ISSN 2250-3153

Improved Hybrid Dynamic Load Balancing Algorithm for Distributed Environment

UrjashreePatil*, RajashreeShedge**

*Department of Information Technology, ThadomalShahani Engineering College
**Department of Computer Engineering, R. A. I. T.

Abstract- With the rapid development of high-speed and Computational resources Load Balancing has become a necessity in emerging distributed environment to address the inherit heterogeneity in computing resources. From two load balancing strategies static and dynamic, dynamic strategy is efficient for these systems. In dynamic also the centralized dynamic approach limits the scalability with the load balancing unit itself becoming a bottleneck. Conversely, the decentralized dynamic approach though overcomes the above problems, suffers from increased communication overhead. The hybrid dynamic approach which uses centralized and decentralized strategy suffers from bottleneck and communication overhead problem for large number of system. In this paper we propose the design of a simple yet effective improved hybrid dynamic load balancing algorithm that overcomes the limitations of hybrid dynamic algorithm and performs competitively for heterogeneous system having large number of heterogeneous nodes.

Index Terms-Bottleneck, Communication Overhead, Distributed Environment, Heterogeneous System, Load Balancing.

I. INTRODUCTION

Load balancing is an operation that involves redistribution of the system workload, as evenly as possible, among the processing elements of a distributed system based on prior analysis of the existing load. Load balancing is to ensure that every processor in the system does approximately the same amount of work at any point of time. Load balancing is one of prerequisites to utilize the full resources of parallel and distributed system [1]. Load balancing is basically to do following benefits as optimal resource utilization, maximum throughput, reduce mean job response time under job transfer overhead, increase the performance of each host, remove starvation that causes to small job[2]. In parallel and distributed system more than one processors processing parallel programs, the amount of processing time needed to execute all processes assigned to a processor is called **workload** of processor [3]. Processes may migrate from one node to another even in the middle of execution to ensure equal workload. A distributed system provides the resource sharing as one of its major advantages, which provide the better performance and reliability than any other traditional system in the same condition [4].

A load balancing has two strategies **Static Load Balancing (SLB)** and **Dynamic Load Balancing (DLB)** depending on the freshness of the input parameters used for determining the load distribution. An associated aim is to avoid system overload by using the processing power of the entire system evenly to smooth out periods of high congestion at individual nodes and network. Dynamic load balancing is more responsive to changes in the system workload and hence is more suitable to heterogeneous environment [5][6]. A DLB algorithm is further based on either centralized OR decentralized (distributed) approach. **Centralized DLB algorithm** approach limits the scalability [6] because the load balancing unit itself becomes a bottleneck and susceptible to failures so **Decentralized DLB algorithm** is preferred [7][8].

The critical issue here is the inter-node communication and synchronization required – that involves an all-to-all broadcast as compared to that in the centralized approach (all-to-one and one-to-all)[9].Thus, though the decentralized approach overcomes the limitations of the centralized approach, it raises the number of messages used for information exchanges. Therefore, the development of an efficient DLB algorithm is really critical design issue – one that must ensure a tradeoff between the scalability and efficiency. In this backdrop, the hybrid dynamic algorithm is designed which uses both centralized and decentralized strategy but which is also less effective in case of large number of nodes in the system [10]. Therefore we propose the design of a simple yet effective improved hybrid dynamic load balancing algorithm. Our approach typically sits between the two extremes mentioned above.

The report proceeds with what other related work we have done before in sectionII. SectionIII The new improved hybrid load balancing approach which is overcoming the drawbacks of hybrid algorithm work that we have done. After that in section IV Analysis of the improved algorithm is done with hybrid system showing that how proposed method is efficient, finally conclusion is produced in chapter V and references that we have taken for whole work in section 6.

II. RELATED WORK

There are many sources available to write literature for the parametric analysis of static and dynamic load balancing algorithm and the design of load balancing algorithm that try to address one or another design issues, but no one of them addressed efficiency concern as per my observation.Load balancing involves the distribution of jobs throughout a networked computer system, thus increasing throughput without having to obtain additional or faster computer hardware [1]. Load balancing policies may be either static or dynamic.

