International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Study of Load Balancing In Distributed Computing Environment

Minal M. Taley[1], Prof. Ranjit R. Keole[2]

[1]PG Student, S.G.B.A.U University, H.V.P.M College of Engineering & Technology, Amravati, Maharashtra, India
[2]Professor, S.G.B.A.U University, H.V.P.M College of Engineering & Technology, Amravati, Maharashtra, India

Abstract: ***The advancements in micro-electronic technology have resulted in availability of fast, inexpensive processors and the availability of highly cost effective, efficient computer networks are the result of advancement in communication technologies. Based on these the price/performance ratio indicates that the use of multiple interconnected processors is more beneficial compared to a single high speed processor. These interconnected processors can be tightly coupled or loosely coupled constitutes distributed and parallel computing environment. This kind of environment is beneficial in many ways such as information sharing among processors, resource sharing, shorter response time, high throughput, better price/performance ratio, higher reliability and extensibility. The major research issue in distributed systems is development of effective techniques for distributing work load among multiple processors. The main goal is to achieve performance goals along with distribution of work load. Load balancing algorithms enables the jobs to move from one processor to another.***

Keywords: SLB, DLB, Distributed Systems, Load and Processor

1. Introduction

Processing speed of a system is always highly intended. From the beginning of the development of computer it is always focused on the system performance that is how to improve the speed or performance of an existing system and thus we reached to the era of supercomputer. Especially the business organizations, defence sectors and science groups need the high performance systems for constant change of their day to day need. So from serial computer to supercomputer, parallel computing and parallel distributed computing have been developed. Massively parallel computers (MPC) are available in the market today. In MPC a group of processors are linked to the memory modules through the network like mesh, hypercube or torus. Super computers are very expensive so a new alternative concept has emerged (although existed) that is called parallel distributed computing in which thousands of processors can be connected either by wide area network or across a large number of systems which consists of cheap and easily available autonomous systems like workstations or PCs. So it is becoming extremely popular for large computing purpose such as scientific calculations as compared to MPC. Recently distributed systems with several hundred powerful processors have been developed.

Distributed computing system provides high performance environment that are able to provide huge processing power. Multicomputer system can be efficiently used by efficient task partitioning and balancing the tasks (loads) among the nodes properly. The parallel systems are highly applicable to the fields like Financial Modeling, Hydrodynamics, Quantum Chemistry, Astronomy, Weather Modeling and Forecasting, Geological 20 Modeling, Prime Numbering Factoring, Biological science and so on. Blue Gene /L (belongs to Blue Gene family of parallel computers) system, which have been configured with as much as 65,536 computing nodes, developed by inter National Business Machines(IBM) having the operational speed of 280.6 teraflops and was fastest computer on Oct, 2005. IBM is currently developing Blue Gene /R and Blue Gene /P of speed 1.40petaflops, the successors of Blue Gene/ L system. Distributed network is mainly heterogeneous in nature in the sense that the processing nodes, network topology, communication medium, operating system etc. may be different in different network which are widely distributed over the globe. Presently several hundred computers are connected to build the distributed computing system. In order to get the maximum efficiency of a system the overall work load has to be distributed among the nodes over the network. So the issue of load balancing became popular due to the existence of distributed memory multiprocessor computing systems. The distribution of loads to the processing elements is simply called the load balancing problem. In a system with multiple nodes there is a very high chance that some nodes will be idle while the other will be over loaded. The goal of the load balancing algorithms is to maintain the load to each processing element such that all the processing elements become either overloaded or idle that means each processing element ideally has equal load at any moment of time during execution to obtain the maximum performance (minimum execution time) of the system.

So the proper design of a load balancing algorithm may significantly improve the performance of the system.

In the network there will be some fast computing nodes and slow computing nodes. If we do not account the processing speed and communication speed (bandwidth), the performance of the overall system will be restricted by the slowest running node in the network. Thus load balancing strategies balance the loads across the nodes by preventing the nodes to be idle and the other nodes to be overwhelmed. Furthermore, load balancing strategies removes the idleness of any node at run time.

