PC 机群上共享存储与消息传递的比较*

章隆兵[1+], 吴少刚[2], 蔡 飞[1], 胡伟武[1]

[1](中国科学院 计算技术研究所,北京 100080)

[2](石油大学(华东) 计算机与通信工程学院,山东 东营 257061)

Shared-Memory Versus Message-Passing on PC Cluster

ZHANG Long-Bing[1+], WU Shao-Gang[2], CAI Fei[1], HU Wei-Wu[1]

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

[2](College of Computer and Communication Engineering, University of Petroleum (East China), Dongying 257061, China)

+ Corresponding author: Phn: +86-10-62559641, Fax: +86-10-62564342, E-mail: lbzhang@ict.ac.cn, http://www.ict.ac.cn

Received 2003-09-08; Accepted 2004-03-01

Zhang LB, Wu SG, Cai F, Hu WW. Shared-Memory versus message-passing on PC cluster. *Journal of Software*, 2004,15(6):842~849.

http://www.jos.org.cn/1000-9825/15/842.htm

Abstract: Two parallel programming models of shared-memory and message-passing are widely adopted. The programmability of message-passing is poor, while that of shared-memory is good. The OpenMP Application Programming Interface is an emerging standard for shared-memory. OpenMP on cluster supplies an OpenMP computing environment on cluster of workstations or PCs, which combines the friendly programmability of shared-memory with the fine scalability of cluster. Taking 7 well-known parallel applications on a cluster of PCs, this paper compares the performance of OpenMP/JIAJIA, an OpenMP system on cluster, with that of MPI, a typical message passing system. Experimental results show that the performance of OpenMP is averagely equal to 81% of MPI for the 7 applications running on 8-nodes, but the former is easier to use than the latter.

Key words: OpenMP; message passing interface; PC cluster; parallel programming model; shared-memory; software distributed shared memory system

摘 要: 共享存储和消息传递是目前两种主流的并行编程模型.一般认为,消息传递的可编程性不及共享存储友好.OpenMP 是目前共享存储编程的实际工业标准.机群 OpenMP 系统在机群上提供了 OpenMP 编程环境,具有易编程和可扩展的特点,但是其性能如何一直是关注的热点.以机群 OpenMP 系统 OpenMP/JIAJIA 和典型的消息传递系

* Supported by the National Natural Science Foundation of China under Grant No.60303016 (国家自然科学基金); the Foundation for the Author of National Excellent Doctoral Dissertation of the Chinese Academy of Sciences (中国科学院全国首届优秀博士学位论文作者基金); the Youth Innovation Foundation of the Institute of Computing Technology, the Chinese Academy of Sciences, under Grant No.20026180-7 (中国科学院计算技术研究所领域前沿青年基金)

作者简介: **章隆兵**(1974－),男,安徽绩溪人,博士,主要研究领域为计算机系统结构,并行计算,微处理器设计;**吴少刚**(1973－),男,博士,讲师,主要研究领域为计算机系统结构,机群计算,虚拟共享存储技术;**蔡飞**(1979－),男,博士生,主要研究领域为计算机系统结构,微处理器设计;**胡伟武**(1968－),男,博士,研究员,博士生导师,主要研究领域为高性能计算机体系结构,并行处理,VLSI 设计.

统 MPI(message passing interface)为例,在分析两种并行编程环境各自特点的基础上,在 PC 机群上采用 7 个应用比较了二者的性能.实验结果表明,对于所测试的 7 个应用而言,OpenMP 版本在 8 个处理机运行时的平均性能为 MPI 版本的 81%,但是采用 OpenMP 编程却比 MPI 简单很多.

