A New Distributed Load Balancing Algorithm

Md Firoj Ali
Department of Computer Science
Aligarh Muslim University
Aligarh, India
firojali.mca@gmail.com

Rafiqul Zaman Khan
Department of Computer Science
Aligarh Muslim University
Aligarh, India
rzk32@yahoo.co.in

***Abstract*—** In this paper, we presented a load balancing algorithm in distributed computing system. We assumed that each node will maintain a local load table to hold the load status of immediate neighbors. The aim of this algorithm is to achieve balanced load among the processors according to their speed of computation and also to reduce communication over heads. This algorithm also targets most powerful nodes for load transfer in the system. We measured the performance of this algorithm which shows better performance over previously existing Ni's drafting algorithm.

Keywords*-distributed computing, load table, average response time, overheads*

*****

I. INTRODUCTION

A distributed computer system is a collection of processors connected through a network that works together for a common purpose. The primary objective of a distributed system is to proper utilization of the available resources in distributed environment. The most crucial resource is CPU speed and bandwidth of the underlying network. The low bandwidth may bottleneck the CPU speed. So, most commonly used mechanism is to share the load among the nodes by transferring some of the loads from a heavily loaded processor to a lightly loaded processor. The tasks may be transferred either due to the processing time for a task in a processor is expected to be sufficiently greater than that of another remote processor or when the imbalance in the workload at various processors is sufficiently large. The load balancing improves the performance of the system by using the processing power of the entire system more effectively.

The distribution of loads to the processing elements is simply called the load balancing problem. In a system with multiple nodes there is a very high chance that some nodes will be idle while the other will be over loaded. The goal of the load balancing algorithms is to maintain the load to each processing element such that all the processing elements become neither overloaded nor idle that means each processing element ideally has equal load at any moment of time during execution to obtain the maximum performance (minimum execution time) of the system [2,3,4, 7, 9,10,11]. So the proper design of a load balancing algorithm may significantly improve the performance of the system.

In the network there will be some fast computing nodes and slow computing nodes. If we do not account the processing speed and communication speed (bandwidth), the performance of the overall system will be restricted by the slowest running node in the network [11]. Thus load balancing strategies balance the loads across the nodes by preventing the nodes to be idle and the other nodes to be overwhelmed. Furthermore, load balancing strategies removes the idleness of any node at run time.

Load balancing is the way of distributing load units (jobs or tasks) across a set of processors which are connected to a network which may be distributed across the globe. The excess load or remaining unexecuted load from a processor is migrated to other processors which have load below the threshold load [8]. Threshold load is such an amount of load to a processor that any load may come further to that processor. In a system with multiple nodes there is a very high chance that some nodes will be idle while the other will be over loaded. So the processors in a system can be identified according to their present load as heavily loaded processors (enough jobs are waiting for execution), lightly loaded processors(less jobs are waiting) and idle processors (have no job to execute). By load balancing strategy it is possible to make every processor equally busy and to finish the works approximately at the same time.

There are two fundamental approaches to the load balancing algorithm design. In static load balancing design approach the tasks are assigned on the basis of a priori knowledge of the system and once the tasks are allocated on the nodes do not change [1, 2]. The performance of the static load balancing algorithms depends on the prior information about the tasks and the system. The decision to transfer the tasks does not depend on the system state change. So this approach is best suited for homogeneous distributed computing system. But the dynamic load balancing algorithms take the decision to transfer the tasks depending on the current state of the system. The tasks are transferred from heavily loaded node to the lightly loaded node [1,2,5]. So the quality of dynamic load balancing algorithms depends on the collection of information on load on different nodes in the system. So this approach is best suited for heterogeneous distributed computing system.

