Performability Analysis of a Cloud System

Xiwei Qiu, Peng Sun, Xun Guo, and Yanping Xiang
Collaborative Autonomic Computing Laboratory, School of Computer Science and Engineering
University of Electronic Science and Technology of China, Chengdu, China
Email: qiu_uestc@hotmail.com, sleisurep@163.com, guoxun@uestc.edu.cn, yanping_xiang@163.com

***Abstract*—Cloud computing has recently emerged as an important filed with numerous novel features, particularly, large-scale resource integration and virtualized resource provisioning. Since a cloud system essentially aims at service-oriented computing, service performance becomes the primary metric that needs analyzing in detail. However, in a realistic scenario, operation of virtual machines (VM) may be interrupted by random resource failures. This demonstrates that service performance is indeed affected by resource reliability. Thus, connecting performance and reliability is essential for making more precise evaluation. In this paper, we present a theoretical modeling approach for performability analysis of cloud services and the cloud system. This flexible modeling approach first builds two tractable submodels that consider an important correlation factor (i.e., available resource capacity that is not only decided by reliability but also has a significant effect on performance) to ensure the required fidelity. Then, a Bayesian method is applied to connect the submodels, which can make our performability model more scalable. In contrast to a monolithic modeling method, our approach that combines interacting submodels can effectively reduce computing complexity for a large-scale cloud system. Numerical examples are illustrated.**

***Index Terms*—Cloud computing, correlation modeling, performance, reliability**

I. Introduction

In recent years, cloud computing has emerged as a new computing paradigm with numerous novel characteristics, such as large-scale resource sharing, virtualized resource provisioning, and service-oriented computing [1]. Many industries focus on developing cloud systems for real-world environments, e.g., Amazon Elastic compute cloud (EC2), Google cloud platform, and Microsoft Azure. These cloud systems have a cloud controller (CC) designed to efficiently manage virtual resource pools. Meanwhile, the use of virtual machines (VM) enables numerous new service modes, including software as a service (SaaS), infrastructure as a service (IaaS), and platform as a service (PaaS) [2].

For providing efficient and stable cloud services, performance becomes an important metric that must be analyzed in detail. Many recent researches have studied cloud service performance for various application scenarios. For example, Zhang *et al.* [3] investigated the cloud network and developed a predict model to analyze performance of web services in advance. Khazaei *et al.* [4] presented an iterative performance model to evaluate service performance of a cloud computing center. Bruneo [5] applied stochastic reward nets to construct an analytical performance model for IaaS clouds. Although these studies systemically investigated cloud service performance, it is inadequate for realistic environments since they do not consider an important correlated factor, i.e., resource reliability.

In practice, cloud service performance is significantly affected by random resource failures. For example, hardware failures of a physical server inevitably lead to interruption of co-located VMs hosted on the server. Moreover, excessive failed servers can result in inadequate resource capacity of the resource pool, which potentially implies more waiting time of user requests and further incurs a higher possibility of discarding new arrival requests due to the full of the request queue. Therefore, there exists an important correlation between performance and reliability, and these metrics should not be considered separately. Many studies about reliability in the literature also demonstrate that performance is indeed a resulting attribute of reliability [6]-[8].

For combining performance and reliability, Meyer *et al.* [9] proposed the notion of performability, and corresponding theoretical models were subsequently studied. For example, Yang *et al.* [10] evaluated expected response delay and execution time of cloud services based on stochastic models in fault recovery. Tokuno and Yamada [11] investigated hardware and software failures to build a performability model for a parallel computing system. Kim *et al.* [12] considered an IaaS cloud scenario consisting of datacenters, hosts, and VMs, and used CloudSim to measure the performability of the IaaS cloud. Tamura and Yamada [13] used open source software (OSS) to construct a cloud computing environment, and mainly adopted a Gaussian jump diffusion process to analyzed the performability of the cloud OSS. However, these models do not consider an important cloud component, i.e., the cloud controller, the brain of the cloud system. In fact, all requests must be first processed by the CC and the service ability of the CC inevitably has a significant effect on performance. Thus, this important factor should be fully captured for achieving a more precise evaluation of cloud performability.

