实验目标

测量多线程 FFT 程序运行时间，考察线程数目增加时运行时间的变化。

实验要求

- 采用 C/C++编写程序，选择合适的运行时间测量方法
- 根据自己的机器配置选择合适的输入数据大小 n，保证足够长度的运行时间
- 对于不同的线程数目，建议至少选择 1 个，2 个，4 个，8 个，16 个线程进行测试
- 回答思考题，答案加入到实验报告叙述中合适位置

思考题

1. pthread 是什么？怎么使用？

Pthread 是线程，通过创建分配消除来完成一个任务得各个部分。

2. 多线程相对于单线程理论上能提升多少性能？多线程的开销有哪些？

理论上来说多线程可以将一个任务分为 n 个线程，如果 n 小于 cpu 的最大核心数，则性能能提升 n 倍。但多线程要求创建线程，创建线程时间较大。

3. 实际运行中多线程相对于单线程是否提升了性能？与理论预测相差多少？可能的原因是什么？

实际并没有提升甚至总的时间还增大了。可能的原因为创建线程时间较长。

实验内容

多线程 FFT 代码

多线程 FFT 的代码可以参考这里。

该代码采用了 pthread 库来实现多线程，其中...(此处请补充 pthread 的使用)

多线程 FFT 程序性能分析

通过分析多线程 FFT 程序代码，可以推断多线程 FFT 程序相对于单线程情况可达到的加速比应为：

(此处请补充多线程 FFT 程序代码的性能分析)

