A Comparative Study of Various Load Balancing Algorithm in Parallel and Distributed Multiprocessor System

Mamta Kumari
APS University
Rewa (M.P)

Rakesh Kumar Katare
HOD, APS University
Rewa (M.P)

ABSTRACT
In modern days parallel and distributed computing is one of the greatest platform for research and innovation in the field of computer science. Rapid growth of communication network and need to solve large scale problem, complexity and efficiency of the system as a whole is the key issue. Load balancing is one of the most important problem in attaining high performance in parallel and distributed systems which may consist of many heterogeneous resources connected via one or more communication networks. Load balancing is the process of distributing or reassigning of load over the different nodes which provide good recourse utilization and better throughput. Although intense work has been done in the algorithm design of load balancing and its performance measure issues, we present a brief overview of various load balancing conditions and its algorithmic classification for tailor made applications. Various criteria were discussed for the classification of load balancing helping designers to compare and choose the most suitable algorithm for the application.

Keywords
Load–balancing, Hetrogeneous-resourse, Resource-utilization

1. INTRODUCTION
One of the key issues in algorithm designs for paral- lel and distributed computing is that of load balancing where n interacting task are allocated among m pro- cessing nodes arranged in a given topology in order to minimize or maximize some criteria. Load balancing improves the distribution of workloads across multiple computing resources such as computer cluster, network link or disk drives. The drive behind this load balancing is two fold- efficiency and extensibility. Various issues related to load balancing are also analysed during the classification [1,2] .

2. ISSUES WHILE DESIGNING LOAD BALANCING ALGORITHM
In distributed system, communication link are of finite bandwidth and the nodes are physical far apart so load balancing algorithm need to take consideration of task migration. There must be some constrained while task partitioning. Also load on each processor as well as system as a whole varies time to time and capacity of the processing node may vary in the system. So, taking into consideration of various issues, Load balancing can be generalized into four basic steps:

- Monitoring processors load and its state.
- Exchanging load and state information between processors
- Calculating the new workload
- Actual data migration between processors / nodes

2.1 Load Balancing Aims

- To improve the performance substantially
- To have a backup plan in case of system failure
- To accommodate future modification in the system
- to optimize resource use and minimise response time and
- avoid overload of any single resources

Load sharing and balancing in a locally distributed system is the process of transparently distributing work submitted to the interconnection network .Shifting work from an heavily loaded nodes to the lightly loaded process performance of the network can be improved substantially [3,4] In a multiprocessor system each processing node exhibits different stage depending upon load poses at that time. It varies time to time just like process. Following are various stages of nodes .

(i) Heavy_load_node-workload more than a threshold value.
(ii) Light_load_node-workload much less than a threshold value.
(iii) Proper_load_node_workload approaching to threshold value.
(iv) Busy_load_node-maximum involvement.
(v) No_load_node- no workload on the node [5,6]

Following are some Important points to be consider while designing an algorithm for load balancing

- Estimation of workload of each node and total workload of the system
- Nature of workload to be transferred
- Comparison of workload of each node with its neighboring nodes
- Stability of different communication net- work system
- Performance of a local and the overall net- work system
- Interaction between various neighboring nodes
- Selecting of nodes etc.

The load considered can be in terms of CPU load, amount of memory used, delay or Network load [7,8]

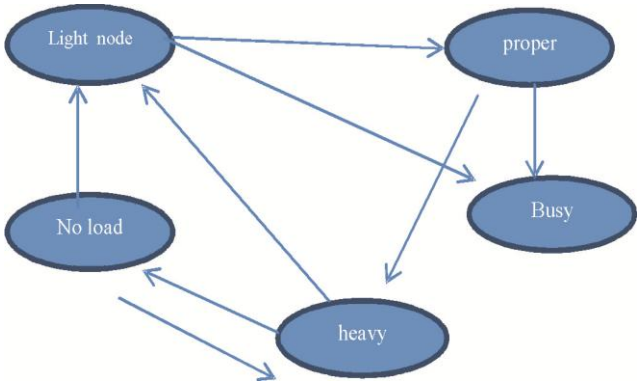

Figure.1 Various stages of nodes/processors during load balancing

3. CLASSIFICATION OF VARIOUS LOAD BALANCING ALGORITHM

A large number of scheduling and algorithm were developed and proposed in the area of load balancing in parallel and distributed system.

