A VM Reservation-Based Cloud Service Broker and Its Performance Evaluation

Heejae Kim, Yoonki Ha, Yusik Kim,
Kyung-no Joo, and Chan-Hyun Youn(✉)

Department of Electrical Engineering, Korea Advanced Institute
of Science and Technology (KAIST), Daejeon, Korea
{kim88l0l9, milmgas, yusiky, eu2l98, chyoun}@kaist.ac.kr

Abstract. We deal with a reservation-based cloud service broker (R-CSB). The main role of the R-CSB is to provide application execution services or Software-as-a-Service. The R-CSB makes a profit by an arbitrage between cloud service consumers and providers, and service fees from the consumers. In this paper, we first present detail concepts and architecture of the R-CSB. Also, to reduce the VM leasing cost, we discuss two schemes. The VM reservation scheme (C-VMR) makes the R-CSB reduce the VM leasing cost via leasing an appropriate number of reserved VMs. In addition to the C-VMR, we also present the VM allocation scheme (C-VMA) to allocate applications to VMs cost-effectively. Performance evaluation results show that the C-VMR has lower cost than other approaches and the C-VMA shows has higher average VM utilization than the conventional methods in most cases.

Keywords: Cloud service brokers · VM reservation · VM allocation

1 Introduction

According to Gartner, cloud service brokers (CSBs) are one of the top 10 strategic technology trends for 2014 [1]. There are many related companies to provide services for selecting best services of multiple clouds, adding monitoring services, metadata managing services, and providing Software-as-a-Service (SaaS). Liu et al. [2] classified these services as three forms: service intermediation to improve services by adding new value-added features, service aggregation to combine and integrate services into new services, and service arbitrage to arbitrage and aggregate service with not fixed services.

We suppose that a CSB operates independently to cloud service providers (CSPs) and cloud service consumers (CSCs). It means that the CSB is a business entity which creates values between CSPs and CSCs, and we call it as a VM reservation-based CSB (R-CSB). The main role of the R-CSB is to provide application execution services or SaaS using virtual machines (VMs) leased from CSPs. Because it is difficult for most CSCs to perform effective VM allocation and application execution management, the R-CSB has advantages of choosing the optimal resources to execute the applications with characteristics of CSCs (e.g. geographical location, network topology, various types of resource requests, etc.), and reallocating VMs in dynamic situations via

V.C.M. Leung et al. (Eds.): CloudComp 2014, LNICST 142, pp. 43–52, 2015.
DOI: 10.1007/978-3-319-16050-4_4

monitoring performance. Also, because we consider the actual and general pricing policies of CSPs, the R-CSB is easily applicable in today's industry.

In this paper, we present detail concepts and architecture of the R-CSB, and its VM reservation and allocation schemes. The VM reservation scheme (C-VMR) and the VM allocation scheme (C-VMA) especially focus on reducing the VM leasing cost. The C-VMR is designed to lease an appropriate number of reserved VMs (RVMs) by demand with time. Also, as an extension of BestFit, the C-VMA is operated with both on-demand VMs (OVMs) and RVMs by considering residual times of VMs.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 presents detail concepts and architecture of the R-CSB. Sections 4 and 5 present the C-VMR and C-VMA respectively. In Sect. 6, we evaluate the C-VMR and C-VMA. Finally, Sect. 7 concludes this paper.

2 Related Work

For VM reservation in clouds, Chaisiri et al. [3] proposed the OVMP algorithm to optimize resource provisioning and VM placement. The authors formulated the optimization problem to minimize the total cost of resource provisioning using stochastic integer programming and presented a method to solve it using Benders decomposition and sample-average approximation algorithms. Wang et al. [4] presented VM reservation strategies to minimize the VM leasing cost using dynamic programming and the corresponding approximation algorithms for CSBs.

