# Availability Analysis of Cloud Computing Centers

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

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

***Abstract*—Accurate availability and performance analysis are important requirements to guarantee quality of services (QoS) for cloud users. In this paper, we develop an analytical performance and availability model using interacting stochastic sub-models. Each sub-model captures a specific aspect of cloud centers. The key performance metrics such as task blocking probability and total delay incurred on user tasks are obtained. Our results can be used by an admission control to prevent the cloud center from entering unsafe regimes of operation. 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 and on top of them an availability model which captures the failure/repair effects of resources. We assume that the cloud center consists of a number 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 user 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]. A general analytic model based approach for an end-to-end performance analysis of a cloud service is proposed in [4]. However the proposed model is limited to the single arrival of requests and the start up delay of cold PMs has not been captured. Also, the effect of virtualization was not reflected in their results. In [5], the cloud center was modeled as an $M/M/m/m+r$ queuing system in which inter-arrival and service times are exponentially distributed, and the system has a finite buffer. The response time was partitioned into waiting, service, and execution periods, assuming that all three periods are independent. Our earlier work [6], [1], [7] presents monolithic analytical models which are quite restrictive compared to this work in terms of extendability simplicity and computational cost. Also, that work does not address the concept of virtualization as well as heterogeneous server pools and PMs. In [8], authors applied classical Erlang loss formula and $M/M/m/K$ queuing system for response time and outbound bandwidth modeling respectively.

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 is arrived, 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). 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. The size of super-task is assumed to be geometrically distributed. However, for practical reason we set a maximum size of super-tasks (MSS) to conduct the numerical analysis (i.e., truncated geometric distribution).

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 on a single PM. In this work, we assume that all PMs and VMs are homogeneous and *total acceptance policy* [9] is adopted. Under this policy the system tries to service all tasks within a super-task at the same time. User requests (super-tasks) are submitted to a global finite queue and then processed on the first-in, first-out basis (FIFO).

Resource Assigning Module (RAM) starts with the hot pool to provision the super-task on a PM machine (Fig. 1). If the process is not successful then RAM examines the warm pool. if there is no PM in warm pool that can accommodate the super-task, RAM refers to the cold pool. Finally, RAM 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 .

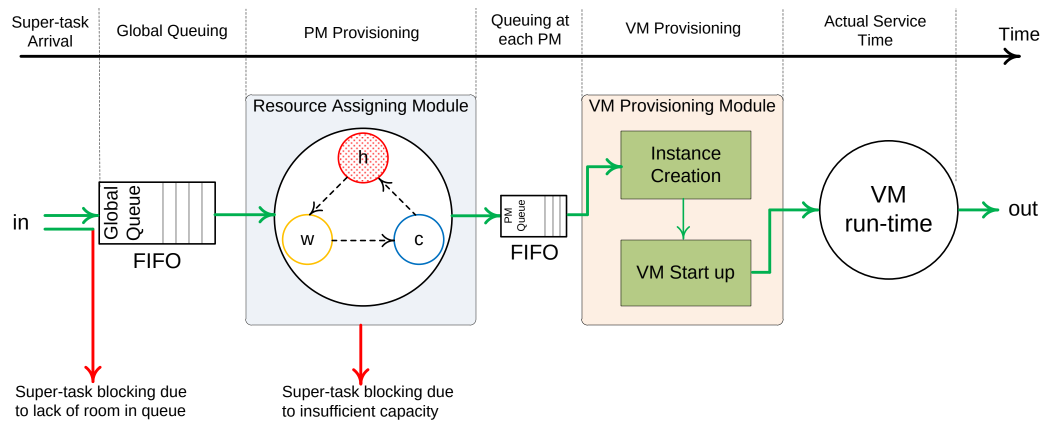


Fig. 1. The steps of servicing and corresponding delays.

We identify four main delays imposed by cloud computing centers on a user requests: at first, it should be determined whether cloud center has enough resources to fulfill the super-task or not. Using RAM, 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 RAM and VMPM. We then connect these sub-models into an overall performance model to compute the task rejection probability and mean response delay. Submitting the outputs of performance model into availability model provides us the effective value of above-mentioned performance metrics. We describe our analysis in the following sections.

