Optimization Service Migration Scheme for Load Balance in Cloud Computing

Mao Yingchi, Wang Jiulong, Zhu Lili, Jie Qing
College of Computer and Information Engineering
Hohai University
Nanjing, China, 210098
maoyingchi@gmail.com

***Abstract*—Large-scale development of Web services and cloud computing bring enterprises and users new experience. In the traditional service system architecture, large-scale number of service access and execution make some of the nodes overload and seriously affect the quality of services. Therefore, it becomes a research hotspot to establish high quality and reliable Web services platform. Cloud computing is a novel computing paradigm to provide better performance on load balance. The paper proposed an overall framework of Web services migration. In this Web service migration framework, two strategies are provided, one is service replication, and the other is service transfer. Under the different situations, the appropriate service migration strategy is adopted based on the current workload. The experiments results show that the proposed Web Services migration scheme can obtain better performance on the load balancing.**

Keywords-Cloud computing; load balance; service migration; service replication; service transferring

I. INTRODUCTION

In the traditional Web Services framework, each service obtains a fixed system resources at the deployment phase. However, in the real applications, the user's access frequency is fluctuate, which results in overloading on some nodes, and seriously affects the quality of service and the overall performance of the Web Services system. Compared with the traditional computing model, Emergence of cloud computing greatly changed the traditional software usage model, and also changed the software development model. Establishing a high quality and reliable Web services framework in the cloud can better support the Web services management and maintain the quality of Web services. In this paper, an overall framework of Web services migration was proposed. In this Web service migration framework, two strategies are provided, one is service replication, and the other is service transfer. Under the different situations, the appropriate service migration strategy is adopted based on the current workload. The amount of the users' access request can be used to measure the load on the virtual machine. The migration of the service for dynamic scheduling and dynamic deployment can ensure better quality of service to be executed at the same time.

II. RELATED WORK

Web services migration refers to migrate services from the original location to the destination for the load balancing in the system [1]. In a dynamic large-scale, and automated execution system, service migration is a more satisfactory service deployment method [2]. Adopting the appropriate service migration scheme, the system can maintain better quality of service. At present, there are two categories migration schemes, static and dynamic schemes. The static migration schemes suspend or terminate the services during the migration process, so the static schemes cannot be adopted in the SaaS systems. As to the dynamic migration schemes, two important issues should be solved: (1) service code and data encapsulation and restart at the destination node after service migration, (2) files configuration, and log information and related documents migration [3]. Lu et al. proposed core migration service method to improve the services viability [4].

In order to meet the flexibility and scalability in the Web Services framework, services migration can be divided into service replication and services transfer based on the granularity of the services migration. Xu adopted services replication scheme to establish a distributed mutual aid system to alleviate the network congestion [5]. After analysis the performance of grid services, considering the dynamical grid services and their status, a stateful grid service replication mechanism based on virtual workspace was presented [6]. Shen et al. discussed the services replication by using the middleware technology in the Web Services framework [7]. In addition, one framework for Web services migration was proposed based on SOA [8]. In the proposed framework, it can determine the services migration based on the servers' workload, services type, which can reduce the communication costs and improve the quality of services.

The above research emphasized service replication or services transfer separately without considering two scheme together. In this paper, an overall framework of Web services migration was proposed. In this Web service migration framework, service replication and service transfer were provided. Under the different situations, the appropriate service migration strategy is adopted based on the current workload.

III. SERVICE MIGRATION FRAMEWORK

This paper proposes service migration scheme to predict and analyze the nodes' workload, and execute the services replication and services transfer in the different situation. In order to load balance, the service migration scheme includes four parts: load measurement and prediction, service migration trigger, migration selection, and services replication or transfer scheme execution.

A. Load Measurement and Prediction

The life cycle of the service instance starts from the request generation. User sends a request to the node and the node begins a thread for processing. If the node has received the large number of user requests, the thread pool will dispatch the extra requests to the request queue and wait for the successive execution. When the request queue is full, the node refuses the successive requests arrival. Therefore, the number of the receiving requests in one unit of time can be used to estimate the workload of the node.

A Web service executing on a virtual node, the maximum accepted requests of the node in one unit of time is the full workload of the node. Assume that one Cloud computing system has n virtual nodes, $N_1, N_2, ..., N_n$, and m Web Services, $S_1, S_2, ..., S_m$. In order to obtain the services execution ability of the node, it can use the following matrix:

$$\begin{array}{cccccc} & S_1 & S_2 & S_3 & ... & S_{\mathrm{m}} \\ N_1 & n_{11} & n_{12} & n_{13} & ... & n_{1m} \\ N_2 & n_{21} & n_{22} & n_{23} & ... & n_{2m} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ N_{\mathrm{n}} & n_{n1} & n_{n2} & n_{n3} & ... & n_{nm} \end{array} \quad (1)$$

Suppose there are j services $S_1, S_2, ..., S_j$ executing on node N_i , j connections of these services and j requests sent via the connections in one unit time with $R_1, R_2, ..., R_j$.

