

# 适合机群 OpenMP 系统的制导扩展

章 隆 兵<sup>1)</sup> 吴 少 刚<sup>2)</sup> 蔡 飞<sup>1)</sup> 胡 伟 武<sup>1)</sup>

<sup>1)</sup>(中国科学院计算技术研究所 北京 100080)

<sup>2)</sup>(石油大学(华东)计算机科学系 东营 257062)

**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。机群 OpenMP 系统在机群上实现了 OpenMP 计算环境, 它将 OpenMP 的易编程性和机群的可扩展性结合起来, 是很有意义的。OpenMP 的编程方式主要有循环级和 SPMD 两种, 其中循环级方式易于编程而 SPMD 方式难于编程。然而在机群 OpenMP 系统中获得高性能 OpenMP 程序, 必需采用 SPMD 方式。该文描述了适合机群 OpenMP 系统的一个简单的 OpenMP 制导扩展子集(包括数据分布制导、循环调度模式), 并在机群 OpenMP 系统 OpenMP/JIAJIA 上进行了实现。应用测试表明, 利用这些制导扩展进行编程, 既保持循环级方式的易编程性又获得与 SPMD 方式相当的性能, 是有效的编程方式。

**关键词** OpenMP; 制导扩展; 数据分布; 软件分布式共享存储系统; JIAJIA

**中图法分类号** TP302

## Directive Extensions of OpenMP System Based on Cluster

ZHANG Long-Bing<sup>1)</sup> WU Shao-Gang<sup>2)</sup> CAI Fei<sup>1)</sup> HU Wei-Wu<sup>1)</sup>

<sup>1)</sup>(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)

<sup>2)</sup>(Department of Computer Science, University of Petroleum (East China), Dongying 257062)

**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 system based on cluster which combines the programmability of OpenMP with scalability of cluster. The OpenMP program could be written in both loop-level parallel style and SPMD style. The SPMD style is not easy to program while the loop-level style is easy. Unfortunately, it is necessary to program in SPMD style for high performance on OpenMP system based on cluster. In this paper, some directive extensions have been illustrated, such as data distribution directive, new loop schedule scheme, etc. and these directive extensions have been implemented on our OpenMP system based on cluster named OpenMP/JIAJIA. Experimental results show that the performance of programs' version written in these directive extensions is as good as the version written in SPMD style while the programmability is as good as loop-level style. So it is an effective programming style to program in these directive extensions on OpenMP system based on cluster.

**Keywords** OpenMP; directive extension; data distribution; software distributed shared memory system; JIAJIA

收稿日期: 2003-02-17; 修改稿收到日期: 2003-03-10. 本课题得到国家自然科学基金(60303016)、中国科学院全国首届优秀博士学位论文作者专项基金资助。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。E-mail: lbzhang@ict.ac.cn. 吴少刚, 男, 1973 年生, 博士研究生, 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。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。胡伟武, 男, 1968 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

OpenMP 是当前支持共享存储并行编程的工业标准。相比于消息传递和 HPF 等并行编程模型，可移植、支持增量并行及良好的可编程性等特点使得 OpenMP 成为并行程序设计的主流模型之一。目前机群系统由于其性价比高和可扩展性好的特点，正逐渐成为主流的并行计算平台。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，对其进行研究是很有意义的<sup>[1~3]</sup>。

目前机群 OpenMP 系统主要利用软件 DSM (Distributed Shared Memory) 系统在机群上构造的类 NUMA (Non-Uniform Memory Access) 结构的虚拟共享存储界面，将 OpenMP 程序转换成软件 DSM 程序在机群上运行。由于 OpenMP 标准是针对 UMA (Uniform Memory Access) 结构制定的，要在 NUMA 结构上获得高性能并不容易，特别是在通信延迟较大的机群系统上。OpenMP 的编程模式主要有循环级和 SPMD (Single Program Multi-Data) 两种编程方式，其中循环级方式编程容易而 SPMD 方式编程困难<sup>[4]</sup>。然而要在 NUMA 结构机器上获得高性能，用户必须采用 SPMD 方式来手工划分数据和计算任务，尽量实现拥有者计算以减少通信。

