Study on Dynamic Load Balancing in Distributed System

Kaushal M. Madhu

ME student, CSE Department, Government Engineering College, Sector28, Gandhinagar, Gujarat, India

S. M. Shah

Associate Professor, CSE Department, Government Engineering College, Sector28, Gandhinagar, Gujarat, India

Abstract

Load balancing is the process of improving the performance of a parallel and distributed system through a redistribution of load among the processor [3]. The main goal of load balancing is to equalize the workload among the nodes by minimizing execution time, minimizing communication delays, maximizing resource utilization and maximizing throughput. The scheduling in distributed system is NP-complete problem even in best conditions, and methods based on heuristic search have been proposed to obtain optimal and suboptimal solutions [2]. This paper presents issues that need to be considered in the development of a dynamic load balancing algorithm.

Keywords: Load Balancing, Distributed system, Scheduler, SLB, DLB, stability

I. Introduction

Load balancing is a mechanism that enables jobs to move from one computer to another within the distributed system. Load balancing is the process of roughly equalizing the work load among all nodes of the distributed system. It strives to produce a global improvement in system performance. In this manner, load balancing goes one step further than load sharing, which only avoids having some nodes idle in the distributed system when other nodes have too much work. Load balancing has been found by to further reduce the mean and standard deviation of task response times more than load sharing would [1].

Some of the main goals of a load balancing algorithm are:

1) To achieve a greater overall improvement in system performance at a reasonable cost.
2) To treat all jobs in the system equally regardless of their origin.
3) To have a fault tolerance.
4) To have the ability to modify itself in accordance with any changes or expand in the distributed system configuration.
5) To maintain system stability. [1]

II. Types of Load Balancing Algorithm

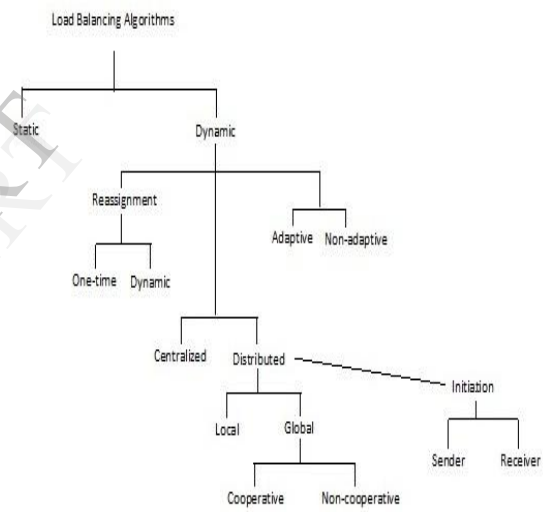

Fig-1 classification of load balancing algorithm [2]

A) Static v/s Dynamic

Static LBA use only information about the average behaviour of the system while dynamic LBA react to the system state that changes dynamically.

B) Distributed v/s Centralized

In the centralized LBA, the responsibility of scheduling physically resides on a single node.

In the distributed LBA, process assignment decisions are physically distributed among the various nodes of the system.

C) Local v/s Global

In the local LBA, each processor polls other processors in its neighbourhood and uses this

local information to decide upon a load transfer. The primary objective is to minimize remote communication as well as efficiently balance the load on the processors.

In the global LBA, global information of all or part of the system is used to initiate the load balancing. This scheme requires a considerable amount of information to be exchanged in the system which may affect its scalability.

D) Cooperative v/s Non-cooperative

In the non-cooperative LBA, each node has autonomy over its own resource scheduling. That is why the node may migrate or allocate tasks based on local performance.

In the cooperative LBA, processes work together toward a common system-wide global balance. Scheduling decisions are made after considering their effects on some global effective measures.

E) One-Time Assignment v/s Dynamic Assignment

In the one-time assignment LBA, a task may be dynamically done but once it is scheduled to a given processor, it can never be rescheduled to another one.

In the dynamic assignment LBA, jobs can migrate from one node to another even after the initial placement is made.

F) Sender/Receiver/Symmetrical Initiated

In the sender initiated LBA, the overloaded nodes transfer one or more of their tasks to several under loaded nodes.

In the receiver initiated LBA, under loaded nodes request tasks to be sent to them from nodes with higher loads.

In the symmetric LBA, both the under-loaded as well as the over loaded nodes may initiate load transfers.

G) Adaptive v/s Non-adaptive

In the adaptive LBA, scheduled decisions take into consideration past and current system performance and are affected by previous decisions or changes in the environment.

In the non-adaptive LBA, parameters used in scheduling remain the same regardless of system's past behaviour.

