PC 机群上 JIAJIA 与 MPI 的比较*

胡明昌+, 史 岗, 胡伟武, 唐志敏, 张福新

(中国科学院 计算技术研究所,北京 100080)

Comparing JIAJIA with MPI on PC Cluster

HU Ming-Chang+, SHI Gang, HU Wei-Wu, TANG Zhi-Min, ZHANG Fu-Xin

(Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: 86-10-62559641, Fax: 86-10-62564342, E-mail: hmc@ict.ac.cn

http://www.ict.ac.cn/chpc/dsm

Received 2002-04-22; Accepted 2002-09-30

Hu MC, Shi G, Hu WW, Tang ZM, Zhang FX. Comparing JIAJIA with MPI on PC cluster. *Journal of Software*, 2003,14(7):1187~1194.

http://www.jos.org.cn/1000-9825/14/1187.htm

Abstract: JIAJIA and MPI (message passing interface) are compared in this paper. JIAJIA and MPI represent the programming model of share memory and message passing respectively. MPI handles message transmission manually and is hard for programming. JIAJIA maintains data coherence by lower layer and provides simple message passing APIs, so it is easy and flexible to program. It takes more time for JIAJIA to initialize applications than MPI. A rough empirical formula about speedup and machine size is presented in this paper. Experimental results show that both JIAJIA and MPI have good parallel performance on PC clusters and their performance difference of most testing applications is lower than 10 percent. Applications with fewer messages have better performance and the gap between JIAJIA and MPI is minor. While messages increase, parallel performance decreases and the gap between JIAJIA and MPI depends on the amount of messages they produce.

Key words: JIAJIA; MPI (message passing interface); shared virtual memory; message passing; speedup; parallel performance; message amount; PC cluster

摘 要: 对 JIAJIA 和 MPI (message passing interface)是进行了比较.JIAJIA 和 MPI 分别代表共享存储和消息传递的编程模式.MPI 显式进行数据传输,编程复杂;JIAJIA 由底层维护数据一致性,并附加提供简单的消息传递函数,编程容易、灵活.JIAJIA 分配共享内存时开销较大,初始化时间比 MPI 长.提出了一个关于并行加速比与进程数目之间关系的近似经验公式,推出 JIAJIA 和 MPI 性能差距随着进程数目的增多而增大的结论.测试结果表明,大部分应用程序的 JIAJIA 和 MPI 版本的并行性能差距不超过 10%.对于通信量很小的应用程序,其 JIAJIA 和 MPI 的性能差距较小,而通信量本身较大的应用程序,其 JIAJIA 和 MPI 的性能差距主要取决于运行时产生的实

* Supported by the National Natural Science Foundation of China under Grant Nos.60073018, 69896250, 69973046 (国家自然科学基金); the Youth Innovation Foundation of the Institute of Computing Technology, the Chinese Academy of Sciences, under Grant Nos.20016280-1, 20026180-7 (中国科学院计算技术研究所领域前沿青年基金)

第一作者简介: 胡明昌(1974－),男,浙江永嘉人,博士生,主要研究领域为网络并行计算.

际通信量.

关键词: JIAJIA;MPI (message passing interface);共享虚拟存储;消息传递;加速比;并行性能;通信量;PC 机群

中图法分类号: TP311 文献标识码: A

并行程序设计环境为用户提供了一些简单的并行编程模型,并高效地支持用户使用这些模型.典型的并行程序设计环境主要有消息传递系统(例如 PVM[1]和 MPI[2]等)和共享存储系统[3~5].MPI(message passing interface)是一个实用的、易移植的、高效的、灵活的消息传递接口标准,统一了现存各种消息传递系统核心库函数的描述,适用于编写并行应用程序.分布式共享存储系统和共享虚拟存储系统(软件分布式共享存储系统)都具有共享存储系统的易编程性,数据访问方便.由于在硬件上直接实现分布式共享存储代价高昂,而且可扩展性一般,近 10 多年来,共享虚拟存储得到广泛的研究,实现了若干软件 DSM(分布式共享存储)系统,如 TreadMarks[3]和 JIAJIA[4]等.中国科学院计算技术研究所开发的 JIAJIA 采用了类似于 NUMA 的结构,能够把多个机器的物理地址空间组成一个更大的共享虚拟地址空间,实现了一种基于锁的一致性协议.

单机计算能力的增强和操作系统的支持使机群在近几年发展迅速.廉价的机群具有很高的性能价格比.本文的测试结果表明,用 PC 机群进行并行计算也具有非常好的并行性能.高速网络(如千兆以太网、Myrinet 和 SCI 等)在机群中的广泛使用,大大缩小了机群与大规模并行处理机之间的性能差距.

