IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 3, March 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203070 |

Taxonomy of Load Balancing Strategies in Distributed Systems

Harsh Bipinbhai Sakariya, Dr Ganesh D

PG Student, School of CS & IT, Jain(Deemed-to-be-University), Bengaluru, India

Professor, School of CS & IT, Jain(Deemed-to-be-University), Bengaluru, India

ABSTRACT: Large-scale parallel and distributed computing systems are becoming more popular as a result of falling hardware prices and improvements in computer networking technologies. Improved performance and resource sharing are potential benefits of distributed computing systems. We have provided a summary of distributed computing in this essay. The differences between parallel and distributed computing, terms related to distributed computing, task distribution in distributed computing, performance metrics in distributed computing systems, parallel distributed algorithm models, benefits of distributed computing, and distributed computing's application domain were all covered in this paper.

KEYWORDS Distributed System, Load balancing Algorithms, Throughput

I. INTRODUCTION

When two or more computers are networked together to share a single computing task, this is referred to as distributed computing. Distributed computing aims to divide the workload among several computers. In the sense that the processing nodes, network architecture, communication medium, operating system, etc [3]. may differ in different networks that are dispersed far around the world, distributed networks are primarily heterogeneous in nature. The distributed computing system is currently constructed using connections between several hundred PCs. The total work load must be divided across the nodes in the network in order to maximise system efficiency [2]. Therefore, the existence of distributed memory multiprocessor computing systems led to the issue of load balancing being well-known. There will be some fast and some slow computer nodes in the network. If processing speed and connection speed (bandwidth) are not taken into consideration, the network's slowest functioning node will have an impact on the system's overall performance. Therefore, load balancing solutions maintain a balance between the loads on the nodes by preventing certain nodes from being idle while the others are overloaded. Additionally, load balancing techniques eliminate any node's run-time inactivity.

II. DIFFERENCES BETWEEN PARALLEL AND DISTRIBUTED COMPUTING

Although there are many parallels between parallel and distributed computing, there are also some significant distinctions in terms of computation, expense, and turnaround time. While distributed computing separates an application into jobs that may be executed at many locations utilising the available networks connected together, parallel computing actually divides an application into small enough tasks that can be executed concurrently [8]. In parallel computing, several processing units coexist on a single machine, each of which is concurrently dedicated to the total system. However, with distributed computing, a number of distinct nodes, each with a potential difference in nature, work together through a network to contribute processing cycles to the system as a whole. While distributed computing makes use of already existing individual machines that are reasonably priced on today's market, parallel computing requires expensive parallel gear to coordinate several processors on the same system.

III. LOAD BALANCING

The technique of spreading load units (jobs or tasks) among a group of processors connected to a network that may be dispersed throughout the world is known as load balancing. Other processors with loads below the threshold load take on the extra or unfinished load from one processor [9]. When a CPU is under threshold load, it is capable of handling any additional load. There is a very high likelihood that certain nodes in a system with several nodes will be idle while

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203070 |

the others will be overloaded. Therefore, the processors in a system can be classified as strongly loaded (enough jobs are waiting to be executed), lightly loaded (fewer jobs are waiting), and idle (no jobs to do) depending on their current load [10]. It is possible to make every processor equally occupied and to accomplish the tasks roughly at the same time by using a load balancing approach.

Three rules make up an operation that balances loads. They are the distribution rule, the selection rule, and the location rule. The selection rule operates in a nonpreemptive or preemptive manner. While the running process may be picked up by the preemptive rule, the newly formed process is always picked up by the non-preemptive rule. Preemptive transfers are more expensive than non-preemptive transfers, which are the better option. Preemptive transfer, however, can occasionally be superior to non-preemptive transfer [12]. Practically, location and distribution rules work together to decide on load balancing. There are two categories of balancing domains: local and global. In the local domain, the balancing decision is made by the closest neighbours by exchanging local workload data, whereas in the global domain, transfer partners are activated throughout the entire system and work load data is exchanged internationally.

