**深 圳 大 学**

**实 验 报 告**

**课程名称****：并行计算**

**实验名称：PSRS排序算法的MPI并行程序**

**姓 名：刘俊楠**

**学 号：2017303010**

### 班 级：计科一班

**实验日期：第14周实验课**

## 一. 实验目的

1. 学会编写简单的MPI程序；

2. 对并行程序进行简单的性能分析。

## 二. 实验环境

1. 软件环境：Microsoft Visual Studio 2013。

## 三. 实验内容

1. 实验要求：用MPI编写7.1.1节中的PSRS排序算法，将数组*a*中的*n*个数排序，结果输出到数组*b*中。

* 初始时，数组*a*存储在进程0中，其他进程没有数据。结束时，数组*b*也存储在进程0中。
* 数组*a*中的每个数都初始化为一个0到1之间的随机double型值（用rand()/double(RAND\_MAX)实现）。
* 添加检测计算结果是否正确的代码。
* 计算执行时间用MPI\_Wtime()函数。

2. 程序代码和说明：

#include<iostream>

#include<mpi.h>

#include<math.h>

#include<algorithm>

#include<vector>

**using** **namespace** std**;**

int n **=** 100000000**;**

//p路归并排序函数

void Mergesort**(**double**\*** pData**,** int**\*** pIndex**,** int len**,** int pnum**)**