关键词: OpenMP;消息传递;PC 机群;并行编程模型;共享存储;软件分布式共享存储系统

中图法分类号: TP311 **文献标识码**: A

随着高速网络和微处理器技术的发展,机群获得了很好的性能.由于性价比高和可扩展性好的特点,机群正逐渐成为主流的并行计算平台.消息传递(如 MPI(message passing interface)[1]、PVM)和共享存储(如 OpenMP[2])是两种典型的并行编程模型.通常认为,共享存储的可编程性比消息传递要好得多.由于机群是一种典型的分布式存储系统,采用消息传递进行编程是很自然且有效的,因此消息传递系统是目前机群上主流的并行编程环境.然而,消息传递难于编程,而共享存储易于编程,这使得在机群上实现共享存储编程环境很有吸引力.目前,OpenMP 是支持共享存储并行编程的工业标准,而机群 OpenMP 系统在机群上提供了 OpenMP 计算环境,它结合了机群的可扩展性和 OpenMP 的易编程性,引起了广泛的研究[3~5].

机群 OpenMP 系统主要利用软件 DSM(distributed shared memory)系统在机群上构造的虚拟共享存储抽象,将 OpenMP 程序转换成等价的软件 DSM 程序,进而在机群上运行.由于机群 OpenMP 系统是以软件 DSM 系统为基础的,所以机群上软件 DSM 系统的性能对机群 OpenMP 系统的性能有很大影响.在机群上进行软件 DSM 系统与消息传递系统的性能比较的工作较多[6~9].文献[8]中的研究表明:对于所测试的大部分应用而言,在 PC 机群上软件 DSM 系统的性能与 MPI 相当.对于机群 OpenMP 系统而言,由于程序语义的转换开销,其性能可能会比软件 DSM 系统要差一些.然而与消息传递系统相比,机群 OpenMP 系统在性能上究竟相差多少,是否在合理范围内,却又不失编程友好性呢?目前,在 PC 机群上进行 OpenMP 与消息传递系统性能比较的工作还未见到.

本文的贡献在于利用我们自己设计的机群 OpenMP 系统 OpenMP/JIAJIA,在一个通用的 PC 机群平台上与 MPI 系统作了详细的性能比较,直接回答了上述性能差距问题.本文的研究结果表明,所测试的 7 个应用的 OpenMP 版本在 8 个处理机运行时的平均性能达到 MPI 版本的 81%.本文第 1 节简要介绍 OpenMP 标准、机群 OpenMP 系统 OpenMP/JIAJIA 以及 MPI 标准.第 2 节介绍用于性能比较的应用程序.第 3 节是测试数据和性能分析.第 4 节是结论.为了描述方便,本文中有关 OpenMP 的制导和子句都采用斜体标示.

1 机群 OpenMP 系统与 MPI

1.1 OpenMP

OpenMP 标准通过定义编译制导、库例程和环境变量规范,给程序员提供了支持 Fortran、C/C++语言的一组功能强大的高层并行结构和一个支持增量并行的共享存储程序设计模型,能满足很大范围的应用需求.目前,OpenMP ARB 发布的分别支持 Fortran、C/C++的最新说明规范是 2.0 的版本[2].OpenMP 允许用户创建、管理可移植的并行程序,其编译制导主要包括 3 种类型:并行及工作共享制导、数据环境制导和同步制导.OpenMP 程序的并行块主要由 *parallel* 制导来描述,并行块以 SPMD 方式在多线程上运行.工作共享制导用于将并行区域中的任务划分成子任务在多个线程上执行,包括 *for*,*sections* 和 *single* 这 3 种模式.工作共享制导可以和某个并行块绑定在一起使用,编程更加简洁.数据环境制导用于在并行执行时控制数据的属性,主要包括 *threadprivate* 制导和一些描述数据区域属性的子句,如 *private*,*shared*,*default*,*firstprivate*,*lastprivate*,*reduction* 和 *copyin* 子句.同步制导主要包括 *master*,*barrier*,*critical*,*atomic*,*flush* 和 *ordered*.另外,OpenMP 的库例程主要提供一些获取线程标识和设置锁变量的接口函数,和环境变量一起便于对程序运行时行为进行控制.

