

# 机群 OpenMP 系统的设计与实现

吴少刚<sup>1,2)</sup> 章隆兵<sup>2)</sup> 蔡 飞<sup>2)</sup> 顾丽红<sup>1)</sup> 唐志敏<sup>2)</sup>

<sup>1)</sup>(石油大学(华东)计算机与通信工程学院 东营 257061)

<sup>2)</sup>(中国科学院计算技术研究所 北京 100080)

**摘要** OpenMP 以其易用性和支持增量并行的特点成为共享存储体系结构的编程标准。目前机群系统已成为高性能计算的主流平台,研究机群 OpenMP 系统对推进并行应用的开发和普及非常有意义。该文作者以软件 DSM 系统 JIAJIA 作为 OpenMP 的运行时系统,结合一个前端编译器 OMP2JIA,在机群系统上实现了 OpenMP/JIAJIA 计算环境,同时在提高性能方面根据机群系统特点扩展了 OpenMP 制导,优化了后端运行时库。通过 11 个 OpenMP 应用,作者比较了该计算环境和一个支持 OpenMP 的硬件 cc-NUMA 系统(SGI 2100)的性能。结果表明,作者的机群 OpenMP 系统的 7 机平均加速比为 4.62;SGI 2100 系统为 4.55,二者性能相当。

**关键词** OpenMP;cc-NUMA;软件 DSM 系统;机群计算;JIAJIA

**中图法分类号** TP302

## Design and Implementation of OpenMP on Cluster Systems

WU Shao-Gang<sup>1,2)</sup> ZHANG Long-Bing<sup>2)</sup> CAI Fei<sup>2)</sup> GU Li-Hong<sup>1)</sup> TANG Zhi-Min<sup>2)</sup>

<sup>1)</sup>(College of Computer and Communication Engineering, University of Petroleum (East China), Dongying 257061)

<sup>2)</sup>(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)

**Abstract** The OpenMP Application Programming Interface (API) is an emerging standard for parallel programming on shared memory multiprocessors because of its ease of use and incremental approach to the parallelization of sequential programs. At present, clusters of workstations or PCs have been becoming the mainstream platform for high performance computing. It is increasingly attractive to develop OpenMP parallel applications on cluster systems. This paper presents the design and implementation of OpenMP computing environment on clusters, which is based on the combination of software DSM and compiler technologies. In order to improve the system performance and enlarge the application range, many methods are introduced to adapt to clusters architecture, such as OpenMP directive extension, runtime library optimization, loop scheduling algorithms etc. Furthermore, eleven OpenMP programs from some standard benchmarks have been measured on a hardware cc-NUMA machine (SGI 2100) and the OpenMP/JIAJIA system on authors' commodity cluster of PCs. The experimental results show that the mean speedup of the OpenMP/JIAJIA system on seven processors is 4.62 and that of the SGI 2100 machine is 4.55.

**Keywords** OpenMP; cc-NUMA; software DSM; cluster computing; JIAJIA

收稿日期:2003-09-08;修改稿收到日期:2004-05-12. 本课题得到国家自然科学基金(60303016)、国家“九七三”重点基础研究发展规划项目基金“大规模科学计算研究”(G1999032800)、中国科学院全国优秀博士学位论文作者专项基金、中国科学院计算技术研究所领域前沿青年基金(20026180-7)资助。吴少刚,男,1973 年生,博士,讲师,主要研究领域为计算机体系结构、共享虚拟存储系统、并行计算。E-mail: wsg@ict.ac.cn. 章隆兵,男,1974 年生,博士后,主要研究领域为计算机系统结构、机群计算、分布式共享存储系统。蔡 飞,男,1979 年生,博士研究生,主要研究领域为机群计算、微处理器设计和计算机系统结构。顾丽红,女,1970 年生,硕士,讲师,主要研究领域为嵌入式系统、分布式系统、Web 数据库。唐志敏,男,1966 年生,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、并行处理和 VLSI 设计。

# 1 引言

大规模科学和工程计算应用对高性能计算的需求是无止境的,例如气象预报、生物信息、飞行力学、药物设计、油藏模拟等应用,都需要 Tflops~Pflops 级的计算能力,并行计算是解决这些挑战性问题的唯一途径。然而并行软件开发很困难,主要原因是并行程序设计方法和手段还很落后。目前主流并行编程模型包括数据并行、消息传递和共享变量。数据并行的应用范围不广,且性能很大程度取决于编译器。消息传递已成为分布式存储系统的主要编程模式,但是它要求程序员显式地安排消息的发送和接收,编程困难,并且不支持增量并行。共享变量编程模型具有易于编程的特点,但是长时间来缺乏像消息传递中的 MPI 或 PVM 那样广为接受的标准,程序的可移植性差。OpenMP 标准委员会于 1997 年推出的 OpenMP 标准<sup>[1]</sup>有望改变这种局面,它得到了业界许多主要软硬件厂商的支持,目前已成为共享存储并行编程的实际工业标准。

OpenMP 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++ 的一组功能强大的高层并行结构和一个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。该标准在不断地扩充和发展,目前最新发布的是 2.0 版。支持增量并行和良好的可编程性使得 OpenMP 成为并行程序设计的主流模型之一。目前机群系统已成为主流的并行计算平台,尤其是基于 SMP (Symmetric Multi-Processor) 结点的机群系统。在这种平台上实现 OpenMP 是一个非常有意义的研究,它结合了 OpenMP 的易编程性和机群系统的可扩展性,将有利于推进并行应用的开发和普及。

