DRSQ - A Dynamic Resource Service Quality Based Load Balancing Algorithm

Anindita Sarkar[1(✉)], Kshitij Pant[2], and Samiran Chattopadhyay[2]

[1] School of Mobile Computing and Communication, Jadavpur University, Kolkata, India
sarkar.anindita5@gmail.com

[2] Department of Information Technology, Jadavpur University, Kolkata, India
kshitijjpant@gmail.com, samirancju@gmail.com

Abstract. In cloud computing domain, the main problem faced by cloud manager is to handle the huge amount of clients request at a single amount. For this reason, there have different types of load balancing algorithms, some are static algorithm and some are dynamic algorithm. But till, when the question is arrived to maintain service quality with the handling of work load balance than the shortage is occurred. In this paper, we proposed a novel dynamic load balancing algorithm named as Dynamic Resource Service Quality Based Load Balancing Algorithm (DRSQ). Besides supporting the load balance, it's main aim is to continue with a good performance rate by utilizing resources in proper way. To reach the goal, back-end resource selection is done based on the provided service quality by the resources. The service quality is measured by considering the request type (e.g., READ, WRITE) and the value set of resource metrics. To prove the usefulness, it goes through some experiments using some real temperature sensor generated datasets in private cloud environment. The most famous load balancing algorithm Round-Robin is considered for comparative analysis.

Keywords: Dynamic load balancing algorithm · Resource service quality · Private cloud · Resource monitoring · Resource metrics

1 Introduction

Cloud computing has recently emerged as a new pattern of hosting and delivering services over the Internet. It works as distributed computing pattern. Now-a-days, cloud computing becomes so popular that applications of different domains (e.g., health, industry, city, home etc.) desire to take facility of it, pointed in Fig. 1. According to Fig. 1, there have two zones. Data zone holds different application domains which are responsible to generate dataset and cloud zone holds resource pool which consists of resources. Here, each application of a single domain produces a large amount of dataset with a varied frequency rate

J. K. Mandal et al. (Eds.): CICBA 2018, CCIS 1031, pp. 97–108, 2019.
https://doi.org/10.1007/978-981-13-8581-0_8

in concurrent manner. It creates a huge challenge to the cloud developer for handling such a vast data traffic. To solve this matter, load balancer [11] acts in between the applications and cloud resource pool [3].

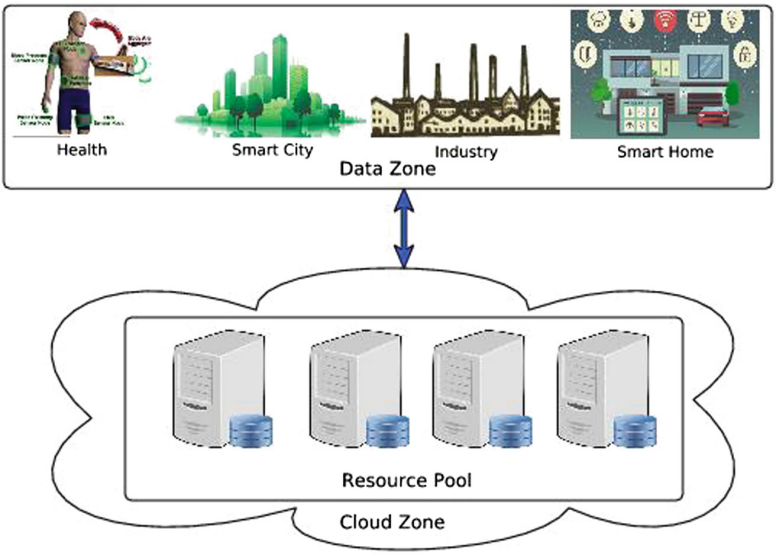

Fig. 1. Overview of cloud utilization by applications

Load balancer manipulates the feature of cloud computing and distributes the data traffics in between the resources of resource pool. The main task of the Load balancer is to ensure that all the resources of the resource pool never be overloaded and supports three factors of cloud computing: availability, reliability and flexibility [4,12]. Here, the question is how load balancer achieve this goal without any data loss. Therefore, load balancer primarily focus upon the resource scheduling tasks, done by load balancing algorithms [7].

