并行算法分析与设计

实验内容

将串行快速排序算法改造成并行算法，并基于 pthread 实现，与串行算法做出比较

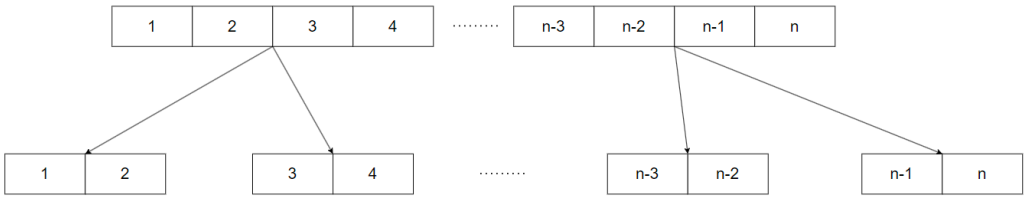
一、实验平台参数

- CPU: 13th Gen Intel(R) Core(TM) i9-13900HX 2.20 GHz 24 个内核
- 内存: 16G
- 操作系统: Windows 11
- IDE: Trae

二、算法原理

并行快速排序算法，采用了分而治之的思想

首先，将数据分区，将整个数据集均匀地分成与线程数量相等的连续区间，每个线程负责对一个区间进行排序将整个数据集均匀地分成与线程数量相等的连续区间，每个线程负责对一个区间进行排序。



然后对于每个数据分区，创建一个线程实现局部排序，选择最右元素作为基准。分区时将小于基准的元素放在左侧，大于基准的放在右侧，递归处理左右子区间。

最后同步各个线程，使用 pthread 屏障(barrier)机制确保所有线程完成局部排序后，主线程才开始合并操作。这避免了主线程在其他线程未完成排序前就开始合并，保证了算法的正确性。当所有线程完成局部排序后，主线程执行多路归并，将多个已排序区间合并成一个完整的有序数组

三、实验准备

使用 generate_test.cpp 生成测试数据，生成 8 种不同规模的测试数据文件（从 100 到 10 亿个数）每个文件中包含随机生成的整数，范围在 1 到 1,000,000 之间