In this paper, we present a theoretical modeling approach for analyzing the performability of cloud services and the cloud system. We assume the cloud system adopts a more flexible and efficient three-phase failure recovery mechanism to handle occurred server failures. Then, the corresponding reliability submodel is proposed for analyzing the probability distribution of available capacity of cloud resources. We also propose a more realistic performance submodel, which fully takes the CC service time and the VM service time into account.

978-1-4673-8590-9/15/$31.00 ©2015 IEEE

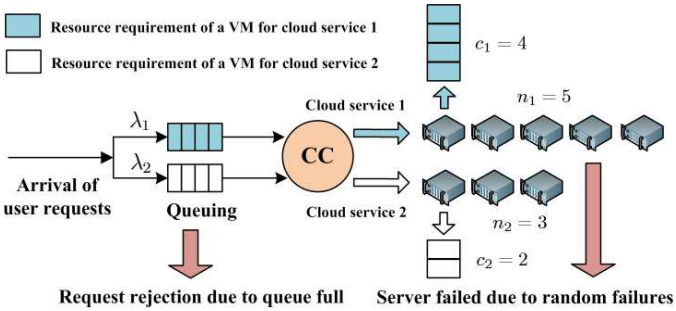

Fig. 1. A topological structure of the cloud system with two cloud services.

Markov models are mainly used to depict the performance and reliability submodels, and then a Bayesian approach is applied to connect the submodels for precisely obtaining an overall evaluation of the performability of cloud services and the cloud system.

The remainder of the paper is organized as follows. In Section II, we describe a topology structure of the cloud system with multiple cloud services. Section III presents the performability modeling approach using interacting stochastic models. Section IV illustrates numerical examples. Finally, we conclude the paper in Section V.

II. Cloud Topological Structure

In principle, the cloud system must have essential capabilities of unified access interface, large-scale resource sharing, and centralized job scheduling. To achieve such distinctive capabilities, the cloud system must possess two elementary components: the CC and the cloud resource pool. The CC receives and parses user requests to initiate and deploy VMs for delivering corresponding cloud services to the users. The cloud resource pool is virtualized cloud infrastructure that hosts VMs to support various cloud services.

In general, the cloud system needs to determine a resource assignment strategy for a cloud service. Suppose that the CC assigns n homogeneous physical servers to the cloud service, and the resource capacity (i.e., CPU, memory, and disk) of each server can support c VMs running simultaneously. Thus, the cloud service can serve a maximal number of $x = c{\cdot}n$ users in parallel. However, in realistic scenarios, physical servers may be down due to random failures, which implies that the available resource capacity for supporting the cloud service is indeed a random variable.

Fig. 1 illustrates a topological structure of the cloud system with two cloud services. Take cloud service 1 shown in the figure as an example, the CC assigns five severs to the cloud service. With each server supports four co-located VMs, the cloud service can serve 20 users simultaneously. Once a server is failed due to some failures, the available resource capacity of the cloud service is correspondingly reduced, which inevitable leads to the decrease of the service performance. To effectively analyze this situation, we present a performability model to combine performance metric with random failures and recovery, which is described in the following Section III.

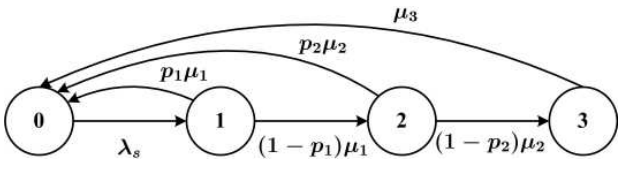

Fig. 2. Markov model for the recovery process of the failed server.

III. Analysis of Cloud Performability

A. Resource Reliability Modeling

In this paper, we assume that physical servers assigned to a cloud service are homogeneous. This is a reasonable division of cloud infrastructure as it can significantly improve the management efficiency of the CC. Meanwhile, servers assigned to different cloud services can be heterogeneous. Note that the CC is the most critical element that ensures the operation of the cloud system, and thus it usually has multiple redundant copies to guarantee its reliability. According to this situation, we assume that the CC is fully reliable.

