Performance of an IaaS Cloud with Live Migration of Virtual Machines

Hamzeh Khazaei, Jelena Mišić and Vojislav B. Mišić
Ryerson University
Toronto, Ontario, Canada
Emails:{hkhazaei,jmisic,vmisic}@scs.ryerson.ca

Abstract—**Cloud centers often use virtualization which requires live migration of virtual machines to improve performance and availability. In this paper, we describe an analytical performance model that measures the important performance indicators, namely, rejection probability and total delay, of a cloud center whilst taking into account live migration of virtual machines. Using the proposed performance model, we show that live virtual machine migration reduces task rejection probability and renders the super-task delay nearly independent of mean service time of individual tasks in a super-task.**

I. Introduction

Cloud computing is rapidly gaining acceptance for making computing resources, including infrastructure, platforms and applications, accessible over the Internet as services [17], [1], [5]. Many cloud solutions rely on virtualization to provide computing resources in the form of Virtual machines deployed on physical servers in the cloud data center. Virtual machines are often migrated between different physical servers in order to save energy, improve resource (i.e., server) utilization and improve performance, as communication between applications running on virtual machines within a single server is often faster than between virtual machines running on different servers. Migration of virtual machines can be done by temporarily suspending the execution of a virtual machine, moving it to another server, and resuming operation there. Recently, technology has become available [18], [2], [15] that allows live migration, during which the virtual machine that is being migrated does not suspend execution, although its execution may become slowed down somewhat due to the migration.

In this paper we analyze the performance of cloud systems with live migration using Markov chain modeling and probabilistic analysis. As the development of a monolithic model is difficult and its analysis may be tedious, we propose to break the model into two interacting stochastic models that correspond to different servicing steps in a cloud system with live task migration. The models are then jointly solved to obtain the final solution.

The rest of the paper is organized as follows. In Section II, we survey related work in cloud performance analysis, in particular the work on live migration of virtual machines. Sections III and IV present our analytical models for resource management and virtual machine provisioning (including live migration), respectively. Section V presents the numerical results, while Section VI summarizes our findings and concludes the paper.

II. Related Work

Cloud computing has attracted considerable research attention, including (lately) performance issues. In [20], a cloud center is modeled as the classic open network with single arrival, from which the distribution of response time is obtained, assuming that both inter-arrival and service times are exponential. Using the distribution of response time, the relationship among the maximal number of tasks, the minimal service resources and the highest level of services was found.

Our earlier work [7], [9] has presented general analytical models that allow for an end-to-end performance analysis of a cloud service, using service availability and provisioning delay as two key QoS metrics. The proposed models reduce the complexity of performance analysis of cloud centers by dividing the overall model into sub-models and then obtaining the overall solution by iteration over individual sub-model solutions. However, these models did not consider live VM migration and different servicing policies.

The performance of cloud computing services for scientific computing workloads was examined in [6]. The authors quantified the presence of Many-Task Computing (MTC) users with real scientific computing workloads and performed an empirical evaluation of the performance of four commercial cloud computing services including Amazon EC2. They have also compared the performance characteristics and cost models of clouds and other scientific computing platforms through trace-based simulation, for general and MTC-based scientific computing workloads.

Live migration and its performance have become an interesting topic only recently, when the technology to accomplish it became available. However, most of the available research results are obtained through experimentation. For example, the authors in [21] studied the live migration efficiency of multiple virtual machines from experimental perspective and investigated different resource reservation methods and migration strategies in the live migration process. Experiments on a virtualized systems were used to construct a performance model using the probabilistic model checker PRISM in [10]. Performance models used to predict migration time from application behavior and resource usage statistics were created through profiling on an experimental Xen-based environment in [19]. A different approach based on CPU scheduling and checkpointing was described in [12], together with experimental data obtained on a prototype. Also, the performance of live migration in two major virtualization environments was reported in [3].

However, analytical performance analyses of live migration are still scarce. This was the motivation for extending our analytical models to include live virtual machine migration.

