A Hierarchical Correlation Model for Evaluating Reliability, Performance, and Power Consumption of a Cloud Service

Xiwei Qiu, Yuanshun Dai, *Member, IEEE*, Yanping Xiang, and Liudong Xing, *Senior Member, IEEE*

***Abstract*—Cloud computing is a new emerging technology aimed at large-scale resource sharing and service-oriented computing. To achieve the efficient use of cloud resources for supporting a cloud service, many important factors need to be considered, particularly, reliability, performance, and power consumption of the cloud service. Evaluation of these metrics is essential for further designing rational resource scheduling strategies. However, these metrics are closely related; they do affect one another. The cloud system should consider correlations among the metrics to make more precise evaluation. Most of the existing approaches and models handle these metrics separately, and thus they cannot be used to study the correlations. This paper presents a new hierarchical correlation model for analyzing and evaluating these correlated metrics, which encompasses Markov models, queuing theory, and a Bayesian approach. Various distinctive characteristics of the cloud system are investigated and captured in the model, such as multiple virtual machines (VMs) hosted on the same server, common cause failures of co-located VMs caused by server failures, and logical mapping mechanisms for multicore CPUs. Moreover, for evaluating and balancing the tradeoff between performance and power consumption, a tradeoff parameter and a pure profit optimization model are developed based on the presented correlation model. Numerical examples are provided.**

***Index Terms*—Cloud computing, cloud reliability, correlation, hierarchical Markov model, power efficiency.**

I. Introduction

CLOUD computing is a newly developed technology with numerous novel characteristics, such as large-scale resource sharing, on-demand resource provisioning, and service-oriented computing [1], [2]. The use of virtual machines (VMs) enables safe isolation of co-located applications and further implements various new service modes, including software as a service (SaaS), platform as a service, and infrastructure as a service (IaaS) [3]. In real-world scenarios, many serious problems existing in the conventional IT systems can be effectively solved by the cloud system. For example, the resource utilization of the conventional IT systems is highly inefficient since most physical servers only operate at 10%–50% of their full capacity most of the time [4], and the conventional IT systems usually keep almost all the servers active even though some of the servers are idle, which results in a massive waste of power consumption [5], [6]. The cloud system applying VM consolidation can significantly improve the resource utilization of physical servers. This is not only has a positive effect on power saving, but also provides a feasible method to improve service performance.

For providing a stable cloud service while efficiently saving the consumed power, performance and power consumption become important metrics of the cloud service that must be simultaneously taken into account. This is not a trivial issue since performance and power consumption are mutually correlated metrics. For example, more VMs serving users in parallel can enhance the performance of the cloud service. However, more VMs also require occupying more physical resources, which inevitably leads to an increase in the power consumption. In fact, there exists a complicated tradeoff between the performance and power metrics [7]. The cloud system requires an important capability of estimating this tradeoff to further develop optimal resource scheduling strategies.

Although many recent studies have proposed various approaches for balancing the performance–power tradeoff [8]–[10], none of them considered reliability, another significant factor of the cloud service. More importantly, in realistic scenarios, both performance and power consumption are indeed affected by reliability. For example, an unreliable cloud system with a higher probability of resource failures inevitably results in more interruption of running VMs, which implies a reduction in the performance of the cloud service. Subsequent failure recovery also incurs additional power consumption. Thus, reliability, performance, and power consumption are closely related and should not be considered separately.

To evaluate these important metrics systemically, theoretical modeling is a feasible and efficient method that can cover a large realistic parameter space to ensure fidelity. Since reliability has a significant influence on the other metrics, it needs

Manuscript received December 22, 2014; accepted April 20, 2015. This work was supported in part by the 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 Grant 2015TD0002. This paper was recommended by Associate Editor H. Tianfield.

X. Qiu, Y. Dai, and Y. Xiang are with the Collaborative Autonomic Computing Laboratory, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: qiu_uestc@aliyun.com; dai@cloudian.org; yanping_xiang@163.com).

L. Xing is with the Department of Electrical and Computer Engineering, University of Massachusetts, Dartmouth, MA 02747 USA (e-mail: ldxing@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2015.2452898

being first analyzed in detail. However, the virtualization technology brings new challenges in reliability modeling. For example, once server failures occur, none of the co-located VMs on the server can operate. This is a new kind of common cause failures (CCFs) that does not exist in the traditional systems. Meanwhile, execution of co-located VMs is noninterfering, which means failures of different VMs are independent. These distinctive characteristics of the cloud system should be investigated for reliability modeling.

On the other hand, for precisely evaluating performance and power metrics, the existing correlations (i.e., random changes of performance and power consumption due to random failures and recovery) must be captured. This is a critical factor that affects accuracy of evaluation. However, for the cloud system with a large-scale structure, directly employing a monolithic model to capture the correlations is difficult because the model always tends to be complex and intractable. In contrast, a more flexible modeling approach such as interacting stochastic submodels [11] can contribute to overcoming this difficulty. Once such a systemic evaluation model is built, solutions of the model can be treated as an efficient measurement of performance and power consumption taking the effect of reliability into account. However, there also exists a critical issue of analyzing the performance–power tradeoff since it is difficult to find a general method to directly estimate the balance of the tradeoff.

To remedy this lack, we propose a new and comprehensive theoretical model for evaluating reliability, performance, and power consumption of a cloud service. According to the precise evaluation of the proposed model, we further design a tradeoff parameter and a pure profit optimization model to analyze the tradeoff of performance and power consumption. The primary innovation of this paper is that it investigates server failures, VM failures, and corresponding recovery to build a reliability model of the cloud service. Another major contribution of this paper is a tractable and flexible hierarchical modeling approach. We first propose a three-layer logical structure in a hierarchical manner, which consists of resource layer, application layer, and management layer for analyzing reliability, performance, and power consumption, respectively. The existing correlations (or interacting parameters) are captured by clearly defining the interfaces between different layers. Then, Markov reward models (MRMs) are used to connect these logical layers, and overall solutions of the hierarchical model can be obtained by using a Bayesian approach.

The remainder of this paper is organized as follows. Section II describes the general cloud computing paradigm, and presents a three-layer logical structure of cloud services in a hierarchical manner. Section III introduces a hierarchical correlation model for evaluating reliability, performance, and power consumption of a cloud service. The tradeoff between performance and power consumption is also studied by designing a tradeoff parameter and developing a pure profit optimization model. Numerical examples are also illustrated in Section III. Section IV describes some related researches followed by highlights of new contributions made by this paper. Section V concludes this paper.

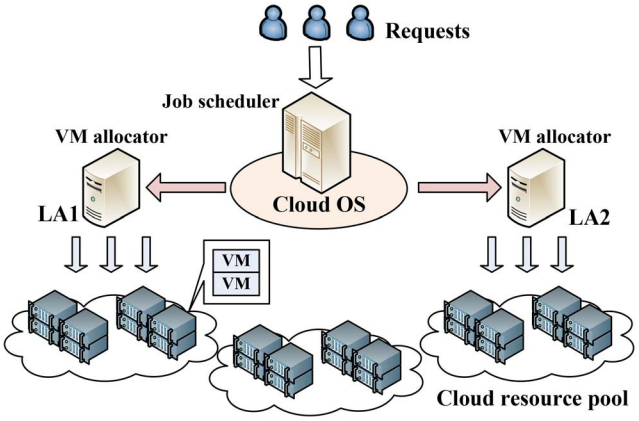

Fig. 1. General structure of a cloud computing system.

II. Analysis of Cloud System and Correlated Metrics

Cloud computing has emerged as an important field, distinguished from the conventional distributed computing systems by its focus on sharing of cloud infrastructure, innovative cloud service modes, and, in some cases, an orientation toward high performance and power saving.

A. Description of the Cloud System and Cloud Services

A general representation of the cloud system is depicted by Fig. 1. The cloud system has an essential component, i.e., cloud operating system (COS), which is the "brain" of cloud computing. The COS receives service requests from users and translates them into "jobs." According to some scheduling strategies, these jobs are sent to some local agents (LAs), which can be treated as VM allocators and supervisors. After the processing of the LAs, the jobs are finally executed by some VMs.

