Research Journal of Applied Sciences, Engineering and Technology 13(8): 632-637, 2016
DOI:10.19026/rjaset.13.3049
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: July 19, 2015 Accepted: August 20, 2015 Published: October 15, 2016

Research Article

Agent Based Load Balancing Mechanisms in Federated Cloud

[1]C.S. Rajarajeswari and [2]M. Aramudhan
[1]Research Scholar, Bharathiyar University, Coimbatore, Tamil Nadu, India
[2]Department of IT, PKIET, Karaikal, India

Abstract: Cloud computing is one of the recent innovation in the field of information technology, which provides services to user on demand and pay per utilization. Single cloud based service model lacks in performance factors like response time, throughput and deadline missing etc., when workload becomes heavy. To overcome this limitation, federated cloud management broker architecture was proposed. Since cloud traffic is unpredictable and busty in nature, there is a possibility of large number of incoming service requests for processing. Hence the workload varies dynamically, some service providers are overloaded and others may be under loaded. In order to balance this situation, to improve the performance of federated cloud broker architecture, load balancing techniques are incorporated at the place of Broker Manager and brokers. Broker Manager (BM) plays a vital role to select appropriate best broker for computing the incoming service requests. Agent based Round Robin Load Balancing Scheduling (ARRS) is proposed at BM for assigning the service requests among the selected brokers by considering the parameters such as workload and queue size of brokers. Another one called Decentralized Agent based Load Balancing (DALB) technique is proposed at the level of brokers to balance the assigned workload in the way of migrating the requests to the under loaded brokers. The result shows that the proposed load balance based broker architecture provides better performance compared to without load balancing based architecture.

Keywords: Broker, broker manager, federated cloud, load balancing, load sharing, migration

INTRODUCTION

Cloud computing provides a wide range of cost effective, dynamic services to users based on the demand through internet. Even though the Service Level Agreement (SLA) is made between user and provider based on functional and non-functional parameters that promote Quality of Service (QoS) (Armbrust *et al*., 2009) due to the nature of internet traffic, extending QoS is a challenging task in cloud environment (Rajarajeswari and Aramudhan, 2014a). In order to maintain the QoS, various changes have been carried out at the level of architecture.

Cloud computing performance depends on the scheduling and proper load balancing algorithms (Kunamneni, 2012). Scheduling algorithms provide a sequence of proper resource allocation in turn throughputs are increased (Randles *et al*., 2010), whereas load balancing algorithm divides the workload between the available resources. Load balancing mechanism is needed to distribute equal workloads to the cloud service providers to achieve optimum outset. Load balancing strategies may be either static or dynamic. Static strategies use information about the average performance of the system and the transfer decisions are independent of the current system state. Dynamic strategies use system state information to make load distribution decisions. Due to uneven job arrival patterns and unequal computing capabilities, some cloud service providers may be overloaded while others may be underutilized. Hence, load balancing is required to redistribute the workload among the available resources in order to achieve optimal resource utilization, maximize throughput, minimum response time and avoid overload.

In federated cloud model, load balancing techniques are used to balance the workloads of cloud service providers by distributing the workload among the brokers. Almost all the load balancing techniques are centralized decision making for forwarding the requests for execution in the federated cloud. The architecture based on the centralized decision making leads to congestion in addition to single point of failure in the federated cloud. The federated cloud architecture proposed by the authors in Rajarajeswari and Aramudhan (2014b) suggested the need of load balancing at the level of BM and brokers to promote QoS. To address these issues two load balancing algorithms namely Agent based Round Robin Load Balancing Scheduling (ARRS) and Decentralized Agent based Load Balancing (DALB) techniques are proposed in this study to maintain the QoS.

Corresponding Author: C.S. Rajarajeswari, Research Scholar, Bharathiyar University, Coimbatore, Tamil Nadu, India

ARRS sits at dispatcher in BM and collects the workload of brokers. Based on the information service requests will be distributed among the selected broker in round robin fashion. DALB is a decentralized load balancing technique that will be invoked by broker. By analyzing the workload of other brokers, perform service request migration.

LITERATURE REVIEW

In algorithms (Xu and Huang, 2009; Rimal *et al*., 2009) the processes are divided between all processors. Each process is assigned to the processor in a round robin order. The process allocation order is maintained locally independent of the allocations from remote processors. Though the work load distributions between processors are equal but the job processing time for different processes are not same. So at any point of time some nodes may be heavily loaded and others remain idle. This algorithm is mostly used in web servers where http requests are of similar nature and distributed equally.

