Resource Allocation with Exponential Model Prediction for Server Virtualization

Dulyawit Prangchumpol[1], Peraphon Sophatsathit[2]
ChidchanokLursinsap[3], PanjaiTantasanawong[4]
*1, 2, 3Department of Mathematics and Computer Science
Faculty of Science, Chulalongkorn University
Bangkok, Thailand
[4] Department of Computer Science
Faculty of Science, Silpakorn University, Bangkok
Thailand
[1]dulyawit@gmail.com, [2]peraphon.s@chula.ac.th, [3]lchidcha@chula.ac.th, [4]panjai@su.ac.th

Journal of Digital Information Management

ABSTRACT: *This paper proposes a practical and effective predicting model for resource allocation and usage for server. The approach rests on virtualization technique to simulate a heterogeneous virtual machine environment. The study is confined to management of number of CPUs and memory units being allocated on three servers to serve user's requests. The objective is fast response time to attain as high user satisfaction as possible. The algorithm employs exponential smoothing technique to analyze trend of data and the relationship among them. Assessment was compared with association rules and ARIMA. The findings reveal that the proposed technique yields more accurate prediction results than other predicting models, having the least MSE and acceptable system response time.*

Subject Categories and Descriptors
K.6.2 [Computing Installation Management]; Pricing and resource allocation

General Terms:
Servers, Resource Allocation

Keywords:
CPU, Memory Unit, Virtual Machines, VIrtualization

Received: 18 July 2015, Revised 20 August 2015, Accepted 22 August 2015

1. Introduction

Virtualization has widely been known in the field of energy saving technology. There are many related products that use this technique, and every enterprise improves its products to compete in the market. One use of such a technique is server virtualization which simulates not only the number of virtual CPUs but also the CPU organization. It imitates a virtual machine with multiple computers, each of which runs its own operating systems and thereby working as though it was a multiple platform system.

For resource usage, virtualization can be used to simultaneously manage heterogeneous workloads of different applications. The typical concerned resources in every application are the number of assigned CPUs and the size of allotted memory units to achieve the highest resource utilization and user satisfaction. An efficient resource management plan to achieve both aspects must be based on the pattern of past resource requests and prediction of future requests. Actually, the performance of any system must be evaluated in both aspects on system and user's sides. The maximum satisfaction of both sides should be as high as possible with minimum compromise factors. Details will be discussed in subsequent sections. Previously, there were many research works focusing on how to improve real-time monitoring workload and allocate resources to the system. None of them involved user satisfaction, which could be in terms of response time and execution cost, as one of their objectives. However, the real-time monitoring approach is too costly in practice because it must be continuously run. Furthermore, the system must waste some execution cycles to monitor

the events prior to the resource allocation. This obviously increases the undesirable workload of the system. To alleviate this situation, the amount of requested resources must be estimated and allocated in advance.

This paper presents a method to estimate the required number of CPUs and amount of memory usage in advance from the past data. The estimated resources must satisfy two previously mentioned aspects as much as possible. The following constraints are imposed in our study.

1. The behaviour of deploying a computing system of any user is unknown.

2. The requested resources are determined by the running processes of the user's submitted task. The user has no privilege to demand the requested resources to the computing system.

The above constraints imply that estimating the requested resources within a short period of time can achieve better performance of resource management than a longer period. Therefore, our method confines the estimation of resource usage to one hour in advance-based on past data. The estimated resources may satisfy the response time but could reduce resource utilization. Therefore these estimated resources must be further adjusted as a consequent process to compromise on resource utilization.

The rest of paper is organized as follows. Related works are presented in Section 2. Material and method are described in Section 3. Experiments and results are explained in Section 4. Finally, Section 5 concludes the work.

2. Related Work

Research on energy saving in computing systems have been paid much attention nowadays. Server virtualization is one of the techniques to reduce the number of physical components to conserve system energy and to maintain the acceptable response time of the system. Several works of energy cost reduction were proposed. Tick et al. [1] emphasized the cost reducing effect of ITS application on server virtualization through two studied cases. Khanna et al. [2] showed monitoring of key performance metrics and used the data to trigger migration of Virtual Machines within physical servers. Their algorithms attempted to minimize the cost of migration and maintained acceptable application performance levels. Hugo H. Kramer et al. [3] proposed an efficient approach to solve a relevant cluster optimization problem which, in practice, could be used as an embedded module to implement and integrated power and performance management solution in a real server cluster.

Server virtualization is also has impact on cloud computing. Casalicchio and Silvestri [4] focused their research on the mechanism for service level agreement (SLA) provisioning in cloud-based service providers. A self-manageable architecture for SLA-based service virtualization to ease interoperable service executions in a diverse, heterogeneous, distributed and virtualized world services were presented by Kertesz et al [5]. Prasad Calyam et al., [6] developed an analytical model called Utility-Directed Resource Allocation Model (URAM) to solve the combined utility-directed resource allocation problem within virtual desktop clouds.

In the meantime, various virtualization techniques were proposed. Steinder et al. [7] explored the usage of server virtualization technology in the autonomic management of data centers running a heterogeneous mix of workloads. Mlynski et al. [8] analyzed the influence of virtualization mechanisms of *pSeries* servers on dynamic resources and partition load manager utilities. Park et al. [9] identified some design considerations for constructing and managing clusters and proposed architectures to support clustering. Padala et al. [10] presented adaptive control of virtualized resources in utility computing environments. Xu et al., [11] introduced predictive control for dynamic resource allocation in enterprise data centers. Resource allocation for quality of service provision in buffered crossbar switches with traffic aggregation was presented by Q. Duan [12]. Q.Duan and J.Daigle [13] presented resource allocation for statistical quality of service provision in buffered crossbar switches. Jianfeng Zhao et al., [14] showed a model of virtual resource scheduling in cloud computing. Speitkamp and Bichler [15] proposed a capacity planning method for virtualized IT infrastructures that combined a specific data preprocessing and an optimization model. Moreover, Koushik Chakraborty et al. [16] used a hardware technique to detect when virtual CPU was waiting for CPU cycles, and to pre-empt that virtual CPU to run a different and more productive virtual CPU. Also, Zhikui Wang et al., [17] used AppRAISE to manage performance of multi-tier applications by dynamically resizing the virtual machines hosting the applications.

Considering the research related to improve workload in server virtualization, there is no study on predicting and allocating future requested resources to reduce the load on server and also to exploit the potential of hardware utilization, investment cost, and energy consumption as a whole.

3. Proposed Methods

3.1. Problem Formulation

The scenario of our virtualization is the following. A server is defined as a collection of homogeneous CPUs and memory units. A user submits a task consisting of a set of processes to the server. Some appropriate numbers of CPUs and memory units are allocated to execute these processes. However, any CPUs and memory units not being allocated to any processes will be idle. If the received task requires more computing resources, the server will turn on additional idle CPUs and memory units to serve

the request.