## A. *Resource Allocation Sub-Model*

The resource assigning module (RAM) may be represented by the resource allocation sub-model (RASM) shown in Fig. 2. RASM 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 $\lambda_{st}$. 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 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 $\lambda_{st}$.
(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 th 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 $(0, w)$, RASM 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, RASM 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 RASM and the next super-task at the head of global queue will go under provisioning. In this sub-model, arrival rate of super-task ($\lambda_{st}$) 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

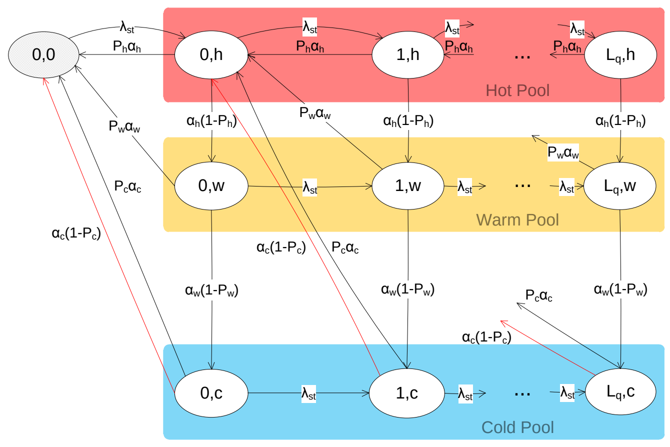


Fig. 2. Resource allocation sub-model.

can be calculated. Two types of blocking may happen to a given super-task:

(a) Blocking due to a full global queue that occurs with the probability of

$$BP_q = \pi_{(L_q,h)} + \pi_{(L_q,w)} + \pi_{(L_q,c)} \tag{1}$$

(b) Blocking due to insufficient resources (PMs) at pools, with the probability of

$$BP_r = \sum_{i=0}^{L_q} \frac{\alpha_c(1-P_c)}{\alpha_c + \lambda_{st}} \pi_{(i,c)} \tag{2}$$

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],

$$Q(z) = \pi_{(0,0)} + \sum_{i=0}^{L_q} (\pi_{(i,h)} + \pi_{(i,w)} + \pi_{(i,c)}) z^i \tag{3}$$

The mean number of super-tasks in the global queue is the first derivative of $Q(z)$ at $z = 1$,

$$\overline{q} = Q'(1) \tag{4}$$

Applying Little's law, the mean waiting time in the global queue is given by:

$$\overline{wt} = \frac{\overline{q}}{\lambda_{st}(1 - P_{reject})} \tag{5}$$

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

$$\overline{lut} = \frac{1/\alpha_h + (1 - P_h)((1/\alpha_w) + (1 - P_w)(1/\alpha_c))}{1 - P_{reject}} \tag{6}$$

## *B. VM Provisioning Sub-Model*

Virtual Machine Provisioning Sub-Model (VMPSM) captures the instantiation, deployment and running of VMs on a PM. In this model, we assume homogeneous PMs, VMs and tasks. We also assume that each PM has 2 Virtual Machine Monitors (VMM) that deploy and monitor VMs on PMs. Fig. 3 shows the VMPSM (a CTMC) for a PM in the hot pool. A PM in warm or cold pool can be modeled with the same VMPSM, though, the arrival and instantiation rates are different. Consequently, each pool (hot, warm and cold) can be modeled as a set of VMPSM with the same arrival and instantiation rate. Each state in Fig. 3 is labeled by $(i, j, k)$ in which $i$ indicates the number of tasks in PM's queue, $j$ denotes the number of task that is under provisioning and $k$ 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. Let $\phi_h$ be the rate at which a VM can be deployed on a PM at hot pool and $\mu$ be the service rate of each VM. So, the total service rate for each PM is the product of number of running VMs by $\mu$. State $(0, 0, 0)$ indicates that the PM is empty and there is no task either in the queue or in the instantiation unit. Depends on the size of arriving super-task, model transits to one of the states in $\{(0, 1, 0), (0, 2, 0), \cdots, (MSS - 2, 2, 0)\}$, in which $MSS$ is the maximum possible size of super-tasks. The arrival rate to each PM in the hot pool is given by:

$$\lambda_h = \frac{\lambda_{st}(1 - BP_q)}{N_h} \tag{7}$$

in which $N_h$ is the number of PMs in the hot pool. Note that $BP_q$, used in (1), is obtained from RASM. In Fig. 3, $\{\lambda_i, i = 1..MSS\}$ is given by $\lambda_i = \lambda_h p_i$; here $p_i$ is the probability of having a super-task of size $i$. The state

