iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** IV **Month of publication:** April 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Resource Management Using Dynamic Load Balancing in Distributed Systems

Prof. Darshika Lothe
Assistant Professor, Imperial College of Engineering & Research, Pune, Maharashtra

Abstract — In a distributed network of computing hosts, the performance of the system can depend on dividing up work effectively across the participating nodes. Dynamic load balancing have the ability of performing better than static load balancing. The overheads involved are much more than dynamic approach. Load balancing is found to reduce significantly the mean and standard deviation of job response times, especially under heavy and unbalanced workload. The performance is strongly dependent upon the load index; queue-length-based indices perform better. The reduction of the mean response time increases with the number of hosts. Load balancing is still very effective when a large portion of the workload is not movable. All hosts, even those with light loads, benefit from load balancing. Similarly, all types of jobs see improvements in their response times, with larger jobs benefiting more and system is more stable. The random arrival of tasks at each processor is likely to bring about uneven processor loads in a distributed system. Various approaches are there to provide the dynamic load balancing. In one scheme, Load balancing can be improved by having one centralized node to handle the uneven load. The project work shows that efficiency can be improved by replacing the centralized node with a number of nodes added. The scheme can reduce the waiting time by significant amount of time.
Keywords— Load Balancing, Migration, Priority, Hosts.

I. INTRODUCTION

A distributed system consists of a collection of autonomous (heterogeneous) computers, connected through a network and distribution middleware, which enables computers to coordinate their activities and to share the number of resources of the system, so that users access the system as a single, integrated computing facility. Some of the basic
Characteristics of the distributed system are Resource Sharing, Openness, Concurrency, Scalability, Fault Tolerance, different types of Transparency. There are many advantages in distributed system as better performance, distribution, reliability, Incremental growth, sharing of resources, communication
Resource Management in Distributed Systems used approach such as Task assignment, Load-balancing and Load-sharing Types of process scheduling techniques Task assignment approach in which User processes are collections of related tasks and Tasks are scheduled to improve performance Load-balancing approach in which Tasks are distributed among nodes so as to equalize the workload of nodes of the system Load-sharing approach In which Simply attempts to avoid idle nodes while processes wait for being processed
Load Balancing based on the idea of migration of excess load from heavily loaded node to lightly loaded ones. The problem starts with to determine when to migrate a process or task. Simple procedure is to determine the load on every processor and decide on which processor to migrate the task .Load estimation is important to balance the load and for load estimation consider the processing power of the load. Processing power includes the overall configuration of node.
In static load balancing, the performance of the processors is determined at the beginning of execution. Then depending upon their performance the work load is assigned by the master processor. The slave processors calculate their allocated work and submit their result to the master. A task is always executed on the processor to which it is assigned that is static load balancing methods are non-preemptive. The goal of static load balancing method is to reduce the execution time, minimizing the communication delays Static algorithms ignore the current state or load of the nodes in the system [1].
Deterministic and Probabilistic is type of static Load balancing *Deterministic* Load balancing *algorithms* use the information about the properties of the nodes and the characteristic of processes to be scheduled Probabilistic Load balancing algorithms use information of static attributes of the system (e.g. number of nodes, processing capability, topology) to formulate simple process placement rules Deterministic approach is difficult to optimize Probabilistic approach has poor performance
In applications with constant workloads, static load balancing can be used to the computation. Other applications have workloads

that are unpredictable or change during the computation; such applications require dynamic load balancers that adjust the decomposition as the computation proceeds. Static algorithms collect no information and make probabilistic balancing decisions, while dynamic algorithms collect varying amounts of state information to make their decisions. Potentially the more information an algorithm can collect the better decision it will make. The problem with the complex balancing algorithms is that they cannot keep up with the rapidly changing state information of the system. The best they can do is to make their decisions based on information passed between two nodes i.e. the node itself and the node that sent it the information concerning its queue length. Load balancing is done basically to do following benefits.

Load balancing reduces mean job response time under job transfer overhead.

Load balancing increases the performance of each host.

Small jobs will not suffer from starvation.

