Research on Resource Management in PaaS Based on IaaS Environment

Peng Xu, Rui Hu, and Sen Su

State Key Lab of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
xupeng@bupt.edu.cn

Abstract. As one of the three service models of cloud computing, PaaS (Platform as a Service) has gained more and more popularity for its capabilities in optimizing development productivity and business agility. However, the traditional PaaS uses the dedicated infrastructure, which generally leads to the low infrastructure utilization rate. To solve the above problem, PaaS based on IaaS (PoI) emerged, in which IaaS (Infrastructure as a Service) is involved to provide PaaS the infrastructure, to decrease the response time of the infrastructure scale and to increase the utilization of the infrastructure. Because PoI has many characteristics, resource management mechanisms used in the traditional PaaS or IaaS could no longer adopted in PoI. In this paper, an adaptive resource management framework and the corresponding scale-up, scale-down algorithms are brought forward to guarantee the QoS of applications deployed in PaaS platform as well as to decrease the rental cost of VMs from IaaS providers. Experimental results show that the resource management mechanisms proposed in this paper can not only guarantee QoS of all applications, but also improve the utilization rate of the infrastructure, thus to make PoI possess the advantages of both PaaS and IaaS.

Keywords: cloud computing, paas, iaas, resource management.

1 Introduction

Cloud computing has gained unprecedented popularity since its inception and becomes a great solution to provide a flexible, on-demand and dynamically scalable computing infrastructure for enterprises. Its *Pay-As-You-Go* pricing model is essentially similar to other public utilities (e.g., electricity, gas and water). Therefore, cloud computing is also called "On-Demand Computing". Cloud computing supports three service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

IaaS, as the basic cloud service model, enables users to rent the infrastructure (server, storage and network etc.) dynamically as needed. Since the increasing requirements to improve the cost-efficiency of Internet Data Centers (IDCs), IaaS has been widely adopted by more and more enterprises to improve the utilization rate of

J. Su et al. (Eds.): ICoC 2013, CCIS 401, pp. 145–157, 2013.

the infrastructure. At the same time, enterprises, especially IT companies, tend to set up PaaS platforms to provide Application Execution Environment (AEE) for their applications and reuse a variety of application components. Because PaaS also has to rely on the infrastructure such as servers and storage, IaaS becomes a candidate to provide PaaS the infrastructure. That is the PaaS based on IaaS (PoI). PoI could provide the features of both PaaS and IaaS: i) to provide AEE for applications and application components as PaaS; ii) to increase the utilization rate of the infrastructure as IaaS.

Resource management is one of the most important issues of cloud computing which decides the efficiency of the platform. In PoI, resource management also plays a very important role as well. It has many characteristics: i) the traffic model and QoS requirements of different applications on PoI are different and changing fast. ii) PaaS could dynamically use the resource of IaaS according to the requirements of applications.

In this paper, a QoS guaranteed and cost-efficient resource management mechanism in PoI is presented. The rest of the paper is organized as follows. The related works are introduced in Section 2. Then the architecture of PoI is illustrated in Section 3. In section 4, the problems which have to be faced by PoI are stated and the resource management framework of PoI is brought forward in detail. In the end, the experimental results are presented in Section 5 followed by the conclusions and future works in section 6.

2 Related Works

Recent research on dynamic resource provisioning in virtualized environments includes [1]. These research attempted to achieve application performance goals with dynamic resource provisioning in virtualized environments. Most recently, people extended the idea into the cloud environment [2]. [3] investigated task scheduling with deadline and budget constraints in the heterogeneous environment. Some research investigated cost-efficiency in the cloud environment. On the cloud providers' point of view, [4][5] discussed the resource allocation and instance consolidation strategies for cloud data centers. The goal is to maximize the profits of cloud providers while maintaining Service Level Agreement (SLAs). [6] discussed power management in cloud environment. On the cloud users' point of view, the hot topic is to build strategies to deploy applications among multiple cloud providers to enhance availability with minimum cost [7].

