MPI动态负载平衡策略的研究与实现

卢 照，张锦娟，师 军，鱼佳欣
（陕西师范大学 计算机科学学院，陕西 西安 710062）

摘 要：集群环境下的并行计算越来越被广泛应用，MPI是集群系统中最重要的编程工具。在并行处理过程中，负载平衡起着很重要的作用，它直接影响到整个算法的效率。文中结合MPI编程环境下的具体特点，提出了基于负载益处估价的方法来判断是否进行任务迁移，给出了负载实时监测和调度的算法，并在每个节点机上间隔性地进行测试。最后在搭建的MPI环境下，运用并行排序方法进行了验证。实验结果表明采用负载前后有了很明显的提高，特别是随着任务量不断增大的情况下提高的效果更加明显。

关键词：集群；负载平衡；任务迁移；MPI并行程序
中图分类号：TP311 **文献标识码**：A **文章编号**：1673－629X(2010)05－0132－04

Dynamic Load Balancing Strategies in MPI Parallel Environment

LU Zhao, ZHANG Jin-juan, SHI Jun, YU Jia-xin
(School of Computer Science, Shaanxi Normal University, Xi'an 710062, China)

Abstract: Cluster parallel computing environment is used more and more widely. MPI is the most important programming tool. In parallel processing, load balancing plays a very important role, it directly affects the efficiency of the entire algorithm. In this paper, under MPI programming environment specific characteristics, and based on the benefits of the method of valuation to determine whether trasfer to the task, given the real－time monitoring of load and scheduling algorithms. In each node interval test and compute. Finally, in the MPI environment, the use of parallel sequencing methods to verify the scheduling algorithms. The experimental results show that the load before and after the improvement has been obvious, especially in the larger task, result is more obvious.

Key words: cluster; load balance; task transfer; MPI parallel programming

0 引 言

随着计算机技术的不断发展，人们对大规模数据量的计算越来越急切，高性能计算一直是人们关心的热点，也是一个国家综合实力的体现，目前并行计算广泛应用于大气与气象预测、石油勘测、水质污染测试、航空航天等方面。对于集群系统的成本低、性能好、规模可扩展性高的优点，目前大多采用集群系统来进行并行计算，也代表了高性能计算机发展的新方向[1]。

在集群系统中，经常会出现某些处理器的计算量过大而另一些处理器机负载很轻甚至空闲，人们试图把负载过重处理机的一部分任务转移到空闲或负载轻的处理机上运行来提高处理机的利用率和系统并行计算的效率，如何采取有效的调度策略来平衡各个节点的负载已是研究热点，负载平衡的目标是：

a. 保持所有处理机处于忙碌，而不是保持它们均分负载；

b. 提供最短的平均任务响应时间；

c. 有自适应性，能够适用于变化的负载；

d. 可靠性，避免负载的轻重跳跃，避免处理器抖动，减少不必要的通讯开销[2,3]。

1 MPI并行程序执行结构特点

MPI是目前被公认的应用前景最好的消息传递库，它采用消息传递的方式实现并行程序间的通信。它的最大优点是移植性好、功能强大、效率高和方便灵活易使用。在并行处理中被大量的用户所使用，几乎被所有的并行环境支持，而且有很多免费高效的版本，如MPICH、LAM、IBM MPL，几乎所有的并行计算机厂商都提供对MPI的支持，成为了事实上并行编程的标准[4]。

收稿日期：2009－09－07；**修回日期**：2009－12－03
作者简介：卢 照(1983－)，男，山西运城人，硕士，研究方向为并行算法、体系结构；师 军，副教授，研究方向为智能信息处理、并行计算。

MPI 为消息传递和相关操作提供了功能强大的库函数,MPI－1 中有 128 个库函数,MPI－2 中有 287 个库函数,从通信角度看都可以分为 6 个基本函数来完成,分别是:

MPI_Init()//启动 MPI 计算环境

MPI_Comm_size()//确定并行环境中进程个数

MPI_Comm_rank()//确定自身进程标识号

MPI_Send()//给指定的进程号机发送消息

MPI_Recv()//从指定的进程号机接收消息

MPI_Finalize()//结束 MPI 运行环境

在 MPI 程序中,每个节点机运行一个进程,且对应一个标识号,由 MPI_Comm_rank()函数获得各自的进程标识号,然后根据进程标识号来分配任务,让不同的进程即节点机去并行执行各自不同的任务,即通过进程号来识别各个节点机。为了更好更快地完成整个任务,各个进程之间要进行相互通讯与合作,实现程序的并行计算。

2　负载平衡策略

2.1　负载平衡问题的描述

运行时间是衡量并行程序性能的重要指标[5]。一个并行程序通常是由多个进程组成的,在集群环境下通常每个节点上运行一个进程,并行程序的运行时间是由最迟完成对应任务的节点决定的。这就需要通过调度与分配算法来实现任务的优化分配,从而有效地减少平均响应时间,降低执行时的额外开销。因此,负载平衡一直是并行程序设计中的一个重要考虑因素,它的一个作用是提高系统性能,缩短用户任务的平均响应时间;它的另一个作用是充分地利用整个系统的资源[6~8]。图 1 所示为串行程序并行化后出现的负载不平衡状况。

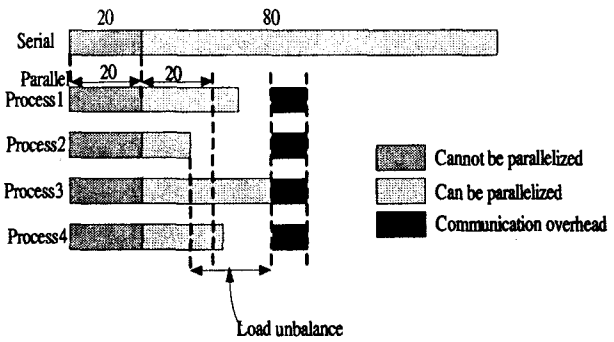

图 1　负载不平衡图

集群系统的负载平衡策略可以分为静态负载平衡和动态负载平衡,即初始放置和任务迁移。

2.2　静态负载平衡

静态负载平衡是根据系统自身的各节点处理能力来做出决策,将工作量相应的分配到各计算节点上,处理能力强的计算节点分配更多的工作,在运行过程中就不能再重新分配。这种负载仅考虑了各计算节点处理能力的差异,而未考虑各计算节点上负载的波动情况,仅当整个计算节点均单独用于该并行应用软件时才有效。

2.3　动态负载平衡

动态负载平衡是在运行过程的各个阶段中,不断根据运行过程中各计算节点的负载情况,将工作量从负载重的节点迁移到负载轻的节点上,尽量平衡各个节点的负载,使任务分配执行达到或接近最优,从而减少运行时间。它是静态负载平衡的有效补充,特别是在任务量估计不准确的情况下动态负载平衡会获得更好的效果[9]。

2.3.1　一些定义

为了能够很好地说明负载状态,根据具体情况从以下几个因素监测来进行综合考虑:

a:CPU 队列长度;

b:CPU 利用率;

c:内存可用空间的大小;

d:上下文切换速度;

e:I/O 吞吐率。

T. Kunz[10]的研究表明,采用不同判断标准时运行的效率差别很大,并在考虑 CPU 队列长度为标准时,可以得到较高的效率。在系统中负载描述如下:设系统中有 n 台处理机节点,分别为 $p_0, p_1, p_2, \cdots, p_{n-1}$,每台处理机拥有的任务数表示其负载,记为 $W(i)(0 \leqslant i \leqslant n-1)$,则整个系统的总负载为 $W = \sum W(i)$,系统的平均负载为 $W^* = W/n$。定义负载上界为 $W_u = W^* + A$,定义负载下界为 $W_b = W^* - A$(其中 A 为负载阈值)。当某个节点的负载值 $W(i) < W_b$ 时定义该节点为轻载节点;当 $W(i) > W_u$ 时定义该节点为重负载;当 $W_b < W(i) < W_u$ 定义该节点为适载节点;当 $W(i) = 0$ 时定义该节点为空载节点[11]。

