Proceedings of ITC-CSCC 2000, Pusan, Korea

Load Balancing Strategies for Network-based Cluster System

Hoon Jin Jung   Choung Shik Park   Sang Bang Choi

Dept. of Electronic Engineering, Inha University

253 Younghyun-Dong, Nam-Gu, Inchon, 402-751, Korea

Tel: +82-32-860-7417, Fax: +82-32-868-3654

Email:sangbang@inha.ac.kr

Abstract: Cluster system provides attractive scalability in terms of computation power and memory size. With the advances in high speed computer network technology, cluster systems are becoming increasingly competitive compared to expensive parallel machines. In parallel processing program, each task load is difficult to predict before running the program and each task is interdependent each other in many ways. Load imbalancing induces an obstacle to system performance. Most of researches in load balancing were concerned with distributed system but researches in cluster system are few. In cluster system, the dynamic load balancing algorithm which evaluates each processor's load in runtime is purpose that the load of each node are evenly distributed. But, if communication cost or node complexity becomes high, it is not effective method for all nodes to attend load balancing process. In that circumstances, it is good to reduce the number of node which attend to load balancing process. We have modeled cluster systems and proposed marginal dynamic load balancing algorithms suitable for that circumstances.

1. Introduction

Recently, the introduction of high performance microprocessor technology and high speed network equipment is toward cluster systems which have cheep price 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.

Nowadays, the cluster system is in positioned between MPP(Massively Parallel Processors) and distributed systems. The number of node for cluster systems are usually around 100, and it's network framework have a switching topology[1]. The cluster system usually use Fast Ethernet for low cost but use ATM, Gigabit Ethernet, Myrinet and SCI network equipment for high performance with expensive cost.

Main obstacle to performance in the cluster system is load imbalancing like parallel computers doing. The cluster system is similar with loosely-connected MPP system. So the communication overhead is more than MPP. Therefore, it needs more effective load balancing algorithm which has low communication overhead.

In this paper, we proposed marginal dynamic load balancing which is suitable for cluster system. And we showed that it is better than existing load balancing algorithms.

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.

Cluster technology permits organizations to boost their processing power using standard technology (commodity hardware and software components) that can be acquired at a relative low cost. The cluster system is a new supercomputing architecture that have advantages such as high performance, high availability, high scalability and high throughput[1].

3. Load balancing in cluster system

Load balancing strategies minimize total execution time of a single application under unexpectable dynamic varying circumstances through load redistribution. They are classified into static load balancing algorithms and dynamic load balancing algorithms. Static load balancing algorithms are task redistribution schemes which make load balancing before application run, and dynamic load balancing algorithms are task redistribution schemes which make load balancing while application is running. Generally, dynamic load balancing algorithm is called load balancing algorithm. So, the term, load balancing which is topic of our thesis means dynamic load balancing.

Cluster system usually supports SPMD style programs. A large number of parallel programs belong to this class: linear algebra problems, partial differential equation solvers, image processing algorithms. Since the workload on each processing unit is a function of the number of elements contained in its sub domain, we can keep the workload balanced by means of data migrations from overloaded to underloaded nodes.

Cluster system for parallel processing can't use absolute thresholds induced from CPU queue length, CPU utilization, and I/O usage which is used in existing distributed system. Therefore, cluster system adopts the number of tasks in the run queue of each node as basis for measuring the load[2].

In MPP systems which have high speed interconnection network, the load balancing cost is low. But in cluster system which have expensive communication cost, we need effective load balancing algorithm which have low overhead.

Fig. 1. shows a definition of nodes in load balancing. We define *maximum load* that is number of task which most overloaded node have. We define *minimum load* that is number of task which most underloaded node have.

We define *average load* that overall number of task are divided by overall number of node. If we divide the area from *maximum load* to *average load* by one hundred, the margin rate α is percentage distance from average load. Also we divide the area from average load to minimum load by one hundred, the margin rate β is percentage distance from average load. Table 1 shows classification of nodes in load balancing. Each node is included in the range.

Task migration of overloaded node is defined by margin rate α. And task migration of underloaded node is defined by margin rate β. So α and β have same value in order to match the number of migration tasks.

Table 1. Classification of nodes.

Classification of nodes	Range
overloaded node	χ > Average + (Max−Average) * α
middle overloaded node	Average < χ ≤ Average + (Max−Average) * α
middle node	Average
middle underloaded node	Average − (Average − Min) * β ≤ χ ≤ Average
underloaded node	χ < Average − (Average − Min) * β

Average: Average load value
Max: Maximum load value
Min: Minimum load value
χ : Random nodes in this condition

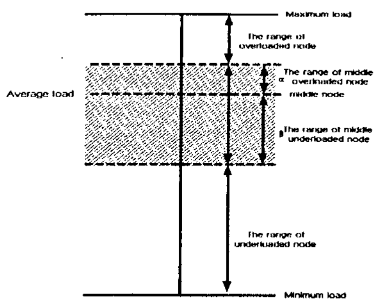
Maximum load
The range of overloaded node
The range of middle overloaded node
α
Average load
middle node
β
The range of middle underloaded node
The range of underloaded node
Minimum load

Fig. 1. A definition of nodes.