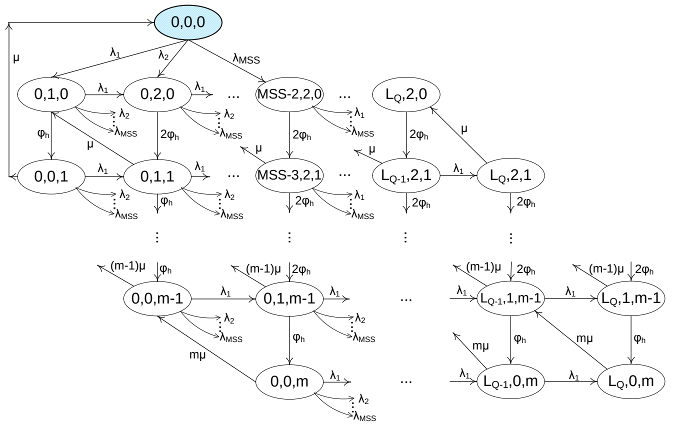


Fig. 3. Virtual machine provisioning sub-model for a PM in the hot pool.

transition in VMPSM can occur due to super-task arrival, task instantiation or service completion. From state $(0, 0, 0)$, system can move to state $(0, 2, 0)$ with rate $\lambda_2$ (i.e., super-task with size 2). From $(0, 2, 0)$, system can transit to $(0, 1, 1)$ with rate $2\phi_h$ (i.e., instantiation rate by 2 VMMs) or upon arriving a super-task with size $k$, moves to state $(k, 2, 0)$.

From $(0, 1, 1)$, system can move to $(0, 0, 2)$ with rate $\phi_h$, transits to $(0, 1, 0)$ with rate $\mu$ (i.e., service completion rate), or again upon arriving a super-task with size $k$, system can move to state $(k-1, 2, 1)$ with rate $\lambda_k$. A PM can accept a super-task, if only it can provision all tasks within super-task. Suppose that $\pi^h_{(i,j,k)}$ is the steady-state probability for the hot PM model (Fig. 3) to be in the state $(i, j, k)$. 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 maximum possible number of VMs on a PM and $F_h$ be the free capacity at each state: $F_h(i, j, k) = m - (i + j + k)$. $P^h_{na}$ is then given by:

$$P^h_{na} = \sum_{x \in \xi} \sum_{t=F_h+1}^{MSS} \pi^h_x p_t \tag{8}$$

where $p_t$ is the probability of arrival a super-task with size $t$ and $\xi = \{(i, j, k) | \, F_h(i, j, k) < MSS\}$. Therefore, probability of success provisioning ($P_h$) in the hot pool can be obtained as

$$P_h = 1 - (P^h_{na})^{N_h} \tag{9}$$

Note that $P_h$ is used as an input parameter in RASM (Fig. 2). The provisioning model for warm PM is 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

$$\lambda_w = \frac{\lambda_{st}(1 - BP_q)(1 - P_h)}{N_w} \tag{10}$$

where $N_w$ is the number of PMs in the warm pool.

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

Instantiation rate in warm pool is the same as in hot pool ($\phi_h$). Like VMPSM for a hot PM (Fig. 3), the model for a warm PM is solved and the steady-states probabilities ($\pi^w_{(i,j,k)}$), are obtained. The success probability for provisioning a super-task in warm pool is:

$$P_w = 1 - (P^w_{na})^{N_w} \tag{11}$$

A PM in the cold pool can be modeled as in the warm and hot pool but with different arrival rate and start up time. The effective arrival rate at each PM in the cold pool is given by:

$$\lambda_c = \frac{\lambda_{st}(1 - BP_q)(1 - P_h)(1 - P_w)}{N_c} \tag{12}$$

in which $N_c$ is the number of PMs in the cold pool. 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 $\gamma_c$. The success probability for provisioning a super-task in the cold pool is given by:

$$P_c = 1 - (P^c_{na})^{N_c} \tag{13}$$

From VMPSM, we can also obtain the mean waiting time at PMs' queue ($\overline{PM_{wt}}$) and mean provisioning time ($\overline{pt}$) by using the same approach as the one that led to Eq. (5). As a result, the total delay before starting of actual service time, is given by:

$$\overline{td} = \overline{wt} + \overline{lut} + \overline{PM_{wt}} + \overline{pt} \tag{14}$$

