Dynamic Load Balancing Algorithm using Execution Time Prediction on Cluster Systems

Wan-Oh Yoon, Jin-Ha Jung, and Sang-Bang Choi
Dept. of Electronic Engineering, Inha University
Yonghyun-dong, Nam-ku, Incheon
402-751, South Korea
Tel:+82-32-860-7417, Fax: +82-32-868-3654 sangbang@inha.ac.kr

Abstract: In recent years, an increasing amount of computer network research has focused on the problem of cluster system in order to achieve higher performance and lower cost. The load unbalance is the major defect that reduces performance of a cluster system that uses parallel program in a form of SPMD (Single Program Multiple Data). Also, the load unbalance is a problem of MPP (Massive Parallel Processors), and distributed system. The cluster system is a loosely-coupled distributed system, therefore, it has higher communication overhead than MPP. Dynamic load balancing can solve the load unbalance problem of cluster system and reduce its communication cost.

The cluster systems considered in this paper consist of P heterogeneous nodes connected by a switch-based network. The master node can predict the average execution time of tasks for each slave node based on the information from the corresponding slave node. Then, the master node redistributes remaining tasks to each node considering the predicted execution time and the communication overhead for task migration. The proposed dynamic load balancing uses execution time prediction to optimize the task redistribution. The various performance factors such as node number, task number, and communication cost are considered to improve the performance of cluster system. From the simulation results, we verified the effectiveness of the proposed dynamic load balancing algorithm.

1. Introduction

Recently, the introduction of high performance microprocessor technology and high speed networks equipment are toward cluster systems which have low cost and high performance. Some examples of cluster systems are NOW (Network of Workstations), Beowulf, HPVM (High Performance Virtual Machine), and Solaris-MC. Such systems use SPMD (Single Program Multiple Data) programming style, which enables the same code to run on several processing nodes while the data space is partitioned among them. The libraries such as MPI (Message Passing Interface) and PVM (Parallel Virtual Machine) are supporting SPMD programming framework[3].

Nowadays, the cluster system is positioned between MPP (Massively Parallel Processors) and distributed systems. The number of nodes for cluster system is usually about 100, and it's network framework has a switching topology[1]. Usually, the cluster systems use Fast Ethernet for low cost but there is tendency for using ATM, Gigabit Ethenet, Myrinet and SCI network equipment to accomplish high performance although expensive cost.

Major defect on performance in the cluster system is load imbalance like parallel computers have. The cluster system is similar to loosely-connected MPP system. So the communication overhead is higher than MPP. Therefore, it needs more effective load balancing algorithm which has low communication overhead.

In this paper, cluster system using master-slave is modeled, and the master node can predict the execution time of each node based on the information from slave node. The master node calculates the task, that will be redistributed to each node on a predicted time and it considers the communication cost as the task move. Dynamic Load Balancing uses execution time prediction to optimize the task redistribution

2. The Cluster system

The cluster system is a type of parallel or distributed system, which consists of a collection of interconnected stand-alone computers working together as a single, integrated computing resource[1]. A computer node can be a single or multiprocessor system (PCs, workstations, or SMPs) with memory, I/O facilities, and an operating system.

The Cluster technology permits organizations to boost their processing power using standard technology (commodity hardware and software components) that can be acquired at relative low cost. The cluster system is a new supercomputing architecture that has efficiencies such as performance, availability, scalability and throughput[2].

3. Load Balancing Algorithm

3.1 Network Model

Fig. 1 shows master-slave network model. The parallel programs simulated follow the master and slave network model, where a master task generates a number of slave tasks[2]. Each slave task carries out some processing and sends result back to the master. After receiving the results from all the slaves, the master task will be terminated.

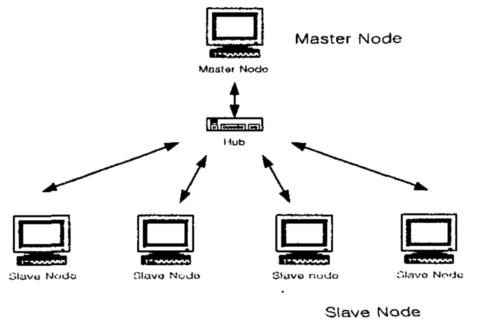


Fig 1. Master-slave network model

Cluster systems considered in this paper consist of N heterogeneous nodes and have switch-based network. The master node is randomly selected among nodes and has additional jobs which distribute tasks and gather results from slave nodes, processing own task as one node. We assume that the communication architecture of the switch-based network is the one-port model. The one-port model restricts a node to exchange messages with at most one node at a time. We also assume that multicast is not supported in hardware.