Benefits of Load Balancing Algorithms

a) Load balancing improves the performance of each node and hence the overall system performance

b) Load balancing reduces the job idle time

c) Small jobs do not suffer from long starvation

d) Maximum utilization of resources

e) Response time becomes shorter

f) Higher throughput

g) Higher reliability

h) Low cost but high gain

i) Extensibility and incremental growth

The load balancing approach becomes a focus of intense research because of the aforementioned advantages.

Over the past few years, a large number of load balancing algorithms have been created, however no single technique is suitable for all applications [11]. The choice of a suitable load balancing depends on hardware aspects like communication overheads as well as application parameters like balancing quality and load production patterns. In general, there are two types of load balancing algorithms: static load balancing and dynamic load balancing.

Static Load Balancing

In a static method, the processes are distributed among the processors at build time based on the nodes' performance. The processes cannot be changed or reassigned at run time once they have been assigned. Each node's number of jobs is fixed in the static load balancing mechanism. Static algorithms do not gather any node-specific data. The incoming time, the amount of resources required, the mean execution time, and inter-process communications are taken into consideration when assigning jobs to processing nodes [12]. Static load balance is also known as a probabilistic method since these factors need to be measured prior to the assignment. As there is no job migration during runtime, there is either no overhead or very little overhead. In static load balancing, it has been found that the load balancing improves as the number of jobs exceeds the number of processors [11].

Local jobs are assigned according to a schematic diagram of static load balancing in Figure. Either a job can be transferred to a distant node or it can be assigned from the assignment queue to the threshold queue. A remote node job will also be allocated to the threshold queue. A job cannot be transferred to another node once it has been put to a threshold queue. Any node in the communication network can process a job that arrives there, or it can send it to another node for remote processing. The static load balancing algorithms can be divided into two sub classes: optimal static load balancing and sub optimal static load balancing [12].

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203070 |

Optimal Static Load Balancing Algorithm

The best static load balancing may be carried out if all of the resources and information pertaining to a system are understood. By using the best load balancing algorithm, a system's throughput can be increased while the use of its resources is maximised. Examples of optimisation techniques include simulated annealing (SA) and genetic algorithms (GA) [10].

Sub-Optimal Static Load Balancing Algorithm

When an ideal solution cannot be discovered, sub-optimal load balancing algorithms will be required for some applications. For sub-optimal algorithms, the thumb-rule and heuristics methods are crucial.