本文介绍我们的机群 OpenMP 系统 OpenMP/JIAJIA 的设计与实现以及提高性能的相关技术。其主要贡献是在 JIAJIA 的基础上实现了机群 OpenMP,同时结合软件 DSM(Distributed Shared Memory)协议和机群系统特点提出了若干性能优化技术,包括合适的制导扩展和后端运行库优化等。测试结果表明,在我们评测的 11 个应用程序中,8 机平均加速比为 4.87。本文第 2 节介绍 OpenMP 标准;第 3 节具体描述了我们的机群 OpenMP/JIAJIA 系统的设计与实现;第 4 节介绍了我们针对机群系统特点所采用的一些优化技术;第 5 节给出了具体的性能测试结果;第 6 节介绍国内外的相关工

作;第 7 节总结全文。

## 2 OpenMP 介绍

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。本文的研究是在机群系统上实现 OpenMP C 版本,因此下文所有对 OpenMP 的描述缺省都是针对 OpenMP C 2.0 说明规范。OpenMP 定义了一个集编译制导、库函数和环境变量为一体的集合,用来描述 C 程序的共享存储并行机制,其目的是提供一个并行编程模型允许并行程序可以在不同厂商的共享存储体系结构的机器上移植。该 OpenMP 标准已被许多编译器商家支持。

OpenMP 允许用户创建、管理可移植的并行程序。编译制导以 SPMD(Single Program Multi-Data) 结构、工作共享结构、同步结构扩展了 C 顺序程序模型,提供对数据共享和私有化支持;库函数和环境变量提供了锁变量和对并行程序运行时行为的控制。OpenMP 采用 fork/join 的并行执行模式:OpenMP 程序首先由 Master 线程执行,直到碰到第一个并行结构(由 parallel 制导构成),接着由 Master 线程产生一组线程,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。并行结构执行完后,线程组隐式同步,接着 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 fork/join 多次。

## 3 OpenMP/JIAJIA 原型系统

OpenMP/JIAJIA 的基本设计思想是利用软件 DSM 系统 JIAJIA<sup>[2]</sup> 在机群上提供的共享存储界面,将 OpenMP 程序映射成等价的 JIAJIA 程序在机群平台上运行。OpenMP/JIAJIA 系统主要包括一个编译处理系统前端和一个支持 fork/join 模式的 JIAJIA 运行库后端。系统框架如图 1 所示。

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,OMP2JIA 编译器由我们自己基于 SUIF 开发,性能优化器是我们在系统设计时考虑采用编译分析技术

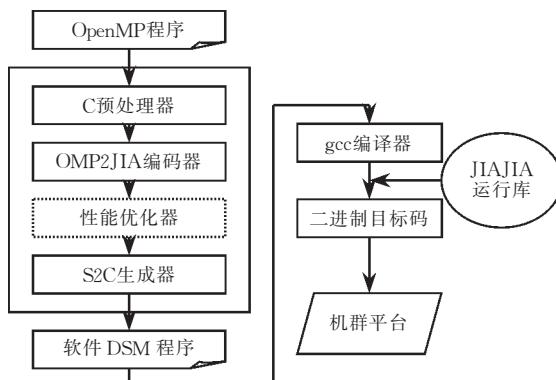


图 1 OpenMP/JIAJIA 系统框架图

提高性能所加入的模块,目前还没有实现。后端运行库由基于 SPMD 模式 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(例如 GNU 的 gcc)链接 JIAJIA 运行库,生成目标程序在机群平台上运行。

### 3.1 编译器 OMP2JIA

编译器的实现需要定位 OpenMP 和软件 DSM 系统之间的语义间隙。首先,OpenMP 假定轻量(light-weight)线程编程模式,整个地址空间缺省是共享的,除了线程私有栈中的数据外,地址空间中的数据可以共享访问,而 JIAJIA 采用重量(heavy-weight)进程编程模式,只有通过特殊分配方式标记为共享的数据才能在处理机之间共享,编译器需要处理这种地址空间全部共享和部分共享之间的语义差别。其次,OpenMP 中所有全局变量缺省为共享,而 JIAJIA 中缺省为私有。最后,编译器的工作是根据 OpenMP 的语义提取共享数据、处理并行区域、生成可执行环境,由于体系结构的差异,OpenMP 程序直接编译成基于 DSM 系统的并行程序,产生令人失望的性能,所以支持一些适应机群体系结构特点的制导扩展也是编译器应有的功能。

我们的 OMP2JIA 编译器基于斯坦福大学的著名 SUIF 编译工具集,在莱斯大学的翻译器框架基础上构成。SUIF 由一个精小的核心和许多以核心为基础的支持编译分析和优化的遍工具集组成。其核心定义了编译器的中间表示,提供访问和操作中间表示的编程接口;工具集包括 C 和 Fortran 的前端、循环级并行和局部性优化器、经过优化的 MIPS 后端和一些编译器开发工具等。OMP2JIA 编译器主要由编译制导识别、数据环境处理、并行任务生成和执行环境构造四个编译遍组成,加上之前的 C 预处理器和之后的目标源程序生成两遍构成了整个编