Conventional services in a distributed computing environment can typically be divided into two categories: large online services and individual services. A large online service such as a social networking service, a traffic information service, and a Web service is usually supported by multiple active servers. These servers are running continuously and simultaneously for serving millions of users in real time. In contrast, an individual service emphasizes on providing various applications (e.g., processing programs, office tools, and professional software) to users for performing some specific functions.

In cloud computing environments, large online services can be supported by IaaS clouds while individual services are better fit with SaaS clouds. In this paper, we focus on the large online service in the IaaS cloud (we use the term "cloud service" to mean a large online service in the following of this paper). Using the virtualization technology, active physical servers in the conventional computing systems become VMs, which can be treated as "logical servers," and these VMs are consolidated on fewer physical servers for more efficient use of cloud infrastructure.

B. Logical Layers for Analyzing Correlated Metrics

To analyze the correlation among reliability, performance, and power consumption, we build a three-layer logical

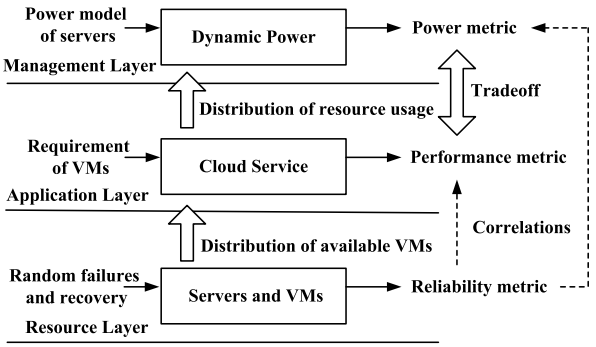

Fig. 2. Three-layer logical structure for the correlated metrics.

structure in a hierarchical manner, as shown in Fig. 2. The resource layer represents physical servers and VMs in the cloud resource pool. The application layer includes various cloud services, each having multiple VMs (i.e., logical servers) processing users' requests in parallel. The management layer focuses on controlling power consumption of servers and the cloud services. The interactions between different layers identify the existing correlations, which are described as follows.

1) *Reliability–Performance Correlation:* For a cloud service, the number of available VMs directly decides its performance. However, these VMs may be failed due to server failures or VM failures. Thus, the number of available VMs is a random variable affected by the reliability parameter, which implies that performance is a resulting attribute of reliability.
2) *Reliability–Power Correlation:* A critical factor deciding dynamic power consumption of servers is resource usage. Sine each VM has a certain resource requirement, the resource usage of servers can be expressed as a function of the number of available VMs. Thus, power consumption is also a resulting attribute of reliability.
3) *Performance–Power Correlation:* The amount of resources assigned to a cloud service has inverse effects on performance and power consumption. This implies that the performance–power correlation is essentially a tradeoff relationship.

Due to the hierarchical nature of the correlated metrics, we find that hierarchical correlation modeling should be a clear and tractable approach for achieving precise evaluation. Resource usage associated with random failures and recovery is the critical parameter that builds the essential connections between these metrics.

C. Analysis of Resource Usage Pattern

CPU frequency is one of the most important computational resources in the cloud resource pool. Modern physical servers widely deploy multicore CPUs and, in general, the essential way to take full advantage of computational capability of a multicore CPU is to implement parallel execution in all of its cores [12]. The cloud system manages multicore CPUs by using frequency scaling and logical mapping.

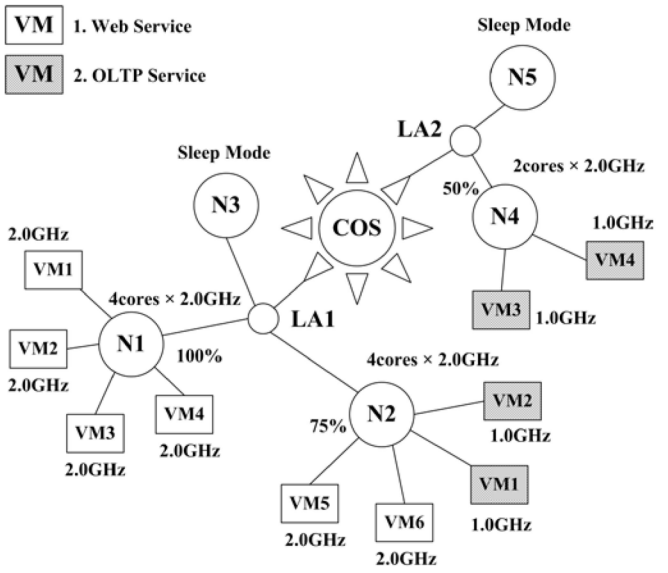

Fig. 3. Typical cloud service scenario.

Logical mapping decides how many cores can be assigned as a logical CPU to a VM. For VMs of a cloud service that need to remain active so as to serve users at any time, one-to-one mapping from a physical core to a logical CPU can be considered as a reasonable resource usage pattern in most cases. With this mechanism, a VM exclusively occupies a physical core and thus absolutely avoids potential resource competition of the other co-located VMs. There also exist two other mapping mechanisms: one-to-many and many-to-one mapping from the physical core to the logical CPU. The former solution is usually used for VMs that do not have strong demand of CPU resource; the latter solution is more suitable for VMs with the capability of parallel computing. In this paper, we assume VMs of a cloud service employ the one-to-one mapping mechanism, meaning that only one VM runs in each core at any time.

Fig. 3 illustrates a typical cloud service scenario, where all servers possess multicore CPU structures. For example, the CPU of N1 consists of 4×2.0 GHz cores, which means that it can support four VMs running simultaneously, with a maximum computational capacity of 2000 MIPS assigned to each VM. In this paper, we apply a virtual tree topology to represent a server and its co-located VMs. This tree structure enables modeling of CCFs of VMs caused by server failures.

Two typical kinds of cloud services, i.e., the Web service and the online transaction processing (OLTP) service, are taken as examples in Fig. 3. Each VM of these services is a logical server that processes users' requests one by one. The logical CPU frequency of a VM decides its computing speed, which also notably affects the mean time to serve a request. For both of these two services, increasing the number of active VMs contributes to improving the capability of serving requests in parallel, ultimately having a positive effect on the performance. As shown in Fig. 3, the Web service and the OLTP service have six VMs with scaled frequencies of 2.0 GHz and four VMs with scaled

frequencies of 1.0 GHz, respectively. The average CPU frequency of N1 is 2.0 GHz, which means all the cores are working at the maximum frequency simultaneously. Although two CPU cores of N4 are hosting VMs, the average CPU frequency of N4 remains 1.0 GHz (i.e., the usage of N4 is 50%) since the VMs do not totally occupy the CPU frequency of N4.

III. Hierarchical Modeling for Evaluating Correlated Metrics

A. Reliability Modeling of Resource Layer

In this paper, the presented reliability model considers not only VM failures but also server failures of the cloud service. The influence of server failures always plays an important role in any traditional services, and this influence is magnified in the cloud service where a server usually hosts multiple co-located VMs. If a server failure occurs, all the co-located VMs will be halted by the COS. Although the analysis of this phenomenon is similar in spirit to the traditional hardware/software co-design [13], the distinctive feature of VM isolation makes the cloud reliability modeling become different. In this paper, we make the following assumptions for the cloud reliability modeling.

1) A server may be down because of a certain hardware failure. The server failure follows a Poisson process with a failure rate θ. The hardware failures of different servers are independent of one another. Once a server failure occurs, all co-located VMs on the server are halted until the failure is resolved.
2) A VM failure is an obvious software failure that can be immediately detected by the COS, which means the running VM is instructed to halt as soon as the VM failure is detected. A cloud service can have multiple VMs for serving users in parallel. Software failures of these VMs follow Poisson processes with the same failure rate λ_s. Meanwhile, failures of different VMs are noninterfering and independent due to the isolating nature of the virtualization technology.
3) Any failure initiates a repair process. The hardware repair time and the software restore time follow exponential distributions with repair rates η and μ_s, respectively.
4) Due to the one-to-one logical mapping mechanism for the cloud service, a server deployed with an M-core CPU is only allowed to simultaneously host a maximum of M co-located VMs.
5) The COS and LAs are fully reliable, which can be justified since a relatively short interval of their computation is required for transmitting users' requests and cloud control commands.