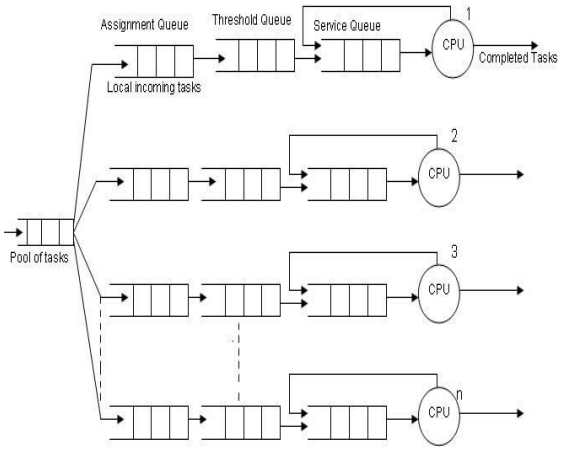

Fig 1: Pool of Tasks and Process Queues

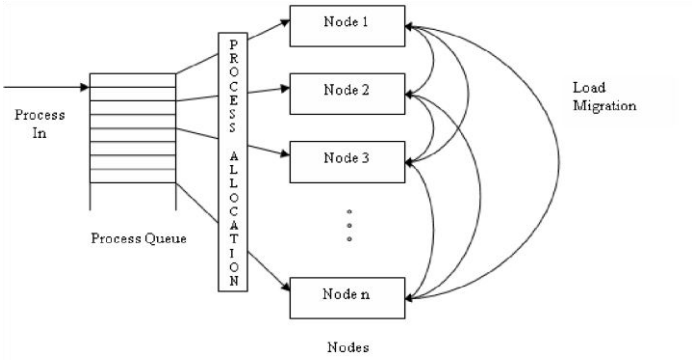

Fig 2: Dynamic Load Balancing

Dynamic Load Balancing

In order to execute the static load balancing, too much information about the system and the jobs must be known. These details might not be accessible beforehand. a detailed analysis of the system's current state and the laborious tasks' preliminary approach. Thus, the dynamic load balancing algorithm was created. Job assignments are made in real time. According to the situation, the load will be moved from strongly loaded nodes to lightly loaded nodes in DLB jobs that are reassigned at runtime [12]. In this situation, communication bottlenecks develop and become worse as the number of processors rises. No choice is made in dynamic load balancing until the procedure is put into action. This tactic gathers data on both the job details and the system condition. An algorithm may be able to make better decisions when more data is gathered by it in a short period of time [11]. Because heterogeneous systems have nodes with varying speeds, varied communication link speeds, different memory capacities, and variable external loads as a result of the many, dynamic load balancing is typically taken into account in these systems. For a system to work well, numerous load balancing solutions have been created and categorised to date. A straightforward dynamic load balancing for

moving jobs from strongly loaded to less loaded nodes is shown in Figure.

Dynamic Load Balancing Strategy

Bidding Strategy

The overburdened processor asks other processors to submit their bids so that it can get assistance with some of its chores. The status of each processor's work load is included in the bid data [8]. The originating processor chooses which processors it will assign parts of its jobs to for completion after receiving all of the bids from involved processors. One significant disadvantage of this approach is the potential for one processor to experience overload as a result of consistently winning bids. The amount of information provided, the choice of offers, and communication overhead all affect how well this method performs.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203070 |

Drafting Strategy

Instead of uniformly spreading the workload across participating processors, the drafting policy accomplishes load balance by keeping every CPU active.

Instead of being triggered by a processor being overloaded as in the bidding technique, lightly laden processors start a process migration.

Each processor keeps track of its workload and indicates whether it is in the H-, N-, or L-load state. When a processor has a heavy load, or H-load, some of its processes may be able to move to another CPU. A normal load, or N-Load, denotes the absence of any plans for process movement [8]. A processor with an L-load, or light-load, is one that will accommodate some migrant processes. Each CPU uses a load table to store this information about other processors and serve as a billboard from which to access the system's overall information. When a processor's load changes, it broadcasts this information to neighbouring nodes, causing them to update their load tables.

The processor will identify additional processors holding the state of H-Load from its load database as it becomes lightly loaded, or L-Load, and send them a draft-request message. The drafting processor is ready to take on extra work, according to this notification [10]. Each remote (drafted) processor will respond by sending a draft-respond message that contains a draft-age information if, at the time it receives this message, it is still in the H-Load state.

The benefit of using this method is that it minimises a number of issues with the bidding algorithm, such as pointless communication messages and the potential for a bid winning processor to become overloaded.

Threshold Strategy

a criterion value T determines whether a task is carried out locally or remotely. The processor's queue length is the threshold.

Until this processor threshold is achieved, a processor will try to locally perform a newly arriving task. This CPU will then randomly choose another processor and probe it to see if doing the work on that processor will push it over the threshold or not. If not, the work is passed to the processor being probed, who must complete it there without attempting to pass it to an other processor. Another processor is chosen in the same way if it does raise it above the threshold [8]. The probing limit is the upper limit at which this process can continue. The work is then carried out locally [5] if no destination processor is discovered after that. It should be emphasised that the threshold technique makes decisions about whether to transmit a task or not without requiring any communication between processors.