在工作站机群(NOW)上进行消息传递和共享存储性能比较的研究较多,文献[6~8]的工作表明,在工作站上,共享虚拟存储具有与消息传递相当的性能,两者的并行性能相差很小.由于 PC 机群有相当大的消息处理开销,在这样的系统中进行消息封装形成共享虚拟存储环境往往不被人重视,认为其性能与消息传递有较大的差距.本文作了 JIAJIA 与 MPI 的比较,而且是在 PC 机群上进行的,回答了消息传递和共享虚拟存储在 PC 机群上性能的差距问题.由于 MPI 标准形成较晚,有些测试程序没有 MPI 的相应版本,本文移植了几个测试程序.本文第 1 节介绍我们所使用的测试环境和测试程序,第 2 节是详细的测试数据和性能比较分析,第 3 节是结论.

1 测试环境和测试程序

1.1 测试环境

本文的硬件环境是我们实验室的一个 PC 机群,软件环境是 RedHat Linux 6.2 操作系统.

我们采用的 PC 机群环境是用百兆位以太网交换机连接的 8 台 PC,每台 PC 有两个 PIII 700 处理器、1GB 内存、一个百兆位网卡,主板是 SuperMicro 公司的 370DLE.软件环境是 RedHat Linux 6.2 操作系统,测试时采用的内核版本是 2.4.3.

1.1.1 MPI

消息传递是广泛应用于并行机的编程模型.为了使一种消息传递系统能够被更多的人使用,能在更多的机器上运行,MPI 标准便应运而生.它吸收了现存的许多系统的最突出优点[2].MPI 提供了一个易移植的编程接口和一个可靠的通信接口,它允许避免内存到内存的拷贝,允许通信重叠,获得了良好的通信性能;它可以在异构系统中透明使用,即能在不同体系结构的处理器上运行;MPI 提供的接口与现存消息传递系统接口(如 PVM,NX,Express,p4 等)相差不大,但提供了更大的灵活性,能在更多的平台上运行.MPI 是一个标准,它没有规定底层必须如何实现,故给实现该标准的厂家带来了更大的灵活性,使 MPI 可扩展性更好.MPI 的一个很成功的实现是 MPICH.我们在 PC 机群上使用的是 MPICH-1.2.3.MPI 提供模块化的函数调用,函数种类和个数都很多,适用于各种场合.同时,对一般的应用程序来说,通常只用到其中的十几个最常用的库函数[9],程序本身比用 PVM 编写的应用程序要直观得多.

1.1.2 JIAJIA

JIAJIA 是一个软件分布式共享存储系统,采用一种基于锁的新型 Cache 一致性协议来实现域存储一致性模型[4].与传统的基于目录的协议相比,基于锁的协议使处理机通过访问附带在锁上的 write-notice(用来记录一个页面是否被修改过)来维护一致性,从而避免由目录引起的存储开销和系统复杂度.为了降低系统开销,JIAJIA

只支持一种存储一致性模型和一种写传播策略(写无效策略),同时采用多写(multiple-writer)协议来避免假共享.JIAJIA 系统的另一个特征是它能够把多个机器的存储器组织起来,形成一个更大的存储空间,其最大容量可达每个机器的局部存储器之和;而近期实现的系统,如 TreadMarks,CVM 和 Quarks,其中的共享地址空间受到一台机器内存空间的限制.由于采用了基于锁的有固定 Home 的一致性协议,JIAJIA 系统避免了 twin 和 diff 的保存、存储空间的释放和收集、局部地址和全局地址之间的转化以及矢量时间戳维护等系统开销,同时可以打开广播、预取、迁移 home、写矢量等优化选项,因此效率比较高[10].

1.1.3 JIAJIA 编程及与 MPI 的关联

JIAJIA 提供了类似于硬件实现的共享存储系统的编程接口.应用程序的各个进程间通过两种基本的原语进行同步:用于控制互斥访问的锁(lock)和用于全局同步的栅障(barrier)[4].函数 jia_lock(*i*)和 jia_unlock(*i*)用于实现一个临界区:如果某个进程已通过 jia_lock(*i*)获得了锁 *i*(*i* 是一定范围内的整数),则任何其他进程调用 jia_lock(*i*)都会导致等待,直至锁被释放.JIAJIA 遵循域一致性模型,即对同一组变量的互斥访问必须通过对同一个锁 *i* 的加锁和开锁操作来实现.函数 jia_barrier()将暂停当前进程,直至并行程序中的所有进程都到达了这个栅障,并完成域内所有的数据同步为止.MPI 也有同步函数 MPI_Barrier(),其语义比 jia_barrier()要弱,所完成的操作仅仅是等齐[9],其功能相当于 jia_wait().

