SLA-Based Optimal Provisioning Method
for Cloud Computing Environments

Seunghwan Yoo1, and Sungchun Kim1
1 Computer Science & Engineering Department, Sogang University, Seoul, South Korea

Abstract — In cloud computing, cloud providers can offer cloud consumers two provisioning plans for computing resources, formal reservation and on-demand plans. Usually, cost of utilizing computing resources provisioned by reservation plan is cheaper than that provisioned by on-demand plan, since cloud consumer has to pay to provider in advance. With the reservation plan(Local Adjustment), the consumer can reduce the total resource provisioning cost. However, the optimal reservation of resources is difficult to be achieved due to fluctuation of consumer’s future demand and providers’ resource prices. So we propose a framework to improve their profits by maximizing the resource utilization and reducing the reconfiguration costs. Then a two-step runtime reconfiguration strategy. The SLA algorithm can provide computing resources for being used in multiple provisioning stages as well as a long-term plan(Global Adjustment), The Service Level Agreement (SLA) based scheduling approach promotes cooperative resource sharing. In this paper, minimizing both under provisioning and over provisioning problems under the demand and price uncertainty in cloud computing environments is our motivation to explore a resource provisioning strategy for cloud consumers; the results show that our framework is effective for maximizing the resource utilization and reducing the costs of the runtime reconfiguration.

Keywords: Provisioning, Service Level Agreement, Optimize, Cloud Computing Platform

1 Introduction

Cloud computing[1] is basically an Internet-based network made up of large numbers of servers - mostly based on open standards, modular and inexpensive. Also, it is popular as a rising application paradigm, where resources, including software, platform and infrastructure, are provided and shared as services. Cloud providers are responsible for various resource demands by determining where to place VMs and how to allocate the resources. The virtualization-based cloud computing can improve resource utilizations, scalabilities, flexibilities and availabilities of applications. Also it can provide good application isolations in multiple levels. Due to these advantages, large-scale distributed applications are preferred to be hosted in a cloud platform. A service-level agreement (SLA) is a part of a service contract where a service is formally defined. In practice, the term SLA is sometimes used to refer to the contracted delivery time (of the service or performance). As an example, internet service providers will commonly include service level agreements within the terms of their contracts with customers to define the level(s) of service being sold in plain language terms. This paper proposes a SLA-Based Optimal resource provisioning method (SAA provisioning Method). SAA provisioning method provides scalable processing power with dynamic resource provisioning mechanisms, where the number of virtual machine used is dynamically adapted to the time-varying incoming request workload. We evaluate the effectiveness of our initial VM deployment method used in sandpiper[19] and through runtime VM reconfiguration, we show the advantage in resource utilization.

The remainder of the paper is organized as follows. Section 2 presents survey related to our work. Section 3 describes our SAA provisioning method on the cloud computing platform. Section 4 presents our adaptive resource provisioning algorithm and its performance evaluation. Section 5 concludes the paper and points out some future research directions.

2 Related Work

Dynamic resource provisioning [4], which has been generally used in web hosting platforms, has proven to be useful in handling multiple time-scale workloads(VMs). However, dynamic provisioning in previous research has been more focus on physical resource allocation, which is not flexible enough for the effective delivering of services. Unlike other computing resources, VMs are flexibly deployed on physical machines, which can be automatically generated for different virtualized applications. Though existing physical capacity provisioning has long been used, overprovisioning or under-provisioning has been a common difficulty for most resource IT vendors. To solve this problem it is necessary to make full use of advantages of adaptive resource provisioning. We propose the design of a virtualized resource allocation framework using the cloud platform, which allocates VMs on demand in order to provide services,
as well as minimizing the cost of using those virtual resources. Nowadays, some researches have focused on the issue of resource management and performance control in cloud computing platform[5,6]. However, new challenges are introduced while service providers benefit from the planning flexibility in technical and economic aspects. Some challenges and opportunities of automated control in cloud computing is discussed in [7]. And other researchers work to improve the resource utilization, such as resource virtualization[8,16], on-demand resource provisioning management based on virtual machines [9, 10], and QoS management of virtual machine [11].

