Computer Networks 91 (2015) 453–470

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Modeling of the resource allocation in cloud computing centers

Shahin Vakilinia*, Mustafa Mehmet Ali, Dongyu Qiu

Department of Electrical and Computer Engineering, Concordia University, 7141 Rue Sherbrooke West, QC H3G 1M8, Canada

ARTICLE INFO

Article history:
Received 25 September 2014
Revised 12 July 2015
Accepted 21 August 2015
Available online 8 September 2015

Keywords:
Cloud computing
Queuing systems
Resource allocation
Markov process

ABSTRACT

Cloud computing offers on-demand network access to the computing resources through virtualization. This paradigm shifts the computer resources to the cloud, which results in cost savings as the users leasing instead of owning these resources. Clouds will also provide power constrained mobile users accessibility to the computing resources. In this paper, we develop performance models of these systems. We assume that jobs arrive to the system according to a Poisson process and they may have quite general service time distributions. Each job may consist of multiple numbers of tasks with each task requiring a virtual machine (VM) for its execution. The size of a job is determined by the number of its tasks, which may be a constant or a variable. The jobs with variable sizes may generate new tasks during their service times. In the case of constant job size, we allow different classes of jobs, with each class being determined through their arrival and service rates and number of tasks in a job. In the variable case a job generates randomly new tasks during its service time. The latter requires dynamic assignment of VMs to a job, which will be needed in providing service to mobile users. We model the systems with both constant and variable size jobs using birth–death processes. In the case of constant job size, we determined joint probability distribution of the number of jobs from each class in the system, job blocking probabilities and distribution of the utilization of resources for systems with both homogeneous and heterogeneous types of VMs. We have also analyzed tradeoffs for turning idle servers off for power saving. In the case of variable job sizes, we have determined distribution of the number of jobs in the system and average service time of a job for systems with both infinite and finite amount of resources. We have presented numerical results and any approximations are verified by simulation. The results of the paper may be used in the dimensioning of cloud computing centers.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Reduced costs of processing and storage technologies brought about rapid growth of computing resources industry. Recently, a new computing paradigm called cloud computing emerged which provides on-demand network access to the computing resources through virtualization. This paradigm offers cost savings because users lease the computing resources from a service provider when needed instead of owning them. Further, clouds will provide mobile users access to computing resources, which is referred to as mobile cloud computing [1]. This is very important as mobile devices are becoming primary computing platform to many users and they have limited processing power and battery life. Cloud computing enables dynamic sharing of the computing resources among the users. A service level agreement (SLA) specifies the quality of service (QoS) to be provided to the user in terms of various performance parameters such as throughput, reliability, blocking probability and response time. Cloud computing services may be classified into three types as Infrastructure-as-a-service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). IaaS refers to providing hardware equipment such as CPU,

* Corresponding author. Tel.: +15148365580.
E-mail addresses: s_vakili@ece.concordia.ca, shahin.vakilinia@gmail.com (S. Vakilinia), Mustafa@ece.concordia.ca (M.M. Ali), dongyu@ece.concordia.ca (D. Qiu).

http://dx.doi.org/10.1016/j.comnet.2015.08.030
1389-1286/© 2015 Elsevier B.V. All rights reserved.

memory and storage as a service, PaaS refers to providing platforms such as software development frameworks, operating systems or multi-tenant application supports as a service and SaaS providing software and applications as a service. In this paper, we only consider cloud computing centers that provide the IaaS service through leasing of virtual machines (VMs) to the users [2].

In general, the topology of a cloud computing center is hierarchical with racks containing a fixed number of blade servers. A blade server contains a number of processors each one consisting of several processing cores. The processing cores, memory and storage space are configured into VMs. VMs may be homogeneous or heterogeneous. In the first case, VMs have the same number of CPUs, memory and storage sizes, while in the second case, there may be different VM types which may differ from each other in number of CPUs, memory and storage sizes [1].

Jobs entering the system may demand different types of services. However most of them require parallel data analysis [3]. This is the main reason for the recent development of MapReduce Programming model [4]. This model relies on parallel processing with a sequential functional approach. Job fragments are executed in parallel to speed up processing of the jobs. MapReduce has usually three phases as fan out, map and reduce. Applications such as Apache Hadoop [5] and platforms such as Pig [5] implement the MapReduce programming model.

This model also applies to bag-of-tasks (BoTs) where a job consists of parallel and sequential tasks. Number of tasks executing in mapping phase will be larger than fan out and reduce phases, thus dynamic resource allocation will benefit this programming model.

Mobile devices such as smartphones and tablet PC are increasingly becoming part of everyday life. These devices provide many capabilities such as GPS, WiFi and cameras. As a result, developers are building more and more complex mobile applications such as gaming, navigation, video editing, etc. for these devices. Though hardware of these devices is becoming more powerful, they are not able to keep up with the computational, storage and energy demands of more complex applications and they have short battery life [6]. Mobile cloud computing (MCC) is a derivative of cloud computing and its goal is to serve mobile users [7]. The MCC is expected to provide on-demand processing power and storage for mobile users in the cloud. This will enable mobile devices to offload their work to the cloud at a finer granularity [8]. Khan et al. [9] provides a survey of the proposed application models for mobile cloud computing. The various application models differ from each other in terms of design and objectives. Depending on the workload of the mobile device, number of VMs assigned to it will be dynamically changing. In [10], a method level offloading to the cloud has been proposed in order to take advantage of the parallelism in the application. In the experiments reported in [10], the average time to resume a VM from the pause state is around 300 ms while from the powered-off state is 32 s.

In this paper, we will consider various cloud computing models that may be used in the dimensioning of these systems. We will consider systems with both homogenous and heterogeneous types of VMs.

We assume that the job arrivals will be according to a Poisson process. A job may consist of multiple numbers of tasks and execution of each task requires a VM. The size of a job in number of tasks may be a constant or may vary dynamically during its service time. In the case of constant job size, the size is chosen from a discrete probability distribution. For this case, we consider two service types, which are simultaneous and individual completion of the tasks. In the simultaneous subcase a job is assigned a service time at the end of which all its tasks terminate. In the second subcase, tasks of a job receive independent and identically distributed service times, which results in individual task service completions.

In the case of variable job size, the size of a job varies during the time that it is in the system. A job initially has a single task, however, it generates new tasks according to a Poisson process during its service time. The service times of the tasks are independent and identically distributed and each one requires a VM for its execution. Thus the number of tasks belonging to a job during its service time will be a random variable. A job is completed when all the tasks belonging to that job complete their service times. A job with variable size may be appropriate for modeling of service demands of mobile devices.

In the following sections of the paper we present performance analysis of the cloud computing models described in the above. Main contributions of this paper are as follows:

- We have considered systems with multiple classes of jobs with constant job sizes in number of tasks with homogeneous VMs. Assuming Poisson arrival of jobs with arbitrary service distributions, we have determined job blocking probabilities of each class and distribution of the utilization of resources under single server, multiple-server and multiple-server pool cases. In multiple-server case, we have determined fragmentation probability of a job's service among multiple servers. We have shown applicability of our results to study a power management algorithm that reduces the power consumption while maintaining a plausible job blocking probability under time-varying traffic load.
- We also derived job blocking probabilities and distribution of the utilization of resources with multiple classes of jobs with heterogeneous VMs.
- We determined probability distribution of the service time and average number of jobs for a system with constant job sizes and independent task completion times.
- We considered a system with jobs arriving to the system according to a Poisson process with variable job size in number of tasks. It is assumed that a job will generate new tasks randomly during its service time in the system. We have derived service time distribution of a job, distribution of the number of jobs and total number of tasks in the system.

The remainder of this paper is organized as follows. Related work is discussed in Section 2. In Section 3, we study systems with homogeneous VMs with constant job sizes and simultaneous task release times. Sections 4 and 5 extend the analysis of Section 3 to systems with heterogeneous VMs and jobs with independent task release times respectively. In Section 6, we present modeling of a system with variable

job size. In Section 7, we give a comparison of our results with the closest previous work that has been referred to in Section 2 and Section 8 contains the conclusions.

2. Related work

A number of papers have been published on performance modeling of the cloud computing [11–16,24–26]. In [11], a cloud computing center has been modeled as a M/G/m/m + r queue where *r* is the size of the buffer that stores the waiting jobs. A new arriving job to a full buffer is lost and the jobs in the buffer are served on FCFS basis. The steady-state distribution of the queue length is determined by writing down the transition probability matrix of the embedded Markov chain at the arrival points and solving the equilibrium equations numerically. The analysis makes the approximation that at most three jobs may be served during an inter-arrival time and the queue length distribution does not have a closed form. In [24], this analysis has been extended to the jobs where each job contains random number of tasks. In [25], performance of cloud computing systems has been studied using stochastic reward networks (SRNs) which are an extension of generalized stochastic Petri Nets (GSPNs). In [26], performance of cloud computing systems with fault recovery has been considered.

In [12], cloud computing capacity has been studied under time-varying traffic load using historical traces with the assumption that idle capacity is turned off through simulation. The time-axis has been divided into slots of 5 min duration and various moving average and autoregressive models have been used to predict the job demand for the next slot using the demands for the present and past slots. Then the needed server capacity for the predicted load was determined using Erlang loss formula, as a result extra capacity may be added or subtracted to/from presently active capacity respectively. It is assumed that it takes one slot to turn on the extra capacity. The unneeded capacity is turned off after one slot to prevent unnecessary on–off turning of the servers. It is assumed that an arriving job will be blocked and lost if there is no available active capacity to serve it. Under this scheduling algorithm, the paper determined job blocking probabilities and unutilized server capacity for prediction models as well as for a model that maintains a fixed reserved capacity using simulation. It has been determined that fixed reserve capacity provides better performance than the prediction models.

In [13], throughput optimal load balancing models has been considered in systems with cluster of servers. The work assumes heterogeneous type of VM configurations. The time-axis is slotted and in each slot a number of job requests arrive to the system. Each job may request a single VM for a number of slots. When the system is busy the arriving jobs are stored in a central queue for each type of jobs. It is shown that server-by-server maxweight job scheduling with preemption and server reconfiguration in each slot is throughput optimal. A non-preemptive algorithm, which is nearly optimal, has also been given. To reduce the communication overhead a more distributed system is also considered where each server maintains its own queues. It has been shown that new arrivals joining to the server with shortest queue and servers using maxweight job scheduling is throughput optimal. The paper also presents simulation results, which show that mean delay performance of centralized and distributed queuing systems are not very different. The paper does not take into consideration QoS requirements of different type of jobs which may not be met in this process.

Stolyar and co-workers [14,15] consider optimization of a cloud computing center w.r.t. communication bandwidth demands. The sum of the bandwidth requirements of VMs on a server may exceed the capacity of a server's network interface. Since bandwidth demands of the VMs are stochastic, statistical multiplexing may be used to place VMs on minimum number of servers such that VMs bandwidth guarantees may be met probabilistically. This problem may be modeled as a Stochastic Bin Packing (SBP) problem. Under the assumption that a VM's bandwidth consumption is normally distributed, the paper presents approximate online and offline algorithms for the optimal assignment of VMs to the servers.

Recently, Amazon introduced a new cloud computing service that sells the idle instances of resources called Spot Instance (SI) through competitive bidding. The price of SI depends on the demand but in general it is lower because no reliability is provided for the services. In [16], a statistical modeling of the SI prices and inter-price durations has been provided through curve-fitting to the experimental data available from Amazon.