For the first and second assumptions, the server and VM failures are assumed to follow Poisson processes, which can be explained as being either within the operational phase or in a steady state after a long-time run [14]. The assumption of independent failures in different physical servers is a good approximation to reality since the servers are uncorrelated. The third assumption of repair time in accordance with the exponential distribution has also been widely accepted in [15].

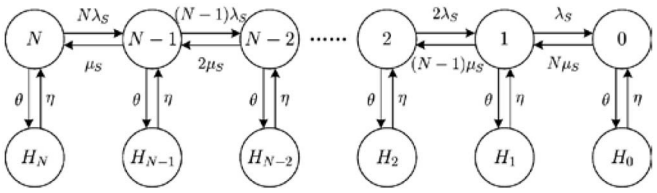

Fig. 4. Markov model for a server hosting multiple VMs.

The fourth assumption is reasonable for a cloud service in most cases, as illustrated in Section II-C.

Suppose a physical server with an M-core CPU has been deployed with $N(N \leq M)$ co-located VMs running simultaneously. States of the VMs and the server can be modeled as a Markov process, which is shown in Fig. 4. The state $n(n = N, N-1, \ldots, 1, 0)$ represents that the number of available VMs is n, which also implies that the other $N-n$ VMs are halted by their own software failures, not a hardware failure of the server. The state $H_n(n = N, N-1, \ldots, 1, 0)$ represents that a hardware failure of the server occurs when n VMs are running, which results in the number of available VMs on the server becoming 0 according to Assumption A1. Note that states $H_N, H_{N-1}, \ldots, H_1$ and H_0 are essentially different states, even though they all represent that no VMs are available on the server. This is because that state H_n implies that n VMs were available before the hardware failure occurred, and thus it can only transit back to state n. Denote π_n and π_{H_n} as the steady probabilities for the server to stay at states n and H_n, respectively. It is easy to derive π_n and π_{H_n} by solving the following Chapman–Kolmogorov equations:

$$(N\lambda_s + \theta)\pi_N = \mu_s\pi_{N-1} + \eta\pi_{H_N} \tag{1}$$

$$\begin{aligned}(n\lambda_s + (N-n)\mu_s + \theta)\pi_n = {} & (n+1)\lambda_s\pi_{n+1} \\ & + (N-n+1)\mu_s\pi_{n-1} \\ & + \eta\pi_{H_n}(n = N-1, N-2, \ldots, 1)\end{aligned} \tag{2}$$

$$(N\mu_s + \theta)\pi_0 = \lambda_s\pi_1 + \eta\pi_{H_0} \tag{3}$$

$$\eta\pi_{H_n} = \theta\pi_n(n = N, N-1, \ldots, 1, 0) \tag{4}$$

$$\sum_{n=N}^{0}\left(\pi_n + \pi_{H_n}\right) = 1(n = N, N-1, \ldots, 1, 0). \tag{5}$$

Solving (1)–(5), the expressions of π_n and π_{H_n} can be obtained as

$$\pi_N = \left[\sum_{k=0}^{N} C_N^k(\lambda_s/\mu_s)^k(1 + \theta/\eta)\right]^{-1} \tag{6}$$

$$\pi_{N-k} = C_N^k(\lambda_s/\mu_s)^k\pi_N(k = 1, 2, \ldots, N) \tag{7}$$

$$\pi_{H_{N-k}} = C_N^k(\lambda_s/\mu_s)^k(\theta/\eta)\pi_N(k = 0, 1, 2, \ldots, N). \tag{8}$$

In fact, we need to pay more attention to the number of available VMs rather than the detailed reasons or types of occurred failures in a realistic environment, as the number of available VMs directly affects the related performance and power consumption. For a server with N co-located VMs, the number of its available VMs is a discrete random variable

denoted as X. From (6)–(8), the probability mass function for X to take a value $x(x = 0, 1, \ldots, N)$ is derived as

$$p(x) = \Pr(X = x) = \pi_x (x = 1, 2, \ldots, N) \tag{9}$$

$$p(0) = \pi_0 + \sum_{n=0}^{N} \pi_{H_n} = (\lambda_s/\mu_s)^N \pi_N + (1 + \lambda_s/\mu_s)^N (\theta/\eta)\pi_N. \tag{10}$$

Suppose the COS assigns K servers to the cloud service for deploying its VMs. The multicore CPUs of the K servers have $M_1, M_2, \ldots, M_K$ cores, respectively. The kth server hosts $N_k (N_k \leq M_k)$ VMs, and there are a total of $N_v = \sum_{k=1}^{K} N_k$ VMs created for the cloud service. Denote the number of available VMs for the cloud service as X_a, which is a discrete random variable that ranges from 0 to N_v. The probability mass function of X_a can be obtained using

$$p_a(x) = \Pr(X_a = x) = \Pr\left(\sum_{k=1}^{K} X_k = x\right)(x = 0, 1, \ldots, N_v). \tag{11}$$

In (11), $X_k (k = 1, 2, \ldots, K)$ is a random variable representing the number of available VMs in the kth server, and its distribution law can be obtained by using (9) and (10). Moreover, $X_1, X_2, \ldots, X_K$ are i.i.d. random variables.

Then, the expected number of available VMs for the cloud service is calculated by

$$E(X_a) = E\left(\sum_{k=1}^{K} X_k\right) = \sum_{k=1}^{K} E(X_k). \tag{12}$$

The reliability of the cloud service can be defined as the probability that all of its VMs being unavailable does NOT occur, which is derived by

$$R = 1 - p_a(0). \tag{13}$$

Illustrative Example 1: Consider the Web service shown in Fig. 3 as an example. The service has four VMs hosted on N1 and two VMs hosted on N2. The number of available VMs at N1 and N2 can be represented as discrete random variables X_1 and X_2, respectively. The software failure rate of each VM is $\lambda_s = 0.0008\ \text{s}^{-1}$, and the repair rate is $\mu_s = 0.0030\ \text{s}^{-1}$. Suppose that N1 and N2 are homogeneous, and their hardware failure rate and repair rate are $\theta = 0.0004\ \text{s}^{-1}$ and $\eta = 0.0015\ \text{s}^{-1}$, respectively. By evaluating (6)–(11), we can obtain the distribution laws of X_1, X_2, and X_a, as given in Table I.

Note that the homogeneous servers used in this example are only for illustration; heterogeneous servers can also be implemented in a similar way for more realistic scenarios. To calculate (12), the expected number of available VMs for the Web service is $E(X_a) = 3.7396$. From (13), the reliability of the Web service can be derived by

$$R = 1 - p_a(0) = 0.9479. \tag{14}$$

Note that assigning more servers to support the Web service can achieve a higher reliability. To demonstrate this situation, we take three resource scheduling strategies with different numbers of assigned servers as examples, all of which host six VMs for supporting the Web service. The strategies are described as follows.

TABLE I
DISTRIBUTION LAWS OF X_1, X_2, AND X_a

x_1	4	3	2	1	0
$p_1(x_1)$	0.3067	0.3271	0.1309	0.0232	0.2121

x_2	2	1	0
$p_2(x_2)$	0.4921	0.2624	0.2455

x	6	5	4	3	2	1	0
$p_a(x)$	0.1509	0.2414	0.2255	0.1261	0.1426	0.0614	0.0521

S1) Two servers are assigned for hosting four VMs and two VMs, respectively, as shown in Fig. 3, denote by (4, 2).
S2) Three servers are assigned and each server hosts two VMs, denoted by (2, 2, 2).
S3) Six servers are assigned and each server hosts only one VM.

The first strategy S1 represents a greedy strategy that always tries to occupy as many cores of a server as possible. The second strategy S2 is a load balance strategy, which makes all assigned servers have the same resource usage as much as possible. The last strategy S3 is essentially the same as a conventional resource scheduling strategy.

From (13), the reliability of the cloud service for S1–S3 are 0.9479, 0.9852, and 0.9971, respectively. Obviously, S3 has the highest reliability among the three designed strategies. In general, the reliability of a cloud service can be improved by increasing the number of VMs and assigning them to more servers.

To verify analytical results obtained by the proposed cloud reliability model, a simulation program has been developed which considers the CCFs of co-located VMs caused by the failures of the corresponding physical server. To obtain a more accurate estimation of the steady probability of the cloud reliability model, the simulation program is designed to capture 500 times of failures (including VM failures and server failures). Then, the simulation program runs 150 times for calculating the reliability of the cloud service for S1–S3, as shown in Fig. 5.