In dynamic load balancing the information may be collected either by centralized or distributed approach. In centralized approach the information is collected by a specially designed central node and in distributed approach each node has the autonomy to collect the information about the load of the system. It has been reported that the collection of information by centralized approach about the system state does not cause any performance degradation for a reasonably large distributed computing systems [6].The drawback of this

ISSN: 2321-8169
2556 – 2559

approach is that the performance of a globally distributed system would be very poor and the cost of state information collection would be too much and maintaining a huge information by a single node will surely cause a performance degradation. In the distributed information collection policy the information is collected either by sender initiative or receiver initiative algorithm. In sender initiative approach the heavily loaded nodes search for lightly loaded nodes for transferring extra load and the receiver initiative approach is the converse of sender initiated approach [12, 13]. In this approach either a sender or a receiver may poll all the nodes in a network for load balancing causing huge overheads. To reduce the overheads the sender or receiver nodes poll a selected number of nodes like nearest neighbors [4,7,13]. Another performance problem with this approach is associated with the inter-arrival times and service times.

II. Computational Model and Assumptions

The distributed system is represented by an undirected graph G = (P, E) where *P* refers to the set of processors and *E* to the set of links. The communication link between any two processors is assumed to be bidirectional. Thus if there is a link (P_i, P_j) that joins P_i with P_j, then P_i is a neighbor of P_j and they can send and receive information and load from each other. We also assumed that there are *N* heterogeneous processors P_n where $n=2, 3, ...N$.

Each processor maintains a local load table that holds the three field of information: processor ID, status and load of neighboring nodes. Status of a node is either -1,0 or +1. 0 implies for normal loaded, +1 for overloaded and -1 for lightly loaded situation. A node will be lightly loaded if it's ready queue becomes less than half the ready queue length. From load table a node will select a least loaded node. If still load becomes excess, then select next least loaded node and so on.

A processor P_n of the system maintains two queues for its tasks: a 'ready' queue and a 'waiting' queue as shown in Fig. 1. A task or a job may come in as input to a processor directly from outside or as a transfer from a neighbor. Two types of tasks may thus be executed in a processor: local tasks and remote tasks. A local task comes in as input to a particular processor directly from outside the system. A remote task, on the other hand, is received at a particular processor as a transfer from one of its neighbors and has come into the system as input at some other processor. The 'ready' queue has a buffer of finite size of length six and all tasks in this queue are executed by the respective processor. The 'waiting' queue has a buffer of fairly large (infinite) length and holds the tasks that arrive into the system externally. A task waiting in the 'local' queue is either transferred to the 'ready' queue of the processor if the 'ready' queue is not full or transferred to another processor. A processor whose 'ready' queue is half filled is taken to be lightly loaded. A processor is assumed to be heavily loaded if its 'ready' queue is full and its 'local' queue is not empty. A task from a heavily loaded processor when transferred to another neighboring processor enters into the 'ready' queue of that processor since it is assumed to be lightly loaded. A processor which has its 'ready' queue full and the 'local' queue empty is called normally loaded. The performance of the system is measured in terms of its overall response time consisting of the service time, the queuing time and the transmission time.