在 JIAJIA 中,所有的共享存储区都必须通过函数 jia_alloc()进行分配,该函数的一些变形还允许程序员指定数据区在各个结点上的分布方式,如分块的大小、起始结点号等.对已分配的共享数据的存取通过引用 jia_alloc 返回的指针进行,用户无须了解相应的数据是在本地还是在其他处理机的内存中.在消息传递系统中,每个进程能直接访问的数据仅有本地数据.jia_alloc 的时间开销比 malloc 要大得多,因此,JIAJIA 初始化时间比 MPI 长.

JIAJIA 提供了全局共享的变量 jiahosts,用于指明参与并行计算的结点的数目,并为每个结点程序提供一个从 0 开始的序列号 jiapid,表示并行计算的不同进程号.MPI 可分别用函数 MPI_Comm_size()和 MPI_Comm_rank()获得功能相当的值.

为了提供编程方便和提高性能,JIAJIA 还提供了类似消息传递的函数 jia_send()和 jia_recv(),功能相当于 MPI 中的 MPI_Send()和 MPI_Recv(),JIAJIA 测试程序 Ocean 利用了函数 jia_send 和 jia_recv,极大地提高了并行性能.

JIAJIA 可以对程序执行过程进行优化,如进行预取、迁移 home 等,这可以通过一个简单的函数调用 jia_config 来完成.

JIAJIA 和 MPI 的初始化和结束分别通过函数 jia_init(),MPI_Init(),jia_exit()和 MPI_Finalize()来完成.

1.2 测试程序

本文用到了 7 个测试程序.这 7 个测试程序是来自 Stanford 大学 SPLASH[11]并行程序组的 LU,OCEAN 和 WATER,来自 NASA Ames 研究中心的 NAS[12]并行程序集中的 EP 和 3DFFT,来自 Rice 大学的 TSP 和 SOR[6],其中 MPI 版本的 EP 和 3DFFT 是直接从 NAS 的网站下载的,另外 5 个 MPI 测试程序是从相应的共享存储版本移植的.在将共享存储的应用程序移植到基于消息传递的 MPI 环境的过程中我们发现,编写基于消息传递的并行程序的工作量比编写基于共享存储的相同程序的工作量要大得多,消息传递需要确定发送或收取什么数据、何时发送或收取数据、向哪个进程发送或获取数据,其中最难的是确定向哪个进程发送或收取数据.

EP(embarrassingly parallel)程序产生高斯伪随机数,统计落在环内的数目,是一个典型的 Monte Carlo 应用——求积分.该程序只在最后交换数据——求和,通信计算比非常小.

SOR 测试程序采用红黑格的逐次超松驰迭代法解偏微分方程.SOR 的并行程序将红黑两个数组分成大小基本相同(最多只差一行)的长方块,由每个处理机负责计算一块数据.由于采用 5 点差分格式,只有涉及到带状区域的边缘行的计算时才会发生通信,其通信量很小.

TSP 程序通过分枝限界算法来求 *N* 个城市的旅行商问题的最优解.MPI 并行程序采用主从管理方式:主进程在自己的私有空间内管理所有主要的数据结构,并维护和更新最短路径;从进程进行遍历,从主进程处得到待分析的路段或向主进程更新最短路径.

WATER 是一个分子动力学模拟程序.Water 的并行算法将水分子数组分成相同大小的连续区段,每个处理机负责其中的一个区段.主要的处理机间的通信发生在力的计算过程中:若将分子数组看成是一个环,则每个处理机对它所负责的每个分子都要计算该分子与数组中紧随其后的 $n/2$ 个分子间的作用力,并依此更新记录中的相应数据.

OCEAN 应用程序研究起伏涨落的和边界的洋流在影响大规模海洋运动时的作用[13].该程序用一个类似于由于地球自转而引起的环流的离散化模型模拟了一个立方形的海洋盆地,空气中的风提供了牵引力,洋底的山墙和平地提供了摩擦力.洋流在牵引力和摩擦力的作用下运动,最后,洋流的涨落和流动达到一个动态的平衡.由于需要计算许多步,每一步需要解一组空间差分方程和椭圆方程,因此计算量非常大.

