Dynamic Load Balancing Algorithm for MPI Parallel Computing

Sun Nian[1], Liang Guangmin[2]
[1]*Information Technology and Management Department Zhejiang Police Vocational Academy, Hangzhou 310018，China*
sunnynian@hotmail.com
[2]*School of Electronics and Information Engineering Shenzhen Polytechnic, Shenzhen 518055, China*
gmliang@oa.szpt.net

Abstract

MPI is the most important parallel programming tool in cluster currently. It implements communication in parallel program by message. Implementing load balance in MPI parallel program is very important. It may reduce running time and improve performance of MPI parallel program, aiming at solving the dynamic balancing problem in homogeneous cluster system. This paper proposes an implementing method in MPI parallel program that can transfer tasks between nodes effectively by node's load. The experiments prove the availability and practicability of the algorithm in parallel computing task.

1. Introduction

Cluster system have the characters of availability and extendibility, Load balancing problems of nodes are main barrier, being one of hot pots of scientific research domain. load balance includes static load balance and dynamic load balance, static load balance is not relate to state of system and it has low efficiency; Dynamic load balancing algorithm can balance the loads of nodes, having good practicability. Research of load balance continues to develop in heterogeneous system, but there are many applications in homogeneous cluster system.

At present, MPI is the most important tool of parallel program [1]. It achieves the communication of parallel program by adopting message passing mechanism. MPI has many merits such as good portability ,strong performance, high efficiency and so on, and it has all kinds of free, practical versions , for example MPICH、LAM、IBM MPL, all of parallel computer companies support MPI, Achieving load balance become very interesting in parallel program, It can improve the performance of MPI program.

This paper researches dynamic load balancing algorithm in homogeneous cluster system and proposes a new balancing algorithm based on MPI.

2. Summary of MPI parallel program

MPI is a typical interface of message passing mechanism. It can exploit parallel program based on message passing. MPI adopts program model of signal process signal data, in other words, each process carries out the same MPI program, MPI program can get the number of current program, program can differ from other program by the number, and each program can carry out different task to communicate with other processes.

MPI provides abundant of functions for message passing and related operation. Having hundred of function interfaces, especially, six functions make up the subset. They can achieve all functions of message passing parallel program:

```
MPI_INIT:
/* start MPI computing*/
MPI_FINALIZE:
/*conclude MPI computing*/
 MPI_COMM_SIZE:
/*decide the number of process*/
MPI_COMM_RANK:
/*decide the id number of process*/
MPI_SEND:  /*send a message*/
MPI_RECV:    /*receive a message*/
```

There is an example of basic structure of MPI application program:

```
#include "mpi.h"
main (argc,argv)
int argc;
char* argv;
{
int count;
int myid;
/* part1, initialization*/
```

978-0-7695-3687-3/09 $25.00 © 2009 IEEE
DOI 10.1109/NISS.2009.171

```
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&count);
MPI_Comm_rank(MPI_COMM_WORLD,&myid);
/* part2, papallel computing*/
if(myid==0)   //MASTER
{compution and communicating;
if(fetal error happens) MPI-About() ;}
else        //SLAVE
{
 Computing and communication;
if(fetal error happens)MPI-About();}
/*part3, exit the MPI environment*/
MPI_Finalize();
```

3. Decision strategy of dynamic load balance

There are two decision methods in dynamic load balancing algorithm, Centralism method which has a master (name center node) can collect loading conditions of every node and make globally decision, and this method can get optimal effect. Because center node has large Communication Workload and easily form communication bottleneck, Nowadays this method can't be adopted in most of systems. In distributional method, every node can only communicate with other nodes in certain scope to make decision; this method is reasonable that it can be adopted by many systems. The algorithm of this paper adopts distributional strategy method.

4. Parameters in dynamic load balancing algorithm

(1) Scale of system. Amount of processor in system is a parameter which can affect load balancing making-decision.

(2) Load state in system. Needing avoid the phenomenon of jolting.

(3) Input flows. Process may arrive to processor in any random Pattern, if processor can measure input flows itself and compare with other processors, it can easily estimate current load condition of the system. Making a decision-making to the task shift

(4) Loading threshold. Trapping loading threshold of task shift is a key parameter in system, because incorrect choice can result in Chain-reaction of system imbalance and task shift.

(5) Task scale. In general, It is inappropriate to shift a little running-time task. Similarly, a large process or process which infers to amounts of data and documents should better carry out in local processor [2]

(6) Cost of management. There are main elements to compose the cost of management, for example: current loading measure in processor, loading information of decision-making of processor, position of decision-making occurs, transferring tasks among processors.

(7) Horizon of load balance. A node may seek for a possible aimed node in neighbor-node scope and carry out this task. The diameter of neighbor-node scope defines for horizon. This parameter establishes quantity of neighbor-node in process of seeking for aimed node.

(8) Requirement of resource. The task can affect its shift to system resource's request. The process which needs many resources continues to wait for resources possibly. The possibility affects response time of system.