Our study concerns three related essential issues. The first issue concerns the estimation of number of requested CPUs and size of memory in the next hour. The second issue focuses on advance resource allocation so that the amount of allocated resources is always larger than the requested resources within a defined constant value. The last issue pertains to the relation between maximum resource utilization and user satisfaction in terms of response time. As previously mentioned, many researches consider only how to improve the performance of the system but totally omit user satisfaction aspect. In this work, the term *utilization* refers to the state of system with no idle components at any time. Generally speaking, system utilization and user satisfaction are controversial. To make a user most satisfy of the response time, more resources must be allocated. But more resources imply that some resources may not be fully used throughout the considered period and the energy consumption is obviously increased. The problem studied in this paper is formulated as follows.

Let $1 \leq j \leq 60$ and $1 \leq k \leq 6$ be the minute index in one hour and the time interval index of 10 seconds within one minute, respectively. Since there are 60 minutes in one hour, the value of j is between 1 and 60. For each minute, there are six equal 10-second time intervals. Thus, the value of k is between 1 to 6. At any hour, define the following variables for our computation.

1. $c_{j,k}$: Percentage of CPU usage at the k^{th} time interval of the j^{th} minute. The CPU time is measured in clock ticks or seconds. This percentage is measured by the ratio of number of deployed CPUs and total number of available CPUs in the server. Suppose a server has 10 CPUs and only 6 CPUs are executing the received tasks at the k^{th} time interval of the j^{th} minute. The other CPUs are idle. Hence the value of $c_{j,k}$ is 60%.

2. $m_{j,k}$: Percentage of memory usage at the j^{th} minute and the k^{th} second interval. The percentage is measured by the ratio between the actual amount of memory units used and the total available memory units.

3. $\hat{c}$: The predicted amount of requested CPU units in the next $(i+1)^{th}$ hour.

4. $\hat{m}$: The predicted amount of requested memory units in the next $(i+1)^{th}$ hour.

3.2. Theoretical Background

Some relevant theoretical concepts of server virtualization and models are briefly summarized in this section. The concepts of server virtualization and models to predict with interesting techniques, exponential smoothing technique, association rule discovery and auto-regressive integrated moving average are addressed in the following sub-sections.

3.2.1. Virtualization Backgrounds

In traditional server machine working one server is used for one application where it was found that the use of resource was not fully employed. This limitation of a traditional server created several consequent problems such as delayed response time, low utilization of resources, and inefficient power consumption. To alleviate this limitation, a set of single servers should be pooled to increase computing power and enhance performance efficiency.

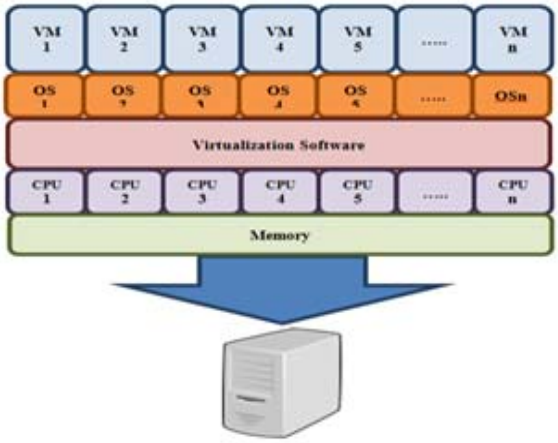

Figure 1.Typical server virtualization architecture consisting of a pool of heterogeneous servers

Server virtualization is a technique that can group different types of server machine to work as a single machine. The concept of virtualization has been widely used in many organizations to substitute for the traditional single server. It can manage the system more dynamically, allocate and de-allocate the resource on demand to improve utilization, and increase the investment value of hardware. Furthermore, it can reduce the number of physical machines, reduce cost of maintenance, and the power consumed to cool the server. This concept reduces the number of servers by combining heterogeneous workloads to work on virtual machines running as one physical machine as shown in Figure 1. In this paper, we will use the architecture of server virtualization as defined in the previous section.

3.2.2. Exponential Smoothing Methods

Exponential smoothing technique is an effective method commonly used to forecast temporal data.It can cope with data having little fluctuation or a small trend.In our study, the data have two patterns of change, i.e., relatively constant and time change with a little trend. Hence, the exponential smoothing method is suitable for these data change patterns to predict the future data values. There are two widely used smoothing approaches which are simple exponential smoothing and double exponential smoothing. Let f_t be the predicted value at time t and x_t be the considered data at time t. The methods are set up as follows.

Simple Exponential Smoothing

This approach focuses on different weighted data in different periods of time. There are three variables involved, namely,

a smoothing constant, the most recent predicted value, and the current data. A smoothing constant $\alpha \in [0, 1]$ is a weight assigned to the latest historical data.The formula of the simple exponential smoothing is defined as follows.

$$f_{t+1} = \alpha x_t + (1 - \alpha) f_t \quad (1)$$

Double Exponential Smoothing

Simple exponential smoothing method does not focus on the trend of data. The trend of data can cause a prediction error due to the fluctuation. To handle this phenomenon, the equation of simple exponential smoothing is rewritten by adding another term for handling a trend within a considered period of time. This term is defined as a function of two consecutively predicted values. Let Δ be a considered period of time. The predicted value computed by double exponential smoothing method is defined as follows.

$$f_{t+\Delta} = s_t + b_t \quad (2)$$

$f_{t+\Delta}$ is the prediction value at time $t+\Delta$, b_t is the trend smoothing value, and s_t is the overall smoothing value. The values of b_t and s_t are defined in the following equations.

$$s_t = \alpha x_t + (1 - \alpha)(s_{t-1} + b_{t-1}) \quad (3)$$

$$b_t = \gamma (S_t - S_{t-1}) + (1 - \gamma) b_{t-1} \quad (4)$$

where $0 \leq \alpha \leq 1$ is the smoothing constant between actual data and predicting value and $0 \leq \gamma \leq 1$ is a smoothing constant between the trend of actual data and the trend of prediction value. In case of seasonal trend pattern, the triple exponential smoothing method known as Winter's method can be applied.

3.2.3. Association Rule Discovery

Association rule is one of the well-known data mining techniques. The technique is used to analyze the relationship of the two data sets or larger data. In our experiments, this approach is compared with our approach in terms of day, time, and the resource usage levels. In association rule discovery, *confidence* (Conf) and *support*(Sup) are two values used to measure the strength of each discovered rules based on a considered set of transactions. A rule consists of left-hand side items and right-hand side items. The left-hand side items are the cause and the right-hand side items are the results. For any rule, *support* is defined as the ratio between the number of transactions having all items in both left-hand and right-hand sides and the total number of considered transactions. But *confidence* is defined as the ratio between the number of transactions having all items in both left-hand and right-hand sides and the number of transactions having only the left-hand side items.

In this study, the items of the left-hand side are the day and time but the items on the right-hand side are the amount of allocated resources in advance which is defined in terms of level unit. Some rules are eliminated by using the measures *sup* and *conf*.

3.2.4. Autoregressive Integrated Moving Average (ARIMA)

ARIMA model or Box-Jenkins technique is a popular univariate time series model prediction. This technique combines autoregressive (AR) model and moving average (MA) model based on historical data. It produces rather appropriate and efficient outcomes for a short period of time. [19].

