Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 215147, 9 pages
http://dx.doi.org/10.1155/2015/215147

Research Article

A New Resources Provisioning Method Based on QoS Differentiation and VM Resizing in IaaS

Rongdong Hu,[1] Guangming Liu,[1,2] Jingfei Jiang,[1] and Lixin Wang[1]

[1]*School of Computer, National University of Defense Technology, Changsha 410072, China*
[2]*National Supercomputer Center, Tianjin 300457, China*

Correspondence should be addressed to Rongdong Hu; rongdonghu@nudt.edu.cn

Received 1 June 2015; Accepted 13 July 2015

Academic Editor: Xiaoyu Song

In order to improve the host energy efficiency in IaaS, we proposed an adaptive host resource provisioning method, CoST, which is based on QoS differentiation and VM resizing. The control model can adaptively adjust control parameters according to real time application performance, in order to cope with changes in load. CoST takes advantage of the fact that different types of applications have different sensitivity degrees to performance and cost. It places two different types of VMs on the same host and dynamically adjusts their sizes based on the load forecasting and QoS feedback. It not only guarantees the performance defined in SLA, but also keeps the host running in energy-efficient state. Real Google cluster trace and host power data are used to evaluate the proposed method. Experimental results show that CoST can provide performance-sensitive application with a steady QoS and simultaneously speed up the overall processing of performance-tolerant application by 20~66%. The host energy efficiency is significantly improved by 7~23%.

1. Introduction

Cloud computing offers near-infinite amount of resources capacity at a competitive rate and allows users to obtain resources on demand with pay-as-you-go pricing model. Industry analyst firm IDC predicted that the global cloud market, including private, public, and hybrid clouds, will hit $118 billion in 2015 and crest at $200 billion by 2018 [1]. The proliferation of cloud computing has resulted in enormous amounts of electrical energy consumption.

IaaS (infrastructure as a service), as one important form of cloud computing, mainly leverages the virtualization technology to create multiple VMs (virtual machines) on a physical host and can support rapid deployment of large-scale applications [2]. Cloud providers can reduce power consumption by consolidating various applications into a fewer number of physical hosts and switching idle hosts to low-power modes. However, virtualization also creates a new problem. The applications performance relies on effective management of VM capacity. One essential requirement of cloud computing is providing steady QoS defined in terms of SLA (service level agreements). SLA violation will bring economic penalties to cloud providers. Therefore, they always strive to ensure the agreed performance of individual VM.

Consequently, performance control and power management are two major research issues in modern data centers. But they are in conflict. Seeking a good tradeoff between power consumption and applications performance is crucial for cloud providers.

The focus of this work is on the energy efficiency of IaaS cloud data center. Two typical applications are considered: performance-sensitive application (sVM) and performance-tolerant application (tVM). CoST places them on the same host and dynamically adjust their sizes based on the load forecasting. From the point of sVM, tVM is an elastic resource pool. The requirement of sVM dominates the host resource allocation. CoST not only guarantees the QoS, but also keeps the host running in energy-efficient state. Experiments with real trace data in CloudSim show that CoST can speed up the overall progress of tVM and improve the host energy efficiency significantly.

The rest of this paper is organized as follows. Section 2 describes the background and motivation. Section 3 discusses our analysis results, the detailed design and implementation

of the proposed method, and Section 4 presents the experimental evaluation. Section 5 examines the related work and Section 6 makes the conclusions.

2. Background and Motivation

2.1. Energy Efficiency. In an IaaS platform, multiple applications are hosted in a virtualized and shared physical infrastructure. Resource management is a complex issue due to the correlation between the power consumption and the resulted performance. Performance-oriented approaches aim to guarantee a performance target and do not have explicit control over power consumption. Power-oriented approaches aim to enforce power budget and disregard the QoS of hosted applications. Recent studies use *energy efficiency* to achieve a tradeoff between performance guarantee and power consumption. Two well-known technologies adopted by many previous researches for energy efficiency are DVFS (dynamic voltage and frequency scaling) and VM dynamic consolidation (i.e., VM migration).

2.2. DVFS Limitation. DVFS is a well-known hardware approach which shifts hardware components from high-power phases to low-power phases to reduce power consumption, such as Intel SpeedStep, Cool'n'Quiet, and AMD PowerNow!. In this technique, the voltage is dynamically adjusted in conjunction with clock frequency, allowing for a corresponding reduction in the power consumption, particularly for memory-bound workloads.

However, recent developments in processor and memory technology have resulted in the saturation of processor clock frequencies, larger static power consumption, smaller dynamic power range, and better sleep modes. Each of these developments limits the potential power savings that resulted from DVFS. Actually, in the analysis below, we can get the conclusion that host can achieve optimal energy efficiency only when it is running at full capacity.

2.3. VM Migration Costs. VM dynamic consolidation is typically achieved by VM migration. The migration process is complex. It may cause performance degradation of multiapplications, both inside and outside the VM being migrated [3]. During a VM migration the power consumption of both the source and the destination hosts is higher than the one before the migration is carried out [4]. Previous studies have demonstrated that transmission rate of an Apache web server slowed down by 12~20% [5] and power consumption may increase by 30% during VM migration [6].

