Research Journal of Applied Sciences, Engineering and Technology 13(10): 794-799, 2016
DOI:10.19026/rjaset.13.3355
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2016 Maxwell Scientific Publication Corp.
Submitted: August 1, 2016 Accepted: September 2, 2016 Published: November 15, 2016

Research Article

A Comprehensive Study on Load Balancing Algorithms in Cloud Computing Environments

[1]S. Sankara Narayanan and [2]M. Ramakrishnan
[1]Department of Computer Science and Engineering, Anna University, Chennai,
[2]School of Information Technology, Madurai Kamaraj University, Madurai,
Tamil Nadu, India

Abstract: Aim of this paper is to study the various load balancing algorithms used in cloud computing environments. Cloud computing is a new paradigm and has become a conventional method of computation and communication over the internet. This new technology is attracted well by the internet community. Cloud computing offers various services on demand with nominal charges. With cloud computing, end users are free from installation, maintenance and licensing hurdles. Thus cloud computing has become an important way of disseminating information and providing computational services over the network. Day by day, the volume of information stored and retrieved on clouds is increasing rapidly. With huge number of requests, both storage and retrieval, flowing to the cloud, it is important to balance the load among existing servers. Load balancing gives user satisfaction and allows best utilization of resources. Hence this study provides an overview on the basics of load balancing techniques used in cloud environments. The paper also presents various performance metrics used to evaluate the load balancing algorithm.

Keywords: Cloud computing, load balancing, performance, resource allocation, response time

INTRODUCTION

Cloud computing is a new distributed computing technology which provides convenient, on-demand services over the network. It mix up with virtual techniques to provide services in an efficient way when and where needed. In cloud environment, customers can get the resources like CPU, memory, bandwidth, etc they want with reduced cost and effort (Zhang *et al*., 2010). Another advantage of cloud computing is that it provides services on different platforms, thereby avoiding infrastructure complexities. Cloud computing allows users to access high valued softwares without purchasing the licensed copy (Buyya *et al*., 2009).

Cloud computing works as follows: All the services, applications, infrastructure, etc are kept in cloud. Network devices send their requests to a cloud service provider and retrieve the results in a timely manner without facing complexities like storage, process, security, protocols, service compositions, communication and distributed computation (Badger *et al*., 2011). Thus cloud is a convenient model that enables on-demand network access to a pool of shared and configured resources, resources that can be easily provisioned and released with minimal efforts (Zhang *et al*., 2010).

It is important for cloud service provider to offer best services to the client and also allows them to utilize the resources in an efficient way (Buyya *et al*., 2010). In other words, the response time should be good and assigning resources to the clients should be fair. In this situation, load balancing concept comes into the picture and it is one of the most challenging tasks in cloud computing domain. Since various resources are available in cloud viz. network links, CPUs, applications, disk drives, etc achieving optimal resource allocation and utilization is important and hence load balancing is an important aspect in cloud computing environments. By load balancing method, response time, throughput can be increased and overload of cloud traffic can be managed (Calheiros *et al*., 2011).

LITERATURE REVIEW

Devi and Uthaiaraj (2016) presented a paper on load balancing in cloud computing environments using improved weighted round robin algorithm for non-preemptive dependent tasks. The objective of the work

Corresponding Author: S. Sankara Narayanan, Department of Computer Science and Engineering, Anna University, Chennai, Tamil Nadu, India

is to evaluate the scheduling and load balancing algorithm considering virtual machine capabilities. The proposed algorithm works in three stages. The static scheduler algorithm, dynamic scheduler algorithm and load balancer is used which evenly distributes the load across all the virtual machines.

Domanal and Reddy (2014) presented a work on optimal load balancing in cloud computing by efficient utilization of virtual machines. Here a novel method of load balancing is proposed which allocates the incoming requests to all available virtual machines in an efficient manner. The method is efficient and provides load balancing without under as well as over resource utilization in cloud environments.

Xu *et al*. (2011) presented an intelligent load balancing algorithm for efficient cloud computing. Generally mapreduce is used for complex job decomposition and sub-task management in supporting cloud computing. This classic Mapreduce model extended towards agent-aid layer and abstracting work load requests for data blocks as tokens. Agents know where to send the token thereby balancing the load in cloud and improving the system performance. The token generation model is heuristic and finds optimal providers with agent's partial observation to global load distribution.