III. Analytical model: resource management

The cloud center consists of a number of physical servers (PSs) that host a number of virtual machines (VMs), up to a certain predefined limit. User requests (super-tasks) are submitted to a the cloud center through a global finite queue and then processed on the first-in, first-out basis (FIFO). Lack of space in the global queue may result in a super-task being rejected. Each user request is fulfilled by instantiating the specified number of VMs on one or more PSs, in the order of request arrivals. A user may ask for one or more VMs in a single request, thus requests are referred to as super-tasks. We assume that VM instances are created from pre-built or customized disk images [7]; without loss of generality, these pre-built images are assumed to be capable of fulfilling all user requests.

When the request comes to the head of the global queue, the system checks whether there are sufficient resources to service it. Namely, if the number of VMs that can be instantiated at this moment is smaller than the number requested by the user, the super-task will be rejected. We assume that an accepted super-task is broken into individual tasks which are sent to appropriate PSs for instantiating the corresponding VMs. Note that different tasks within the super-task may be provisioned on VMs running on separate PSs. This approach offers flexibility and reduced probability that a super-task will be rejected due to insufficient resources. At the same time, the communication overhead between the tasks in this case may be somewhat higher than in the case where all tasks (and all resulting VMs) from a super-task are provisioned on the same PS.

Our first analytical model, the Resource Management Module, captures the details of queuing at the global admission control and at the PS input queue. The resource assignment process is modeled with a two dimensional continuous time Markov chain shown in Fig. 1(a).

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 [4] with rate λ_{st}. super-tasks are lined up in the global finite queue to be processed in a first-in, first-out (FIFO) basis. Each state of Markov chain is labeled as (i, j), where i indicates the number of super-tasks in queue and j denotes the admission control mode: 'A' for accept mode, and 'R' for reject mode.

Initial state $(0, A)$ corresponds to an empty system which means there is no request in the queue and the arrival request will be accepted. When the value of j is R, it indicates that last PS provisioning has been unsuccessful. The global queue size is set to L_q.

Let P_s be the success probability of finding a PS that can accept the current super-task. We assume that $1/\alpha_s$ is the mean look up delays for finding an appropriate PS in the cloud center. Upon arrival of first super-task, system moves to state $(1, A)$ which means the super-task will be provisioned immediately. Afterward, depending on the upcoming event, three possible transitions can occur:

(a) Another super-task has arrived and system transits to state $(2, A)$ with a rate of λ_{st}.
(b) A PS accepts the super-task so that system moves back to state $(0, A)$ with a rate of $P_s\alpha$.
(c) If none of PSs in the cloud center has enough capacity to accept the super-task, the system will go to state (1,R) with rate $\alpha(1 - P_s)$. In this state, system moves to $(0, A)$ with a rate of α which represents both successful and unsuccessful provisioning. In other words, system will give a second chance to the current super-task at the head of queue for provisioning; in case of successful provisioning system will get back to $(0, A)$ with a rate of $\alpha(1 - P_s)$ and otherwise it moves back to $(0, A)$ with a rate of $P_s\alpha$.

The rejection at (x,R) states is due to insufficient resources at the moment and rejection at states (L_q,A) and (L_q,R) is due to lack of space in the global queue.

The super-task leaves the queue once it receives a decision from the resource management module, 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 delay ($1/\alpha_s$) are exogenous parameters and success probability (P_s) is calculated from the VM provisioning model (which is discussed in detail in the next Section). The resource management model is described by the balance equations

$$\begin{aligned}
\pi(0, A) &= \frac{\alpha(P_s\pi(1, A) + \pi(1, R))}{\lambda} \\
\pi(1, A) &= \frac{\lambda\pi(0, A) + \alpha P_s\pi(2, A) + \alpha\pi(2, R)}{\alpha + \lambda} \\
\pi(1, R) &= \frac{\alpha(1 - P_s)\pi(1, A)}{\alpha + \lambda} \\
&\vdots \\
\pi(L_q, A) &= \frac{\lambda\pi(L_q - 1, A)}{\alpha} \\
\pi(L_q, R) &= \frac{\alpha(1 - P_s)\pi(L_q, A)}{\alpha}
\end{aligned} \tag{1}$$

from which, together with the normalization equation

$$\pi(0, A) + \sum_{i=1}^{L_q} \pi(i, A) + \pi(i, R) = 1 \tag{2}$$

we can obtain steady-state probabilities $\pi_{(i,j)}$s and, eventually, performance metrics such as blocking probability

$$P_{reject} = BP_q + BP_r \tag{3}$$

where the terms correspond to two types of blocking described above, blocking due to a full global queue:

$$BP_q = \pi_{(L_q,A)} + \pi_{(L_q,R)} \tag{4}$$

and blocking due to insufficient resources in the system:

$$BP_r = \sum_{i=1}^{L_q} \frac{\alpha(1-P_s)}{\alpha+\lambda} \pi_{(i,R)} \tag{5}$$

