Modeling the Performance of Heterogeneous IaaS Cloud Centers

Hamzeh Khazaei
Computer Science Department
University of Manitoba, Winnipeg, Canada
Email: hamzehk@cs.umanitoba.ca

Jelena Mišić, Vojislav B. Mišić and Nasim Beigi Mohammadi
Computer Science Department
Ryerson University, Toronto, Canada
Emails: {jmisic,vmisic,nbeigimo}@scs.ryerson.ca

***Abstract*—Performance modeling of cloud computing centers is a challenging task due to complexity, shared resources and large scale of such systems. Introducing heterogeneous resources and workloads make the analysis even more complicated. In this paper, we propose a layered stochastic performance model applicable to cloud centers with wide range of heterogeneous physical and virtual resources as well as task characteristics without sacrificing the scalability, granularity and simplicity. Under various parameters and configurations, key performance metrics such as task blocking probability and total delay incurred on user tasks are obtained. The results also reveal practical insights into capacity planning for cloud computing centers.**

I. Introduction and Related Work

Service availability and response time are two important quality measures in cloud's users perspective [1]. Quantifying and characterizing such performance measures requires appropriate modeling; the model ought to include a large number of parameters while still being tractable. A monolithic model may suffer from intractability and poor scalability due to vast parameter space [2]. Instead, in this paper we construct separate sub-models for different servicing steps in a complex cloud center. We assume that the cloud center consists of different types of Physical Machines (PM) that are allocated to users in the order of task arrivals. User requests may share a PM using virtualization technique. Cloud users may request more than one virtual machine (VM) by submitting a single request; such requests are referred as *super-tasks* in this context.

Cloud computing has attracted considerable research attention, but only a small portion of the work done so far has addressed performance issues by rigorous analytical models [3]. In [4], authors applied classical Erlang loss formula and $M/M/m/K$ queuing system for response time and outbound bandwidth modeling respectively. A general analytical model for an end-to-end performance analysis of a cloud service is proposed in [2]. However the proposed model is limited to single task arrival per request and also the effect of virtualization has not been explicitly addressed in performance results. In [5], the authors studied the response time in terms of various metrics, such as the overhead of acquiring and realizing the virtual computing resources, and other virtualization and network communication overhead.

Our previous work [1], [6] presents monolithic analytical models which are quite restrictive compared to this work in terms of extendability, simplicity and computational cost. Also, [1], [6] do not address the concept of virtualization as well as heterogeneous server pools and PMs. In [7], we employed a homogeneous integrated performance model to overcome the complexity of cloud computing centers. In this work however, we introduce heterogeneity to our performance model. More specifically, we permit PMs to be different among pools and also VMs are to be different in terms of number of virtual CPUs (vCPU).

The rest of the paper is organized as follows. Section II presents our model and the details of the analysis. Section III presents the numerical results obtained from the analytical model. Finally, section IV summarizes our findings and concludes the paper.

II. Analytical Model

In IaaS cloud, when a request arrives, a pre-built or customized disk image is used to create one or more VM instances. We assume that PMs are categorized into three server pools: hot (i.e., with running VMs), warm (i.e., turned on but without running VMs) and cold (i.e., turned off). Each pool includes specific type of PMs. PMs in pools are different in terms of virtual CPU (vCPU), number of cores per CPU, RAM, disk and the degree of virtualization (i.e the max number of running VMs on the PM).

Instantiation of a VM from an image and provisioning it on hot PMs has the minimum delay compared to warm and cold PMs. Warm PMs require some time to be ready for provisioning and cold PMs require additional time to be started up before a VM instantiation. In this work we allow users to request more than one VM by submitting a *super-task*; a super-task may contain more than one task, each of which requires one VM. Each VM needs one vCPU and each vCPU may run on multiple physical CUP cores [8]. The size of super-task (i.e., number of VMs) and size of vCPU (i.e., number of cores assigned to a vCPU) can be generally distributed. For numerical validation the super-task size is assumed to be uniformly or geometrically distributed but we truncate the distribution of the value by MSS (Maximum Super-task Size). MSS is the maximum number of VMs running on a single PM. A super-task is shown in Fig. 1. We also use uniform distribution for the vCPU size in the numerical section.