The modeling in our paper is closest to the work in [11]. The differences with [11] arise from the structure of job requests and service discipline. We allow more complicated job requests than in [11]. We consider systems where jobs may generate new tasks during their service. They allow limited queuing of the job requests while as in [13], we assume blocking when resources are not available. Finally, we also consider systems with heterogeneous VMs and in general our results have closed forms. We study the performance tradeoffs using an analytical model under time-varying traffic load.

3. Modeling of a system with homogeneous VMs, constant job sizes and simultaneous release times

In this section, we assume multiple classes of jobs. Each class of jobs arrives at the system according to a Poisson process with a different parameter and each class has a different service rate and job size. The size of a job is determined by the number of tasks that it has and the job size remains constant during its service time. Each task requires a VM for its execution. Distribution of the service times of jobs may have rational Laplace transforms with a different mean service times for each class. Service time of a job begins with its arrival to the system and at the end of that service time all its tasks terminate simultaneously. In other words, processing units related to an arriving job are provisioned and released together. The notation has been introduced in Table 1.

We will consider single and multiple servers and multiple server pools cases. We assume finite resources, thus a job will be blocked if there are no enough number of idle VMs to serve it. The objective of the following analysis will be to determine joint distribution of the number of jobs from each class, job blocking probabilities and distribution of the utilization of resources. We will also show applicability of our results into power management in a cloud computing center.

Table 1
Parameter definitions.

R	number of classes of jobs
λ_r	arrival rate of class r jobs
λ_T	total job arrival rate
μ_r	service rate of class r jobs
k_T	total number of busy VMs
b_r	number of VMs required by a class r job
n_r	number of class r jobs in the system

Let us define state of the system as number of jobs from each class in the system, $\vec{n} = (n_1, n_2, \ldots, n_r, \ldots n_R)$, and $p(\vec{n})$ as the distribution of $\vec{n}$.

3.1. Single server model

First, we consider a system with finite resources of S VMs all located at a single server. In this case, an arriving job will be lost if there are not enough number of idle VMs to serve it. This model is same as blocking in shared resources environment studied in [17]. From there, the joint probability distribution of the number of jobs in the system is given by,

$$p(\vec{n}) = \frac{1}{G}\prod_{r=1}^{R}\frac{\rho_r^{n_r}}{n_r!} \tag{1}$$

where G is the normalization constant, which may be determined through a recursion [17] and $\rho_r = \frac{\lambda_r}{\mu_r}$. Let j denote number of the busy VMs at the computing center, then, $j = \vec{n}.\vec{B}^T$ where $\vec{B} = [b_1, \ldots b_r, \ldots b_R]$. Defining probability distribution of the number of busy VMs in the computing center as,

$$q(j) = Pr(j = \vec{n}.\vec{B}^T)$$

from [17], $q(j)$ is given by the following recursion,

$$jq(j) = \sum_{r=1}^{R} b_r \rho_r q(j - b_r) \tag{2}$$

Then average number of busy VMs in the system is given by,

$$E[k_T] = \sum_{j=1}^{S} jq(j) \tag{3}$$

Let $\tilde{q}(\ell)$ denote probability distribution of the number of idle VMs, then, $\tilde{q}(\ell) = q(S - \ell)$. Defining PB_r as the probability that a class r job will be blocked, then from [17],

$$PB_r = \sum_{\ell=1}^{b_r - 1}\tilde{q}(\ell) = 1 - \frac{G(S - b_r, R)}{G(S, R)} \tag{4}$$

where $G(S - b_r, R)$ may be calculated recursively [17].

The overall job blocking probability is given by,

$$P_b = \frac{1}{\lambda_T}\sum_{i=1}^{R}\lambda_i PB_i$$

where $\lambda_T = \sum_{i=1}^{R}\lambda_i$.

As may be seen from (1), the joint distribution of the number of jobs in the system depends on the service time only through its mean.

3.2. Multiple servers model

Next, we consider a system with M servers where each server has S VMs.

As before, an arriving job will be blocked if the total number of idle VMs in the computing center is less than the number of VMs needed to serve the arriving job. Thus as far as job blocking probabilities are concerned the system may be considered as a single server with a total of MS VMs. However, in this case, it is possible that no server may have enough number of idle VMs to serve an accepted job to the system and the job may need to be assigned VMs from multiple servers which will be referred to as fragmented service. As a result, these jobs will experience additional performance penalty due to the need for communication among the servers. Henceforth, we determine the probability that assigned VMs to an accepted job will be fragmented among servers. Let us introduce the following additional notation,

V	total number of VMs at the computing center.
j	total number of busy VMs in the computing center.
$\vec{\varepsilon}$	$(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m, \ldots \varepsilon_M)$ where corresponds to the number of idle VMs at an arbitrary time in the mth server.

The total number of VMs in the datacenter is given by:

$$V = MS \tag{5}$$

Let ℓ denote the total number of idle VMs in the computing center,

$$\ell = \sum_{m=1}^{M}\varepsilon_m = V - j \tag{6}$$

Since b_r denote the number of VMs required to provide service to a class r job, depending on the value of the ℓ, the following possibilities exist for a class r job:

$$\begin{cases} \text{job will be blocked, if } \ell < b_r \\ \text{job may receive fragmented service,} \quad b_r \le \ell < Mb_r \\ \text{job will receive service from a single server,} \quad Mb_r \le \ell \end{cases}$$

Assuming a load balancer is operating in the system, then probability distribution of the number of idle VMs in each server will be identical. Given that total number of idle VMs is equal to ℓ, let $P(\ell, M, b_r)$ denote the conditional probability that none of the M servers have b_r or more idle VMs:

$$P(\ell, M, b_r) = Pr(\varepsilon_1 < b_r, \ldots, \varepsilon_m < b_r, ldots, \varepsilon_M < b_r) \tag{7}$$

Distribution of the number of idle VMs in servers is analogous to the traditional balls urn model, where each ball is placed into one of the urns with equal probability. Then, distribution of the number of idle VMs in each server will be the same as distribution of the balls in the urns model [18]. $P(\ell, M, b_r)$ does not have a closed form solution but it could be obtained recursively [18],

$$P(\ell + 1, m, b_r) = P(\ell, m, b_r) - \binom{\ell}{b_r} P(\ell - b_r, m - 1, b_r)\frac{(m-1)^{\ell - b_r}}{m^{\ell}} \tag{8}$$

with the following initial condition,

$$P(\ell, m, b_r) = 1 \quad \text{for} \quad \{1 \le \ell \le b_r, 1 \le m \le M\}$$

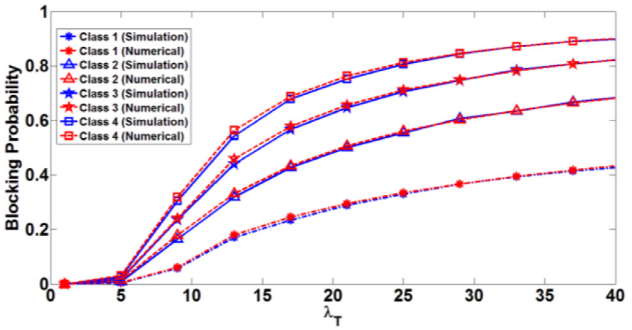

Fig. 1. Numerical and simulation results for blocking probabilities of different classes of jobs as a function of total job arrival rate.

The following result may be used to simplify the above recursion,

$$\binom{\ell}{b_r}\frac{(m-1)^{n-b_r}}{m^n}=\left(\frac{m-1}{m}\right)\left(\frac{\ell}{\ell-b_r}\right)\left\{\binom{\ell-1}{b_r}\frac{(m-1)^{\ell-1-b_r}}{m^{\ell-1}}\right\} \tag{9}$$

Note that $P(\ell, M, b_r)$ gives the probability of a class r job receiving fragmented service when $b_r \le \ell < Mb_r$. If $Mb_r \le S$ then $\ell \le S$, then all assignment combinations of idle instances of V resources into servers are feasible. But if $S < Mb_r$ it is possible that $\ell > S$, then some assignment of idle VMs to the servers will not be admissible because it will result in allocation of more idle VMs to a server than the capacity of that server. The non-admissible assignments of idle VMs have to be excluded through normalization. Let $\tilde{P}(\ell, M, b_r)$ denote the probability that a class r job receives fragmented service, thus:

$$\tilde{P}(\ell, M, b_r)=\begin{cases}P(\ell, M, b_r), & \text{if } \ell \le S\\ \frac{P(\ell, M, b_r)}{1-\sigma}, & \ell > S\end{cases} \tag{10}$$

where $\sigma=\sum_{k=S}^{\ell} P(\ell, M, k)$ and $P(\ell, M, k)$ is obtained from (8). Next, unconditioning the above result w.r.t. the distribution of the number of idle VMs leads to the probability that a class r job will receive fragmented service. Defining,

$PF_r = Pr$ (an accepted class r job receives fragmented service)

Then, it is given by,

$$PF_r=\frac{\sum_{\ell=b_r}^{Mb_r-1}\tilde{P}(\ell, M, b_r)\tilde{q}(\ell)}{1-PB_r} \tag{11}$$

In the above, denominator normalizes the fragmentation probability with the probability of accepting a job.

Next, we present numerical and simulation results for a computing center with multiple servers. Discrete event-based simulation has been developed to determine accuracy of the assumption in the analysis that the number of idle VMs is uniformly distributed over multiple servers. Simulation implements a practical load balancer to be described below to achieve fair distribution of the load among the servers. In simulation also it has been assumed that jobs arrive into the system according to a Poisson process and job service times are exponentially distributed.

We consider a system with $M = 5$ servers with $S = 50$ VMs per server. We assume 4 classes of jobs with the following VM requirements and job arrival rates,

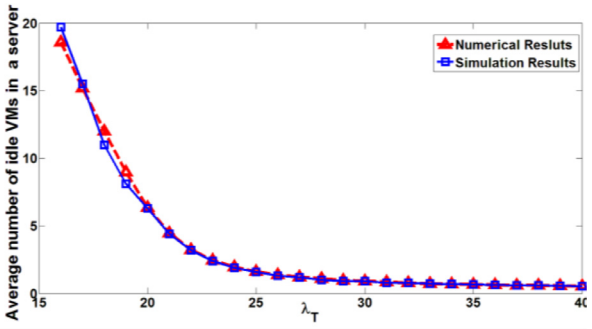

Fig. 2. Numerical and simulation results for the average number of idle instances of resources per server as a function of total job arrival rate.

$$\vec{B}=[b_1\ b_2\ b_3\ b_4]=[1\ 2\ 3\ 4] \tag{12}$$

$$\lambda=[\lambda_1\ \lambda_2\ \lambda_3\ \lambda_4]=[0.4\ 0.3\ .2\ 0.1]\lambda_T \tag{13}$$

It may be seen that jobs with smaller VMs requirements have been assigned higher arrival rates. Fig. 1 presents blocking probabilities of different classes of jobs as a function of the total job arrival rate. As may be seen, blocking probabilities increases with the number of VMs required by a job class. As expected, there is total agreement between numerical and simulation results as the analysis for calculation of job blocking probabilities is exact.