LU 程序用块 Gauss 消去算法将一个稠密矩阵分解为一个上三角阵和一个下三角阵的乘积.SPLASH 采用连续块分配的 LU 分解,连续块分配把那些在原来的数组中并不连续的矩阵块分配在连续的空间内,并将所有块按块列均匀分布到对它们进行操作的处理机内存中,相应的处理机成为该矩阵块的属主.块 LU 分解算法逐步进行,每一步分解一列块.每一步首先是当前对角块的属主进程将该对角块进行 LU 分解,将分解结果广播给所有其他进程;然后,其他进程根据收到的对角块分解该对角块所在列下边的其他块和所在行右边的其他块,并将行块和列块传送给其他要使用的对应进程;最后,所有进程更新右下角矩阵的所有块.

3DFFT 程序利用 FFT 算法求解三维偏微分方程.假定输入数组 A 大小为 $n1\times n2\times n3$,按行存放,3DFFT 首先执行每个 $n1\times n2$ 复杂矢量上 $n3$ 个点的一维 FFT,接着执行每个 $n1\times n3$ 复杂矢量上 $n2$ 个点的一维 FFT,然后,结果组转置成 $n3\times n2\times n1$ 的复杂数组 B,最后在每个 $n2\times n3$ 复杂矢量上进行 $n1$ 个点的一维 FFT.

2 测试数据和结果分析

我们在 PC 机群上测试了 7 个应用程序在 1 个、2 个、4 个、8 个和 16 个进程时的并行(或串行)执行时间,见表 1,得到了加速比与进程数的关系,如图 1 所示,每个应用程序的规模见表 2.图 1 的横轴是进程数,纵轴是并行加速比的对数(以 2 为底数),J 前缀代表 JIAJIA,M 前缀代表 MPI.从图 1 可以看出,随着进程数目的增加,应用程序的加速比增大.无论是 MPI 还是 JIAJIA 应用程序,其并行性能都很好.

Table 1 Parallel application runtime (s)

表 1 应用程序的执行时间 (秒)

Application	Type	Number of processes				
		1	2	4	8	16
EP	JIAJIA	24.23	12.13	6.07	3.11	1.57
	MPI	27.44	13.74	6.91	3.35	1.82
SOR	JIAJIA	98.11	50.37	25.98	13.60	8.62
	MPI	84.57	42.43	21.50	10.84	7.56
TSP	JIAJIA	99.69	52.92	28.08	15.53	10.83
	MPI	76.29	38.55	20.90	11.55	6.51
WATER	JIAJIA	155.07	81.84	42.98	24.58	15.55
	MPI	257.01	131.69	66.01	33.66	18.76
OCEAN	JIAJIA	62.43	36.39	21.17	13.36	*
	MPI	38.41	19.29	10.20	5.484	*
LU	JIAJIA	301.20	164.24	88.29	53.13	38.53
	MPI	300.42	169.15	88.03	50.71	30.19
3DFFT	JIAJIA	166.61	137.38	109.33	73.57	48.61
	MPI	117.60	110.64	81.93	49.79	35.79

The 16-process benchmark is not convergent at the given error and the running time is not obtained.

