ISSN: 2321-7782 (Online)
Volume 2, Issue 12, December 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

A comparative study of static and dynamic Load Balancing Algorithms

Payal Beniwal[1]
Research Scholar /MMICT&BM
Maharishi Markandeshwar University, Mullana
Ambala, Haryana - India

Atul Garg[2]
Associate Professor/MMICT&BM
Maharishi Markandeshwar University, Mullana
Ambala, Haryana, India

Abstract: ***A distributed system can be viewed as an environment in which, number of computers/nodes are connected together and resources are shared among these computers/nodes. But unfortunately, distributed systems often face the problem of load imbalance, which can degrade the performance of the system. Load balancing is used to improve scalability and overall system throughput in distributed systems. Load balancing improves the system performance by dividing the work load effectively among the participating computers/nodes. Many algorithms were proposed for load balancing and their performance is measured on the basis of certain parameters such as, response time, resource utilization and fault tolerance. Load balancing algorithms are broadly classified in two categories- static and dynamic load balancing. This paper presents the study of performance analysis of load balancing algorithms. This analysis can further help in the design of new algorithms.***

Keywords: Distributed systems, load balancing, static load balancing, dynamic load balancing and performance parameters.

I. INTRODUCTION

Technology has completely changed the modern life standard and web services play crucial part in this technology. Internet is one such service that has connected us all in a way that was barely imaginable. It is widely involved in all aspects of our life, as it serves many purposes like communication, business, entertainment, education, social network etc., with this outstanding progress in computer technology the demand of high speed processing has risen and so the need of high scalability, availability and rapid response. Today millions of users use internet, this rapid growth has lead to the storage of tremendous amount of data and rise in web traffic. This has further lead to the problems of network congestion, web service delay and slow response time. These factors increased the trend of using distributed systems. A distributed system consists of computing nodes connected together by a communication network [12]. Apart from sharing data and I/O devices, distributed systems also share the computational power among nodes, which improves the system's performance. However, users get a single system image of this physically distributed system [12]. To increase the efficiency of network multiple servers are deployed connected to each other to form a server farm. Client request is transferred to one of the back end server in the server farm. It replies to client request without the client ever knowing about the internal distribution. The process of distributing the client request among servers is performed by load balancing technique. Load balancing is defined as a process of allocating the total work load to the individual nodes of the distributed systems to improve resource utilization and response time, also avoiding the condition in which some nodes are overloaded while others are under loaded or server failure [24]. In general, as the complexity of distributed systems grows, their load balancing requires more advancement functionality, such as the ability to tolerate faults, install new load balancing algorithms at run time and to create replica to handle busty clients. Distributed systems performance and scalability will be affected due to the lack of above functionalities. A load balancing algorithm should be general and transparent to the applications, also it should provide minimum overhead to the system [15]. The load balancing process is defined in three rules, first one is location rule which gives information about the resource domain to be included in the

balancing operations and second rule is distribution rule which is used in case redistribution among nodes is needed. Lastly selection rule which decides whether the load balancing can be performed or not [13].

According to researchers [24] load balancing perform following functions (Figure 1):

» Intercepts the load sent by the user.

» Divides the load among individual nodes and decides which node should receive and process the request.

» Maintains the log of available nodes and ensures that they are responding to the load. If not they are taken out of rotation.

» Provide redundancy by deploying separate units for failures.

