Cloud Resource Provisioning and Bottleneck Eliminating for Meshed Web Systems

Yamin Lei*, Zhicheng Cai*†, Hang Wu* and Rajkumar Buyya †
*School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
† The Cloud Computing and Distributed Systems (CLOUDS) Laboratory,
School of Computing and Information Systems, The University of Melbourne, Australia
Corresponding Author: Zhicheng Cai (caizhicheng@njust.edu.cn)

***Abstract*—Most of existing resource provisioning methods are designed for traditional Web applications with linear structures. However, Web systems with the meshed topology are becoming widespread. Meshed connections among different tiers make Virtual Machine (VM) provisioning and bottleneck elimination complex. In this paper, a Jackson network based Proactive and Reactive VM auto-scaling Method (JPRM) is proposed. In JPRM, request transition behaviors among tiers are modeled as a finite-state Markov stochastic process. A transition probability matrix is studied on-line to predict resource requirements based on M/M/N queuing models as proactive control. For reactive provisioning, the final increased request rate of each tier is determined based on stable state checking and Jackson equilibrium equation solving to eliminate bottleneck tiers and avoid bottleneck shifting. The JPRM is evaluated in a simulation environment established using CloudSim. Experimental results show that JPRM avoids bottleneck shifting with reasonable additional VM rental costs compared with existing methods.**

***Index Terms*—Resource provisioning; Bottleneck eliminating; Meshed Web systems; Jackson network; Cloud computing**

I. Introduction

More and more Web system vendors migrate their applications to virtual data center built on elastically rented VMs [1]. Most existing works assume that Web systems have linear multi-tier structures. However, due to various machine learning algorithms have been used in big data processing to generate better service experiences [2], services provided by one Web system are becoming diverse which makes the Web system have a mesh structure. It is crucial to design Cloud resource auto-scaling methods for mesh-structured Web systems (called Meshed Web systems) to minimize resource rental costs and guarantee Quality of Service (QoS).

Mesh structures make the resource provisioning problems complex. Most existing works mainly focus on single-tier Web systems or single tiers of multi-tier Web systems. For multi-tier systems, some works treat the Web system as a whole using black-box based methods, making it hard to decide which tiers to increase resources confronted with bottlenecks. Some other works provision for each tier independently using methods designed for a single-tier directly. This kind of divide-and-conquer based method is helpful to find bottlenecks and eliminate them but cannot avoid their shifting among tiers.

Only a few methods have been designed to deal with the bottleneck shifting of multi-tier systems. The ratio of the request arrival rate in each tier to the total arrival rate to the whole system is measured on-line. Then, resources are increased in proportion to the ratios when there are bottlenecks [3]. For single-function Web systems, request ratios are relatively stable. However, in a Meshed Web system, there are requests accessing different services and the proportions of users generating these requests may change as time goes [4], which decrease the performance of this proportion-based method.

In this paper, a Jackson network based Proactive and Reactive VM auto-scaling Method (JPRM) is proposed which uses the Markov stochastic process to describe the request transition behaviors among tiers in Meshed Web systems. The main contributions are as follows:

- A method based on stable state checking of queuing models is proposed to estimate the additionally passed request rate of each bottleneck tier after the bottleneck is eliminated.
- Queuing network and Jackson equilibrium equation based reactive method is proposed to predict the final increased request arrival rates and VM requirements of all tiers after eliminating bottleneck tiers to avoid bottleneck shifting.

The structure of the rest in this paper is as follows. Section II reviews related works. The problem is described in Section III. Section IV explains the JPRM. Section V shows experimental results and conclusions are depicted in Section VI.

