MPI 并行计算在图像处理方面的应用*

吕 捷,张天序,张必银

(华中科技大学 图像识别与人工智能研究所,湖北 武汉 430074)

摘 要:介绍了 MPI 并行计算的基本概念和并行计算集群的实现,着重论述了图像处理算法的并行实现方法。列举了提高并行算法效率的一些措施。在并行计算集群上实现了灰度相关匹配算法和快速傅里叶变换算法,对它们的实验结果进行了分析和对比,通过实验数据说明了并行计算通信量和并行计算效率的关系,提出了并行计算在图像处理方面的适用范围。

关键词:消息传递接口; 并行计算; 灰度相关; 集群; 快速傅里叶变换

中图分类号:TP31 **文献标识码**:A **文章编号**:1007-2276(2004)05-0496-04

Applications of MPI parallel-computing on image processing*

LÜ Jie,ZHANG Tian-xu,ZHANG Bi-yin

(Institute for Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: The concept of parallel-computing based MPI and the implementation of parallel-computing cluster are introduced. The parallel way of image processing arithmetic is presented. Some methods of impoving parallel efficiency are put forward. A matching arithmetic based on intensity-based correlation and the FFT arithmetic are implemented and run on the parallel-computing cluster. The results are analyzed and compared. The expermental data show the relationship of computing efficiency and the quantities of communication. The feasibility scope of parallel computing in image processing is put forward.

Key words: Message passing interface; Parallel-computing; Intensity-based correlation; Cluster; FFT

0 引 言

随着科学技术的飞速发展,越来越多的大规模科学和工程计算问题对计算机的速度提出了非常高的要求。在图像处理方面,大规模的地形匹配、神经网络计算及其他计算量大的任务都需要计算机具有强大的计算性能。近年来,微处理器的性能不断提高,高速局域网的不断发展,可以利用相对廉价的微机通过高速局域网构建高性能的并行集群计算系统。与传统的超级计算机相比,并行集群计算系统具有较高的性价比和良好的可扩展性,可以满足不同规模的大型计算问题。

目前,MPI (Message Passing Interface) 是比较

收稿日期:2003-10-08; **修订日期**:2003-12-02

* **基金项目**:国家自然科学基金重点项目(60135020);航天科技创新基金项目(2001042)

作者简介:吕捷(1979-),男,湖北武汉人,硕士,主要从事并行集群计算方面的研究工作。

流行的并行计算开发环境之一。MPI 是一个并行计算消息传递接口标准,由 MPI 论坛(MPI Forum)推出,制定该标准的目的是提高并行程序的可移植性和开发效率。MPI 论坛是由欧美主要的并行计算机生产商、大学、政府实验室和工厂研究人员组成的一个非官方组织。MPI 论坛在 1994 年 6 月正式推出了 MPI 的第一个版本 MPI1.0,又于 1995 年 6 月推出了 MPI1.1,对原有的版本进行了修改、完善和补充。1997 年 7 月推出的 MPI2.0 版本中,又加入了远程存储访问、并行 I/O、动态进程管理等内容。MPI 现在已经成为产业界广泛支持的并行计算标准。

1　并行集群计算环境的建立

目前,国际上有许多对 MPI 标准的实现,其中包括 Argonne 国家实验室和 MSU 开发的 MPICH、Ohio State University 开发的 LAM、Edinburgh 开发的 CHIMP 等[1]。MPICH 是一个免费软件,它支持 LINUX、Windows NT/2000/XP 等操作系统,我们选用 MPICH.NT.1.2.5 作为并行计算开发环境。

2　并行图像处理算法的实现

图像处理的并行求解过程,一般分为以下几个步步骤:

(1) 对图像处理问题进行抽象,建立算法串行模型;

(2) 对算法串行模型进行分析,找出算法模型中需要并行处理的部分,确定算法并行实现方法,建立算法并行模型的描述;

(3) 用并行计算语言实现并行算法;

(4) 在并行集群计算系统上运行,调试并行算法。

2.1　建立算法并行模型[2]

在以上步骤中,建立算法并行模型至关重要。建立算法并行模型就是解决算法如何并行的问题。并行计算的目的是充分利用计算集群系统的资源,缩短计算时间,提高计算效率。并行计算的实现方法就是把一个大计算量的计算任务分解成多个子任务,分配给各个节点并行进行计算。由于计算上的内在关联性,计算节点之间必须进行数据交换,所以并行计算不可避免地会引入额外节点间的通信时间。额外的通信时间过大,将会降低并行计算的运行效率。如果额外通信时间大于算法并行计算节省的计算时间,那么并行集群计算系统运行速度就会低于一台计算机的运行速度,并行计算就会完全失去意义。

建立算法并行模型过程分三步:(1) 对算法模型进行分析,找出算法中运算量最大的几个模块,运算量最大的部分也就是需要并行计算的部分。(2) 对需要并行计算的模块进行分析,找出并行实现的方法。(3) 分析并行计算可行性,如果计算耗时远大于并行计算需要的额外通信耗时,那么并行计算是可行的、否则就是低效的[3]。

