# Two Level Scheduling on Private Cloud System: Cloud Resources Scheduling and Real Time Scheduling on Virtualized Servers

Hsu Mon Kyi, Thnn Thu Naing
*University of Computer Studies, Yangon*
*hsumonkyi.ucsy@gmail.com, ucsy21@most.gov.mm*

## Abstract

*Cloud computing is deployed a large set of virtualized computing resources in different infrastructures and various development platforms. One of the key challenges in cloud computing system is virtual resources and virtual machines (VMs) are rapidly provision in order to meet the cloud user's requirement. To address this challenge, this system contributes two level scheduling systems: (i) virtual resource allocation and scheduling on private cloud infrastructure and (ii) real time scheduling that is invoked for multimedia applications running on virtual machines. First is resource level scheduling and second is application level scheduling. This system analyzes first level scheduling steps by applying an analytical performance model using Stochastic Markov chain. Moreover, a real time scheduling algorithm is presented for application level to analyze real time multimedia applications running on virtualized servers. According to performance evaluation, this system describes the detail analysis of virtual resources and allocation steps based on the criteria such as user request completion probability, mean response time. Then, this system also shows the analysis results for real time applications running on virtualized servers. This scheduling algorithm contributes to reduce the rate and ratio of missing deadline. As a testbed infrastructure, this system evaluates and analyzes an academic-oriented private cloud system which is implemented using Eucalyptus open source system.*

***Keywords**: Cloud Computing; Virtual Machine; Scheduling; Virtualization; Stochastic Markov Chain; Eucalyptus*

## 1. Introduction

A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on Service Level Agreements (SLA) established through negotiation between the service providers and consumers [9]. There are four deployment models of cloud computing environment such as Public, Private, Community and Hybrid cloud. This research is only emphasis on the private cloud model and data and processes are managed within the organization that a limited number of people behind a firewall. Eucalyptus open source provide cloud system is configured to provide IaaS services in the system.

Resource scheduling is a key process for cloud system. Cloud infrastructure commonly take virtual machine (VM) as scheduling unit, be allocated on physical resources. Some of the classical cloud-based applications include Social Networking, Web Hosting, Content Delivery, and Real-Time Instrumented data processing. It is very difficult to quantify the performance of scheduling and allocation policy on cloud infrastructures for different applications under varying workload and system size. The reason why resource allocation and scheduling brings new research issues in cloud computing system. To address this challenge, this system contributes two level scheduling systems. The detail explanation of these scheduling are described in next section.

The paper is organized as follows: Section 2 discusses related work to this topic and design of two levels scheduling architecture is present in section 3. Then, the system model for resource allocation is presented in section 4. This paper defines steps of the model approach in section 5. Then, numerical performance evaluation results are presented in section 6 and followed by real time scheduling algorithm on virtualized servers is presented in section 7. Section 8 shows evaluation results for real time application. Finally, Section 9 concludes the paper.

## 2. Related Work

Since Eucalyptus [2] and Usher [7] are the open source systems for cloud infrastructure and development, they provide VM creation and resources allocation across a Physical Machine on cluster servers. However, they could not support the efficient VM scheduling policies to consolidate or redistribute VMs.

Rodrigo N. Calheiros et al. [10] presented analytical performance (queuing network system model) to improve the efficiency of the system. This proposed provisioning technique detects changes in workload intensity (arrival pattern, resource demands) that occurs over time and allocates multiple virtualized IT resources accordingly to achieve application QoS targets. H.M.Kyi et al. [4] proposed stochastic markov model approach for virtual machines scheduling on private cloud environment. This approach analyse performance of system based on state probability of the system model.

Luqun Li [6] discussed an optimistic differentiated service job scheduling system for cloud computing service users and providers. This system uses non-preemptive priority M/G/1 queuing model for these job services. Hongbin Liang et al. [3] proposed Semi-Markov Decision Process model for resource allocation on mobile cloud environment. This system aims to allocate the cloud resource to maximize the system resources.

O.Khalid et al. [8] proposed a dynamic and adaptive real-time virtual machine scheduling technique for HPC workloads on the Grid. The objectives of the system are to increase overall jobs throughput in the system and meet their deadline.