III. Components of Dynamic Load Balancing Algorithm [1]

A DLBA is required to make load distribution decisions based on the current work load at each node of the distributed system. A DLBA has three main components: Information Strategy, Transfer Strategy and Location Strategy

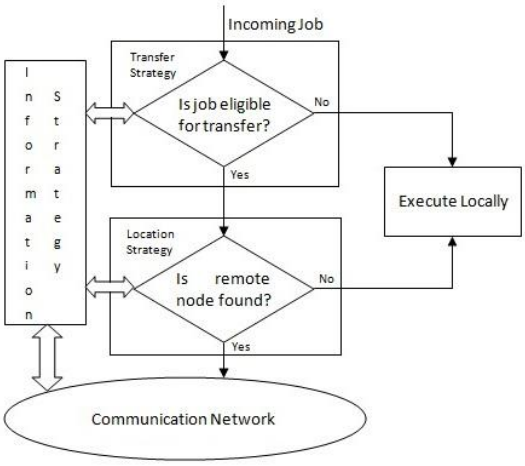

Fig-2 Interaction among components of a dynamic load balancing algorithm [1]

As shown in Fig. 2, incoming jobs are intercepted by the transfer strategy which decides whether or not it should be transferred to a remote node for the purpose of load balancing. If the transfer strategy decides that a job should be transferred, the location strategy is triggered in order to find a remote node for the job. Information strategy provides both transfer and location strategies with the necessary information to build their decisions. Now let them discuss in detail:

A) Information Strategy:

The part of a dynamic load balancing responsible for collecting information about nodes in the system is referred to as information strategy. It is responsible for providing location and transfer strategies at each node with the necessary information needed to make their load balancing decisions. It keeps all nodes of the distributed system updated on the global system state but generates extra traffic and hence increases the overhead generated by the algorithm. Therefore, there is a trade-off between the amount of information exchanged and the frequency of the exchange of this information.

B) Transfer Strategy:

The part of a dynamic load balancing algorithm which selects a job for transfer from a local node to a remote node is referred to as transfer strategy.

Two main issues concerning load balancing activity that depend on the transfer strategy employed are: (1) when is the right time to start it and (2) what jobs are subjected to it. Two approaches are commonly used to start the load balancing activity: the time a new job arrives or is

created at a node and the time a finished job departs from a node.

Algorithms which make load balancing decisions at the arrival or creation of a new job are referred to as sender-initiated, while algorithms which make load balancing decisions at the departure of a finished job are referred to as receiver-initiated. It has been shown sender-initiated algorithms are suitable when the system is light to moderately loaded while receiver-initiated algorithms are suitable when the system is heavily loaded.

Two approaches that determine which jobs are eligible for transfer are: consider-new-only and consider-all. The consider-new-only approach, only considers newly arrived or created jobs for load balancing. The consider-all approach considers all jobs eligible for load balancing. This approach is more complex than the previous one because it employs an extra mechanism for the selection of the appropriate job out of a set of active jobs.

C) Location Strategy:

The part of a dynamic load balancing algorithm which selects a destination node for a transferred task is referred to as location strategy. One of the main decisions performed by a load balancing algorithm is the selection of a destination node for a job transferred for load balancing. This decision represents the sole purpose for load balancing: a heavily loaded node tries to find a lightly loaded node to help in executing some of its jobs. This decision is performed by the location strategy. The selection of a remote node is based on the current work load present at that node. Some of the approaches used to select a destination node for a transferred job are:

1) Random: Under a random location strategy, a local node selects a remote node randomly and transfers the job there for execution.

2) Probing: Location strategies which employ probing work as follows: a local node selects a random subset of nodes and polls them to find a suitable destination node for a transferred job. A suitable node is the one which will provide the transferred task with a better service.

3) Negotiation: Under this location strategy, which is usually applied in distributed dynamic algorithms, nodes negotiate with each other for load balancing purposes in order to select a suitable destination node for transferred jobs.

D) Other Issues:

1) Load Measurement: DLB depends on the current work load in the system. For this reason, one important parameter used by a DLBA is the load descriptor. It employs to define the work load present at each node of the system. Some load descriptors are: CPU queue length, CPU utilization, job resource requirements, context switch rate, percentage of idle CPU time, and the amount of unfinished work at a node. CPU queue length is believed to be a good load descriptor because it gives a good estimate of job response time. The advantage of queue length as a load descriptor is the simplicity to obtain its value.

2) Performance Measurement: A LBA should adopt a performance index by which this performance improvement is measured. Since there is more than one index that can be utilized, the selection usually differs from one algorithm to another. A performance index could be system performance oriented, user-oriented, or both.

3) System Stability: It is very important that a DLBA maintain stability in the distributed system. A load balancing algorithm is stable if it does not cause thrashing, if the load on any two nodes of the distributed system does not differ by some threshold and if the response time to any sudden arrival burst does not exceed a certain limit.

IV. Dynamic Load Balancing Algorithms [8]

A) Random (RAND) Algorithm

As soon as a workload (greater than threshold load) is generated in a processor, it is migrated to a randomly selected neighbour. It does not check state information of a node. This algorithm neither maintains any local load information nor sends any load information to other processors.

It is simple to design and easy to implement. But it causes considerable communication overheads due to the random selection of lightly loaded processor to the nearest neighbours.