2.2　并行计算的编程实现

并行算法设计的基本思路是:在各计算节点启动计算进程,找出算法中可以并行的部分,把计算任务分块,每个进程完成一部分计算工作,进程之间通过 MPICH 提供的消息传递函数进行通信。由于图像是按行存放的,所以按行对图像进行分块计算是最方便的并行方法。把图像根据计算节点的个数平均分割成 N 个子图像,每个子图像在一个计算节点上运行。以下这段程序演示了对图像按行分块并行处理的方法:

```
//获取参与计算节点个数,保存到 size 变量
MPI_Comm_size(MPI_COMM_WORLD, &size);
//获取当前进程序号,保存到 rank 变量
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
//计算每一个节点要处理的行数,sum 为要计算的总行数
num = sum / size;
//计算当前进程要处理的起始、中止行
start_line = num * rank;
end_line = start_line + num;
if (rank == size-1) end_line = sum;
//开始计算
for (int i=start_line; i<end_line; i++){
…… //各节点进行行计算
}
```

2.3　提高并行算法效率的方法

并行计算的任务就是降低算法运行时间。如何在最短的时间内完成算法,提高运行效率可以从以下几个方面考虑:

(1) 减少通信量。并行计算进程之间肯定存在

数据通信，在图像处理中，通信量一般比较大。要去掉不必要的数据通信，尽量减少数据通信量和通信时间。

(2) 减少通信次数。每一次数据通信都要根据相应的协议对数据进行打包等操作，会消耗额外 CPU 时间。要减少通信次数，图像数据的传送应避免多次按行发送，过于分散数据应该打包一次传送。

(3) 利用通信与计算的重叠提高运行效率。MPI 提供了非阻塞的通信方式，进程在通信没有完成时可以继续进行计算。正确的使用通信与计算的重叠能提高算法的整体运行效率。

(4) 采用性能相同的计算机构建并行计算集群。采用相同的计算机目的是使各节点的性能一致，同时简化系统的管理，方便并行编程时计算任务的分割，提高计算速度。

灰度相关匹配算法[4]输入两幅图像：参考图和实时图。对实时图和参考图进行相关系数计算，得到相关面。取相关面的最大值作为匹配结果。

3 灰度相关匹配算法的并行计算

灰度相关匹配算法要进行多次图像相关度计算，计算量很大，特别是在大尺寸雷达图像与红外图像匹配时，单台微机计算时间难以令人满意。采用并行集群计算，可以大大降低算法运行时间。

3.1 算法的并行实现方法

计算灰度相关值是算法中计算量最大的部分，并行计算的思想就是把计算灰度相关值的任务分解到多个计算节点上并行执行。算法并行实现过程为：首先根据参与计算的节点数，把参考图按行分成 N 个子图像，由每个计算节点计算出相关面的局部最大值及最大值所对应的坐标，然后通过各个节点上的进程把结果传递给进程 0，进程 0 综合得出整幅参考图的相关面最大值，求出最后的匹配结果。

3.2 并行计算结果分析

评价并行计算性能需要引入两个参数：加速比 S_p 和并行效率 f_p。

$$S_p = T_p / T_s$$
$$f_p = S_p / p$$

式中 p 为节点个数；T_p 为在 p 个节点上并行计算耗时；T_s 为单机运算耗时。S_p 是衡量并行计算速度的参数，而 f_p 是衡量并行计算效率的参数，表示并行计算对计算机资源的利用率。

在 1～4 台计算机上并行运算灰度相关匹配算法，参考图尺寸为 256×360，实时图尺寸为 128×256。并行计算的结果如表 1 所示。

表 1 灰度相关匹配算法并行计算结果

Tab. 1 The result of parallel matching arithmetic based on intensity-based correlation

p	Total time/s	Communication time /s	Parallel computing time /s				S_p	f_p /%
			Node 0	Node 1	Node 2	Node 3		
1	12.60	0	12.60	—	—	—	—	—
2	7.89	～0	6.32	7.89	—	—	1.60	80
3	5.22	0.01	4.16	5.21	5.09	—	2.41	80
4	3.91	～0	3.12	3.91	3.71	3.22	3.22	81

在 4 个节点上并行计算灰度相关匹配算法的时空图如图 1 所示，从上到下分别是节点 0、1、2、3，横坐标为时间轴。最左边的灰色方块为并行同步，各节点从硬盘读出图像以后同时开始并行计算，最右边的