5. Dynamic load balancing algorithm

5.1. Definition of load state

In order to estimate the load state of node, It is general to consider one or synthesis many facts as follows [3]:

[1]: length of CPU queue;
[2]: availability of CPU;
[3]: available memory capacity;
[4]: speed of context switching

T.Kunz [4] researches and indicates that different factor has different efficiency and the difference is large. However the simplest factor is length of CPU queue which can get higher efficiency. An algorithm which this paper proposes adopts length of CPU queue as load appraisal factor, the definition as follows:

Definition 1:

Implementing ability of nodes in homogeneous cluster system , in other words in the unit time CPU can carry out the most numbers of processes which can describe with C, threshold value describes with A, which is 10% of C , processes in current CPU queue can describe with Q . Load state of node can describe with S which is decided by the blow function:

S=f(Q)

= **Heavy** Q>(C+A)

Normal $(C-A) \le Q \le (C+A)$

Light Q<(C-A)

Especially, **Heavy** presents that node is at overload condition, needing to transfer out partial processes; **Normal** presents that node is at suitable condition , it neither migrates the process to other nodes nor accepts the migration process; **Light** presents that the node is at unload condition ,may accept the processes from the overload node.

Migration process should be satisfied with the condition of time as follows:

Horchol-Balter and Downey [5] believe that the least implementing time of process which can be transferred among is that:

$$AGE_{\min} = \frac{f + u/b}{Qi - Vs/Vd \cdot (Qj+1)}$$

Especially, Vs presents implementing speed of source node that migrates out the process: Vd presents implementing speed of aimed node that accepts process; Qi presents numbers of process of source node before migration; Qj presents numbers of process of aimed node before migration; The cost of process migration is $f+u/b$, f presents cost of fixed migration of preemptive migration; u presents the cost of memory of migration process; b presents transferring bandwidth of memory.

5.2. A new task scheduling strategy

There are many tasks to carry out in the same time in parallel computing, it is vital that tasks are scheduled by certain method. Task scheduling strategy donates that tasks are dynamically migrated among computing nodes, transferring tasks from overload node to unload node, balancing load of computing node as far as possible in order to reduction program running time, according to each computing node's current load in MPI parallel program process. However, there are many difficulties to accomplish, needing some extra cost of communication. In general, three questions should be solved [6]:

1) How to collect current load information of system.
2) Deciding whether or not migrating tasks and where to migrate.
3) How to migrate the tasks,

Corresponding, task scheduling should include three parts:

1) Collection of information: Making load threshold value which can measure load information of node as well as making method of information collection.
2) Migration making-decision: judging whether or not migrating one task to other computing node according to task and load of node.
3) Implementation of migration: The task which is suitable to migrate to other computing node should choose aide node to migrate.

Above three parts interact with in different ways, "implementation of migration" uses the load information that "collection of information" provides. The task can migrate after it is judged as suitability by "Migration Making decision". Thus dynamic load balance needs a process to collect the load information and decide which node needs to migrate out, which node can accept task; the process is definite as scheduling process. Theory of the task scheduling strategy which this paper proposes is that the scheduling process's number is 0, other processes regard as computing processes. Because the computing scope of scheduling process is not large, scheduling process and one computing process carry out together in the same node. Specific method as follows:

1) each computing process sends information to scheduling process and inform the numbers of its rest numbers of task Q. if the rest numbers of task is Q>A+C, state function of node is f(Q)=heavy, sending message to scheduling process to migrate out task, if rest numbers of task is Q<C-A, state function of node is f(Q)=light; waiting for message of other computing processes; if the numbers of task is $C - A \le Q \le A + C$, state function of node is f(Q)=normal; and continue to implement.
2) scheduling process receives the message of computing process i, the rest numbers of task of computing process is Qi , The time of migrating process is Ti, then the state function of the node is f(Qi)=light and Ti= AGE_{min} , Registering the process as idle and waiting for message of other computing process. If the node f(Qi)=heavy and the other process j, f(Qj)=light and Ti= AGE_{min}, some of tasks of computing process i shoud be migrated to computing process j.
3) Computing process i receives number j of idle process from scheduling process, if f(Qj)=normal, continue to implement ; if f(Qj)=light and Tj= AGE_{min} , scheduling process will send message of task migration to computing process j.
4) If any computing process k, f(Qk)=light, then parallel program is concluded.

