STOCHASTIC MARKOV MODEL APPROACH FOR EFFICIENT VIRTUAL MACHINES SCHEDULING ON PRIVATE CLOUD

Hsu Mon Kyi[1] and Thinn Thu Naing[2]

[1]University of Computer Studies, Yangon, Myanmar
hsumonkyi.ucsy@gmail.com

[2]University of Computer Studies, Yangon, Myanmar
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 significant issues in cloud computing system is the scheduling and allocation of virtual resources and virtual machines (VMs). To address this issue, this paper proposed an efficient approach for virtual machines scheduling in cloud infrastructure resource management and allocation also called EVMSA (Efficient Virtual Machines Scheduling Algorithm) that provides the effective and efficient resource allocation. To analyze the performance of this scheduling and allocation on cloud infrastructure, an analytical performance model approach using Stochastic Markov model is proposed to measure the scalability and tractability for infrastructure resource of private cloud. This model intends to analyze an academic-oriented private cloud system which is implemented using Eucalyptus open source system. According to performance evaluation, the effective mean response time of the system to improve the performance of IaaS services in the system.

KEYWORDS

Cloud Computing, Virtual Machine, Scheduling, Stochastic Markov Model, Eucalyptus Private Cloud

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 [10]. 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.

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 is that resource provision is more complex, since virtual machines and their host resources both need to be considered and compared with resource allocation in the traditional parallel and distributed system, virtual machines contain more properties which are used for scheduling, for example,

DOI : 10.5121/ijccsa.2011.1301

memory size, software packages, and access to specific devices. Moreover, cloud resource allocation and servicing is difficult for realistic analyze of cloud servicing request.

Existing Eucalyptus's default VM scheduler is ***Greedy scheduler*** and ***Round Robin scheduler***. The weakness of these approaches is basic single queuing systems that lead to maximize execution time due to longer jobs necessitating for the deployment of a better scheduling strategy at the cluster level. In order to improve throughput and minimize response time of system, we enhanced ***an efficient scheduling algorithm*** which is called Efficient Virtual Machines Scheduling Algorithm (EVMSA). In this algorithm, we use multiple queues and follow the FCFS principle. This algorithm uses three effective policies for resource allocation. These policies are (1) To decrease delay time for large VM request, this system uses three queues instead of single queue. Waiting VM requests are inserted to their appropriate queue according to the user request VM types. (2) To be fair all VM requests from three queues, the system compare the arrival time of first three queue for select queued VM request. (3) To be efficient utilize the resource, if selected VM request is not sufficient the resources, then we choose VM request from another queue due to small VM request size more sufficient resource than large VM request. According to these policies, the scheduler launches the specified VM(s) on physical resources according to FCFS principle. However, not every VM(s) request will be accepted by the scheduler since there may be not enough computing resource available in some clouds. Therefore, if there is no sufficient resource for the request, the scheduler will reject it and place this request on the appropriate queue. Then we choose VM request from another queue according to three scheduling policies.

To analyze the performance of efficient scheduling and allocation on cloud infrastructure, we use an analytic modelling approach using stochastic Markov models. Analytical performance model allows the system to predict the effects of a provisioning schedule on target QoS such as response time, throughput, and resource utilization. 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.

The two major contributions of this system are to:

- Develop an efficient scheduling mechanism to provide heterogeneous VM request types, for instances, variation in the number of cores, sizes of memory and sizes of storage.
- To analyze heterogeneous service request, we use Interacting Stochastic Markov model approach. This model result is to generate mean response time of request VM and system availability for user request.

The next sections will describe in detail: Section 2 discusses related work to this system and the overview of proposed cloud system is presented in section 3. This paper defines steps of the model approach in section 4. Then, numerical performance evaluation results are presented in section 5. Finally, Section 6 concludes the paper.

2. RELATED WORK

Since Eucalyptus [1] 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.

O.Khalid et al. [9] proposed a dynamic and adaptive real-time virtual machine scheduling technique for HPC workloads on the Grid. The primary objective is to increase overall job throughput in the system. L.Wang et al. [2] presented vGreen design to manage VM scheduling across different PMs with the objective of managing the over performance and system level energy savings.