Depending upon the current status and condition, load balancing strategies can be classified as [1,2]

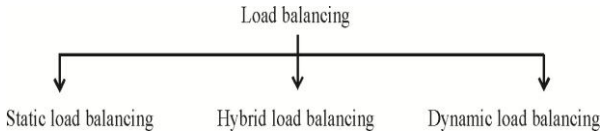

Fig. 2 Classification of Load Balancing

3.1 Static load balancing

Static load balancing is used when computation and communication requirement of a given problem are known prior .Assignment of task to the various processors is performed before execution. Load balancing decisions are made deterministically or probabilistically at compile time according to the performance of computing node and remains constant during runtime. Number of task are fixed in this approach. Static load balancing scheme is non preemptive in nature. Now, Static load balancing are further categories as [7,8]

3.1.1 Classic Round –Robin Algorithm.

Round Robin is undoubtedly the most widely used algorithm. This algorithm assign task sequentially and evenly to all nodes. It's easy to implement and easy to understand. Here's how it works. Let's say you have 2 servers waiting for requests behind your load balancer. Once the first request arrives, the load balancer will forward that request to the 1st server. When the 2nd request arrives (presumably from a different client), that request will then be forwarded to the 2nd server. Because the 2nd server is the last in this cluster, the next request (i.e., the 3rd) will be forwarded back to the 1st server, the 4th request back to the 2nd server, and so on, in a cyclical fashion. In this algorithm inter-process communication is not required. This scheme is useful for job of equal processing time and a node of same capabilities. But not efficient when uneven tasks and nodes having different capacities. [5 ,7]

3.1.2. Weighted Round Robin Algorithm

For the 2nd scenario mentioned above, i.e., Server 1 having higher specs than Server 2, you might prefer an algorithm that assigns more requests to the server with a higher capability of handling greater load. One such algorithm is the Weighted Round Robin. The Weighted Round Robin is similar to the Round Robin in a sense that the manner by which requests are assigned to the nodes is still cyclical, albeit with a twist. The node with the higher specs will be apportioned a greater number of requests. Basically, when you set up the load balancer, you assign "weights" to each node. The node with the higher specs should of course be given the higher weight. You usually specify weights in proportion to actual capacities. So, for example, if Server 1's capacity is 5x more than Server 2's, then you can assign it a weight of 5 and Server 2 a weight of 1. It is one of the simplest scheduling algorithms that utilize the principle of time slices. Here time is divided into multiple slices and each node is given a particular time interval. Each node is given a quantum and in this given quantum node has to perform its operations. If the user request completes within time quantum then user should not wait otherwise user have to wait for its next slot. It means that this algorithm selects the load randomly, while in some case some server is heavily loaded or someone is lightly loaded [5, 7]

3.1.3 Central Manager algorithm

The Central Manager Algorithm uses central node as a coordinator to distribute the workload among the slave processors. The rule that is followed to choose the slave processor is to assign the job to the processor that have the least load. The central processor is able to gather all slave processors load information and take the decision of load balancing depending on this information so we expected a good performance when applying this algorithm. The main disadvantage of this type is the high degree of inter-process communication that could make a bottleneck state [16,17]

3.1.4 Threshold Algorithm

In this algorithm, the processes are assigned immediately to the server nodes upon creation . Server nodes for new job are selected locally without sending remote messages. Each server node keeps a copy of the system's current load. The load of a processor can be characterized by one of the three levels: under loaded, medium, and overloaded. Two threshold parameters t-under and t-upper can be used to describe these levels:

Underloaded: workload <t_under
Medium : t_under d<workload d<t_upper
Overloaded: workload >t_upper

At first, all processor are assumed to be under-loaded. When the load of a processor changes, it sends a message to all other processors that are related with the new load state, updating them as to the actual current load state of the entire system. Each process gets allocated locally when the processor is under-load , otherwise, a remote under-loaded processor is selected, and if no such host exists, the process is also allocated locally . Thresh-olds algorithm have large number of local process allocations so it has low inter-process communication that decreases the overhead of the whole system which leads to improve the performance . The main disadvantage of the threshold algorithm is that all processes are assigned locally when the processors are overhead. It does not take the execution time in consideration which impacts the performance of the entire system.[3, 8]

3.1.5 Randomized Algorithm

Randomized Algorithm (RA) uses random number *i* selecting a computing node for processing having any information about current or previous load on the node. The computing nodes are selected randomly following random number generated based on a statistical distribution. Basically it works

for particular special purpose application .No inter-process communication is not required. Not much efficient algorithm as response time is poor [9 ,13]

3.1.6 Throttled Load Balancing Algorithm

This algorithm is totally based on the allocation of request to virtual machine. Here client will first request the load balancer to check the right virtual machine which access that load easily and performs the operations request by client or user. In this algorithm the load balancer maintains an index table of virtual machines as well as their states (Available or Busy). Therefore the client first requests the load balancer to find a suitable Virtual Machine to perform the required operations. These dynamic algorithms are being experimentally performed using the cloud analyst tool which gives the output with respect to virtual machine.

[10]

3.2 Hybrid Load balancing:- In hybrid load balancing both static and dynamic are merged together exploit the benefit of both algorithm.

3.3 Dynamic load balancing

Dynamic load balancing method is applied in situation where no prior estimation of load distribution are possible .At the time of parallel program execution it is decided that how much work is being assigned to each processor, in many cases static load balancing is either impossible to implement or lead to load imbalance. Dynamic load balancing works well on heterogeneous system. Task can be redistributed to any processor while runtime hence overloading and underloading problem becomes minimum. But high communication overhead occurs. Also, system overhead increases because it is preemptive in nature. [4,7]

3.3.1 Policies or Strategies in dynamic Load balancing

In order to define a Load-Balancing Algorithm completely, the main four sub-strategies have to be defined. This will provide as a framework for describing and classifying different existing load balancing algorithm facilitating the task of identifying a suitable load balancing strategies.

- Initialization policy:- The initialization approach specifies the system, which invokes the load balancing behaviour. This may be episodic or event-driven initiation. Episodic initiation is a time based initiation in which load information is exchanged every preset time interval. Event-driven is usually a load dependent policy based on the observation of local load.
- Location policy:-This policy specifies the location at which the algorithm itself is executed. Location policy may be central or distributed. Distributed algorithm are further classified as-Synchronous and asynchronous. A synchronous load balancing algorithm must be executed by all nodes present in the system simultaneously and in asynchronous algorithm, it can be executed at any moment in a given node, with no dependency on what is being executed at other nodes.
- Information exchange policy:-This specifies the information and load flow thought the network. The information used by the dynamic load balancing algorithm for decision making can be local information or gathered from the surrounding processors.
- Load selection policy:-This policy is the most important part of the whole system in which the processing node decide from which node to exchange load [13, 15]

In a distributed system, dynamic load balancing can be done in two different ways: distributed and non-distributed. In the distributed one, the dynamic load balancing algorithm can be further classified as cooperative and non-cooperative. In the first, the nodes work side-by-side to achieve a common objective like to improve the overall response time. In the second form, each node works independently toward a goal local to it, that is to improve the response time of a local task. Dynamic load balancing algorithms of distributed nature usually generate more messages than the non-distributed ones because, each of the nodes in the system needs to interact with every other node. A benefit of this is that even if one or more nodes in the system fails, it will not cause the total load balancing process to halt, it instead would effect the system performance to some extent. Distributed dynamic load balancing can introduce immense stress on a system in which each node needs to interchange status information with every other node in the system. It is more advantageous when most of the nodes act individually with very few interactions with others. In non-distributed type, either one node or a group of nodes do the task of load balancing [9, 11] Non-distributed dynamic load balancing algorithms can take two forms: centralized and semi-distributed.