Advantage-

It minimizes communication overhead. Another advantage is that the threshold policy avoids extra transfer of tasks to processors that are above the threshold.

Greedy Strategy

The current state of each processor P in this method is represented by the function f(n), where n is the total number of tasks being handled by each processor. A processor searches for a remote processor whose state is less than or equal to f(n) if a task arrives at P and the number of tasks n is more than zero. The task is sent to a distant processor if one is discovered that has this characteristic [8]. The function f(n) that is chosen will determine how well this technique works.

The greedy method has the advantage of using a cyclic probing process rather than the random selection employed by the threshold strategy. In this cyclic probing approach, processor i searches for a suitable destination processor by probing processor (i+j) mod N, where N is the system's total number of processors.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203070 |

Load Balancing Algorithm

Round Robin load balancing method

The simplest and most widely used load balancing algorithm is round robin. Application servers receive client requests in a straightforward rotation. The first client request is sent to the first application server in the list, the second client request is sent to the second application server, the third client request is sent to the third application server, the fourth client request is sent to the first application server, and so on [9].

Preference- When the number of compute nodes are fixed and all have the same capability then this Algorithm may perfrom better.

Advantage- It is very easy to implement

Disadvantage- It is only distributing request evenly not workload. So it might cause problem When number of nodes are not fixed.

Least Connection load balancing method

A dynamic load balancing mechanism called least connection load balancing distributes client requests to the application server that has the fewest active connections at the moment the client request is received. This technique takes the active connection load into account [9]. When application servers have identical specs, one server may be overwhelmed due to longer-lasting connections.

Preference- When we are not sure about the processing time of each request.

Advantage- We don't need to know about the capability of computation power. So implementation will be easy.

Disadvantage- There is not any major disadvantage.

Weighted Least Connection load balancing method

It is built on the least connection load balancing algorithm. We need to assign weight to each computing node [10].

Preference- Same as least connection load balancing.

Advantage- More effective distribution of the work load.

Disadvantage- One problem will be there which is how to assign weight to each compute node.

Source IP Hash load balancing method

The source and destination IP addresses of the client request are utilised in the source IP hash load balancing technique to create a special hash key that is used to assign the client to a certain server [11].

Preference- It is useful when particular user need to be directed to the same server after session is broken.

Advantage- It always redirects the client to the same compute node.

Disadvantage- It does not balance the work load effectively.

Performance Parameters in Distributed Computing

There are many performance parameters which are mostly used for measuring parallel computing performance. Some of them are listed as follows:

Execution Time

Execution time is the amount of time needed to finish an application from the time it is sent to a computer. The execution time is known as serial execution time when the application is submitted to a serial computer and is denoted by TS [4]. The execution time is known as parallel execution time when the programme is submitted to a parallel computer and is denoted by TP.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203070 |

Throughput

It is defined as the quantity of tasks finished in a certain amount of time. The throughput varies with the size of the jobs. For large processes, the throughput might be one per hour, whereas for minor processes, it might be twenty per second. The underlying architecture and the size of the active processes on that architecture are completely deterministic [5].

Speed Up

The speedup of a parallel algorithm is the ratio of the algorithm's execution time when run sequentially to its execution time when run simultaneously by many processors. The formula for speed increase is: Sp=Ts /Tp, where Ts stands for sequential execution time and Tp for parallel execution time. In an ideal scenario, the speedup is equal to the number of processors running in parallel, but in practise, the speedup is never as fast as it could be due to other key cluster-related issues as communication and memory access delays [5].

Efficiency

It is a measurement of a processor's parallel contribution to an algorithm. Ep= Sp/p (0>Ep1), where Sp is the speed up and p is the number of parallel processors, can be used to calculate efficiency. An effective algorithm is shown by the value of Ep being close to 1 [5].

System Utilization

This is an extremely crucial variable. The engagement of resources inside a system is measured by system utilisation. The range could be anywhere from 0% to 100% [4].