Many kinds of repair actions can be applied to restore a server failure. In general, a hierarchical recovery mechanism consisting of multiple kinds of repair actions is more flexible and efficient than a single kind of repair actions [14]. In this paper, the hierarchical recovery mechanism is assumed to have three typical kinds of repair actions, which can be expressed as a three-phase recovery process. In the first phase, a *rapid repair action* with a relatively short mean repair time $1/\mu_1$ is used according to the phenomenon of the failure and previous experience of recovery, which has a probability of p_1 to successfully remove the failure. Once it is failed to remove the failure, the process enters the next phase that adopts a *diagnostic repair action*. This kind of repair action tries to find the cause of the failure, and then applies a more pertinent repair action based on the diagnostic result. Similarly, this phase also has a mean repair time $1/\mu_2$ and a probability p_2 to remove the failure. The most serious situation is that the failure cannot be remove in the first and second phases, and the CC has to apply a *complete repair action* (with the mean time of $1/\mu_3$) to fully restore the failed server, which is usually time-consuming. Without loss of the generality, the inequalities $\mu_1 > \mu_2 > \mu_3$ and $p_1 < p_2$ are held. Thus, the recovery process of the failed server can be modeled as a Markov model, which is shown in Fig. 2.

In this paper, we assume that the failure time and the repair time are both exponentially distribute [15]. The state 0 represents the initial state of the server that no failure exists. The model translates from 0 to 1 with the failure rate λ_s, which means that a server failure occurs. The states 1, 2, and 3 represent the first, second, and final phases of the recovery process, respectively. Denote q_k as the steady probability for the Markov model stays at state k $(k = 0, 1, 2, 3)$, which can be obtained by solving the following *Chapman-Kolmogorov* equations:

$$\lambda_s q_0 = p_1\mu_1 q_1 + p_2\mu_2 q_2 + \mu_3 q_3 \tag{1}$$

$$\mu_1 q_1 = \lambda_s q_0 \tag{2}$$

$$\mu_2 q_2 = (1 - p_1)\mu_1 q_1 \tag{3}$$

$$\mu_3 q_3 = (1 - p_2)\mu_2 q_2 \tag{4}$$

$$\sum_{k=0}^{3} q_k = 1. \tag{5}$$

Solving (1)-(5), the probability that the server is available can be obtained as $p_a = q_0$. Then, for the cloud service with n assigned servers, the number of available servers is a discrete random variable denoted by Y, which takes values from 0 to n. Since these servers are homogeneous and independent, the probability distribution function (pdf) of Y can be derived as

$$p_Y(y) = \Pr(Y = y) = C_n^y (p_a)^y (1 - p_a)^{n-y}. \tag{6}$$

Obviously, the available resource capacity (i.e., the maximal number of VMs that can be hosted on all available servers simultaneously) is also a random variable determined by Y. Denote it as X and the equation $X = c \cdot Y$ is satisfied. From (6), the pdf of X can be easily written as

$$p_X(x) = p_Y(x/c) \quad x = 0, c, 2c, \ldots, nc. \tag{7}$$

Although we assume n servers supporting the same cloud service are homogeneous, our reliability submodel can also be extended for more complicated scenarios that all servers are heterogeneous. For the i-th server ($1 \leq i \leq n$), the maximal number of co-located VMs can be expressed as a discrete random variable Z_i. From (1)-(5), its probability distribution is written as

$$p_{Z_i}(c_i) = \Pr(Z_i = c_i) = p_{a_i} \tag{8}$$

$$p_{Z_i}(0) = \Pr(Z_i = 0) = 1 - p_{a_i}. \tag{9}$$

Note that the availability states of these heterogeneous servers are independent, thus the probability distribution of X can be derived as

$$p_X(x) = \sum_{\substack{c_1, c_2, \ldots, c_n \\ c_1 + \ldots + c_n = x}} \left(\prod_{i=1}^{n} p_{Z_i}(c_i) \right). \tag{10}$$

B. Service Performance Modeling

Most existing work does not consider the service time of the CC. However, in realistic scenarios, the service capability of the CC is indeed critical since it decides the efficiency of parsing user requests. To remedy this lack, our performance model focuses on capturing not only the VM service time but also the CC service time.