For node N_i , it can compute the weighted average of the maximum of services requests, denoted as the maximum workload of the node N_i at the time t . That is, the maximum workload of services requests at Node N_i can be calculated as follows:

$$Load_{node\_i} = \sum_{k=1}^{i} \left(\frac{R_k}{\sum_{k=1}^{i} R_k} n_{jk} \right) \quad (2)$$

For node N_i , the real-time workload of node N_i at the time N_i is denoted as $ACT^t_{node\_i}$, which can be calculated as the ratio of the real-time requests to the maximum requests. The detail is as follows:

$$ACT^t_{node\_i} = \frac{\sum_{k=1}^{i} R_k}{Load_{node\_i}} \quad (3)$$

From the formula (3), the current workload of virtual node N_i can be calculated via the number of requests.

By adapting the appropriate time interval, it stores the real-time workload of each node $ACT^t_{node\_i}$ at every interval. All of the stored real-time workload $ACT^t_{node\_i}$ can combined into the time sequence of real-time workload to predict the workload at the next interval.

Due to the self-similarity of the variances of workload, it is feasible to analyze and forecast the nodes' workload based on the time sequence of workload. In this paper, exponential smoothing method were adapted to predict the workload on the node. The predictive value of the workload at the next time $t+1$ is denoted as $PRE^{t+1}_{node\_i}$ and calculated as follows:

$$PRE^{t+1}_{node\_i} = \alpha ACT^t_{node\_i} + (1-\alpha) PRE^t_{node\_i} \quad (4)$$

The service migration model will utilize $PRE^{t+1}_{node\_i}$ to trigger the service migration.

B. Services Migration Trigger

To further avoid the workload Ping-Pong effect on the node, the dual-threshold scheme was adopted to trigger the services migration. The workload of the overloaded node drops below the threshold after the services migration, while the node with low workload is over the threshold. The migration procedure can result in the frequent migration from the overloaded node to the destination with the low workload. That is called Ping-Pong effect.

In the dual-threshold scheme, it set one high threshold λ_{high} and one low threshold λ_{low} . Node N_i compares the prediction value of workload $PRE^{t+1}_{node\_i}$ with the high threshold λ_{high} . If the $PRE^{t+1}_{node\_i} > \lambda_{high}$, it means that the node is overloaded. The services executing on this node will be migrated. If $PRE^{t+1}_{node\_i} < \lambda_{low}$, it means that this node is idle. The extra services can be migrated to the idle node. If $\lambda_{low} \le PRE^{t+1}_{node\_i} \le \lambda_{high}$, it means that the node has enough resource to support the services execution without any migration. The dual-threshold scheme can reduce the workload of node over the threshold λ_{high} as well as avoid the Ping-Pong effect.

C. Services Migration Selection

After a Web service migration has been triggered, the overloaded node will decide to replicate or transfer the service based on the service's workload. If the overloaded service takes up the most of node's workload, it may result in the overloading at the destination just transferring the heavy service. Under this situation, we adopt the service replication scheme to copy the same service to other node with the light load. Then, the users' requests can be distributed to these nodes. Two nodes execute the same service to reduce the workload of each node. If the overloaded service just takes up small parts of load, the overloaded service will be transferred to the light-loaded node. When all of the services evenly occupying the workload of the node, it will reduce the workload of the node after transferring one service. Thus, one of appropriate service on the node is selected and transferred to other node with light load. Meanwhile, the service on the source node is shut down to reduce the workload on the source node.

The detail selection migration procedure as follows. Assume that there are m services $S_1, S_2, ..., S_m$ in one virtual node, the workload of each services are $L_1, L_2, ..., L_n$, respectively. For any service S_k , it can compute the

workload L_k. Firstly, it can store the time sequence of the weighted average L_{node} of the maximum service requests on this node, and the time sequence of the real-time requests $R_1, R_2, ..., R_k, ..., R_m$ corresponding to each service. Adopting exponential smoothing methods, it can compute the weighted average L'_{node} of the maximum service requests and the real-time requests to the services $R'_1, R'_2, ..., R'_k, ..., R'_m$ at the next time $t+1$. Thus, it can obtain the workload of each service $L_1, L_2, ..., L_k, L_n$ at the time $t+1$ as follows:

$$L_k = \frac{R'_k}{L'_{node}} (1 \le k \le m) \quad (5)$$

When an overloaded node triggered the service migration, if the current workload of service $L_k > \lambda_{high}$, it think that the service occupied the large part of resources of the node, the system will trigger the service replication. If $L_k < \lambda_{low}$, the service can share the resource evenly, then the system will execute the service transferring.

