IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 4, Ver. V (Jul.-Aug. 2016), PP 153-158
www.iosrjournals.org

A Survey Report on Distributed System Using Load Balancing Approach

Arju Malik [1],PankajPratap Singh[2]
[1 2] *Department of Computer Science & Engineering and Information Technology,*
[1 2] *Dr. A.P.J. Abdul Kalam Technical University, Uttar Pradesh, India*

Abstract:*A number of load balancing algorithm were developed in order to improve the execution of a distributed application in any kind of distributed architecture. Load balancing involves assigning tasks to each processor and minimizing the execution time of the program. In practice, it would be possible even to execute the applications on any machine of worldwide distributed system. This results in a significant performance improvement for the users. This paper describes the necessary, newly developed, principal concepts for several load balancing techniques in a distributed computing environment. This paper also includes various types of load balancing strategies, their merits, demerits and comparison depending on certain parameters. Load balancing is the process of distributed the load among various nodes of a distributed system to improve both job response time and resource utilization while also avoiding a situation where some of the nodes are heavily loaded while other nodes are idle or lightly loaded.*

Keywords:*Load balancing, Distributed system, Internet, Server, QoS.*

I. Introduction

Load balancing is dividing the amount of work that a computer has to do between two or more computers so that more work gets done in the same amount of time and, in general, all users get served faster. Load balancing can be implemented with hardware, software, or a combination of both. Typically, load balancing is the main reason for computer server clustering. On the Internet, companies whose Web sites get a great deal of traffic usually use load balancing. For load balancing Web traffic, there are several approaches. For Web serving, one approach is to route each request in turn to a different server host address in a domain name system (DNS) table, round-robin fashion[1]. Usually, if two servers are used to balance a work load, a third server is needed to determine which server to assign the work to. Since load balancing requires multiple servers, it is usually combined with failover and backup services. The approach considers the heterogeneity in the processing rates of the nodes as well as the randomness in the delays imposed by the communication medium. The optimal one-shot load balancing policy is developed and subsequently extended to develop an autonomous and distributed load-balancing policy that can dynamically reallocate incoming external loads at each node[2]. This adaptive and dynamic load balancing policy is implemented and evaluated in a two-node distributed system.

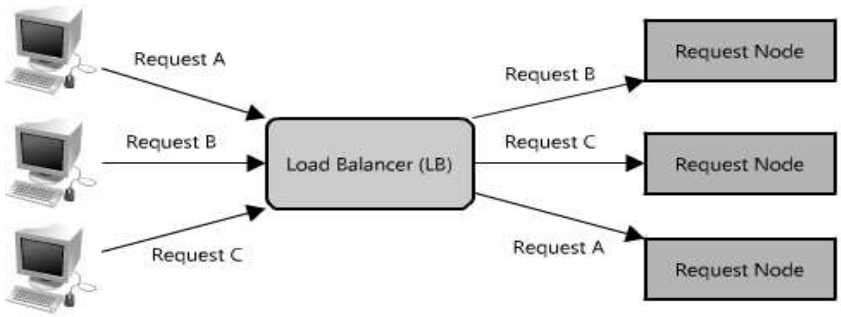

Fig1. Load Balancing Approach

Distributed computing is a field of computer science that studies distributed systems. A distributed system consists of multiple computers that communicate through a computer network. The computers interact with each other in order to achieve a common goal. A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs. Distributed computing also refers to the use of distributed systems to solve computational problems. In distributed computing, a problem is divided into many tasks, each of which is solved by one or more computers which communicate with each other by message passing. The word distributed in terms such as "distributed system",

"distributed programming", and "distributed algorithm" originally referred to computer networks where individual computers were physically distributed within some geographical area[3]. The terms are nowadays used in a much wider sense, even referring to autonomous processes that run on the same physical computer and interact with each other by message passing.Before there were any commercially available, purpose-built load balancing devices, there were many attempts to utilize existing technology to achieve the goals of scalability and HA. The most prevalent, and still used, technology was DNS round-robin. Domain name system (DNS) is the service that translates human-readable names into machine recognized IP addresses. DNS also provided a way in which each request for name resolution could be answered with multiple IP addresses in different order.