Now, question arises when the load balancing algorithm will be applied? There are two situations firstly at the time of compilation which is known as static load balancing (SLB)algorithm in which the assignment of the tasks to the processors are done before runtime. After the assignment of jobs no change is possible that means redistribution of tasks is not possible. Secondly at the time of execution which is known as dynamic load balancing (DLB)algorithm in which the task are dynamically distributed across the nodes and redistribution of tasks is possible. In SLB priori knowledge about the tasks and the system are known, so the task are distributed according to the performance of the nodes and the other factors like communication speed, input output buffer size etc. Once assignment of works is done there assignment is frizzed, so the communication over-heads is negligible here. But in DLB jobs are reassigned at the runtime when the load will be transferred from heavily loaded nodes to the lightly loaded or idle nodes. So considerable communications over-heads occur and become more when number of processors increase. Quality of a load balancing algorithm is dependent on two factors. Firstly number of steps that are needed to get a balanced state. Secondly the extent of load that moves over the link to which nodes are connected.

2. Literature Review

In a distributed system characterized by a resource multiplicity combined with a stochastic nature of the workload, there is a high probability for the occurrence of the 'wait while idle' state whereby some hosts in the pool are idle while other hosts have multiple jobs queued up. In his profiling of a network of workstations, Mutka has shown that processors are idle 70% of the time while Livny showed that this probability depends on the system load and the size of the pool, and that load balancing can improve performance even in the case of homogeneous process arrival rates. This environment is characterized by changing and uneven loads on the hosts and a lack of up-to-date system state information. For an effective use of the resource multiplicity inherent in such systems, and to satisfy the diverse and sometimes conflicting users' performance expectations, the design of efficient distributed scheduling algorithms and mechanisms for processor allocation has been a research challenge for over a decade. These algorithms deal with the global scheduling of system workload through local and remote job placement, while allocation of local resources is left to the local scheduling component.

Although the common objective of load balancing is to improve the performance of the computer system, the nature of the performance objective differs with the computing environment involved:

- For a general purpose distributed computer system based on a local area network, it is to reduce the average system response time with a minimum degradation of the performance for individual users. An, alternative objective is greedy scheduling where each job is allocated to the node where it has the best response time regardless of the effect on other jobs.
- For a real time distributed system, it is to provide a guaranteed response time.
- For a parallel computer system, it is to reduce the total execution time of a program composed of several modules.

In this review load balancing is addressed at two levels. On the distributed system level different architectural models and perspectives are surveyed. The communication model assumed is based on the broadcast device. On the algorithm level, first previous work on dynamic algorithms is reviewed then the approaches to the adaptability problem are considered [4].

3. Load Balancing

Load balancing is the way of distributing load units (jobs or tasks) across a set of processors which are connected to a network which may be distributed across the globe. The excess load or remaining unexecuted load from a processor is migrated to other processors which have load below the threshold load. Threshold load is such an amount of load to a processor that any load may come further to that processor. In a system with multiple nodes there is a very high chance that some nodes will be idle while the other will be over loaded. So the processors in a system can be identified according to their present load as heavily loaded processors (enough jobs are waiting for execution), lightly loaded processors(less jobs are waiting) and idle processors (have no job to execute). By load balancing strategy it is possible to make every processor equally busy and to finish the works approximately at the same time. A load balancing operation consists of three rules. These are location rule, distribution rule and selection rule. The selection rule works either in preemptive or in non-preemptive fashion. The newly generated process is always picked up by the non-preemptive rule while the running process may be picked up by the preemptive rule.

Preemptive transfer is costly than non-preemptive transfer which is more preferable. However preemptive transfer is more excellent than non-preemptive transfer in some instances. Practically load balancing decisions are taken jointly by location and distribution rules. The balancing domains are of two types: local and global. In local domain, the balancing decision is taken from a group of nearest neighbors by exchanging the local workload information while in global domain the balancing decision is taken by triggering transfer partners across the whole system and it exchanges work load information globally.

4. Static Load Balancing Algorithm (SLB)

In static algorithm the processes are assigned to the processors at the compile time according to the performance of the nodes. Once the processes are assigned, no change or reassignment is possible at the run time. Number of jobs in each node is fixed in static load balancing algorithm. Static algorithms do not collect any information about the nodes. The assignment of jobs is done to the processing nodes on the basis of the following factors: incoming time, extent of resource needed, mean execution time and inter-process communications. Since these factors should be measured

before the assignment, this is why static load balance is also called probabilistic algorithm. As there is no migration of job at the runtime no overhead occurs or a little over head may occur. In static load balancing it is observed that as the number of tasks is more than the processors, better will be the load balancing.

Fig 1 shows a schematic diagram of static load balancing where local tasks arrive at the assignment queue. A job either be transferred to a remote node or can be assigned to threshold queue from the assignment queue. A job from remote node similarly be assigned to threshold queue. Once a job is assigned to a threshold queue, it cannot be migrated to any node. A job arriving at any node either processed by that node or transferred to another node for remote processing through the communication network. The static load balancing algorithms can be divided into two sub classes: optimal static load balancing and sub optimal static load balancing [2].