For VM allocation in clouds, Genaud et al. [5] divided strategies for it into four categories: 1VM4ALL, 1VMPerJob, Bin-Packing, and Relax. 1VM4ALL allocates every job to a single VM, 1VMPerJob allocates each job to a VM, Bin-Packing allocates jobs to VMs using heuristics such as FirstFit, BestFit, and WorstFit, and Relax considers SLAs by including a bound on the waiting time. Leitner et al. [6] presented a scheme to minimize the sum of changes of the VM leasing cost and the SLA penalty cost. They described the change of the VM leasing cost after the VM κ is selected to allocate the application as depicted in Eq. (1) where T^{req} is the execution time of the application requested to execute, T^{res}_{κ} is the residual time of the VM κ, BTU_{κ} is billing time unit (BTU) of the VM κ, and p^{BTU}_{κ} is the price of the VM κ in the BTU.

$$\Delta lc_{\kappa} = \begin{cases} \left| \frac{T^{req} - T^{res}_{\kappa}}{BTU_{\kappa}} \cdot p^{BTU}_{\kappa} \right|, & if\ T^{req} > T^{res}_{\kappa} \\ 0, & if\ T^{req} \leq T^{res}_{\kappa} \end{cases}. \tag{1}$$

We note that selecting a VM to minimize the change of the VM leasing cost is the extension of BestFit which selects a VM whose residual time is longer than the execution time and the nearest to it. We call it as modified BestFit (MBF) in the remainder of this paper. In addition, WorstFit is identical to the BestFit except that it selects a VM whose residual time is the farthest to the execution time. We call the corresponding extension of the WorstFit as modified WorstFit (MWF). Shen et al. [7] presented a scheme using a portfolio of integer programming problems (IPP) and heuristics-based

approaches. In the scheme, VM allocation strategies are produced by the IPP and the various heuristics in limited time, and the best strategy is selected as its VM allocation decision. They also extended the scheme to consider both OVMs and RVMs by determining the number and types of RVMs. A scheme presented by Deng et al. [8] used a trace-based simulator to select a suitable strategy for each VM provisioning, job selecting, and VM selecting in a portfolio. In addition, they propose an algorithm to enlarge the chance of selecting the best policy in limited time.

3 Reservation-Based Cloud Service Broker

The R-CSB executes applications on behalf of CSCs or provides SaaS using VMs leased from CSPs. A profit of the R-CSB is made by an arbitrage between CSCs and CSPs, and service fees from CSCs. To increase the profit, the VM leasing cost of the R-CSB should decrease, and we solve it via cost-effective VM reservation and allocation. The VM reservation is based on the following fact. The resources provided by CSPs is generally divided by OVMs and RVMs. The OVMs and the RVMs refers to VMs leased in comparatively short BTUs (e.g. an hour) and long BTUs (e.g. a month, a year) respectively. Prices of RVMs during unit time is set to be cheaper than those of OVMs, and the VM reservation can reduce the VM leasing cost. However, because BTUs of RVMs are much longer than those of OVMs, the cost-effectiveness of the VM reservation can rather decrease if utilizations of the RVMs are low. Therefore, the R-CSB should lease an appropriate number of RVMs.

In addition to the VM reservation, the VM leasing cost can be reduced by the effective VM allocation via increasing average VM utilization. If the number of leased RVMs is greater or equal to the current demand to the R-CSB, it is enough to allocate applications to the RVMs, and the OVM leasing cost is not imposed. Otherwise, an additional OVM should be leased to allocate the application. Therefore, increasing average VM utilization decrease the number of OVMs leased additionally, and it results in the reduction of the VM leasing cost.