To calculate the mean waiting time in queue, we must first establish the probability generating function (PGF) for the number of super-tasks in the global queue [16] as

$$Q(z) = \pi_{(0,A)} + \sum_{i=1}^{L_q} (\pi_{(i,A)} + \pi_{(i,R)}) z^i \tag{6}$$

The mean number of super-tasks in queue may be obtained as

$$\overline{q} = Q'(1) \tag{7}$$

Applying Little's law [11], the mean waiting time in the queue is given by

$$\overline{wt} = \frac{\overline{q}}{\lambda_{st}(1-P_{reject})} \tag{8}$$

IV. Analytical model: virtual machine provisioning

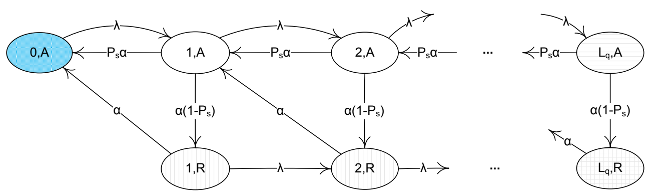
The virtual machine provisioning model captures the instantiation, deployment, live migration and provisioning of VMs on a PS, as well as the actual service time during which the task executes on the assigned VM. The model attempts to provision all VMs from a super-task on a single PS; if this can't be accomplished, it will split the super-task into individual tasks and provisions the corresponding VMs on different PSs. VM migration is undertaken periodically in order to reduce the number of VMs running on the same PS which improves performance, as shown in [8].

Fig. 1(b) shows the virtual machine provisioning model for a PS in the cloud center. The model is represented as a continuous-time Markov chain where states are labeled as (i, j, k): i denotes the number of tasks at the deployment unit (i.e., a PS), j denotes the number of CPU cores on the PS, and k denotes the number of VMs currently running on the PS. Also, β_i and β_o denote the rates of inward and outward migration of VMs, respectively; μ_x denotes the service rate when x VMs are concurrently active on a PS; ϕ denotes the instantiation rate; and λ denotes the super-task arrival rate. Note that the queue size (i.e., deployment unit size) at each PS is set to be to the maximum number of VMs that can be deployed on a PS.

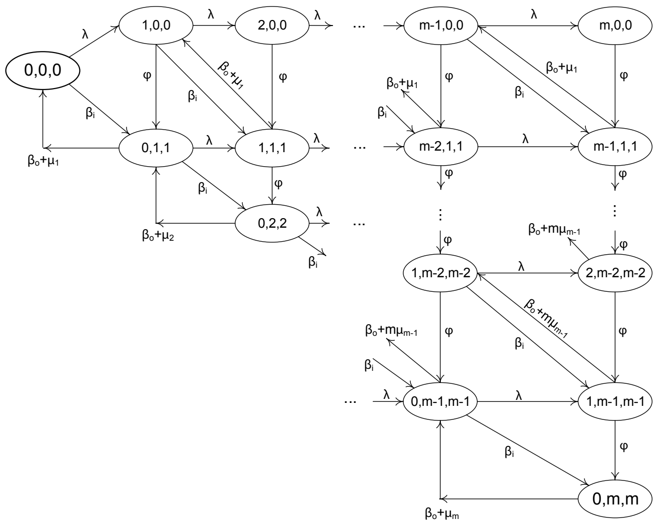
Let ϕ be the rate at which a VM can be deployed on a PS, and let μ_x be the service rate of each VM when x VMs are running. Considering the effect of virtualization degree on the service rate, the total service rate for each PS is the product of the number of running VMs and the service rate μ_x. The arrival rate to each PS in the cloud center is given by

$$\lambda = \frac{\lambda_{st}(1-BP_q)}{N} \tag{9}$$

where N is the number of PSs in the center. The value of BP_q used in (4) is actually obtained from the resource management model described above.



