Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 6, June 2014, pg.843 – 851

RESEARCH ARTICLE

A Semi-Distributed Load Balancing Algorithm Using Clustered Approach

Shweta Rajani[1], Renu Bagoria[2]

[1]Computer Science, JaganNath University, India

[2]Computer Science, JaganNath University, India

[1] shwetarajani28@yahoo.in ; [2] renubagoria@gmail.com

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. In a distributed computing system made up of different types of processors, each processor in the system may have different performance and reliability characteristics. In order to take advantage of this diversity of processing power, a modular distributed program should have its modules assigned in such a way that the applicable system performance index, such as execution time or cost, is optimized. This situation can be handled either by increasing the size of servers or by effectively distributing the workload among multiple servers. The presented paper discusses various techniques of load balancing and proposes a new design and algorithm which uses a clustered and semi-distributed approach to perform dynamic load balancing.

Keywords— Load Balancing, Load Balancing taxonomy, Static Load Balancing, Dynamic Load Balancing, Issues in designing load balancing algorithms, semi-distributed approach

I. INTRODUCTION

Load Balancing refers to distributing the processes to the nodes in the system so as to equalize the workload among the nodes [1]. 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. Load balancing enhances the performance of the servers, leads to their optimal utilization and ensures that no single server is overwhelmed [2]. Traditional load balancing techniques are not that efficient to provide healthy environment for today's requirements. By studying pros and cons of different techniques used for load balancing, we are specifically giving priority to Dynamic load balancing method rather than Static load balancing.

Load Balancing based on the idea of migration of excess load from heavily loaded node to lightly loaded ones. The problem starts with how to determine, when to migrate a process or task. This solution is typically based on local load situation: for example, a simple procedure may be the comparison of the load between various nodes with those of the neighbouring node and a determination of the node to which the task is to be migrated. But the two nodes, each one having two task may not be equally loaded as in distributed environment; the nodes are of heterogeneous nature. Now, load estimation can be estimated by means of processing power of the node. Processing power does not mean only the processing speed of Processor; it includes the overall configuration of node [3].

With all these factors taken into account, load balancing can be generalized into four basic steps:

- Monitoring processor load and state
- Exchanging load and state information between processors
- Calculating the new work distribution
- Actual data movement.

Distributed-computing system is a collection of heterogeneous and geographically dispersed computing elements that allows nodes to cooperatively execute applications in a parallel fashion [4]. Load balancing is one of the most important problems in attaining high performance in distributed systems which may consist of many heterogeneous resources connected via one or more communication networks. In these distributed systems it is possible for some computers to be heavily loaded while others are lightly loaded. This situation can lead to a poor system performance. The goal of load balancing is to improve the performance of the system by balancing the loads among the computers. Load balancing is one of prerequisites to utilize the full resources of parallel and distributed systems. Load balancing may be centralized in a single processor or distributed among all the processing elements that participate in the load balancing process [5].

Centralized load balancing algorithms suffer from scalability problems, especially on machines with relatively small amount of memory. Fully distributed load balancing algorithms, on the other hand, tend to yield poor load balance on very large machines. The paper presents a dynamic and clustered load balancing method that overcomes the scalability challenges of centralized schemes and poor solutions of traditional distributed schemes. This is done by creating multiple clusters each with a group of nodes as its part. A cluster is treated like a load balancing domain. Whenever a cluster suffers overload, the extra load is transferred to a supporting node.

II. LOAD BALANCING TAXONOMY

A load balancing algorithm can be characterized as *static* or *dynamic* depending upon its requirement and the nature of the strategy applied [1]. These categories can be sub-classified into various schemes as illustrated in Fig. 1.

