


On dynamic performance estimation of fault-prone Infrastructure-as-a-Service clouds

International Journal of Distributed
Sensor Networks
2017, Vol. 13(7)
© The Author(s) 2017
DOI: 10.1177/1550147717718514
journals.sagepub.com/home/ijdsn


Wanbo Zheng¹, Yuandou Wang¹, Yunni Xia¹, Quanwang Wu¹, Lei Wu²,
Kunyin Guo¹, Weiling Li¹, Xin Luo³ and Qingsheng Zhu¹

Abstract

The cloud computing paradigm enables elastic resources to be scaled at run time satisfy customers' demand. Cloud computing provisions on-demand service to users based on a pay-as-you-go manner. This novel paradigm enables cloud users or tenant users to afford computational resources in the form of virtual machines as utilities, just like electricity, instead of paying for and building computing infrastructures by their own. Performance usually specified through service level agreement performance commitment of clouds is one of key research challenges and draws great research interests. Thus, performance issues of cloud infrastructures have been receiving considerable interest by both researchers and practitioners as a prominent activity for improving cloud quality. This work develops an analytical approach to dynamic performance modeling and trend prediction of fault-prone Infrastructure-as-a-Service clouds. The proposed analytical approach is based on a time-series and stochastic-process-based model. It is capable of predicting the expected system responsiveness and request rejection rate under variable load intensities, fault frequencies, multiplexing abilities, and instantiation processing times. A comparative study between theoretical and measured performance results through a real-world campus cloud is carried out to prove the correctness and accuracy of the proposed prediction approach.

Keywords

Infrastructure-as-a-Service cloud, performance prediction, Pareto distribution

Date received: 30 March 2017; accepted: 8 June 2017

Academic Editor: Sang-Woon Jeon

Introduction

Cloud computing, as an emerging technology, is featured by the ability of elastic provisioning of on-demand computing resources ranging from applications to storage over the Internet on a pay-per-use manner. Cloud computing brings in numerous benefits for companies and end customers. For example, end customers can invoke computational resources for almost all possible types of workload when resources are reachable. This erases the conventional need for IT (information technology) administrators to build, provision, and maintain resources. Moreover, companies can dynamically scale upward or downward to meet the fluctuations of varying demands. Therefore,

¹Chongqing Key Laboratory of Software Theory and Technology, Chongqing University, Chongqing, China

²School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, China

³Chinese Academy of Sciences, Chongqing Institute of Green and Intelligent Technology, Chongqing, China

Corresponding authors:

Yunni Xia, Chongqing Key Laboratory of Software Theory and Technology, Chongqing University, Chongqing 400030, China.
Email: xiayunni@hotmail.com

Quanwang Wu, Chongqing Key Laboratory of Software Theory and Technology, Chongqing University, Chongqing 400030, China.
Email: wqw@cqu.edu.cn

Kunyin Guo, Chongqing Key Laboratory of Software Theory and Technology, Chongqing University, Chongqing 400030, China.
Email: css@cqu.edu.cn



Creative Commons CC-BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License

(<http://www.creativecommons.org/licenses/by/4.0/>) which permits any use, reproduction and distribution of the work without

further permission provided the original work is attributed as specified on the SAGE and Open Access pages (<http://www.uk.sagepub.com/aboutus/openaccess.htm>).

investments into physical computing infrastructures, which are substituted by virtual environments connected to cloud data centers, are thus no longer necessary. In this way, computational resources are priced at a granular level, allowing customers to pay only for what they actually require and use. Through the provision of on-demand access to hardware/software/communication/storage components, clouds deliver services at multiple levels: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). IaaS clouds¹ provision customers or tenant users virtual machine (VM) instances as computational resources. These instances are created and executed in the owner data center. In contrast, PaaS and SaaS use the web to deliver platforms and applications that are managed by a third-party vendor and whose interface is accessed on the clients' side. PaaS and SaaS are typically used by application developers, while IaaS is used by network architects. In other words, PaaS and SaaS provide a mechanism to develop applications, whereas IaaS provides only the infrastructure necessary to run the code developed by application developers.

Cloud users consistently concern about the quality of cloud service delivered. Thus, performance promised by cloud providers is crucial to the success of cloud service offered. A major performance metric frequently used is expected response delay. When running mission-critical web-facing applications in cloud environments, one of the major requirements is the expected response time, usually specified as the expected waiting interval between a request submission and its requested outcome returned. Specifically, this work considers expected instantiation response delay, that is, the expected waiting interval between a cloud request arrival and its corresponding VM instance created, as the metric of system responsiveness. Note that instantiation-creation-time differs from task execution time in that the latter is actually the sum of VM instantiation response delay and VM usage time. VM usage time can be application-dependent and vary from one request to another. Thus, task execution time of IaaS cloud is hard to model and predict. Moreover, the last thing cloud users want to see is the rejection of their requests. We therefore consider request rejection rate as another important performance metric, which directly determines users' satisfaction. This rate is expressed by the probability that a submitted request receives no response or a negative response (cloud system being unable to instantiate its VM request). Note that the two performance metrics are determined by various system factors and parameters, for example, service rates, buffer scale, multiplexing ability, and fault rates. As will be shown later in this work, a successful VM instantiation on IaaS cloud has to go through several provisioning steps, each of which can be subject to unexpected faults. Fault-handling activities are thus inevitable and such

activities could strongly affect the overall cloud performance. However, a careful investigation into related works (discussed in next section) suggests that, for model simplicity, most existing approaches assume reliable and fault-free cloud provisioning and thus have less difficulty in theoretical derivation of performance metrics. Although some recent performance/quality of service (QoS) models consider fault/fault-handling, they rely on measurement/test data to analyze cloud performance. Measurement-based approaches have limited value in optimization and bottleneck analysis. In contrast, comprehensive theoretical models capable of modeling unreliable cloud provisioning with accurate prediction of performance metrics are in high need.

Instead of assuming reliable cloud and fault-free cloud provisioning by most existing related studies, we introduce a performance modeling and trend prediction framework for runtime performance prediction of unreliable IaaS cloud with faulty VM instantiation. For this purpose, a stochastic-process and time-series-based approach is developed, and product-form results of multiple performance metrics are derived. To prove the effectiveness and accuracy of the proposed stochastic model, we employ runtime performance data from an actual campus cloud built on OpenStack and show that runtime experimental performance results closely converge to predictive ones.