译处理系统。

### 3.2 JIAJIA 运行库

软件 DSM 系统由于结合了共享存储系统的易编程性和机群系统的可扩展性而受到广泛的研究。软件 DSM 系统用软件的方法把分布于各结点的多个独立编址的存储器转化为一个统一编址的共享虚拟存储空间,在分布式存储系统平台上提供共享存储的编程抽象。通常,相比消息传递的并行程序而言,对应的软件 DSM 并行程序在通信量方面要大一些,然而软件 DSM 所提供的共享存储编程环境的可编程性是消息传递所不及的。

JIAJIA 是一个完全建立在操作系统之上的用户级软件 DSM 系统,目前支持常见的类 Unix 平台和 Windows NT 平台。JIAJIA 的存储组织采用类似硬件 cc-NUMA (cache coherent Non-Uniform Memory Access) 方式,共享存储空间中的每个地址都有一个确定的宿主(home)结点;JIAJIA 采用基于锁的高速缓存一致性协议<sup>[3]</sup>来维护共享数据在不同处理机中的多个备份的一致性。另外,JIAJIA 在提高系统性能方面提出了许多协议优化策略,如单写的识别、数据自动迁移、写向量、基于 SMP 的优化、数据预取等。目前绝大多数软件 DSM 系统都是采用 SPMD 编程模式,JIAJIA 也不例外。为了匹配 OpenMP 的 fork/join 执行模式,我们扩充了 JIAJIA 系统调用接口,使得 JIAJIA 支持 fork/join 执行模式。

## 4 OpenMP/JIAJIA 性能优化

当前的 OpenMP/JIAJIA 的实现基本上满足了 OpenMP 的并行语义到 JIAJIA 语义直接映射的要求,然而 OpenMP 标准是针对 UMA (Uniform Memory Access) 共享存储结构制定的,由于这种结构不需要考虑数据分布和远程数据访问延迟问题,使得 OpenMP 标准对于非 UMA 结构的并行系统不是很有效。尤其是在机群这种通信开销较大的环境中,这种直接语义对照翻译,性能非常低下。为了在机群环境中获得满意的性能,我们需要对 OpenMP 制导进行扩展以及优化后端运行库。

### 4.1 OpenMP 制导扩展

#### 4.1.1 数据分布制导

OpenMP 标准是针对 UMA 共享存储体系结构制定的,不用考虑数据分布,然而在分布存储系统中,数据分布是影响性能的重要因素,在以软件 DSM

为基础的共享存储机群平台上更是如此。虽然共享存储编程不需要像消息传递编程那样考虑数据划分,但是为了性能需要考虑共享数据在处理间的分布。不过数据分布与数据划分是两个不同的概念,消息传递编程中的数据划分需要改变对被划分对象的引用,包括数组的名字以及数组下标;而数据分布只改变分配部分,不需修改引用部分,二者本质是不同的。

在 cc-NUMA 系统的 OpenMP 实现中,例如 SGI<sup>[4]</sup> 和 Compaq<sup>[5]</sup> 的系统等,都引入了数据分布制导来扩展 OpenMP 语言,大多数这些制导都源自 HPF 语言且与相应的 HPF 制导的语义相似。我们在 OpenMP/JIAJIA 实现中也仿效 HPF,扩展了 OpenMP 语言引入数据分布制导,不过由于软件 DSM 系统以页为粒度进行数据分布,且 JIAJIA 系统基于 home 维护数据一致性,因此在实现数据分布时需要对齐页且保证数据分配空间连续性,数据分布制导的功能受到一些限制。

#### 4.1.2 有效调度算法

OpenMP 提供了丰富的循环调度算法,然而这些算法的提出都只是从负载平衡角度考虑而没有考虑数据分布。正如前面所述,在分布存储系统中,数据分布情况和程序访问数据的模式极大地影响性能,因此在 OpenMP 的机群实现中,设计与数据分布匹配的循环调度算法是提高性能的关键。

循环调度性能的优劣主要受三个因素的影响:循环分配开销(loop allocation overhead)、负载不平衡(Load imbalance)、远程数据通信(remote data communication)开销。这三个因素之间是相互冲突的,为了减少远程数据通信的开销,应该充分利用处理器和存储器的局部性。如果一个处理器只对分布在本地存储器中的数据进行计算,就无需进行远程通信,通信开销很小。然而一味地考虑数据局部性则可能会引起负载不平衡,例如假设数据在处理器间的分布不均匀,则严格按照数据局部性就会造成处理器负载不平衡。可见数据局部性和负载均衡之间存在一定的矛盾,需要权衡折中。

Markatos 和 LeBlanc 在文献[6]中分析了硬件共享存储系统中负载平衡和局部性的重要性,他们得出的结论是局部性比负载平衡要重要得多。其结果表明:网络的速度越慢,局部性就越重要。由于我们的目标环境是普通的商用网络互连的机群系统,且软件 DSM 系统大多以页作为数据分配和通信的单位,通信开销很大,因此数据局部性比负载平衡重要得多。文献[7]中也清楚地表明了这一点。

针对机群系统结构的特点,我们提出了两种有效利用数据局部性的静态调度算法 LBS(Locality-Based Scheduling)和动态调度算法 LBDS(Locality-Based Dynamic Scheduling),充分体现拥有者计算的原则。在独占环境下,静态 LBS 非常有效,而在非独占的元计算环境下,某些负载极不平衡的应用则采用动态 LBDS 更有效。我们将这两种调度算法通过扩展 OpenMP 的循环调度子句的模式供用户使用。