3.2.5. Resource Utilization

Every organization prefers to achieve high computing system utilization with respect to the worthy investment. The term system refers to CPUs and memory units in this research. However, high system utilization may conflict with system performance in terms of response time and user satisfaction [20], [21] and [22]. With the factors mentioned above, this research focused on compromising the conflict between resource utilization and system response time to satisfy users' requests. Researches study on the relationship between utilization and response time were reported in [20], [21], [22], [23], [24] and [25]. These studies did not relate the problem of best resource allocation to the problems of resource utilization and response time.

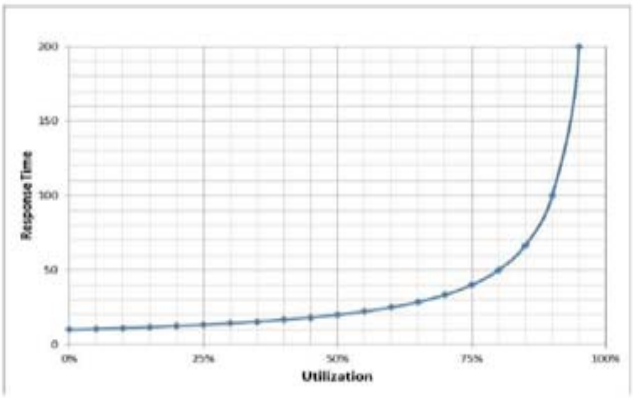

Figure 2. An example of the relationship between resource utilization and response time [20]

Figure 2 shows an example of the relationship between system utilization and response times [20]. It can be seen when the utilization is almost 100%, the response time becomes very slow. High utilization of the allocated resources implies that the resources are fully functioning, but it does not imply that the processing speed of any task is maximum. The compromised values of utilization and response time should be at the *knee point* of the graph as shown in Fig. 2. This point is where the gradient of the curve suddenly changes. In this figure, the *knee point* is at 75% of utilization.

Figure 3 illustrates different knee values [22]. The upper line denotes the current relationship between utilization and response time. When allocating more resources to the system, this line is shifted downwards as indicated by the lower line. The knee value is moved from 0.81% to

0.66% utilization. This reduces utilization but increases response time. The location of *knee value* depends upon the resource configuration and the policy of each organization.

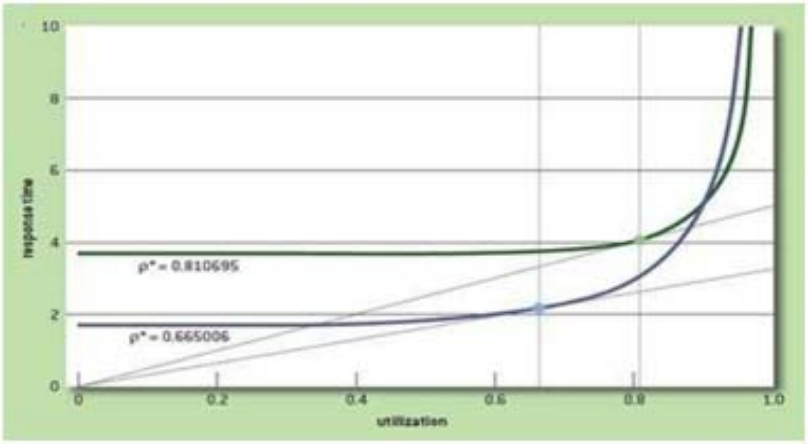

Figure 3. Response time curves showing knee values [22]

3.3. Proposed Algorithm

The algorithm consists of two main steps. The first step is to predict the amount of requested resources by applying double exponential smoothing method. In fact, it is rather difficult to make the prediction precise. In case of imprecision, some processed tasks may be interrupted due to insufficient resources and the response time must be prolonged. This degrades user satisfaction although utilization may be maximum. The second step is to adjust the predicted amount of resources to compromise the utilization and system response time. The resources refer to CPU and memory units. The prediction is performed one hour in advance. The following variables will be referred in the algorithm. For any hour *i*, let

1. $c_{j,k}$ be the percentage of CPU usage at the j^{th}minute time and k^{th} second time interval.
2. $m_{j,k}$ be the percentage of memory usage at the j^{th}minute time and k^{th} second time interval.
3. $s_i^{(cpu)}$ be the overall smoothing value at the i^{th} hour. This has the same meaning as s_t in Section 3.2.2. The superscript *(cpu)* denotes that this overall smoothing value is for CPU.
4. $b_i^{(cpu)}$ be the trend smoothing value at the i^{th} hour. This has the same meaning as b_t in Section 3.2.2. The superscript *(cpu)* denotes that this trend smoothing value is for CPU.
5. $s_i^{(mem)}$ be the overall smoothing value at the i^{th} hour. This has the same meaning as s_t in Section 3.2.2. The superscript *(mem)* denotes that this overall smoothing value is for memory.
6. $b_i^{(mem)}$ be the trend smoothing value at the i^{th} hour. This has the same meaning as b_t in Section 3.2.2. The superscript *(mem)* denotes that this trend smoothing value is for memory.
7. $\bar{m}_i$ be the mean allocated memory units within thei^{th} hour.
8. $\bar{c}_i$ be themean allocated CPU units within thei^{th} hour.
9. $\hat{m}$ be thepredicted amount of requested memory units in the next $(i+1)^{th}$ hour.
10. $\hat{c}$ be thepredicted amount of requested CPU units in the next $(i+1)^{th}$ hour.
11. u be thecompromising factor of utilization versus user satisfactions, $1 \leq u \leq 100$.

Double exponential smoothing is used in our prediction process because the prediction error of this method is less than that of the simple exponential smoothing. The detail of how to predict CPU and memory requests is given in Algorithm 1.

4. Experiments and Results

To signify the merit of our proposed concept, three servers of different purposes, i.e., database server, application server, and web server, were involved in the experiment. Each type of server had its behavioural characteristics. The database server was used for internally and externally collecting, creating, and sending documents. The application server acted as anti-virus scanner and was also used for installing, updating patches as well as signature, deploying patches to the users, and monitoring the system. The web server was used only for web services. Details of data collection and the set-up of relevant parameters were given in the following sections.

4.1. Experimental Set-up

Data concerning the use of CPU and memory were collected from the above three servers at Suan Sunandha Rajabhat University in every 10 seconds during a period of 60 days. The values of comprising factor in step 10 of Algorithm 1 were set to 65. The data were separated into two parts. The first part was the set of data collected within the first 45 days used for creating model and the second part was the remaining data collected within the other 15 days for testing.

The values of α and γ in simple and double exponential smoothing methods were differently set up as shown in Table 1 to achieve as high accuracy as possible. Table 1(a) summarizes only the *a* values for simple exponential smoothing method, while Table 1(b) summarizes both α

and γ values for double exponential smoothing method. Note that the values of α in both methods were set to the same values.

4.2. Experimental Results

Since our approach concerns two related issues, i.e., predicting the requested resources and resource allocation to satisfy the response time, the experimental results relevant to each issue will be separately addressed as follows.

Algorithm 1 Predicting and allocating resources

Input : The resources usage in every k period $c_{j,k}$ and $m_{j,k}$.

Output : Resources allocated for next hour $\tilde{m}_i$, $\tilde{c}_i$

Step1: Let T = 360.