Related works

Recently, performance estimation of cloud computing systems is attracting increasing attention. Various model-driven methods in this direction are developed for analytical performance prediction of cloud computing systems and cloud data centers. For example, Xiong and Chen² work on percentile of job execution times and response delays and present an M/M/1 queue model, that is, a single-server queue with Poisson arrival, for QoS prediction of cloud services. It assumes reliable and fault-free cloud provisioning to simplify the derivation of performance results. Our recent work in Xia et al.³ and Li et al.,⁴ however, shows that faults exist at different levels of IaaS cloud and the overhead required to counter such faults intensively affects performance especially when cloud is heavily loaded. Wang et al.⁵ introduce a comprehensive performance model by considering machine error and repair. They also obtain the theoretical distributional functions of cloud task execution durations. He et al.⁶ consider a multivariate QoS model to analyze efficiency of VM-PM (physical machine) mapping strategies with reliable cloud provisioning. Bruneo and colleagues^{7,8} model the aging process of cloud PMs and derive performance results under different request load intensities. Our work differs in the ways below: (1) Bruneo and

colleagues^{7,8} consider reliable provisioning but our work captures fault/fault-handling activities; (2) Bruneo and colleagues^{7,8} ignore waiting periods, which are actually non-negligible. Ghosh et al.⁹ develop a scalable performance and energy prediction model for describing resource placement, dynamic speed scaling, and job failure. To reduce model complexity, their model decomposes the provisioning process into sub phases and obtains performance results of each phase separately. Finally, an integration method composes performance results of separate phases together. For simplicity and tractability, the interacting performance model is subject to accuracy loss because such separate phases of IaaS clouds actually have inter-dependence with each other. Our framework, instead, considers a monolithic provisioning control-flow model where all provisioning activities and phases co-decide final performance. Khazaei et al.^{10,11} develop a more refined work using the Continuous-time Markov chain, that is, CTMC model. They derive closed-form expressions of executing times and failure rate as the QoS metrics but use the fault-free assumption.

In comparison with the above-mentioned work, the contributions of this work are multifold: (1) instead of assuming reliable cloud provisioning, we consider a fault-prone one and propose a stochastic model to quantify cloud performance under variable fault frequencies, load intensities, multiplexing abilities, and instantiation processing distributions; (2) instead of using separate modeling approaches at the cost of accuracy loss, we consider that provisioning steps work simultaneously to determine the final performance and develop a monolithic performance model; (3) instead of static performance analysis and assumptions of Markovian processes (It is worth noting that although various works assume general distributions of performance results, these works are still limited. For analyzing the distributional functions of performance metrics, Hwang et al.¹² and Xia and colleagues^{13,14} consider the historical empirical distribution to be the theoretical distribution. By doing so, they consider the historical empirical distribution to be able to describing the future fluctuations of performance. A major limitation of doing so lies in that the historical empirical distribution merely employs the density of samples as their distributional probabilities but ignores its trend. As a case in point discussed in our earlier work,¹⁵ a distribution of response delay with increasing occurrences of long delays with time clearly suggests a deteriorating performance which is caused by, for example, increasing failures of cloud components. However, its empirical distribution may be quantitatively identical to that of another response delay type with stable occurrences of long delays with time. Such limitation of using the historical empirical distribution could be well avoided using a time-series-based analysis instead.), we

introduce a dynamic prediction approach (using the Autoregressive-Moving-Average-Model (ARMA)¹⁶ series model) with special attention to the runtime trend of cloud performance in real-world environments; (4) based on real-world performance data, we compare the theoretical and empirical performance results to validate the correctness and accuracy of our proposed performance model.

The ARMA model

ARMA¹⁶ models aim at modeling and predicting performance of computing,¹⁷ communication, traffic control manufacturing system, and so on. It captures the dynamic feature of a time-varying distribution by employing an integration of an autoregressive (AR) type and a moving average (MA) type.

For a series X_t with stationary trends, an $ARMA(p, q)$ process can be defined as

$$\phi(B)X_t = \theta(B)Z_t \quad (1)$$

where $\phi(B)$ indicates the polynomial with degree p , $\theta(B)$ indicates the polynomial with order q , and Z_t is assumed to be independently and identically distributed with zero mean and variance σ^2

$$\begin{aligned} \phi(B) &= 1 - \phi_1 B - \dots - \phi_p B^p \\ \theta(B) &= 1 + \theta_1 B + \dots + \theta_q B^q \end{aligned} \quad (2)$$

and B is the back-shift operator which is defined as

$$B^i X_t = X_{t-i} \quad (3)$$

where $i = 0, \pm 1, \pm 2, \dots$

The Box-Jenkins method is highly effective in prediction of the future value of an established $ARMA(p, q)$ series, on condition that series are stationary. A time-series can be seen as stationary on condition that its residuals are statistically independent of each other and constant in mean and variance over time. For example, the time-series is stationary if its fluctuation has a constant mean value without a trend.

The Box-Jenkins-based prediction is carried out through the following steps:

- Series stationarity;
- Model identification;
- Optimization and selection;
- Model diagnostic checking.

As discussed earlier, performance of IaaS cloud systems is mainly decided by instantiation processing time, request arrival rate, fault frequency, and the physical capacity, among which instantiation processing time is the most varying and fluctuating one. We can feed the historical instantiation processing time-series into the

ARMA model and obtain the predicted future instantiation processing time (as shown in Figures 4–6), that is, T_I . T_I serves as fluctuating input parameter in the stochastic performance model. Note that other input parameters are considered to be stable according to our experimental performance test, and thus, we employ them as non-fluctuating inputs.

System view and its stochastic modeling

The IaaS cloud paradigm is featured by elastic provisioning of virtualized computing entities on the web. In a typical IaaS cloud, third-party providers own hardware, software, communication, storage, system maintenance, backup, resiliency planning, and other infrastructure entities for its customers. The capability of IaaS cloud provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud physical infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components. IaaS cloud offers highly dynamic resources that can be invoked and used on-demand at run time. Such elasticity is well suited for highly fluctuating workload and unexpected increase in request intensity.

In a typical IaaS cloud architecture, a cloud management unit is responsible for admitting incoming requests. The request input flow can be usually

captured by the input rate, λ . We use c to represent the size of the admission buffer. Such size can be determined before use (e.g. such size can be determined by the FRAME-SIZE filed in OpenStack). Requests incoming may depart according to rate θ or enter the retrial step by rate $1 - \theta$ if the admission buffer is fully occupied. The cloud management unit is responsible for processing requests out of the admission buffer based on the first-in-first-output style and trying to instantiate corresponding VMs on PMs, that is, PMs. For the performance prediction purpose, we need to evaluate instantiation response delay, that is, the expected time that a request takes to see its corresponding VM created and ready for invocation.