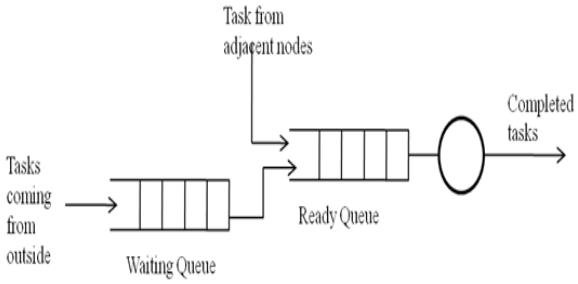

Figure 1. Simulation Model

III. Simulation Study

We considered a mesh topology of sixteen heterogeneous nodes for the simulation purpose. Each queue follows the M/M/1 queue model. Once a task is assigned in ready queue cannot be migrated. Only the tasks from the waiting queue are allowed to be migrated. We also considered a node as under loaded if it has load below the half of the length of a ready queue; a node is considered to be moderate loaded if its waiting queue is empty but ready is not empty and a node is over loaded if its waiting queue is not empty. Whenever a node either becomes under loaded or over loaded, it informs its status to its neighbors and the neighboring nodes would immediately update their load tables. A over loaded node will compare its amount of load with the loads from its load table with -1 status. Once the over loaded node finds the under loaded node having highest load deficiency, it will set an agreement with that under loaded node that it wants to transfer the load. If the under loaded node agrees then load would be transferred. The over loaded node may transfer its load to more than one nodes if its load is more than the load deficiency of a single node. Figure 2 represents a situation of node P6 and its neighbor P2, P5, P7 and P10. Each node represents two numbers: first number shows the status and second number shows the load. Table 1 represents the load table at P6 for that moment. Suppose that P6 has total tasks 12, and then it has extra 6 tasks which are to be transferred. 4 out of 6 is transferred to P5 and rest 2 tasks is transferred to P2 node.

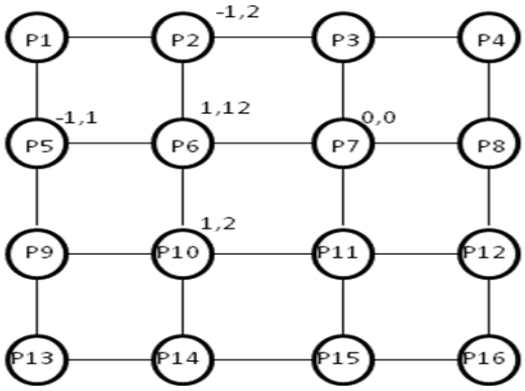

Figure2. Heterogeneous mesh topology of node 16

ISSN: 2321-8169
2556 – 2559

Node	Status	Load
P2	-1	2
P5	-1	1
P7	0	0
P10	1	2

Figure 3. Load Table

We measured the response time with the arrival rate and we compared our algorithm with the algorithm without load balancing and Ni's drafting algorithm [14,15]. Figure 3 shows the comparison below.

