Balancing Load in Parallel Computing through Program Slicing

Dr. P.A.Tijare[1], Dr. P.R.Deshmukh[2]
[1]*Associate Professor, Sipna College of Engineering & Technology, Amravati,*
[2]*Professor, Govt. College of Engineering, Nagpur, India*
[1]*pritishtijare@rediffmail.com,* [2]*pr_deshmukh@yahoo.com*

Abstract—Execution time of a program can be diminished and performance can be upgraded by balancing the load. Our work proposes parallel task execution. By watching the load on the system, tasks can be executed for may iterations through various nodes. Our analyses demonstrate the proposed algorithm lessens finishing time in parallel computing task.

Keywords— *dynamic load balancing, parallel computing, slicing*

1. Introduction

Parallel Computing is a kind of calculation where we can accomplish load balancing by partitioning computer problems into littler jobs and afterward execute parallel on different systems.

The ongoing advancement in the parallel processing has given numerous facilities in transmission and control of information through network. Notwithstanding, these facilities and advancement came with the challenges in parallel computing. High complexity of building parallel application is regularly referred as one of the significant obstacles to the standard selection of Parallel Computing. Numerous scientific computations require a lot of computing time, the computation duration can be diminished by isolating an issue more than a few processors.

There are various load balancing algorithms exists which encourages the task of recognizing a reasonable load balancing methodology. Distinctive groupings have just been anticipated; however every classification was centered on specific applications. Couples of them are associated with process scheduling in distributed operating systems and with job scheduling in parallel applications dependent on functional disintegration.

It is a challenge to balance load on multi computers since the processors are independent and furthermore the correspondence overhead expense is incorporated to discover present load information. For solving/running distributed and parallel program applications, it is smarter to utilize parallel and distributed processing condition[1].

2. Related Work

To improve the eventual outcomes of the system, researches are carried out by using parallel computing. At present, Message Passing Interface is the one of the parallel programming tool in cluster computing [2].

.

In [3], proposed strategy in Message Passing Interface parallel program which transfers the tasks between nodes adequately based on system load. Load balance incorporates static and dynamic load balance, static load balance is not relate to state of system and it has low efficiency; while dynamic load balancing algorithm can balance the loads of nodes, having great practicability [3].

In research paper [4] , they have given a nonexclusive model to the clients who aim to compose parallel projects over the conveyed framework with CORBA middle ware [5]. They have dynamically created slave object to satisfy the master/slave parallel computing paradigm by utilizing the idea of factory object.

A paper [6], the Design Pattern and Distributed Processes model use MIMD Processor Architecture and Operating system which makes process and pass message for correspondence. In their model, parallel program is represented by a directed graph. Every node in the graph has a lot of input and output ports utilized for accepting and sending messages individually. Output port of a node is associated with an input port of another node.

At the point when a node sends a message to one of its output ports it reaches the input ports of the connected node which can get this message. They have discovered the abstraction of nodes and ports very significant in designing and utilizing design-pattern based system. It has enabled the model to stay free of the particulars of the fundamental message passing models, for example, PVM [7], Sockets [8] or MPI [9].

3. Calculation and distribution of load

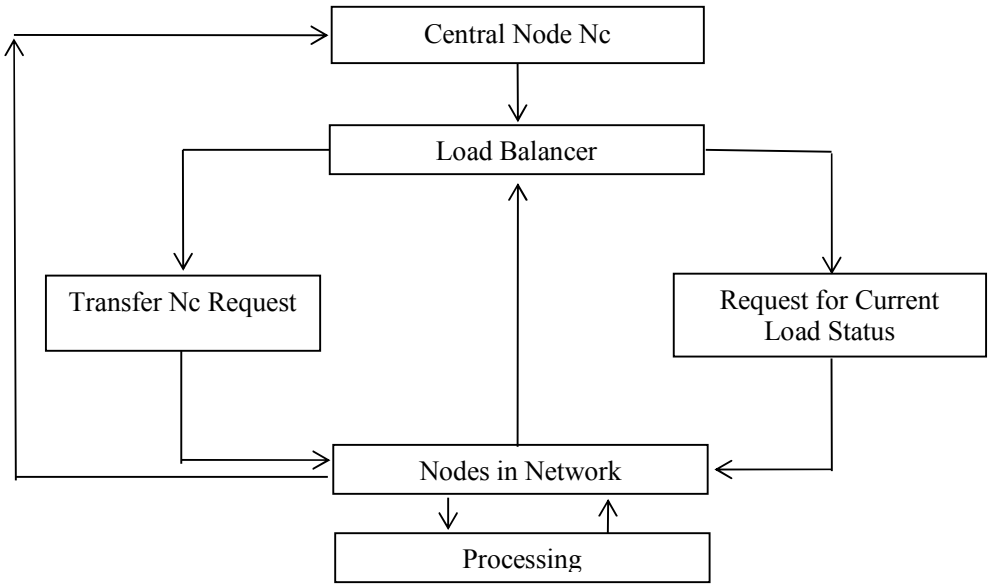

Figure 1. Dynamic load balancing architecture

Above figure 1 demonstrates the architecture, in which the central node Nc ask for current status of load to all nodes present in the network Wi. Each and every node available in the network will respond with their individual load information to central node. In view of this information central node takes choice whether to move the load or not. In the wake of getting demand, nodes in the cluster prepared the information and results back to the central node.

Given a parallel system with N processors, every processor i is estimated to have compute load of Wi. For load balancing strategy we are not considering correspondence cost. So the aggregate CPU load Li for every processor is spoken to by

$$Li = Wi$$

At first demand is sent from central node Nc to every single other nodes Ni present in system to discover the present load Wi of every node Ni. The load is nothing but the

number of processes CPU is executing at that time. At that point every node Ni present in the network will respond with current load Wi it is taking care of.

Ventures for Workload Calculation and dissemination

1. Network checking for number of systems present in the network.
2. Each Node Ni gets load status ask for from Central node Nc.
3. Each node Ni responds with their load status to Nc
4. Each node load status Li is looked by Nc.
5. Compare load status Lc of Nc
6. Distribute equivalent load among all Ni where i=1,2,3,….n
7. Repeat the above strides for any new approaching load.

4. Program Slicing

The method for making cuts of the program dependent on some slicing criteria is nothing but program slicing. Diverse methodologies on Program Slicing of various looks into are there for various kinds of program. Our primary center is essentially making the genuine utilization of slicing algorithm in the computing world.

The genuine utilization of slicing is just when the slices are autonomous in all regard i.e. running freely by taking the essential information, reliability with respect to necessary output. Traditional program slicing is done explicitly with slicing criteria. The Slicing algorithm created in research work use slicing point as spine of the algorithm. A proposed Slicing Algorithm is planned just for Iterative Program.

At slicing point, the logic of the program is gets repeat number of time till the completion of the program. In this work we refer “For Loop” as the slicing point. The Iterative Program contains redundancy of the similar logic number of times, and for the completion of the reiteration, the system holds up till the test condition is finished.

Slicing Algorithm Steps

The Slicing Algorithm is depicted in the accompanying Steps:

1) Form CFG of the program.
2) Identify slicing point of the program.
3) Find initial and last value of the iteration in slicing point.
4) From Last Value of the iteration divide the slicing point in number of iteration.
5) In recently created slice, include the program beginning articulation and the statement after the slicing point to the slice, so that the slice can be prepared for execution.

5. Results

Three systems in the network are considered for the experiment purpose. The number of tasks is coming to central node Nc, are to be transferred for execution to every one of the nodes, based on the threshold value. On the off chance that threshold value is greater than that of the processes running on Nc, at that point load will be transferred else it will be executed itself by Nc

Here we considered three numbers of systems present in the network. The task is sliced and sent to number of nodes for partial execution.