From Fig. 5, one can see that the simulation results for S1–S3 fluctuate around the theoretical values of the reliability (i.e., 0.9479, 0.9852, and 0.9971 calculated from our analytical model), respectively. Note that the fluctuation ranges for S3, S2, and S1 increase gradually. The main reason resulting in such a situation is the CCFs of co-located VMs caused by the server failures. Since S1 has the most number of co-located VMs (i.e., four VMs) compared to S2 and S3, a server failure of S1 can lead to the most serious effect on the reliability. This phenomenon shows the significance of our reliability model on the analysis of this kind of CCFs.

Let random variables X_a^{S1}, X_a^{S2}, and X_a^{S3} represent the random number of available VMs for Strategies S1–S3, respectively. The distributions of X_a^{S1}, X_a^{S2}, and X_a^{S3} obtained

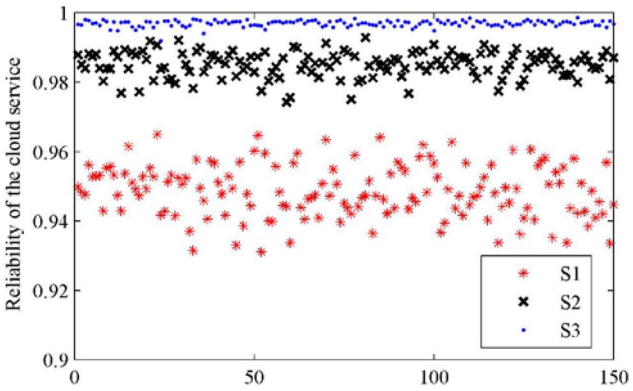

Fig. 5. Reliability estimation of 150 runs of the simulation program.

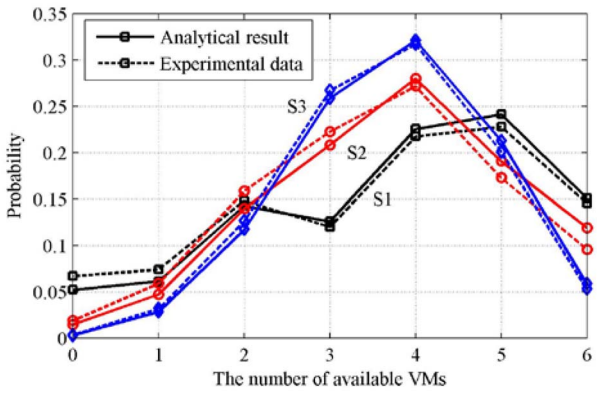

Fig. 6. Analytical results versus simulation results.

by the simulation are depicted in Fig. 6. As shown in the figure, the analytical results calculated by the reliability model are very close to the simulation results, which witness that the proposed cloud reliability model is justified.

In the special case where the servers are homogeneous, each VM of the cloud service has the same parameters λ_s, μ_s, θ, and η, which means the operational state of these VMs are identically distributed random variables that can be denoted as X_{vm}. Thus, even though S1–S3 are different, we can find that the following equation is satisfied:

$$E\left(X_a^{S1}\right) = E\left(X_a^{S2}\right) = E\left(X_a^{S3}\right) = 6E(X_{\text{vm}}) = 3.7396. \quad (15)$$

However, due to the CCFs of co-located VMs caused by server failures, these VMs become dependent, e.g., four VMs hosted on N1 in Fig. 3. Thus, the distribution of X_a associated with S1–S3 are very different, as shown in Fig. 6. According to the distribution of X_a derived from the proposed reliability model, we can further evaluate performance metric in the upper application layer.

B. Performance Modeling of Application Layer

To make performance modeling tractable while capturing the correlation between the reliability and performance metrics, we first take the value of X_a as an input parameter during the performance modeling, and then use a Bayesian approach

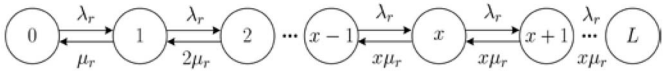

Fig. 7. Birth–death process for the request queue.

to remove it. The following assumptions are made for the application layer.

B1) For a cloud service, the arrival of user requests follows a Poisson process with an arrival rate λ_r. Limitation of the request queue length is L. When a new request arrives, it is blocked if the queue is full, which causes a blocking failure and results in the request being discarded by the COS.

B2) All available VMs of the cloud service serve the arrived requests in parallel. The service time of a request is exponentially distributed with the parameter μ_r.

B3) A request has its due time T_d. If the sojourn time of a request exceeds its due time T_d, a time-out failure occurs.

Assumption B1, which states that the arrival of requests follows a Poisson process, can be justified as a memoryless process for unknown users to submit requests. B2 is also a commonly accepted assumption, which is more general than assuming a fixed/constant service time.

Given x available VMs of the cloud service, the resulting birth–death process $Y(t)$ is modeled in Fig. 7. The state $i (i = 0, 1, \ldots, L)$ represents the number of requests in the queue. If $i < x$, all i requests can be immediately served by i available VMs, so the exit rate of any one request is equal to $i \cdot \mu_r$. If $i \geq x$, only x requests can be served by the x available VMs, so the exit rate is $x \cdot \mu_r$. Denote q_i as the steady probability of the request queue staying at state i, which can be derived as

$$q_0 = \left[\sum_{i=0}^{x-1} \frac{\lambda_r^i}{i!\mu_r^i} + \sum_{i=x}^{L} \frac{\lambda_r^i}{x^{i-x}x!\mu_r^i}\right]^{-1} \quad (16)$$

$$q_i = \frac{\lambda_r^i}{i!\mu_r^i} q_0 \ (1 \leq i < x) \quad (17)$$

$$q_i = \frac{\lambda_r^i}{x^{i-x}x!\mu_r^i} q_0 \ (x \leq i < L). \quad (18)$$

If the request queue is full, new requests cannot be added into the queue, which causes a blocking failure. At this point, the COS discards the new requests so as to avoid overflowing of the queue. Therefore, the probability of a blocking failure occurring is

$$p_{\text{block}} = q_L \quad (19)$$

where q_L is obtained by solving (16)–(18). Let λ_e represent the effective arrival rate of the requests, giving the loss rate of requests of $\bar{\lambda}_e = \lambda_r - \lambda_e$, which can be obtained as

$$\lambda_e = \lambda_r(1 - q_L). \quad (20)$$

Let $\Pr(i)$ represent the probability that there already exist i requests in the queue when a new request is added into the queue. If $i < x$, the added request can be served immediately. If $x \leq i < L$, the request needs to wait since all x available VMs are occupied. Here, we adopt the rule of

"first come first serve" for the request queue. The probability $\Pr(i)$ can be determined by

$$\begin{aligned}\Pr(i) &= \Pr(Y = i | i < L) \\ &= \frac{q_i}{1 - q_L} \ (i = 0, 1, \ldots, L-1).\end{aligned} \tag{21}$$

Denote the waiting time of the added request as T_w. The probability that the request is served immediately without a waiting period, that is, the probability of $i < x$, is given by

$$F_w(0) = \Pr(T_w \le 0) = \sum_{i=0}^{x-1} \Pr(i). \tag{22}$$

For the other situation where $x \le i < L$, there already exist $i - x$ requests waiting to be served before the newly added request. The time to complete any one request follows an exponential distribution with the parameter $x \cdot \mu_r$. Thus, the waiting time of the new request is equal to the time for $i-x+1$ requests to be completed, which is a random variable denoted as T_{i-x+1}, in accordance with the Erlang-$(i-x+1)$ distribution, with the probability density function (pdf) defined as

$$f_{i-x+1}(t) = \frac{(x\mu_r t)^{i-x}}{(i-x)!} x\mu_r e^{-x\mu_r t} \ (t \ge 0, i > x). \tag{23}$$

From (21) to (23), we can obtain the cumulative probability distribution function of T_w as

$$F_w(t) = \begin{cases} F_w(0) & t = 0 \\ F_w(0) + \sum_{i=x}^{L-1} \Pr(i) \int_0^t f_{i-x+1}(\tau) d\tau & t > 0. \end{cases} \tag{24}$$

Obviously, the sojourn time of a request is the sum of its queue time T_w and its execution time T_e. As mentioned above, the execution time of the request follows an exponential distribution with parameter μ_r. Hence, the pdf of the sojourn time T_s can be obtained as

$$f_s(t) = \sum_{i=0}^{x-1} \Pr(i) f_e(t) + \sum_{i=x}^{L-1} \Pr(i) f_{i-x+1}(t) \otimes f_e(t) \tag{25}$$

where "$\otimes$" represents the convolution operator of two functions and $f_e(t)$ is the pdf of the execution time T_e.

