International Journal of Research in Advent Technology, Vol.2, No.4, April 2014
E-ISSN: 2321-9637

A Dynamic Approach for Load Balancing using Clusters

Shweta Rajani [1], Renu Bagoria [2]
Computer Science[1,2],Global Technical Campus, Jaipur[1],JaganNath University, Jaipur[2]
Email: shwetarajani28@yahoo.in[1]

Abstract-Rapid increase and advancement in the use of computer and internet has increased the demand for resource sharing since it has increased the amount of load across internet to a vast level. This situation can be handled either by increasing the size of servers or by effectively distributing the workload among multiple servers. The paper discusses various techniques of load balancing and a newly proposed design and algorithm with a clustered approach to perform dynamic load balancing.

Index Terms-Load Balancing; Static Load Balancing; Dynamic Load Balancing; Issues in designing load balancing algorithms; clustered approach.

1. INTRODUCTION

Load Balancing refers to distributing the processes to the nodes in the system so as to equalize the workload among the nodes. Load balancing algorithm tries to balance the total system load by transparently transferring the workload from heavily loaded nodes to lightly loaded nodes to ensure good overall performance.A Load Balancer is a software program which listens to the port where external clients connect to access services[1]. A load Balancer may be combined with a decision making system itself or it should report the load conditions to one or more back-end servers to make balancing decision. However, the basic goal of all load balancing algorithms is to maximum total system throughput.

Load Balancing Technique has several advantages:

- Increases the performance of the system
- Reduces mean job response time
- Increases Processor Utilization
- Maximum resource utilization leads to maximum throughput.
- Ensures that no server is overwhelmed

For any Load Balancing Algorithm, there are three major parameters that define the strategy[1]:

i. Who makes the load balancing decision ?
ii. What information is used to make the load balancing decision ?
iii. Where the load balancing decision is made ?

The first question classifies the technique into two types: Sender-initiated and Receiver-initiated. In sender-initiated strategy, highly loaded nodes search for lightly loaded nodes and in receiver-initiated strategy, lightly loaded nodes search for highly loaded nodes.

The second question again classifies the technique into two types: Global and Local. In global strategy, all nodes in the network are considered while searching for lightly loaded node and in local strategy, nodes are divided into groups and balancing decision is made locally.

The third question classifies the technique into two types: Centralized and Distributed. In centralized strategy, all nodes share their load information with one single node which makes the load balancing decision and in distributed strategy, all nodes perform a broadcast of their load information and each node makes the balancing decision.

2. TYPES OF LOAD BALANCING ALGORITHMS

Depending upon the basis of number of processes and load in the system, the load balancing algorithms can be categorized into two types:

2.1 *Static Load Balancing:*

Static Load Balancing is performed when the system load and number of processes is fixed and known at compile time. All parameters are fixed for the system. Static Load Balancer makes balancing decision on the basis of average workload of the system. Hence the Static Load Balancing takes less time to execute and is simpler. But it is not suitable

for the environments with changing workloads. Hence, a dynamic approach is required.

2.2 *Dynamic Load Balancing:*

Dynamic Load Balancing is performed when the system load and number of processes is likely to change at run time. In this case, there is a need of consistently monitoring the system load. This increases the overhead and makes the system more complex. Dynamic Load Balancer makes balancing decision on the basis of current state of the system.

Hence Static Load Balancing is simpler, faster and cost-effective than Dynamic Load Balancing but is not suitable for the system with changing workloads. Therefore, Dynamic approach is much more efficient for distributed networks.

3. ISSUE IN DESIGNING LOAD BALANCING ALGORITHMS:

There are several issues to be considered while designing a load balancing algorithm. These are discussed briefly in[3]:

3.1. *Load Estimation Policy*

A node's workload can be estimated by following parameters

- Total no. of processes at the node
- Resource demand of these processes
- Instruction mixes of these processes
- Architecture and speed of node's processor
- Sum of remaining service times of all the processes in the networks

3.2. *Process Transfer Policy*

These are the policies used to decide whether the node is heavily loaded or lightly loaded. This is done by deciding a threshold value for the workload. These are of two types: Static and Dynamic. In Static policy, there is a predefined threshold value for each node which does not vary with dynamic changes in the workload. In dynamic policy, threshold value is calculated as a product of average workload of all the nodes and a predefined constant C. C depends on the processing capability of a node relative to processing capability of all other nodes. Threshold policies are used to decide the region to which node belongs.

3.3. *Location Policy*

These policies are used to select the destination node for the process's execution. We can adopt either of the following types of location policies:

3.3.1. *Threshold policy:* A node is selected at random and a check is made to determine whether the transfer to that node leads it to an overloaded condition, if not, the process is transferred to that node, if yes, then another node is selected at random and probed in the same way until a probe limit L is reached.

3.3.2. *Shortest Policy:* In this, L distinct nodes are chosen at random and each is polled to determine its load. The process is transferred to the node with minimum load value.