Because PaaS platform is essentially a distributed system, it faces similar resource management problems like other distributed systems. Computing resources (e.g., CPU, memory, disk I/O and network bandwidth) are limited in distributed systems; therefore how to manage these resources with high efficiency is critical. PaaS platform is asynchronous soft real-time system [8]. On one hand, PaaS platform is asynchronous because requests of applications on PaaS are unpredictable and nondeterministic, and the distribution of the requests cannot be precisely described by mathematical models. On the other hand, it is a soft real-time system because there are some constraints on the response time, but the failure of timeliness will only lead to

some small losses rather than disastrous consequences. The workload PaaS platform confronted fluctuates within a wide range. Over-provisioning system resources to accommodate the potential peak will lead to the waste of the resource. As a consequence, it is important for PaaS platform to maintain efficient resource utilization under a wide workload conditions without increasing the possibility of failure of timeliness. [9] proposed a new algorithm called FC-LRU which integrates feedback control with LRU algorithm to perform adaptive resource management in PaaS platform. However, the idea of FC-LRU algorithm is based on the assumption that the PaaS platform is using its dedicated infrastructure. Therefore, the resource management mechanism in traditional PaaS platform is no longer practical in PoI. New mechanism for PoI should be adopted.

3 Architecture of PaaS Based on IaaS

Some enterprises adopt PaaS to provide AEE for their applications and application components, because PaaS is famous for optimizing development productivity and business agility. However, the benefits of the traditional PaaS are limited, because the cost of the infrastructure scale-down and scale-up is high and with low efficiency in the typical PaaS. At the same time, IaaS has many advantages in optimizing infrastructure utilization. In order to maximize business agility and development productivity, enterprises are trying to set up PaaS on IaaS, namely PoI. PoI provides application development, testing, execution and provision environment which enables both PaaS' ease of application deployment and IaaS' efficient infrastructure management.

In the traditional PaaS, the infrastructure is dedicated for its usage. In the idle hours, some infrastructure is idle and lead to great waste; while in the busy hours, the infrastructure is inadequate, and QoS of applications could not be guaranteed. In PoI, when it is the idle hours, the infrastructure of PaaS could scale down by using the capability of the underlying IaaS; when it is the busy hours, the infrastructure of PaaS could scale up. This mechanism could greatly increase the Cost Efficiency of the platform.

The architecture of PoI is shown in Figure 2. It can be divided into two layers: PaaS Layer and IaaS Layer. The PaaS Layer dynamically rents VMs from the IaaS according to the workload.

- *App Execution Engine* hosts users' applications. It provides multiple AEEs and supports applications written by several programming languages (e.g., Java, PHP). As one application is scaled up, the App Execute Engine will launch more instances of this particular application.
- *Front End* is the entry to the platform: it accepts all requests from users and forwards them to the appropriate App Execution Engine. In essence, *Front End* is a load balancer that knows the correspondence between requests and applications.
- *Service Pool* manages a set of services including data storage and many other capabilities that can be used by developers via APIs.

- *Platform Master* is the key component of PoI. It is responsible for application scheduling, application resource allocation and interfacing with *VM Management* to acquire or release infrastructure resource from/to IaaS.
- *VM Management* is the key component of IaaS to perform the infrastructure resource scheduling and providing corresponding APIs.

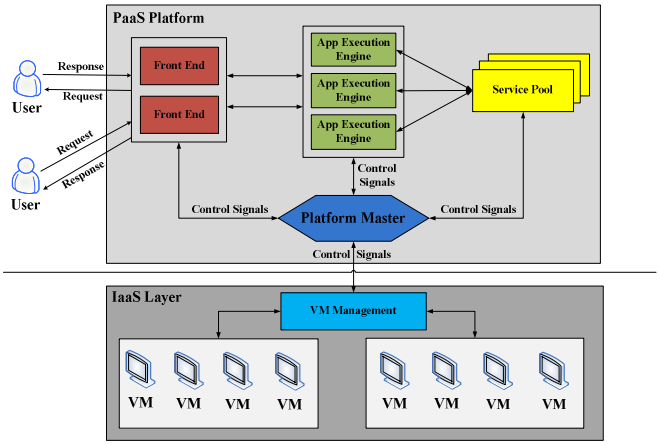


Fig. 1. Architecture of POI

In the architecture illustrated above, App Execution Engine and Front End are allowed to have more than one instance running simultaneously. Users' applications are deployed in App Execution Engine. All components of PaaS, including App Execution Engine, are deployed in the VMs provided by IaaS. This architecture ensures the scalability that all applications can easily be accommodated through scale-up and scale-down. When one application's workload increases and the requirements for computing resources increases accordingly, Platform Master can increase application's instances, either based on current set of VMs or acquires more of them from IaaS Layer to guarantee QoS of the applications. When one application's workload decreases, over-provisioning instances will lead to the waste of computing resources; thus Platform Master can decrease its instances and decide whether to return over-provisioning VMs to IaaS Layer in order to save VM rental cost.