Figure 1 depicts architecture of the R-CSB. A VM reservation module is to determine the number of RVMs to be leased by time. The VM reservation strategizing in the VM reservation module is performed based on demand monitoring and prediction. RVMs leased by the VM reservation module and OVMs additionally leased are managed in a VM pool management module. We divide the VM pool into two kinds: VM pools which contains VMs whose status is idle (an idle VM pool) and VMs on which the applications is executed (an active VM pool). For application execution requests of CSCs via a user interface, the R-CSB parses the requests and profiles the applications if the profiling isn't done before. The applications are scheduled and allocated to appropriate VMs in the idle VM pool, and VM scaling is performed if it is empty. Then, the application execution module starts to execute the applications via a cloud interface. To guarantee performances of applications in the dynamic nature of public clouds, the R-CSB is designed to reallocate VMs if performance degradations are monitored or current VM allocation cannot satisfy SLAs of CSCs. In addition, we need a security and authentication management module for a secure service of R-CSB.

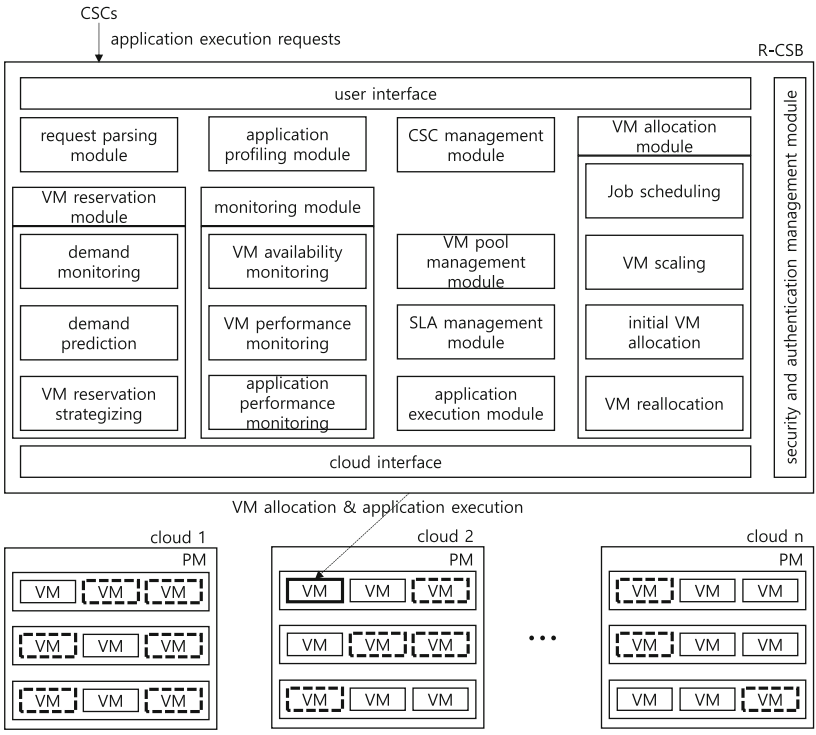


Fig. 1. Architecture of the R-CSB.

4 C-VMR

The C-VMR is presented to determine the appropriate number of RVMs to be leased in the VM reservation module. In this section, we suppose that BTUs of OVMs and RVMs are fixed as BTU_{OVM} and BTU_{RVM} respectively. The basic idea of the C-VMR is adaptively determining the number of RVMs to be leased based on predicted demand. The demand refers to the number of VMs needed to service application execution requests of CSCs.

The C-VMR uses the auto regressive integrated moving average (ARIMA) model to predict demand based on Brockwell et al. [9], Fang et al. [10], and Ha et al. [11]. The demand prediction is performed with the following three steps. First, time series of the demand are preprocessed to apply the ARIMA model. Because the ARIMA model is only applicable with stationary time series, the non-stationary time series of the demand should be processed to be stationary. It can be achieved by obtaining derivations of the time series. Second, the order of the ARIMA model is determined based on the auto correlation function (ACF) and the partial auto correlation function (PACF) of the preprocessed time series. Validity of the ARIMA model is also checked in this step. Third, the ARIMA model is applied to predict the demand.