W. Tsai et al. [11] proposed a framework for real-time service-oriented cloud computing. This system aims to schedule tasks for the multi-tenancy SaaS applications. C.Vecchiola et al.[1] present deadline- driven provisioning mechanism. This mechanism shows that Aneka cloud application platform is able to efficiently allocate resources from different sources in order to reduce application execution times.

To the best of our knowledge, the paper presents Stochastic Markov Model approach for resource allocation and the heterogeneous VM request servicing of IaaS properties. And then, real time scheduling algorithm is presented to analyze the performance evaluation for real time applications running on virtualized servers.

## 3. Design of Two Level Scheduling Architecture

This section presents two level scheduling systems on the cloud computing architecture. First is resource level scheduling and second is real time application level scheduling. First level analyzes mean response time and throughout of system based on the effects of variations in workload such as job arrival rates, job service time and system capacity (number of NCs in each pool) on IaaS cloud service. Second level focus on deadline meet rate of real time application running on virtualized server.

For analysis first level resource allocation and scheduling steps, we use an analytic modeling approach using stochastic Markov models for analyzing performance evaluation. First, we construct separate sub-models for resource allocation and servicing steps of a cloud service and then the overall solution is obtained by iteration over individual sub-model solutions. The detailed steps of the model are described in the next section.

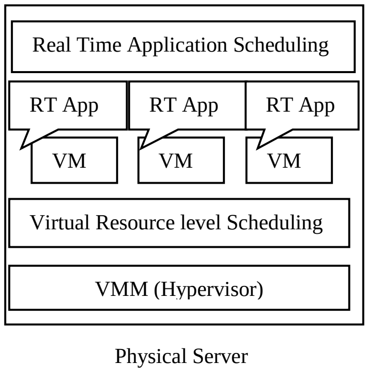
Real Time Application Scheduling
RT App
RT App
RT App
VM
VM
VM
Virtual Resource level Scheduling
VMM (Hypervisor)
Physical Server

**Figure 1. Layer of cloud Scheduling**

For second level, we also present a real-time scheduling algorithm for real-time multimedia application in order to reduce the deadline miss rate of task. The algorithm is based on EDF (Earliest Deadline First) scheduling policy in order to arrange real time tasks to meet deadline. The analysis demonstrates that the system is reduced deadline miss of system in real time video conferencing cloud service on virtualized server.

## 4. System Model for Resource Allocation and Scheduling

This system model is constructed based on Eucalyptus private cloud Infrastructure architecture. In such system, several virtual machine (VM) types are offered according to the users' requirements. These VM types with specific CPU, RAM and storage capacity are provisioned after creation an instance. These user request VM are deployed on node controller (NCs) each of which may be shared by multiple VMs. The Eucalyptus Infrastructure offers two types of resource pools. These pool are running (turn on) and pending (turn on, but not ready) pool. User requests several VM types are submitted to a resource allocation decision module that processes request on a first-come first-serve (FCFS) basis as follows. The request at the head of the queue is provisioned on a running server if there is capacity to run a VM on one of the running servers. If no running NC is available, a NC from pending pool is used for provisioning the requested VM. If none of these servers are available, the request is rejected and placed this request on appropriate queue. For the above described scenario, we investigate the effects of varying job arrival rates, job service rates, system capacity on the QoS metrics.

## 5. Proposed Model Approach

Shown in Figure 2 is the life cycle of VM instance request in cloud system. When cloud user request VM services from web interface, the resource allocation decision phase (RADP) checks whether its resource availabilities can meet the requirement of this VM request. If the request capacity is not sufficient in the system, the request place queue. If the request is accepted, it goes to a specific machine for VM creation. After creating the VM, this VM already deploy in the cloud. Then, the VM runs in the cloud and releases the VM when it finishes.

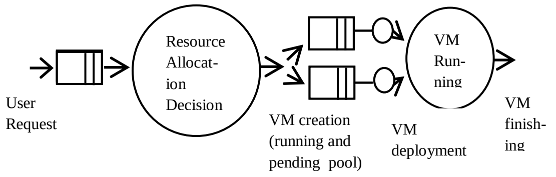
User Request
Resource Allocat-ion Decision
VM creation (running and pending pool)
VM deployment
VM Run-ning
VM finish-ing

**Figure 2. Life cycle of VM instance request**

From this instance life cycle, the cloud service are decomposed the three major steps. They are (i) resource allocation decision (ii) VM usage (creation and deployment) and (iii) VM execution. These steps are translated into analytical model. These models are described below.