As shown in Fig. 1, the CC keeps multiple separate queues for different cloud services. These queues are assumed to be processed by the CC in parallel. Requests in the same queue are served on the First-Come-First-Serve (FCFS) discipline. In this paper, the following assumptions are made for the performance model of a cloud service.

A1) The arrival of requests for the cloud service follows a Poisson process with an arrival rate λ. Limitation of the request queue length is L. When a new request arrives, it is discarded if the queue is full.

A2) The service times of the CC and a VM are exponentially distributed with service rates μ_c and μ_v, respectively.

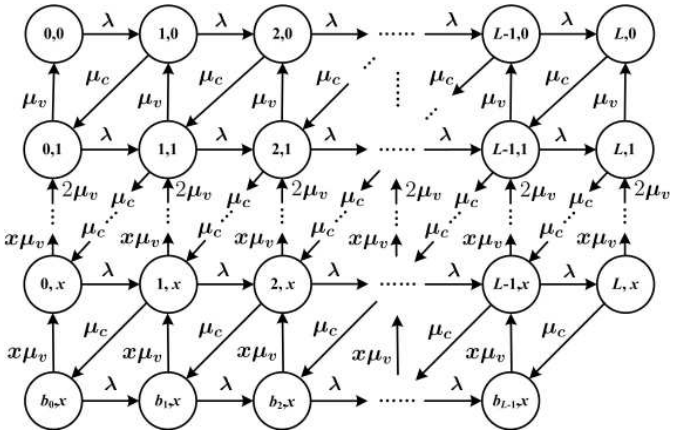

Fig. 3. Performance model for the cloud service.

The service process of a user request mainly includes two operations: a *parse operation* and a *serving operation*. The parse operation is executed by the CC, which translates the request to a service command and tries to find a server with adequate resources (e.g., CPU, memory, and disk) to host a new VM. If all available resources have been occupied, this request has to wait in the queue until some occupied resources are released, which also makes the other requests in the queue cannot be served by the CC (i.e., the CC becomes 'busy' state). After the service command is sent to a server, the subsequent serving operation can be operated by the server, which initiates and deploys a new VM for delivering the cloud service to the user. Once the VM successfully serves the user, the occupied resources are immediately released and can be used for serving the next request.

We present the performance model for the cloud service, as shown in Fig. 3. States of the model in Fig. 3 are indexed by (i, j), where i denotes the number of requests in the queue ($i = 0, 1, \ldots, L, b_0, b_1, \ldots, b_{L-1}$) and j represents the number of running VMs ($j = 0, 1, \ldots, x$). Note that b_k represents that one request has been parsed by the CC but cannot be served due to inadequate resources, and there also are k requests in the queue waiting for the process of the CC. We briefly describe the state transitions in the model:

1) For the situation that a new request arrives, if the queue is not full, i.e., $i \neq L$ and $i \neq b_{L-1}$, the new request makes the model move from (i, j) to state $(i+1, j)$ with the arrival rate λ, where $b_k + 1 \equiv b_{k+1}$.
2) For the situation that a request is parsed by the CC, if $j < x$, which means that there still exists adequate resources can be used for hosting a new VM, the model moves from (i, j) to $(i - 1, j + 1)$ with the service rate μ_c. If $j = x$, the parsed request makes the CC become busy state, and the model translates

from (i, x) to (b_{i-1}, x) with the service rate μ_c.

3) For the situation that a request is complete, if $i \neq b_k$, when j $(1 \leq j \leq x)$ VMs are simultaneously running in the resource pool, the model moves from (i, j) to $(i, j-1)$ with the exit rate $j \cdot \mu_v$. If $i = b_k$, the model move from (b_k, x) to (k, x) with the exist rate $x \cdot \mu_v$, this is because the first request waiting in the queue has already been parsed by the CC and thus it can be served immediately.

