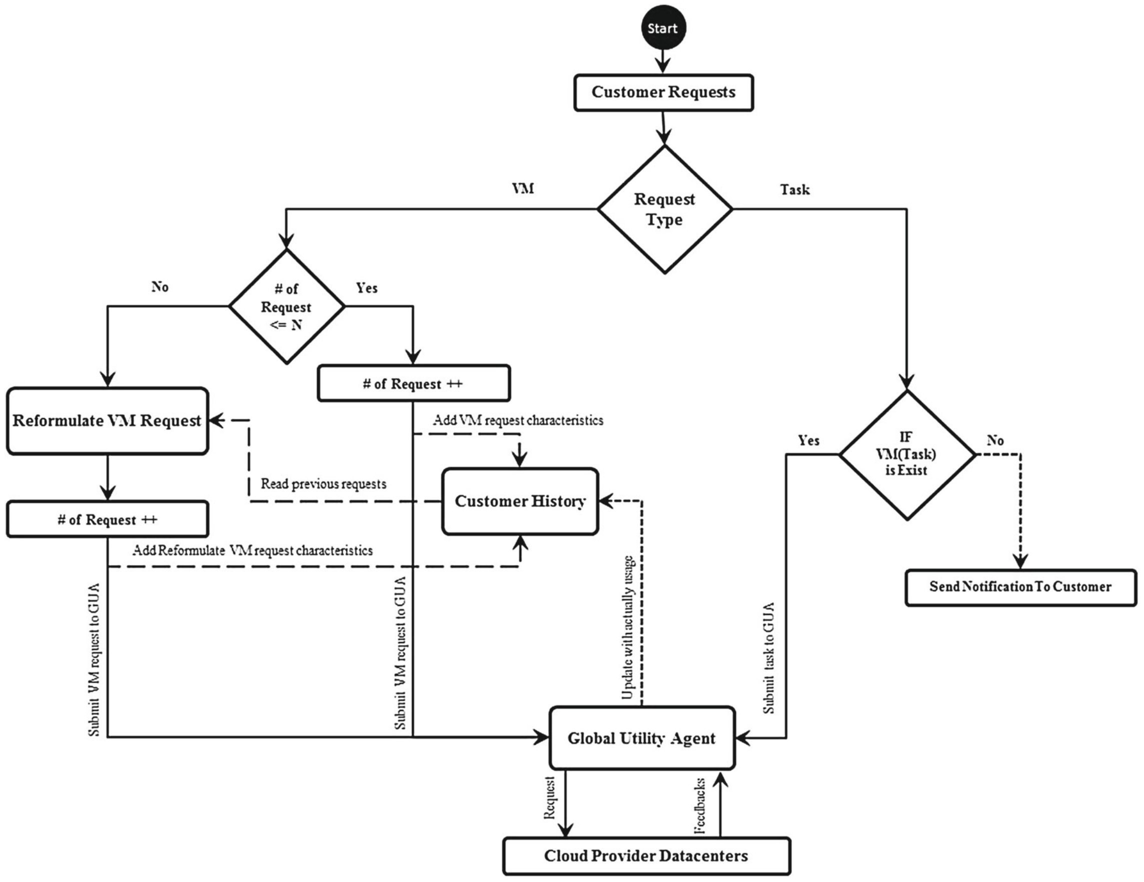


Fig. 5 Flowchart of local utility agent

where C_1 and C_0 are computed as follows.

$$C_1 = \frac{\sum_{i=1}^{C}[(PRC_i - \overline{PRC}) \times (WC_i - \overline{WC})]}{\sum_{i=1}^{C}[(PRC_i - \overline{PRC})^2]},$$
$$C_0 = \overline{WC} - C_1 \times \overline{PRC}.$$

In the above equations, $\overline{X}$ represents the average of X and PRC is an array containing the latest C requests made by the same customer.

3.3 The global utility agent

The main responsibility for the global utility agent is to manage the reformulated requests coming from the local utility agents. This constitutes communicating with the hypervisors of the physical machines to provision the requested resources, notifying the local utility agents of the details of the provisioned resources and notifying the hypervisors of the physical machines upon the completion of a customer's job to reclaim the allocated resources. While determining how to allocate resource, the global utility agent must take into account several performance goals such as reducing the number of active physical machines, reducing the number of VM migrations, etc. Figure 6 depicts the architecture of the global utility agent.