N.Bobroff et al [8] proposed virtual machine placement algorithm to allocate virtual machines while minimizing the number of PMs activated without violating the SLA agreement. Similarity, L.Wangy et al. [6] presented a Multi-Dimensional Scheduling Algorithm (MDSA) for task scheduling in virtual machine based SOA environments. Each virtual machine is pre-installed with some application level software packages. This algorithm is static based scheduling algorithm.

Rodrigo N. Calheiros et al. [11] 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.

Luqun Li [5] 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.

To the best of our knowledge, the proposed efficient virtual machine scheduling (EVMSA) algorithm is appropriate for resource allocation and the heterogeneous VM request servicing of IaaS properties, for example, CPU, memory size, storage and software packages. And then Stochastic Markov Model approach is suitable for analyzing the performance of scheduling and allocation services of IaaS cloud system.

3. PROPOSED SYSTEM OVERVIEW

In this section, we present a component of Eucalyptus architecture and system model for this architecture.

3.1. The Components of Eucalyptus Architecture

In this section, we will briefly explain the overview architecture of Eucalyptus open source system. Eucalyptus architecture is deployed with some components: Cloud Controller (CLC) as front-end interface component, the several Cluster Controllers (CCs) in which Storage Controllers (SCs) are attached to provide the EBS block storage. Then the several Node Controllers (NCs) are working as back-end nodes. According to networking architecture point of view, the front-end node is configured with two network interfaces: one is connected to public campus network and another one is connected to private VM networks into Node Controllers (back-end node).

The main functions of Eucalyptus components are: ***Cloud Controller*** **(CLC) -** The CLC is responsible for exposing and managing the underlying virtualized resources (machines (servers), network, and storage) via user-facing APIs.

Walrus Storage Controller **(WS3)-** WS3 implements scalable "put-get bucket storage." The current implementation of Walrus is interface compatible with Amazon's S3 (a get/put interface for buckets and objects), providing a mechanism for persistent storage and access control of virtual machine images and user data.

Storage Controller **(SC) -** The SC provides block-level network storage that can be dynamically attached by VMs. The current implementation of the SC supports the Amazon Elastic Block Storage (EBS) semantics.

Cluster Controller **(CC)** - The CC controls the execution of virtual machines (VMs) running on the nodes and manages the virtual networking between VMs and external users.

Node Controller **(NC)** - The NC (through the functionality of a hypervisor) controls VM activities, including the execution, inspection, and termination of VM instances.

The Eucalyptus private cloud system architecture is shown in below.