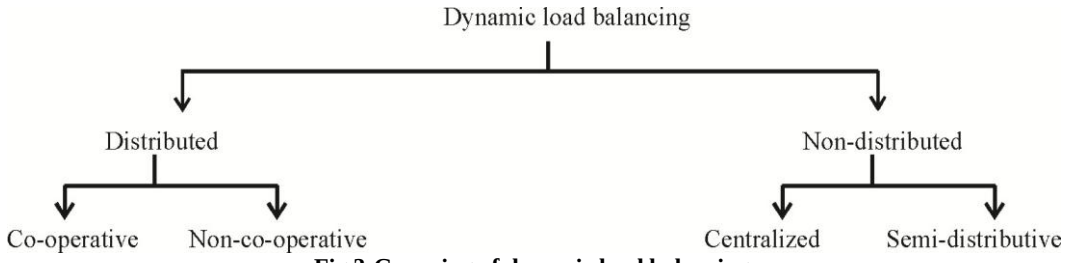

Fig.3:Grouping of dynamic load balancing

In the centralized scheme, the load balancing algorithm is executed only by a single node in the whole system: the central node. This node is solely responsible for load balancing of the whole system. The other nodes interact only with the central node. In semi-distributed form, nodes of the system are partitioned into clusters, where the load balancing in each cluster is of centralized form. A central node is elected in each cluster by appropriate election technique which takes care of load balancing within that cluster. Hence, the load balancing of the whole system is done via the central nodes of each cluster. Centralized dynamic load balancing takes fewer messages to reach a decision, as the number of overall interactions in the system decreases drastically as compared to the semi-distributed case. However, centralized algorithms can cause a bottleneck in the system at the central node and also the load balancing process is rendered useless once the

central node crashes. Therefore, this algorithm is most suited for networks with small size [17 ,18]

3.3.2 Central Queue Algorithm

Central Queue Algorithm [4] stores new activities an unfulfilled requests on a cyclic FIFO queue on the main host. Each new activity arriving at the queue manager is inserted into the queue. Then, whenever a request for an activity is received by the queue manage, it removes the first activity from the queue and sends it to the requestor. If there are no ready activities in the queue, the request is buffered, until a new activity is available. When a processor load falls under the threshold, the local load manager sends a request for a new activity found in the process-request queue, or queues the request until a new activity arrives. This is a centralize initiated algorithm and need high communication among nodes.

3.3.3 Local Queue Algorithm

This algorithm supports inter-process migration. The main concept in local queue algorithm is static allocation of all new process with process migration initiated by the host when its load falls under the predefine minimum number or ready processes (threshold limit). Initially, new processes created on the main host are allocated on all under loaded hosts. From then on, all the processes created on the main host and all other hosts are allocated locally. When the local host gets under load it request for the activities from the remote host. The remote hosts that look up its local list for ready activities and compares the local number of ready activities with the received number. If the former is greater than the later, then some of the activities are passed on to the requestor host and get the acknowledgement form the host. This is a distributed co-operative algorithms and required inter-process but lesser as compared to central queue algorithm [7, 9]

There are some other basic dynamic load balancing parallel algorithmic paradigm conditions prevails in parallel and distributed computing which is follows as:

- *Sender initiated Diffusion method (SID)*

The SID strategy is a, local, nearest-neighbor dif-fusion approach which employs overlapping balancing domains to achieve global balancing. For an N processor system with a total system load L, a diffusion approach, such as the SID strategy, will cause each processor's load to converge to L/N. Balancing is performed by each processor whenever it receives a load update message from a neighbor indicating that the neighbors load, i<Ideal Load, where Ideal Load is a preset threshold. Each processor is limited to load information from within its own domain, which consists of itself and its immediate neighbors [9,12]