Finally, the probability of a timeout failure (i.e., the request is not successfully completed within the due time T_d) can be computed by

$$p_{\text{timeout}} = \Pr(T_s > T_d) = 1 - \int_0^{T_d} f_s(t) dt. \tag{26}$$

According to specific requirements of real-world scenarios, many measures can be treated as performance metrics. For the large online service considered in this paper, it needs to run continuously, serving numerous requests from millions of users. Thus the throughput of the cloud service is of critical importance, which can be defined as the average number of successfully completed requests (i.e., the requests are completed within their due times) per unit of time. Denote this performance metric of the cloud service as γ. When the application has x available VMs serving user requests, from (19) and (26), we can obtain $\gamma(x)$ as

$$\gamma(x) = \lambda_r (1 - p_{\text{block}})(1 - p_{\text{timeout}}). \tag{27}$$

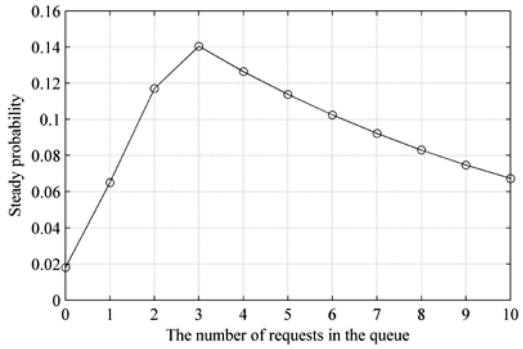

Fig. 8. Expected probability for the queue length to be i given $x = 4$.

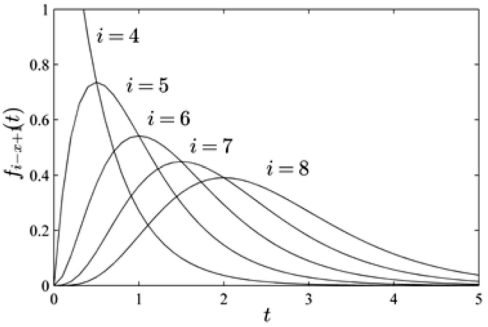

Fig. 9. Probability density function of $f_{i-x+1}(t)$.

Now, we can implement a MRM to connect reliability with performance. In addition to the steady probability $p_a(x)(x = 1, 2, \ldots, N_v)$ derived from (11), each state x also has a reward value, i.e., $\gamma(x)$. For the cloud service with N_v VMs in total, we can obtain the expected performance using

$$E(\gamma) = \sum_{x=1}^{N_v} p_a(x)\gamma(x). \tag{28}$$

Illustrative Example 2: Continue the example described in Section III-A. Suppose that the maximum allowable number of requests waiting in the request queue is 10 (i.e., $L = 10$), the arrival rate of the users' requests of the Web service is $\lambda_r = 1.8\ \text{s}^{-1}$ and the completion rate of each request is $\mu_r = 0.5\ \text{s}^{-1}$. The Web service has six VMs in total (i.e., $N_v = 6$). For the purpose of illustration, we first numerically set the number of available VMs as $x = 4$. From (16)–(18), we can obtain q_i, which is depicted in Fig. 8.

Given this scenario, the probability that a new request is discarded by the COS can be obtained as $p_{\text{block}} = 0.0672$ by using (19). Substituting the given parameters in (23) gives the pdf $f_{i-x+1}(t)$ of the different values of x, which are depicted in Fig. 9.

We can then substitute the values and functions of Figs. 8 and 9 into (21)–(25) to obtain the pdf of the sojourn time as

$$\begin{aligned}f_s(t) = {} & 1.0796e^{-0.5t} - 0.8970e^{-2t} - 1.2100te^{-2t} \\ & - 0.7856t^2e^{-2t} - 0.3196t^3e^{-2t} - 0.0869t^4e^{-2t} \\ & - 0.0142t^5e^{-2t}.\end{aligned} \tag{29}$$

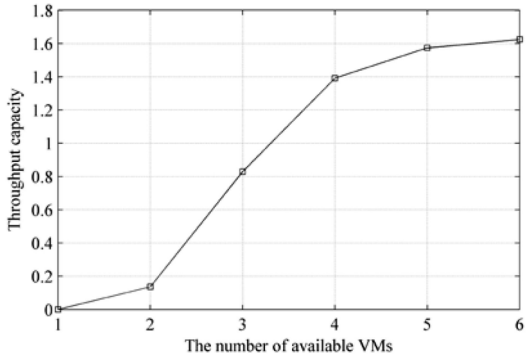

Fig. 10. Conditional expected values of $\gamma(x)$.

Numerically, we set the due time for successfully finishing a service request as $T_d = 5$ s. Then, from (26), the probability that a time-out failure occurs can be obtained as

$$p_{\text{timeout}} = 1 - F_s(T_d) = 0.1716. \tag{30}$$

Therefore, given four available VMs, the conditional expected values of the performance metrics is $\gamma(4) = 1.3910\ \text{s}^{-1}$. Fig. 10 illustrates all conditional expected values of $\gamma(x)$. Note that there also exists a special condition with $x = 0$ indicating that all the VMs are down. The conditional expected value for $x = 0$ becomes $\gamma(0) = 0$, which indicates that no requests are completed.

With these results and the distributions of X_a^{S1}, X_a^{S2}, and X_a^{S3} obtained from Section III-A, as shown in Fig. 6, we can substitute them into (28) to obtain the expected performance of the cloud service with three different strategies S1–S3 as $1.0627\ \text{s}^{-1}$, $1.0746\ \text{s}^{-1}$, and $1.1074\ \text{s}^{-1}$, respectively. The corresponding expected request completion rates (i.e., γ/λ) are 59.04%, 59.70%, and 61.52%. These results show that the greedy strategy S1 and the load-balance strategy S2 lead to a slight decrease in the service performance compared to the conventional strategy S3. Note that the request completion rates seem low, mainly due to the short queue capacity and the relative small value of N_v designed in the example for simplifying illustration.

For most cloud services in real-world scenarios, it should be noted that the logical frequency of each VM has a significant influence on the service performance, as described in Section II-C. Usually, a request is ultimately translated into an executable task, and the processing of the task includes a set of computing operations, so the work requirement of the request can be measured by the number of commands or instructions to be executed. On the other hand, the logical frequency of a VM can be defined by an MIPS rating, which means the logical frequency directly affects the service rate of the VM. Therefore, for a cloud service with the computing intensive feature, we can establish an important relationship between the request service rate and the logical frequency of the VM, i.e.,

$$\mu_r \propto f_{\text{vm}}. \tag{31}$$

C. Power Modeling of Management Layer

In this paper, the power consumption of a cloud service is defined as the total power consumption of all the servers used for hosting VMs. In practice, the power model of a server needs to be calibrated using some statistical methods. However, the CPU power is always considered as the largest and most variable component of the server power [16], [17]. In this paper, we simplify the power model of the server by assuming that the power consumption of all other components (e.g., disk, memory, and network) is essentially constant regardless of system activities [18]. Thus, the dynamic power consumption of the server depends on its CPU voltage and frequency as well as its CPU utilization. Meanwhile, we also assume that the active cores of the CPU are fully utilized since VMs hosted on the cores run continuously. The validity of this assumption is supported by results from prior research reported in the literature, which concluded that the intensity of workload directed at the VMs does not affect the power consumption [19]. Denote the power consumption of the server as $P(f)$. It can be described using the following formula:

$$P(f) = c_0 + c_1 \cdot f^2 \tag{32}$$

where c_0 is a constant representing the basic power consumption of the server, and c_1 is the power coefficient. f in (23) represents the average frequency of all the cores, which has a significant influence on the dynamic power consumption $c_1 \cdot f^2$ [20]. For convenience, f is normalized in terms of a ratio to the maximum average frequency of the multicore CPU, and thus $0 < f \leq 1$. Note that the basic power consumption c_0 always has a relatively large value, which implies that there is significant overhead involved with running the server, and thus it is inefficient to keep the server working with a low average CPU frequency.

