# A Pattern-Based Prediction Model for Dynamic Resource Provisioning in Cloud Environment

**Hyukho Kim, Woongsup Kim and Yangwoo Kim**
Dept. of Information and Communication Engineering, Dongguk University
Seoul, 100-715, South Korea
[e-mail: {hulegea, woongsup, ywkim}@dongguk.edu]
*Corresponding author: Yangwoo Kim

*Received April 7, 2011; revised August 16, 2011; accepted September 22, 2011; published October 31, 2011*

## Abstract

Cloud provides dynamically scalable virtualized computing resources as a service over the Internet. To achieve higher resource utilization over virtualization technology, an optimized strategy that deploys virtual machines on physical machines is needed. That is, the total number of active physical host nodes should be dynamically changed to correspond to their resource usage rate, thereby maintaining optimum utilization of physical machines. In this paper, we propose a pattern-based prediction model for resource provisioning which facilitates best possible resource preparation by analyzing the resource utilization and deriving resource usage patterns. The focus of our work is on predicting future resource requests by optimized dynamic resource management strategy that is applied to a virtualized data center in a Cloud computing environment. To this end, we build a prediction model that is based on user request patterns and make a prediction of system behavior for the near future. As a result, this model can save time for predicting the needed resource amount and reduce the possibility of resource overuse. In addition, we studied the performance of our proposed model comparing with conventional resource provisioning models under various Cloud execution conditions. The experimental results showed that our pattern-based prediction model gives significant benefits over conventional models.

***Keywords:*** Cloud computing, service, resource provisioning, prediction, pattern

The part of this paper was presented in the ICONI (International Conference on Internet) 2010, December 16-20, 2010, Philippines. This research was supported by the MKE(Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency)(NIPA-2011-C1090-1101-0008). This work was supported by the Dongguk University Research Fund of 2011.

**DOI: 10.3837/tiis.2011.10.002**

## 1. Introduction

Cloud computing is a new computing paradigm composed of a combination of grid computing and utility computing concepts. Cloud provides dynamically scalable virtualized computing resources as a service over the Internet; hence, they can handle task scheduling, resource provisioning, and allocation over the virtual clusters [1]. Cloud computing is implemented with a data center or computing center based on virtualization technology, where data centers are used to provide services to clients through the Internet. Due to its dependency on virtualization technology, Cloud can handle more sophisticated task scheduling and resource allocation mechanisms, such as self-managing and self-healing resource provisioning systems [2]. Therefore, virtualization can potentially benefit data centers, reducing the need for over-provisioning and hence the costs.

Virtualization technologies allow one to create multiple Virtual Machines (VMs) on one or more physical servers and allocate the necessary network and data center resources in dynamic ways. IaaS (Infrastructure-as-a-Service) providers run customers' VMs as many as possible to fully utilize their data center capacity. Once management systems assign VMs on physical machines, they execute VMs to run applications which provide services. The number of VMs deployed on physical nodes should be enough to accommodate all of the service requests from users, while the number of deployed VMs should not be assigned over the cloud capacity.

In Cloud, applications providing services are run on VMs, and their operations are dynamically determined to best match the delivery needs for the current user requests. As the population of user requests changes, the population of VMs provisioning services should vary to accommodate all of the service requests. For maximum service availability, Cloud management systems need to decide the number of VMs on physical machines in advance, calculating each VM's resource capability. In addition, as it is always possible that deployed VMs run idle occupying resources and consuming energy, virtualized technology should always simultaneously take the redundancy elimination ability into account.

In Cloud, both guaranteed service provisioning and maintaining higher resource utilization are required, which are dependent on the optimization strategy over VM allocation and release. To support guaranteed service provisioning and high computing resource utilization simultaneously, Cloud management systems need to have the ability to predict the application performance to decide when to turn-on and turn-off VMs on physical nodes. A VM turn-on/turn-off process usually involves a VM booting time and an on-line migration time for service reallocation, which could cause delays in service delivery. From our experiment, in a normal environment, VM booting time takes more than 1 minute, and systems should wait that amount of time, resulting in service delays. Performance prediction can be used for detecting when to prepare for VM booting and service migration in advance. Hence, VM preparation is complete before VM allocation/release is actually needed. Good prediction, we believe, creates guaranteed service delivery by eliminating VM preparation time in dynamic service provisioning environments. Performance is typically predicted by means of aspects of system behavior under load. From this context, heavy load prediction should increase more VM deployment, and light load prediction should withdraw the existing VM which runs idle.

In this paper, we propose a pattern-based prediction model for resource provisioning which facilitates best possible resource preparation by analyzing the resource utilization and deriving resource usage patterns. The focus of our work is on predicting future resource requests by

optimized dynamic resource management strategy that is applied to a virtualized data center in a Cloud computing environment. To this end, we build a prediction model that is based on user request patterns and make a prediction of system behavior for the near future.

We consider two fundamental aspects when we design a prediction model. First, the population of Cloud user requests varies over time. Thus, the prediction should count recent user requests more than older ones, and be updated incrementally on on-line basis over time: old usage information should be discarded and the most recent usage information should be added at the same time. Second, the pattern selection process should be simple enough to reduce computational overhead for the Cloud management system and finish within a short time interval of two consecutive information collecting actions.

Our proposed model generates the usage patterns for the given timeframe with information which is from user history on VMs. We noticed that well-defined usage patterns could be updated incrementally, weighting usage history recorded recently. Moreover, using well-defined patterns could save prediction time as the system can avoid complex numeric computations, such as simple pattern matching algorithms. As a result, we have defined patterns that are incrementally updatable on-line and designed a prediction model to use those patterns.

We also employ sliding windows to collect the number of recent usage histories and support incremental updates, in order to capture of the change in resource utilization. Our resource usage pattern consists of resource types, service types, and time using the services. Our model can do resource provisioning in advance because it can predict the expected resource amount through the generated resource patterns. To reduce computational overhead, we employ a simple pattern matching algorithm. As a result, this model can save time for predicting the needed resource amount and reduce the possibility of resource overuse. We studied the performance of our proposed model using comparative experiments with two conventional models, such as threshold-based and Fuzzy-based models, under various Cloud execution conditions. The experimental results show that our pattern-based prediction model gives significant results, compared to other resource provisioning models.

The rest of this paper is organized as follows. Section 2 summarizes related works for Cloud and virtualization technologies as well as approaches in resource prediction in Cloud. In section 3, we introduce our pattern-based prediction models. Section 4 presents the detail of our findings and performance evaluations. Finally, we conclude our contributions and discuss future works in section 5.

## 2. Related Work

### 2.1 Cloud Technologies

Recently, a great interest in Cloud computing has been manifested from both academic and industry, and numerous projects from industry and academia have been conducted. The concept of cloud computing comes from the Amazon Elastic Compute Cloud (EC2) [3], which is based on a simple idea: to offer a set of web services and a command line interface that lets users manage (create, destroy, migrate, etc.) virtual machine images on the Amazon data center. Amazon sells CPU time, which is used by users' virtual images.

Starting from EC2, a large set of technologies has been successively developed. In commercial contexts, it is worth mentioning IBM's Blue Cloud [4], Sun Microsystem's Network.com [5], Microsoft Azure Services Platform [6], Google App Engine, and Dell Cloud

computing solutions [7]. Most of these commercial systems adopt proprietary solutions, such as the virtualization engine by VMWare [8], and relatively few details are available on the adopted architecture. Even if the cloud concept is born in the commercial environment, it is simply an evolution of the virtualization techniques that have been the object of research in recent years. The scientific world has offered many similar solutions in the past, such as the idea of the Cluster on Demand proposed in [9]. In the state-of-the-art context, the most advanced research project is the Reservoir project [10], which includes technologies such as OpenNebula [11]. The most widely adopted virtualization engine is Xen [12], followed by less popular alternatives such as Virtualbox [13] and KVM [14].