## 5.1. Resource Allocation Decision Model

To calculate the resource allocation decision process, we design a continuous time Markov chain (CTMC) shown on Figure 3.

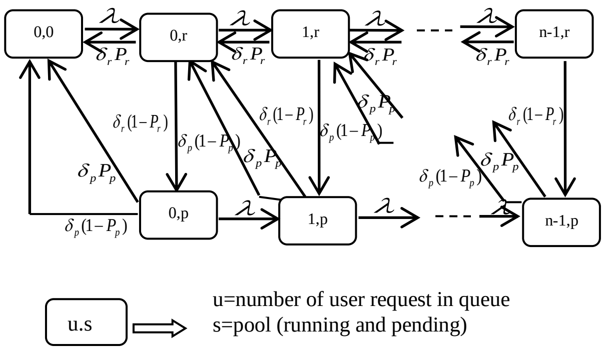

**Figure 3. Resource allocation decision model**

The system users arrive at the system with the Poisson rate λ. In this model, arrival user is u (u $\in \{1,\ldots,n\}$). States in the model in Figure 2 are labelled as (u,s), where u denotes the number of users currently waiting in the queue and s denotes the type of pool that the user' requested VM is undergoing allocation decision. In this model, state (0,0) indicate a user has not arrived at the system. From state (0,0) model transits to state (0,r) with rate λ, due to arrival of a user. State (0,r) describe the RADP is deciding if at least one running NC can accept the user requested VM for allocation. Similarity, state (0,p) indicate the RADP is deciding if any pending NC can accept the request for allocation. This system assumes that $1/\delta$ is the mean searching delay to fine a NC for allocation in RADP. In state (0,r), three possible outgoing events can occur: (a) job is accepted for allocation on one of the running NCs, and the model goes to state (0,0) with rate , $\delta_r P_r$ . (b) user request VM cannot be accepted for allocation on any running NC, and the model goes to state (0,p) with rate $\delta_r(1-P_r)$,(c) arrival of new request and the model goes to state (1,r) with rate λ. If no running NC is available, a transition occurs from state (0,r) to state (0,p).In state (0,p), three possible outgoing event are same transaction with the state (0,r). Next State (1,r) represents the condition that one request is waiting in the decision queue and request job is undergoing allocation decision. In this model, input and out parameters discussed in the following.

### 5.1.1 Model Input and Output

Input parameters in this model, cloud user in according to the Poisson distribution rate λ is assumed to be given, the delay parameters $\delta_r, \delta_p$ can be measure from Greedy search and $P_r, P_p$ are compute from VM usage model. Outputs of this model are

(i) First, user request rejection probability due to buffer full and is denoted by $P_{block}$

$$P_{block} = \pi_{(n-1,r)} + \pi_{(n-1,p)} \tag{1}$$

(ii) Probability that a user request will be rejected due to insufficient NC capacity ($P_{insufficient}$).

$$P_{insufficient} = \sum_{u=0}^{n-1} \frac{\delta_p(1-P_p)\pi_{(u,p)}}{\lambda} \tag{2}$$

(iii) User request service unavailable probability ($Service_{unavailabe}$) is sum of $P_{block}$ and $P_{insufficient}$

(iv)Measure of service availability that user request will be available

$$Service_{available} = 1 - Service_{unavailable} \tag{3}$$

(v) Average waiting time in resource allocation decision phase $E[W_{RADP}]$ = $E[W_{q\_dec}]$(queuing delay for resource allocation decision)+$E[W_{dec}]$ (decision delay)

$$E[W_{RADP}] = \frac{\sum_{i=0}^{n-1} i(\pi_{(i,r)} + \pi_{(i,p)}) + \lambda(\frac{1}{\delta_r} + \frac{(1-P_r)}{\delta_p})}{\lambda(1 - Service_{unavailable})} \tag{4}$$