Two broad categories of load balancing algorithms are commonly recognized. In source-initiative algorithms, the hosts where jobs arrive and take the initiative to transfer the jobs, whereas, in server-initiative algorithms, hosts able and willing to receive transferred jobs go out to find such jobs. In order to maximize the performance Dynamic load balancing can be used with a number of approach.

A. Load Balancing in Distributed Systems

Load balancing is the technique to distribute the workload in distributed environment within multiple computers and many other resources within system.in order to optimize resource utilization ,maximize throughput minimize response time and avoid overload. Several issues in load balancing such as 1) If client is access an overloaded web server 2)Processes in overloaded database server take a long time to search through the database 3) load is extremely high and server become crash

1) *Task assignment approach:* In this approach Processes have been split into tasks and Computation requirement of tasks and speed of processors, Cost of processing tasks on nodes, Communication cost between every pair of tasks, Resource requirements and available resources on node are known. Reassignment of tasks is not possible in this approach. Task assignment approach achieve goals are Quick turnaround time of process, High degree of parallelism, Efficient utilization of resources.
2) *Load-balancing approach:* For Load-balancing algorithms To balance the workload on all the nodes of the system, it is necessary to decide how to measure the workload of a particular node Some measurable parameters (with time and node dependent factor) can be the following:
   a) Total number of processes on the node
   b) Resource demands of these processes
   c) Instruction mixes of these processes
   d) Architecture and speed of the node's processor

Several load-balancing algorithms use the **total number of processes** to achieve big efficiency

B. Load Balancing Taxonomy

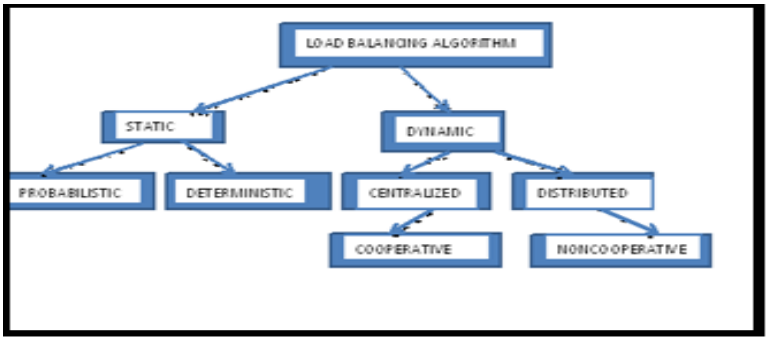


Fig 1: Taxonomy Load-balancing

1) *Static and Dynamic:* Static algorithms use only information about the average behavior of the system Static algorithms ignore the current state or load of the nodes in the system Dynamic algorithms collect state information and react to system state if it changed Static algorithms are much more simpler Dynamic algorithms are able to give significantly better performance
2) *Centralized and Distributed:* Centralized approach collects information to server node and makes assignment decision Distributed approach contains entities to make decisions on a predefined set of nodes Centralized algorithms can make efficient decisions, have lower fault-tolerance Distributed algorithms avoid the bottleneck of collecting state information and react faster
3) *Cooperative and Noncooperative:* In Noncooperative algorithms entities act as autonomous ones and make scheduling decisions independently from other entities In Cooperative algorithms distributed entities cooperate with each other Cooperative algorithms are more complex and involve larger overhead Stability of Cooperative algorithms are better

C. Approach For Dynamic Load Balancing

Dynamic algorithms allocate processes dynamically when one of the processors becomes under loaded. Instead, they are buffered in the queue on the main host and allocated dynamically upon requests from remote hosts Dynamic algorithms are able to give significantly better performance [2].