A data center is a collection of computers connected through a high-speed network. Data centers tend to run a large number of physical servers at a low resource utilization level [15][16]. Virtualization could enable server consolidation, increasing the resource utilization level and decreasing the necessary investment in equipment. Server consolidation could result in effectively reducing over-provisioning, but at the expense of increasing resource management complexity. Now there is not only one computer system running on the physical server, but several. It is necessary to monitor each virtual machine's workload and dynamically adjust resource allocation on-demand. Different virtual machines may run applications that have competing quality of service requirements that may not be satisfied at a given point in time. Thus, for server consolidation to be effective, improved resource management mechanisms need to be devised. Mechanisms are needed that can automatically respond in a timely fashion to the unpredictable, time-varying demand experienced by the virtual machines running in a physical server and for the hundreds of servers that could be running together in a data center.

### 2.2 Resource Provisioning and Prediction under Virtualization

Considerable work has been published to address different issues of resource management in large data centers. Since Walsh [17] began to explore the use of utility function in practical autonomic computing systems, more recent work [18][19][20][21] has focused on utility-based approaches rather than traditional action policies or goal policies. Similarly, in this paper, a utility function concerning both the provisioning policy and the amount of unused servers is employed in order to maximize or minimize resource utilization. The following briefly summarizes other work containing some common elements with this paper’s approach.

There has been significant research in the area of dynamic resource partitioning, which means that resources can be acquired and released on demand. Such architectures can be classified as employing share server utility (many services share a server at the same time) or full server utility (each server offers one service at a time) models. Several researchers have focused on approaches utilizing the full server utility model. In the work of [17][18][19][20], a utility-based self-optimizing architecture was proposed, and a prototype system was established and then commercialized by collaborating with current available products. Bennani [21] addressed the resource allocation problem in a data center hosting several application environments. Analytic performance models were used to design controllers that dynamically switch servers from one application environment to another as needed. Besides, a cluster manager control algorithm called QuID was devised in [22], which focused on issues such as how many servers to allocate to an application and how to migrate applications. These resource optimization techniques rely on the ideal environment that all servers are virtualized. Our model also uses similar environment based virtualization technology. However, our model differs from these papers in that herein, a full virtual server model is adopted, which means that physical resources are shared.

There are considerable efforts about predicting CPU availability using time series analysis. Wolski et al. [44][45] presented and evaluated nine linear CPU load prediction models in network weather service. Yang et al. [46] also proposed time series based prediction methods considering ascending and descending CPU load dynamics. Yaik et al. [47] applied data mining methodologies for time series load prediction using associate rules. Methodologies using time series analysis are designed and work reasonably well for one step-ahead predictions. However, these methodologies reveals weakness for long term predictions required for efficient management in virtualized resource provisioning. In addition, rule-based approaches are proposed using a set of event-condition-action rules that are triggered when some precondition is satisfied (e.g. when some metrics exceed a predefined threshold). For example, the HP-UX Workload Manager [23] allowed the relative CPU utilization of a resource partition to be controlled within a user-specified range, and the approach of Rolia et al. [24] observed resource utilization (consumption) by an application workload and uses some "fixed" threshold to decide whether current allocation is sufficient or not for the workload. With the growing complexity of systems, even experts find it difficult to define thresholds and corrective actions for all possible system states.

Approaches based on control theory have been applied to resource management to achieve performance guarantees. Most of the work assumed a linear relationship between the QoS parameters and the control parameters, and involved a training phase with a given workload to perform system identification. Typically, control parameters must be specified or configured offline and on a per-workload basis. Abdelzaher et al. [25] investigated this approach for QoS adaptation in Web servers. In [26][27], a nonlinear relation between response time and CPU allocation to a Web server was studied, and a bimodal model was used to switch between underload and overload operating regions. To deal with time-varying workloads, more recent work applied adaptive control theory, in which models were automatically adapted to changes using online system identification. Model-based research efforts [28] [29] [30] [31][32] have been trying to model computer systems from different perspectives. Bennani et al. [33] predicts the response time and throughput for both online and batch workloads using multiclass open queuing networks. Liu et al. [34] used AR models to map CPU entitlement to the mean response time with a fixed workload. Chandra et al. [28] modeled the resource using a time-domain queuing model which relates the resource requirements to its workload. Some of these approaches made simplifying assumptions, such as using a single queue to model the whole system, which could fail to capture complexities of the relationship between application workload and resource usage. Some models were validated only using simulations. However, our work is based on the analysis of the usage pattern between several performance metrics and pattern values. We apply the pattern values to resource utilization analysis for allocation and release request prediction.

Some researches [35][36][37][38][39][40] also proposed similar model-based approaches, whereas the unique aspect of the proposed approach is that it has combined patterns and the utilization dynamics to predict the transient behavior of application workloads. Diao et al. [41] proposed a profit-oriented feedback control system for maximizing SLA profits in web server systems. The control system applied fuzzy control to automate the admission control decisions in a way that balanced the loss of revenue due to rejected work against the penalties incurred if admitted work had excessive response times.

Our proposed approach differs from the prior works in the following aspects: the resource allocation and release process is automatically done without any human intervention. Our approach relates the application resource requirements to their dynamic changing workload characteristics on an on-line basis. The predictions are updated continuously as new data

arrives, which enables our pattern-based model to capture transient or unexpected workload changes.

## 3. Pattern Based Prediction Model

### 3.1 Requirements of Usage Prediction

In Cloud computing, dynamic optimization virtual machine technology can play an important role for high resource utilization, especially for both service availibility and energy efficiency. To achieve high resource utilization, a Cloud management system should be able to allocate an additional VM to the service when the resource usage is expected to be overcrowded with user requests and withdraw VMs when there are VMs running idle. However, VM allocation/release involves system preparation time, such as booting time and service migration time, which could delay service delivery. Therefore, we need to use prediction to avoid delays from preparation periods, as accurate prediction mechanisms would reduce preparation time as much as possible in VM allocation and withdrawal involved in dynamic virtualization strategy.

In designing a model for accurate prediction, we identified two major aspects that work as requirements in dynamic virtualization strategy in Cloud as follows.

1. The prediction model should be on on-line basis. The model should incrementaly insert the most recent resource usage history and delete the most non-important usage history automatically. In addition, the size of usage history data should be enough to make predictions covering at least VM preparation time and should be small enough not to cover old history, which may influence prediction accuracy.
2. The prediction function should be as simple as possible such that minimal computational overhead is guaranteed. The overall computations needed for prediction should not draw computational overhead in the Cloud management system, and a prediction computation time should not exceed two consecutive usage collection intervals such that the most recent usage history is included when making predictions.

We designed a pattern-based prediction model that satisfies the requirements described above. To satisfy the first requirement, we employed a sliding window in which new history data was inserted, and the oldest data is removed in a given time interval. The slidng window enables incremental usage data management, and we can decide appropriate data sizes or time simply by selecting and adjusting the windows size. We also design a prediction algorithm such that predictions can be computed in an incremental manner. For the second requirement, we employed a simple method, where patterns are identified into numeric values, and they can be added up to be used for VM allocation/release decisions later. Our pattern-based prediction is composed of two phases: 1) the phase of extracting pattern values, and 2) the phase of pattern-based prediction. We will describe these two phases in section 3.2 and 3.3, respectively.

### 3.2 Usage Patterns

Our proposed model utilizes the flow of resource usage history to generate usage patterns. Resource usage flows are obtained from log history data that is collected from virtual machines. We basically identified two general cases of resource usage flow: 1) usage increase

and 2) decrease. Then, we identified the current gradient of utilization increase/decrease dynamics (**Fig. 1**), and categorized the gradient into several types.

**Table 1** shows patterns types (referred to as pattern values as they are expressed numerically) and their conditions. Each pattern type is expressed with its own numeric value such that they can be summed up and used for predictions later. The numeric pattern value is obtained from the condition described in **Table 1**, and the condition indicates the gradient of recent resource utilization dynamics. We obtained the gradient of the utilization flow using tangent function and trigonometrical function in **Fig. 2** .