II. Related Work

Resource provisioning techniques for a single-tier of Web systems can be mainly categorized into time series analysis [5], [6], queuing model [7], [8], [9], [10], [11], control theory [12] and reinforcement learning [13]. Different time series analysis models are usually used to predict workloads of Web applications, based on which reasonable resources are determined by various queuing systems such as M/M/N [7], [8], [9], M/G/N/N+K [10], MMPP/PH/1 [11] and M/GI/1 [12]. A Proportional Integral (PI) controller [12] and other control techniques are then applied to control QoS precisely. Reinforcement learning models performance characteristics more flexibly, but needs more training time [13].

Based on these techniques, resource provisioning methods for multi-tier Web systems include provisioning for each tier independently [4], [14], [15], [16], queuing networks [17], [18], [19], embedded Markov chain [20] and reinforcement

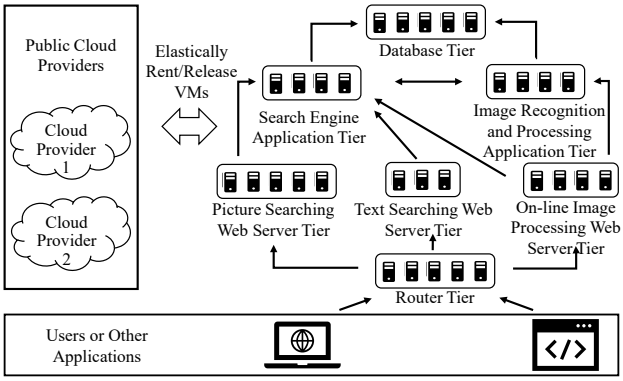


Figure 1: An example of Meshed Web systems providing multiple services.

learning [21]. However, most of them focus on predicting workloads and resource requirements of different tiers as accurately as possible, but predicted results still deviate from the actual requirements inevitably which is likely to generate bottlenecks. Urgaonkar et al. [3] proposed a Proportional Provisioning Method (PPM) for multi-tier systems. To eliminate bottlenecks, VMs are added in proportion to the ratio of the request arrival rate in each tier to the total arrival rate. However, request ratios change when the proportions of user types change which decreases the performance of PPM.

In this paper, JPRM models the request transmitting relations among different tiers using a transition probability matrix. When bottleneck tiers are eliminated, the impacts on other tiers are also predicted based on the transition probability matrix which is helpful to avoid bottleneck shifting.

III. Problem Description

A Meshed Web system usually provides diverse services (functions) such as video processing, machine learning and so on in one platform with a network structure. Fig. 1 is an example of such systems which provide multiple types of services. All requests from users or other applications first arrive at the Router tier and then are distributed to other tiers. Requests in a tier may invoke other tiers with different probabilities. Virtual cluster of each tier consists of multiple elastically rented homogeneous VMs with hour-based pricing models. The Service Level Agreement (SLA) of Web systems usually specifies a mean response time limitation W_{sla} (called sub-SLA). The objective of this paper is to design a resource auto-scaling method for Meshed Web applications to minimize VM rental costs and guarantee W_{sla}.

IV. Proposed Resource Provisioning Method

As shown in Fig. 2, the proposed JPRM mainly consists of four parts: (1) The Jackson queuing network is used to describe the Meshed Web system and a request transition probability matrix is measured on-line. (2) A Jackson network based Proactive VM Scaling-Up method (JPSU) is applied to provision VMs based on obtained transition probability matrix and M/M/N queuing models for every time interval T^p. (3) When a bottleneck is detected for every time interval T^r, the Jackson network based Reactive Bottleneck Elimination method (JRBE) is employed to obtain final estimated arrival

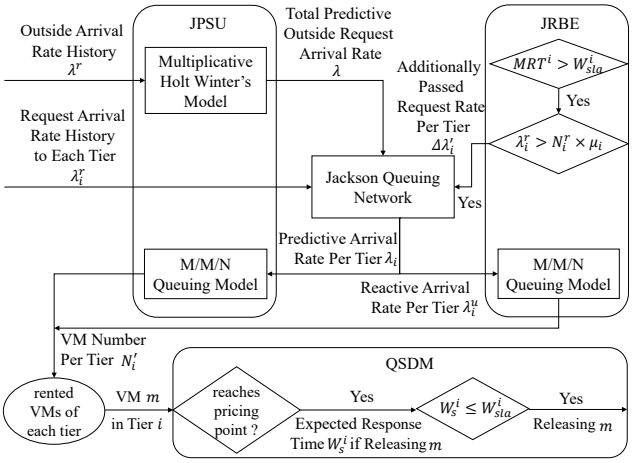


Figure 2: The overall framework of JPRM.

