Dynamic Scheduling Method of Virtual Resources Based on the Prediction Model

Dongju Yang(✉), Chongbin Deng, and Zhuofeng Zhao

Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, Research Center for Cloud Computing, North China University of Technology, Beijing 100144, China
{yangdongju,edzhao}@ncut.edu.cn, xgybdcb@163.com

Abstract. Deploying applications to the cloud has become an increasingly popular way in the industry due to elasticity and flexibility. It uses virtualization technology to provide storing and computing resources to the applications. So how to efficiently schedule virtual resources to ensure the quality of services during the peak, and avoid the waste of resources during the idle is an important research topic in the cloud computing, which aims to minimize the execution cost and to increase the resource utilization. The way based on the monitoring data to scale up or scale down the virtual resources may let virtual resources suffer from over seriously. In this paper, we present a dynamic scheduling method for the virtual resources based on the prediction model. Firstly, we use prediction model to predict the request quantity. And then we combined the prediction result with the load capacity of current resources to compute whether to increase or decrease the virtual resources. Finally, we choose the suitable physical machine to create or recycle the virtual machine. The experimental results show that the prediction model can fit our scene well, and the resource scheduling algorithm can be used to ensure the quality of service in a timely and effective manner.

Keywords: Cloud application · Surge in traffic · Quality of service · Prediction model · Dynamic resource scheduling

1 Introduction

Cloud computing provides an on-demand and scalable delivery model for the users [1]. It has been used to solve the complicated computation and storage problems by more and more governments, research institutions and industries [2] due to provide resources as a service to the users. Effective virtual resource scheduling can improve the utilization of resources and meet the needs of users. The virtual resource scheduling problem is considered to be a combinatorial optimization problem, but also a NP complete problem [3].

The workload of each virtual machine (VM) is always changing, and some may exhibit cyclical changes. We usually tend to over allocate virtual resources in order to ensure the application to have a better performance during the peak [4], which will inevitably lead to low utilization of resources. To operate and manage resources more conveniently, effective monitor is often used [5]. But there will be a traffic surge for

S. Wang and A. Zhou (Eds.): CollaborateCom 2016, LNICST 201, pp. 384–396, 2017.
DOI: 10.1007/978-3-319-59288-6_35

quite a time after application is deployed and then decline gradually based on the monitoring. For example, in the school's teaching information system, a few students will visit this website in ordinary time, but a huge amount of traffic will be generated when students need to select the course. Then the application system needs to extend resources to deal with the requests traffic. We usually monitor the system and warn system managers to deal with the lack of resources. But there is often a delay to schedule resources by monitor and warning.

To deal with the above challenges and to improve the utilization of virtual resources and flexibility of scheduling, we propose a method to dynamically schedule the virtual resources based on the Autoregressive Integrated Moving Average Model (ARIMA) prediction model. Resources will be allocated in advance on the basis of the prediction data. Our paper addresses the following problems:

Find a prediction model to accurately predict the possible application workload of the next time interval.

Dynamic scheduling of virtual machine resources according to the change of workload and prediction data to ensure more efficient use of resources.

The rest of our paper is organized as follows. Section 2 is an overview of the current state of the academic research on the virtual resource scheduling. Section 3 describes the system architecture. Section 4 focuses on the prediction model we will use. Section 5 describes the virtual machine scheduling strategy and the corresponding experiment details are illustrated in Sect. 6. Section 7 presents conclusions and future work.

2 Related Works

Since cloud computing uses virtualized resources, scheduling and resource allocation are the important research topics [6]. It is a hot research topic that how to use effective scheduling strategies to reduce the cost of execution and improve the utilization of resources.

Silpa et al. [6] discuss the current scheduling algorithm that has been published in cloud computing, in this paper, 15 different algorithms are studied and compared, such as fuzzy genetic algorithm based on task scheduling.

Some resource scheduling frameworks are put forward. Singh et al. [7] present an efficient cloud workload management framework under the premise of certain workload and quality of service. Shuja et al. [8] introduce a resource scheduling framework for efficient utilization.

Hassan et al. [9] present Nash bargaining to save resources and optimize the number of servers. Singh et al. [10] propose service quality indicators to optimize the execution time and avoid waste of resources. And some studies about resource optimization algorithms, such as the article [11, 12].

