THE STUDY ON LOAD BALANCING STRATEGIES IN DISTRIBUTED COMPUTING SYSTEM

Md. Firoj Ali[1], Rafiqul Zaman Khan[2]
Department of Computer Science
Aligarh Muslim University, Aligarh (India).
[1]firojali.mca@gmail.com, [2]rzk32@yahoo.co.in

ABSTRACT

A number of load balancing algorithms 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 systems. However, the 'distributed system' becomes popular and attractive with the introduction of the web. 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.

KEYWORDS: *Load Balancing, SLB, DLB, Parallel Computing, Distributing Computing*

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. Specially the business organizations, defense 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 [8]. Super computers are very expensive so a new alternative concept has emerged (although existed) that is called parallel distributed computing in which thousand 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 [13]. 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 [1].

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 [3]. 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

DOI : 10.5121/ijcses.2012.3203

Oct, 2005. IBM is currently developing Blue Gene /R and Blue Gene /P of speed 1.40 petaflops, the successors of Blue Gene/Lsystem [3].

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 [10, 15]. Presently several hundred computers are connected to build the distributed computing system [13]. In order to get the maximum efficiency of a system the over all 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 [11].

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 neither overloaded nor 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 [4, 8, 11, 14,15]. 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 [15]. 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 [9]. 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 [9]. 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 the reassignment 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 [10]. 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. 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 [9]. 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 [2, 5, 7, 12, 17]. 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 [17].

Practically load balancing decisions are taken jointly by location and distribution rules [5, 17]. 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.

2.1. Benefits of Load balancing

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

For the above benefits the load balancing strategy becomes a field of intensive research.

So many load balancing algorithms have been developed in the past several years but no single algorithm is appropriate for all applications. The selection of an appropriate load balancing depends on application parameters like balancing quality, load generation patterns and also hardware parameters like communication overheads. Generally load balancing algorithms are of two types: first is static load balancing and second one is dynamic load balancing.

2.2. Static Load Balancing

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 [10, 7]. 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 [11]. 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 can not 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 [6, 11].

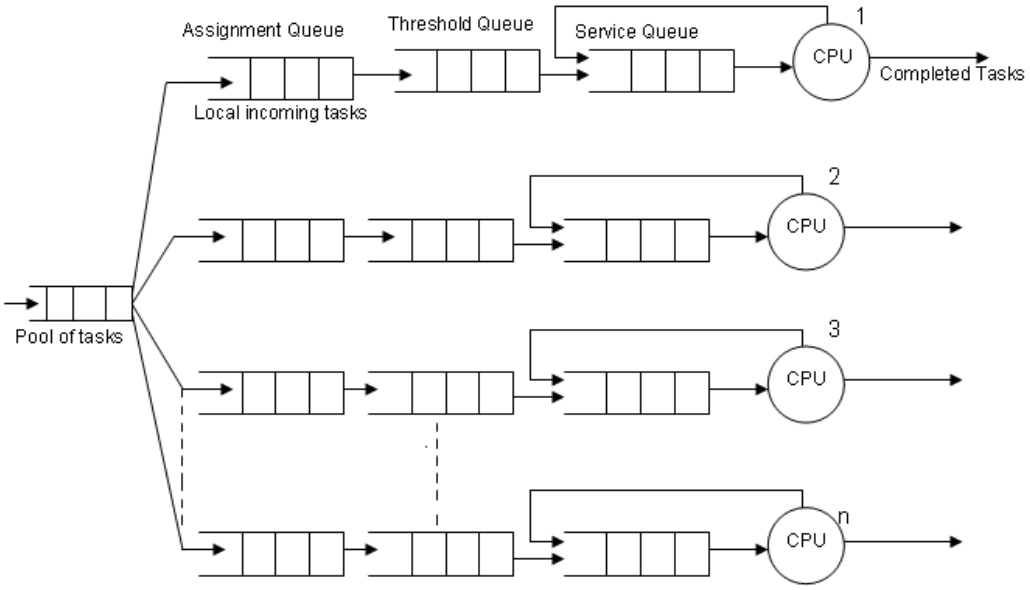

Figure 2. Model of Processing Node [source: C. LU and S. LAU, 1996.]

