基于MPI的并行程序设计

张翠莲,刘方爱,王亚楠
(山东师范大学 信息管理学院,山东 济南 250014)

摘 要:在介绍消息传递接口标准(MPI)和分析并行程序设计方法的基础上,提出了在并行程序设计中需要进行算法级分析和程序级测试,以此来对影响具体的并行程序执行效率的因素进行分析,并用实例验证了分析结果。最后对MPI的实现之———MPICH1.2.5版本的不足,提出了改进的方法。

关键词:消息传递;通信;MPI;并行程序
中图分类号:TP311.1 **文献标识码**:A **文章编号**:1673-629X(2006)08-0072-03

Parallel Program Design Based on MPI

ZHANG Cui-lian, LIU Fang-ai, WANG Ya-nan
(College of Information Management, Shandong Normal University, Jinan 250014, China)

Abstract: In this paper, MPI(message passsing interface) is introduced and parallel programming technic is analysed. Then put forward applying algorithm analysis and programming test to find out the factors affecting the specific program. At last, methods are proposed to improve the deficiency of the MPICH1.2.5.

Key words: message passing; communication; MPI; parallel program

0 引 言

计算机技术极大地促进了计算科学的发展,与此同时科学计算对计算速度也提出了更强大的需求,例如人类基因、全球气候准确预报、海洋环流循环、核爆炸模拟等等,没有万亿次以上的高性能计算机是无法解决的,这些问题使得计算机的性能亟待改善。但在实践中,由于受到物理器件极限速度和技术水平的限制,使得单处理机远远满足不了许多领域中的大规模计算课题对计算资源的需求。而多个处理器协同的求解问题极大地提高了计算速度,满足了大规模数据处理的需求。对并行处理的需求极大地促进了并行技术的发展,因此许多大规模并行计算机系统相继问世,如PVP(Parallel Vector Processors,并行向量处理机)、SMP(Symmetric Multiprocessors/Shared Memory Processors,共享内存多处理机)、MPP(MassivelyParallel Processors,大规模并行处理机)、DSM(Distributed Shared Memory,分布式共享存储多处理机)[1]等。但传统的并行系统的高成本性、专用性、系统规模的不可伸缩性等使其难以推广到普通的商业应用和科学计算中[2]。高性能集群系统因其性能价格比高、高可复用性、强可扩展性、用户编程方便等优点在科学研究中得到了广泛的应用。在2005年6月刚刚发布的全球高性能计算TOP500中,集群系统占到TOP500的60.8%。并行计算机系统的出现就需要对程序进行并行设计,这种需求使得各种不同的并行编程环境得到了很大的发展。现行高性能计算机系统中使用的并行编程环境主要有两种:PVM(Parallel Virtual Machine)和MPI(Message Passing Interface)[3]。PVM的开发始于1988年,由美国橡树岭国家试验室发起。目前很多人采用MPI作为并行开发环境。

收稿日期:2005-11-13

作者简介:张翠莲(1980-),女,山东泰安人,硕士研究生,研究方向为并行算法、图像恢复;刘方爱,教授,博导,研究方向为并行处理、并行计算模型和互联网络。

1 MPI简介

MPI是为开发基于消息传递模型的并行程序而制定的工业标准,其目的是为了提高并行程序的可移植性和易用性,用户必须显式地通过发送和接受消息来实现处理器之间的数据交换。为了开发一个高标准并具有可移植性的消息传递库,由厂商(如IBM, Intel等)、软件开发商(Parasoft, KAI等)、研究中心(ANL, GMP等)和大学(Endinburgh, Maryland等)共同于1992年成立了MPI论坛[4]。经过共同研究和使用,MPI论坛先后于1994年和1997年后提出了MPI-1和MPI-2[5]。MPI具有移植性好、功能强大、效率高等多种优点,而且有多种不同的免费、高效、实用的实现版本,几乎所有的并行计算机厂商都提供对它的支持,这是其它并行环境所无法比拟的。MPI只是一个并行编程语言标准,要编写基于MPI的并行程序,还必须借助MPI的某种具体实现。MPICH是最重要的一种MPI实现,是一个与MPI规范同步发展的版本。每当MPI推出新的版本时,就会有相应的MPICH的实现