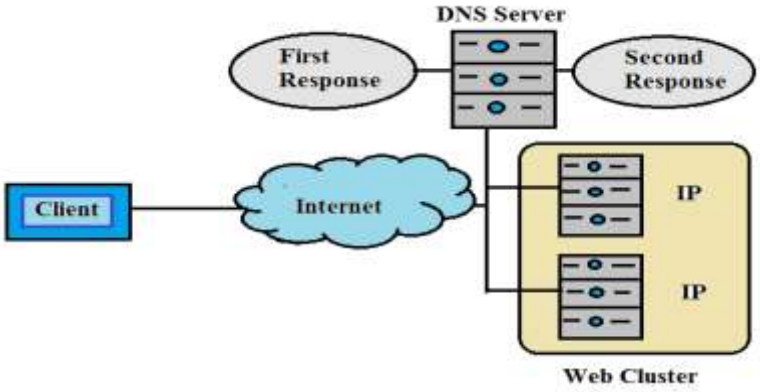

Fig2. Basic DNS Response for Redundancy

II. Literature Review

When you submit your Enhance Load Rebalance Algorithm for Distributed File Systems in Clouds[4]**Kokilavani .K**discussed on basic concepts of Enhance distributed load rebalancing algorithm to cope with the load imbalance factor, movement cost, and algorithmic overhead. The load rebalance algorithm is compared against a centralized approach in a production system and the performance of the proposal implemented in the Hadoop distributed file system for cloud computing applications. We investigate to implement security provided for cloud computing and Evaluate the Quality of Service-QOS (Ex. Response Time) of whole system. In cloud computing one server controls number of sub servers, files, it can add, delete, and append dynamically.A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing[5]**M. Randles, D.Lamb, and A.Taleb-Bendiab**discussed on Distributed Load Balancing Algorithms are Honeybee Foraging Behavior, Biased Random Sampling, and Active Clustering. Honeybee Foraging Behavior investigated a decentralized honeybee-based load balancing technique that is a nature-inspired algorithm for self-organization. It achieves global load balancing through local server actions. Performance of the system is enhanced with increased system diversity but throughput is not increased with an increase in system size. It is best suited for the conditions where the diverse population of service types is required. Biased Random Sampling investigated a distributed and scalable load balancing approach that uses random sampling of the system domain to achieve self-organization thus balancing the load across all nodes of the system. The performance of the system is improved with high and similar population of resources thus resulting in an increased throughput by effectively utilizing the increased system resources. It is degraded with an increase in population diversity. Active Clustering investigated a self-aggregation Load Balancing Techniques that is self-aggregation algorithm to optimize job assignments by connecting similar services using local re-wiring. The performance of the system is enhanced with high resources thereby increasing the throughput by using these resources effectively. It is degraded with an increase in system diversity. Game-theoretic static load balancing for distributed systems[6]. **Penmatsa and Chronopoulos**discussed on static load balancing strategy based on game theory for distributed systems. And this work provides us with a new review of the load balance problem in the cloud environment. As an implementation of distributed system, the load balancing in the cloud computing environment can be viewed as a game.

The Comparative Study on Load Balancing Techniques in Distributed Systems[8]**P. BeaulahSoundarabai, Sandhya Rani A., Ritesh Kumar Sahai, Thriveni J., K.R. Venugopaland L.M. Patnaik**discussed on the Scalability becomes the basic need for distributed systems. With the increase of users, load on application serversalso keep increasing. This requires a critical action to balance the load on servers. Load balancing is the concept of balancing load on servers using various load balancing techniques. In this paper we discuss various load-balancing techniques that are currently available. We discuss inherent characteristics, operational process, advantages and disadvantages of various static load-balancing techniques and dynamic load-balancing techniques. Also we made a comparative study on various parameters of the load-

balancing techniques.A server program that is installed on a targeted system has limited resources. These resources include system memory, hard disk space, and processor speed. Since the server capacity is limited, so it can handle only certain number of clients. With more number of clients a server will be overloaded and that may lead to slow down of the performance, hang and crash issues. So, it is crucial to balance the load on server and this can be achieved by keeping copies of servers and distribute the load among them. Load balancing is the process of making a group of servers participate in the same service and do the same work[9]. The general purpose of load balancing is to increase availability, improve throughput, reliability, maintain stability, optimize resource utilization and provide fault tolerant capability. As the number of servers grows, the risk of a failure anywhere increases and such failures must be handled carefully.