Denote $\pi_{i,j}$ as the steady probability for the cloud service stay at state (i, j), which can be obtained by solving the following *Chapman-Kolmogorov* equations:

$$\lambda\pi_{0,0} = \mu_v\pi_{0,1} \tag{11}$$

$$\mu_c\pi_{L,0} = \lambda\pi_{L-1,0} + \mu_v\pi_{L,1} \tag{12}$$

$$(\lambda + x\mu_v)\pi_{b_0,x} = \mu_c\pi_{1,x} \tag{13}$$

$$x\mu_v\pi_{b_{L-1},x} = \lambda\pi_{b_{L-2},x} + \mu_c\pi_{L,x} \tag{14}$$

$$(\mu_c + x\mu_v)\pi_{L,x} = \lambda\pi_{L-1,x} \tag{15}$$

$$(\lambda + x\mu_v)\pi_{0,x} = \mu_c\pi_{1,x-1} + x\mu_v\pi_{b_0,x} \tag{16}$$

$$(\lambda + \mu_c)\pi_{i,0} = \lambda\pi_{i-1,0} + \mu_v\pi_{i,1},\ i = 1, \ldots, L-1 \tag{17}$$

$$(j\mu_v + \mu_c)\pi_{L,j} = \lambda\pi_{L-1,j} + (j+1)\mu_v\pi_{L,j+1}, \quad j = 1, 2, \ldots, x-1 \tag{18}$$

$$(\lambda + j\mu_v)\pi_{0,j} = \mu_c\pi_{1,j-1} + (j+1)\mu_v\pi_{0,j+1}, \quad j = 1, 2, \ldots, x-1 \tag{19}$$

$$\begin{aligned}(\lambda+\mu_c + j\mu_v)\pi_{i,j} &= \lambda\pi_{i-1,j} \\ &+ \mu_c\pi_{i+1,j-1} + (j+1)\mu_v\pi_{i,j+1}, \\ &i = 1, 2, \ldots, L-1, j = 1, 2, \ldots, x-1\end{aligned} \tag{20}$$

$$(x\mu_v + \mu_c)\pi_{i,x} = \mu_c\pi_{i+1,i-1} + \lambda\pi_{i-1,x} + x\mu_v\pi_{b_i,x}, \quad i = 1, 2, \ldots, L-1 \tag{21}$$

$$(x\mu_v + \lambda)\pi_{b_i,x} = \lambda\pi_{b_{i-1},x} + \mu_c\pi_{i+1,x},\ i = 1, \ldots, L-1 \tag{22}$$

$$\sum_i \sum_j \pi_{i,j} = 1 \tag{23}$$

Given the condition that available resources supporting x VMs at most, the conditional probabilities that the CC remains 'busy' state and the resource pool is 'saturation' state (i.e., all available resources have been occupied) can be respectively obtained as

$$p_{busy}(x) = \sum_{k=0}^{L-1} \pi_{b_k,x} \tag{24}$$

$$p_{sat}(x) = \sum_{i=0}^{L} \pi_{i,x} + \sum_{k=0}^{L-1} \pi_{b_k,x}. \tag{25}$$

Moreover, the conditional probability that a new request is discarded due to the full of the queue is written as

$$p_{dis}(x) = \sum_{j=0}^{x} \pi_{L,j} + \pi_{b_{L-1},x}. \tag{26}$$

The first and second output indices (i.e., $p_{busy}(x)$ and $p_{sat}(x)$) can be used to analyze the states of the CC and the resource pool, respectively. According to this valuable information, the cloud provider can rationally determine the limitation of the queue and the number of assigned servers. The third index $p_{dis}(x)$ is an important measurement that describes the performance of the cloud service.

C. Evaluation of Performability

Now, we can use Markov reward models to evaluate the performability of the cloud service. For each state x, it has a probability $p_X(x)$ and a reward value $r(x)$ (it could be $p_{busy}(x)$, $p_{sat}(x)$ or $p_{dis}(x)$). From (7), use a Bayesian approach to remove the parameter x, the expected reward value is obtained as

$$r = \sum_x r(x)p_X(x). \tag{27}$$

From (27), the expected indices p_{busy}, p_{sat}, and p_{dis} can be obtained. In this paper, we take the expected throughput as the performability metric of the cloud service, which are expressed as