2.3.2　负载开销

负载移动的开销主要包括两个方面:一是负载信息的采集开销;二是负载的移动开销[12]。在各个节点机上开辟两个线程,让其中一个线程专门用于负载信息的采集,尽量多地收集相关负载信息,这样就会更加准确地进行下一步的调度。负载移动会使计算节点接收大量的数据,这里用 MPI 自带函数 MPI_Recv() 来从指定节点上收集。为了减少移动次数和任务在计算节点的不断移动而不执行的颠簸现象,规定只有当某

个任务执行完成时才启动负载平衡策略，从进程标识号相邻的节点机接收新任务。

2.3.3 益处评估方法

在移动负载时，当移动的太少，会造成移动中的通讯开销和在移到的节点计算时间之和大于在源节点的计算时间，这样就划不来了，花费的时间会更多。这里定义一个函数 f 来判断是否需要迁移。设负载 m 要从 n_j 迁移到 n_i 上，估计函数为

$$f = m/s_j - m/s_i - c_{i,j}(m) \quad (1)$$

其中 s_j，s_i 分别表示 j 节点和 i 节点的计算速度。从式中分析可知，当 $f > 0$ 时，说明可以提高运行时间，从式中可以推出当负载 $m > c_{i,j}(m) * s_j * s_i/(s_i - s_j)$ 时才有必要进行迁移。

2.3.4 移动负载的粒度问题

在迁移负载时，到底迁移多少就成了要解决的问题，我们知道并行计算的运行时间取决于最慢的进程所花费的时间，当迁移任务过大过小都不合适，这里以上面为例，最好是让节点 i 和节点 j 能同时完成任务，即要使得它们花费的时间相同：$t_i = t_j$，即 $(W_j - W)/s_j = W/s_i$，式中 W 表示要迁移的负载大小。这样就可以求出[13]：

$$W = W_i * s_i/(s_i + s_j) \quad (2)$$

由于在 MPI 中任务量是以数据块进行传递的，当任务量大花费时间很多时，传递的时间可以忽略掉。

2.4 动态负载平衡算法

在前面提到，为了防止负载的来回迁移，只有当本身节点的任务计算完成后才启动负载平衡机制，这里采用接受者启动策略。且任务之间的迁移只在相邻标识号进程之间进行。在每个节点计算机上间隔调用测试程序。

(1)调度算法：

```
设本节点的进程标识号 MyRank=i,
While(1)
{
计算本节点现有的任务;
if(本节点现有任务完成)
{ MPI_Send(进程标识号为 i+1 的节点发送负载信息);}
监测 i+1 号进程的响应;
MPI_Recv(Load_Infomation);
//获得 i+1 进程的负载信息
求出(1)式中估价函数 f 的值;
if(f>0)
{根据(2)式计算所需要的迁移粒度;通知进程 i+1,发送任务数到 i;}
if(f<0 && Wi=0)//本节点任务完成且迁移条件不满足,退出
break;
}
```

(2)调度实时监控方法：

```
While(1)
{记录负载信息到 Load_struct 中;
if(有请求信息){
开启中断,延时 100ms;
将 Load_struct 信息发送出去;
While(1)
{ MPI_Recv 来自进程为 i 的分析结果;
if(结果为 T)
MPI_Send(向 i 进程发送所需的任务量);
else
关闭中断;
if(f<0 && Wi=0)
//本节点任务完成且迁移条件不满足,退出
break;}}}
```

3 试验分析

3.1 试验环境及配置

本试验是通过 3 台 PC 机(联想 P4 /1.86GHz/0.99GB * 2,P4 /2.4GHz/2GB * 1)在 Windows 系统下构建一个集群来实现的。通过应用 Visual C++ 6.0 编译器，要将 MPICH 提供的 include 文件和 lib 文件分别添加到 Visual C++ 6.0 编译器相应的库中。其中，由 0 号处理器作为主进程，对任务进行初始划分和结果的输出。

