装订线 答题时不要超过此线

中 国 科 学 技 术 大 学

2013－2014 学年第一学期考试试卷

考试科目：并行程序设计　　得分：_____

学生所在系：_____　姓名：_____　学号：_____

一、 描述以下循环 1 和循环 2 中的依赖关系；并尝试通过循环交换、分布和逆转等多种方法来尝试向量化和/或并行化变换：（2×15=30 分）

参考解答：

```
for i = 2 to N+1 do
   for j = 2 to M+1 do
      for k = 1 to L  do
         A[i,j,k] = A[i,j-1,k+1]+A[i-1,j,k+1]
      endfor
   endfor
endfor     // 循环 1，其中 N、M 和 L 均为常数
```

循环 1 的依赖分析：

显然，三重循环中存在依赖向量为(0,1,-1)和(1,0,-1)的依赖关系，可知最外层的循环 i 和中间层循环 j 均不可并行化，最内层循环 k 可以向量化或并行化，只是并行化的粒度不大。

可以考虑针对最内层循环 k 的逆转，接着再与最外层进行循环交换，则可得依赖向量为(1，1，0)和(1,0,1)。如此一来，尽管最外层循环（k）不能并行，但是中间层循环 j 却因此可以并行化，而最内层循环（i）可以并行化或向量化。程序变换结果如下：

```
for k = L downto 1 do
   for j = 2 to M+1 para-do
      for  i = 2 to N+1 para-do   //最内层并行化
        A[i,j,k] = A[i,j-1,k+1]+A[i-1,j,k+1]
      endfor
   endfor
endfor
```

最内层向量化如下：

```
for k = L downto 1 do
   for j = 2 to M+1 para-do
     A[2: N+1,j,k] = A[2: N+1,j-1,k+1]+A[1: N,j,k+1]
   endfor
endfor
```

```
for i = 1 to N do
   for j = 1 to M  do
      S:  A[i, j]    =  A[i, j]   + X;
      T:  B[i+1, j] =  A[i, j]   + B[i, j];
      U:  C[i, j+1] =  A[i, j]   + C[i, j];
      V:  D[i+1, j] =  B[i+1, j] + C[i, j] + D[i, j];
   endfor
endfor  //循环 2  其中 N、M 均为常数
```

循环 2 的依赖分析：

显然，二重循环中存在如下依赖关系：

S δ^f T，依赖向量(0,0)，关于数组 A 的。

S δ^f U，依赖向量(0,0)，关于数组 A 的。

T δ^f T，依赖向量(1,0)，关于数组 B 的。

T δ^f V，依赖向量(0,0)，关于数组 B 的。

U δ^f U，依赖向量(0,1)，关于数组 C 的。//内层循环不能向量化

U δ^f V，依赖向量(0,1)，关于数组 C 的。

V δ^f V，依赖向量(1,0)，关于数组 D 的。

上述依赖关系的存在，我们知道外层循环 i 不能并行，内层循环 j 也不能并行或向量化！

可以考虑循环分布变换！

先考虑内层循环中的语句依赖图（图略）。可知，S 最先执行，包含语句 S 的外层或内层循环可并行化，或内层向量化均可以。而 T 和 U 可以同时进行--

其中，包含语句 T 的内层循环可以向量化或并行化，而包含语句 U 的外层循环可以并行化。在 T 和 U 执行完成后，最后再执行语句 V，包含语句 V 的内层循环可以向量化或并行化。

程序变换结果如下：（仅给出并行化版本）

```
for i = 1 to N para-do
    for j = 1 to M  para-do
    S:  A[i, j]    =  A[i, j]   + X;
    endfor
endfor

for i = 1 to N do
    for j = 1 to M  para-do // 该内层循环可向量化
    T:  B[i+1, j] =  A[i, j]   + B[i, j];
    endfor
endfor

for i = 1 to N para-do
    for j = 1 to M  do
    U:  C[i, j+1] =  A[i, j]   + C[i, j];
    endfor
endfor

for i = 1 to N do
    for j = 1 to M  para-do  //该内层循环可向量化
   V:   D[i+1, j] =  B[i+1, j] + C[i, j] + D[i, j];
   endfor
endfor
```

二、 用基本的 MPI_Send 和 MPI_Recv 函数实现 MPI_AlltoAll 函数功能，并简评你的实现方法。（20 分）

参考解答：

MPI_Alltoall 函数完成数据全交换；类似矩阵转置效果。注意，实现时应避免通讯上的死锁。程序略。

三、 设有两个进程 A 和 B，以及 N×N 的双精度数矩阵 C。现在，进程 A 将矩阵 C 的某一列数据发送给进程 B。请用三种不同的 MPI 实现方式来完成发送操作。（30 分）

参考解答：

三种实现方式如下：

（1） 给出循环，直接使用 MPI_Send 逐个发送列数据；
（2） 采用 MPI_Type_vector 定义列消息数据类型，完成整列数据的发送；
（3） 使用 MPI_Pack 等打包函数，先将列数据完整打包至缓冲区，再行发送。
程序略！

四、 矩阵相乘的另一种并行算法是 Fox 算法：将待相乘的矩阵 A 和 B 分成 p 个方块 $A_{i,j}$ 和 $B_{i,j}(0 \leqslant i,j \leqslant \sqrt{p}-1)$,每块大小为$(n/\sqrt{p}) \times (n/\sqrt{p})$,并将它们分配给 $\sqrt{p} \times \sqrt{p}$ 个处理器$(P_{0,0},P_{0,1}, \cdots ,P_{\sqrt{p}-1,\sqrt{p}-1})$。开始时处理器 $P_{i,j}$ 存放有块 $A_{i,j}$ 和 $B_{i,j}$，并负责计算块 $C_{i,j}$。然后 Fox 算法执行以下各步总计 $\sqrt{p}$ 次迭代即可完成：

① 选中对角块 $A_{i,i}$ 并将其向所在行的 $\sqrt{p}-1$ 个处理器进行一到多播送；
② 各处理器将所收到的 A 阵的块与 B 阵原有的块进行乘–加运算；
③ B 阵的块向上循环 1 步；
④ 如果 $A_{i,j}$ 是本次播送的块，则下次应选块 $A_{i,(j+1) \bmod \sqrt{p}}$ 向同行的 $\sqrt{p}-1$ 个处理器播送，然后转第②步。

请写出 Fox 算法的 MPI 并行程序实现。你的程序必须**重点**体现上述①～④步的通信与计算。（20 分）

参考解答：

注意给出的实现须包含：
(1) 一维的进程 rank 与二维的进程行列位置(row,col)之间的转换函数；
(2) 按行来划分子通信域以及在该子通信域内的广播；
(3) 最好使用 MPI_Sendrecv 来循环收发矩阵子块。

具体程序略！