装订线 答题时不要超过此线

中 国 科 学 技 术 大 学

2014－2015 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、 分析以下 3 个循环中存在的依赖关系；分别通过循环交换、分布和逆转等多种方法来尝试向量化和/或并行化变换：（3×10=30 分）

```
for i = 1 to 100 do   //循环 1
   A[i] =  A[i] + B[i-1];
   B[i] =  C[i-1] * 2 ;
   C[i] =  1 / B[i] ;
   D[i] =  C[i] * C[i] ;
endfor
```

```
for i = 1 to 999 do   // 循环 2
   A[i] = B[i] + C[i];
   D[i] = ( A[i] + A[ 999-i+1 ] ) / 2 ;
endfor
```

```
for i = 1 to 100 do   // 循环 3
   for j = 1 to 100 do
      A[3*i+2*j, 2*j] = C[i,j] * 2     ;
      D[i,j]          = A[i-j+6, i+j] ;
   endfor
endfor
```

二、 假设某种 MPI 广播通信方案如下：将 P 个进程看作 $\sqrt{p}\times\sqrt{p}$ 的二维拓扑结构，并且将各个行或列进程组划分为单独的子通信域。这样，root 进程可先在其行子通信域中进行广播，然后该行中的所有进程在各自的列通信子域中再广播。给出该广播方案的 MPI 具体实现。（20 分）

三、 设有两个进程 A 和 B，以及结构变量 stu。现在，进程 A 将 stu 发送给进程 B。请用三种不同的 MPI 实现来完成进程 A 的发送操作。（3×10=30 分）

```
struct Student {int id; char name[10];double mark[3]; char pass;
} stu;
```

四、 以下是单处理器上的矩阵求逆算法：

```
Begin
  for i=1 to n do
    (1) a[i,i]=1/a[i,i]
    (2)for j=1 to n do   if (j≠i) then a[i,j]=a[i,j]*a[i,i] end if   end for
    (3)for k=1 to n do
         for j=1 to n do if ((k≠i and   j≠i)) then a[k,j]=a[k,j]-a[k,i]*a[i,j] end if   end for
       end for
    (4)for k=1 to n do   if (k≠i) then a[k,i]= -a[k,i]*a[i,i]   end if   end for
  end for
End
```

矩阵求逆的过程中，依次利用主行 ***i*(*i*=0,1,…,*n*-1)**对其余各行 ***j*(*j*≠*i*)**作初等行变换，由于各行计算之间没有数据相关关系，因此可以对矩阵 ***A*** 按行划分来实现并行计算。考虑到在计算过程中处理器之间的负载均衡，对 ***A*** 采用行交叉划分：设处理器个数为 ***p***，矩阵 ***A*** 的阶数为 ***n***，$m=\lceil n/p \rceil$，对矩阵 ***A*** 行交叉划分后，编号为 ***i*(*i*=0,1,…,*p*-1)**的处理器存有 ***A*** 的第 ***i*, *i*+*p*,…, *i*+(*m*-1)*p*** 行。在计算中，依次将第 **0,1,…,*n*-1** 行作为主行，将其广播给所有处理器，这实际上是各处理器轮流选出主行并广播。发送主行数据的处理器利用主行对其主行之外的 ***m*-1** 行行向量做行变换，其余处理器则利用主行对其 ***m*** 行行向量做行变换。

请写出矩阵求逆算法的 **MPI** 并行实现。（20 分）

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_Reduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)
MPI_Bcast( void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm )
MPI_Gather(void *sendbuf, int sendcnt, MPI_Datatype sendtype, void *recvbuf, int recvcnt, MPI_Datatype recvtype,
int root, 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_Pack(void *buf, int count, MPI_Datatype type, void* packBuf,int packSize,int *Position,MPI_Comm comm)
MPI_Pack_size(int incount, MPI_Datatype datatype,MPI_Comm comm,int *size)
MPI_Type_vector(int count, int blocklength, int stride, MPI_Datatype old_type, MPI_Datatype *newtype_p)
MPI_Comm_dup(MPI_Comm comm, MPI_Comm *newcomm)
MPI_Comm_split(MPI_Comm comm, int color, int key, MPI_Comm *newcomm)
MPI_Wait(MPI_Request *request, MPI_Status *status)
MPI_Type_struct(int count,int blocklens[],MPI_Aint indices[],MPI_Datatype old_types[], MPI_Datatype *newtype)
MPI_Address( void *location, MPI_Aint *address )
MPI_Type_commit(MPI_Datatype *datatype)
```