#### 4.1.3 其它制导扩展

针对一些特定类型的应用,JIAJIA 系统提出了许多优化技术,例如动态数据预取<sup>[8]</sup>、自动 home 迁移<sup>[9]</sup>、写向量技术<sup>[10]</sup>等。这些优化技术对提高特定程序性能有很大好处,因此我们通过扩展 OpenMP 语言方式由程序员决定是否在系统后端采用这些优化技术。我们定义了一个绑定于 parallel 制导的 options 子句,该子句在并行区域的入口处打开相应的优化功能,并且在出口处自动关闭。

### 4.2 运行库优化

运行库优化集中在影响系统性能的关键制导和子句的实现上。reduction 子句是 OpenMP 的 for 制导中应用比较多的子句,优化归约操作是提高程序性能的有效途径之一。在 OpenMP 语义中,当 reduction 绑定在 parallel 或 for 制导上,表示一个全局的归约操作;当绑定在 sections 制导上,表示一个部分归约操作,且该子句中说明的归约变量为一个共享标量,然而在许多工程应用中,归约变量为向量的情况比较多,所以我们在实现中扩展了 OpenMP 的标准,支持 reduction 子句中说明归约向量。

在基于软件 DSM 的 OpenMP 实现中,为了实现对共享的归约变量的互斥访问,通常的做法是给该变量加锁保护。我们最初的实现也是采用这种方法,然而在软件 DSM 系统中,在处理器数目较多的情况下,锁的使用开销比较大,例如在 JIAJIA 中,锁的申请需要全局锁管理器的授权,同时释放锁需要维护一致性。为了替换加锁机制,我们采用类似 MPI 中的归约方法,即用传递消息的方式将各线程的分量发送给该共享归约变量的 home 所在的那个线程,通常是 Master 线程,然后由该线程完成相应的归约操作。这种实现的好处是:首先消除了引入锁带来的锁管理开销;其次由于各分量在执行归约操作的处理器上有备份,这样节省了远程取页、创建 diff、服务中断等开销。reduction 优化前后的性能如图 2 所示,从中可以看出,在机群 OpenMP 中, reduction 的优化效果随处理器数的增加变得更加明

显,这是因为在软件 DSM 系统中,处理机数越多,对锁的竞争越激烈,导致的开销就越大.两机情况下,优化后时间为优化前的 54.6%;而在 16 机时,优化前归约操作开销为  $49247.72\mu s$ ,优化后为  $16797.68\mu s$ ,为前者的 34.1%.

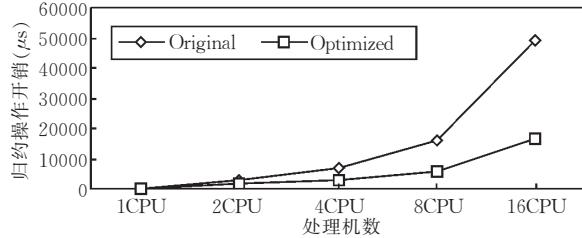


图 2 reduction 操作优化前后性能比较

## 5 性能测试

我们的 PC Cluster 由 8 个 PC 结点构成,每个结点支持两个主频为 700MHz 的 Pentium III 处理器. 该处理器拥有指令和数据分离的、容量各为 16KB 的一级 Cache, 每个 Cache 采用 2 路组相联设计,而且每个处理器还拥有一个 256KB 的二级 Cache.

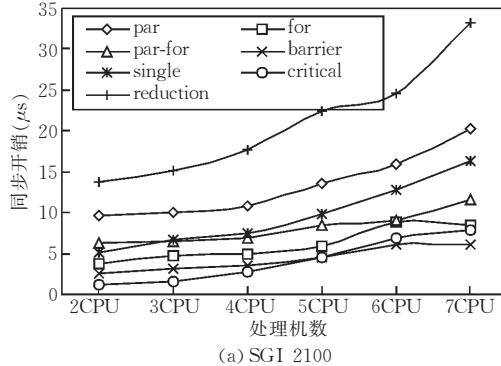


图 3 OpenMP 同步开销

从图 3 中可以看出:首先,SGI 2100 上的制导开销属于微秒级,而 PC Cluster 上的制导开销则于毫秒级,这充分反映了硬件 cc-NUMA 和软件共享存储机群在通信性能上的差异.其次,在 PC Cluster 同步开销曲线中,许多制导的曲线与 barrier 制导的曲线相类似.这主要是因为许多制导都隐含了对共享数据一致性维护要求,例如在并行区域的入口和出口以及 for 和 single 制导的出口都需要维护一致性.由于我们的数据一致性维护采用 barrier 操作实现,而在软件 DSM 中 barrier 操作开销比较大,因此这些曲线都与 barrier 制导曲线类似.这也说明优化同步是减少 OpenMP 制导开销的主要途

径,例如同步消除、同步弱化技术等.另外,OpenMP 制导在两种平台上的同步开销曲线的变化趋势很类似,这说明就 OpenMP 编程模型来说,OpenMP/JIAJIA 系统和 cc-NUMA 的性质是一样的,主要差别体现在性能级别上.也就是说,无论 Cache 一致性是采用软件实现还是硬件实现,其本质是一样的.

与之比较的硬件 cc-NUMA 结构的机器为 SGI 公司于 1999 年推出的 SGI 2100 服务器,这是一款支持包括科学工程计算、生物信息和视频流媒体服务等计算密集型应用的高性能产品.该服务器采用全配置的 4 个结点板(node board)集成,每个结点板支持两个主频为 250MHz 的 MIPS 64-bit R10000 处理器. R10000 处理器拥有指令和数据分离的、容量各为 32KB 的一级 Cache,每个 Cache 采用 2 路组相联设计,而且每个处理器还拥有一个 4MB 的二级 Cache.系统配备了 4GB 的主存,结点板上的内存峰值带宽为 780MB.操作系统为 IRIX 6.5,编译器为 MIPSpro 7.3,采用-O2 级优化.