# of vCPUs per VM	# of Cores per vCPU	RAM (Gig)	Disk (Gig)	# of VMs

Fig. 1. Specification of a typical super-task submitted by users.

Due to high interaction among tasks within a super-task, each super-task will be provisioned on the same PM. A super-task may request up to all available VMs or CPU cores on a single PM. In this work, all VMs are the same in terms of RAM and disk while each VM may request a different number of CPU cores.

Server Assigning Module (SAM) starts with the hot pool to provision the super-task on a PM machine. If the process is not successful then SAM examines the warm pool. if there is no PM in warm pool that can accommodate the super-task, SAM refers to the cold pool. Finally, SAM either assigns a PM in one of the pools to a super-task or rejects the super-task. As a result, user requests (super-tasks) may get rejected either due to lack of space in global queue or insufficient resource at the cloud center. When a running task finishes, the capacity used by that VM is released and becomes available for servicing the next task.

We identify four main delays imposed by cloud computing centers on a user requests: at first, it should be determined whether cloud center has sufficient resources to fulfill the super-task. Using SAM, we capture the details of delays in the global queue as well as decision process. Then, VM Provisioning Module (VMPM) undertakes the provisioning of VMs. Therefore each task within super-task may experience queuing delay at PM's queue and VM provisioning process delay. After all, the actual service can start. Therefore, the response time can be considered as the sum of four above mentioned delays and the actual service time.

In order to model each module precisely, we design two stochastic sub-models for SAM and VMPM. We then connect these sub-models into an overall performance model to compute the task rejection probability and mean response delay. We describe our analysis in the following sections.

A. Server Allocation Sub-Model

The server assigning module (SAM) may be represented by the server allocation sub-model (SASM) shown in Fig. 2. SASM is a two-dimensional Continuous Time Markov Chain (CTMC) in which we take care of number of super-tasks in the global queue as well as the current pool on which provisioning is taking place. Since the number of potential users is high and a single user typically submits super-tasks at a time with low probability, the super-task arrival can be adequately modeled as a Poisson process [9] with rate λ_{st}. Super-tasks are lined up in the finite global queue to be processed in a first-in, first-out (FIFO) basis. Each state in Markov chain is labeled as (i, j), where i indicates the number of super-tasks in the global queue and j denotes the pool on which the leading super-task is under provisioning. State $(0, 0)$ indicates that system is empty which means there is no super-task under provisioning or in the global queue. Index j can be h, w or c that indicates current super-task is undergoing provisioning on the hot, warm or cold pool respectively. The size of global queue is L_q and one more super-task can be at the deployment unit for provisioning thus, the capacity of system is $L_q + 1$. Let P_h, P_w and P_c be the success probabilities of finding a

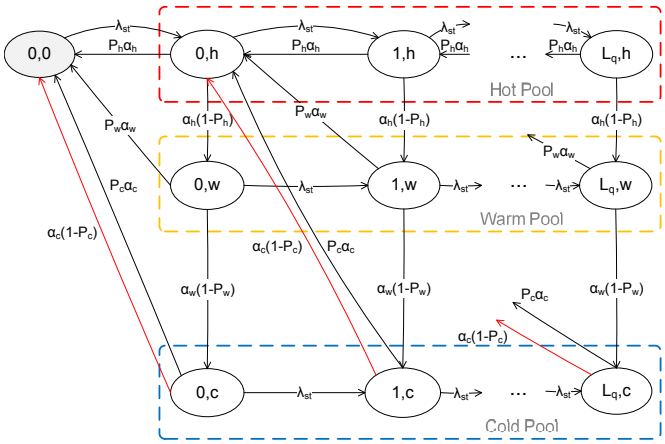


Fig. 2. Server allocation sub-model.

PM that can accept the current super-task in the hot, warm and cold pool respectively. We assume that $1/\alpha_h$, $1/\alpha_w$ and $1/\alpha_c$ are the mean look up delays for finding an appropriate PM in the hot, warm and cold pool respectively. Upon arrival of first super-task, system moves to state $(0, h)$ which means the super-task will be provisioned immediately in the hot pool. Afterwards, depending on the upcoming event, three possible transitions can occur:

(a) Another super-task has arrived and system moves to state $(1, h)$ with rate λ_{st}.
(b) A PM in the hot pool admits the super-task so that system moves back to state $(0, 0)$ with rate $P_h\alpha_h$.
(c) None of PMs in the hot pool has enough capacity to accept the super-task, so the system will examine the warm pool (i.e., moves to state $(0, w)$) with rate $(1 - P_h)\alpha_h$.