3. Method

3.1. VM Resizing. In IaaS, applications share the underlying hardware by running in isolated VMs. Each VM, during its initialization, is configured with a certain amount of resources (such as CPU, memory, and I/O), called size. A key factor for achieving energy efficiency is resource provisioning. The objective of VM resizing is to ensure that VM capacity is matched with the load. Existing virtualization technologies can adjust the capacity of a live VM locally based on time division multiplexing to maximize the resource utilization.

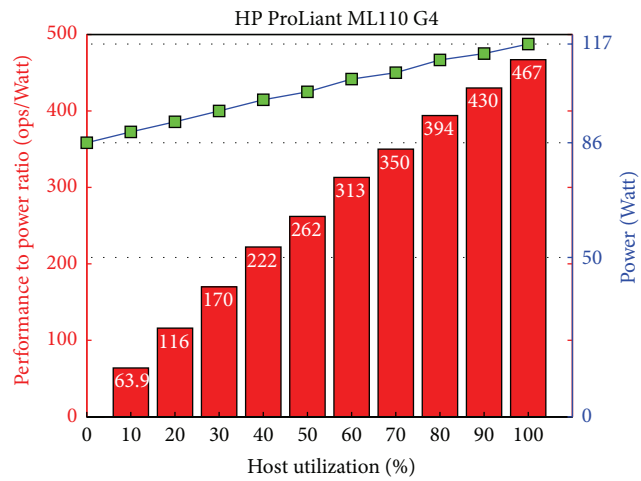

Figure 1: HP ProLiant ML110 G4 server energy efficiency data.

Implementation of any policy is accompanied by operating costs. Chen et al. [7] set up a simulated environment and perform a preliminary experiment to illustrate the VM resizing effect on three types of applications (CPU, memory, and network I/O intensive). Experimental results show that the application performance degradation during the VM resizing is smaller than that during VM migration, and the time of performance degradation is also shorter. In conclusion, we have good reason to believe that the VM resizing is worth considering firstly among various energy-saving technologies including DVFS and VM migration.

3.2. Optimal Energy Efficiency State. The power consumption of an idle server often exceeds half of its peak power consumption, because even when a server is not loaded with user tasks, the power needed to run the operating system and to maintain hardware peripherals (such as memory, disks, mainboard, PCI slots, and fans) at an acceptable state is not negligible [8].

SPECpower is an industry-standard benchmark for measuring both the performance and the power consumption of physical servers [9]. The data in Figure 1 demonstrate the relationship between server performance and power consumption, derived from HP ProLiant ML110 G4 server equipped with Intel Xeon 3040 processor with enhanced SpeedStep DVFS technology.

The highest performance (i.e., the maximum load the server can process) is marked as 100%. Each test runs at some percentage of the calibrated 100% performance in increments of 10%. Within the capacity of the server, increasing the load can improve the throughput of the system and exhibit better external performance (ops, throughput per second), also accompanied with power consumption increasing. When the server is idle, power consumption is 86 watts. It increases to 117 watts as is running at full load (point of 100%). However, it is worth noting that the performance to power ratio

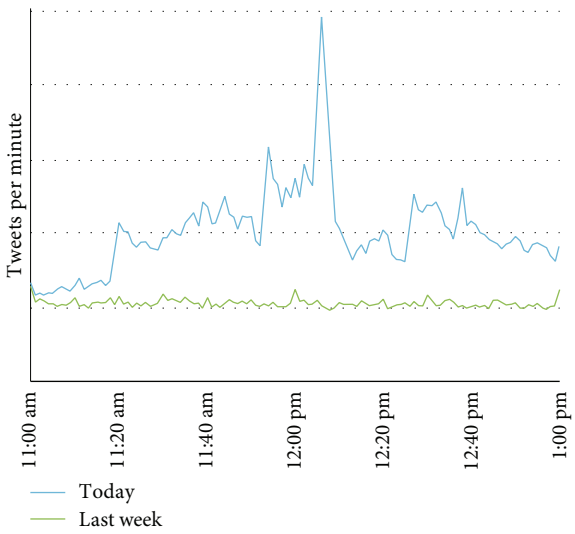

Figure 2: Load of Twitter on Obama's inauguration day.

(ops/watt) continues to increase with performance improving. It is only 63.9 ops/watt when the load is ten percent of the maximum. When the server is running at full capacity, the ratio reaches 467 ops/watt, more than sevenfold improvement. Through analysis, we found that almost all the datasets of SPECpower have the similar characteristics: as the load increases, the same amount of power can process more tasks.

Accordingly, there is every reason to believe that, within its capacity, keeping the host running as fully as possible is an effective way to improve the energy efficiency.

3.3. Performance-Sensitive/Tolerant Applications. A variety of applications are running on the cloud platform simultaneously. Some are highly performance sensitive and others need vast amounts of resources but do not require hard deadlines on execution.

3.3.1. Performance-Sensitive Applications. Internet applications are typical performance-sensitive, such as web service, streaming media, cloud gaming, and e-commerce. The web server may need to ensure that the response time experienced by its clients is less than 1 second on average. Similarly, a streaming media server may wish to ensure that the streaming rate is high enough to avoid playback discontinuity. For these applications, while the power control is needed to consider, good performance guarantee is the most important precondition.

Efficient resource provisioning for these applications is usually difficult. They often have interactive load, which consists of a number of client transactions to process in a given instant. Resource utilization of these loads is highly variable over time, as it depends on the number of simultaneous clients interacting with the application and on the particular interaction that each client performs. Take Figure 2, for example. It depicts a real-world scenario wherein Twitter experienced dramatic load fluctuations on Obama's inauguration day [10]. Such scenario is very typical in modern cloud applications.