Zheng et al. [13] point out that virtual machine placement is also a way to optimize the resources, two kinds of virtual machine placement method, incremental placement and consolidated placement, are discussed and a new virtual machine placement strategy

is proposed. Wang et al. [14] explore an energy and QoS-aware VM placement optimization approach based on particle swarm optimization.

Liu et al. [15] present an approach to process the dynamic user service requests more cost-effectively. Zhou et al. [16] introduce a dynamically adjust the virtual resource rental strategy to help cloud service providers maximizing profits.

There are also some researches on prediction methods. Salah et al. [17] use Markov chains to estimate the number of virtual machine instances that are required to be allocated under a given Service Level Object (SLO) standard. Shyama et al. [18] study a Bayesian model to predict resource needs of CPU and memory intensive applications in the short term and long term. However, the article [17] focus on load balance, and in [18] mainly focus on CPU cores and RAM, but they are not very suitable for our scenarios.

3 System Architecture

3.1 Architecture Description

The system architecture used in this paper showed in Fig. 1 includes the following layers User Layer, Application Layer, Control Layer, Virtual Layer, Physical Layer.

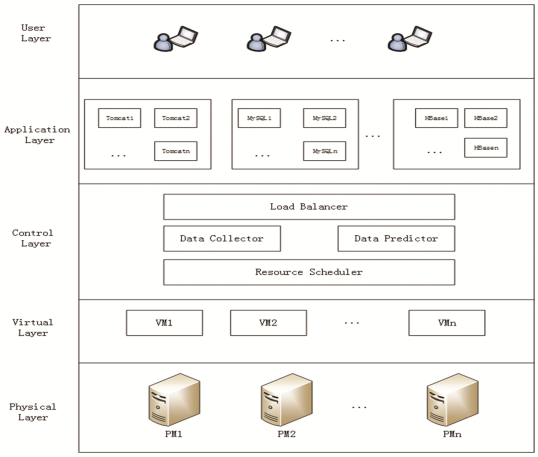

Fig. 1. System architecture diagram

User Layer: Users who use the application in cloud plateform.

Application Layer: It provides basic application environment, such as tomcat server, MySQL database, Hbase, etc.

Control Layer: It includes Load Balance Module, Data Collector Module, Data Predictor Module, Resource Scheduler Module.

Load Balancer Module: All requests submitted by the users will be forwarded through the Load Balancer to our web server. We use Nginx as a load balancing server in the system architecture.

Data Collector Module: This module is mainly to get the data through real-time monitoring. Some data will be collected by this module, such as CPU, memory, request quantity, etc., and then data will be pre processed. We store the requests per second to database in order to analyze historical data in the future. The module will calculate the response time for each request, which is ready for us to assess the application response time.

Data Predictor Module: The module uses the data collected in the Data Collector Module to calculate the number of requests to be reached at the next time interval of the application. The next time interval is the amount of user requests that the application will achieve after 5 min. On one hand, we mainly study the prediction of the application workload in the short term; On the other hand, the high frequency statistics and calculation will cause some pressure on our server. Generally each virtual machine can be completed in 30–96 s from creation to deployment [17], so the time is enough for us to deal with the coming pressure on the system. Of course, you can adjust the time interval.

Resource Scheduler Module: Using the Data Predictor Module to get the predicted requests of the application to decide to increase or reduce the virtual resource.

Virtual Layer: It supply virtual resource and composes a virtual resource pool.

Physical Layer: It is infrastructure and mainly includes the physical servers and the virtual machines deployed on the physical servers, which provides the underlying resources for the application.

4 Workload Prediction Methods

In this section we will discuss several methods for predicting the workload. We simplify the workload to the number of user requests in our scenario. But the requested quality of service has been taken into account, we assume that the response time of the request is satisfied in the range of 0 to 2 s, the maximum workload of the server is considered under such a condition. We mainly discuss three prediction methods: Moving Average method, Polynomial Fitting method, ARIMA Model method. We have also introduced the error to analyze the three methods which is more suitable for our scenario.