A mechanism of the C-VMR is as follows. The C-VMR is operated every period T_r. At each time t at which the C-VMR is operated, the demand from the time t for T_p is predicted. Then, $n^l_{RVM}(t)$ which denotes the number of RVMs to be leased at the time t is determined as Eq. (1) where $D_p(t)$ is the predicted demand during $[t, t+1]$, and $n^e_{RVM}(t)$ is the number of RVMs in the VM pool at time t.

$$n^l_{RVM}(t) = \left\lfloor \frac{1}{T_p} \sum_{k=t}^{t+T_p} D_p(k) - n^e_{RVM}(k) \right\rfloor. \tag{2}$$

Algorithm 1. C-VMR

Input: historical demand information during $[t-T_h, t-1]$ where T_h is a period to be used as the historical demand in the demand prediction

1: predict the demand during $[t, t+T_p]$

2: Obtain $n^l_{RVM}(t)$

3: Lease additional RVMs as much as $n^l_{RVM}(t)$

5 C-VMA

The C-VMA is operated as online VM allocation. We assume that the applications are indivisible. To reduce the VM leasing cost, the C-VMA focuses on increasing VM utilization. Whenever each application execution request is arrived, the C-VMA is operated as follows. If there exist OVMs which satisfy $\alpha <$ the residual time – the predicted application execution time $< \beta$ in the idle VM pool, the MBF is applied for the OVMs. Otherwise, if there exist RVMs in the idle VM pool, one of the RVMs is selected. Finally, if there does not exist the corresponding OVM or RVM, the MBF is applied for all the VMs in the idle VM pool. Obviously, an additional OVM is leased and added to the idle VM pool if it is empty.

Algorithm 2. C-VMA

Input: an application execution request

1: **if** there exists OVMs which satisfy $\alpha <$ the residual time – the predicted application execution time $< \beta$ **then**

2: apply the MBF for the OVMs

3: **else if** there exists RVMs in the idle VM pool then

4: allocate the application to one of the RVMs

5: **else**

6: apply the MBF for all the OVMs in the idle VM pool

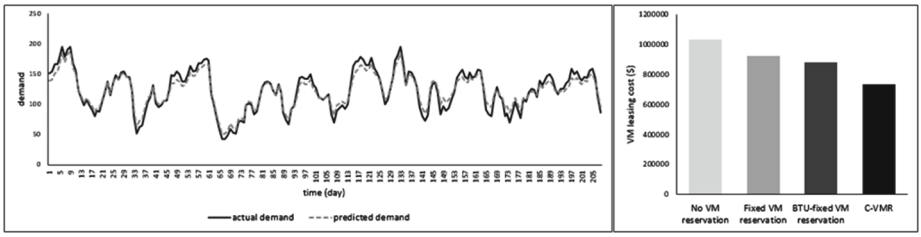
7: **end if**

6 Performance Evaluation

6.1 C-VMR

In this section, we evaluate the C-VMR. For the evaluation, the actual demand is generated for 4 years as depicted in Fig. 2(a) as a solid line. The actual demand is unit of the average number of VMs needed to service application execution requests of CSCs per hour. We use ASTSA package in R [12] for the demand prediction using the ARIMA model, and the prediction result is also depicted in Fig. 2(a) as a dotted line. In the evaluation, we suppose that there are 5 RVMs in the VM pool initially, BTUs of OVMs and RVMs are 1 h and 1 month, and the leasing costs are $0.24 per an hour and $131.4 per a month respectively. We note that the VM leasing costs are from the pricing policy for large standard cloud servers in GoGrid [13]. Also, we set T_r, T_p, and T_h as 1 week, 1 month, and 30 weeks respectively.

We compare the C-VMR with three other approaches: No VM reservation, Fixed VM reservation, and BTU-fixed VM reservation). The No VM reservation is the approach which uses no RVM. Therefore, application execution requests are serviced using only OVMs. The Fixed VM reservation is the approach which reserves an optimal fixed number of RVMs for every time. The optimal fixed number is determined to minimize the VM leasing cost. The BTU-fixed VM reservation is the approach which reserves an optimal fixed number of RVMs for each period as much as BTU. Therefore, the optimal fixed number is recomputed at each period. Figure 2(b) shows the result. The result shows that the VM leasing cost of C-VMR is less than other three approaches.