### 5.1 实现开销

为了理解 OpenMP 制导在 SGI 2100 和 PC Cluster 两种平台上的实现开销,本文使用文献[11]中的 microbenchmark 程序进行了测试.目前我们主要关心同步开销,所以仅使用了其中的同步测试程序,结果如图 3 所示.

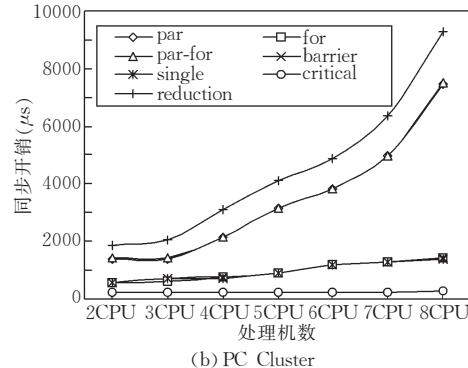


图 3 OpenMP 同步开销

径,例如同步消除、同步弱化技术等.另外,OpenMP 制导在两种平台上的同步开销曲线的变化趋势很类似,这说明就 OpenMP 编程模型来说,OpenMP/JIAJIA 系统和 cc-NUMA 的性质是一样的,主要差别体现在性能级别上.也就是说,无论 Cache 一致性是采用软件实现还是硬件实现,其本质是一样的.

### 5.2 测试程序

我们使用了一些被广泛采用的基准程序:NAS 的并行程序集中的蒙特卡罗模拟程序 EP、多网格计算程序 MG、共轭梯度算法求最小特征值程序 CG;SPEC OMP2001 程序集中的浅水模型程序 Swim、神经网图像识别程序 Art 和地震模拟程序 Equake;

SPLASH2 程序集中的水分子模拟程序 Water; 莱斯大学 OpenMP-Now 项目<sup>[12]</sup>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<sup>[13]</sup>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。

EP 程序的主要目的是产生一组高斯分布的数据对。该程序特别适合于并行，程序中唯一的通信是在程序的最后进行一次累加操作。在程序的主循环处采用 parallel reduction 制导并行化。

MG 程序在固定的边界条件下用 multigrid 算法在一个三维立方体上求方程  $\nabla^2 u = v$  中  $u$  的近似解。multigrid 是一种基于多级范例的快速线性迭代算法，可以和常用的离散化技术一起使用，其典型应用是求二维或多维椭圆偏微分方程的数值解。该算法中主要的共享数据是 17 个三维数组。计算的并行部分采用 parallel for 制导，不同的节点分别计算当前 level 对应矩阵的不同横切面（相当于将立方体横切成多个大小相等的三维矩阵，平均分配给各个节点进行计算，对应于在最外层循环划分并行任务），在对每个横切面进行计算的时候需要访问与其上下相邻的两个面。

CG 程序采用共轭梯度 (conjugate gradient) 算法计算大型对称正定稀疏矩阵的最小特征值，其中的稀疏矩阵随机产生。该程序可用来测试系统对于非规则应用的远程通信能力，程序中多个主循环采用 parallel reduction 制导。

Swim 程序是一个为天气预报建模的程序，它使用有限差分方法求解浅水方程组 (shallow water equations)。程序中定义了 14 个大小相同的共享二维数组，每个数组都采用了 block 分布模式制导。程序结构分为初始化和迭代模拟两部分。每个迭代步中包含 3 个函数调用和一个 reduction 操作。

Art 程序利用 ART 2 (Adaptive Resonance Theory 2) 神经网络来识别红外图像中的物体。程序分训练和识别两个阶段。其主要数据结构是 art-2 神经网络和一个用来记录识别物体匹配度的一维数组 mat\_con。每个处理机拥有负责维护自己的 art-2 神经网络（每个进程的 art-2 神经网络都是相同的），而 mat\_con 数组以 block 分布模式在各处理机中共享。

Equake 程序是一个模拟地震波在大型、具有地质多样性的盆地中的传播，以推演盆地中任意一点在发生特定的地震事件时受地震波影响所发生的位置变化的程序。该程序利用非结构的有限元方法模拟了 1994 年发生在美国南加州 San Fernando 盆地

中的 Northridge 地震的余震对盆地的影响。程序中使用一个 3 维的非结构网格对该盆地中  $50\text{km} \times 50\text{km} \times 10\text{km}$  大小的区域的地质结构进行建模。Equake 分两个部分：初始化部分读取输入文件中的数据，包括该区域的地质参数和震源及震中位置等；模拟部分包含 3334 个迭代步（我们只计算了前 240 个迭代步），每个迭代步中主耗时部分是对函数 SMVP 的调用（计算稀疏矩阵和向量的乘积），程序的并行也是针对该函数。

Water 是一个水分子动力学模拟程序，逐步地模拟分子的运动状态。其主要的数据结构为一个共享的一维数组，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。Water 采用 parallel 制导并行，在每个时间步，每个处理机都需要计算出本机上的每个分子与其它分子之间的作用力，该计算封装在 parallel 制导的并行区域内。for 制导采用块调度，尽可能使用 nowait 子句减少不必要的同步开销。