OpenMP 采用 fork/join 并行执行模式:OpenMP 程序首先由 Master 线程执行,直到碰到第 1 个并行结构(由 *parallel* 制导构成),则由 Master 线程创建(fork 操作)一组线程,且 Master 线程成为线程组的主线程.除了工作共享结构外,每个线程都执行并行动态扩展域中的代码.而工作共享结构表明任务被划分成子任务,线程组中的每个线程分别执行对应的子任务,所有线程在工作共享结构结束处需要隐式同步,而且在并行结构的出口处执行

合并操作(join 操作).并行结构执行完后,Master 线程继续执行.程序中可以说明多个并行结构,所以程序在执行时创建和合并多次.OpenMP 标准只规定用户直接并行的语义,它的实现本身并不检查依赖、死锁等导致程序错误的问题,完全由用户保证采用 OpenMP 结构的程序正确性.

1.2 OpenMP/JIAJIA

OpenMP/JIAJIA 是我们自己开发的一个基于软件 DSM 系统 JIAJIA[10]的机群 OpenMP 系统.OpenMP/JIAJIA 利用 JIAJIA 在机群上提供的共享存储界面,将 OpenMP 程序映射成等价的 JIAJIA 程序在机群上运行.OpenMP/JIAJIA 主要包括编译处理系统前端和支持 fork/join 模式的 JIAJIA 运行库后端.系统框架如图 1 所示.

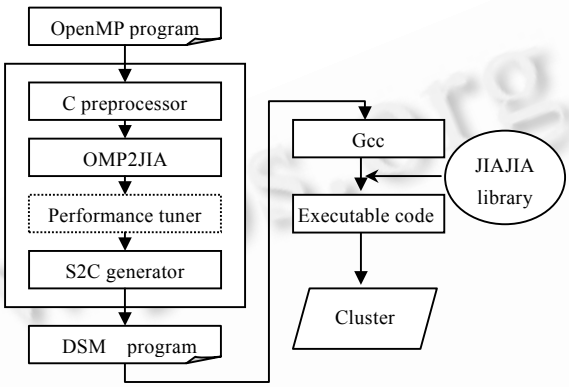

Fig.1 Framework of OpenMP/JIAJIA system

图 1 OpenMP/JIAJIA 系统框架

图 1 中,编译处理系统由 C 预处理器、OMP2JIA 编译器、性能优化器以及 S2C 源码生成器组成,其中 C 预处理器和 S2C 生成器直接取自 SUIF 工具集[11],OMP2JIA 编译器由我们自己基于 SUIF 开发,性能优化器是计划采用编译分析技术来提高性能的模块,目前没有实现.后端运行库由基于 SPMD 模式的 JIAJIA 改造而成.整个系统处理流程包括由编译系统前端处理 OpenMP 源程序,生成等价的软件 DSM 源程序,然后通过标准编译器(如 GNU Gcc)链接 JIAJIA 运行库,生成可执行程序在机群上运行.

由于软件 DSM 系统 JIAJIA 在机群上构造的是一个类 NUMA 的虚拟共享存储系统,为了在 NUMA 系统中获得高性能,需要对 OpenMP 标准进行扩展.与硬件 CC-NUMA 机器 SGI Origin 2000 类似,OpenMP/JIAJIA 中提供了合适的数据分布制导扩展 *distribute*.*distribute* 制导支持 Block 和 Cyclic 两种数据分布方式,程序员利用 *distribute* 数据分布制导控制数据在处理机间的分布.另外,OpenMP/JIAJIA 还提供了基于局部性优先的静态和动态循环调度模式.结合数据分布制导和合适的调度模式,可以充分实现拥有者计算,从而提高程序性能.

