ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

MEASURING PERFORMANCE OF DYNAMIC LOAD BALANCING ALGORITHMS IN DISTRIBUTED COMPUTING APPLICATIONS

Priyesh Kanungo[1]

Professor and Senior Systems Engineer (Computer Centre), School of Computer Science and Information Technology

Devi Ahilya University, Indore-452001, India[1]

Abstract: In distributed applications, performance issues have become more critical due to proliferation of heterogeneous devices, large variety of communication medium and increased security concerns. This paper highlights the issues in performance measurement in Dynamic Load Balancing Algorithms (DLB) used for distributed scheduling. Various parameters used to measure the performance of scheduling algorithms have been described. The simulation model has been used illustrate performance issues associated with load balancing.

Keywords: Performance Measurement, Distributed Scheduling, Slowdown, Response Time, Processor Utilization, Dynamic Load Balancing.

I. INTRODUCTION

With the growing demand of resource intensive distributed computing applications, the need of using sophisticated performance improvement techniques has also increased. DLB is one of the techniques used extensively to improve scalability and overall throughput in distributed systems in the rapidly growing resource intensive distributed applications. It is responsible for task scheduling as well as monitoring load variation in the system. In such distributed applications, uneven process arrival may cause load imbalance, where some nodes are overloaded while some other nodes are idle [2, 7]. The fundamental aspect of load balancing in large clusters is that it needs to take into account many different parameters for driving process transfer decisions. DLB approach can create additional overhead in collecting system state, analysing the data, making load balancing decisions and transferring the processes from one node to another [8,10]. Thus, even though it is established that load balancing facility is necessary for improving the performance of a distributed system, the important issues of performance measurement parameters and quality of algorithm needs to be investigated further and are being considered in this paper.

II. PERFORMANCE MEASUREMENT

Load balancing facility improves the performance of the distributed system. Overall system performance can be measured by the following parameters [1, 3, 5]:

A. Mean Response Time

Performance of a load balancing algorithm can be measured by the response time. Response time is the time elapsed between start of execution of a process and its completion. To achieve the good response time, processes must be distributed evenly among the nodes using appropriate load balancing technique. Good response time also means that the processes don't have to wait too much in the system. Response time can be computed as follows:

$$T = F - A$$

Where,

T is Response Time

F is finish time of current CPU burst
S is start time of current CPU burst

Note that a process arrives and executed several times on the central processing unit during its lifetime.

B. Processor Utilization

Utilization of processor means the percentage of time for which the node is busy in running processes. This index is useful at lower load conditions. At the higher load

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

conditions, even after maximum utilization of the processor, some of the processes are waiting for execution. These processes can't be taken into account for measuring load.

C. Mean Slow Down

Sometimes response time or waiting time cannot give correct idea about the suffering of processes, particularly when there is huge variation in their processing times. *Slowdown* or *penalty ratio* can be used to measure proportionate suffering of processes in the system irrespective of whether the process is long or short. *Slowdown* of a process is defined as the ratio of total time spent by the process in the system to the execution time of the process and is defined as:

$$P = (t + w + m) / t$$

where,

P is mean slowdown or penalty ratio

t is execution time

w is missed or waiting time of a process in queue

m is migration time

III. DESCRIPTION OF THE ALGORITHM

DLB algorithms can be defined by their implementation of policies for load estimation, process transfer, status information exchange, node allocation and process migration. Load estimation policy specifies what workload information is to be collected, when it is to be collected and from where the information is to be gathered and what load indices will be used e.g. queue length, execution time or process age. Process transfer policy detects if the load imbalance conditions are prevailing and decides appropriate period of triggering the load balancing operation. State information exchange policy is necessary for exchange of load information among the nodes in the system to identify the nodes, which are either overloaded or under-loaded. Polling, broadcasting or on-demand techniques may be used to exchange state information. Broadcasting technique may lead to increased network traffic whereas other techniques allow selected information exchange [4, 6]. Node Allocation policy is needed to define the processes on an overloaded source node and to select an under-loaded destination node, where these processes will be migrated. Bidding, threshold, shortest or pairing technique may be used to decide a destination node. Process migration technique is required to actually transfer processes from source node to the destination node [9]. The formal algorithm for comparing various load indices and performance parameters is formally described below:

Algorithm *performance-measurement;*
/*Algorithm for measuring performance in DLB. Following load indices have been used: 1=No load balancing, 2 = Queue Length, 3=Process Age, 4= Execution Time.*/
{
store *threshold, avg-queue-length, avg-age, avg-exec-time* of the system

for each node *P* in the system
store its *ready-queue, fptr, rptr, load-level, mean-response-time, mean-utilization, mean-slowdown, mean-queue-length, mean-age, mean-exec-time;*