GS 是一个计算  $M$  个  $N$  维向量的正交基的应用。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。每次迭代的正交化部分采用 parallel for 制导并行执行。 $M$  个  $N$  维向量以一个二维数组存放，在并行区域中共享。

SOR 程序用红黑逐次超松弛迭代法解偏微分方程。数组中的红黑元素交叉，每个红元素周围是 4 个黑元素（边界元素除外），反之亦然。在每次迭代中，每个数组元素更新为相邻元素的平均值，采用 parallel for 制导并行化计算循环。红黑数组的元素平均分配在所有处理机上。

LAP 程序用雅可比迭代法求解二维拉普拉斯方程。新旧两个二维数组平均分配在所有处理机上。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，此步我们采用 parallel for 并行化；然后采用 parallel for reduction 制导并行计算新旧数组迭代误差；最后使用了 parallel for 并行更新旧数组。

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。在每次迭代中，对角线元素所在行处理完后，采用 parallel for 制导并行更新当前元素右下角矩阵。矩阵以一个二维数组存放，分解后的  $L$  矩阵和  $U$  矩阵存放在原来的数组中。

### 5.3 测试结果及分析

表 1 列出了每个应用程序的特性,包括测试规模和所占用的共享空间。其中 NAS 程序集的规模从小到大依次为 S, M, A, B 和 C; 而 SPEC OM-PL2001 程序集的规模依次为 test, train 和 ref。

表 1 应用程序特征

程序	规模	共享空间	程序	规模	共享空间
EP	Class W	24KB	Water	1728 分子	1MB
MG	Class A	432MB	GS	$4096 \times 4096$	64MB
CG	Class B	428MB	LU	$4096 \times 4096$	128MB
Swim	Train	191MB	SOR	$4096 \times 4096$ , 迭代 100 次	128MB
Art	Ref	552KB	LAP	$4096 \times 4096$ , 迭代 100 次	256MB
Equake	Ref, 迭代 240 次	355MB			

根据共享数据量和数据访问模式的特点,我们将上述程序分成四类,如表 2 所示。

表 2 应用程序分类

应用程序特点	共享数据量大	共享数据量小
数据访问规则	规则程序	易并行程序
数据访问不规则	非规则程序	易共享程序

(1) 易并行程序。这类程序的共享数据量少,容易开发并行,而且通信非常少。例如计算圆周率以及本文的 EP 程序等。EP 程序只有 24KB 的共享数据,而且基本没有同步,只在程序结束时进行少量通信。

(2) 易共享程序。这类程序通过一些简单的私有化手段可以使得程序的共享数据较少,容易共享并行,然而并行任务间交互需要频繁地更新共享数据,消息量也不小。例如 Water 程序的共享空间只有 1.1MB,但是所有处理机频繁访问共享空间,这对一致性维护协议提出了较高要求。还有 Art 程序,共

享空间只有 0.5MB,也需要并行进程间频繁交互。

(3) 规则程序。这类程序尽管共享数据量大,然而数据访问比较规则,可以通过合适的数据分布策略挖掘局部性。例如 SOR 和 LAP 的访存行为非常规整,采用 BLOCK 数据分布策略可以较好地实现拥有者计算。GS 和 LU 的计算模式很相似,GS 并行修改本次迭代之后的向量,而 LU 只并行修改右下角矩阵,采用 CYCLIC 数据分布策略可以很好地实现拥有者计算。由于 LU 的每次迭代的计算量递减,计算负载不平衡,其性能不及 GS。MG 程序的数据访问模式也比较规则,但是数据分布不容易对齐,在 A 规模下需要 432MB 的共享空间,而且同步比较多。

(4) 非规则程序。这类程序共享数据量大而且数据访问模式不规则,很难采取合适的数据分布策略,导致远程数据访问频繁,通信量极大。例如 CG 程序和 Equake 程序中都涉及到一个很大的稀疏矩阵和向量的乘积运算,使用的共享空间分别为 428MB 和 355MB,数据访问模式不规则。

表 3 列出了 11 个应用程序在多机情况下的计算时间(注:SGI 2100 服务器有一个处理器已坏,我们在测试中只能使用 7 个处理器,在 PC Cluster 上我们测试了 8 个处理器时的执行时间)。从表 3 中可以看出,虽然在 SGI 2100 和 PC Cluster 两种平台上运行是相同规模的同一程序,然而在单处理器运行时性能相差较大,主要因为:(1) 编译器差异。MIPSpro 7.3 编译器和 gcc 2.96 编译器采用 O2 级别的优化程度不一样,前者编译出的目标代码效率比后者高;(2) 体系结构差异,尽管 SGI 2100 的 R10000 处理器的主频比较低,但是其一级和二级 Cache 都远大于机群的 Pentium III 处理器。除了 EP, CG, Swim, Equake 和 Water 5 个程序的机群单处理器性能不及 SGI 2100 外,其它 6 个程序都优于后者。

表 3 应用程序计算时间

(单位:s)