3.3.2. Performance-Tolerant Applications. In contrast, performance-tolerant applications (such as weekly batch jobs, document converting, and media transcoding and compressing) usually have relatively loose deadlines that allow for being flexible about when to process the tasks.

These applications usually have batch loads consisting of a collection of tasks. Normally, a task is processed as fast as the machine can cope with. For instance, for computing-intensive application such as the media transcoding, the utilization of CPU tends to be around 100% during its execution. In other words, this type of applications always tends to run out of all computing resources allocated to it before its task is completed. Combined with the analysis of the previous Section 3.2, these applications can often achieve good energy efficiency.

Cloud service providers can develop a more rational pricing mechanism to guide customers to choose more efficient resource usage patterns. For example, providers can use relatively low prices to attract customers with performance-tolerant applications to sign a more rational SLA, which has relatively loose constraints on the QoS and allows for temporary performance degradation. Alternatively, providers can charge those batch jobs by task size. If the task completion time exceeds the negotiated deadline, customers can get some compensation. It is another issue worthy of study. In this work, we take advantage of this type of applications to improve the host energy efficiency without SLA violation.

3.4. CoST. An ideal resource provisioning strategy is to seek the best tradeoff between the energy consumption and application performance. It is a common management task in modern virtualization-based cloud. The topic also has been widely studied in the research community. Many earlier works only considered each VM separately. Obviously, the potential of these methods is quite limited. In contrast, recent researches began to focus on the affinity or conflict relationship between applications/VMs (such as similar memory content sharing, consolidation based on communication patterns). Our study further exploits the potential of their relationship for energy efficiency in IaaS.

3.4.1. Overview. According to analysis conclusions of Sections 3.1 and 3.2, in order to achieve better energy efficiency, our method tries to make the physical host running at full capacity by using VM resizing. Because the existing virtualization technology cannot completely isolate all types of physical resources, full capacity operation of host will cause intense competition for resources between the colocated VMs, resulting in application performance degradation and SLA violation. Inspired by Section 3.3, we make use of the fact that the two types of applications usually have different QoS requirements, to solve this problem.

Our method, named CoST, coallocates two different types of VMs in one host. The VM running the performance-sensitive/tolerant application is labeled as sVM/tVM. As the performance-sensitive application has relatively higher performance requirements, we let the resource requirement of sVM to dominate the host resource allocation. Specifically, we

use the mature load forecasting technique to obtain the future resource requirements of sVM. Then, resource provisioning must meet this predicted value as precise as possible to avoid the SLA violation punishment. At the same time, the colocated tVM plays a role of elastic resource pool for its sVM roommate. If the predicted sVM resource requirement in future interval will increase by Δx, CoST cuts down the same amount of tVM size and reallocates the deprived resource to sVM. Conversely, redundant resources of sVM will be reclaimed and reallocated to tVM. In addition to guaranteeing sVM performance, another major purpose of CoST is to keep the host running at full capacity to improve energy efficiency. Obviously, during the VM resizing process, sVM always has a stable performance (high QoS), and the tVM, which is in a passive position and has uncertain resource (low QoS), mainly plays the role of energy efficiency improving. Because the application running in tVM is performance-tolerant, its temporary performance degradation is acceptable and will not lead to SLA violation punishment. CoST takes the advantage of the QoS differentiation between sVM and tVM, to achieve a good tradeoff between the energy consumption and performance and to obtain better energy efficiency.

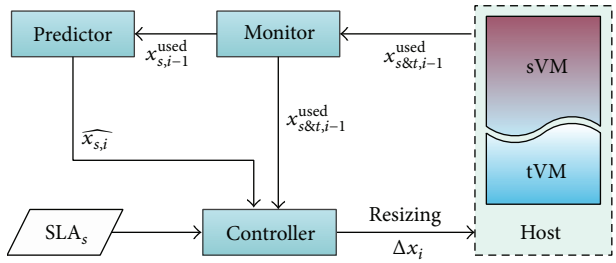

Figure 3: Control process of CoST.

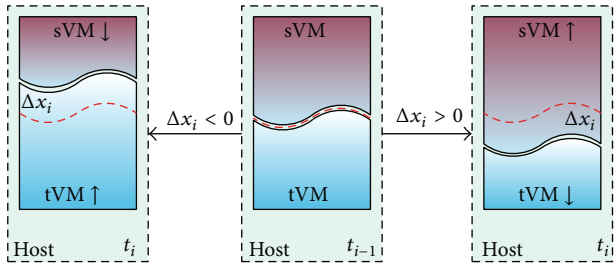

Figure 4: VM resizing.

3.4.2. CoST Design. Resource utilization is commonly used by data center operators as a proxy for application performance because of the monotonic relationship between them and the fact that utilization is easily measurable at the OS level. In-depth researches on this relationship were also conducted, such as [11]. As it lies outside the sphere of this work, without loss of generality, we also adopt resource utilization to indicate QoS.