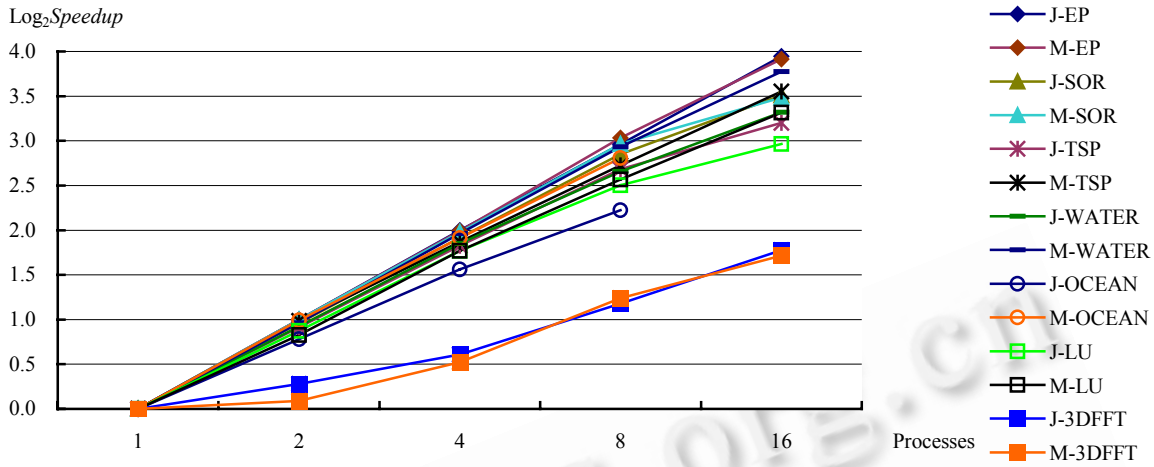

Fig.1 Logarithmic speedup of benchmarks

图 1 基准测试程序的对数加速比

Table 2 Problem size of parallel benchmarks and message sum of eight processes

表 2 并行基准测试程序的规模和 8 个进程时的消息总量

Benchmark	Problem size	Message count		Message sum (bytes)	
		JIAJIA	MPI	JIAJIA	MPI
EP	2^{25}	98	5	61 260	32
SOR	4096×2048, 100 iterations	10 025	2 800	23 640 984	45 825 200
TSP	20 cities	11 863	3 577	45 706 928	50 980
WATER	1728 mols, 10 iterations	13 619	1 447	40 394 276	19 029 104
OCEAN	1024×1024	39 876	11 929	45 337 492	20 594 472
LU	4096×4096	71 810	33 532	274 822 596	274 628 608
3DFFT	256×256×128, 16 iterations	159 000	136	845 595 388	536 871 040

2.1 测试结果总体分析

从表 1 可以看出,大部分应用程序单进程运行时间的 JIAJIA 版本稍大于 MPI 版本,但 Water 的 JIAJIA 版本执行明显比 MPI 快,这是由于像 Water 这类应用程序,对其主要数据结构的访问很不适合消息传递,为了实现消息传递,需要做大量的额外工作.

从图 1 可以看出,图中的大部分折线近似为直线,这意味着,对特定的应用程序来说,并行加速比的对数约正比于进程数的对数,设进程数为 n,对应的加速比为 $S(n)$,将折线直线化,则得到近似公式 $\log_2 S(n)=\beta\cdot\log_2 n$,即

$$S(n)=n^{\beta},\text{其中 }\beta<1\text{ 且 }n\text{ 较小.} \tag{1}$$

从图 1 还可以看出,从 8 个进程到 16 个进程的加速比增幅变缓.当一个应用程序并行化以后,其运行时间由两部分组成,一部分是计算时间,另一部分是通信时间.一般来说,计算时间总是能随着计算进程的增加而减小,如果应用程序并行度很高,那么计算时间随着进程的增加而呈双曲线形减小.并行程序的通信总量一般随着节点的增加而保持不变或增加,单个进程的平均通信量下降,但下降得没有像计算量那么明显,从而通信时间在总运行时间中的比重加大,造成通信瓶颈.如果能减小通信时间在总运行时间中的比例,那么能够明显提高并行效果.这里引用的并行测试程序并行度都很高,应用程序的通信情况和网络性能对加速比的影响明显,当从 8 个进程到 16 个进程以后,每个进程的网络带宽占有率从 1 网卡带宽/进程降到 0.5 网卡带宽/进程,因此其加速比增幅变缓,但如果应用程序通信量很小,则增幅影响不大.

假定一个应用程序的 JIAJIA 和 MPI 版本都满足近似式(1)中的加速比与进程数之间的关系,即

$$S_J(n)=S_J(2^k)=n^{\beta_J}=2^{k\cdot\beta_J}, \tag{2}$$

$$S_M(n)=S_M(2^k)=n^{\beta_M}=2^{k\cdot\beta_M}, \tag{3}$$

则 JIAJIA 版本与 MPI 版本的加速比之比值近似为

$$\eta(n)=\eta(2^k)=\frac{S_J(n)}{S_M(n)}=\frac{2^{k\cdot\beta_J}}{2^{k\cdot\beta_M}}=[2^{(\beta_J-\beta_M)}]^k, \tag{4}$$

从而加速比比值$\eta(2^k)$近似为一个关于 k 的指数函数.由于指数函数是一个单调函数,故式(4)单调增或单调减,因此,JIAJIA 和 MPI 的性能差异近似为随着进程数的增多而逐渐增大.JIAJIA 和 MPI 的真实性能差异如图 2 所示.图 2 的横轴为并行运行的进程数目,纵轴为 JIAJIA 应用程序与 MPI 应用程序的加速比比值.从图 2 可以看出,大部分应用程序的 JIAJIA 和 MPI 性能相差不大,在 8 个进程并行计算时,7 个应用程序的 JIAJIA 性能比 MPI 稍低,其中有 5 个应用程序性能差别不超过 10%,有一个应用程序(Water)相差 17%,还有一个应用程序(Ocean)相差较大,达到 33%.图 2 中大部分交点落在 0.9~1.1 的条带内,说明 JIAJIA 性能非常好,与 MPI 相当.随着进程的增多,JIAJIA 的加速比增幅略小于 MPI,并行性能与 MPI 的差距稍微变大,但从 8 个进程到 16 个进程时,部分应用程序的这个趋势出现了变化,例如,SOR,3DFFT 和 EP 的 JIAJIA 性能与 MPI 的差距变小,并超过 MPI.下面我们来解释这一现象.

