Journal of Advances and Scholarly Researches in Allied Education
Vol. 16, Issue No. 11, November-2019, ISSN 2230-7540

Node Clustering Distributed Networking Load Balancing

Ravi Shanker Sharma[1]* Gireesh Kumar Dixit[2]

[1] Research Scholar

[2] Department of Computer Science, Shyam University, Dausa, Rajasthan

Abstract – "Cloud computing" encompasses a wide range of technologies, including virtualization, distributed computing, networking, software, and online services. Customers, data centres, and distributed servers are all part of a cloud architecture.. On-demand services and low TCO are just a few of the features it offers, in addition to excellent uptime and fault tolerance. Developing a load balancing algorithm is the focus of these concerns. It is possible to have a lot or a little of each form of stress on the system. Dispersed systems can be more efficient if the burden is distributed among the system's nodes rather than concentrated on a few nodes that are overworked while others remain idle or perform very little work. Load balancing ensures that all system and network nodes are doing roughly the same amount of work at any one time. Techniques that are either sender initiated, receiver initiated, symmetric, static or dynamic, centralised or distributed are all feasible. Depending on the time of day, up to 80% of workstations are idle, making them a valuable resource. The idle time and compute resources of a processor can be used to reduce the cost of processing. Explain the concept of load balancing, different types of load balancing algorithms, and the various policies that can be used in load balancing algorithms in general, as well as provide an overview of various distributed load balancing algorithms that can be used in cloud environments. Our goal is to

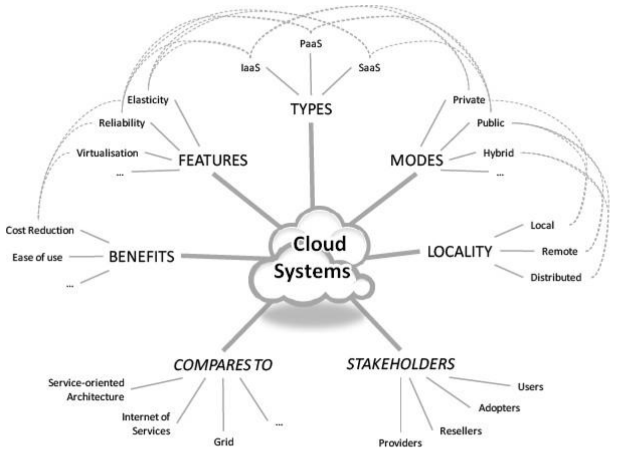
Key Words – Software as a Service, Platform as a Service ,sensors, IOT, load balancer, Cloud-Computing, Sharing-Resources

- X -

1. INTRODUCTION

Instead of using remote servers or local machines, Cloud computing allows services to be accessed from a wide range of resources. Instead of using remote servers or local workstations, Using cloud computing, services may be accessed from a variety of locations and devices. As far as I can tell, there isn't a standard way to describe cloud computing. In most cases, it is made up of a number of disparate components.

In a network, servers known as masters give requested services and resources to a variety of clients known as customers. Distributed computers offer on-demand services. Physical resources and software resources are only a few instances of services. Amazon EC2 is an example of a cloud computing service. [2]



Components of the Cloud

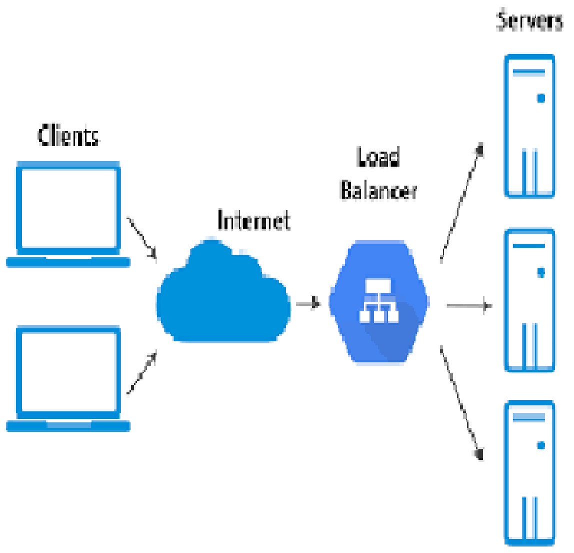
The three major components of a Cloud system are clients, datacenters, and distributed servers. Each component provides a distinct function and serves a certain goal.