B) Prioritized Random (PRAND) Algorithm

Modification is done on RAND and ACWN for the non-uniform workload to get prioritized RAND (PRAND) and prioritized ACWN (PACWN) respectively. In these algorithms the work loads are assigned index numbers on the basis of the weight of their heaps.

PRAND is similar to RAND except that it selects the second largest weighted load from the heap and transfers it to a randomly selected neighbour. On the other hand, PACWN selects the second largest weighted workload and transfer it to the least loaded neighbour.

C) 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 in 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 does require maintaining the local load information and also the load information of the neighbours for exchanging the load periodically. Hence, RAND is different from the ACWN in a respect that ACWN always finds the target node which is least loaded in neighbours.

D) 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 last system to which a process was sent.

E) Probabilistic Algorithm

Each node keeps a load vector including the load of a subset of nodes. The first half of the load vector holding also the local load 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 this algorithm is not ideal, its extensibility is poor and insertion is delayed.

F) Threshold and Least

They both use a partial knowledge obtained by message exchanges. A node is randomly selected for accepting a migrated load in Threshold. If the load is below threshold load, the load accepted by there. Otherwise, polling is repeated with another node for finding appropriate node for transferring the load. After a maximum number of attempts if no proper recipient has been reported, the process is executed locally.

Least is an instant of Threshold and after polling least loaded machine is chosen for receiving the migrated load.

Threshold and Least have good performance and are of simple in nature. Furthermore, up-to-date load values are used by these algorithms.

G) Reception Algorithm

In this algorithm, nodes having below the threshold load find the overloaded node by random polling for migrating load from overloaded node.

H) Shortest Expected Delay (SED) Algorithm

In this algorithm, efforts to minimize the expected delay of each job completion so the destination node will be selected in such a way that the 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 ongoing arrival process. To find out the destination node the source node has to get state information from other nodes for location policy.

I) Never Queue (NQ) Algorithm

NQ algorithm is a separable strategy, in which the sending server estimates the cost of sending a job to each final destination or a subset of final destinations and the job is placed on the server with minimal cost. This algorithm always places a job to a fastest available server. This algorithm minimizes the extra delay into successive arriving jobs so that the overall delay will be minimized by NQ policy. Furthermore, a server does not transfer incoming job to the servers until fastest server than it is available.

V. Comparison of Dynamic Load Balancing Algorithms [8]

Table-1 Comparison among the diff. Dynamic Load Balancing Algorithms [8]

Sr. No.	Algorithms	State information check	Performance
1	Random	No	Excellent
2	Prioritized Random	Partial	Excellent
3	Adaptive Contracting With Neighbour (ACWN)	Yes	Good
4	Prioritized ACWN	Yes	Good
5	Cyclic	Partial	Slightly better than Random
6	Probabilistic	Partial	Good
7	Threshold	Partial	Better

8	Least	Partial	Better
9	Reception	Partial	Not so good
10	Shortest Expected Delay	Yes	Good
11	Never Queue	Yes	Good

VI. Conclusion

In this paper, Load estimation, performance indices, amount of information exchanged among nodes, job resource requirements estimation, job selection for transfer, remote nodes selection, are some of the issues that have been discussed.

More efficient load balancing algorithm more is the performance of the computing system. There exists no absolutely perfect balancing algorithm but one can use depending on the need. The study provides a view of the load balancing algorithms as well as offers practical guidelines to researchers in designing efficient load balancing algorithms for distributed computing systems.

VII. References

[1] Ali M. Alakeel, "A Guide to Dynamic Load Balancing in Distributed Computer Systems", IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010

[2] Purnima Shah and S. M. Shah, "Load Balancing in Distributed System Using Genetic Algorithm", Special issues on IP Multimedia Communications

[3] Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma, "Performance Analysis of Load Balancing Algorithms", World Academy of Science, Engineering and Technology 38 2008

[4] M. Nikravan and M. H. Kashani, "A Genetic Algorithm for Process Scheduling In Distributed Operating Systems Considering Load Balancing", Proceedings 21st European Conference on Modelling and Simulation Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni ©ECMS 2007.

[5] Amit Chhabra, Gurvinder Singh, Sandeep Singh Waraich, Bhavneet Sidhu, and Gaurav Kumar "Qualitative Parametric Comparison of Load Balancing Algorithms in Parallel and Distributed Computing Environment" World Academy of Science, Engineering and Technology 2006.

[6] K. Goswami, M. Devarakonda, and R. Iyer, "Prediction-Based Dynamic Load-Sharing Heuristics," IEEE Transactions on Parallel and Distributed Systems, Vol. 4, No. 6, pp. 638-648, June 1993.

[7] Pradeep Sinha, "Distributed Operating Systems – Concepts and Design", PHI Learning Private Limited, New Delhi, 2009

[8] Md. Firoj Ali, Rafiqul Zaman Khan, "The Study on Load Balancing Strategies In Distributed Computing System", International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.2, April 2012