Also, many researchers [12, 13] focus on improving resource utilization as well as guaranteeing quality of the hosted services via on-demand local resource scheduling models or algorithms within a physical server. However, most of them could not be good solutions to tradeoff between resource utilization and SLA. For example, [12] present a novel system-level application resource demand phase analysis and prediction prototype to support on-demand resource provisioning. The process takes into consideration application’s resource consumption patterns, pricing schedules defined by the resource provider, and penalties associated with SLA violations. The authors in [13] improve resource utilization and performance of some services by hugely reducing performance of others. How to improve resource utilization, as well as guarantee SLA, is a challenge in a VM-based cloud data center.

In the context of the dynamic resource provisioning, the author in [16] introduce three mechanisms for web clusters. The first mechanism, QuID [14], optimizes the performance within a cluster by dynamically allocating servers on-demand. The second, WARD [15], is a request redirection mechanism across the clusters. The third one is a cluster decision algorithm that selects QuID or WARD under different workload conditions.

For multi-tier internet applications, the modeling is proposed that a provisioning technique which employs two methods that operate at two different time scales : predictive provisioning at the time-scale of hours or days, and reactive provisioning at time scales of minutes to respond to a peak load[17].

In this section, we first discuss the service level agreements (SLAs) that we use in the paper. Then we give a high-level description of the test bed and three types of workload generators for our experimental studies. Finally, we describe the control system architecture that we use throughout the paper.

### 2.1 Service Level Agreements

Service level agreements (SLAs) are firm contracts between a service provider (IT Bender) and its clients (Users). SLAs in general depend on certain chosen criteria, such as latency, reliability, availability, throughput and security, and so on. In this paper, we focus on end-to-end latency, or maintain cost. Although SLA cost function may have various forms, we believe that a staircase function is a natural choice used in the real-world contracts as it is easy to describe in natural language [18]. We use a single step function for SLA in our paper as a reasonable approximation. We assume that if response time is shorter than arranged time, then the service provider will earn some revenue. Otherwise, the service provider will pay a penalty back to the client. As a result, in order to minimize the SLA penalty cost, our method should keep the response time right below arranged time.

### 3 Proposed Scheme

#### 3.1 SAA Framework

This section presents a scalable framework for virtualized applications on the cloud computing platform. The framework deals with the scenario that hosted on a cloud computing platform, handle many virtual machines simultaneously according to the incoming user requests. Since the amount of incoming requests changes with time and the cloud platform is a pay-per-use service, the application has to dynamically assign the resources it uses to maintain guaranteed response time and reduce the total owner cost under various workloads. In the framework, server pool, combining a distinct computing server, is capable of processing multiple hybrid workload requests.

To efficiently utilize resources, there are two main issues considered in the cloud computing platforms. The first is finding the least loaded resource for dispatching incoming requests. The second issue deals with SAA provisioning for adaptively handling dynamic user’s requests. With resource state monitoring, each workflow enactment request will be sent to the least loaded resource for service. The effectiveness of least load dispatching largely depends on how to accurately capture the computing load on each resource.

![Fig. 1 Proposed SAA-Provisioning Framework](image-url)
Fig. 1 shows an overview of the framework in handling user requests for Cloud Computing Environments. The architecture consists of four main components that Initial Deployment, Runtime Monitor, Modeler & Predictor, and Runtime VM Configurator which is a loops architecture. The goal is to meet the user requirements while adapting cloud architecture to workload variations. Usually, each request requires the execution of virtualized application allocated on the VM of each physical server. A cloud computing resource amount enables multiple virtualized applications may be increased when request increases and reduced when request reduces. This dynamic resource provisioning allows flexible response time in a cloud platform where peak workload is much greater than the normal steady state.

Our Framework provides a high-level dynamic resource provision architecture for cloud computing platform, which shows relationships between heterogeneous server resources pool and self-management function. Server pool contains physical resources and virtualized resources. A lot of VMs hold several Server Pool sharing the capacity of physical resources and can isolate multiple applications from the underlying hardware. VMs of a virtualized application may correspond to a physical machine.

