作业 6 Parallel

1 理论分析

单线程时，易知计算斐波那契数列的时间复杂度为 $O(2^n)$。

在多线程时，时间复杂度与可创建的最大线程数有关，最大线程数越大，时间复杂度越低；此外，创建和销毁线程会带来一定的开销。

在不考虑创建和销毁的开销情况下，对多线程的时间复杂度进行定性分析。假设线程个数超过 n，则在递归树中，每一层都可以同时运算，时间复杂度为 O(n)。线程个数为 k（k<n）时，可以认为先以 O(k) 的时间复杂度运行前 k 层，此后并行进行 k 次单线程 $O(2^{n-log_2^k})$ 操作，总时间复杂度为 $O(2^{n-log_2^k})+O(k)$。

但实际情况下，时间复杂度不可能下降至 $O(2^{n-log_2^k})+O(k)$。不能简单地认为 O(1) 的线程相关操作（创建、同步、销毁）和 O(1) 的单线程计算所用时间等价。此外，多线程计算可能还涉及到任务分配不均、CPU 核心数量不足等问题。

2 实验结果与分析

k（最大线程数）分别为 2、4、8、16 时，某次单线程与多线程得到的第 35 个斐波那契数、耗时、耗时比分别如图 1、图 2、图 3和图 4。

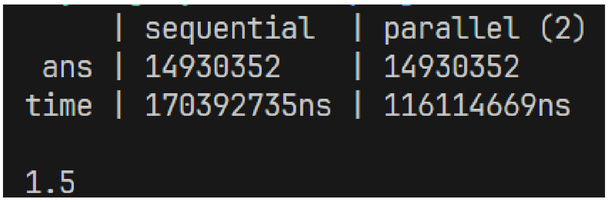


图 1: k=2 时的测试结果

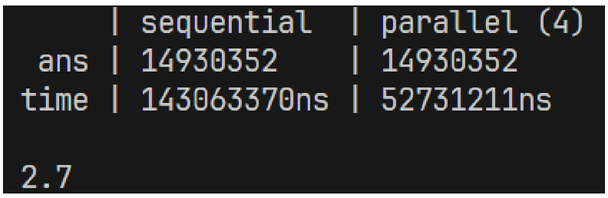
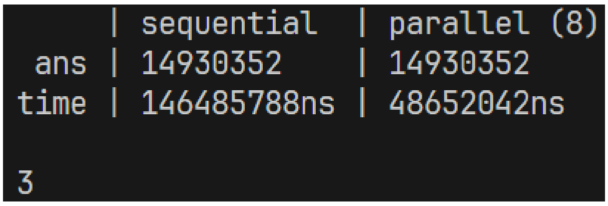
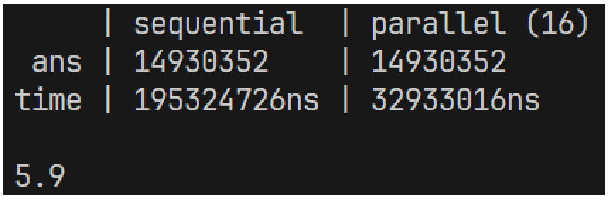


图 2: k=4 时的测试结果


```
     | sequential   | parallel (8)
 ans | 14930352     | 14930352
time | 146485788ns | 48652042ns

3
```

图 3: k=8 时的测试结果


```
     | sequential   | parallel (16)
 ans | 14930352     | 14930352
time | 195324726ns | 32933016ns

5.9
```

图 4: k=16 时的测试结果

各运行 20 次后，得到的平均耗时比如表 1。

最大线程数	2	4	8	16
平均耗时比	1.6	2.5	3.4	4.0

表 1: k=2、4、8、16 时的平均耗时比、加速比

最大线程数与时间复杂度关系折线图如图 5。

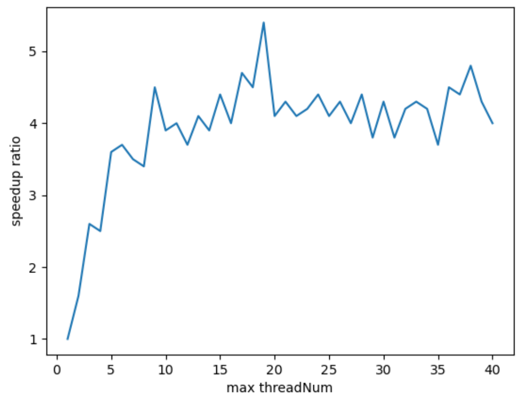


图 5: 最大线程数与时间复杂度关系折线图

可以看到，并行远远达不到 $O(2^{n-log_2^k})+O(k)$ 的时间复杂度，但多线程运算确实起到了提升性能的效果。对于“计算第 35 个斐波那契数”这一测试，在线程增加到 15 左右时，线程增加不再带来显著的性能提升。并且，并行的效率受任务分配、CPU 核心数等参数影响，性能会出现波动。

3 结论

并行可以提升性能，但是受到 CPU 核心数、最大线程数、线程开销等的制约。在一定程度内，增加线程数可以提升性能，但是线程数足够多时，性能提升不会再增加。