3.4 The host fault detection (HFD) algorithm

In general, a VM *migration* occurs when its hosting physical machine is facing an overloading problem or an underloading problem. If a physical machine is being largely underutilized, the VM placement algorithm [such as inter quartile range (IQR), local regression (LR), local regression robust (LRR), median absolute deviation (MAD) and static threshold (THR)] migrates *all* hosted VMs from the underloaded host to another one so that the underloaded hosts can be turned off to save on electricity. On the other hand, if a physical machine is being overloaded, the VM placement algorithm calls one of VM selection algorithms (such as Maximum Correlation (MC), Minimum Utilization (MU), Random Selection (RS) and Minimum Migration Time (MMT) to determines which VMs to migrate [2].

In this study, the global utility agent uses a novel VM selection algorithm called Host Fault Detection (HFD). This algorithm determines which VMs to migrate when the hosted

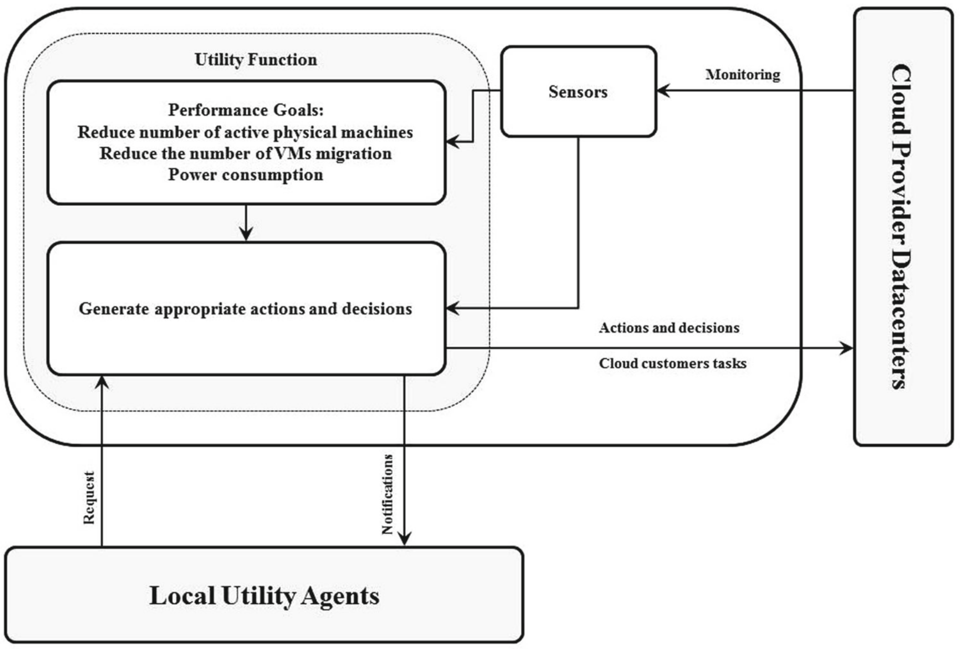


Fig. 6 Global utility agent architecture

physical machine is being overloaded depending on the cause of the overload. It then chooses the VM with the maximum effect on the cause of the overload. For e.g., if the RAM is the cause of the overload, then HFD selects the VM with the max allocated RAM to be migrated. It uses the power-aware best fit decreasing (PABFD) VM placement algorithm to determine where to migrate VMs. Despite its simplicity, the HFD algorithm outperforms other selection algorithm as shown in the following section.

4 Experiments and results

To evaluate the performance of the proposed DRPM system (along with its novel RR and HFD algorithms), an extended version of CloudSim [4] called CloudExp [10] is employed. CloudSim is one of the earliest and most widely accepted simulation environments for the cloud computing paradigm.

The first part of the experiment is concerned with the RR algorithm and its performance. For this part, we customers who are conservative with respect to the way they make their requests. Such customers always request resources exceeding their expected needs to avoid resorting to the generally expensive task of requesting additional resources during the execution of the job. We evaluate the effect of the RR algorithm by measuring SLA violations in addition to free and wasted capacities. Similar to the discussion in last section, the focus here is on one specific resource, CPU (measured in MIPS), to simplify the presentation of ideas and results.