本文针对机群 OpenMP 系统的特点，描述了一个简单的 OpenMP 制导扩展子集，包括合适的数据分布制导扩展和充分利用拥有者计算原则的循环调度模式 LBS (Locality-Based Scheduling) 和 LBDS (Locality-Based Dynamic Scheduling)，并在我们自己的机群 OpenMP 系统 OpenMP/JIAJIA 上进行了实现。应用测试表明，采用这些制导扩展编程，既获得了循环级方式的易编程性，又获得了与 SPMD 方式相当的性能。本文第 2 节介绍机群 OpenMP 系统 OpenMP/JIAJIA；第 3 节介绍数据分布制导扩展以及循环调度模式；第 4 节是性能评测；第 5 节是相关工作；最后是本文的结论和未来的工作。

## 2 机群 OpenMP 系统

### 2.1 OpenMP

OpenMP 标准定义了一个集编译制导、库例程和环境变量为一体的集合，可以方便程序员开发高可移植的共享存储程序。目前 OpenMP ARB 发布

的分别支持 Fortran, C/C++ 的最新规范是 2.0 版本<sup>[5]</sup>。本文中对 OpenMP 的描述缺省都是针对 OpenMP C 2.0 规范。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三种类型：并行和工作共享制导、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任务划分成子任务在多个线程上执行，包括 for, sections, single。工作共享制导可以和一个并行块绑定在一起。数据环境制导用于在并行执行时控制数据环境，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。OpenMP 的库函数和环境变量主要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 并行执行模式。完成并行结构后，线程组隐式同步。程序中可以说明多个并行结构，因此在执行时 fork/join 多次。

### 2.2 JIAJIA 系统

JIAJIA<sup>[6]</sup>是本实验室开发的一个基于 Home 的软件 DSM 系统，采用基于锁的新型高速缓存一致性协议来实现域一致性。JIAJIA 的存储器组织方式类似于 NUMA 结构。在 JIAJIA 中，每一页都有一个相应的 Home 结点，处理机的本地存储器分为 Home 区域和软件 Cache 区域。Home 区域用来存放那些 Home 结点是自己的内存页。当处理机访问 Home 分布在本地的那一部分共享内存时，只进行本地访问操作；当处理机访问 Home 分布在其他结点上的页面时，需要进行远程通信，将该页取回到本地的软件 Cache 中。通常当一个页的 Home 结点指定后，就固定不变，数据的初始 Home 分布对程序的性能影响很大。JIAJIA 提供了可以灵活控制数据初始 Home 分布的编程接口 *jia\_alloc()*。JIAJIA 获得了与消息传递系统可比的性能<sup>[7]</sup>。

### 2.3 OpenMP/JIAJIA

OpenMP/JIAJIA 是我们自己开发的一个基于 JIAJIA 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 主要包括源—源 编译器 OMP2JIA 和改造过的 JIAJIA 运行库后端。

OMP2JIA 主要将 OpenMP 程序转换成等价的 JIAJIA 程序，它的主要工作是根据 OpenMP 的语义提取共享数据，处理并行区域，生成可执行环境。

OMP2JIA 实现主要基于 Stanford 大学的 SUIF 编译工具集。SUIF 是一个功能强大、应用灵活且开放源码的编译基础设施系统,由一个精小的核心和许多基于此核的支持编译分析和优化的遍工具集组成。OMP2JIA 编译器结构由四遍组成,依次为编译制导识别、数据环境处理、并行任务生成、执行环境构造。加上原有的 C 预处理和目标程序生成两遍,构成了整个编译处理系统。另外由于 JIAJIA 采用 SPMD 执行模式,为了匹配 OpenMP 的 fork/join 执行模式,我们扩充了 JIAJIA 系统调用接口,使之支持 fork/join 执行模式。