(a) (b)

Fig. 2. An example of VM allocation in the R-CSB: (a) VM allocation when it is not necessary to lease OVMs, (b) VM allocation when it is necessary to lease OVMs additionally.

6.2 C-VMA

In this section, we evaluate the C-VMA. The evaluation is performed in our cloud testbed based on OpenStack Essex [14] as depicted in Fig. 3. Each physical machine (PM) in the testbed uses two quad-core processors with Hyper-Threading [15] (Intel® Xeon® Processor E5620). It also has 14 GB for the main memory and 1000 GB for the hard disk. For the evaluation, we developed the several modules highlighted in Fig. 3, and the modules are operated on Apache Tomcat 7.0 [16]. The experimental procedure

is as follows. We start the experiment after building VMs which have 1 VCPU, 1 GB of memory, and 10 GB. Five VMs of them are set for the initial RVMs in an idle VM pool, and we suppose that BTU of the RVMs are longer than the experiment period. Application execution requests are arrived to the R-SPCSB via RESTful web services via Jersey [17] during an hour by inter-arrival times which follows a Poisson distribution whose mean is 5 s. For the applications, we use three applications whose expected execution times are 15.99, 38.23, and 60.08 s and they are based on MapChem [18] which performs a high performance bio and chemical analysis. We note that the inter-arrival time and the application for each request is predetermined randomly before the experiment. Application execution requests are written in JavaScript Object Notation (JSON) [19] via Gson [20] and transmitted to the R-CSB via a POST method. After receiving each request, the R-CSB parses the request using Gson, and a VM to execute the application is selected by a VM allocation strategy. Then, the application is executed on it. We suppose that BTU of OVMs is 100 s, and extra delay to lease new VMs including transaction time is 5 s. These are considered as scaled down values of BTU and extra delay in real world respectively.

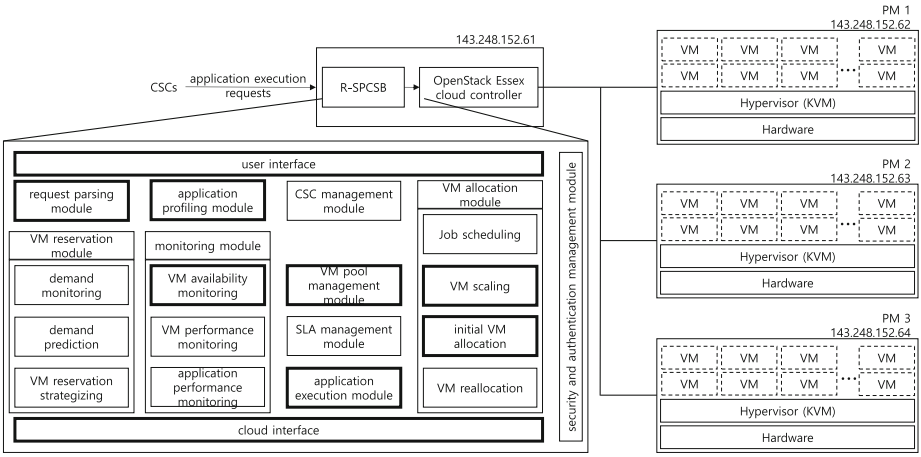


Fig. 3. The experimental environment for evaluating the C-VMA.

We evaluate the C-VMA for six cases by difference between predicted and expected application execution times.

Case 1 (predicted application execution times ≪ expected application execution times). Predicted application execution times of the three applications are 10.99, 26.23, and 40.08 s respectively. $\alpha = 7$ and $\beta = 27$.