- *Receiver initiated Diffusion method (RID)*

The RID strategy is converse of SID in that it is a receiver initiated approach as opposed to sender initiated approaches .However, besides the fact that in the RID strategy under loaded neighbor request from overloaded nodes, certain subtle difference exit between them. Firstly balancing process initiated by any node whose load drops below threshold value. Secondly, upon receipt of a load request, a processor will fulfill the request only up to an amount equal to half of its current load. [19]

- *Symmetric*

This algorithm is combination of both sender initiated diffusion and receiver initiated diffusion method.

- *Hierarchical load balancing Method (HBM)*

The HBM strategies organizes the multicomputer system into hierarchy of balancing domains, thereby decentralizing the balancing process. Specific processors are designated to control the balancing operations at different levels of hierarchy [19, 20]

- *Tree-Based Parallel Load Balancing Method*

Here we present three tree based parallel load balancing methods to efficiently deal with load unbalance problem on distributed memory interconnection network.

- *The maximum Cost spanning Tree Parallel Load Balancing Method (MCSTPLB)*

The main idea of MCSTPLB method is to find a maximum cost spanning tree from processors graph ob-tainted from initial partitioning of graph and it tries to balance the load of processor.

- *The Binary Tree Parallel Load Balancing (BTPLB) Method*

The BTPLB method is similar to the MCSTPLB method .The only difference between these two method is that the MCSTPLB method is based on maximum cost spanning tree to balance the computational load of the processors while the BTPLB method is based on binary tree .

- *The Condensed Binary Tree Parallel Load-Balancing (CBTPLB) Method*

The main idea of the CBTPLB method is based on to group processor graph into meta-processors. Each meta-processor is a hypercube this group processor graph is called as condensed processor graph [19, 20]

- Gradient method:- The gradient model is a demand driven method .Basic concept is that underloaded processors inform other processors in the system of their state, and overloaded processors respond by sending a portion of their load to the nearest lightly loaded processor in the system.[10,11]
- Dimension Exchange Method(DEM) A DEM is similar to the HBM method in which small domain are balanced first and these are combined to form large domains until the entire system is balanced. This is a synchronized scheme designed for hypercube system but may be applied to other topologies with some modification. In case of N-processor hypercube, balancing is performed iteratively in each of the log N dimensions, it facilitates debugging and performance analysis.
- Divide and Conquer (DAC):- In this approach a big problem is divided into small problems and then we start solving it (i.e conquer). In this way dynamic load balancing a parent process divides its workload into several smaller pieces and assigns them to a number of child processes. The child process compute their workload in a parallel fashion and the result are mergers by parent.
- Pipeline: In this algorithm the output of one stage works as an input for the next stage, hence a pipe is created called virtual pipe. A number of processor creates a virtual pipe. A continuous data stream is fed into the pipeline and the processes execute at different pipeline stages simultaneously in an overlapping fashion.

4. CONCLUSION AND FUTURE WORK

This paper focuses on the various load balancing algorithm which help developer to compare and design a new application paradigm. Dynamic load balancing techniques give better performance than static load balancing technique. Good load balancing algorithms depended on good task scheduling techniques. There are many parameters to measure the efficiency of load balancing techniques such as response time, resource utilization, overhead associated, fault tolerant, centralized or decentralized, reliability, stability, adaptability, cooperative, process migration, scalability and throughput. This classification compare and analyze different algorithm and design a new tailor for needs. In future, we intend to develop a framework for application with load balancing that utilizes this classification and help to design and tailor his own algorithm.

5. REFERENCES

[1] M. Willebeek-LeMair and A. P. Reeves, ‘‘A general dynamic load balancing model for parallel computers,’’ Tech: Rep. EE-CEG-89-1) Comell School of Electrical Engineering. 1989.