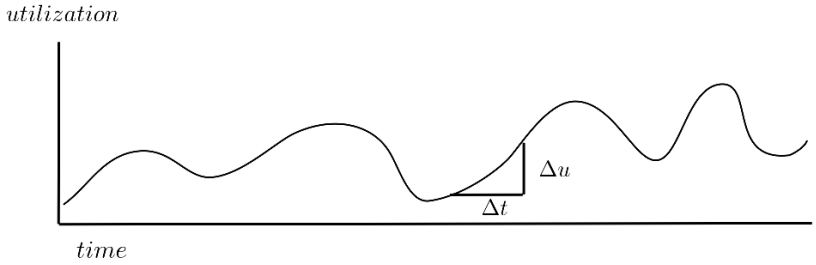


**Fig 1**. Gradient from utilization flows

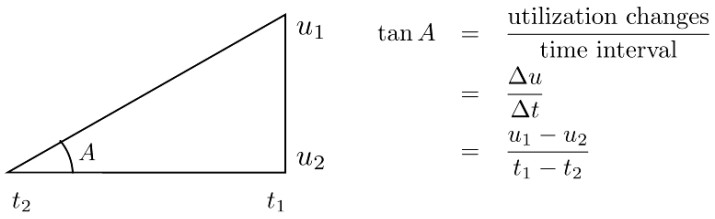


**Fig 2**. Trigonometrical function

**Table 1**. Patterns values and their description

| Type (Pattern Value) | Description | Condition |
|---|---|---|
| 0 | The gradient of resource utilization increase is less than π/12 | 0 ≤ *tan A* < 0.27 |
| 1 | The gradient of resource utilization increase is within [ π/12, π/6 ] | 0.27 ≤ *tan A* < 0.58 |
| 2 | The gradient of resource utilization increase is within [ π/6, π/4 ] | 0.58 ≤ *tan A* < 1 |
| 3 | The gradient of resource utilization increase is greater than π/4 | 1 ≤ *tan A* |
| -0 | The gradient of resource utilization decrease is less than π/12 | -0.27 ≤ *tan A* < 0 |
| -1 | The gradient of resource utilization decrease is within [ π/12, π/6 ] | -0.58 ≤ *tan A* < -0.27 |
| -2 | The gradient of resource utilization decrease is within [ π/6, π/4 ] | -1 ≤ *tan A* < -0.58 |
| -3 | The gradient of resource utilization decrease is greater than π/4 | *tan A* < -1 |

Let the two consecutive time-intervals be $t_1$ and $t_2$, and resource utilization rate at each time interval be $u_1$ and $u_2$; then, we can obtain the degree tan. After we obtain the gradient of utilization flow, we can extract patterns of current utilizations dynamics using **Table 1**. For example, if $\Delta u$ is 30 and $\Delta t$ is 15 from **Fig. 1**, we can calculate *tan A* and get 2. Then, we can extract its corresponding pattern value from **Table 1**. As *tan A* is 2 (more than 1), we can identify that the gradient belongs to pattern value 3.

### 3.3 Resource Allocation and Release Prediction from Patterns

Our prediction approach uses pattern values obtained from the recent gradients of utilization flow, as described in the previous section. In this section, we present a prediction algorithm that, using pattern values, is implemented to predict resource usages in the near future. Like the pattern values obtained with simple calculation, our prediction model is designed to reflect two apsects of 1) simple calculation and 2) incremental update, while attempting to maintain prediction accuracy.

Our algorithm for predictions is described as follows. Let $pv_i$ be the pattern values measured at time $t_i$, and let $k$ be the number of utilization measurements used in the prediction. Then, we define $U_i$ , the predictive utility measured at time $t_i$, as

$$U_i = \sum_{j=i-k}^{i} pv_j$$

Then, the utility $U_i$ is obtained from pattern values within a time frame. Our model utilizes the utility $U_i$ to decide whether to allocate a new VM or to release the existing VM. Our model maintains two types of threshold values for the VM allocation/release decision: 1) pv threshold ($U^L$,$U^H$) and 2) utilization threshold ($u^L$, $u^H$), where $U^L$ is the lower threshold utiliy value, $U^H$ represents the upper utiliy value, $u^L$ stands for the lower threshold regarding resource utilization rate, and $u^H$ denotes the upper threshold regarding the resource utilization rate. A *pv* threshold is checked when we compute utility $U_i$, whereas the utilization threshold is checked when we measure resource utilization rate $u_i$ , at time $t_i$ . Our model decides to release the existing VM only when $U_i < U^L$ and $u^i < u^L$, and it allocates a new VM only when $U_i > U^H$ and $u^i > u^H$.

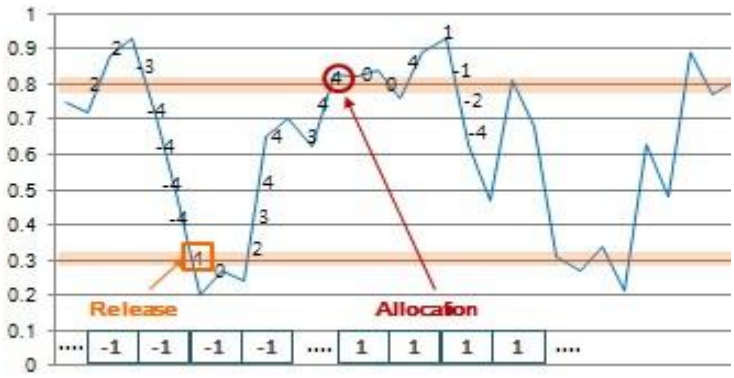


**Fig. 3**. Pattern values from resource utilization flow

**Fig. 3** shows pattern values which are derived from utilization data collected at five-minute interval and illustrates how upper/lower threshold values and pattern values work together. Our proposed model uses these values to decide whether the system needs to allocate additional resources or release some of currently assigned resources. In our model, system collects resource utilization rate at each of five-minute timeframe and then compares utilization rates collected in adjacent timeframes to derive pattern values. The numeric values in right-bottom boxes in **Fig. 3** show pattern values measured at each timeframe. As pattern values in four adjacent time-frames in **Fig. 3** are (-1, -1, -1, -1) totaled -4, the system compares current utilization rate with lower threshold value. At a point current utilization rate is under 30% and totaled pattern value is -4 (shown as a boxed number marked as "Release" in **Fig. 3**), systems decide to release an available resource. Likewise, values in middle-bottom boxes are (1, 1, 1, 1) totaled 4 and resource utilization rate is over 80% (the circled number marked "Allocation" in **Fig. 3**). So the system makes decision to allocate an additional resource.

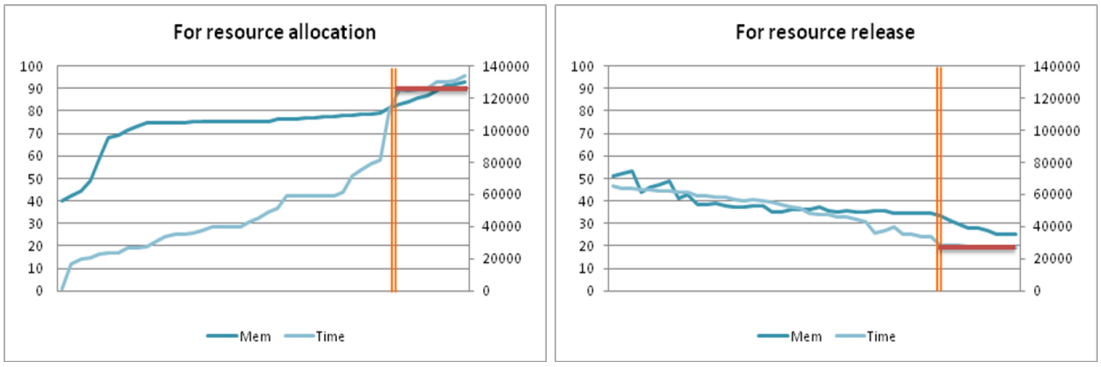


**Fig 4**. Response times for resource (memory) usage rates

Since threshold values provide a critical role in deciding when to allocate/release VMs, we evaluated memory usage rate and the corresponding response time to figure out which threshold value can provide the most efficient VM management performance. **Fig. 4** illustrates the relationship between VM's response time and resource utilization rate. As a result, we determined upper threshold to be 80% because as the response time increases dramatically at the point around the memory usage rate 80% and we thought the system needs to allocate new VM to reduce response time at that point. And we determined the release point to be 30% because the response time is almost same when the resource utilization is less than 30%. Therefore we conclude it would be OK to release a VM when the system usage rate is less than 30%.