① Moving Average: Moving Average is one of widely known technical indicator used to predict the future data in time series analysis [19]. In statistics, a moving average is a calculation to analyze data points by creating series of averages of different subsets of the full data set. It includes simple moving average, the cumulative moving average and weighted moving average [20]. We use simple moving average in this paper.

We assume that the requests for a certain time of application is r_i, then the sequence of requests can be expressed as $R = \{r_1, r_2 \ldots r_i \ldots r_n \mid n < T\}$, where T is our measurement time. According to the definition of the moving average, we can predict the amount requested at time $n + 1$ is:

$$Pre\_R = \left(r_1 + r_2 + \ldots + r_n\right) \Big/ n. \tag{1}$$

Pre_R is the predicted value at time $n + 1$ in Eq. (1), n denotes the average movement cycle, r_1 to r_n are the first n values.

② Polynomial Fitting method: Polynomial regression is a form of linear regression in which the relationship between the independent variable x and the dependent variable y is modelled as an nth degree polynomial [21]. Linear relationship is not a good description of the relationship between the amounts of application requests at different times, so the polynomial method is one aspect to consider.

We assume that the ti time corresponds to a request amount of r_i, and (t_i, r_i) is a point on a two curve, namely:

$$r_i = at_i^2 + bt_i = c. \tag{2}$$

Since it is the conic section, we only need to use three sets of values to be able to seek out a, b and c solution. If we predict r_{i+1}, just need three points, that is, $(t_{i-2}, r_{i-2}), (t_{i-1}, r_{i-1}), (t_i, r_i)$.

③ ARIMA Model: The ARIMA model is built based on Markov random process, which not only absorbs the dynamic advantages of the regression analysis, but also the advantages of moving average [22]. Non-seasonal ARIMA model use ARIMA (p, d, q) to express, wherein p, d, q are non-negative integers, p is the order of autoregressive model; d indicates the degree of differencing, q represents the moving average model order. Seasonal ARIMA model using ARIMA (p, d, q) (P, D, Q) m, m refers to the number of cycles per season, P, D, Q, respectively, refers to the autoregressive, moving average and differential [23].

5 Scheduling Algorithm and Implementation

5.1 VM Provisioning Algorithm and Implementation

We assume that a certain time point T, the requests for our application is R. We simplify the application workload to the user's request. Our machine (Web Server) number is N in the current state. We adopt a polling workload allocation strategy. The number of requests for each web server shared by the load balancer is R/N.

We use Data Collector Module to collect raw data and pre-processing data. We use $T_{response}$ to denote the response time for each request, and pass the request of the pre processed data to the ARIMA module. Each time the user requests for the application will be recorded in the database, we can calculate the amount of user requests of each interval, abbreviated R_i. We predict requests of the next time interval, assumed to be $S_{predict}$ based on R_i. $S_{predict}$ and $S_{current\_max}$, the maximum workload that the virtual machines can bear under the current scale, will be passed to the Dynamic Scheduling Module. If $S_{predict}$ is greater than $S_{current\_max}$, then the DSM will find the right virtual machine template in physical servers to configure virtual resources. The way we are using is to randomly select a virtual machine that is providing services for the application and obtain its information (the using operation system, CPU kernel number, memory, etc.). Finding the information from the template in physical servers and here we assume that each virtual machine that has been used is created by a template in the physical machine server. In order to reduce the false positive rate, we can set the number of times to meet the judgment. We begin to configure virtual resources, when the times are more

than a specified number of times. Assuming that each virtual machine created by virtual machine template can withstand the maximum workload of MHVMW. The difference between the predicted workload and the maximum workload that the current size of the virtual machines can withstand is the workload that we need to create virtual machines to support. We can roughly calculate the number of virtual machines that need to be created at a time combined with MHVMW, using N_{need} to denote the calculated number. To determine the location of the virtual machine created, first of all, physical machines workload will be sorted from least to most, and we use the word "size" to denote the number of physical machines that meet the conditions for creating virtual machines, then taking the remainder of "size" from 0 to N_{need}. The purpose of this is to create a virtual machine in a low workload physical machine. Finally, the strategy returns the positions of the virtual machine to be created.

Symbol definition:

$LVMW_i$: Lowest Virtual Machine Workload, the minimum workload value allowed by each virtual machine, if the current workload is lower than the value, we will consider it to be an idle virtual machine.