Next we present the results concerning service fragmentation. In simulation, using a load balancer, it is assumed that a server selection algorithm attempts to achieve fair distribution of the load among the servers. An accepted job if possible will be given service without fragmentation otherwise with fragmentation. If a job receives service without fragmentation, then it is assigned to the server with highest number of idle VMs. On the other hand, if a job receives service with fragmentation the scheduling algorithm aims to minimize the number of fragments depending on the distribution of the number of idle VMs in the servers. In Figs. 2 and 3, we present average number of idle VMs in a server and job fragmentation probabilities for each class as a function of the total job arrival rate. The jobs with higher VM requirements experience higher fragmentation probabilities at any total arrival rate. From Fig. 3, the fragmentation probability of class 4 jobs reaches to 30% at the total job arrival rate of 30. Job

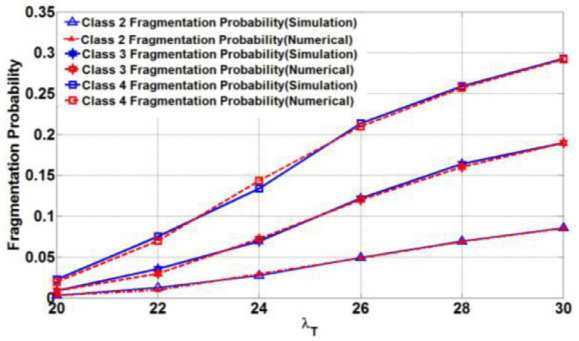

Fig. 3. Numerical and simulation results for fragmented service probabilities of different classes of jobs as a function of total job arrival rate.

fragmentation will increase the communication latency between the VMs, which will increase job service times. As may be seen, there is a close agreement between numerical and simulation results in both figures, which validates the assumption in the analysis that the number of idle VMs is uniformly distributed across the multiple servers.

3.3. Multiple server pools model

In this subsection, we extend our model to cloud computing centers with pools of servers. Pool management techniques attempt to reduce power consumption of the system, which represents a significant component of the operating cost of a cloud computing center. Topology of the cloud computing center under consideration is shown in Fig. 4. These techniques turn off a server pool to save power if its servers are not currently serving any job. Let us assume that there are N server pools in the system, which are numbered as, $n = 1 \ldots N$. We assume that scheduling algorithm always assigns a job to the server pool with the smallest index number that has enough idle resources. It is assumed that a job will not be assigned resources from multiple server pools to keep communication overhead low. Thus a job will be served by the pool $n + 1$ with enough idle resources if pool n does not have enough idle resources. As before the total job arrival process at the system will be according to a Poisson process. The first pool of servers will see the total job arrival process while any other pool of servers will see the overflow traffic from the preceding pool. We assume that the overflow processes are Poisson which is an approximation to be verified by simulation. Within a pool, if possible, a job will be placed in a single server otherwise it will be fragmented. Thus VMs of each pool may be considered as a completely shared resource without the need to make a distribution among its servers. Let us define,

λ_{rn}	arrival rate of class r jobs to the nth server pool.
λ_{Tn}	total arrival rate of the jobs to the nth server pool.
PB_{rn}	probability that a class r job will be blocked by the nth server pool.
PB_n	overall job blocking probability at the nth server pool.
g	number of active server pools.
g_n	$Pr(g = n)$

Then, we have the following,

$$\lambda_{rn} = \lambda_{r(n-1)} PB_{r(n-1)} = \lambda_{r1} \prod_{i=1}^{n-1} PB_{ri}, \quad n \geq 2.$$

$$\lambda_{Tn} = \sum_{r=1}^{R} \lambda_{rn}$$

where $\lambda_{T1} = \lambda_T$, $\lambda_{r1} = \lambda_r$

$$PB_n = \frac{1}{\lambda_{Tn}} \sum_{r=1}^{R} \lambda_{rn} PB_{rn} \tag{14}$$

Assuming that each pool server has M servers with S VMs per server, then, $q_n(j)$ will be determined by (2) with finite

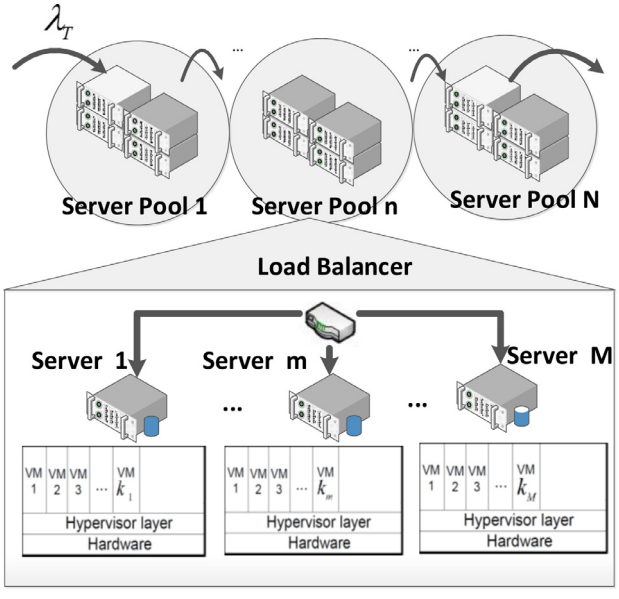

Fig. 4. Topology of the cloud computing center.

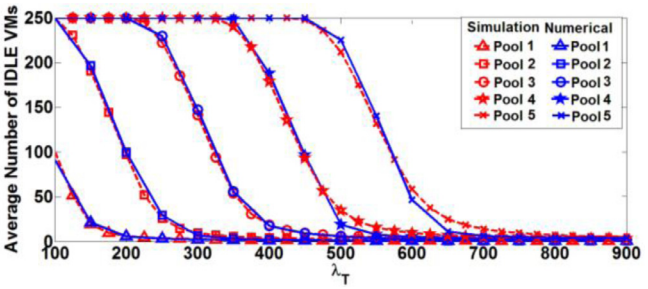

Fig. 5. Numerical and simulation results for the average number of idle VMs of different server pools as a function of total job arrival rate.

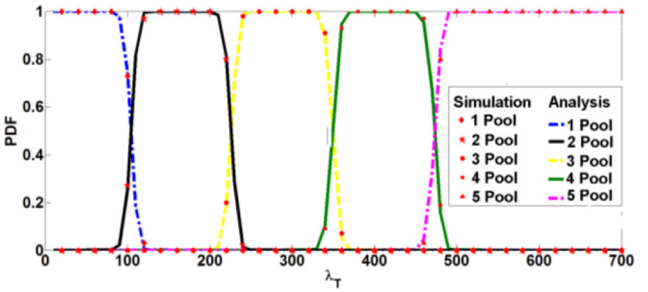

Fig. 6. Numerical and simulation results for probability distributions of number of active server pools as a function of total job arrival rate.

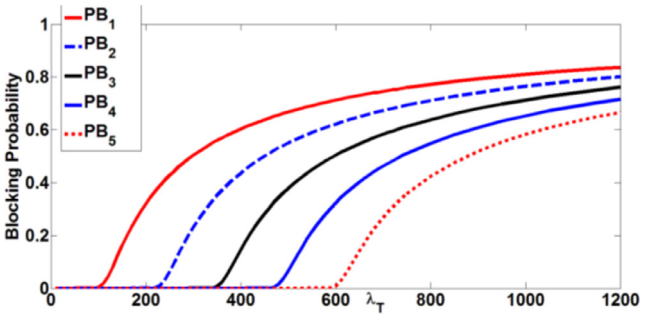

Fig. 7. Job blocking probabilities of server pools as a function of total job arrival rate.

resources of *MS* and overflow traffic from the pool $(n-1)$ as job arrival process. Then,

$$g_n = \prod_{i=n+1}^{N} q_i(0) \tag{15}$$

$$\vec{g} = [g_0, \ldots, \quad g_n, \ldots, \quad g_N]$$

We have tested the accuracy of the Poisson approximation of the overflow processes in the analysis through discrete event based simulation. In simulation also arrival of the jobs is according to a Poisson process and job service times are exponentially distributed. We assumed four job classes defined in (12,13) with $N = 5$ server pools, $M = 5$ servers/pool and $S = 50$ VMs/server. In Figs. 5 and 6, we have plotted numerical and simulation results for the average number of idle VMs and the probability distribution of the number of active server pools in the system as a function of the total job arrival rate respectively. As may be seen, there is a close agreement between numerical and simulation results, which justifies Poisson assumption of overflow processes. From Fig. 6, it is seen that at any arrival rate with probability one there will be only single number of active server pools except in the narrow transition regions. This plot shows that system operation does not result in frequent on–off switching of the server pools if the job arrival rate is not time-varying. Fig. 7 presents overall job blocking probabilities of the server pools as a function of the total job arrival rate. As may be seen, job blocking probabilities of server pools drop with the increasing index value with PB_5 giving the overall job blocking probability of the system. The results in this figure may be used to determine number of needed active server pools to support a given traffic load at an acceptable level of job blocking probability.

Next, we will assume that the total job arrival rate to the system is time-varying. It will be assumed that job arrival rate will be changing according to a discrete-time Markov chain. The time-axis will be slotted with slot durations equaling to server set-up time. We will let number of active servers to denote state of the system with the state of the system changing at the discrete-times. There will be set-up times for turning an off machine to on, while turning an on machine off will be instantaneous. As may be seen from the previous

results, the domain of the total arrival rate may be divided into intervals during which number of active server pools has a non-zero probability only for a single value during an interval. Let λ'_{Tn} denote the total arrival rate at the midpoint of the interval for $g_n = 1$. In calculation of job blocking probabilities during the transition from state i to state j, where $j > i$,we will assume that the total job arrival rate is given by λ'_{Tj}.

Letting p_{ij} denote the transition probability from state i to state j and P the corresponding transition probability matrix, then the steady-state probability distribution of the number of active server pools is determined by,

$$\vec{g} = \vec{g}P \tag{16}$$

Defining $\bar{g}$ as average utilization of the server pools in the system,

$$\bar{g} = \frac{1}{N}\sum_{i=0}^{N} i g_i \tag{17}$$

Given the rising cost of energy, with the growing scale of cloud computing datacenters, the expenditure on enterprise power usage and server cooling prevents facility owners to keep all server pools active. On the other hand, switching a server pool on requires setup time, which can adversely affect system performance in terms of job blocking rate. Hence, we consider a dynamic power management approach similar to that in [11] aiming to reduce power wastage while keeping job blocking probabilities and consequently loss of revenue at an acceptable level. In the following we consider four schemes, which will be referred to as always-on, reactive, proactive and optimal prediction and compare their performances. In the always-on case, there is no power management and all the idle server pools remain on. In the reactive case, idle server pools are turned off and they are turned on according to the demand. This scheme includes set-up times during which job losses occur. Reactive scheme responds to load increases with the time lag of one slot. In proactive case, an additional pool is kept in idle state to meet any load increases. The optimal prediction scheme predicts the job arrival rate for the next slot and turns on enough number of off servers to meet the demand.