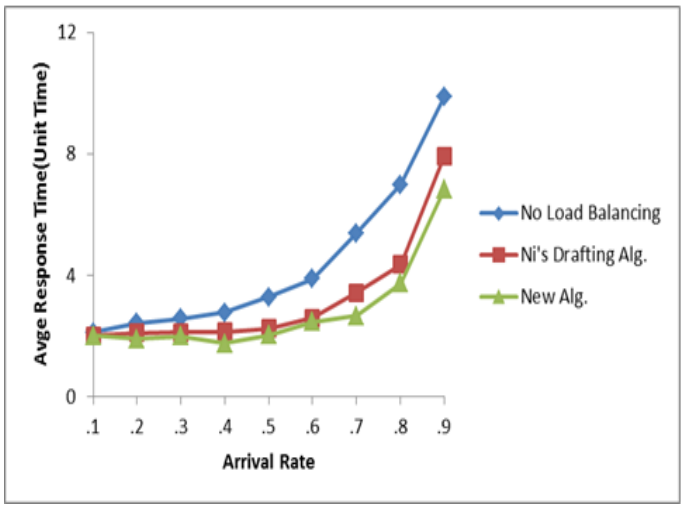

Figure 4. Average Response Time VS Arrival Rate

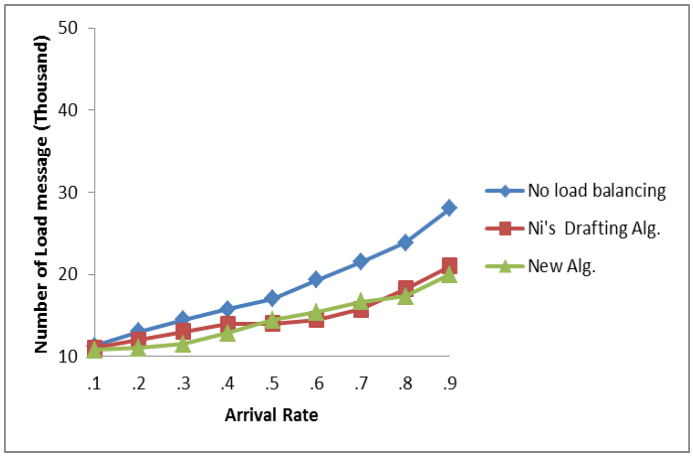

Figure 5. Number of Load Message VS Arrival Rate

IV. CONCLUSION

In this paper we presented a distributed load balancing algorithm. In this algorithm each node maintains a load table holding the current load situation of its neighbors. A new concept in threshold load in each node avoids unexpected over loading and also ensures more job allocation towards more powerful nodes. As the nodes inform their load situation on status change, the over loaded or under loaded nodes do not need to poll for transfer of jobs or to invite the jobs from the neighbors and thus causes low congestion in the network. We measured response time and overheads by applying our concept in mesh topology of sixteen nodes and compared our algorithm with Ni's drafting algorithm. Our algorithm produces better response time and over heads with respect to the arrival rate than Ni's drafting algorithm.

REFERENCES

[1] Ahmad I., Ghafoor A. and Mehrotra K. "Performance Prediction of Distributed Load Balancing on Multicomputer Systems". ACM, 830-839, 1991.

[2] Ali M.F. and Khan R.Z. "The Study on Load Balancing Strategies in Distributed Computing System", International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.2, April 2012.

[3] Antonis K., Garofalakis J., Mourtos I., and Spirakis P., A hierarchical adaptive distributed algorithm for load balancing, Journal of Parallel and Distributed Computing, 64 (1) (2004) 151-162.

[4] Arora M., Das S.K., Biswas R., "A de-centralized scheduling and load balancing algorithm for heterogeneous Grid environments", Proceedings of the International Conference on Parallel Processing Workshops, 18–21 August 2002, pp. 499–505.

[5] Barak and La'adan O., "The MOSIX multicomputer operating system for high performance cluster computing", Future Generation Computer Systems, 13 (4-5) (1998) 361-372.

[6] Berenbrink P., Friedetzky T. and Steger A. "Randomized and Adversarial Load Balancing". *CiteSeer*^{x}, 1997.

[7] Bernard G., Steve D. and Simatic M. "A Survey of Load Sharing in Networks of Workstations". The British Computerm Society, The Institute of Electrical Engineers and IOP Publishing Ltd, 75-86,1993.

[8] Berten V., Goossens J., and Jeannot E., "On the distribution of sequential jobs in random brokering for heterogeneous computational Grids", IEEE Transactions on Parallel and Distributed Systems, 17 (2) (2006) 113-124.

[9] Dandamudi S. P., Lo K. C. M., "A hierarchical load sharing policy for distributed systems", Proceedings of the Fifth International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '97), 12-15 Janauary 1997, pp:3 – 10.

[10] Eager D.L., Lazowska E.D., Zahorjan J., "The limited performance benefits of migrating active processes for load sharing", ACM SIGMETRICS Performance Evaluation Review, 16 (1) (1988) 63–72 .

[11] Gu D. Z., Yang L. and Welch L. R., A Predictive, Decentralized Load Balancing Approach, in: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Denver, Colorado, 04-08 April 2005.

[12] Khan R.Z. and Ali M.F. "An Efficient Local Hierarchical Load Balancing Algorithm (ELHLBA) in Distributed Computing", IJCSET, Vol 3, Issue 11, 427-430, November 2013.

ISSN: 2321-8169
2556 – 2559

[13] Khan R.Z. and Ali M.F. "An Efficient Diffusion Load Balancing Algorithm in Distributed System", I.J. Information Technology and Computer Science, Vol. 08, 65-71, July, 2014.

[14] Ni L.M., Xu C.W. and Gendreau T.B., "A Distributed Drafting Algorithm for Load Balancing", IEEE Trans. on Software Engineering, Vol. SE-13, No. 10, October, 1985,pp.1153-1161.

[15] Suen T.T.Y and Wong J.S.K. "Efficient Task Migration Algorithm for Distributed Systems", IEEE trans. on Parallel and Distributed systems Vol. 3, N0. 4, July, 1992.