1.3 MPI

消息传递是一种广泛应用的并行编程模型.为了使消息传递系统能被更多的人使用,能在更多的机器上运行,MPI 标准便应运而生.在吸收了现存许多系统的最突出优点的基础上,学术界和工业界的研究人员共同设计并制订了该标准.MPI 标准定义了用 C 和 Fortran 编写消息传递应用程序所用到的核心库例程的语法和语义,具有很多特点.首先,MPI 提供了一个易移植的编程接口和一个可靠的通信接口,允许避免内存到内存的拷贝,允许通信重叠,具有良好的通信性能;其次,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;再者,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,却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,它没有规定具体的实现细节,这给实现该标准的厂家带来了很大的灵活性,使 MPI 可扩展性更好.MPICH 是一个成功的 MPI 实现,我们在机群上使用的是 MPICH-1.2.3 版本.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 10 几个最常用的库函数,程序本身比用 PVM 编写的程序要直观得多.

2 应用程序简介

我们使用了一些被广泛采用的基准程序,包括 NPB[12]中的蒙特卡罗模拟程序 EP,SPLASH2[13]中的水分子模拟程序 WATER,SPEC OMPL2001[14]中的浅水模拟程序 SWIM,Rice 大学 OpenMP-Now 项目[3]提供的计算多个向量正交基程序 GS,逐次超松弛迭代程序 SOR,Omni 项目[4]提供的拉普拉斯方程求解程序 LAP,以及我们自己编写的非分块 LU 分解程序.MPI 版本的 EP 是我们直接从 NAS 的网站上下载的,其他 6 个程序的 MPI 版本是从相应的共享存储程序移植过来的.在程序的 OpenMP 版本中,我们采用了针对机群 OpenMP 系统扩展的 *distribute* 数据分布制导来控制数据分布,并且使用了基于局部性优先的静态循环调度模式.在所有程序中,除了 EP 的 MPI 版本是 Fortran 程序,其余的都是 C 程序.

EP(embarrassingly parallel)程序产生高斯伪随机数,统计落在环内的数目,是一个典型的蒙特卡罗应用——求积分.该程序只在最后求和时交换数据,计算通信比非常大.

WATER 是一个水分子动力学模拟程序.WATER 的主要数据结构是一个由记录构成的一维数组,每个数组元素记录了一个分子的特性参数,包括分子的质心、受力、位移和 6 个方向的导数等.在每一个时间步都要计算分子和分子间的势.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.主要的处理机间通信发生在力的计算过程中:若将分子数组看成一个环,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.

SOR 采用红黑格的逐次超松弛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.在每个迭代中,先根据黑数组的值来计算红数组,然后利用红数组来计算黑数组.由于采用 5 点差分格式,只有涉及到带状边缘行的计算才会发生通信.

LAP 采用雅可比迭代法求解二维拉普拉斯方程.LAP 的并行程序将新旧两个数组按行均匀划分成基本相同(最多只差一行)的长方块,由每个处理机负责计算一块数据.在每次迭代中,先计算旧数组中每个元素的相邻 4 个元素的平均值,存放在新数组的对应元素中,然后计算新旧数组的迭代误差,最后并行更新旧数组.当计算涉及到长方块边缘时需要进行通信.

SWIM 是一个天气预报建模程序,它使用有限差分方法求解浅水方程组(shallow water equations).SWIM 的数据结构主要是 14 个二维数组.SWIM 的并行程序将所有数组按行均匀分布到处理机上,每个处理机负责计算分配给自己的行.SWIM 程序结构包含一个初始化阶段和一个迭代模拟阶段.SWIM 在计算自己拥有的数据边缘或者需要和别的处理机相互交换数据时进行通信.

LU 分解将一个稠密矩阵分解成上三角阵 U 和下三角阵 L.该程序没有采用块分解算法,而是普通的基于行主元的分解.LU 的数据结构是一个二维数组.在每次迭代中,当对角线元素所在行处理完以后,就利用该行内容并行更新当前元素右下角矩阵.考虑到负载平衡问题,LU 程序将数组按行以 Cyclic 方式分配给处理机,每个处理机负责计算分配给自己的行.当处理机要取对角线元素所在行来更新自己拥有的行时就要进行通信.