Let k_p denote the cost of per unit power consumption (standard fee per watt) and k_r denote per hour rental rate of a VM. Also, p_{on} and p_{idle} denote the average power usage of a VM in active and idle states respectively. Next, we determine the net cost of transition (NC) to a higher state per slot for each of the four schemes, which is the difference between revenue and cost of power consumption. In the following equations, earned and lost revenue has negative and positive signs respectively.

$$NC_{\text{always-on}} = \bar{\zeta}\sum_{i=0}^{N-1} g_i \left\{ k_p (N-i) MS p_{\text{idle}} - \sum_{j=i+1}^{N} p_{ij} \sum_{r=1}^{R} \left(k_r r \lambda'_{rj} PB_{ri} \frac{1}{\mu_r} \right) \right\} \tag{18}$$

$$NC_{\text{reactive}} = \bar{\zeta}\sum_{i=0}^{N-1} g_i \left\{ \sum_{j=i+1}^{N} p_{ij} \left[k_p (j-i) MS p_{\text{on}} + \sum_{r=1}^{R} \left(k_r r \lambda'_{rj} PB_{ri} \frac{1}{\mu_r} \right) \right] \right\} \tag{19}$$

$$NC_{\text{proactive}} = \zeta \sum_{i=0}^{N-1} g_i \left\{ k_p MS p_{\text{on}} + \sum_{j=i+1}^{N} p_{ij} \sum_{r=1}^{R} \left(k_r r \lambda'_{rj} PB_{r(i+1)} \frac{1}{\mu_r} \right) - \sum_{j=i+1}^{N} p_{ij} \sum_{r=1}^{R} \left[k_r r \lambda'_{rj} (PB_{ri} - PB_{r(i+1)}) \frac{1}{\mu_r} \right] \right\} \tag{20}$$

$$NC_{\text{Optimal prediction}} = \bar{\zeta}\sum_{i=0}^{N-1} g_i \left\{ \sum_{j=i+1}^{N} p_{ij} \left[k_p (j-i) MS p_{\text{on}} - \sum_{r=1}^{R} \left(k_r r \lambda'_{rj} PB_{ri} \frac{1}{\mu_r} \right) \right] \right\} \tag{21}$$

In the above, the terms with k_p and k_r correspond to cost and revenue items respectively. Clearly, the scheme with the most negative net cost value will be performing better than the others. We need to know transition probabilities of the imbedded Markov chain for calculation of the net cost of the transitions. In practice, these values will be determined from the measurements, however, next we illustrate the utilization of our results through an example. We assumed the same job classes that have been defined in (12) with the additional parameter values given below,

k_p	$0.055 \frac{\$}{\text{kWh}}$ (HydroQuebec rate)
k_r	$0.085 \frac{\$}{\text{h}}$ (Microsof Azure Small VM)
N	$5,\ M = 5,\quad S = 50,\quad R = 4$
p_{on}	$405w,\quad p_{\text{idle}} = 225w,$ (Intel Atom Centerton 1.6 GHz CPU)
$\bar{\zeta}$	300 s

where p_{on} is the required power to turn a CPU on. Next we assume that the transition probabilities for the discrete-time Markov chain are given by,

$$p_{ij} = \begin{cases} {\gamma_i}^{i-j}, & 0 \le j < i \le N \\ \alpha_i, & j = i \\ {\beta_i}^{j-i}, & 0 \le i < j \le N \end{cases} \tag{22}$$

where α_i, β_i and γ_i are state dependent parameters. As may be seen the transition probability between states i and j is given by a power of β_i or γ_i where the power is determined by the distance between the two states. Thus probability of transition between two states decreases with the increasing distance between them. Next, we will relate state dependent parameters α_i, β_i to each other. It has been found that average utilization of a cloud computing center is presently about 30%, $\bar{g} = 0.3$ [19]. As a result, the system will spend more time in state 1 than the other states. We will designate state 1 as the base state and express all the α_i, β_i as a function of α_1, β_1 respectively. Next we assumed that $\beta_i = \tau^{|1-i|}\beta_1, \quad \alpha_i = \sigma^{|1-i|}\alpha_1$ where σ, τ are proportionality constants, $0 \le \sigma, \tau \le 1$. We note that γ_i is determined from the normalization condition of the transition probabilities of each state. High value of α_1 (low values of β_1, γ_1) indicates

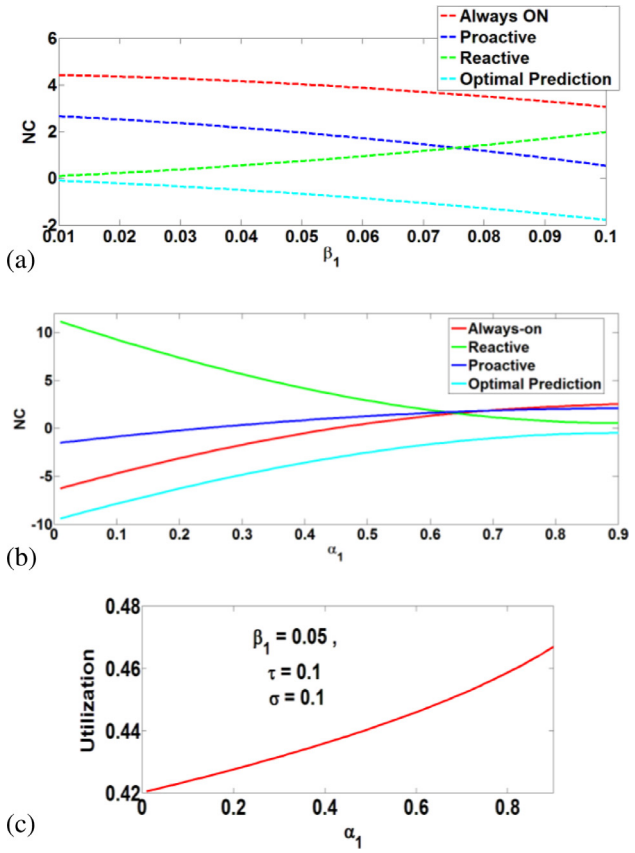

Fig. 8. (a) Net cost of a transition for *always-on, reactive, proactive* and *optimal prediction* schemes as a function of β_1 for $\alpha_1 = 0.88, \sigma = \tau = 0.1$. (b) Net cost of a transition for *always-on*, *reactive*, *proactive* and *optimal prediction* schemes as a function of α_1 for $\beta_1 = 0.05, \sigma = \tau = 0.1$. (c) Utilization as a function of α_1.

a system with slowly varying job arrival rate, on the other hand low value of α_1 (higher values of β_1, γ_1) indicates a system with fast varying job arrival rate, the latter being a more dynamic system.

In Figs. 8a and 8b, we present plots of NC for the four schemes as a function of β_1 and α_1 respectively. As expected, in both cases, optimal prediction gives the best performance as its net cost has the most negative value. In Fig. 8a, reactive scheme always performs better than always-on and most of the time better than proactive scheme because the system spends a lot of time in state 1 due to high value of α_1. In Fig. 8 b, the system is more dynamic for low values of α_1 compared to its high values. Since reactive scheme's response has a lag time, it gives the worst performance for $\alpha_1 < 0.65$. It may be seen that the performance of various schemes depend on degree of time-variation of the traffic load. Fig. 8c shows the utilization of the system as a function of parameter α_1 with the other parameters fixed. As may be seen, utilization increases with increasing value of α_1.

Findings in this section may give insight to the selection of appropriate system operation policy, i.e. proactive to reactive or vice versa. For example, in a static scenario (large values of α_1) reactive approach is good enough while for more dynamic systems the proactive approach gives better performance.

4. Modeling of a system with heterogeneous VMs, constant job size and simultaneous release times

In this section, we extend the results of the previous section to a single server with heterogeneous types of VMs. The VM types may differ from each other in the amount of resources allocated to a VM, such as in number of CPUs, memory and storage sizes. We assume that there are L types of VMs and a job may request up to J VMs of a single type. The type and number of VMs requested will define class of a job. Thus a class $j\ell$ job will request j VMs of type ℓ, $j = 1 \dots J, \quad \ell = 1 \dots L$. Let us introduce the following notation,

F	number of resource types.
C_f	number of units of resource f, $f = 1 \dots F$.
$b_{\ell f}$	number of units of resource f required by a type ℓ VM, $\ell = 1 \dots L, \quad f = 1 \dots F$.
$\lambda_{j\ell}$	arrival rate of class $j\ell$ jobs that require j number of type ℓ VMs, $j = 1 \dots J, \quad \ell = 1 \dots L$.
$\mu_{j\ell}$	service rate of class $j\ell$ jobs.
$n_{j\ell}$	number of class $j\ell$ jobs in the system.
$\vec{b}_\ell$	$(b_{\ell 1}, \dots b_{\ell f}, \dots, b_{\ell F})$
$\vec{C}$	$(C_1, \dots C_f, \dots, C_F)$
$\vec{n}$	$(n_{11}, \dots, n_{j1} \dots, n_{J1}, \dots, n_{1\ell}, \dots, \quad n_{j\ell}, \dots n_{J\ell}, \dots, \dots n_{1L}, \dots, \quad n_{jL}, \dots, n_{JL})$

$$\vec{n}_{\bar{j\ell}} \qquad (n_{11},\ldots,n_{j1}\ldots,n_{J1},\ldots,n_{1\ell}-1,\ldots n_{J\ell},\ldots n_{1L},\ldots,\quad n_{jL},\ldots,n_{JL})$$

Total arrival rate of the jobs is given by,

$$\lambda_T = \sum_{j=1}^{J}\sum_{\ell=1}^{L}\lambda_{j\ell}$$

Defining B as the resource matrix of VM types,

$$B = \begin{bmatrix} b_{11} & \cdots\ b_{1f}\ \cdots & b_{1F} \\ \vdots & b_{\ell f}\ddots & \vdots \\ b_{L1} & \cdots\ b_{Lf}\ \cdots & b_{LF} \end{bmatrix}$$

Next defining N and Λ as matrices of the number of each class of jobs and their arrival rates respectively,

$$N = \begin{bmatrix} n_{11} & \cdots\ n_{1\ell}\ \cdots & n_{1L} \\ \vdots & n_{j\ell}\ddots & \vdots \\ n_{J1} & \cdots\ n_{J\ell}\ \cdots & n_{JL} \end{bmatrix} \tag{23}$$

$$\Lambda = \begin{bmatrix} \lambda_{11} & \cdots\ \lambda_{1\ell}\ \cdots & \lambda_{1L} \\ \vdots & \lambda_{j\ell}\ddots & \vdots \\ \lambda_{J1} & \cdots\ \lambda_{J\ell}\ \cdots & \lambda_{JL} \end{bmatrix}$$

As before, we assume that the distribution of the service time of each class of jobs has a rational Laplace transform.

We note that this model is an extension of blocking in shared resources environment studied in [17] to a system with multiple types of resources. Following the analysis in [17], we will determine joint probability distribution of the number of jobs in the system and derive a multi-dimensional recursion for the distribution of the utilization of resources. First, we will write the local balance equation (LBE) of this system. An LBE equates the flow due to a departure of a job from a network state to the flow due to an arrival of a job to a network that will return the system to the same state, thus,

$$n_{j\ell}\mu_{j\ell}p(\vec{n}) = \lambda_{j\ell}p\left(\vec{n}_{\bar{j\ell}}\right) \tag{24}$$

Let us assume the following joint probability distribution of the number of different classes of the jobs in the system,

$$p(\vec{n}) = \frac{1}{G}\prod_{j=1}^{J}\prod_{\ell=1}^{L}\frac{{\rho_{j\ell}}^{n_{j\ell}}}{n_{j\ell}!} \tag{25}$$

where G is the normalization constant and $\rho_{j\ell} = \frac{\lambda_{j\ell}}{\mu_{j\ell}}$.

It may be shown by substitution that (25) satisfies (24). Since $p(\vec{n})$ satisfies the LBE, it also satisfies the global balance equations (GBEs), and therefore (25) is the correct distribution.