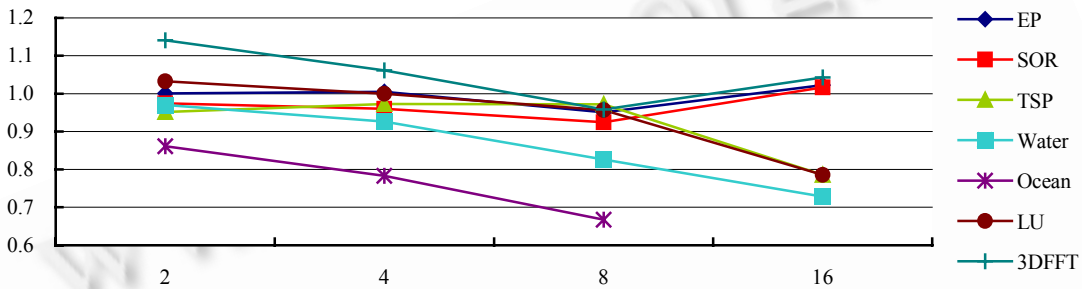

Fig.2 Speedup ratio of JIAJIA and MPI applications
图 2 JIAJIA 和 MPI 应用程序加速比比值

Intel P3 700 CPU 有 16KB 一级数据 cache(高速缓存),SuperMicro 主板有 256KB 二级 cache,每台主机有 1GB 内存,1 个速率为 100Mbps 的网卡,主机存储器带宽和通信带宽见表 3.从表中可以看出,访问主存的速度是网络速度的 17~31 倍.我们还测出:从内存读一个整数,假如 cache 命中,则所花时间(即从 cache 读一个整数的时间)约为 2.6ns;假如 cache 不命中,分为两种情况:(1) 访问大块连续的内存,平均每次最小访问时间(每次访问在一个 cache line 的内部)为 18ns;(2) 极端情况,每次访问都需要先把 cache line 中的数据写回内存,然后再从内存取一个 cache line 大小的数据,这样的操作重复多次,平均一次访问时间为 219ns.结点内 JIAJIA 和 MPI 发送一个小消息的时间分别为 65 000ns 和 62 000ns,而结点间 JIAJIA 和 MPI 发送一个小消息的时间分别为 105 000ns 和 101 000ns.由此可见,在我们配置的机群中,一次访存的系统开销和一次通信的系统开销相差至少 300 倍,因此需要减小系统开销在一次通信中所占的比重——减小消息个数,并尽量发送大消息.当 JIAJIA 应用程序运行 16 个进程时,由于结点内部采用 SMP 中的共享内存进行通信(结点内 JIAJIA 进程的 home 是共享的),结点内带宽(即内存带宽)远大于消息传递形式的结点内带宽,消息量也减少,因此,结点内通信比 MPI 强,而 SOR,3DFFT 在同一结点的两个进程之间有很大的通信量,采用 JIAJIA 共享 home 形式可以大大减小结点内部的通信时间,获得比 MPI 更高的并行性能.

Table 3 Problem size of parallel benchmarks and message sum of eight processes (MB/s)
表 3 PC 机群的存储器带宽和通信带宽 (MB/s)

Type	Memory bandwidth			Type	Communication bandwidth	
	First cache	Second cache	Main		Internal	External
Write	3 275	1 834	285	JIAJIA	59.9	9.11
Copy	2 893	1 374	191	MPI	38.1	11.0

2.2 具体应用在8个进程时的并行性能比较和分析

下面我们具体分析各个应用程序的 JIAJIA 和 MPI 性能与通信之间的关系.应用程序的 8 个进程加速比如图 3 所示.从图 3 可以看出,MPI 的并行性能比 JIAJIA 要好,但对大部分应用程序来说,两者的性能差距不超过 10%.8 个进程的消息个数和消息总大小见表 2,从表 2 可以看出,JIAJIA 的消息大小和消息个数都比 MPI 大,两

者通信量的差别与具体的应用程序、进程数量和具体执行有关.结合表 3 的数据,我们可以得出:对于通信量很小的应用程序,其并行性能非常好,并且 JIAJIA 和 MPI 的性能差距也小;而通信量本身较大的应用程序,其并行加速比相应降低,JIAJIA 和 MPI 的性能差距主要取决于运行时产生的实际通信量(包括消息个数和大小).