The ability to maintain unaffected service during any number of simultaneous failures is termed as high availability. And "Distributed Algorithms for QoS Load Balancing[10]" **Heiner Ackermann, Simon Fischer, Martin Hoefer, Marcel Schongens**discussed on the consider a dynamic load balancing scenario in which users allocate resources in a non-cooperative and selfish fashion. The perceived performance of a resource for a user decreases with the number of users that allocate the resource. In our dynamic, concurrent model, users may reallocate resources in a round-based fashion. As opposed to various settings analysed in the literature, we assume that users have quality of service (QoS) demands. A user has zero utility when falling short of a certain minimum performance threshold and having positive utility otherwise. Whereas various load-balancing protocols have been proposed for the setting without quality of service requirements, we consider protocols that satisfy an additional locality constraint: The behaviour of a user depends merely on the state of the resource it currently allocates. This property is particularly useful in scenarios where the state of other resources is not readily accessible. For instance, if resources represent channels in a mobile network, then accessing channel information may require time-intensive measurements [11]. We consider several variants of the model, where the quality of service demands may depend on the user, the resource, or both. For all cases we present protocols for which the dynamics converge to a state in which all users are satisfied. More importantly, the time to reach such a state scales nicely. It is only logarithmic in the number of users, which makes our protocols applicable in large-scale systems.

III. Load-Balancing Method

1. Primary Approach for Dynamic Load Balancing:

A distributed system consists of independent workstations which are connected usually by a local area network. Static load balancing fulfil the requirements for load balancing. As in static load balancing, number of jobs at a station is fixed[7]. Dynamic load balancing does the process while job are in execution. Jobs are allocated to host or node. Processes are migrated from heavily loaded node to light weighted node.

2. Centralized Approach for Load Balancing:

If a heavily loaded node does not find node in its cluster and due to congestion in network, node fail to search the node far away cluster. It is better that if heavily loaded node finds a temporary node in same cluster to handle the over load. So, in centralized approach one centralized node is provided in each cluster. The overload from nodes is transferred to centralized node to increase the output of each node.

3. Modified Approach for Dynamic Load Balancing:

In Centralized approach there is single node, so it processes the load at high speed by using switching but still a limitation exists. Away to remove the limitation is to split the centralized node into small nodes called supporting nodes (SNs). But still here supporting node are not allotted load initially. Many times supporting nodes is idle or they are not properly loaded as only overload is assigned to supporting nodes. This is wastage of power of supporting nodes. We can also use the free time of SN by making them busy for free time[12].

4. Algorithm for Modified Approach:

There are two types of nodes. They are Primary and supporting nodes. Primary nodes are the main nodes and supporting are used to handle overload .Primary node tries to approach supporting node and will find suitable supporting node, after finding suitable and interrupts SN for execution of its process. Conclusion for dynamics of a distributed computing system in the context of load balancing, a Modified Model has been formulated[13]. The centralized model was used for solving the purpose of load balancing initially. The aim in distributed system is to execute the process at minimum cost i.e. time is most important factor can be considered in cost calculation.