Again, it may be seen that joint distribution of the number of jobs depends on service time of a job through its mean. Let us define,

u_f number of units of resource f that is busy.

$$\vec{u} = \left(u_1,\ldots,u_f,\ldots,u_F\right) \tag{26}$$

Let $q(\vec{u})$ denote joint probability distribution of the utilization (number of busy units) of different type of resources. In the appendix, we derive the following multi-dimensional recursion for determining this distribution,

$$u_f q(\vec{u}) = \sum_{l=1}^{L}\sum_{j=1}^{J} jb_{\ell f}\rho_{j\ell}q\left(\vec{u}-j\vec{b}_{\ell}\right) \tag{27}$$

Then, the average utilization vector is given by,

$$E(\vec{u}) = \sum_{\vec{u}|\ (\forall f\ \in\ F,\quad u_f \le C_f)} \vec{u}q(\vec{u}) \tag{28}$$

The probability that demand for a type ℓ VM will be blocked is given by,

$$PB_{\ell} = \sum_{\vec{u}|\ (\forall f\ \in F,\quad u_r+b_{lf}>C_f)} q(\vec{u}) \tag{29}$$

Next we will give an example based on a system with three VM types given in Table 2 with the following resource vector,

$$\vec{C} = (160\ GB, 200\ Core, 10000\ \ GB) \tag{30}$$

From Table 2., resource matrix of VM types is given by,

$$B = \begin{bmatrix} 2 & 2 & 100 \\ 16 & 6 & 400 \\ 8 & 8 & 200 \end{bmatrix} \tag{31}$$

Assuming the following arrival rate matrix for classes of jobs with ($J = 4$),

$$\Lambda = \begin{bmatrix} 0.2 & 0.1 & 0.1 \\ 0.15 & 0.075 & 0.075 \\ 0.1 & 0.05 & 0.05 \\ 0.05 & 0.025 & 0.025 \end{bmatrix}\lambda_T \tag{32}$$

It should be noted that in the above job classes with higher resource requirements have lower arrival rates. Figs. 9, 10 and 11 show the cumulative probability distributions of memory, CPU and storage utilizations respectively with the total job arrival rate as a parameter. These results may be used to determine bottleneck resources and redundancy in the system. It may be seen that at the total job arrival rate

Table 2
Representative VMs specifications.

VM type	Memory	CPU cores	Storage
Standard	2(GB)	2	100 (GB)
High memory extra large	16(GB)	6	400 (GB)
High CPU extra large	8(GB)	10	200 (GB)

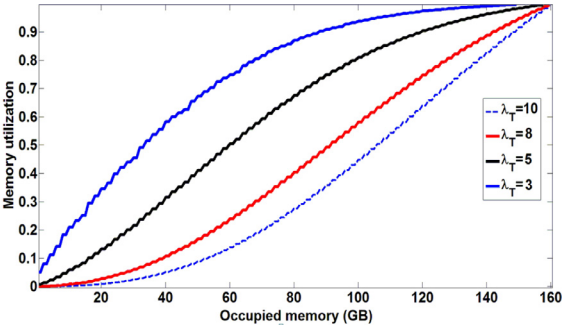

Fig. 9. Cumulative distribution of memory utilization with λ_T as a parameter.

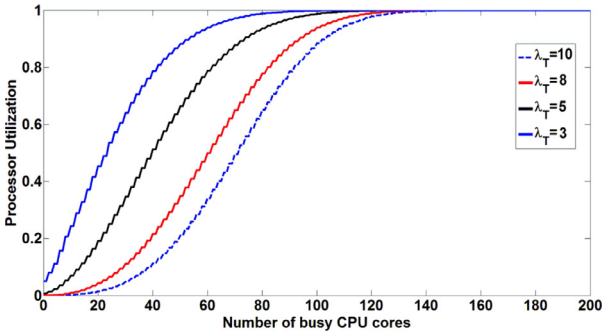

Fig. 10. Cumulative distribution of CPU utilizatilization with λ_T as a parameter.

of 10, the values of memory, CPU and storage corresponding to cumulative probabilities of unity are 160 GB, 130 cores and 4500 GB respectively. Since at this arrival rate all the available memory may be busy, the system cannot support a higher traffic load. As a result, the number of cores beyond 130 and storage beyond 4500 GB will not be utilized and they will be redundant.

Fig. 12 shows the blocking probabilities of the requests for different types of VMs as a function of the total job arrival rate. As may be seen, VMs differ in their blocking probabilities pertaining to their resource requirements.

5. Modeling of the system with constant job size, homogeneous VMs and independent release times

In this section, as in Section 3, we assume constant job sizes with multiple classes as defined in Table 1. This model differs from the model of that section in the service given to the tasks. It is assumed that service times of the tasks of a job are i.i.d with exponential distribution with parameter μ, which results in the independent as opposed to simultaneous task completion times. We assume finite resources with S VMs and model the system with birth–death processes.

Let p_j denote probability that there will be j tasks in the system, then GBE of the system may be written as,

$$\begin{cases} \left(\sum_{r=1}^{R}\lambda_r + j\mu + j\mu\right)p_j = (j+1)\mu p_{j+1}\sum_{r=1}^{R} p_{j-b_r}\lambda_r, \\ \quad 0 < j < S \\ \sum_{r=1}^{R}\lambda_r p_0 = \mu p_1, \quad j = 0 \\ S\mu p = \sum_{r=1}^{R} p_{s-b_r}\lambda_r, \quad j = S \end{cases} \tag{33}$$

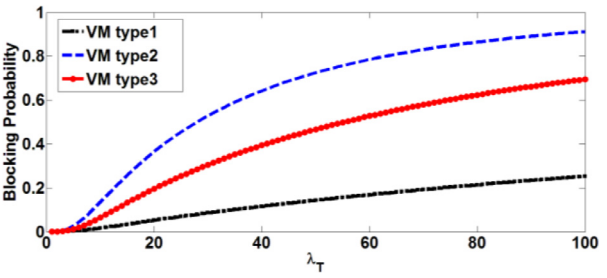

Fig. 12. Blocking probabilities of different types of VMs as a function of job arrival rate λ_T.

The above equations cannot be solved through the transform analysis, but the distribution of the number of busy VMs may be determined from the above recursive equations together with the normalization condition. Then average of the total number of the busy VMs is given by,

$$E[k_T] = \sum_{j=0}^{S} jp_j$$

Let P_{B_r} denote the blocking probability of class r jobs, then it is given by,

$$P_{B_r} = \sum_{j=S-r+1}^{S} p_j$$

Next we will determine pdf of the service time of a class r job. Let T_r and $f_{T_r}(t)$ be this service time and its pdf respectively. Then,

$$T_r = \max\left(t_1, t_2, \ldots, t_j, \ldots, t_r\right)$$

where t_j is the service time of the jth task. Since service times of the tasks are i.i.d. with exponential distribution,

$$Pr(T_r < t) = \prod_{j=1}^{r} P\left(t_j < t\right)$$

From the above, the pdf of T_r is given by,

$$f_{T_r}(t) = r\mu e^{-\mu t}\left(1 - e^{-\mu t}\right)^{r-1}$$

The average service time of a class r job is given by,

$$\overline{T_r} = \frac{1}{\mu}\sum_{i=1}^{r}\frac{\binom{r}{i}}{i}(-1)^{i+1} \tag{34}$$

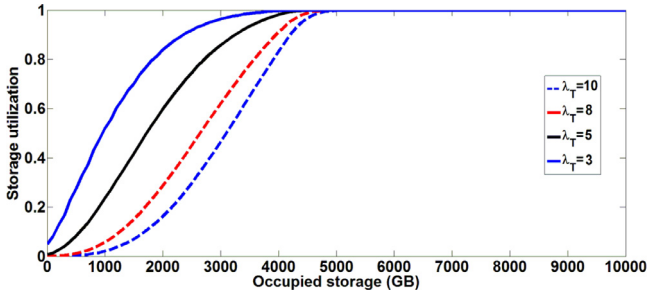

Fig. 11. Cumulative distribution of storage utilization with λ_T as a parameter.

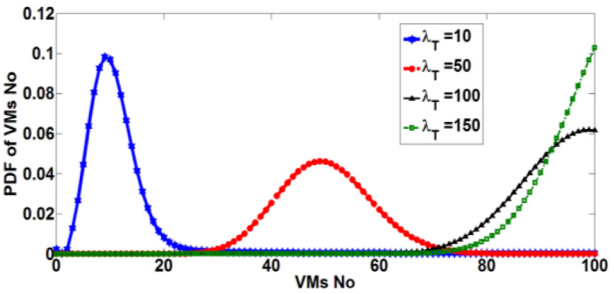

Fig. 13. Distribution of busy VMs under low, medium, heavy and very heavy load ($R = 4, S = 100, \mu = 1$).

Let n_r denote number of class r jobs in the system, then from the Little's result its average is given by,

$$E[n_r] = \lambda_r(1 - P_{B_r})\overline{T_r} \tag{35}$$

Fig. 13 presents probability distribution of the number of busy VMs for a system with four classes of jobs with equal arrival rates with total arrival rate as a parameter for a fixed number of VMs in the system. As may be seen, probability distribution shifts to the right with increasing total arrival rate. Further, the distribution has the largest spread at the medium job arrival rate. Fig. 14 presents the average number of jobs from each class in the system as a function of the total arrival rate. It may be observed that average of the number of class 4 jobs in the system decreases faster than the other classes with increasing total arrival rate.

6. Modeling of the system with dynamic service demand

In this section, we propose a performance model for systems with dynamic service demand where job size in number of tasks varies during service. As explained in the introduction, this model will be more appropriate to mobile cloud computing systems. We assume that the size of a job in number of tasks varies randomly during the time that job is in the system. The arrival of the jobs to the system will be according to a Poisson process with parameter λ jobs/sec. We assume that a new arriving job to the system initially demands service for a single task. A job generates random number of tasks according to a Poisson process with parameter α α task/job/sec during its service time in the system. We assume that each task requires a VM for its execution and task execution times are exponentially distributed with parameter μ. Service time of a job begins with its arrival to the system and it is completed when there are no more tasks belonging to that job left in the system. Clearly, a job will have a general service type distribution. In this section, a birth–death process is proposed to model this type of cloud computing systems. Fig. 15 indicates the state transition diagram for the tasks of a job in the system. The objective of this analysis is to determine distribution of the number of jobs in the system, service time distribution of a job and average of the total number of tasks. We will consider systems with both infinite and finite number of VMs.

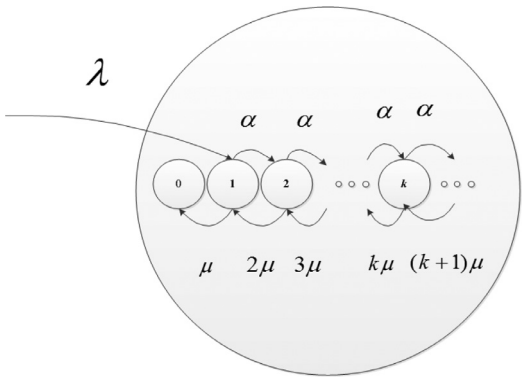

Fig. 15. State-transition-rate diagram for the tasks of a job in the system.

6.1. Infinite resource model

First, we consider infinite resource model where there is always an idle VM available for the execution of each newly generated task to begin immediately. In this case the number of jobs in the system can be modeled as an $M/G/\infty$ queuing system. Next, we will determine main performance measures of this system.

6.1.1. Distribution of the number of jobs in the system

Let p_n denote the steady state probability of having n jobs in the system and $N(z)$ its probability generating function (PGF). From the results for the $M/G/\infty$ queuing system [20],

$$p_n = \frac{(\lambda\bar{x})^n}{n!}e^{\lambda\bar{x}} \tag{36}$$

$$N(z) = e^{-\lambda\bar{x}(1-z)} \tag{37}$$

where $\bar{x}$ denotes the average service time of a job which is determined below.

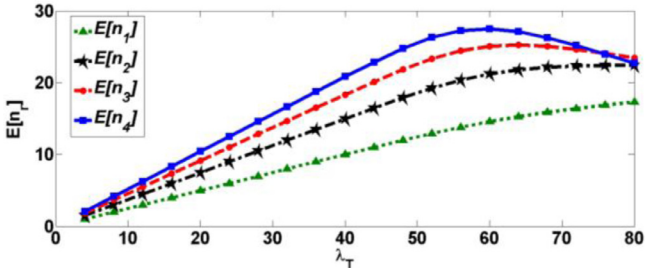

Fig. 14. Average number of jobs from each class as a function of the total job arrival rate ($R = 4, S = 100, \mu = 1$).

