装订线 答题时不要超过此线

中 国 科 学 技 术 大 学

2011－2012 学年第一学期考试试卷

参考答案

考试科目：并行程序设计　　得分:_____

学生所在系:_____　姓名:_____　学号:_____

本试卷共五个大题！

一、　描述以下循环中的依赖关系及语句依赖图：（20 分）

```
for i = 4 to 100 do
S:  A(i) = B(i) + c(i)
T:  B(i+2) = A(i-1) + A(i-3) + C(i-1)
U:  A(i+1) = B(2*i+3) + 1
endfor
```

参考解答:

循环中语句依赖关系如下：

- 语句 T 流依赖于语句 S，即 $S\ \delta^f\ T$，满足依赖关系的偶对集合为：

  $\{ <S(i), T(j)> \mid i = j - 1;\ \ 5 \leqslant j \leqslant 100 \} \cup \{ <S(i), T(j)> \mid i = j - 3;\ \ 7 \leqslant j \leqslant 100 \}$

- 语句 S 流依赖于语句 T，即 $T\ \delta^f\ S$，满足依赖关系的偶对集合为：

  $\{ <T(i), S(j)> \mid i = j - 2;\ \ 6 \leqslant j \leqslant 100 \}$

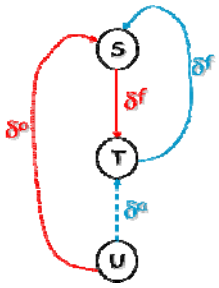
- 语句 S 输出依赖于语句 U，即 $U\ \delta^o\ S$，满足依赖关系的偶对集合为：

  $\{ <U(i), S(j)> \mid i = j - 1;\ \ 5 \leqslant j \leqslant 100 \}$

- 语句 T 反依赖于语句 U，即 $U\ \delta^a\ T$，满足依赖关系的偶对集合为：

  $\{ <U(i), T(j)> \mid j = 2*i + 1;\ \ 4 \leqslant i \leqslant 49 \}$

语句依赖图如下：



二、编写 MPI 程序：依据所有 MPI 进程运行所在节点集合的大小，进行 MPI_COMM_WORLD 通信域划分，使得运行在相同节点上的 MPI 进程都将被分到相同的子通信域。（20 分）

参考解答：

基本思想是：

首先，每个进程均收集所有进程的运行节点信息（通过 MPI_Allgather 完成）；

其次，根据所有节点信息和自身的运行节点名，每个进程确定自己在该节点上的位序，即 key；

最后，根据所有节点信息和自身的运行节点名，每个进程确定该节点名在所有节点信息中的第一次出现的位置，即 color。

MPI 程序主要部分如下：

```
int rank,Group_Size;
char host[128];
char *allhost; //存放所有节点名称字符串的数组；
int color,key;
MPI_Comm myComm, nodeComm;
… …
MPI_Init(&argc, &argv);//MPI 初始化
MPI_Comm_dup(MPI_COMM_WORLD, &myComm);//复制 MPI_COMM_WORLD 通信域
MPI_Comm_rank(myComm,&rank);
MPI_Comm_size(myComm,&Group_Size);
allhost = (char*)malloc(128*Group_Size);
// 分配节点名数组空间 allhost,大小为进程总数 X 128 字节
gethostname(host, 128); // 获得 MPI 进程运行所在节点的名称

MPI_Allgather(host,128,MPI_BYTE,allhost,128,MPI_BYTE, myComm );
// 每个 MPI 进程收集全部进程运行节点名称，并按照 MPI 进程编号由小到大的顺序，
// 存到 allhost 数组。示例如下表所示：每个节点名占据 128 个字节。
```

rank	0	1	2	3	4	5	6	… …
allhost	Node0	Node0	Node3	Node4	Node5	Node4	Node4	… …

```
// 进程编号(rank)为 i 的 MPI 进程运行所在节点名称,即 host,应该存放在 allhost
的第 i 项，即 allhost 的第 i*128 个字节偏移处存放的字符串。
```

```
//现在开始获得通信子域划分所需的 color 和 key
   color = 0; key = 0; // 初值为 0
   for(i=0;i<rank;i++){
      // 每个进程统计在编号小于它的进程中，有多少和其节点名称相同；
      if(strcmp(allhost+i*128,host)) continue;// 不同，则下一个进程
      else   key++; //相同，key 增一。
   } //循环结束，key 即为当前进程的所在通信域的编号（KEY）

   for(i=0;i<=rank;i++){
      if(!strcmp(allhost+i*128,host)) break;
      //节点名称相同，则跳出循环
   }
   color = i;//对于 rank==0 的主进程而言，总是 color 为 0。
  // 每个进程在 allhost 数组中确定第一个和其节点名称相同的进程所在位置，
  // 并以此作为通信域划分中的 COLOR。
```