/* load level may be heavily-loaded, moderately-loaded or lightly-loaded*/

for each process in the system
store its *creation-node, PID, arr-time, ser-time, response-ratio, dep-time;*
/ new processes arrive at Pi */*

CreateProcessorQueue (struct processes ());

while (true) **do**
compute-threshold (*T*);

for every processor *P* in the DCS /*at every node concurrently*/
{
calculate-load (*W, P*);

if *W* > *T* then
load-level=heavily-loaded;
else
load-level= lightly-loaded;

while (*P* = heavily-loaded) **do**
{
choose-shortest-dest-node(*dest-node*);
if (destination-node-found)
{
select (*newprocess*);
migrate-process (*newprocess, P,dest-node*);
execute (*newprocess, dest-node*);
}
}
}
}
calculate-load (*W, P*)
{
if *load-index* = qlength

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

W= sum of queue lengths on processor;
if *load-index* is process-age
W = sum of age of processes on processor queue;
if *load-index* is execution-time
W= sum of execution times of processes on processor queue;
}
compute-threshold (*load-level*);
{
if *load-index* = *qlength*
{
T = *avg-queue-length* of processors;
}
if *load-index*= *process-age*
{
T= *avg-age* of the processes;
}
if *load-index*=exec-time
{
T = *avg-exec-time* of the processes;
}
} **End of Algorithm**

A. Simulation and Result Discussion

The simulator was designed and implemented to evaluate load indices for different performance measure parameter parameters viz. Processor utilization and mean slowdown. The simulator used artificial workload to carry out comparisons as this provides greater flexibility as compared to real workload. We assumed random process arrival and random service time distribution. The load balancer consists of server module that collects load information and makes job placements and migration module for remote execution of processes. Table I, Table II, Fig. 1 and Fig. 2 compare the results of simulation.

TABLE I
COMPUTATION OF PROCESSOR UTILIZATION USING DIFFERENT LOAD INDICES

Node Id.	Utilization of Processors			
	Without DLB	Queue Length	Process Age	Exec. Time
3	35	56	68	79
5	42	61	70	82
4	51	65	71	83
7	59	69	77	87
8	65	74	81	89
6	71	77	83	90
2	79	83	87	92
1	95	85	89	93

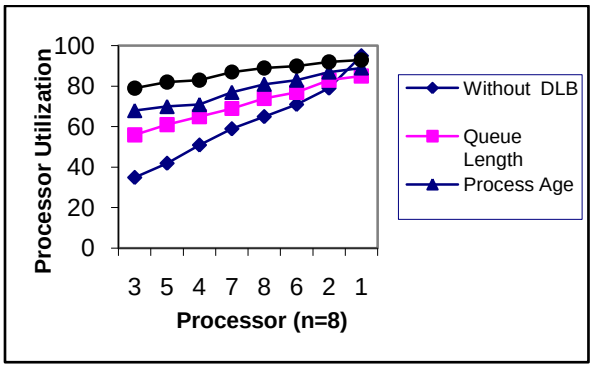

Fig. 1. Processor utilization and different load indices (node numbers are in ascending order of mean response time).

TABLE II
COMPUTATION OF MEAN SLOWDOWN USING DIFFERENT LOAD INDICES

Node Id.	Mean Slowdown Time of Processes			
	Without DLB	Queue Length	Process Age	Exec. Time
3	1.4	1.6	1.7	1.9
5	1.6	1.7	1.9	1.95
4	2.0	2.1	2.0	2.1
7	2.3	2.4	2.3	2.3
8	2.7	2.9	2.5	2.4
6	3.8	3.1	2.9	2.6
2	4.2	3.5	3.1	2.7
1	5.6	3.8	3.3	2.8

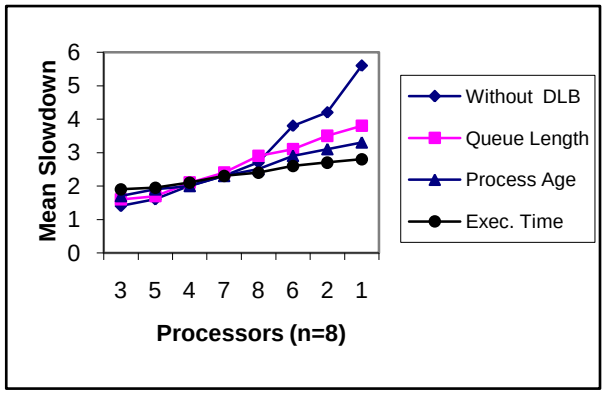

Fig. 2. Mean slowdown and different load indices (node numbers are in the ascending order of mean response time.

