装订线 答题时不要超过此线

中 国 科 学 技 术 大 学

2016－2017 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、 (1)分析以下循环中的依赖关系：（3×10=30 分）

```
for i = 2 to 10 do   //循环 1
   for j = 2 to 10
      A[i,j] =  ( A[i-1,j-1] + A[i+1,j+1] )* 0.5;
   endfor
endfor
```

```
for i = 2 to 20 do   // 循环 2
   A[2*i+2] = A[2*i-2] + B[i];
endfor
```

```
for i = 2 to 20 do  // 循环 3
    if A[i] > 0 then
       B[i] = C[i-1] + 1
    else
       C[i] = B[i] – 1
    endif
endfor
```

(2) 给出以上循环 2 的 OpenMP 并行实现（10 分）

二、 设有 N*N 个进程按照 N*N 二维拓扑排列。给出 MPI 代码片段，实现（1）进程 rank 与其二维坐标(i,j)的转换；（2）将所有进程按照主对角线（平行）方向来分组；（3）在主对角线组内，进行 rank 值的循环传送。（25 分）

三、 假设线程 A 和线程 B 轮流输出。首先线程 A 输出 1，2，3；然后线程 B 输出 4，5，6；最后线程 A 输出 7，8，9。给出相应的 OpenMP 实现。（10 分）

四、 设有 N 个 MPI 进程，每进程有 M*N 双精度浮点矩阵。给出 MPI 代码，实现 N 个进程轮流按行广播其矩阵数据。（10 分）

五、 乘幂法求解矩阵最大特征值的并行算法：

乘幂法求矩阵特征值由反复进行矩阵向量相乘来实现，因而可以采用矩阵向量相乘的数据划分方法。设处理器个数为 ***p***，对矩阵 ***A*** 按行划分为 ***p*** 块，每块含有连续的 ***m*** 行向量，这里 $m=\lceil n/p \rceil$，编号为 ***i*** 的处理器含有 ***A*** 的第 ***im*** 至第(***i***+1)***m***-1 行数据，(***i***=0,1, …,***p***-1)，初始向量 ***v*** 被广播给所有处理器。具体算法框架描述如下：

输入：系数矩阵 $A_{n\times n}$，初始向量 $v_{n\times 1}$，ε

输出：最大的特征值 ***max***

Begin

对所有处理器 **my_rank(my_rank=0,…, *p*-1)**同时执行如下的算法:

while (|diff|>*ε*) do /* **diff** 为特征向量的各个分量新旧值之差的最大值*/

(1) for *i*=0 to *m*-1 do /*对所存的行计算特征向量的相应分量*/

(1.1) *sum*=0

(1.2) for *j*= 0 to *n*-1 do *sum*=*sum*+*a*[*i*,*j*]*x*[*j*] end for

(1.3) *b*[*i*]= *sum*

end for

(2) *localmax*=|*b*[0]| /*对所计算的特征向量的相应分量，求新旧值之差的最大值 ***localmax*** */

(3) for *i*=1 to *m*-1 do

if (|*b*[*i*]|>*localmax*) then *localmax*=|*b*[*i*]| end if

end for

(4) 用 **Allreduce** 操作求出所有处理器中 ***localmax*** 值的最大值 ***max*** 并广播到所有处理器中

(5) for *i*=0 to *m*-1 do *b*[*i*] =*b*[*i*]/*max* end for /*对所计算的特征向量归一化 */

(6) 用 **Allgather** 操作将各处理器中计算出的特征向量的分量的新值组合并广播到所有处理器

(7) diff = *max* - *oldmax*, *oldmax* = *max*

end while

End

给出上述算法的 MPI 实现。（15 分）

（需包括主/从进程间数据分发和最后结果的显示，但不必描述文件读写）

本试卷答题过程中可能用到的函数原型：

MPI_Comm_size(MPI_Comm comm, int *size)

MPI_Comm_rank(MPI_Comm comm, int *rank)

MPI_Allreduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)

MPI_Bcast(void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm)

MPI_Allgather(void *sendbuf, int sendcnt, MPI_Datatype sendtype, void *recvbuf, int recvcnt, MPI_Datatype recvtype, MPI_Comm comm)

MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)

MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)

MPI_Sendrecv(void *sendbuf, int sendcount, MPI_Datatype sendtype,int dest, int sendtag, void *recvbuf, int recvcount, MPI_Datatype recvtype, int source, int recvtag, MPI_Comm comm, MPI_Status *status)

MPI_Wait(MPI_Request *request, MPI_Status *status)