Load balancing algorithm (Werstein *et al*., 2006) can also be based on least connection mechanism which is a part of dynamic scheduling algorithm. It needs to count the number of connections for each server dynamically to estimate the load. The load balancer records the connection number of each server. The number of connection increases when a new connection is dispatched to it and decreases the number when connection finishes or timeout happens.

Equally spread current execution algorithm (Nitika *et al*., 2012) process handle with priorities. It distribute the load randomly by checking the size and transfer the load to that virtual machine which is lightly loaded or handle that task easy and take less time and give maximize throughput. It is spread spectrum technique in which the load balancer spread the load of the job in hand into multiple virtual machines.

Throttled algorithm (Nitika *et al*., 2012) is completely based on virtual machine. In this client first requesting the load balancer to check the right virtual machine which access that load easily and perform the operations which is give by the client or user. In this algorithm the client first requests the load balancer to find a suitable Virtual Machine to perform the required operation.

Fang *et al*. (2010) and Buyya *et al*. (2010) discussed a two-level task scheduling mechanism based on load balancing to meet dynamic requirements of users and obtain high resource utilization. It achieves load balancing by first mapping tasks to virtual machines and then virtual machines to host resources thereby improving the task response time, resource utilization and overall performance of the cloud computing environment.

Min-Min Algorithm begins with a set of all unassigned tasks. First of all, minimum completion time for all tasks is found. Then among these minimum times the minimum value is selected which is the minimum time among all the tasks on any resources. Then according to that minimum time, the task is scheduled on the corresponding machine. Then the execution time for all other tasks is updated on that machine by adding the execution time of the assigned task to the execution times of other tasks on that machine and assigned task is removed from the list of the tasks that are to be assigned to the machines. Then again the same procedure is followed until all the tasks are assigned on the resources. But this approach has a major drawback that it can lead to starvation (Ray and Sarkar, 2012).

Max-Min is almost same as the min-min algorithm except the following: after finding out minimum execution times, the maximum value is selected which is the maximum time among all the tasks on any resources. Then according to that maximum time, the task is scheduled on the corresponding machine. Then the execution time for all other tasks is updated on that machine by adding the execution time of the assigned task to the execution times of other tasks on that machine and assigned task is removed from the list of the tasks that are to be assigned to the machines (Kokilavani and Amalarethinam, 2011).

Historical based service submission records (Sotiriadis *et al*., 2012) explored the performance of an Inter-Cloud to measure the utilization levels among their sub-clouds for various job submissions. The experimental analysis shows that based on basic historical records, an Inter-Cloud can decide the number of datacenters to be utilized based on the number of jobs submitted to the system. A future direction is to incorporate cloud datacenters and allow tasks to be migrated between different hosts belonging to various datacenters.

LOAD BALANCING TECHNIQUE FOR FEDERATED CLOUD BROKER ARCHITECTURE

At the start, users submit request to Broker Manager (BM) (Rajarajeswari and Aramudhan, 2014a). Dispatcher component is used to schedule the incoming requests of Broker Manager. Since the cloud traffic is dynamic and unpredictable, dispatcher disseminates the incoming workload to the brokers with the help of Broker Monitoring Agent (BMA).

Broker Monitoring Agent (BMA): BMA monitors the workload of all the brokers and inform the status to broker manager. BMA also maintains a BLI (Broker Load Index) table which records the information such as broker-id, length of jobs in a waiting queue, length of jobs in service etc. for each broker. At each time t, BMA counts the number of request in the queue and update into the load index table.

Dispatcher: Dispatcher component uses this BLI table to schedule the incoming jobs. Dispatcher decision is