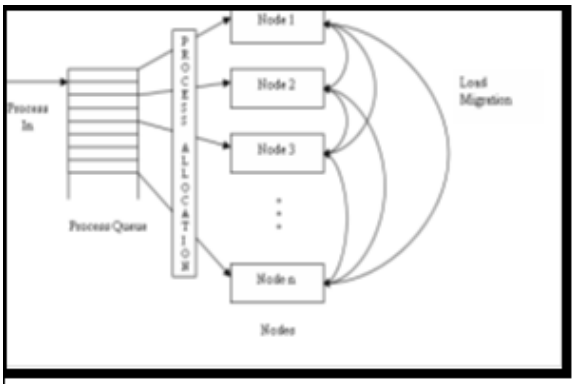


Fig 2 . Simple dynamic Load balancing to avoid overload on heavily loaded node by transferring process to light weighted node.[5]

As shown in diagram, initially processes are stored in queue. If these are placed in queue, processes are allotted one by one to primary nodes. Processes are migrated from heavily loaded node to light weighted node. Process migration is greatly affected by the network bandwidth and work load. In order to reduce the traffic, nodes are grouped into clusters. First a light weighted node is checked in the same cluster, if suitable node not found then after nearby cluster is searched and after getting a required node transfer takes place if a protocol is satisfied for load transfer .

D. Policies or Strategies in dynamic load balancing

There are different policies in dynamic load balancing [10]:

1) *Transfer Policy:* The part of the dynamic load balancing algorithm which selects a job for transferring from a local node to a remote node is referred to as Transfer policy or Transfer strategy.
2) *Selection Policy:* It specifies the processors involved in the load exchange (processor matching).
3) *Location Policy*: The part of the load balancing algorithm which selects a destination node for a transferred task is referred to as location policy or Location strategy.
4) *Information Policy*: The part of the dynamic load balancing algorithm responsible for collecting information about the nodes in the system is referred to as Information policy or Information strategy.
5) *Load estimation policy*: which determines how to estimate the workload of a particular node of the system?
6) *Process transfer policy:* which determines whether to execute a process locally or remotely?
7) *Priority assignment policy:* which determines the priority of execution of local and remote processes at a particular node?
8) *Migration limiting policy*: which determines the total number of times a process, can migrate from one node to another.

Dynamic load balancing is useful in a system consisting of a network of workstations (NOW) in which the primary performance

goal is maximizing utilization of the processing power instead of minimizing execution time of the applications. The major disadvantage of dynamic load balancing schemes is

a) The run-time overhead due to the load information transfer among processors,

b) The decision-making process for the selection of processes and processors for job transfers

c) The communication delays due to task relocation itself

E. Centralized Approach For Load Balancing

Many times whenever a heavily loaded node don't find node in its cluster and due to congestion in network, node fail to search the node far away cluster. It would be better if heavily loaded node finds a temporary node in same cluster to handle the over load. So, in centralized approach one centralized node is provided in each cluster. Whenever a primary node is over loaded, first it search the other light weighted primary nodes, if such primary node is available, load transfer take place between these two node and load is balanced, otherwise if such light weight node is not available, one centralized node is available to accommodate the overload of a primary node.

This centralized node is not assigned any process initially; it is given only the overload of primary nodes. Centralized node has some better structure as compared to other nodes in the cluster. Traffic between centralized node and primary nodes kept minimum to avoid network delay.

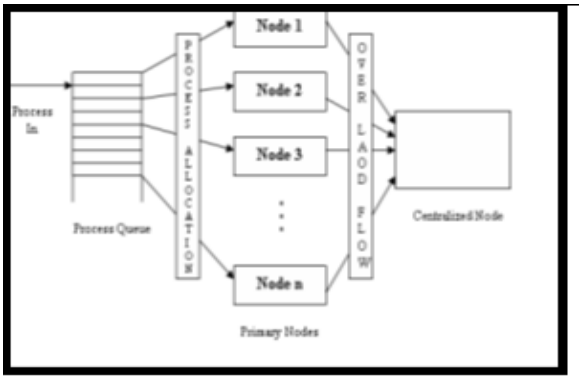


Fig3. Centralized node based Load balancing [5]

F. Modified Approach For Dynamic Load Balancing

In Centralized approach there is single node, so process the load at high so still a limitation is there. An approach is there to remove the limitation is to split the centralized node into small nodes called supporting nodes (SNs). But still here supporting node are not allotted load initially. Many times supporting nodes is idle or they are not properly loaded as only overload is assigned to supporting nodes. This is wastage of power of supporting nodes. We can also use the free time of SN by making them busy for this free time. So a further approach is developed here in which supporting nodes are given some load initially and SNs maintain a priority list of process or order in which the process at the SN will execute.[5]