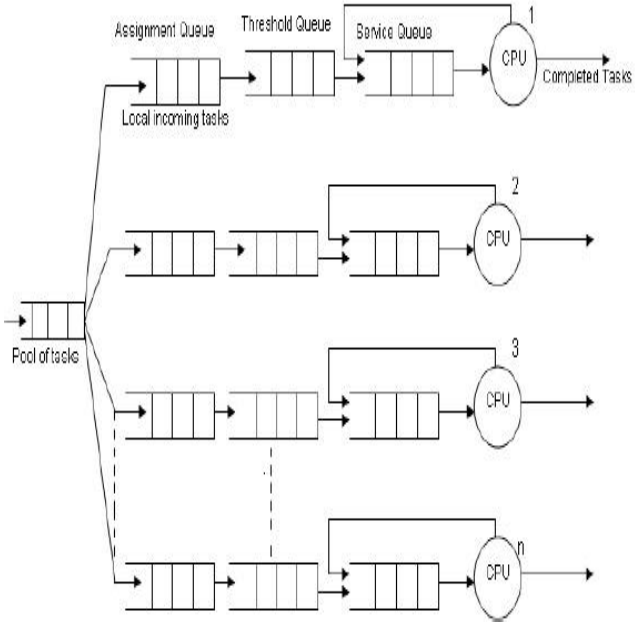

Figure 1: Model of Processing Node

4.1 There are four types of static load balancing algorithms:

(i) Round Robin Algorithm: It distributes jobs evenly to all processors. Processors perform locally on each process independent of allocation of other processor.

(ii) Randomized Algorithm: It uses random numbers to choose the processors. The random numbers are generated based on a statistic distribution.

(iii) Central Manager Algorithm: In this a central process will choose a slave processor and assign a job. The selection is based the least load.

(iv) Threshold Algorithm: Here the processors are assigned immediately upon creation to host two threshold parameter t_under and t_upper are used to describe the three main levels, under loaded, medium loaded and over loaded.

4.2 The general advantages of SLB are:

1) The methods are very simple.
2) Minimal communication delay.
3) No execution overhead.

4.3 The general disadvantages of SLB are:

1) No accurate methods to estimate execution time.
2) The process allocation cannot be changed during execution.
3) These methods do not consider data distribution complications

5. Dynamic Load Balancing Algorithm (DLB)

During the static load balancing too much information about the system and jobs must be known before the execution. These information may not be available in advance. A thorough study on the system state and the jobs quite tedious approach in advance. So, dynamic load balancing algorithm came into existence. The assignment of jobs is done at the runtime. In DLB jobs are reassigned at the runtime depending upon the situation that is the load will be transferred from heavily loaded nodes to the lightly loaded nodes. In this case communication over heads occur and becomes more when number of processors increase. In dynamic load balancing no decision is taken until the process gets execution. This strategy collects the information about the system state and about the job information. As more information is collected by an algorithm in a short time, potentially the algorithm can make better decision. Dynamic load balancing is mostly considered in heterogeneous system because it consists of nodes with different speeds, different communication link speeds, different memory sizes, and variable external loads due to the multiple. The numbers of load balancing strategies have been developed and classified so far for getting the high performance of a system Fig. 2 shows a simple dynamic load balancing for transferring jobs from heavily loaded to the lightly loaded nodes [2].

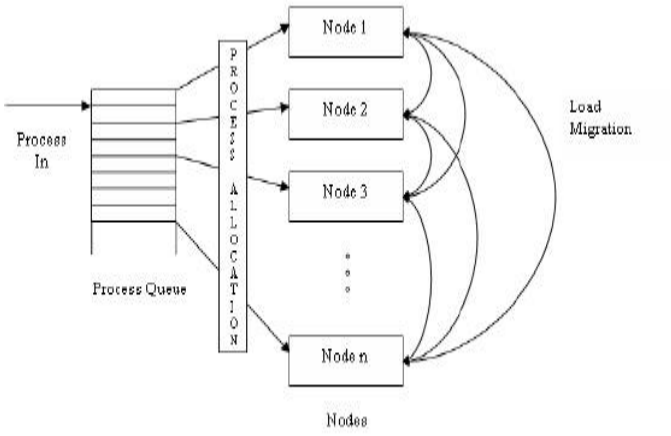

Figure 2: Job Migration in Dynamic Load Balancing Strategy

5.1There are different strategies required by DLB algorithm.