## C. Availability Model

The availability model deals with the failure of PMs, repairing process and employment of cold PMs in case of failure in running machines. This model realizes by a three-dimensional CTMC in which $(i, j, k)$ indicate the initial number of PMs in the hot, warm and cold pool respectively. Fig. 4 depicts an example of availability model when the cloud center has 2, 1 and 2 PMs in the hot, warm and cold pool respectively. The hot and warm PMs can fail with rate $\theta_h$ and $\theta_w$. To maintain a certain level of availability, upon failure of a hot PM, one warm PM substitutes immediately. Repair units repair broken PMs and return them to the hot, warm or cold pool. Each repair unit repairs a PM with rate $\beta$. If the hot pool is in need of a PM, the repaired PM goes to the hot pool; otherwise it goes to warm and if none of them (i.e., the hot and warm pools) require a PM, the repaired one shots down (i.e., goes to the cold pool). When the number of PMs in either hot or warm pool gets lower than a pre-defined value (e.g., in Fig. 4 lower than 2 and 1) cold PMs are fired up to serve the target pool. The mean start up time is assumed to be $1/SU_c$.

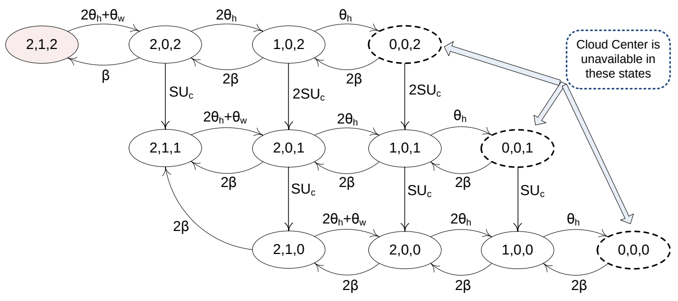


Fig. 4. The availability model for 5 PMs in the pools.

## D. Interaction among sub-models

The interactions among sub-models is depicted in Fig. 5. 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 resource allocation sub-model (RASM). 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 (VMPSM_cold). The resource allocation sub-model (RASM) computes the blocking probability, $BP_q$, which is the input parameter to VM provisioning sub-models. Another type of interaction is between performance model (including RASM and VMPSMs) and the availability model. For each state in the availability model the desired statistics are calculated and then finally the effective value of each performance metrics is obtained. As can be seen, there is an inter-dependency among performance sub-models. This cyclic dependency is resolved via fixed-point iterative method [4] using a modified version of successive substitution approach.

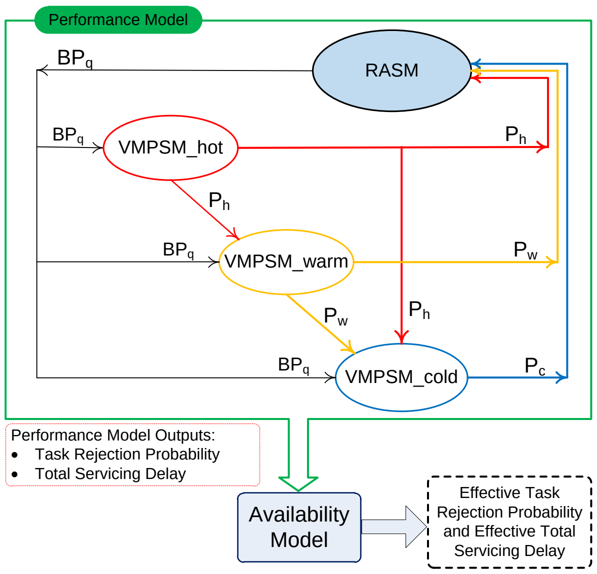

Performance Model
RASM
BPq
VMPSM_hot
Ph
VMPSM_warm
Pw
VMPSM_cold
Pc
Performance Model Outputs:
• Task Rejection Probability
• Total Servicing Delay
Availability Model
Effective Task Rejection Probability and Effective Total Servicing Delay

Fig. 5. Interaction between sub-models.