采用并行快速排序做验证试验。试验搭建系统结构图如图 2 所示。

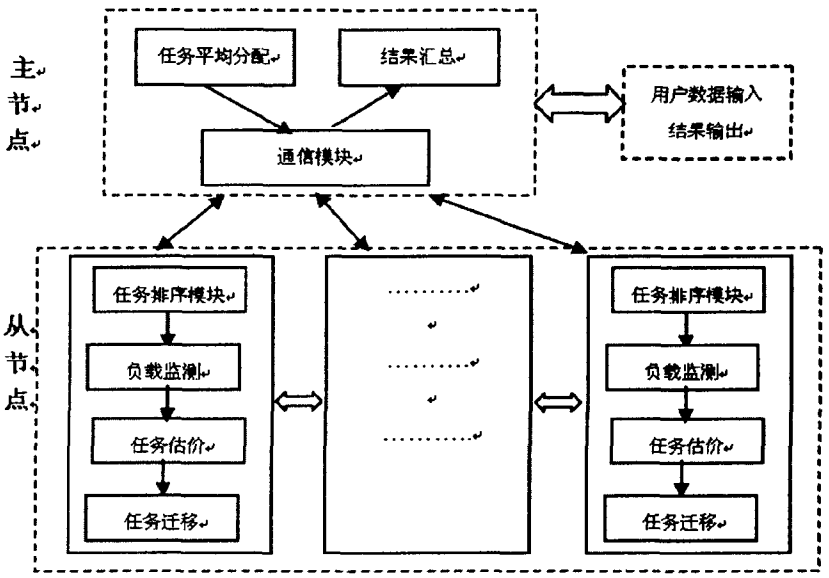

图 2 试验系统结构图

试验程序设计中的部分说明，在搭建的 MPI 并行

表 1　运行时间表

时间(s)	排序数据量(万)									
	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
单机	22.45	83.78	182.74	317.80	469.91	710.33	956.42	1255.50	1572.98	1953.96
3 台无负载平衡	19.89	79.16	160.44	277.13	381.32	492.15	678.66	896.78	1129.79	1440.87
3 台有负载平衡	19.03	77.25	144.99	249.45	339.26	445.01	569.36	699.36	839.32	1107.07

环境中,用快速并行排序算法求解来验证,采用主从式进行交互的,主进程将数据量均匀分配给各个从节点,从节点根据的计算速度只对所分数据中的部分数据进行排序,其他剩下的将作为负载迁移时的任务。在程序设计过程中,每个节点机都要分两部分进行:一是进行任务的计算,二是间隔调用监测程序段进行负载测试,决定是否需要迁移任务。这样一直继续,直到计算完成后,从节点将所有结果发给主进程节点机进行汇总输出。

3.2　试验说明及结果分析

程序设计中负载平衡信息 C 语言的定义结构体:

```
struct Load_struct
{
int My_Rank;//本进程的标识号
unsigned long Leave_Data;
//剩下的未进行排序数的数目
unsigned long Compute_Index;
//当前计算到的数据序列位置
double Compute_Time;
//该进程已经花费的时间
double Computer_Cpu_Speed;
//当前计算节点的计算速度
unsigned long Buffer_long;
//当前节点开辟缓冲区剩余的大小
unsigned short Procs_num;
//当前节点的进程数量
}
```

实际上在运算过程中主要用到了 CPU 计算速度,当前剩余数据未进行排序数据的起始位置,因为验证试验是对数据进行排序的一个过程,是计算密集型,在试验中也可以看出计算时 CPU 的使用率几乎达到了饱和状态,所以这里采用以 CPU 的使用率作为判断指标。以计算机计算速度即 CPU 主频来作为第一次计算时各个节点所分的数据大小的指标。

试验中对不同数据量进行排序后进行比较,数据量不大时体现不出优势,分别在单机、三台机的环境下进行了测试,用 MPI 函数 MPI_Wtime()前后做差来测试时间,需要说明的是这里只测试排序算法中的核心排序所花费的时间,数据输入输出的消耗都不计算在时间内。