$HVMW_i$: Highest Virtual Machine Workload, the maximum workload value allowed by each virtual machine, if the current workload is higher than the value, we will believe that the quality of service provided can not meet the needs of users.

$S_{predict}$: The predicted workload value obtained from the prediction model.

$N_{current}$: Number of current virtual machines.

$S_{current\_max}$: The maximum workload that the virtual machines can bear under the current scale.

$$S_{current\_max} = \sum_{i=1}^{Ncurrent} HVMW_i \tag{3}$$

S_{curent_min}: The minimum workload required for the virtual machines at the current scale.

$$S_{curent\_min} = \sum_{i=1}^{Ncurrent} LVMW_i . \tag{4}$$

MHVMW: Model Highest Virtual Machine Workload, the virtual machine can be made into a template, this variable represents the maximum workload that it can take after the template is turned into a virtual machine.

PM: Physical Machine, $PM = \{PM_1, PM_2 \ldots PM_i \mid i < PMCount\}$.

VM: Virtual Machine, $VM = \{VM_1, VM_2 \ldots VM_j \mid j < VMCount\}$.

Algorithm 1. VM Provisioning

Input: $S_{predict}$, $N_{current}$, $S_{current\_max}$, MHVMW, PM, VM
Output: postions that VM will place.

```
1: while true do
2:   if  S_predict >= S_current_max  then
3:              MHVMW = findTemplate(PM,VM)
4:              N_need = (S_predict  - S_current_max) / MHVMW
5:      return addVMPos(PM, Nneed)
6:   else
7:             break;
8: end while
9:   function findTemplate(PM,VM)
10:         randomly select VM[j] from VM
11:         for(template in PM)
12:           if(VM[j]== template)
13:             MHVMW = VM[j]
14:             break;
15:           else
16:             continue
17:         end for
18:         return MHVMW
19: function  addVMPos(PM, Nneed)
20:          list = {}
```

```
21:       for pm in PM do
22:      if (pm.currentWorkload  <
pm.higestworkload())&& pm.hasResource() then
23:          list.add(pm)
24:       end for
25:       sortByAsc(list)
26:       size = list.size()
27:       for i=0;i< N_need ;i++ do
28:         positions.add(list.get(i % size))
29:       end for
30:       return positions
```

Algorithm 2: VM Recycling

Input: $S_{predict}$, S_{curent_min} , VM

Output: Recycling virtual machine success (true) or failure (false)

```
1:        while true do
2:        if S_predict <= S_curent_min then
3:            rmPos = delVMPos(VM)
4:         return destoryVM(rmPos)
5:        else
6:         break;
7:        end while
8:        function delVMPos(VM)
9:         sortByAsc(VM)
10:        return VM.get(0)
```

5.2 VM Recycling Algorithm and Implementation

It is obviously a waste of resources that virtual resources remain the largest scale after the peak of the traffic is over. VM recycling Algorithm will be enabled when the current workload is less than the specified minimum workload. We will carry out an ascending sort to workload of all virtual machines when the recycling algorithm is enabled. In order to reduce the false positive rate, we can also use the method that predicted times reach the number of count times specified to enable VM Recycling Algorithm. We will poll the VM workload and recycle the VM that is the minimum workload until the remaining one virtual machine to provide services.

6 Experimental Design and Results

6.1 Experimental Environment

In order to verify whether the prediction model and the resource scheduling algorithms are effective, we do some experiments in this part. The experimental environment used is: 2 LoadRunner servers, 1 load balance server, 4 application servers, 1 MySQL servers and 3 physical machines (Table 1).