(a) Resource management model.



(b) Virtual machine provisioning.

Fig. 1. Markov chains for the two sub-models.

The state transition in the Markov chain from Fig. 1(b) can occur due to super-task arrival, task instantiation, live migration or service completion. From the initial state $(0, 0, 0)$, the system can move to state $(1, 0, 0)$ with a rate of λ. From $(1, 0, 0)$, system can either transit to $(0, 1, 1)$ with a rate of ϕ (which corresponds to VM instantiation) or, upon the arrival of a task, to state $(2, 0, 0)$ with a rate of λ. From $(0, 1, 1)$, system can either move to $(0, 2, 2)$ with rate β_i (i.e., inward live migration), to $(0, 0, 0)$ with rate $(\mu_1 + \beta_o)$ (i.e., service completion or outward live migration), or (again upon a task arrival) to state $(1, 1, 1)$ with a rate of λ. Using these probabilities, we may describe the system with balance equations and the additional normalization equation, in the manner similar to (1) and (2).

A super-task will be rejected for provisioning if the total number of spare VMs, taking all PSs into account, is lower than the number required by the super-task. The probability that the super-task can't be provisioned can be calculated as

$$P_{na} = \sum_{i=1}^{MBS} p_i \sum_{j=N-i+1}^{N} \binom{N}{j} P_f^j (1-P_f)^{N-j} \tag{10}$$

where p_i is the probability of the super-task having size i and P_f is the probability of having no spare VMs on a PS:

$$P_f = \pi_{(m,0,0)} + \pi_{(m-1,1,1)} + \cdots + \pi_{(1,m-1,m-1)} + \pi_{(0,m,m)} \tag{11}$$

where $\pi_{(i,j,k)}$ is the steady-state probability for the model from Fig. 1(b) to be in the state (i, j, k).

Then, the probability of successful provisioning can be obtained as

$$P_s = 1 - P_{na} \tag{12}$$

This value is used as an input parameter for the resource management model, as explained above.

From the virtual machine provisioning model we can also obtain the mean waiting time at PS queue ($\overline{wt_{PM}}$), mean provisioning time ($\overline{pt}$) and fragmentation of service time due to live migration delay. As the down time of VMs during the live migration is of the order of milliseconds, i.e., 60 milliseconds according to [2], we can safely ignore the last component (pause time) in the calculation of the total delay, which is given by

$$\overline{td} = \overline{wt} + \overline{lut} + \overline{wt_{PM}} + \overline{pt} \tag{13}$$

We note that the steady state probability, P_s, that a super-task gets successfully provisioned is calculated by the virtual machine provisioning model and fed to the resource management model as an input parameter. The resource management model computes the blocking probability, PB_q, which is then fed back to the VM provisioning model to calculate improved probability values. This cyclic dependency is resolved via fixed-point iteration [13] using a modified version of successive substitution method.

V. Numerical validation

The analytical model described above has been implemented and solved using Maple 15 from Maplesoft, Inc. [14]. We have calculated two most important performance metrics, namely rejection probability and total delay imposed on super-tasks; the other parameters were set as shown in Table I.

TABLE I
Parameter setting for the analytical modeling.

Parameters	Mean Value(s)
VM service time	[60,90,···,210] (min)
Max # of VMs per PS	10
Optimum # of VMs per PS	4
# of VMMs on each PS	2
VM start-up	3 (min)
VM live migration	2.5 (min)
Live migration down time	60 (millisec)
PS Look-up time	2 (sec)
Global queue size	50 (super-task)
Super-task (super-task) size	6 (task)
Arrival rate	[10,20,···,60] (super-task/min)

The look-up time for checking the spare capacity of a PS was set to 30 searches per minute, and assumed to be independent of the number of PSs in the cloud center. Each PS can run up to 10 VMs and the optimum number of running VMs was 4. Here 'optimum' is meant as the limit below which there is no degradation of VM throughput; above that value, we have used the results from [8] to calculate the performance penalty (i.e., degradation of VM service rate) with respect to the virtualization degree (i.e., the number of VMs running concurrently on the same PS). We have also assumed geometric probability distribution for the super-task sizes.