测试结果如表 1 所示。

对应三种测试结果曲线图如图 3 所示。

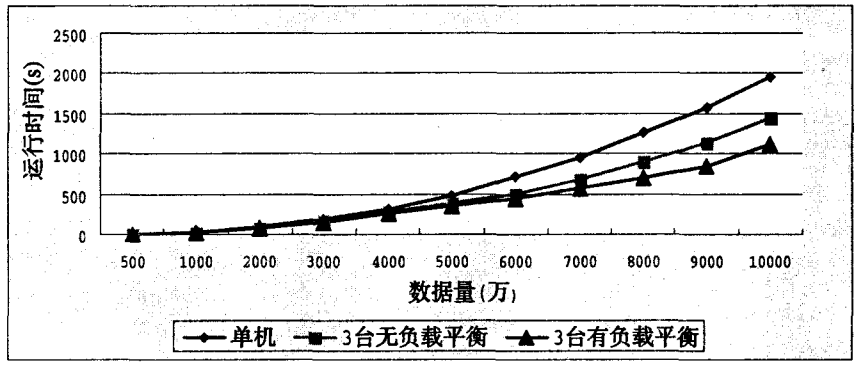

图 3　测试结果曲线图

试验中发现当运行程序时,可以看到 CPU 的利用率一下跳到了 50%左右,并保持不变,直到运行结束,从而,当在计算密集型的情况下,CPU 的计算速度及利用率可以作为很好的负载指标。

4　结束语

在并行集群下,负载平衡成为并行程序必须解决的问题,文中结合 MPI 程序的特点,在多机下实现负载平衡,提高了运算效率,对负载平衡方面的研究提供了参考。在试验中,对于实时监测时间控制比较关键,CPU 计算速度较快,很可能造成监测的负载信息已经是过去的负载了,造成计算不准确;同样根据任务类型不同综合考虑各方面因素来测试,从而具有更好的使用范围,这有待下一步深入研究。

参考文献:

[1] 陈国良. 并行算法的设计与分析[M]. 北京:高等教育出版社,2002.

[2] 焦素云,徐中宇. 分布式系统的动态负载平衡[J]. 长春光学精密研究机械学院学报,1999,22(2):41－43.

[3] 陈　涛,陈启买. 分布式计算机系统负载平衡研究[J]. 计算机技术与发展,2006,16(5):33－35.

(下转第 149 页)