(i) SLA Definition. As a crucial element for QoS differentiation, SLA model of sVM is defined as follows:

$$\frac{1}{i-1}\sum_{j=1}^{i-1} F\left(x_{s,j}^{\text{alloc}} > x_{s,j}^{\text{used}}\right) = \text{cslaV}_i \le \text{slaV},$$
$$F(y) = \begin{cases} 0, & \text{if } y \text{ is true,} \\ 1, & \text{if } y \text{ is false.} \end{cases} \tag{1}$$

$x_{s,j}^{\text{alloc}}$ and $x_{s,j}^{\text{used}}$ separately represent actual resources allocation value and real resources usage of sVM in time interval j. cslaV$_i$ is the cumulative average SLA violation rate before interval i. It is the indication of the average QoS for sVM and is restrained by slaV which is confirmed after the negotiation between cloud service providers and customers. slaV represents the user's tolerance for performance degradation. The smaller the value, the greater the sVM performance requirement. It is usually smaller than 5% for performance-sensitive applications. As long as VM resource utilization is below 100% (of course, a certain safety margin is also permitted), that is, $x_{s,j}^{\text{alloc}} > x_{s,j}^{\text{used}}$, we judge that the resource is enough and there is no SLA violation.

(ii) Controller. The block diagram for the CoST is shown in Figure 3. At the beginning of the control interval i, the VMM (virtual machine manager) monitor collects real resource consumption of both sVM and tVM during the previous interval ($x_{s\&t,i-1}^{\text{used}}$). Predictor estimates the resource requirements of sVM ($\widehat{x_{s,i}}$) according to the historical data from monitor. Controller calculates the resource provisioning variation (Δx_i) in the interval i according to the cumulative average SLA violation rate and host resource allocation in last interval:

$$\Delta x_i = s\widehat{x_{s,i}} - x_{s,i-1}^{\text{alloc}},$$
$$s = a^{\text{cslaV}_i - \text{slaV}/10}, \tag{2}$$

where s is an incremental coefficient which is highly correlated with QoS of sVM. Its value depends on the gap between the actual application performance (cslaV$_i$) and SLA (slaV). The greater the gap, the bigger the s. Bigger s means allocating more resources and fewer SLA violations. Because the application in sVM is sensitive to SLA violation, we adopt cslaV$_i$ − slaV/10 instead of cslaV$_i$ − slaV. That is, when SLA violation rate exceeds one-tenth of the allowed value, the controller will adaptively adjust control parameters and respond quickly to increase the supply of resources. It is a proactive (predictor) and adaptive (s) control process driven by QoS of sVM.

Controller makes the resizing scheme and sends it to VMM to execute. The resizing operation has two cases as shown in Figure 4. If the estimated resource requirement of sVM is increasing ($\Delta x_i > 0$), CoST cuts down the resource of tVM by Δx_i and reallocates it to sVM. Conversely, if sVM does not need that much resource as in the last interval ($\Delta x_i < 0$), CoST will reallocate the idle resource of sVM to tVM. We can see that, during the whole process, the host is kept running at full capacity and the performance of sVM is always guaranteed well. While it is a seemingly simple resource provisioning strategy based on QoS differentiation,

good energy efficiency is achieved without SLA violation. Moreover, CoST has the advantage of high feasibility and light additional overhead.

In IaaS virtualization platform, there are many mature tools (such as Xen, KVM, and VMware) available for system manager to perform the monitoring and resizing operations. For instance, we can use the Xen *xm* command to collect the CPU utilization of VM and use the Xen credit scheduler to set the CPU capacity limit of VM for resizing. More importantly, there are some good forecasting algorithms with high accuracy and low overhead for selection, such as KSwSVR [12] adopted in this work.

4. Experimental Results

It should be noted that the experiments in this paper mainly focus on CPU. So, the experimental conclusions surely apply to CPU-intensive applications. However, CoST is also applicable to other types of applications (such as the memory/disk/network-intensive ones), because the existing virtualization technology can dynamically split these types of resources in a fine-grained way and the forecasting algorithm also applies to these resource objects.

4.1. Experiment Setup

4.1.1. CloudSim. CoST is implemented on the CloudSim [13], one of the most sophisticated and comprehensive simulators for cloud computing. It is a framework for modeling and simulating the cloud computing services and infrastructures, including basic cloud resources, various scheduling algorithms, and resource provisioning policies, and is a repeatable and controllable environment.

In the experiment, a bottom resource line is set for tVM, making it always have a certain amount of resources and be kept running. However, this constraint is not necessary, because we can architect applications to be resilient against VM interruption by splitting work into small tasks (via Hadoop or queue based architectures) or checkpointing work-in-progress. So, we can also strip tVM of all resources and switch host offline. Besides, because existing virtualization technology can make VMs flexibly share resources across multiple CPU cores, we treat the CPU as a whole in the experiment. This does not affect the assessment of the effectiveness of CoST.

4.1.2. Google Workload Trace. The effectivity of CoST is evaluated by using real-world Google cluster workload traces (version 2011_2) (http://code.google.com/p/googleclusterdata/) rather than historical data generated by ourselves, for the purpose of giving comparable and reproducible results. Specifically, a job is an application that consists of one or more tasks. At run time, the usage of a task is measured as the actual consumption of each type of resources in five-minute interval. Our current work mainly focuses on CPU. However, as previously mentioned, it is straightforward to extend our method to consider other resources such as memory and disk space. For confidentiality reasons, the values reported in Google cluster traces were normalized between 0 and 1 before release.