In this environment, we have examined the effects of varying task service time and arrival rate on rejection probability and total delay. As can be seen in Fig. 2(a), rejection probability increases almost linearly as the service time get longer in the system without live migration. When live migration is introduced, variation of super-task service time does not significantly affect the rejection probability, as the live VM migration acts as an effective load balancing method that prevents PSs from reaching their full capacity. Increasing service time has virtually no effect on the total delay, as could be expected due to nearly constant rejection rate in the system with live migration, Fig. 2(a). However, total delay decreases rapidly in the system with live migration, but only because many super-tasks are rejected; those that manage to get through will get into service slightly faster than in the system with live migration.

By increasing the rate of super-task arrivals, rejection probability increases both with and without live migration (Fig. 3(a)). Live migration does mitigate the rejection slightly by balancing the workload among PSs. In both scenarios total delay decreases with the increase in super-task arrival rate, 3(a). At higher arrival rates, new arrivals are quickly rejected due to lack of capacity; this leads to almost empty queues at both system and PS level, which in turn reduces the queuing delay for some of the new arrivals, in particular after some of the tasks currently executing finish and release resources.

VI. Conclusions

In this paper we have presented an analytical model suitable for performance evaluation of IaaS cloud centers with live migration of VMs. We have examined the effects of super-task arrival rate and task service time on rejection probability and total delay for super-tasks. Our future work will focus on optimal policies for live task migration with the goal of optimizing relevant parameters such as energy consumption and cloud center performance.

References

[1] An amazon.com company. Amazon Elastic Compute Cloud, Amazon EC2. Website, Last accessed October 2012. http://aws.amazon.com/ec2.

[2] Citrix Systems, Inc. Xen Hypervisor. Website, Last accessed January 2013. http://www.xen.org.

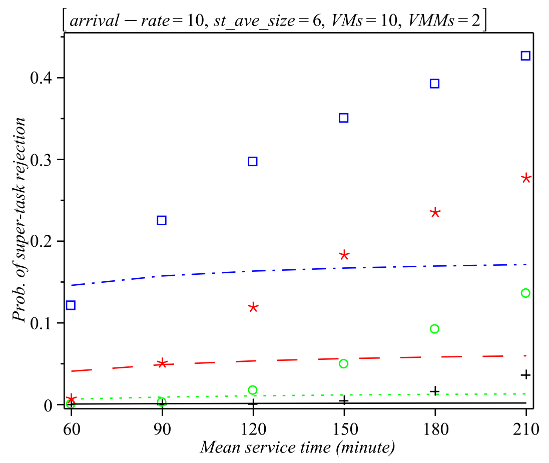
[3] X. Feng, J. Tang, X. Luo, and Y. Jin. A performance study of live VM migration technologies: VMotion vs XenMotion. In *Communications and Photonics Conference and Exhibition*, 2011.

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

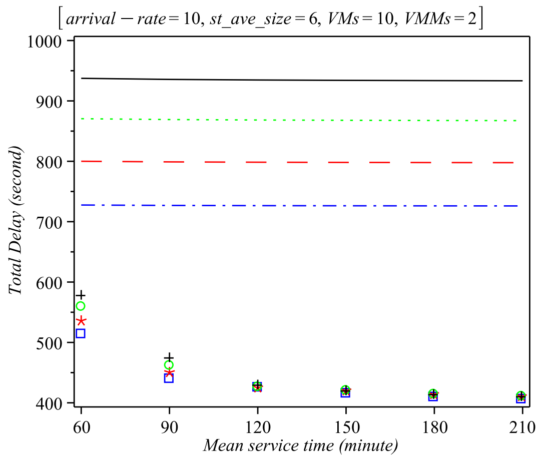
[5] IBM. IBM Cloud Computing. Website, Last accessed July 2012. http://www.ibm.com/ibm/cloud/.

[6] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema. Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(6):931–945, June 2011.

[7] H. Khazaei, J. Mišić, and V. B. Mišić. A fine-grained performance model of cloud computing centers. *IEEE Transactions on Parallel and Distributed Systems*, 99(PrePrints):1, 2012.