4 Resource Management of PaaS Base on IaaS

4.1 Problem Statement

In this paper, a resource management framework of PoI is brought forward. On one hand, PoI is an asynchronous soft real-time system. Its requests are unpredictable and nondeterministic, and their distribution cannot be precisely described by the mathematical models. On the other hand, it is a soft real-time system because there are some

constraints on the response time, but the failure of timeliness will lead to some small losses rather than disastrous consequences. In addition, computing resources that PaaS needs are provided by IaaS Layer as a set of VMs. For those enterprises willing to adopt PoI, it is important to minimize the usage of VMs in order to minimize the rental cost as well as guarantee QoS of their applications.

QoS of the application is measured based on the number of requests missed their deadlines. Missed Deadline Ratio (denoted as MissRatio) is defined as the percentage of requests that missed their deadlines; the formula used to calculate this value is given by (1):

$$\text{MissRatio} = \frac{\text{number of requests missing their deadlines}}{\text{number of total requests}} \tag{1}$$

In terms of resource utilization, CPU utilization is involved in this paper because CPU is the scarcest computing resource. It is important for PoI to maintain CPU utilization at a high level. Different instances of an application may have different CPU utilization, the CPU utilization of the application (denoted as U_{cpu}) is defined as the average of CPU utilization of all instances; the formula used to calculate this value is given by (2):

$$U^{j}_{CPUi} = \frac{\sum_{i=0}^{n} U_{cpu_i}}{nM} \tag{2}$$

Where n is the number of instances. M is the maximum CPU utilization that each instance is allowed to use (CPU utilization of one application's instance can be limited by cgroup in recently released Linux version). M means that CPU utilization of an instance should be limited below the M (e.g., 20%). U^{j}_{CPUi} is used to represent the CPU utilization of i_{th}instance of application j. The overall CPU utilization of PaaS can be relatively high if the CPU utilization of each application is high.

Since users' applications are running in VMs provided by IaaS, it is equally important to reduce the usage of VMs in order to lower the rental cost as much as possible. Traditionally IaaS providers charge according to Pay-As-You-Go pricing model. Therefore the rental cost is determined by how long and how many VMs are used. The Usage of VM is defined as the total usage time (denoted as T_{VM}); the formula used to calculate this value is given by (3):

$$T_{VM} = \sum_{1}^{n} T_{VM_i} \tag{3}$$

Where T_{VM_i} represents the usage time of i_{th} VM and n represents the number of rented VMs.

In PoI, the resource management framework is designed following three objectives:

- Guaranteed QoS of the application, described by *Missed Deadline Ratio*.
- High resource utilization rate, described by *CPU Utilization*.
- Less VM usage, described by *Time Usage* of VMs.

4.2 Resource Management Framework

In [9], the resource management framework is mentioned and resource management problems are simplified into feedback control problems. The first two objectives mentioned above can be achieved by feedback control. PID (Proportional-Integral-Derivative) feedback control is adopted to maintain high CPU utilization. Besides, VM scheduling algorithm is designed to manage VMs in order to achieve cost efficiency as well as QoS guaranteed applications.

With its three-term functionality covering treatments to both transient and steady-state responses, PID control offers the simplest and yet most efficient solutions on many real-world control problems [10]. The reasons why PID control is employed in this paper include: i) PoI is essentially a dynamic system and PID control technique is widely accepted technique in dynamic systems; ii) precise mathematical model of the system is not required in PID control technique. Instead, PID control technique can achieve satisfactory performance based on an approximate model. The complexity of PoI makes it difficult to be described precisely by mathematical model, which makes PID control technique a great candidate for underlying resource management.

A classic feedback control system is composed of a controller, a plant (the object to be controlled no matter what it is) and sensors (the object to measure the output of the plant) [11]. Controlled Variables are the variables to be controlled. Set Points represent the correct and expected values of the Controlled Variables. The difference between the current value and the Set Point is the Errors. The whole feedback and control loop is aimed to reduce the Errors.

- The sensor periodically monitors and compares the *Controlled Variables* to the *Set Points* to determine the *Errors*.
- The controller generates control signals through control function based on the *Errors*.
- The actuator takes actions to control the plant based on the signal generated by the controller, which is aimed to reduce the *Errors*.