rates of all tiers after eliminating bottleneck tiers based on stable state checking and the transition probability matrix. (4) Whenever there is a VM reaching its pricing point, a Queuing model based VM Scaling-Down Method (QSDM) is used to decide whether to release the VM.

A. Jackson Queuing Network based Modeling

Although inter-arrival time and processing time of requests may have different distributions, it is acceptable to assume that both of them follow the exponential distribution [8], [18]. Since each tier consists of homogeneous VMs, the M/M/N queuing model is applied to model each tier. All K tiers have different probabilities to generate new requests to other tiers which can be described by a transition matrix

$$P = \begin{bmatrix} p_{0,0} & ... & p_{0,j} \\ ... & ... & ... \\ p_{i,0} & ... & p_{i,j} \end{bmatrix}$$

where $i \in \{0, 1, ..., K\}$ and $j \in \{0, 1, ..., K\}$. P can be measured on-line by the request number of each tier, which is studied from the request history.

B. Jackson Network based Proactive VM Scaling-Up Method

Predicting workloads is helpful to prepare VMs for workload changes in advance. Multiplicative Holt-Winter's model [6], which obtains well performance, is applied. In order to decrease the time spent on training data, the total outside arrival rate λ is predicted based on the real total workloads λ^r of each interval T^p. According to the Jackson queuing network theory [22], the estimated arrival rate λ_i of each tier can be obtained based on the transition probabilities as follows.

$$\lambda_j = \sum_{i=0}^{K} \lambda_i \times p_{ij}, j \in \{1, 2, ..K\} \quad (1)$$

$$\lambda_0 = \lambda \quad (2)$$

Then, based on λ_i and sub-SLA W^i_{sla}, the M/M/N queuing model is used to predict the required VM number of each tier. Given λ_i, N VMs and service rate per VM μ_i for Tier i, its expectation of response time $W^i_s(N, \lambda_i, \mu_i)$ can be obtained based on queuing theory [8]. However, the function from W^i_s to N cannot be deduced directly, an exhausted search method

is applied to find the minimum number of VMs $N_i^{'}$ to fulfill W_{sla}^i as follows.

$$N_i^{'} = \min_{N \in Z^+} \{N | W_s^i(N, \lambda_i, \mu_i) \leq W_{sla}^i\} \quad (3)$$

The formal description of JPSU is shown in Algorithm 1. Firstly, the number of VMs in each tier should not be smaller than $\lfloor \lambda_i/\mu_i \rfloor + 1$ according to the stable requirements of M/M/N queuing models. Then, an exhausted search is applied to determine Φ, based on which new VMs are rented.

Algorithm 1: Jackson network based Proactive VM Scaling-Up method (JPSU)

Input: sub-SLA W_{sla}^i and service rate μ_i of each tier
begin
  Initialize renting plan $\Phi \leftarrow \phi$;
  Predict λ based on Holt-Winter's model;
  Calculate λ_i of each tier by solving (1) and (2);
  for *each tier* $i \in \{1, 2, ..., K\}$ **do**
    Initialize $N \leftarrow \lfloor \lambda_i/\mu_i \rfloor + 1$;
    Calculate $W_s^i(N, \lambda_i, \mu_i)$;
    while $W_s^i > W_{sla}^i$ **do**
      $N \leftarrow N + 1$;
      Calculate $W_s^i(N, \lambda_i, \mu_i)$;
    $N_i^{'} \leftarrow N$ and $\Phi \leftarrow \Phi \bigcup N_i^{'}$;
  Scaling up VMs of each tier according to Φ;

C. Jackson Network based Reactive Bottleneck Elimination

Due to inevitable inaccuracy of Holt-Winter's prediction model, workload fluctuations and the proportion changes of requests accessing different services, existing resources may be not sufficient to cope with the real workloads. It is crucial to detect bottlenecks, eliminate them and avoid their shifting among tiers in every interval T^r.