Suppose a process Pi is currently executed by SNi and a Primary node Ni is overloaded so that it finds a supporting node SNi suitable for transferring its overload, so Ni will interrupt the SNi, then SNi will assign Priority to the coming process and call the interrupt service routine to handle the interrupt. Interrupt Service Routine actually compares the priority of each coming process with the currently executing process and perform the switching between the currently executing process and process coming from the primary nodes, Otherwise, each supporting node is maintaining a priority queue in which process to be executed are sorted according to the priority, in which coming process are stored in this queue with a priority. In figure 3. Each node whether primary node or secondary node (assuming initially process is there) is assigned some Task .Process and transferred from PN to SN if overload

Occur at any PN.[5]

Volume 3 Issue IV, April 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

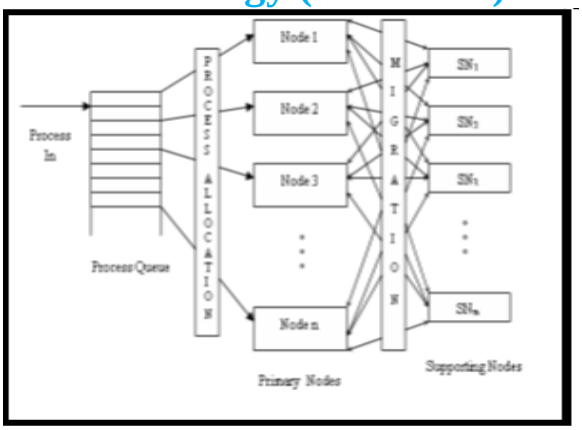


Fig4. Modified Approach with Load balancing handling multiple request.[5]

A load balancing approach using one supporting node with each primary node and using a priority scheme to schedule tasks at supporting nodesLet 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. 3. In case of overload atnode 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.

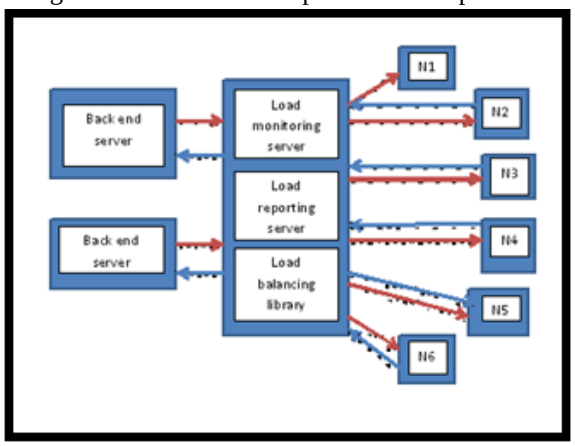


Fig.4: Proposed design for Load Balancing using[8]

1) *Disadvantage of centralized and Load Balancer approach:* As the number of nodes increase, the load increases at the central node in both the approaches.

The proposed approach overcomes the drawbacks 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.

G. Proposed Design

The proposed design is uses a clustered approach in which each cluster maintains three nodes and each cluster has a supporting node (as shown in Fig. 4). The load balancer maintains a queue for each cluster to store the load of its nodes. This reduces the cost of infrastructure used in the design proposed in [6], and improves the service offered by [7] 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.[8]

Volume 3 Issue IV, April 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

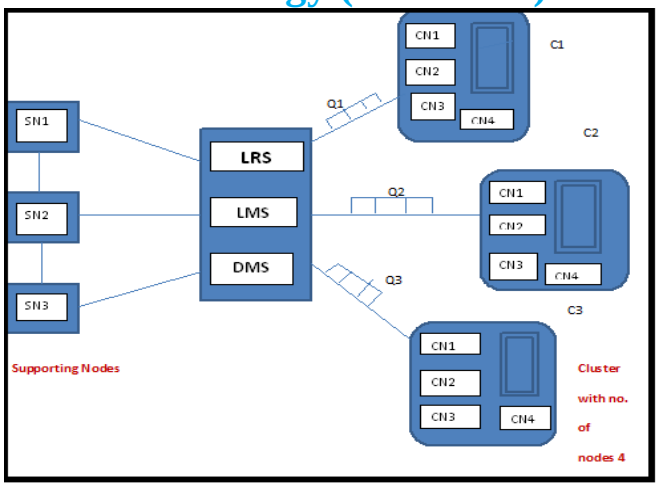


Fig5: Proposed design for Load Balancing using a supporting node with each cluster of three nodes [8]

