Week 15

软73 沈冠霖 2017013569

June 6, 2019

1 T1

1 思路：先初始化C让C全是0，然后每次分治先并行算出前四个乘积加到C上，再并行算出后四个乘积加到C上。伪代码如下。

```
1 n = A.rows
2 if n == 1
3       c11 += a11b11
4 else partition A, B,C into n/2 × n/2 submatrices: A11, A12, A21, A22; B11,
B12, B21, B22;C11,C12,C21,C22;respectively
5       spawn P-MATRIX-MULTIPLY-RE(C11, A11, B11)
6       spawn P-MATRIX-MULTIPLY-RE(C12, A11, B12)
7       spawn P-MATRIX-MULTIPLY-RE(C21, A21, B11)
8       P-MATRIX-MULTIPLY-RE(C22, A21, B12)
9       sync
10        spawn P-MATRIX-MULTIPLY-RE(C11, A12, B21)
11        spawn P-MATRIX-MULTIPLY-RE(C12, A12, B22)
12        spawn P-MATRIX-MULTIPLY-RE(C21, A22, B21)
13        P-MATRIX-MULTIPLY-RE(C22, A22, B22)
14        sync
```

2 工作量：$T_1(n) = 8T_1(\frac{n}{2}) + \theta(1), T_1(1) = \theta(1)$
。持续时间：$T_\infty(n) = 2T_\infty(\frac{n}{2}) + \theta(1), T_\infty(1) = \theta(1)$

3 根据主定理，工作量$T_1 = \theta(n^3)$，持续时间$T_\infty = \theta(n)$,并行度为$\theta(n^2)$。
n=1000时，并行度为10^6，是先前算法10^7的十分之一。

2 T2

测试环境 CPU:Inter Core i5-6300HQ,2.3GHZ
内存：12G
环境：VS2017,release模式 比较不同数据规模下串行和并行归并，快排的运行时间。两个都用的naive算法。

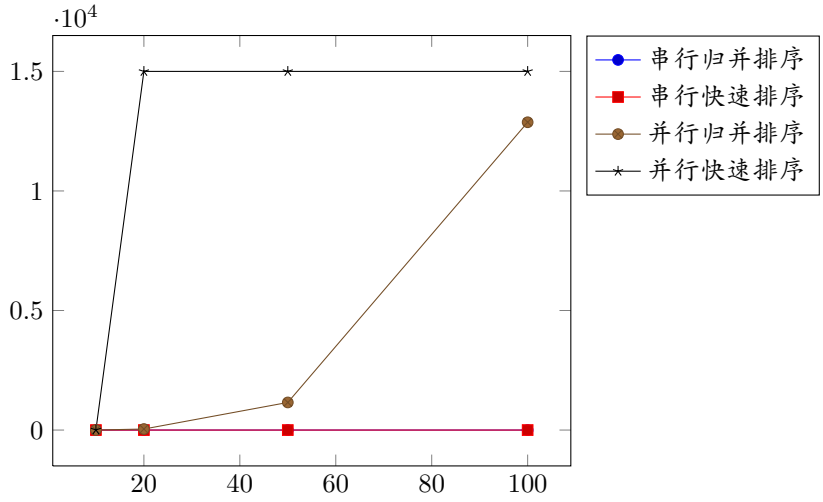
为了保证数据安全，我在并行的每个函数都开了全新的数组，复制元素过来进行排序，因此会自带$\theta(n)$的复杂度。

结果分析　程序目前来看结果是正确的。
可以看出，使用多线程在递归上是非常慢的，并行归并比串行慢了成千上万倍，而且随着数据的增大，每增大2倍时间要增加10倍以上。我猜测是线程开的太多了，每个线程的算力太低了。并行快排比归并还慢很多，我猜测是线程切分不均匀导致的。

Table 1: 不同数据规模下排序的时间

测量序号	1	2	3	4
数据范围	10	20	50	100
串行归并排序时间 (ms)	0	0	0.2	0
串行快速排序时间 (ms)	0	0	0	0
并行归并排序时间 (ms)	1.2	42.8	1154.6	12872.8
并行快速排序时间 (ms)	6	太大	太大	太大

注：每组数据都是运行5次后取的平均值



注：时间过大我用15s来表示了。