Since Platform Master communicates with all components and collect information (distribution of applications' instance, workload of each application's instance, resource utilization rate of each VM etc.) of the whole PoI and perform application scheduling, application resource allocation and VM management (by interfacing with VM Management of IaaS), Platform Master could be reconstructed to support the feedback-control loop, thus the resource management framework is mainly implemented in Platform Master, shown in figure 3.

In the framework shown in figure 3, Missed Deadline Ratio and average CPU Utilization are adopted as the Controlled Variables. Because Controlled Variables are application-independent, a small, non-zero value is used as the Set Point of Missed Deadline Ratio for each application. The Set Point of Missed Deadline Ratio of application j is denoted as $\mathrm{MissRatio}_s^j$. Similarly, an expected percentage is used as the Set Point of CPU Utilization for each application. The Set Point of CPU Utilization of application j is denoted as $U_{cpu_s}^j$. Note that the workload of each application in PaaS is unpredictable, it is impossible to achieve 100% CPU utilization and 0% missed deadline ratio. Therefore, a tradeoff between these two metrics is inevitable.

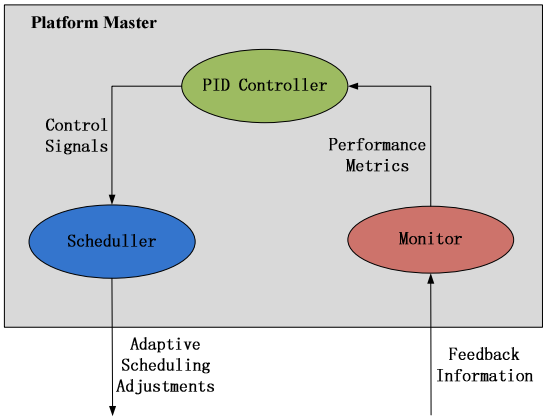


Fig. 2. Resource Management Framework

The sampling time is defined to be the end of each period of the arrival of the feedback data. Resource management decisions are made at the beginning of each period based on the feedback data collected previously. Then error_k^j for application j could be calculated with the following formula (4):

$$\mathrm{error}_k^j = w_1 \times \left(\mathrm{MissRatio}_k^j - \mathrm{MissRatio}_s^j\right) + w_2 \times \left(U_{\mathrm{cpu}_i}^j - U_{\mathrm{cpu}_s}^j\right) \quad (4)$$

Where k is the sampling instant and $\mathrm{MissRatio}_k^j$ is the Missed Deadline Ratio in the k_{th}sampling instant and Coefficients w_1 and w_2 are tunable. Based onerror_k^j, the number of instances of application j to be changed can be calculated in the current period with the following PID control formula (5):

$$\Delta\mathrm{Instance}^j = C_p \times \mathrm{error}_k^j + C_I \times \sum_{IW} \mathrm{error}_k^j + C_D \times \frac{\mathrm{error}_k^j - \mathrm{error}_{k-DW}^j}{DW} \quad (5)$$

Where C_p, C_I, C_D, IW and DW are tunable coefficients. The number of instances of application j can be changed according to $\Delta\mathrm{Instance}^j$. $\Delta\mathrm{Instance}^j > 0$ means that the application should scale up (i.e., the number of instances of application j should be increased), and $\Delta\mathrm{Instance}^j < 0$ means that the application should scale down (i.e., the number of instances of application j should be decreased), and $\Delta\mathrm{Instance}^j = 0$ means that current number of instances of application j is appropriate and does not need change.

4.3 VM Scheduling Policy

How to set up a feedback control system in PoI and how to determine the number of instances of each application in order to guarantee the QoS of each application are stated above, then the policy to schedule the rented VMs will be illustrated. The decisions on when to scale up and scale down are made on the objective: Low Time Usage of VMs. Specifically, as the ΔInstance has been computed, strategies of

scale-up and scale-down which can lower the Time Usage of VMs as much as possible should be determined. The VM Scheduling Policy can be divided into two algorithms: scale-up algorithm and scale-down algorithm.