## 2.4 编程方式

OpenMP 主要有两种编程方式:细粒度的循环级方式和粗粒度的 SPMD 方式。循环级方式是 OpenMP 的一种主流编程方式,它主要开发程序的循环级并行性,一般是通过使用“# pragma omp for”制导,将循环迭代分布到多个线程上并行执行。SPMD 方式是一种粗粒度并行的编程方式,主要利用“# pragma omp parallel section”制导来开发任务级并行性。SPMD 方式的策略是尽量将数据私有化,最少化共享数据,实现最大的拥有者计算。在 SPMD 方式下,我们将数据在处理机间分布,并将分布到本地的数据私有化。为了实现 OpenMP 程序线程间的通信,需要分配共享缓冲区来实现通信,一个线程向共享缓冲区中写,另一个线程从中读以实现通信。由于是在循环级方式中,程序员无需控制数据及计算任务在处理机间的划分;而在 SPMD 方式中,程序员不仅需要负责数据及计算任务在处理机间的划分,而且需要管理处理机间的通信。因此一般说来,采用 SPMD 方式编程比循环级方式困难得多。实际上,SPMD 程序类似于消息传递程序。

由于软件 DSM 在机群上构造的共享存储抽象,类似于 NUMA 结构,存取处理机本地内存比远地内存快得多,这就使得实现拥有者计算原则非常重要。由于 OpenMP 标准中没有提供数据分布制导,因此利用循环级编程方式,不能够控制数据分布以及任务划分来实现拥有者计算。而在 SPMD 方式下,用户通过手工划分数据和计算任务,来实现拥有者计算。因此一般来说,在机群 OpenMP 系统中,SPMD 程序要比循环级程序的性能高得多。

## 3 合适的 OpenMP 制导扩展

### 3.1 数据分布制导

OpenMP 标准是针对 UMA 的共享存储结构制

定的,不用考虑数据分布。然而在 NUMA 结构的分布式共享存储系统中,由于访问本地存储器比访问远程存储器快得多,因此数据分布严重影响性能。在硬件实现的 CC-NUMA 系统中几乎都引入了数据分布制导来扩展 OpenMP 标准,例如 SGI<sup>[8]</sup> 和 Compaq<sup>[9]</sup> 的系统等。这些数据分布制导大部分从 HPF<sup>[10]</sup> 中借鉴而来。软件 DSM 系统在机群上提供了类 CC-NUMA 结构的计算平台。在这种松散耦合的计算平台上支持 OpenMP 编程模型,从性能角度出发,更迫切需要显式的数据分布制导。这主要是:一方面适合于硬件 CC-NUMA 结构的 First-touch 隐式数据分布策略在机群平台的多操作系统下实施起来困难较大;另一方面这种松散结构系统远比硬件支持的紧密结构系统更需要发挥数据局部性。我们引入数据分布制导来扩展 OpenMP 标准,使得程序员显式控制数据在结点 Home 中的分布模式。我们在 OpenMP/JIAJIA 中实现了该数据分布制导。下面具体介绍数据分布制导。

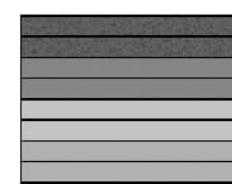
数据分布制导说明被制导的数组对象在各结点 Home 中的分布模式,该制导只能应用于全局共享的数组变量。由于数据分布制导以页为单位且不支持嵌套并行,我们只制导数组的最高维,所以制导语法非常简单:

```
# pragma omp distribute (array_name,layout_mode)
layout_mode := BLOCK|CYCLIC|CYCLIC(chunksize)
```

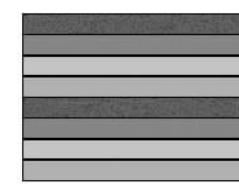
其中, BLOCK 方式是将数组 array\_name 按最高维长度均分在每个处理机上;而 CYCLIC(chunksize) 是将数据最高维按 chunksize 大小以 round-robin 方式在处理机间循环分配(在实现中进行页对齐)。 CYCLIC 则表示以 CYCLIC(1) 方式进行分布。例如:

```
double A[1024][1024],B[1024][1024];
#pragma omp distribute(A,BLOCK)
#pragma omp distribute(B,CYCLIC(128))
```

在上述定义中,数组 A 和 B 的大小都为 8MB,每个数组均占 1024 个页(假设页大小为 8KB)。所以上述制导的数据分布情况如图 1 所示(假设映射到 P 个处理机上,这里 P=4)。



(a) BLOCK 方式



(b) CYCLIC 方式

图 1 采用 distribute 制导的数据分布情况

图 1 中(a)表示数组 A 以最高维按 BLOCK 方式分布在各结点的 Home 中的分布情况,即第 1 块的 256 个页分布在结点 0 上;第 2 块分布在结点 1 上,依次类推. 图 1(b)表示数组 B 以最高维按 CYCLIC 方式分布在各结点的 Home 中的分布情况,每次分配 128 个页,即第 1 个和第 5 个 128 页分布在结点 0 上;第 2 个和第 6 个 128 页分布在结点 1 上,依次类推.