Gasior and Seredynski (2012) proposed an load balancing method in cloud computing system through formation of coalitions in a spatially generalized prisoner's dilemma game. This method is highly parallel and distributed which works in environments where local information alone is available. This method uses self-organization phenomena in a game theoretical spatially generalized Prisoner's Dilemma Model which is represented in two-dimensional cellular automata space. It forms the temporal coalitions of participants during load balancing.

Maguluri *et al*. (2012) presented a paper on stochastic models of load balancing and scheduling in cloud computing clusters. In the proposed model, jobs are arrived according to stochastic process and request virtual machines which are specific in terms of resources. Resource allocation problem alone is handled in this model, concentrating on load balancing among servers and scheduling algorithms to configure virtual machines. The policies enforced in this model improves the throughput and better than best fit policy.

Cloud service models: Cloud is an internet based service model offering on-demand services to user without providing implementation and maintenance hurdles to the customers. Many cloud service providers offer wide range of services such as content management, messaging, social computing, collaboration, identity management, storage CRM and many more (Hogan *et al*., 2011). These services are categorized generally into three types and they are:

IaaS-Infrastructure as a Service: IaaS provides customers with the provision of processing, storing, networking and other important computing resources which a customer is able to deploy and run arbitrary software (Zissis and Lekkas, 2012). Customer need not manage or control the cloud infrastructure. Instead, have control over operating system, storage and applications. IaaS is easily used for website hosting, disaster recovery preparedness, testing and developing virtual machines and many more (Lenk and Tai, 2014). With IaaS model, service provider outsources the elements of infrastructure like virtualization, storage, networking, load balancing and so on.

PaaS-Platform as a Service: PaaS provides customers capability to deploy onto the cloud infrastructure customer created applications using programming languages, libraries, tools and services supported by the cloud provider (Subashini and Kavitha, 2011). Here the customer is having control over deployed applications and can configure settings for the application hosting environments. PaaS can be more extensible and provides a set of customer-ready features thereby delivering great flexibility and security. PaaS allows customers to build their own applications by delivering services and applications that are served by upper model (Rimal *et al*., 2009).

SaaS-Software as a Service: SaaS allows customers to user cloud provider's applications running on cloud infrastructure. In other words, it allows applications to remotely deployed and hosted in clouds (Armbrust *et al*., 2010). With browser to interact with applications, it provides support without installing programs. SaaS is an important service model and it meets the business expectations of IT company (Armbrust *et al*., 2009). SaaS moves the task of managing software and its deployment to third party services. SaaS tends to reduce the software cost ownership by removing the need for technical staff to install, manage and upgrade applications (Armbrust *et al*., 2010). SaaS also provides customers the relief of licensing problem.

The above discussed three service models are widely recognized by the researchers. Variations of the primary service models have been proposed, each one combines IT resources for specific applications. Other service models are Storage as a Service, Database as a Service, Security as a Service, Communication as a Service, Integration as a Service, Testing as a Service and Process as a Service (Krutz and Vines, 2010).

Load Balancing is the process of assigning the total load available to individual nodes so that response time is good and cloud resources are utilized in an efficient way. The load can be CPU capacity, total memory, network traffic, etc and hence it is necessary to distribute these loads equally to all the nodes in the network (Gkantsidis and Rodriguez, 2005). Load balancing divides the traffic among all the servers so as to make data communication without delay. Load balancing affects the overall system performance (Gkantsidis and Rodriguez, 2005).

Load balancing does not consider the previous or future state of the system. It is concerned with the current status of the system and hence load balancing algorithms are dynamic in nature. The objective of load balancing algorithm in cloud computing is to improve the system/network performance substantially, to backup the data so that any failure in the system can be tackled, to maintain system stability and to accommodate future modification in the system (Chaczko *et al*., 2011). Load balancing maximizes application performance and reliability, ease of scaling the applications up or down to match the demand.

Though the basic objective of all the load balancing algorithms is to evenly distribute the load across the network, based on the working style of the algorithm, they are classified into two categories.