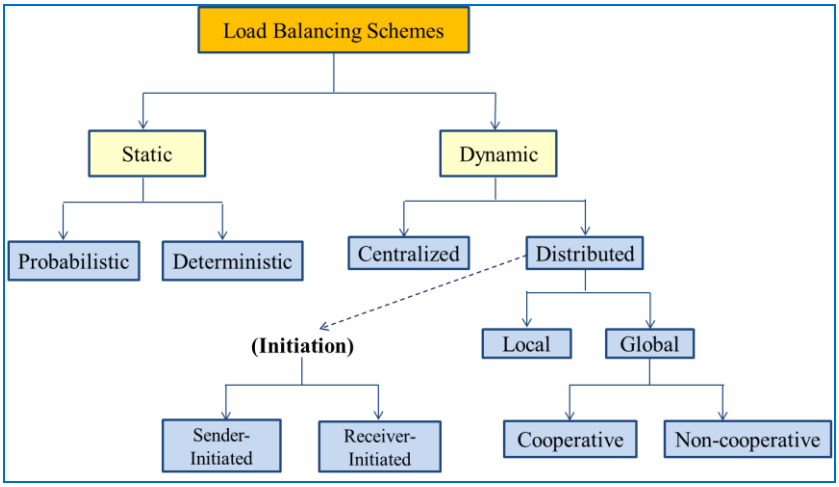

Fig. 1: Load Balancing Taxonomy

A. Static Load Balancing

Static Load Balancing policies base their decision on statistical information about the system. They do not take into consideration the current state of the system. 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.

Static algorithms are of two types: Probabilistic and deterministic. Probabilistic approach uses the attributes of the system such as number of nodes; processing capability of each node; network topology etc. It is easier to implement but has lower performance. On the other hand, deterministic approach uses the properties of nodes and characteristics of processes to allocate processes to nodes. It is difficult to optimize and costs comparatively higher.

B. Dynamic Load Balancing

Dynamic policies base their decision on the current state of the system. They are more complex than static policies. 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 more cost-effective than Dynamic Load Balancing but is not suitable for the systems with changing workloads. Therefore, Dynamic approach is much more efficient for distributed networks.

The Load Balancing strategies can be classified depending upon three parameters of the system; viz. "who makes the load balancing decision?", "what information is used to make the load balancing decision?" and "where the load balancing decision is made?". Based on these three parameters, there are three classifications of load balancing strategies.

1) *Centralized vs. Distributed:* In centralized strategy, all nodes share their load information with one single node which makes the load balancing decision and in distributed strategy, the load balancer is replicated on all workstations and all nodes perform a broadcast of their load information and each node makes the balancing decision at its own.The centralized strategy has one single point of control that can limit the scalability and even lead to the problem of bottleneck. The distributed scheme helps solve the scalability problem and avoids the problem of bottleneck but the expense of "all-to-all" broadcast of profile information between workstations may be high.
2) *Local vs. Global:* 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 such that each group has nearly equal aggregate computational power and balancing decision is made locally by considering the nodes within a single group. Global strategy requires additional communication and synchronization between the various workstations. The benefit of a local strategy is that performance information is exchanged only within the group. But the reduced communication and synchronization between workstations may also be a downfall in cases where the various groups exhibit major differences in performance.
3) *Sender-initiated vs. 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 sender-initiated strategy is well suited for low to moderate system loads; because here, the probability of finding a lightly-loaded node is higher than that of finding a heavily-loaded node. Receiver-initiated strategy works better at high system loads, since it is much easier to find a heavily-loaded node in such systems.
4) *Cooperative vs. non-cooperative:* In cooperative scheme, distributed entities cooperate with each other to make scheduling decisions. These are stable but more complex and involve large scheduling and communication overhead. While in non-cooperative scheme, individual entities act as autonomous entities and make scheduling decisions independently of actions of other entities. These are instable but simple and involve less scheduling overhead

III. ISSUES IN DESIGNING LOAD BALANCING ALGORITHMS

There are several issues to be considered while designing a load balancing algorithm. These are discussed in brief:

A. Load Estimation Policy

Before starting for load balancing in a system, we need to identify the load of the computing nodes in order to recognize heavily loaded and lightly loaded nodes. 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.

CPU utilization is an important aspect of load estimation policy. CPU utilization is the number of cycles executed per unit time. CPU utilization is measured by setting up a timer to observe the CPU state

B. 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. Threshold policies are of two types: Single threshold and Double threshold. A node should only transfer one or more of its processes to another node if such a transfer greatly

improves the performance of the rests of its local processes. A node should accept remote processes only if its load is such that the added workload does not significantly affect the service to the local ones.