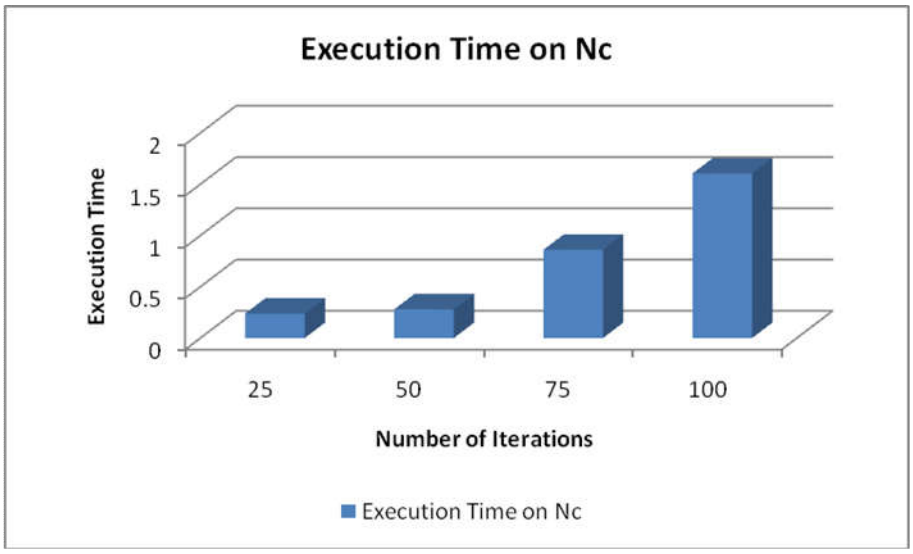

Figure 2. Execution Time on Nc

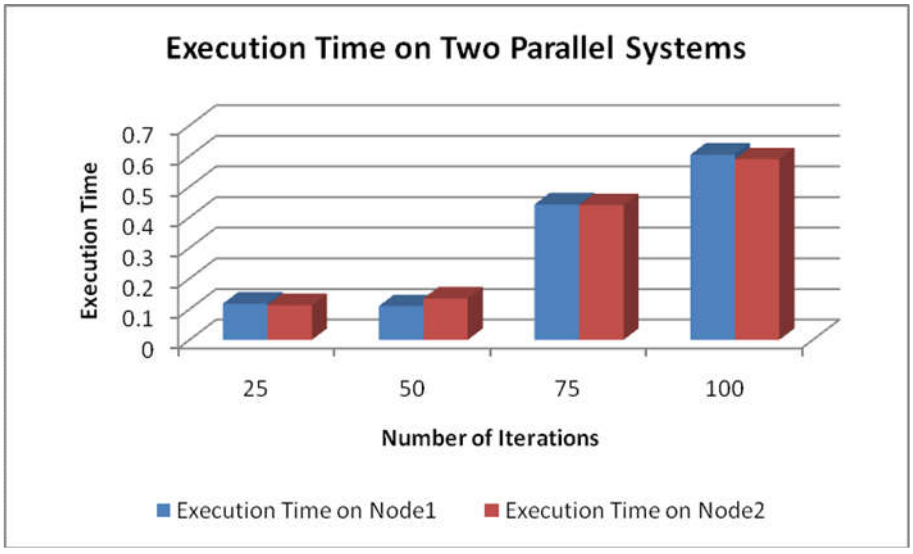

Figure 3. Execution Time on Two Parallel Systems

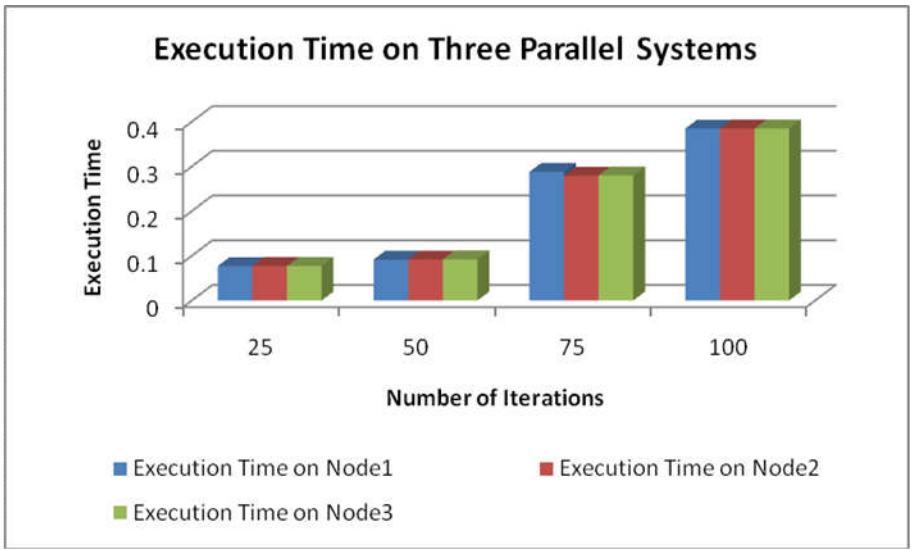

Figure 4. Execution Time on Three Parallel Systems

6. Conclusion

We have actualized execution of a task in parallel that can keep running for number of iterations between nodes adequately by node's load. The investigations demonstrate the execution of the algorithm in parallel computing task reduces completion time. Load balancing in parallel processing is accomplished through the slicing the iterative program and running the slice on other nodes to enhance the execution of the system.

References

[1] A. Chhabra, G. Singh, S. S Waraich, B. Sidhu, and G. Kumar Z, "Qualitative Parametric Comparison of Load Balancing Algorithms in Parallel and Distributed Computing Environment" Proc. World Academy of Science, Engineering and Technology, Vol 16, pp. 39-42, Nov 16, 2006.

[2] Z. Yongzhi, Z. Yan, W. Ronghui, "Constructing and Performance Analysis of a Beowulf Parallel Computing System Based on MPICH", Computer Engineering and Application 2006.

[3] S. Nian, L. Guangmin, "Dynamic Load Balancing Algorithm for MPI Parallel Computing", International Conference on New Trends in Information and Service Science 2009.

[4] Chi-Chang Chen Meng-Xiang Chen, "A Generic Parallel Computing Model for the Distributed Environment", IEEE Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies, 2006,

[5] G. Brose, A. Vogel, and K. Duddy, " Java Programming with CORBA". 3rd Edition, John Wiley & Sons, 2001.

[6] S. Siu and A. Singh, "Design Patterns for Parallel Computing Using a Network of Processors", Sixth IEEE International Symposium on High Performance Distributed Computing, 1997.

[7] G. Geist and V. Sunderam, "Network-based concurrent computing on the PVM system", Concurrency: Practice and Experience, 4(4):29:3-311, 1992.

[8] S. Leffler, M. McKusick, M. Karels, and J. Quarterman, "The design and implementation of 4.3 BSD UNIX Operating System", Addison- Wesley Publishing Company, Inc., 1990.

[9] D. Walker, "The design of a standard message passing interface for distributed memory concurrent computers", Parallel Computing, 20(4):657-673, 1994.