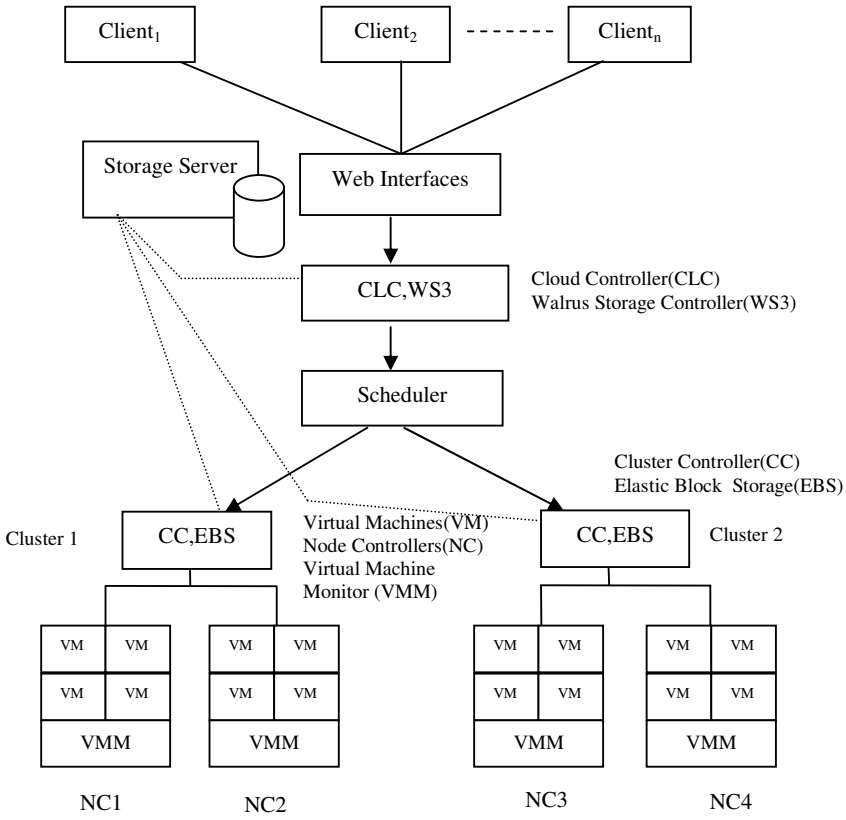

Figure 1. The Eucalyptus system architecture

3.2. System Model for Eucalyptus Architecture

This system model is constructed based on Eucalyptus private cloud infrastructure architecture. In such system, several 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. This system model uses the efficient virtual machine scheduling algorithm (EVMSA) using proposed effective scheduling policies to enhance FCFS scheduling policy. Using the EVMSA algorithm, the instances will be scheduled to run on proper physical machines so that it will have a higher performance.

4. STEPS OF PROPOSED MODEL APPROACH

This section describes hierarchical steps of the developed models and interaction among the models. Step by step processes of VM request in Eucalyptus cloud is shown in Figure 2 and the detail analysis of steps are described in below.

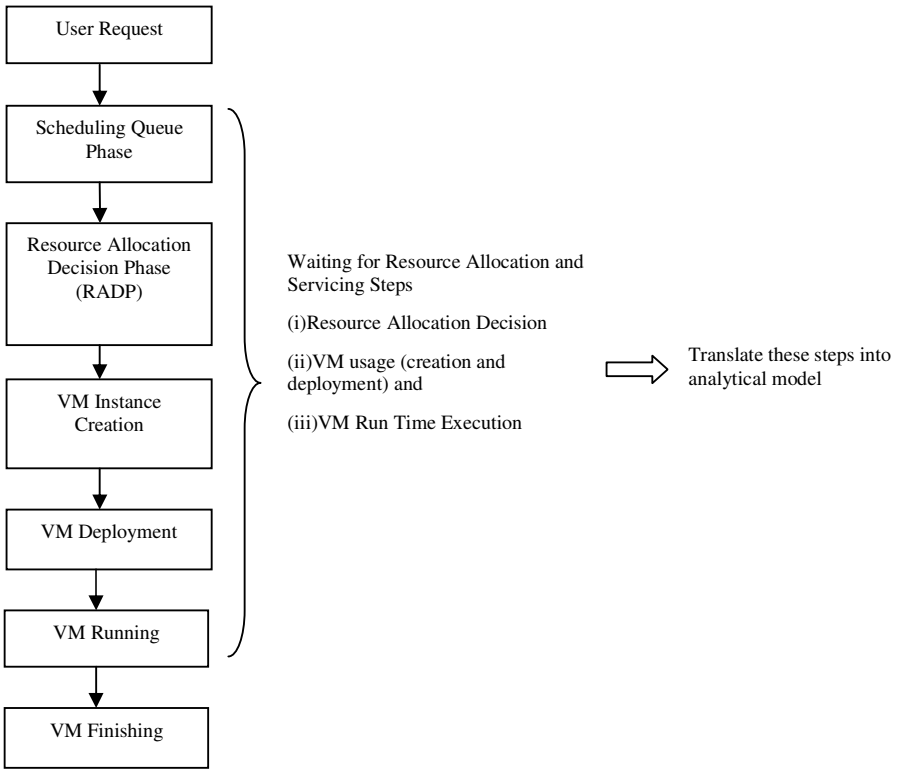

Figure 2. The Process of resource allocation and servicing

Resource allocation and servicing steps describe in figure translate three interactive sub models for analyze the performance of system. These steps based on Markov model; (1) resource allocation decision model, (2) VM usage model and (3) VM execution model respectively. These models are described below.