**{**

double**\*** pSort **=** **new** double**[**len**];** // 归并后的p路有序数组

int**\*** idx **=** **new** int**[**pnum**];** // p段对应p个下标

// 下标初始化

**for** **(**int i **=** 0**;** i **<** pnum**;** i**++)** idx**[**i**]** **=** pIndex**[**i**];**

// p路归并

int k **=** 0**;**

**for** **(**int i **=** 0**;** i **<** len**;** i**++)**

**{**

int tidx **=** 0**;**

double tData **=** 2**;**

// 找到各个指针指向的最小值

**for** **(**int j **=** 0**;** j **<** pnum**;** j**++)**

**if** **(**idx**[**j**]** **<** pIndex**[**j **+** 1**]** **&&** pData**[**idx**[**j**]]** **<** tData**)**

**{**

tData **=** pData**[**idx**[**j**]];**

tidx **=** j**;**

**}**

pSort**[**i**]** **=** tData**;** // 赋值

idx**[**tidx**]** **++;** // 对应指针右移

**}**

**for** **(**int i **=** 0**;** i **<** len**;** i**++)** pData**[**i**]** **=** pSort**[**i**];**

**}**

//串行PSRS函数，使用快排替代

void PSRS\_single**(**double**\*** a**)** **{**

sort**(**a**,** a **+** n**);**

**}**

int main**(**int argc**,**char**\*\***argv**)** **{**

int rank**,** size**;** //分别是进程号与进程数目

int times **=** 5**;** //执行次数

MPI\_Status status**;** //MPI状态变量

double avg\_time**[**5**];** //并行时间数组

double end1**;** //串行时间

MPI\_Init**(&**argc**,** **&**argv**);**//初始化MPI

MPI\_Comm\_rank**(**MPI\_COMM\_WORLD**,** **&**rank**);**

MPI\_Comm\_size**(**MPI\_COMM\_WORLD**,** **&**size**);**

**while** **(**times**--)** **{**

int num **=** n **/** size**;** //分段长度

double**\*** a **=** **new** double**[**n**];** //初始数组

double**\*** b **=** **new** double**[**n**];** //结果数组

double**\*** b\_a **=** **new** double**[**num**];** //各个进程存放的分段数组

double**\*** temp **=** **new** double**[**n**];** //串行计算的结果数组

double**\*** sample **=** **new** double**[**size**];** //单个进程的样本元素数组

double**\*** sample1 **=** **new** double**[**size**\***size**];** //所有进程的样本元素数组

double**\*** mainyuan **=** **new** double**[**size **-** 1**];** //主元数组

int**\*** sendindex1 **=** **new** int**[**size **+** 1**];** //发送主元所在下标的数组

int**\*** sendlength1 **=** **new** int**[**size **+** 1**];** //发送各个进程主元段的长度的数组

int**\*** recvindex1 **=** **new** int**[**size **+** 1**];** //各个进程接受主元所在下标的数组

int**\*** recvlength1 **=** **new** int**[**size **+** 1**];** //各个进程接受主元段长度的数组

//主进程初始化

**if** **(**rank **==** 0**)**

**{**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)** **{**

a**[**i**]** **=** **(**double**)**rand**()** **/** **(**RAND\_MAX**);**

temp**[**i**]** **=** a**[**i**];**

**}**

**}**

double start**;**

//串行计算

**if** **(**rank **==** 0 **&&** times **==** 4**)** **{**

start **=** MPI\_Wtime**();**

PSRS\_single**(**temp**);**

end1 **=** MPI\_Wtime**()** **-** start**;**

cout **<<** "串行时间为：" **<<** end1 **<<** endl**;**

**}**

MPI\_Barrier**(**MPI\_COMM\_WORLD**);**

start **=** MPI\_Wtime**();**

//将a分成num段分发给各个进程

**if** **(**rank **==** 0**)**

//Scatter需要分进程执行，否则除主进程外其余进程不能接收到其散播的数据

MPI\_Scatter**(**a**,** num**,** MPI\_DOUBLE**,** b\_a**,** num**,** MPI\_DOUBLE**,** 0**,** MPI\_COMM\_WORLD**);**

**else**

MPI\_Scatter**(NULL,** num**,** MPI\_DOUBLE**,** b\_a**,** num**,** MPI\_DOUBLE**,** 0**,** MPI\_COMM\_WORLD**);**

//段内比较排序

sort**(**b\_a**,** b\_a **+** num**);**

//每段选出size个值

int p **=** 0**;**

**for** **(**int i **=** 0**;** i **<** num**;** i**+=**n**/(**size**\***size**))** **{**

sample**[**p**++]** **=** b\_a**[**i**];**

**}**

//将各个进程的sample数组里的size个样本元素汇集给0号进程的sample1数组中

MPI\_Gather**(**sample**,** size**,** MPI\_DOUBLE**,** sample1**,** size**,** MPI\_DOUBLE**,** 0**,** MPI\_COMM\_WORLD**);**

//主进程对size平方的值进行排序并且选出p个主元

**if** **(**rank **==** 0**)** **{**

//对p^2个主元进行排序

sort**(**sample1**,** sample1 **+** size **\*** size**);**

int k **=** 0**;**

**for** **(**int i **=** 0**;** i **<** size**-**1**;** i **++)** **{**

mainyuan**[**k**]** **=** sample1**[(**i**+**1**)\***size**];**

k**++;**

**}**

**}**

//将主进程的主元广播出去，让各个进程接收到相同的主元

MPI\_Bcast**(**mainyuan**,** size **-** 1**,** MPI\_DOUBLE**,** 0**,** MPI\_COMM\_WORLD**);** //将最终选取的主元广播出去

//找到各进程需要找到的下标,以及长度 找下标使用的是二分法

sendindex1**[**0**]** **=** sendlength1**[**0**]** **=** 0**;**

**for** **(**int i **=** 1**;** i **<** size**;** i**++)** **{**

//对每个进程里的主元求秩，找到他们对应的位置

int l **=** 0**,** r **=** num**-** 1**;**

double x **=** mainyuan**[**i **-** 1**];**

**while** **(**l **<** r**)** **{**

int mid **=** l **+** r **>>** 1**;**

**if** **(**b\_a**[**mid**]** **>** x**)** r **=** mid**;**

**else** l **=** mid **+** 1**;**

**}**

sendindex1**[**i**]** **=** r**;**

**}**

sendindex1**[**size**]** **=** num**;**//标兵下标

//根据找到的主元段下标算主元段的长度

**for** **(**int i **=** 0**;** i **<** size**;** i**++)** sendlength1**[**i**]** **=** sendindex1**[**i **+** 1**]** **-** sendindex1**[**i**];**

//全局交换发送和接收长度 各个进程将各自的主元段长度发给相应的进程

MPI\_Alltoall**(**sendlength1**,** 1**,** MPI\_INT**,** recvlength1**,** 1**,** MPI\_INT**,** MPI\_COMM\_WORLD**);**

//计算接收下标,totalnum=接收长度总长

int totalnum **=** 0**;**

**for** **(**int i **=** 0**;** i **<** size**;** i**++){**

recvindex1**[**i**]** **=** totalnum**;**

totalnum **+=** recvlength1**[**i**];**

**}**

recvindex1**[**size**]** **=** totalnum**;**

double**\*** result **=** **new** double**[**totalnum**];**

//将每个进程的b\_a数组从sendindex1发送sendlength1个长度给result数组，其接收长度为recvlength1，下标为recvindex1