On state $(0, w)$, SASM tries to provision the super-task on the warm pool; if one of the PMs in warm pool can accommodate the super-task, the system will get back to $(0, 0)$, otherwise, SASM checks the cold pool (i.e., transition from state $(0, w)$ to $(0, c)$). If none of the PMs in the cold pool can provision the super-task as well, the system moves back from $(0, c)$ to $(0, 0)$ with rate $(1 - P_c)\alpha_c$ meaning that the user request (super-task) will get rejected due to insufficient resources in the cloud center. During the provisioning in the cold pool, another super-task may arrive and takes the system to state $(1, c)$ which means there is one super-task in deployment unit and one awaiting in the global queue. Finally, the super-task under provisioning decision leaves the deployment unit, once it receives a decision from SASM and the next super-task at the head of global queue will go under provisioning. In this sub-model, arrival rate of super-task (λ_{st}) and look up delays

($1/\alpha_h, 1/\alpha_w$ and $1/\alpha_c$) are external parameters and success probabilities (P_h, P_w and P_c) are calculated from the VM provisioning sub-model. The VM provisioning sub-model will be discussed in the next section.

Using steady-state probabilities $\pi_{(i,j)}$, blocking probability can be calculated. Super-tasks may experience two kinds of blocking events:

(a) Blocking due to a full global queue that occurs with the probability of $BP_q = \pi_{(L_q,h)} + \pi_{(L_q,w)} + \pi_{(L_q,c)}$.
(b) Blocking due to insufficient resources (PMs) at pools, with the probability of

$$BP_r = \sum_{i=0}^{L_q} \frac{\alpha_c(1-P_c)}{\alpha_c + \lambda_{st}} \pi_{(i,c)}$$

The probability of rejection is, then, $P_{reject} = BP_q + BP_r$. In order to calculate the mean waiting time in the global queue, we first establish the probability generating function (PGF) for the number of super-tasks in the queue [9],

$$Q(z) = \pi_{(0,0)} + \sum_{i=0}^{L_q} (\pi_{(i,h)} + \pi_{(i,w)} + \pi_{(i,c)}) z^i \qquad (1)$$

The mean number of super-tasks in the global queue is $\overline{q} = Q'(1)$. Applying Little's law, the mean waiting time in the global queue is given by

$$\overline{wt} = \frac{\overline{q}}{\lambda_{st}(1 - P_{reject})} \qquad (2)$$

Look up time among pools can be considered as a Coxian distribution with 3 steps. Thus it can be calculated as,

$$\overline{lut} = \frac{1/\alpha_h + (1-P_h)((1/\alpha_w) + (1-P_w)(1/\alpha_c))}{1 - P_{reject}} \qquad (3)$$

B. VM Provisioning Sub-Model

Virtual Machine Provisioning Sub-Model (VMPSM) captures the instantiation, deployment and running of VMs on a PM. Fig. 3 shows the VMPSM (a 4-dimensional CTMC) for a PM in the hot pool. For the sake of simplicity in representation, Fig. 3 just shows the VMPSM when each super-tasks has only one task. A PM in warm or cold pool can be modeled as a hot PM though, with some minor differences. Consequently, each pool can be modeled as a set of VMPSM with the same arrival and instantiation rate. Each state in Fig. 3 is labeled by (i, j, k, l) in which i indicates the number of tasks in PM's queue, j denotes the number of task that is under provisioning, k indicates the number of busy cores and l is the number of VM that are already deployed on the PM. Note that we set the queue size at each PM equal to the maximum number of VMs that can be deployed on a PM. Therefore, a super-task can be accepted by a PM if first, there is enough room in the PM's queue for all tasks within the super-task and second, if the PM has sufficient number of free cores for the given super-task. Let ϕ_h be the rate at which a VM can be deployed on a PM at hot pool and μ be the service rate of each VM. So, the total service rate for each PM is the product of number of running VMs by μ. State $(0, 0, 0, 0)$ indicates that the PM is empty and there is no task either in the queue or in the instantiation unit. The arrival rate to each PM in the hot pool is given by:

$$\lambda_h = \frac{\lambda_{st}(1 - BP_q)}{N_h} \qquad (4)$$

in which N_h is the number of PMs in the hot pool.