应用 程序	1 处理机		2 处理机		4 处理机		7 处理机		8 处理机	
	SGI	Cluster	SGI	Cluster	SGI	Cluster	SGI	Cluster	SGI	Cluster
EP	26.25	44.08	13.30	22.06	7.51	11.02	4.06	6.81	—	5.55
MG	57.69	55.16	38.01	39.38	22.27	28.98	18.16	26.23	—	25.00
CG	1475.29	2846.18	912.06	1685.60	698.94	1133.13	554.44	1173.12	—	1138.75
Swim	1307.48	2226.71	837.61	1145.74	424.62	682.68	315.94	617.70	—	512.76
Art	23391.61	21678.97	14470.28	12154.03	9471.96	6723.93	6237.62	3990.27	—	3488.84
Equake	570.68	676.32	419.35	653.55	219.06	452.80	160.88	322.42	—	324.96
Water	142.52	156.88	73.03	86.82	37.37	47.93	21.56	31.41	—	27.96
GS	3301.46	1681.90	1673.28	880.97	851.83	484.45	515.65	393.07	—	369.77
LU	1965.36	1624.72	994.16	808.44	709.83	448.43	574.71	417.50	—	387.46
SOR	194.67	90.05	100.45	42.72	60.46	23.00	34.24	15.22	—	13.77
LAP	326.00	277.97	199.61	134.98	100.32	70.11	77.37	48.70	—	43.61

图 4 给出了上述 11 个应用在 SGI 2100 和 PC Cluster 两种平台上的多机加速比。由于在两种平台上程序执行的绝对时间存在差异,比较加速比是一个合适的方式。在上述四类程序中,易并行程序 EP 在 SGI 2100 和 PC Cluster 上都获得了接近线性的加速比;易共享程序 Art 和 Water 在两种平台上的 4 机平均加速比分别为 3.1 和 3.2,7 机平均加速比分别为 4.7 和 5.2,由于 JIAJIA 在处理普通访存失效方面作了许多优化,这两个程序的多机加速比都比较好;LU,GS,SOR,LAP,Swim 和 MG 6 个规则

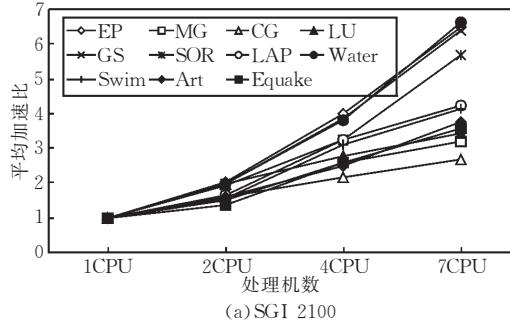


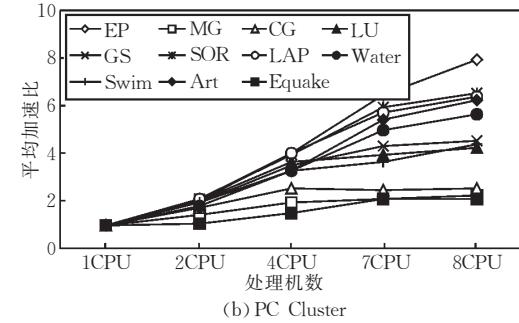
图 4 OpenMP 应用程序多机加速比

从上述分析中可以看出,基于 OpenMP/JIAJIA 的机群系统表现出与硬件 cc-NUMA 相似的共享存储结构特点;在这 11 个比较有代表性的应用方面,OpenMP/JIAJIA 系统能获得与硬件 cc-NUMA 系统相近的性能,能满足一定范围的应用领域;通信开销是非规则应用性能的决定因素,所以适当的硬件支持远程数据访问是软件共享存储机群系统的重要研究方向。

## 6 相关工作

针对基于 SMP 的机群系统,近年来许多项目组在研究结点间利用 MPI 消息传递、SMP 结点内利用 OpenMP 共享存储的混合编程模式,这种两级并行方法充分匹配两种体系结构的特点,不过其缺点是对程序员提出了更高的要求,需要掌握消息传递和共享存储两种编程方法,增加了程序员负担。从编程友好性出发,采用共享存储编程模式是机群系统的理想编程方法,目前在这种平台上的 OpenMP 实现都是以某个软件 DSM 系统作为后端的运行时库。美国莱斯大学的 Lu<sup>[12]</sup> 等人最先在机群系统上基于 TreadMarks 所提供的虚拟 COMA 平台实现了 OpenMP 子集,但是其实现没有很好地结合机群系统特点,合适的应用并不多;日本的 Omni 项目

程序在两种平台上的 4 机平均加速比分别为 3.1 和 3.3,7 机平均加速比分别为 4.5 和 4.4,在数据分布得当的情况下,规则程序的加速比也比较好;CG 和 Equake 两个非规则程序在两种平台上的 4 机平均加速比分别为 2.2 和 2.0,7 机平均加速比分别为 3.1 和 2.2,它们的计算和通信比值较小,扩展性不好。因此,对于前三类程序,PC Cluster 与 SGI 2100 的性能相当;对于非规则程序,由于通信开销很大,两种平台上的性能都不太理想,而且 PC Cluster 性能不及 SGI 2100。



组<sup>[13]</sup>实现了一个基于软件 DSM 系统 scash 的 OpenMP 全集,且有条件地扩展了 OpenMP 循环调度和支持数据分布,不过性能还不够理想;美国普渡大学 Basumallik<sup>[14]</sup>等人用手动翻译的方式分析了 OpenMP 应用在基于软件 DSM 的机群上的性能问题;从事类似研究的还有瑞典的 Odin 项目组<sup>[15]</sup>以及美国休斯敦大学等。