$$\varphi = \lambda(1 - p_{dis}). \tag{28}$$

For the cloud system with multiple cloud services, its performability is defined as the efficient service ratio of all requests. Suppose the cloud system runs M different cloud services. Although VMs supporting these cloud services may have different configuration (e.g., software, OS, and resource requirement), the cloud isolation technology effectively guarantees non-interfering runs of VMs, which implies that service processes of user requests can be treated as independent. Thus, the overall performability of the cloud system can be calculated by

$$\rho = \frac{\sum_{m=1}^{M} \varphi_m}{\sum_{m=1}^{M} \lambda_m} \tag{29}$$

where φ_m and λ_m are the expected throughput and the arrival rate of the m-th $(m = 1, 2, \ldots, M)$ cloud service, respectively.

IV. Numerical Examples

Consider cloud service 1 shown in Fig. 1 as an example. The cloud service has five homogeneous servers and each server can host four VMs simultaneously. Suppose the failure rate of each server is $\lambda_s = 0.0009\ s^{-1}$. The repair rates of the first, second and final phases are $\mu_1 = 0.01\ s^{-1}$, $\mu_2 = 0.006\ s^{-1}$, and $\mu_3 = 0.001\ s^{-1}$, respectively. The first phase and the second phase have probabilities of $p_1 = 0.75$ and $p_2 = 0.85$ to successfully remove the failure, respectively. From (1)-(5), the probability that a server is unavailable due to random failures and recovery can be calculated as $1 - p_a = 0.1389$. Then, substitute this value into (6), the pdf of the number of available servers can be furthermore obtained, as shown in Fig. 4. Now, given a specific resource capacity $x = c \cdot n$, the conditional indices can be calculated by using our performance model. Suppose the limitation of the queue for the cloud service is $L = 10$ and the arrival rate of the user requests is $\lambda = 2.6\ s^{-1}$. The CC service rate and the VM service

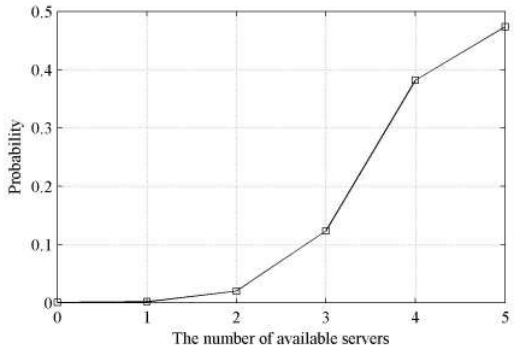

Fig. 4. The probability distribution of the number of available servers.

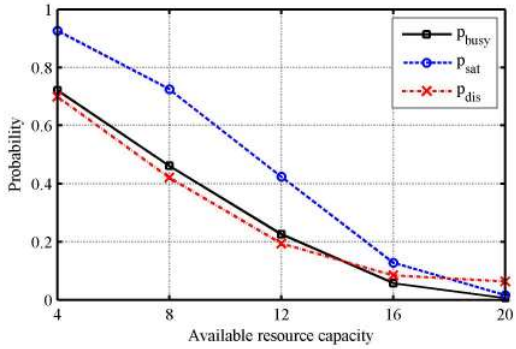

Fig. 5. Conditional reward values as functions of available resource capacity.

rate are $\mu_c = 2.8\ s^{-1}$ and $\mu_v = 0.2\ s^{-1}$, respectively. From (24), (25), and (26), we can calculate the corresponding reward values $p_{busy}(x)$, $p_{sat}(x)$, and $p_{dis}(x)$, as shown in Fig. 5. Note that there also exists a special condition of $x = 0$ indicating that all the resources are unavailable due to server failures. The conditional indices for $x = 0$ are set as $p_{busy}(0) = p_{sat}(0) = p_{dis}(0) = 1$, which represents that no requests are completed.