的开发和管理。

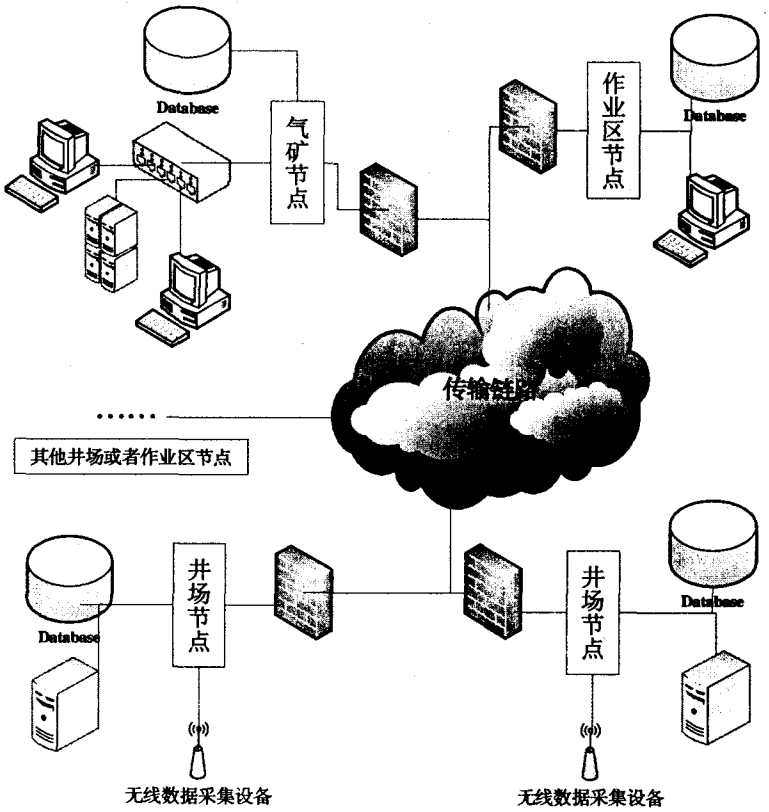

图 1　分布式数据库系统结构图

4　结束语

利用 Oracle 远程数据库访问技术和分布式事务处理能力,可以实现远程数据库间的数据共享、存储和更新[7]。

油气田安全生产系统数据库具有分布范围广、数据量大、数据提交适时性要求高等特点,分布式数据库具有数据共享、局部自治、可用性好、扩充性好等特点[8],将分布式数据库技术应用到安全生产领域,有效地解决了数据分散和集中管理的矛盾,减轻了气矿节点数据计算、管理和维护的压力,提高了油气田开发信息系统的各类数据的整合性,实现了数据的共享和交换,并建立了数据异地备份机制和双路网络传输链路,保障了数据的安全性和可靠性。实践证明,分布式数据库技术在远程数据管理中具有不可替代的作用。

参考文献:

[1] 潘群华. 分布式数据库系统中数据一致性维护方法[J]. 计算机工程,2002,28(9):251－252.

[2] 郦文华,姚　健,焦建栋,等. 运用 J2EE 框架技术构建公共卫生信息平台[J]. 计算机技术与发展,2008,18(12):192－196.

[3] 葛卫民. 基于 Oracle 高级复制的分布式数据库系统应用研究[J]. 计算机工程与应用,2003,39(2):180－181.

[4] 李　磊. Oracle 分布式数据库技术在四川地震检测中的应用研究[J]. 科技信息,2008(22):398－399.

[5] Oracle 数据库 DBA 管理手册[EB/OL]. 2001－04－19. http://www.china－pub.com/第三章.

[6] Oracle 数据库链接建立技巧与实例讲解[EB/OL]. 2008－10－18. http://www.bitscn.com/ 中国网管联盟.

[7] 姚文琳,存　刚. 基于 Oracle 的分布式数据库设计与技术[J]. 计算机工程,2006,32(20):89－91.

[8] Stonebraker M, Aoki P M, Litwin W, et al. Mariposa: a wide－area distributed database system[J]. The VLDB Journal, 1996(5):48－63.

(上接第 135 页)

[4] 都志辉. 高性能并行编程技术—MPI 并行程序设计[M]. 北京:清华大学出版社,2001:36－37.

[5] Phillippe M, Jean_Lue D. Data_Parallel Load Balancing Strategies[J]. Parallel Computing,1998,24:1665－1884.

[6] Gtama A, Gupta A, Karypis G, et al. Introduction to Parallel Computer[M]. 张　武等译. 北京:机械工业出版社,2005:81－95.

[7] 李冬梅,施海虎. 负载平衡调度问题的一般模型研究[J]. 计算机工程与应用,2007,43(8):121－125.

[8] 唐　丹,金　海,张永坤. 集群动态负载平衡系统的性能评价[J]. 计算机学报,2004,27(6):803－811.

[9] 卢克中,林晓辉. MPI 并行程序设计的负载平衡实现方法[J]. 微计算机信息,2007,23(5－3):226－237.

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

[11] 马艳琨,马胜甫,田俊峰,等. 一种用于 PC 存储集群的动态负载平衡策略[J]. 计算机工程与应用,2004(29):119－128.

[12] 廖湘科. 网络并行计算中的负载平衡[J]. 小型微型计算机系统,1995,16(9):32－36.

[13] 刘振英,方滨兴,胡铭曾,等. 一种有效的动态负载平衡方法[J]. 软件学报,2001,12(4):563－568.