For any tier i, it is considered a bottleneck tier when the real response time MRT_i is larger than assigned sub-SLA W_{sla}^i and the queuing model can be used to analyze its additionally passed request rate. Tier i is still in a stable state if the real arrival rate λ_i^r is still smaller than its current processing rate $N_i^r \times \mu_i$. Increasing additional VMs to this tier will not increase the arrival rates of other tiers. Otherwise, it is in a nonstable state, its queue length will increase continually if no VMs are added and the passed request rate is smaller than its arrival rate. After additional VMs are added, the additional request rate passing through this tier which also generates additional workloads to other tiers can be obtained by

$$\triangle\lambda_i^{'} = \begin{cases} \lambda_i^r - (N_i^r \times \mu_i) & \lambda_i^r > N_i^r \times \mu_i \\ 0 & Otherwise \end{cases} \quad (4)$$

However, complex request transition relationships among tiers make it complicated to calculate the impacts of eliminating all bottleneck tiers simultaneously on other tiers. The Jackson queuing network [22] can solve this problem well by the Jackson equilibrium equation as follows.

$$\triangle\lambda_j = \triangle\lambda_j^{'} + \sum_{i=1}^{K} \triangle\lambda_i \times p_{ij}, j \in \{1, 2, ..K\} \quad (5)$$

By solving this linear system, the final increased arrival rate $\triangle\lambda_i$ of each tier is obtained by the additional arrival rate of the current tier and additional transferring arrival rates from other tiers. $\triangle\lambda_i$ will be added to the original real arrival rate λ_i^r to obtain the final reactive arrival rate λ_i^u after eliminating bottlenecks. Specially, the original arrival rate of a bottleneck tier is $N_i^r \times \mu_i$ when $\lambda_i^r > N_i^r \times \mu_i$. Based on obtained λ_i^u, the M/M/N queuing model is used to calculate the required VM number of each tier $N_i^{'}$ by (3) as shown in Algorithm 2.

Algorithm 2: Jackson network based Reactive Bottleneck Elimination method (JRBE)

Input: assigned sub-SLA W_{sla}^i for each tier;
mean response time MRT_i of each tier;
begin
  Initialize renting plan $\Phi \leftarrow \phi$;
  set of bottleneck tiers $Bt \leftarrow \phi$;
  for *each tier* $i \in \{1, 2, ..., K\}$ **do**
    if $MRT_i > W_{sla}^i$ **then**
      $Bt \leftarrow Bt \bigcup i$;
  for *each tier* $i \in Bt$ **do**
    Calculate $\triangle\lambda_i^{'}$ based on (4);
  Compute $\triangle\lambda_i, i \in \{1, 2, ..., K\}$ by solving (5);
  for *each tier* $i \in \{1, 2, ..., K\}$ **do**
    if $\lambda_i^r > N_i^r \times \mu_i$ **then**
      $\lambda_i^u \leftarrow N_i^r \times \mu_i + \triangle\lambda_i$;
    else
      $\lambda_i^u \leftarrow \lambda_i^r + \triangle\lambda_i$;
    Calculate $N_i^{'}$ based on λ_i^u using (3);
    $\Phi \leftarrow \Phi \bigcup N_i^{'}$;
  Scaling up VMs of each tier according to Φ;

D. Queuing Model based VM Scaling-Down Method

To save rental costs, VMs are only released when rented hours for them are used up. Whenever a VM m of Tier i reaches its pricing point, it is checked that whether the left capacity after releasing this VM can fulfill the resource requirement of this tier. Let N_i^r be the existing real VM number of Tier i then $N = N_i^r - 1$ be the number of left VMs. If $W_s^i(N, \max(\lambda_i, \lambda_i^u, \lambda_i^r), \mu_i) \leq W_{sla}^i$, VM m is released. Otherwise, the next hour is still rented on this VM.