测试

测试平台

在如下机器上进行了测试：

部件	配置
CPU	core i5-6500U
内存	DDR4 8GB
操作系统	Ubuntu 1604 LTS

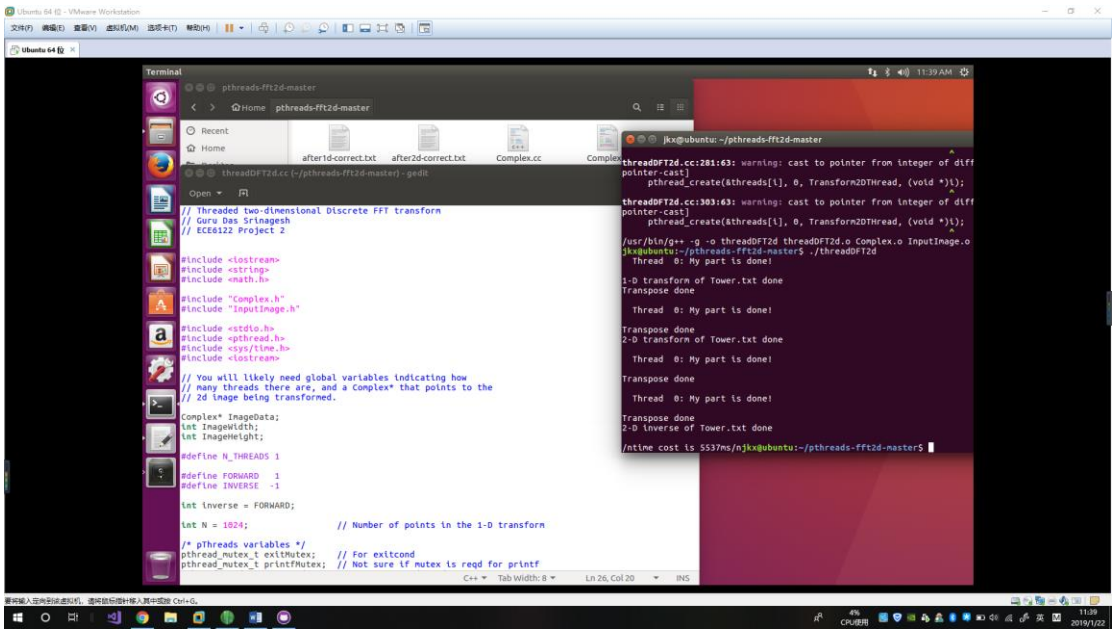
测试记录

多线程 FFT 程序的测试参数如下：

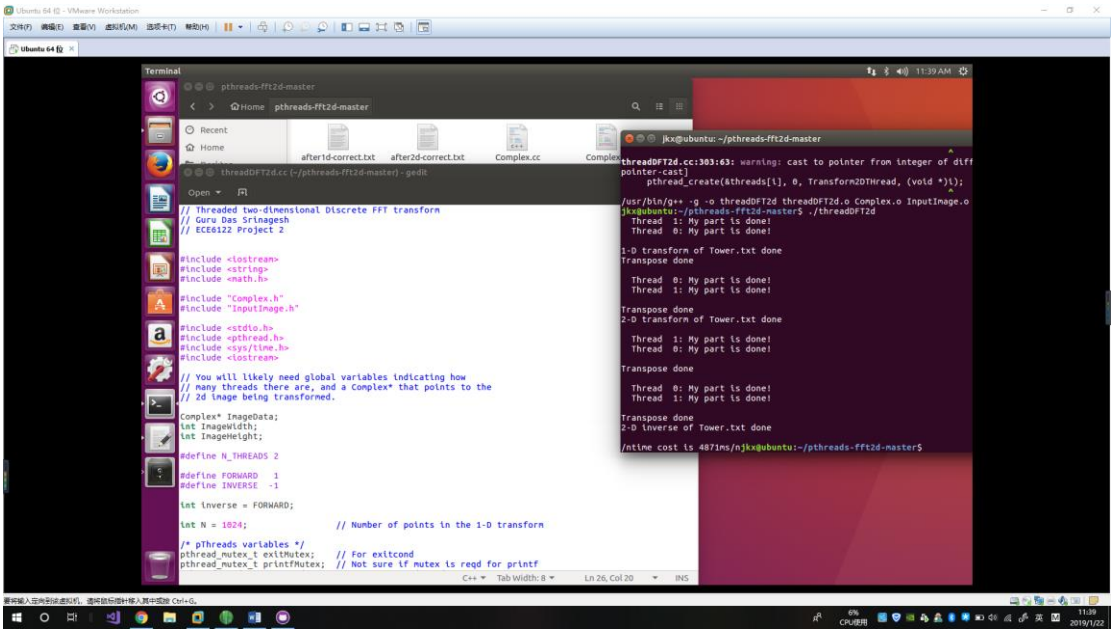
参数	取值
数据规模	1024 或其它
线程数目	1,2,4,8,16,32

多线程 FFT 程序运行过程的截图如下：