1. Scale-Up Algorithm

In order to decrease the Time Usage of VMs, the CPU capacity of every VM should be monitored. In addition, there is a threshold for each VM to limit its CPU utilization in a safe level (e.g., 90%). The objective of scale-up algorithm is to make sure all instances are converging to a small number of VMs as well as keep each VM's CPU utilization in a safe level. A memory management algorithm "BEST FIT" is introduced to achieve the above objective. BEST FIT is a famous algorithm for OS memory management in order to avoid using larger blocks unnecessarily. In BEST FIT, the block list is searched for the block that is smallest but greater than or equal to the request size [12].

The suitable VM, into which new instances of the particular application to be expanded are deployed, is chosen based on BEST FIT. Next this algorithm will be further illustrated with an example. Assuming the utilization of all VMs is as figure 4:

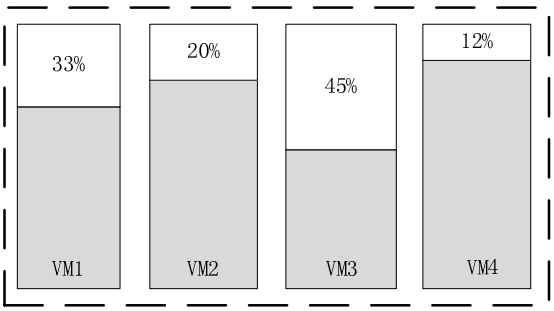


Fig. 3. CPU Utilization of VMs

In figure 4, there are 4 VMs. White box represents the percentage of CPU unused. In this example, an instance of application j need to be created and this instance will occupy 10% CPU utilization of a single VM. The safe threshold of VMs is 90%. Therefore, the actual percentages of CPU which could be used are 23%, 10%, 35% and 2% respectively. According to the BEST FIT algorithm, VM2 is chosen as the target to bear application j's instance. If there is more than one instance to be scaled up, the above procedures should be performed iteratively based on BEST FIT algorithm. By using BEST FIT algorithm, the CPU of every rented VMs will be fully exploited over the time. Scale-Down Algorithm

The scale-down algorithm is designed following the same objective as the scale-up algorithm. Differing from scale-up algorithm, the scale-down algorithm tries to determine which instance to be removed in order to acquire as many idle VMs as possible and return them to IaaS Layer.

k_{th} VM is denoted as $VM_k\{\text{Application Sequence}\}$, where Application Sequence is the sequence of applications ID whose instances are running in VM_k. For example, if 1 instance of application 1, 1 instance of application 2 and 1 instance of application 3 are running in k_{th} VM, this VM can be denoted as $VM_k\{1,2,3\}$. Application Sequence probably has duplicate application ID since there could be more than one instances of one application running simultaneously on the same VM. The above objective is achieved by introducing the heuristic algorithm and removed the instances of the VM which has the least number of running instances and its instances are the subset of the instances to be removed. Assuming there are 4 VMs: $VM_1\{1,2\}$, $VM_2\{1\}$, $VM_3\{2,2,3\}$, $VM_4\{3,4,5\}$, and the instances to be shrunk is {1,1, 2, 2, 2, 3}. Based on our scale-down algorithm, the instances should be removed according to the following sequence: $VM_2\{1\}$, $VM_1\{1,2\}$ and $VM_3\{2,2,3\}$. VM 1, 2 and 3 are set idle after the scale-down and can be returned to IaaS Layer.

4.4 Resource Management Process

In summary, the resource management process in PoI is essentially a feedback-control loop. At the beginning of every period, Platform Master collects all feedback data from other components of the platform and makes resource management decisions according to the feedback data. Every feedback-control loop can be divided into 4 phases.

- To collect performance metrics including Missed Deadline Ratio and CPU Utilization of every application deployed on the platform.
- To compute $\Delta\text{Instance}$ for every application based on PID control function mentioned above.
- To deploy or remove application instances based on scale-up or scale-down algorithms. Rent VMs from IaaS Layer if needed.
- To return all idle VMs to IaaS Layer in order to save rental cost.

5 Experiments

In this chapter, the comprehensive evaluation of the resource management framework of PoI is presented. In Section 5.1, the details of the experimental setup are specified. In Section 5.2, the proposed framework under different experimental parameters is illustrated, then the analysis of the experimental results is given.

5.1 Experimental Setup

For the evaluation of the proposed resource management framework in PoI, workloads of different patterns as the workloads of the platform are prepared. The different workloads are provided by The Grid Workloads Archive, which could provide anonymous workload traces from grid environments to researchers and to practitioners alike [13]. 3 workloads of different patterns within 24 hours are chosen as shown in Figure 5.