版本,目前常用的版本是 mpich－1.2.6,在 Windows 下的一个常见版本是 mpich.nt.1.2.5,文中的所有实验均采用此版本作为实验环境。CHIMP 是 Edinburgh 开发的另一个免费 MPI 实现,是在 EPCC(Edinburgh Parallel Computing Centre)的支持下进行的。LAM(Local Area Multicomputer)是 Linux 平台下另一个免费的 MPI 实现。它由 Ohio 州立大学开发,主要用于异构的网格计算并行系统。

2　基于 MPI 的程序设计方法

并行程序设计可以通过多种方法实现,它们各有特点,并适合于不同的场合。最常用的是共享变量方法和消息传递方法。共享变量方法的主要特点是利用共享存储器中共享变量来实现处理机间的通信,它主要用在共享存储结构中。消息传递方法的主要特点是不同处理机间必须通过网络传送消息来相互通信,并达到任务间需要的同步操作,它主要应用于分布式存储结构中。MPI 是基于消息传递模型的,在这种模型中,把任务分成若干部分(进程),让每个处理器(节点)独自运行不同或者相同的部分(进程)。驻留在不同处理器(节点)上的进程可以通过网络传递消息来互相通信,从而达到并行计算的效果,减少计算的时间,其执行过程如图 1 所示。其中进程的分配、结果的收集均通过消息传递来完成。在 MPI 中,通过指定通信因子和组来对进程进行一种逻辑上的划分,通讯因子定义了进程组内或组间通讯的上下文。

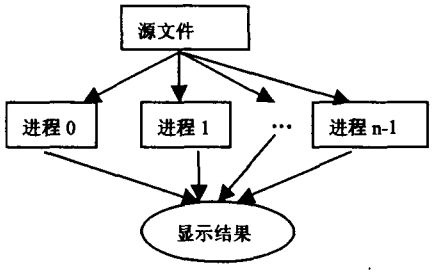

图 1　消息传递模型的执行过程

消息传递的多处理机编程可以通过下面的方法实现:

(1)设计全新的并行编程模型;

(2)扩展现有串行高级编程语言的语法和保留字来实现消息传递;

(3)使用现有的高级串行编程语言并提供消息传递的扩展链接库。

MPI 是运用方法(3)来实现的。采用 MPI 构建一个并行计算环境主要用到 6 个函数。这些函数的函数名为 MPI_Init, MPI_Comm_Size, MPI_Comm_Rank, MPI_Send, MPI_Recv, MPI_Finalize。MPI_Init 和 MPI_Finalize 实现 MPI 环境的初始化和结束。MPI_COMM_WORLD 通信因子是在 MPI 环境初始化过程中创建的。MPI_Comm_Size 获取缺省组(Group)的大小;MPI_Comm_Rank 获取本进程[调用进程]在缺省组中的逻辑号(从 0 开始);MPI_Send 和 MPI_Recv 实现消息的传递。一个基本的 MPI 程序的流程图如图 2 所示。

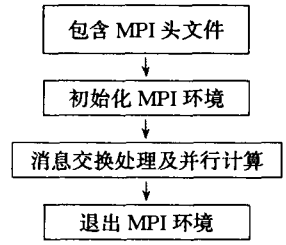

图 2　MPI 程序设计流程图

设计并行程序的目的是为了提高算法的效率,尽可能地减少时间复杂性。但并不是所有的算法都能在并行机上并行运算,而且能在并行机上并行运算的算法并不一定都能达到加速的目的,所以分析并行算法效率,对于并行程序设计具有重要的指导意义。在将一个并行算法写成并行程序之前,先进行算法级的分析,再在并行程序运行的过程中进行程序静态测试和程序动态测试,及时地修改调整程序,使其能更高效地执行。算法级的分析主要是对时间复杂度的分析。在消息传递系统中,整个的执行时间由两部分组成,即计算时间和通信时间。可用下式表示:

$$T_p = t_{comp} + t_{comm}, \text{而 } T_{comm} = t_{startup} + nt_{data}$$

其中 $t_{startup}$ 为启动时间,包含在源进程处将消息打包以及在目的进程处将消息解包所需要的时间;t_{data} 表示发送一个数据字所需的传送时间。程序级的调试主要是包含对并行程序任务的分配、数据的划分方法、消息的大小、负载的均衡等方面。对于同一个应用程序、不同的数据划分方法,在时间复杂度相同的情况下,运行时间可能是不同的;对于不同的应用程序,在其他条件相同的情况下,要传递的消息的大小对整个的通信时间有着很大的影响。对于集群而言,负载的均衡也是影响并行效率的关键因素之一。