上述数据分布制导易于使用,但是该制导的一个缺陷是以页为划分粒度,这对于制导较小的数组和使用 CYCLIC 分布模式效果不好. 为了解决这个问题,SGI 提供了 Distribute\_reshape 制导来实现数组元素级数据分布而不管页边界问题. 由于这种制导使得编译器可能重新组织数组,使得数组元素索引不再是一种顺序序列,所以使用这种制导有特定的限制. 例如: 初始化数据不能使用该制导; 由 malloc 显式分配且通过指针引用的数组不能使用该制导等等. 另外,HPF<sup>[10]</sup> 提供了 Align 制导,可以用来将两个数组的对应元素映射到同一个结点的存储器中. 这对于提高针对两个数组元素的操作的性能是很有好处的. 为了简单起见,我们没有在 OpenMP/JIAJIA 中实现 distribute\_reshape 制导和 Align 制导.

### 3.2 循环调度模式扩展

OpenMP 提供了丰富的循环调度模式,包括静态调度(static)、动态调度(dynamic)、制导调度(guided)等,然而这些调度模式都只考虑了负载平衡而没考虑数据分布. 数据分布制导使得用户能够控制数据在各结点的 Home 中的分布情况. 结合数据分布制导,我们将描述两种有效利用 Home 中数据局部性的静态调度模式 LBS(Locality-Based Scheduling) 和动态调度模式 LBDS (Locality-Based Dynamic Scheduling).

```
#define N 1024
double A[N][N]
#pragma omp distribute(A,CYCLIC)
#pragma omp parallel
{
    for(i=0;i<N;i++) {
        #pragma omp master
        { for(j=i+1;j<N;j++) A[i][j] /= A[i][i]; }
    }
}
```

图 2 采用 LBS 调度模式的 LU 代码

### LBS 调度模式

LBS 是一种基于局部性的静态调度模式,它针对的目标系统是独占运行的、结点计算能力相同的同构机群. LBS 充分利用数据的 Home 分布信息,使得处理机只计算 Home 分布在自身的数据,实现对 Home 中数据的拥有者计算. 我们扩展 OpenMP 的 schedule 子句以实现 LBS,其语法格式为

```
LBS_schedule_clause = schedule(LBS,LBS_target)
LBS_target := array_name | array_name([LBS_subscript])*
LBS_subscript := expr | *
```

其中 expr 的表达式必需是  $scale * identifier + offset$  的格式, identifier 为循环控制变量, scale 和 offset 为循环不变量. 实际上, expr 表达的是循环迭代与该迭代所计算的该维数据之间的映射关系; \* 则表示循环迭代计算该维的所有数据. 目前我们的实现仅支持 scale 为 1 且 expr 为最高维的情况. LBS 子句说明按照数组 array\_name 在处理机 Home 中的分布情况进行基于局部性的调度.

LBS 的实现思路是利用数据分布制导来获得数据的初始 Home 分布; 在进行循环调度时,利用数据的 Home 分布信息来进行基于局部性的循环调度,即如果循环迭代  $i$  所计算的数据分布在处理机  $j$  的 Home 中,就调度处理机  $j$  执行迭代  $i$ . LBS 是一种体现数据局部性优先的静态调度模式,这样对于数据分布不平衡的应用就可能造成负载不平衡,用户可以通过数据分布制导语句中采用最佳的数据分布粒度来解决. 图 2 是在 LU 程序中使用 LBS 调度模式的例子. 在本例中,数组 A 是以 CYCLIC(1) 模式在处理机间分布,采用 LBS 调度模式可以很好实现拥有者计算.

```
# pragma omp for schedule(LBS,A[j]*[])
for (j=i+1;j<N;j++) {
    for (k=i+1;k<N;k++) {
        A[j][k] -= A[i][k] * A[j][i];
    }
}
//end of for
//end of parallel
```