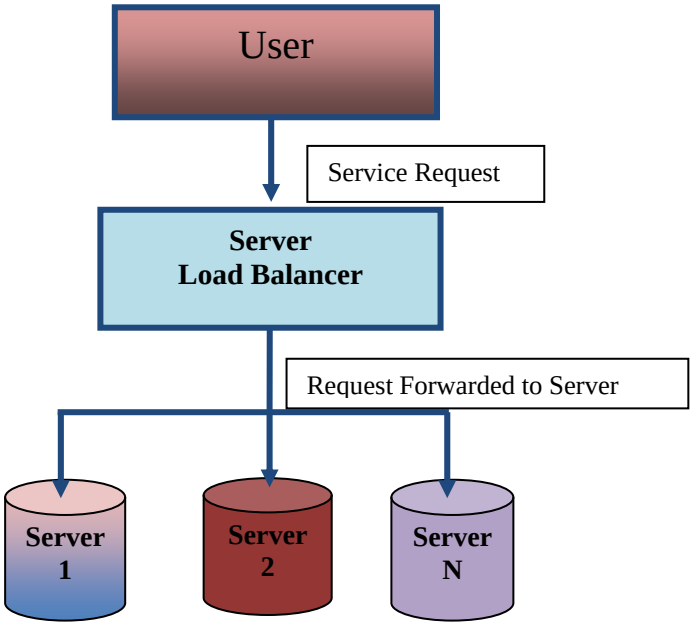

Figure 1: Load Balancer

In general load balancing algorithms are classified as static or dynamic and centralized or distributed [12]. In case of static load balancing load distribution depends on the load at the time of selection of node whereas dynamic load balancing performs load distribution at run time. It uses current load information for making distribution decisions [10]. Some dynamic algorithms are adaptive i.e. the algorithms can be modified as the system state changes. Applications having constant work load will perform better with static algorithms, whereas adaptive applications where workload is unpredictable or change during execution dynamic algorithms give desired performance. Potentially the more information an algorithm can collect the better decision it will make. Although dynamic algorithms give better performance they are difficult to implement, static algorithms also have their merits and demerits. In centralized algorithm there is a central node which gathers information from all nodes and makes the load distribution decisions accordingly. This method gives efficient result in small applications. Whereas in distributed approach there is no central node, rather all connected nodes have copy of information regarding each other. The information is updated as the state of any node is changed. This sharing of information has additional overhead of inter process communication [13]. Importance of load balancing in World Wide Web has lead to the development of various techniques that use behavior of species for finding optimal solution to problems. Atul et al., [20] proposed one such algorithm which uses behavior of ants to explore the best suited path to reach the destination.

In this paper seven load balancing algorithms are compared using various parameters. In next section the related work done by various researchers is discussed. In section III various algorithms used for loud balancing are analyzed and compared. Last section conclude the research work done in this paper.

II. Literature Review

Tremendous amount of research is done on load balancing and is still going on. This section provides the briefing of work done by researchers. Luis & Azer [25] presented the low overhead distributed packet rewriting (DPR) technique to redirect TCP connections. In the research work proposed by Luis & Azer [25], each host keeps information about the remaining hosts in the system. Load information is maintained using periodic multicast amongst the cluster hosts. Performance measurements suggest that this prototype outperforms both pure RR-DNS and the stateless DPR solutions. Ruud et al., [7] derived ant - like agents to implement load balancing in telecommunication networks. The performance of the network is measured by the proportion of calls which fail. The results are compared with those achieved by using fixed shortest-path routes, and also by using an alternative algorithmically based type of mobile agent. The technique proved to be promising but need to be improved for overcoming the various technical problems in the network. Kyeongmo & Myong [11] proposed a fair-ready scheduling method by using the memory utilization parameter of real servers for load balancing in Linux web server cluster. The method is compared to its previous version ABSS which was based on CPU utilization parameter. Result shows that later designed method is much more efficient and provides steady service under heavy request traffic. Mohsen et al., [15] presented an echo system of adaptive fuzzy ants to meet the challenges of load balancing. The ants in this environment can create new ones and may also commit suicide depending on existing conditions. Theoretical analysis is also done to prove that this new mechanism surpasses its predecessor. More work is to be done to implement the mechanism in realistic environment. Also the framework doesn't address the issues of security. Aissatou & Foroski [19] Proposed dynamic distributed load balancing for D-GIS in order to quickly render information to client interface by involving a set of GIS servers which process clients' requests depending of an algorithm. Concept of load hash table and failover callback function is also defined. The load-balancing algorithm proposed intercepts the incoming HTTP traffic and directs it to one of the servers in the cluster. The solution provides high scalability but still need to be implemented for business purposes. Fengyun et al., [14] presented a simple and effective approach of load balancing for MMORPGs and many more online games that use clusters of server for high scalability. Behavioral approach to load balancing is taken into consideration as compared to geographical approach as it suffers from problem of crowding. The method showed high rate of scalability but there is need to conduct much research in this area. Sandeep et al., [16] presented the performance analysis of static and dynamic load balancing algorithms. Comparison is done the various parameters of overload rejection, fault tolerance, accuracy and stability etc. Selection of load balancing algorithm is based on situation in which work load is assigned i.e. at run time or compile time. Static algorithms are proved to be more stable as compare to dynamic load balancing algorithms. Atul et al., [21] presented the comparison of various evolutionary search methods developed for query optimization on web servers. Evolutionary algorithms are analyzed and compared on the basis of their behavior, flexibility and transmission methods etc. Results show that performance of PSO is better as compared to other algorithms buts lack behind in processing time. Dimple & Atul [20] proposed an ant based framework to balance the load. In their work the author proposed an active ant at client side and the server side both. The client ant is responsible for the request whereas the server ant is responsible for replying the request. The authors improved the server performance in their work.

