# 《高性能计算》实验指导书

## 实验1：用MPI点对点通信函数进行分布式内存编程

### 一、实验目的

1. 熟练掌握常用的MPI点对点通信函数，并能够应用其进行并行编程
2. 通配符的使用

### 二、实验内容

1. 采用点对点的通信方式编写梯形积分法的MPI程序：
2. 梯形积分区间的端点和划分的个数是固定；
3. 允许用户输入梯形积分区间的端点和划分的个数；
4. 修改梯形积分法，使其能够在comm\_sz无法被n整除的情况下，正确估计积分值(假设n>=comm\_sz);
5. 采用点对点通信方式实现编写一个MPI程序，采用如下的树形通信结构来计算全局总和（考虑comm\_sz是2的幂的特殊情况）.

|  |
| --- |
| f03-06-9780123742605.eps |
| 图1 |

|  |
| --- |
| 表1 MPI通信计算全局总和程序 |
| #include <stdio.h>  #include <stdlib.h>  #include <mpi.h>  int Global\_sum(int my\_int, int my\_rank, int comm\_sz, MPI\_Comm comm);  const int MAX\_CONTRIB = 20;  int main(void) {  int i, sum, my\_int;  int my\_rank, comm\_sz;  MPI\_Comm comm;    MPI\_Init(NULL, NULL);  comm = MPI\_COMM\_WORLD;  MPI\_Comm\_size(comm, &comm\_sz);  MPI\_Comm\_rank(comm, &my\_rank);  srandom(my\_rank + 1);  my\_int = random() % MAX\_CONTRIB;    sum = Global\_sum(my\_int, my\_rank, comm\_sz, comm);    if ( my\_rank == 0) {  printf("Sum = %d\n",sum);  }  MPI\_Finalize();  return 0;  } /\* main \*/  /\*-------------------------------------------------------------------  \* Function: Global\_sum  \* Purpose: Implement a global sum using tree-structured communication  int Global\_sum(  int my\_int /\* in \*/,  int my\_rank /\* in \*/,  int comm\_sz /\* in \*/,  MPI\_Comm comm /\* in \*/) {    int partner, recvtemp;  int my\_sum = my\_int;  unsigned bitmask = 1;    while (bitmask < comm\_sz) {  partner = my\_rank ^ bitmask;    if (my\_rank < partner) {  MPI\_Recv(&recvtemp, 1, MPI\_INT, partner, 0, comm,  MPI\_STATUS\_IGNORE);  my\_sum += recvtemp;  bitmask <<= 1;  } else {  MPI\_Send(&my\_sum, 1, MPI\_INT, partner, 0, comm);  break;  }  } /\* while \*/  return my\_sum;  } /\* Global\_sum \*/ |

### 三、实验过程

### 四、实验要求

1. 记录实验过程代码及结果，并进行必要的分析及自己的理解