3.2 Dynamic Load Balancing Algorithm using Execution Time

First we propose the dynamic load balancing using predictable execution time without consideration of communication cost. In order to achieve the dynamic load balancing, master node has to calculate a task to move using the load information of slave node. Cluster system for parallel processing can't use absolute thresholds induced from CPU queue length, CPU utilization, and I/O usage which are used in the existing distributed system. Therefore, cluster system adopts the number of tasks in the *ready queue* of each node as basis for measuring the load[2].

N is the number of task that distributed with the nodes of the number of P equally. master node per time t_j collects the information of task number(n_i) in *ready queue* of each slave node i, $i=1, \cdots, P$. Master node calculates average time that each slave node executes a task through the collected information during period t_j.

$$T_{task_i} = \frac{t_j}{N - n_i}\ , \quad i = 1, \cdots, P \qquad (1)$$

N_{total} is the total number of tasks on slave node, Then

$$N_{total} = \sum_{i=1}^{P} n_i \qquad (2)$$

Ideally, in order to get the best performance of cluster system, all slave nodes should finish the task simultaneously. Thus, as shown in the following equation, all slave nodes should be redistributed by the task in the ratio of the performance of each node.

$$n'_1 \times T_{task_1} = n'_2 \times T_{task_2} = \cdots = n'_P \times T_{task_P} \qquad (3)$$

Solving the above two equations(i.e.,equations 2 & 3), we can obtain the following formula for n'_i :

$$n'_i = \frac{1}{T_{task_i} \sum_{j=1}^{P} \frac{1}{T_{task_j}}} \times N_{total}\ , \quad i = 1, \cdots, P \qquad (4)$$

In MPP systems which have high speed interconnection network, the load balancing cost is low. However, in cluster system which have expensive communication cost, we need an effective load balancing algorithm which has low overhead. In this paper, n'_i is calculated with prediction of total task execution time (E_i) considering communication cost. We define average task execution time of slave node that overall numbers of task execution time ($T_{average}$) are divided by overall number of node. The node which T_{task_i} is bigger then $T_{average}$ is called *overloaded* node, and the one which is smaller than $T_{average}$ is called *underloaded* node.

All slave nodes calculate the communication cost concerned E_i value with increasing the task from *overloaded* node to *underloaded* node achieve the task quantity calculated from formula 4.

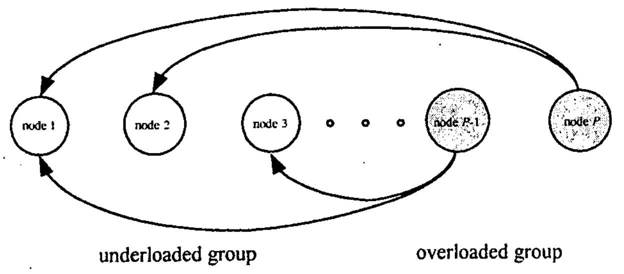
Herewith, the communication cost of task *m*(bytes) migration is as following.

$$C_t = \text{delay time} + \text{transmission delay}$$

Delay time is the time which need for system to migrate the task and transmission delay is the network overhead which takes to migrate the task *m*(bytes). Cluster system's total execution time is equal to the biggest value of each node's execution time E_i, therefore node's select is migrating the task from *overloaded* node group to *underloaded* node group (Fig 2). Everytime task migration, each node's E_i value is recalculated with following formula.

overloaded : $E_i = (n_i - k_i) \times T_{task_i} + k_i C_t$

underloaded : $E_j = (n_j + k_j) \times T_{task_j} + k_j C_t$



where, $1 \le i,\ j \le N$ and $\sum k_i = \sum k_j$.

It gives the information on value of E_i that is the smallest number of E_i when all nodes are the same number of task number in formula 3.

4. Simulations and Implementation

4.1 Simulation and Analysis

In this paper, we studied dynamic load balancing using a simulation model. We compared general dynamic load balancing algorithms such as central dynamic load balancing (CLB), decentral load balancing (DLB) with our proposed dynamic load balancing algorithm using execution time and marginal central dynamic load balancing (MCLB). Existing implemented cluster systems employ switch topology and the group nodes by multiple of 2 such as 4, 8, 16, 32, 64, 128 nodes group. Task model in each node uses as follows[4]. In our simulation model, each node has 200 tasks. But each task execution time has random value. Execution time of a task j of node i is $r_j(i)$. And distribution of $r_j(i)$ have a uniform random distribution with the range, $0 < r_j(i) < 2r_j(i)$. $r_j(i)$ is average task execution time in node i. $r_j(i)$ is distributed in the range, $0 < r_j(i) < 2E(r)$. $E(r)$ is average task execution time in all node. We varied C_{msg} from 1% of $E(r)$ to 5% of $E(r)$. C_{task} (one task migration time) is set into 10 times of C_{msg}.