GS 是一个计算 M 个 N 维向量的正交基的应用.GS 的数据结构是一个二维数组.GS 的并行程序在第 i 次迭代时,首先规格化第 i 个向量,随后把所有的第 j 个向量($j>i$)与第 i 个向量正交化.与 LU 类似,GS 程序将数组按行以 Cyclic 方式分配给处理机,每个处理机负责计算分配给自己的行.当处理机要取规格化后的向量来更新自己拥有的向量时要进行通信.

3 测试数据和结果分析

本文的硬件测试环境是 8 个结点的一个 PC 机群,采用百兆位交换式快速以太网互联,每个结点有 2 个 PIII 700 处理器、1GB 内存以及 SuperMicro 公司的 370 DLE 主板.软件环境是 RedHat Linux 7.3 操作系统,内核版本为 2.4.18.采用的 C 编译器为 Gcc 2.96,Fortran 编译器为 g77 2.96.机群 OpenMP 系统为我们自己的 OpenMP/JIAJIA,MPI 的版本为 MPICH-1.2.3.

表 1 为所测试应用的规模及其串行版本的执行时间.一般情况下,应用的串行版本执行时间应该比其并行版本的单机执行时间短,这主要是因为并行版本的单机执行时间中包含了并行化所带来的开销.然而,由于应用

的串行版本与其并行版本的程序结构不同,由编译器生成的可执行代码的效率不同,故应用串行版本的执行时间与其并行版本的单机执行时间与具体程序的相关,并不一定很有规律.

Table 1 Problem size and sequential-version execution time of applications

表 1 应用规模及串行版本执行时间

Application	Problem size	Sequential-Version execution time (s)
EP	2^{25}	43.96
WATER	1728 mols, 10 iterations	241.54
SOR	4096×4096, 100 iterations	123.83
LAP	4096×4096, 100 iterations	289.45
SWIM	1335×1335, 120 iterations	380.96
LU	2048×2048	286.02
GS	2048×2048	341.91

3.1 测试结果总体分析

表 2 是所有应用的 OpenMP 版本与 MPI 版本的单机和多机执行时间.图 2 为应用的 OpenMP 版本与 MPI 版本的多机并行加速比(O 前缀表示 OpenMP 版本,M 前缀表示 MPI 版本).图 3 为应用的 OpenMP 版本与 MPI 版本的多机加速比比值.为了公正比较,本文以应用的并行版本多机执行时间与串行版本执行时间的比值作为多机加速比.

Table 2 Execution time of applications written in OpenMP and MPI

表 2 应用的 OpenMP 版本与 MPI 版本的执行时间

Application	Version	Execution time (s)			
		1 node	2 nodes	4 nodes	8 nodes
EP	OMP	44.08	22.06	11.02	5.55
	MPI	39.59	19.86	9.93	4.98
WATER	OMP	243.67	130.94	69.96	38.75
	MPI	236.59	124.67	68.13	35.46
SOR	OMP	130.0	70.94	36.32	20.32
	MPI	141.94	71.24	36.73	19.35
LAP	OMP	311.29	160.52	81.64	44.33
	MPI	290.6	144.99	78.7	42.0
SWIM	OMP	444.95	237.79	138.27	83.75
	MPI	459.8	236.33	122.56	63.67
LU	OMP	288.46	139.7	82.51	70.08
	MPI	276.48	143.51	72.66	41.0
GS	OMP	340.24	180.9	107.84	77.76
	MPI	351.5	180.44	92.42	47.9

