**深 圳 大 学**

**实 验 报 告**

**课程名称****：并行计算**

**实验名称：简单排序的OpenMP并行程序**

**姓 名：刘俊楠**

**学 号：2017303010**

**班 级：计科一班**

**实验日期：第8周实验课**

## 一. 实验目的

1. 实现简单排序的OpenMP并行程序；

2. 掌握for编译制导语句；

3. 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：排序数组*a*：

* 数组大小*n*和线程数*p*都是可输入的参数。
* 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）。
* 先将数组*a*分为*p*份，每个线程调用sort函数排序一份，然后调用merge函数将*p*个有序序列归并。
* 添加检测排序结果是否正确的代码。
* 添加计算排序时间的代码，注意不包含数组的初始化时间。

2. 程序代码和说明：

#include<iostream>

#include<algorithm>

#include <omp.h>

#include<vector>

using namespace std**;**

int n**,** p**;** //数组大小n和线程数p都是可输入的参数。

//判断数组是否有序

bool judge**(**double **\***a**,** double**\***c**)** **{**

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

**if** **(**a**[**i**]** **!=** c**[**i**])** **{**

cout **<<** "error!" **<<** endl**;**

**return** **false;**

**}**

**}**

**return** **true;**

**}**

//借助temp 数组b来合并两段有序的数组a

void merge**(**double **\***a**,** double **\***b**,** int first**,** int middle**,** int endaddress**)**

**{**

int i **=** first**,** j **=** middle **+** 1**,** k **=** first**;**

**while** **(**i **<=** middle **&&** j **<=** endaddress**)** **{**

**if** **(**a**[**i**]** **>** a**[**j**])** **{**

b**[**k**++]** **=** a**[**j**++];**

**}**

**else** **{**

b**[**k**++]** **=** a**[**i**++];**

**}**

**}**

//合并剩下多余的

**while** **(**i **<=** middle**)** b**[**k**++]** **=** a**[**i**++];**

**while** **(**j **<=** endaddress**)** b**[**k**++]** **=** a**[**j**++];**

//重新赋值

**for** **(**i **=** first**;** i **<=** endaddress**;** i**++)** a**[**i**]** **=** b**[**i**];**

**}**

int main**()**

**{**

clock\_t start**,** end**;**

//数组大小n和线程数p

cout **<<** "请输入数组大小：" **<<** endl**;**

cin **>>** n**;**

cout **<<** "请输入线程数：" **<<** endl**;**

cin **>>** p**;**

**for** **(**int pi **=** 0**;**pi **<** 7**;**pi**++)** **{**//每次线程数\*2，循环七次

cout **<<** "+++++++++++++++++++++++++++++++线程数为" **<<** p **<<** "时：++++++++++++++++++++++++" **<<** endl**;**

//记录并行运行时间的数组

double multi\_time**[**5**];**

//归并函数中用作辅助的数组temp

double**\*** temp **=** new double**[**n**];**

memset**(**temp**,** 0**,** n**);**

**for** **(**int times **=** 0**;**times **<** 5**;**times**++)** **{**//循环五次

cout **<<** "---------------------第" **<<** times **+** 1 **<<** "次并行计算：----------------------" **<<** endl**;**

start **=** 0**;**end **=** 0**;**

//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

double**\*** a **=** new double**[**n**];**//a数组存储并行结果

double**\*** b **=** new double**[**n**];**//b数组存储标准结果

omp\_set\_num\_threads**(**p**);**//设置并行线程数

int i**;**

//数组初始化

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

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

b**[**i**]** **=** a**[**i**];**

**}**

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

//并行开始计时

start **=** clock**();**

#pragma omp parallel shared(a,temp) private(i)

**{**

//对数组分成p路排序

int num **=** n **/** p**;**

#pragma omp for schedule(static,1)