Self-management function means mechanisms to automate the VMs of configuring and tuning the virtualized application so as to maintain the guaranteed response time for requirements of the diverse users. As previously stated, four main components more detail explanation are as follows:

1. Runtime Monitor: Collects the runtime information, including resource usage, network load, and request arrival rate, such as the response time, the request arrival rate, the average service time, and the CPU utilization, etc.. All information is sampled periodically without affecting application performance significantly

2. Modeler & Predictor: Use data from Runtime Monitor to calculate the objective values and predict the future state.

3. Runtime VM Configurator: decides when and how to reconfigure the VMs. To reduce the runtime reconfiguration costs

4. In conclusion, Fig. 1 is presented the dynamic resource provisioning method. Our research is a great help of on the improved design of resource scheduler for requested workload. The goal is to minimize the using of resources for request workload while satisfying different users for the guaranteed response time.

### 3.2 Proposed SAA provisioning Algorithms

In this section, we propose an auto-control algorithm denoted as SAA provisioning method (SLA Aware Adaptive) to dynamically provide an adequate amount of resources to virtualized application. To maintain acceptable response time and cost efficiency, it would find the configuration value which the Sum of cloud platform profits is maximized. Considering all of virtual machine system parameters observed by monitor, especially response time and usage cost, we compute the profit value of each VMs. Through equation (1), our method calculates the optimized next step setting value. Resource scheduler receives the modified configuration parameter. Then it reflects the value next schedule period.

![Table 1](image)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>r(RA)</td>
<td>Rate of Arrival Request</td>
</tr>
<tr>
<td>r(SLAi)</td>
<td>Rate of SLA Satisfied</td>
</tr>
<tr>
<td>r(VMi)</td>
<td>Rate of VM Failed</td>
</tr>
<tr>
<td>c(VMA)</td>
<td>Active VM maintain Cost</td>
</tr>
<tr>
<td>c(VMI)</td>
<td>Idle VM maintain Cost</td>
</tr>
<tr>
<td>α</td>
<td>Created Value (per Application)</td>
</tr>
<tr>
<td>β</td>
<td>Weight Value (per Application)</td>
</tr>
</tbody>
</table>

\[
\text{Profit}(P_i) = \alpha \times \{r(RA) \times r(SLA_{Ai}) - r(VM_{Pi})\} - \beta \times \{c(VMA_{Ai}) + c(VMI_{Ai})\} \quad (1)
\]

After each Server-Pool Profit(Pi) is calculated, Periodically it is updated and check SLA-requirements. After the specific point which variability is minimized, Our Scheme elect optimized parameter for Global Profit.

\[
\max \{-profit_{global} = \sum_{i=1}^{n} P_i \} \quad (2)
\]

After all, our mechanism collect local profit and calculate Global profit as shown in (2). It would find the optimal value for certain period. Also it is adaptively perform in the course of time. Since it has sufficient information about virtualized application. For example, there are a little difference between current parameter and next-step parameter. It check prefixed threshold. If it is not exceed, retain the system current parameter. Finally, Our SAA provisioning algorithms would
find SLA- guaranteed response time and low maintenance cost.

4 Performance Analysis

4.1 Experiment

In the following experiments, we evaluate our dynamic resource provisioning technique for virtualized applications. We establish a prototype system of cloud environment such that each of the server nodes was run on Intel Xeon 3.2GHz processors with 24GB RAM. Processing capacity of each VM server is equal in cloud platform.

We evaluate the effectiveness of our initial VM deployment method used in sandpiper[18]. The VM template capacities and application demands are distributed in the following sets: CPU-{0.25*2.4, 0.5*2.4, 1*2.4, 1.25*2.4, 1.5*2.4, 2*2.4, 3*2.4, 4*2.4}, memory-{0.5, 1, 1.5, 2.0, 3.0, 4.0, 6.0}, network I/O-{4, 6, 10, 15, 20}. For example, Physical server’s total capacity is {4*3.4GHZ, 24GB, 100M}.