MPI\_Alltoallv**(**b\_a**,** sendlength1**,** sendindex1**,** MPI\_DOUBLE**,** result**,** recvlength1**,** recvindex1**,** MPI\_DOUBLE**,** MPI\_COMM\_WORLD**);**

//各个进程对其中的主元段进行p路归并

Mergesort**(**result**,**recvindex1**,**totalnum**,**size **);**

//将各个进程算出来的总长度以及结果数组发送给主进程

**for** **(**int i **=** 0**;** i **<** totalnum**;** i**++)**

b**[**i**]** **=** result**[**i**];**

**if** **(**rank **!=** 0**)** **{**

MPI\_Send**(&**totalnum**,** 1**,** MPI\_INT**,** 0**,** 0**,** MPI\_COMM\_WORLD**);**

MPI\_Send**(**result**,** totalnum**,** MPI\_DOUBLE**,** 0**,** 0**,** MPI\_COMM\_WORLD**);**

**}**

**if(**rank**==**0**)**

**for** **(**int i **=** 1**;** i **<** size**;** i**++)** **{**

int recvsum**;**

MPI\_Recv**(&**recvsum**,** 1**,** MPI\_INT**,** i**,** 0**,** MPI\_COMM\_WORLD**,** **&**status**);**

MPI\_Recv**(**b **+** totalnum**,** recvsum**,** MPI\_DOUBLE**,** i**,** 0**,** MPI\_COMM\_WORLD**,** **&**status**);**

totalnum **+=** recvsum**;**

**}**

double end **=** MPI\_Wtime**();**

//校对结果

**if(**rank**==**0**&&**times**==**4**)**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)** **{**

**if** **(**b**[**i**]** **-** temp**[**i**]>**1e-6**)** **{**

cout **<<** i **<<** " | error | b[i]="**<<**b**[**i**]<<**" | temp[i]="**<<**temp**[**i**]** **<<** endl**;**

exit**(**0**);**

**}**

**}**

//输出结果

**if** **(**rank **==** 0**)** **{**

avg\_time**[**times**]** **=** end **-** start**;**

cout **<<** size **<<** "个进程并行时间为:" **<<** end **-** start **<<** endl**;**

**}**

//删除数组，节省空间

**delete[]** a**;**

**delete[]** b**;**

**delete[]** b\_a **;**

**delete[]** temp **;**

**delete[]** sample**;**

**delete[]** sample1 **;**

**delete[]** mainyuan **;**

**delete[]** sendindex1**;**

**delete[]** sendlength1**;**

**delete[]** recvindex1 **;**

**delete[]** recvlength1 **;**

**delete[]** result**;**

**}**

//结束MPI

MPI\_Finalize**();**

**if** **(**rank **==** 0**)** **{**

double alltime **=** 0**;**

**for** **(**int i **=** 0**;** i **<** 5**;** i**++)** **{**

alltime **+=** avg\_time**[**i**];**

**}**

cout **<<** "平均时间为：" **<<** alltime **/** 5 **<<** endl**;**

cout **<<** "加速比为：" **<<** **(**5 **\*** end1**)** **/** alltime**;**

**}**

**}**

3. 实验结果和分析：测试并行程序在不同进程数下的执行时间和加速比（串行执行时间/并行执行时间），并分析实验结果。其中，*n*固定为100000000，进程数分别取1、2、4、8、16、32、64时，为减少误差，每项实验进行5次，取平均值作为实验结果。

表1 并行程序在不同进程数下的执行时间（秒）和加速比

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 线程数  执行时间 | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
| 第1次 | 14.20 | 5.59 | 5.80 | 4.47 | 4.31 | 3.89 | 3.55 |
| 第2次 | 10.44 | 6.45 | 4.51 | 5.46 | 4.06 | 3.64 | 3.46 |
| 第3次 | 10.42 | 5.54 | 7.29 | 4.09 | 4.66 | 3.67 | 3.57 |
| 第4次 | 10.08 | 5.82 | 5.63 | 5.21 | 4.87 | 3.55 | 3.26 |
| 第5次 | 9.98 | 6.49 | 5.35 | 4.01 | 4.02 | 3.49 | 3.32 |
| 平均值 | 11.03 | 5.98 | 5.71 | 4.65 | 4.38 | 3.64 | 3.43 |
| 加速比 | 0.78 | 1.48 | 1.50 | 1.84 | 1.96 | 2.46 | 2.62 |