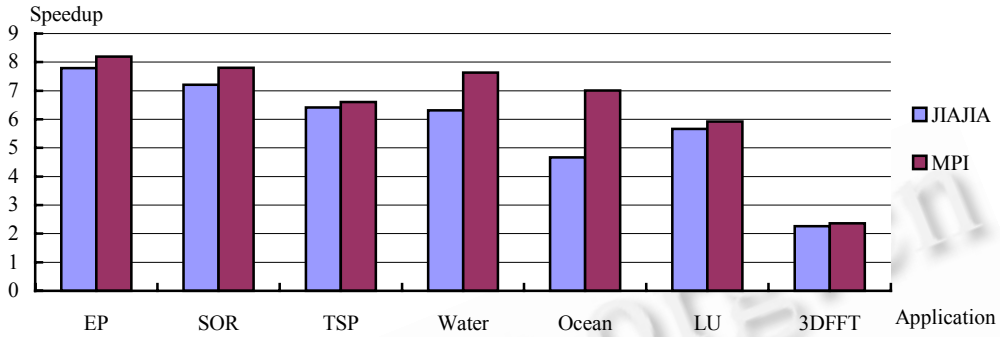

Fig.3 Eight-Process speedup of JIAJIA and MPI benchmarks

图 3 JIAJIA 和 MPI 应用程序的 8 个进程加速比

EP 的消息个数和消息大小在 7 个应用程序中最小,故其加速比最高,接近线性加速比.SOR 消息量较小,但其 JIAJIA 版本和 MPI 版本的消息个数相差很大,JIAJIA 是 MPI 的 3.58 倍,达到 737 个消息/(秒·8 个进程),而这个数目的消息量已经不可忽略,因此,JIAJIA 性能比 MPI 性能低 7.7%.在 TSP 程序中可以看到共享虚拟存储和消息传递的典型区别:共享虚拟存储方式共享粒度大,一般是 1 页,故数据不在本地命中时需要到远程取 1 页大小的数据,消息传递只取需要的数据,经常能减小消息大小,在本例中,JIAJIA 产生的消息总大小是 MPI 的 900 倍(接近 1 页与一个整数大小的比值);共享虚拟存储需要维护共享数据一致性,而消息传递没有一致性开销,这样,共享虚拟存储的消息个数比消息传递要多,在本例中,JIAJIA 版本产生的消息个数是 MPI 的 3.32 倍;尽管 TSP 的 JIAJIA 版本的消息大小比 MPI 大得多,但性能相差不大,仅有 3%.TSP 与 SOR 具有相近的消息量和计算时间,并且理论并行度极高,但并行性能比 SOR 差不少,引起这种差别的主要原因是 SOR 在两个同步点之间的计算量均匀,而 TSP 虽然理论并行度很高,但受到不确定因素的影响——分支限界算法对相同规模的不同输入具有不同的计算时间,故并行效果比 SOR 差.Water 的通信量不大,但 JIAJIA 版本的小消息个数是 MPI 的 9.4 倍,在快速以太网上小消息延迟非常大,至少 100μs 以上,而且不少是获取或释放锁的操作,与锁有关的操作的开销非常大,因此并行加速比要比 MPI 低 17%.Ocean 的通信特点是每个进程有不少方型矩阵,每个矩阵需要与它相临的矩阵交换边界(包括行和列)数据,为了减少消息个数和大小,在 JIAJIA 版本中采用了消息传递(利用 jia_send 和 jia_recv),将矩阵左(或右)边界所有列的数据(每列或每两列一个 double 数)合并到一个消息内发送,即使这样,JIAJIA 版本还有不少消息个数,是 MPI 版本的 3.34 倍,达到 2 985 个消息/(秒·8 个进程),消息个数便成为并行瓶颈,其并行性能比 MPI 差 33%.LU 的通信量大,并行性能有所下降,但还处在较合理的水平——大于 5.5,由于每次通信都是完整的页,是 JIAJIA 通信的理想情况,因此,JIAJIA 的性能仅比 MPI 低 4%.3DFFT 通信量极大,其并行性能很差,加速比<2.5,JIAJIA 通信量达到 2 162 个消息/(秒·8 个进程)和 11.6MB/(秒·8 个进程),MPI 版本的消息个数和总大小比 JIAJIA 小得多,但是 MPI 的归约操作(如 MPI_Reduce,MPI_Alltoall 和 MPI_Allreduce 等)效率非常低,其并行性能比 JIAJIA 仅高 4%.