4. Load balancing process

Fig. 2. shows master-slave computation model. The parallel programs simulated follow the master and slave computation 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 terminate.

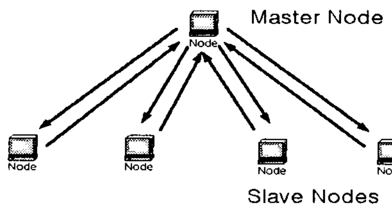

Fig. 2. Master-slave computation model.

Cluster systems considered in this paper consist of N homogeneous nodes and have switch-based network. Master node is selected out randomly among nodes and have 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.

The following 5 steps are performed in each load balancing time[3].

1) Broadcast of Balancing Start: A master node broadcasts a message indicating the start of load balancing except itself.

$$C_{master} = (N - 1) \times C_{msg}$$

2) Collection of Global Information: Each slave node send their load information to master node using global gathering method.

$$C_{response} = (N - 1) \times C_{msg}$$

3) Migration node determination by margin policy: Master node determines Average Load and chooses overloaded nodes and underloaded nodes by Margin policy. Two lists are created. First overloaded node list is current node whose load is greater than α margin quota. Second underloaded node list is current node whose load is smaller than β margin quota. The two lists are sorted in the increasing order of load value for task migration node matching.

4) Multicast for task migration: A master node broadcasts migration instruction to each node. Both overloaded node and underloaded node are received this instruction. The middle nodes, the middle overloaded node, and the middle underloaded node are not included in communication cost because they don't attend to load balancing process.

$$C_{master} = (N - 1 - M) \times C_{msg}$$

5) Task migration: Each overloaded node sends χ tasks to underloaded nodes. χ is a excessive task load. The cost of task migration is proportional to the task's size. In order to synchronize, two additional C_{msg} cost is also added.

$$C_{task} = \textit{unit time} \times \textit{the number of task}$$
$$C_{mig} = (2 \times C_{msg}) + C_{task}$$

Table 2. Simulation parameters for load balancing

Parameters	Meaning	Used value
T_{period}	load balancing period	1000 unit time
T_{total}	overall task sum of program	30000 unit time
C_{msg}	a node information sending cost	1~5% of E
C_{master}, $C_{receiver}$	communication cost between master node and slave nodes	$(N-1) \times C_{msg}$
C_{task}	a task migration cost	unit time × the number of task
C_{mig}	overall task migration cost	$(2 \times C_{msg}) + C_{task}$
α, β	load margin rate	10~40%
N	the number of node	31, 63, 127 nodes
M	the number of margin node	χ nodes
S	switch overhead	$C_{msg} \times 1.4$

5. Simulations & Results

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 marginal central dynamic load balancing (MCLB) and marginal decentral dynamic load balancing(MDLB). Table 2 shows simulation parameters for load balancing. Existing implemented cluster systems employ switch topology and group the nodes by multiple of 2 such as 2, 4, 8, 16, 32, 64 nodes group. So we simulated with same way. Also we made to simulation model like uniform network group and nonuniform network group. All of the communication cost were assumed unit time. C_{msg} can be said 1 unit time when it takes to send a unit message from one node to the other node in sub switch. And a message through main switch takes 1.4 unit time. We assumed that a parallel application is partitioned into independent tasks which don't have unlimited number. This is a fundamental assumption in load balancing. Even if tasks are distributed into each node with same number, it is impossible to know the exact total execution time of a node because each task is characterized by unpredictable execution time.

Task model in each node use as follows[4]. In our simulation model, each node have 100 tasks. But each task execution time have random value. Execution time of a task j of node i is $\gamma_j(i)$. And distribution of $\gamma_j(i)$ have a uniform random distribution with the range, $0 < \gamma_j(i) < 2\gamma(i)$. $\gamma(i)$ is a average task execution time in node i. $\gamma(i)$ is distributed in the range, $0 < \gamma(i) < 2E(r)$. E(r) is a 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}. C_{task} is a data segment movement time. So the overhead is large. Fig. 3. is an example of network configuration in nonuniform group.