4.1. Resource Allocation Decision Model

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

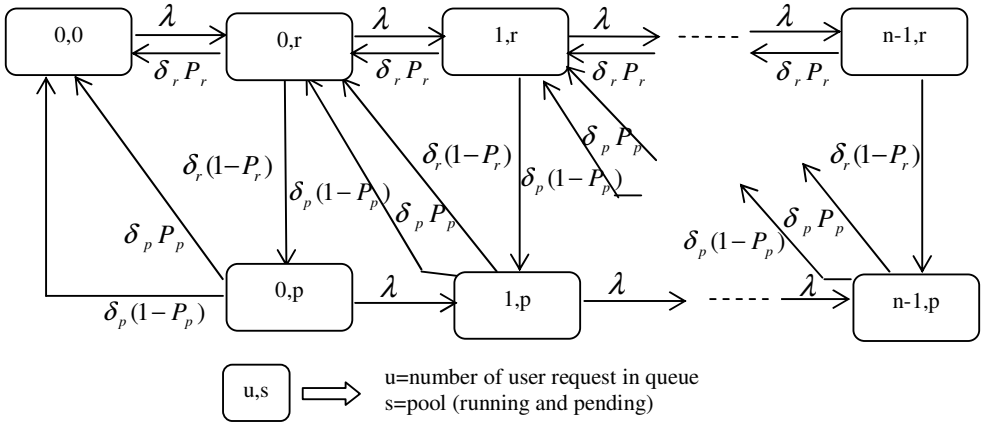

Figure 3. Resource allocation decision model

The system users arrive at the system with the Poisson rate λ. In this model, arrival user is u (u ∈ {1,…,n}). States in the model in Figure 3 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.

4.1.1. 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 δ_r, δ_p can be measure from Greedy search and P_r, P_p are compute from VM usage model. Outputs of this model are

(i) Average request service unavailable probability ($Service_{unavailabe}$) that a user request will be rejecting due to insufficient capacity.

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

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

$$Service_{available} = 1 - Service_{unavailable} \tag{2}$$

(iii) 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)

$$E[W_{RADP}] = \frac{\sum_{i=0}^{n-1} i(\pi_{(i,r)} + \pi_{(i,p)}) + \lambda(\frac{1}{\delta_r} + \frac{(1-P_r)}{\delta_p})}{\lambda(1-Service_{unavailable})} \tag{3}$$

4.2. Virtual Machines Usage Model

VM usage models capture the instantiation creation and deployment of a VM on a NC. In this model, we design separate VM usage models according to user request VM types. Three kind of request types are type1 (one CPU core for each request), type2 (two CPU cores for each request) and type3 (four CPU cores for each request). Figure 4 shows VM usage model for each user request for one VM which runs on one CPU core of a server in the running pool.

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 4 are indexed by (i,j,k), where, i denotes number of request VM in the queue, j denotes number of VMs currently being provisioned, k denotes the number of CPU cores on a NC which have already been deployed. In Figure 4, Q_r is buffer size in each PM and M is maximum number of VMs that can be deployed on a NC for the type 1 request. In this model, From state (0,0,0), after a job arrival, model goes to state (0,1,0), with rate λ_r. In state (0,1,0), a VM instance is created. Mean time to creation a VM on a running PM, is $1/\beta_r$ and the model moves from (0,1,0) to (0,0,1) with rate β_r. Upon service completion, VM instance is removed and the model moves from (0,0,1) to (0,0,0) with rate μ; this rate is computed as an output from the VM execution model. When a VM is being provisioned in state (0,1,0), arrival of a new job will take the model to state (1, 1,0), where the new job is waiting in the queue. In this usage model, input and output parameters are discussed in the following.