A.Static Load Balancing(SLB)

Static load balancing policies are generally based on the information about the average behavior of system; transfer decisions are independent of the actual current system state. SLB algorithm collects no information and makes probabilistic

balancing decisions[2]. Processes are assigned to the processors before the execution starts.

BDynamic Load Balancing(DLB)

Dynamic load balancing algorithm collect varying amount of state information to make their decisions. DLB can reassign the processes. Dynamic policies, on the other hand, react to the actual current system state in making transfer decisions. This makes dynamic policies necessarily more complex than static ones, and truly optimal dynamic policies are known only for special systems.Load balancing algorithms vary in their complexity where complexity is measured by the amount of communication used to approximate the least loaded node and cost of transferring a job from one node to another. Potentially the more information an algorithm can collect the better decision it will make [3]. Figure 1 shows the general dynamic load balancing system [5].

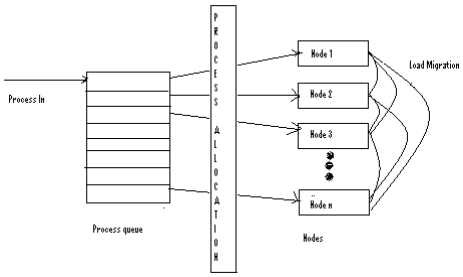

Figure 1: Simple dynamic load balancing

The problem with the complex load balancing algorithm is that they cannot keep up with the rapidly changing state information of the system. An associated aim is also to avoid system overload by using the processing power of entire system evenly to smooth out periods of high congestion at individual nodes and network [4].

In a distributed network of computing hosts, the performance of the system can depend crucially on dividing up work effectively across the participating nodes. Dynamic load balancing is still very effective when a large portion of the workload is immobile. 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. System instability is possible, but can be easily avoided. The random arrival of tasks at each processor is likely to bring about uneven processor loads in a distributed system [5]. Dynamic Load balancing can be improved by having one centralized node to handle the uneven load.DLB can be centralized or decentralized.

1) Centralized DLB Algorithm

A central node acts as the load balancing node, working on the jobs from different nodes in the system. However, this approach limits the scalability because the load balancing unit itself becomes a bottleneck and susceptible to failures.

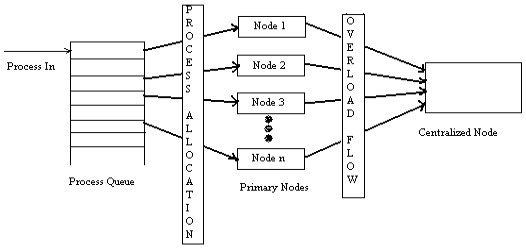

Figure2:Centralized node based load balancing

Figure2 shows the centralized node based balancing in which one central node handles all the load distribution of the overloaded node [6].

2) Decentralized DLB Algorithm

In decentralized algorithm all nodes participate in load balancing. Decentralized dynamic approach though overcomes the above problems, suffers from increased communication overhead.The project work shows that efficiency can be improved by replacing the centralized node with a number of nodes added with interrupt service which is shown in figure 3 [6]. The scheme can reduce the waiting time by significant amount of time [7]. This approach supports scalability and has better fault tolerance [8].The decentralized approach is preferred for the heterogeneous system for balance allocation [9][10].

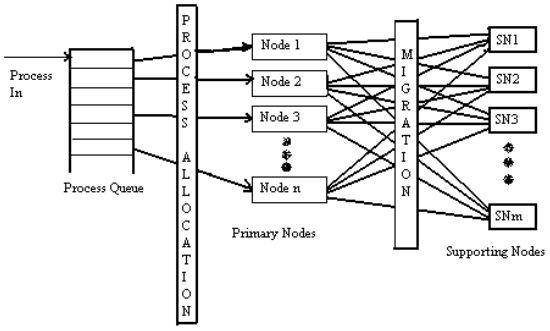

Figure3: Centralized node replaced by number of supporting nodes

3) Hybrid DLB algorithm:-

To overcome the problem of centralized and decentralized DLB the hybrid load balancing algorithm is developed which gives greater performance over decentralized approach increasing response time[11].

In figure 4, the MasterNode MSN are replaced by multiple supporting nodes *S1, S2, S3, ..., Sp*. Each node in the system is allowed to access any supporting node. When a node becomes overloaded, it first queries the member nodes of its group and collects the load information in a decentralized fashion. There is a high probability that the Overloaded node will find a lightly loaded node in the same group Here, when a highly loaded node does not find a lightly loaded node in the same group, it randomly selects one supporting node and now supporting node follows the same procedure as pursued by MSN. The algorithm gives very high performance with large reduction of response time for large system having large number of processes.

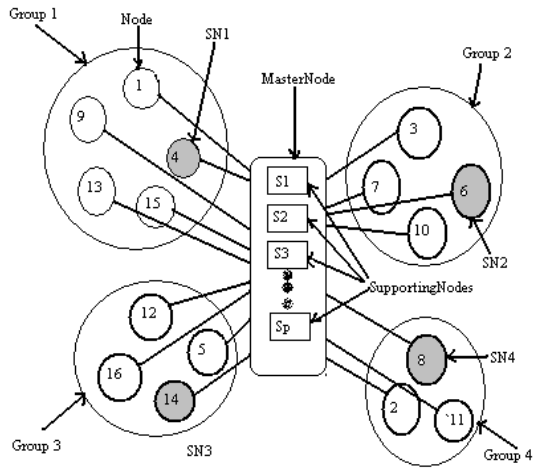

Figure 4: View of groups in simplified distributed system with supporting nodes

The disadvantage of the system is that communication overhead will be very large in case of large number of nodes since the nodes are directly connected to the supporting node of master node in this algorithm [11]. So Here, we propose the design of a simple yet effective improved hybrid dynamic load balancing algorithm that overcomes limitations of hybrid approach performs competitively for heterogeneous system. Under this new hybrid approach, a highly loaded node is expected to discover lightly loaded node in lesser time as compared to hybrid dynamic approach, minimizes the communication overhead, and is also expected to produce improved performance under variety of workloads.

III. PROPOSED APPROACH

A.Design issues

There are various factors pertaining to design of a load balancing algorithm. For the proposed hybrid approach, all these factors are discussed in the following

1) Heterogeneity

The proposed algorithm is designed to consider system heterogeneity that is to be associated with varied hardware and software resources of the nodes. We address heterogeneity with respect to diverse set of CPUs, memories, and disks machine platform. Specifically, we characterize each node *ni*by its CPU speed *Ci*, the storage capacity *Mi*and the disk speed *Di*.

2) Target applications

A real distributed system can have workload consisting of applications that may be using significant amount of CPU, memory, and/or I/O. Therefore, to prevent performance fall under variety of applications, it is essential to make a decision upon intended applications while designing a DLB algorithm.

3) Metrics for load measurement

Our design takes into account the following parameters to measure the load at each node in the system; based on which the decision for further distribution is done.

- maximum CPU queue length
- maximumCPU utilization
- maximummemory queue length
- maximummemory utilization
- maximumI/O queue length
- maximumI/O utilization
- maximumcontext switch rate

An appropriate combination of these parameters is used to constitute the load index. Considering maximum load measurement matrices over short interval improves instantaneous values, and overcome the use of too long averaging interval that reduces the responsiveness of the index to load changes[11]. Hence, we use maximum load of resources over an interval of 4 seconds.

B. Components of proposed algorithm

Various kinds of components exist, and they should be chosen according to the desired environment, such as application types or system environment. The improved hybrid dynamic algorithm uses same component as that of hybrid dynamic algorithm with little changes named as follows,

1) Information Policy

The proposed ***information policy***uses an effective mechanism to exchange state information that considerably reduces the communication overhead while quickly updating the state information in large system environment. We use traditional *demand-driven* information policy same as hybrid approach. When any node becomes highly loaded, initially, load information of local nodes from any one supporting node that resides within the same group is collected. If a lightly loaded node is not found within the same group, other groups are searched to find a lightly loaded node. In former case, lesser number of messages will be used while in latter case, the number of messages will increase gradually.

2) Transfer policy

We use a threshold policy as the ***transfer policy***that is dynamic in nature. However, for simplicity single threshold policy is considered. Defining a suitable threshold value for a particular computing environment is a challenging task. Therefore, one node in each group is selected as a *MasterNode*. The MasterNode periodically collects the load information of other nodes in the group to compute the average load of the group. This average load is set as a new threshold value *T* and is broadcast to other nodes in the group.

3) Selection policy

Selection policy is same which is used in hybrid dynamic load balancing algorithm[13].

4) Location policy

Location policyconsidered is polling (or probing). Under this policy, a MasterNode polls all the nodes in the same group to find out a lightly loaded node. The node whose load is lowest taking into account all the parameters is considered as the destination node. Thus, this approach makes an effort to choose the best node for the process transfer. The destination node executes the process regardless of its resource usage at the time of arrival of the transferred process.

C. Working of Proposed Approach

The proposed algorithm is designed for a system in which $N = \{ n1, n2, n3, ..., nn \}$ nodes are connected via a high speed

communication network. Each node in the system is composed of a combination of various resources including processor, memory, disk, network connectivity and every node differs in its capacity of processor, memory and disk. A system is divided into m groups, where 1 < m <n/2. For simplicity, static groups are created. Moreover, each group can have different size where a number of nodes in each group can be minimum 2 and maximum n/2. In each group one node is selected as MasterNode and which is also divided into p number of *MasterSupportingNode*(MSN) where 1<p<m/2, which is the central node for that group that periodically collects the load of the other nodes in the group to set new threshold value. Communication between groups is possible via number of supporting node SN.

Table 1:Definition of notations

Notations	Definition
N	Number of nodes in distributed system
Ni where 1<i<n	Different nodes in system
M	Number of groups
MN	Master node
Msn	MasterSupportingNode
SN	Supporting node
Li	Load of the node ni
Pi	Overload of the node ni
T	Threshold value to decide status of the node

Figure 5 shows groups created for simplified distributed system. Here, system consists of n=13 nodes that are divided into m=4 groups, where group1 and group 3 has 4 nodes, group2 has 4 nodes, and group4 has 2 nodes. Nodes 3, 7, 8, and 12 are MasterNode. Each MN is divided into Msn. Each Msn node has access to SN. Msn is responsible for monitoring available resources of other nodes in the group.

When a node becomes overloaded, it first queries any one of the Msn of its group and Msn collects the load information in a centralized fashion. There is a high probability that the Msn will find a lightly loaded node in the same group for large system (unlikely if the majority of nodes in group are highly loaded).

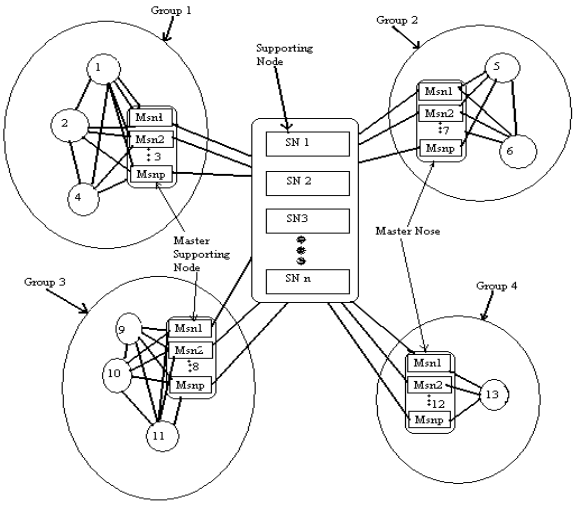

Figure 5:View of groups in simplified distributed system with supporting nodes

However, when the Msn does not find a suitable destination node in the same group to transfer its overload, it requests to one of the supporting nodes to search for the lightly loaded node in the other groups. The SN randomly selects one group and collects the load status of all nodes in this group from Msn to ensure the availability of destination node. If any lightly loaded node is found, the overload will be transferred to this node, otherwise Msn selects another supporting node and process will be repeated until either a lightly loaded node is found or all groups are searched. In the latter case, the overload will be executed on the original node.Since the groups contains very large number of nodes so there is no possibility that Msn becomes a bottleneck and a single point of failure occur due to presence of number of Msn. Each Msn in the group is allowed to access any supporting node. We will see algorithm utilized by highly loaded, Master supporting and supporting node.

Algorithm utilized by a highly loaded node

Assumption: Each node is having some load
Input: load Li of the node and threshold T
Output: destination node to transfer overload
Algorithm LoadBalancingProcedure (Li, T)
1. if Li > T for any node Ni
2. Consult Msn (or randomly selected MasterSupportingNode) to find lightly loaded node within the group
3. if (destination node found)
4. dispatch Pi = Li – T load to this node
5. exit

Algorithm utilized by MasterSupportingNode

Assumption: Each node is having some load
Input: load Li of the node and threshold T
Output: destination node to transfer overload

algorithm LoadBalancingProcedure Msn(Li, T)
1. if Li > T for any node Ni
2. collect the load status of all nodes in the same group
3. take out lightly loaded nodes
4. destination node = most lightly loaded node
5. if (destination node found) send ID of the destination node to overloaded node
6. transfer overload to destination node

7. exit
8. else
9. Consult SN (or randomly selected supporting node) to find lightly loaded node in other group

Algorithm utilized by SupportingNode

Assumption: Each node is having some load
Input: load Li of the node and threshold T
Output: destination node to transfer overload

Algorithm LoadBalancingProcedureSN (Li, T)
1. while (destination node is not found OR all groups are not searched)
2. FOR each group in the system
3. obtain value of threshold T for the selected group from Msn
4. take out lightly loaded nodes
6. destination node = most lightly loaded node
7. if (destination node found)
8. send ID of destination node to Msn
9. Transfer overload to destination node
10. exit
11.else
12. Execute overload on the original node

IV. PERFORMANCE ANALYSIS

The proposed improved hybrid algorithm is evaluated using following two parameters: Communication overhead and Response time for large system having node in of large number.

Table 2: Comparison of decentralized approach and proposed approach

	Communication Overhead	
	Hybrid dynamic Approach	Proposed Improved Hybrid dynamic approach
Best Case	7	5
Average Case	18 or 26	16 or 22
Worst Case	30	30

Table 2 summarizes the number of messages utilized by hybrid and improved hybrid approaches in order to achieve load balancing for the given example.
Consider figure 5, let's assume that node 4 in group-1 has become overloaded. As per the proposed approach, initial number of messages used to collect the load information from the same group is equal to 7 (4 request messages and 4 response messages). If lightly loaded node is found in this group, then 1 message will be used to transfer overload to destination node. Thus, the number of messages is limited to 9 if lightly loaded node is found in the same group. However, if a destination node is not found in the same group, the highly loaded node sends a request to Msn to locate lightly loaded node from the other group. Msn will consult with SN1 and SN1 will search group2, group3, and group4 until lightly loaded node is found with 18, 26, and 30 messages respectively.
This analysis shows that proposed approach has higher probability of lesser communication overhead. For the above mentioned example, it takes only 9 messages to transfer overload in best case while the same communication in decentralized approach would have taken 31 messages. In average case, it takes 18 or 26 messages. However, in worst case, it may need 30 messages to transfer the overload, but the likelihood of worst case is less. Moreover, it has been reported in literature that lesser communication overhead minimizes response time. Thus, proposed approach is expected to improve performance of system in terms of communication overhead and response time. Thus, from all the above study we will come to conclusion which is explained in next chapter.

V. CONCLUSION

With increasing number of nodes in the system due to large number of resources used by user the previous load balancing algorithms failed to perform efficiently to provide greater performance with efficient work distribution. We have designed an effective improved hybrid dynamic load balancing algorithm that fulfills main objectives to overcome several limitations of hybrid load balancing method, provides Load measurement policy that gives load status of major affecting parameters using Effective information policy. This improved hybrid load balancing approach significantly minimizes the communication overhead along with demand-driven information policy for system containing large number of nodes.Our algorithm is expected to perform efficiently for CPU-intensive, memory-intensive, I/O intensive applications, and any combination of these because we have defined the load index utilizing load information of critical resources viz. CPU, I/O, memory to measure the load in large system.

REFERENCES

[1] Md. Abdur Razzaque and Choong Seon Hong."Dynamic Load Balancing in Distributed System: An Efficient Approach"; T.N.Anitha et al / Indian Journal of Computer Science and Engineering (IJCSE) Vol. 3 No. 2 Apr-May 2007.

[2] Pawandeep Kaur, Harshpreet Singh**, "**Adaptive dynamic load balancing in grid computing an approach"; International Journal of Engineering Science & Advanced Technology(IJESAT),112-125,May-Jun 2012.

[3] Ali M. Alakeel. "A Guide to Dynamic Load Balancing in Distributed Computer Systems"; International Journal of Computer Science and Network Security, Vol. No.10, Issue No. 6, 153—160, Jun 2010.

[4] Belabbas Yagoubi, and Yahya Slimani. "Dynamic Load Balancing Strategy for Grid Computing"; Transactions on Engineering, Computing and Technology (World Academy of Science), Vol. No. 13, 260—265, May 2006.

[5] Sandip Kumar Goyal, R.B. Patel, Manpreet Sing, "Adaptive and Dynamic Load Balancing Methodologies For Distributed Environment: A Review"; International Journal of Engineering Science and Technology , Vol.3 No. 3,2011.

[6] Parveen Jain, and 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 (Academy Publisher), Vol. No. 1, Issue No. 1, 232—236, 2009.

[7] Sunita Bansal, Divya Gupta, and Chittaranjan Hota, "Adaptive Decentralized Load Sharing Algorithms with Multiple Job Transfers In Distributed Computing Environments"; International Journal of Recent Trends in Engineering, Vol 2, No. 2, November 2009.

[8] Issam Al-Azzoni, and Douglas G. Down. "Decentralized Load Balancing for Heterogeneous Grids"; Proceedings of the Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, IEEE Computer Society, 545—550, 2009.

[9] P.Neelakantan, "Decentralized load balancing in Heterogeneous systems using diffusion Approach"; International Journal of Distributed and Parallel Systems (IJDPS) Vol.3, No.1, january 2012.

[10] Sunita Bansal, Divya Gupt1, and Chittaranjan Hota, "Adaptive Decentralized Load Sharing Algorithms with Multiple Job Transfers In Distributed Computing Environments"; International Journal of Recent Trends in Engineering, Vol 2, No. 2, November 2009.

[11] Mayuri Mehtaand Devesh Jinwala, "A Hybrid Dynamic Load Balancing Algorithm for Heterogeneous Environments" ; International Journal of Computer Science & Emerging Technologies(IJCSET), Vol-2 No 5 October, 2011

[12] http://publib.boulder.ibm.com/infocenter/cicsuc/v6r0m0/index.jsp?topic=%2Fcom.ibm.cicsuc600.doc%2Fcclaa0270.htm

[13] Pradeep Kumar Sinha, "Didtributed operating system"; IEEE Press, 1997 - 743 pages,2011.

AUTHORS

First Author – UrjashreePatil., M.E.(Computer Pursuing), Department of Information Technology, ThadomalShahani Engineering College.Email: patil4urjashree@gmail.com

Second Author –RajashreeShedge,M.E.(Computer) Department of Computer Engineering, R. A. I. T.
Email: rajashree.shedge@gmail.com