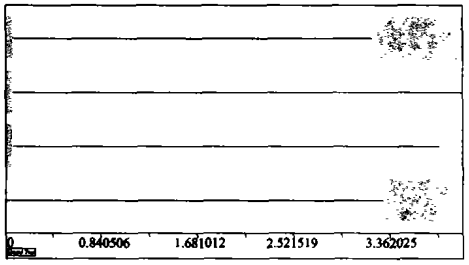

图 1 灰度相关匹配算法 4 节点并行计算时空图

Fig. 1 Time-space figure of four-node parallel matching arithmetic based on intensity correlation

灰色方块表示进程为结束并行同步而处于等待状态，当各个节点都计算完毕以后把结果传递给节点 0。两个方块中间就是并行计算相关面的部分。

结果分析：该算法并行计算的通信量很小，通信时间约为 0，随着计算节点增多，S_p 增大，说明并行计算的速度随节点增多不断提高。从图 1 可以看出，节点 1 计算性能最差，计算耗时最长，节点 0 必须等到节点 1 计算完毕后才能得出整幅图像匹配结果，节点 1 的性能偏低导致了其他计算节点的额外等待，所以并行计算集群应该尽量采用性能相同或相近的微机。

如果采用性能完全相同的微机，该算法的理论 f_p 应该接近 100%。

4　快速傅里叶变换的并行计算

并行计算适用于计算量大的算法。有些算法计算量并不大，但并行实现以后会引入大量的数据通信，耗费 CPU 资源和网络带宽。快速傅里叶变换(FFT)算法就是一个例子，对傅里叶变换算法进行了并行实现并对其并行效率进行了测试。

4.1　算法原理及并行实现

二维傅里叶变换就是对图像进行两次一维 FFT：首先对图像进行行变换，然后再对行变换的结果进行列变换。

并行实现快速傅里叶变换算法的思路就是把行变换和列变换的计算任务分配到各个节点并行完成。并行傅里叶变换的过程为：各节点进行行变换→各节点把行变换的结果发送给根进程→根进程对行变换结果进行转置运算→根进程把转置结果发送到所有计算节点→各计算节点再进行行变换→各节点把变换结果发送到根进程→根进程进行转置。

图 2 示出了并行计算过程。从上至下分别是节点 0(根节点)、1、2、3，水平的直线表示计算，带箭头的斜线表示数据的传送方向，深色方块表示发送或接收数据，浅色方块表示广播数据(根节点把数据传送到所有计算节点)。其中，根节点的 4 段水平直线分

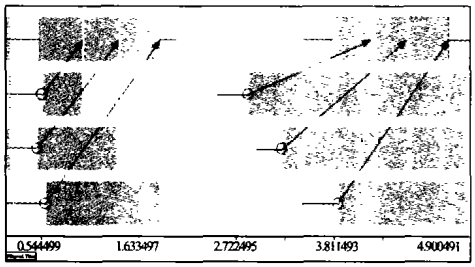

图 2　FFT 算法 4 节点运算时空图

Fig. 2　Time-space figure of four-node parallel FFT arithmetic

别表示行变换计算、转置计算、行变换计算、转置计算。其他节点只有行变换计算，没有转置计算。

4.2　并行计算结果分析

对一幅 1024×1024 的图像进行了傅里叶变换并行计算，在 1～4 节点运行并行算法的结果如表 2 所示。

表 2　快速傅里叶变换并行计算结果

Tab. 2　The results of parallel FFT arithmetic

p	Total time /s	FFT computing time /s	Transposing time /s	Quantities of communication /Byte	Communication time /s	S_p	f_p /%
1	2.537	2.214	0.323	—	—	—	—
2	2.939	1.112	0.324	24M	1.503	0.86	43
3	3.604	0.674	0.332	32M	2.598	0.70	23
4	4.975	0.566	0.314	36M	4.095	0.51	13

结果分析：从表 2 数据可以看出，并行计算加速比 S_p 小于 1，说明并行计算未能提高计算速度，随着计算节点增多，f_p 不断减小，4 节点并行计算时 f_4 只有 13%，效率很低。虽然并行计算后计算 FFT 所用时间明显减少了，但是并行计算引入了很大的数据通信量，引入的通信时间大于并行计算节省的计算时间，所以快速傅里叶变换算法是不适合并行计算的。

5　结　论

从以上两个实验的结果可以看出，节点之间的通信耗时是影响并行计算加速比和并行效率的主要原因。要尽量加大计算耗时与通信耗时的比值，该比值越大，并行效率越高，如果该比值小于 1，算法就不适合并行计算。

参考文献：

[1]　都志辉. 高性能计算之并行编程技术——MPI 并行程序设计[M]. 北京：清华大学出版社，2001.

[2]　Fadlallah G, Lavoie M. Parallel computing environments and methods[A]. International Conference on Parallel Computing in Electrical Engineering[C]. 2000, 8. 2-7.

[3]　Barry Wilkinson, Micheel Allen. Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers[M]. Prentile hall. 1999.

[4]　钟胜，桑农，张天序. 基于灰度相关的雷达与可见光景象匹配算法[J]. 红外与激光工程，1999，28(5)：22-25.