III. Analysis And Comparison

This section of paper presents the various algorithms chosen for the comparative analysis. For load balancing there are two approaches and each approach have their own algorithms. These approaches with respect to their algorithms are discussed below.

Static Load Balancing: In this approach load balancing is achieved by providing priori information about the system. The performance of the node is determined at the commencement of execution. Nodes calculate their allotted work and submit the result to remote node. Then depending on the performance work load is distributed in start without considering the current load [13]. Static load balancing methods are non-preemptive i.e. once the load is allocated to the node it cannot be transferred to another node. This method requires less communication hence reduces the execution time [16]. The main drawback of this

approach is that it does not take current state of the system while making allocation decisions. This has the major impact on the overall performance of the system due to load fluctuation in distributed system [18].There are three types of static load balancing algorithms: round robin, central manager, threshold algorithm and randomized algorithm.

a) Round Robin Algorithm [12]: In this algorithm the load is distributed evenly to all nodes. Work load is distributed in round robin order, where equal load is assigned to each node in circular order without any priority and will be back to the first node if the last node has been reached. Each node maintains its load index independent of allocations from remote node. Round robin is easy to implement, simple and starvation free. It does not require inter process communication and gives best performance for special purpose applications. It cannot give expected result in general case and when the jobs are of unequal processing time.

***b) Central Manager Algorithm [16*]:** In this algorithm a central node selects the slave node for transferring the load. Slave node having the least load is selected and assigned the job. The central node maintains the load index of all slave nodes connected to it. Whenever, load is changed, a message is send by the slave nodes to the central node. This algorithm needs high level of inter process communication, which can sometimes lead to the bottleneck state. This algorithm performs better when dynamic activities are created by different hosts.

c) Threshold Algorithm [13]: According to this algorithm the load is assigned immediately upon creation of node. Nodes are selected locally without sending remote messages. Each node keeps a private copy of the system's load. The load is characterized is in three levels: underload, medium and overloaded. Two parameters tunder and tupper are used to describe these levels.

Under loaded - load<tunder

Medium - tunder<= load<=tupper

Overloaded - load>tupper

Initially, all the nodes are considered in under loaded level. When the load state of a node exceeds a load level limit, then it sends messages regarding the new load state to all remote nodes, regularly updating them as to the actual load state.

If the local state is not overloaded then the load is allocated locally. Otherwise a remote under loaded node is selected and if no such node exists it is also allocated locally. This algorithm has low inter process communication and large number of local process allocations. The later reduces the overhead of remote process allocation and the overhead of remote memory access, which leads to improvement in performance.

d) Randomized Algorithm [15]: in this algorithm the node is selected on random selection, without having any information about the current or previous load on the node. As the algorithm is static in nature it is best suited when the system has equal load on each node. Gives best performance for special purpose applications. Each node maintains its own load record hence no inter process communication is required. But sometimes it may cause a single node overloaded while the other node is under loaded.

Dynamic Load Balancing: These algorithms monitor changes on the system work load and redistribute the work accordingly. This algorithm usually composed of three strategies: transfer strategy, location strategy and information strategy. Transfer strategy decides on which tasks are eligible for transfer to other nodes for processing. Location strategy nominates a remote node to execute a transferred task. Information strategy is the information center for load balancing algorithm [24]. It is responsible for providing location and transfer strategies to each node. Dynamic algorithms can take three different controlling forms: centralized, distributed, or semi-distributed. In centralized load distribution, a single (central node) node in the network is nominated to be responsible for all load distribution in the network. In distributed the responsibility is divided among all nodes equally [24]. In a semi- distributed the network is segmented into clusters where each cluster is centralized. Load balancing of

whole system is achieved through the cooperation of central nodes of all clusters [24]. There are three types of dynamic algorithms: central queue , local queue and least connection.