The state transition in VMPSM can occur due to super-task arrival, task instantiation or service completion. From state $(0, 0, 0, 0)$, system can move to state $(0, 1, 0, 0)$ with rate λ_h; at this state, because the instantiation unit is free the super-task will go under provisioning immediately. From $(0, 1, 0, 0)$, system may transit to $(1, 1, 0, 0)$ upon arrival of another super-task. From state $(0, 1, 0, 0)$ system also may move to one of the states $\{(0, 0, 1, 1), (0, 0, 2, 1), \cdots, (0, 0, m, 1)\}$ with the rate of $p_i\phi_h$. Probability p_i is the probability by which a super-task my request i cores for a single VM. From states $\{(0, 0, 1, 1), (0, 0, 2, 1), \cdots, (0, 0, m, 1)\}$ system can get back to $(0, 0, 0, 0)$ with rate μ (i.e., service completion rate), or again upon arriving a super-task, system can move to one of the states $\{(0, 1, 1, 1), (0, 1, 2, 1), \cdots, (0, 1, m, 1)\}$ with rate λ_h. Now consider the state $(0, 0, 3, 2)$ indicates two VMs are running in which one of them employs one core and the other one engages two cores. System can move to $(0, 0, 2, 1)$ or $(0, 0, 1, 1)$ with rate $2\mu p_1$ (i.e., completed VM releases one core) or $2\mu p_2$ (i.e., completed VM releases two cores) respectively. Also, system may move to $(0, 1, 3, 2)$ upon arrival of another super-task.

Suppose that $\pi^h_{(i,j,k,l)}$ is the steady-state probability for the hot PM model (Fig. 3) to be in the state (i, j, k, l). Using steady-state probabilities, we can obtain the probability that at least on PM in hot pool can accept the super-task for provisioning. At first we need to compute the probability that a hot PM cannot admit a super-task for provisioning (P^h_{na}). Let m be the total number of cores on a PM and $MCS \leq m$ be the maximum number of cores that can be requested by a super-task. Since each VM requires at least one vCPU (i.e., one core), the maximum number of VMs on a single PM (MSS) is not going beyond m. In Fig. 3, let $F_h(i, j, k, l) = m-(i+j+l)$ and $G_h(i, j, k, l) = m - k$ be the free capacity and cores at each state respectively. P^h_{na} is then given by:

$$P^h_{na} = \sum_{x \in \xi} \sum_{t=F_h+1}^{MSS} \pi^h_x P^*_t + \sum_{y \in \zeta} \sum_{s=G_h+1}^{MCS} \pi^h_y p_s \qquad (5)$$

where P^*_t is the probability of arrival a super-task with size t and ξ is a set of states in which the system might block an arrival super-task due to lack of space:

$$\xi = \{(i, j, k, l) |\ F_h(i, j, k, l) < MSS\}$$

and p_s is the probability by which a super-task may request for s cores and ζ is a set of states in which the system might block an arrival super-task due to lack of free cores:

$$\zeta = \{(i, j, k, l) |\ G_h(i, j, k, l) < MCS\}$$

Therefore, probability of successful provisioning (P_h) in the hot pool can be obtained as $P_h = 1 - (P^h_{na})^{N_h}$. where N_h is