Static Algorithms divide the network traffic equally among servers. Traffic is forced on servers that make them imperfect. Allocation of traffic is done using prior knowledge about the system resources and performance of processors (Patel *et al*., 2016). Hence imposing traffic on a system does not require the current status of the system. By this method, all the nodes will get equal load and it is fair. However, tasks are assigned to the processor only if it is created and that tasks cannot be shifted during its execution to any other machine for load balancing.

Dynamic Algorithms work based on the current status of the system and based on that, load transfer decisions are made. In dynamic algorithms, the server which is having low utilization is searched and allocated for balancing the load (Patel *et al*., 2016). It checks the real time status of the system for selecting light weight server. Here the processes are allowed to move from one machine to another in real time. Dynamic load balancing algorithms can be executed simultaneously by all the nodes in the network and scheduling task is shared among them. This setup is said to be distributed dynamic load balancing algorithm. In contrast to this, nodes can work personally to achieve a common optimization goal. This is termed as non-distributed load balancing algorithms.

Metrics for load balancing: To evaluate the effectiveness of load balancing algorithms, few parameters are used, which are considered important. In this section, we will discuss them:

- **Throughput:** This measure reflects the number of tasks executed. Generally high throughput is expected to have improved performance over the system.
- **Migration time:** It represents the time needed for jobs or resources to migrate from one node to another. In order to achieve high performance in the system, migration time should be minimal.
- **Overhead:** Amount of load produces by the load balancing algorithm. Generally it is expected to have minimum overhead.
- **Response time:** The time taken by a load balancing algorithm to give response to a task in a system. For better performance of a system, response time should be less.
- **Scalability:** It is the ability of the system to perform uniformly when finite number of nodes are added in the network.
- **Fault tolerance:** It is the measure representing the system behaviour incase of failure. Load balancing algorithm should be fault tolerant.
- **Resource utilization:** Load balancing algorithm should utilize the resources in a maximum extent.
- **Performance:** It represents the measure satisfying all the above parameters by a load balancing algorithm.

MATERIALS AND METHODS

Round Robin algorithm are extremely simple algorithm, round robin method uses time slicing mechanism to balance the load in cloud environments. Each node is given a time slice and nodes have to wait for their turn. If a node finishes the task within the time slice, it is removed from the execution list else it has to wait for next turn of time slice to execute (Smith *et al*., 2007). Here the time is divided into equal slots and based on First Come First Serve (FCFS), nodes are selected and executed in a round robin fashion. Advantage of using this algorithm is that method does not yield starvation and response time is high. But this method treats all the nodes equally and time slices are allocated to nodes that are temporarily down in network. Different processes have different processing times and hence few nodes may be heavily loaded while others are underutilized in this algorithm.

Variant of round robin algorithm are weighted round robin and dynamic round robin methods (Venkataraman *et al*., 2006). In the first variant, weights are assigned to each node based on certain criteria and nodes with higher weights are given the chance. Dynamic round robin method assign weights to nodes dynamically based on real-time data about nodes currently loaded and idle capacity.

Min-Min load balancing algorithm is a static load balancing algorithm where the cloud manager identifies the execution and completion time of the task waiting in the queue (Calheiros *et al*., 2011). Cloud manager deals with tasks having minimum execution time and processors are assigned to them according to the capability of completing the job in specific completion time.

Minimum completion time for all the nodes is calculated and node with minimum completion time gets the chance to get assigned the job. This algorithm is also simple and fast thus yielding improved performance (Krishnamachari and Ghosh, 2010). The problem with this algorithm is that larger tasks are keep on in waiting list leading to starvation. Moreover, this algorithm is less fault tolerant and less scalable.

Max-Min load balancing algorithm is a variant of min-min algorithm. It also finds the minimum execution time of all the nodes. Now cloud manager deals with tasks having maximum execution time and assigned task is removed from the list (Yu *et al*., 2010). Fresh calculation of the minimum execution time for all the available nodes is calculated and procedure is repeated. Meta tasks containing details about completion and execution time can be used as an enhanced version of max-min algorithm.

Honey Bee works using the concept of honey bee which searches its food and then informs to others using waggle dance (Bitam *et al*., 2010). This dance represents the amount of food present. In the same way, user requests on virtually overloaded machine is forwarded to next less loaded virtual machine, thereby achieving global load balancing using local server actions.