The collective system load must be reassigned to the individual nodes in order to increase resource efficiency and task response time while also eliminating a scenario where some nodes are overloaded while others are underloaded. Instead of taking into consideration the system's past state or behaviour, a dynamic load balancing algorithm looks at how the system is behaving at the time in question. Load estimate and comparison, system stability and performance, node interaction, the nature of the task being transmitted and node selection are all important considerations when developing such an algorithm [13]. CPU consumption, memory usage, delay or network utilisation may all be used to quantify this strain. As long as a job is assigned to any of the available cluster nodes, it may be executed effectively. As a result, a strategy for selecting the nodes with these resources is required. Scheduling is a component or method that is in charge of determining which cluster node a certain process will be assigned to. The load balancing state will be investigated using this mechanism [9,10]. As a result, scheduling requires algorithms to overcome such issues. In the actual world, load balancing is primarily influenced by three elements [11]:

- the environment in which the load is to be balanced.
- the type of the load itself.

2. THE TOOLS FOR LOAD BALANCING THAT ARE AVAILABLE

In order to increase the overall system's resource efficiency and task response time, the task of redistributing the system's total load across the system's nodes is being carried out. A dynamic load balancing algorithm ignores the system's previous state or behaviour and instead focuses on the system's current behaviour. Load estimate and comparison, as well as stability and performance of systems and nodes as well as interactions between such nodes, are all important factors to take into account while creating such an algorithm [13]. CPU, memory, latency, and network utilisation can all be used to quantify the strain. When a workload is assigned to any cluster node, the available resources are fully utilised, and the workload can be completed quickly. As a result, a mechanism for determining which nodes have these resources is needed. Scheduling is a component or mechanism that determines which cluster node will be assigned to a specific process. This mechanism will be used to explore the load balancing situation [9,10]. As a result, scheduling necessitates the use of algorithms to address these challenges. Load balancing is generally influenced by three factors in the real world [11]:



- **The load must be balanced in a specific setting.**

Load balancing algorithms come in a variety of shapes and sizes.

Which of these three algorithms for load balancing is employed depends on who initiated the process [4].

Sender-Initiated Load Balancing Algorithm: If the sender initiates the load balancing procedure.

Receiver-Initiated Load Balancing Process: If the algorithm for load balancing is started by the receiver.

Symmetric: It is the result of both sender and recipient initiating the communication.

Load balancing algorithms can be classified into two groups based on the present condition of the system, as shown in [4]:

Static: It is independent of the system's current condition. It is necessary to have prior knowledge of the system.

Dynamic: The present status of the system is taken into account while making decisions about load balancing. Participation is open to everyone with no prior expertise. Because of this, a dynamic approach is recommended. Discussed here are a variety of dynamic load balancing methods for clouds of varied sizes.

3. ALGORITHM FOR DYNAMIC LOAD BALANCING

distributed and non-distributed methods of dynamic load balancing are available in a distributed system.

All nodes in the distributed system participate in the dynamic load balancing algorithm, which distributes the load balancing job. Cooperative and non-cooperative interactions among nodes can be used to achieve load balancing [4].

In the first, the nodes collaborate to achieve a common goal, such as improving the overall reaction time of the system. When Voml. 2oIssueo2f, Ftehberuary- 2013 nodes work independently with limited interactions with others, it is more favourable.

In a non-distributed system, load balancing is the responsibility of a single node or a group of nodes.

An algorithm that is both centrally and semi-distributed is available. A single system node does all of the work involved in load balancing in type one. The load balancing of the system is solely the responsibility of this node. It is just the core node that communicates with other nodes.

Semi-distributed load balancing is implemented in each cluster, with the system's nodes partitioned into clusters. Nodes in clusters are elected using an appropriate mechanism to achieve load balancing within each cluster. Thus, the cluster's core nodes are responsible for the entire system's load balancing [4].