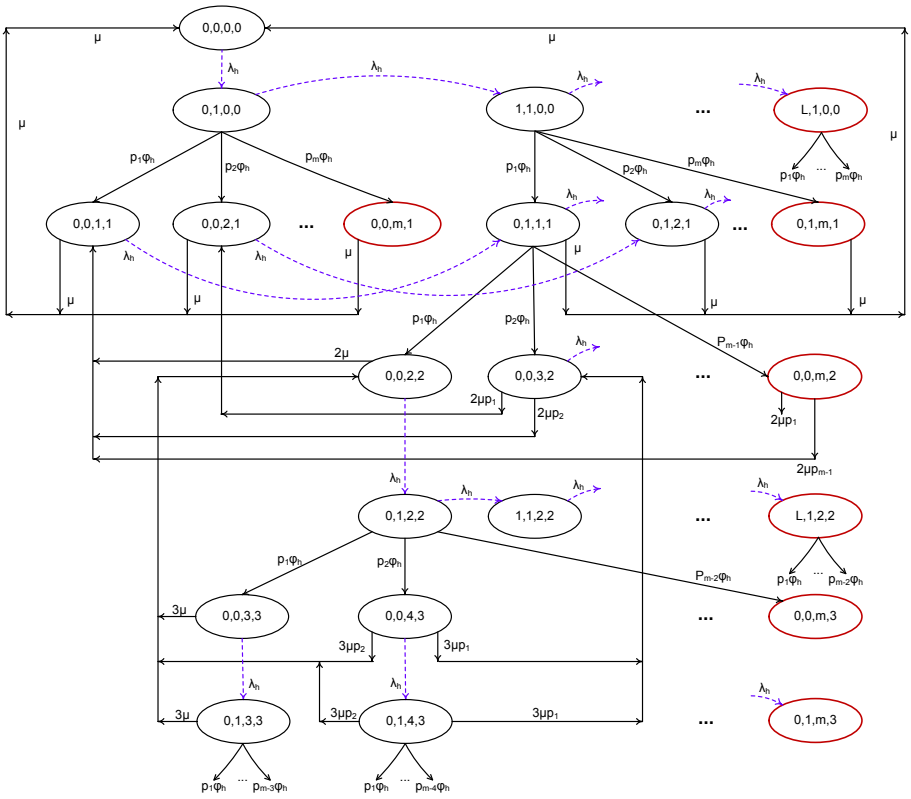


Fig. 3. Virtual machine provisioning sub-model for a PM in the hot pool.

the number of PMs in the hot pool. Note that P_h is used as an input parameter in SASM (Fig. 2). The provisioning model for warm PM is almost the same with the one for hot PM, though, there are some differences in parameters:

(a) The arrival rate to each PM in warm pool is

$$\lambda_w = \frac{\lambda_{st}(1 - BP_q)(1 - P_h)}{N_w} \tag{6}$$

(b) Every PM in warm pool requires extra time to be ready for first instantiation. This time is assumed to be exponentially distributed with mean value of $1/\gamma_w$.

(c) The first instantiation is occurred by rate ϕ_w and the rest by ϕ_h.

(d) The specifications of PMs in the warm pool are different from hot pool. (i.e., MSS, MCS and m)

Like VMPSM for a hot PM (Fig. 3), the model for a warm PM is solved and the steady-states probabilities ($\pi^w_{(i,j,k,l)}$), are obtained. The success probability for provisioning a super-task in warm pool is $P_w = 1 - (P^w_{na})^{N_w}$. A PM in the cold pool can be modeled as in the warm pool but with different arrival rate, instantiation rate and start up time. The effective arrival rate at each PM in the cold pool is given by:

$$\lambda_c = \frac{\lambda_{st}(1 - BP_q)(1 - P_h)(1 - P_w)}{N_c} \tag{7}$$

A cold PM (off machine), requires longer time than a warm PM to be ready (hot) for the first instantiation. We assume that the start up time is also exponentially distributed with mean value of $1/\gamma_c$. The success probability for provisioning a super-task in the cold pool is $P_c = 1 - (P^c_{na})^{N_c}$. From VMPSM, we can also obtain the mean waiting time at PMs' queue ($\overline{PM_{wt}}$) and mean provisioning time ($\overline{PM_{pt}}$) by using the same approach as the one that led to Eq. (2). As a result, the total delay before starting of actual service time, is given by:

$$\overline{td} = \overline{wt} + \overline{lut} + \overline{PM_{wt}} + \overline{PM_{pt}} \tag{8}$$

Note that all success probabilities (i.e., P_h, P_w and P_c) are input parameters to the server allocation sub-model (Fig. 2).