D. Web Services Replication Stategy

In order to compute the number of connections, and effectively control the workload of source and destination node simultaneously, the Web services replication scheme was executed by adopting the sender-driven strategy to reduce the Ping-pong effect.

The detail of Web services replication scheme is as follows. Firstly, the node with light workload below λ_{low} is selected as the destination node. After the services are migrated to the selected destination, the users' connections are also migrated to the destination in batch. For example, there are 100 randomly selected users' connections each time. When the users' status were migrated, the number of connection requests were recorded in the load analysis model.

The extended sender-driven strategy is as follow. Firstly it select light load node that the node's load below λ_{low} as destination node. After the service is migrated, the user connections are also migrated. For example, there are 100 random users connected to the requested services. When the users' connections are migrated, it will store the ratio of the number of connections to the number of requests in the workload analyzer. If the workload of the source node is greater than λ_{high}, the service replication scheme will be activated and the appropriate services will be migrated. After the services have been migrated, if the workload of source node is still greater than λ_{high} and the workload of the selected destination is greater than λ_{low}, it will select one node with the light node as the destination node where the extra services are migrated till the workload is lower than λ_{low} ultimately.

E. Web Services Transferring Strategy

In the service transferring strategy, the selection of transferred services is modelled from the view of the workload and the transferred data. The most important factor of migration cost is the amount of migrated data, thus, the amount of migrated data is used to measure the migration cost.

Assume that there are *n* virtual nodes $N_1, N_2, ..., N_n$ in the system, the data provided by each service are $C_1, C_2, ..., C_n$, respectively, and the workload of each service are $L_1, L_2, ..., L_n$. For any one of service S_k, its workload can be obtained by the above formula (5).

It set the parameters $a_1, a_2, ..., a_i, ..., a_n (1 \le i \le n)$ to 1 or 0. If $a_i = 0$, the service S_i cannot be transferred. If $a_i = 1$, the service S_i will be transferred. The transferred workload of one node L_{trans} is as follow.

$$L_{trans} = a_1 L_1 + a_2 L_2 + ... + a_k L_k + ... + a_n L_n = \sum_{i=1}^{n} a_i L_i \quad (6)$$

Therefore, the amount of transferred data is as follows:

$$C_{trans} = a_1 C_1 + a_2 C_2 + ... + a_k C_k + ... + a_n C_n = \sum_{i=1}^{n} a_i C_i \quad (7)$$

On the one hand, it should maximize the amount of transferred data C_{trans} before the workload reducing below the high threshold λ_{high} during the service migration. On the other hand, it should minimize the amount of transferred data C_{trans} after the workload have decreased below the high threshold λ_{high}. Therefore, this problem can be converted to the standard knapsack problem. It can be solved via the knapsack problem dynamic programming method. The optimal solution can be calculated as follows:

$$C_{trans} = f[m][\lambda_{high}] = Max \begin{cases} f[m-1][\lambda_{high}] \\ f[m-1][\lambda_{high} - L_m] + C_m \end{cases} \quad (8)$$

When C_{trans} is set to the maximum value, it can obtain every $a_i (1 \le i \le n)$. If $a_i = 1$, the service S_i can be migrated to the destination so that the workload can be reduced below the threshold and save the migration cost.

IV. EVALUATION

A. Experiments Setup

The proposed service migration framework is provided for the large-scale cloud computing system. In the simulations, a cloud simulation platform CloudSim is used to evaluate the performance of the proposed service migration strategies in the large-scale computing environment.

In the CloudSim, 150 identical virtual nodes are set to configure the Web services. Every virtual node is configured to one core CPU, 1,000 MIPS execution rate, 2GB RAM, and 30GB storage. In addition, European LHC Distributed Computing Center's workload data are used as simulation data set.

B. Experiments Results

Firstly, the performance of load balancing is evaluated. As shown in Figure 1, the proposed service replication strategy can ensure the variance of workload between the

high threshold and low threshold, which can maintain the load balance among the nodes. This is because the dual-threshold scheme can migrate the appropriate services on the overloaded nodes triggered by the low and high thresholds. Moreover, the proposed strategy can provide the resource utilization rate with the same workload of the system.