Each workload pattern above represents a typical scenario. For example, workloads of Application 1 and Application 3 slightly differ from each other but show a very common pattern that fluctuates consistently and reaches peaks in the middle of the day. In addition, workload of Application 2 shows a very steady pattern with several wiggles.

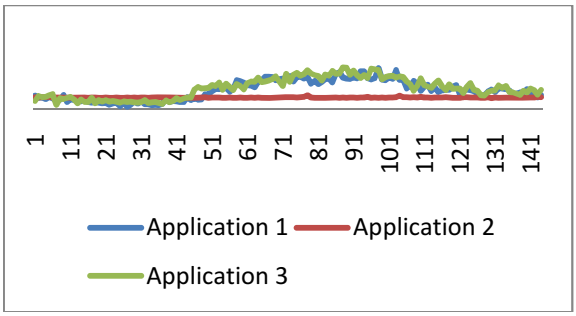


Fig. 4. Workload Patterns of Application 1, 2 and 3

The fixed parameters of the experiment are shown in Table 1. Each workload is split into 144 intervals with 10 minutes per interval. Each application is initialized and deployed with 1 instance. Coefficients shown in Table 1 are obtained based on experiences. The values of these coefficients could be determined by using PID tuning techniques [14].

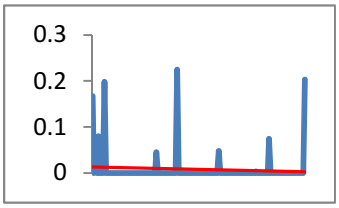
Table 1. Fixed Experimental Parameters

Number of Applications	3
Minutes per Interval	10
Number of Intervals	144
Initial Instances of each Application	1
Coefficient w_1	2
Coefficient w_2	1.5
Coefficient C_p	1
Coefficient C_i	0.5
Coefficient C_d	1

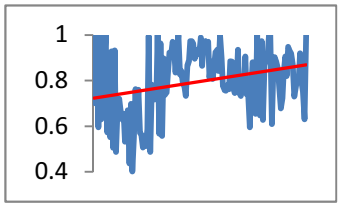
5.2 Evaluation of Resource Management Framework

Besides the fixed experimental parameters shown in Table 1, there are also three kinds of tunable experimental parameters including Set Point of Missed Deadline Ratio for each application (denoted as MissRatio_s), Set Point of CPU Utilization for each application (denoted as U_(cpu_s)) and VM Safe Threshold. For the clarification and simplicity of the following analysis, MissRatio_s and U_(cpu_s) for all applications are set together. The proposed resource management framework is evaluated under different settings of tunable experimental parameters.

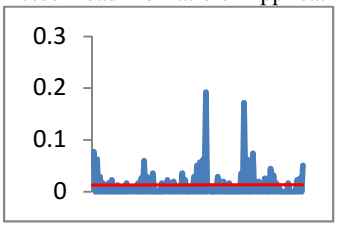
Figure 6 shows the Missed Deadline Ratio and CPU Utilization of different applications in the platform with the tunable parameters settings in above table. The proposed resource management framework keeps both the Missed Deadline Ratio and CPU Utilization of all applications around the expected values in response to the workloads of different patterns. The proposed framework also has the predictive capability to the changing workload. Take Application 1 as an example, as its workload increases suddenly and keeps the increasing trend for periods, its Missed Deadline Ratio only increases for a short time and returns to the expected value. The proposed framework gains this predictive capability through PID control function based on which it scales up Application 1 to deal with the future workload increase, with the cost of the temporary fall of CPU Utilization.



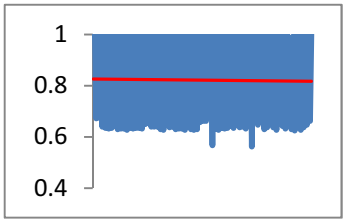
Missed Deadline Ratio of Application 1



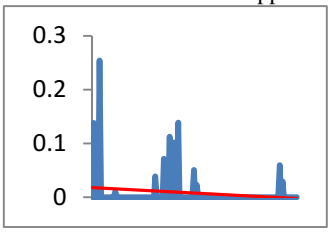
CPU Utilization of Application 1



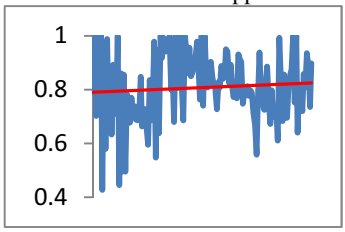
Missed Deadline Ratio of Application 2



CPU Utilization of Application 2



Missed Deadline Ratio of Application 3



CPU Utilization of Application 3

Set Point of Missed Deadline Ratio	0.05
Set Point of CPU Utilization	0.8
VM Safe Threshold	0.9