Table 1 VMs characteristics

Number of VMs	50
VMs types	4
MIPS of CPU	[2500, 2000, 1000, 500]
Number of PEs	1
RAM capacity	[870, 1740, 613]
Bandwidth	100 Mbit/s
Storage capacity	2.5 GB

The second part of the experiments is concerned with the HFD VMs selection policy. We compare its performance with other VMs selection polices like MC, MU, RS and MMT and evaluate it on several aspects such as energy consumption, number of VM migrations, number of host shutdowns, and SLA violations.

4.1 Experiments setup

We use for each experiment the same settings used in [2]. We use four types of virtual machines and two types of physical machine deployed in a single datacenter. All experiments are carried out under the dynamic workload with 50 cloudlets for a simulation period of 24 hours in which the scheduling interval (new resource usage time) is 300 seconds. Tables 1, 2, 3 and 4 show the characteristics of the VMs, the physical machines, the datacenter and the cloudlets, respectively.

Table 2 Physical machines characteristics

Number of Hosts	50
VMs types	2
MIPS of CPU	[1860, 2660]
Number of PEs	2
RAM capacity	[4096]
Bandwidth	1 Gbit/s
Storage capacity	50 GB

Table 3 Datacenter characteristics

System architecture	x86
Operating system	Linux
Virtual machine monitor (VMM)	Xen
Time zone	10.0
Cost per processing	0.03 cents
Cost per memory	0.05 cents
Cost per storage	0.001 cents
Cost per bandwidth	0.01 cents

Table 4 Cloudlets characteristics

Number of cloudlets	50
Cloudlet length	216000000
File size	300
Output size	300
Number of PEs	1

4.2 Request reformulation (RR) algorithm experiments

As mentioned in Sect. 3, the RR algorithm uses regression analysis to estimate the amount of resources a customer would actually use based on the current and past resource requests from the same customer and how much of them were actually used. This section shows several experimentation result on the RR algorithm. We assume that the customers are conservative in their requests. This means that a customer always requests more resources than it will actually use to avoid resorting to the generally expensive task of requesting additional resources for a customer's job during the execution of the job. All experiments are conducted under dynamic workload with different values for B ranging from -15 to 15 %. In each one of these experiments, the value of B is set at the beginning and it remains fixed throughout the experiment. However, B is not necessarily a fixed parameter; it can be set dynamically. We perform an additional experiment with adaptive value of B computed as the average of previous MIPS violations normalized against the range of requested MIPS.

The results of this experiment are shown in Figs. 7, 8 and 9 highlighting the advantages and disadvantages of using the proposed system for different values of B (including the adaptive one). Figure 7 shows the advantages of the proposed system manifested in MIPS savings. Figures 9 and 8, on the other hand, show the cost of the proposed system manifested in the total wasted MIPS (Fig. 9) as well as the SLA violations (Fig. 8). Specifically, the figure shows the percentage of requests for which an SLA violations related to insufficient MIPS occur (henceforth referred to as MIPS violations).

To study the gains from using the DRPM system, consider Fig. 7, which shows the percentages of MIPS savings for different values of B. These percentages are computed by dividing the amount of saved MIPS over the difference between the amount of requested MIPS and the amount of actually used MIPS. For example, consider a customer requesting a virtual machine with 2000 MIPS and the RR algorithm reformulates the request to become 1900 MIPS, if the customer actually uses 1800 MIPS, then it can be viewed that RR reformulated the request to attain $\frac{1900-1800}{2000-1800} = 50\,\%$ of the potential savings. The figure shows that the proposed system steadily achieves high percentages of possible savings with large variations in the value of B having minimal impact. The figure also shows that the best savings are achieved for