Table 1. Configuration information

Name	CPU	Memory	Hard disk
LoadRunner1	2 cores 2.0 GHz	4 GB	130 GB
LoadRunner2	2 cores 2000 MHz	4 GB	100 GB
LoadBalance	1 core 2000 MHz	2 GB	100 GB
VM1 (Application Server)	1 core 2000 MHz	2 GB	100 GB
VM2 (Application Server)	1 core 2000 MHz	2 GB	100 GB
VM3 (Application Server)	2 core 2600 MHz	2 GB	100 GB
VM4 (Application Server)	1 core 2000 MHz	2 GB	100 GB
MySQL	2 core 2.67 GHz	4 GB	100 GB
PM1	48 core 2.6 GHz	64 GB	830 GB
PM2	48 core 2.6 GHz	32 GB	550 GB
PM3	48 core 2.6 GHz	32 GB	550 GB

6.2 Experiment and Result Analysis of the Predict Methods

We deploy our application in two groups of two virtual machines, one group using the default policy, another group using the proposed strategy in this paper. Using 2 LoadRunner servers to simulate the request, and then using our prediction model to predict. We will show the number of requests for the site below and the curve drawing of the real value and the predictive value obtained by using the methods in the Sect. 4.

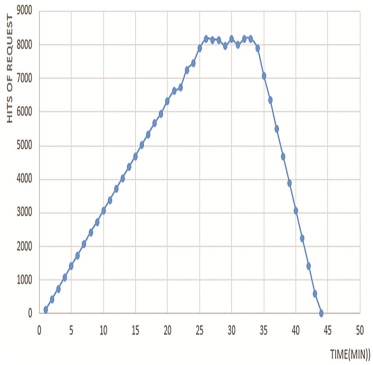

Fig. 2. Site request volume graph

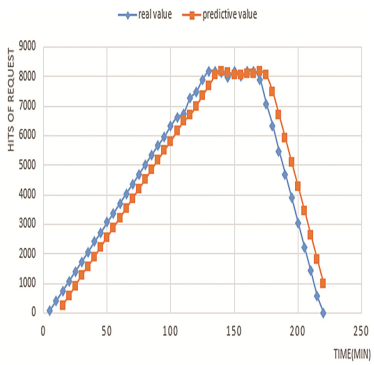

Fig. 3. The real value and predictive value of MA

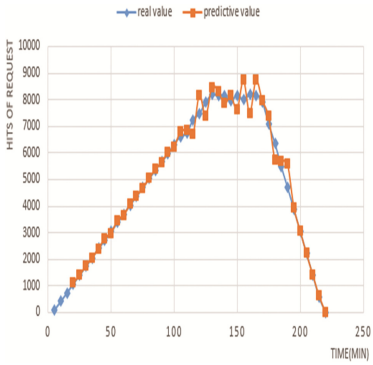

Fig. 4. The real value and predictive value of polynomial fitting

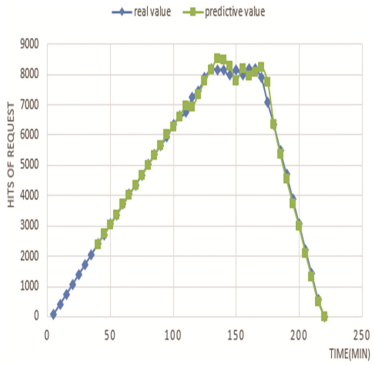

Fig. 5. The real value and predictive value of ARIMA

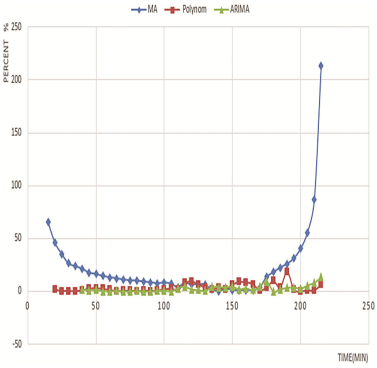

Fig. 6. Relative error curve of MA, Polynomial and ARIMA

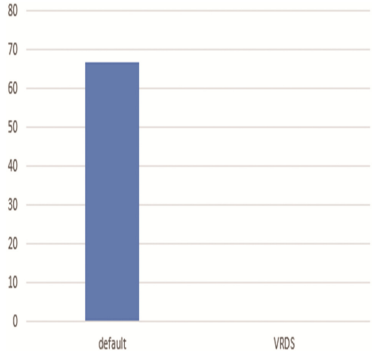

Fig. 7. Default policy and VRDS comparison graph with failed requests

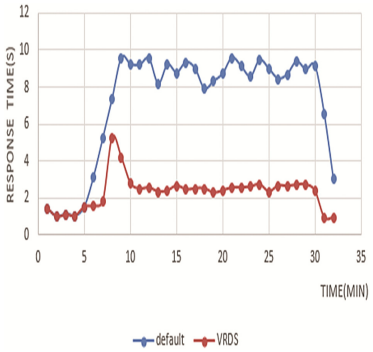

Fig. 8. Default policy and VRDS response time comparison graph

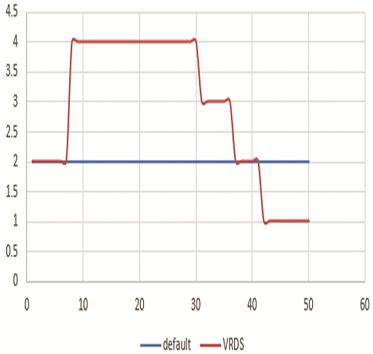

Fig. 9. Changes in the number of VMs under the default policy and VRDS

Figure 2 is the general trend of accepting requests from our website. The number of requests for the site is a gradual increase from less to more, after a certain period of time, the requests will gradually decline. This is the scenario we need to deal with in the project. Figure 3 uses moving average method to fit our scenarios, and the overall trend is well fitted, but we can find out from the graph, the fitting curve of this method is relatively backward, which cannot be very good to help us to predict the future trend of requests. Figure 4 method is very good at the request of the amount of the increase and decrease of the scene, but the volatility of prediction results is larger in the peak period of requests. In our scenario, the Fig. 5 method can be well fitted with both increasing and decreasing values. In order to observe the real value and the predicted value of the scene, we introduce the error analysis. Figure 6 shows the relative error of the three methods.

From Fig. 6, we can see that the relative error of ARIMA is relatively stable, and the error is the smallest of the three methods, we will use the model to predict in our scene.

6.3 Resource Scheduling Experiment Results and Analysis