In OpenStack, the time-stamp of VM creation can be obtained in the INSTANCE-SPAWNED property. As suggested by Figure 1, the above process involves multiple phases and interactions. The last phase is the time required for the cloud management unit to create a VM, that is, instantiation processing time. In OpenStack, such time is denoted by the interval between INSTANCE-BUILDING and INSTANCE-SPAWNED. As mentioned earlier, instantiation processing time is highly sensitive to system load and network connectivity and thus strongly fluctuating with time. We thus employ its predicted value through the ARMA method as the input into the stochastic model presented in the next section.

With the support of VM multiplexing¹⁸ powered by modern multi-core/multi-threading technologies, more than one VM can be instantiated on a single PM. However, such number is usually bounded and we thus

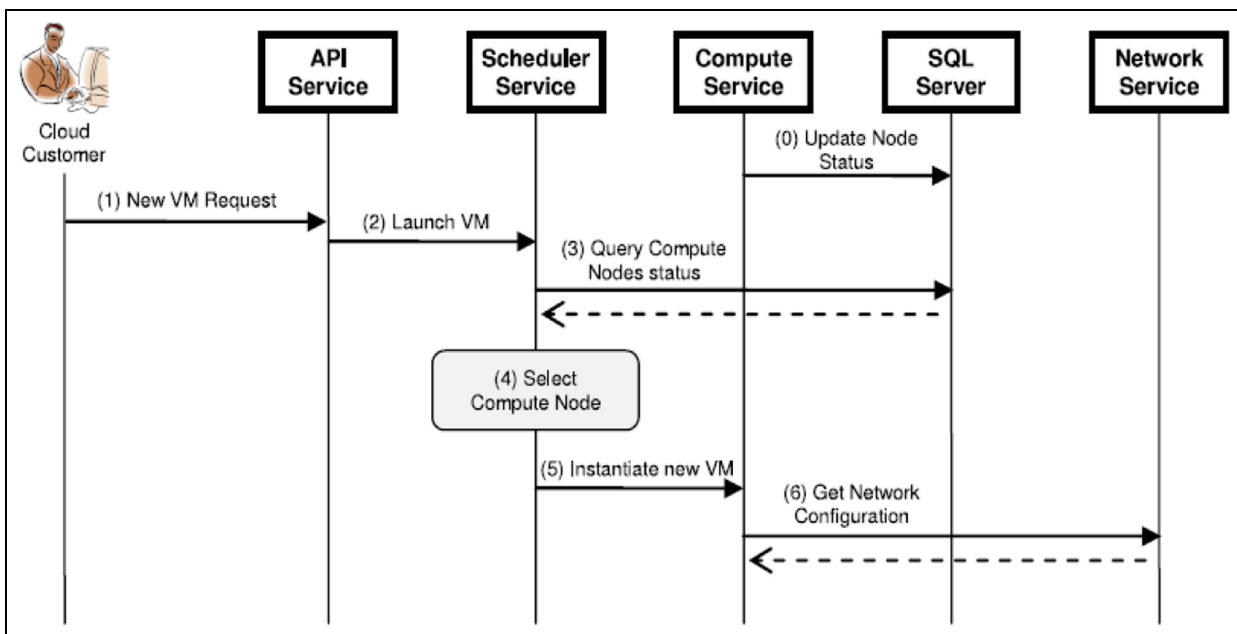


Figure 1. Sequence chart of VM instantiation in OpenStack.

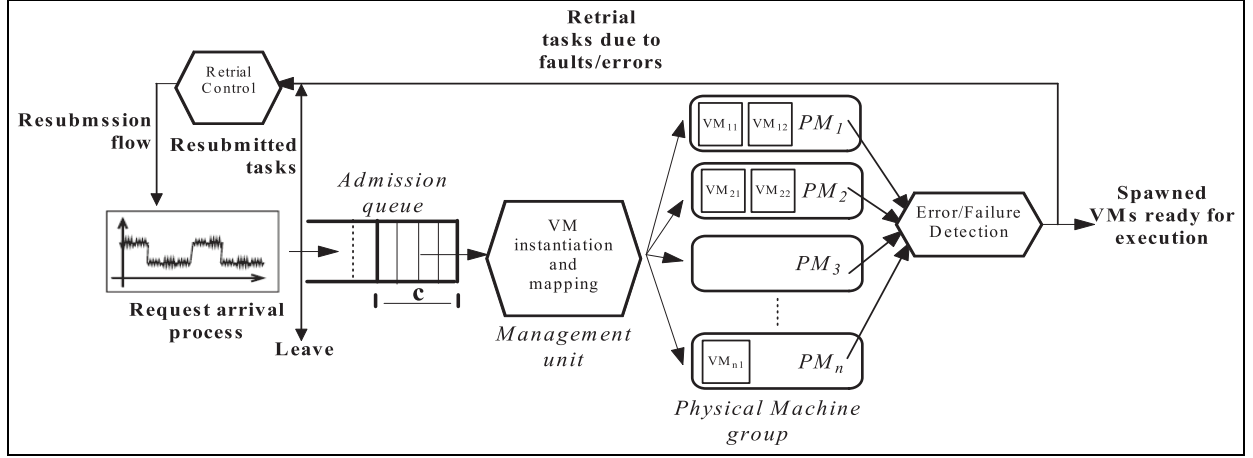


Figure 2. IaaS cloud provisioning control flow.

use m to denote such bound. Note that m denotes the maximum number of VMs being instantiated rather than the number of VMs being executed. Such bound can be specified through, for example, OpenStack scripts. VM multiplexing helps to achieve considerable resource saving in comparison with individual VM-based resource provisioning. However, a high multiplexing level is not necessarily profitable due to the fact that process interference can lead to performance and stability deterioration.

As shown in Figure 1, all the provisioning phases are subject to faults, and unsuccessful VM instantiation can thus be inevitable since internal or external communications are sometimes fault-prone. Such faults are usually caused by, for instance, temporary connection failure to remote database, communication congestion, inappropriate input/output sequence, gateway failure, database access failure, and unexpected user quit. The overhead required to counter such faults, for example, compensation/transactional rollback/ re-instantiation, can have intensive impact on system performance. This is especially evident when system is heavily loaded as discussed later in the section of case study.