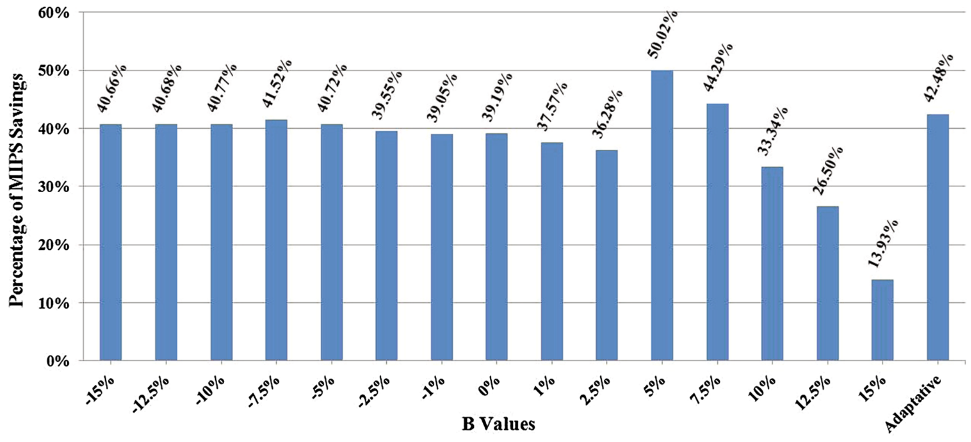


Fig. 7 The percentages of MIPS savings for different values of B

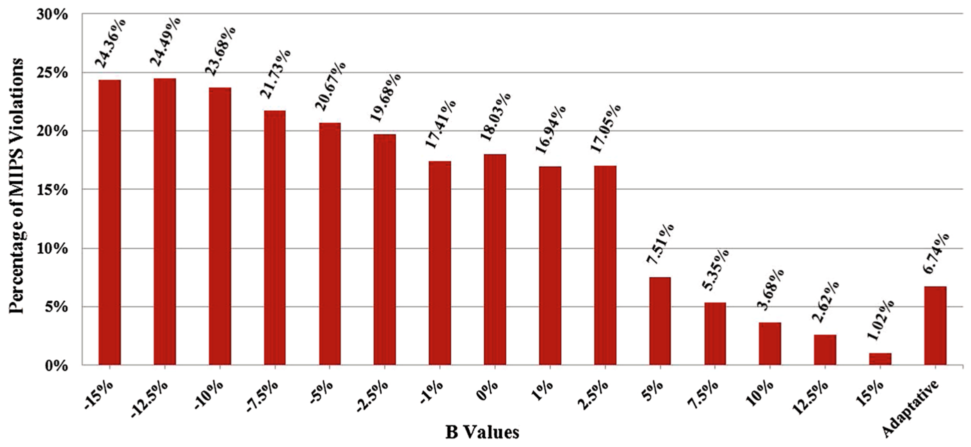


Fig. 8 The percentages of MIPS violations for different values of B

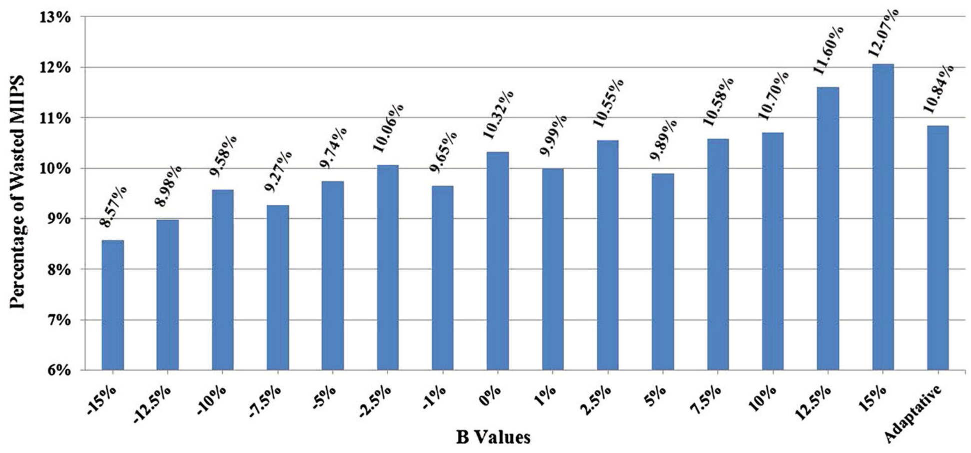


Fig. 9 The percentages of wasted MIPS for different values of B

$B = 5\,\%$ with an average of about 50 % MIPS savings in each request. This is a huge percentage compared with allocation algorithms that do not employ RR or any similar algorithm to reformulate the customers requests to better match their actual usage.

To build confidence in our results, we performed a t-test to test the null hypothesis that the average wasted CPU without using the RR algorithm is the same as the average wasted CPU with using the RR algorithm.

The average improvement ($\mu = 214.12, \sigma = 1.918$ and $N =$ 41,637) was significantly greater than zero, $t(7) = 111.6$, two-tail $p = 0$, providing evidence that the RR algorithm is in fact effective in reducing wasted CPU. A 95 % C.I. about the average improvement is (210.37, 217.89).

For this test, we focus on the experiments with the adaptive way of computing the value of B. Similar results are obtained for other values of B.