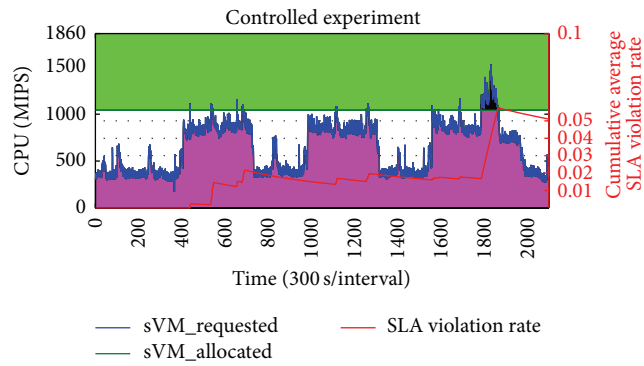

Figure 5: Fixed-size VM: CPU capacity remains unchanged.

We randomly selected some long duration jobs as the applications in sVM, extracted its more than one week of data (2100 intervals) as the load, and linearly converted its CPU usage for our experiment. The CPU usage of the job at a given time is computed by summing up the resource usage of all its running tasks at that time.

4.1.3. Experimental Results. By convention of CloudSim, CPU capacity is represented as Million Instructions per Second (MIPS) in this experiment. Naturally, Million Instructions (MI) is adopted to measure the amount of effective work. Energy efficiency (EE) is defined as the ratio between the finished effective work and the consumed energy to do it:

$$\begin{aligned} \mathrm{EE} &= \frac{\text{effective Work (MI)}}{\text{energy (J)}} = \frac{\text{effective Work (MI)}}{\text{energy (W} \cdot \text{S)}} \\ &= \frac{\text{performance (MIPS)}}{\text{power (W)}}. \end{aligned} \tag{3}$$

Note that EE is equivalent to the ratio between the amount of effective work finished each second (performance, MIPS) and the electric quantity during that second (power, W).

We set the parameters of the host CPU capacity and power consumption according to the HP ProLiant ML110 G4 server described in Section 3.2. As only ten levels of power consumption can be obtained from [9], linear interpolation is adopted to calculate the power consumption for other load levels. The slaV is set as 5%; that is, the time in which resource is lacking cannot be more than 8.75 hours during a week.

First, we designed a controlled experiment with randomly selected sample job430861762. In many existing commercial IaaS, such as Amazon EC2, the fixed-size VMs are provided. We make a sVM collocate with a tVM in the same host and also keep their CPU sizes (MIPS) remaining unchanged during runtime. The result is shown in Figure 5. We can see that the CPU usage of sVM (sVM_requested) continues to fluctuate. This is a typical feature of many performance-sensitive applications. We calculate the size of sVM (sVM_allocated, 1041 MIPS) beforehand, according to the SLA violation rate threshold (slaV). The rest of the host capacity is allocated to tVM (819 MIPS). During runtime, tVM always tends to run

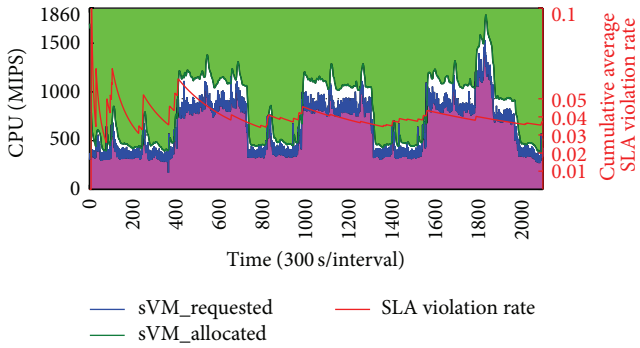

Figure 6: CoST: the requirement of sVM dominates the resizing.

out of all computing resources allocated to it (green area). In contrast, sVM only uses a portion of the resource allocated to it (magenta area) most of the time, and considerable resources are idle (white area). It is worth noting that although the cumulative average SLA violation rate is controlled within 5% eventually and the overall QoS of sVM is guaranteed, the concentrated load increasing brings continuous lack of resources (black area), which will result in about continuous 6.3 hours of performance degradation. It will lead to a very poor user experience. The root cause of the problem is that the fixed-size VM method uses more redundancy resource to guarantee the overall average QoS but cannot respond quickly to transient changes in the load. The performance is guaranteed at the expense of the host energy efficiency. EE is 12,665 MIPS/Watt.

It is worth reminding that, in this controlled experiment, we have got the resources requirement data of all the time intervals without any error in advance, and then the size of sVM is figured out manually based on the host capacity and QoS requirements. However, in actual applications, the long-term resource requirements cannot be obtained in advance, and even the short-term resource requirement estimated by other methods (such as forecasting or modeling techniques) will have at least some errors. Therefore, the optimal CPU size is hardly obtained in the method with fixed VM size. As a matter of course, the actual host energy efficiency is certainly lower than 12,665 MIPS/Watt.

Then, we verify the effectiveness of CoST with the same parameters. Different from the controlled experiment, the future requirement cannot be got directly. CoST predicts it based on historical data. The result is shown in Figure 6. At the beginning, the prediction error is large because of limited historical data used for training predictor. This, coupled with the small overall base, makes the cumulative average SLA violation rate change dramatically, exceeding the allowable range. However, with the accumulation of historical data, the accuracy of the predictor gradually increases. The predictor and controller make the CPU size of sVM change along with its requirement. Existing mature virtualization technology can quickly perform the resizing scheme, ensuring the sVM has the capacity to deal with load fluctuation. The SLA violation rate can be effectively controlled below 5%.