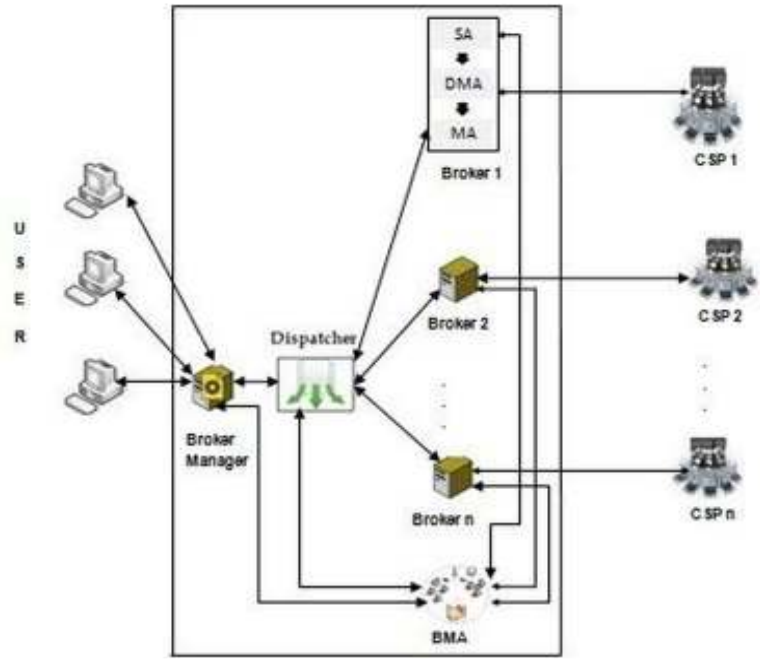

Fig. 1: Functional architecture of federated cloud

based on the total time needed for the completion of the previous assigned request in the brokers. Dispatcher component is used to allocate or reallocate the incoming user request to the selected broker.

The functional model of the federated cloud broker architecture is shown in Fig. 1.

Workload of a broker is defined as the number of service requests assigned to the specific broker by the broker manager and time taken to execute all the assigned requests. To balance the workload, two load balancing techniques are proposed at the level of broker manager and brokers. The proposed load balancing techniques are given below:

- Agent based Round Robin Load Balancing Scheduling Algorithm (ARRS)
- Decentralized Agent based Load Balancing Algorithm (DALB)

Agent based Round Robin Load Balancing scheduling is similar to round robin scheduling of CPU that assigns the service requests among the selected brokers. To balance the workload of brokers, a Decentralized Agent based Load Balancing (DALB) is proposed at the level of brokers.

In the proposed federated cloud broker model, service requests that are arrived at the broker manager are forwarded to the dispatcher. The functionality of the dispatcher is to redistribute the incoming service request among the selected broker for the task. The selected brokers are ranked by using Poincare plot method (Sotiriadis *et al*., 2012). Each cloud service provider is managed with a broker. Broker Monitoring Agent (BMA) in the proposed architecture is used to monitor the workload of brokers, inform periodically to the dispatcher about the workload status of brokers.

PROPOSED LOAD BALANCING MODEL

To balance the workload of the federated cloud broker environment two load balancing techniques are proposed at the level of broker manager and brokers. The load balancing techniques are:

- Agent based Round Robin Load Balancing Scheduling Algorithm (ARRS)
- Decentralized Agent based Load Balancing Algorithm (DALB)

Agent based Round Robin Scheduling algorithm (ARRS): BM assigns unique ID for each service request and selects broker for computing the task. It is working on the principle of centralized decision making technique. Generally BM chooses the top broker for executing the incoming request (Rajarajeswari and Aramudhan, 2014b).This leads to congestion at the level of brokers. This congestion can be avoided with the help of ARRS load balancing technique.

To balance the workload at BM, a round robin scheduling is used as a load balancing technique to assign the service requests equally among the selected brokers. ARRS sits at dispatcher, to distribute the service workload among the selected brokers by BM. ARRS performs among the selected brokers, identify least workload and assign the task based on Poincare plot method and compatibility decision matrix (Rajarajeswari and Aramudhan, 2014a) provided by BM.

The function of BM in case of failure of the dispatcher is identified as follows. If the dispatcher fails, BM can act as a dispatcher for short time and inform the status of failure to the cloud administrator. BMA informs the status of brokers to the dispatcher. If BMA fails to receive any response from dispatcher before exceed of timer value, re-inform the status to the broker manager. Timer value is twice the propagation delay between the dispatcher and the BMA.

Consider simple scenario; there are 5 brokers namely B1, B2, B3, B4 and B5 in the federated cloud broker architecture. B2, B4 and B5 are selected as matched brokers for the incoming service requests and store in the compatibility decision matrix. B2, B4 and B5 are ranked using Poincare plot method. Dispatcher verifies the workload of the ranked brokers and assigns the service request to the selected broker using round robin technique.