Step2:for 1 ≤i≤ 24 **do**

Step3: Let $\bar{c}_i = \frac{1}{T}(\sum_{j=1}^{60}\sum_{k=1}^{6} c_{j,k})$

Step4:end for

Step5:for 1 ≤i≤ 24 **do**

Step6: Let $\bar{m}_i = \frac{1}{T}(\sum_{j=1}^{60}\sum_{k=1}^{6} m_{j,k})$

Step7:end for

Step8: Compute $\tilde{c}_{i+1}$ by double exponential smoothing method as follows.

$$\tilde{c}_{i+1} = \propto \bar{c}_i + (1-\propto)(s_{i-1}^{(cpu)} + b_{i-1}^{(cpu)}) + \gamma(s_i^{(cpu)} - s_{i-1}^{(cpu)}) + (1-\gamma)b_{i-1}^{(cpu)}$$

Step9: Compute $\tilde{m}_{i+1}$ by double exponential smoothing method as follows.

$$\tilde{m}_{i+1} = \propto \tilde{m}_i + (1-\propto)(s_{i-1}^{(mem)} + b_{i-1}^{(mem)}) + \gamma(s_i^{(mem)} - s_{i-1}^{(mem)}) + (1-\gamma)b_{i-1}^{(mem)}$$

Step10: Adjust the prediction $\tilde{c}_{i+1}$ and $\tilde{m}_{i+1}$ by

$$\tilde{c}_{i+1} = \frac{100}{u}\tilde{c}_{i+1}$$

$$\tilde{m}_{i+1} = \frac{100}{u}\tilde{m}_{i+1}$$

Server	Resource	α
Database	CPU	0.6
	Mem	1.0
Application	CPU	0.5
	Mem	1.0
Web	CPU	0.2
	Mem	1.0

(a) Simple Exponential

Server	Resource	α	ã
Database	CPU	0.6	0.2
	Mem	1.0	0.0
Application	CPU	0.5	0.2
	Mem	1.0	0.0
Web	CPU	0.2	0.0
	Mem	1.0	0.0

(b) Double Exponential

Table 1.The set up values of α and γ for both exponential smoothing methods. (a) The α value for simple exponential smoothing method. (b) The a and γ values for double exponential smoothing method

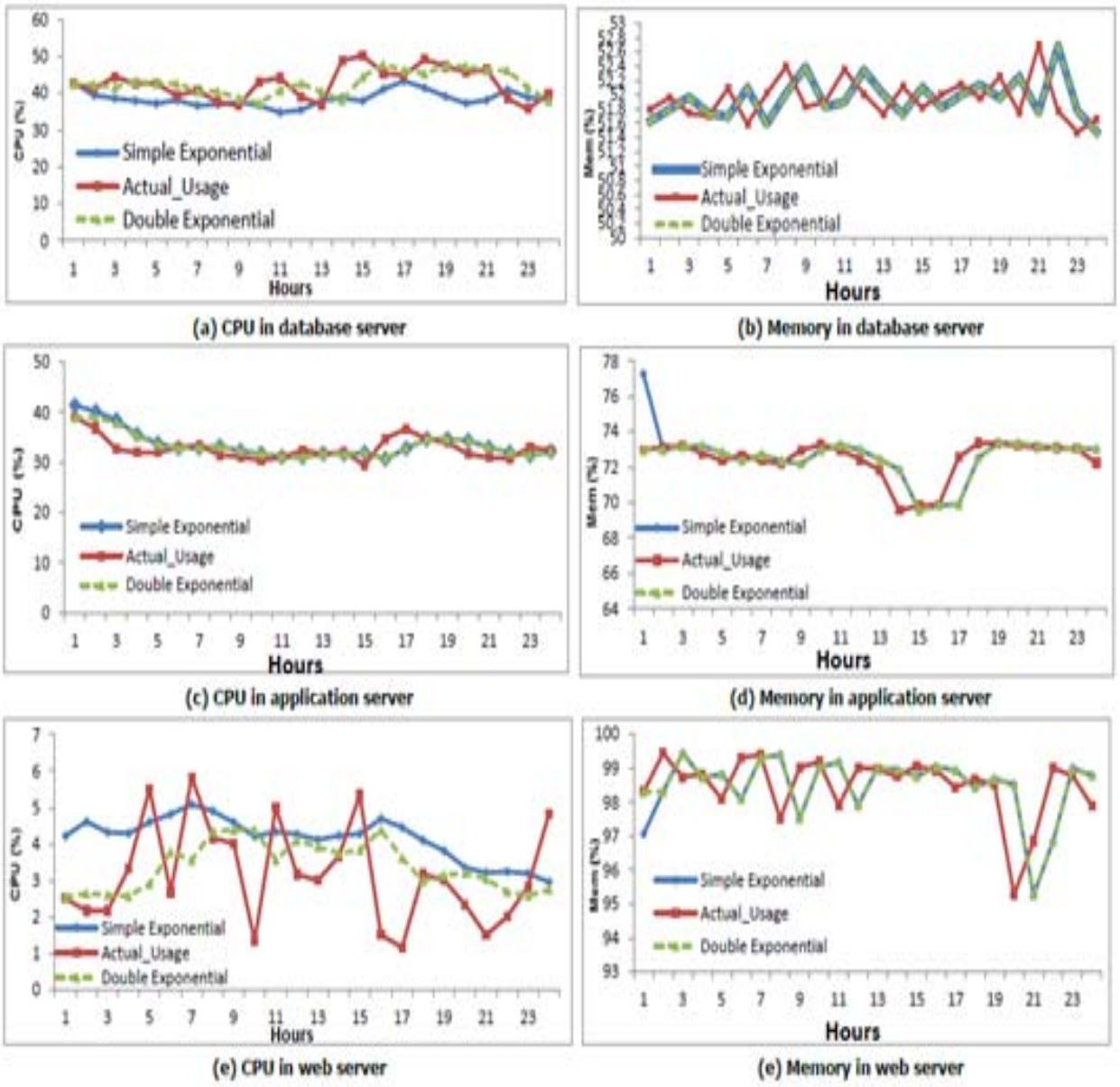

Figure 4. Resource prediction by exponential smoothing method for three servers

4.2.1. Results of Exponential Smoothing Methods

The percentage of CPU and memory usages in each hour predicted by simple exponential and double exponential smoothing methods are compared and shown in the figure 4. The predicted CPU usages of database, application, and web servers are in Figure 4(a), (c), and (e), respectively. Predicted memory usages of those three different servers are in Figure 4(b), (d), and (f), respectively. It is noticeable that the predicted CPU and memory usage of all servers by double exponential smoothing method, which was adopted in our work, are more accurate than those from simple exponential smoothing method.

4.2.2. Effect of Compromising Factor on Resource Allocation

Generally, the predicted number of resources may be higher or lower than the actual number of requested resources. If the predicted number of resources is higher than the actual requested one, then the best response time can be achieved. But if the predicted number of resources is lower than the requested one, then some additional resources must be augmented to the predicted one to maintain the best response time. After the prediction by exponential smoothing method, the predicted resources are augmented by allocating some additional resources, i.e. more number of CPUs and memory units. However, the augmented resources must not be too many to affect optimizing the number of idle resources and power consumption of those idle resources. Determining the optimal number of augmented resources is not easy since the actual behaviour of CPU and memory requests is unknown in advance. The only known information is the predicted request. In our experiment, an additional compromising factor *u* was empirically set to 65%-75% of resource utilization. Fig. 5 shows the CPU allocation, actual CPU usage, and the predicted values. Fig. 6 shows the memory allocation, memory usage, and the predicted

values. The value of compromising factor *u* can directly affect the performance if utilization is close to the knee value. On the other hand, if utilization is set too low, it would be wasteful since there may be too many idle resources.