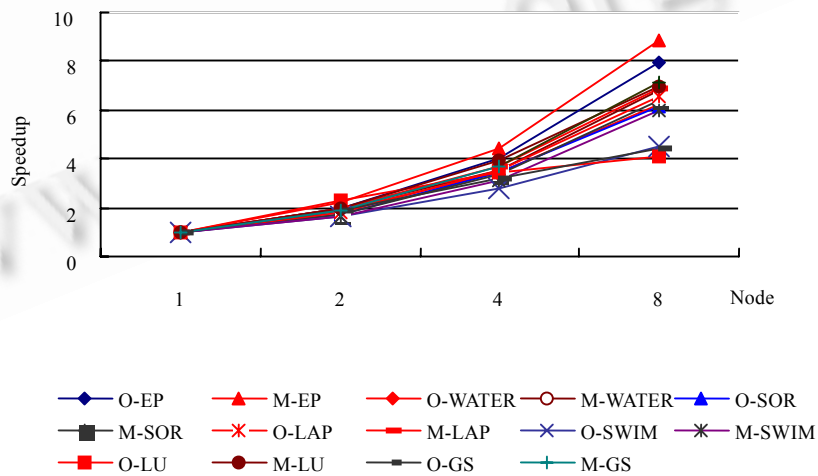

Fig.2 Speedup of applications written in OpenMP and MPI

图 2 应用的 OpenMP 版本和 MPI 版本的多机加速比

由图 2 可知,几乎所有应用的 OpenMP 版本的性能都比 MPI 要差,但两者性能相差不大.例如,在 2 个处理机

运行时,所有应用的 OpenMP 版本的平均性能相当于 MPI 版本的 96.8%;在 4 个处理机运行时为 92.2%;在 8 个处理机运行时则为 81%.另外,从图 3 中可看出,随着处理机数目的增加,应用的 OpenMP 版本与 MPI 版本的性能差距变大.例如,在 2 个处理机运行时,所有应用的 OpenMP 版本与 MPI 版本性能相差不超过 10%;在 4 个处理机运行时性能差距不超过 15%;而在 8 个处理机运行时,有些应用程序(LU,GS)性能相差较大,达到 40%.这表明 OpenMP/JIAJIA 的可扩展性比 MPI 要差.

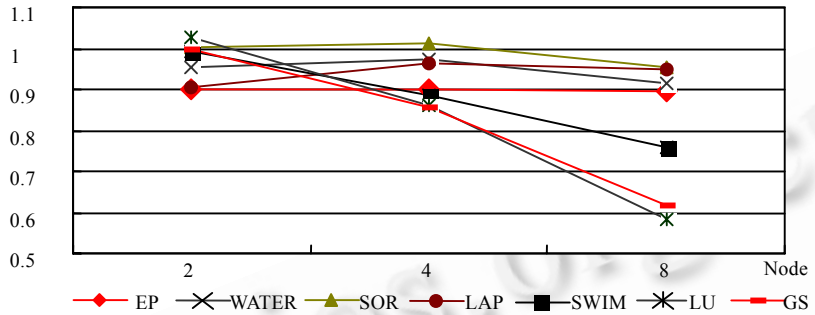

Fig.3 Speedup ratio of applications written in OpenMP and MPI

图 3 应用的 OpenMP 版本与 MPI 版本的加速比比值

3.2 应用在8个处理机时的并行性能比较和分析

下面我们具体分析各个应用的 OpenMP 版本与 MPI 版本的性能和通信之间的关系.图 4 为应用的 8 机加速比.从图 4 中可知,所有应用的 MPI 版本的性能均比 OpenMP 版本要好.对于其中的 4 个应用(EP,WATER,SOR,LAP),OpenMP 版本的性能达到 MPI 版本的 90%以上;有一个应用(SWIM)的 OpenMP 版本性能达到 MPI 版本的 76%;而有两个应用(LU,GS),其 OpenMP 版本的性能只相当于 MPI 版本的 60%左右.总的来说,所测试的 7 个应用在 8 处理机运行时,OpenMP 版本的平均性能相当于 MPI 版本的 81%.

表 3 为 8 个处理机运行时,应用发送的消息数和消息量.从表 3 中可以看出,应用的 OpenMP 版本发送的消息个数和消息量都比 MPI 版本要大.这主要是因为 OpenMP 版本翻译后是软件 DSM 程序,而软件 DSM 系统为了支持远程内存访问和维护整个共享存储空间的一致性,需要传送大量的消息.而对于精心设计的 MPI 程序,却可以将发送的消息数量减少到最低限度.所以从这个角度来说,MPI 版本发送的消息数是 OpenMP 版本的下界.