After taking a look at the advantages of the proposed system, it is time to look into the price at which these gains come by considering Figs. 8 and 9. Figure 8 shows the percentage of MIPS violations, which is the main cost metric (disadvantage) of using the DRPM system. It can be clearly seen that with the increase of the value of B, the percentage of MIPS violations decreases, which is in accordance with the arguments of Sect. 3. Following the same argument, having a conservatively large value of B will definitely increase the amount of wasted resources (see Fig. 9).

Figures 7, 8 and 9 show the trade-off between the saved resources and the SLA violations in an evident way. Now, it is up to the cloud provider to balance these two factors to serve its best interest. Note that, from these figures, adaptively setting the value of B gives a similar behavior to statically setting it to around 7.5 %, which represents a very good balance between the gains (in terms of saved MIPS) and the losses (in terms of SLA violations).

4.3 Host fault detection (HFD) policy experiments

The experiments in this section are dedicated to study the performance of the Host Fault Detection (HFD) VM selection policy and compare it with other policies. HFD as well as the other policies are tested under dynamic workload and different VM allocation polices. Figure 10 shows the result of comparisons between HFD and other VMs selection polices. Specifically, we consider the following four selection policies: the Maximum Correlation (MC) policy, the Minimum

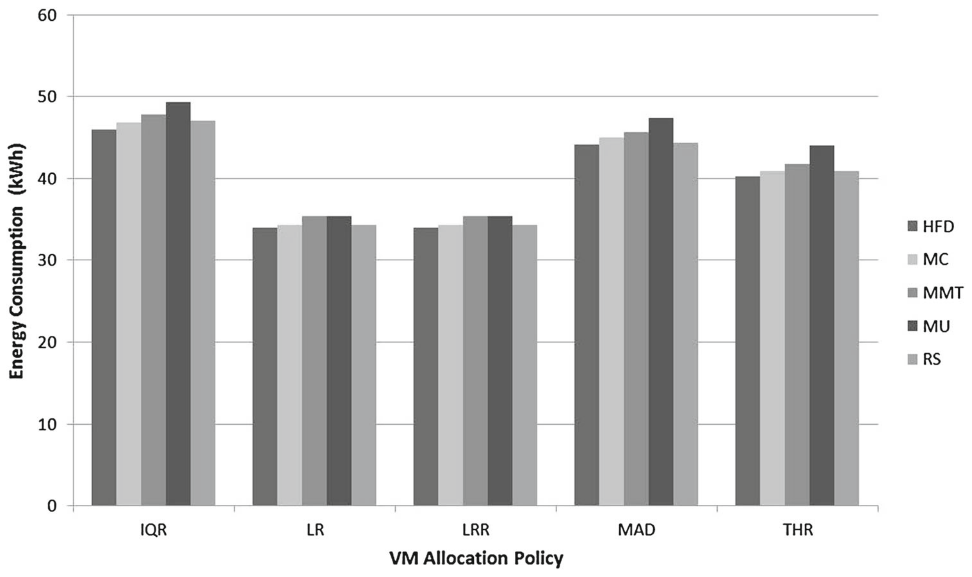


Fig. 10 Comparison of the VM selection policies in terms of energy consumption (kWh)