2.2.1. Optimal Static Load Balancing Algorithm

If all the information and resources related to a system are known optimal static load balancing can be done. It is possible to increase throughput of a system and to maximize the use of the resources by optimal load balancing algorithm. Genetic algorithms (GA) and simulated annealing (SA) are the example of optimization techniques [6].

2.2.2. Sub-Optimal Static Load Balancing Algorithm

Sub-optimal load balancing algorithm will be mandatory for some applications when optimal solution is not found. The thumb-rule and heuristics methods are important for sub-optimal algorithm.

2.3. Dynamic Load Balancing

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 [7, 10, 12]. 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 [10]. 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 [10]. Fig. 4 shows a simple dynamic load balancing for transferring jobs from heavily loaded to the lightly loaded nodes.

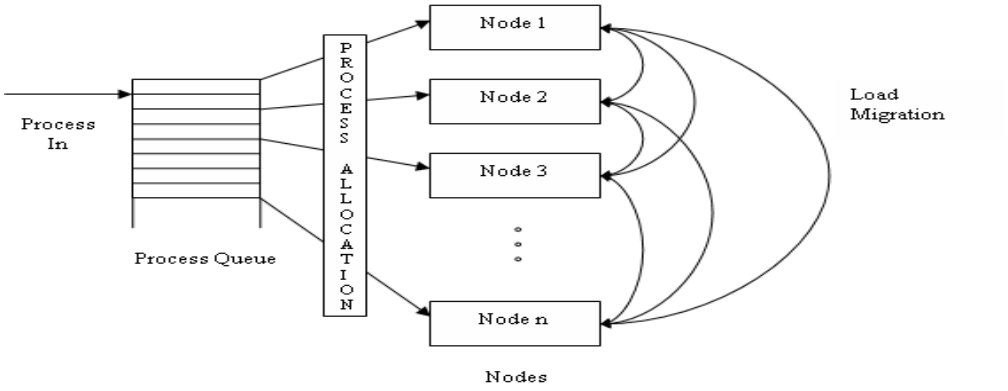

Figure 3. Job Migration in Dynamic Load Balancing Strategy (source: Jain P. and Gupta D., 2009.)

2.3.1. Dynamic Load Balancing Issues

During the design of a dynamic load balancing algorithm the following issues are considered [11, 17]:

a) Job Assignment which assigns the jobs to the processors according to the situation in a system.

b) Load Calculation which tells how to calculate the workload of a particular node in a system.

c) Job Transfer which determines whether a job is to be executed locally or remotely. This also defines when a node becomes overloaded.

d) System State which tells whether a node is overloaded or lightly loaded.

e) Priority Assignment which tells the priority of execution of local and remote processes at a particular node.

f) Information Exchange which tells how to exchange the system load information among the nodes. The information policy includes the following steps [17].

- On demand: processors begin to collect load information of other nodes when load balancing operation is going to start.
- Periodical: processors inform their load information at regular interval to the other nodes which either may not be interested.
- On-state -change: when a processor changed its state, it immediately informs the others by passing information.

g) Migration limiting which determines frequency of transfer that how many of times a process can migrate from one node to another.

3. COMPARISON BETWEEN SLB and DLB ALGORITHM

Some qualitative parameters for comparative study have been listed below.

3.1. Nature

Whether the applied algorithm is static or dynamic is determined by this factor.

3.2. Overhead Involved

In static load balancing algorithm redistribution of tasks are not possible and there is no overhead involved at runtime. But a little overhead may occur due to the inter process communications. In case of dynamic load balancing algorithm redistribution of tasks are done at the run time so considerable over heads may involve. Hence it clear that SLB involves a less amount of overheads as compared to DLB.