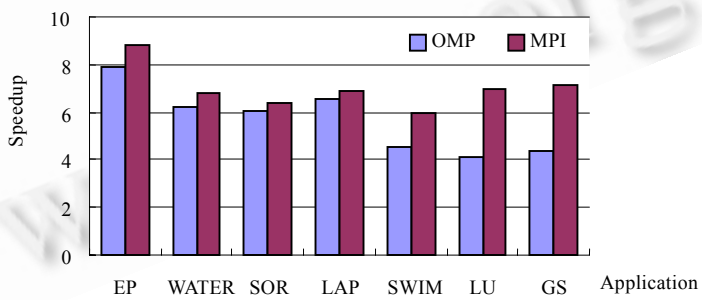

Fig.4 Eight-Nodes speedup of applications written in OpenMP and MPI

图 4 应用的 OpenMP 版本和 MPI 版本的 8 机加速比

Table 3 Message count and sum of applications running on eight-nodes

表 3 8 机情况下应用发送的消息数和消息量

Application	Message count		Message sum (byte)	
	OMP	MPI	OMP	MPI
EP	257	32	129740	256
WATER	13 619	1 447	40 394 276	19 262 144
SOR	10 025	2 800	23 640 984	22 937 600
LAP	14 647	1 400	58 376 684	45 875 200
SWIM	98 301	13 074	326 155 480	123 180 982
LU	75 786	14 336	178 934 684	117 383 168
GS	94 260	14 336	120 901 080	117 420 512

下面,根据通信特征的不同,我们将应用程序分为以下 5 类:

(1) 通信很少的程序,如 EP 等.EP 程序只在计算完成后做一个归约操作.对于这类应用而言,OpenMP 版本的性能与 MPI 版本相似,都接近线性加速比.图 4 中 EP 的 8 机加速比出现了超线性情况,这主要是由于 EP 的 MPI 版本是 Fortran 程序,而串行版本是 C 程序,Fortran 程序的执行效率要高一些.

(2) 对于使用较少共享存储空间的不规则问题,如 WATER 等.这类程序的计算量较大,计算与通信比高,通信不是程序性能瓶颈.尽管 OpenMP 版本发送消息数目和消息量比 MPI 版本多很多,但是由于软件 DSM 系统 JIAJIA 中采用了多种通信与计算重叠的优化技术,而且通信本身不是程序的性能瓶颈,所以增加的通信对程序性能影响并不大.这类应用的两种版本的性能相差不大.

(3) 访问规则的程序,如 SOR,LAP 等.这类程序由于访存规则,使用 OpenMP 的 *distribute* 制导扩展可以很好地实现拥有者计算,因此 OpenMP 版本的通信量与 MPI 版本相差不大.对于这类应用,OpenMP 版本的性能与 MPI 版本相差不大.

(4) 访存较为规则,但读写的数据存在碎片,如 SWIM 等.这类应用的 OpenMP 版本的通信量比 MPI 版本大很多,这主要是因为软件 DSM 系统以页粒度(通常为 4KByte)进行通信,由于不是数据页中的所有数据都会被访问,将造成浪费.对于这类程序而言,由于程序的计算与通信比不高,通信对性能影响较大,因此,OpenMP 版本的性能与 MPI 版本有一定差距.例如,SWIM 的 OpenMP 版本的 8 机加速比只有 MPI 版本的 76%.

(5) 访存比较规则,但存在访问冲突情况,如 GS,LU 等.由于这类程序具有单生产者/多消费者的访存特点,存在严重的通信竞争,即多个处理机同时向同一个处理机取数据.对于这类应用而言,尽管 OpenMP 版本的发送消息量与 MPI 版本相差不大,但是在处理机数目较大的情况下,性能相差很大.例如,GS 和 LU 的 OpenMP 版本的 8 机加速比只相当于 MPI 版本的 60%左右.这也反映出 OpenMP/JIAJIA 的可扩展性没有 MPI 好.