The biggest difference between Figures 5 and 6 is that the white areas reduced significantly. It means that only a small amount of resources are idle. As mentioned in Section 3.2, this is an effective way to improve energy efficiency. In addition, we did not find visible black areas in Figure 6. In other words, there is no continuous lack of resources and no significant performance degradation in more than one week simulation time. The host energy efficiency EE is 14,944 MIPS/Watt.

Table 1: Experimental results for job430861762.

Method	Fixed VM size	CoST	Improvement
sVM SLA violation rate	5.10%	3.72%	27.06%
tVM total work (billion MI)	539.83	708.89	31.32%
Energy efficiency (MIPS/watt)	12,665.33	14,944.60	18.00%

The detailed experimental results are listed in Table 1. As the performance-sensitive application in sVM places a higher value on QoS, CoST uses load forecasting technology to timely expand the VM size in response to the load increasing, leading to a 27.06% decrease in the rate of sVM cumulative average SLA violation. In contrast, the performance-tolerant application in tVM is relatively more cost-conscious and always makes full use of resources allocated to it and does not care about the short-term performance fluctuations. CoST uses virtualization resizing to quickly allocate idle host resources to it as much as possible.

Although this mechanism will lead to lower short-term performance, from an overall viewpoint, it will not necessarily postpone task completion time of the performance-tolerant application. For example, in our experiments, CoST speeds up the tVM process by 31.32% ultimately. CoST is always trying to make the host running at full capacity. The experimental result verifies the conclusion of Section 3.2: energy efficiency is improved by 18.00%.

In order to verify the universality of this CoST, we randomly selected some other data of Google trace to test. Experimental results show that CoST can always guarantee the performance of sVM and significantly improve the overall energy efficiency of the host. Figures 7 and 8 are part of the test results. Taking the fixed-VM-size method as benchmark index, all data in three-dimensional histogram are converted linearly for more clear presentation. Similar to Figure 6, there is almost no visible black area (lack of resources), and white area (wasting of resources) is little. Moreover, the overall process of tVM is also speeded up obviously, by 20~66%. It can be seen that CoST, respectively, improved the host energy efficiency by 18%, 23%, 7%, and 20%.

The implementation mechanism of CoST is simple, and this means lower management costs. This method is based on load forecasting, and it applies to any higher accuracy prediction algorithm that may arise in the future. As described in Section 3.3, these two types of applications exist extensively in the cloud, so this method has good applicability.

Different from the other energy-saving technologies such as DVFS, VM migration, and VM replication, CoST is a feasible and effective method running in host, and it improves the host energy efficiency by keeping it running at nearly full capacity. But it is worth noting that they are not in conflict but

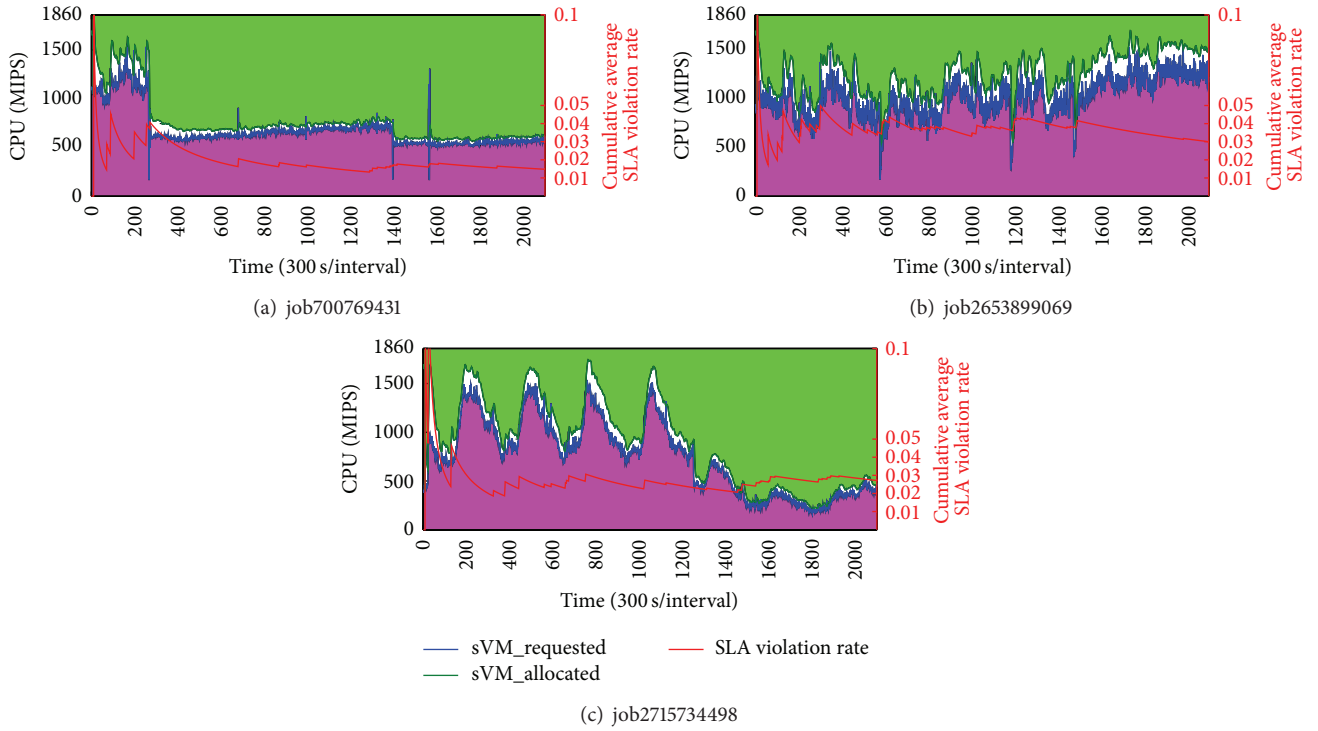

Figure 7: Experimental results of CoST for Google traces.