Based on the above observations, the control-flow view of cloud provisioning on fault-prone IaaS cloud, in terms of a queuing network, is shown in Figure 2. This model erases implementation contents of cloud infrastructures while focusing on the control-flow contents adequate for stochastic modeling. Based on the stochastic control-flow model, the following section shows how closed-form expression of performance metrics can be derived and the theoretical effects of changing input loads (request load, fault load, and capacity load).

Performance prediction

We denote ρ as the equivalent cloud utilization

$$\rho = \frac{\lambda' \times \mathbb{E}(T_I)}{n \times m} \quad (4)$$

where T_I denotes the predicted instantiation processing time using the ARMA method (its predicted values are illustrated later in Figures 4–6), $\mathbb{E}(T_I)$ is the expectation of T_I , and λ' is the effective request arrival rate. Note that λ' is different from λ since resubmitted requests join the original arrival flow.

It has been widely recognized that heavy-tailed distributions are well suited for modeling job processing and request handling activities in computer systems and networks. The Pareto distribution is often presented in terms of its survival function (or reliability function, or tail function), which gives the probability of seeing larger values than a given value. Thus, we employ it for the VM instantiation processing time. In particular, we choose the Pareto distribution where T_I is lower bounded by \hat{x} . The tail of T_I can therefore be obtained as

$$\mathbb{P}\{T_I \geq \hat{x}\} = \left[\frac{x}{\hat{x}}\right]^{-v} \quad (5)$$

where $x \geq \hat{x}$ and $v \in \{1, 2\}$. The above equation suggests that it is more likely to observe a longer instantiation processing time than the light-tailed one and its expectation is

$$\mathbb{E}(T_I) = \frac{v \times \hat{x}}{v - 1} \quad (6)$$

Let R denote the rejection rate due to the capacity limit, we have

$$R = \frac{1}{(nm)^b \times (nm)!} (\lambda' \times \mathbb{E}(T_I))^{nm+b} P_0 \quad (7)$$

where P_0 denotes the probability that the waiting buffer is empty

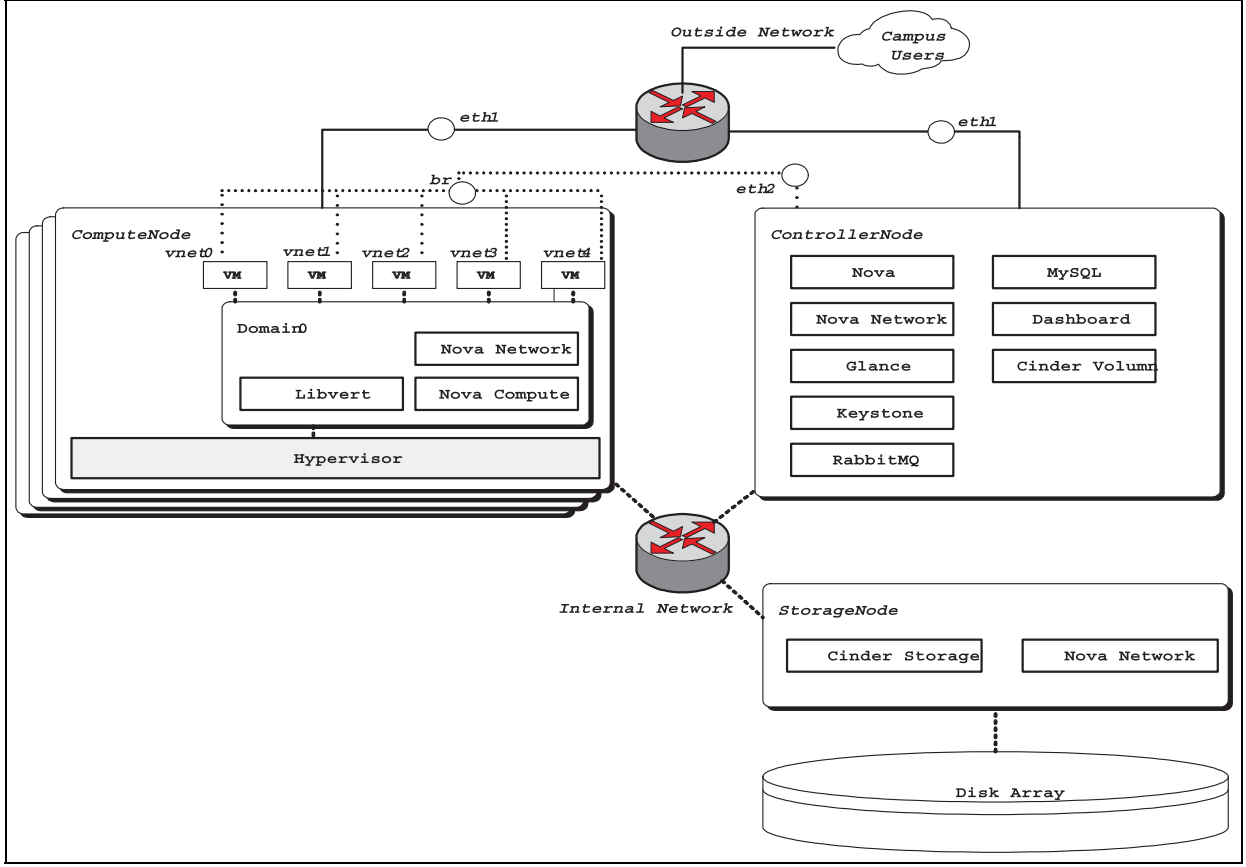


Figure 3. The architectural view of the campus cloud.

$$P_0 = \left[\sum_{i=0}^{nm-1} \frac{(\mathbb{E}(T_I) \times \lambda')^i}{i!} + \sum_{i=nm}^{nm+b} \frac{(\mathbb{E}(T_I) \times \lambda')^i}{(nm)^{i-nm} (nm)!} \right]^{-1} \quad (8)$$

Let λ'' denote the resubmission rate due to the capacity limit, we have

$$\frac{\lambda''}{\lambda'} = \frac{R \times (1 - \theta)}{1 - R} \quad (9)$$

λ'' is consequently be expressed as

$$\lambda'' = \lambda' \times \frac{R \times (1 - \theta)}{1 - R} \quad (10)$$

By combining the results derived above, we have

$$\begin{aligned} \lambda' &= (\lambda' \times f + \lambda'' + \lambda) \times (1 - R) \\ &= \left(\lambda' \times f + \lambda' \times \frac{R \times (1 - \theta)}{1 - R} + \lambda \right) \times (1 - R) \end{aligned} \quad (11)$$

Note that R is determined by λ'' , b , m , n , and T_I according to equations (7) and (8). Thus, the above equation suggests that λ' can be calculated if λ , b , m , and n are given and the distribution of T_I is known. However, the product-form expression of λ' does not

exist. Fortunately, we can obtain its numerical solution using tools, for example, the `solve()` tool in **MATLAB**.