1. Load distribution decision based on current work load.

2. Mechanism for collecting and managing system state information.
3. Mechanisms to assist each node in deciding which job is eligible for load balancing and assist job transfer from local to remote node.
4. Mechanism on which the destination node is selected.

5.2 Based on the above requirements three main strategies are used.

1. **Information strategy**: This is responsible for collecting the information about the nodes.

2. **Transfer strategy**: This select the job for selection of a job for transfer from a local node to remote node.

3. **Location strategy**: This is to select a destination node for transferred task.

5.3 There are two types of DLB algorithm.

(i) Central Queue Algorithm: It works on the principle of dynamic distribution. It stores new activities and unfulfilled request as a cyclic FIFO queue on the main host.

(ii) Local queue Algorithm: It works on the principle of process migration. Initially the static allocation of the processors is done and process migration is initiated by a host when the load falls under some user defined threshold limit.

6. Qualitative Parameters

There are many parameters on which both SLB& DLB can be compared. We have taken few for comparison and understanding purpose:

(i) **Nature:** This factor determines the prerequisite of an algorithm.SLB algorithm are static in nature so planning is very much required. Every processor load is planned and estimated before execution.DLB algorithms are dynamic in nature i.e., process allocation will be done at run time hence no preplanning is required.

(ii) **Overhead Associated:** This factor is related with the amount of overhead involved in implementing algorithm. The overhead may be due to relocation of tasks inter process and processor communication.SLB algorithm incurs less overhead due to pre planning of job allocation. DLB algorithms incur more overhead due to their adaptive nature towards the situation.

(iii) **Resource Utilization**: This factor is used to check how efficiently the resources are utilized. No processors at any point of time should be idle or over loaded, no tasks should be waiting in the queue.SLB algorithm have lesser resource utilization as job allocation is static in nature due to which some process finish their job early and be idle due to lack of work.DLB algorithm have relatively better resources utilization as the job allocation makes equal distribution of work to all the processors.

(iv) **Preemptiveness**: This factor is related with checking the fact that whether tasks in execution can be transferred to other processors or not. In SLB algorithms no reallocation of task is done hence nonpreemptive. DLB algorithms are both preemptive and nonpreemptive based on the task.

(v) **Processor Thrashing:** Processor thrashing occurs when more of processor time is wasted due to process migration. SLB algorithms are free from processor thrashing where no relocation of task is done. DLB algorithm incurs substantial processors thrashing.

(vi) **Reliability:** This factor is related with the adaptability of algorithms in case of some machine failures occurs. SLB algorithms are less reliable as no change can be done during execution time. DLB algorithms are more reliable as its supports process migration.

(vii)**Adaptability:** This factor is used to check whether the algorithm is adaptive to varying situations. SLB algorithms are not adaptive as the process allocation is fixed and cannot be changed. DLB algorithms are adaptive towards every situation.

(viii)**Response Time:** How much time a distributed system using a particular load balancing algorithm is taking to respond? The lesser the overhead minimum will be the response time. SLB algorithms have shorter response time due to less overhead. Whereas DLB algorithms may have relatively higher response time

(ix)**Predictability:** This factor is to predict outcome of an algorithm. It is related with deterministic or nondeterministic factor with which one can predict the outcome. SLB algorithms are predictable in nature as most of the things like workload assignment, average execution time are fixed at compile time. DLB algorithms are unpredictable as all decisions are done at run time.

(x) **Stability:** This factor is related with the exchange of present workload information among the processors. The information exchange done during processing time should be accurate and in time so that the information shared should be updated and useful. SLB algorithms are considered as stable as no information sharing is allowed. DLB algorithms in this context can be considered as less stable as there a probability that the exchanged information is not updated [1].

7. Comparison in Tabular Format

The comparison work in tabular form is shown in Table 1[1].

Table 1: Showing comparison work

Sl No	Load Balancing parametres	SLB Algorithm	DLB Algorithm
1	Nature	Static(Workload is assigned before runtime)	Dynamic (Workload is assigned at runtime)
2	Overhead Associated	Less	More
3	Resource Utilization	Less	More
4	Preemptiveness	Nonpreemptive	Preemptive and Nonpreemptive
5	Processor Thrashing	No thrashing	Substantial thrashing
6	Reliability	Less	More
7	Adaptability	Less adaptive	More adaptive
8	Response Time	Less	More
9	Predictability	More predictable	Less predictable
10	Stability	More	Less

8. Applications

(i)One of the most commonly used applications of load balancing is to provide a single Internet service from multiple servers, sometimes known as a server farm. Commonly load-balanced systems include popular web sites, large Internet Relay Chat networks, high-bandwidth File Transfer Protocol sites, Network News Transfer Protocol (NNTP) servers and Domain Name System (DNS) servers. Lately, some load balancers have evolved to support databases; these are called database load balancers.

