实验一：并行排序编程实验

一、实验内容与方法

用 pthreads 编程实现并行快速排序程序，编程语言：C/C++

– 数组大小≥2000 万，数组元素为双精度浮点数

– 至少测试线程个数为 1、2、3、4 时的性能，如平台允许，可更多

– 注意：

• 快速排序算法时间复杂性与数据相关，测试时注意数据集的一致性

• 统计计算时间时，应排除准备数据的时间(如果有 I/O)

• 并行程序的性能可扩展性：线程个数可变、局部性、…

• 快速排序算法时间复杂度为 O(nlogn)，流程如下：

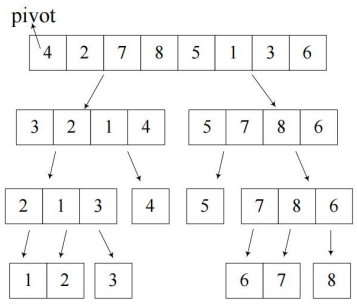
① 从数组序列中选出一个元素，称为中间值(pivot)

② 把数组序列分成两个子序列，比中间值小的元素放在一个子序列，比中间值大的元素放在另一个子序列。该操作称为分割(partition)

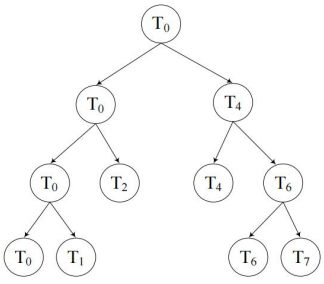
③ 递归地对两个子序列分别重复上述操作，直到子序列中元素个数为 1

• 典型的并行化方法

– 从一个线程开始，递归产生的两个子序列中，一个留给自己处理，另一个传递给另一个线程，由此产生一个树结构



排序过程



工作分配

注意并行划分和粒度控制
•线程个数如何控制？
•每个线程负责哪些任务？

实验设备：MacbookAirM3

实验平台：clion+c+++pthreads 编程

二、实验过程

并行快速排序算法思路

1、快速排序算法：

(1) 从数组中选择一个基准值（pivot）。

(2) 将数组分为两个子序列：一个子序列的元素小于基准值，另一个子序列的元素大于基准值。

(3) 递归地对这两个子序列进行快速排序，直到子序列只剩一个元素 。

2、并行化方法：

(1) 初始线程执行快速排序，递归地将数组划分为子序列。

(2) 当子序列达到一定大小时，创建新的线程来处理子序列，从而实现并行排序 。

(3) 需要控制线程的数量，避免过多的线程创建开销。

三、实验结果与分析

1、实验结果

```
/Users/liujiaojiao/CLionProjects/Parallel/cmake-build-debug/quicksort
数组大小: 20000000
1 线程排序耗时: 3883 毫秒
2 线程排序耗时: 2008 毫秒
4 线程排序耗时: 1083 毫秒
8 线程排序耗时: 763 毫秒

进程已结束，退出代码为 0
```

图 1 quicksor 执行结果

线程数	运行时间	加速比	理论最大加速比(线性)	效率
1	3883	1.0x	1x	1
2	2008	1.93x	2x	0.965
4	1083	3.58x	4x	0.895
8	763	5.08x	8x	0.636

表 1 加速比

2、可扩展性分析

(1) 随着线程数的增加，排序时间减少，加速比增加，这表明并行化提高了排序性能。

(2) 但是，加速比并不是线性增长的，效率随着线程数的增加而降低，这说明存在并行开销，

例如线程创建、同步和数据合并等。

(3) 8 线程时的效率明显下降，可能表明此时并行开销已经比较显著，或者硬件资源已经接近瓶颈。