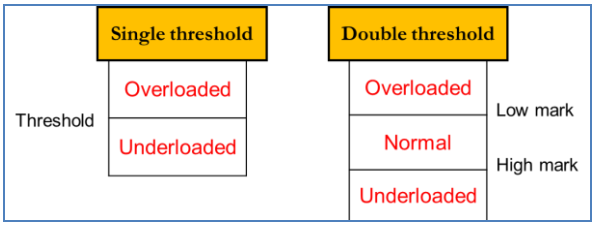

Fig. 2: Threshold policies

C. 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:

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_p is reached.
2) *Shortest Policy:* In this, L_p 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) *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, than the contractor nodes return the bids to manager. The manager then chooses the best bid and transfers the process to the winning contractor node.
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.

D. State Information Policy

State Information policy is used to decide when, what and where the information about the state of other nodes in the network should be collected. Several policies can be adopted to exchange node state information among the nodes in the network.

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.
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) *On-Demand Exchange:* In this a node broadcast a *StateInformationRequest* message when its state switches from normal load region to either over-loaded or under-loaded. 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.
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.

E. 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:

1) *Selfish rule:* In this, local processes are given higher priority than remote processes, but this yields the worst response time.
2) *Altruistic rule:* In this, remote processes are given higher priority than local processes, and it gives the best response time.
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.

F. Selection Policy

Selection policy selects a job to be transferred. A basic criterion that a process selected for transfer should satisfy is that the overhead incurred in the transfer of the process should be compensated by the reduction in the response time. There are several factors to consider in the selection of a process.

- The overhead incurred by the transfer should be minimal. For instance, process of small size carries less overhead.
- The number of location-dependent system calls made by the selected process should be minimal. Location-dependent calls must be executed at the node where the process originated because they use resources such as windows, the clock or the mouse, that only exist at the node.
- The selected task should be long lived so that it is worthwhile to incur the transfer overhead.
- A process should be selected for transfer only if its response time will be improved upon transfer.

G. 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.

1) *Uncontrolled:* In this, remote process is treated same as the local process and it is allowed to migrate any number of times, but this is instable.
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.

IV. RELATED WORK

Load balancing is a challenging problem and has been studied extensively in the past. Load balancing strategies can be divided into two broad categories – those for applications where new tasks are created and scheduled during execution (i.e. task scheduling) and those for iterative applications with persistent load patterns (i.e. periodic load balancing). Much work has been done to study scalable load balancing strategies in the field of task scheduling, where applications can be expressed through the use of task pools. In the proposed strategy, two concepts are mainly referenced.

A. A load balancing approach with centralized Load Balancer and two back-end servers

In [2], 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. Additional servers are used for the sake of reliability. If one server fails, the other can take its job and the system continues to work properly. This design is illustrated in the Fig. 3.

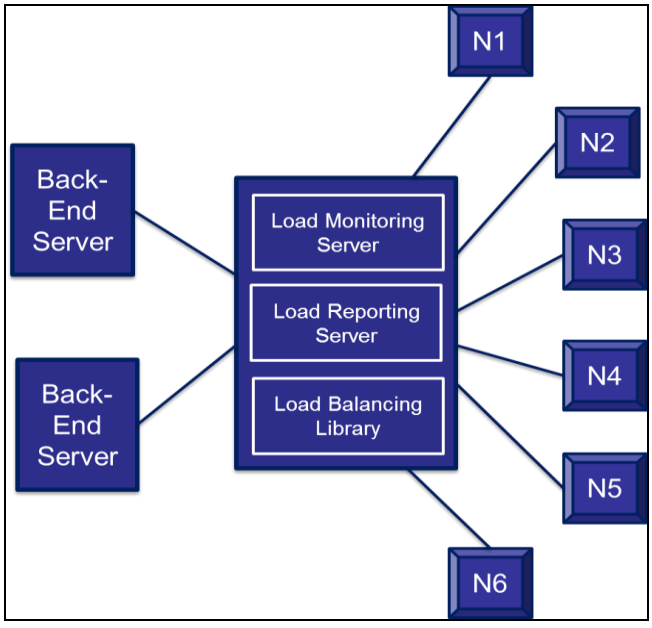

Fig. 3: Proposed design for Load Balancing in [1]

The central load balancer has three parts: Load monitoring server, Load reporting server and load balancing library while the two back-end servers only comprise of Load monitoring server and Load reporting server (as shown in Fig. 4).