There have two types of load balancing algorithm: static load balancing algorithm [2,19] and dynamic load balancing algorithm [20]. The difference between static and dynamic load balancing algorithm is at the time resource scheduling static load balancing algorithm does not consider present status of resources for resource selection but dynamic load balancing algorithm considers it. [18] This resource selection approach makes dynamic load balancing algorithm better by increasing decision accuracy rate higher. [21] Static load balancing algorithm are more stable than dynamic load balancing algorithm The example of the dynamic load balancing algorithms are ant colony [15], honey bee [10] etc. Round robin [9], Random [1,17], Threshold [16] are the example of static load balancing algorithm.

In this paper, we propose a novel dynamic load balancing algorithm named as dynamic resource service quality based load balancing algorithm. It helps load balancer to support the diverse frequency rate of application's requests at

database level. To achieve this goal, load balancer selects the proper resource from resource pool based on the provided data service quality. To judge the service quality, we consider present status of certain resource metrics. They are CPU CLOCK SPEED, LOAD in last 15 min, number of processes, number of running processes, Number of CPU CORES, free RAM size, free SWAP memory size, free DISK size, BYTES IN and BYTES OUT. The contributions of this paper are,

- Without considering a single one, a set of resource metrics are assessed to understand the machine capability against of work load handling. In such way a proper resource is selected for a particular task. This concept helps to utilize resources in better way.
- This algorithm avoids the usage of back-end resources which are in dead state or in bottleneck situation.

The rest of the paper is organized in such way. The background study of different load balancing algorithm in cloud domain is explained in Sect. 2. Section 3 represents the structure of DRSQ algorithm in brief way with resource metrics. The comparative study between Round-Robin algorithm and DRSQ is described in Sect. 4 with experimental setup. Section 5 concludes this paper with mentioning the extension of this work in near future.

2 Related Work

Load balancing is a load allocation strategy which distributes the incoming load to the back-end servers equally, and eliminates any load imbalance problems, thus optimizes the overall performance of the system. There are two type of Load balancing algorithms [3]: static load balancing [19] and dynamic load balancing [1].

Static load balancing [12,16] is also called state independent balancing. It select a back-end server before the incoming request arrives, i.e. the algorithm does not consider the real time load status of the back-end server while assign the request, but instead makes the decisions and assign the request to the back-end server on the basis of known system static information. The advantages of static load balancing are that the simple to implement, incoming request can be quickly allocated to the back-end server. But the main disadvantage of static load balancing algorithm is that it does not consider the real time load of the back-end server, due to which the load allocation to the back-end server would be uneven distributed [18].

Dynamic load balancing [18] select a back-end server, by evaluating the real time load of each back-end server, the incoming request are allocated to the back-end server dynamically.

Active monitoring load balancer algorithm [13] proposed by Hemant S. Mahalle, Parag R. Kaveri and Vinay Chavan, distributes the incoming request between the back-end servers or virtual machines. When ever the load balancer gets an incoming request, it forwards the request to the least loaded Virtual Machine.

Modified throttled algorithm [6] was proposed by Shridhar G. Domanal and G. Ram Mohana Reddy is a dynamic algorithm, which distributes the incoming request between the back-end servers or virtual machines. Cloud Analyst simulator was used to check the result of the algorithm.

The main advantages of dynamic load balancing is that it can select the back-end server on the basis of real time load of the back-end server. Thus the back-end server selection can readjust its selection on the basis of load in the back-end server. The disadvantage of Dynamic load balancing algorithm is that the algorithm frequently collect the status of back-end server to select a back-end server to handle the incoming request which accounts to network overhead, which result in wastage of network bandwidth [18].

3 Overview of Workload Balance

3.1 Resource Performance Metric

In heterogeneous paradigm, each and every resources have different service quality. We use this knowledge in the dynamic resource service quality based load balancing algorithm to design it. A set of metrics are used to measure the performance of each resource in resource pool. The values of these metrics are changed frequently. On the basis of a single parameter at a unit time, not will be a good decision to judge a resource service quality.