3.3. Utilization of Resource

Though the response time is minimum in case of SLB, it has poor resource utilization capability because it is impractical to get all the submitted jobs to the corresponding processors will completed at the same time that means there is a great chance that some would be idle after completing their assigned jobs and some will remain busy due to the absence of reassignment policy. In case of dynamic algorithm since there is reassignment policy exist at run time, it is possible to complete all the jobs approximately at the same time. So, better resource utilization occurs in DLB.

3.4. Thrashing or Process Dumping

A processor is called in thrashing if it is spending more time in migration of jobs than executing any useful work [1]. As the degree of migration is less, processor thrashing will be less. So SLB is out of thrashing but DLB incurs considerable thrashing due to the process migration during run time.

3.5. State Woggling

It corresponds to the frequent change of the status by the processors between low and high. It is a performance degrading factor [1].

3.6. Predictability

Predictability corresponds to the fact that whether it is possible to predict about the behavior of an algorithm. The behavior of the SLB algorithm is predictable as everything is known before compilation. DLB algorithm's behavior is unpredictable, as everything is done at run time.

3.7. Adaptability

Adaptability determines whether an algorithm will adjust by itself with the change of the system state. SLB has no ability to adapt with changing environment. But DLB has that ability.

3.8. Reliability

Reliability of a system is concerned with if a node fails still the system will work without any error. SLB is not so reliable as there is no ability to adapt with the changing of a system's state. But DLB has adaptation power, so DLB is more reliable.

3.9. Response Time

Response time measures how much time is taken by a system applying a particular load balancing algorithm to respond for a job. SLB algorithm has shorter response time because

processors fully involved in processing due to the absence of job transferring. But DLB algorithm has larger response time because processors can not fully involved in processing due to the presence of job transferring policy.

3.10. Stability

SLB is more stable as every thing is known before compilation and work load transfer is done. But DLB is not so stable as SLB because it involves both the compile time assignment of jobs and distribution of work load as needed.

3.11. Complexity Involved

SLB algorithms are easy to construct while DLB algorithms are not so easy to develop because nothing is known in advance. Although the dynamic load balancing is complex phenomenon, the benefits from it is much more than its complexity [10].

3.12. Developing Cost

More complexity implies more cost to develop an algorithm. So SLB algorithms incur less cost as compared to the DLB algorithms.

Now the comparisons done so far between SLB and DLB algorithm are summarized in a tabular form.

Table 1. Comparison between SLB and DLB

S.NO.	Factor	Load Balancing	
		SLB Algorithm	**DLB Algorithm**
1	Nature	Work load is assigned at compile time	Work load is assigned at run time
2	Overhead involved	Little overhead due to IPC	Greater overhead due to process redistribution
3	Resource utilization	Lesser utilization	Greater utilization
4	Processor thrashing	No thrashing	Considerable thrashing
5	State woggling	No woggling	Considerable woggling
6	Predictability	Easy to predict	Difficult to predict
7	Adaptability	Less adaptive	More adaptive
8	Reliability	Less	More
9	Response time	Short	Longer
10	Stability	More	Less
11	Complexity	Less	More
12	Cost	Less	More

4. COMPARISON of SOME DYNAMIC LOAD BALANCING ALGORITHMS

Now some dynamic load balancing algorithms are studied below.

4.1. Nearest Neighbor Algorithm

With nearest neighbor algorithm each processor considers only its immediate neighbor 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 neighbors [17]. By exchanging the load successively to the neighboring nodes the system attains a global balanced load state. The nearest neighbor algorithm is mainly divided into two categories which are diffusion method and dimension exchange method.

With this method a heavily or lightly loaded processor balances its load simultaneously with all its nearest neighbors at a time while in dimension exchange method a processor balances its load successively with its neighbor one at a time.

4.2. Random (RAND) Algorithm

As soon as a workload (greater than threshold load) is generated in a processor, it is migrated to a randomly selected neighbor. It does not check state information of anode. This algorithm neither maintains any local load information nor sends any load information to other processors [17]. Furthermore, 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 neighbors.