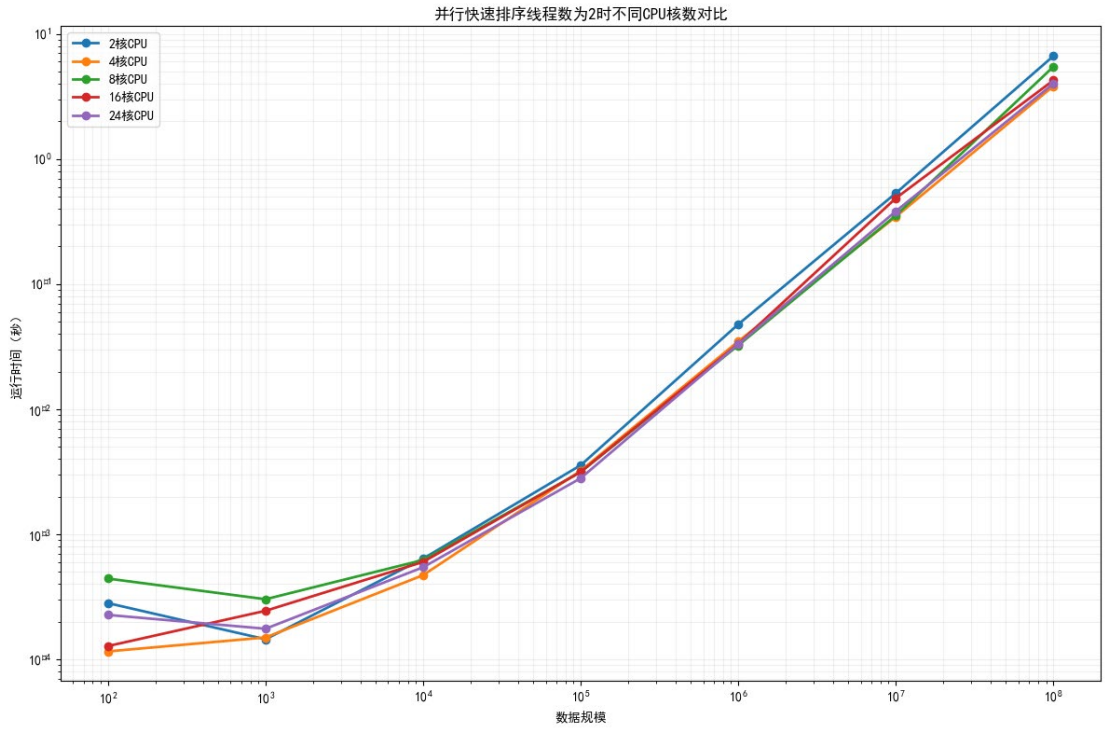
五、实验结果分析

实验采用了控制变量方法，分别控制 CPU 核数和线程数两个变量，来分析并行快速排序的实验结果，所有黄色的标记都是时间最短的

控制 CPU 数量，线程数为 2，并行阈值为 10000，单位均为（秒/s）

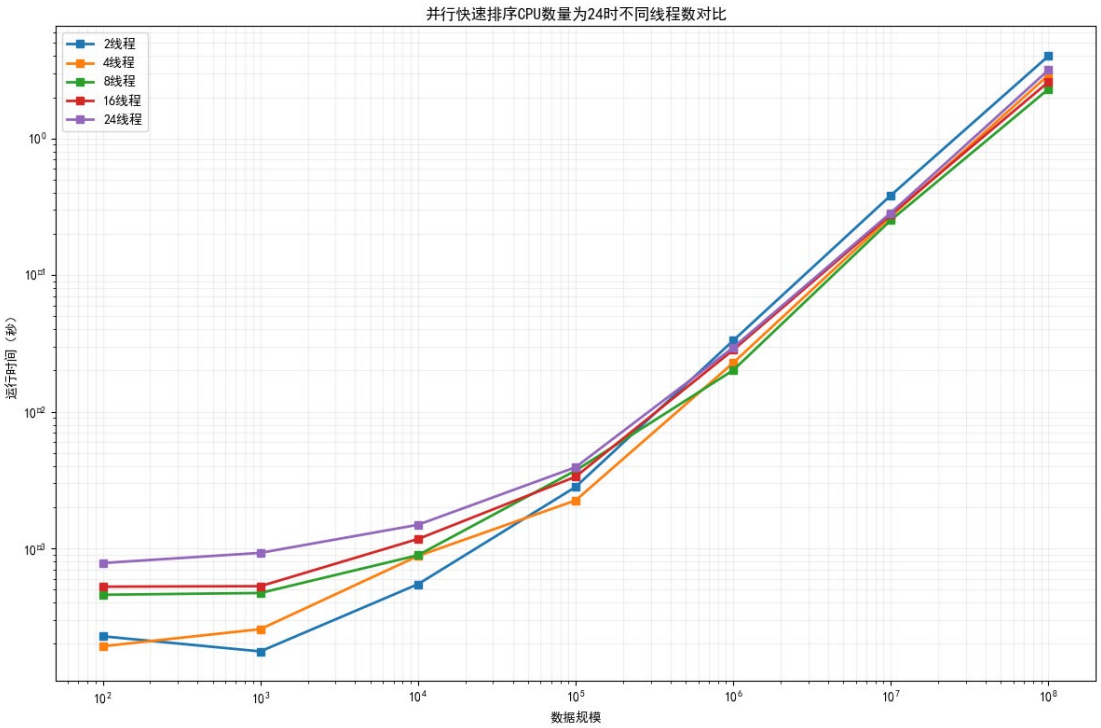
CPU 数	100	1000	1 万	10 万	100 万	1000 万	1 亿

2 核	0.000282	0.000145	0.00064	0.003568	0.04777	0.533648	6.653
4 核	0.000133	0.000146	0.00048	0.003599	0.03854	0.407283	4.03388
8 核	0.000443	0.000303	0.00062	0.003149	0.03239	0.353737	5.46609
16 核	0.000128	0.000245	0.00060	0.003155	0.03334	0.48597	4.2534
24 核	0.000227	0.000176	0.00054	0.00281	0.03310	0.382442	4.00138