In order to make an informed decision, centralised dynamic load balancing requires many fewer communications than semi-distributed load balancing. However, if the central node falls, the load balancing mechanism is rendered worthless in the case of centralised approaches. Small networks are more suited to the use of this method than larger ones.

4. DYNAMIC LOAD BALANCING POLICIES OR STRATEGIES

There are four policies in total [4]:

Move policies or Transfer strategies are algorithms that pick which workloads to transfer from a local node to a distant one.

It is determined by the selection policy which processors will participate in the load exchange

As part of the load balancing mechanism, location policy (or location strategy) is used to choose which node will receive the transferred job.

There are two ways to refer to the dynamic load balancing algorithm, one being "information policy" and the other being "information strategy."

5. LOAD BALANCING METRICS IN CLOUDS

Methods for optimising cloud load balancing now in use look at factors like throughput and response time as well as scalability and fault tolerance as well as migration time and overhead. But parameters like energy usage and carbon emissions must be taken into account for an efficient load balancing.

An algorithm's Overhead Associated parameter is used to determine the amount of overhead required to implement it. Task mobility and communication between processors and processes contribute to this overhead. Keeping this to a minimal is necessary to ensure that a load balancing approach works effectively.

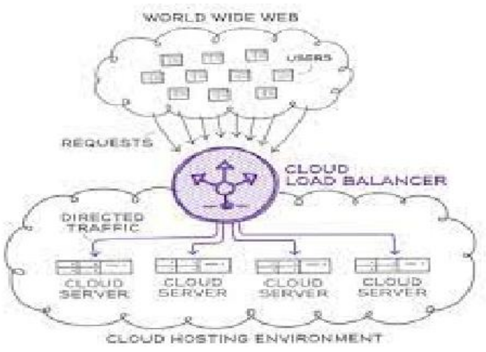
The term "throughput" refers to the number of tasks that have been completed. It should be set to a high value in order to improve the system's performance.

The term "performance" refers to the process of determining the system's efficiency. It must be enhanced at a fair cost, such as reducing reaction time while maintaining acceptable delays.

Resources are "utilised" in order to determine how well they are being used. To maximise efficiency, it should be optimised for load balancing.

Scalability refers to an algorithm's capacity to balance the load of a system with any number of nodes. This metric needs to be improved upon.

The response time of a distributed system's load balancing algorithm is referred to as Response Time. This parameter should be kept to a minimum.



default Tolerance refers to an algorithm's capacity to provide uniform load In the case of a node or connection failure, rebalancing is necessary. For load balancing to be effective, it must be both dependable and fault-tolerant.

The term "migration time" refers to the amount of time it takes to move tasks or resources across nodes. In order to improve the system's performance, it should be kept to a minimum.

The mVoroel and carbon emissions go hand in hand. The greater the amount of 2ndIsesueen2e, Frgebyruary-2013 ingested, the greater the carbon footprint. As a result, it should be lowered for an energy-efficient load balancing system. Load Balancing Types

Load balancing can be accomplished in two ways:

1) software load balancing and

2) hardware load balancing.

3) Load balancing hardware

Load Balancing in Software:

Software load balancing is placed in the application layer of the stack. Using software load balancing, an arbiter decides which engine should be used to process a given task, based on the software's recommendation. With a web application, you may implement software load balancing by using a proxy such as HA Proxy to divert some requests toward application server-x and some toward application server-y.

Load Balancing Hardware:

It is called hardware load balancing when many computers are used to distribute the work and raise efficiency, which in turn improves the program's speed. As a front, a load balancer software redirects requests to the most appropriate computer for processing. The price of hardware is now less than the need for performance in the age of the cloud. As a result, for applications that demand massive scalability, hardware load balancing is highly recommended.

6. CLOUD LOAD BALANCING VIA DISTRIBUTED LOAD BALANCING LOAD

balancing is extremely important in complicated and huge systems. The employment of techniques that operate on the components of the cloud in such a manner that the load on the entire cloud is balanced is one way to simplify global load balancing. The honeybee foraging method, biassed random sampling using a random walk approach, and Active Clustering are all examples of distributed system solutions [7]. Honeybee Foraging Algorithm

The habit of honey bees for discovering and reaping food inspired this algorithm. Forager bees are a type of bee that forages for food sources and then returns to the nest when they discover one.

