Homework7

洪方舟

2016013259

Email: hongfz16@163.com

2018 年 4 月 21 日

1. 实验目的

a. 使用 *openmp* 平台编写归并排序与快速排序的并行版本
b. 通过实验验证理论是否与实际相符
c. 掌握多线程编程方法

2. 实验环境

操作系统：Windows 10
处理器：Intel Core i7-7700k CPU @ 4.20GHz × 8
编程语言：C++
IDE: Visual Studio

3. 实验方法

a. 首先编写无符号整形，并行版本的归并排序与快速排序
b. 容易通过遍历来验证正确性
c. 使用不同的数量级的数据测试两种算法
d. 将测试结果与理论情况比较并进行分析
e. 将测试结果与非并行版本进行比较

4. 实验结果

Table 1: 不同数量级下几种排序算法所耗时间 (ms)

数组长度	归并排序（并行）	快速排序（并行）	归并排序（非并行）	快速排序（非并行）
10^4	3	0	0.916	0.838
10^5	49	5	6	4
10^6	474	53	84	54
10^7	5024	563	951	638
10^8	57867	7130	11069	7276
10^9	687494	70697	129550	81765

5. 分析与总结

a. 可以看到，两种排序方法使用 *openmp* 的并行版本并没有达到预期的效率，归并排序的并行版本的效率有大幅的下降，而快速排序则只有一点微小的效率提升。
b. 横向比较发现快速排序的速度在并行实现上依然远高于归并排序。
c. 并行版本的速度没有得到提升的可能原因有三点：首先 *openmp* 平台为了规划并行消耗了较多的计算资源；其次，每次申请一个新的线程都需要耗费一定的时间；最后，由于 *Visual Studio* 平台上的 *openmp* 版本较老，不支持 *task* 功能，因此在递归过程中实际上并没有充分利用计算机上富余的物理内核；如果强行开启 *nest* 选项，则会由于递归过深创建线程个数过多反而导致效率下降甚至爆栈。
e. 在现代内核上，由于单核的计算速度已经足够快，并且有缓存机制作为保障，使得这种基础算法没有太大的必要使用多线程运行来提升效率，有时候反而会因为多开线程而导致额外的开销。

6. 源代码及可执行文件说明

a. 源代码存放在 *src/psort* 文件夹下。
b. 可执行文件存放在 *bin/psort* 文件夹下。