装订线 答题时不要超过此线

中 国 科 学 技 术 大 学

2017－2018 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、 针对以下两个循环:（本题小计 40 分）

```
for i = 1 to M do   //循环 1  M , N, C 均是常量
   for j = 1 to N
      A[i+1,j+1] =  A[i,j] + C;
   endfor
endfor
```

(1) 给出迭代依赖示意图。(5 分)

(2) 简述能否逆转外层的 i 循环？能否交换内外循环次序？（10 分）

(3) 尝试用 OpenMP 并行化此循环。（5 分）

```
for i = 1 to 100 do // 循环 2  N 是常量
    X[i] = Y[i] + 10; // 语句 S1
    for j = 1 to 100 do
        B[j] = A[j, N]; // 语句 S2
        for k = 1 to 100 do
            A[j+1, k] = B[j] + C[j, k]; // 语句 S3
        endfor // loop-k
        Y[i+j] = A[j+1, N]; // 语句 S4
    endfor // loop-j
endfor // loop-i
```

(1) 给出此循环的语句依赖图。（10 分）

(2) 尝试向量化/并行化此循环。（10 分）

二、给出如下 C 程序片段的 OpenMP 实现。（15 分）

```
double factor = 1.0, sum = 0.0, Pi = 0.0;
int k;
for(k = 0;k < N; k++){ // N 是常量
   sum = sum + factor / ( 2 * k + 1);
   factor =  -  factor;
}
Pi = 4.0 * sum;
```

三、采用 MPI_Sendrecv()给出 cannon 乘法在 N*N 进程拓扑下，进行 A、B 子块矩阵初始对齐以及随后计算中移位对齐操作的主要代码。（15 分）

四、设有长度为 N 的双精度数向量 M。给出构造新的消息类型且发送 M 中编号可被 3 整除的所有分量元素的 MPI 代码。设编号由 0 开始。（10 分）

五、 *参数服务器系统*的 MPI 模拟。

设系统中总计有 N 个进程，其中 P 个进程作为参数*服务器进程*，而 Q 个进程作为*工作进程*（N = P + Q， 且 0 < P < < Q）。工作进程和服务器进程的互动过程如下：

1. 第 n 个工作进程首先产生一个随机数，发送给第 n%P 个参数服务器进程。然后等待并接收它对应的参数服务器进程发送更新后的数值，之后，再产生随机数，再发送……。
2. 每个参数服务器进程等待并接收来自它对应的所有工作进程的数据，在此之后，经通信，使所有的参数服务器获得所有工作进程发送数据的平均值。
3. 每个参数服务器发送该平均值给它对应的所有工作进程，然后再等待……。

试给出上述互动过程的 MPI 程序实现。（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_Allreduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)
MPI_Allgather(void *sendbuf, int sendcnt, MPI_Datatype sendtype, void *recvbuf, int recvcnt, MPI_Datatype recvtype, MPI_Comm comm)
MPI_Bcast( void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm )
MPI_Alltoall( 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)
MPI_Wait(MPI_Request *request, MPI_Status *status)
MPI_Type_vector(int count, int blocklength, int stride, MPI_Datatype old_type, MPI_Datatype *newtype_p)
MPI_Type_commit(MPI_Datatype *datatype)
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_Comm_dup(MPI_Comm comm, MPI_Comm *newcomm)
MPI_Comm_split(MPI_Comm comm, int color, int key, MPI_Comm *newcomm)
```