However, higher utilization will deteriorate the response time because the allocated resources may not be enough to run the given tasks, which implies that all allocated resources must be busy all the time. For the example in Table 2, at 2 o'clock, the predicted value is 37.17. If the value of *u* is set to 65% (*u*65), then allocated value becomes (37.17*100)/65 = 57.19. But the actual resource usage is equal to 38. Then the real utilization is (38*100)/57.19 = 66.45. This value is over the utilization boundary but not over 75%.

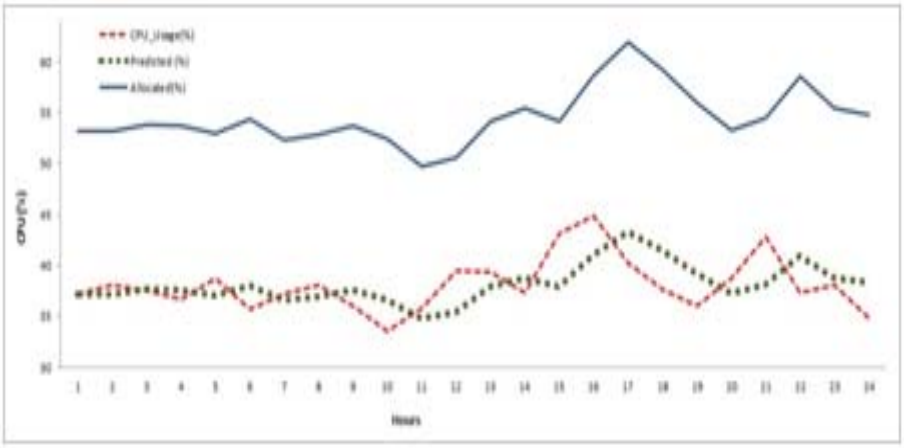

Figure 5. CPU allocation with compromising factor *u* = 65% for database server

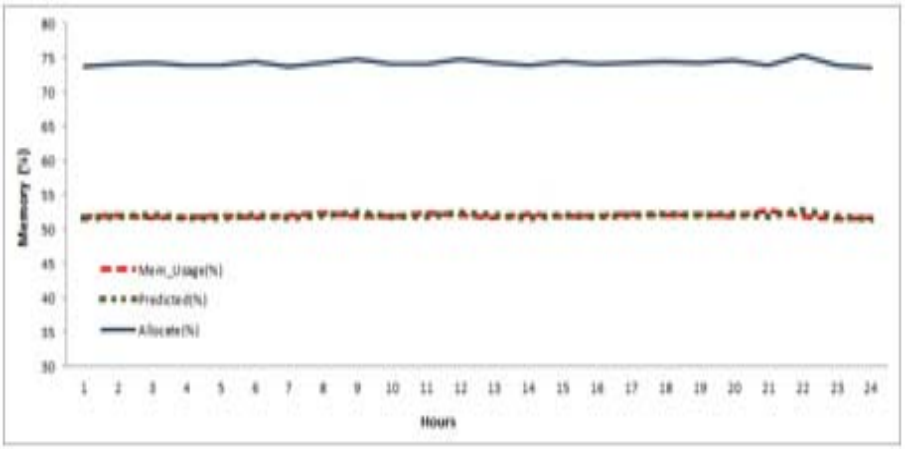

Figure 6. Memory allocation with compromising factor *u* = 65% for database server

In contrast to previous 65% utilization, at 2 o'clock, if the value of *u* is set to 75% (*u*75) of the predicted value of 37.17, then the allocated resource becomes (37.17*100)/75 = 49.56. But the actual resource usage is 38. Hence the real utilization is (38*100)/49.56 = 76.67, which exceeds utilization boundary of 75%.

Table 3 demonstrates the utilization for different values of *u*, denoted by *u*65, *u*66, *u*67, *u*68, *u*69, *u*70, and *u*75. The result indicated that when the value of *u* is increased, the utilization of system may also be increased in some cases. This may reduce the response time of the system.

4.3. Comparisons with Other Predicting Models

Resource prediction is an important part of this research since the prediction values will be considered in the allocation policy. Two predicting methods, i.e. association rule and ARIMA, were compared with ours in terms of prediction accuracy.

4.3.1. Comparing with Association Rule

The resource usage predicted by association rule in our experiment was classified into the following five levels: *1* for low level; *2* for medium low level; *3* for medium level; *4* for medium high level; and *5* for high level. Each level is determined by a set of rules written in the forms of $(D, T) \rightarrow L$ where D is the day, T_j is the time when the usage occurs, and L is the level. For example, the rulemeans the resource usage on Monday at 06:00PM is in level 3.

(Monday, 06:00 PM) $\rightarrow$ 3

Table 4(a) shows an example of rules for predicting memory usage levels on Monday and Tuesday at 06:00 PM in database server. Confidence and support values are used for rule selections. It may be possible that there are more than one established rule with equal confidence. In this situation, the rule with maximum support will be selected. But if there exists a set of rules whose confidence

and support values are equal, then the first rule of this set will be selected [18]. Table 4 (b) shows an example association rule prediction of memory usage by the database server during 24x7 hours of service.

Time	Predicted (%)	CPU_ Usage	Allocated With *U65*	Utilization (65%)	Allocated With *U75*	Utilization (75%)
1	37.17	37.17	57.18	65	49.56	75
2	37.17	38	57.19	66.45	49.56	76.67
3	37.67	37.5	57.96	64.7	50.23	74.66
4	37.57	36.67	57.8	63.44	50.1	73.2
5	37.03	38.67	56.97	67.87	49.38	78.31
6	38.02	35.67	58.49	60.99	50.69	70.37
7	36.61	37.17	56.32	65.99	48.81	76.15
8	36.95	38	56.84	66.85	49.26	77.14
9	37.58	36	57.82	62.26	50.11	71.84
10	36.64	33.5	56.36	59.43	48.85	68.58
11	34.76	35.83	53.47	67.01	46.34	77.31
12	35.41	39.5	54.47	72.51	47.21	83.67
13	37.87	39.33	58.26	67.51	50.49	77.9
14	38.75	37.33	59.62	62.62	51.67	72.25
15	37.9	43.17	58.31	74.03	50.54	85.42
16	41.06	44.83	63.18	70.96	54.75	81.88
17	43.33	40.17	66.66	60.26	57.77	69.53
18	41.43	37.67	63.75	59.09	55.25	68.19
19	39.18	36	60.27	59.73	52.24	68.92
20	37.27	38.67	57.35	67.43	49.7	77.81
21	38.11	42.83	58.64	73.04	50.82	84.28
22	40.95	37.33	63	59.26	54.6	68.37
23	38.78	38	59.67	63.69	51.71	73.48
24	38.32	34.83	58.95	59.08	51.09	68.17

Table 2.Resource utilization with allocation at *u*65 and *u*75

From association rule with confidence and support values in Table 4(b), the level of memory usage predicted for database server in 7 days and in each hour period are summarized in Table 5(a). To compare with our predicting method, each level number must be converted to a numeric value corresponding to the resource usage as follows. At level *i*, there may be different numbers of resource usage. Let $R^{(i)} = \{r_1^{(i)},\ldots, r_n^{(i)}\}$ be the set of different resource usage $r_j^{(i)}$ at level *i*. The numeric value of resource usage at level *i* is equal to $\sum_{j=1}^{n} \frac{r_j^{(i)}}{|R^{(i)}|}$

Table 5(b) shows the numeric value of corresponding CPU usage after converting from its level. The average percentage of resource usage for each hour is converted from the level usage. Fig. 7 shows the comparison of prediction results obtained from double exponential smoothing, association rule, and actual resource usage for database, application, and web in three different servers. The results of our method are more accurate than others for three servers.

4.3.2. Comparing with ARIMA

ARIMA is an efficient statistical forecasting method. The results from our method were compared with ARIMA for CPU and memory usage for database, application, and web in three servers. Fig. 8 shows the comparison results obtained from double exponential smoothing, ARIMA, and actual resource usage. Generally, the proposed method produced higher accuracy than ARIMA except the case of CPU usage of web server. In this case, the errors from

Time	*U65*	*U66*	*U67*	*U68*	*U69*	*U70*	*U75*
1	65.00	66.00	67.00	68.00	69.00	70.00	75.00
2	66.45	67.47	68.49	69.51	70.54	71.56	76.67
3	64.70	65.70	66.69	67.69	68.68	69.68	74.66
4	63.44	64.41	65.39	66.37	67.34	68.32	73.20
5	67.87	68.92	69.96	71.01	72.05	73.09	78.31
6	60.99	61.93	62.86	63.80	64.74	65.68	70.37
7	65.99	67.01	68.03	69.04	70.06	71.07	76.15
8	66.85	67.88	68.91	69.94	70.96	71.99	77.14
9	62.26	63.22	64.18	65.14	66.09	67.05	71.84
10	59.43	60.35	61.26	62.18	63.09	64.01	68.58
11	67.01	68.04	69.07	70.10	71.13	72.16	77.31
12	72.51	73.63	74.75	75.86	76.98	78.09	83.67
13	67.51	68.55	69.59	70.63	71.67	72.70	77.90
14	62.62	63.58	64.54	65.51	66.47	67.43	72.25
15	74.03	75.17	76.31	77.45	78.59	79.73	85.42
16	70.96	72.05	73.14	74.24	75.33	76.42	81.88
17	60.26	61.19	62.11	63.04	63.97	64.90	69.53
18	59.09	60.00	60.91	61.82	62.73	63.64	68.19
19	59.73	60.65	61.57	62.49	63.40	64.32	68.92
20	67.43	68.47	69.51	70.54	71.58	72.62	77.81
21	73.04	74.17	75.29	76.41	77.54	78.66	84.28
22	59.26	60.17	61.08	61.99	62.90	63.81	68.37
23	63.69	64.67	65.65	66.63	67.61	68.59	73.48
24	59.08	59.99	60.90	61.81	62.72	63.63	68.17

Table 3. Resource utilization with different utilization boundary

Rule	**Conf (%)**	**Sup (%)**
Monday , 06:00 PM → 3	37.5	0.219
Monday , 06:00 PM → 4	50	0.292
Monday , 06:00 PM → 2	12.5	0.07
Monday , 06:00 PM → 4	50	0.292
Tuesday , 06:00 PM → 1	12.5	0.073
Tuesday, 06:00 PM → 3	25	0.146
Tuesday, 06:00 PM → 4	50	0.292
Tuesday, 06:00 PM → 4	50	0.292

(a) Examples of rules for predicting

both methods were almost the same and rather high. Our best result was the memory usage prediction of database and web servers.

4.3.3. Summary of Comparison

We use mean square error (MSE to evaluate the performance of each method with respect to the actual

No.	rule	Conf (%)	Sup (%)
1	Monday , 0:00 → 3	62.5	0.365
…	……	…	…
24	Monday , 23:00 → 3	62.5	0.365
25	Tuesday , 0:00 → 3	50	0.292
…	……	…	…
168	Sunday , 23:00 → 3	55.56	0.365