3　并行程序测试

3.1　测试环境

硬件环境是由 3 台计算机组成的集群,配置均为 CPU:Intel PentiumIII,733MHz,内存 192MB;节点机操作系统为 Windows 2000 Professional 和 Windows XP;网络类型:100Mbps 以太网;消息传递系统:MPICH－1.2.5.nt;网络协议:TCP/IP;编程语言:Visual C++ +6.0。

确保集群上的每台节点机都安装 mpich.nt.1.2.5,并以相同的用户名和密码(要求密码不为空)进行注册,然后对主控计算机进行配置。并对 Visual C++ +进行相应的设置,使其能与 MPICH 整合,并行程序的执行利用了 mpich.nt.1.2.5 的图形化的应用程序 guiMPIrun。

3.2　测试及相关分析

测试实例之一使用并行方式计算 π,令函数 $f(x) = 4/(1+x^2)$,则 $\int_0^1 f(x)\mathrm{d}x = \pi$,求解思想是在微小间隔内用求和运算近似积分运算,这种方法本身具有很好的可并行性。设 n 值为将[0,1] 区间所分的间隔数,并行时间复杂度为 $O(n)$,但比串行程序的数量级要小,并且消息的

传递只涉及 n 以及各个处理机计算得到的部分和，通信开销比较小，适合并行化。程序中数据的划分依据是每个进程的进程号，可以将连续的数据块分配给某个进程，或者按照某种规律(例如所有分配给同一进程的数据为首项不同的等差数列)进行分配。程序运行过程中对数据划分方式的改变使得程序的运行效率及结果的精确性有所变化。实验结果表明后者的分配方法使得程序的效率高于前者。以下实验结果均采取第二种数据划分方法。表 1 给出了在不同数量的处理机下的实验结果，本程序的加速比接近最大加速比，并行效率较高、通信量较小，其中每个节点的计算能力是影响算法的主要因素。为加快算法的执行效率，可采用主频较高的处理器或者减少节点的工作负载。

表 1 第二种数据划分方法的实验结果 $n = 10000000$

处理机台数	测量值与实际值的误差(真实 π 值取前 25 位)	运行时间(s)	加速比
3	0.0000000000001155	0.556220	2.942
2	0.0000000000001918	0.831805	1.967
1	0.0000000000000622	1.636433	

测试实例之二是将一个文件中的若干个数据相加，保存数据的文件是一个记事本文件，存放在各个节点的硬盘上，处理机个数为 numprocs。设总共有 n 个数据(本程序中取 $n = 100000$)，程序代码如下所示。本程序中涉及两次消息传递操作，根进程将 n 个数据广播给所有进程，各进程将部分计算结果汇集给根进程，故总的通信时间为：

$$T_{commtotal} = (t_{startup} + nt_{data}) + (t_{startup} + t_{data})$$

总的计算时间包括各个进程并行计算部分和的时间以及一次根进程对部分和求和的时间：

$$t_{comptotal} = (n/\text{numprocs} + 1)t_{comp}$$

故 $T_p = 2t_{startup} + (n + 1)t_{data} + (n/\text{numprocs} + 1)t_{comp}$

从分析来看，本程序中通信时间在总的执行时间中占了主导地位，对通信延迟比较大的工作站网络来说，多机处理的效率未必优于单机。

```
void main(int argc,char * argv[])
{ double startwtime,endwtime;ifstream istrm;
int myid,numprocs;int data[MAXSIZE];
int i,x,low,high,myresult=0,result;
  MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&myid);
if(myid==0)
{istrm.open("f3.dat");
  for(int i=0;i<MAXSIZE;i++)
istrm>>data[i];
istrm.close();}
MPI_Bcast(data,MAXSIZE,MPI_INT,0,MPI_COMM_
WORLD);
startwtime=MPI_Wtime();
x=MAXSIZE/numprocs;low=myid*x;high=low+x;
  for(i=low;i<high;i++)myresult+=data[i];
printf("I got %d from %d\n",myresult,myid);
MPI_Reduce(&myresult,&result,1,MPI_INT,MPI_SUM,0,
MPI_COMM_WORLD);
if(myid==0){endwtime=MPI_Wtime();
printf("The sum is%d\n",result);
printf("wall clock time=%f\n",endwtime-startwtime);}
MPI_Finalize();
}
```

