第 34 卷 第 6 期 计 算 机 工 程 2008 年 3 月
Vol.34 No.6 Computer Engineering March 2008
·软件技术与数据库· 文章编号：1000—3428(2008)06—0080—02 文献标识码：A 中图分类号：TP311

有限元单元计算子程序的 OpenMP 并行化

宋 刚[1,2,3]，蒋孟奇[1,2,3]，张云泉[2,3]，李玉成[2]
(1. 中国科学院研究生院，北京 100080；2. 中国科学院软件研究所并行计算实验室，北京 100080；
3. 中国科学院计算机科学国家重点实验室，北京 100080)

摘 要：Intel 和 AMD 双核乃至 4 核处理器的推出，使得并行计算已经普及到 PC 机。为了充分利用多核，需要对原有程序进行多线程改造，使其充分利用多核处理带来的性能提升。该文利用共享存储编程的工业标准 OpenMP 对有限元方法涉及的单元计算子程序进行了并行化实现。在机群的一个双 CPU 的 SMP 节点上的测试表明，共享并行化使得该单元子程序的性能提高了一倍。
关键词：并行编程；多线程；多核；有限元

OpenMP Parallelization of Element Computation Subroutine of Finite Element Method

SONG Gang[1,2,3], JIANG Meng-qi[1,2,3], ZHANG Yun-quan[2,3], LI Yu-cheng[2]
(1. Graduate University of Chinese Academy of Sciences, Beijing 100080; 2. Parallel Computing Lab, Institute of Software, Chinese Academy of Sciences, Beijing 100080; 3. State Key Lab of Computer Science, Chinese Academy of Sciences, Beijing 100080)

【Abstract】 With the release of the Intel and AMD's dual-core and muti-core processor, parallel computing becomes pervasive. To make effective use of the muti-core processor, the original sequential program has to be parallelized. In this paper, the OpenMP is used to parallelize the element computation subroutine of finite element method. And the test on a SMP node with dual-CPU of a cluster shows that the performance is improved almost twice compared with the original sequential program.
【Key words】 parallel programming; multithread; multi-core; finite element

计算数学专家梁国平先生是飞箭软件创始人之一，于 1990 年研制成功有限元程序自动生成系统(Finite Element Program Generator, FEPG)，1995 年获国家科技进步二等奖。FEPG采用元件化编程思想和有限元语言的表达方式，为各领域有限元问题的求解提供了一个有利的工具[1]。本文针对有限元程序中的单元计算子程序E和E程序调用的A程序进行并行化实现，在机群的一个SMP节点上的测试表明，性能得到了大幅度的提升。

1 OpenMP

OpenMP[2]标准是共享存储体系结构上的一个编程模型，是由一组硬件厂商与软件供应商联合制定的，详细情况可以参考http://www.openmp.org。现在已经有许多硬件和软件供应商提供支持OpenMP的编译器，如Intel, HP, Sun, SGI和IBM等，且包括Unix和Windows NT两种操作系统平台。当前计算机体系结构的发展朝着更有利于OpenMP的方向发展，PC机方面，Intel和AMD双核以致 4 核处理器的推出，人们充分利用多核的优势来为自己服务；在高性能计算领域，IBM的“片上多核”技术的不断成熟，科学计算领域更要充分利用片上多核来提高大规模数学计算问题求解的速度。OpenMP强大的生命力使编程者有理由在共享存储的环境下优先采用OpenMP编程。OpenMP具有使用简单、增量并行、对于中等规模问题获得加速比相对容易的优点，并且处理器数目不断增加的时候更容易实现移植[3]。

OpenMP包括一套编译制导语句和一个用来支持它的函数库，这些编译制导语句和函数库用于为共享存储计算机创建并行程序[4]。OpenMP是通过与标准C, C++和Fortran语言结合来工作的。为了叙述的简洁，本文将使用OpenMP一小部分编译制导语句来介绍OpenMP并行化过程，代码示例采用Fortran语言。