5. Comparison of Some Dynamic Load Balancing Algorithms:

Now some dynamic load balancing algorithms are studied below.

i. Nearest Neighbour Algorithm

With nearest neighbour algorithm each processor considers only its immediate neighbour processors to perform load balancing operations. A processor takes the balancing decision depending on the load it has and the load information to its immediate neighbours. By exchanging the load successively to the neighbouring nodes the system attains global balanced load state. The nearest neighbour algorithm is mainly divided into two categories which are diffusion method and dimension exchange method.

ii. Random (RAND) Algorithm

As soon as a workload (greater than threshold) is generated in a processor, it is migrated to a randomly selected neighbour. It does not check state information of anode. This algorithm neither maintains any local load information nor sends any load information to other processor. Furthermore, it is simple to design any easy to implement. But it causes considerable communication overheads due to the random selection of lightly loaded processor to the nearest neighbours.

iii. Adaptive Contracting with Neighbour (ACWN)

As soon as the workload is newly generated, it is migrated to the least loaded nearest neighbour processor. The load accepting processor keeps the load in its local heap. If the load is its heap is less than to its threshold load then no problem otherwise it sends the load to the neighbour processor which has load below the threshold load. So, ACWN in a respect that ACWN always finds the target node which is least loaded in neighbours[14].

iv. CYCLIC Algorithm

This is the outcome of RAND algorithm after slight modification. The workload is assigned to a remote system in a cyclic fashion. This algorithm remembers always the least system to which a process was sent.

v. Probabilistic

Each node keeps a load vector including the load of a subset of nets. The first half of the load vector holding also the local is sent periodically to a randomly selected node. Thus information is revised in this way and the information may be spread in the network without broadcasting. However, the quality of the algorithm is not ideal, its extensibility is poor and insertion is delayed.

vi. Distributed information and Distributed Decision

Each node in OFFER broadcasts its load situation periodically and each node keeps a global load vector. Performance of this algorithm is poor.

vii. The Shortest Expected Delay (SED)Strategy

These strategy efforts to minimize the expected delay of each job completion so the destination node will be selected in such a way that delay becomes minimal. This is a greedy approach in which each job does according to its best interest and joins the queue which can minimize the expected delay of completion. The average delay of a given batch of jobs with no further successive arrival is minimized by this approach. SED does not minimize the average delay for an on-going arrival process. To find out the destination node the source node has to get state information from other nodes for location policy.

viii. Greedy Throughput (GT) Strategy

This strategy is different from SED and NQ strategies. GT strategy deals with the throughput of the system that is the number of jobs completed per unit time would be maximum before the arrival of new job instead of maximizing only the throughput rate at the instant of balancing. This is why it is called Greedy Throughput (GT) policy.

S.N.	Algorithms	State Information	Performance
i.	NNA	Yes	Good
ii.	RAND	No	Excellent
iii.	ACWN	Yes	Good
iv.	CYCLIC	Partial	Slightly better than
v.	PROBABILISTIC	Partial	Good
vi.	OFFRE	Yes	Poor
vii.	SED	Yes	Good
viii.	GT	Yes	Good

Table 1: Comparison of Different Dynamic Load Balancing Algorithm

IV. Problem Statement

The main focus that we are giving here is to improve the quality of services in distributed system so that data packets requested by the users should take shortest path to travel from one node to another node, where when one user generally requests for required data or file from their respective servers provided that server must respond taking less time to avoid any kind of delay from server's side. But on the other hand, due to increase in number of users there is major challenge that we are facing is time delay problem as in very less time the servers should respond to every user efficiently within no time or taking very less time. This is how servers can provide better quality of services to their respective nodes.

Here come various qualities of services that we are investigating the following are:

1. Throughput
2. Jitter
3. Dropped packets
4. Out of order delivery

Here we are basically investigating the time delay problem so that every user can download its respective files from the servers in very less time.

V. Conclusion

In this paper we studied the load balancing strategies lucidly in detail. Distributed system using load balancing is the most thrust area in research today as the demand of heterogeneous computing due to the wide use of internet. Due to communication delay among servers, the load balancing process may be using out dated load information from local servers to compute the balancing flows. As we have shown, this would significantly affect the performance of the load balancing algorithm. We have just reviewed this work for our understanding. We have calculated the average data rate for the servers i.e. time under which they have to accomplish their work. The comparative study not only provides an insight view of the load balancing algorithms, but also offers practical guidelines to researchers in designing efficient load balancing algorithms for distributed computing systems.