C. Interaction among sub-models

The interactions among sub-models is depicted in Fig. 4. VM provisioning sub-models (VMPSM) compute the steady state probabilities (P_h, P_w and P_c) that at least one PM in a pool (hot, warm and cold, respectively) can accept a super-task for provisioning. These probabilities are used as input parameters to the server allocation sub-model (SASM). Hot PM sub-model (VMPSM$_{hot}$) computes P_h which is the input parameter for both warm and cold sub-models; warm PM sub-model (VMPSM$_{warm}$) computes P_w which is the input parameter for the cold sub-model. The server allocation sub-model (SASM) computes the blocking probability, BP_q, which is the input parameter to VM provisioning sub-models. As can

be seen, there is an inter-dependency among performance sub-models. This cyclic dependency is resolved via fixed-point iterative method [2] using a modified version of successive substitution approach.

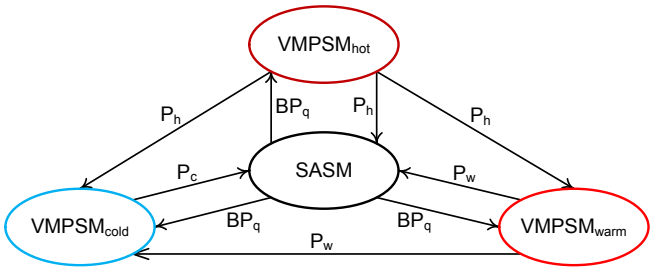


Fig. 4. Interaction among sub-models.

III. Numerical validation

The resulting sub-models have been implemented and solved using Maple 15 from Maplesoft, Inc. [10]. The successive substitution method is continued until the difference between the previous and current value of blocking probability in global queue (i.e., BP_q) is less than 10^{-6}. The integrated model converges to the solution in 8 iterations or less. Table I shows the specifications of PMs in each pool that we use for the numerical validation.

TABLE I
Specifications of PMs in each pool.

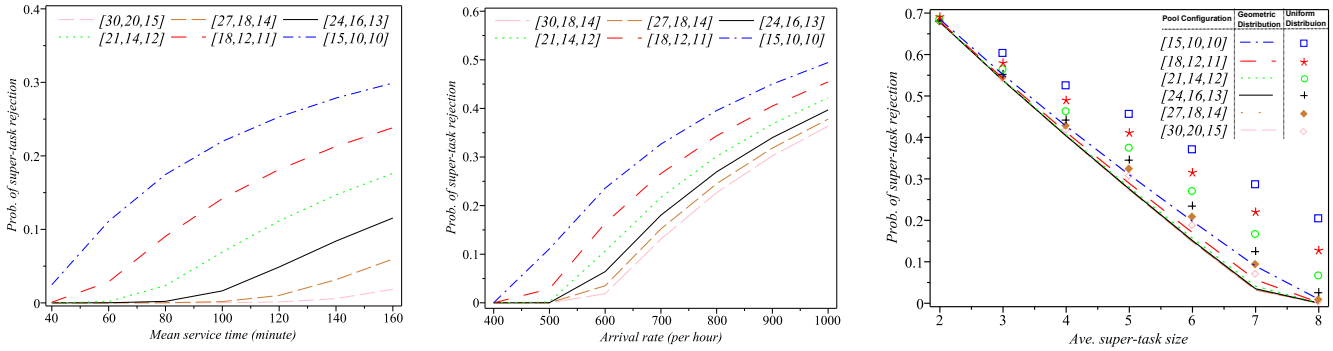
Parameters	Hot Pool	Warm Pool	Cold Pool
# of CPU cores	10	8	4
RAM (GB)	40	25	10
Disk (GB)	500	250	200
Max # of VMs	10	8	2
# of cores per vCPU	1,2,3 or 4	1,2 or 4	2 or 4

We show the effects of changing arrival rate, task service time, number of PMs in each pool, number of VMs on each PM and super-task size on the interested performance indicators. Arrival rate ranges from 400 to 1000 STs per hour. Mean service time of each task within STs ranges form 40 to 160 minutes. We assume 35 to 65 PMs in pools. The look-up time for finding a proper PM is assumed to be independent of the number of PMs in the pool and the type of pool (i.e., hot, warm and cold). Look-up rate is set to 625 and 825 searches per hour for warm and cold pools. Mean preparation delay for a warm and cold PM to become a hot PM (i.e., be ready for first VM instantiation) are assumed to be 1 to 3 and 5 to 10 minutes, respectively. Mean time to deploy a VM on a hot PM is assumed to be between 4 and 10 minutes. First VM instantiation on a warm and cold PM are to be between 4 to 7 and 8 to 12 minutes, respectively. After first VM instantiation on a warm or cold PM, it has already become a hot PM so that the next VM can be deployed just like a hot PM. The size of global queue is set to 100 super-tasks (ST) and the number of cores per vCPU is assumed to be uniformly distributed.