The instantiation response delay, T , is defined as the interval time between request arrival and the corresponding VM being successfully spawned and ready for execution. To analyze T , we first have to analyze the distribution of the time that a request resides in the IaaS cloud on condition that it never enters the rejection and retrial phases, T' . T' also stands for the sojourn time of a request on condition that it enters the request buffer once and only once.

Since the instantiation processing time, T_I , is heavy tailed, the tail of T' can be asymptotically approximated as

$$\mathbb{P}\{T' \geq \hat{t}\} \sim \mathbb{P}\{T_I \geq (1 - \rho)\hat{t}\} \quad (12)$$

In the following, we have to calculate the probability that a cloud request enters the retrial control unit, P_r

$$P_r = (1 - \theta)R + f(1 - R) \quad (13)$$

As can be seen from Figure 2, a successfully instantiated request may experience several retrials before its final successful instantiation due to the capacity limit and instantiation faults. The expected number

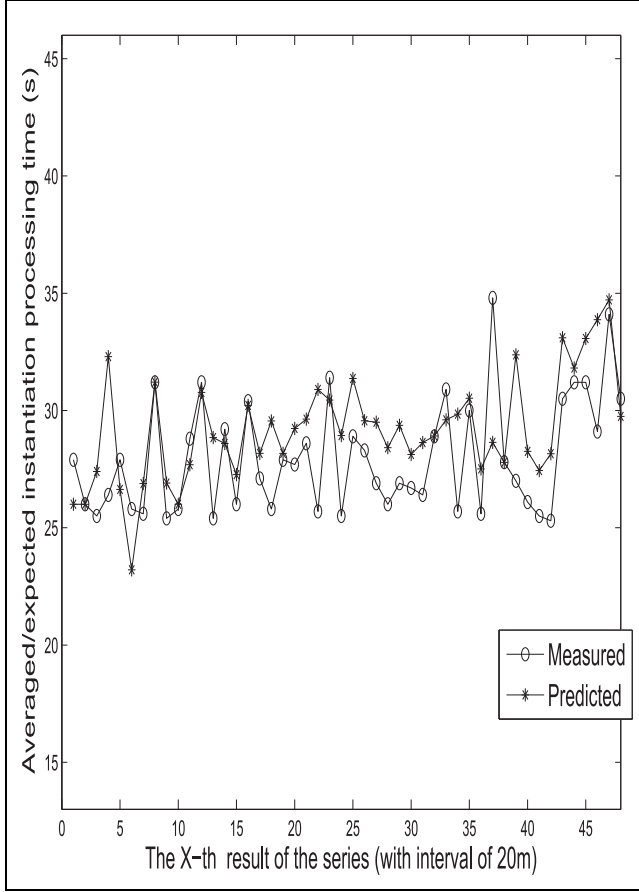


Figure 4. Predicted versus measured averaged instantiation time at high load.

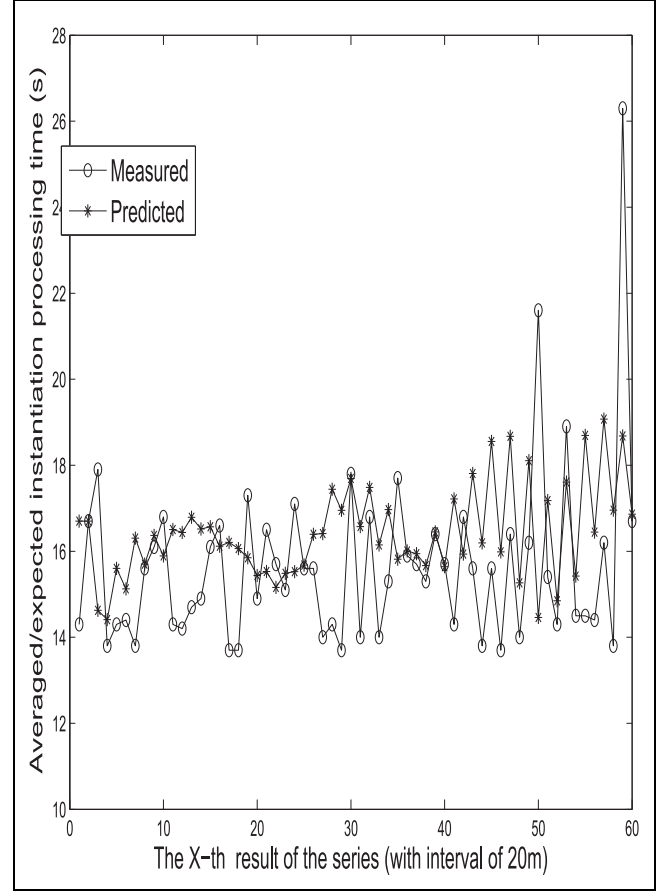


Figure 5. Predicted versus measured averaged instantiation time at medium load.

of retrials of a cloud request on condition that it is finally successfully instantiated, N_r , can therefore be obtained as

$$N_r = \frac{1}{1 - P_r} - 1 \quad (14)$$

N_r has two part, namely, the expected number of retrials due to the capacity limit, N_{rc} , and the expected number of retrials due to faulty instantiations, N_{rf}

$$N_{rc} = N_r \times \frac{f(1 - R)}{P_r} \quad (15)$$

$$N_{rf} = N_r \times \frac{(1 - \theta)R}{P_r} \quad (16)$$

Instantiation response delay can therefore asymptotically approach the sum of the time needed for retrials due to the capacity limit, the time needed for retrials due to faulty instantiations, and the time needed for the last successful trial

$$\mathbb{P}\{T \geq t\} \sim \mathbb{P}\{T' \times N_{rf} + d \times N_{rc} + T' \geq t\} \quad (17)$$

Case study and model validation

To prove the effectiveness of the prediction framework, we present a comparative study on a campus IaaS cloud. It is built following the IaaS architecture presented in Figure 2 and is developed by the XenServer and OpenStack toolkits. Figure 3 shows its physical architecture.