Suppose a CPU has M cores with $x(1 \leq x \leq M)$ of them being active cores. If the scaled frequency of the jth $(1 \leq j \leq x)$ active core is f_{core_j}, the average CPU frequency of the multicore CPU can be obtained as

$$f(x) = \frac{1}{M} \sum_{j=1}^{x} f_{\text{core}_j} \left(0 < f_{\text{core}_j} \leq 1\right) \tag{33}$$

where f_{core_j} is expressed as a ratio to the maximum frequency of the jth core. The equation $f = 1$ is satisfied when all the cores are simultaneously active and working at their maximum frequencies.

Since f is dynamically changed due to the random server failures, VM failures, and failure recovery, we make the following assumptions to capture these details.

- C1) After a VM failure occurs, the VM is suspended and its hosting core remains inactive until a restoration action is complete. The relatively small increase in average CPU frequency caused by the restoration action (e.g., rollback of the failed VM) is negligible since it is usually not a computing intensive action.
- C2) A server hardware failure results in all of its hosted VMs being unable to operate. The power consumption of the server is $P(0_+) = \lim_{f \to 0} P(f) = c_0$ during the failure recovery phase, which means the related large basic power consumption c_0 still exists during this period.
- C3) For a cloud service in a realistic scenario, the COS can dynamically change the number of VMs to fit the

fluctuation of the workload (i.e., the arrival rate of requests). Thus, the logical frequency of the VM is assumed to remain constant.

Suppose there are $K(K \geq 1)$ servers assigned to the cloud service, and the kth $(1 \leq k \leq K)$ server has x_k co-located VMs. According to Assumption C2, we can calculate the power consumption of the kth server using

$$P_k(f(x_k)) = \begin{cases} P_k\left(\frac{1}{M}\sum_{j=1}^{x_k} f_{\text{core}_j}\right) & x_k \neq 0 \\ c_0 & x_k = 0. \end{cases} \tag{34}$$

Then, denote P_s as the power consumption of the cloud service, which can be obtained by

$$P_s = \sum_{k=1}^{K} P_k(f(x_k)). \tag{35}$$

The average CPU frequency $f(x_k)$ also describes the resource usage of the kth server. Note x_k VMs on the kth server have the same logical CPU frequency f_{vm}. According to the one-to-one logical mapping described in Section II-C, the average frequency of the kth server can be simplified as $f(x_k) = x_k f_{\text{vm}}/M$, which shows that the resource usage of the server is a function of x_k.

As mentioned in Section III-A, the number of available VMs on the kth server is a random variable denoted by X_k, and its probability distribution is determined by the random failure and recovery of the resource layer. Since the parameter x_k is the value of random variable X_k, the expected power consumption of the cloud service can also be obtained by using a Bayesian approach, which is written as

$$\begin{aligned} E(P_s) &= E\left(\sum_{k=1}^{K} P_k(X_k)\right) \\ &= \sum_{k=1}^{K}\sum_{x_k} P_k(f(x_k)) \cdot p(x_k) \end{aligned} \tag{36}$$

where $p(x_k) = \Pr(X_k = x_k)$ is the distribution law of X_k, which can be derived from our presented reliability model.

Illustrative Example 3: Take the Web service shown in Fig. 3 as an example. Suppose the basic power consumption c_0 and the peak power consumption $c_0 + c_1$ of the homogeneous servers are 90 and 160 w, respectively. The quadratic function for evaluating the power consumption of the servers can be written as

$$P(f) = 90 + 70 \cdot f^2. \tag{37}$$

The logical frequency of the VM on each active core is the maximum frequency 2.0 GHz (i.e., $f_{\text{vm}} = 1$). Substituting the values of Table I into (36), we obtain the expected power consumption of N1 and N2 for running the Web service as follows:

$$\begin{aligned} E(P_1) &= \sum_{x=0}^{4} P_1(x/4)p_1(x) = 126.74\ w \\ E(P_2) &= \sum_{x=0}^{2} P_2(x/4)p_2(x) = 99.76\ w. \end{aligned} \tag{38}$$

From (36), the expected power consumption of the Web service using S1 is

$$E(P_s) = E(P_1) + E(P_2) = 226.5\ w. \tag{39}$$

Similarly, the expected power consumptions of the Web service using S2 and S3 are 299.28 and 556.36 w, respectively. The results show that the conventional strategy S3 leads to a tremendous increase in the expected power consumption compared to the other strategies S1 and S2. Clearly, S3 is an inefficient strategy from a power consumption perspective since it requires more servers to remain in an active state and thus induces very large basic power consumption.

D. Analysis of Performance–Power Tradeoff

In this paper, we present two methods to analyze the performance–power tradeoff of a cloud service: 1) using a new tradeoff parameter to describe the power efficiency of a resource assignment strategy and 2) optimizing a pure profit function that takes performance and power as inputs to find the balance of the tradeoff.

In a realistic cloud scenario, the cloud provider usually needs to develop a resource assignment strategy for deploying the VMs of the cloud service. However, even though the number of VMs, i.e., N_v, is known, different assignment strategies may result in totally different power efficiency. To describe the power efficiency of a resource assignment strategy, we propose a tradeoff parameter called performance–power efficiency ratio (PPER), which is defined as

$$\rho = \frac{E(\gamma)}{E(P_s)} = \frac{m}{E(P_s) \cdot \Delta t} = \frac{m}{\Delta E_s} \tag{40}$$

where $m = E(\gamma) \cdot \Delta t$ and $\Delta E_s = E(P_s) \cdot \Delta t$ are the expected number of completed requests and the expected energy consumption over the time period Δt, respectively. This tradeoff parameter means the average number of completed requests per unit of energy consumption, which is considered as a measure of the power efficiency of a resource assignment strategy for supporting the cloud service.

The second method finds the balance of the tradeoff by solving a pure profit optimization problem. In general, assigning more resource (i.e., servers and VMs) to the cloud service can improve the performance but incur more power consumption, which results in more cost. As for this situation, we assume that each server hosts the same number of VMs, and thus the number of servers, N, needs to be optimized for achieving the maximum of the pure profit. Note that both performance and power metrics are functions of N, so (28) and (36) can be expressed as functions $G_1(N)$ and $G_2(N)$, respectively. The pure profit optimization model associating with the performance and power metrics is developed as follows:

$$\begin{aligned} &\text{Objective: Max profit}(N) = G_1(N) \cdot \Delta t \cdot c_r - G_2(N) \cdot \Delta t \cdot c_p \\ &\text{Subject to: } N = 1, 2, \ldots, N_{\max} \end{aligned} \tag{41}$$

where c_r and c_p are the mean profit of finishing a request and the mean cost of consuming per unit of energy, respectively.

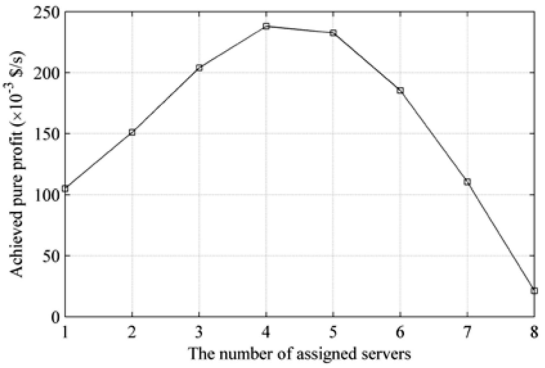

Fig. 11. Pure profit for the number of assigned servers.

$N_{\max}$ is the limitation on the number of servers. The optimization model can be solved by using a convergent searching algorithm (see the Appendix). The optimal solution denoted by N^* implies the balance of the performance–power tradeoff.

Illustrative Example 4: For the first method using the tradeoff parameter, take the above mentioned Web service with $N_v = 6$ as an example. We calculate the PPERs of strategies S1–S3 as 16.9×10^3, 12.9×10^3, and 7.2×10^3 kwh^{-1}, respectively. Clearly, S1 is the most power-efficient strategy as it provides the most benefit in terms of the service performance per kilowatt hour of consumed energy.