The waggle dance was created by the beehive to promote this. Through the performance of this dance, the quality, quantity, and distance of the food from the beehive are communicated. As soon as the foragers arrived at the food supply, the scout bees immediately began collecting it. Waggle dances signal how much food is left, which leads to more food being exploited or abandoned, so they return to their hive and do it again.

Services are dynamically assigned to accommodate the changing demands of users as demand on a web server increases or falls. Virtual servers (VS) are used to consolidate the servers, and each VS has its own dedicated virtual service queue. A profit or an incentive is generated for each server that executes a request from its queue, as demonstrated by the bees' waggle dance. Using the CPU's processing time as an indicator is one approach to measure this incentive. It's like a billboard for honey bees as they dance on the dance floor. The colony's profit is also posted on this board.

As a forager or scout, each server takes on the function assigned to them. If a server makes money from a request, they may decide to promote their success on the message boards.

There are two ways that the server may demonstrate foraging/exploring behaviour: it can use probability calculations to choose which VS queue to use, or it can look for advertising and fulfil those requests. Px is established after taking into account the server's profit and comparing it to the colony's profit. Even if the profit was little, the server will likely be advertised for if the profit is big enough to keep it on the current virtual server. The server switches back to foraging or scouting mode if this value was too low.

1. Use random solutions to start the population.

2. Assess the population's fitness.

3. Creating a new population.

4. Pick a few sites for a neighbourhood search.

5. Recruit bees for certain locations and assess fitness.

6. Choose the most fit bee from each patch.

7. Using a random search, evaluate the fitness of the surviving bees.

8. Put an end to it.

Server Allocations Using the Honey Bee Technique of Foraging (adopted from [14])

Random Sampling with Bias

As each node in this virtual graph is connected to other nodes, this virtual graph represents the amount of demand on the server. An individual server's free resources are represented by the graph as degrees at each node.

By eliminating an incoming edge, the amount of available resources is reduced when the task performed by the node is completed. One way to tell when free resources are becoming more readily available is when the node makes an incoming edge. To add and delete processes from the database, random sampling is performed.

Any node can be the starting point of the walk, and a random neighbour will be chosen at each step. Load is distributed to the last node. Alternatively, another approach for picking a node for load allocation can be utilised, which involves selecting a node based on particular factors such as computational efficiency, etc. Another option is to allocate load to the node that is the most underloaded, i.e. the one with the highest degree. If b is the walking distance, then increasing b enhances load allocation efficiency. For practical purposes, we set a b-threshold value that is approximately equal to log n.

It is only possible for a node to do a task if its current walk time is greater than or equal to the job's length.

As a criteria Otherwise, the job's travel distance is lengthened and a new neighbour node is randomly selected. When a node completes a task, it removes an incoming edge from the graph. Edges are then formed between the two nodes that started and finished the load allocation operation after the task is done.

We end up with a graph that is directed. This method of load balancing is totally decentralized, making it ideal for large network systems like those seen in cloud computing. Involved clustering Active Clustering is based on the idea of grouping related nodes together and focusing on them.

The procedure is as follows:

Once a node has found the right matchmaker node, it moves on to the next step in the process by selecting one of its neighbours that is distinct from the prior one. Using the so-called "matchmaker node," the starting node is connected to its neighbours who are also of the same kind as the initial node.

The link between the matchmaker node and the starting node is then severed.

The processes listed above are repeated iteratively.

7. CONCLUSION

An overview of distributed load balancing algorithms for cloud computing is provided in this article, In this chapter, we'll look at how dynamic load balancing algorithms work, the many types of load balancing algorithms that exist, and the various rules that may be used in each.

REFERENCES

1. Armbrust, M., A. Fox, R., Griffith, A.D., Joseph, R., Katz, A., Konwinski, et al. (2010)―A View of Cloud Computing,‖ Communications of the ACM (53)4, pp. 50–58.

2. Andrew Downie (2008-04-21). "The World's Worst Traffic Jams". Time. Retrieved 2008- 06-20.

3. Aslam, U., I. Ullah, and S. Ansari (2010)―Open Source Private Cloud Computing,‖ Interdisciplinary Journal of Contemporary Research In Business (2)7, p. 399.