Algorithm for distributed load balancing

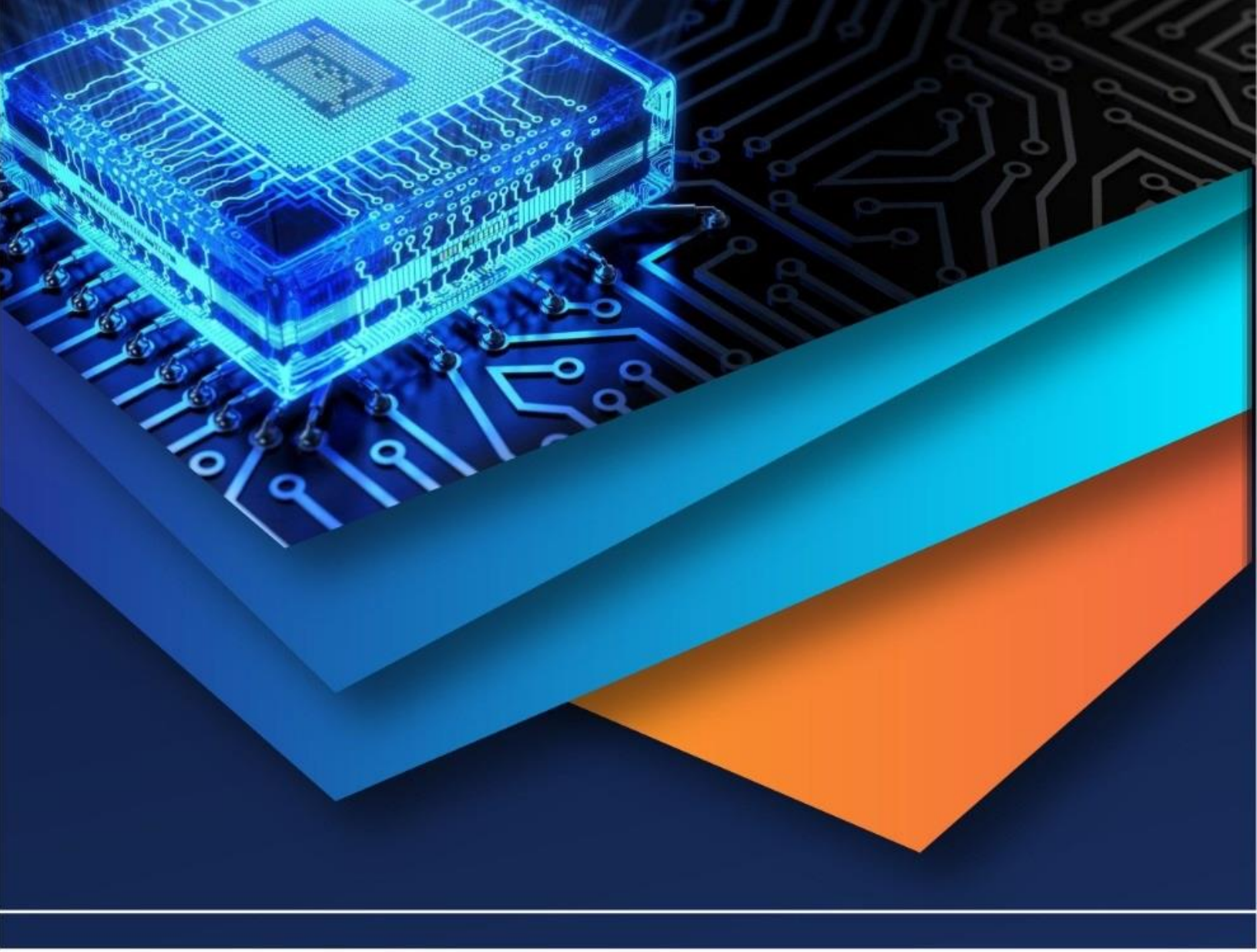
CENTRAL_LOAD_BALANCER( )
Step 1: Create clusters C[i:1 to n]
Step 2: Create four Cluster nodes in each cluster CN[j:1 to 4]
Step 3: Read threshold for each cluster in T
Step 4: The initial load at each node is its process turn-around time
Step 5: Call LOAD_REPORTING_SERVER()
Step 6: sum of loads that is total load on each cluster
Step 7: Create empty **waiting queue** and a **priority queue** with initial processes sorted according to their priorities.
Step 8: Create Supporting node(SN) for each cluster SN[i:1 to n]
Step 9: Call LOAD_MONITORING_SERVER()
Step 10: Monitor load at SN,
If (SN[i].load==0) then:
Call DECISION_SERVER(Process P)
Step 11: Exit.
LOAD_REPORTING_SERVER( )
Step 1: Collect load at each nodeC N[j]
Step 2: Store load of NodeC N[j] in Load Queue Q[j]
Step 3: Return Q.
LOAD_MONITORING_SERVER(Process P)
Step 1: Set Overload= Overload + P.time
Step 2: Transfer Process P to waiting queue.
Step 3: Return Overload[8]
DECISION_MAKING_SERVER(WQ, P)