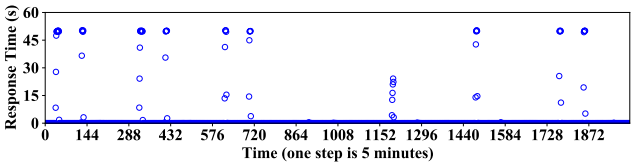
V. Performance Evaluation

Since PPM [3] is one of a few methods considering bottleneck eliminating in multi-tier Web systems, JPRM is compared with PPM first which is modified by using the M/M/N queuing models for fair comparison. JPRM is also compared with JPSU which uses the Jackson queuing network as a proactive resource provisioning technique [18]. They are evaluated on CloudSim [23], in which on-demand VM type "c4.2xL" of Amazon EC2 with the price of $0.419 per hour is simulated.

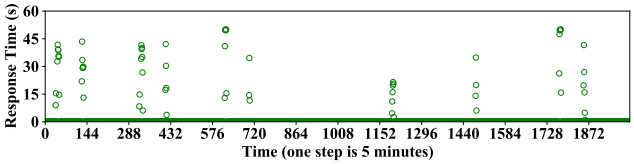
The realistic Wikipedia traces [24] are applied. A Meshed Web system with eight tiers is used as the test-bed. Because the original Wikipedia trace data has no information about

TABLE I: PARAMETERS OF THE MESHED WEB SYSTEM

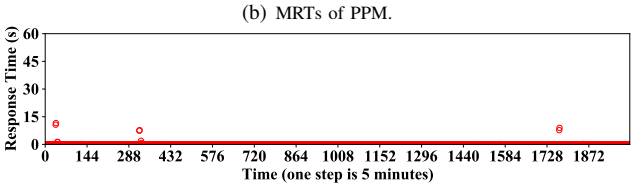
Parameters	Tiers 1 to 5	Tiers 6 to 8
Service rate μ_i (requests/s)	38.75	3.875
Sub-SLA W^i_{sla} (second)	0.03	0.275
Mean task length (MIPS)	400	4000



(a) MRTs of JPSU.



(b) MRTs of PPM.

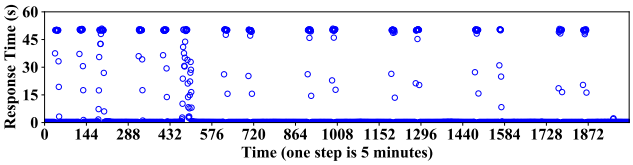


(c) MRTs of JPRM.

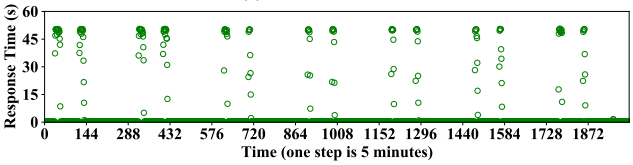
Figure 3: MRTs of requests in Tier 3.

request transition among different tiers, hierarchical folders in the Uniform Resource Locator (URL) of requests are mapped to eight tiers to simulate the request transition among tiers. Each original request in the Wikipedia trace data is considered as a session, which will generate multiple requests to different tiers in the order of hierarchical folders of its URL. Tier 0 is the Router tier. Tier 1 to Tier 5 are parallel Web Server tiers for five different services. Tier 6 and Tier 7 are two common Application tiers and Tier 8 is the Database tier. The service rate per VM, assigned sub-SLA and average task length of requests in each tier are shown in Table I. And the proactive and reactive control intervals are set to be $T^p = 1$ hour and $T^r = 5$ minutes, respectively. To simulate proportion changes of request accessing different services, user sessions to Tier 3 are increased to $2 \times \lambda_3^r$ and $3 \times \lambda_3^r$ in two sequential thirty-minute intervals then decreased in opposite order in another two sequential thirty-minute intervals.