For the second method, we suppose each server hosts two VMs for the Web service. Since the limitation of the queue is usually related to the number of servers, we set it as $L = 2 \cdot N$. Suppose c_r and c_p are 0.1×10^{-3} dollars/request and 0.0002×10^{-3} \$/$(w \cdot s)$, respectively. The optimal solution N^* can be obtained using (41), as shown in Fig. 11.

To show the trend, Fig. 11 depicts the pure profit for different numbers of assigned servers ranging from 1 to 8. It can be observed that the pure profit increases at first, and then decrease after $N = 4$, which means that the increased profit achieved by adding one server is less than the additional cost caused by running the server. Thus $N^* = 4$ is the optimal solution maximizing the pure profit of the Web service, which also potentially represents the balance between performance and power consumption.

IV. Literature Review and Discussion

Since cloud computing has been widely adopted, reliability modeling of the cloud system is an important research field. Although the cloud system is essentially a distributed system, the traditional reliability modeling approaches for distributed systems [21], [22] are not suitable for the cloud system due to some unrealistic assumptions, such as constant operational probabilities of various resources. There are also many other reliability models for software, hardware, or small-scale distributed systems (see [14], [15], [23], [24]), which have a more rational assumption that failure probabilities increase with working time. However, these models cannot be directly implemented for studying cloud reliability as the distinctive features of the cloud system (e.g., cloud isolation and cloud consolidation) were not considered. Many recent research about cloud reliability focused on reducing the complexity of the reliability model caused by large scale [25], using fault-tolerant technologies to improving cloud reliability [26], and connecting service failures with network failures [27], but none of them concerned the CCFs of co-located VMs caused by failures of the host server. Our cloud reliability model systemically analyzes this new kind of CCFs, while taking fault-tolerance into account. Moreover, our reliability model can easily connect with network failures since both server failures and VM failures are independent with network failures.

In principle, reliability is always associated with random failures and recovery of various kinds of resources. It inevitably has a significant effect on other evaluation metrics, e.g., the performance and power metrics. For combining reliability with performance, Meyer [28] proposed the notion of performability, and corresponding theoretical models and approaches were subsequently studied [29], [30]. For evaluating the performability of cloud systems, many important performance indices such as expected response delay, expected waiting time, and expected execution time of cloud services were evaluated based on different stochastic models in reliability [11], [31]. However, the correlation between power consumption and reliability does not attract much attention, which to the best of our knowledge has not been done in prior work. Meanwhile, the existing performability models cannot be directly used or extended for being further associated with power consumption, as they did not concern the usage of resources that significantly affects power consumption. Our correlation model takes the resource usage as the critical interacting parameter between different logical layers, which thus can effectively capture the random changes of the performance and power metrics caused by failures and recovery.

There also exist many research studies on power efficient computing, which try to analyze the tradeoff between performance and power consumption. Gelenbe and Lent [8] formulated an optimization problem from both power and QoS perspectives for rationally selecting a mobile application. Kliazovich *et al.* [9] investigated the correlation between power consumption and network performance and proposed a composite evaluation function. Xia *et al.* [10] presented a queuing-network-based performance framework which also applies dynamic voltage scaling to reduce power consumption. Subirats and Guitart [32] considered the flow of power consumption decided by batch cloud workloads, and used aforementioned forecast to support on-demand performance. Lee and Zomaya [33] applied processor utilization scaling and task consolidation to reduce power consumption as well as to ensure the minimal performance degradation of consolidated cloud services. These models and approaches contribute to combining performance with power consumption, but ignore significant influence of reliability, which results in inaccurate evaluation of performance and power consumption. Our correlation model can effectively remedy this lack. Moreover, we also present two feasible methods for effectively analyzing the performance–power tradeoff based on our correlation model.

The methods, which use the tradeoff parameter to evaluate the performance–power efficiency and maximize the pure profit to balance the tradeoff, contribute to making rational VM deployment strategies and efficient server assignment strategies for cloud services.

V. Conclusion

Over the last few years, cloud computing has been widely deployed to provide various online services (i.e., cloud services). To achieve the efficient use of cloud infrastructure, the cloud provider must estimate multiple important metrics of cloud services, particularly, reliability, performance, and power consumption. This is a difficult issue as these metrics are indeed correlated. Most of the existing research usually treats these metrics separately with no effective model to correlate the metrics, which makes it difficult for the cloud provider to further develop a rational resource scheduling strategy for balancing the tradeoff between the correlated metrics.

This paper is original in that it systematically studies the correlations amongst reliability, performance, and power consumption of a cloud service. The logical structure consisting of the resource layer, application layer, and management layer is designed to capture the existing correlations. The hierarchical modeling based on this logical structure makes the evaluation of correlated metrics clear and tractable by identifying the resource usage as critical interfaces between the layers. Markov models, queuing theory, and Bayesian theory are mainly used to build the presented hierarchical model.

The numerical examples in this paper illustrated the procedures for modeling, analyzing, and evaluating the reliability, performance, and power metrics of a cloud service. These examples also showed that the presented correlation model can effectively help the cloud provider with a performance-power tradeoff analysis. The typical resource scheduling strategies described in the examples are representative in realistic cloud scenarios. The presented model can also be extended to evaluate the reliability, performance and power metrics for a more complicated public cloud scenario with numerous different cloud services. Generally speaking, an optimal resource scheduling strategy in such a complicated and large-scale cloud scenario can be described as a multiobjective optimization problem, which is usually an NP complete problem and thus needs heuristic algorithms to find approximate optimal solutions. The design of resource scheduling strategies for optimizing the correlated metrics in more complicated cloud scenarios is an important topic that will be studied in our future work.

Appendix

The following lists the steps of the convergent searching algorithm for the optimization model (41).

Step 1: Initialize $N = 1$ and compute $G_1(1)$ and $G_2(1)$ from (28) and (36), respectively.

Step 2: Increase N by one at each iteration until $N_{\max}$.

Step 3: Use (28) and (36) to obtain $G_1(N)$ and $G_2(N)$, and then calculate $\Delta G_1(N)$ and $\Delta G_2(N)$.

Step 4: If

$$\frac{\Delta G_1(N)}{\Delta G_2(N)} < \frac{c_p}{c_r} \tag{42}$$

then continue to the following step 5. Otherwise, repeat steps 2–4.

Step 5: The optimal solution of (41) is $N^* = N - 1$.

A. Convergence

For the situation that $N_{\max}$ can be seen as infinity, we can find that $\lim_{N\to\infty} \Delta G_1(N) = 0$. Note that $\Delta G_2(N)$ is a constant, thus $\lim_{N\to\infty} \Delta G_1(N)/\Delta G_2(N) = 0$. There must exist a finite integer N_0 to satisfy the inequality (42). For the other situation that $N_{\max}$ has a certain value, the algorithm is naturally stoppable (reaches $N_{\max}$ at most). Thus, the searching algorithm is convergent without an infinite loop.

B. Optimality

The inequality (42) means that the increased profit caused by adding one server is less than the additional cost due to running the server. Thus, the server should not be added since it must lead to the decrease of the pure profit, and $N^* = N - 1$ must be the optimal solution of the optimization model. Specially, if the pure profit function is observed to be a monotonously decreasing function of N, the optimal solution can be directly obtained as $N^* = 1$.

Acknowledgment

The authors would like to thank the anonymous referees and the Associate Editor for their constructive comments that further improved this paper.

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 Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, Jun. 2009.

[2] P. Barham *et al.*, "Xen and the art of virtualization," *ACM SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, Dec. 2003.

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

[4] L. A. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[5] X. Fan, W. D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proc. 34th Annu. Int. Symp. Comput. Archit.*, New York, NY, USA, 2007, pp. 33–37.

[6] J. Baliga, R. W. Ayre, K. Hinton, and R. Tuchker, "Green cloud computing: Balancing energy in processing, storage, and transport," *Proc. IEEE*, vol. 99, no. 1, pp. 149–167, Jan. 2011.

[7] A. Berl *et al.*, "Energy-efficient cloud computing," *Comput. J.*, vol. 53, no. 7, pp. 1045–1051, Sep. 2010.

[8] E. Gelenbe and R. Lent, "Energy-QoS trade-offs in mobile service selection," *Future Internet*, vol. 5, no. 2, pp. 128–139, Sep. 2013.

[9] D. Kliazovich, P. Bouvry, and S. U. Khan, "DENS: Data center energy-efficient network-aware scheduling," *Cluster Comput.*, vol. 16, no. 1, pp. 65–75, Mar. 2013.

[10] Y. Xia, M. C. Zhou, X. Luo, S. C. Pang, and Q. Zhu, "A stochastic approach to analysis of energy-aware DVS-enabled cloud datacenters," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 1, pp. 73–83, Jan. 2015.

[11] 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 *Proc. 16th IEEE Pac. Rim Int. Symp. Depend. Comput.*, Tokyo, Japan, 2010, pp. 125–132.
[12] K. Lakshmanan, R. Rajkumar, and J. P. Lehoczky, "Partitioned fixed-priority preemptive scheduling for multi-core processor," in *Proc. ECRTS*, Dublin, Ireland, 2009, pp. 239–248.
[13] G. D. Micheli and R. K. Gupta, "Hardware/software co-design," *Proc. IEEE*, vol. 85, no. 3, pp. 349–365, Mar. 1997.
[14] M. Xie, Y. S. Dai, and K. L. Poh, *Computing Systems Reliability: Models and Analysis*. New York, NY, USA: Kluwer, 2004.
[15] K. S. Trivedi, *Probability and Statistics With Reliability, Queuing, and Computer Science Applications*. New York, NY, USA: Wiley, 2001.
[16] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A comparison of high-level full-system power models," in *Proc. HotPower*, Berkeley, CA, USA, 2008, p. 3.
[17] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proc. 1st ACM Symp. Cloud Comput.*, New York, NY, USA, 2010, pp. 39–50.
[18] P. Bohrer *et al.*, "The case for power management in Web servers," in *Power Aware Computing*. New York, NY, USA: Springer, 2002, pp. 261–289.
[19] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via look ahead control," *Cluster Comput.*, vol. 12, no. 1, pp. 1–15, Mar. 2009.
[20] I. Takouna, W. Dawoud, and C. Meinel, "Accurate multi-core processor power models for power-aware resource management," in *Proc. 9th IEEE Int. Conf. Auton. Secure Comput.*, Sydney, NSW, Australia, 2011, pp. 419–426.
[21] V. K. P. Kumar, S. Hariri, and C. S. Raghavendra, "Distributed program reliability analysis," *IEEE Trans. Softw. Eng.*, vol. 12, no. 1, pp. 42–50, Jan. 1986.
[22] D. J. Chen and T. H. Huang, "Reliability analysis of distributed systems based on a fast reliability algorithm," *IEEE Trans. Parallel Distrib. Syst.*, vol. 3, no. 2, pp. 139–154, Mar. 1992.
[23] C. R. Das and J. Kim, "A unified task-based dependability model for hypercube computers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 3, no. 3, pp. 312–324, May 1992.
[24] Z. T. Kalbarczyk, R. K. Iyer, S. Bagchi, and K. Whisnant, "Chameleon: A software infrastructure for adaptive fault tolerance," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 6, pp. 560–579, Jun. 1999.
[25] H. Cui, Y. Li, J. Chen, and Y. Liu, "Methods with low complexity for evaluating cloud service reliability," in *Proc. 16th Int. Symp. Wireless Pers. Multimedia Commun.*, Atlantic City, NJ, USA, 2013, pp. 1–5.
[26] Z. Zheng, T. C. Zhou, M. R. Lyu, and I. King, "Component ranking for fault-tolerant cloud applications," *IEEE Trans. Services Comput.*, vol. 5, no. 4, pp. 540–550, Oct. 2012.
[27] Y. S. Dai, B. Yang, J. Dongarra, and G. Zhang, "Cloud service reliability: Modeling and analysis," in *Proc. 15th IEEE Pac. Rim Int. Symp. Depend. Comput.*, Shanghai, China, Nov. 2009, pp. 1–17.
[28] J. F. Meyer, "On evaluating the performability of degradable computing systems," *IEEE Trans. Comput.*, vol. 100, no. 8, pp. 720–731, Aug. 1980.
[29] R. M. Smith, K. S. Trivedi, and A. V. Ramesh, "Performability analysis: Measures, an algorithm, and a case study," *IEEE Trans. Comput.*, vol. 37, no. 4, pp. 406–417, Apr. 1988.
[30] K. R. Pattipati, Y. Li, and H. A. P. Blom, "A unified framework for the performability evaluation of fault-tolerant computer systems," *IEEE Trans. Comput.*, vol. 42, no. 3, pp. 312–326, Mar. 1993.
[31] B. Yang, F. Tan, and Y. S. Dai, "Performance evaluation of cloud service considering fault recovery," *J. Supercomput.*, vol. 65, no. 1, pp. 426–444, Jul. 2013.
[32] J. Subirats and J. Guitart, "Assessing and forecasting energy efficiency on cloud computing platforms," *Future Gener. Comput. Syst.*, vol. 45, pp. 70–94, Apr. 2015.
[33] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resource in cloud computing systems," *J. Supercomput.*, vol. 60, no. 2, pp. 268–280, May 2012.

Xiwei Qiu received the B.S. degree in electronic and information engineering from Jilin University, Changchun, China, in 2004, and the M.S. degree in software engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2009, where he is currently pursuing the Ph.D. degree in computer science with the School of Computer Science and Engineering.

His current research interests include cloud computing, reliability modeling and optimization, and energy-efficient computing.

Yuanshun Dai (M'03) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2000, and the Ph.D. degree in system engineering from the National University of Singapore, Singapore, in 2003.

He is currently the Dean of the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, China, where he is also a Chaired Professor and the Director of the Collaborative Autonomic Computing Laboratory. He has been the Chairman of the Professor Committee in the School of Computer Science and Engineering, UESTC since 2012 and the Associate Director of the Youth Committee of the "National 1000er Plan" in China. His current research interests include cloud computing, dependability, security, big data, and autonomic computing. He has published over 100 papers and five books, with 62 papers indexed by SCI, including 31 IEEE/ACM Transactions papers.

Prof. Dai was the Program Chair of the 12th IEEE Pacific Rim Symposium on Dependable Computing. He was also the Founder and the General Chair of the IEEE Symposium on Dependable Autonomic and Secure Computing. He served as a Guest Editor of the IEEE TRANSACTIONS ON RELIABILITY. He is on the editorial boards of several journals.

Yanping Xiang received the B.S. and M.S. degrees in computer science from Shanghai Jiao Tong University, Shanghai, China, in 1993 and 1996, respectively, and the Ph.D. degree in system engineering from the National University of Singapore, Singapore, in 2003.

She was a Research Fellow with the Industrial and System Engineering Department, National University of Singapore. In 2008, she joined the School of Computer Science, University of Electronic Science and Technology of China, Chengdu, China, where she is currently a Professor with the Collaborative Autonomic Computing Laboratory. Her current research interests include cloud computing, decision making, reliability, and autonomic computing.

Liudong Xing (S'00–M'02–SM'07) received the B.E. degree in computer science from Zhengzhou University, Zhengzhou, China, in 1996, and the M.S. and Ph.D. degrees in electrical engineering from the University of Virginia, Charlottesville, VA, USA, in 2000 and 2002, respectively.

She is a Professor with the Department of Electrical and Computer Engineering, University of Massachusetts (UMass), Dartmouth, MA, USA. Her current research interests include reliability modeling and analysis of complex systems and networks.

Prof. Xing was a recipient of the UMass Dartmouth Leo M. Sullivan Teacher of the Year Award in 2014, the Scholar of the Year Award in 2010, the Outstanding Women Award in 2011, and the 2007 IEEE Region 1 Technological Innovation (Academic) Award and the co-recipient of the Best Paper Award at the 2009 IEEE International Conference on Networking, Architecture, and Storage. She is an Associate Editor of the *International Journal of Systems Science*, the *International Journal of Systems Science: Operations and Logistics*, and the *Journal of Computational Engineering*. She is also an Assistant Editor-in-Chief of the *International Journal of Performability Engineering*, and an Editorial Board Member of *Reliability Engineering and System Safety*.