Step 1: Repeat step [2] for i=1 to n
Step 2: If (SN[i].load==0) then:
Set SN[i].load :=P.time
Set SN[i].process := P
Set SN.pri := P2.pri
Step 3: Set Process P1 := Pop (WQ)
Step 4: If (P1.pri > SN[i].pri) then:
(i) Interrupt (SN[i].process)

(ii) Push (SN[i].process) to WQ
(iii) Set SN[i].proc := P1
(iv) Set SN[i].load := P1.time
(v) Set SN.pri := P2.pri
Step 5: Else Push P1 to W.
Step 4: Return[8]

REFERENCES

[1]Abhijit A. Rajguru, S.S. Apt" 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.
[2] Yongzhi Wang, Tom Z. J. Fu and Dah-Ming Chiu," Analysis of Load Balancing Algorithms in P2P Streaming",
[3]Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma," Performance Analysis of Load Balancing Algorithms", World Academy of Science, Engineering and Technology 38 2008",
[4] William Leinberger, George Karypis, Vipin Kumar, "Load Balancing Across Near Homogeneous Multi-Resource Servers", 0-7695-0556- 2/00, 2000 IEEE.
[5]"An Algorithm for Dynamic Load Balancing in Distributed Systems with Multiple Supporting Nodes by Exploiting the Interrupt Service" Parveen Jain1, Daya Gupta2 1,2Delhi College of Engineering, New Delhi, India 1 International Journal of Recent Trends in Engineering, Vol 1, No. 1, May 2009
[6] Parveen Jain, Daya Gupta, "An Algorithm for DynamicLoad 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]
[7] 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
[8] SSRG International Journal of Mobile Computing & Application (SSRG-IJMCA) – volume 2 Issue 1 Jan to Feb 2015 ISSN: 2393 - 9141 www.internationaljournalssrg.org Page 1 "A Clustered Approach for Load Balancing in Distributed Systems" Shweta Rajani1, Niharika Garg.
[9] Harchol-Balter M, Crovella M, Murta C. "On choosing a task assignment policy for a distributed server system." Journal of Parallel and Distributed Computing 1999;59:204–28.
[10]Jayasinghe M, Tari Z, Zeephong sekult P.' A scalable multi-tier task assignment policy with minimum excess load." In: Proceedings of the IEEE symposium on computers and communications, Riccione, Italy, 22–25 June. IEEE; 2010. p. 913–8
[11] Distributed O.S Concepts and Design‖, P.K.Sinha, PHI
[12] Advanced concepts in Operating Systems‖, Mukesh Singhal & N.G.Shivaratri, TMH



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY