

An Algorithm for Dynamic Load Balancing in Distributed Systems with Multiple Supporting Nodes by Exploiting the Interrupt Service

Parveen Jain¹, Daya Gupta²

^{1,2}Delhi College of Engineering, New Delhi, India

^{1,2}E-mail: parveenjain1@gmail.com, dgupta@dce.ac.in

Abstract — 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 have the potential of performing better than static strategies, they are inevitably more complex. The overheads involved are much more. But one can not negate their benefits. Load balancing is found to reduce significantly the mean and standard deviation of job response times, especially under heavy and/or 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, but levels off beyond a few tens of hosts. 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. 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 with interrupt service. The scheme can reduce the waiting time by significant amount of time.

Index Terms— Load Balancing, Complexity, Migration, Priority, Hosts.

1. INTRODUCTION

Rapid growth in use of computer has increased the number of resource sharing application and ultimately increased the amount of load across internet. Problem can be solved by increasing the size of servers or distribute the applications in effective manner across different servers. Distribution is termed as 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 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 neighboring 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. In applications with constant workloads, static load balancing can be used as a pre-processor to the computation. Other applications, such as adaptive finite element methods, have workloads that are unpredictable or change during the computation; such applications require dynamic load balancers that adjust the decomposition as the computation proceeds. Load balancing algorithms vary in their complexity where complexity is measured by the amount of communication used to approximate the least loaded node. Static algorithms collect no information and make probabilistic balancing decisions, while dynamic algorithms collect varying amounts of state information to make their decisions. The most significant parameter of the system was found to be the cost of transferring a job from one node to another. It is this cost that limits the dynamic algorithms, but at the high end of complexity are the dynamic algorithms which do collect varying amounts of information. 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 increase 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. Dynamic load balancing is complex but the benefits form dynamic approach is much more than its complexity.

2. PRIMARY APPROACH FOR DYNAMIC LOAD BALANCING

A distributed system consists of independent workstations connected usually by a local area network. Static load balancing doesn't fulfill the requirements for load balancing. As in static load balancing, number of jobs at a station is fixed. Dynamic load balancing does the process while jobs are in execution. Jobs are allocated to host or node. Load at each post is calculated (as number of process, structure of node, network bandwidth etc.) dynamically. As sender initiated or receiver initiated approaches are available to find the desired load for transferring the load.

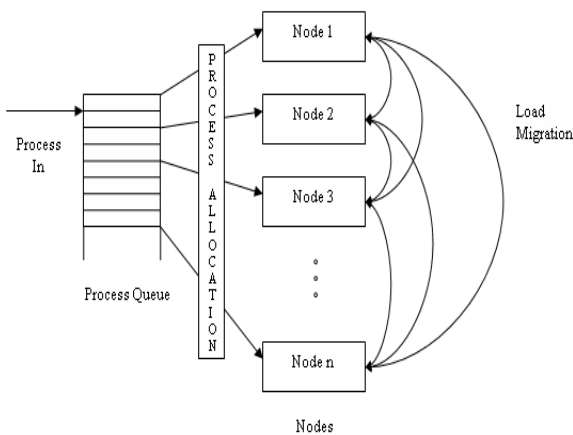


Figure 1. Simple dynamic Load balancing to avoid overload on heavily loaded node by transferring process to light weighted node.

As shown in diagram, initially processes are stored in queue or process can be allotted as they arrive. 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.

3. 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.

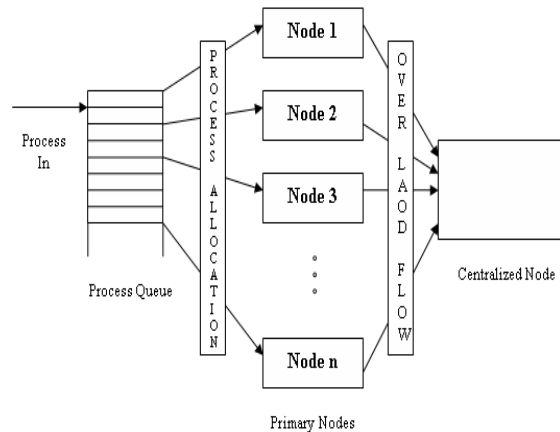


Figure 2. Centralized node based Load balancing

As in figure 2. The overload from nodes is transferred to centralized node to increase output of each node.

4. MODIFIED APPROACH FOR DYNAMIC LOAD BALANCING

In Centralized approach there is single node, so process the load at high speed by using switching but 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. Suppose a process P_i is currently executed by SN_i and a Primary node N_i is overloaded so that it finds a supporting node SN_i suitable for transferring its overload, so N_i will interrupt the SN_i , then SN_i 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

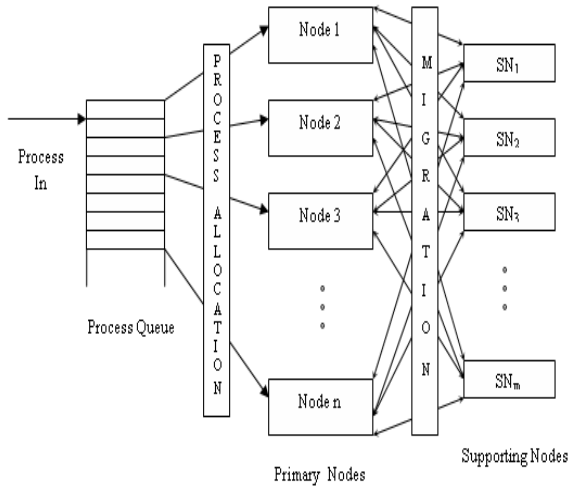


Figure3. Modified Approach with interrupt service for Load balancing Task .Process and transferred from PN to SN if overload occur at any PN.

5. ALGORITHM FOR MODIFIED APPROACH

Two types of node are there- Primary and supporting nodes. As name indicate what each will do. Primary are main nodes and supporting are used to handle overload. When a Primary Node is overloaded, it will search other primary node (light weighted) within a cluster if such node is found; overloaded primary node will maintain the load balancing with available primary node. If suitable primary node is not found.

Algorithm:

- N_i : List of primary nodes
- SN_j : List of supporting nodes
- PI: Priority index maintained by supporting node
- P_k : List of processes in Process Queue
- $i, j, k \in N, j < i$ // $N \leftarrow$ Natural Number

Initially: 1. Assuming each node is having some load,
 2. Some supporting nodes may have load
 $N_i \leftarrow$ Load, $S_j \leftarrow$ Load // load defines some process is there

Procedure: Main ()

```

{
I. Suppose a node  $N_t$  is heavily loaded with load  $\xi$ 
   where  $0 < t < i, \xi \in P_k$ 
// Two situations are possible as in step II.
II. If ( Search_node ) // Search_node will find
// out whether a light weighted primary node is
// available, if primary node is available it will give
// index of available node.
{
    Available_Node  $\leftarrow \xi$  // Overload is assigned
    //to Available node given by Seach_node ()
    Load ( $N_t$ ) = Load ( $N_t$ ) -  $\xi$ 

```

```

}
Else
{
    Call Search_S_node ( ) // Search_node will
    //find out a light weighted or minimum
    //weighted supporting node with index as
    //Available_S_node.
    Available_s_node  $\leftarrow \xi$ 
    Load ( $N_t$ ) = Load ( $N_t$ ) -  $\xi$ 
    IN_S (Available_s_node,  $\xi$ )
}
}}
```

Procedure: IN_S (SN_node, ξ)

```

{
I. Priority is assigned to SN_node as PI
   ( $SN\_node$ ) = t
II. If  $t > PI$  (RP) // RP is the currently
//process on Selected supporting node.
{
    // P_List is list of pending process shorted
    //according to priority.
    P_List  $\leftarrow$  RP // RP is added to Pending
//List
    Now RP  $\leftarrow \xi$  //  $\xi$  is now allotted to
//the Selected Supporting node.
}
Else
    P_List  $\leftarrow \xi$  //  $\xi$  is added to Pending
//List
}
}
```

Procedure: Search_node ()

```

{
I. for each  $N_i$  except node initiating the search_
   node procedure
{
    Check the node with minimum load
    //Minimum load includes the number of
    //processes as well as structure or configuration
    //of node and node should be able to accept
    //node.
}
II. If Desired Node available
{
    return (index of available node) // index
    //defines the property to identify the Node
}
}
}
```

Procedure: Search_S_node ()

```

{
I. for each  $SN_j$  except node initiating the search
   node procedure
{
    Check the Supporting node with minimum
    load

```

```

// Minimum load includes the number of
//processes as well as structure or
//configuration of node
}
II. return (index of available Supporting node)
// index defines the property to identify the
// Node
}

```

Primary node will try to approach supporting node and will find suitable supporting node, after finding suitable node it will interrupt SN for execution of its process. Sn will execute the ISR (Interrupt Service Routine) for handling the interrupt. In ISR, Each supporting node maintain a priority queue, each process is maintained there according to their priority. Process from PN is also assigned some priority. If the process from PN is having greater priority than the priority of current process running on SN. Then Process from PN is current process and process running on SN is stored in priority queue with suitable priority. And if the process from PN is having less priority than the priority of current process running on S. Then Process from PN stored in priority queue.

Conclusion

A Modified Model has been formulated for dynamics of a distributed computing system in the context of load balancing. Initially the centralized model was used for solving the purpose of load balancing. The Model considers two approaches (i) the centralized node used for balancing can be partitioned into various small size nodes. The partitioned node can be heterogeneous in nature. Nodes are given extra load from primary nodes or can say nodes are used to maintain to load balancing. (ii) many times partitioned nodes remain idle. Idle time of nodes can be used by allocating them some load. Switching take place whenever a process with high priority comes.

