# 实验4实验报告

实验题目：利用MPI实现并行排序算法

实验环境：实验所使用电脑操作系统为 windows10 操作系统， IDE 为 Visual Studio2017，

OpenMP 实验环境由 VS2017 提供， MPI 使用 MS-MPI，代码编写和测试均由 VS2017 实

现。

算法设计与分析：

首先我们要明确并行排序算法的过程：

(1)均匀划分：将n个元素A[1..n]均匀划分成p段，每个pi处理A[(i-1)n/p+1..in/p]

(2)局部排序：pi调用串行排序算法对A[(i-1)n/p+1..in/p]排序

(3)选取样本：pi从其有序子序列A[(i-1)n/p+1..in/p]中选取p个样本元素

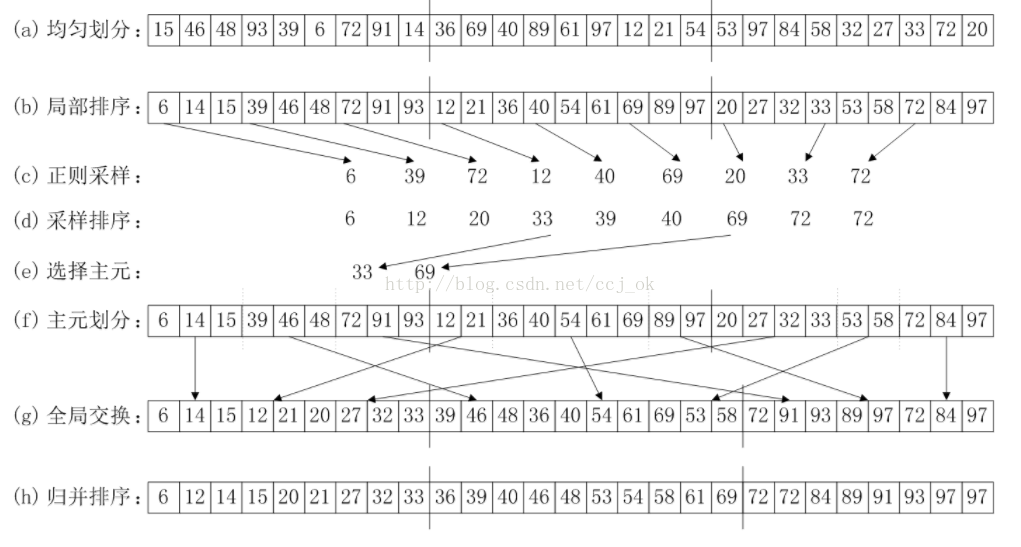
(4)样本排序：用一台处理器对p2个样本元素进行串行排序

(5)选择主元：用一台处理器从排好序的样本序列中选取p-1个主元，并播送给其他pi

(6)主元划分： pi按主元将有序段A[(i-1)n/p+1..in/p]划分成p段

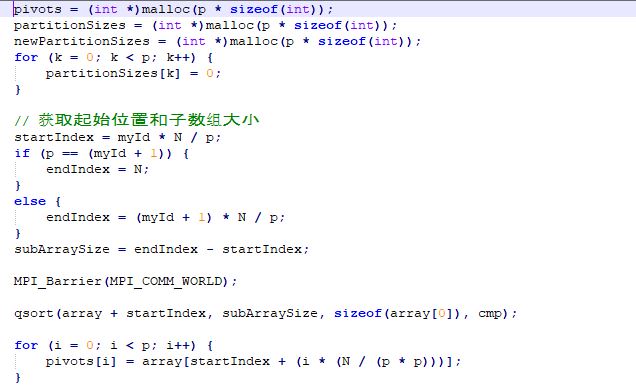
(7)全局交换：各处理器将其有序段按段号交换到对应的处理器中

(8)归并排序：各处理器对接收到的元素进行归并排序

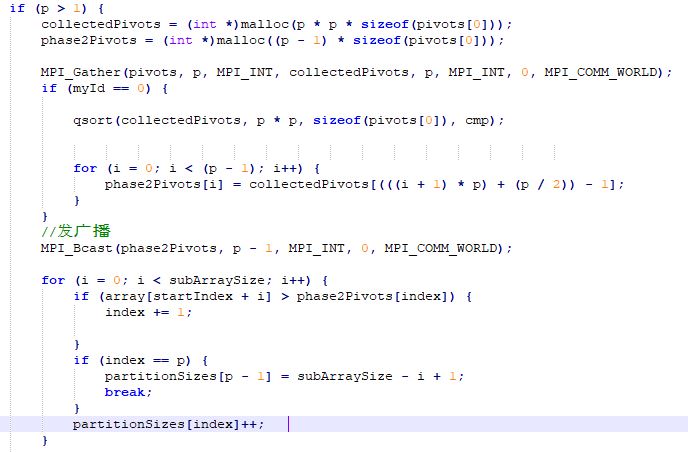


本次算法实现严格按照并行排序算法执行步骤编写。使用具体的MPI函数接口实现各个进程间通信和同步。

核心代码：

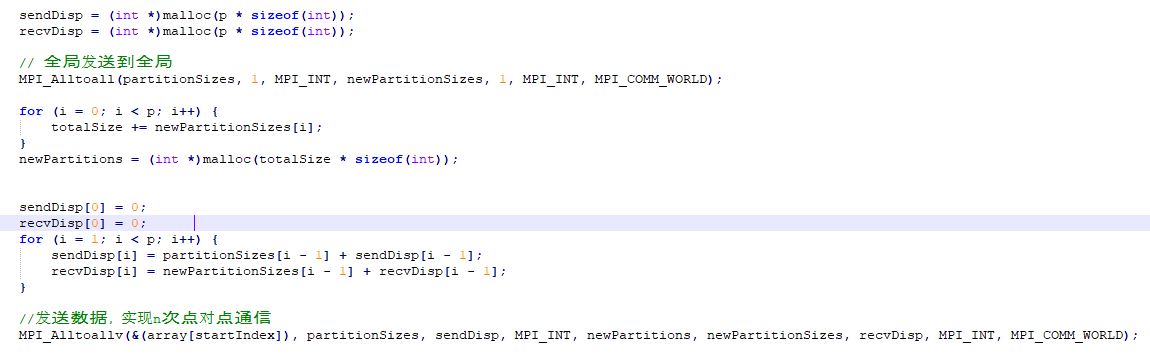


如上图，首先对整个数组进行分组，获取各个进程所分配到的数组数据数。随后先调用库函数对分配到的数据进行排序，然后进行采样获取p个采样数据。

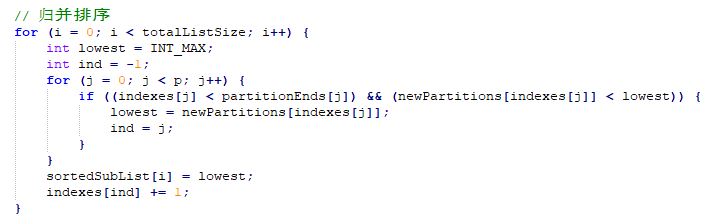


随后在并行的情况下，所有进程把采样数据发送给根进程，根进程负责把这p^2个采样点进行排序，随后从其中选取p-1个数据作为分割点广播出去。

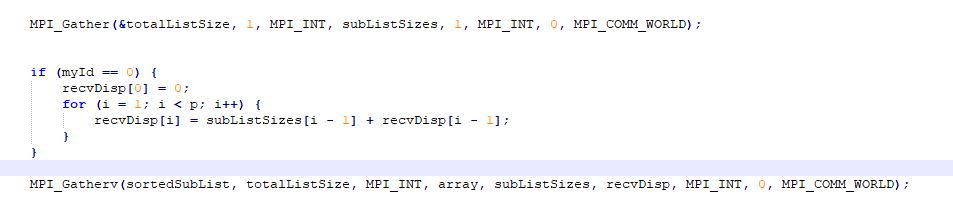
每个进程获取到分割点后，按照分割点把自己负责的数据进行分割，并记录每个分割的大小。



首先调用MPI\_Alltoall()函数使得每个进程都知道自己即将接受的p-1段数据（不算自己的）的规模。求和计算总的数据规模。在各个进程都明确自己将要发送和接收的数据和数据的规模后使用MPI\_Alltoallv()函数，使得进程i向进程j发送自己分割得到的第j段数据，并接收其他p-1个进程发给自己的数据段。



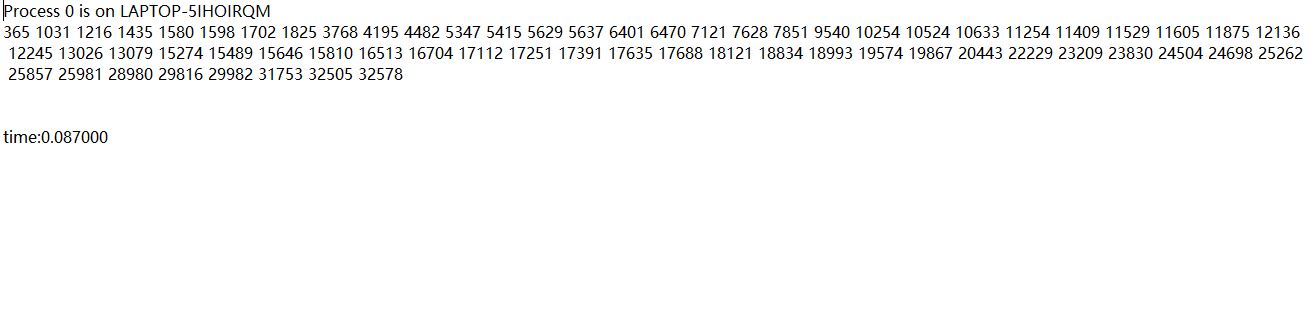
随后每个处理器对自己的p段新的数据进行归并排序。



最后各个进程把自己的已排序完成的数据规模发送给根进程，然后根进程按照接收到的数据规模信息进行数据的接收和存放。完成排序。

实验结果：

使用freopen()函数把命令行输出重定向到文件：



运行时间

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 规模/并行度 | 1 | 2 | 3 | 4 |
| 10000000 | 4.725s | 3.092s | 2.252s | 2.798s |

加速比：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 规模/并行度 | 1 | 2 | 3 | 4 |
| 10000000 | 1 | 1.528 | 2.098 | 1.689 |

实验总结与分析：

并行实验的难点就是进程间的相互通信，这次的通信中使用到的MPI\_Gather(), MPI\_Alltoall(), MPI\_Alltoallv()都需要深刻理解函数含义才能正确使用。

分析这次实验的实验和测试结果容易知道，并行编程的优越性常常在数据规模较大时才能展现出性能方面的优越性。且随着并行度的增加，进程间通信成本也在逐渐攀升，因此我们必须具体情况具体分析，在需要把并行编程应用到工作和生活中时，根据实际情况选择串行编程和并行编程，根据具体情况选择并行编程下的并行度，使得效率达到最高。例如在这次实验排序一千万数据时，在并行度从1开始增加时，运行时间在逐步减小，但当并行度从3上升到4时，通信成本增加使得排序的性能低于并行度为3时候的情况。