(b) Example of predicting by association

Table 4. Association rule with confidence and support values

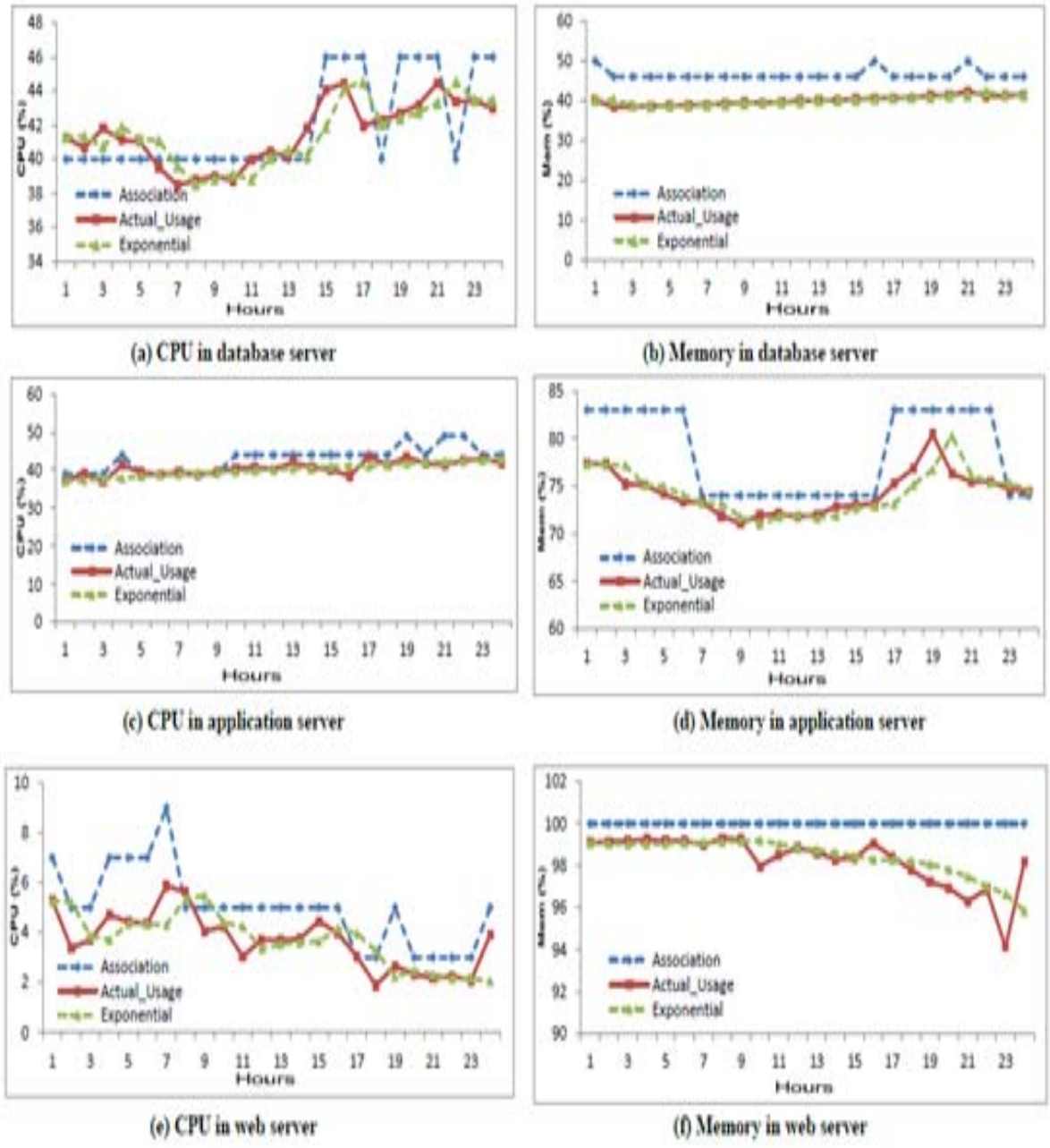

Figure 7. Comparison of resources prediction by association rule and our double exponential smoothing method for three servers

Time	sat	sun	mon	tue	wed	thu	fri
0	4	2	3	3	2	4	2
1	4	2	3	3	3	2	3
2	3	2	3	3	3	4	4
3	3	2	3	3	3	4	3
4	4	2	4	3	2	5	2
5	4	2	4	2	3	1	2
6	2	2	3	3	3	4	2
7	5	2	4	3	3	4	2
8	4	2	3	3	2	4	2
9	3	2	3	3	3	2	2
10	4	2	4	3	3	3	3
11	4	2	2	3	3	2	3
12	2	2	3	3	4	3	3
13	2	2	3	3	4	2	4
14	2	3	3	3	4	2	4
15	4	3	3	3	4	3	4
16	2	3	3	3	4	4	3
17	2	2	3	4	4	3	3
18	2	3	4	4	4	2	4
19	3	3	2	3	4	2	4
20	3	3	3	3	4	3	3
21	3	2	3	3	4	3	4
22	3	3	3	2	4	3	4
23	2	3	3	2	4	3	2

(a) Level for resource usage

Time	sat	sun	mon	tue	wed	thu	fri
0	52	40	46	46	40	52	40
1	52	40	46	46	46	40	46
2	46	40	46	46	46	52	52
3	46	40	46	46	46	52	46
4	52	40	52	46	40	100	40
5	52	40	52	40	46	34	40
6	40	40	46	46	46	52	40
7	100	40	52	46	46	52	40
8	52	40	46	46	40	52	40
9	46	40	46	46	46	40	40
10	52	40	52	46	46	46	46
11	52	40	40	46	46	40	46
12	40	40	46	46	52	46	46
13	40	40	46	46	52	40	52
14	40	46	46	46	52	40	52
15	52	46	46	46	52	46	52
16	40	46	46	46	52	52	46
17	40	40	46	52	52	46	46
18	40	46	52	52	52	40	52
19	46	46	40	46	52	40	52
20	46	46	46	46	52	46	46
21	46	40	46	46	52	46	52
22	46	46	46	40	52	46	52
23	40	46	46	40	52	46	40

(b) Resource usage in percentage

Table 5. CPU usage in 24 hours by association rule

values. MSE is defined by the following equation. Suppose there are n data points.

$$MSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 / n \qquad (5)$$

Y_i is the actual value at time i. $\hat{y}_i$ is the predicted value at time i. Table 4.8 shows MSE for all method predictions in three servers. Association rules provide the highest prediction error. This is probably because resource usage and the period of time in each day are less correlated. Furthermore, ARIMA model prediction is more efficient than association rules. Simple and Double exponential techniques give slightly different values. Double exponential smoothing technique gives the least MSE. Due to data from three servers having different functions, resource usage of CPU and memory units also varied. Double exponential can predict data whose behaviors are different from stationary or non-stationary because this model adjusts the smoothing constant between actual data and predicting value (α) and the smoothing constant between the trend of actual data and the trend of prediction value (γ).

5. Conclusions and Future Work

The problem of predicting resource usage, especially CPU and memory usage, with the constraints on resource utilization in order to achieve the best response time of a virtualization system was studied. Unlike other predictions, our prediction involves consideration of resource utilization with the prediction to compromise between the system response time and resource utilization. High utilization may prolong system response time. On the other hand,