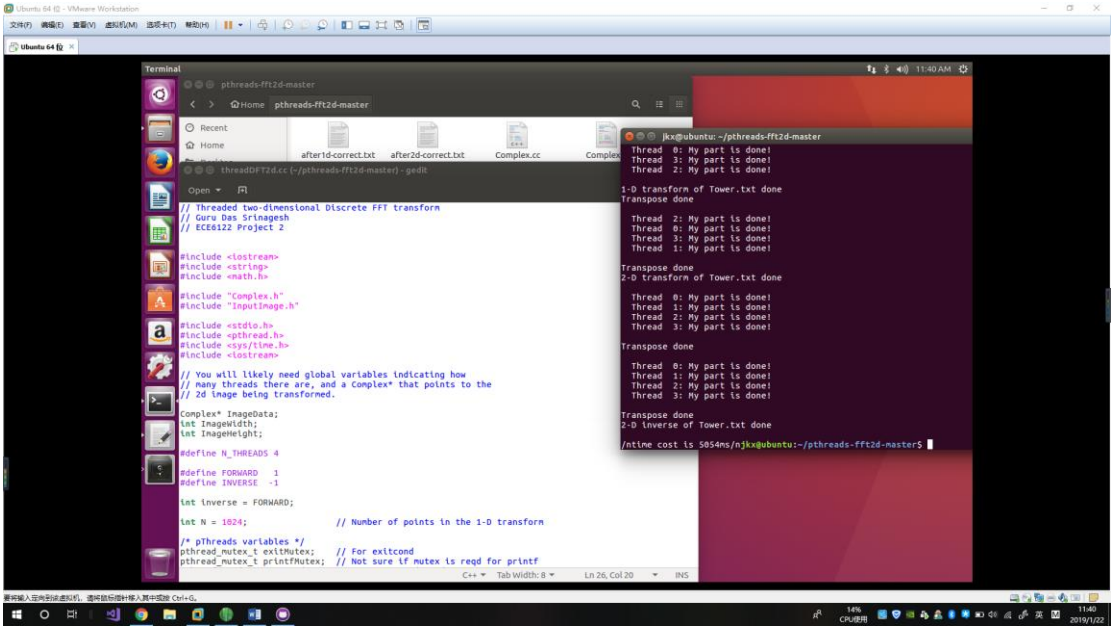
FFT 程序的输出



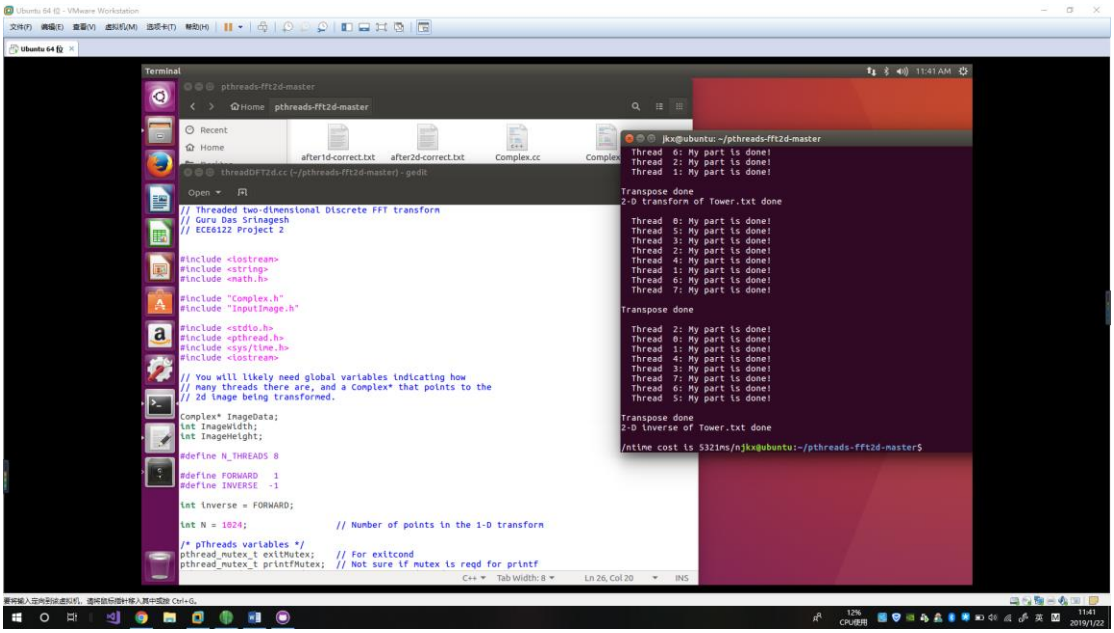
线程数为 1，时间为 5537ms



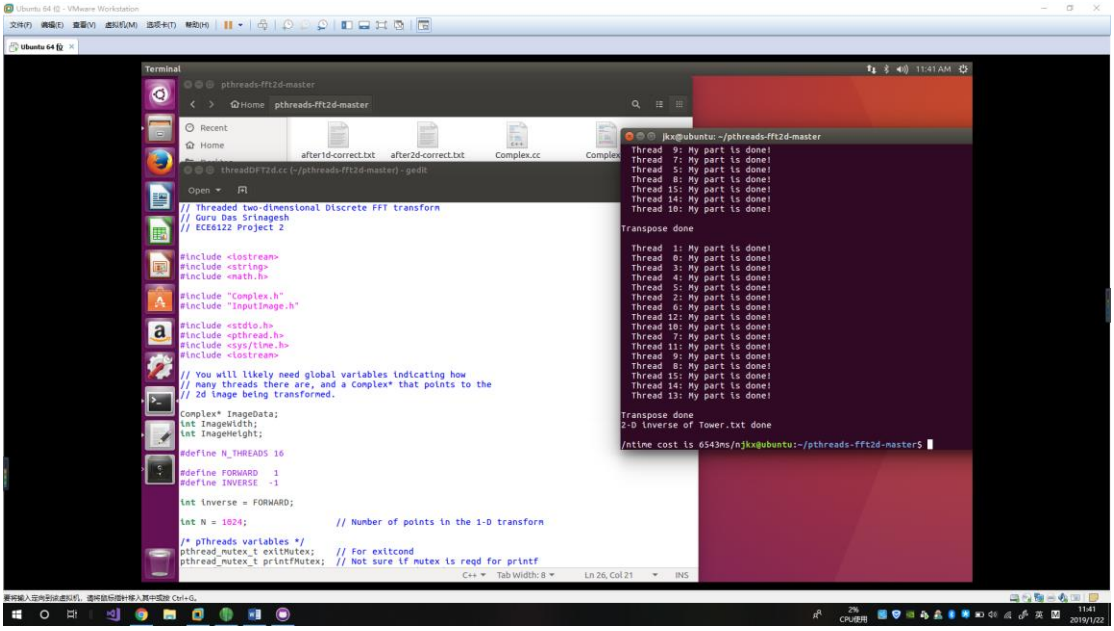
线程数为 2，时间为 4871ms



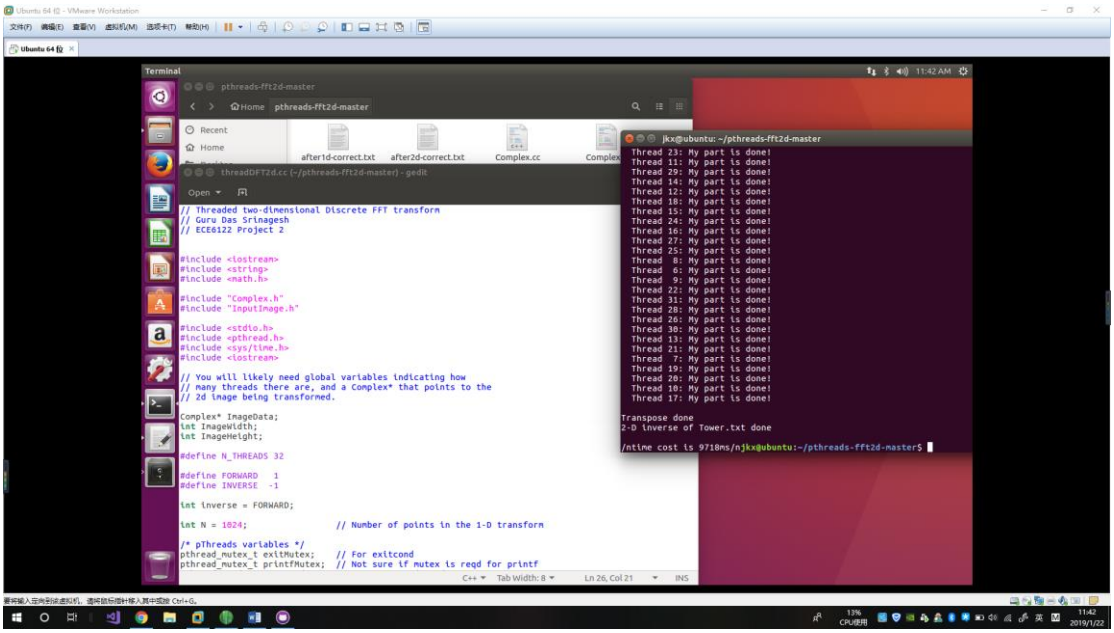
线程数为 4，时间为 5054ms



线程数为 8，时间为 5321ms



线程数为 16，时间为 6543ms



线程数为 32，时间为 9718ms

分析和结论

从测试记录来看，FFT 程序的执行时间随线程数目增大而增大，但这是相对于总时间而言。如果对单个线程，因为运算量的减少，单个线程的运算时间也会减少，所以在选定一个特定的线程数，比如 2。Cpu 能分配给程序两个核心参与运算，同时只创建了两个线程，此时是最适合该程序运行的情况。而对于 32 线程，因为 cpu 没有足够的核心数去满足，同时创建线程开销极大，所以花费时间远大于 2 核心数。

vvv