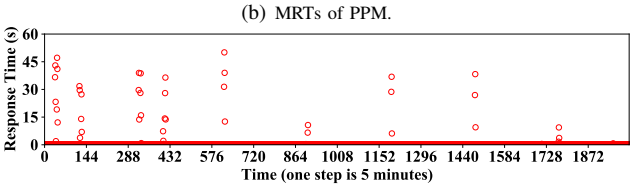
Experimental results show that workloads can be predicted by Holt-Winter's model accurately with a very small mean prediction error of 2.4%, but there are still large deviations leading to bottlenecks in some cases. As shown in Fig. 3(a) and Fig. 4(a), because there are no reactive strategies in JPSU, requests will block all related tiers one by one, e.g., SLA violations of Tier 3 and Tier 7 increase to about 50 seconds and last for a long time until workloads decrease. When bottlenecks occur in the Web system, PPM increases VMs for each tier in proportion to the ratio of the number of requests arriving at each tier to the number of sessions arriving at the whole system. These ratios computed on-line will change when the request proportions change. Therefore,



(a) MRTs of JPSU.

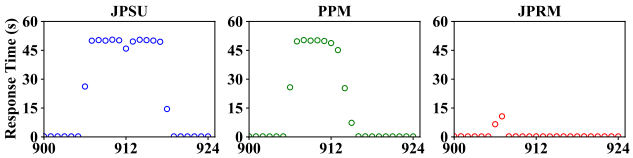


(b) MRTs of PPM.

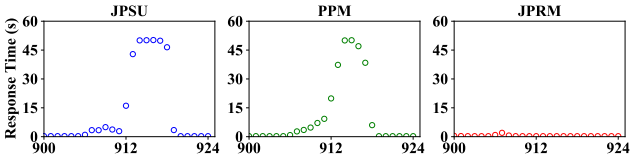


(c) MRTs of JPRM.

Figure 4: MRTs of requests in Tier 7.



(a) MRTs of Tier 7 (one step is 5 minutes).



(b) MRTs of Tier 8 (one step is 5 minutes).

Figure 5: A comparison of request MRTs in the 7th workload fluctuation.

there is great deviation between real ratio and historical ratio of each tier after the proportion of requests belonging to Tier 3 increases greatly. Tier 3 and Tier 7 are both under provisioned by PPM, which makes their MRTs smaller than those of JPSU but still larger than their corresponding sub-SLA as shown in Fig. 3(b) and Fig. 4(b). In contrast, JPRM uses the transition probability matrix which can predict the impacts of bottleneck tiers on other tiers accurately to determine where and how many VMs are added to eliminate bottlenecks and avoid bottleneck shifting. Therefore, JPRM's MRTs in Tier 3 and Tier 7 drop greatly when additional VMs are rented as shown in Fig. 3(c) and Fig. 4(c). Then, requests from Tier 7 generate new requests to Tier 8, causing that Tier 8 will be a bottleneck tier when there is a bottleneck in Tier 7. Because of the request transmission delays, the SLA violation peak of Tier 8 appears several minutes later than that of Tier 7 while using JPSU and PPM as MRTs within a short time shown in Fig. 5. On the contrary, there is no or only a minor SLA violation peak for JPRM, which shows that JPRM has the ability to avoid bottleneck shifting among tiers.