3.3.3. *Bidding Policy:* In this, each node can be either a contractor or a manager. Manager is the node having a process to be transferred; Contractor is the node that is able to accept a remote process. Manager broadcasts request-for-bid messages to all the nodes in the network, and then the contractor nodes return the bids to manager. The manager then chooses the best bid and transfers the process to the winning contractor node.

3.3.4. *Pairing Policy:* In this, two nodes with greatly varying loads are selected and paired and load balancing is carried out between them, several such pairs may be created in a network. The processes to be migrated are selected by comparing their expected completion time on the current node to that on the partner node, including the migration delay. During the time a pair is in force, both members will reject any other pairing requests.

3.4. *State Information Policy*

Several policies can be adopted to exchange node state information among the nodes in the network.

3.4.1. *Periodic Broadcast:* In this, each node broadcasts its state information to after every t units of time. But this process has certain demerits; it increases the traffic, may result into fruitless messages and it is not scalable at all.

3.4.2. *Broadcast when state changes*: In this, a node broadcasts its state information only when its state changes. This can be even improved by observing that is not necessary to report every small change, rather it should broadcast only when it can participate in the load balancing process.

3.4.3. *On-Demand Exchange:* In this, a node broadcasts a StateInformationRequest message when its state switches from normal load region to either overloaded region or under loaded region. On receiving this message, then other nodes send their

current state to the requesting node. This method can be further improved if we include the status of the requesting node in the request message and only those nodes should reply that can contribute to the load balancing process.

3.4.4. *Exchange by polling:* This method is adopted to reduce the network traffic. In this, a node can search for a contributing partner at random by polling the other nodes one by one. Therefore, the state information is exchanged only between the polling node and the polled node.

3.5. *Priority Assignment Policy*

When the process migration has been accomplished using the series of policies, there is a need to devise a priority assignment rule to schedule local and remote processes at a particular node. Any of the following three rules can be adopted:

3.5.1. *Selfish rule:* In this, local processes are given higher priority than remote processes, but this yields the worst response time.

3.5.2. *Altruistic rule:* In this, remote processes are given higher priority than local processes, and it gives the best response time.

3.5.3. *Intermediate rule:* In this, the priority depends on the no. of local or remote processes at that node. If local processes are more than remote processes then local processes are preferred, otherwise remote processes are preferred.

3.6. *Migration Limiting Policy*

It is an important policy that decides "how many times a process should be allowed to migrate?" Two policies are there to decide this: Uncontrolled and Controlled.

3.6.1. *Uncontrolled:* In this, remote process is treated same as the local process and it is allowed to migrate any no. of times, but this is instable.

3.6.2. *Controlled:* A stable approach is to distinguish the remote process by a local process and use a migration count to limit the no. of migrations. For normal size processes the migration limit is generally set to 1, and for large processes it can be set greater than 1.

4. PROPOSED DESIGN

Our proposed design uses a clustered approach for load balancing. Workstation clusters are being recognized as the most promising computing resource of the near future. A large-size cluster, consisting of locally connected workstations, has power comparable to a supercomputer, at a fraction of the cost. Distributing the total computational load across available processors is referred to as load balancing [4].

Effective load-balancing of a cluster of computing nodes in a distributed computing system relies on accurate knowledge of the state of the individual nodes. This knowledge is used to judiciously assign incoming computational tasks to appropriate node, according to some suitable load balancing strategy[4].

In the proposed strategy, two concepts are mainly referred.

In [1], there is a design with one Load Balancer communicating with all the nodes and monitoring their load. This load balancer reports the load to two back-end servers. The servers finally make the balancing decision and return the address of the suitable node to which the overload should be transmitted.

In [2], there is a different approach, it uses one supporting node with each primary node and in case of overload at node Ni, an interrupt service routine generates an interrupt and the overload is transferred to its supporting node and it also uses a priority scheme, if the priority of the incoming process at the supporting node is greater than that of the currently running process, then the current process is interrupted and assigned to a waiting queue and the incoming process is allowed to run at the supporting node. Otherwise the current process continues and incoming process is in waiting state until the current process is completed.

The proposed design uses a clustered approach in which each cluster maintains three nodes and each cluster has a supporting node. Each cluster maintains a queue for to store the load of its nodes.The load balancer maintains priorities of the process in the system. This design reduces the cost of infrastructure used in [2], and improves the service offered by [1] by using clusters as the central load balancer has to communicate with cluster manager rather than individual nodes. Fig.1 illustrates the proposed design.

4.1 Notations:

- Each cluster $C_i$consists of 3 nodes.
- Each cluster C_i is associated with a supporting node $SN_i$to balance the overload at the cluster.

- The nodes inside the cluster are denoted by N_{ij} i.e. *jth node of ith cluster*
- For each cluster, there is a load queue $Q_i$of size 3 that stores the load of node NodeN_{ij}atQ_{ij}
- There is a priority queue ***P***that contains processes arranged in order of their priorities.
- It also contains a waiting queue ***W***that contains the processes that are currently interrupted and waiting for processor. These processes are also arranged in order of their priorities.