## 5.2. Virtual Machines Usage Model

VM usage models capture the instantiation creation and deployment of a VM on a NC. We assume that all event times (e.g., VM request inter-arrival time, service time, VM provisioning time etc.) considered in this model are exponentially distributed. Service time for each VM request type: μ obtained from run time model. We design separate VM usage models for running, pending pool of NCs. States of the model in Figure 3 are indexed by (i,j), i denotes number of VMs currently being provisioned and j denotes the number of VM on a NC which have already been deployed. In this model, from state (0,0), after a job arrival, model goes to state (1,0), with rate $\lambda_r$. In state (1,0), a VM instance is created. Mean time to creation a VM on a running PM, is $1/\beta_r$ and the model moves from (1,0) to (0,1) with rate $\beta_r$. Upon service completion, VM instance is removed and the model moves from (0,1) to (0,0) with rate $\mu$; this rate is computed as an output from the VM execution model. In this usage model, input and output parameters are discussed in the following.

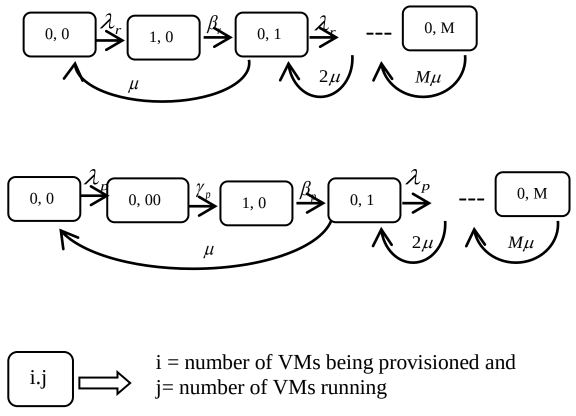

**Figure 4. Virtual Machine usage model for running and pending pool**

### 5.2.1 Model Input and Output

This model assumes total $H_r$ NCs in the running pool, the arrival rate $\lambda_r$ to each running NC is given by:

$$\lambda_r = \frac{\lambda(1-P_{block})}{H_r} \tag{5}$$

the mean time to creation a VM on the running NC is $1/\beta_r$ and service rate μ are obtained from the VM run time model. Outputs of this model are

(i) the steady state probability ($\pi_r$) that all VM on the running server are busy and probability of running pool that a user request can be accepted

$$P_r = 1-(\pi_r)^{H_r} \tag{6}$$

For a pending NC is similar to the running NC model, with few differences:

(i) the arrival rate $\lambda_p$ to each pending NC is given by:

$$\lambda_p = \frac{\lambda(1-P_{block})(1-P_r)}{H_p} \tag{7}$$

(ii) the pending NC requires some additional start-up time to make it ready to use. Time to make a pending NC ready for use, is assumed to be exponentially distributed with mean $1/\gamma_p$. (iii) Mean time to provision a VM on a pending NC is $1/\beta_p$ for the first VM to be deployed on this PM; mean time to provision VMs for subsequent jobs is the same as that for a running NC, i.e., $1/\beta_r$. After solving the pending NC, we can compute the steady state probability ($\pi_p$) that a pending NC can not accept a request for VM provisioning and overall pool model is a set of $H_p$. The probability of pending pool can accept the request is given by:

$$P_p = 1-(\pi_p)^{H_p} \tag{8}$$

From VM usage models, we can also compute average waiting time in VM usage $E([W_{usage}]) =(E[W_{vm\_q}])$(queuing delay)+$(E[W_{prov}])$ (provision delay). According to their Resource Allocation Decision Model and VM usage Model, we can compute average response time for a VM request. This is given by:

$$E[T_{resp}]=E([W_{usage}])+E[W_{RADP}] \qquad (9)$$

### 5.3. Virtual Machine Execution Model

Once a VM request is successfully allocated, it utilizes the resources until its execution is completed.