4 结 论

在机群中采用消息传递编程时,程序员必须对计算任务和数据进行划分,并安排在并行计算时处理机间所有的通信.在某些情况下,程序员安排处理机间通信很困难,特别是确定何时向哪个处理机发送或接收数据.OpenMP 标准的易编程性,特别是提供增量并行的能力,使得在机群上采用 OpenMP 比采用消息传递系统编程容易得多,因此很具有吸引力.但是,由于机群上 OpenMP 程序的性能与消息传递程序相比有差距,这种性能差距阻碍了机群 OpenMP 系统的广泛应用.

本文基于我们自己开发的机群 OpenMP 系统 OpenMP/JIAJIA 和典型的消息传递系统 MPI,利用广泛使用的测试程序,在 PC 机群上比较了这两者的性能.测试结果表明,所测试的 7 个应用的 OpenMP 版本在 8 个处理机运行时的平均性能达到 MPI 版本的 81%.这个结果在一定程度上反映了机群 OpenMP 系统与消息传递系统的性能差距.当然,对于不同类型的应用,性能差别情况可能不一样.

总之,机群 OpenMP 系统与消息传递系统有各自的优缺点:消息传递系统编程困难但性能较高,OpenMP 编程容易但性能差一些.因此,机群上的这两种编程模式会共存下去.我们今后的工作是在机群上将 OpenMP 与消息传递两种编程模型结合起来,以组合两者的优点,也包括进一步提高 OpenMP/JIAJIA 的性能.

References:

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

[2] OpenMP Architecture Review Board. OpenMP C and C++ Application Program Interface, Version 2.0. 2002.

[3] Lu HH, Hu YC, Zwaenepoel W. OpenMP on networks of workstations. In: Benton V, ed. Proc. of the Supercomputing'98. Orlando: IEEE Computer Society. 1998. 1~15

[4] Sato M, Sato S, Kusano K, Tanaka Y. Design of OpenMP compiler for an SMP cluster. In: Proc. of the 1st European Workshop on OpenMP. 1999. 32~39.

[5] Brunschen C, Brorsson M. OdinMP/CCP—A portable implementation of OpenMP for C. Concurrency and Computation: Practice and Experience, 2000,12(12):1193~1203.

[6] Lu HH, Dwarkadas S, Cox AL, Zwaenepoel W. Message passing versus distributed shared memory on networks of workstations. In: Redelfs A, ed. Proc. of the 1995 ACM/IEEE Supercomputing Conf. San Diego: ACM/IEEE Computer Society, 1995. 865~906

[7] Lu HH, Dwarkadas S, Cox AL, Zwaenepoel W. Quantifying the performance difference 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] 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 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1187.htm

[10] 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. Proc. of the HPCN Europe'99. LNCS 1593, Amsterdam, 1999. 463~472.

[11] Amarasinghe SP, Anderson JM, Lam MS, Tseng CW. An overview of the SUIF compiler for scalable parallel machines. In: David H, ed. Proc. of the 7th SIAM Conf. on Parallel Processing for Scientific Computing. Philadelphia: SIAM, 1995. 662~667.

[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] Woo SC, Ohara M, Torrie E, Singh JP, Gupta A. The SPLASH-2 programs: Characterization and methodological considerations. In: Proc. of the 22nd Annual Symp. on Computer Architecture. New York: ACM Press, 1995. 24~36.

[14] Aslot V, Domeika M, Eigenmann R, et al. SPEComp: A new benchmark suite for measuring parallel computer performance. In: Eigenmann, Michael JV, eds. Proc. of the Workshop on OpenMP Application and Tools. LNCS 2104, 2001. 1~10.

附中文参考文献:

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

[9] 胡明昌,史岗,胡伟武,唐志敏,张福新.PC 机群上 JIAJIA 与 MPI 的比较.软件学报,2003,14(7):1187~1194.http://www.jos.org.cn/1000-9825/14/1187.htm