6. Experiment and performance analysis

6.1. Environment of experiment

There are 9 PCs in this experiment of clusters system, hardware and software conditions of this experiment as follows:

1) Configuration of each PC is that:
CPU: Intel(R) Pentium(R) 4 CPU 3.20GHZ 3.20GHZ
Board: ASUS A7V8X
Memory: 256MB
Hard Disk: 80GB

2) Equipment of network
Router: 16 ports 100Mb/s
Network adapter: 100Mbps
Auto-adapted network: adapter×16
Twisted pair wiring: some
3) Software Configuration of each PC is that:
Operation system: Linux RadHat9.0
Environment of parallel program: MPI

6.2. Data of experiment

This experiment adopts typical example of List Ranking Problem, List Ranking Proble is that each element k has a rank number (rank(k)) in the list, rank(k) can be regard as distance from k to tail of list. Therefore, every of k has a indicator of element, which can be described as next(k).if k is the last element in the list, next(k)=k, there is the specific algorithm:

Begin
(1) for all k∈L par-do
(1.1) P(k)=next(k)
(1.2) if P(k)≠k then distance(k)=1
else distance(k)=0
end if
end for
(2) repeat⌈logn⌉times
(2.1) for all k∈L par-do
if P(k) ≠P(P(k)) then
(I) distance(k)=distance(k)+
distance(P(k))
(Π) P(k)= P(P(k))
end if
end for
(2.2) for all k∈L par-do
rank(k)=distance(k)
end for
end repeat
end

This experiment makes use of the above parallel program to get the result as follows:
When scale of system is fixed, for example there is 10 PCs, Run time is as follows:

Table 1 Run time for data Scale

Scale of data	Run time of DNBLI	Run time of NBLB
50	0.237	0.221
100	0.656	0.561
150	0.837	0.781
200	0.955	0.823
250	1.236	1.056
300	2.245	2.012
350	3.656	3.321
400	5.213	4.981

When scale of data is fixed, for example, in above program, there are 200 elements in list, Run time is as follow:

Table 2 Run time for Node Numbers

Number of node	Run time of DNBLI	Run time of NBLB
1	4.407	4.109
2	3.055	2.850
4	1.976	1.401
6	1.230	0.985
8	1.089	0.889
10	0.865	0.791

According to above table1 and Table2, we can get the curve graphs as follows:

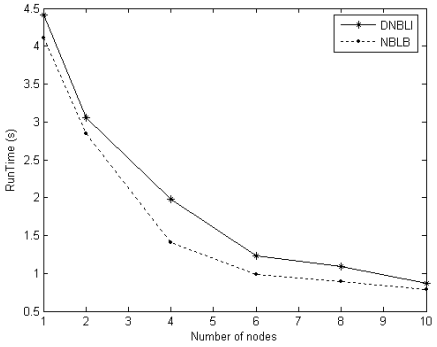

Figure 1 Run time for Node Numbers

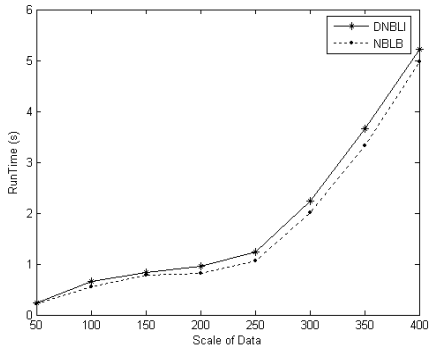

Figure 2 Run time for Scale data

Especially, DNBLI is load balance algorithm which is proposed by Franco, Zambonelli [7]. NBLB is a dynamic load balance algorithm of this paper.

6.3. Experiment data analysis

(1) If scope of data is the same, with increasing of scope of system, running time of algorithms is little. It indicates that RPDLB can get more interests of time

than other dynamic load balancing algorithm according to Figure 1.

(2) The difference of the cost of time among algorithms is also small when scope of data is small according to Figure 2. This indicates that the cost of load information that DNBLI and RPDLB cope with affects system efficiency when the numbers of data are little. With increasing of data scope, NBLB has better performance than other algorithm.

7. Conclusion

Nowadays, applications of homogeneous cluster system are much wider. The problems of dynamic load balance need to be solved in order to make cluster system have high performance. In this paper, we propose a better dynamic load balancing algorithm. By experiment we can prove it has the availability and practicability in parallel computing task.

8. References

[1] Zhu Yongzhi, Zhao Yan, Wei Ronghui, "Constructing and Performance Analysis of a Beowulf Parallel Computing System Based on MPICH", Computer Engineering and Application 2006.14:132

[2] Chan Guoliang, "Parallel Computing –Structure Algorithm Program", Higher education Press 2004.7

[3] Liu Bing, Shi Feng, "Research on Dynamic Load Balancing Algorithm Based on Message Passing Mechanism", Computer Engineering, 2007.5:58

[4] Kunz T, "The Influence of Different Workload Descriptions on a Heuristic Load Balancing Scheme [J]. IEEE Trans. on Software Eng", 1991, 17(7): 725-730.

[5] Balter M H, Downey AB, "Exploiting Process Lifetime Distributions for Dynamic Load Balancing [J] ACM Transactions on Computer Systems", 1997, 15(3): 253-285.

[6] Lu Kezhong, Chen Xiaohun, "Implementing Load Balance in MPI Parallel Program", Information of Microcomputer, 2007, 3 :227.

[7] Zambonelli.F, "Exploiting Biased Load Information in Directneighbour Load Balancing Policies [J]", Parallel Computing, 1999, 25(6): 745-766.