4.3. Adaptive Contracting with Neighbor (ACWN)

As soon as the workload is newly generated, it is migrated to the least loaded nearest neighbor 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 neighbor processor which has load below the threshold load. So, ACWN does require maintaining the local load information and also the load information of the neighbors for exchanging the load periodically. Hence, RAND is different form the ACWN in a respect that ACWN always finds the target node which is least loaded in neighbors [17].

4.4. Prioritized Random (PRAND) Algorithm

In both RAND and ACWN the work load is supposed to be uniform in the sense of their computational requisites. 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 neighbor. On the other hand, PACWN selects the second largest weighted workload and transfer it to the least loaded neighbor [17].

4.4.1 Averaging Dimension Exchange (ADE)

ADE algorithm follows the concept of local averaging load distribution operation and balances the workload with one of its neighbors [17].

4.4.2. Averaging Diffusion (ADF) Algorithm

It averages the local load distribution as ADE but exchanges the load with all its neighbors.

4.3. 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.

4.4. PROBABILISTIC

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 [5].

4.5. 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 [5]. THRESHOLD and LEAST have good performance and are of simple in nature. Furthermore, up-to-date load values are used by these algorithms.

4.6. RECEPTION

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

4.7. Centralized Information and Centralized Decision

In this class of algorithms the information about the system is stored in a single node and the decision is also taken by that single node. CENTRAL is a subclass of this algorithm. When a heavily loaded node wants to migrate a job, it requests a server for a lightly loaded node. Every node in the system informs the server machine whether a lightly loaded node is available or not. CENTRAL afford very efficient performance results [5]. But this algorithm suffers from a very serious problem that if the server is crashed, no facility will be provided by this algorithm.

4.8. Centralized Information and Distributed Decision

In GLOBAL, collection of information is centralized while decision making is distributed [5]. The load situation on the nodes is broadcasted by the server. Through this information an overloaded processor finds the lightly loaded node from its load vector without going through the server. This algorithm is very efficient due to the less inclusion of message information and it is robust in nature because the system remains alive even when the server is in recovery state. GLOBAL algorithm gathers large information but information is not up-to-date. As a result greater overheads occur in the system.

4.9. 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.

In RADIO, both the information and decision are distributed and there is no broadcasting without compulsion. In this algorithm, a distributed list consisting of lightly loaded nodes in which each machine is aware of its successor and predecessor. Furthermore, each node is aware of the head of the available list that is called manager. The migration of a process from a heavily loaded node to the lightly loaded node is done directly or indirectly through the manager. Broadcasting occurs when manager crashes or a node joins the available list [5].

4.10. 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 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 [11].

4.11. Modified SED

A term communication delay is introduced to the SED policy. The communication delay is experienced by a job due to the transfer of job from source node to the destination node. Another modification is done due to the limited size of input queue of a node [11]. If a job is sent to a node whose input queue is saturated, the job is not be able to enter the queue and it will be transferred to second best node which may also have the input queue saturated and so on. If no node is available for a job then it is transferred to the original node anyway. This job will be stored temporarily at this node if the queue there is still saturated and make an attempt to enter again after a fixed interval of time. This delay is considered in modified SED strategy as well.

4.12. The Never Queue (NQ) Strategy

NQ policy 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 [16]. This algorithm always places a job to a fastest available server [11]. 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.

4.13. Modification of NQ strategy

NQ strategy wants to find always the fastest idle server. If no idle server is available, the server with shortest expected delay is selected. If a server is incapable of accepting the job, it is transferred to another server with shortest expected delay and so on. If finally no server with shortest expected delay is found, the job will be backed to the original server anyway. The job may have to wait temporarily there due to the saturation of input queue. So the job will get the chance when the input queue has enough space to accommodate it after some interval of time. This delay should be considered in modified NQ strategy [11].

4.14. 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 [11, 16].

5. COMPARISON of DLB

The comparison of above discussed dynamic load balancing algorithms has been shown in Table 2.