Finally, use a Bayesian approach to remove the parameter x, the corresponding expected values are written as $p_{busy} = 0.0628$, $p_{sat} = 0.1247$, and $p_{dis} = 0.0952$. According to these valuable information, the cloud provider can comprehensively evaluate the working state of the CC, the usage of the resource pool, and the throughput of the cloud service. From (28), the performability metric of the cloud service is $\varphi_1 = 2.3525\ s^{-1}$. To evaluate the performability of the cloud system, we assume cloud service 2 in Fig. 1 has the parameters of $\lambda' = 1.8\ s^{-1}$, $\mu'_c = 2.2\ s^{-1}$, $\mu'_v = 0.3\ s^{-1}$, and $L' = 10$. The performability metric of cloud service 2 can also be calculated as $\varphi_2 = 1.3320\ s^{-1}$. Then, from (29), the performability metric of the cloud system (i.e., the request completion ratio) is written as $\rho = 83.47\%$.

To verify analytical results obtained by the proposed cloud performability model, a simulation program based on the Monte Carlo method has been developed, which is designed to simulate the service process of 2000 requests of a cloud service considering the random change of the performance

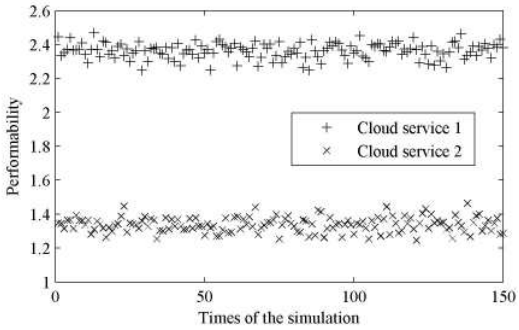

Fig. 6. Performability estimation of 150 runs of the simulation program.

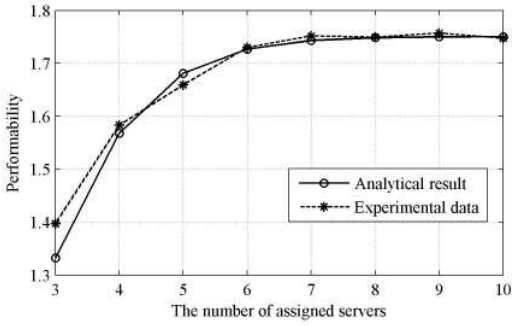

Fig. 7. Performability vs. the number of servers at $\mu_v = 0.3\ s^{-1}$.

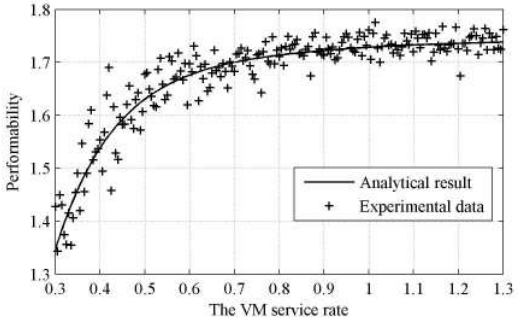

Fig. 8. Performability vs. the VM service rate at $n = 3$.

caused by random server failures and recovery. The simulation program runs 150 times for calculating the performability of cloud service 1 and 2, as shown in Fig. 6. One can see that the simulation results for cloud service 1 and 2 fluctuate around the corresponding theoretical values (i.e., $\varphi_1 = 2.3253\ s^{-1}$ and $\varphi_2 = 1.3320\ s^{-1}$ calculated from our analytical model), which witnesses that the proposed performability model is justified.

In principle, the performability of a cloud service is significant affected by the number of assigned servers n and the VM service rate μ_v. Take cloud service 2 in Fig. 1 as an example, the changes of its performability caused by different n and μ_v are depicted in Fig. 7 and Fig. 8, respectively. It can be observed that the performability is dramatically improved at first, but the effects on improving the performability become

gradually slight with the increase of n and μ_v. Note that the performability tends to remain constants from $n = 6$ in Fig. 7 and $\mu_v = 0.09\ s^{-1}$ in Fig. 8. These can be treated as estimation of optimal solutions of n and μ_v, which implies the maximum performability of the cloud service is almost achieved.

V. Conclusion

In this paper, we systemically studies a theoretical model for evaluating the performability of cloud services and the cloud system. We first analyze the reliability of cloud resources (i.e., physical servers) for the cloud system adopting a three-phase failure recovery mechanism, which is more flexible and efficient than a single kind of repair actions. Then, the CC service time and the VM service time are fully taken into account to establish a more realistic performance model. Finally, the evaluation of the performability metric is obtained by using a Bayesian method.