在相同的条件下，程序运行结果如表 2，从表中可以看到从运行时间和结果的精确性来说，单机均优于多机。本程序的通信量较大，为减少通信量，可采用多种方法：

①对每个节点机，仅传递其计算所需的数据；

②采用高速网络，减少消息延时；

③使用 SMP(Shared Memory Processing)机群，充分利用节点内固有的高速通信。

表 2 若干个数据求和运算结果

处理机台数	运行时间(s)	求和结果
3	0.013221	100002
2	0.019311	100001
1	0.002876	100000

4 结 论

MPI 提供了良好的并行程序接口，通过调用 MPI 的库程序，可以达到并行化程序的目的，但并行程序的设计相对于串行程序的设计来说要复杂得多，文中提出了影响并行程序设计的诸多因素，并通过实例来演示了可以通过算法级测试和程序级测试来及时调整具体的应用程序。文中使用的 MPICH 为并行程序的实现提供了并行环境，并且可以根据具体应用程序的不同来选择处理机，基本达到了测试并行程序的目的。但它是先进行单纯的后台计算，然后给用户输出一组时间数据，即求解同一个问题使用单个到多个处理器的时间值。用户必须比较时间值并额外计算加速比、并行效率，才可以对并行程序进行评价，这种方式，只是提供给用户几个呆板的数据，具体的分析还得靠用户去做，并且不能让用户深刻地体会到并行的理念和优越性。如果将衡量并行程序性能的指标集成到此软件中，并且让后台在进行并行计算的同时，前台能同步地以生动的图像(例如进度条)来演示各处理机计算的情况，则既可以向用户演示并行程序执行效率的情况，又可以生动地演示各处理机的使用情况。总之，并行程序设计涉及到诸多方面的技术，只依靠某一种方法或某一个软件工具是难以设计出高质量的程序的，需要用多种方法，研究多种软件工具有机地协调应用。

参考文献：

[1] 曾庆华，陈天麟. 可扩展并行计算机系统结构和发展现状

(下转第 76 页)

$a,e>$, $<b,b>$, $<b,c>$, $<b,e>$, $<c,c>$, $<c,e>$, $<d,d>$, $<d,e>$, $<e,e>$|,试判定 R 上的盖住集 COVR。

该例中所给的二元关系是以序对集合的形式给出的,两个元素之间的关系不明显,直接根据上述定义不好判定该偏序关系的盖住集 COVR。

2 偏序关系中"盖住"的等价定义及应用

通过对上述定义的分析,给出以下的等价定义。

定义' 对于任意 $<a,b>\in R$ 且 $a=b$,则 $<a,b>\in I_R$,令 $R_1=R-I_R$,则 $R_1-(R_1\bigcirc R_1)$ 为盖住集。

文中所给的等价定义是立足于集合的观点,应用此等价定义判定偏序关系的盖住集,不仅有条理,而且步骤清晰。可从以下几步求盖住集:

(1) 首先从 R 中找出所有 $a=b$ 的序对集合 I_R。

(2) 计算集合 R 和 I_R 的差 R_1,即 $R_1=R-I_R$。

(3) 将 R_1 与其自身复合,得集合 R_2。

(4) 计算集合 R_1 和 R_2 的差,则 R_1-R_2 为盖住集。

因此,实际用于偏序关系的盖住集的判定,不仅效率高,而且准确。

下面证明文中所给的等价定义与现有教材中的定义是等价的。将现有教材中的定义的前提条件分为前提1和前提2:

前提1:对于任意 $a,b\in A$,当 $<a,b>\in R$,$a\neq b$;

前提2:且没有其它元素 c 满足 $<a,c>\in R$ 和 $<c,b>\in R$。

对于前提1实际上排除了 R 中所有 $a=b$ 的序对,而在文中所给的等价定义中"对于任意 $<a,b>\in R$ 且 $a=b$,则 $<a,b>\in I_R$,令 $R_1=R-I_R$",也排除了 R 中所有 $a=b$ 的序对。即 $R_1=\{<a,b>|<a,b>\in R\wedge\neg(<a,b>\in I_R)\}$。