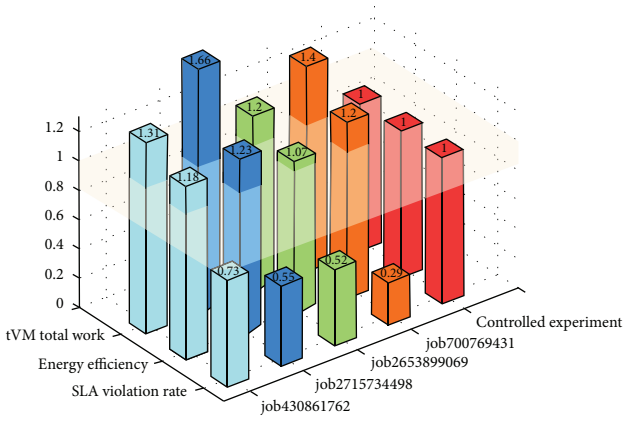

Figure 8: Improvement of CoST.

complementary to each other. For instance, if VM resizing cannot meet the needs of the individual sVM, it can be migrated to a larger capacity host. If there is not enough tVM to match the sVM, DVFS still can play a role. In extreme cases, if sustained heavy loads of sVM severely delay the progress of tVM, VM replication (horizontal scaling) can be enabled to speed it up.

5. Related Work

The research activities related to the energy efficiency issues in IaaS concern two main topics that are strictly correlated to our work: VM placement and VM resizing.

5.1. VM Placement. VM placement (VMP) is to establish mapping relationships between VMs and physical machines (PMs) so as to select the most suitable host for each VM. Van et al. [14] proposed a constraint programming VMP algorithm that can reduce the number of PMs and maximize the global utility function under the constraints of achieving SLA and operating costs. Meng et al. [15] propose a TVMPP (traffic-aware VM placement problem) algorithm which only considered the network constraints. Bobroff et al. [16] introduced a dynamic placement algorithm based on Bin Packing. Chaisiri et al. [17] proposed an optimal virtual machine placement (OVMP) algorithm based on stochastic integer programming (SIP), to minimize the cost spending in reservation and on-demand plans in a multiple cloud provider environment under future demand and price uncertainty. The VMP scheme proposed by Nakada et al. [18] is based on genetic algorithm, which is suitable for the situations where objective function changes dynamically. Agrawal et al. [19] modeled the VMP problem as a vector packing problem with conflicts (VPC) and used the grouping genetic algorithm (GGA) to minimize the number of servers and maximizes the packing efficiency. Fan et al. [20] proposed a method for deploying scientific applications. It takes not only the computing qualities of cloud nodes into account, but also the communication performance between different nodes.

Different from the existing works, our method uses VMP to improve energy efficiency from a new angle. CoST uses the cooperation (coallocation and resizing) between two types of applications to improve the energy efficiency.

5.2. VM Resizing. The fine-grained resource allocation of VM resizing provides the users with more flexibility to cope with the varying demands. Lu et al. [21] present a tool for automatic vertical scaling of VMs based on throughput, average response time, and percentile response time. In [22], VM-level CPU shares (aka. weights) were used to ensure that the most critical applications achieve their performance goals. Different from our method based on load forecasting, they used dynamic weight management to deal with the varying workloads. Dawoud et al. [23] proposed a new fine-grained vertical scaling architecture and compared it with multi-instances horizontal scaling. CloudScale [24] uses resource requirement predictions to scale a VM vertically. However, a user still has to manually determine thresholds for a given QoS target. Yazdanov and Fetzer [25] use an autoregressive prediction model to predict the resource requirements of a VM in order to dynamically hot-plug CPUs in Xen. The approach does not consider the application performance in the scaling decision. VScaler [26] uses reinforcement learning to decide when to scale up or down and presented a novel parallel learning approach to speed up the learning process of reinforcement learning.

CoST also applies VM resizing to scale the capacity of VMs. Our work not only uses forecasting technology, but also drives the resource provisioning with QoS feedback to make up for the prediction error. The most important is that we do not focus only on application performance or resource utilization but take into account energy efficiency which is a more reasonable metrics.

6. Conclusions and Future Work

In this work, we proposed a fine-grained host resource provisioning method, CoST, to improve the host energy efficiency of IaaS. CoST is based on QoS differentiation and VM resizing. It takes advantage of this fact that different types of applications have the different requirement for performance and cost. Performance-sensitive applications like to pursue stable performance, while the performance-tolerant applications prefer a reasonable overall cost. CoST places two different types of VMs on the same host and dynamically adjusts their sizes based on the load forecasting and QoS feedback. It not only guarantees the performance of sVM, but also keeps the host running in energy-efficient state. Real Google cluster trace and host power data are used to evaluate our method. Experimental results show that CoST can provide sVM with a steady QoS and also speed up the overall process of tVM by 20~66%. The host energy efficiency is significantly improved by 7~23%.