The cloud system bases itself on six Intel I450 servers (4-CPU Intel Xeon 5506/128G RAM/15TB storage but only 3-CPU/8G RAM/4TB storage is available for users), each of which serves as a PM. Each PM is capable of supporting eight VMs in parallel at most. The capacity of the waiting buffer for requests is 16 as specified in OpenStack. The fault rate is 0.13%–0.79% based on an experimental test conducted from 14:20 to 18:35 on 5 May 2016. In the test, the IaaS cloud continuously processes requests from students and tries to instantiate each request into a corresponding VM instance. Each VM is deployed on only one PM for data consistency and integrity. The test logfile covers time-stamps of each request's arrival, departure, failure, and successful instantiation times in consecutive periods.

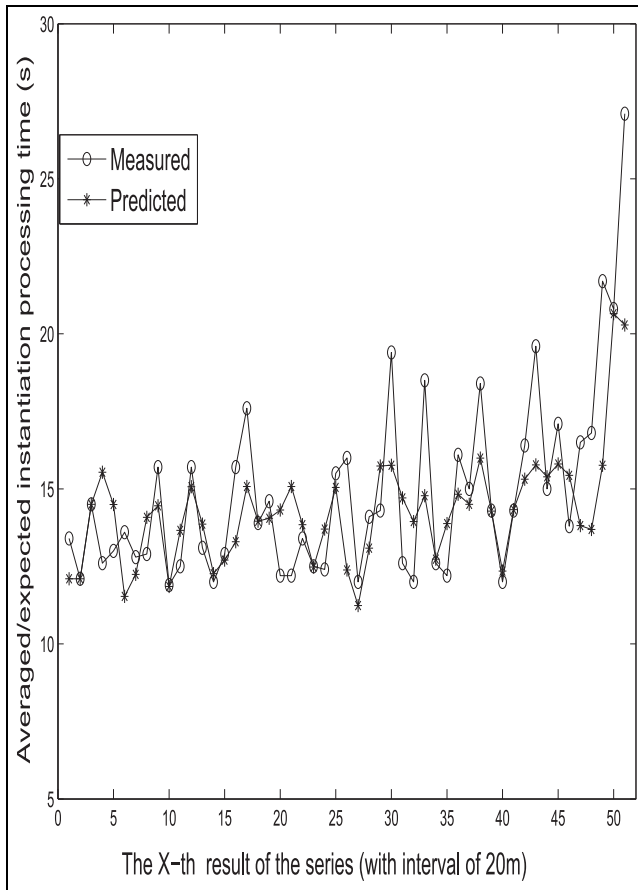


Figure 6. Predicted versus measured averaged instantiation time at low load.

The request arrival rate and instantiation processing time fluctuate and we thus employ the ARMA model to predict their future values as the inputs of the stochastic performance model. As discussed earlier, we consider instantiation processing time as a fluctuating random variable and thus employ its predicted future value (illustrated in Figures 4–6 using the ARMA method), T_I , as the model input into the stochastic performance model.

Measured performance results and their corresponding predicted ones are compared in Figures 7–12. As suggested, our proposed prediction approach achieves satisfactory trend-curve-fitting, and predicted performance results show good convergence to measured ones. Note that measured results show more fluctuations than predicted ones due to the fact that the ARMA method tries to converge to the overall trend with least squares regression and find the values of the parameters which minimize the error term. Individual fluctuations are thus less captured. However, as can be seen that the overall long-term trends of experimental results are well followed by predicted ones.

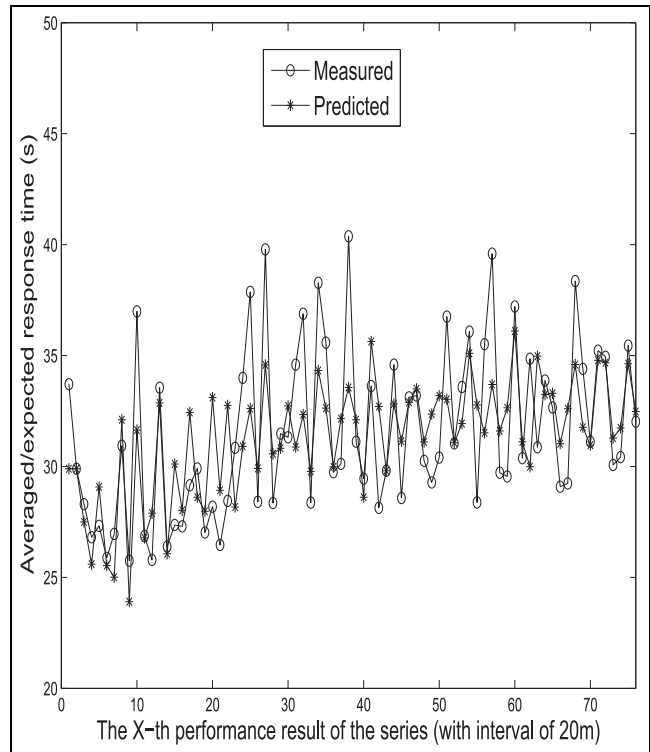


Figure 7. Predicted versus measured expected response delays at high load.

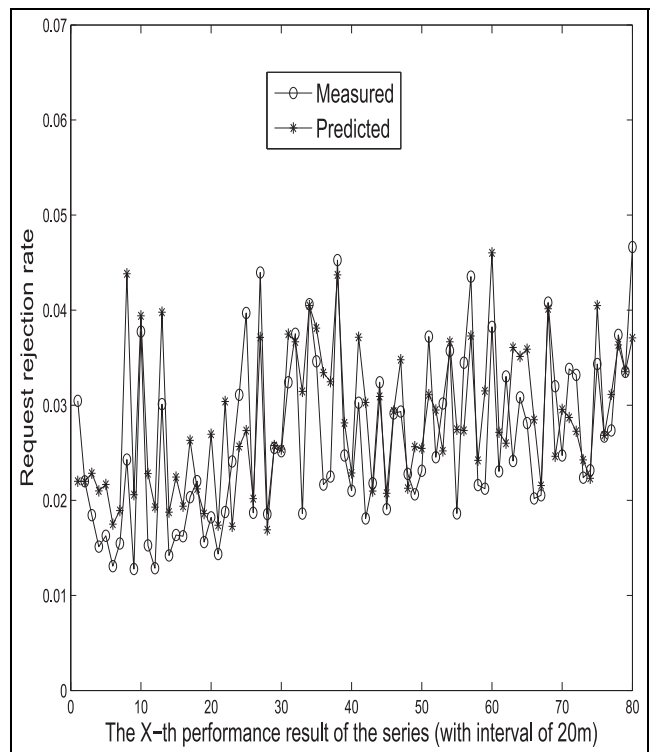


Figure 8. Predicted versus measured request rejection rates at high load.