Table II describes the VM rental costs of different tiers which shows that JPSU is the cheapest because it never reacts to bottlenecks. PPM's proportional provisioning strategy

TABLE II: VM RENTAL COSTS OF DIFFERENT TIERS

	JPSU	PPM	JPRM
Tier 1	196.092	198.187	196.511
Tier 2	188.969	192.740	191.064
Tier 3	405.173	406.849	414.810
Tier 4	219.975	225.841	222.489
Tier 5	141.622	143.298	141.622
Tier 6	901.688	928.085	1507.562
Tier 7	3484.823	3510.382	3909.270
Tier 8	3421.554	3464.292	3920.583

makes Tier 3, Tier 7 and Tier 8 under-provisioned, but Tier 1, Tier 2, Tier 4 and Tier 5 over-provisioned. Therefore, PPM's costs of Tier 3, Tier 7 and Tier 8 are lower than those of JPRM while costs of Tier 1, Tier 2, Tier 4 and Tier 5 are higher than those of JPRM. Moreover, JPRM's cost of Tier 6 is also larger than other two algorithms' costs because this tier becomes a bottleneck caused by original workload fluctuations in some cases. JPRM has the highest total rental cost because it needs an appropriate amount of additional resources to eliminate bottlenecks quickly.

VI. Conclusions and Future Work

In this paper, the Jackson queuing network is applied to model the Meshed Web system, based on which a proactive and reactive resource auto-scaling and bottleneck eliminating method is proposed. Experimental results show that the M/M/N queuing model and Jackson equilibrium equation based method can estimate the impacts of eliminating bottleneck tiers on other tiers accurately. Bottlenecks are not shifted to other tiers after existing bottleneck tiers are eliminated. VMs are reasonably rented for different tiers incurring appropriate additional rental costs to eliminate bottleneck tiers. Combining control theory to improve the controllability of the balance between costs and performance of provisioning algorithms for Meshed Web systems is promising future work.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No.61972202 and 61602243), the Fundamental Research Funds for the Central Universities (No.30919011235 and NO.30920120180101).

References

[1] D. Vohra, *Using the Amazon EC2. Pro Docker*. Apress, Berkeley, 2016.

[2] G. Huang, C. Luo, K. Wu, Y. Ma, Y. Zhang, and X. Liu, "Software-defined infrastructure for decentralized data lifecycle governance: Principled design and open challenges," in *39th IEEE International Conference on Distributed Computing Systems (ICDCS)*. Dallas, TX, USA, USA: IEEE, 7-10 July 2019, pp. 1674–1683.

[3] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood, "Agile dynamic provisioning of multi-tier internet applications," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 3, no. 1, pp. 1–39, 2008.

[4] R. Singh, U. Sharma, E. Cecchet, and P. Shenoy, "Autonomic mix-aware provisioning for non-stationary data center workloads," in *7th International Conference on Autonomic Computing (ICAC)*, Reston, VA, USA, 7-11 June 2010, pp. 21–30.

[5] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *4th IEEE International Conference on Cloud Computing (CLOUD)*. Washington, DC, USA: IEEE, 4-9 July 2011, pp. 500–507.

[6] M. Balaji, G. S. V. R. K. Rao, and C. A. Kumar, "A comparitive study of predictive models for cloud infrastructure management," in *14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, Chicago, IL, USA, 26-29 May 2014, pp. 923–926.

[7] S. Pal and P. K. Pattnaik, "A simulation-based approach to optimize the execution time and minimization of average waiting time using queuing model in cloud computing environment," *International Journal of Electrical & Computer Engineering*, vol. 6, no. 2, pp. 743–750, 2016.

[8] J. Jiang, J. Lu, G. Zhang, and G. Long, "Optimal cloud resource auto-scaling for web applications," in *3rd IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid)*. Delft, Netherlands: IEEE, 13-16 May 2013, pp. 58–65.

[9] A. Melikov, A. Rustamov, and J. Sztrik, "Queuing management with feedback in cloud computing centers with large numbers of web servers," in *5th International Conference on Distributed Computer and Communication Networks (DCCN)*. Moscow, Russia: Springer, 17-21 Sept. 2018, pp. 106–119.