This algorithm is dynamic and distributed, builds advert that stores the status of each node and job has to be assigned to a node. By this way, parameter utilization is very high with less overhead. But his algorithm does not assign task to proper virtual machines, quality of service is not considered and throughput is less.

Active Clustering algorithm nominates one node as cluster head and it enhances the system performance. A group of clusters are made and active clustering groups similar nodes and then process these nodes (Abbasi and Younis, 2007). Creating cluster revolves around the concept of match maker node and a node selects its neighbor node which is of different type. This match maker node searches for its neighbor node which is of same type and makes connection. This process is repeated until all the nodes are accommodated in any one of the clusters. This set up increases the performance of the system where resources are highly available (Abbasi and Younis, 2007). Moreover, throughput is also increased due to efficient utilization of resources.

Ant Colony algorithm is an optimization method searches optimal path between food and colony of ants on the basis of their behaviour. It aims at efficient distribution of work load among the nodes (Adnan *et al*., 2013). Regional load balancing node is selected as head node. This algorithm searches the overloaded nodes and then traverse back to fill the under loaded nodes so that loads can be balanced. Ant Colony algorithm exhibits fault tolerance, resource utilization and good scalability. This algorithm is dynamic in nature, uses less network overhead (Adnan *et al*., 2013). Complex networks are the drawbacks of this algorithm.

Random Sampling algorithm treats network as a virtual graph and each server is taken as a vertex of node, in-degree represents the available free resources the nodes have (Wang and Wu, 2013). Based on the in-degree value of the nodes, algorithm allocates the job. Nodes having least in-degree value gets the job and in-degree value of the node is decremented by 1 when a job is allocated to that node.

It is a centralized method and indicates the maximum traversal from one node to destination node. This distance is known as walk length (Hui *et al*., 2011). For a request, algorithm selects a node randomly and compares the current walk length with threshold value. If it is equal or higher, job is executed at that node else walk length of the job is incremented and another node is selected randomly.

Equally Spread Current Execution, ESCE, handles many requests with their priorities. It randomly selects the virtual machines which are lightly loaded and transfer the load. ESCE takes less time to transfer the load with maximum throughput. An index table is maintained which holds the record of incoming requests (Sockut and Iyer, 2009). When a virtual machine completes the task, a request is communicated to data centre and further to load balancer. This makes load balancer to assign fresh request to virtual machine and this method is dynamic, centralized. Always there exists a communication between load balancer and data centre, which results in updation of index table. This algorithm is well suited for dynamic and centralized environments.

CONCLUSION

Cloud computing is a dominant technology adopted by It industry. Among various issues in cloud computing, load balancing is an important issue that need to be addressed. Load balancing aims to satisfy the user and effective utilization of resources. It is essential in distributing extra dynamic local workload consistently to the entire network to attain uniformity. In this study, a survey on various load balancing mechanisms is presented. The methods discussed are round robin, min-min, max-min, honey bee, active clustering, any colony, random sampling and ESCE. The paper also presents the parameters used to evaluate the effectiveness of any load balancing algorithm.

REFERENCES

Abbasi, A.A. and M. Younis, 2007. A survey on clustering algorithms for wireless sensor networks. Comput. Commun., 30(14): 2826-2841.

Adnan, M.A., M.A. Razzaque, I. Ahmed and I.F. Isnin, 2013. Bio-Mimic optimization strategies in wireless sensor networks: A survey. Sensors, 14(1): 299-345.

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, Electrical Engineering and Computer Sciences, University of California, Berkeley.

Armbrust, M., A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, 2010. A view of cloud computing. Commun. ACM, 53(4): 50-58.

Badger, L., T. Grance, R. Patt-Corner and J. Voas, 2011. Draft Cloud Computing Synopsis and Recommendations. NIST Special Publication 800-146. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8930, pp: 84.

Bitam, S., M. Batouche and E.G. Talbi, 2010. A survey on bee colony algorithms. Proceeding of the IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum (IPDPSW, 2010), pp: 1-8.

Buyya, R., C.S. Yeo, S. Venugopal, J. Broberg and I. Brandic, 2009. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comp. Sy., 25(6): 599-616.

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. Busan, Korea, pp: 13-31.

Calheiros, R.N., R. Ranjan, A. Beloglazov, C.A.F. De Rose and R. Buyya, 2011. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software Pract. Exper., 41(1): 23-50.