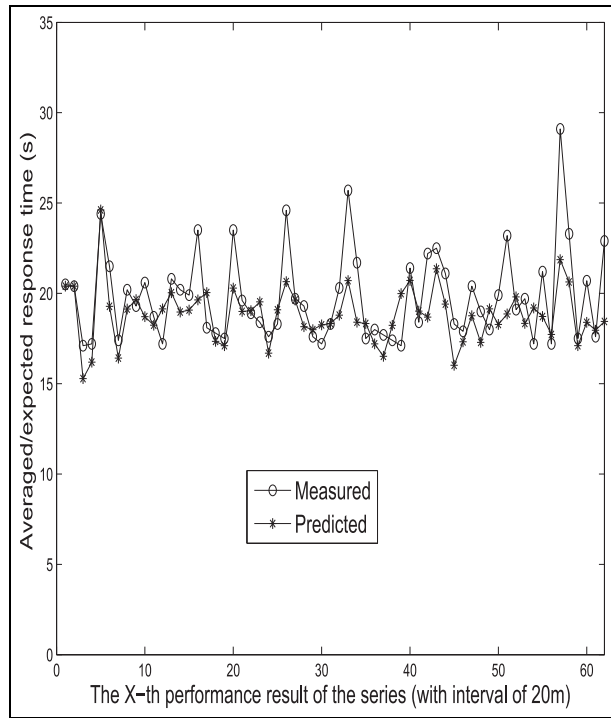


Figure 9. Predicted versus measured expected response delays at medium load.

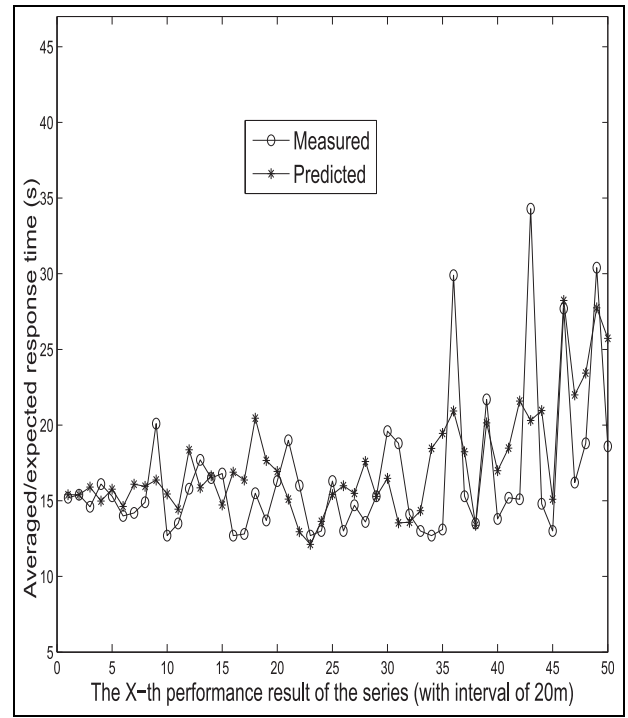


Figure 11. Predicted versus measured expected response delays at low load.

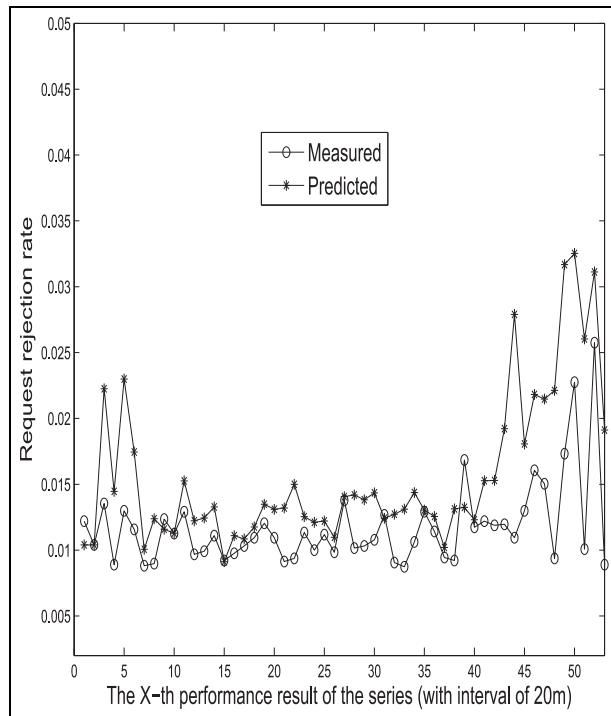


Figure 10. Predicted versus measured request rejection rates at medium load.

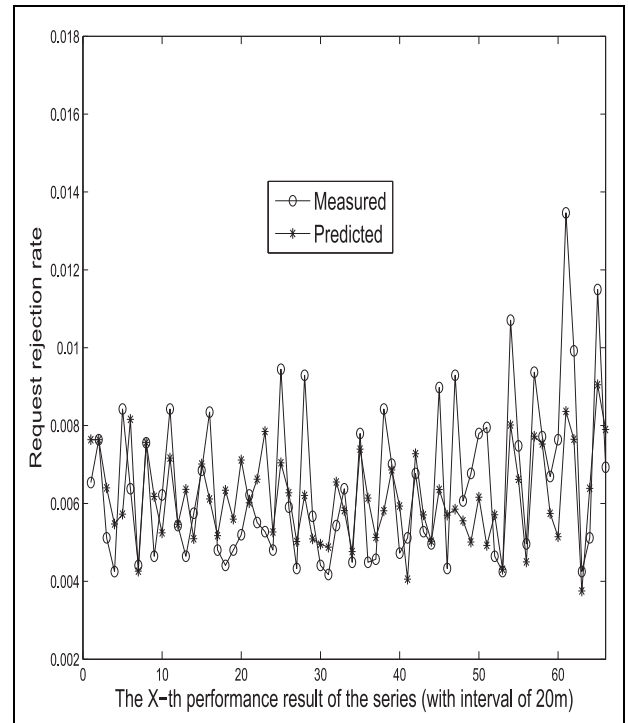


Figure 12. Predicted versus measured request rejection rates at low load.

Conclusion and further studies

In this work, we propose a comprehensive performance estimation and trend prediction approach for fault-prone IaaS clouds with request rejection and resubmission. Based on a stochastic queuing model and a ARMA time-series, our approach is capable of predicting request responsiveness and request rejection rate under variable system conditions. For the model validation purpose, in the case study based on a real-world campus cloud, we compare predicted and measured performance results and show their satisfactory convergence.