Table 2. Comparison among the different algorithms is summarized

S.N.	Algorithms	State information check	Performance
1	RAND	No	Excellent
2	PRAND	Partial	Excellent

3	ACWN	Yes	Good
4	PACWN	Yes	Good
5	CYCLIC	Partial	Slightly better than RAND
6	PROBALISTIC	Partial	Good
7	THRESHOLD	Partial	Better
8	LEAST	Partial	Better
9	RECEPTION	Partial	Not so good
10	CENTRAL	Yes	Excellent
11	GLOBAL	Yes	Good
12	OFFER	Yes	Poor
13	RADIO	Yes	Good
14	SED	Yes	Good
15	NQ	Yes	Good

6. CONCLUSION

In this paper we studied the load balancing strategies lucidly in detail. Load balancing in distributed systems is the most thrust area in research today as the demand of heterogeneous computing due to the wide use of internet. More efficient load balancing algorithm more is the performance of the computing system. We made a comparison between SLB and DLB introducing some new parameters. We have enumerated the facilities provided by load balancing algorithms. Finally, we studied some important dynamic load balancing algorithms and made their comparison to focus their importance in different situations. There exists no absolutely perfect balancing algorithm but one can use depending one the need.

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.

REFERENCES

[1] Ahmad I., Ghafoor A. and Mehrotra K. “Performance Prediction of Distributed Load Balancing on Multicomputer Systems”. ACM, 830-839, 1991.

[2] 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.

[3] Archer C.J., Mullins T.J, Sidelnik A. and Smith B.E. “Parallel Computing System Using Coordinator and Master Nodes for Load Balancing and Distributing Work”. United State Patent, 2010.

[4] Berenbrink P., Friedetzky T. and Steger A. “Randomized and Adversarial Load Balancing”. CiteSeer^{x}, 1997.

[5] Bernard G., Steve D. and Simatic M. “A Survey of Load Sharing in Networks of Workstations”. The British Computerm Society, The Institute of Electrical Engineers and IOP Publishing Ltd, 75-86,1993.

[6] Chhabra A., Singh G., Waraich S.S., Sidhu B. and Kumar G. "Qualitative Parametric Comparison of Load Balancing Algorithms in parallel and Distributed Computing Environment". Word Academy of Science, Engineering and Technology, 39-42, 2006.

[7] Dandamudi S.P. "Sensitive Evaluation of Dynamic Load Sharing in Distributed System". IEEE,1998.

[8] Hamdi M. and Lin C.K. "Dynamic Load Balancing of Data Parallel Applications on a Distributed Network". In 9th International Conference on Supercomputing, ACM, 170-179, 1995.

[9] Haddad E. "Dynamic Optimization of Load Distribution in Heterogeneous Systems". IEEE, 29-34, 1994.

[10] Jain P. and Gupta D. "An Algorithm for Dynamic Load Balancing in Distributed Systems with Multiple Supporting Nodes by Exploiting the Interrupt Service". Academy Publisher, 232-236, 2009.

[11] Kabalan K.Y., Smari W.W. and Hakimian J.Y. "Adaptive load Sharing in Heterogeneous system: Policies, Modifications and Simulation". CiteSeer^x, 2008.

[12] Lin H. C. and Raghavendra C.S. "A Dynamic Load Balancing Policy with a Central Job Dispatcher (LBC)". IEEE,1992.

[13] Lüling R., Monien B. and Ramme F. "Load Balancing in Large Networks: A Comparative Study. In Proceedings of the 3rd IEEE Symposium on Parallel and Distributed Processing". CiteSeer^x, 1991.

[14] Penmatsa S. and Cronopoulos A.T. "Dynamic Multi User Load Balancing in Distributed Systems". IEEE International Parallel and Distributed Processing Symposium, 2007.

[15] Sánchez D., Macías E. Mª. and Suárez Á. "Effective Load Balancing on a LAN-WLAN Cluster". PDPTA 2003: 473-479

[16] Weinrib and Shenker S. "Greed is not Enough: Adaptive Load Sharing in Large Heterogeneous Systems". INFOCOM, 986-994, 1988

[17] Xu C. and Lau F.C.M. "Load Balancing in Parallel Computers: Theory and Practice". Kluwer Academic Press, 1997.