In the first experiment, the results of which is shown in Figs. 5(a) and 5(d), STs have one task each, and PMs are permitted to run only one VM. Fig. 5(a) shows (at a constant arrival rate of 500 STs/hr and different numbers of PMs in each pool) that by increasing the mean service time the rejection probability increases almost linearly. The configuration [21,14,12] seems to be a reasonable choice to maintain the rejection probability below 15%. In addition, it can be observed that by increasing the capacity of system (i.e., having more PMs in the pools) the maximum gain occurs at the longest service time (i.e. 160 minutes). Fig 5(d) shows that a longer service time will result in longer total delay on STs. Unlike the rejection probability, Fig. 5(a), there is no unique service time that yield the maximum gain for all pool configurations.

Under different arrival rates, we also determine the two performance metrics while the average service time is set to 60 min, Figs. 5(b) and 5(e). One turning point can be noticed in rejection probability curves at which rejection probability increases suddenly. Such points, e.g., Fig. 5(b), 500 STs/hour for configuration [27,18,14], may suggest the appropriate time to upgrade or inject more resources (i.e., PMs). However, in case of total delay, two turning points (three regimes of operation) can be identified, Fig. 5(e): the first regime (i.e., stable regime) is the region in which the cloud providers would like to operate. (Admission control may use provided information here and helps the cloud centers to operate in such regime by not accepting extra requests) Transient regime is in between two turning points which the total delay increases sharply. Cloud centers should avoid entering such region since the violation of SLA is about to happen. As the transient regime is not too narrow, it is possible for cloud center to return to stable zone after a short time (i.e., the admission control has enough time to adjust the new admissions). The transient regime is attributed to the size of global queue. The longer the global queue is, the wider the transient regime is going to be. In saturation regime, the total delay for some configurations such as [15,10,10], [18,12,11] and [21,14,12] is even getting decreased at the expense of high rejection rate of super-tasks. This area of operation is totally unpleasant for both cloud providers and users since the cloud center is mostly unavailable for majority of the users.

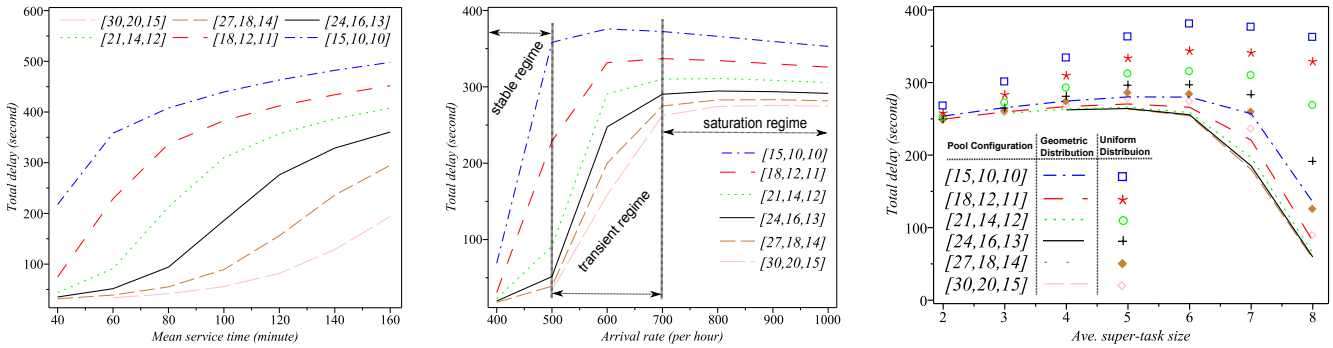
In the last experiment, Figs. 5(c) and 5(f), we examine the effect of super-task size on task rejection probability and total delay using of geometric (shown with lines) and uniform (shown with symbols) distributions for super-task size. Each VM will be assigned one vCPU and each vCPU may use multiple cores (details in Table I). Number of cores per vCPU is assumed to be uniformly distributed. A super-task may request up to all of the available VMs and cores on a PM. Note that the aggregate arrival rate is set to 3500 tasks/hr, however the effective arrival rate of super-tasks is various for different super-task size. For instance, if the average super-task size was five, then the effective arrival rate of super-tasks would be 700 STs/hr. As can be seen by increasing the size of super-tasks the rejection probability reduces sharply for both geometric and uniform distribution. Since the size of super-