The graph in Figure1compares the three load indices on the basis of mean response time of the processes. Figure 1 compares the different load indices on the basis of processor utilization on different nodes. Figure 2 compares the load indices on the basis of mean slowdown of the

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

load indices is better than no load balancing at all. Among the three load indices, execution time as a load index gives better results. But we know that if is difficult to estimate execution time of the processes before actually executing them. However, it works as standard to compare other implementable algorithms. Process age as load indices gives better results than queue length.

IV. CONCLUSION

In this paper, we have investigated various issues in DLB and studied various parameters for effective performance measurement in a computing cluster with respect to different load indices. We have also compared various indices used to measure the load on the nodes using different performance parameters.

REFERENCES

[1] S.P. Ahuja, R. Eaggen and A.K. Jha, "A Performance Evaluation of Distributed Algorithms on Shared Memory and Message Passing Middleware Platforms," *Informatica*, Vol. 29, pp. 327-333, 2005.

A.M. Alakeel, "A Guide to Dynamic Load Balancing in Distributed Computing Environment," International Journal of Computer Science and Network Security, Vol. 10, No 6, pp. 153-160, June 2010.

V. Cardellini, A. Colajanni and P. Yu, "Geographic Load Balancing for Scalable Distributed Web Systems," Proceedings of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. IEEE International performance, San Francisco, CA, U.S.A, pp. 20-27., 2000.

[2] J.D. Herbsleb and A. Mockus "An Empirical Study of Speed and Communication in Globally Distributed Software Development," *IEEE Transactions on Software Engineering*, Vol. 29, No. 6, pp. 481-494, June 2003.

D. James, S. Olivier, J. Prins and C.W. Tseng, "Dynamic Load Balancing of Unbalanced Computations using Message Passing," Workshop on Performance Modelling, Evaluation and Optimization of Parallel and Distributed Systems, CA, pp. 26 – 30, 2007.

[3] H. Mehta, P. Kanungo and M. Chandwani, "A Modified Delay Strategy for Dynamic Load Balancing in Cluster and Grid Environment," *International Conference on Information Science and Applications (ICISA 2010 (IEEE))*, , Seoul, South Korea, Technically sponsored by IEEE, (Paper Available on IEEE Xplore), April 21-23, 2010.

[4] H. Mehta, P. Kanungo and M. Chandwani, "EcoGrid: A Dynamically Configurable Simulation Environment for Economy-Based Grid Scheduling Algorithms," 3rd ACM Annual Conference Compute-2011, Bangalore, March 25-26, 2011(ISBN: 978-1-4503-0750-5).

[5] M.A. Razzaque and C.S. Hong, "Dynamic Load Balancing in Distributed Systems: An Efficient Approach," Joint Conference on Communication and Information (JCCI), Pheonix Park Korea, May 2-4, 2007.

[6] R. K. Sharma, P. Kanungo and M. Chandwani, "A Destination Capability Aware Dynamic Load Balancing Algorithm for Heterogeneous Computing Environment," Proceedings of International Conference on High Performance Architecture and Grid Computing (HPAGC-2011), Department of Computer Application, Chitkara University Punjab in Association with University of Applied Science, Osnabueck, Germany Communication in Computer and Information Science, Volume 169, pp. 186-191 Published by Springer Verlogue, Berlin, 19-20 July, 2011.

[7] L.F. Wilson and W. Shen, "Experiments in Load Migration and Dynamic Load Balancing in SPEEDS," *Proceedings of Winter Simulation Conference*, pp. 483-490, 1998.

BIOGRAPHY

Dr Priyesh Kanungo is working as a Professor and Senior System Engineer (Computer Centre) in Computer Science, at Devi Ahilya Vishwavidyalaya, Indore. He also worked as a Principal in Patel College of Science and Technology, Ralamandal, Indore for two years. He received ME, M Phil and Ph D in Computer Engineering from DAVV, Indore.

His areas of research are Distributed Computing, Grid Computing and Cloud Computing. He is having more than 23 years of teaching experience in M Tech, MCA, MBA, BE etc. in DAVV, Indore and RGPV, Bhopal. He has guided one Ph D and is presently guiding 8 Ph Ds in the area of Distributed Computing, e-Governance and Cloud Computing at School of Computer Science and Information Technology and Institute of Engineering and Technology, Devi Ahilya University, Indore (India). He has published around 40 research papers in reputed international journals and conferences.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013