Turnaround Time

It is described as the amount of time the job took to complete from the time it was submitted. The total turnaround time includes the time it takes to enter memory, wait in the ready queue, execute on the processor, and perform input/output activities [7].

Waiting Time

It is the entire amount of time a processor waits in queue to access a resource. In other terms, waiting time refers to the amount of time a process spends waiting for a resource to become available [6]. The same factors that affect turnaround time also affect waiting times.

Response Time

Response time is the period of time between a request being sent and receiving the first response. The computing system's output devices have the ability to limit this time.

Overheads

A single function known as the overhead function expresses the overheads provided by a parallel programme. The symbol To is used to represent an overhead function in a parallel system [5]. When all processing components are added together, the total overhead in addressing a problem is pTP. As a result, (1), where TS time is free from overhead, is the formula for the overhead function (To).To equal pTP-TS (1)

Reliability

Reliability ensures operations without fail under any specified conditions for a definite period of time

IV. CONCLUSION

An overview of distributed systems is provided in this work. The strategies for load balancing are described. The most crucial component of distributed systems is load balancing. The performance of the computing system increases with an efficient load balancing method. The various dynamic load balancing algorithms are described, as well as the various dynamic load balancing strategy types. There are many factors that must be taken into account while creating a distributed system.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203070 |

REFERENCES

1. Dynamic Load Balancing in Distributed Systems in the Presence of Delays: A Regeneration-Theory Approach Sagar Dhakal, Majeed M. Hayat, Senior Member, IEEE, Jorge E. Pezoa, Cundong Yang, and David A. Bader, Senior Member, IEEE IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 18, NO. 4, APRIL 2007
2. G. Pallis, "Cloud Computing: The New Frontier of Internet Computing", IEEE Journal of Internet Computing, Vol. 14, No. 5, September/October 2010, pages 70-73
3. A. Khiyati, M. Zbakh, H. El Bakkali, D. El Kettani "Load Balancing Cloud Computing: State Of Art", IEEE, 2012.
4. Mladen A. Vouk, "Cloud Computing – Issues, Research and Implementations", Proceedings of the ITI 2008 30th Int. Conf. on Information Technology Interfaces, June 23-26, 2008, Cavtat, Croatia
5. J. Sahoo, S. Mohapatra and R. lath "Virtualization: A survey on concepts, taxonomy and associated security issues" computer and network technology (ICCNT), IEEE, pp. 222-226. April 2010
6. Giuseppe Aceto, Alessio Botta, Walter de Donato, Antonio Pescapè, "Cloud monitoring: A survey", SciVerse ScienceDirect's Computer Networks 57, PP- 2093–2115, ASOC 1894 1–12, © 2013 Elsevier B.V
7. Roughgarden T, Tardos E. How bad is selfish routing? Proceedings of the 41st IEEE Symposium on Foundations of Computer Science, Redondo Beach, CA, November 2000; 93-102
8. Bharadwaj v, Ghose D , Mani V, Robertazzi TG Scheduling Divisible Loads in Parallel and Distributed Systems. IEEE Computer Society Press: Loas Alamitos, CA, U.S.A., 1996
9. A modified round-robin load-balancing algorithm for cluster-based web servers IEEE, Zongyu Xu ; Xingxuan Wang
10. Sidra Aslam Munam, Ali Shah "Load Balancing Algorithms in Cloud Computing: A Survey of Modern Techniques" 2015 National Software Engineering Conference (NSEC 2015),IEEE, pp:30-35
11. Shinde, Vinayak D., Anas Sami Ahmed Dange and Muhib Lambay. "Load Balancing Algorithms in Cloud Computing." (2016).
12. Load Balancing Algorithm Based on Estimating Finish Time of Services in Cloud Computing Nguyen Khac Chien, Nguyen Hong Son, Ho Dac Loc , 2016 18th International Conference on Advance Communication Technology (ICACT)

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.379

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details