We employ three types of applications, CPU-intensive (CI), Memory-intensive (MI) and Network I/O-intensive (NI), with multiple instances. Multiple VM templates are provided and allocated to these applications. TABLE 2, 3, 4 describe the servers’ uses, VM template configurations and application instances respectively.

4.2 Experiment result

Existing method, focus on maximizing resource utilization is approximately demonstrated 87% SLA-satisfied rate. We give consideration to improve SLA-satisfied ratio. Our mechanism indicate settlement for content better SLA-satisfied rate and diminish maintain cost.

Fig. 3 is our deployed VM template simulation result. Each type application requires unpredictable demand. So there are variable response time. We should stable cloud platform performance due to such fluctuation.

<table>
<thead>
<tr>
<th>ID</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>File system for VM migration</td>
</tr>
<tr>
<td>S2</td>
<td>Client workload generator for applications</td>
</tr>
<tr>
<td>S3</td>
<td>Request router, distributing client requests</td>
</tr>
<tr>
<td>S4 ~ S7</td>
<td>Hosting applications packaged into VM</td>
</tr>
</tbody>
</table>

Table.3 VM Template

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>Common</td>
<td>0.5*2.4GHZ, 1GB RAM, 10M I/O</td>
</tr>
<tr>
<td>V2</td>
<td>High-CPU</td>
<td>2*2.4GHZ, 1.5GB RAM, 20M I/O</td>
</tr>
<tr>
<td>V3</td>
<td>High-Mem</td>
<td>1*2.4GHZ, 3GB RAM, 20M I/O</td>
</tr>
<tr>
<td>V4</td>
<td>High-I/O</td>
<td>1*2.4GHZ, 1GB RAM, 30M I/O</td>
</tr>
</tbody>
</table>

Table.4 Application Instances and Allocation

<table>
<thead>
<tr>
<th>App Type.</th>
<th>App. ID</th>
<th>VM Template</th>
<th>Instance Number</th>
<th>Server ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU-Intensive</td>
<td>CI-1</td>
<td>V2</td>
<td>2</td>
<td>S4, S5</td>
</tr>
<tr>
<td></td>
<td>CI-2</td>
<td>V2</td>
<td>2</td>
<td>S4, S5</td>
</tr>
<tr>
<td>Mem-Intensive</td>
<td>MI-1</td>
<td>V3</td>
<td>2</td>
<td>S4, S5</td>
</tr>
<tr>
<td>I/O-Intensive</td>
<td>NI-1</td>
<td>V4</td>
<td>2</td>
<td>S4, S5</td>
</tr>
</tbody>
</table>
Our method conducts initiation and removal of VMs before each interval while considering the utilization of the previous interval. It is noticeably cost-aware. And the results of response times and costs are shown in Fig. 4 and Fig. 5.

5 Conclusions

In this paper, it is argued that dynamic provisioning of virtualized applications environment raises new challenges not addressed by prior work on provisioning technique for cloud computing platform. We presented an optimal autonomic virtual machine provisioning architecture. We proposed a novel dynamic provisioning technique, which was algorithms for virtualized applications in cloud computing platform. Hence the efficiency and flexibility for resource provisioning were improved in cloud environment.

Currently many server applications adjust the amount of resources at runtime manually. So we address the problem of the VM deployment and reconfiguration. The framework in this paper allows applications to automatically manage the amount of resources according to the system workload. It offers application providers the benefits of maintaining QoS-satisfied response time under time-varying workload at the minimum cost of resource usage. Also, we adopt Service Level Agreement (SLA) based negotiation of prioritized applications to determine the costs and penalties by the achieved performance level. If the entire request cannot be satisfied, some virtualized applications will be affected by their increased execution time, increased waiting time, or increased rejection rate.

However, there are still some limitations in our work, including: 1) the prediction techniques may have observational error which affect the runtime VM reconfiguration decisions; 2) the interferences between VMs on a hybrid server type are ignored. In future work, we will focus on these limitations by applying the much more accurate prediction techniques and consider hybrid server architecture.

ACKNOWLEDGMENTS

This work was all-supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) (No. 2012R1A1A2009558), and part - supported by Technology Innovation Development Program funded by Small & Medium Business Administration(No. S2057016)

6 References