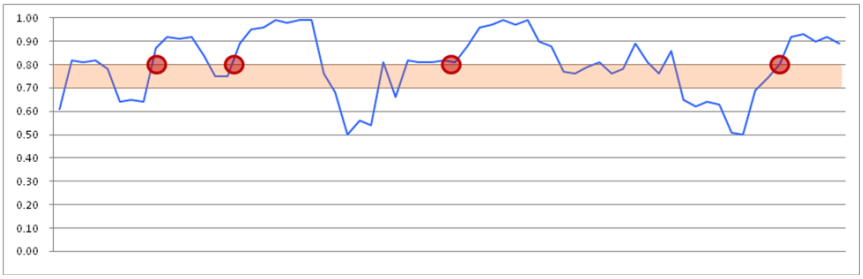


**Fig 5**. The resource utilization flow graph

In addition, we conducted an experiment to find the most appropriate timeframe to allocate/release resources. We set three requirements for deciding time interval values: 1) to reflect utilization dynamics sufficiently avoiding incorrect optimistic decisions, 2) to reduce mis-prediction of allocation/release actions, and 3) to keep the VM preparation and migration time within each time-interval. **Fig. 5** is the resource utilization flow graph that it is generated from our experimental environment with a number of *random data*. Red colored circles in **Fig. 5** indicate timeframes after which resource utilization rates continue to be over 80% for at least 25 minutes. **Table 2** shows prediction performance – the rate of successive prediction - with various time-intervals such as 2min, 5min, 10min, 20min, and 30min. As you can see from **Table 2,** 2 minute interval is too small as there can be too many mis-predictions, and 10 minute-interval is too long as it misses a chance of needed VM allocation and may result in a service delay. Thus we pick 5-minute as the best time interval for our model. In **Table 2**, *appropriate* requests is defined to be the request after which resource utilization rates continue to be over upper threshold for some periods (in our experiment, 20 minutes). If resource utilization rate decreases to the rate under upper threshold right after additional resources are allocated, then the system may have unnecessary resource allocation as system can manage all the resource usages without additional resources. We define requests occurred in this case as *inappropriate* requests. Likewise, if resource utilization rates continue to be over upper threshold for some time periods without any additional resource allocation, the system may have service delay due to resource over-utilization. We call requests occurred in this case as *missing* requests.

**Table 2**. Prediction success rate based on time-interval values

| ***Time Interval*** | ***Total number of allocation/release requests*** | ***Appropriate requests*** | ***Inappropriate or missing requests*** | ***Prediction Success Rate*** |
|---|---|---|---|---|
| 2min | 9 | 4 | 5 | 44% |
| 5min | 5 | 4 | 1 | 80% |
| 10min | 3 | 3 | 1 | 80% |
| 20min | 3 | 3 | 1 | 80% |
| 30min | 2 | 2 | 2 | 50% |

## 4. Evaluation

### 4.1 Experiment Configuration

For our experiments, we used 6 physical nodes. Each physical node is configured as dual core 3.00 GHz CPUs, 1GB Memory, 120 GB local disk space, and connected to the others by High-speed ethernet. We used only a 1Gbps Ethernet for the network communication. Every node is operated under Fedora Linux Core 8, and has own local disk and shares the home directory through NFS. To implement virtualization, each physical node includes Xen and Globus and is capable of operating at most two VMs (**Fig. 6)**. To provide sufficient load on each physical node for measuring of system performance, we designed and implemented Indexer Service in Grid IR system. We used TRAC data (784MB of 49,026 HTML files) for

indexing. And we used additional tools for experiments such as Apache Ant 1.6.5, Apache Lucene 2.4.1, and Apache Log4j-1.2.8.

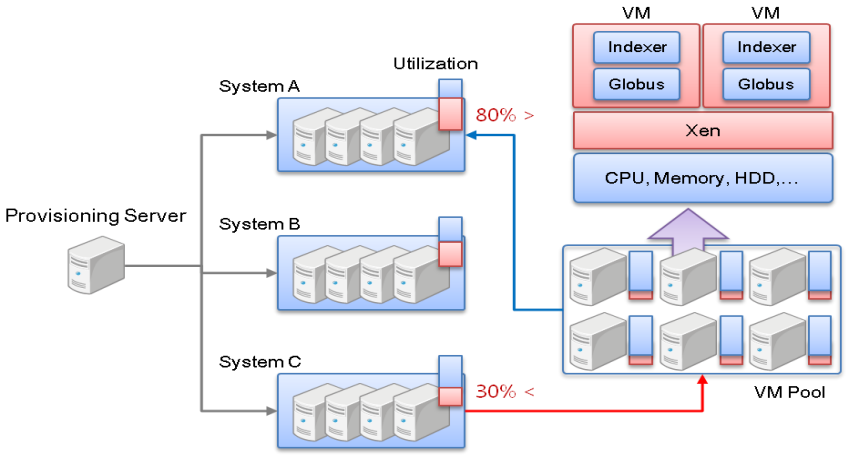


**Fig 6**. The Experimental environment

## 4.2 Overheads for Virtualization

Because we use Xen and Globus middleware for virtualization, we conducted experiments to know how the virtualization impacts on indexing performance. We compared 4 types of system configuration – None, GT, Xen, and GTX (GT and Xen) for overhead analysis. We used the processing time and resource utilization as virtualization overhead indicators.

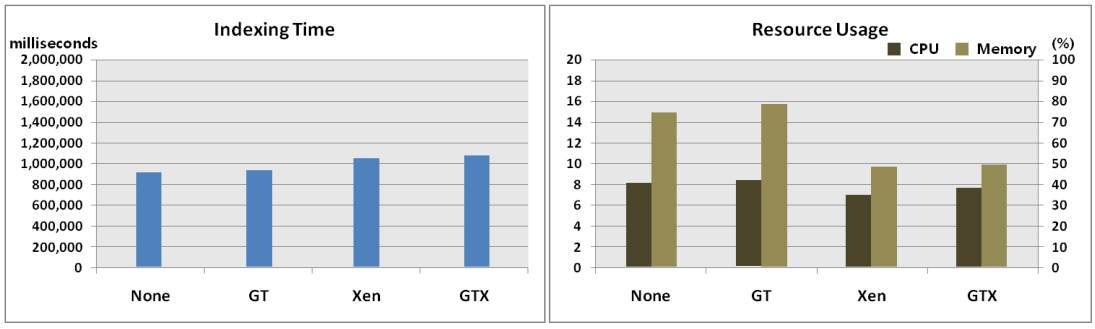


**Fig 7**. Performance comparisons with one VM for one physical node

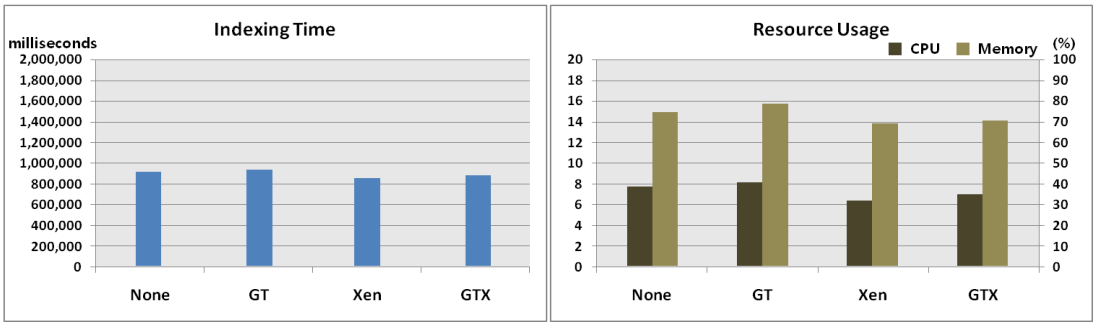


**Fig 8**. Performance comparisons with two VMs for one physical node

**Fig. 7** and **Fig. 8** show indexing time and resource utilization rate with the various number of VMs per one processor. As a result, we found that virtualization middlewares such as Xen and Globus do not bring on noticeable system overhead. We also identified the similar results in [42].