We divide the set of resource performance metrics into four dimensions mentioned in Fig. 2: Memory, Processor, Load and Input/Output. A subset of metrics which are in memory dimensions represents the information about cache memory, swap memory, disk and memory itself. Processor dimensions represents the information about CPU and running processor number. Input/output and Load dimensions informed about direct impact with outsider. Therefore, each and every dimension have an role at the selection of best resource at a particular time.

In DRSQ, we use this concept by considering free RAM size, free SWAP memory size, free DISK size from memory dimension, number of processes, number of running processes, Number of CPU CORES, CPU CLOCK SPEED from Processor dimension, LOAD in last 15 min from Load dimension and BYTES IN and BYTES OUT from Input/Output dimension. These metrics act as a group and eligible to find out a proper resource at that time. This resource comparatively served better performance than others at that time.

3.2 Load Balancing Algorithm

The load balancer run the load balancing algorithm to reduce the gap between application generated concurrent dataset and the limited number of resources of resource pool by distributing the workload in between the resources. Performance of the load balancing algorithm is somehow depends upon the proper resource selection. Based on this concept we implement dynamic resource service quality

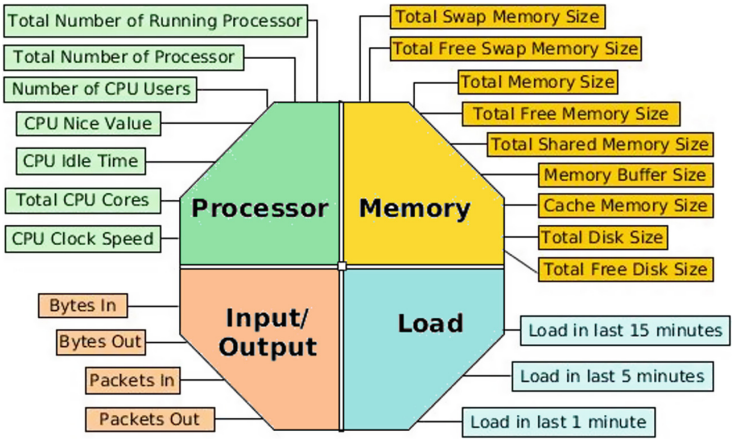

Fig. 2. Dimensions of resource performance metrics

based load balancing algorithm where resource is selected using performance metrics value-set.

DRSQ works under three phases: (a) Decision Making about the type of application requests (b) Configuration of resource list and (c) calculate the ultimate resource. The detailed algorithmic view of the phases are mentioned in algorithm 1. The working procedure of each phase is given in below.

- In first phase, Our algorithm will wait for the incoming request from the client. Once the request arrive at the load balancer, it will register the request. After the request is register, it will check the type of operation which need to be perform i.e. it is a WRITE query or a READ query. The main objective of this module is to check the type of query which is required to select the parameters on the basis of which we will select a back-end server.
- After knowing the type of operation in the Decision Making module, we will fetch the information of the back-end server depending on the type of operation in second phase. If incoming request will perform a WRITE operation, the algorithm will take into consideration the free primary memory, free swap memory, free secondary memory, bandwidth utilization and load in last fifteen minutes each active back-end server. Otherwise the incoming request will perform READ operation in which we will consider the CPU clock speed, Total processes in the server, Total number of running processes in the server, CPU cores, bandwidth utilization and load on the server in last fifteen minutes of each active back-end server. The algorithm will fetch these values and store it in system informations LIST for each back-end server. We have used Ganglia Monitoring Tool to fetch the system information from the back-end servers. This system informations LIST for each back-end server is forwarded to the Computing related jobs based on LIST components module for further processing.

Algorithm 1. Algorithmic view of dynamic resource service quality based load balancing algorithm