## III. Numerical validation

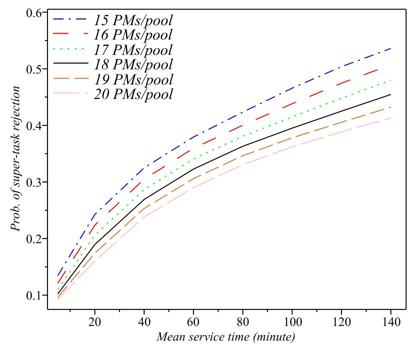
The resulting sub-models have been implemented and solved using Maple 15 from Maplesoft, Inc. [10]. The successive substitution method is continued until the difference between the previous and current value of blocking probability in global queue (i.e., $BP_q$) is less than $10^{-6}$. The integrated model converges to the solution in 10 iterations or less.

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 250 to 600 STs per hour. Mean service time of each task within STs ranges form 20 to 140 minutes. We assume 15 to 20 PMs in each pool and each PM can run 2 VMMs and up to 6 VMs simultaneously. The look-up time for finding a proper PM is assumed to be independent of the number PMs in the pool and the type of pool (i.e., hot, warm and cold). Look-up rate is set to 825 searches per hour. 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 5 to 20 and 15 to 30 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 Mean Time To Failure (MTTF) for hot and warm PMs ($1/\theta_h$ and $1/\theta_w$) are assumed to be in the range of 20 to 1000 and 100 to 3000 hours, respectively. The number of repair units is set to 5 and the range of 1 to 5 hours assumed for mean repair time ($1/\beta$) of a PM. The size of global queue is set to 50 super-tasks (ST).

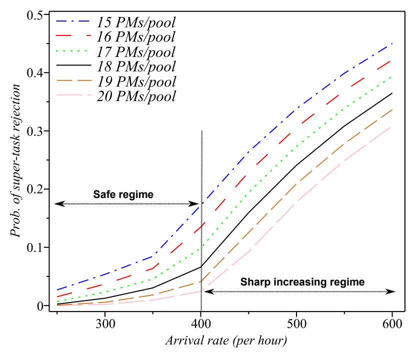
In the first experiment, the results of which is shown in Figs. 6(a) and 6(d), STs have one task and PMs are permitted to run 2 VMMs each of which hosts one VM. Fig. 6(a) shows, at a constant arrival rate (900 STs/hr) and different numbers of PMs in each pool, that by increasing the mean service time the rejection probability increases almost linearly. Also, increasing the system capacity (i.e., the number of PMs in each pool) reduces the rejection probability in a constant manner. Fig 6(d) shows that a longer service time will result in a longer imposed delay on STs. In addition, it can be observed that by increasing the capacity of system (i.e., having more PMs in the pools) the maximum gains occurs at the longest service time (i.e. 140 minutes).

We also determine the two performance metrics under different arrival rates while fixing the average service time at 120 minutes (Figs. 6(b) and 6(e)). One turning point can be noticed in rejection probability curves (Fig. 6(b)) at which rejection probability increases suddenly. Such points (e.g., Fig. 6(b), 400 STs/hour for 20 PMs in each pool) 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. 6(e)): the first regime (i.e., safe regime) is the region in which the cloud providers would like to operate. An 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 in which the total delay increases sharply. Cloud centers should avoid entering such region since the violation of SLA is about to happen. However, since the transient regime is not too narrow, it is possible for cloud center to return to safe 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 super-tasks are experiencing the highest possible amount of delay and this zone is not in the interest of both cloud service providers and cloud customers.