We use our prediction model in the virtual machine scheduling algorithm. By prediction, it will inform our Virtual Resource Dynamic Scheduling (VRDS) algorithm when the user's traffic continues to increase. And then our VRDS algorithm calculates the size of the required resources, select a reasonable resource scheduling and decide to create or recycle the virtual machine. In the default policy we give a fixed number of virtual machines, and in the VRDS strategy will base on the load situation to do resource scheduling. In Fig. 7, it shows the comparison of the number of requests for the users to access the web site under the two different strategies. VRDS strategy can effectively reduce the number of failed requests. It is assumed that the maximum response time for each user to request is 2 s. The response time of the user request is illustrated in the case of Fig. 8 with two strategies, which are continuously increasing with the number of requests on the website. VRDS strategy can be more close to the response time we set.

Figure 9 shows the change in the number of VMs, VRDS can create and recycle VMs in different stages.

In summary, in our scenario, the prediction model used in this article can be more fitting site requests constantly increasing amount of requests to the maximum amount and gradually decreasing after the scene. VRDS algorithm combined with the prediction model in this paper can effectively and timely response to the site of the high workload situation.

7 Conclusion and Future Work

With the advent of big data era, the data is growing geometrically. Our web site or application is likely to generate a huge surge in traffic because of sudden or hot events. Relying solely on the traditional way apparently is unable to cope with such pressure, and cloud computing brings us a new revolution. Deploying our applications in the cloud will help us to avoid the collapse of the application because of heavy workload. However, there are still many deficiencies in the dynamic scheduling of cloud resources.

In this paper, we proposed a method for dynamic scheduling of virtual resources based on prediction. The prediction will help us to make the decision to deal with the load too much earlier, and change the passive into the initiative. By actively calculating the size of the virtual resources that are needed to cope with the current workload and the decision to create a reasonable location for the virtual machine, we will be more rapid in response to the heavy workload of cloud applications and ensure that the application can easily cope with the massive use of access.

Of course, that we simplify the server workload to the user's request for the application is not enough to completely express the actual situation of the workload, and the workload prediction method is still not fine enough, the scene is relatively simple. In future work we will consider more factors that are more close to the actual situation and simulate our experiments, and apply our algorithm to more practical scenarios.

Acknowledgments. This work is supported by Key Program of Beijing Municipal Natural Science Foundation "Theory and Key Technologies of Data Space Towards Large Scale Stream Data Processing" (No. 4131001).