前提2实际上是在满足了前提1的基础上,再排除所有满足"存在其它元素 c 满足 $<a,c>\in R$ 和 $<c,b>\in R$" 的 $<a,b>$ 序对,R 中剩下的序对集合即为盖住集。

而在文中所给的等价定义中提出 $R_1-(R_1\bigcirc R_1)$ 为盖住集,R_1 满足了前提1的条件,$R_1\bigcirc R_1$ 运算的结果正是所有满足"存在其它元素 c 满足 $<a,c>\in R$ 和 $<c,b>\in R$" 的 $<a,b>$ 序对集合。而 $R_1-(R_1\bigcirc R_1)$ 排除了所有满足"存在其它元素 c 满足 $<a,c>\in R$ 和 $<c,b>\in R$" 的 $<a,b>$ 序对。

故两种定义在实质上是等价的。

应用文中所给的等价定义可以很快地判定前面例2的盖住集。

例3 设集合 $A=\{a,b,c,d,e\}$,R 是 A 上的偏序关系。

$R=\{<a,a>,<a,b>,<a,c>,<a,d>,<a,e>,<b,b>,<b,c>,<b,e>,<c,c>,<c,e>,<d,d>,<d,e>,<e,e>\}$,试判定 R 上的盖住集 COVR。

解:$I_R=\{<a,a>,<b,b>,<c,c>,<d,d>,<e,e>\}$

$R_1=R-I_R=\{<a,b>,<a,c>,<a,d>,<a,e>,<b,c>,<b,e>,<c,e>,<d,e>\}$

$R_1\bigcirc R_1=\{<a,c>,<a,e>,<b,e>\}$

$\mathrm{COVR}=R_1-(R_1\bigcirc R_1)=\{<a,b>,<a,d>,<b,c>,<c,e>,<d,e>\}$

3 结束语

文中通过对教材中偏序关系中"盖住"定义的深入分析,将定义"对于任意 $a,b\in A$,当 $<a,b>\in R$,$a\neq b$ 且没有其它元素 c 满足 $<a,c>\in R$ 和 $<c,b>\in R$,则称元素 b 盖住元素 a,并且记 $\mathrm{COVR}=\{<a,b>|a,b\in A;b$ 盖住 $a\}$" 改为"对于任意 $<a,b>\in R$ 且 $a=b$,则 $<a,b>\in I_R$,令 $R_1=R-I_R$,则 $R_1-(R_1\bigcirc R_1)$ 为盖住集",得出一种等价的定义形式。文中所给的等价定义是立足于集合的观点,应用此等价定义判定偏序关系的盖住集,不仅有条理,而且步骤清晰。

参考文献:

[1] 左孝凌,李为鉴,刘永才. 离散数学[M]. 上海:上海科学技术文献出版社,1982.

[2] Kolman B,Busby R C,Ross S C. Discrete Mathematical Structures(Fourth Edition)[M]. Beijing:Higher Education Press,2001.

[3] 何光明,王海艳. 离散数学学练考[M]. 北京:清华大学出版社,2004.

[4] 杨思春,周云霞. 二元关系反对称性的判定[J]. 微机发展,2004,14(3):93-94,

[5] 杨思春,王小林. 二元关系传递性研究[J]. 微机发展,2003,13(10):88-89,

(上接第74页)

[J]. 计算机科学,2003,30(9):158-161.

[2] 何敬鹏,周 容,俞建新. 一个小型集群系统的建立和初步应用[J]. 计算机应用研究,2004(1):227-230.

[3] Gropp W,Lusk E,Doss N,et al. A high-performance Portable Implementation of the MPI Message-passing Interface standard[J]. Parallel Computing,1996,22(6):789-828.

[4] 王萃寒,赵 晨,许小刚,等. 分布式并行计算环境:MPI[J]. 计算机科学,2003,30(1):25-26.

[5] 张 岳,陈 渝,孙亦嘉,等. MPI设计结构的分析和比较[J]. 计算机科学,2004,31(2):163-166.