### 4.3 Analysis Using Resource Usage History

We conducted experiments with the existing resource history which was compiled in advance. We compared experimental results with those of threshold-based provisioning and Fuzzy-based provisioning [43]. In the threshold-based model, the upper threshold value is set to 80% for resource allocation and the lower threshold value is set to 30% for resource release. Fuzzy-based model used the fuzzy logic to deal with the complexity of the virtualized data center and the uncertainties of the dynamically changing workloads.

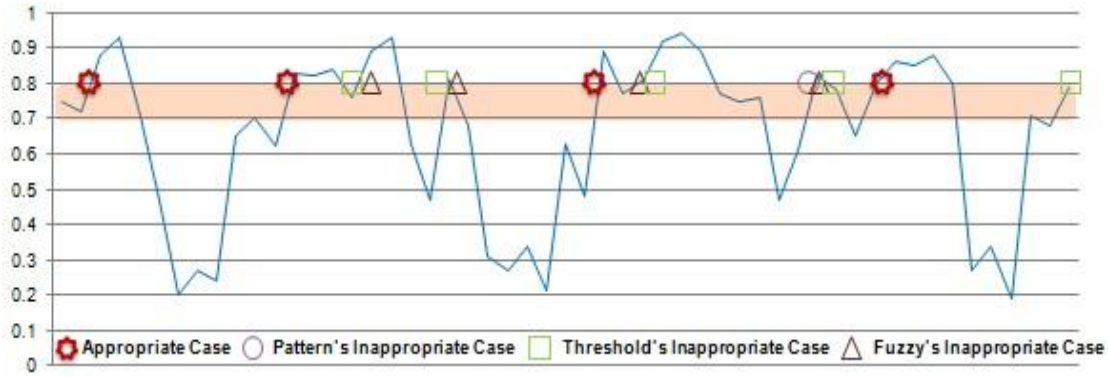


**Fig 9**. Allocation analysis from the existing resource usage data

**Fig. 9** shows analytical results in terms of resource allocation. We compared the results from three approaches - pattern-based, threshold-based, and fuzzy-based one. The number of resource allocation in threshold-based model was eight and only four of the eight requests were *appropriate*. In fuzzy-based model, the result is similar to that of threshold-based mode. In contrary, the number of resource allocation in pattern-based model was five and four of the five allocations were *appropriate*.

**Table 3**. Comparisons of three approaches for resource allocation decision success rate

| | ***Appropriate*** | ***Inappropriate*** | ***Total*** | ***Rate*** |
|---|---|---|---|---|
| Threshold | 4 | 4 | 8 | 50% |
| Fuzzy | 4 | 4 | 8 | 50% |
| Pattern | 4 | 1 | 5 | 80% |

The results in **table 3** indicate that the success rate of the threshold-based resource allocation and fuzzy-based resource allocation were 50% each. However, the *appropriate* rate of pattern-based resource allocation was 80%. As a result, the proposed approach seems to be 30% better than other approaches.

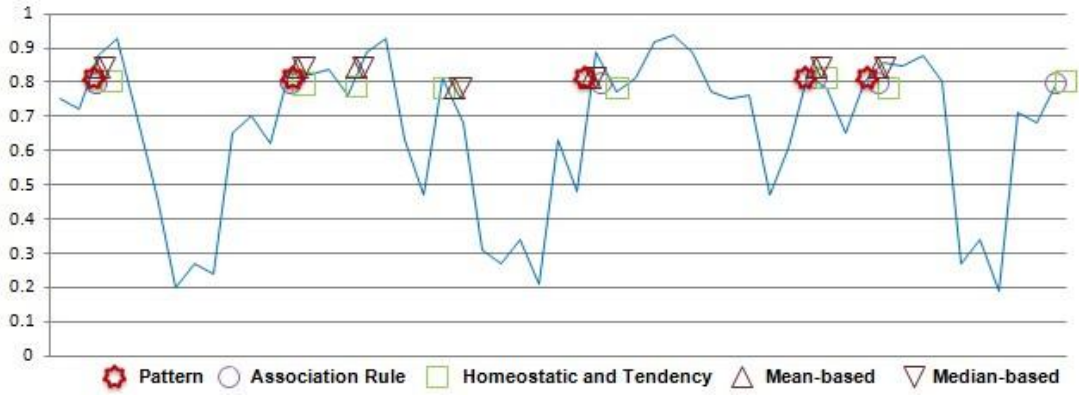


**Fig 10**. Performance comparison for time-series predictors

**Fig. 10** illustrates performances of existing predictors using time series analysis (mean-based [44], median-based [44], association rule[47], homeostatic and tendency [46]) and our approach (pattern) using 5-minute interval usage history data. Results show that existing predictors make more *inappropriate* allocation requests than our approach does. We speculate the reason would be that existing predictors focos on only one step-ahead predictions and hence reveal weakness in terms of long-term prediction, which is required to make *appropriate* prediction case. As shown in **Fig. 10**, association rule based approach generates eight allocation requests, homeostatic and Tendency approach has five allocation requests, and both mean-based approach and median-based approach make seven allocation requests. However our proposed approach has five allocation requests, which is minimum.

**Fig. 11** shows analytical results for the resource release case from the pattern-based, threshold-based, and fuzzy-based approach. The number of resource release requests in the threshold-based model was five and only three times of total requests were detected *appropriate*, and the result in fuzzy-based model was similar to threshold-based model's one. However, the pattern-based model makes only three requests and all of the requests were *appropriate*.

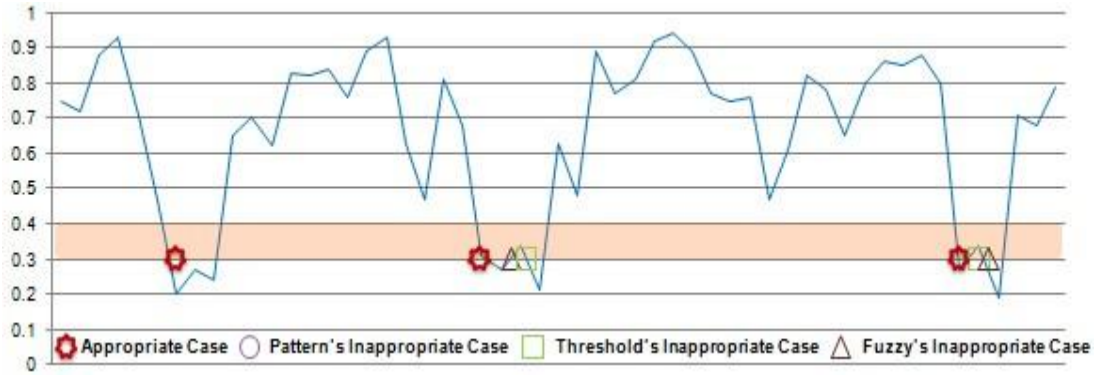


**Fig 11**. Resource release with the existing resource history

The result in **table 4** shows that success rate of threshold-based resource release and fuzzy-based resource release was 60% and the success rate of pattern-based model was 100%. As a result, the proposed model is presented 40% better than other two models.