References

1. Somasundaram, T.S., Govindarajan, K.: CLOUDRB: a framework for scheduling and managing High-Performance Computing (HPC) applications in science cloud. Future Gener. Comput. Syst. **34**, 47–65 (2014)
2. Zhao, Y., Li, Y., Raicu, I., Lu, S., Tian, W., Liu, H.: Enabling scalable scientific workflow management in the Cloud. Future Gener. Comput. Syst. **46**, 3–16 (2015)
3. Yuan, H., Li, C., Du, M.: Optimal virtual machine resources scheduling based on improved particle swarm optimization in cloud computing. J. Softw. **9**(3), 705–708 (2014)
4. Huang, Q., Shuang, K., Xu, P., Li, J., Liu, X., Su, S.: Prediction-based dynamic resource scheduling for virtualized cloud systems. J. Netw. **9**(2), 375–383 (2014)

5. Aceto, G., Botta, A., de Donato, W., Pescape, A.: Cloud monitoring: a survey. J. Comput. Netw. **57**(9), 2093–2115 (2013)
6. Silpa, C.S., Basha, M.S.S.: A comparative analysis of scheduling policies in cloud computing environment. Int. J. Comput. Appl. (0975–8887) **67**(20), 16–24 (2013)
7. Singh, S., Chana, I.: QRSF: QoS-aware resource scheduling framework in cloud computing. J. Supercomput. **71**, 241–292 (2015)
8. Shuja, J., Bilal, K., Madani, S.A., Khan, S.U.: Data center energy efficient resource scheduling. Cluster Comput. **17**, 1265–1277 (2014)
9. Hassan, M.M., Alamri, A.: Virtual machine resource allocation for multimedia cloud: a Nash bargaining approach. Procedia Comput. Sci. **34**, 571–576 (2014)
10. Singh, S., Chana, I.: Q-aware: quality of service based cloud resource provisioning. Comput. Electr. Eng. **47**, 138–160 (2015)
11. Liu, Z., Zhou, H., Fu, S., Liu, C.: Algorithm optimization of resources scheduling based on cloud computing. J. Multimedia **9**(7), 977–984 (2014)
12. Shao, Y.: Virtual resource allocation based on improved particle swarm optimization in cloud computing environment. Int. J. Grid Distrib. Comput. **8**(3), 111–118 (2015)
13. Zheng, Q., Li, R., Li, X., Shah, N., Zhang, J., Tian, F., Chao, K.-M., Li, J.: Virtual machine consolidated placement based on multi-objective biogeography-based optimization. Future Gener. Comput. Syst. **54**, 95–122 (2016)
14. Wang, S., Zhou, A., Hsu, C.H., et al.: Provision of data-intensive services through energy- and QoS-aware virtual machine placement in national cloud data centers. IEEE Trans. Emerg. Top. Comput. **4**(2), 290–300 (2016)
15. Liu, Z., Wang, S., Sun, Q., et al.: Cost-aware cloud service request scheduling for SaaS providers. Comput. J. **57**(2), 291–301 (2014)
16. Zhou, A., Wang, S., Sun, Q., et al.: Dynamic virtual resource renting method for maximizing the profits of a cloud service provider in a dynamic pricing model. In: International Conference on Parallel and Distributed Systems, pp. 944–945. IEEE Computer Society (2013)
17. Salah, K., Elbadawi, K., Boutaba, R.: An analytical model for estimating cloud resources of elastic services. J. Netw. Syst. Manage. **24**, 285–308 (2016)
18. Shyama, G.K., Manvi, S.S.: Virtual resource prediction in cloud environment: a Bayesian approach. J. Netw. Comput. Appl. **65**, 144–154 (2016)
19. Hansun, S.: A new approach of moving average method in time series analysis. In: 2013 Conference on New Media Studies (CoNMedia), pp. 1–4 (2013)
20. Wikipedia. https://en.wikipedia.org/wiki/Moving_average
21. Wikipedia. https://en.wikipedia.org/wiki/Polynomial_regression
22. Li, J., Shen, L., Tong, Y.: Prediction of network flow based on wavelet analysis and ARIMA model. In: International Conference on Wireless Networks and Information Systems, 2009, WNIS 2009, pp. 217–220 (2009)
23. Wikipedia. https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average