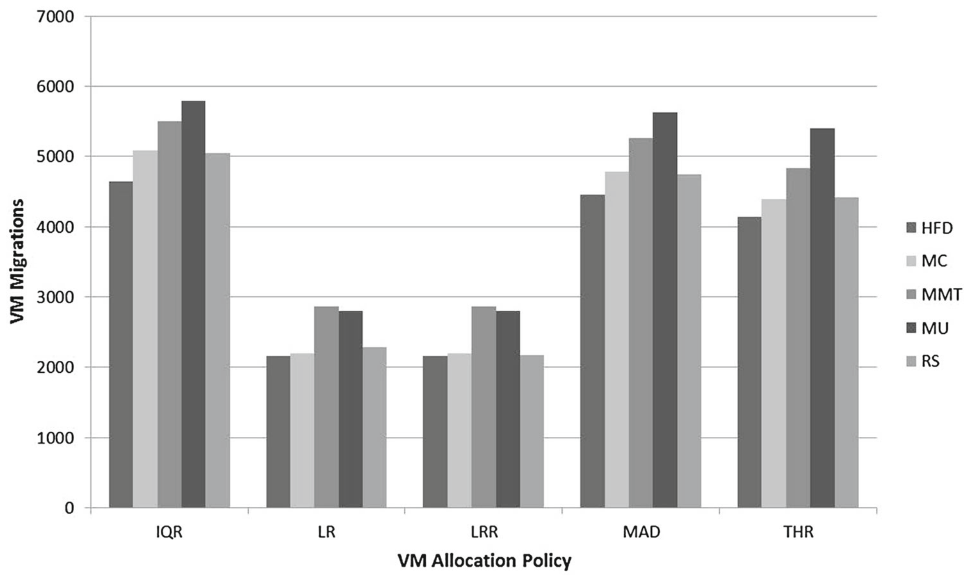


Fig. 11 Comparison of the VM selection policies in terms of VM migrations

Migration Time (MMT) policy, the Minimum Utilization (MU) policy and the Random Selection (RS). We compare the different selection policies under consideration with five different allocation policies: Inter Quartile Range (IQR), Local Regression (LR), Local Regression Robust (LRR), Median Absolute Deviation (MAD) and Static Threshold (THR). The performance of the selection policies is evaluated from different aspects. Specifically, we consider energy consumption, VM migrations, SLA violations and host shutdowns.

Figure 10 shows the total energy consumption of the five selection policies under consideration. The five parts of the figure (parts (a) through (e)) are generated considering the IQR, LR, LRR, MAD and THR allocation policies, respectively.

From these figures, it can be seen that HFD always outperforms all other selection policies under all settings with MC and RS being the closest to HFD. However, the advantage of using HFD over the other policies is sometimes small as it does not exceed 2 % for MC, 4 % for MMT, 9 % for MU and 2 % for RS.

Figure 11 shows the number of VM migrations of the five selection policies and the five allocation policies under consideration. Similar to the energy consumption comparison, HFD outperforms all other selection policies under all settings with MC and RS being the closest to HFD. However, the advantage of using HFD over the other policies is more obvious here as it reaches 9 % for MC, 25 % for MMT, 23 % for MU and 8 % for RS.