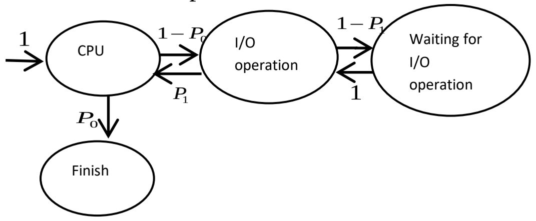

**Figure 5. Virtual Machines execution model for each virtual machine request**

VM execution model is used to determine the mean time for a VM service completion. We use a Discrete Time Markov Chain (DTMC) to capture the details of VM execution. From the initial state labeled CPU, a VM can finish its execution with a probability $P_0$ or go for some I/O operations with probability (1- $P_0$).A transition can occur from local I/O to waiting I/O with a probability (1- $P_1$) or from local I/O to CPU with probability p1 . Assuming the mean service times on the CPU, local I/O and waiting I/O to be $1/\mu_c$, $1/u_l$ and $1/\mu_w$ respectively, we compute the mean VM service time:

$$\frac{1}{\mu}=\frac{1}{P_0\mu_c}+\frac{(1-P_0)}{P_0P_1\mu_l}+\frac{(1-P_0)(1-P_1)}{P_0P_1\mu_w} \qquad (10)$$

## 6. Numerical Performance Evaluation Results

We evaluated cloud user VM request services are two solutions- (1) User request completion probability and (2) mean response time for resource allocation and servicing. In this system model, we show the effect of changing job arrival rates, job service time and system capacity (number of servers in each pool). We assumed exponential distribution for inter-arrival times and service times.

In our example scenario, buffer size in front of RADP to be 20, and buffer size within each NC to be zero. System capacity for each NC has CPU (2x2 GHz), Memory (4 GB) and disk space (320 GB) are considered in this system model. In this stochastic model, resource allocation decision model (in our example, 41 states) and VM usage models (for each model respective numbers of states depend on number of VMs) are solved in this system.

For the performance analysis, academic-oriented Private Cloud Testbed measurements are used in these model parameters. Our testbed analysis, system allows available number of virtual machines according to the user request as shown in the Table 1.

**Table 1. Maximum number of VMs on each NC**

| Name | Max number of VMs | CPUs | Memory (MB) | Disk (GB) |
|---|---|---|---|---|
| c1.medium | 16 | 1 | 256 | 5 |
| m1.large | 8 | 2 | 512 | 10 |
| c1.xlarge | 2 | 4 | 2048 | 40 |

All models were solving using SHARPE [5] software package. Values of key parameters are shown in Table 2.

**Table 2. Values of key parameters**

| Symbol | Meaning | Value |
|---|---|---|
| $1/\delta_r$, $1/\delta_p$ | Mean search delays for resource allocation decision phase: from a particular pool (running and pending) | 4 seconds |
| $1/\beta_r$ | Mean time to VM for creation and deployment a VM on a running server | 8 minutes |

| | | |
|---|---|---|
| $1/\beta_p$ | Mean time to VM for creation and deployment a VM on a pending server | 12 minutes |
| $1/\gamma_p$ | Mean time to prepare on pending state for ready to use | 20 seconds |
| $1/\mu$ | Mean VM service time | 15minutes |
| $\lambda$ | Cloud user request VM arrival time | 300-500 request/hr |

In our experiment, Figure 6(a) shows, at a fixed mean service time (15 minute) and fixed number of NCs in each pool (e.g., 8 NCs in each pool) and when increasing arrival rate, decreases user request completion probability. If we will increase the capacity (NCs in each pool), user request completion probability will rises.

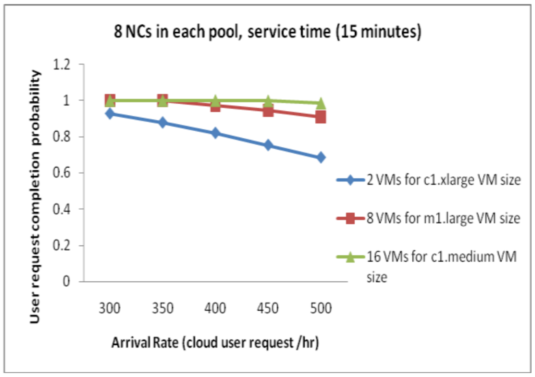

**Figure 6(a). User request completion probability for different arrival rate and fixed mean service time (15 minutes)**

Figure 6(b) shows that with increasing arrival rate, mean response delay increases for a fixed number of NCs in each pool. In Figure 6(b), observe arrival rate at 300, 350, 400, 450 and 500 user request VM an hour, for analysis different VM request type.