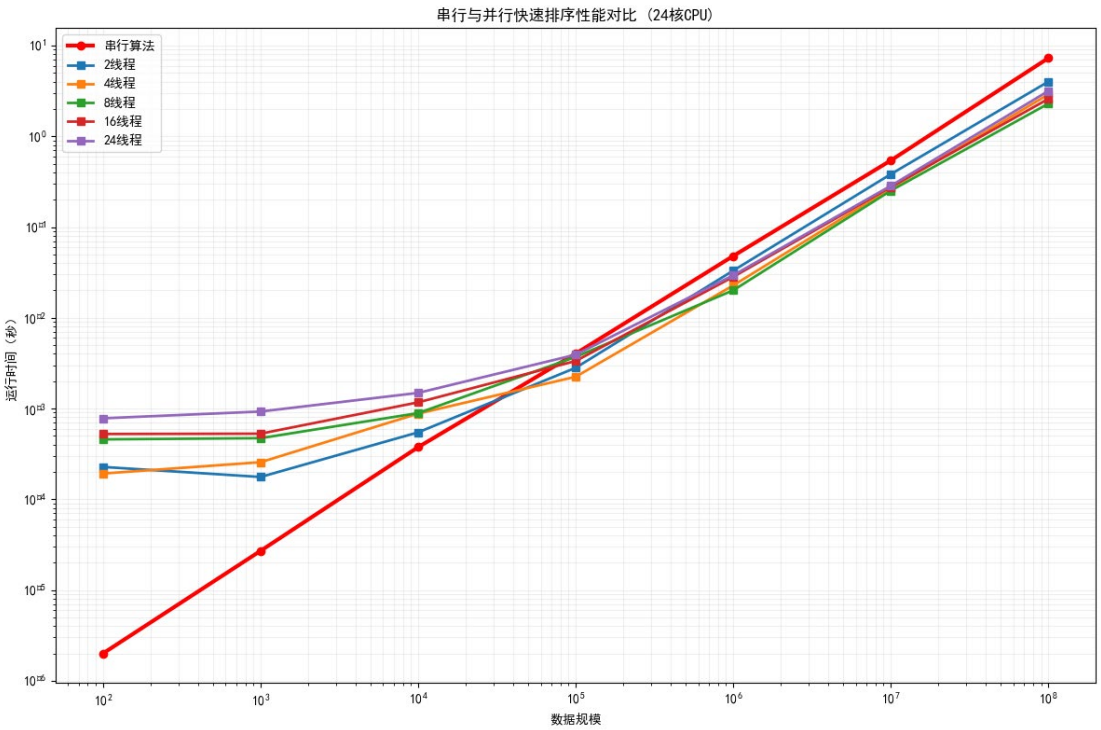


控制线程数量，CPU 数量为 24，并行阈值为 10000，单位均为（秒/s）

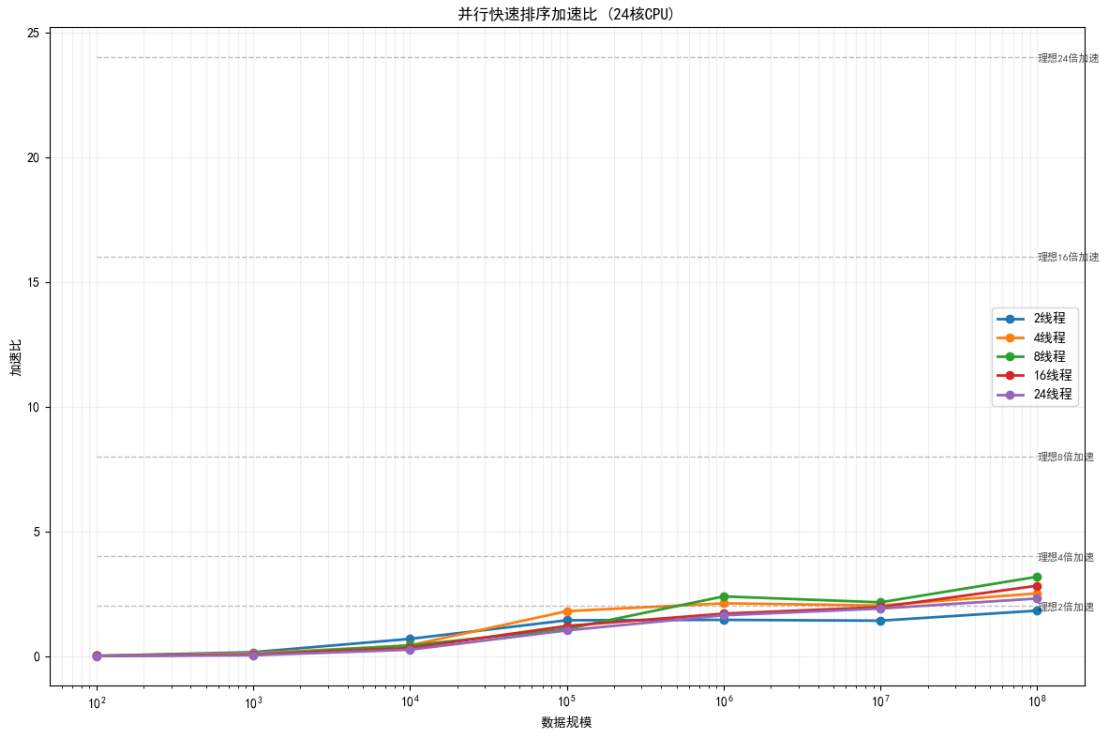
线 程 数	100	1000	1 万	10 万	100 万	1000 万	1 亿
2	0.000227	0.000176	0.00054	0.00281	0.03310	0.382442	4.00138
4	0.000192	0.000256	0.00087	0.002239	0.02272	0.267387	2.89689
8	0.000457	0.000471	0.00089	0.003704	0.02007	0.251689	2.29035
16	0.000524	0.000528	0.00116	0.003339	0.028254	0.276288	2.5834
24	0.00078	0.000926	0.00148	0.003919	0.029414	0.284623	3.1573



串行算法和并行算法的性能对比



并行算法的加速比



四、实验结果

通过对实验数据的分析，我发现并行快速排序的性能受多种因素影响，并不是简单的"核数越多越快"或"线程越多越快"的关系。

控制 CPU 核数（线程数固定为 2）的分析

从数据可以看出:

- 对于小规模数据（100-10000），CPU 核数对性能影响不明显，甚至有时核数少的反而更快
- 对于中等规模数据（10 万-1000 万），8 核 CPU 表现最佳