The Load Reporting Server is used to collect the machine load information on which it is running. The collected data is sent to Load Monitoring Server which is located on same machine. Load Monitoring Server stores the collected data in data structure. Then an all-to-all broadcast of the load information is carried out. Then, at the central load balancer, when request comes, the Load Balancing Library finds least loaded machine and return the address of that machine.

The drawback of the design is the huge communication overhead involved as it follows a global strategy of load balancing. Secondly the cost of this all-to-all broadcast of load information is high.

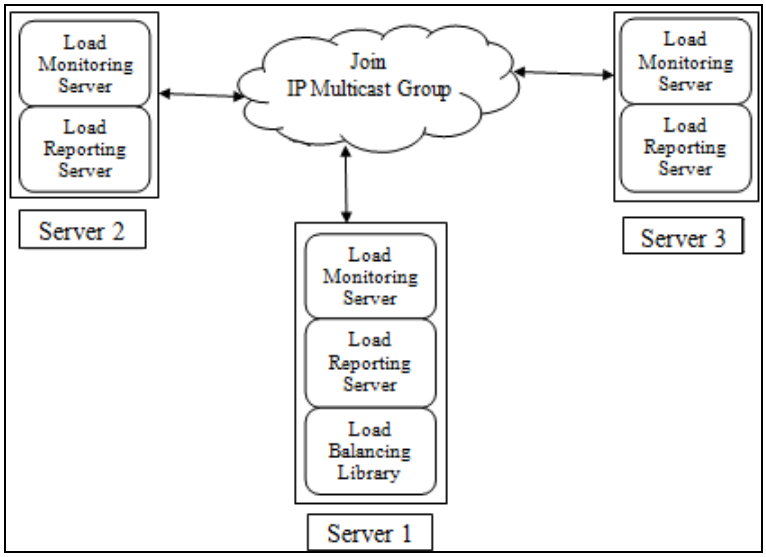

Fig. 4: Applied Load Balancing Strategy in [2]

B. A load balancing approach using one supporting node with each primary node and using a priority scheme to schedule tasks at supporting nodes

Let us now have a look on another approach proposed in [3], it uses one supporting node (denoted as SNi) with each primary node (denoted as Ni) as depicted in Fig. 5. 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.

This approach has a drawback of its complexity and the cost of such a huge infrastructure. The priority scheme makes it more dynamic and suitable for distributed systems as well as handling real time tasks.

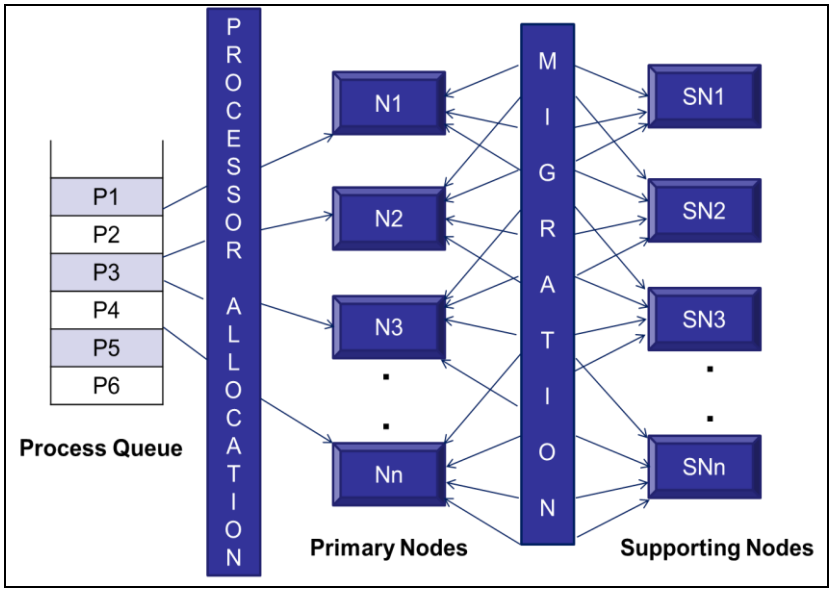

Fig. 5: Proposed design for Load Balancing using multiple supporting nodes [3]