Case 2 (predicted application execution times < expected application execution times). Predicted application execution times of the three applications are 10.99, 30.23, and 50.08 s respectively. $\alpha = 5$ and $\beta = 25$.

Case 3 (predicted application execution times = expected application execution times = actual application execution times). Predicted application execution times of the three applications are 15.99, 38.23, and 60.08 s respectively. $\alpha = 0$ and $\beta = 20$.

Case 4 (predicted application execution times = expected application execution times). Predicted application execution times of the three applications are 15.99, 38.23, and 60.08 s respectively. $\alpha = 0$ and $\beta = 20$.
Case 5 (predicted application execution times > expected application execution times). Predicted application execution times of the three applications are 20.99, 46.23, and 70.08 s respectively. $\alpha = -5$ and $\beta = 25$.
Case 6 (predicted application execution times $\gg$ expected application execution times). Predicted application execution times of the three applications are 20.99, 52.23, and 80.08 s respectively. $\alpha = -7$ and $\beta = 27$.

We note that the experiment for the case 3 was performed with just waiting (i.e., executing sleep()) during the corresponding expected application execution times instead of executing application actually.

Figure 4 depicts the results. We compare the C-VMA with the MBF, the MWF, and reservation-based MBF (R-MBF). We note that the R-MBF is identical to the MBF except that it tries to allocate applications to RVMs first. The results of the MBF shows that the average VM utilization gets lower as the predicted application execution times are farther from the expected application execution times. Because the MBF is based on the prediction of application execution times, the MBF can fail to select VMs whose residual time is the closest to the actual application execution times. In addition, the unnecessary VM leasing cost can occur if the predicted application execution times are shorter than the actual application execution times in the selected VMs. Therefore, the C-VMA is designed to overcome these problems via using the specific bound of the residual time – the predicted application execution time. On the other hand, the results

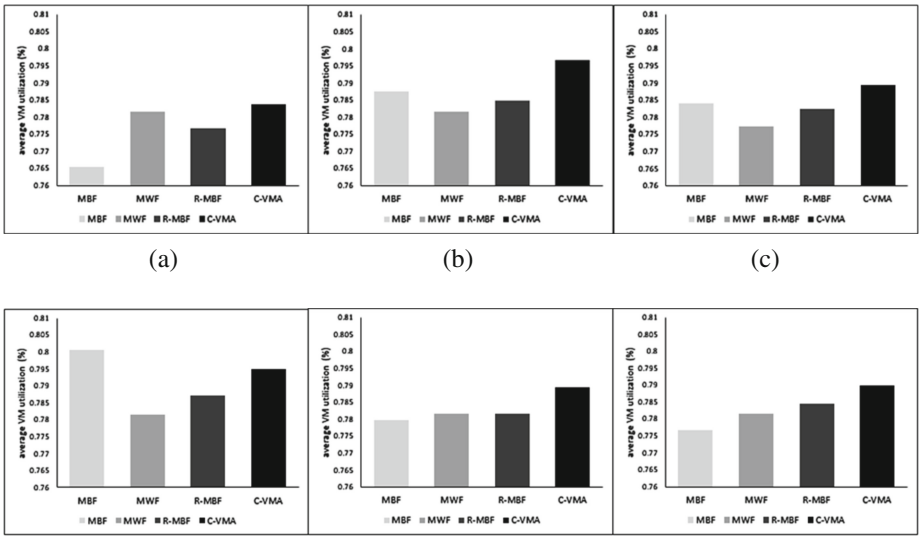


Fig. 4. The average VM utilization of the MBF, the MWF, the R-MBF, and the C-VMA: (a) case 1, (b) case 2, (c) case 3, (d) case 4, (e) case 5, and (f) case 6.

of the MWF are the same for all the cases except for the case 3 because it selects a VM whose residual time is the longest regardless of the predicted application execution times. In the results of the R-MBF, the average VM utilizations of the R-MBF are higher than those of the MWF if the predicted application execution times are close to the expected application execution times. Otherwise, the average VM utilizations are higher than that of the MBF because the R-MBF checks RVMs whether there exist RVMs available to allocate first. Finally, the average VM utilization of the C-VMA is the highest for all the cases except for the case 4.