- 对于大规模数据（1 亿），24 核 CPU 性能最优

原因分析:

- 并行开销与收益平衡 小规模数据下，线程创建和管理的开销可能超过并行带来的收益
- 核数增加会带来更多的缓存一致性维护开销
- 内存带宽限制 ：快速排序是内存密集型算法，当核数增加时，内存访问竞争加剧，8 核可能是当前硬件架构下内存带宽与计算能力的最佳平衡点

控制线程数（CPU 核数固定为 24）的分析

从数据可以看出:

- 小规模数据（100-1000）下，2-4 线程性能最佳
- 中等规模数据（10 万-1000 万）下，4-8 线程性能最佳
- 大规模数据（1 亿）下，8 线程性能最优，而非 24 线程

原因分析:

- 线程调度开销，程数增加会导致更多的上下文切换和调度开销。程数超过实际工作负载需求时，这种开销会抵消并行带来的收益
- 负载均衡问题，排序的工作负载可能不均衡，某些分区比其他分区需要更多处理时间。过多线程可能导致一些线程闲置，而其他线程过载。
- 归并阶段的瓶颈，代码中的归并操作是单线程执行的，线程数增加只能加速排序阶段。当线程数过多时，归并阶段成为主要瓶颈
- 资源竞争，线程数超过物理核心数会导致线程间竞争 CPU 资源，24 线程在 24 核 CPU 上理论上可以一一对应，但实际上会有其他系统线程占用资源

算法实现也存在一些问题

由于是静态分区，这使得有一些提前完成计算的线程，但可能导致负载不均衡，特别是当数据分布不均匀时。

五、代码实现

我已经放到 github 上了请参考：[h161020716/Parallel-Algorithm-Analysis-and-Design]