C. Drawbacks in above described designs

The design proposed in [2] as illustrated in Fig. 3 is a centralized approach. Hence, as the number of nodes increase, the load increases at the central load balancer and also results in performance bottleneck. Moreover, in today's era of large and highly distributed networks, centralized approach is not suitable. It only does well for small networks that contain less number of nodes. Furthermore, it does not involve any priority scheme to differentiate more important and less important processes. It suffers from the problem of starvation.

Another design proposed in [3] as illustrated in Fig. 5 is a distributed approach. It uses a supporting node for each node to balance the workload at that node. It uses a complex design and huge infrastructure costs as the number of computing elements is just double the actual number of nodes.

The proposed approach illustrated in the next section overcomes the drawbacks of both these concepts and develops a faster and efficient algorithm that uses a semi-distributed approach. It also uses priority scheme while assigning the processes so as to avoid the problem of starvation and large waiting times.

V. PROPOSED DESIGN

The proposed design is a slight extension and modification to that proposed in [1]. It uses a clustered approach in which each cluster maintains three nodes and each cluster has a supporting node (as shown in Fig. 6). The load balancer maintains a queue for each cluster to store the load of its nodes. This reduces the cost of infrastructure used in [3], and improves the service offered by [2] by using clusters rather than individual nodes. It uses a threshold approach to decide whether a node is heavily loaded or not.

The Load Balancer has three parts: Load Monitoring Server (LMS), Load Reporting Server (LRS) and Decision Making Server (DMS). The Load Monitoring Server and the Load Reporting Server have similar tasks of calculating and collecting the system load information. The decision making server runs only whenever some node is overloaded and finds the most appropriate node to which the overload is to be transferred.

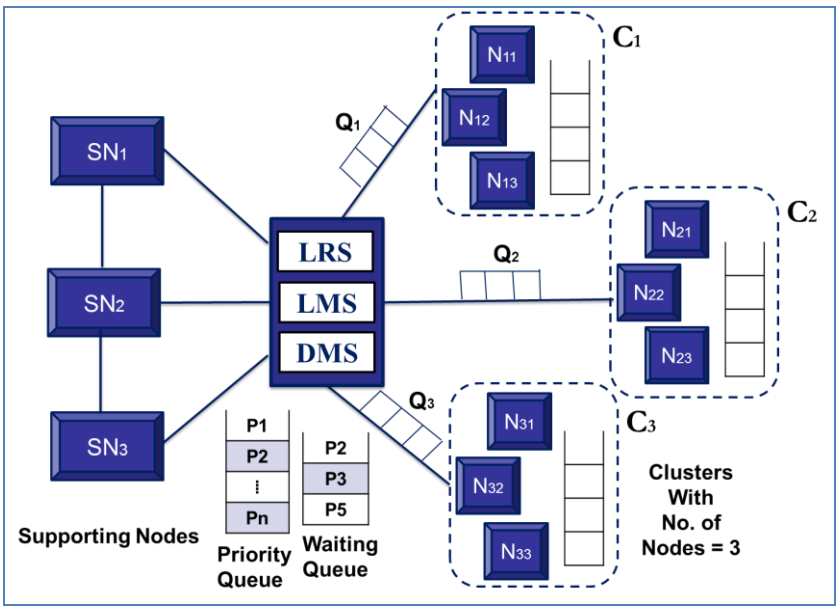

Fig. 6: Proposed design for Load Balancing using a supporting node with each cluster of three nodes

VI. PROPOSED ALGORITHM

A. Algorithm

The following steps are carried out in order to perform dynamic load balancing.