## 7 总结和未来的工作

并行处理系统要取得真正成功,易用性是重要因素。由于机群 OpenMP 结合了 OpenMP 的易用性和机群系统的可扩展性,因此机群 OpenMP 的研究对推进并行应用的开发和普及非常有意义。本文介绍了基于 JIAJIA 系统的机群 OpenMP 的设计和实现,以及提高系统性能的许多优化措施。通过对 11 个应用程序的测试,我们的机群 OpenMP/JIAJIA 原型系统获得了与硬件 cc-NUMA 机器(SGI 2100)相当的性能。我们的经验表明机群 OpenMP 的关键问题在于如何提高性能,而提高性能的方法主要包括合适的 OpenMP 制导扩展、后端软件 DSM 系统优化、前端编译器的自动优化以及对软件 DSM 协议和 OpenMP 语义的专门硬件支持等。目前我们课题组正在研究有关硬件支持技术,而编译器优化技

术也是我们今后进一步努力的方向。

**致谢** 感谢中国科学院数学与系统科学研究院张林波研究员为我们提供 SGI 2100 计算平台;感谢美国莱斯大学 Honghui LU 提供给我们部分测试程序和编译技术指导;感谢日本 Omni 项目组提供部分 NAS 的 OpenMP 并行程序集。

## 参 考 文 献

- 1 OpenMP Architecture Review Board. OpenMP C and C++ Application Program Interface, Version 2.0, March 2002. <http://www.openmp.org/>
- 2 Hu W. W., Shi W. S., Tang Z. M.. JIAJIA: A software DSM system based on a new cache coherence protocol. In: Proceedings of HPCN Europe'99, Amsterdam, 1999, 463~472
- 3 Hu W. W., Shi W. S., Tang Z. M., Li M.. A lock-based cache coherence protocol for scope consistency. Journal of Computer Science and Technology, 1998, 13(2):97~109
- 4 Chandra R., Chen D. K., Cox R. et al.. Data distribution support on distributed shared memory multiprocessors. In: Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'97), Las Vegas, 1997, 334~345
- 5 Bircsak J., Craig P., Crowell R. et al.. Extending OpenMP for NUMA machines. Scientific Programming, 2000, 8(3):163~181
- 6 Markatos E. P., LeBlanc T. J.. Load balancing vs. locality management in shared memory multiprocessors. In: Proceedings of the International Conference on Parallel Processing (ICPP '92), Innsbruck, Austria, 1992, 258~267
- 7 Shi W. S., Tang Z. M., Hu W. W.. A more practical loop scheduling for home-based software DSMs. In: Proceedings of the ACM-SIGARCH Workshop on Scheduling Algorithms for Parallel and Distributed Computing—from Theory to Practice, Greece, 1999, 154~161
- 8 Hu W. W., Zhang F. X., Liu H. M.. Dynamic data prefetching in home-based software DSMs. Journal of Computer Sciences and Technology, 2001, 16(3):231~241
- 9 Hu W. W., Shi W., Tang Z. M.. Home migration in home-based software DSMs. In: Proceedings of the 1st Workshop on Software Distributed Shared Memory, Greece, 1999, 162~173
- 10 Hu W. W., Shi W. S., Tang Z. M.. Adaptive write detection in home-based software DSMs. In: Proceedings of HPDC'99, California, 1999, 27~38
- 11 Bull J. M.. Measuring synchronization and scheduling overheads in OpenMP. In: Proceedings of the 1st European Workshop on OpenMP (EWOMP'99), Lund, Sweden, 1999, 99~105
- 12 Lu H., Hu Y. C., Zwaenepoel W.. OpenMP on networks of workstations. In: Proceedings of Supercomputing'98, Orlando, USA, 1998, 1~15
- 13 Sato M., Sato M., Kusano K., Tanaka Y.. OpenMP design for an SMP cluster. In: Proceedings of the 1st European Workshop on OpenMP (EWOMP'99), Lund, Sweden, 1999, 106~112
- 14 Basumallik A., Min S. J., Eigenmann R.. Towards OpenMP execution on software distributed shared memory systems. In: Proceedings of the International Workshop on OpenMP: Experiences and Implementations (WOMPEI'02), Kyoto, Japan, 2002, 457~468
- 15 Brunschen C., Brorsson M.. OdinMP/CCP: A free portable OpenMP implementation for C. In: Proceedings of the 1st European Workshop on OpenMP (EWOMP'99), Lund, Sweden, 1999, 123~129



**WU Shao-Gang**, born in 1973, Ph. D., lecturer. His research interests include system architecture, software distributed shared memory system, parallel computing.

**ZHANG Long-Bing**, born in 1974, Ph. D., postdoctor. His research interests include system architecture, cluster computing, software distributed shared memory system.

## Background

This work is supported by the National Natural Science Foundation of China under Grant No. 60303016 and other foundations. This project is focused on how to implement a high performance OpenMP computing environment on clusters. The group have developed an OpenMP system towards clusters platform named OpenMP/JIAJIA by integrating

**CAI Fei**, born in 1979, Ph. D. candidate. His research interests include cluster computing, microprocessor design and system architecture.

**GU Li-Hong**, born in 1970, M. S., lecturer. Her research interests include embedded system, distributed system, web database.

**TANG Zhi-Min**, born in 1966, Ph. D., researcher, Ph. D. supervisor. His research interests include high performance computer architecture, parallel processing.

complier and software distributed shared memory technologies. They are studying some other technologies to achieve good performance and scalability of the OpenMP/JIAJIA system, such as the various programming styles, complier optimization, special hardware support, and so on.