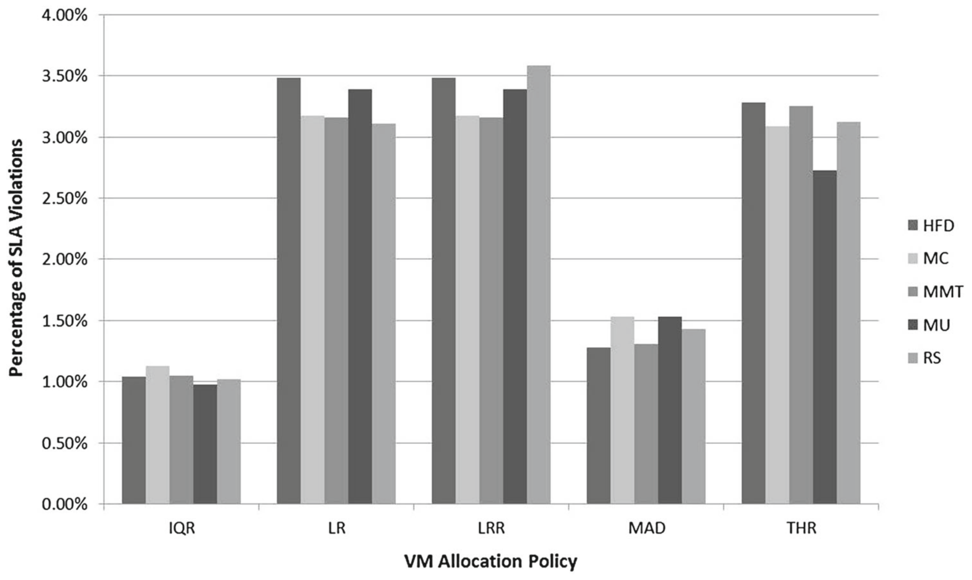


Fig. 12 Comparison of the VM selection policies in terms of SLA violations

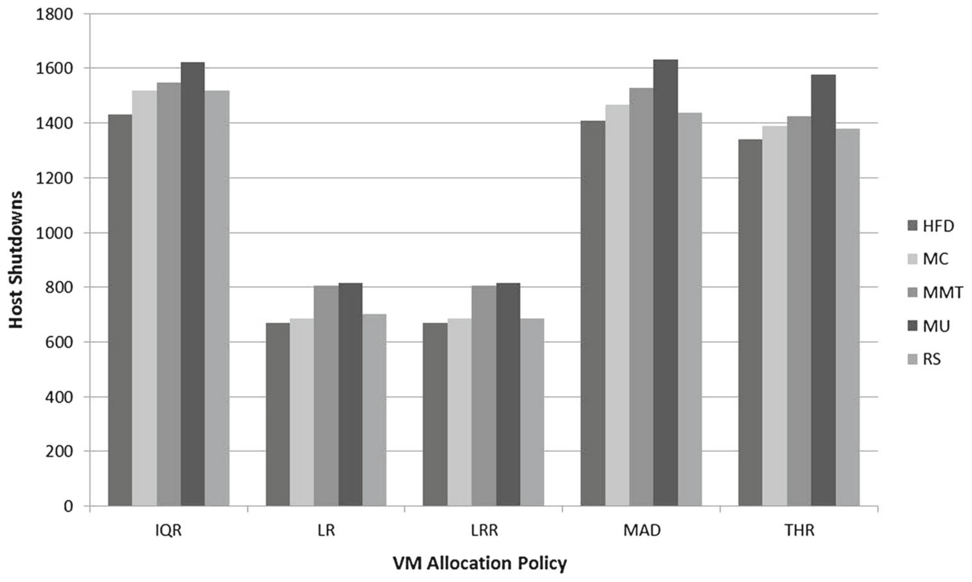


Fig. 13 Comparison of the VM selection policies in terms of hosts shutdowns

Figure 12 shows the percentage of SLA violations of the five selection policies and the five allocation policies under consideration. Unlike the previous comparisons, HFD outperforms all other selection policies under only the MAD allocation policy. On the other hand, HFD is being outperformed by all other selection policies under the LR and THR allocation policies. As for the remaining allocation policies (IQR and LRR), HFD outperforms some of the selection policies under consideration while being outperformed by others.

Figure 13 shows the number of host shutdowns of the five selection policies and the five allocation policies under consideration. Similar to the energy consumption and number of migrations comparisons, HFD outperforms all other selection policies under all settings with MC and RS being the closest to HFD. Specifically, the advantage of using HFD over the other policies reaches 6 % for MC, 17 % for MMT, 18 % for MU and 6 % for RS.

Finally, it is worth mentioning the noticeable gain of LR and LRR over the other allocation policies regardless of the selection policy. This is true for three of the four metrics under consideration. The only exception is the percentage of SLA violations for which IQR and MAD produce the lowest percentages.

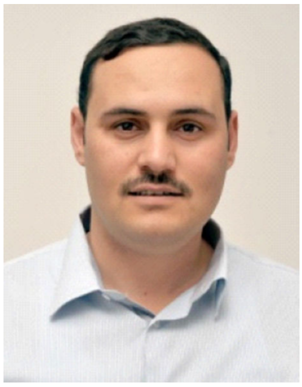
5 Conclusion

This study presented the Dynamic Resources Provisioning and Monitoring (DRPM) system, a multi-agent system to manage the cloud provider's resources while taking into account the customers' quality of service (QoS) requirements as determined by the service-level agreement (SLA). Moreover, DRPM includes a new Virtual Machine (VM) selection algorithm called the Host Fault Detection (HFD) algorithm. The proposed DRPM system is evaluated using the CloudSim tool. The results show that the DRPM system allows the cloud provider to increases resource utilization and decreases power consumption while avoiding SLA violations.