Decentralized Agent based Load Balancing technique (DALB): This technique is performed at the level of brokers, by considering the service completion time. Service may be migrated among the selected brokers that execute the service earlier. In decentralized load balancing technique, each selected broker has making decision, either to execute the request or transfer to any under loaded brokers. Each broker provides three agents namely Stationary Agent (SA), Decision Making Agent (DA) and Migration Agent (MA).

Stationary agent sits permanently at the broker, monitors the workload of the broker and informs the status of workload to the decision making agent. Decision making agent collects the workload information of other brokers through BMA and decide any service request in the queue that may have possible transfer to other under loaded broker by considering the value of the performance index.

Performance Index (PI) is calculated as the difference between the time taken by the request to execute in the broker assigned by the dispatcher and the time taken for request transfer, waiting time and execution time of the request in other under loaded brokers. Performance index consists of two values +1 and -1.

If the time difference between assigned broker is less than the under loaded broker then the performance index value is +1, otherwise the performance index value is -1. If the performance index value is -1 then decision making agent transfers the service request to the under loaded broker, otherwise it will be executed in the broker assigned by the dispatcher.

Migration agent is invoked by the decision making agent, to transfer service request from the queue to other under loaded broker based on the performance index value.

Let A, B, C be the brokers that are connected with different cloud service providers. Let $queuelength_A$, $queuelength_B$, $queuelength_C$ be the queue size of the brokers respectively at a given point of time. The brokers A, B, C collect the information from Broker Monitoring Agent (BMA). The broker which transfers the services to other brokers is considered as source and the broker where the services are received and manipulated is considered as the destination. Job re-direction between brokers is based on the following algorithm.

Step 1: If $queuelength_{Desiination} \succ (queuelength_{Source1}$ and $queuelength_{Source2}$ and… $queuelength_{Sourcen})$ then

Step 2: Compute q_x such that q_x = min ($queuelength_{Source1}$, $queuelength_{Source2}$ … $queuelength_{Sourcen}$)

Step 3: Compute n such that n = ($queuelength_{Destination}$-$q_x$)/2

Step 4: Transfer the last n jobs from $queuelength_{Destination}$ to q_x broker, if it holds condition 1.

A job 'j' on broker 'x' is reallocated to a broker 'y' only when:

$$\sum_{i=1}^{j} p_{ix} \succ j\_rt_x + j\_rp_{xy} + \sum_{i=n}^{i=1} p_{iy} + p_{jy} \quad (1)$$

where,

P_{ix} = Processing Time of i^{th} request at broker x
j_rt_x = Transmission Time of j^{th} request by broker x
j_rp_{xy} = Propagation Time of j^{th} request from broker x to broker y
p_{iy} = Processing Time of i^{th} request at broker y
p_{jy} = Processing Time of j^{th} request at broker y

Algorithm 1: Decentralized agent based load balancing: The purpose of computing n is to redistribute the jobs in order.

EXPERIMENTAL RESULTS AND DISCUSSION

The proposed framework is implemented in Cloudsim (Calheiros *et al*., 2009; Barrett *et al*., 2011) using Java. A software simulator was designed and implemented to model the DALB load balancing technique in the federated cloud broker environment.

The workload of a broker is determined by the number of requests processed at each broker. DALB is applied to minimize the workload difference between the brokers. The performance of DALB load balancing scheme is evaluated and compared with without load balancing scheme.

Table 1 compares the load distribution generated by the DALB scheme and without load balancing scheme on three brokers at different moment. It also includes the average deviation of load on the three

Table 1: Load distribution on three brokers

Load Balancing using DALB	Total no. of requests	Brokers B1	B2	B3	Average deviation	Overall average deviation
	100	39	21	40	8.22	
	200	103	59	38	24.22	
	300	93	66	141	27.33	
	400	140	127	133	4.44	
	500	195	125	180	24.44	
	600	280	179	141	53.33	27.64
	700	240	196	264	24.89	
	800	292	233	275	22.44	
	900	271	309	320	19.33	
	1000	255	310	435	67.77	
Without load Balancing	Total no. of requests	Brokers B1	B2	B3	Average deviation	Overall average deviation
	100	48	11	41	14.89	
	200	140	55	5	48.88	
	300	75	65	160	40	
	400	150	115	135	12.22	
	500	205	91	204	50.44	
	600	303	144	153	68.66	54.81
	700	215	99	286	89.55	
	800	302	188	310	52.44	
	900	215	322	363	36.66	
	1000	156	309	535	34.44	

brokers. It shows that DALB scheme has lower deviation than without load balancing scheme in most of the cases. That means DALB can distribute user request more evenly to the cloud service providers.