在 OpenMP 中，编译制导指令在每一行以 C$OMP 或!$OMP 开头。编译制导指令 parallel 和 end parallel 定义了一个所谓的并行域。一旦进入一个这样的并行域，就会生成另外的一些线程来并行执行这段代码。除非有一些其他的制导指令，否则对于每一个线程都会执行并行域中的所有代码。有许多方法可以控制线程的行为，具体见文献[5]。进入并行域生成的线程会在遇到 end parallel 的时候结束。在并行域里面，线程不仅可以执行相同的程序代码，还可以分担繁重的计算任务，尤其是 loop 循环。如果一个循环被制导指令 do 和 end do 包含，那么它的迭代将根据不同的分配策略分配到不同的线程中去执行。也就是说这两个制导指令可以合并在一起成为一对 parallel do 和 end parallel do 结构。OpenMP 并行循环示例代码如下：

```
C$OMP PARALLEL DO PRIVATE(I, tmp)
```

基金项目：国家自然科学基金资助项目(60303020)；国家自然科学基金资助重点项目(60533020)；国家“973”计划基金资助项目(2005CB321702)；国家“863”计划基金资助项目(2006AA01A102, 2006AA01A125)；北京邮电大学网络与交换技术国家重点实验室开放课题基金资助项目(2005-05)
作者简介：宋 刚(1982－)，男，硕士研究生，主研方向：并行编程，大规模科学与工程计算的方法与软件；蒋孟奇，硕士研究生；张云泉，副研究员、博士；李玉成，研究员
收稿日期：2007-03-30 **E-mail：**hevensun@126.com

```
C$OMP&SHARED(x, y, z)
      do i = 1,L
        temp = sin(z(i))
        x(i) = y(i) * tmp
      end do
C$OMP END PARALLEL DO
```

在共享编程的环境下，区分变量是所有线程共享还是每个线程都有一个私有拷贝很重要。例如在一个循环中，循环计数器必须为私有，因为每一个线程都计算不同的迭代，而循环计数的界限值因为对每个线程都一样，所以可以共享。默认的情况下，所有变量都为共享，OpenMP 制导指令可以设置哪些变量共享，哪些变量私有。

为了更好地理解上述内容，考虑 OpenMP 并行循环示例代码，变量 *x, y* 和 *z* 是长度为 *L* 的数组。制导指令 parallel do 表明循环的不同迭代可以被分配到不同线程中并行执行。在这个例子中，每个线程都可以访问数组 *x, y* 和 *z*，但每个线程都使用这些数组自己的部分。语句 shared 被用来标明需要共享的变量 *x, y* 和 *z*，语句 private 标明每个线程对变量 *i* 和 temp 都有自己的一份拷贝。

2 单元计算子程序的并行化

由于飞箭软件公司的有限元单元计算子程序都是用 Fortran 语言编写的，因此以下都是以 OpenMP＋Fortran 方式介绍并行化过程的。单元计算子程序的主要程序结构是一个循环结构，并且循环里面调用的一个子程序大约占到程序执行时间的 80%，所以，只要将这个循环并行尤其是其中的子程序并行，性能就应该会有很大的提高，测试数据验证了这一推断的正确。以下分别介绍程序并行化过程涉及的问题以及处理方法。

2.1 区分共享与私有变量

OpenMP 编程的难点之一就是并行化过程中区分共享变量与私有变量。由于单元计算子程序并行化操作主要针对 DO FOR 循环，因此区分好循环体中的变量至关重要，基本策略就是对于只有读操作的变量可以设为共享，而有写回操作的变量一般应设为私有，即对循环体中出现的每个变量都进行读写分析。下面是经过分析得到的共享与私有变量，并且已经用编译制导语句表示出来：

```
C$OMP PARALLEL DO
C$OMP&PRIVATE(NE,J,I,INOD,IDGF,JNOD,R,
C$OMP&nr,imate,prmt,LM,es,em,ef,estifv)
C$OMP&SHARED(U,A,NA,NUMCOL,NODVAR,
C$OMP&EU,knode,kdgof,kcoor,kelem,k,kk)
C$OMP&DEFAULT(private)
```