VI. Future Work

A regeneration-theory approach is undertaken to analytically characterize the average overall completion time in a distributed system. The approach considers the heterogeneity in the processing rates if the nodes as well as the randomness in the delays imposed by the communication medium. The optimal one-shot load balancing policy is developed and subsequently extended to develop an autonomous and distributed load-balancing policy that can dynamically reallocate incoming external loads at each node. This adaptive and dynamic load-balancing is implemented and evaluated in a two-node distributed system. The performance of the proposed dynamic load-balancing policy is compared to that of static policies as well as existing dynamic load-balancing policies by considering the average completion time per task and the system processing rate in the presence of random arrivals of the external loads.So Here by we conclude that with the increase in number of users there is problem of time delay that arises,, we are investigating this time delay problem and improving it using the method called LOAD BALANCING method and algorithm.

References

[1]. Md. Firoj Ali and RafiqulZaman Khan. "The Study On Load Balancing Strategies In Distributed Computing System". International SJournalOf Computer Science & Engineering Survey (IJCSES) Vol.3, No.2, April 2012

[2]. Ahmad I., Ghafoor A. and Mehrotra K. "Performance Prediction of Distributed Load Balancing On Multicomputer Systems". ACM, 830-839, 1991.

[3]. Antonis K., Garofalakis J., Mourtos I. and spirakis P. "A Hierarchical Adaptive Distributed Algorithm for Load Balancing". Journal of Parallel and Distributed Computing, Elsevier Inc. 2003.

[4]. Kokilavani .K . "Enhance Load Rebalance Algorithm for Distributed File Systems in Clouds". International Journal of Engineering and Innovative Technology (IJEIT). Volume 3, Issue 6, December 2013.

[5]. M. Randles, D. Lamb, and A. Taleb-Bendiab, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing", Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, Perth, Australia, April 2010.

[6]. S.Penmatsa and T.Chronopoulos, Game-theoretic static load balancing for distributed systems, Journal of Parallel and Distributed Computing, vol.71, no.4, pp.537-555, Apr. 2011.

[7]. E. Altman, T. Basar, T. Jimenez, and N. Shimkin.Routing in two parallel links: Game- theoretic distributed algorithms. J. Parallel and distributed computing, 61(9):1367-1381, September 2001.

[8]. P. BeaulahSoundarabai, Sandhya Rani A., Ritesh Kumar Sahai, Thriveni J., K.R. Venugopaland L.M. Patnaik, "Comparative Study on Load Balancing Techniques in Distributed Systems",International Journal of Information Technology and Knowledge Management, pages 63(7), 2009.

[9]. Heiner Ackermann, Simon Fischer, and Martin Hoefer. Distributed algorithms for QoS load balancing. In Proc. 21st Symp. Parallelism in Algorithms and Architectures (SPAA), pages 197{203, 2009.

[10]. Heiner Ackermann, Simon Fischer, Martin Hoefer, Marcel Schongens, Distributed Algorithms for QoS Load Balancing.In Proc. 19th Symp. Discrete Algorithms (SODA), pages 314{322, 2008.

[11]. Petra Berenbrink, Tom Friedetzky, Leslie Ann Goldberg, Paul Goldberg, Zengjian Hu, and RusselMartin. Distributed sel_sh load balancing. SIAM J. Comput., 37(4):1163{1181, 2007.
[12]. Petra Berenbrink, Tom Friedetzky, ImanHajirasouliha, and Zengjian Hu. Convergence to equilibria indistributed, sel_sh reallocation processes with weighted tasks. In Proc. 15th European Symposium onAlgorithms (ESA), pages 41{52, 2007.
[13]. Petra Berenbrink, Martin Hoefer, and Thomas Sauerwald. Distributed sel_sh load balancing on networks.In Proc. 22nd Symp.Discrete Algorithms (SODA), 2011.
[14]. Benjamin Doerr and Leslie Goldberg. Drift analysis with tail bounds. In Proc. 11th Intl. Conf. ParallelProblem Solving From Nature (PPSN), volume 1, pages 174{183, 2010.