1) First of all, the LRS located at each node N_{ij} collects the load at its machine and stores that load in the Load Queue Q_i at Q_{ij}.
2) Each cluster maintains three parameters:
   - A data structure τ called threshold that stores the maximum allowable load without any degradation of performance.
   - A Load Queue Q_i.
   - A waiting queue WQC_i
3) Each cluster has a manager that sends the value of these three parameters to the central load balancer.

4) The central Load Balancers runs in three parts: Load Monitoring Server (LMS), Load Reporting Server (LRS) and a Decision Making Server (DMS).
   i. The LMS monitors the incoming parameters i.e. Load Queue Q_i and threshold τ. If for any cluster, total load at Q_i, plus total load at WQC_i is greater than τ, then C_i is said to be overloaded.
   ii. The LMS returns the amount of overload at C_i and transfers the incoming process to waiting queue.
5) Now for each node C_i, the LRS checks the load of SN_i.
   i. If SN_i is free, then the overload is transferred to SN_i.
   ii. Otherwise, LRS reports the Load of each SN_i to Decision Making Server along with the amount of overload.
6) The DMS maintains two parameters:
   i. A priority queue in which processes are sorted in the order of the priorities.
   ii. A central waiting queue in which waiting processes are sorted in the order of their priorities
7) DMS compares the priority of the currently running process at SN_i and the priority of the incoming process
   i. If the incoming process has higher priority, then the currently running process is interrupted and inserted to waiting queue and the incoming process is allowed to execute on SN_i.
   ii. Otherwise, the current process continues to run and the incoming process is inserted to the waiting queue.
8) Whenever any SN_i becomes free (i.e. Load(SN_i)=0), then the LRS sends the address of that SN_i to DMS and DMS transfers the highest priority process from the waiting queue to SN_i for execution.

B. Advantages

The above presented algorithm has following advantages

- Produce faster and appropriate results than both the referred designs.
- Reduce average response time of processes.
- Implement priority scheme so as to prevent higher priority processes from large waiting times.
- Balance the workload so as to maximize total system throughput.

VII. CONCLUSION

The presented design and algorithm works well for distributed systems and ensures that no process suffers starvation and no processor is overwhelmed. The presented design works for cluster with number of nodes=3. The proposed design has several advantages over the referred designs comprising reduced communication overhead, semi-distributed architecture, dynamic approach to load balancing and reduced cost and complexity.

For further research, it would be a significant task to design a modified approach to work for 'n' number of clusters with 'n' number of nodes. We can apply a dynamic priority scheme and also add some functionality to make it work for heterogeneous networks. The optimization of the algorithm is another task for future research.

REFERENCES

[1] Shweta Rajani, Renu Bagoria, "A Dynamic Approach for Load Balancing using Clusters", *International Journal of Research in Advent Technology*, Vol.2, No.4, April 2014, E-ISSN: 2321-9637

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

[3] 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

[4] Rutuja Jadhav, Snehal Kamlapur, 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

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

[6] Mayuri A. Mehta, Devesh J. Jinwala, "Analysis of Significant Components for Designing an Effective Dynamic Load Balancing Algorithm in Distributed Systems", *Third International Conference on Intelligent Systems Modeling and Simulation* 2012 IEEE

[7] Wenzheng Li, Hongyan shi, "Dynamic Load Balancing Algorithm Based On FCFS", *Fourth International Conference on Innovative Computing, Information and Control* 2009 IEEE

[8] 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

[9] Yong Meng TEO, Rassul AYANI, "Comparison of load balancing strategies on cluster-based web servers", *Transactions of the Society for Modeling and Simulation*, 2001

[10] Jean Ghanem, "Implementation of Load Balancing Policies in Distributed Systems", *The University of New Mexico Albuquerque, New Mexico*, June 2004

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

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

[13] Abhijit A. Rajguru, S.S. Apte, "A Comparative Performance Analysis Of Load Balancing Algorithms In Distributed System Using Qualitative Parameters", *International Journal of Recent Technology and Engineering (IJRTE)*, ISSN: 2277-3878, Volume-1, Issue-3, August 2012