a) Central Queue Algorithm [13]: This algorithm stores new activity and unfulfilled requests in a cyclic FIFO queue. Each new activity is inserted in the queue. Then, whenever a request for an activity is received the first activity is removed from the queue. If there is not any requested activity in the queue then the request is buffered until a new activity is available. This is a centralized initiated algorithm and need high communication among nodes.

b) Local Queue Algorithm [16]: this algorithm supports inter process migration. This idea is static allocation of all new process with process migration initiated by the host when its load falls under the predefined minimum number of ready processes. When the host gets under load it request for the activities from the remote hosts. The remote hosts than look up its local list for ready activities and some of the activities are passed on to the requestor host and get the acknowledgement from the host. This is a distributed co-operative algorithms requires inter process communication but lesser as compared to central queue algorithm.

c) Least Connection Algorithm [24]: This algorithm decides the load distribution on the basis of connections present on a node. The load balancer maintains the log of numbers of connections on each node. The number increases when a new connection is established and decreases when connection finishes or time out. The nodes with least number of connections are selected first.

Further, based on various dynamic and static algorithms various parameters are selected as per the requirement of the modern era. The following table (Table I) shows the list of parameters with their description:

TABLE I: LIST OF PARAMETERS

PARAMETERS	DESCRIPTION
Nature	Determines the behavior of the algorithm i.e. whether static or dynamic.
Overhead	Determines the amount of overheads like inter-process communication, migration of tasks etc. involved while implementing the algorithm. This should be minimum so that algorithm can work effectively.
Throughput	It is used to calculate the no. of tasks whose execution has been completed. It should be high for better performance of algorithm.
Process Migration	It is the time to migrate the jobs from ne node to other. It should be minimum to enhance the system performance.
Response Time	It is the amount of time taken by a load balancing algorithm to complete a task. It should be minimized.
Resource Utilization	It is used to check the utilization of resources, should be optimized in order to get good performance.
Fault Tolerant	It enables any algorithm to work continuously in the event of some failure; even a small failure can degrade the performance of an algorithm.
Waiting Time	It is defined as the time period spends in the ready queue, less the waiting time better the performance.
Scalability	It is defined as the ability of an algorithm to give optimized result with any finite number of nodes.
Performance	It is used to check the efficiency of the system.

The algorithms are compared using the parameters mentioned above in table I, the resulted analysis is given in table II below.

TABLE II: COMPARATIVE ANALYSIS OF LOAD BALANCING ALGORITHMS

Algorithms / Parameters	Round Robin	Central Manager	Threshold	Randomized Algorithm	Central Queue	Local Queue	Least Connection
Nature	Static	Static	Static	Static	Dynamic	Dynamic	Dynamic
Overhead	Low	High	Low	Low	High	High	High
Throughput	Low	Low	Low	High	High	High	Low
Process Migration	No	No	No	No	No	Yes	No
Response Time	Less	Less	Less	Less	More	More	Less
Resource Utilization	Less	Less	Less	Less	Less	More	More
Fault Tolerant	No	Yes	No	No	Yes	Yes	No
Waiting Time	More	More	More	More	Less	Less	Less
Scalability	High	Low	High	Less	Low	High	High
Performance	Low	Low	Low	Less	High	High	High

IV. CONCLUSION

Load balancing is required to distribute the workload evenly across all nodes to achieve high performance; with minimum overheads. With proper load balancing waiting time can be kept to a minimum which will further maximize the response time. In this paper, comparison of different load balancing algorithms is carried out on the basis of certain parameters. The above comparison shows that static load balancing algorithms are more stable than dynamic algorithms but due to capability of performing accurate in distributed systems, dynamic load balancing is chosen over static load balancing algorithms. This analysis further can also help in designing new load balancing algorithms.