Overall average deviation in Table 1 is the mean of the average deviation of all moments. The overall average deviation of the DALB scheme is lower than without load balancing scheme strategy. Overall average deviation of DALB is twice lesser than the existing without load balancing technique in federated cloud management system.

Proposed load balancing algorithm executes the incoming request based on the current load of cloud service providers with the help of the agents in each broker.

CONCLUSION

This study examines two new algorithms for improving the performance of the federated cloud broker architecture through load balancing approach. ARRS algorithm distributes the service workload among the selected brokers by broker manager. It is working on the principle of centralized decision making technique. But load imbalance is still persists in the architecture. Proposed DALB algorithm has several advantages. First, decision making in brokers is decentralized and improves the response time. Second, use of Stationary agent, Decision Making Agent and Migration Agent provides high flexibility in the migration process. Third, the result shows that no providers remain idle at any time while other providers are processing more requests. Thus the proposed DALB framework proves that the effective distribution of workload among the brokers in the federated cloud broker environment is achieved.

REFERENCES

Armbrust, M., A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica and M. Zaharia, 2009. Above the clouds: A berkeley view of cloud computing. Technical Report No. UCB/EECS-2009-28, University of California at Berkeley, pp: 1-25.

Barrett, E., E. Howley and J. Duggan, 2011. A learning architecture for scheduling workflow applications in the cloud. Proceeding of the 9th IEEE European Conference on Web Services, pp: 83-90.

Buyya, R., R. Ranjan and R.N. Calheiros, 2010. InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services. Proceeding of the 10th International Conference on Algorithms and Architectures for Parallel Processing, pp: 13-31.

Calheiros, R.N., R. Ranjan, C.A.F. De Rose and R. Buyya, 2009. CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services. Technical Report No. GRIDS-TR-2009-1, Grid Computing and Distributed Systems Laboratory, the University of Melbourne, Australia.

Fang, Y., F. Wang and J. Ge, 2010. A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing. In: Wang, F.L. *et al*. (Eds.), Web Information Systems and Mining. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 6318: 271-277.

Kokilavani, T. and D.I.G. Amalarethinam, 2011. Load balanced min-min algorithm for static meta-task scheduling in grid computing. Int. J. Comput. Appl., 20(2): 43-49.

Kunamneni, V., 2012. Dynamic load balancing for the cloud. Int. J. Comput. Sci. Electr. Eng., 2315: 33-37.

Nitika, M., G. Shweta and G. Raj, 2012. Comparative analysis of load balancing algorithms in cloud computing. Int. J. Adv. Res. Comput. Eng. Technol., 1(3): 34-38.

Rajarajeswari, C.S. and M. Aramudhan, 2014a. Differentiated services at application level for SLA resource provisioning management. Proceeding of the International Conference on Mathematical Sciences, pp: 639-642.

Rajarajeswari, C.S. and M. Aramudhan, 2014b. Ranking model for SLA resource provisioning management. Int. J. Cloud Appl. Comput., 4(3): 68-80.

Randles, M., D. Lamb and A. Taleb-Bendiab, 2010. A comparative study into distributed load balancing algorithms for cloud computing. Proceeding of the IEEE 24th International Conference on Advanced Information Networking and Applications and Workshops, pp: 551-556.

Ray, S. and A.D. Sarkar, 2012. Execution analysis of load balancing algorithms in cloud computing environment. Int. J. Cloud Comput. Serv. Archit., 2(5): 1-13.

Rimal, B.P., E. Choi and I. Lumb, 2009. A taxonomy and survey of cloud computing systems. Proceeding of the 5th International Joint Conference on INC, IMS and IDC, pp: 44-51.

Sotiriadis, S., N. Bessis and N. Antonopoulos, 2012. Exploring inter-cloud load balancing by utilizing historical service submission records. Int. J. Distrib. Syst. Technol., 3(3): 72-81.

Werstein, P., H. Situ and Z. Huang, 2006. Load balancing in a cluster computer. Proceeding of the 7th International Conference on Parallel and Distributed Computing, Applications and Technologies, pp: 569-577.

Xu, Z. and R. Huang, 2009. Performance study of load balancing algorithms in distributed web server systems. CS213 Parallel and Distributed Processing Project Report.