The numerical examples in this work illustrated the procedures for modeling, analyzing and evaluating the performability of cloud services and the cloud system. We quantify the effects of variations in the number of assigned servers and the VM service rate on the performability of cloud services, which also show that the proposed model can effectively help the cloud provider estimate optimal solutions of the number of servers and the VM service rate. The analytical results calculated by our performability model are very close to the simulation results, which can effectively validate our performability model.

The presented model can also be extended to a more complicated public cloud scenario with heterogeneous servers. According to the analysis of the theoretical model, optimal resource scheduling strategies for multiple cloud services can be further studied. Generally speaking, such an optimal resource scheduling strategy in a large-scale cloud scenario can be described as a multi-objective optimization problem, which is usually an NP complete problem, and thus needs heuristic algorithms to find approximately optimal solutions. The design of resource scheduling strategies for optimizing the performability of multiple cloud services is an important topic that will be studied in our future work.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grant 61170042, in part by the Fundamental Research Funds for the Central Universities under Grant ZYGX2011Z001, and in part by the Innovational Team Project of Sichuan Province under Grand 2015TD0002.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 3, pp. 599-616, Jun. 2009.

[2] M. N. Huhns, and M. P. Singh, "Service-oriented computing: Key concepts and principles," *IEEE Internet Computing*, vol. 9, no. 1, pp. 75-81, Jan. 2005.

[3] Y. Zhang, Z. Zheng, and M. R. Lyu, "An online performance prediction framework for service-oriented systems," *IEEE Transactions on System, Man, and Cybernetics: Systems*, vol. 44, no. 9, pp. 1169-1181, Sept. 2014.

[4] H. Khazaei, J. Misic, V. B. Misic, "A fine-grained performance model of cloud computing centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 11, pp. 2138-2147, Nov. 2013.

[5] D. Bruneo, "A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems," *IEEE Transactions on Parallel Distributed Systems*, vol. 25, no. 3, pp. 560-560, Mar. 2014.

[6] Y. S. Dai, Y. Pan, and X. K. Zou, "A hierarchical modeling and analysis for grid service reliability," IEEE Transactions on Computer, vol. 56, no. 5, pp. 681-691, May. 2007.

[7] K. Vishwanath, and N. Nagappan, "Characterizing cloud computing hardware reliability," in *Proceeding of the 1st ACM Symposium on Cloud Computing*, 2010, pp.193-204.

[8] E. Bauer, and R. Adams, *Reliability and availability of cloud computing*, Wiley, 2012.

[9] J. F. Meyer, "On evaluating the performability of degradable computing systems," *IEEE Transactions on Computers*, vol. 100, no. 8, pp. 720-731, 1980.

[10] B. Yang, F. Tan and Y. S. Dai, "Performance evaluation of cloud service considering fault recovery," *Journal of Supercomputing*, vol. 65, no. 1, pp. 426-444, 2013.

[11] K. Tokuno, and S. Yamada, "Codesign-oriented performability modeling for hardware-software systems," *IEEE Transactions on Reliability*, vol. 60, no. 1, pp. 171-179, Mar. 2011.

[12] J. H. Kim, S. M. Lee, D. S. Kim, and J. S. Park, "Performability analysis of IaaS cloud," *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, 2011, pp. 36-43.

[13] Y. Tamura and S. Yamada, "Performance evaluation and dependability analysis for open source cloud computing," *International Transactions on Systems Science and Applications*, vol.8, pp. 1-11, Dec. 2012.

[14] Y. S. Dai, Y. Xiang, Y. Li, L. Xing and G. Zhang, "Consequence oriented self-healing and autonomous diagnosis for highly reliable systems and software," *IEEE Transactions on Reliability*, vol. 60, no. 2, pp. 369-380, 2011.

[15] K. S. Trivedi, *Probability and Statistics With Reliability, Queuing, and Computer Science Applications*, New York: Wiley, 2001.