[2] Y. T. Wang and R.I. T. Morris, ‘‘Load sharing in distributed systems,’’ IEEE Trans. Comput., Vol. C-34, pp. 204-211, Mar. 1985. M. J. Berger and S.H. Bokhari, ‘‘A partitioning strategy for non-uniform problems on multiprocessors,’’ IEEE Trans. Corput., vol. C-36, pp. 570-580, May 1987.

[3] Fonlupt, C., Marquet P. and Dekeyser, J.: Dataparallel load-balancing strategies. Parallel Computing 24 (1998) 1665-1684.

[4] Dekeyser, J. L., Fonlupt, C. and Marquet, P. : Anal-ysis of Synchronous Dynamic Load Balancing algo-rithms’’, Parallel Computing: State-of the Art Per-spective (ParGo95), Volume 11 of Advances in Parallel Computing, pages-455-462, Gent, Belgium (September 1995).

[5] Y. Wang and R. Morris, ‘‘Load balancing in distributed systems.’’ IEEE Trans. Computing. C 34, no.3, pp. 204-217, Mar. 1985.

[6] Mr. Gaurav Sharma and Jagjit Kaur Bhatia, ‘‘A review on different approaches for load balancing in computational grid’’, Journal of Global Research in Computer Science, Volume 4, No. 4, April 2013.

[7] Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma, ‘‘Performance Analysis of Load Balancing Algorithms’’, World Academy of Science, Engineering and Technology, 2008.

[8] S. F. EI-Zoghdy and S. Ghoniemy, ‘‘A Survey of Load Balancing In High-Performance Distributed Computing Systems,’’ International Journal of Advanced Computing Research, Volume 1, 2014

[9] G. Cybenko, ‘‘Dynamic load balancing for distributed memory multiprocessors.’’ J. Parallel and Distributed Comput., Vol. 7279-301, October, 1989.

[10] Hamidzadeh, B., Lilja, D. J. and Atif, Y. : Dynamic scheduling. techniques for heterogeneous computing systems. Concurrency : Practice and Experience, vol. 7 (1995) 633-652.

[11] Saletore, V. A : A distributive and adaptive dynamic load balancing scheme for parallel processing of medium grain tasks. Proceedings of the 5th Distributed Memory Conference (April 1990) 995-999

[12] K. G. Shin and Y.C. Chang. ‘‘Load sharing in distributed real time systems with state-change broad-casts,’’ IEEE Trans. Comput., pp. 1124-1142, Aug. 1989. V. A. Saletore, ‘‘A distr

[13] F. C. H. Lin and R. M. Keller, ‘‘The gradient model load balancing method,’’ IEEE Tran. Software Engineering 13, 1987, pp. 32-38.

[14] Zaki, M. J., Li, W. and Parthasarathy, S. : Customized dynamic load balancing for a network of workstations. Proceedings of the 5th IEEE Int. Symp., HPDC (1996) 282-291.

[15] D. P. Bertsekas and J. N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods. Englewood Cliffs, NJ: Prentice-Hall.

[16] K.M. Baumgarther, R. M. Kling, and B.W. Wah, “Implementation of GAMMON: An efficient load balancing strategy for a local computer system,” in Proc. 1989 Int. Conf Parallel Processing, Vol. 2, Aug. 1989, pp. 77-80.

[17] William Leinberger, George Karypis, Vipin Kumar, “Load Balancing Across Near-Homogeneous Multi-Resource Servers”, 0-7695-0556-2/00, 2000 IEEE.

[18] H.S. Stone, “Critical Load Factors in Two-Processor Distributed Systems,” IEEE Trans. Software Engg., Vol. 4, No. 3, May, 1978.

[19] C.H. Lin and R.M. Keller, ‘‘The gradient model load balancing method,’’ IEEE Tran. Software Eng., vol. 13, no. 1, pp.32-38, Jan. 1987.

[20] G.C. Fox, ‘‘A review of automatic load balancing and decomposition methods for the hypercube,’’ California Institute of Technology, C3P-385, Nov. 1986.