量实现对 Home 中数据的拥有者计算. 我们扩展 OpenMP 的 schedule 子句以实现 LBDS,其语法格式为

```
LBDS_schedule_clause = schedule(LBDS,LBDS_target)
```

### LBDS 调度模式

LBDS 是一种基于局部性的动态调度模式,主要适用于非独占的元计算环境以及应用负载不平衡的情况. LBDS 的基本思想和 LBS 是一样的,即尽

```

get,granularity)
LBDS_target := array_name | array_name([LBDS-
    subscript])*
LBDS_subscript := expr| *

```

其中 *expr* 的表达式必需是 *scale \* identifier + offset* 的格式, 其中 *identifier* 为循环控制变量, *scale* 和 *offset* 为循环不变的整数量. LBDS 子句说明按照数组 *array\_name* 的 Home 数据分布来进行基于局部性的动态调度, 并且将循环迭代按任务粒度 *granularity* 大小来分配给各个线程.

LBDS 的实现包括三步: 静态分配、本地调度、远程调度. LBDS 所采用的是一个全局共享循环迭代任务队列, 但这个队列的组织形式与传统的集中式队列不同. 它根据处理机数目被划分成 *P* 个段, 其中第 *i* 个段分配给第 *i* 个处理机. 队列中各段和处理机间的对应关系由数据在处理机间的初始 Home 分布以及循环迭代与该迭代所计算数据之间的映射关系来决定. 在本地调度时, 处理机只获得那些计算数据在本地 Home 区域中的迭代进行计算. 当负载不平衡时, 空闲处理机进行远程调度, 通过任务队列找到最忙的处理机并从它的任务段中取出迭代进行计算. 具体步骤如下:

1. 静态分配. 根据循环的下边界、上边界和步长、循环迭代与所计算数据之间的映射关系以及数据的 Home 分布模式(由 distribute 制导指定)进行循环调度共享任务队列的初始化;

2. 本地调度. 每个处理机从属于自己的任务段中, 取出 *granularity* 个迭代进行计算, 一直到自己的任务段为空; 如果最后一次, 自己的任务段中剩下不足 *granularity* 个迭代, 则全部取出;

3. 远程调度. 当处理机自己的任务段为空时, 找到共享队列中余下任务最多的任务段(即将该任务段所对应的处理机作为最忙处理机), 从该任务段中取出 *granularity* 个迭代进行计算. 一直到共享任务队列为空时, 停止调度.

图 3 是采用 LBDS 调度模式的例子. 该调度子句表明循环中第 *i* 个迭代计算数组 *A* 的第 *i*+1 行; 并且分配任务粒度为 10.

```

#pragma omp for schedule(LBDS,A[i+1][*],10)
for(i=0;i<100;i++)
for(j=0;j<100;j++)
A[i+1][j]=i+j;

```

图 3 采用 LBDS 调度模式的例子

对于操纵单个数组的循环而言, 如何采用 LBS 和 LBDS 是很显然的. 对于操纵多个数组的循环而言, 用户需要从最大化拥有者计算角度来选取其中一个数组作为基准数组, 并依此来采用 LBS 或者 LBDS 调度模式. 为了获得高性能, 需要使其它数组与基准数组的数据分布相同或满足某种关系. 这可以通过对各数组精心采用 distribute 制导来部分实现. 当然在 HPF 中, 采用 Align 制导会更加方便. 由于在 OpenMP/JIAJIA 中没有实现 Align 制导, 我们只能采用 distribute 制导来部分解决该问题.

## 4 性能评价

我们在 OpenMP/JIAJIA 中实现了数据分布制导扩展和循环调度模式. 本文的测试平台为 8 个结点的机群, 每个结点有两个 Intel PIII 700MHz 的处理器、主存为 1GB, 结点间以 100M 快速以太网互连, 主板是 SuperMicro 公司的 DLE. 结点操作系统为 Redhat 7.3(Linux 内核版本为 2.4.18).