As main aim in distributed system is to execute the process at minimum cost i.e. time is most important factor can be considered in cost calculation. This is attributable to the fact that New dynamic load balancing policy achieves a higher success, in comparison to the previously used load balancing techniques, in reducing the likelihood of nodes being idle while there are tasks in the system, comprising tasks in the queues. Our future work considers the implementation and evaluation of the complexity of the modified approach for load balancing. Ant colony optimization is used to minimize the complexity for the purpose.

6. REFERENCE

- [1] Bahi J.M., Vivier C., and Couturier R., "DynamicLoadBalancing and Efficient Load Estimators for Asynchronous Iterative Algorithms," IEEE Trans. Parallel and Distributed Systems, vol. 16, no. 4, Apr. 2005.
- [2] Bridgewater S.A. Jesse, Boykin Oscar P., and Roychowdhury P. Vwani, "Balanced Overlay Networks (BON): An Overlay Technology for Decentralized Load Balancing", IEEE Transactions on Parallel and Distributed Systems, Vol. 18, pp. 1122-1133, No. 8, August 2007.
- [3] Bahi J.M., Vivier C., and Couturier R., "DynamicLoadBalancing and Efficient Load Estimators for Asynchronous Iterative Algorithms," IEEE Trans. Parallel and Distributed Systems, vol. 16, no. 4, Apr. 2005.
- [3] Cortes A., Ripoll A., Senar M., and Luque E., "Performance Comparison of Dynamic Load-Balancing Strategies for Distributed Computing," Proc. 32nd Hawaii Conf. System Sciences, vol. 8, p. 8041, 1999.
- [4] Cybenko G., Dynamic load balancing for distributed memory multiprocessors, Journal of Parallel Distributed Computing, Vol 7, pp. 279-301, 2001.
- [5] Dhakal S. , "On the optimization of load balancing in distributed networks in the presence of delay, Advances in Communication Control Networks," LNCSE vol. 308, pp. 223–244, Springer-Verlag, 2004.
- [6] Dhakal S., Hayat M. M.,Elyas M., Ghanem J. and Abdallah C. T., "Load Balancing in Distributed Computing Over Wireless LAN: Effects of Network Delay" , IEEE Communications Society / WCNC 2005, Vol. 2, pp. 1755-1760, 2005.
- [7] Dhakal Sagar , Hayat Majeed M., Pezoa Jorge E., Yang Cundong, and Bader David A., "Dynamic Load Balancing in Distributed Systems in the Presence of Delays: A Regeneration-Theory Approach" IEEE Transactions on Parallel and Distributed Systems, vol. 18,no. 4, pp. 485-497, April 2007.
- [8] Gelenbe E. and Kushwaha R., "Incremental Dynamic Load Balancing in Distributed Systems", submitted for publication, 1993.
- [9] Ghanem J. et al., "Load balancing in distributed systems with large time delays: Theory and experiment,"Proceedings of the IEEE/CSS 12th Mediterranean Conference on Control and Automation (MED '04), Aydin, Turkey, June 2004.
- [10] Hayat M.M., Dhakal S., Abdallah C.T., Birdwell J. D., and Chiasson J, "Dynamic Time Delay Models for Load Balancing. Part II: Stochastic Analysis of the Effect of Delay Uncertainty," Advances in Time Delay Systems, vol. 38, pp. 355-368, Springer-Verlag, 2004.
- [11] K. Kabalan, W. Smari, and J. Hakimian, "Adaptive Load Sharing in Heterogeneous Systems: Policies, Modifications, and Simulation," Int'l J. Simulation Systems Science

- and Technology, vol. 3, nos. 1-2, pp. 89-100, June 2002.
- [12] Kafil M. and Ahmed I., "Optimal Task Assignment in Heterogeneous Distributed Computing Systems", IEEE Concurrency, May 1998.
- [13] Kremin O. And Kramer J. , "Methodical Analysis of Adaptive Load Sharing Algorithms," IEEE Trans. on Parallel and Distributed Systems, vol. 3, pp. 747-760, November 2005.
- [14] Mitzenmacher M., "The Power of Two Choices in Randomized Load Balancing", IEEE Transaction on Parallel and Distributed Systems, Vol. 12, pp. 1094-1104, No. 10, Oct. 2001
- [15] Philp R. Ian, "Dynamic Load Balancing in Distributed Systems", IEEE Trans. Parallel and distributed system.
- [16] Tantawi, A.N. and Towsely, D. "Optimal Load Balancing in Distributed Computer Systems", Research Report, IBM Research Division.
- [17] Zhou Songnian," A Trace-Driven Simulation Study of Dynamic Load Balancing", IEEE transactions on Software Engineering, vol. 14, pp. 1327-1341, No 9, Sept 2006.