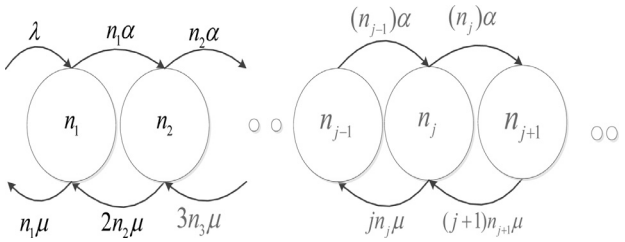

Fig. 16. State-transition diagram for the stages of the system.

As stated above, each job initially requires service for a single task; however, it generates new tasks according to a Poisson process during its service time in the system. Since we have assumed infinite resource model, each newly generated task immediately begins to receive service. Since task execution times are also exponentially distributed, service time of a job corresponds to the busy period of an $M/M/\infty$ queue, where the number of customers served during the busy period corresponds to the total number of tasks generated by the job. Fig. 16 shows the state-transition-rate diagram for the tasks of a job in the system. From [21], Laplace transform of the probability distribution of the busy period of an $M/M/\infty$ queue with arrival and service rates of α and μ is given by,

$$B(s) = 1 + \alpha^{-1}(s - (\int^{e^{-st-\alpha\int_0^t(1-G(v))dv}})) \tag{38}$$

where $G(v)$ denotes the service time distribution of a task in the system, which has exponential distribution.

Then average service time of a job is given by the mean busy period of $M/M/\infty$ queue,

$$\bar{x} = \frac{e^{\alpha/\mu} - 1}{\alpha} \tag{39}$$

From the Little's result the average number of jobs in the system is given by:

$$E[n] = \lambda\bar{x} \tag{40}$$

6.1.2. Average number of tasks generated by a job during its lifetime in the system

Next, we determine average of the total number of tasks generated by a job during its life-time in the system, which is given by the ratio of average service time of a job to the service rate seen by its tasks in the system. Thus, first, we will determine the service rate seen by the tasks of a job.

Let q_k denote probability that there will be k customers in an $M/M/\infty$ queuing system at the steady-state. From [20], q_k has Poisson distribution given by,

$$q_k = \frac{(\alpha/\mu)^k}{k!} e^{-\alpha/\mu}, \quad k \geq 0 \tag{41}$$

Letting q'_k denote probability that there will be k customers at an arbitrary time during a busy period in an $M/M/\infty$ queuing system, then:

$$q'_k = \frac{q_k}{1 - q_0}, k \geq 1 \tag{42}$$

Let μ_k denote service rate of the tasks of a job, which has k tasks in the system at an arbitrary time. Since $\mu_k = k\mu$, the average service rate of the tasks generated by a job is given by

$$\bar{\mu} = \mu \sum_{k=1}^{\infty} k q'_k = \frac{\alpha}{1 - e^{\frac{-\alpha}{\mu}}} \tag{43}$$

Defining $\bar{r}$ as the average number of tasks generated by a job during its service time in the system, then it is given by,

$$\bar{r} = \frac{\bar{x}}{\bar{\mu}} = e^{\frac{\alpha}{\mu}} \frac{\left(1 - e^{\frac{-\alpha}{\mu}}\right)^2}{\alpha^2} \tag{44}$$

6.1.3. Joint distribution of the number of jobs in each stage of the system

We define a job to be in stage j if it has j tasks in execution at that time within the system. Let n_j denote number of jobs in stage j at an arbitrary time. Fig. 16 shows the state-transition rate diagram for stages of the system. Next, we will determine joint distribution of the number of jobs in each stage of the system.

Proposition 1. *n_j has a Poisson distribution.*

Proof. Let us define Bernoulli random variable k_{ij} as,

$$k_{ij} = \begin{cases} 1 & i^{\text{th}} \text{ job has } j \text{ tasks in the system} \\ 0 & \text{otherwise} \end{cases} \tag{45}$$

Then, PGF of the distribution of k_{ij} is given by,

$$K_{ij}(z) = q'_j z + 1 - q'_j \tag{46}$$

From the above, n_j may be expressed as,

$$n_j = \sum_{i=1}^{n} k_{ij} \tag{47}$$

Let $N_j(z)$ denote PGF of the probability distribution of n_j, then,

$$N_j(z) = N(z)|_{z=K_{ij}(z)} = e^{-\lambda\bar{x}q'_j(1-z)} \tag{48}$$

where we substituted from (37) and (46) in the above. The inversion of the above PGF gives,

$$p_{n_j} = \frac{\left(\lambda\bar{x}q_j\right)^{n_j}}{n_j!} e^{-\lambda\bar{x}q_j} \tag{49}$$

which completes the proof.

Now, we will determine the joint distribution of the number of jobs at each stage of the system. Let state of the system denoted by the vector $\vec{n} = (n_1, \ldots, n_i, \ldots n_\infty)$. We will show

that the joint probability distribution of $\vec{n}$ has a Poisson distribution given by,

$$p(\vec{n}) = \prod_{j=1}^{\infty} \left[\frac{\left(\lambda \bar{x} q'_j\right)^{n_j}}{n_j!} e^{-\lambda \bar{x} q'_j} \right] \tag{50}$$

Let us define the following vectors that differ from $\vec{n}$ at most in two components:

$$\begin{aligned}
\vec{n}_j^{+} &= (n_1, \ldots, n_j^{+}, \ldots n_\infty) \\
\vec{n}_j^{-} &= (n_1, \ldots, n_j^{-} \ldots n_\infty) \\
\vec{n}_{ij}^{+-} &= (n_1, \ldots, n_i^{+}, \ldots, n_j^{-}, \ldots n_\infty) \\
\vec{n}_{ij}^{-+} &= (n_1, \ldots, n_i^{-}, \ldots, n_j^{+}, \ldots n_\infty)
\end{aligned} \tag{51}$$

where $n_j^{+} = n_j + 1, \quad n_j^{-} = n_j - 1$.

Next, we will write the LBEs for the state $\vec{n}$,

$$\begin{cases}
j n_j \mu p(\vec{n}) + n_j \alpha p(\vec{n}) = (j+1)(n_{j+1}+1)\mu \; p(\vec{n}_{j,j+1}^{-+}) \\
\quad + (n_{j-1}+1)\alpha p(\vec{n}_{j-1,j}^{+-}), \quad j > 1 \\
n_1 \mu p(\vec{n}) + n_1 \alpha p(\vec{n}) = 2(n_2+1)\mu \; p(\vec{n}_{12}^{-+}) + \lambda \mathrm{p}(\vec{n}_1^{-}), \\
\quad j = 1
\end{cases} \tag{52}$$

By means of substitution it can be shown that (50) satisfies the LBEs in (52) and therefore it is the correct distribution.

6.1.4. Distribution of the total number of tasks in the system

Next, we will determine distribution of the total number of tasks in the system. Let us introduce the following notation,

$$\begin{aligned}
r_j &= j n_j \\
\vec{r} &= (r_1, \ldots, r_j, \ldots, r_\infty) \\
\vec{z} &= (z_1, \ldots, z_j, \ldots, z_\infty) \\
\vec{n} &= (n_1, \ldots, n_j, \ldots, n_\infty)
\end{aligned}$$

where, r_j corresponds to the total number of tasks that belong to the jobs in stage j. Let us define PGF of the distribution of $\vec{r}$ as,

$$\begin{aligned}
R(\vec{z}) &= E[\vec{z}^{\vec{r}}] = E\left[\prod_{j=1}^{\infty} z_j^{r_j}\right] = E\left[\prod_{j=1}^{\infty} z_j^{j n_j}\right] = E\left[\prod_{j=1}^{\infty} \left(z_j^{j}\right)^{n_j}\right] \\
R(\vec{z}) &= E\left[\prod_{j=1}^{\infty} \left(z_j^{j}\right)^{n_j}\right] \\
R(\vec{z}) &= \sum_{n_1=0}^{\infty} \cdots \sum_{n_j=0}^{\infty} \cdots \sum_{n_\infty=0}^{\infty} \left[\prod_{j=1}^{\infty} \left(z_j^{j}\right)^{n_j}\right] p(\vec{n})
\end{aligned} \tag{53}$$

Substituting for $p(\vec{n})$ from (50),

$$R(\vec{z}) = \sum_{n_1=0}^{\infty} \cdots \sum_{n_j=0}^{\infty} \cdots \sum_{n_\infty=0}^{\infty} \left[\prod_{j=1}^{\infty} e^{-\lambda \bar{x} q'_j} \frac{\left(\lambda \bar{x} q'_j z_j{}^{j}\right)^{n_j}}{n_j!}\right]$$

Interchanging the order of summations and multiplications,

$$R(\vec{z}) = \prod_{j=1}^{\infty} e^{-\lambda \bar{x} q'_j} e^{\lambda \bar{x} q'_j z_j^{j}} = \prod_{j=1}^{\infty} e^{-\lambda \bar{x} q'_j (1 - z_j^{j})}$$

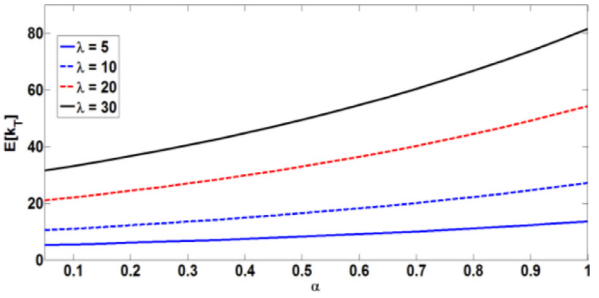

Fig. 17. Average of the total number of the tasks as a function of α and λ as a parameter.

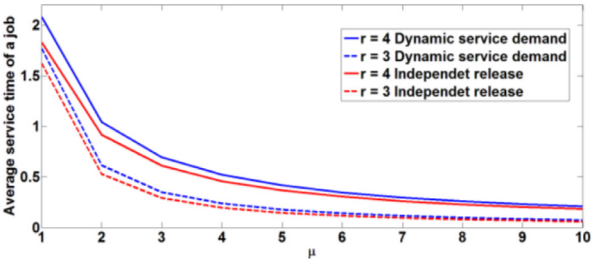

Fig. 18. Average service time of a job as a function of task service rate for dynamic service and independent release time models.

$$R(\vec{z}) = e^{-\lambda \bar{x} \sum_{j=1}^{\infty} q'_j \left(1 - z_j^{j}\right)} = e^{-\lambda \bar{x} \left(1 - \sum_{j=1}^{\infty} q'_j z_j^{j}\right)} \tag{54}$$

Next let us define k_T as the total number of tasks in the system and $K_T(z)$ as the PGF of its distribution, then,

$$k_T = \sum_{j=1}^{\infty} r_j$$

$$K_T(z) = E[z^{k_T}] = R(\vec{z})|_{z_i = z, i = 1, \ldots, \infty} = e^{-\lambda \bar{x} \left(1 - \sum_{j=1}^{\infty} q'_j z^{j}\right)} \tag{55}$$

Substituting in the above from (36), (39) gives,

$$K_T(z) = e^{-\lambda \bar{x} \left[1 - \frac{e^{\frac{-\alpha(1-z)}{\mu}} - e^{\frac{-\alpha}{\mu}}}{1 - e^{\frac{-\alpha}{\mu}}}\right]} \tag{56}$$

Finally, from the above average of the total number of tasks in the system are given by,

$$E[k_T] = \frac{\lambda \alpha \bar{x}}{\mu\left(1 - e^{\frac{-\alpha}{\mu}}\right)} = \frac{\lambda}{\mu} e^{\frac{\alpha}{\mu}} \tag{57}$$

Fig. 17 presents average of the total number of the tasks in the system as a function of the task arrival rate with job arrival rate as a parameter. Fig. 18 presents the average service time of a job with dynamic service time and the independent release time of the previous section from (39) and (34) respectively. We plotted the results for class 3 and 4 jobs for the independent release times. For fair comparison, average of the number of tasks generated by a job with dynamic service time, (44), has been set equal to the number of tasks in each class of jobs for the independent release time. Thus for each value of $\boldsymbol{\mu}$, task generation parameter $\boldsymbol{\alpha}$ has been chosen such that $\bar{\boldsymbol{r}} = \boldsymbol{r}$. As may be seen, under these assumptions the average service times of a job in the two models are close to each other.

