实验报告

实验名称（多线程 FFT 程序性能分析和测试）

物联 1601 201608010628 曾彤芳

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？
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？
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？

实验内容

多线程 FFT 代码

该代码采用了 pthread 库来实现多线程，其中 pthread

Pthread 是一套通用的线程库，它广泛的被各种 Unix 所支持，是由 POSIX 提出的. 因此，它具有很好的可移植性.

```
#include <pthread.h>

//新建线程

int pthread_create(pthread_t *restrict tidp, constpthread_attr_t *restrict attr,void*(*start_rtn)(void*),void *restrict arg);

//线程终止

void pthread_exit(void *rval_ptr);//线程自身主动退出

int pthread_join(pthread_t tid, void **rval_ptr);//其他线程阻塞自身，等待 tid 退出

//线程清理

voidpthread_cleanup_push(void(*rtn)(void*), void *arg);

voidpthread_cleanup_pop(intexecute);

pthread_create()：创建一个线程
```

pthread_exit()：终止当前线程

pthread_cancel()：中断另外一个线程的运行

pthread_join()：阻塞当前的线程，直到另外一个线程运行结束

pthread_attr_init()：初始化线程的属性

pthread_attr_setdetachstate()：设置脱离状态的属性（决定这个线程在终止时是否可以被结合）

pthread_attr_getdetachstate()：获取脱离状态的属性

pthread_attr_destroy()：删除线程的属性

pthread_kill()：向线程发送一个信号

pthread_equal(): 对两个线程的线程标识号进行比较

pthread_detach(): 分离线程

pthread_self(): 查询线程自身线程标识号

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：N

测试

测试平台

在如下机器上进行了测试：

部件	配置	备注
CPU	core i5-6200U	
内存	DDR3 4GB	
操作系统	Ubuntu 16.04 LTS	中文版

测试记录

多线程 FFT 程序的测试参数如下：

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

多线程 FFT 程序运行过程的截图如下：

1：

```
Performance counter stats for './threadDFT2d':

      3983.696078      task-clock (msec)         #    1.559 CPUs utilized
               34      context-switches          #    0.009 K/sec
                0      cpu-migrations            #    0.000 K/sec
            4,248      page-faults               #    0.001 M/sec
   14,389,730,597      cycles                    #    3.612 GHz
   35,692,311,951      instructions              #    2.48  insn per cycle
    7,080,823,378      branches                  # 1777.451 M/sec
        7,754,472      branch-misses             #    0.11% of all branches

      2.555842229 seconds time elapsed
```

2：

```
Performance counter stats for './threadDFT2d':

      4999.619802      task-clock (msec)         #    2.072 CPUs utilized
               113      context-switches          #    0.023 K/sec
                0      cpu-migrations            #    0.000 K/sec
            4,254      page-faults               #    0.851 K/sec
   14,663,381,072      cycles                    #    2.933 GHz
   36,718,322,129      instructions              #    2.50  insn per cycle
    7,334,055,443      branches                  # 1466.923 M/sec
        7,782,660      branch-misses             #    0.11% of all branches

      2.412815387 seconds time elapsed
```

4：

```
Performance counter stats for './threadDFT2d':

     15727.357427      task-clock (msec)         #    3.905 CPUs utilized
              133      context-switches          #    0.008 K/sec
                4      cpu-migrations            #    0.000 K/sec
            4,275      page-faults               #    0.272 K/sec
   19,886,143,371      cycles                    #    1.264 GHz
   46,718,086,531      instructions              #    2.35  insn per cycle
    9,839,616,865      branches                  #  625.637 M/sec
        7,879,278      branch-misses             #    0.08% of all branches

      4.027276076 seconds time elapsed
```

8：

```
Performance counter stats for './threadDFT2d':

     67095.958580      task-clock (msec)         #    6.972 CPUs utilized
            2,904      context-switches          #    0.043 K/sec
              166      cpu-migrations            #    0.002 K/sec
            4,314      page-faults               #    0.064 K/sec
   45,627,728,796      cycles                    #    0.680 GHz
   83,644,663,926      instructions              #    1.83  insn per cycle
   19,063,443,211      branches                  #  284.122 M/sec
        8,719,631      branch-misses             #    0.05% of all branches

      9.623471239 seconds time elapsed
```

16：

```
Performance counter stats for './threadDFT2d':

     36091.433294      task-clock (msec)         #    6.482 CPUs utilized
            3,679      context-switches          #    0.102 K/sec
              141      cpu-migrations            #    0.004 K/sec
            4,373      page-faults               #    0.121 K/sec
   63,479,756,322      cycles                    #    1.759 GHz
  119,514,509,468      instructions              #    1.88  insn per cycle
   28,035,551,954      branches                  #  776.792 M/sec
        8,538,029      branch-misses             #    0.03% of all branches

      5.567592222 seconds time elapsed
```

思考题解答

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？

多线程相对于单线程理论上能提升性能为多线程的数目；

多线程的开销：

多线程中两个必要的开销：线程的创建、上下文切换

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？

实际运行中多线程相对于单线程是提升了性能，但并不如预期值，这是因为多线程的必要开销，并且线程越多，阻塞的时间也就越长。

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减少，其相对于单线程情况的加速比分别为：

线程数 1：

加速比：1

线程数 2：

加速比：2*2.55/2.41=2.15

线程数 4：

加速比：4*2.55/4.03=2.53

线程数 8：

加速比：8*2.55/9.62=2.12

线程数 16：

加速比：16*2.55/5.56=7.34