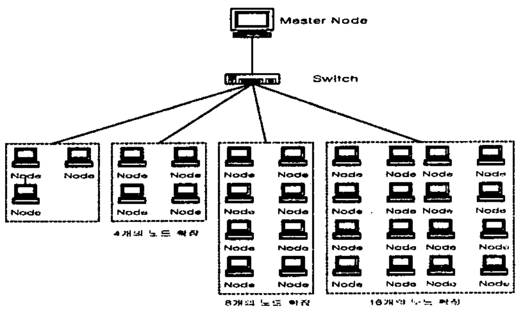


Fig 3. An example of the network configuration (the number of nodes is 32)

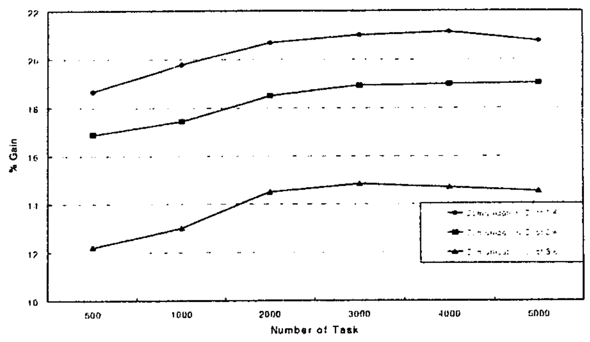


Fig 4. The gain versus the number of tasks for various communication overheads (the number of nodes is 16).

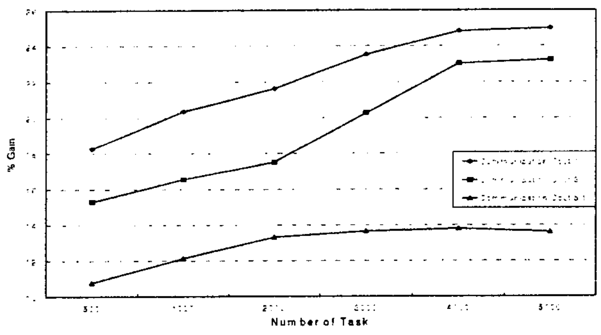


Fig 5. The gain versus the number of tasks for various communication overheads (the number of nodes is 32).

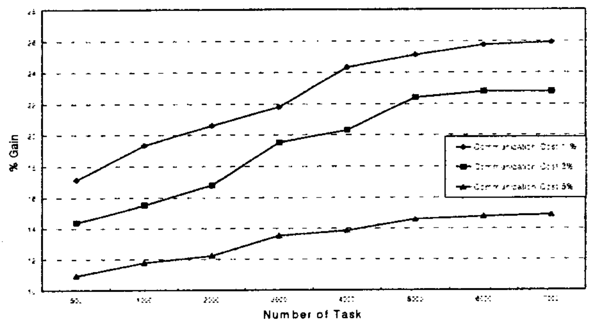


Fig 6. The gain versus the number of tasks for various communication overheads (the number of nodes is 64).

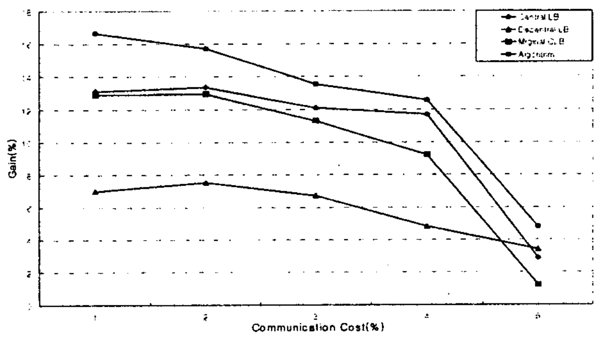


Fig 7. The comparison of gains obtained from four algorithms (the number of nodes is32)

C_{task} is a data segment movement time. So the overhead is large. Fig. 3 is an example of network configuration. Fig. 4~6 show simulation results in network consisting of 16, 32, and 64 nodes. The gain, *G*, is the ratio of total execution time in no load balancing to total execution time in load balancing.

$$G = \frac{ET_{NOLB} - ET_{LB}}{ET_{LB}}$$

ET_{NOLB} = *Execution time without load balancing*

ET_{LB} = *Execution time with load balancing*

Fig. 7 shows to make a comparison between gain rate of proposed algorithm and one of other algorithms.