其中，PRIVATE 指定的为私有变量，即每个线程都私有；SHARED 指定的为共享变量，所有线程共享这些变量；DEFAULT 指定其他所有未指定的变量都是私有的。PRIVATE 也可以省略，因为 DEFALT 语句已经指定默认的都为私有，但是为了让读者看得更清晰，所以全部列了出来。

2.2 SAVE 的使用

对于主程序中的一个数组 sml，单元计算子程序调用的 A 程序都是利用这个数组来返回值，串行程序中不存在问题，并行程序中如果 A 程序被并行执行，若不经过任何处理，A 程序的返回值就可能被别的线程冲掉，所以，在主程序中将这个数组私有化，对于所有的线程来说，每一个线程都有一个这样的数组空间，私有化的时候需要将这个数组增加 save 属性，如下所示：

```
dimension sml(100000)
save sml
C$OMP threadprivate(sml)
```

2.3 临界区的处理

对于循环体中的共享变量 *U* 存在重新计算以后更新值的问题，如果一个线程计算完其值以后还没来得及更新，这时候被其他线程重新计算了，原先计算的结果就被冲掉了，所以将值的计算与更新这段代码设置了临界区，使这段代码同一时刻只能有一个线程执行。且这段代码的执行并不会占用太多的时间，所以对性能影响不是很大。加入临界区后的代码如下：

```
C$OMP   CRITICAL (auu)
C     //省略若干行
          U(IDGF,NODI)=U(IDGF,NODI)+ef(i)
C     //省略若干行
          IF (INV.LT.0)
              U(JDGF,NODJ) = U(JDGF,NODJ)
         &-ESTIFN(J,I)*U(IDGF,NODI)
C     //省略若干行
C$OMP END CRITICAL (auu)
```

2.4 common 块的处理

COMMON 块是 Fortran77 中使用“全局变量”的方法，它用来定义一块共享内存空间。可以通过全局变量的使用让不同程序之间或是主程序跟函数之间使用相同的内存位置。这也是一种在不同程序间传递参数的方法。在单元计算子程序循环中调用的 A 程序里有许多 common 块，如果对各个线程都共享这些 common 块，显然对于写回 common 块的操作就会发生冲突，所以把 A 程序中以及 A 程序调用的子程序中出现的 common 块进行私有化。子程序中 common 块的处理如下：

```
common /raec8/ru(8,32),cu(8,4)
common /vaec8/rctr(3,3),crtr(3,3)
common /daec8/ refc(3,8),gaus(8)
C$OMP threadprivate(/raec8/,/vaec8/,
C$OMP&/daec8/)
```

2.5 库函数的修改

由于循环中调用的 A 程序所调用的子程序对库(飞箭软件公司编译生成的)中函数定义的一些 common 具有写回操作，因此对库中并行化涉及到的这些 common 块进行私有化后重新编译链接。对库中 common 块处理如下：

```
common /coord/ coor(3),coora(27,3)
C$OMP threadprivate(/coord/)
```

通过以上一系列的处理，得到单元计算子程序的共享并行版本。

3 性能测试与分析

3.1 “清华探索 3 号”测试平台机群

“清华探索 3 号”每个节点采用 2 个 Itanium 2 的 64 位处理器，峰值速度达 1.331 TFlop/s，Linpack 持续性能达到 1.126 TFlop/s；其内存为 514 GB，存储能力为 13 TB，可提供大规模科学计算服务。操作系统为 Redhat Linux As3.0 ia64，编译器选择 Intel icc8.0。由于使用的是共享存储编程模式，因此使用其中的一个 SMP 节点(双 CPU)进行性能测试。

3.2 测试数据与性能分析

测试由飞箭软件公司工程师帮助提供，测试数据来源于

(下转第 84 页)