Input: Application Request (READ/WRITE) operation
Output: Backend resource which will handle the request
Phase 1: Decision making about the type of request
Phase 1.1: *Wait for the incoming request*
Phase 1.2: *Register New Request*
Phase 1.3: *Decide the type of request either it can be READ or WRITE*
Phase 2: Configuration of Resource LIST
Phase 2.1: *If the operation to be performed is READ*
Phase 2.1.1: Select and store a back-end resource with maximum CLOCK_SPEED,minimum LOAD in last 15 minutes, minimum number of processes, minimum number of RUNNING processes, maximum Number of CPU CORES respectively in a LIST
Phase 2.1.2: Append the Server name in the LIST till all the parameters are checked
Phase 2.2: *If the operation to be performed is WRITE*
Phase 2.2.1: Select and store a back-end resource with maximum FREE RAM, maximum FREE SWAP, maximum FREE DISK, minimum LOAD in last 15 minutes respectively in a LIST
Phase 2.2.2: Append the resource in the LIST till all the parameters are checked.
Phase 3: Computing ultimate resource based on LIST components
Phase 3.1: *Select the resource which has maximum occurrence in LIST*
Phase 3.2: *Forward the request to the selected resource*

– In third phase, we will compare each field of the system informations LIST which is created in the previous module (containing system informations) of each back-end server. The most suitable value of each field is taken into consideration and the name of that server is stored in a newly created resource LIST. After we have compared all the fields of the system informations LIST, we will check which server has dominated the resource LIST. After getting the name of that server, We will forward the registered incoming request to the selected server.

4 Experiment

4.1 Setup

The main aim of the load balancing algorithm is to control the network traffic. We also try to achieve this goal by using proposed Dynamic Resource Service Quality based Load Balancing Algorithm. Figure 3 describes the experimental system setup. This setup is divided into two parts, *Data Zone* responsible to client request and *Cloud Zone* presents the information about resources and load balancer. Here, four application domain is placed, each of them consists of temperature sensors. *Load Balancer* runs the DRSQ to select the resource

for handling network traffic generated by the clients requests. According to this setup client's request is based on database related like read or write.

The efficiency of DRSQ is measured by the varied incoming data frequency rate means increasing and decreasing the network data traffic. We use real temperature sensor data to do the experiment. Here, total 6 temperature sensor is used, the data generation frequency rate of each sensor is varied. According to the setup in Fig. 3, at minimum range each application domain consists of a single temperature sensor and at maximum range each application domain holds two temperature sensors.

To do the experiment, five virtual machines and one load balancer machine are considered. MongoDB [8], Apache Tomcat [5] and Ganglia [14] are used to configure the machines according to their needs presented in Fig. 3. Ubuntu 14.04 is used as operating system of these machines. To collect the resource metrics value set corresponding of each machine, Ganglia plug-ins are used like, Gmetad 3.6.0 in Load balancer and Gmond 3.6.0 in resource part. MongoDB 3.6.19 is used as back-end database to put the incoming dataset. The used java based web applications are run on tomcat server 8.5.28.

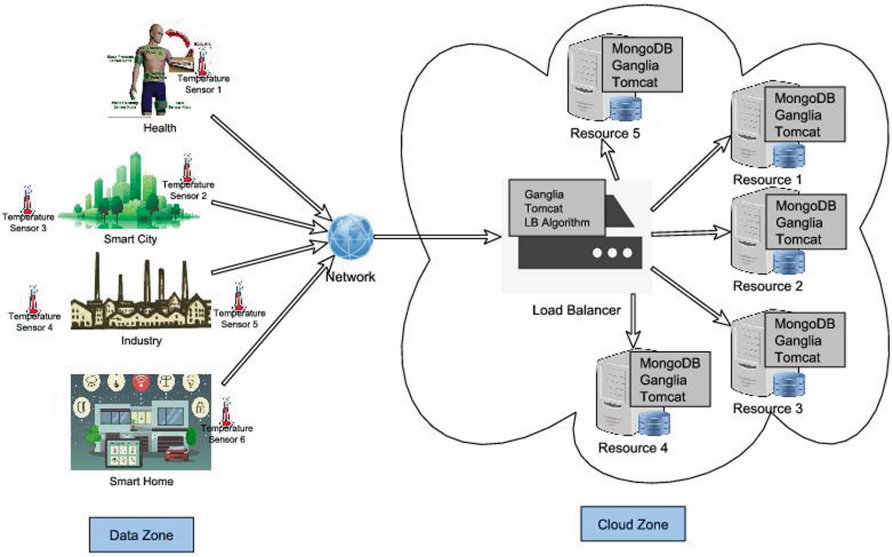

Fig. 3. Visualization of experimental system setup

4.2 Illustration

We choose Round-Robin load balancing algorithm for doing the comparative study because it is considered as standard load balancing algorithm by the load