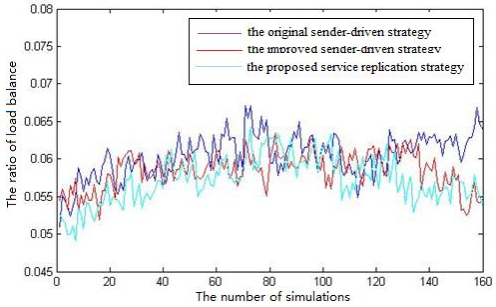

Figure 1. load balance performance with service replication strategy

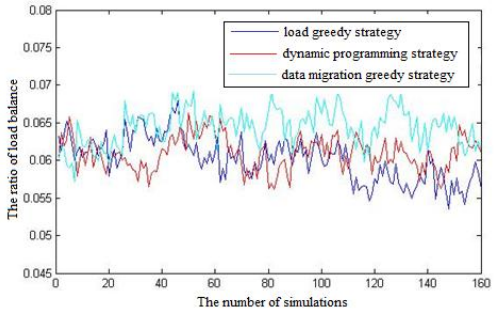

Figure 2. load balance performance with service transferring strategy

To evaluate the performance of the proposed service transferring strategy, the dynamic programming strategy is adopted for service transferring. We compared it with the load greedy strategy and data migration greedy strategy. As shown in Figure 2, the dynamic programming strategy can achieve better performance on load balance than that of two other schemes. It can keep the workload of the nodes in a reasonable range.

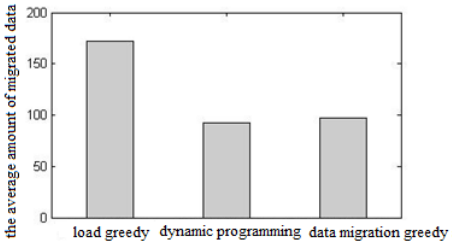

Figure 3. The average amount of migration data with different strategy

In addition, it should evaluate the performance in terms of the average amount of data, the number of services in every migration. The simulation results are shown in Figure 3 and 4, respectively. The simulation results illustrate that the amount of migrated data via the load greedy strategy is significantly high than that of the other two strategies. From the point of view of the number of migrated services, the data migration greedy strategy should migrate more services, which results in the decrease of the system efficiency.

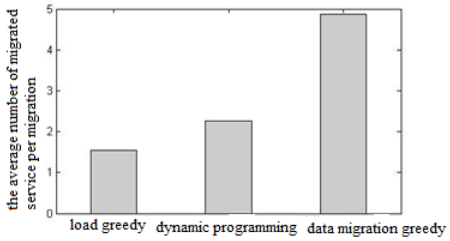

Figure 4. The average numbert of migrated service with different strategy

V. CONCLUSION

Load balance is one of the most important issues in the cloud computing systems. To support good performance on load balance, the Web services migration framework was proposed. In the framework, the service replication and the service transferring strategies were discussed. Under the different situations, the appropriate service migration strategy is adopted based on the current workload. The migration of the service for dynamic scheduling and dynamic deployment can ensure better quality of service to be executed at the same time.

ACKNOWLEDGMENT

This research is partially supported by the National Key Technology Research and Development Program of the Ministry of Science and Technology of China under Grant No. 2013BAB06B04; National Science Foundation of China under Grant No. 60903018, 61272543; Nature Science Fund of Jiangsu Province under Grant No. BK2012584, Fundamental Research Funds for the Central Universities 2013B06914; Open Funds of Huaian Research Institute Hohai University.

REFERENCES

[1] T. Setzer, K. Bhattacharya, H. Ludwig, "Decision support for service transition management Enforce change scheduling by performing change risk and business impact analysis," Network Operations and Management Symposium, (NOMS 2008), IEEE, pp. 200-207, 2008.

[2] K. Oikonomou, I. Stavrakakis, A. Xydias, "Scalable service migration in general topologies," International Symposium on World of Wireless, Mobile and Multimedia Networks, WoWMoM 2008.

[3] J. Meehean, M. Livny, "A service migration case study: Migrating the Condor schedd," Midwest Instruction and Computing Symposium. 2005.

[4] T. Lu, N. Gu, "Enabling the Analysis of Successful and Safe Transition between Service Cores for Service-Based System Survivability," 2nd International Conference on Pervasive Computing and Applications, ICPCA 2007, 408-413, IEEE.

[5] S. Xu, "Services replication scheme research under the network congestion," Disseration of Master Degree, Tsinghua University, 2007.

[6] X. Li, "Stateful grid services replication mechansim based on the virtual workspace," Dissertation of Master Degree, Liaoning Univeristy, 2008.

[7] Y. Shen, Y. Huang, S, Xie, "Research and implementation on Web Service replication middleware," Journal of Information Engineering Univesity, Vol. 10, No.4, pp. 546-549, 2009.

[8] L. Zheng, S. Wu, "An infrastructure for web services migration in clouds," 2010 International Conference on Computer Application and System Modeling (ICCASM), 2010, V10-554-556, IEEE.