[10] X. Chang, B. Wang, J. K. Muppala, and J. Liu, "Modeling active virtual machines on iaas clouds using an m/g/m/m+k queue," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 408–420, 2014.

[11] A. Rajabi and J. W. Wong, "Provisioning of computing resources for web applications under time-varying traffic," in *22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*. Paris, France: IEEE, 9-11 Sept. 2014, pp. 152–157.

[12] A. Kamra, V. Misra, and E. M. Nahum, "Yaksha: A self-tuning controller for managing the performance of 3-tiered web sites," in *12th IEEE International Workshop on Quality of Service (IWQOS)*. Montreal, Que., Canada: IEEE, 9 June 2004, pp. 47–56.

[13] L. Yazdanov and C. Fetzer, "Lightweight automatic resource scaling for multi-tier web applications," in *7th IEEE International Conference on Cloud Computing (CLOUD)*. Anchorage, AK, USA: IEEE, 27 June-2 July 2014, pp. 466–473.

[14] A. Ashraf, B. Byholm, J. Lehtinen, and I. Porres, "Feedback control algorithms to deploy and scale multiple web applications per virtual machine," in *38th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. Cesme, Izmir, Turkey: IEEE, 5-8 Sept. 2012, pp. 431–438.

[15] J. Bi, Z. Zhu, R. Tian, and Q. Wang, "Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center," in *3rd International Conference on Cloud Computing (CLOUD)*. Miami, FL, USA: IEEE, 5-10 July 2010, pp. 370–377.

[16] J. Bi, H. Yuan, M. Tie, and W. Tan, "Sla-based optimisation of virtualised resource for multi-tier web applications in cloud data centres," *Enterprise Information Systems*, vol. 9, no. 7, pp. 743–767, 2015.

[17] F. Seracini, M. Menarini, I. Krueger, L. Baresi, S. Guinea, and G. Quattrocchi, "A comprehensive resource management solution for web-based systems," in *11th International Conference on Autonomic Computing (ICAC)*, Philadelphia, PA, 18-20 June 2014, pp. 233–239.

[18] Y. Shi, J. Huang, X. Zhao, L. Liu, S. Liu, and L. Cui, "Integrating theoretical modeling and experimental measurement for soft resource allocation in multi-tier web systems," in *23rd IEEE International Conference on Web Services (ICWS)*. San Francisco, California, USA: IEEE, 27 June-2 July 2016, pp. 522–529.

[19] A. Gandhi, P. Dube, A. Karve, A. Kochut, and L. Zhang, "Adaptive, model-driven autoscaling for cloud applications," in *11th International Conference on Autonomic Computing (ICAC)*, Philadelphia, PA, 18-20 June 2014, pp. 57–64.

[20] K. Salah, K. Elbadawi, and R. Boutaba, "An analytical model for estimating cloud resources of elastic services," *Journal of Networks and Systems Management*, vol. 24, no. 2, pp. 285–308, 2016.

[21] W. Iqbal, M. N. Dailey, and D. Carrera, "Unsupervised learning of dynamic resource provisioning policies for cloud-hosted multitier web applications," *IEEE Systems Journal*, vol. 10, no. 4, pp. 1–12, 2015.

[22] J. R. Jackson, "Networks of waiting lines," *Operations research*, vol. 5, no. 4, pp. 518–521, 1957.

[23] T. Goyal, A. Singh, and A. Agrawal, "Cloudsim: Simulator for cloud computing infrastructure and modeling," *Procedia Engineering*, vol. 38, no. 4, pp. 3566–3572, 2012.

[24] G. Urdaneta, G. Pierre, and M. van Steen, "Wikipedia workload analysis for decentralized hosting," *Elsevier Computer Networks*, vol. 53, no. 11, pp. 1830–1845, 2009.