Table 1. Performance result-set of Dynamic resource service quality and Round-Robin load balancing algorithm

Algorithm	Execution time	CPU utilization	Memory utilization
Dynamic resource service quality based load balancing algorithm	1.98 s	0.451%	6 MB
Round-Robin load balancing algorithm	0.401 s	0.028%	5 MB

balancer designer in each and every application domain. If we compare the overall performance (i.e., CPU Utilization, Memory Utilization and Execution Time) between Round-Robin and DRSQ than we find out Round-Robin algorithm perform in all aspects, shown in Table 1.

To analyze the performance behavior in streaming data, we consider 5 data request from each sensor of applications (Fig. 3) which sends data request in 5 KB/s, 10 KB/s, 15 KB/s, 20 KB/s, 25 KB/s and 30 KB/s. We notice that in Fig. 4, when the data frequency rate is changed after few minute or hour than the response time pick is changed in DRSQ algorithm otherwise it maintains a line. Figure 5 represent characteristics of response times of each request under different frequency rates in more elaborate way. Here, we can see that, in each and every frequency rate only for 'Request 1' (Fig. 5(a)) DRSQ algorithm takes more time than Round-Robin and in other requests (Fig. 5[(b), (c), (d), (e)]) it takes less time.

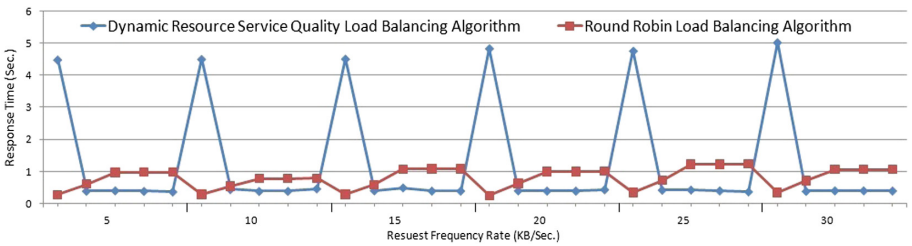

Fig. 4. Pattern of response times for streaming data requests

Dynamic Resource Service Quality algorithm selects the resource based on the system monitoring status value collected from ganglia generated log files. Initially the algorithm invests time to trigger the ganglia monitoring daemons to monitor the resources, connect with the back-end servers, fetch the log files generated by ganglia daemons and finally parse the desired values from the log files. When the next request is received by load balancer after few micro or nano second, it just modify the generated log file contents, analyzed the new value and

send the incoming request to the selected resource. Hence reducing the ganglia triggering up time which diminishes the response time.