本文使用了一些被广泛采用的测试程序, 包括 SPEC OMP2001 的浅水模拟程序 SWIM<sup>[11]</sup>、Rice 大学 OpenMP on 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 以及我们自己编写的非分块 LU 分解程序. 我们分别采用 SPMD 方式以及数据分布制导方式重写了这些程序, 即程序的 SPMD 版本和 DIST 版本. 表 1 列出了每个应用程序的特性以及串行执行时间. 从表 1 中可知, SPMD 版本的共享空间较小, 它主要是用作通信缓冲区, 而不是用来存储计算数据.

表 1 程序特征及串行执行时间

程序	规模	共享内存		Barrier 数		串行执行时间(s)
		SPMD	DIST	SPMD	DIST	
SOR	4096×4096(100iter)	80KB	128MB	205	204	133.15
SWIM	1335×1335 (120iter)	400KB	195MB	2405	1925	472.32
LU	2048×2048	16KB	32MB	2051	4099	230.10
GS	2048×2048	8KB	16MB	2053	4099	336.50

我们测试了所有应用在 2,4,8,16 个进程时的并行执行情况。表 2 即为程序的 SPMD 和 DIST 版本的执行情况,包括执行时间、Barrier 时间和通信量。从表 2 中可知,除了 SWIM 之外的所有程序的 SPMD 版本的通信量都比 DIST 版本要小,这主要是由于 SPMD 方式几乎将数据完全私有化,尽量减少通信。对于 SWIM 的 SPMD 版本而言,由于在并行循环执行前和执行后,一个处理机都需要与相邻处理机进行通信,因此 SWIM 的 SPMD 版本通信量较大,超过了 DIST 版本。另外,SOR 的 SPMD 版本在 16 个进程时的通信量比 DIST 版本大,这主要是由于 JIAJIA 采用的一些通信优化措施造成的。

一般认为程序的 SPMD 版本的 Barrier 数应该

比 DIST 版本少,这是因为在 SPMD 版本中程序员可以通过精心安排使 Barrier 数减到最少。而在 DIST 版本中,由于 Barrier 隐含在制导(如 parallel, for)实现中,难于消去。但是由于在 SPMD 版本中,当利用共享缓冲区进行通信时需要利用 Barrier 实现同步,因此程序的 SPMD 版本可能会比 DIST 版本的 Barrier 更多。实际应用表明,两种方式的 Barrier 数目实际上取决于程序的情况。例如,SWIM 的 SPMD 版本的 Barrier 数就比 DIST 版本多;SOR 基本持平,LU,GS 则有明显减少。从表 2 中可看出 Barrier 数目对程序的可扩展性影响较大。例如 LU、GS 程序在 16 个进程情况下,SPMD 版本的性能比 DIST 版本有较大改进(LU 为 16%,GS 为 14%)。

表 2 测试结果

程序	2 个进程时的测试结果			4 个进程时的测试结果			8 个进程时的测试结果			16 个进程时的测试结果		
	时间(s)	Barrier (s)	消息(MB)	时间(s)	Barrier (s)	消息(MB)	时间(s)	Barrier (s)	消息(MB)	时间(s)	Barrier (s)	消息(MB)
SOR	SPMD	68.65	0.53	5.9	36.83	2.34	20.4	22.60	2.60	58.7	16.76	3.41
	DIST	69.08	1.56	37.6	39.96	2.15	60.2	20.65	3.04	79.6	14.37	3.66
SWIM	SPMD	244.25	10.14	43.3	155.10	28.16	139.2	125.43	41.24	303.9	101.34	45.87
	DIST	263.06	20.06	51.3	160.96	26.79	117.9	112.95	34.63	247.8	84.50	32.25
LU	SPMD	124.33	8.14	19.6	76.67	15.72	61.5	80.05	24.13	147.1	88.20	30.01
	DIST	130.87	14.79	30.6	93.16	24.98	78.3	89.21	33.87	166.9	102.60	45.23
GS	SPMD	172.46	5.20	17.4	100.96	11.37	52.0	76.72	19.22	120.2	65.24	22.71
	DIST	177.93	7.12	34.5	111.93	20.57	77.5	84.40	26.18	150.6	74.40	28.30