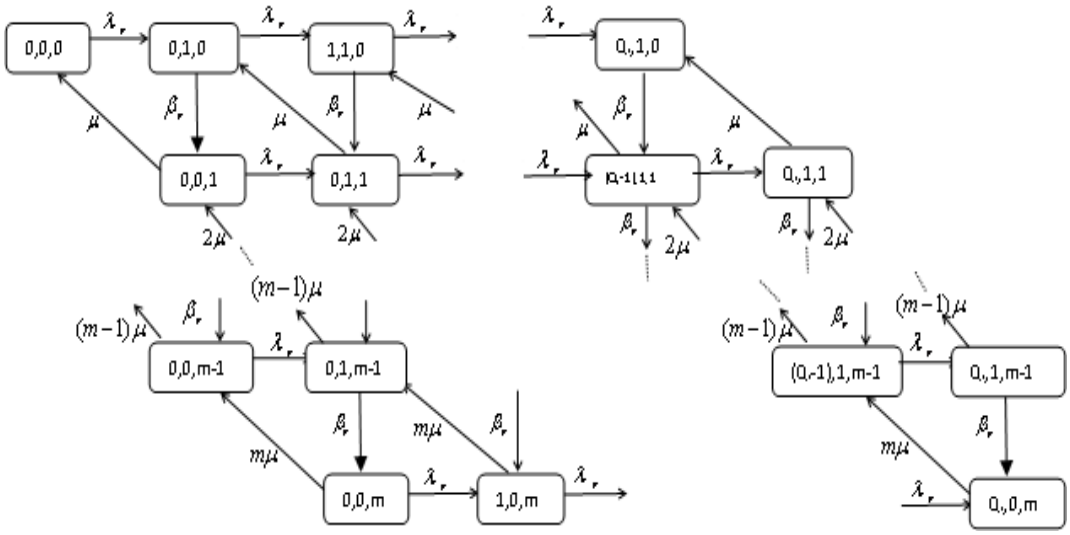

Figure 4. Virtual Machines usage model

4.2.1. Model Input and Output

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

$$\lambda_r = \frac{\lambda}{H_r} \tag{4}$$

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 (π_r) that a running NC cannot accept a job for all request VM type provisioning:

$$\pi_r = \sum_{i=0}^{M-1} \pi^r_{(Q_r,1,i)} + \pi^r_{(Q_r,0,M)} \tag{5}$$

(ii) Probability for all VM request type that a user request can be accepted in the running pool

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

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

(i) the arrival rate λ_p to each pending NC is given by:

$$\lambda_p = \frac{\lambda(1-P_r)}{H_p} \quad (7)$$

(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 (π_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:

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

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:

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

4.3. Virtual Machine Execution Model

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

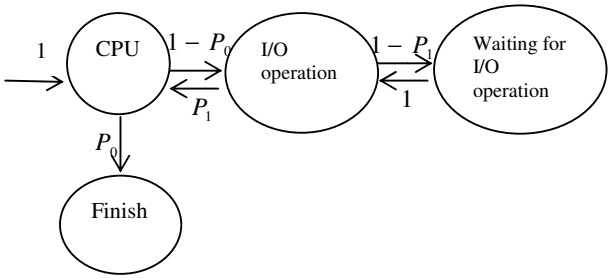

Figure 5. Virtual Machines execution model for each Virtual Machine request

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 labelled 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 $1/\mu_c$, $1/\mu_l$ and $1/\mu_w$ respectively, we compute the mean VM service time:

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

5. NUMERICAL PERFORMANCE EVALUATION RESULT

We evaluated cloud user VM request services are two solutions- (1) Service request available 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.

An example two scenarios are considered for this model output. First is maximum of one VM on each NC, buffer size in front of RADP to be 20, and buffer size within each NC to be zero. In this stochastic model, resource allocation decision model (in our example, 41 states) and VM usage models (for each PM model respective number of states are 3and 4) are solved in this system. Next scenario is similar to the first except eight VM on each NC.

All models were solving using SHARPE [4] software package. Assume values of key parameters are shown in Table 1.

Table 1. 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 instantiation and deployment a VM on a running server	8 minutes
$1/\beta_p$	Mean time to VM for instantiation 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	15-30 minutes
λ	Cloud user request VM arrival time	300-500 request/hr
H_r	Number of running NC in running pool	8-16 NCs
H_p	Number of pending NC in pending pool	8-16 NCs

Table 2 shows numerical result of a fixed mean service time (15 minutes for each VM request) and different arrival rate at different number of NCs. In this experiment, decrease user request service available probability at increasing arrival rate. Observe arrival rate at 300, 350, 400, 450 and 500 user request VM an hour, for an increase in capacity from 8 to 16 NCs in each pool

Table2. User request service available probability at different arrival rate

Arrival rate (VMs request/hr)	Comparison of service availability result in running and pending pools					
	(8,8)		(12,12)		(16,16)	
	$Service_{available}$ (1VM)	$Service_{available}$ (8VMs)	$Service_{available}$ (1VM)	$Service_{available}$ (8VMs)	$Service_{available}$ (1VM)	$Service_{available}$ (8VMs)
300	0.695	0.99585	0.9776	1	0.999986	1

350	0.620	0.98620	0.9495	1	0.999749	1
400	0.537	0.96710	0.9090	0.999999	0.998870	1
450	0.442	0.93770	0.8610	0.999990	0.992190	1
500	0.373	0.89700	0.8090	0.999836	0.986200	1

Different service time and fixed arrival rate (350 user request VM/hr) results are shown in Table 3. In these tables result, service available probability increase at increasing number of VM in each pool.