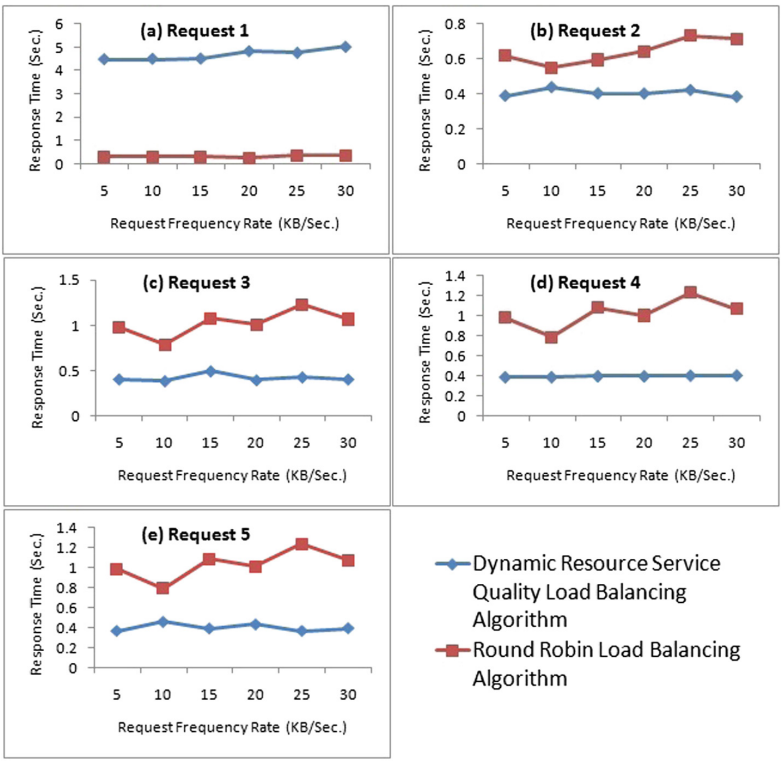

Fig. 5. Characteristics of each requests under different frequency rates

On the other end, Round-Robin algorithm selects the resources in circular way, one after another. In such situation, there are large chances that a crashed server or a heavily loaded server is selected as there will be many requests pending with it all the requests may cause a bottleneck situation at the back-end server which upsurge the response time. Therefore, some times response time is very low and some times it is very high. For the DRSQ, there is no chance to select a resource which is in bottleneck condition. For this reason, except first stage, the generated response times show the more or less similar value.

Also we consider 'Task Accuracy Rate' to justify the usefulness of DRSQ Load Balancing algorithm. For doing this experiment, we changed our setup as shown in Fig. 3 little bit where Resource 5 is in dead condition. Here, we made 10,000 'Write' requests continuously with a data bundle consists of 1000 data records in each request. According to the working principle of Round-Robin algorithm, Resource 5 was requested 2,000 times and every times it failed to store the data.

Whereas DRSQ does not select Resource 5 at a single time because before making request it checks the resource is in workable state or not. If we assume 1% packet loss is occurred by network issues than the computed accuracy percentage for both algorithms are shown in Fig. 6. For DRSQ, data loss is occurred only by the network issues but for Round-Robin algorithm inaccuracy rate is calculated by data loss in network faults and selection of crashed resources.

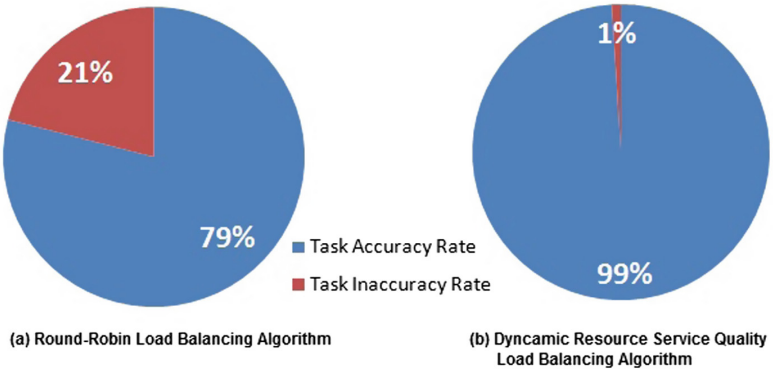

Fig. 6. Task accuracy rate in percentile for (a) Round-Robin and (b) Dynamic resource service quality load balancing algorithm in mentioned scenario

We come to the point that If we see the overall performance than Round-Robin algorithm provides better performance. At the respect of application domain, usage as load balancing algorithm by load balancer some confusions are occurred In task accuracy rate or performance rate for streaming data, it does not provide satisfactory performance. Whereas, in such situations DRSQ provides good performance.

5 Conclusions and Future Work

In this paper, we present a novel dynamic load balancing algorithm DRSQ which not only balance the work load and also provides an efficient application services. This algorithm makes the resource selection based on the system monitoring status. So, it omits the possibility to select a resource which is in dead state or in bottleneck situation.

In our future work, we will extend our algorithm with further optimization features. Firstly, it is important that we reduce the decision making time to select the back-end resource as it is a dynamic algorithm. To achieve this feature we need to integrate some optimization algorithms with our existing algorithm. Secondly, we wish to segregate the incoming request on the type of information rather than type of operation to be performed. To achieve this feature, we need to analyze the incoming data packet and check the type of data it is carrying.