In the future, we plan to design a more intelligent method to automatically judge the type of application and will make coallocation recommendations for the user to choose. A more rational pricing mechanism which tries to guide users to actively accept such energy-efficient method is also on the agenda.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (NSFC) under Grant no. 61303070.

References

[1] F. Gens, *IDC Predictions 2015: Accelerating Innovation—and Growth—on the 3rd Platform*, International Data Corporation, Framingham, Mass, USA, 2015.

[2] Z. Zhang, Z. Li, K. Wu et al., "Vmthunder: fast provisioning of large-scale virtual machine clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3328–3338, 2014.

[3] W. Dargie, "Estimation of the cost of VM migration," in *Proceedings of the 23rd International Conference on Computer Communication and Networks (ICCCN '14)*, pp. 1–8, IEEE, Shanghai, China, August 2014.

[4] K. Rybina, W. Dargie, A. Strunk, and A. Schill, "Investigation into the energy cost of live migration of virtual machines," in *Proceedings of the 3rd International Conference on Sustainable Internet and ICT for Sustainability (SustainIT '13)*, pp. 1–8, IEEE, October 2013.

[5] C. Clark, K. Fraser, S. Hand et al., "Live migration of virtual machines," in *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation—Volume 2*, pp. 273–286, USENIX Association, 2005.

[6] Q. Huang, F. Gao, R. Wang, and Z. Qi, "Power consumption of virtual machine live migration in clouds," in *Proceedings of the 3rd International Conference on Communications and Mobile Computing (CMC '11)*, pp. 122–125, IEEE, April 2011.

[7] W. Chen, X. Qiao, J. Wei, and T. Huang, "A two-level virtual machine self-reconfiguration mechanism for the cloud computing platform," in *Proceedings of the 9th International Conference on Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC '12)*, pp. 563–570, IEEE, September 2012.

[8] G. Chen, W. He, J. Liu et al., "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI '08)*, vol. 8, pp. 337–350, San Francisco, Calif, USA, April 2008.

[9] Hewlett-Packard company proliant ml110 g4, 2011, http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110124-00338.html.

[10] Inauguration Day on Twitter, 2009, http://blog.twitter.com/2009/inauguration-day-twitter.

[11] S. Kundu, R. Rangaswami, A. Gulati, M. Zhao, and K. Dutta, "Modeling virtualized applications using machine learning techniques," *ACM SIGPLAN Notices*, vol. 47, no. 7, pp. 3–14, 2012.

[12] R. Hu, J. Jiang, G. Liu, and L. Wang, "Efficient resources provisioning based on load forecasting in cloud," *The Scientific World Journal*, vol. 2014, Article ID 321231, 12 pages, 2014.

[13] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency Computation Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[14] H. N. Van, F. D. Tran, and J.-M. Menaud, "Autonomic virtual resource management for service hosting platforms," in

Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing (CLOUD '09), pp. 1–8, IEEE, Vancouver, Canada, May 2009.

[15] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of the IEEE (INFOCOM '10)*, pp. 1–9, IEEE, San Diego, Calif, USA, March 2010.

[16] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," in *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM '07)*, pp. 119–128, IEEE, May 2007.

[17] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *Proceedings of the IEEE Asia-Pacific Services Computing Conference (APSCC '09)*, pp. 103–110, IEEE, Singapore, December 2009.

[18] H. Nakada, T. Hirofuchi, H. Ogawa, and S. Itoh, "Toward virtual machine packing optimization based on genetic algorithm," *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, pp. 651–654, 2009.

[19] S. Agrawal, S. K. Bose, and S. Sundarrajan, "Grouping genetic algorithm for solving the server consolidation problem with conflicts," in *Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pp. 1–8, ACM, June 2009.

[20] P. Fan, J. Wang, Z. Chen, Z. Zheng, and M. R. Lyu, "A spectral clustering-based optimal deployment method for scientific application in cloud computing," *International Journal of Web and Grid Services*, vol. 8, no. 1, pp. 31–55, 2012.

[21] L. Lu, X. Zhu, R. Griffith et al., "Application-driven dynamic vertical scaling of virtual machines in resource pools," in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS '14)*, pp. 1–9, IEEE, May 2014.

[22] S. Blagodurov, D. Gmach, M. Arlitt, Y. Chen, C. Hyser, and A. Fedorova, "Maximizing server utilization while meeting critical SLAs via weight-based collocation management," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM '13)*, pp. 277–285, IEEE, Ghent, Belgium, May 2013.

[23] W. Dawoud, I. Takouna, and C. Meinel, "Elastic virtual machine for fine-grained cloud resource provisioning," in *Global Trends in Computing and Communication Systems*, vol. 269 of *Communications in Computer and Information Science*, pp. 11–25, Springer, Berlin, Germany, 2012.

[24] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, p. 5, ACM, 2011.

[25] L. Yazdanov and C. Fetzer, "Vertical scaling for prioritized VMs provisioning," in *Proceedings of the 2nd International Conference on Cloud and Green Computing (CGC '12)*, pp. 118–125, IEEE, November 2012.

[26] L. Yazdanov and C. Fetzer, "Vscaler: autonomic virtual machine scaling," in *Proceedings of the IEEE 6th International Conference on Cloud Computing (CLOUD '13)*, pp. 212–219, IEEE, July 2013.