多线程排序算法 实验报告

吴佳龙 2018013418

摘要

本次实验对多线程的归并排序和快速排序进行了算法分析和编程实现，验证了算法实现的结果正确性，并比较了不同设置下排序算法的效率，初步体会了多线程算法的特点。

1 问题

在 OpenMP 平台上实现并比较多线程的归并排序和快速排序算法。

2 实验环境

操作系统：Windows 10

IDE：Visual Studio 2019, OpenMP 2.0

处理器：3.1 GHz 双核 Intel Core i5

3 并行归并排序

3.1 算法描述与分析

将串行的归并排序的两个递归过程并行，之后的归并仍然串行，则这样算法的功仍为

$$T_1(n) = 2T_1(\frac{n}{2}) + \Theta(n) = \Theta(n \lg n)$$

持续时间为

$$T_\infty(n) = T_\infty(\frac{n}{2}) + \Theta(n) = \Theta(n)$$

若将归并并行，参见课件上 P-MERGE 过程，P-MERGE 的持续时间为

$$M_\infty(n) = M_\infty(\frac{3}{4}n) + \Theta(\lg n) = \Theta\left(\lg^2 n\right)$$

这样总的归并排序的持续时间为

$$T_\infty(n) = T_\infty(\frac{n}{2}) + \Theta(\lg^2 n) = \Theta(\lg^3 n)$$

[1] https://en.wikipedia.org/wiki/Segment_tree

3.2 实际优化

串行归并 虽然并行的归并过程 P-MERGE 带来了理论上的优化，但在实验过程中我们发现：可能由于多线程调度带来大常数因子的时间开销，调用 P-MERGE 过程的归并排序并没有串行归并的归并排序实际排序效率高。

截断 理论分析中假设的多线程模型可能过于理想，且在个人计算机上计算资源有限，为了实际降低排序时间，我们设定在并行递归深度达到一个阈值之后，就转而调用原始的串行归并排序算法。实验表明这能够降低排序时间。

4 并行快速排序

4.1 算法描述与分析

将串行的快速排序的两个递归过程并行，Partition 操作仍然串行，则算法的功和持续时间与串行归并的归并排序相同：

$$T_1(n) = 2T_1(\frac{n}{2}) + \Theta(n) = \Theta(n \lg n)$$

$$T_\infty(n) = T_\infty(\frac{n}{2}) + \Theta(n) = \Theta(n)$$

4.2 Partition 的并行化

Partition 操作也可并行化，参见 CLRS Exercise 27.3-3。由于实现较为复杂，且与 P-MERGE 类似，实现并行的 Partition 可能反而会带来实际效率的降低，我们仅简要描述其算法流程：

1. 在待进行 Partition 操作的数组区间上构建一棵 Segment tree [1]

2. 通过在 Segment tree 分治的递归操作，可以对于 $a_{l..r}$ 中的每个元素计算出 $low_{l..r}$ 和 $high_{l..r}$，其中 low_j 表示在 a_j 左边小于等于主元 $pivot$ 的数的个数，$high_j$ 表示在 a_j 右边大于 $pivot$ 的数的个数

3. 依据 low 和 $high$ 将 a 中的每个元素移动到临时数组 tmp 中 Partition 过程后对应的位置，然后将 tmp 复制回 a

4.3 实际优化

与归并排序类似，我们也使用了**截断**的技巧，即在并行归并的层数达到一定阈值后调用串行的排序算法。

5 结果分析

5.1 结果正确性

首先我们验证了算法实现的正确性，验证方法为：给定 n，随机生成长度为 n 的 32 位无符号整数数组，并调用不同并行或串行的排序算法，比较他们的排序结果。

实验表明，对于随机生成的数据，不同算法给出的排序结果总是相同的，由此可以认为我们对于算法的实现是无误的。

5.2 排序时间

在本次实验中，设置 OpenMP 的参数 num_procs=4, num_threads=4；**截断**都发生在递归深度达到 3 的时候，也就是已经产生了递归树上的 8 个叶子节点后转而调用串行的排序。

令 $n \in [10^6, 10^7]$，不同设置的排序算法的排序时间统计如表 1 和图 1。

P-MERGE 的影响 可以看到，使用了 P-MERGE 的归并排序，排序效率都慢于使用串行归并的归并排序。尤其是，在表格中第三行，使用了 P-MERGE 的没有截断的归并排序所需的时间远远超过了其他算法。可能的原因是：分治实现的 P-MERGE 带来的多线程调度的时间花销的常数因子较大。

截断的影响 比较不截断与截断的归并排序和快速排序，可以看到截断可以带来排序时间的降低。这可能是因为减少了多线程调度的时间花销。这说明人为的干预可以实现更优的资源分配。