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
2. Beloglazov, A., Buyya, R.: Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr. Comput. **24**(13), 1397–1420 (2012)
3. Bonvin, N., Papaioannou, T.G., Aberer, K.: Autonomic SLA-driven provisioning for cloud applications. In: Proceedings of the 2011 11th IEEE/ACM international symposium on cluster, cloud and grid computing, IEEE Computer Society, pp. 434–443 (2011)
4. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software **41**(1), 23–50 (2011)
5. Chatterjee, S., Hadi, A.S.: Regression analysis by example. Wiley, Hoboken (2013)
6. Duong, T.N.B., Li, X., Goh, R.S.M.: A framework for dynamic resource provisioning and adaptation in iaas clouds. In: Proceedings of the IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), 2011, pp. 312–319 (2011). IEEE
7. Hategan, M., Wozniak, J., Maheshwari, K.: Coasters: uniform resource provisioning and access for clouds and grids. In: Proceedings of the Fourth IEEE International Conference on Utility and Cloud Computing (UCC), pp. 114–121 (2011). IEEE
8. Herbst, N.R., Kounev, S., Reussner, R.: Elasticity in cloud computing: what it is, and what it is not. In: Proceedings of the 10th International Conference on autonomic computing (ICAC 2013), San Jose, CA (2013)
9. Huang, H., Wang, L.: P&p: a combined push-pull model for resource monitoring in cloud computing environment. In: Proceedings of the Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, pp. 260–267 (2010). IEEE
10. Jararweh, Y., Jarrah, M., Kharbutli, M., Alsaleh, M.N., Al-Ayyoub, M.: CloudExp: a comprehensive cloud computing experimental framework. Simul. Model. Pract. Theory **49**, 180–192 (2014)
11. Marshall, P., Keahey, K., Freeman, T.: Elastic site: using clouds to elastically extend site resources. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, EEE Computer Society, pp. 43–52. I (2010)
12. Siddiqui, U., Tahir, G.A., Rehman, A.U., Ali, Z., Rasool, R.U., Bloodsworth, P.: Elastic jade: dynamically scalable multi agents using cloud resources. In: Proceedings of the Cloud and Green Computing (CGC), 2012 Second International Conference on, pp. 167–172 (2012). IEEE
13. Vaquero, L.M., Rodero-Merino, L., Buyya, R.: Dynamically scaling applications in the cloud. ACM SIGCOMM Comput. Commun. Rev. **41**(1), 45–52 (2011)
14. Vecchiola, C., Chu, X., Buyya, R.: Aneka: a software platform for.net-based cloud computing. High Speed Large Scale Sci. Comput. 267–295 (2009)
15. Venticinque, S., Tasquier, L., Di Martino, B.: Agents based cloud computing interface for resource provisioning and management. In: Proceedings of the Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on, pp. 249–256 (2012). IEEE

Mahmoud Al-Ayyoub received his B.S. degree in computer science from the Jordan University of Science and Technology Irbid, Jordan, in 2004. He received his M.S. and Ph.D. degrees in computer science also from the State University of New York at Stony Brook, Stony Brook, NY, USA, in 2006 and 2010, respectively. He is currently an assistant professor at the Computer Science Dept at the Jordan University of Science and Technology, Irbid, Jordan. His research interests include wireless and cellular networks, game theory, artificial intelligence, machine learning and cloud computing.



Yaser Jararweh received his Ph.D. in Computer Engineering from University of Arizona in 2010. He is currently an assistant professor of computer sciences at Jordan University of Science and Technology, Jordan. He has co-authored about sixty technical papers in established journals and conferences in fields related to cloud computing, HPC, SDS, security and Big Data. He was one of the TPC Co-Chair, IEEE Globecom 2013 International Workshop on Cloud Computing Systems, and Networks, and Applications (CCSNA). He is a steering committee member for CCSNA 2014 and CCSNA 2015 with ICC. He is the General Co-Chair in IEEE International Workshop on Software Defined Systems SDS -2014 and SDS 2015. He is also chairing many IEEE events such as ICICS, SNAMS, BDSN, IoTSMS and many others. Dr. Jararweh served as a guest editor for many special issues in different established journals. Also, he is the steering committee chair of the IBM Cloud Academy Conference.

Mustafa Daraghmeh received his B.Sc. degree in computer science from Al-Balqa' Applied University, Al-Huson University College, Irbid, Jordan, in 2009. He received his M.S. degree in computer science also from the Jordan University of Science and Technology, Irbid, Jordan, in 2014. His main research interests include cloud computing, operating systems, multi-agent systems and software development.

Dr. Qutaibah Althebyan is an Assistant professor in the department of Software Engineering at Jordan University of Science and Technology (JUST). He has been there since August of 2008. Dr. Qutaibah Althebyan finished his Ph.D. degree in 2008 in Computer Science from University of Arkansas-Fayetteville and his Master degree in 2004 in Computer Information Systems from the University of Michigan-Dearborn. Dr. Althebyan published several papers in high ranked journals and conferences. He is also a reviewer for many journals and conferences. Dr. Althebyan main research interests are, but not limited to, in information security, database security, security in the cloud, big data management, health information systems, information assurance, software metrics and quality of open-source systems. Lately, he has been working in different security, e-health and software engineering projects, namely; Large Scale Insider Threat Assessments and damage assessment in the cloud in the area of cloud security. Also, studies of Power laws and their effects in object oriented metrics in the area of software engineering.