rank	0	1	2	3	4	5	6	… …
allhost	Node0	Node0	Node3	Node4	Node5	Node4	Node4	… …
Key	0	1	0	0	0	1	2	… …
COLOR	0	0	2	3	4	3	3	… …

```
// 如此一来，可以划分通信域了！
MPI_Comm_split(myComm, color, key, &nodeComm);
```

三、 MPI 构造函数 MPI_Type_indexed 可从相同类型元素组成的连续数据区中提取若干不同长度、不同索引位置的数据块组合为派生类型消息，其原型如下：

```
int MPI_Type_indexed(int count,int blocklens[],
                     int indices[],MPI_Datatype old_type,
                     MPI_Datatype *newtype )
```

参数含义如下：

count： 数据块的个数；也是参数 indices 和 blocklens 两个数组的有效长度。

blocklens：数组，存放每个块中类型为 old_type 的元素个数。

indices：数组，存放每个块的首元素在 old_type 数据区中的索引。

old_type：元素的基本类型。newtype ：新的派生消息类型。

设有二维矩阵 int SA[N][N]。试写出 MPI 程序片段来定义：（20 分）

（1）由下三角矩阵构成的派生消息类型 lowTriangle;

（2）由主对角线（N 个元素）及其上下两侧辅对角线（N-1 元素）组成的主条带派生消息类型 mainStripe。

（3）编写两个 MPI 进程组成的程序，其中 0 号进程向 1 号进程发送由 SA[0][11]开始的 5×5 的上三角矩阵。

参考解答：

依题意，主要是完成函数 MPI_Type_indexed 的参数填写工作。

（1） 由下三角矩阵构成的派生消息类型 lowTriangle;

```
count=N; // 下三角矩阵包含 N 个数据块
for(i=0;i<N;i++) blocklens[i] = i+1;
//每块数据含 1、2、…、N 个数据
for(i=0;i<N;i++) indices[i] = i*N;
//每块首元素偏移从 0、N、2*N…、(N-1)*N
MPI_Type_indexed(count,blocklens,indices,MPI_INT,&lowTriangle);
MPI_Type_commit(&lowTriangle);
```

（2） 主条带派生消息类型 mainStripe

```
count=N; // 主条带矩阵包含 N 个数据块
blocklens[0] = 2;//首个数据块包含 2 个 old_type 数据
blocklens[N-1] = 2; //最后的数据块包含 2 个 old_type 数据
for(i=1;i<N-1;i++) blocklens[i] = 3;
//其余各块均包含 3 个 old_type 数据
indices[0] = 0;//首块偏移为 0；
for(i=1;i<N;i++) indices[i] = i*(N+1);//其余各块偏移。
MPI_Type_indexed(count,blocklens,indices,MPI_INT,&mainStripe);
MPI_Type_commit(&mainStripe);
```

（3） 5×5 的上三角矩阵 upTriangle5

```
count = 5; //含 5 个数据块
for(i=0;i<5;i++) blocklens[i] = 5-i;
// 每块长度依次为 5、4、3、2、1
indices[0] = 0;//首块偏移为 0；
for(i=1;i<5;i++) indices[i] = i*(N+1);//其余各块偏移
MPI_Type_indexed(count,blocklens,indices,MPI_INT,&upTriangle5);
MPI_Type_commit(&upTriangle5);

if(rank==0){
  MPI_Send(&SA[0][11],1, upTriangle5,
           1,2012,MPI_COMM_WORLD);
}
else if(rank==1){
  MPI_Recv(&SA[0][11],1,upTriangle5,
           0,2012,MPI_COMM_WORLD,&status);
}
```