不同的排序算法 归并排序的不同版本中，最优的是：不使用 P-MERGE 的且使用截断的并行归并排序，它大约能比串行的归并排序减少 30% 的排序时间。

快速排序的不同版本中，较优的是：串行的快速排序和使用截断的并行快速排序。并行大约能够减少 10% 的排序时间。

实际上，以上三者的排序时间差异并不显著。

6 总结

本次实验通过对多线程的归并排序和快速排序的理论时间和实际计算时间的分析，初步感受了多线程算法的特点，并认识到了理论分析和工程实践之间的差异。

算法	P-MERGE or P-Partition	截断	n ($\times 10^6$)									
			1	2	3	4	5	6	7	8	9	10
串行归并排序			0.12	0.26	0.38	0.53	0.66	0.80	0.95	1.10	1.26	1.39
并行归并排序			0.26	0.45	0.84	0.90	1.03	1.59	1.38	1.98	1.82	1.98
	✓		4.06	8.41	10.72	13.24	23.57	20.36	31.52	25.03	28.53	32.02
		✓	0.09	0.18	0.24	0.33	**0.42**	0.50	0.59	0.67	0.77	**0.84**
	✓	✓	0.87	1.39	1.95	2.57	3.23	3.77	4.52	5.02	5.47	7.60
串行快速排序			**0.08**	0.18	0.26	0.35	0.46	0.55	0.63	0.74	0.83	0.94
并行快速排序			0.24	0.38	0.55	0.76	1.00	1.01	1.34	1.44	1.68	2.12
		✓	0.09	**0.16**	**0.23**	**0.32**	**0.42**	**0.41**	**0.58**	**0.57**	**0.70**	0.89

Table 1: 不同算法在不同输入规模下的排序时间（秒）：“P-MERGE or P-Partition”一列表示是否使用并行化的 Merge 或者 Partition；“截断”一列表示是否在递归深度达到阈值后调用串行算法。每一列最优的排序时间用粗体标出。

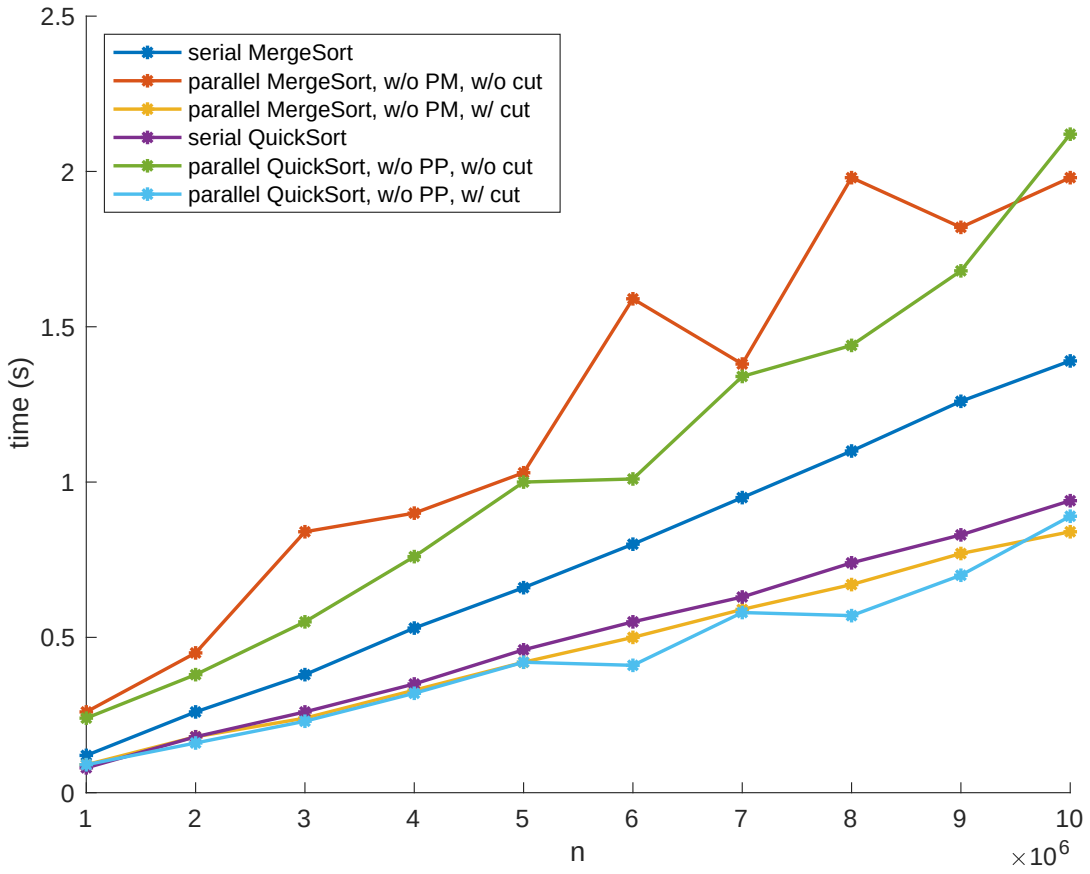

Figure 1: 不同算法在不同输入规模下的排序时间：PM 或者 PP 表示 P-MERGE 或者 P-Partiton；cut 表示截断