Fig. 5. Missed Deadline Ratio and CPU Utilization of Application 1, 2 and 3

6 Conclusion and Future Works

It's becoming a trend for enterprises to adopt PoI. In summary, PoI has many advantages on the flexibility, reliability and cost-efficiency. In this paper, a QoS guaranteed and cost-efficient resource management framework in PoI is proposed. The framework is composed of a feedback control system and two scaling algorithms. Experimental results show that the resource management framework can not only maintain Missed Deadline Ratio of all applications at an expected value, but also improve the CPU Utilization of all applications around an expected value. According to experimental results, it can be found that Time Usage of VMs has negative correlation with the Set Point of Missed Deadline Ratio, the Set Point of CPU Utilization and VM Safe Threshold.

In this paper, the resource management framework is designed based on a relatively simplified IaaS Layer, e.g., the action of renting and returning VMs can be finished immediately. However this assumption usually does not hold true. For example, Amazon Elastic Compute Cloud (Amazon EC2), an IaaS provider, charges on hourly basis, which makes it impossible to return VMs to IaaS providers immediately. In the future, a more practical abstraction of IaaS Layer will be involved, e.g., the potential delay in renting and returning VMs needs to be taken into account.

References

1. Chandra, A., Gong, W., Shenoy, P.: Dynamic Resource Allocation for Shared data centers using online measurements. In: Proceedings of the 11th International Workshop on Quality of Service (2003)
2. Ruth, P., McGachey, P., Xu, D.: VioCluster, "Virtualization for Dynamic Computational Domains". IEEE International on Cluster Computing, 1–10 (September 2005)
3. Menasc, D., Casalicchio, E.: A Framework for Resource Allocation in Grid Computing. In: Proceedings of the 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, pp. 259–267 (2004)
4. Yazir, Y., Matthews, C., Farahbod, R., Neville, S., et al.: Dynamic Resource Allocation in Computing Clouds using Distributed Multiple Criteria Decision Analysis. In: 3rd International Conference on Cloud Computing, Miami, Florida, USA (2010)
5. Chang, F., Ren, J., Viswanathan, R.: Optimal Resource Allocation in Clouds. In: 3rd International Conference on Cloud Computing, Miami, Florida, USA (2010)
6. Mazzucco, M., Dyachuk, D., Deters, R.: Maximizing Cloud Providers Revenues via Energy Aware Allocation Policies. In: 3rd International Conference on Cloud Computing, Miami, Florida, USA (2010)
7. Bossche, R., Vanmechelen, K., Broeckhove, J.: Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads. In: 3rd International Conference on Cloud Computing, Miami, Florida, USA (2010)
8. Cristian, F., Fetzer, C.: The Timed Asynchronous Distributed System Model. IEEE Transactions on Parallel and Distributed Systems (June 1999)
9. Hu, R., Li, Y., Zhang, Y.: Adaptive Resource Management in PaaS Platform Using Feedback Control LRU Algorithm. In: 2011 International Conference on Cloud and Service Computing (2011)

10. Ang, K.H., Chong, G., Li, Y.: PID Control System Analysis, Design and Technology. IEEE Transactions on Contr. Syst. Tech. 13(4), 559–576 (2005)
11. Astrom, K.J.: PID Controllers: Theory, Design, and Tuning. Instrument Soc. Amer. Research Triangle Park (1995)
12. Best Fit Allocation Algorithm, http://www.cs.rit.edu/~ark/lectures/gc/03_03_03.html (access on January 2013)
13. The Grid Workloads Archive, http://gwa.ewi.tudelft.nl/pmwiki/pmwiki.php?n=Home.GWA (access on January 2013)
14. Wang, Q.G., Lee, T.H., Fung, H.W., Bi, Q., Zhang, Y.: PID Tuning for Improved Performance. IEEE Trans. Contr. Syst. Tech. 7, 3984–3989 (1999)