**Table 4**. Comparisons of three approaches for resource release deicsion success rate

| | *Appropriate* | *Inappropriate* | *Total* | *Rate* |
|---|---|---|---|---|
| Threshold | 3 | 2 | 5 | 60% |
| Fuzzy | 3 | 2 | 5 | 60% |
| Pattern | 3 | 0 | 3 | 100% |

### 4.4 Analysis in a physical node

We also conducted experiments with a server that are overloaded from a task generator in real time. We compared results with threshold-based provisioning, fuzzy-based provisioning, and our pattern-based provisioning.

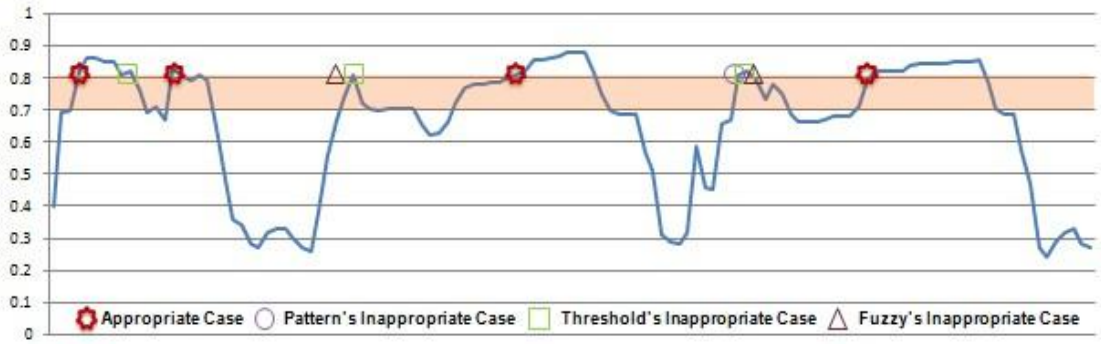


**Fig 12**. Decisions for resource allocation from real time usage monitoring

**Fig. 12** shows results from experiments we conducted with three different models. The number of resource allocation in threshold-based model is seven and only four times of total requests are found appropriate. And fuzzy-based model is six and four times of total requests are found appropriate. In contrast, four times of total requests are appropriate among the five times of total resource requests in the pattern based model.

**Table 5**. Comparisons of decision success rate for resource allocation

| | *Appropriate* | *Inappropriate* | *Total* | *Rate* |
|---|---|---|---|---|
| Threshold | 4 | 3 | 7 | 57% |
| Fuzzy | 4 | 2 | 6 | 60% |
| Pattern | 4 | 1 | 5 | 80% |

**Table 5** is the overall results. The success rate of threshold-based model is 57%, fuzzy-based model is 60% and the success rate of pattern-based model is 80%. As a result, the proposed model is over 20% better than threshold-based model.

**Fig. 13** shows analytical results the three models. The number of resource releases in threshold-based model and fuzzy-based one was five each, and only two of total requests were appropriate. In contrary, the number of resource releases in pattern-based model was two and all requests were appropriate.


**Fig 13**. Release decisions from real time usage monitoring

**Table 6**. Comparisons of success rate for resource release from real time usage monitoring

| | *Appropriate* | *Inappropriate* | *Total* | *Rate* |
|---|---|---|---|---|
| Threshold | 2 | 3 | 5 | 40% |
| Fuzzy | 2 | 3 | 5 | 40% |
| Pattern | 2 | 0 | 2 | 100% |

**Table 6** shows that the success rate of the threshold-based model is 40%, fuzzy-based model is 40% and the success rate of pattern-based model is 100%. Again, the proposed model is illustrated that it is much more effective than other two models.

### 4.5 Analysis in multiple physical nodes

Finally, we performed a test on multiple servers. These servers are overloaded by a task generator. Job processing is conducted by a master server and two work nodes. If a master server needs additional resources, it assigns its jobs to two work nodes. In such case, job scheduler in the master server primarily allocates jobs to the work node 1, and if work node 1 is detected working - the resource utilization is over the work node threshold -, the job scheduler allocates them to work node 2.

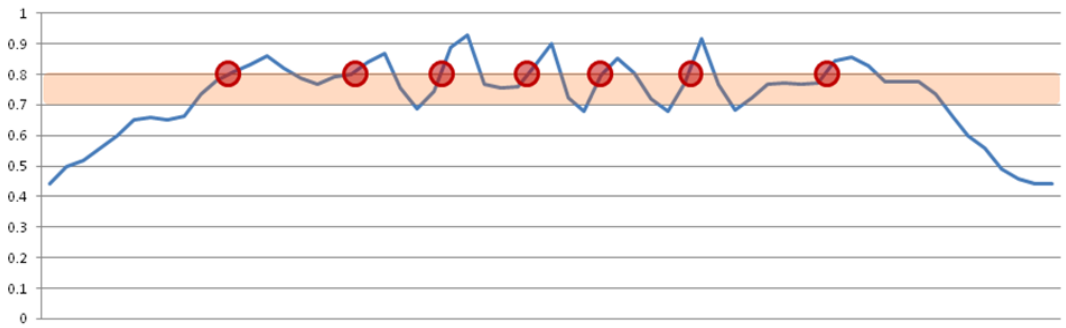


**Fig 14**. Resource usages and allocation decisions on threshold based approach

As you can see in **Fig. 14** and **Table 7**, in threshold based approach, a master server utilization rate is averaged 72%, and the two work nodes have 28% average utilization rates.

The total resource utilization of a master server and two slave work nodes has a 50% percent averaged. Note the utilization rate in the threshold-based approach varies frequently from 69% to 92% during experiment time span.

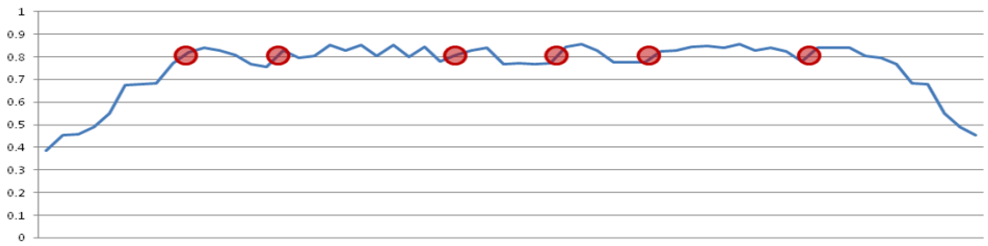


**Fig 15**. Resource usages and allocation decisions on fuzzy based approach

The result of the fuzzy-based approach is shown in **Fig. 15** and **Table 7**. As you can see the result, utilization flow is more stable than the threshold-based approach. And the request number of resource allocation and resource utilization is better than threshold-based one. A master server of all work nodes has 75% of the whole utilization and two work nodes have 27% of the whole utilization averagely. The total resource utilization of a master server and two work nodes has 51% averagely.

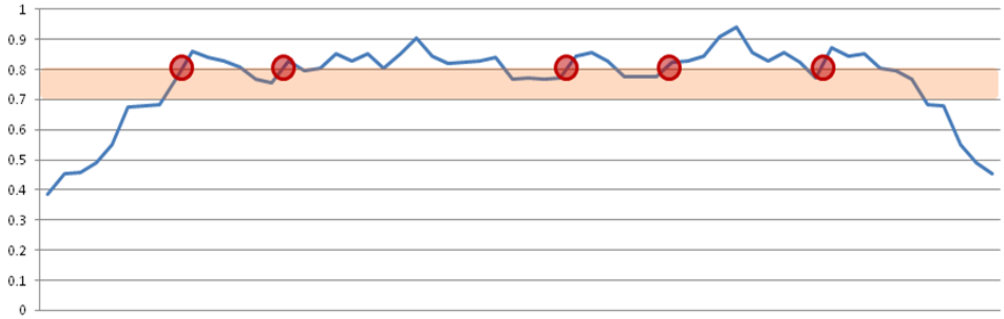


**Fig 16**. The analysis of resource utilization based on pattern based approach

**Table 7**. The analysis based on threshold-based approach

| | *Total # of requests* | *Master Server Avg. Utilization* | *Two Work nodes Avg. Utilization* | *Overall Avg. Utilization* |
|---|---|---|---|---|
| Threshold | 7 | 72% | 28% | 50% |
| Fuzzy | 6 | 75% | 27% | 51% |
| Pattern | 5 | 76% | 24% | 50% |

The result of the proposed approach is shown in **Fig. 16** and **Table 7**. In contrast, the system with the pattern-based approach had a small difference in the resource utilization flow with time. Furthermore, this approach is better than threshold-based approach and fuzzy-based approach as it operates with 50% percent on average due to the fact that the average resource utilization of two other types of work nodes operates with 24% on average.