(ii)For Internet services, the load balancer is usually a software program that is listening on the port where external clients connect to access services. The load balancer forwards requests to one of the "backend" servers, which usually replies to the load balancer. This allows the load balancer to reply to the client without the client ever knowing about the internal separation of functions. It also prevents clients from contacting back-end servers directly, which may have security benefits by hiding the structure of the internal network and preventing attacks on the kernel's network stack or unrelated services running on other ports.

(iii) Load balancing is often used to implement failover—the continuation of a service after the failure of one or more of its components. The components are monitored continually (e.g., web servers may be monitored by fetching known pages), and when one becomes non-responsive, the load balancer is informed and no longer sends traffic to it. When a component comes back on line, the load balancer begins to route traffic to it again. For this to work, there must be at least one component in excess of the service's capacity. This can be much less expensive and more flexible than failover approaches where each single live component is paired with a single backup component that takes over in the event of a failure. Some types of RAID systems can also utilize hot spare for a similar effect.

(iv) Load balancing can be useful in applications with redundant communications links. For example, a company may have multiple Internet connections ensuring network access if one of the connections fails. A failover arrangement would mean that one link is designated for normal use, while the second link is used only if the primary link fails. Using load balancing, both links can be in use all the time. A device or program monitors the availability of all links and selects the path for sending packets. The use of multiple links simultaneously increases the available bandwidth[5].

9. Conclusion and Future Directions

The algorithm adopted for load balancing is closely related to the type and amount of load and job information assumed to be known to the decision-making modules**.** If process execution times and inter processor communication time can be estimated a priori i.e. at compile time which is practically difficult, then SLB algorithms should be employed otherwise dynamic load balancing algorithms should be used.

This paper concludes with following results in terms of performance comparison of SLB and DLB algorithms.

1) Parameters like Associated overhead, Processor thrashing, Predictability, Planning in terms of nature, response time and Stability are in favor of SLB algorithms.
2) Whereas other parameters like Resource utilization, Preemptiveness, Adaptability and Reliability favor DLB algorithms. In order to balance the workload efficiently, one has to make a choice between SLB and DLB algorithms. In case of SLB algorithms efforts must be done to develop methods, which estimate process execution times and other resource requirements a priori i.e. at compile time. Also other parameters like Resource utilization, Adaptability and Reliability must be thoroughly entertained to achieve best possible results.

While In case of DLB algorithms effort must be done to develop methods that help to reduce associated overhead. Also other parameters like Predictability and Stability must be taken care of to get best possible results.

In future this work can be extended to develop a virtual environment to study these load balancing algorithms based on identified comparative parameters quantitatively. The supporting data will be generated on the basis of target algorithms and it will become a background for evaluating load-balancing algorithms [4].

References

[1] Arathi P., "Comparative Study of Load Balancing Algorithms With Qualitative Parametric Comparison In Distributed Computing ",IEEE Sponsored International Conference On Empowering Emerging Trends In Computer, Information Technology & Bioinformatics International Journal of Computer, Information Technology & Bioinformatics (IJCITB) ISSN: 2278-7593, Volume-2, Issue-2.

[2] Md. Firoj Ali and 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, Department of Computer Science Aligarh Muslim University, Aligarh (India).[Online] Available:http://www.academia.edu/2066646/THE_STUDY_ON_LOAD_BALANCING_STRATEGIES_IN_DISTRIBUTED_COMPUTING_SYSTEM [Accessed: April 4,2015]

[3] Kouider Benmohammed-Mahieddine, "An Evaluation of Load Balancing Algorithms for Distributed Systems".

[4] Amit Chhabra, Gurvinder Singh,"Qualitative Parametric Comparison of Load Balancing Algorithm in Distributed Computing Environment", Conference Paper [Online], Available:http://www.researchgate.net/publication/42687783_Qualitative_Parametric_Comparison_of_Load_Balancing_Algorithms_in_Distributed_Computing_Environment [Accessed:29 march,2015].

[5] Use of load balancing in distributed system[Online], Available:http://en.wikipedia.org/wiki/Load_balancing_(computing), [Accessed: 29 march, 2015].