注:2,4,8 台处理器是在每个结点上运行一个进程;16 台处理器是在每个结点上运行两个进程。

另外,从表 2 中可看出 LU 和 GS 加速比不好,这主要取决于程序本身的特性,跟采用何种编程方式关系不大。LU 和 GS 程序都具有单生产者/多消费者的访存特点,且负载也不平衡,这对 LU 和 GS 的扩展性影响很大。此外,由于本文的 16 个进程时的数据是在每个结点上运行两个进程,尽管我们机群的结点是 SMP 结构,但是当两个 CPU 上运行的进程都具有访存密集特点时,会造成严重内存总线竞争,影响程序性能。这也会影响到本文的 16 进程时执行时间的准确性。

总之,大部分测试程序的 SPMD 版本的执行时间比 DIST 版本有改进,但改进不大(<17%)。程序性能改进不多的主要原因是:由于程序的访存比较规则,利用 distribute 制导和基于局部性的调度模式,这些程序基本上都能较好地实现拥有者计算。对于这些程序而言,通信已经不再是性能瓶颈,尽管采用 SPMD 方式能进一步减少通信量,但是对提高性能作用不大。

## 5 相关工作

自从 OpenMP 标准于 1997 年发布以来,扩充 OpenMP 以有效支持 NUMA 结构成为研究热点。尽管 OpenMP ARB 还没有发布分布式 OpenMP 的计划,但是许多公司在自己的商业 OpenMP 实现中都利用数据分布制导扩展来支持 NUMA 结构,例如 SGI<sup>[8]</sup>, Compaq<sup>[9]</sup>, PGI<sup>[12]</sup>, KAP 等。文献[8]中描述了在 SGI Origin2000 的 MIPSPro 编译器中实现了数据分布制导扩展 distribute 以及相应的循环调度模式 Affinity。针对数据分布制导扩展,文献[8]中不仅提出了以存储页粒度来进行数组分布的 distribute 制导,而且提出了按照数组元素来进行分布的 Distribute\_reshape 制导。文献[8]中提出 Affinity 调度模式类似于本文的 LBS,但是没有提出类似于 LBDS 的动态调度模式。Compaq<sup>[9]</sup>的制导扩展实现了 distribute 数据分布制导,并采用 ON HOME 和

OMP NUMA 制导结合来实现 Affinity 调度模式功能,而且还进一步实现了 Align 制导.

另外,也有研究者认为不对 OpenMP 标准进行显式数据分布扩展,也可以在 NUMA 结构上有效实现,例如文献[13]中提出了一种通过在数据和处理线程之间隐式建立亲和关系以便进行循环调度时使用的方法,用户无需关心数据分布亦可提高性能.但是采用该方法,还是需要对循环调度制导进行扩展,且其效率受限于应用模式,对于非规则应用程序开销较大.另外,该文工作是在 SGI Origin2000 机器上做的,所提出的方法需要同 First-touch 页放置策略结合起来,而该策略在软件 DSM 系统上难于实现.

目前国际上还有其他一些研究组进行机群 OpenMP 系统的研究.日本的 Omni 项目组<sup>[2]</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 支持数据分布和循环调度,但是在调度的实现中没有很好地权衡数据分布和负载平衡的关系,性能不够理想;美国 Rice 大学的 Lu 等人<sup>[1]</sup>基于软件 DSM 系统 TreadMarks 实现了一个 OpenMP 子集,但是没有针对机群特点进一步优化,在其实现中没有扩充数据分布制导,仅实现了有限的拥有者计算调度;其他的研究组,例如瑞典的 Odin 项目组<sup>[3]</sup>、美国 Houston 大学和 Purdue 大学也在从事相关研究,但未见在公开文献发表研究成果.

## 6 结论和未来工作

由于 OpenMP 标准是针对 UMA 结构制定的,不太适合 NUMA 结构的机器,特别是对于通信开销大的机群系统.如果不进行制导扩展,要想在机群 OpenMP 系统上获得高性能的 OpenMP 程序,就必须采用 SPMD 方式编程.然而与循环级编程方式相比,SPMD 方式编程困难,这就违背了制定 OpenMP 标准的初衷.本文描述了一个简单制导扩展子集(包括数据分布制导、循环调度模式),并在机群 OpenMP 系统 OpenMP/JIAJIA 中实现.对于访存规则的程序,利用 distribute 制导扩展编程,不仅获得了与 SPMD 方式相当的性能,而且其易编程性与循环级编程方式类似.可见,利用 distribute 制导扩展进行编程是机群 OpenMP 系统中一种有效的编程方式.