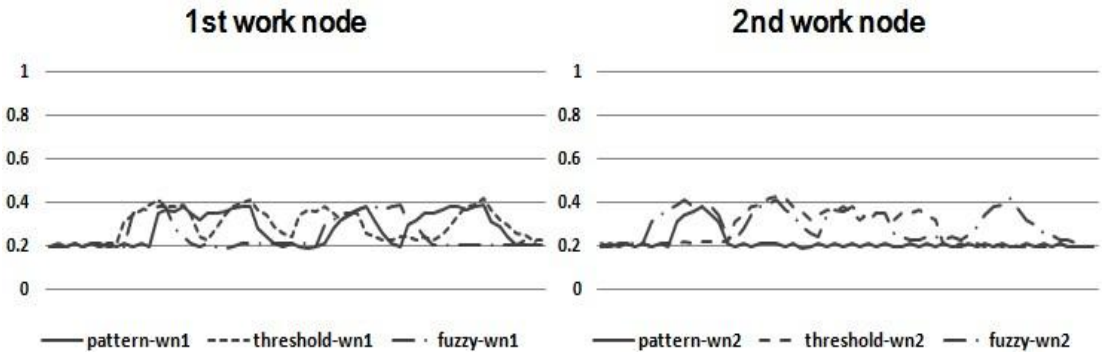


**Fig 17**. Resource utilization of work nodes in all approaches

**Table 8**. Standard deviation of each work node utilization rates

| | *Avg. Utilization* | *Std. in Utilization* |
|---|---|---|
| Threshold: work node1 | 29.8% | 0.07102 |
| Threshold: work node2 | 27.0% | 0.07905 |
| Fuzzy: work node1 | 25.0% | 0.06947 |
| Fuzzy: work node2 | 29.9% | 0.07050 |
| Pattern: work node1 | 28.0% | 0.07543 |
| Pattern: work node2 | 21.8% | 0.04303 |

**Fig. 17** and **Table 8** shows dynamics of resource utilization in two work nodes, and each approach exhibits its own dynamics in terms of resource utilization rate due to variation in job scheduler's allocation policy. As threshold based approach invokes more frequent resource allocation requests, the approach brings the most utilization rate from work nodes. Similarly, fuzzy based approach has more work node utilization rate than pattern-based approach, as it has more requests than pattern based approach.

In addition, in pattern based approach, work node 2 runs idle more than half of experimental time periods continuously. That makes more opportunities that work node 2 moves into sleep mode with minimum chances of sleep/wake-up process. Therefore, pattern-based approach can provide more efficient resource management in terms of power consumption and service delay.

## 5. Conclusion

In this paper, we proposed a pattern-based prediction model that enables high resource utilizations in a virtualized Cloud environment. Our prediction model enables to predict and prepare the needed resources before users begin to request them, in order to earn time to prepare VM setup and migration and hence to provide Cloud environments that reduce the chances of encountering delay. To this end, we developed and employed pattern-based models that show dynamics of resource utilization flow. Our model is designed simple and hence can save computational overhead to predict the amount of needed VMs in near future.

To validate our approach, we conducted various types of experiments to demonstrate benefits from the proposed model, in comparison with the conventional service provisioning environment: threshold based provisioning and fuzzy based provisioning. Our experiments include simulations with request history and run-time resource utilization monitoring with

data from load generator, and we found our approach is better than other two approaches in all of evaluation circumstances we conducted. The success rate of resource allocation/release and the efficiency of resource management in the proposed model show our approach is better than the conventional threshold-based model and fuzzy model.

The evaluation results demonstrate comparative advantages of our pattern based approach against other approaches including the traditional CPU load predictors and how our model is compatible to the needs of dynamic VM management. The presented model, as experimental results have shown, is highly effective at reducing the number of VM allocations and releases in diverse Cloud environment. As a result, our model can minimize the number of unnecessary VM allocations and releases, while maintaining the number of available VMs to accommodate all the needed amount of service requests with minimum delay.

Although the proposed model has several benefits, there are still areas for further improvements. We plan to expand our pattern types to reflect geographical dynamics in Cloud environment. For example, physical servers to be used can be distributed geographically and service quality may vary depending on the location of physical servers. Therefore, VM migration decision should reflect that.

## References


[1] Ian Foster, et al., "Cloud Computing and Grid Computing 360-Degree Compared," in *Proc. of the Grid Computing Environments Workshop*, pp. 1-10, Dec. 2008. Article (CrossRef Link)

[2] K. Keahey et al., "Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid," *Scientific Programming Journal*, pp. 265-275, Jan. 2006.

[3] Amazon Inc., "Elastic compute cloud," http://aws.amazon.com/ec2.

[4] Google Inc., "Google application engine," http://code.google.com/intl/itIT/appengine.

[5] Dell Co., "Dell cloud computing solutions," http://www.dell.com/cloudcomputing.

[6] J.S. Chase, D.E. Irwin, L.E. Grit, J.D. Moore, S. Sprenkle, "Dynamic virtual clusters in a grid site manager," in *Proc. of IEEE HPDC*, pp. 90-100, June 2003. Article (CrossRef Link)

[7] Distributed Systems Architecture Research Group, "Opennebula project," Universidad Complutense de Madrid, *Tech. Report*, 2009.

[8] WMWare Staff, "Virtualization overview," White Paper.

[9] Sun Inc., "VirtualBox," http://www.virtualbox.org/.

[10] Qumranet, "KVM," http://www.linux-kvm.org/page/.

[11] Purdue University, "Wispy project," http://www.rcac.purdue.edu/teragrid/resources/#wispy.

[12] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, "Xen and the art of virtualization," in *Proc. of SIGOPS Operating Systems Review*, pp. 164–177, Dec. 2003. Article (CrossRef Link)

[13] Masaryk University, "Kupa project," http://meta.cesnet.cz/cms/opencms/en/docs/clouds.

[14] R. Aversa, A. Mazzeo, N. Mazzocca, U. Villano, "Heterogeneous system performance prediction and analysis using PS," *IEEE Concurrency*, pp. 20-29, Mar. 1998. Article (CrossRef Link)

[15] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, K. Salem, "Adaptive control of virtualized resources in utility computing environments," *SIGOPS Operating Systems Review*, pp. 289-302, June 2007. Article (CrossRef Link)

[16] D.M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," *Technical Report UCB/EECS-2009-28*, EECS Department, University of California, Berkeley, pp. 1-23, Feb. 2009.