7 Conclusion

In this paper, we presented a VM reservation-based cloud service broker and its performance evaluation. Among many issues enabling to be addressed in the R-CSB, we focused on reducing the VM leasing cost. To achieve it, the C-VMR and the C-VMA were presented for cost-effective VM reservation and allocation respectively. The evaluation for the C-VMR showed that the VM leasing cost of the C-VMR is the lowest compared with the other methods. To evaluate the C-VMA, we implemented a prototype of the R-CSB. The results showed that the average VM utilization of the C-VMA is the highest compared with the conventional methods in most cases. As on-going and future work, we are extending the C-VMR and the C-VMA to consider various types of OVMs and RVMs of multiple CSPs, and constraints such as budget, performance, etc.

Acknowledgments. This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2014, and the MSIP under the ITRC (Information Technology Research Center) support program (NIPA-2014(H0301-14-1020)) supervised by the NIPA (National IT Industry Promotion Agency).

References

1. Top 10 Strategic Technology Trends for 2014. http://www.gartner.com
2. Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., Leaf, D.: NIST cloud computing reference architecture. NIST Special Publication 500-292 (2011)
3. Chaisiri, S., Lee, B.S., Niyato, D.: Optimization of resource provisioning cost in cloud computing. IEEE Trans. Serv. Comput. **5**(2), 164–177 (2012)
4. Wang, W., Niu, D., Li, B., Liang, B.: Dynamic cloud resource reservation via cloud brokerage. In: 33rd IEEE International Conference on Distributed Computing Systems, pp. 400–409 (2013)
5. Genaud, S., Gossa, J.: Cost-wait trade-offs in client-side resource provisioning with elastic clouds. In: 4th IEEE International Conference on Cloud Computing, pp. 1–8 (2011)
6. Leitner, P., Hummer, W., Satzger, B., Inzinger, C., Dustdar, S.: Cost-efficient and application SLA-aware client side request scheduling in an infrastructure-as-a-service cloud. In: 5th IEEE International Conference on Cloud Computing, pp. 213–220 (2012)
7. Shen, S., Deng, K., Iosup, A., Epema, D.: Scheduling jobs in the cloud using on-demand and reserved instances. In: Wolf, F., Mohr, B., an Mey, D. (eds.) Euro-Par 2013. LNCS, vol. 8097, pp. 242–254. Springer, Heidelberg (2013)

8. Deng, K., Song, J., Ren, K., Iosup, A.: Exploring portfolio scheduling for long-term execution of scientific workloads in IaaS clouds. In: 25th International Conference on High Performance Computing, Networking, Storage and Analysis (2013)
9. Brockwell, P.J., Davis, R.A.: Introduction to Time Series and Forecasting. Springer, New York (2002)
10. Fang, W., Lu, Z., Wu, J., Cao, Z.Y.: RPPS: a novel resource prediction and provisioning scheme in cloud data center. In: 9th IEEE International Conference on Services Computing, pp. 609–616 (2012)
11. Ha, Y., Youn, C.H.: A study on efficient VM reservation method for cloud broker. In: 41st Conference of the KIPS (2014)
12. R. http://www.r-project.org
13. GoGrid. http://www.gogrid.com
14. OpenStack Essex. http://www.openstack.com
15. Intel Hyper-Threading Technology. http://www.intel.com
16. Apache Tomcat 7.0. http://tomcat.apache.org
17. Jersey. http://jersey.java.net/
18. Ren, Y.: A Cloud Collaboration System with Active Application Control Scheme and Its Experimental Performance Analysis, Master's thesis, KAIST (2012)
19. JavaScript Object Notation. http://www.json.org
20. Gson. https://code.google.com/p/google-gson