Table3. User request service available probability at different service time

Service time for each VM request(min)	Comparison of service availability result in running and pending pools					
	(8,8)		(12,12)		(16,16)	
	$Service_{available}$ (1VM)	$Service_{available}$ (8VMs)	$Service_{available}$ (1VM)	$Service_{available}$ (8VMs)	$Service_{available}$ (1VM)	$Service_{available}$ (8VMs)
15	0.620	0.98620	0.9495	1	0.999749	1
20	0.563	0.92104	0.9175	1	0.998670	1
25	0.493	0.08163	0.8640	0.999999	0.993460	1
30	0.404	0.04630	0.8060	0.999987	0.980700	1

In our experiment for first scenario, figure 6(a) shows, at a fixed arrival rate (350 cloud user request VM /hr) and mean response time increase at increasing service time. Figure 6(b) shows that with increasing arrival rate, mean response time 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 an increase in capacity from 8 to 16 NCs in each pool.

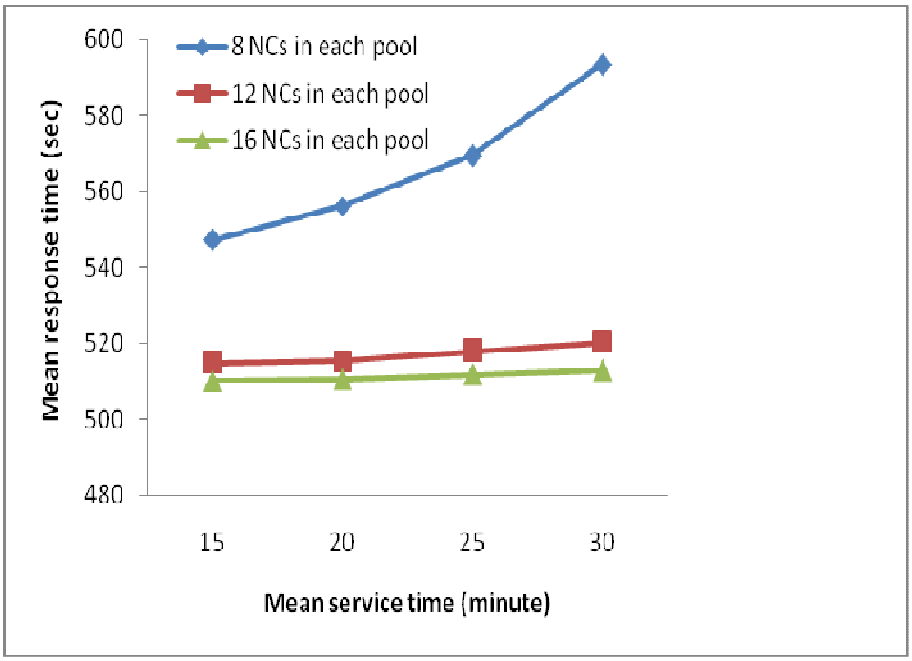

Figure 6(a). Mean response time for different service time and fixed arrival rate (350 user request VM/hr) at different number of NCs