Chaczko, Z., V. Mahadevan, S. Aslanzadeh and C. Mcdermid, 2011. Availability and load balancing in cloud computing. Proceeding of the International Conference on Computer and Software Modeling. Singapore, 14: 134-140.

Devi, D.C. and V.R. Uthariaraj, 2016. Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks. Sci. World J., 2016: 14.

Domanal, S.G. and G.R.M. Reddy, 2014. Optimal load balancing in cloud computing by efficient utilization of virtual machines. Proceeding of the 6th International Conference on Communication Systems and Networks (COMSNETS, 2014), pp: 1-4.

Gasior, J. and F. Seredynski, 2012. Load balancing in cloud computing systems through formation of coalitions in a spatially generalized prisoner's dilemma game. Proceeding of the 3rd International Conference on Cloud Computing, GRIDs and Virtualization, Nice, France, pp: 201-205.

Gkantsidis, C. and P. Rodriguez, 2005. Network coding for large scale content distribution. Proceeding of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 4: 2235-2245.

Hogan, M., F. Liu, A. Sokol and J. Tong, 2011. Nist Cloud Computing Standards Roadmap. NIST Special Publication, New York, pp: 35.

Hui, P., J. Crowcroft and E. Yoneki, 2011. BUBBLE Rap: Social-based forwarding in delay-tolerant networks. IEEE T. Mobile Comput., 10(11): 1576-1589.

Krishnamachari, B. and A. Ghosh, 2010. Algorithmic aspects of throughput-delay performance for fast data collection in wireless sensor networks. Ph.D. Thesis, University of Southern California.

Krutz, R.L. and R.D. Vines, 2010. Cloud Security: A Comprehensive Guide to Secure Cloud Computing. Wiley Publishing, Indianapolis, Indiana.

Lenk, A. and S. Tai, 2014. Cloud Standby: Disaster Recovery of Distributed Systems in the Cloud. In: Villari, M., W. Zimmermann and K.K. Lau (Eds.), Service-Oriented and Cloud Computing. Springer, Berlin, Heidelberg, 8745: 32-46.

Maguluri, S.T., R. Srikant and L. Ying, 2012. Stochastic models of load balancing and scheduling in cloud computing clusters. Proceeding of the IEEE INFOCOM, pp: 702-710.

Patel, D.K., D. Tripathy and C.R. Tripathy, 2016. Survey of load balancing techniques for Grid. J. Netw. Comput. Appl., 65: 103-119.

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 (NCM '09), pp: 44-51.

Smith, G., A. Chaturvedi, A. Mishra and S. Banerjee, 2007. Wireless virtualization on commodity 802.11 hardware. Proceeding of the 2nd ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization, pp: 75-82.

Sockut, G.H. and B.R. Iyer, 2009. Online reorganization of databases. ACM Comput. Surv., 41(3), Article No. 14.

Subashini, S. and V. Kavitha, 2011. A survey on security issues in service delivery models of cloud computing. J. Netw. Comput. Appl., 34(1): 1-11.

Venkataraman, V., K. Yoshida and P. Francis, 2006. Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. Proceeding of the 14th IEEE International Conference on Network Protocols, pp: 2-11.

Wang, Y. and X. Wu, 2013. Preserving differential privacy in degree-correlation based graph generation. Trans. Data Privacy, 6(2013): 127-145.

Xu, Y., L. Wu, L. Guo, Z. Chen, L. Yang and Z. Shi, 2011. An intelligent load balancing algorithm towards efficient cloud computing. Proceeding of the 8th AAAI Conference on AI for Data Center Management and Cloud Computing, pp: 27-32.

Yu, L., W.T. Tsai, X. Chen, L. Liu, Y. Zhao, L. Tang and W. Zhao, 2010. Testing as a service over cloud. Proceeding of the 5th IEEE International Symposium on Service Oriented System Engineering (SOSE, 2010), pp: 181-188.

Zhang, Q., L. Cheng and R. Boutaba, 2010. Cloud computing: State-of-the-art and research challenges. J. Internet Serv. Appl., 1(1): 7-18.

Zissis, D. and D. Lekkas, 2012. Addressing cloud computing security issues. Future Gener. Comp. Sy., 28(3): 583-592.