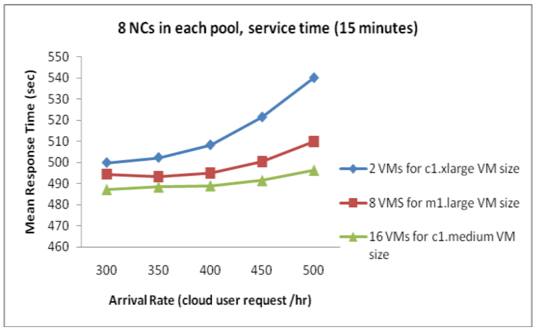

**Figure 6(b). Mean response time for different arrival rate and fixed mean service time (15 minutes)**

## 7. Real-Time Scheduling Algorithm on Virtualized Servers

The second part of the paper is real-time application scheduling algorithm. The real-time algorithm can be applied to meets the deadline of the cloud services when cloud users use the web portal to request multimedia application running on virtualized server on cloud platform.

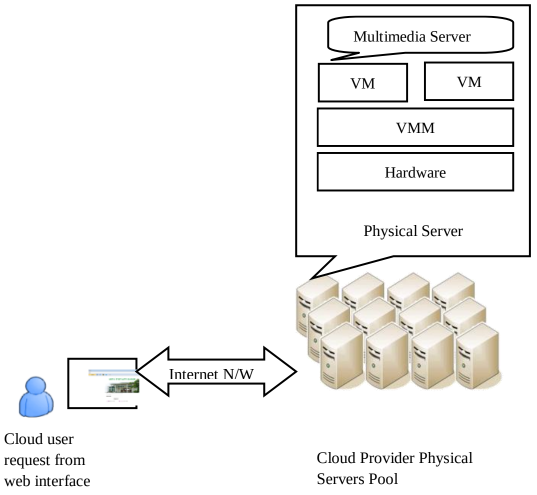
Multimedia Server
VM
VM
VMM
Hardware
Physical Server
Internet N/W
Cloud user request from web interface
Cloud Provider Physical Servers Pool

**Figure 7. Cloud platform architecture**

First the system can be considered on a number of tasks and other parameters that include the system. Then applied the algorithm based on the following parameters; execution time, arrival time, start transmission time, and deadline.

In a video conferencing system, video frames and audio samples are arranged at the virtual resource. In this service, the system decides which task is the nearest deadline to schedule. It makes a schedule based on earliest deadline first (EDF) scheduling policy in which task should be sent first. The following algorithm shows how to process the proposed scheme. In this system, tasks are schedule based on deadline nearest. Then, we find the difference between finishing time and deadline. Then, the value of the difference is less than the zero, this task is schedule meet the deadline.

```
Alogrithm: Real Time Scheduling Algorithm
Begin
$t_i=\{a_i,e_i,d_i\}$ where $1\leq i\leq m$, $m>0$
Schedule these tasks whose deadlines are the nearest and send them by orderly.
$f_i=s_i+e_i$
If ($d_i- f_i <0$)then
Deadline-miss=1
Else
Deadline-miss=0
End
```

**Table 3. Notation of real time algorithm**

| Symbol | Meaning |
|---|---|
| $T=\sum_{i=1}^{m} t_i$ | A set of tasks in a real time application |
| $a_i$ | Arrival time of task i |
| $e_i$ | Execution time of task i |
| $d_i$ | Deadline of task i |
| $s_i$ | Start transmission time of task i |
| $f_i$ | Finish time of task i |

## 8. Evaluation Results

In this section, the performance evaluations of the real time multimedia application are presented. The proposed algorithm tests with video data tasks on virtualized running server. The result shows that the number of tasks dropped due to deadline miss is significantly reduce in the system.

In figure 8(a), real time statistics on video transmission. The result shows that the deadline miss loss occurs without applying algorithm.