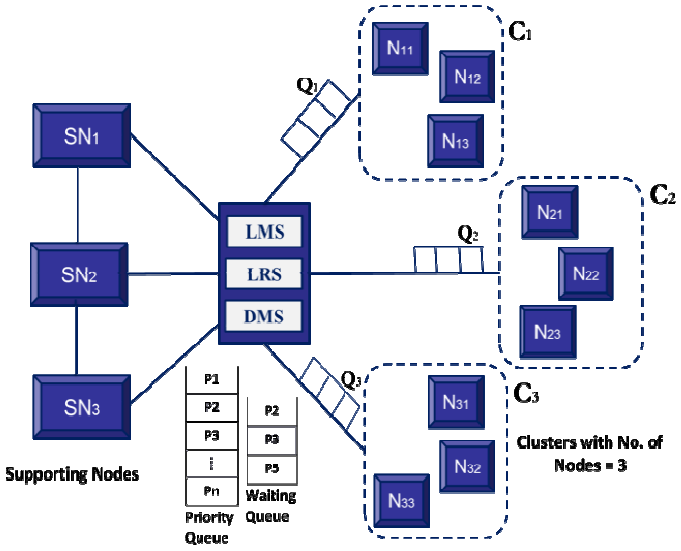

Fig. 1: Proposed design for Load Balancing using cluster of three nodes

4.2 Description of Design:

- The Load Balancer comprises of three types of servers: Load Monitoring Server (LMS), Load Reporting Server (LRS) and Decision Making Server (DMS).
- A Load Reporting server is located at each node to collect and report its load, stores the load of Node N_{ij} in the Load queue $Q_i$at Q_{ij}. For example, if loads at nodes of first cluster are 2, 4, 3; then the queue Q**1** will be

2	4	3

- The Load Reporting Server at central load balancer reports the load of all Supporting Nodes.
- The Load Monitoring Server monitors the load queue of each cluster and compares the load with maximum load threshold.
- We decide a maximum allowable load for the nodes as the threshold. Suppose every node can withstand a load of 5 units without any degradation in the performance.
- In this case, whenever any element of queue gets over the maximum specified load, an interrupt is generated and the Load Balancer makes the balancing decision and transfers the overload to the suitable supporting node.
- If a supporting node is free then the overload is transferred to it, otherwise the priority of the currently running process and that of the incoming process is compared by the Decision Making Server (DMS).
- If the incoming process has higher priority, then the currently running process is interrupted and inserted to waiting queue$W_i$and the incoming process is allowed to execute on SN_i.Otherwise, the current process continues to run and the incoming process is inserted to the waiting queue.
- The waiting is queue is checked periodically and the waiting processes are scheduled.

5. CONCLUSION

The above presented algorithm works well and ensures that no process suffers starvation and no processor is overwhelmed. The presented design works for cluster with number of nodes=3. We are planning to design a modified approach to work for 'n' number of clusters. The optimization of algorithm is another task for future research. As the era of distributed networks and systems increasingly comes into practice, the demand for more organized and less complex structures rises. The presented design is likely to facilitate such demand in an easy way and lesser implementation cost.

References

[1] Ankush P. Deshmukh, Prof. KumarswamyPamu, *Applying Load Balancing: A Dynamic Approach*,International Journal of Advanced Research in Computer Science and Software Enginnering, Volume 2, Issue 6, June 2012

[2] Parveen Jain, Daya Gupta,*An Algorithm for Dynamic Load Balancing in Distributed Systems with Multiple Supporting Nodes by Exploiting the Interrupt Service*, International Journal of Recent Trends in Engineering, Vol 1, No. 1, May 2009

[3] RutujaJadhav, SnehalKamlapur, I Priyadarshini, *Performance Evaluation in Distributed System using Dynamic Load Balancing*,International Journal of Applied Information systems(IJAIS)–ISSN:2249-0868, Volume 2, No.7, February 2012

[4] JeanGhanem, *Implementation of Load Balancing Policies in Distributed Systems*, The University of New Mexico Albuquerque, New Mexico, June 2004

[5] Ahmad Dala'ah, *A Dynamic Sliding Load Balancing Strategy in Distributed Systems*, The International Journal of Information Technology, Vol 3, No.2, April 2006.

[6] Mayuri A. Mehta, *Designing an Effective Dynamic Load Balancing Algorithm Considering Imperative Design Issues in Distributed Systems*, International Conference on Communication Systems and Network Technologies 2012 IEEE.

[7] Sandeep Sharma, Sarabjit Singh, Meenakshi Sharma, *Performance Analysis of Load Balancing Algorithms*, World Academy of Science, Engineering and Technology 14 2008

[8] Yong Meng TEO, Rassul AYANI, *Comparison of load balancing strategies on cluster-based web servers*, Transactions of the SocietyforModeling and Simulation, 2001

[9] Mayuri A. Mehta, Devesh C. Jinwala, *Analysis of Significant Components for Designing an Effective Dynamic Load BalancingAlgorithm in Distributed Systems*, Third International Conference on Intelligent Systems Modelling and Simulation, 2012

[10] Hao Jiang, Luo, Feng, Tang, Yin, *DALB: A Dynamic Application-sensitiveLoad Balancing Algorithm*, International Conference on Computer Science and Service System, 2012