4.2 Implementation

In this paper, we implements a linux cluster system to evaluate performance of proposed load balancing algorithm. It measures the performance using Netpipe (Network Protocol Independent Performance Evaluator)[6] that evaluate the performance of network protocol and NPB (NAS Parallel Benchmark). Maximum bandwidth in experiment is about 7.8Mbps which is unreached to the maximum values 10Mbps in theory (Fig 8 and 9). Therefore, the data transfer time C_t from benchmarking test is $C_t = 242 + 235 \times m(byte/\mu\text{sec})$. This value is average one from many tests.

To apply the algorithm proposed in this paper, we need parallel program model. We obtain gain rate by

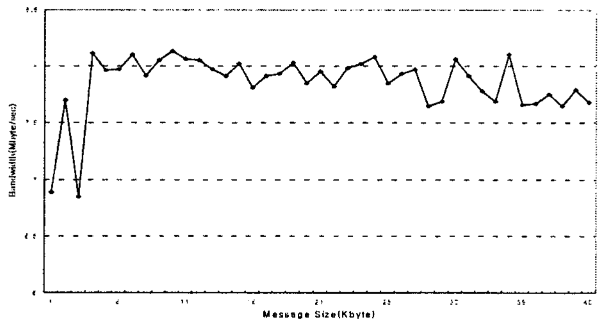


Fig 8. The value of bandwidth by the Netpipe

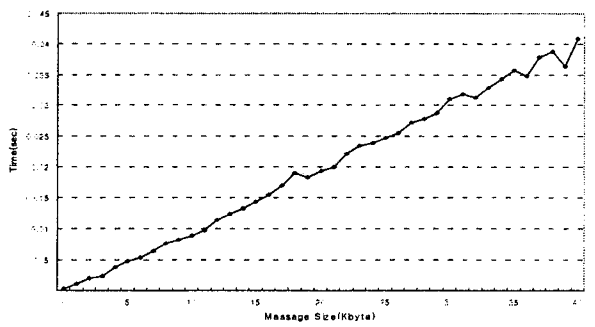


Fig 9. The message transfer time of point-to-point communication by the Netpipe

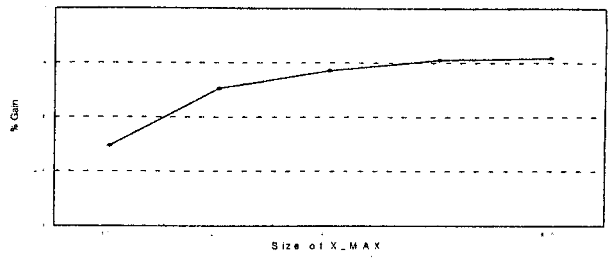


Fig 10. The gain from the proposed algorithm

applying heat conduction model with proposed algorithm. It's about 5.6% gain rates (Fig 10).

5. Conclusions

In this paper, the proposed dynamic load balancing uses execution time prediction to optimize the task redistribution. The various performance factors such as number of nodes, number of tasks and communication cost are considered to improve the performance of cluster system. From the simulation results, we verified the effectiveness of the proposed dynamic load balancing algorithm. We implemented four node Beowulf cluster system to find the speedup from the cluster system implemented using the algorithm proposed in this paper. The performance of the implemented cluster system is measured using Netpipe benchmark program for network performance evaluation and NPB benchmark program to measure the computing capability of cluster system. The network bandwidth measured in the real execution of Netpipe program is less than the value obtained from the simulations in the 10Mbps Ethernet environment. We also run the heat conduction model using the implemented cluster system to evaluate the performance of the proposed algorithm and achieved the remarkable performance improvement.

References

[1] Rajkumar Buyya, "High Performance Cluster Computing Volume 1 - Architectures and Systems"

[2] M. Cermele, M. Colajanni, and G. Necci, "Dynamic Load Balancing of Distributed SPMD Computations with Explicit Message-Passing," IEEE97, ISSN:0-8186-7879-8/97

[3] Wentong Cai, Bu-Sung Lee, Alfred Heng, and Li Zhu "A Simulation Study of Dynamic Load Balancing for Network-based Parallel Processing," IEEE 1997, ISSN : 1087-4089/97

[4] Marc H. Willebeek-LeMair, and Anthony P. Reeves, "Starategies for Dynamic Load Balancing on Highly Parallel Computers," IEEE Transactions on Parallel and Distibuted Systems, Vol 4, No. 9

[5] Niranjan G. Shivaratri, Philip Krueger, and Mukesh Singhal, "Load Distributing for Locally Distributed Systems," IEEE Computer, Vol 25, No. 12, Dec. 1992.

[6] Netpipe, http://www.scl.ameslab.gov/netpipe.

[7] mpi, http://www.-unix.mcs.anl.gov/mpi/