3 结 论

JIAJIA 和 MPI 分别代表共享存储和消息传递的编程模式.MPI 手工进行数据传输,必须完全了解并行程序运行时数据流动的时刻和方向,编程复杂.JIAJIA 由底层维护数据一致性,编程方便,可移植性好,大大降低了串行程序并行化的工作量,但同时增加了并行程序的消息量.JIAJIA 还提供简单的消息传递函数,增加了编程的灵活性,克服了通信粒度大的缺点,提高了并行性能.

JIAJIA 能充分利用多个机器的物理内存来组成一个更大的共享虚拟空间,但分配共享内存时开销较大,因

此,JIAJIA 初始化时间比 MPI 长.

对于理论并行度非常高和消息量较少的并行程序来说,其 JIAJIA 和 MPI 版本在进程数较少的情况下都满足近似公式(1),即并行加速比近似为进程数的幂函数.由近似公式可以推出 JIAJIA 和 MPI 性能差距随着进程数目的增多而增大的结论.

测试结果表明,大部分应用程序的 JIAJIA 和 MPI 版本的并行性能差距不超过 10%.对于通信量很小的应用程序,其并行性能非常好,并且 JIAJIA 和 MPI 的性能差距也小,而对于通信量本身较大的应用程序,其并行加速比相应降低.JIAJIA 和 MPI 的性能差距主要取决于运行时产生的实际通信量(包括消息个数和大小),与数据组织和编程方式有关.

本文用到的测试程序都是国际上常用的并行基准程序,其并行性能一般随着规模的增大而增强.本文没有用到气象、油藏、生物方面的测试程序,这需要将来做更为细致的研究.

References:

[1] Sunderam VS. PVM: A framework for parallel distributed computing. Concurrency: Practice and Experience, 1990,2(4):315~339.

[2] Snir M, Otto S, Huss-Lederman S, Walker D, Dongarra J. MPI: The Complete Reference. London: MIT Press, 1996.

[3] Hu WW, Shi WS, Tang ZM. JIAJIA: A software DSM system based on a new cache coherence protocol. In: Sloot PMA, Bubak M, Hoekstra AG, Hertzberger B, eds. Proceedings of the HPCN Europe 1999. LNCS 1593, Amsterdam, 1999. 463~472.

[4] Amza C, Cox AL, Dwarkadas S, Keleher P, Lu H, Rajamony R, Yu W, Zwaenepoel W. TreadMarks: Shared memory computing on networks of workstations. IEEE Computer, 1996,29(2):18~28.

[5] Hu WW. Shared Memory Architecture. Beijing: Higher Education Press, 2001 (in Chinese).

[6] Lu HH, Dwarkadas S, Cox AL, Zwaenepoel W. Message passing versus distributed shared memory on networks of workstations. In: Proceedings of the 1995 ACM/IEEE Supercomputing Conference. San Diego: ACM/IEEE Computer Society, 1995.

[7] Lu HH, Dwarkadas S, Cox AL, Zwaenepoel W. Quantifying the performance differences between PVM and TreadMarks. Journal of Parallel and Distributed Computation, 1997,43(2):65~78.

[8] Tang ZM, Shi WS, Hu WW. Message-Passing versus shared-memory on dawning 1000A. Chinese Journal of Computers, 2000, 23(2):134~140 (in Chinese with English abstract).

[9] Al-Tawil K, Moritz CA. Performance modeling and evaluation of MPI. Journal of Parallel and Distributed Computing, 2001,61(2): 202~223.

[10] Hu WW, Shi WS, Tang ZM. Reducing system overheads in home-based software DSMs. In: Proceedings of the 13th International Parallel Processing Symposium and the 10th Symposium on Parallel and Distributed Processing. IEEE Computer Society, 2000. 167~173.

[11] Woo SC, Ohara M, Torrie E, Singh JP, Gupta A. The SPLASH-2 programs: Characterization and methodological considerations. In: Proceedings of the 22nd Annual International Symposium on Computer Architecture. Portofino: ACM/IEEE, 1995. 24~36.

[12] Bailey D, Harris T, Saphir W, van der Wijngaart R, Woo A, Yarrow M. The NAS parallel benchmarks 2.0. Technical Report, NAS-95-020, 1995.

[13] Singh JP, Hennessy JL. Finding and exploiting parallelism in an ocean simulation program: Experience, results and implications. Journal of Parallel and Distributed Computing, 1992,15(1):27~48.

附中文参考文献:

[5] 胡伟武.共享存储系统结构.北京:高等教育出版社,2001.

[8] 唐志敏,施巍松,胡伟武.曙光 1000A 上共享存储与消息传递的比较.计算机学报,2000,23(2):134~140.