6.2. Finite resource model

Next, we consider the finite resource model where the computing center has finite number of VMs given by S. A new arriving job will be blocked if all the VMs are occupied. In this model, we assume that each job is assigned a fixed number of VMs, c, for its service. When the number of tasks belonging to a job is more than c, then the excess tasks are queued. Let us assume that S is an integral multiple of c, then the number of jobs in the system can be modeled as an $M/G/N/N$ queuing system where $N = S/c$.

The service time of a job may be modeled by the busy period of an $M/M/c$ queue, where customers are the tasks generated by the job. The average service time of a job is given by the mean busy period of the $M/M/c$ queue, which is from [22],

$$\bar{x} = \begin{cases} \dfrac{1}{\mu\left(1-\frac{\alpha}{c\mu}\right)} & \text{for } c \le 2 \\ \dfrac{1}{\alpha}\left[\dfrac{\left(\frac{\alpha}{\mu}\right)^c}{\left(1-\frac{\alpha}{c\mu}\right)c!} + \dfrac{1}{\alpha}\sum_{k=1}^{c-1}\dfrac{\left(\frac{\alpha}{\mu}\right)^k}{k!}\right] & \text{for } c > 2 \end{cases} \tag{58}$$

Let k denote the number of tasks in the system that belongs to a job, then it may be determined from the distribution of the number of customers in an $M/M/c$ queuing system, [22],

$$Pr(k=i) = \begin{cases} \dfrac{Pr(k=0)}{1-Pr(k=0)}\dfrac{\left(\frac{\alpha}{\mu}\right)^i}{i!} & 0 < i \le c \\ \dfrac{Pr(k=0)}{1-Pr(k=0)}\dfrac{\left(\frac{\alpha}{\mu}\right)^k}{c!c^{k-c}} & i > c \end{cases} \tag{59}$$

where,

$$Pr(k=0) = \left[\sum_{k=0}^{c-1}\frac{\left(\frac{\alpha}{\mu}\right)^k}{k!} + \frac{\left(\frac{\alpha}{\mu}\right)^c}{c!\left(1-\frac{\alpha}{c\mu}\right)}\right]^{-1}$$

Let y denote the number of busy VMs from those that assigned to a job, then,

$$Pr(y=i) = \begin{cases} Pr(k=i) & 0 < i < c \\ \sum_{\ell=c}^{\infty} Pr(k=\ell) & i = c \end{cases} \tag{60}$$

From the $M/G/N/N$ queuing results, probability distribution of the number of jobs in the system is given by, [23],

$$p_n = \begin{cases} p_0\dfrac{(N\rho)^n}{n!} & n < N \\ p_0\dfrac{(N\rho)^N}{N!} & n = N \end{cases} \tag{61}$$

where $p_0 = [\sum_{j=0}^{N-1}\frac{(N\rho)^j}{j!} + \frac{(N\rho)^N}{N!}]^{-1}$ and $\rho = N\lambda\bar{x}$.

We note that blocking probability of a job is given by p_N. Let k_T denote total number of tasks in the system, then its average is given by,

$$E[k_T] = E[n]E[k] \tag{62}$$

The above average needs to be determined numerically from (59) and (61).

Fig. 19 shows the average number of VMs in the system as a function of task arrival rate and job arrival rate as a parameter. We assumed that $N = 40$ and $c = 10$. As illustrated, due to hard limitation on maximum number of tasks of a job, task arrival rate is dominant in creation of VMs compared to job arrival rate. With increasing the job arrival rate, job saturation probability shifts to the left.

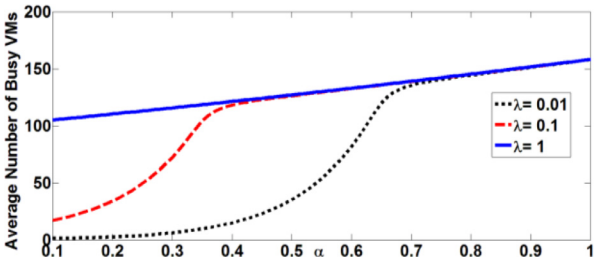

Fig. 19. Average number of the VMs as a function of task arrival rate and job arrival rate as a parameter. ($c = 10$, $N = 40$).

7. Comparison with the related work

In this section, we give a comparison of our results with the closest previous work that has been referred to in Section 2. There is an overlap between our work and that in [11,24–26], though we studied several more models not considered in those works.

In [11], a cloud computing center has been modeled as an $M/G/m/m+r$ queue, where m is the number of VMs in the system and r is the size of the buffer that stores the waiting jobs. A new arriving job to a full buffer is lost and the jobs in the buffer are served on FCFS basis. It is assumed that each job requires a single VM for its execution. The steady-state distribution of the queue length is determined by writing down the transition probability matrix of the embedded Markov chain at the arrival points. The analysis makes the approximation that at most three jobs may be served during an inter-arrival time. The equilibrium equations had to be solved numerically, thus the queue length distribution could not be obtained in a closed form. This model corresponds to our single server model with one class of jobs, when no buffering is allowed, $r = 0$. In Fig. 20, we plot average number of busy VMs for both our and their model under the assumption of no buffering, $r = 0$, as a function of the job arrival rate. The results have been plotted both for exponential and deterministic service times. As may be seen, the approximate results of [11] are very close to our exact results. Further, as our analysis shows it the results do not depend on the service time distribution.

In [24], the analysis in [11] has been extended to the jobs where each job contains random number of tasks and execution of each task demands a VM. In this model, the tasks of a waiting job are stored in the buffer with each task occupying one position. All the tasks of a job need to start execution simultaneously. If the tasks of a new arriving job cannot be served immediately and there is no enough storage in the buffer to store all the tasks, then that job is rejected. Since jobs are still served on a FCFS basis, this results in head-of-line (HOL) blocking until enough servers become available to serve the HOL job. Service times of the tasks are i.i.d with a general distribution, thus the tasks of a job have independent release times. Letting number of tasks to denote the system state, then the system has been analyzed by

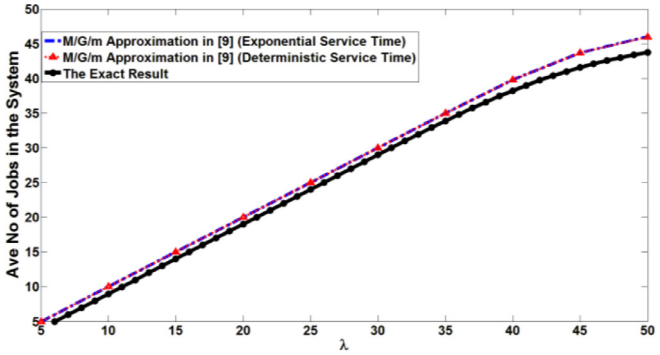

Fig. 20. Average number of the jobs in the system as a function of job arrival rate for M/G/m approximation and the exact results for $\mu = 1$.

embedding a Markov chain at the job arrival points. Similar to the original model, it is assumed that a VM cannot serve more than three tasks during a job inter-arrival time. The transition probabilities are determined assuming constant number of tasks in a job, which needs to be unconditioned numerically afterwards. Further, an important weakness of the analysis is that the probabilities involving transitions from a state with number of idle VMs require knowledge of the distribution of the idle VMs, which is part of the solution that is being determined. The distribution of the number of idle VMs had to be determined through simulation. After determination of the transition probability matrix, which is quite tedious, the equilibrium equations have been solved numerically. This model becomes identical to our model for a system with multiple classes of jobs and independent task release times under the assumptions of exponential task service times and no buffering, $r = 0$ (Section 5.). For this case, we also determined distribution of the service time of a job and average number of jobs from each class in the system (latter had been plotted in Fig. 14), which are not available in [24]. We note that our single server models apply to systems with multiple job classes and simultaneous task completion times for both homogeneous and heterogeneous VMs under no queuing assumption. The joint distribution of the number of jobs has been presented in Eqs. (1) and (25) for homogeneous and heterogeneous VMs cases respectively.

In [25], the performance of cloud computing systems has been studied using stochastic reward networks (SRNs). It is assumed that cloud center has N servers that may support up to M VMs where $N \geq M$. The arrival of the jobs is either according to a homogeneous Poisson process or a Markov Modulated Poisson Process (MMPP) which allows time variations in the arrival rate. It is assumed that each job requires a single VM for its execution and service times are exponentially distributed. However, mean service time is a function of the number of busy VMs on a server. The system has a finite queue, which is managed according to the FCFS discipline and a job arriving to a full queue is lost. The models of [25] and [11] become identical for Poisson arrivals and exponentially distributed service times with a constant mean value, when number of servers and number of VMs are equal to each other, $N = M$. For this case, the two models have been compared in [25] and the presented numerical results show very close agreement. This also means that our results agree with that of [25] for the case of single task per job scenario with no buffering, since all the three models become same for this special case. The main weakness of the model in [25] is that it is numerical and lacks closed form results.

In [26], performance of cloud computing systems has been studied considering fault recovery. It is assumed that arrival of jobs is according to a general stochastic process and each job has random number of tasks. The system has a server with S VMs and each task requires a VM for its execution. Task service times are i.i.d with exponential distribution, which results in independent task completion times. The system has a finite queue and each task of a job occupies a position in the queue. A job is lost if not all of its tasks can be accepted to the system. The system has been modeled as a $GI^X/M/S/N$ queue where N corresponds to the maximum number of allowed tasks in the system. The steady-state probability distribution of the number of tasks in the system is determined by writing down the transition probability matrix for the embedded Markov chain and solving numerically the equilibrium equations. We note that, the analysis does not result in the distribution of the number of jobs in the system. For fault modeling, it is assumed that VMs fail according to a Poisson process and VM recovery times are exponentially distributed. Following recovery, the execution of a task resumes from the point of failure. Under the approximation that all the tasks of a job begin receiving service simultaneously, job service times have been determined. However, probability distribution of the number of tasks in the system with fault tolerance could not be obtained because the queuing model only allows exponential service times. Again, this model under the assumption of Poisson arrival of jobs with no queuing and simultaneous service completion of the tasks of a job corresponds to our single server model with single class of jobs studied in Section 3.1. Simultaneous service completion means that whenever a VM assigned to a task fails, all the tasks belonging to the job as the failed task are also delayed until recovery. Then for fault tolerance scenario, our model gives the distribution of the number of jobs in the system from Eq. (1), since the analysis applies for any service time distribution. Let μ, γ denote parameters of the exponential distributions for service and recovery times respectively and α parameter of the Poisson distribution for failure. Then, mean service time of a job is given by,

$$\bar{\bar{b}} = \frac{\alpha + \gamma}{\mu\gamma}$$

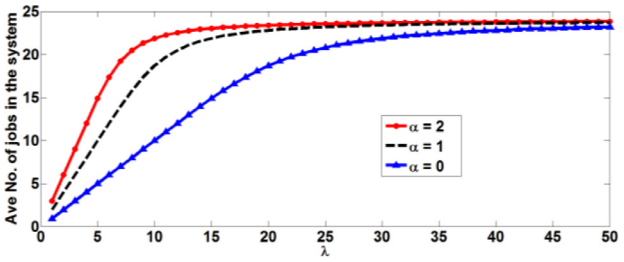

Fig. 21. Average number of the jobs in the system as a function of job arrival rate with α as a parameter for $= \gamma = 1, S = 100$ and four tasks per job.