(a) Rejection probability: arrival-rate=500 tasks/hour.

(b) Rejection probability: service-time=60 minute.

(c) Rejection probability: service-time=60 minute, aggregate arrival-rate=4500 tasks/hour.



(d) Total delay on a super-task: arrival-rate=500 tasks/hour.

(e) Total delay on a super-task: service-time=60 minute.

(f) Total delay on a super-task: service-time=60 minute, aggregate arrival-rate=4500 tasks/hour.

Fig. 5. Effective rejection probabilities and total servicing delays under different input parameters.

tasks is truncated to the maximum number of VMs allowed on each PM (here, up to 8 VMs), the bigger super-task size will result in the lower number of arrivals as well as lower deviation of super-task size. As can be seen, by increasing the capacity the maximum delay occurrs when the mean size of STs is 6, and the maximum reduction of delay is obtained when the mean size of STs is 8. For STs of size 6 or bigger, the dominant imposed delays are the processing delays (i.e, resource assigning and VM provisioning delays) as opposed to the queuing delays. Due to higher deviation, the uniform distribution imposes more rejections and delay on super-tasks.

IV. Conclusions

In this paper, we present a performance model suitable for cloud computing centers with heterogeneous requests and resources using interacting stochastic models. We quantify the effects of variation in super-task arrival, task service time, super-task size and VM size on quality of cloud services. Our model provides a clear picture of cloud center operation regimes so that a service provider can manage in advance to stay in the desired operation region. Moreover, under different configurations, the behavior of cloud center is characterized so that an effective admission control can be achieved. We plan to extend the heterogeneity in super-tasks (i.e., RAM and disk) and server pools (i.e., different PMs in each pool) in our future works.

References

[1] H. Khazaei, J. Mišić, and V. B. Mišić, "Performance analysis of cloud computing centers using $M/G/m/m+r$ queueing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 5, p. 1, 2012.

[2] R. Ghosh, K. S. Trivedi, V. K. Naik, and D. S. Kim, "End-to-end performability analysis for infrastructure-as-a-service cloud: An interacting stochastic models approach," in *Proceedings of the 2010 IEEE 16th Pacific Rim International Symposium on Dependable Computing*, ser. PRDC '10, Washington, DC, USA, 2010, pp. 125–132.

[3] A. Iosup, N. Yigitbasi, and D. Epema, "On the performance variability of production cloud services," in *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May. 2011, pp. 104–113.

[4] H. Qian, D. Medhi, and K. S. Trivedi, "A hierarchical model to evaluate quality of experience of online services hosted by cloud computing," in *Integrated Network Management (IM), IFIP/IEEE International Symposium on*, May. 2011, pp. 105–112.

[5] N. Yigitbasi, A. Iosup, D. Epema, and S. Ostermann, "C-meter: A framework for performance analysis of computing clouds," in *CCGRID09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009, pp. 472–477.

[6] H. Khazaei, J. Mišić, and V. B. Mišić, "Performance analysis of cloud centers under burst arrivals and total rejection policy," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, Dec. 2011, pp. 1–6.

[7] H. Khazaei, J. Mišić, V. B. Mišić, and S. Rashwand, "Analysis of a pool management scheme for cloud computing centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, no. PrePrints, 2012.

[8] VMware, Inc., "VMware VMmark 2.0 benchmark results," Website, Sept. 2012, http://www.vmware.com/a/vmmark/.

[9] G. Grimmett and D. Stirzaker, *Probability and Random Processes*, 3rd ed. Oxford University Press, Jul 2010.

[10] Maplesoft, Inc., "Maple 15," Website, Mar. 2011, http://www.maplesoft.com.