从本文中可以看出,机群 OpenMP 系统在将

OpenMP 程序转换成软件 DSM 程序后引入的 Barrier 数对程序的可扩放性影响较大,因此如何引入尽量少的 Barrier 以及如何减少 Barrier 操作的执行时间,是我们未来工作的重点.我们的未来工作还包括克服目前以页为粒度进行数据分布的限制.由于数据分布制导只适用于访存规则的程序,对于非规则应用不是很有效.我们未来的工作也包括研究适合非规则应用的制导扩展、提高 OpenMP/JIAJIA 性能的编译优化以及专门硬件支持的技术.

## 参 考 文 献

- 1 Lu H., Hu Y.C., Zwaenepoel W.. OpenMP on networks of workstations. In: Proceedings of the Supercomputing'98, Orlando, USA, 1998, 1~15
- 2 Sato M., Sato S., Kusano K., Tanaka Y.. Design of OpenMP compiler for an SMP cluster. In: Proceedings of the 1st European Workshop on OpenMP, Lund, Sweden, 1999, 32~39
- 3 Brunschen C., Brorsson M.. OdinMP/CCP——A portable implementation of OpenMP for C. Concurrency and Computation: Practice and Experience, 2000, 12(12): 1193~1203
- 4 Krawezik G., Alleon G. et al.. SPMD OpenMP versus MPI on a IBM SMP for 3 kernels of the NAS Benchmarks. In: Proceedings of the 4th International Symposium on High Performance Computing, Kansai Science City, Japan, 2002, 425~436
- 5 OpenMP Architecture Review Board. OpenMP C and C++ Application Program Interface, Version 2.0, March 2002. <http://www.openmp.org/>
- 6 Hu W., Shi W., Tang Z.. JIAJIA: A software DSM system based on a new cache coherence protocol. In: Proceedings of HPCN Europe'99, Amsterdam, 1999, 463~472
- 7 Tang Zhi-Min, Shi Wei-Song, Hu Wei-Wu. Message-passing versus shared-memory on dawning 1000A. Chinese Journal of Computers, 2000, 23(2): 134~140(in Chinese)  
(唐志敏,施巍松,胡伟武.曙光 1000A 上消息传递和共享存储的比较.计算机学报, 2000, 23(2): 134~140)
- 8 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
- 9 Bircsak J., Craig P., Crowell R. et al.. Extending OpenMP for NUMA machines. Scientific Programming, 2000, 8(3): 163~181
- 10 High Performance Fortran Forum. High Performance Fortran Language Specification, Version 2.0, January 1997
- 11 Aslot V., Domeika M., Eigenmann R., et al.. SPEComp: A new benchmark suite for measuring parallel computer performance. In: Proceedings of the Workshop on OpenMP Application and Tools, West Lafayette, USA, 2001, 1~10

- 12 Merlin J., Miles D., Schuster V.. Distributed OpenMP: Extensions to OpenMP for SMP clusters. In: Proceedings of the 2nd European Workshop on OpenMP, Edinburgh, 2000, 48~58
- 13 Nikolopoulos D., Artiaga E., Ayguadé E., Labarta J.. Exploiting memory affinity in OpenMP through schedule reuse. ACM SIGARCH Computer Architecture News, 2001, 29(5): 49~55



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

**WU Shao-Gang**, born in 1973, Ph. D. candidate, lectu-

er. His research interests include system architecture, software distributed shared memory system, parallel computing.

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

**HU Wei-Wu**, born in 1968. Ph. D., Ph. D. supervisor. His research interests include high performance computer architecture, parallel processing and VLSI design.

## Background

The project's name is "Research on OpenMP towards Cluster Systems". This project is supported by the National Natural Science Foundation of China under grant no 60303016. This project aims to research on some techniques to implement a high performance OpenMP computing environment on clusters which enable users to develop OpenMP applications on clusters. The group have developed an

OpenMP system based on cluster named OpenMP/JIAJIA. In this paper, some directive extensions have been illustrated to get high performance on OpenMP/JIAJIA. Now, the group is studying some other techniques to improve the performance of OpenMP on cluster, such as compiler optimization, special hardware support, etc.