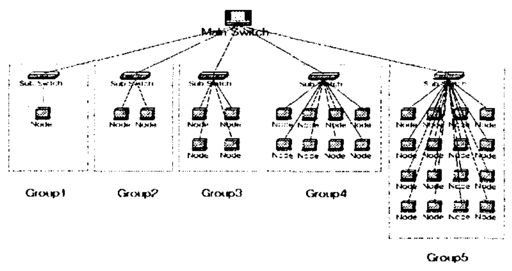

Fig. 3. An example of network configuration. (nonuniform group)

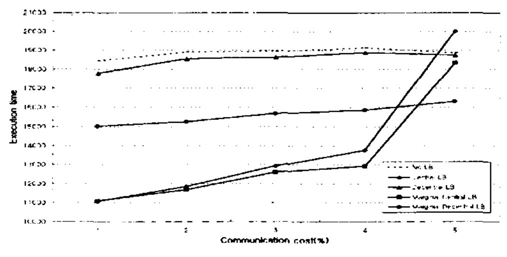

Fig. 4. The comparison of each algorithm's execution time. (31 nodes, nonuniform group, α = 30% margin)

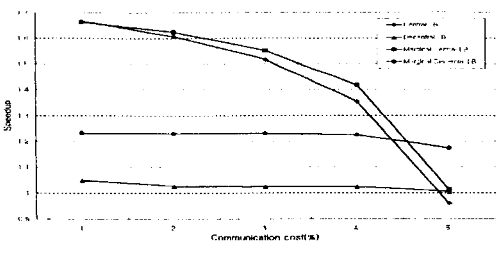

Fig. 5. Speedup variation by communication cost. (31 nodes, nonuniform group)

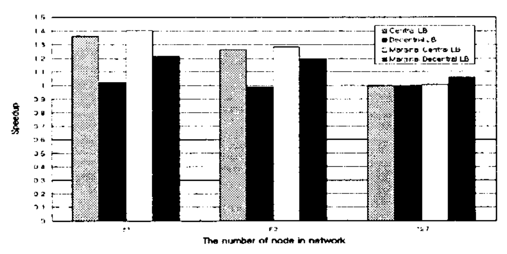

Fig. 6. Execution time variation by the number of node. (nonuniform group)

Fig. 4~6 show simulation results in network consists of nonuniform group. Speedup is the ratio of total execution time in no load balancing to total execution time in load balancing. Fig. 6. shows that speedup is getting worse in proportion to number of nodes because complexity is increased.

$$Speedup = \frac{T_{NoLB}}{T_{LB}}$$

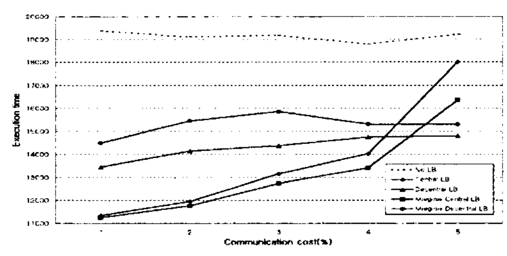

Fig. 7. The comparison of each algorithm's execution time. (31 nodes, uniform group, α = 30% margin)

Fig. 7~9 show simulation results in network consists of uniform group. They have similar results as nonuniform group except DLB algorithm.

6. Conclusions

We proposed marginal dynamic load balancing algorithm which vary the number of node attending load balancing dynamically according to communication cost in cluster system. In MDLB(Marginal decentral dynamic load balancing) algorithm, additional group communication to determine overall margin range is needed. But the advantage of MDLB is that cost of collecting overall node information is low.

Blocking Phenomenons happen very frequently in DLB because groups in DLB can't communicate each other. MDLB showed always better performance than DLB in nonuniform group network because MDLB solved the load unbalancing in small group network.

Each groups consist of same node number in uniform group network. In this case, the number of node in a group is enough to solve load unbalancing by itself, so DLB have better performance than MDLB. MCLB(Marginal central dynamic load balancing) algorithm showed always better performance than CLB because load balancing cost is reduced by marginal migration scheme. But the speedup of MCLB algorithm is getting worse than MDLB as node number increase. Consequently, MCLB is suitable for small group network and low communication network. MDLB is suitable for nonuniform group network and high communication network.

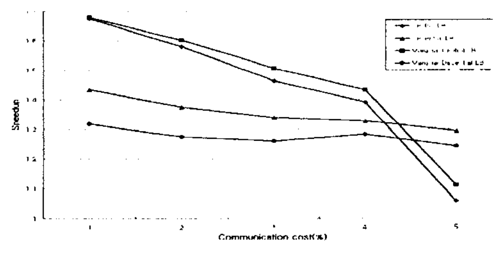

Fig. 8. Speedup variation by communication cost. (31 nodes, uniform group)

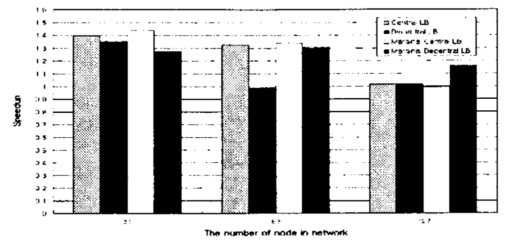

Fig. 9. Execution time variation by the number of node. (uniform group)

7. References

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

[2] M. Cermele, M. Colajanni, 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, 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, Anthony P. Reeves, "Starategies for Dynamic Load Balancing on Highly Parallel Computers", IEEE Transactions on Parallel and Distibuted Systems, Vol 4, No. 9