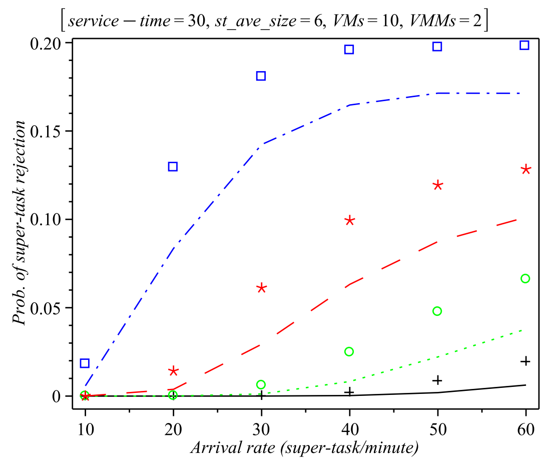


(a) Rejection probability.

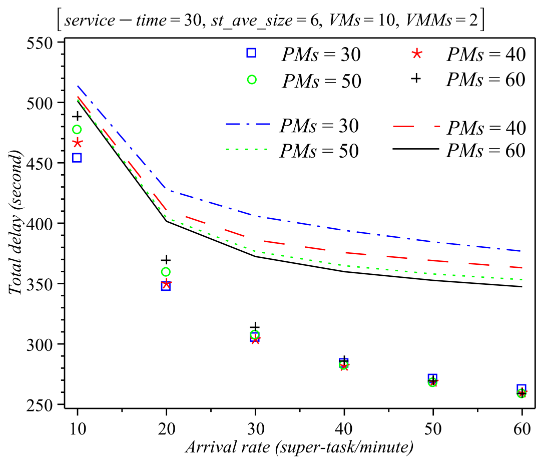


(b) Super-task delay.

Fig. 2. Rejection probability and delay as functions of mean task service time. Lines: system with task migration; symbols: system without task migration.



(a) Rejection probability.



(b) Super-task delay.

Fig. 3. Rejection probability and delay as functions of mean task arrival rate. Lines: system with task migration; symbols: system without task migration.

[8] H. Khazaei, J. Mišić, and V. B. Mišić. Performance of cloud centers with high degree of virtualization under batch task arrivals. *IEEE Transactions on Parallel and Distributed Systems*, 99(PrePrints):1, 2012.

[9] 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*, 99(PrePrints), 2012.

[10] S. Kikuchi and Y. Matsumoto. Performance modeling of concurrent live migration operations in cloud computing systems using PRISM probabilistic model checker. In *IEEE Int. Conf. Cloud Computing (CLOUD)*, pages 49–56, 2011.

[11] L. Kleinrock. *Queueing Systems, Volume 1, Theory*. Wiley-Interscience, 1975.

[12] W. Liu and T. Fan. Live migration of virtual machine based on recovering system and CPU scheduling. In *6th IEEE Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, volume 1, pages 303–307, 2011.

[13] V. Mainkar and K. S. Trivedi. Sufficient conditions for existence of a fixed point in stochastic reward net-based iterative models. *Software Engineering, IEEE Transactions on*, 22(9):640–653, September 1996.

[14] Maplesoft, Inc. Maple 15. Website, Last accessed March 2011. http://www.maplesoft.com.

[15] Microsoft, Inc. Microsoft Hyper-V Server 2012. Website, Last accessed January 2013. http://www.microsoft.com/en-us/server-cloud/hyper-v-server.

[16] H. Takagi. *Queueing Analysis*, volume 1: Vacation and Priority Systems. North-Holland, 1991.

[17] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1), June 2009.

[18] VMware, Inc. VMware vSphere Hypervisor. Website, Last accessed January 2013. http://www.vmware.com/products/vsphere-hypervisor.

[19] Y. Wu and M. Zhao. Performance modeling of virtual machine live migration. In *IEEE Int. Conf. Cloud Computing (CLOUD)*, pages 492–499, 2011.

[20] K. Xiong and H. Perros. Service performance and analysis in cloud computing. In *IEEE 2009 World Conference on Services*, pages 693–700, February 2009.

[21] K. Ye, X. Jiang, D. Huang, J. Chen, and B. Wang. Live migration of multiple virtual machines with resource reservation in cloud computing environments. In *IEEE Int. Conf. Cloud Computing (CLOUD)*, pages 267–274, 2011.