We intend to work on the following topics as our future research: (1) more, for example, throughput, mobility, and consolidation overhead should be considered; (2) Petri nets^{19–22} can be used as an alternative model formalism to facilitate structural reduction techniques and reduce computational complexity; (3) dynamic VM-PM-mapping strategies should be considered and modeled instead of static ones assumed in this article where PMs are equally likely to host incoming VM instances. According to a dynamic mapping strategy, the cloud management unit decides the mapping plan based on the utilization and working status of each PM at run time. However, theoretical performance estimation and derivation of response delay distribution of dynamic-mapping-based IaaS cloud can be very difficult.^{23,24}

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the NSFC under Grant No. 61472051, the Fundamental Research Funds for the Central Universities under Project Nos 106112014CDJZR185503 and CDJZR12180012, the Science Foundation of Chongqing under Nos cstc2014jcyjA40010 and cstc2014jcyjA90027, Chongqing Social Undertakings and Livelihood Security Science and Technology Innovation Project Special Program No. cstc2016shmszx90002, China Postdoctoral Science Foundation No. 2015M570770, Chongqing Postdoctoral Science special Foundation No. Xm2015078, and Universities' Sci-tech Achievements Transformation Project of Chongqing No. KJZH17104.

References

- Shiau WL and Chau PYK. Understanding behavioral intention to use a cloud computing classroom: a multiple model comparison approach. *Inform Manage* 2016; 53(3): 355–365.
- Xiong K and Chen X. Ensuring cloud service guarantees via Service Level Agreement (SLA)-based resource allocation. In: *Proceedings of the IEEE 35th international conference on distributed computing systems workshops (ICDCSW)*, Columbus, OH, 29 June–2 July 2015, pp.35–41. New York: IEEE.
- Xia Y, Zhou M, Luo X, et al. A comprehensive QoS determination model for infrastructure-as-a-service clouds. In: *Proceedings of IEEE international conference on automation science and engineering*, Madison, WI, 17–20 August 2013, pp.122–127. New York: IEEE.
- Li W, Wu L, Xia Y, et al. On stochastic performance and cost-aware optimal capacity planning of unreliable infrastructure-as-a-service cloud. In: *Proceedings of the international conference on algorithms and architectures for parallel processing*, Granada, 14–16 December 2016, pp.644–657. New York: IEEE.
- Wang C, Xing L, Wang H, et al. Processing time analysis of cloud services with retrying fault-tolerance technique. In: *Proceedings of the IEEE international conference on communications in China*, Beijing, China, 15–17 August 2012, pp.63–67. New York: IEEE.
- He S, Guo L, Ghanem M, et al. Improving resource utilization in the cloud environment using multivariate probabilistic models. In: *Proceedings of the IEEE 5th international conference on cloud computing*, Honolulu, HI, 24–29 June 2012, pp.574–581. New York: IEEE.
- Bruneo D. A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems. *IEEE T Parall Distr* 2014; 25(3): 560–569.
- Bruneo D, Distefano S, Longo F, et al. Workload-based software rejuvenation in cloud systems. *IEEE T Comput* 2013; 62(6): 1072–1085.
- Ghosh R, Naik VK and Trivedi KS. Power-performance trade-offs in IaaS cloud: a scalable analytic approach. In: *Proceedings of the IEEE/IFIP 41st international conference on dependable systems and networks workshops*, Hong Kong, China, 27–30 June 2011, pp.152–157. New York: IEEE.
- Khazaei H, Misisic JV, Misisic VB, et al. Analysis of a pool management scheme for cloud computing centers. *IEEE T Parall Distrib* 2013; 24(5): 849–861.
- Khazaei H, Misisic JV and Misisic VB. Performance of cloud centers with high degree of virtualization under batch task arrivals. *IEEE T Parall Distrib* 2013; 24(12): 2429–2438.
- Hwang SY, Wang H, Tang J, et al. A probabilistic approach to modeling and estimating the QoS of web-services-based workflows. *Inform Sci* 2007; 177(23): 5484–5503.
- Xia Y, Wan N, Dai G, et al. A non-Markovian stochastic Petri net-based approach to performance evaluation of ontology-based service composition. *Concurr Comp Pract E* 2012; 24(18): 2255–2267.
- Xia Y, Zhou M, Luo X, et al. Stochastic modeling and performance analysis of migration-enabled and error-prone clouds. *IEEE T Ind Inform* 2015; 11(2): 495–504.
- Xia Y, Ding J, Luo X, et al. Dependability prediction of WS-BPEL service compositions using Petri net and time series models. In: *Proceedings of the 2013 IEEE seventh international symposium on service-oriented system*

- engineering, Redwood City, CA, 25–28 March 2013, pp.192–202. New York: IEEE.
16. Dashevskiy M and Luo Z. Time series prediction with performance guarantee. *IET Commun* 2011; 5(8): 1044–1051.
 17. Branch P. ARMA(1,1) modeling of Quake4 Server to client game traffic. In: *Proceedings of the 6th ACM SIGCOMM workshop on network and system support for games*, Melbourne, VIC, Australia, 19–20 September 2007, pp.70–74. New York: ACM.
 18. Meng X. Efficient resource provisioning in compute clouds via VM multiplexing. In: *Proceedings of the 7th international conference on Autonomic computing*, Washington, DC, 7–11 June 2010, pp.11–20. New York: ACM.
 19. Bi J, Yuan H and Zhou M. A Petri net method for compatibility enforcement to support service choreography. *IEEE Access* 2016; 4: 8581–8592.
 20. Wang S, You D, Zhou M, et al. Characterization of admissible marking sets in Petri nets with uncontrollable transitions. *IEEE T Automat Contr* 2016; 61(7): 1953–1958.
 21. Yu W, Yan C, Ding Z, et al. Modeling and verification of online shopping business processes by considering malicious behavior patterns. *IEEE T Autom Sci Eng* 2016; 13(2): 647–662.
 22. Wang SG, You D and Zhou MC. A necessary and sufficient condition for a resource subset to generate a strict minimal siphon in S4PR. *IEEE T Automat Contr* 2017. DOI: 10.1109/TAC.2017.2677859.
 23. Wu QW and Ishikawa F. Heterogeneous virtual machine consolidation using an improved grouping genetic algorithm. In: *Proceedings of the HPCC/CSS/ICSS conferences*, New York, 24–26 August 2015, pp.397–404. New York: IEEE.
 24. Deng S, Huang L, Taheri J, et al. Zomaya: mobility-aware service composition in mobile communities. *IEEE Trans Syst Man Cybern: Syst* 2017; 47(3): 555–568.