4. Avetisyan, A.I., R. Campbell, I. Gupta, M.T. Heath, S.Y. Ko, G.R. Ganger, et al. (2010)―Open Cirrus: A Global Cloud Computing Tested,‖ Computer (43)4, pp. 35–43.

5. Banerjee, P., R. Friedrich, C. Bash, P. Goldsack, B.A. Huberman, J. Manley, et al. (2011) ―Everything as a Service: Powering the New Information Economy,‖ Computer (44)3, pp. 36–43.

6. Barki, H., S. Rivard, and J. Talbot (1993) ―A Keyword Classification Scheme for IS Research Literature: An Update,‖ MIS Quarterly, June, pp. 209–225.

7. Barnhill, D.S. (2010) ―Cloud Computing and Stored Communications: Another Look at Z. Shi, *Advanced Artificial Intelligence*, World Scientific, Singapore, 2011.

8. M. S. Ali, M. Adnan, S. M. Noman, and S. F. A. Baqueri, “Estimation of traffic congestion cost-A case study of a major arterial in karachi,” *Procedia Engineering*, vol. 77, pp. 37–44, 2014.View at: Publisher Site | Google Scholar

9. W. Cao and J. Wang, “Research on traffic flow congestion based on Mamdani fuzzy system,” *AIP Conference Proceedings*, vol. 2073, 2019.View at: Publisher Site | Google Scholar

10. X. Kong, Z. Xu, G. Shen, J. Wang, Q. Yang, and B. Zhang, “Urban traffic congestion estimation and prediction based on floating car trajectory data,” *Future Generation Computer Systems*, vol. 61, pp. 97–107, 2016.View at: Publisher Site | Google Scholar

11. Q. Yang, J. Wang, X. Song, X. Kong, Z. Xu, and B. Zhang, “Urban traffic congestion prediction using floating car trajectory data,” in *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing*, pp. 18–30, Springer, Zhangjiajie, China, November 2015.View at: Google Scholar

12. W. Zhang, Y. Yu, Y. Qi, F. Shu, and Y. Wang, “Short-term traffic flow prediction based on spatio-temporal analysis and CNN deep learning,” *Transportmetrica A: Transport Science*, vol. 15, no. 2, pp. 1688–1711, 2019.View at: Publisher Site | Google Scholar

13. T. Adetiloye and A. Awasthi, “Multimodal big data fusion for traffic congestion prediction,” *Multimodal Analytics for Next-Generation Big Data Technologies and Applications*, Springer, Berlin, Germany, 2019.View at: Publisher Site | Google Scholar

14. F. Wen, G. Zhang, L. Sun, X. Wang, and X. Xu, “A hybrid temporal association rules mining method for traffic congestion prediction,” *Computers & Industrial Engineering*, vol. 130, pp. 779–787, 2019.View at: Publisher Site | Google Scholar

15. J. Wang, Y. Mao, J. Li, Z. Xiong, and W.-X. Wang, “Predictability of road traffic and Congestion in urban areas,” *PLoS One*, vol. 10, no. 4, Article ID e0121825, 2015.View at: Publisher Site | Google Scholar

16. Z. He, G. Qi, L. Lu, and Y. Chen, “Network-wide identification of turn-level intersection congestion using only low-frequency probe vehicle data,” *Transportation Research Part C: Emerging Technologies*, vol. 108, pp. 320–339, 2019.View at: Publisher Site | Google Scholar

17. K. M. Nadeem and T. P. Fowdur, “Performance analysis of a real-time adaptive prediction algorithm for traffic congestion,” *Journal of Information and Communication Technology*, vol. 17, no. 3, pp. 493–511, 2018.View at: Publisher Site | Google Scholar

18. H. Zhao, X. Jizhe, L. Fan, L. Zhen, and L. Qingquan, “A peak traffic Congestion prediction method based on bus driving time,” *Entropy*, vol. 21, no. 7, p. 709, 2019.

Corresponding Author

Ravi Shanker Sharma*

Research Scholar