实验题目:

使用 MPI 实现 PSRS 算法
（注意请输入 2 的整数次方的数，该程序并没有处理边界条件。最多支持 8 线程并行）

算法设计与分析:

(注：排序数据是利用随机数生成的)
(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
————————————————————
原文链接：https://blog.csdn.net/ccj_ok/java/article/details/72848671

核心代码:

```
//局部排序
    qsort(a+(n/numprocs)*myid,n/numprocs,sizeof(int),cmp);
    MPI_Barrier(MPI_COMM_WORLD);

    //将局部排序结果汇总到 0 号线程中
    if(myid!=0)
    MPI_Bsend((a+((n/numprocs)*(myid))),4*(n/numprocs),
        MPI_BYTE,0,myid,MPI_COMM_WORLD);


    //全局交换后的排序
    for(i=0;i<numprocs;i++)
    if(myid==i)
    {
        printf("this is thread %d\n",myid);
        qsort(newpartitions,totalSize,sizeof(int),cmp);
        for(i=0;i<totalSize;i++)
            printf("%3d",newpartitions[i]);
        printf("\n\n\n");
```

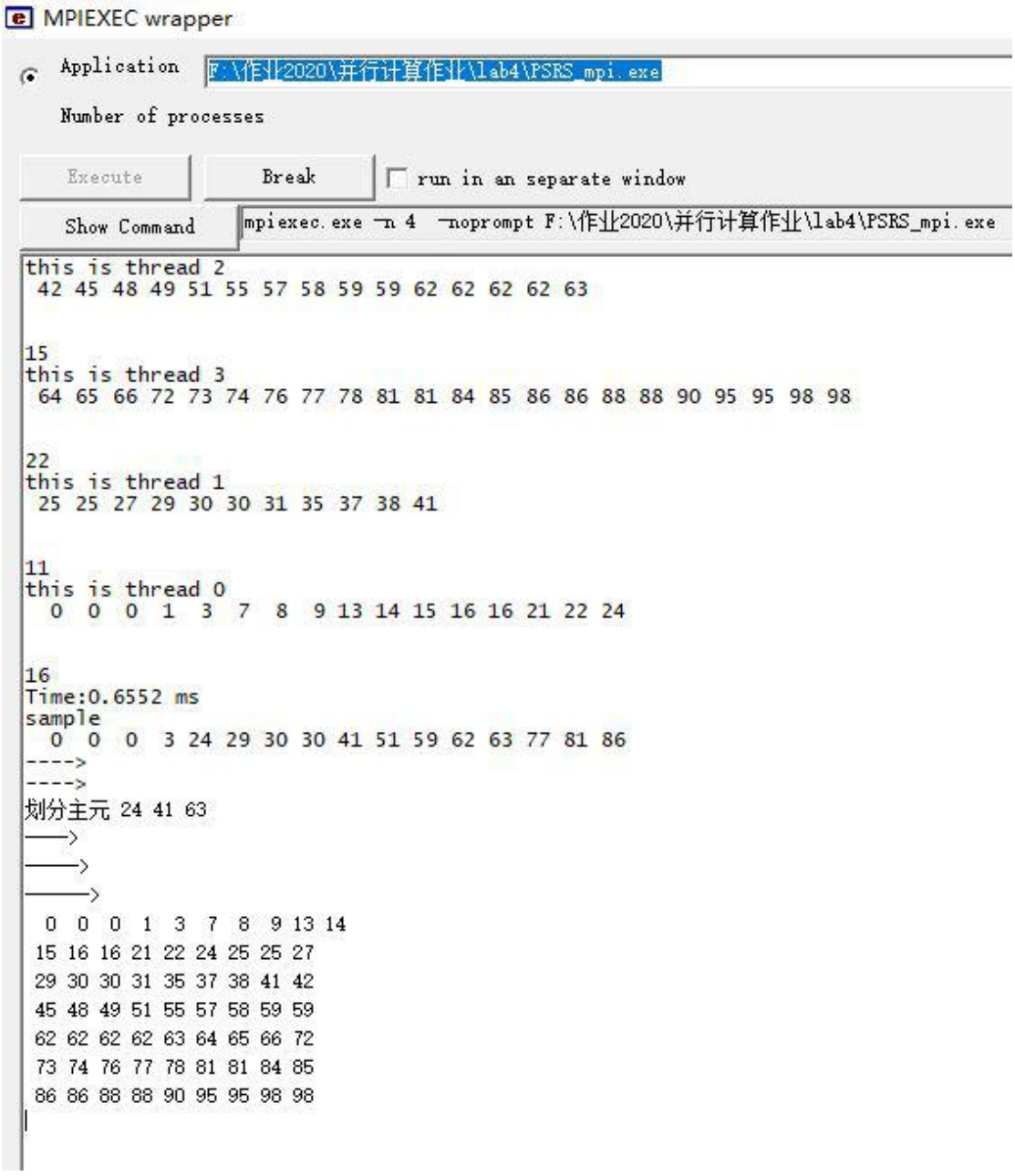
```
        printf("%d\n",totalSize);
    }
```

实验结果:

排序 5,000,000 个数:

进程数	1	2	4	8
运行时间	427.9ms	364.3ms	229.4ms	783.9ms
加速比	1	1.18	1.87	/

8 线程速度慢了可能和物理内核数不足有关。


MPIEXEC wrapper
Application
F:\作业2020\并行计算作业\lab4\PSRS_mpi.exe
Number of processes
Execute
Break
run in an separate window
Show Command
mpiexec.exe -n 4 -noprompt F:\作业2020\并行计算作业\lab4\PSRS_mpi.exe
this is thread 2
42 45 48 49 51 55 57 58 59 59 62 62 62 62 63
15
this is thread 3
64 65 66 72 73 74 76 77 78 81 81 84 85 86 86 88 88 90 95 95 98 98
22
this is thread 1
25 25 27 29 30 30 31 35 37 38 41
11
this is thread 0
0 0 0 1 3 7 8 9 13 14 15 16 16 21 22 24
16
Time:0.6552 ms
sample
0 0 0 3 24 29 30 30 41 51 59 62 63 77 81 86
---->
---->
划分主元 24 41 63
---->
---->
---->
0 0 0 1 3 7 8 9 13 14
15 16 16 21 22 24 25 25 27
29 30 30 31 35 37 38 41 42
45 48 49 51 55 57 58 59 59
62 62 62 62 63 64 65 66 72
73 74 76 77 78 81 81 84 85
86 86 88 88 90 95 95 98 98