四、 生成质数的经典方法就是 Eratosthenes 筛选法（Eratosthenes Filtering），在这种方法中，所有的一系列整数(设其小于 n)从 2 开始产生。第一个数 2 是质数给予保留，然后序列中所有 2 的倍数因不可能是质数都被删除；将序列中所有 3 的倍数都删除；将序列中所有 5 的倍数都删除…将序列中所有$\lfloor\sqrt{n}\rfloor$的倍数都删除，这样剩下的未被删除的数就是 2~n 的序列中的所有质数。按此，请给出基于流水线设计技术的 OpenMP 并行程序。（20 分）

参考解答：

```
//数组 a 存放 0、1、2～N，sn 为 N 的平方根
//如果某个数不是素数，则其在 a 中对应项置为 0；
p = 2;
while(p<sn)
{
   #pragma omp parallel for private(i)
   for(i=p+1;i<N;i++)
     if (a[i]==0) continue;
     else if( a[i] % p == 0 ) a[i] = 0;
   while( a[++p] ==0 ) ;
}
```

五、 矩阵相乘的另一种并行算法是 Fox 算法（Fox Algorithm）：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$\sqrt{p}$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。

请写出实现 Fox 算法的 MPI 并行程序。（20 分）

参考解答：

(1) 有关辅助工作及函数：

(1.1)依题意，所有进程拓扑为$\sqrt{p}\times\sqrt{p}$二维划分。

因此，进程 **rank** 和行 **row**、列 **col** 转换如下：

```
int  sp =  sqrt(p) ;//√p
int  Sub_Matrix_Size = (N / sp) * ( N / sp );
int  get_Rank(int row, int col){ return  row * sp + col; }
int  get_Row(int rank){ return  rank / sp; }
int  get_Col(int rank){ return  rank % sp; }
int upRow(rank){
  return get_Rank((sp+get_Row(rank)-1)%sp,get_Col(rank))
}
int lowRow(rank){
  return get_Rank((get_Row(rank)+1)%sp,get_Col(rank))
};

void Copy_SubMatrix(SUB_A,SUB_B);
//将子矩阵 SUB_A 拷贝到子矩阵 SUB_B;

void Mul_ADD_SubMatrix(SUB_A,SUB_B,SUB_C);
//子矩阵 SUB_A 和 SUB_B 相乘，结果累加到子矩阵 SUB_C;

// 即 SUB_C += SUB_A * SUB_B
```

(1.2) 此外，依题意，具有相同行 row 的进程之间需要进行广播通信。因此，为简化编程，可以将行 row 相同的进程可以分在一个子通信域内。

```
int color = get_Row(rank);
int key   = get_Col(rank);
MPI_COMM  rowComm;
MPI_Comm_split(MPI_COMM_WORLD,color,key,&rowComm);
```

(2) Fox 算法的 MPI 实现主要部分

```
int i;
int row,col;
int root;
MPI_Status stat;
MPI_DOUBLE *local_SUB_A, *local_SUB_B,*local_SUB_C;
MPI_DOUBLE *temp_SUB_A,  *temp_SUB_B;


row  = col = get_Row(rank);// 取 Ai,i 的下标

root = get_Rank(row,col); // 设置首次广播的 root


for( i=0; i<sp; i++){

   //算法步①

    Copy_SubMatrix(local_SUB_A,temp_SUB_A);
   MPI_Bcast(temp_SUB_A,Sub_Matrix_Size,MPI_DOUBLE,
             root, rowComm);

   //算法步②

   Mul_ADD_SubMatrix(temp_SUB_A,local_SUB_B,local_SUB_C);

   //算法步③

MPI_Sendrecv(
 local_SUB_B,Sub_Matrix_Size,MPI_DOUBLE,upRow(rank),9999,
 temp_SUB_B,Sub_Matrix_Size,MPI_DOUBLE,lowRow(rank),9999,
 MPI_COMM_WORLD,&stat
);
   Copy_Sub_Matrix(temp_SUB_B,local_SUB_B);

   //算法步④

   col = (col+1) % sp
   root = get_Rank(row,col);

}
```

本试卷答题过程中可能用到的函数原型：

MPI_Type_commit(MPI_Datatype *datatype)

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

MPI_Comm_split(MPI_Comm comm, int color, int key, MPI_Comm *newcomm)

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_Allgather(void *sendbuf, int sendcount, MPI_Datatype sendtype,void *recvbuf, int recvcount, 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)

gethostname(char *name, int namelen)