References

1. P.L.McEntire, J.G.O'Reily "Distributed Computing: concepts and implementation. New York: IEEE Press, 1984.
2. Y.Wang and R.Morrois, " Load balancing in distributed systems", IEEE Trans.Computing.C-34, no.3, pp. 204-217, Mar.1985.
3. Derek L. Eager, Edward D.Lazowska,'Adaptive load sharing in homogenous distributed systems", IEEE trans. On software engg., v.12 n.5, p.662-675, May 1986.
4. L.Rudolph, M. Slivkin-Allalouf, " A simple load balancing scheme for task allocation in parallel machines", Proceedings of 3rd ACM Symposium on Parallel Algorithms and Architectures, pp.237-245, July 1991.
5. Zong Xu, Rong Hung, " Performance study of load balancing algorithms in distributed web server systems", CS213 Parallel and Distributed Processing Project Report.
6. M.Zaki, W.Li and S.Parthasarathy," Customized dynamic load balancing for a network of workstations ". Journal of parallel and distributing computing: Special issue , June 1997.
7. Ruud Schoonderwoerd, Owen Holland & Janet Bruten, "Ant-like agents for load balancing in telecommunications networks.", Agnets'97,CA,USA(August 1997)
8. S.P.Dandamundi, " Sensivity evaluation of dynamic load sharing in distributed systems", IEEE p.no- 62-72, 1998.
9. S.Malik, " Dynamic load balancing in a network of workstations", 95.515 research report, 19 NOV. 2000.
10. William Leinberger, George Karpis, Vipin Kumar, "Load balancing across near-homogeneous multi-resource servers", 0-7695-0556-2/00, 2000 IEEE.
11. Kyeongmo Kang, MinHwan Ok and Myong-soon "Agent-Based Fair Load Distribution in Linux Web Server Cluster" S. R. Das, S. K. Das (Eds.): IWDC 2003, LNCS 2918, pp. 143-152, 2003.

12. Abubakar, Haroon Rashid and Usman, " Evaluation of load balancing strategie", National Conference on Emerging Technologies, 2004.
13. Daniel Grousa, Anthony T. ," Non-Cooperative load balancing in distributed systems". Journal of Parallel and Distributing Computing, 2005.
14. Fengun, Simon and Graham," Load balancing for massively multiplayer online games ", Netgames 06, Oct 30-31, 2006,Singapore.
15. Mohsen & Hossein Delda " Balancing Load in a Computational Grid Applying Adaptive, Intelligent Colonies of Ants". Informatica 32 (2008) 327–335.
16. Sandeep Sharma, S.Singh and Meenakshi, "Performance analysis of load balancing algorithms", World academy of science, 2008.
17. Parveen Jain and Daya Gupta ," An algorithm for dynamic load balancing in distributed systems with multiple supporting nodes by exploiting the interrupt service", IJRTE, Vol 1, no.1, May 2009.
18. Ali M.Alakeel ," a guide to load balancing in distributing computer systems", IJCSNS, vol 10, no.6, June 2010.
19. Aissatou Diasse "Dynamic-Distributed Load Balancing for Highly-Performance and Responsiveness Distributed-GIS (D-GIS)" Journal of Geographic Information System, April 2011.
20. Dimple Juneja and Atul Garg," Collective Intelligence based framework for load balancing of web servers", IJICT, Vol 3 No-1 Jan-2012.
21. Atul Garg and Dimple Juneja,"A Comparison and analysis of various extended techniques of query optimization", IJICT Vol-3 no-3 July-2012.
22. Abhijit and S.S.Apte, "A comparative performance analysis of load balancing algorithms in distributed systems using qualitative parameters", IJRTE, vol. 1, issue-3, August 2012.
23. Ali M.Alakeel, "A fuzzy dynamic load balancing algorithm for homogeneous distributed systems", world academy of science, engineering and technology, vol:6 2012-01-21.
24. Suriya and Prashanth ," Review of load balancing in cloud computing" . IJCS, vol.10 issue.1, Jan 2013.
25. Luis Aversa & Azer Bestavros, "Load Balancing a Cluster of Web Servers Using Distributed Packet Rewriting" NSF research grants CCR-9706685.

AUTHOR(S) PROFILE

Payal Beniwal, received the M. Sc degree in computer science & applications from Panjab University, Chandigarh in 2010. Presently research scholar in Maharishi Markandeshwar University, Mullana (Ambala), Haryana.

Atul Garg received degree of Master of Computer Applications from Kurukshetra University, Kurukshetra in 2004 and completed his Ph. D degree from Maharishi Markandeshwar University, Mullana (Ambala) in 2013. Currently, he is working as an Associate Professor at M.M.I.C.T&B.M., Maharishi Markandeshwar University, Mullana (Ambala), Haryana. He is Senior Member of the association of Universal Association of Computer & Electronics Engineers (UACEE), Australia, member in the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (ICST), Belgium and Member in the International Association of Engineers, Hong Kong. His area of interest is web, Query Optimizations and mobile ad hoc networks.