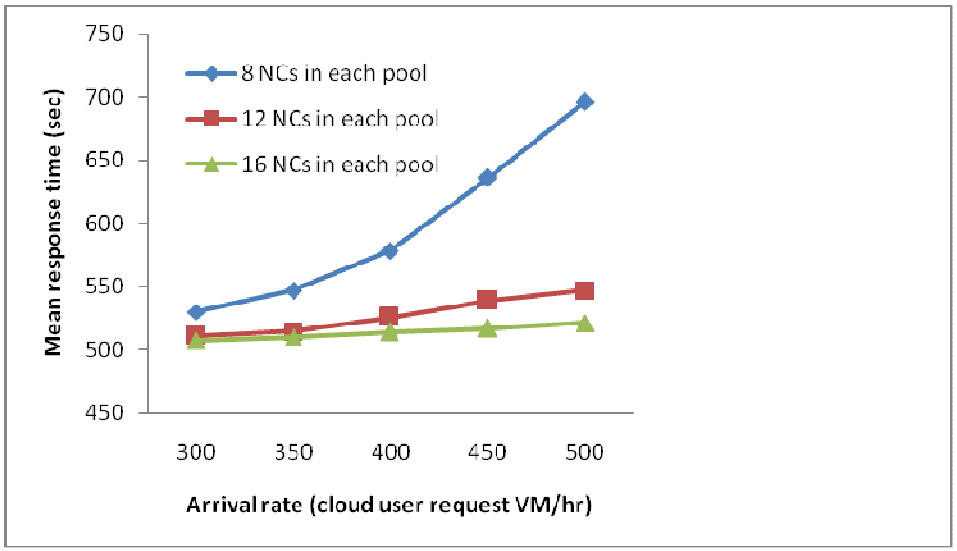

Figure 6(b). Mean response time for different arrival rate and fixed mean service time (15 minutes) at different number of NCs

In second scenario, figure 7(a) shows that a fixed arrival rate (350 cloud user request VM /hr) and, increases mean response time at increasing mean service time. And also, Figure 7(b) shows that with increasing arrival rate, mean response time increases for a fixed number of NCs in each pool. According to these scenarios, the more number of VM in each NC the more performance for this system.

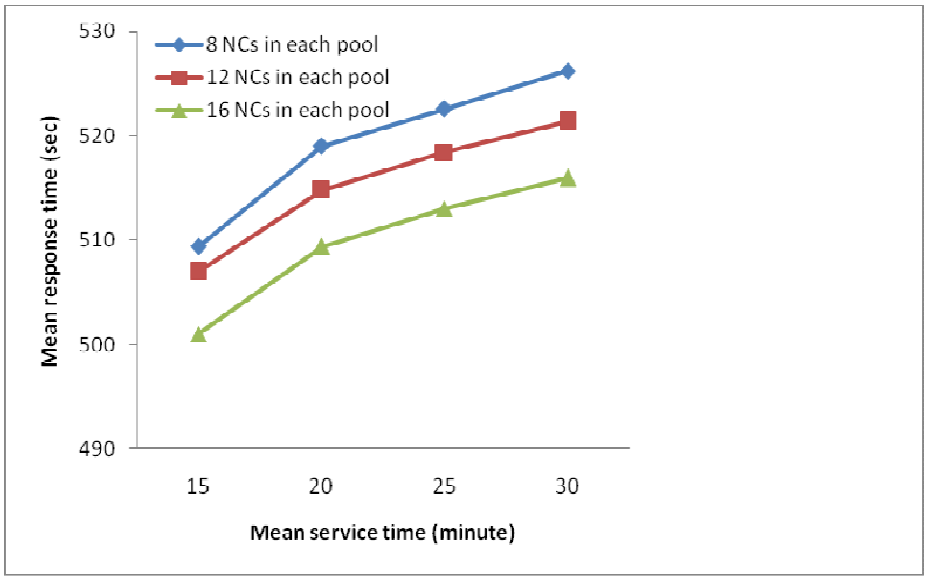

Figure 7(a). Mean response time for different service time and fixed arrival rate (350 user request VM /hr) at different number of NCs