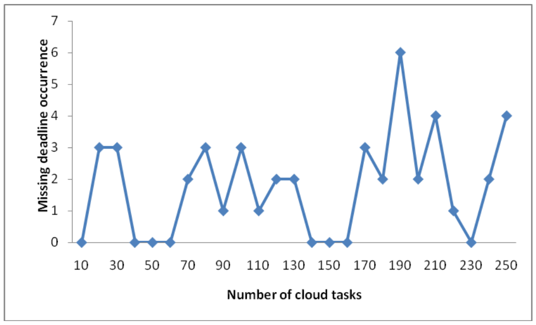

**Figure 8(a). Deadline miss occurrence without applying algorithm**

In figure 8(b), the same experiment is tested by applying the algorithm, the result shows that the deadline miss occurrence is significantly reduced within the same time series. The result shows that the deadline miss occurrences reduce to 28%.

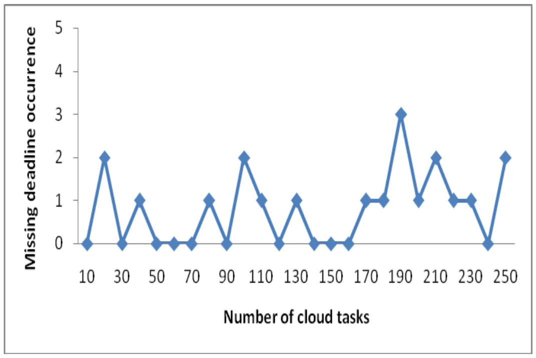

**Figure 8(b). Deadline miss occurrence with applying algorithm**

## 9. Conclusion

It has been widely accepted that virtual machines can be employed as computing resources for high performance computing. Therefore, present Stochastic Markov model for evaluate the performance of resource scheduling and allocation on private cloud system. This method is suitable for analyzing the VM request service of large sized IaaS clouds, with reduced complexity of analysis. In the second part, a real time scheduling algorithm for real time multimedia applications is proposed. This algorithm shows that significantly reduce the deadline miss rate of real time application.

## References

[1] C.Vecchiola, R.N.Calheiros, D.Karunamoorthy "Deadline-driven provisioning of resources for scientific applications in hybrid cloud with Aneka", ACM,*In Proc.ISLPED*,2011.

[2] D.Nurmi, R.Wolski, C.Grzegorczyk, G. Obertelli, S.Soman, L.Youseff, and D.Zagorodnov. "The Eucalyptus open-source cloud-computing system", *In Proceedings of Cloud Computing and Its Applications*,2008.

[3] H.Liang, D.Huang, L..Cai, X. (Sherman) Shen and D. Peng, "Resource Allocation for Security Services in Mobile Cloud Computing", *IEEE,pp. 191-195* ,May 2011.

[4] H.M.Kyi and T.T.Naing. "Stochastic Markov Model Approach for Virtual Machines Scheduling on Private Cloud", *International Journal on Cloud Computing: Services and Architecture (IJCCSA),*Vol.1, No.3, pp 1-13, November 2011.

[5] K. S. Trivedi and R. Sahner, "SHARPE at the age of twenty two," *ACM Sigmetrics Performance Evaluation Review*, vol. 36, no. 4, pp. 52–57, March 2009.

[6] L.Li "An Optimistic Differentiated Service Job Scheduling System for Cloud Computing Service Users and Providers", *IEEE Third International Conference on Multimedia and Ubiquitous* Engineering, pp.295-299, 2009.

[7] M. McNett, D. Gupta, A. Vahdat and G. M. Voelker. "Usher: an extensible framework for managing custers of virtual machines". In Proc. LISA, 2007.

[8] O. Khalid, I. Maljevic, R.Anthony, M.Petridis, K.Parrott and M. Schulz. "Deadline Aware Virtual Machine Scheduler for Grid and Cloud Computing" , *IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, pp.85-90,2010.

[9] R.Buyya, C. S. Yeo and Venugopal, "Market oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities". *Proceeding of the 10th IEEE International Conference on the High Performance Computing and* Communications,2008.

[10] R. N. Calheiros, R. Ranjany, and R.Buyya, "Virtual Machine Provisioning Based on Analytical Performance and QoS in Cloud Computing Environments", 2011.

[11] W.T.Tsai, Q.Shao, X.Sun and J.Elston "Real-Time Service-Oriented Cloud Computing",2011.