References

1. Alakeel, A.M.: A guide to dynamic load balancing in distributed computer systems. Int. J. Comput. Sci. Netw. Secur. (IJCSNS) **10**, 153–160 (2010)
2. Belkar, S.P., Handur, V.: Comparative study of static load balancing algorithms in distributed system using cloudsim. Int. J. Adv. Res. Basic Eng. Sci. Technol. (IJARBEST) **5**, 26–30 (2017)
3. Casavant, T.L., Kuhl, J.G.: A taxonomy of scheduling in general-purpose distributed computing systems. IEEE Transact. Softw. Eng. **14**, 141–154 (1988)
4. Chang, H., Tang, X.: A load-balance based resource-scheduling algorithm under cloud computing environment. In: Luo, X., Cao, Y., Yang, B., Liu, J., Ye, F. (eds.) ICWL 2010. LNCS, vol. 6537, pp. 85–90. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20539-2_10
5. Chopra, V., Li, S., Genender, J.: Professional Apache Tomcat 6. Wiley, Hoboken (2007)
6. Domanal, S.G., Reddy, G.R.M.: Load balancing in cloud computingusing modified throttled algorithm. In: 2013 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), October 2013
7. Fang, Y., Wang, F., Ge, J.: A task scheduling algorithm based on load balancing in cloud computing. In: Wang, F.L., Gong, Z., Luo, X., Lei, J. (eds.) WISM 2010. LNCS, vol. 6318, pp. 271–277. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16515-3_34
8. Hows, D., Membrey, P., Plugge, E.: MongoDB Basics. Apress, New York (2014)
9. Jain, S., Saxena, A.K.: A survey of load balancing challenges in cloud environment. In: 2016 International Conference System Modeling Advancement in Research Trends (SMART), pp. 291–293 (2016)
10. Kaur, A., Kaur, B.: Load balancing in tasks using honey bee behavior algorithm in cloud computing. In: 2016 5th International Conference on Wireless Networks and Embedded Systems (WECON), pp. 1–5. IEEE (2016)
11. Khan, R.Z., Ahmad, M.O.: Load balancing challenges in cloud computing: a survey. In: Lobiyal, D.K., Mohapatra, D.P., Nagar, A., Sahoo, M.N. (eds.) Proceedings of the International Conference on Signal, Networks, Computing, and Systems. LNEE, vol. 396, pp. 25–32. Springer, New Delhi (2016). https://doi.org/10.1007/978-81-322-3589-7_3
12. Kunz, T.: The influence of different workload descriptions on a heuristic load balancing scheme. IEEE Transact. Softw. Eng. **17**, 725–730 (1991)
13. Mahalle, H.S., Kaveri, P.R., Chavan, V.: Load balancing on cloud data centers. Int. J. Adv. Res. Comput. Sci. Softw. Eng. **3**(1), 1–4 (2013)
14. Massie, M.L., Chun, B.N., Culler, D.E.: The ganglia distributed monitoringsystem: design, implementation, and experience. Parallel Comput. **30**, 817–840 (2004)
15. Nishant, K., Sharma, P., Krishna, V., Gupta, C., Singh, K.P., Nitin, Rastogi, R.: Load balancing of nodes in cloud using ant colony optimization. In: 2012 UKSim 14th International Conference on Computer Modelling and Simulation, pp. 3–8 (2012)
16. Rahmawan, H., Gondokaryono, Y.S.: The simulation of static load balancing algorithms. In: 2009 International Conference on Electrical Engineering and Informatics, pp. 640–645 (2009)
17. Rathore, N.: Dynamic threshold based load balancing algorithms. Wireless Pers. Commun. **91**, 151–185 (2016)

18. Sharma, S., Singh, S., Sharma, M.: Performance analysis of load balancing algorithms. World Acad. Sci. Eng. Technol. **38**, 269–272 (2008)
19. Tantawi, A.N., Towsley, D.: Optimal static load balancing in distributed computer systems. J. ACM **32**, 445–465 (1985)
20. Wang, M., Guan, J.: An adaptive dynamic feedback load balancing algorithm based on QoS in distributed file system. J. Commun. Inf. Netw. **2**, 30–40 (2017)
21. Zhou, L., Wang, Y.C., Zhang, J.L., Wan, J., Ren, Y.J.: Optimize block-level cloud storage system with load-balance strategy. In: 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), pp. 2162–2167. IEEE (2012)