[17] W.E. Walsh, G. Tesauro, J.O. Kephart, R. Das, "Utility functions in autonomic systems," in *Proc. of International Conference on Autonomic Computing*, pp.70-77, May 2004. Article (CrossRef Link)

[18] D.M. Chess, G. Pacifici, A. Tantawi, "Experience with Collaborating Managers: Node Group

Manager and Provisioning Manager," in *Proc. of 2nd International Conference on Automatic Computing*, pp. 39-50, June, 2005. Article (CrossRef Link)
[19] D.M. Chess, A. Segal, I. Whalley, "Unity: Experiences with a Prototype Autonomic Computing System," in *Proc. of 1st International Conference on Autonomic Computing (ICAC'04)*, pp. 140-147, May 2004. Article (CrossRef Link)
[20] R. Das, J.O. Kephart, I.N. Whalley, P. Vytas, "Towards Commercialization of Utility-based Resource Allocation," in *Proc. of IEEE on Autonomic Computing*, pp. 287-290, June 2006. Article (CrossRef Link)
[21] M.N. Bennani, D.A. Menasce, "Resource allocation for autonomic data center using analytic performance models," in *Proc. of IEEE on Autonomic Computing*, pp. 229-240, June 2005. Article (CrossRef Link)
[22] S. Ranjan, J. Rolia, H. Fu, E. Knightly, "QoS-driven server migration for Internet data centers," in *Proc. of 10th IEEE International Workshop*, pp.3-12, May, 2002. Article (CrossRef Link)
[23] HP-UX Workload Manager, http://docs.hp.com/en/5990-8153/ch05s12.html.
[24] J. Rolia et al., "Configuring Workload Manager Control Parameters for Resource Pools," in *10th IEEE/IFIP Network Operations and Management Symposium*, pp.127-137, April, 2006. Article (CrossRef Link)
[25] T. Abdelzaher et al., "Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach," in *Proc. of IEEE Trans. on Parallel and Distributed Systems*, pp.80-96, Jan. 2002. Article (CrossRef Link)
[26] Z. Wang et al., "Utilization and SLO-Based Control for Dynamic Sizing Of Resource Partitions," in Proc. of *16th IFIP/IEEE Distributed Systems on Operations and Management*, pp.24-26, Jan. 2005. Article (CrossRef Link)
[27] X. Zhu, Z. Wang, S. Singhal, "Utility-Driven Workload Management using Nested Control Design," in *Proc. of American Control Conference*, pp.1-8, June, 2006. Article (CrossRef Link)
[28] A. Chandra, W. Gong, P. Shenoy, "Dynamic Resource Allocation for Shared Data Centers Using Online Measurements," in *Proc. of International Workshop on Quality of Service*, pp.300-301, June, 2003. Article (CrossRef Link)
[29] R. Doyle et al., "Model-Based Resource Provisioning in a Web Service Utility," in *Proc. of Internet Technologies and Systems on USENIX Symposium*, pp.5, March, 2003. Article (CrossRef Link)
[30] L. Sha, et al., "Queueing Model Based Network Server Performance Control," in *Real-Time Systems Symposium*, pp.81-90, Dec. 2002. Article (CrossRef Link)
[31] W. Xu et al., "Predictive Control for Dynamic Resource Allocation in Enterprise Data Centers", in *Proc. of IEEE/IFIP Network Operations and Management Symposium*, pp.115-126, April, 2006. Article (CrossRef Link)
[32] B. Urgaonkar et al., "An Analytical Model for Multitier Internet Services and its Applications", in *Proc. of ACM SIGMETRICS*, pp. 291-302, June 2005. Article (CrossRef Link)
[33] M. N. Bennani, D. A. Menascé, "Resource Allocation for Autonomic Data Centers using Analytic Performance Models," in *Proc. of International Conference Autonomic Computing*, pp. 229-240, Sep. 2005. Article (CrossRef Link)
[34] X. Liu et al., "Adaptive Entitlement Control Of Resource Containers On Shared Servers," in *Proc. of IFIP/IEEE Intl. Symposium on Integrated Network Management*, pp.163-176, June, 2005. Article (CrossRef Link)
[35] G. Tesauro, N.K. Jong, R. Das, M.N. Bennani, "A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation," in *Proc. of IEEE International Conference on Autonomic Computing*, pp. 65-73, June 2006. Article (CrossRef Link)
[36] C. Lumezanu, S. Bhola, M. Astley, "Utility Optimization for Event-Driven Distributed Infrastructures," in *Proc. of 26th IEEE International Conference on Distributed Computing Systems*, pp. 24, July 2006. Article (CrossRef Link)
[37] S. Ranjan, J. Rolia, H. Fu, E. Knightly, "QoS-driven server migration for Internet data centers," in *Proc. of 10th IEEE International Workshop on Quality of Service*, pp. 3-12, May 2002. Article (CrossRef Link)

[38] S.R. Mahabhashyam, "Dynamic Resource Allocation of Shared Data Centers Supporting Multiclass Requests," in *Proc. of the First International Conference on Autonomic Computing*, pp. 222-229, May 2004. Article (CrossRef Link)
[39] J. Almeida, V. Almeida, D. Ardagna, C. Francalanci, M. Trubian, "Resource Management in the Autonomic Service-Oriented Architecture," in *Proc. of IEEE International Conference on Autonomic Computing*, pp. 84-92, June 2006. Article (CrossRef Link)
[40] R.P. Doyle, J. Chase, O. Asad, W. Jin, A. Vahdat, "Model-Based Resource Provisioning in a Web Service Utility," in *Proc. of the Fourth USENIX Symposium on Internet Technologies and Systems (USITS)*, pp. 848-851, May 2003. Article (CrossRef Link)
[41] Y. Diao, J.L. Hellerstein, S. Parekh, "Using Fuzzy Control To Maximize Profits In Service Level Management," *IBM System Journal*, vol. 41, no. 3, pp. 403-420, Apr. 2002. Article (CrossRef Link)
[42] H. Kim, W. Kim, Y. Kim, "Experimental Study to Improve Resource Utilization and Performance of Cloud Systems Based on Grid Middleware," *Journal of Communication and Computer*, vol. 7, no. 12, pp. 32-43, Dec. 2010.
[43] J. Xu et al, "On the Use of Fuzzy Modeling in Virtualized Data Center Management," in *Proc. of 4th International Conference on Autonomic Computing (ICAC'07)*, pp. 25, June 2007. Article (CrossRef Link)
[44] R. Wolski, "Dynamically Forecasting Network Performance Using the Network Weather Service," *Journal of Cluster Computing*, vol. 1, pp. 119-132, July 2006. Article (CrossRef Link)
[45] R. Wolski, N. Spring, J. Hayes, "Predicting the CPU Availability of Time-shared Unix Systems," in *Proc. of 8th IEEE High Performance Distributed Computing Conference (HPDC 1999)*, pp. 102-115, Aug. 1999. Article (CrossRef Link)
[46] L. Yang, I. Foster, J.M. Schopf, "Homeostatic and Tendency-based CPU load Predictions," in *Proc. of International Symposium on Parallel and Distributed Processing (IPDPS'03)*, pp. 42-50, April, 2003. Article (CrossRef Link)
[47] O.B. Yaik, C.H. Yong, F. Haron, "Time Series Prediction Using Adaptive Association Rules," in *Proc. of the First International Conference on Distributed Frameworks for Multimedia Applications (DFMA'05)*, pp. 310-314, Feb. 2005. Article (CrossRef Link)

**Hyukho Kim** He received the B.A degree in Information and Communications engineering from Daejeon University in 2003. He received the M.S. and Ph.D. degrees from Dongguk University in 2005 and 2011, respectively. Since 2011, he has been an engineer of R&D Innovation Center at Samsung Advanced Institute of Technology. His current research interest areas are Grid computing, Cloud computing, GPU computing and Parallel and Distributed processing.

**Woongsup Kim** He received the B.A. degree in Computer Engineering from Seoul National University in 1998. He received his M.S. degree in Computer and Information Science from University of Pennsylvania, and Ph.D. degrees in Computer Science from Michigan State University in 2001 and 2006, respectively. Since 2007, he has been an Assistant Professor of Departement of information and communications engineering at Dongguk University. Research interests include software engineering, semantic web and service oriented architecture.

**Yangwoo Kim** He received the M.S. and Ph.D. degrees from Syracuse University in 1986 and 1992, respectively. Since 1996, he has been with Donogguk University as a professor. His current research interest areas are Grid computing, Cloud computing, P2P computing and Parallel and Distributed processing.