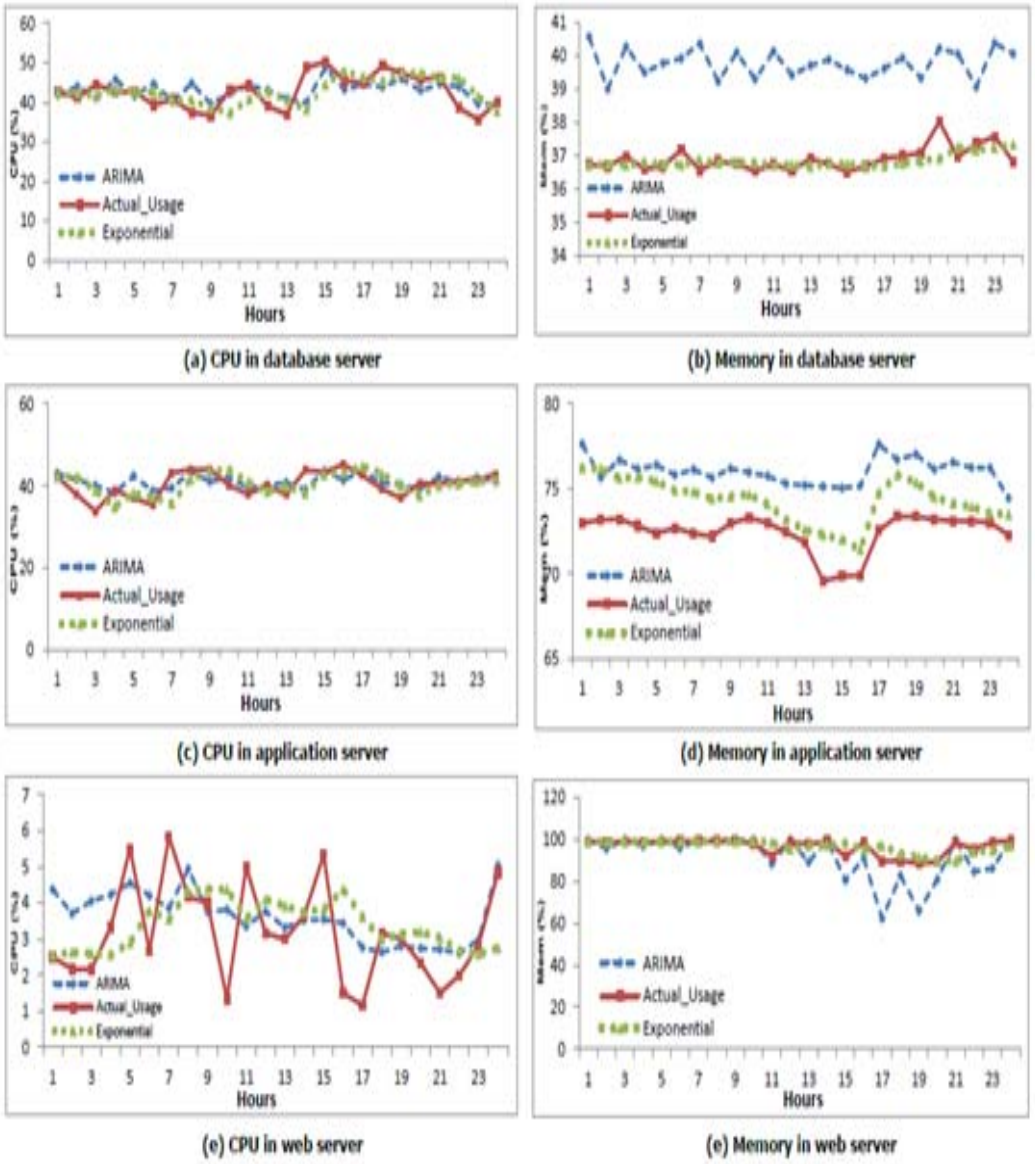

Figure 8.Comparison of resources prediction by ARIMA and our double expoential smoothing method for three servers

low utilization may shorten system response time and leave some idle resources. The method of double exponential smoothing technique with utilization compromising factor were introduced in this paper. The proposed method was tested with the actual data and compared with association rule and ARIMA. Our prediction produced the higher accurate results than those of the other methods. The proposed compromising factor can be effectively used to allocate resources to meet the acceptable system response time.

Typically, the behaviour of request for resources of a server is rather inconsistent. It may depend upon operation time of the system, the number of users, or the demands of users' programs. The parameters used in the predicting algorithm as well as the utilization compromising factor must be periodically adjusted to conform with up-to-date situation. Identifying the resource requesting behaviour of the system for different periods of time must be further studied.

References

[1] Tick, J., Tiszai, T. (2007). Server Virtualization in Intelligent Traffic Control System. in Logistics and Industrial Informatics, LINDI 2007. International Symposium on.

[2] Khanna, G., et al. (2006). Application Performance Management in Virtualized Server Environments. in Network Operations and Management Symposium, NOMS 2006. 10th IEEE/IFIP.

[3] Kramer, H. H., et al. (2012). A column generation approach for power-aware optimization of virtualized heterogeneous server clusters. *Comput. Ind. Eng*., 63 (3) 652-662.

[4] Casalicchio, E., Silvestri, L. (2013). Mechanisms for SLA provisioning in cloud-based service providers. *Computer Networks*, 57 (3) 795-810.

[5] Kertesz, A., Kecskemeti, G., Brandic, I. (2014). An interoperable and self-adaptive approach for SLA-based

service virtualization in heterogeneous Cloud environments. *Future Generation Computer Systems*, 32 (0). 54-68.

[6] Calyam, P., et al. (2011). Utility-directed resource allocation in virtual desktop clouds. *Computer Networks*, 55 (18) 4112-4130.

[7] Steinder, M., et al. (2007). Server virtualization in autonomic management of heterogeneous workloads. *In*: Integrated Network Management, 2007. IM '07. 10th IFIP/ IEEE International Symposium on.

[8] Mlynski, M. (2008). The Analysis of Influence of IBM pSeries Servers' Virtualization Mechanism on Dynamic Resources Allocation in AIX 5L. *In*: Parallel and Distributed Computing, 2008. ISPDC '08. International Symposium on.

[9] Jong-Geun, P., et al. (2008). Cluster Management in a Virtualized Server Environment. *In*: Advanced Communication Technology, ICACT 2008. 10th International Conference on.

[10] Padala, P., et al. (2007). Adaptive control of virtualized resources in utility computing environments, *In:* Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, ACM: Lisbon, Portugal. p. 289-302.

[11] Wei, X., et al. (2006). Predictive Control for Dynamic Resource Allocation in Enterprise Data Centers. *In: Network Operations and Management Symposium*, 2006. NOMS 2006. 10th IEEE/IFIP.

[12] Duan, Q. (2007). Resource allocation for quality of service provision in buffered crossbar switches with traffic aggregation. *Int. J. Comput. Appl*., 2007. 29 (3) 283-290.

[13] Duan, Q., Daigle, J.N. (2008). Resource allocation for statistical quality of service provision in buffered crossbar switches. *International Journal of Communication Systems*, . 21 (6) 609-629.

[14] Zhao, Jianfeng., Liu, W.Z., Miu., Li, Guangming (2012). A model of Virtual Resource Scheduling in Cloud Computing and Its Solution using EDAs. JDCTA: *International Journal of Digital Content Technology and its Applications*, 6. p. 102-113.

[15] Speitkamp, B., Bichler, M. (2010). A Mathematical Programming Approach for Server Consolidation Problems in Virtualized Data Centers. *Services Computing*, *IEEE Transactions on*, 3 (4) 266-278.

[16] Chakraborty, K., Wells, P. M., Sohi, G. S. (2012). Supporting Overcommitted Virtual Machines through Hardware Spin Detection. *Parallel and Distributed Systems, IEEE Transactions on*, 23 (2) 353-366.

[17] Zhikui, W., et al. (2009). AppRAISE: application-level performance management in virtualized server environments. *Network and Service Management*, *IEEE Transactions on*, 6 (4) 240-254.

[18] Prangchumpol, D., Sanguansintukul, S., Tantasanawong, P. (2012). Improving heterogeneous workload performance in server virtualization based on user behaviors. *Journal of Convergence Information Technology (JCIT)*, 7.

[19] Raghuveera, T. (2011) P.V.S.K.a.K.S.E. , Adaptive Linear Prediction Augmented Autoregressive Integrated Moving Average Based Prediction for VBR Video Traffic. *Journal of Computer Science*,. 7.

[20] Nan, G., et al. Parametric Utilization Bounds for Fixed-Priority Multiprocessor Scheduling. *In:* Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International. 2012.

[21] González-Horta, Francisco A., R.A.E.-C., Ramírez-Cortés, Juan M., Martínez-Carballido, Jorge Buenfil-Alpuche, Eldamira (2011). Mathematical Model for the Optimal Utilization Percentile in M/M/1 System: A Contribution about Knees in Performance Curves, in The Third International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE 2011). 2011.

[22] Millsap, Cary (2010). M.R.C., Thinking Clearly about Performance. Performance, 8.

[23] Deji, C., Mok. A. K., Tei-Wei, K. (2003). Utilization bound revisited. Computers, *IEEE Transactions on*, 52 (3) 351-361.

[24] Nan, G., et al. (2012). Parametric Utilization Bounds for Fixed-Priority Multiprocessor Scheduling. *In:* Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International.

[25] Craciunas, S. S., Kirsch, C. M., Sokolova, A. (2010). Response Time versus Utilization in Scheduler Overhead Accounting. *in Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2010 16th IEEE.

Author biographies

D. Prangchumpol was born in Thailand in 1980. He received B.Sc in statistic degree from Thamasat University and M.S. in computer science from Slipakorn University. He studies doctorate degree at Chulalongkorn University in field of computer science and information technology. He works for SuanSunandhaRajabhat University, major information technology. His research work is in the areas of Virtual system, and Network System