In Fig. 21, we plotted average number of jobs in the system as a function of the job arrival rate λ with α as a parameter for constant values of μ, γ and S. It is assumed that number of tasks per job is four. As may be seen, average number of jobs in the system increases with increasing value of VM failure rate at any job arrival rate.

The models in [11,24–26] are more general than ours in one aspect, that they allow limited queuing of the jobs. However, the presented analyses have approximations, results are numerical and they do not have easy to use closed forms. The closed form results that we have derived show that distribution of the number of jobs in the system depend on service time only through its mean. We also note that queuing for real-time jobs will not be important. We have also studied a number of models not considered in those works. In the case of jobs with random number of tasks, we analyzed the system for both independent and simultaneous task release times, while the analyses in [24,26] apply only to the independent task release times. Further, our analysis allows heterogeneous VMs with different resource requirements, while their models only allow homogeneous VMs. Finally, we have also studied models that allow a job to generate new tasks during its service time, which will be appropriate to mobile cloud computing environment. The models studied in the work advances state-of-the art in performance modeling of cloud computing centers.

8. Conclusion

In this paper, we have studied performance modeling of cloud computing systems. We have derived job blocking probabilities and distribution of the utilization of resources as a function of the traffic load under various scenarios for systems with both homogenous and heterogeneous VMs. We have determined service fragmentation probabilities and have shown application of the derived results in power management techniques under time-varying traffic loads. We have obtained results for systems that resource requirements of jobs may vary dynamically during their service times, which may be appropriate to mobile cloud computing environment. The derived results of this paper will be useful in dimensioning of cloud computing systems.

Appendix

Proposition 2. $q(\vec{u})$, probability distribution of the utilization of resources may be determined by following multi-dimensional recursion,

$$u_f q(\vec{u}) = \sum_{l=1}^{L} \sum_{j=1}^{J} j b_{\ell f} \rho_{j\ell} q(\vec{u} - j\vec{b}_\ell)$$

Proof. Let us define,

a_ℓ	number of type ℓ VMs that is busy.
$\vec{a}$	$[a_1, a_2, \ldots, a_\ell, \ldots a_L]$
$\vec{j}$	$(1, 2 \ldots, j, \ldots J)$

From the above definitions, we have,

$$a_l = \sum_{(j=1)}^{J} j n_{jl}, u_f = \sum_{(l=1)}^{L} a_l b_l f = \sum_{(l=1)}^{L} \sum_{(j=1)}^{J} j b_{lr} n_{jl} \quad \text{(A. 1)}$$

Then,

$$\vec{a} = \vec{j}N \; \vec{u} = \vec{a}B \quad \text{(A. 2)}$$

$q(\vec{u})$ is given by,

$$q(\vec{u}) = Pr(\vec{a}B = \vec{u}) = \sum_{\vec{n}|\vec{a}B=\vec{u}} p(\vec{n}) \quad \text{(A. 3)}$$

Let us rewrite LBE in Eq. (24) as follows,

$$n_{j\ell} p(\vec{n}) = \rho_{j\ell} p(\overrightarrow{\vec{n}}_{j\ell}^{-}) \quad \text{(A. 4)}$$

Multiplying both sides of (A. 4) by $j b_{\ell f}$ and summing over j and ℓ,

$$p(\vec{n}) \sum_{\ell=1}^{L} \sum_{j=1}^{J} j b_{\ell f} n_{j\ell} = \sum_{\ell=1}^{L} \sum_{j=1}^{J} j b_{\ell f} \rho_{j\ell} p(\overrightarrow{\vec{n}}_{j\ell}^{-})$$

Substituting from (A. 1) on the LHS,

$$u_f p(\vec{n}) = \sum_{\ell=1}^{L} \sum_{j=1}^{J} j b_{\ell f} \rho_{j\ell} p(\overrightarrow{\vec{n}}_{j\ell}^{-}) \quad \text{(A. 5)}$$

Next let us sum both sides of equation (A. 5) over the states $(\vec{n}|\vec{a}B = \vec{u})$,

$$\sum_{\vec{n}|\vec{a}B=\vec{u}} u_f p(\vec{n}) = \sum_{\vec{n}|\vec{a}B=\vec{u}} \sum_{\ell=1}^{L} \sum_{j=1}^{J} j b_{\ell f} \rho_{j\ell} p(\overrightarrow{\vec{n}}_{j\ell}^{-})$$

Substituting from (A. 1) on the LHS and interchanging the order of summations on the RHS,

$$u_f q(\vec{u}) = \sum_{\ell=1}^{L} \sum_{j=1}^{J} j b_{\ell f} \rho_{j\ell} p \sum_{\vec{n}|\vec{a}B=\vec{u}} p(\overrightarrow{\vec{n}}_{j\ell}^{-}) \quad \text{(A. 6)}$$

We note from (A. 1), $(\vec{n}|\vec{a}B = \vec{u}) = (\vec{n}|\vec{j}NB = \vec{u})$
Then $(\vec{n}|\vec{a}B = \vec{u})$ means that,

$$(\overrightarrow{\vec{n}}_{j\ell}^{-} | \vec{j} \overrightarrow{\vec{n}}_{j\ell}^{-} B = \vec{u} - j\overrightarrow{\vec{b}_\ell}) \quad \text{(A. 7)}$$

Substituting (A. 7) in (A. 6) completes the proof.

References

[1] Brian J.S. Chee, C. Franklin Jr., Cloud computing: technologies and strategies of the ubiquitous data center, CRC, New York, 2010.
[2] S. Vakilinia, D. Qiu, M.M. Ali, Optimal multi-dimensional dynamic resource allocation in mobile cloud computing, EURASIP Journal on Wireless Communications and Networking 1 (1) (2014) 1–14.
[3] T. Cordeiro, D. Damalio, et al., Open source cloud computing platforms, in: the Proceedings of 9th IEEE International Conference on Grid and Cooperative Computing (GCC), 2010, pp. 366–371.
[4] D. Jeffrey, S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.
[5] T. White, Hadoop: the definitive guide. O'Reilly, 2012.
[6] H.T. Dinh, C. Lee, D. Niyato and P. Wang, A Survey of mobile cloud computing: architecture, applications and approaches, Wireless Communications and Mobile Computing, 2011.
[7] K. Kumar, Y.H. Lu, Cloud Computing for mobile users: Can offloading Computation Save Energy, IEEE Computer Magazine, pp. 51–56, April 2010.
[8] E. Curvo, A. Balasubramanian, D. Chao, A. Wolman, S. Saroiu, R. Chandra, P. Bahl, MAUI: making smartphones last longer with code offload, in: Proceedings of the 8th international conference on Mobile systems applications (MobiSys), and services, ACM, Chicago, 2010, pp. 49–62.
[9] A. Khan, M. Othman, S. Madani, S. Khan, A survey of mobile cloud computing application models, IEEE Communication Surveys & Tutorials 16 (1) (2014) 393–413.
[10] S. Kosta, A. Aucinas, P. Hui, R. Mortier, X. Zhang, ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading, in: the Proceeding of IEEE INFOCOM, 2012.
[11] H. Khazaei, J. Mīsíc, V.B. Mīsíc, Performance of cloud centers with high degree of virtualization under batch task arrivals, IEEE Transactions on Parallel and Distributed Systems 24 (12) (2013) 2429–2438.
[12] B. Bouterse, H. Perros, Scheduling cloud capacity for time-varying customer demand, in: the Proceeding of IEEE cloud Networking (CLOUDNET), 2012, pp. 137–142.
[13] S.T. Maguluri, R. Srikant, L. Ying, Stochastic models of load balancing and scheduling in cloud computing clusters, in: the Proceeding of IEEE INFOCOM, 2012, pp. 702–710.
[14] A. Stolyar, An infinite server system with general packing constraints, Operations Research 61 (5) (2013) 1200–1217.
[15] D. Breitgand, A. Epstein, Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds, in: the Proceeding of IEEE INFOCOM, 2012, pp. 2861–2865.
[16] B. Javadi, R.K. Thulasiram, R. Buyya, Statistical modeling of spot instance prices in public cloud environments, in: the Proceedings of the IEEE Utility and Cloud Computing (UCC), 2011, pp. 219–228.
[17] J.F. Kaufman, Blocking in a Shared Resource Environment, IEEE Trans. Communications 29 (1981) 1474–1481.
[18] M.V. Ramakrishna, An exact probability model for finite hash table, in: the Proceeding of IEEE Fourth International Conference on Data Engineering, 1988, pp. 362–368.
[19] A. Gandhi, M. Harchol-Balter, R. Das, Ch. Lefurgy, Optimal power allocation in server farms, ACM SIGMETRICS Performance Evaluation Review 37 (1) (2009) 157–168.
[20] L. Kleinrock, Queuing Systems, vol. I, John Wiley & Sons, New York, NY, USA, 1975.
[21] M.A.M. Ferreira, M. Andrade, The M/G/∞ queue busy period distribution exponentially, Journal of Appl. Math. 4 (3) (2011) 249–260.
[22] J.R. Artalejo, M.J. Lopez-Herrero, Analysis of the busy period for the M/M/c queue: An algorithmic approach, Journal of Applied Probability 38 (1) (2001) 209–222.
[23] T. Kimura, A transform-free approximation for the finite capacity M/G/s queue, Operations Research 44 (6) (1996) 984–988.
[24] H. Khazaei, J. Misic, V.B. Misic, Performance Analysis of Cloud Computing Centers Using M/G/m/m+r Queuing Systems, IEEE Transactions on Parallel and Distributed Systems 23 (5) (May 2012) 936–943.
[25] D. Bruneo, A Stochastic Model to Investigate Data Center Performance and QoS in IaaS Cloud Computing Systems, IEEE Transactions on Parallel and Distributed Systems 25 (3) (March 2014) 560–569.
[26] B. Yang, F. Tan, Y. Dai, Performance Evaluation of Cloud Service Considering Fault Recovery, Journal of Supercomputing 65 (2013) 426–444 Springer.

Shahin Vakilinia (S'07) received the B.Sc. degree from University of Tabriz, Tabriz, Iran and the M.Sc. degree from Sharif University of Technology, Tehran, Iran, both in electrical engineering in 2008 and 2010 respectively. He has got his Ph.D. in the Department of Electrical and Computer Engineering at Concordia University, Montreal, QC, Canada in 2015.

Mustafa Mehmet-Ali (M'88) received the B.Sc. and M.Sc. degrees in electrical engineering from Bogazici University, Istanbul, Turkey, in 1977 and 1979, respectively, and the Ph.D. degree in electrical engineering from Carleton University, Ottawa, ON, Canada, in 1983. Until the end of 1984, he was a Research Engineer with Telesat Canada. Since 1985, he has been with the Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada, where he is currently a Professor.

Dongyu Qiu (M'04) received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, and the Ph.D. degree from Purdue University, Indiana, USA, in 2003. He is currently an Associate Professor in the Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada. His research interests are in the areas of peer-to-peer networks, TCP/IP networks, cloud computing, queuing analysis, network security, and wireless networks.