**for** **(**i **=** 0**;** i **<** p**;** i**++)** **{**

sort**(**a **+** i **\*** num**,** a **+** **(**i **+** 1**)** **\*** num**);**

**}**

**}**

//利用temp数组对部分排序的a数组进行归并

**for** **(**int block **=** n **/** p**;** block **<** n**;** block **+=** block**)**

**{**//分块归并，每次循环扩大块一倍

**for** **(**int first **=** 0**;** first **<** n **-** block**;** first **+=** **(**block **+** block**))**

**{**//确定块内数组first middle end 的address

int middle **=** first **+** block **-** 1**;**

int endaddress **=** first **+** block **+** block **-** 1**;**

**if** **(**endaddress **>=** n**)** **{**

endaddress **=** n **-** 1**;**

**}**

merge**(**a**,** temp**,** first**,** middle**,** endaddress**);**

**}**

**}**

end **=** clock**();**

//存储并行时间

multi\_time**[**times**]** **=** **(**end **-** start**)** **/** 1000.0**;**

cout **<<** "并行时间为" **<<** multi\_time**[**times**]** **<<** endl**;**

//判断并行的结果数组是否有序

judge**(**a**,** b**);**

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

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

**}**

cout **<<** "--------------------------------------串行计算---------------------------------" **<<** endl**;**

double**\*** c **=** new double**[**n**];**//串行测试数组

double**\*** d **=** new double**[**n**];**//标准数组

memset**(**temp**,** 0**,** n**);**

//数组初始化

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

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

d**[**i**]** **=** c**[**i**];**

**}**

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

//开始计时

start **=** clock**();**

//按照线程分开排序

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

sort**(**c **+** i **\*** **(**n **/** p**),** c **+** **(**i **+** 1**)** **\*** **(**n **/** p**));**

**}**

//p路归并

**for** **(**int block **=** n **/** p**;** block **<** n**;** block **+=** block**)**

**{**

**for** **(**int first **=** 0**;** first **<** n **-** block**;** first **+=** **(**block **+** block**))**

**{**

int middle **=** first **+** block **-** 1**;**

int endaddress **=** first **+** block **+** block **-** 1**;**

**if** **(**endaddress **>=** n**)** **{**

endaddress **=** n **-** 1**;**

**}**

merge**(**c**,** temp**,** first**,** middle**,** endaddress**);**

**}**

**}**

end **=** clock**();**

//判断是否数组已经有序

judge**(**c**,** d**);**

double single\_time **=** **(**end **-** start**)** **/** 1000.0**;**

cout **<<** "串行消耗时间为：" **<<** single\_time **<<** "秒" **<<** endl**;**

//计算并行平均时间

double average**;**

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

average **+=** multi\_time**[**i**];**

**}**

average **/=** 5**;**

cout **<<** "并行平均时间为：" **<<** average **<<** "秒" **<<** endl**;**

cout **<<** "平均加速比为" **<<** single\_time **/** average **<<** endl**;**

p **\*=** 2**;**

delete **[]**c**;**

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

**}**

**}**

3. 实验结果和分析：测试并行程序在不同线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，数组大小*n*固定为100000000，线程数分别取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 并行程序在不同线程数下的执行时间（秒）和加速比

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 线程数  执行时间 | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
| 第1次 | 10.025 | 6.357 | 5.682 | 5.563 | 6.024 | 6.537 | 6.683 |
| 第2次 | 9.828 | 6.189 | 5.29 | 5.347 | 5.542 | 6.039 | 6.504 |
| 第3次 | 9.906 | 6.356 | 5.21 | 5.243 | 5.505 | 6.466 | 6.478 |
| 第4次 | 9.79 | 6.269 | 5.274 | 5.207 | 5.478 | 6.076 | 6.544 |
| 第5次 | 9.791 | 6.219 | 5.034 | 5.363 | 5.889 | 6.171 | 6.461 |
| 平均值 | 9.868 | 8.2516 | 6.94832 | 6.73426 | 7.03445 | 7.66469 | 8.06694 |
| 加速比 | 1.00466 | 1.25527 | 1.56498 | 1.71867 | 1.71442 | 1.65643 | 1.63668 |