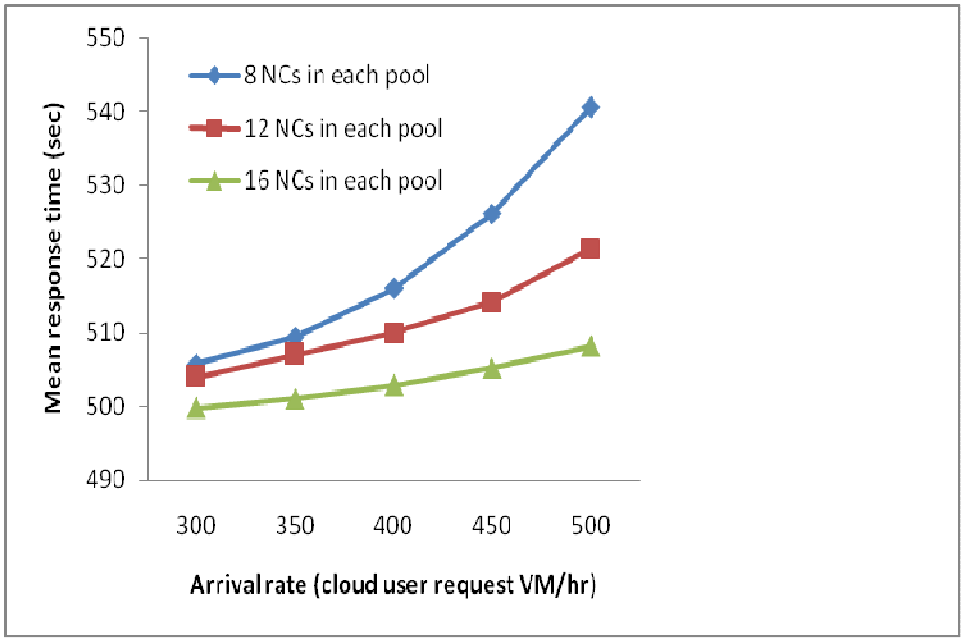

Figure 7(b). Mean response time for different arrival rate and fixed mean service time (15 minutes) at different number of NCs

6. CONCLUSIONS

It has been widely accepted that virtual machines can be employed as computing resources for high performance computing. Thus, virtual machine scheduling and resource allocation is essential in cloud computing environment. Therefore, we present Stochastic Markov model for evaluate the performance of resource scheduling and allocation on Eucalyptus private cloud system. In this paper, we quantify the effects of variations in workload (e.g., user request arrival rate, service rate of VM) and system capacity (NCs in each pool) on cloud service quality. Our approach is tractable and captures many realistic features of a large sized cloud, with reduced complexity of analysis.

REFERENCES

[1] 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.

[2] G. Dhiman, G.Marchetti, T. Rosing. "vGreen: A System for Energy Efficient Computing in Virtualized Environments", ACM, In Proc.ISLPED,2009.

[3] Hongbin Liang, Dijiang Huang, LinX.Cai, Xuemin (Sherman) Shen and Daiyuan Peng, "Resource Allocation for Security Services in Mobile Cloud Computing", IEEE,pp. 191-195 ,May 2011.

[4] 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.

[5] 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.

[6] L.Wangy, G.V.Laszewskiy, M.Kunzez and J.Taoz "Schedule Distributed Virtual Machines in a Service Orientedent Environment", 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] N.Bobroff, A.Kochut and K.Beaty "Dynamic placement of virtual machines for managing SLA violations",2009

[9] O.Khalid,I.Maljevic and R.Anthony. "Dynamic Scheduling of Virtual Machines Running HPC Workloads in Scientific Grids"IEEE,2009.

[10] 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.

[11] Rodrigo N. Calheiros, Rajiv Ranjany, and Rajkumar Buyya, "Virtual Machine Provisioning Based on Analytical Performance and QoS in Cloud Computing Environments", 2011.

Authors

Hsu Mon Kyi. She is received the Bachelor of Computer Science degree and Master of Computer Science degree from University of Computer Studies, Yangon in 2003 and 2007 respectively. Currently she is a Ph.D student in University of Computer Studies, Yangon (Myanmar) as well as currently, she is an assistant lecturer in Computer Science at University of Computer Studies, Yangon (Myanmar). Her research interests are Distributed Computing, Cloud Computing and Virtualization Technology.

Thinn Thu Naing. She obtained her Ph.D degree in Computer Science from University of Computer Studies, Yangon in 2004, Bachelor of Computer Science degree and Master of Computer Science degrees in 1994 and 1997 respectively from University of Computer Studies, Yangon. Currently, she is a Professor in Computer Science at University of Computer Studies, Yangon (Myanmar). Her specialization includes Cluster computing, Grid computing, Cloud computing and Distributed computing.