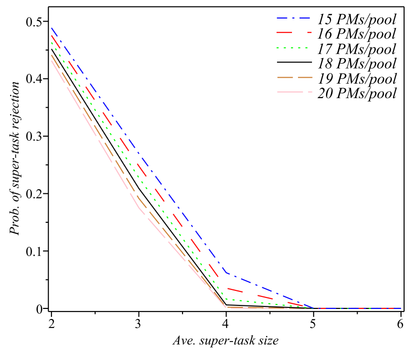
In the last experiment, (Figs. 6(c) and 6(f)) we examine the effect of super-task size on task rejection probability and total delay. Each PM run 2 VMMs and every VMM hosts 3 VMs; a super-task may request up to all of the available VMs in a PM (i.e., all 6 VMs). Note that the aggregate arrival rate is set to 2500 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 500 STs/hr. As can be seen by increasing the size of super-tasks the rejection probability and total delay reduce sharply. Since the size of super-tasks is truncated up to the maximum number of VMs allowed on



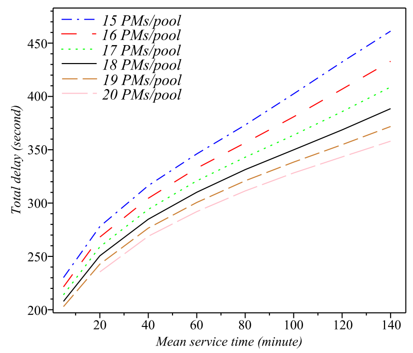
(a) Rejection probability: arrival-rate=900 tasks/hour.



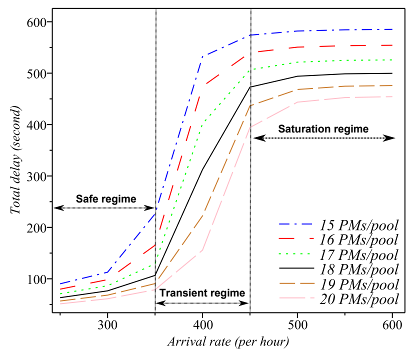
(b) Rejection probability: service-time=120 minute.



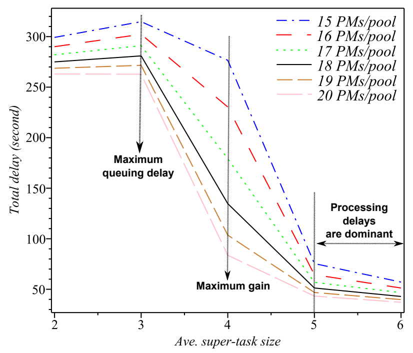
(c) Rejection probability: service-time=120 minute, arrival-rate=2500 tasks/hour.



(d) Total delay on a super-task: arrival-rate=900 tasks/hour.



(e) Total delay on a super-task: service-time=120 minute.



(f) Total delay on a super-task: service-time=120 minute, aggregate arrival-rate=2500 tasks/hour.

Fig. 6. Effective rejection probabilities and total servicing delays under different input parameters: in all experiments #VMMs is 2.

each PM (here, up to 6 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, the maximum delay is happened on size 3 and the maximum gain (i.e., reduction of delay) by increasing the capacity is obtained at size 4. After size 5, the dominant imposed delays on super-tasks are the processing delays (i.e, resource assigning and VM provisioning delays) as opposed to be the queuing delays.

## IV. Conclusions

In this paper, we present a performance and availability model suitable for cloud computing centers using interacting stochastic models. We quantify the effects of variation in super-task arrival, task's service time, super-task size and failure of PMs 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. Cloud providers can examine the amount of improvement on performance criteria by adding extra resources to their center. Therefore, a trade-off analysis can be carried out for capacity planing based on the provided insights in this paper.

## 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. PP, no. 99, p. 1, 2011.

[2] F. Longo, R. Ghosh, V. K. Naik, and K. S. Trivedi, "A scalable availability model for infrastructure-as-a-service cloud," *Dependable Systems and Networks, International Conference on*, pp. 335–346, 2011.

[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] 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.

[5] B. Yang, F. Tan, Y. Dai, and S. Guo, "Performance evaluation of cloud service considering fault recovery," in *First Int'l Conference on Cloud Computing (CloudCom) 2009*, Beijing, China, Dec. 2009, pp. 571–576.

[6] H. Khazaei, J. Mišić, and V. B. Mišić, "Modeling of cloud computing centers using $M/G/m$ queues," in *The First International Workshop on Data Center Performance*, Minneapolis, MN, Mar. 2011.

[7] ——, "Performance analysis of cloud centers under burst arrivals and total rejection policy," in *Globecom 2011: Communications QoS, Reliability, and Modeling Symposium*, Houston, TX, Dec. 2011.

[8] 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.

[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.