Lab1 performance test report

1. 实验概要

多线程编程是高性能编程的技术之一，实验1将针对数独求解问题比较多线程与单线程的性能差异、同一功能不同代码实现的性能差异以及多线程在不同硬件环境下的性能差异。

1.1 程序输入

程序将在控制台接收用户输入，该输入应为某一目录下的一个数独谜题文件，该文件包含多个数独谜题，每个数独谜题按固定格式存储在该文件中。

1.2 程序输出

实验中把数独的解按与输入相对应的顺序直接输出到屏幕上。

1.3 Sudoku算法

我们采用了2种不同的数独算法：arity cache和basic，多线程测试中两种代码对应了两个不同的版本。

1.4 性能指标

实验以求解完单个输入文件里的所有数独题并把数独的解按顺序写入文件所需要的时间开销作为性能指标。一般而言，可以用加速比直观地表示并行程序与串行程序之间的性能差异（加速比：串行执行时间与并行执行时间的比率，是串行与并行执行时间之间一个具体的比较指标）。

为了精确地测量性能，时间开销均在数独求解进程/线程绑定CPU的某个核的条件下测得，这样保证了该进程/线程不被调度到其他核中运行，但不保证该进程/线程独占某个核。更精确的测量方法可以先把CPU的某个核隔离，而后再绑定在某个进程/线程上，这样该CPU核心不会运行其他的用户程序。当CPU资源充足时（CPU核心数足够多，当前正在运行的进程/线程足够少），是否把核隔离并没有多大影响，因为操作系统的调度策略不会频繁的对线程/进程进行无谓的调度。

1.5 实验环境

Linux内核版本为4.15.0-88-generic；8GB内存；CPU型号为Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz，共1个物理CPU；每个CPU有4个物理核心，共4个物理核心；不使用超线程技术

1.6 代码实现版本

实验中共使用两份不同的代码：**Code1**和**Code2**。

Code1: 原生的数独求解代码，即本实验中所提供的代码，只能以单线程模式运行。

Code2: 为适应多线程而在Code1上进行了一系列的修改和增添而成。在Code2中，可通过参数的调节而控制线程数量。与Code1相比，Code2的代码量多了一些内容。code中使用的是同构线程以及sudoku_basic方法解决数独问题。code执行流程如下：先完成文件读写操作，然后根据输入的线程数量建立多线程。多线程运行Deal函数，该函数会循环调用readPuzzle，读取与求解数独，直到所有数独求解完毕。readPuzzle函数会尝试从临界区中获取数独题目，如果获取成功，则对其求解并调用solved函数对解进行验证，并将解按照题目的行号记录到输出数组（该数组暂时只开到容纳1000个数独）中。线程执行结束后对解进行输出。

如无特别说明，默认使用Code2。

2. 性能测试

程序的性能会受到诸多因素的影响，其中包括软件层面的因素和硬件层面的因素。本节将分析比较多线程程序与单线程程序的性能差异、同一功能不同代码实现的性能差异，以及同一个程序在不同硬件环境下的性能差异。

2.1 多线程与单线程性能比较

test1,即有1组输入时

1线程

```
Process time:2.42074
```

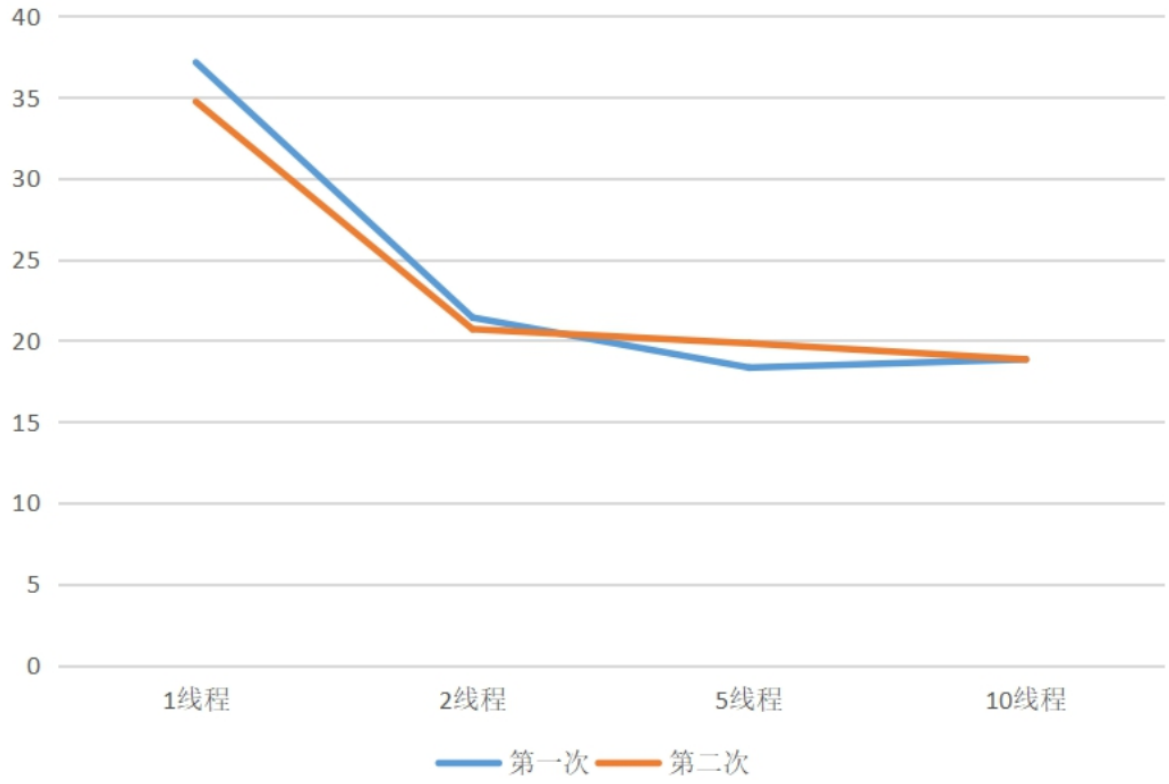
时间:

2.42074s

test40,即有40组输入时

成表

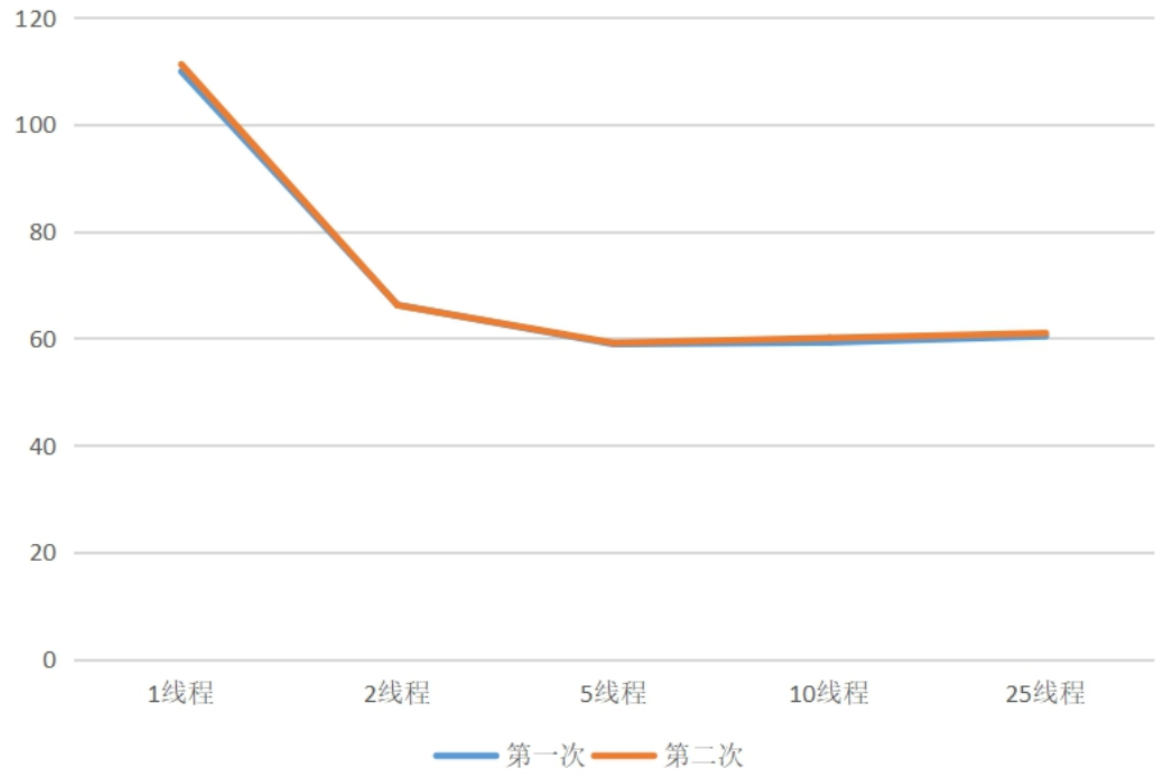
线程	第一次	第二次
1	37.1514	34.729
2	21.4199	20.7044
5	18.3367	19.833
10	18.8456	18.8455



test100,即有100组输入时

成表

线程	第一次	第二次
1	109.948	111.29
2	66.2479	66.2254
5	58.9295	59.1397
10	59.2442	60.0764
25	60.4115	60.9794



test1000,即有1000组输入时

成表

线程	2	32
时间	682.121	695.604

由test40和100两组可以看出，在1-5个线程时，随线程增多，效率有所提升，随后即使线程增多，效率也趋于稳定。一开始我们推测，在线程到达 总输入开方 个之后效率达到最高，而结果表明，在5个线程左右时效率基本已达到最高。同时线程增加到一定数量后时间反而略有增加，说明线程过多也会影响效率

2.2 不同代码实现性能比较

这里测试了另外一组代码Code3，code中使用的是同构线程以及sudoku_basic方法解决数独问题。code执行流程如下：先完成文件读写操作，然后根据输入的线程数量建立多线程。多线程运行Deal函数，该函数会循环调用readPuzzle，读取与求解数独，直到所有数独求解完毕。readPuzzle函数会尝试从临界区中获取数独题目，如果获取成功，则对其求解并调用solved函数对解进行验证，并将解按照题目的行号记录到输出数组（该数组暂时只开到容纳1000个数独）中。线程执行结束后对解进行输出。

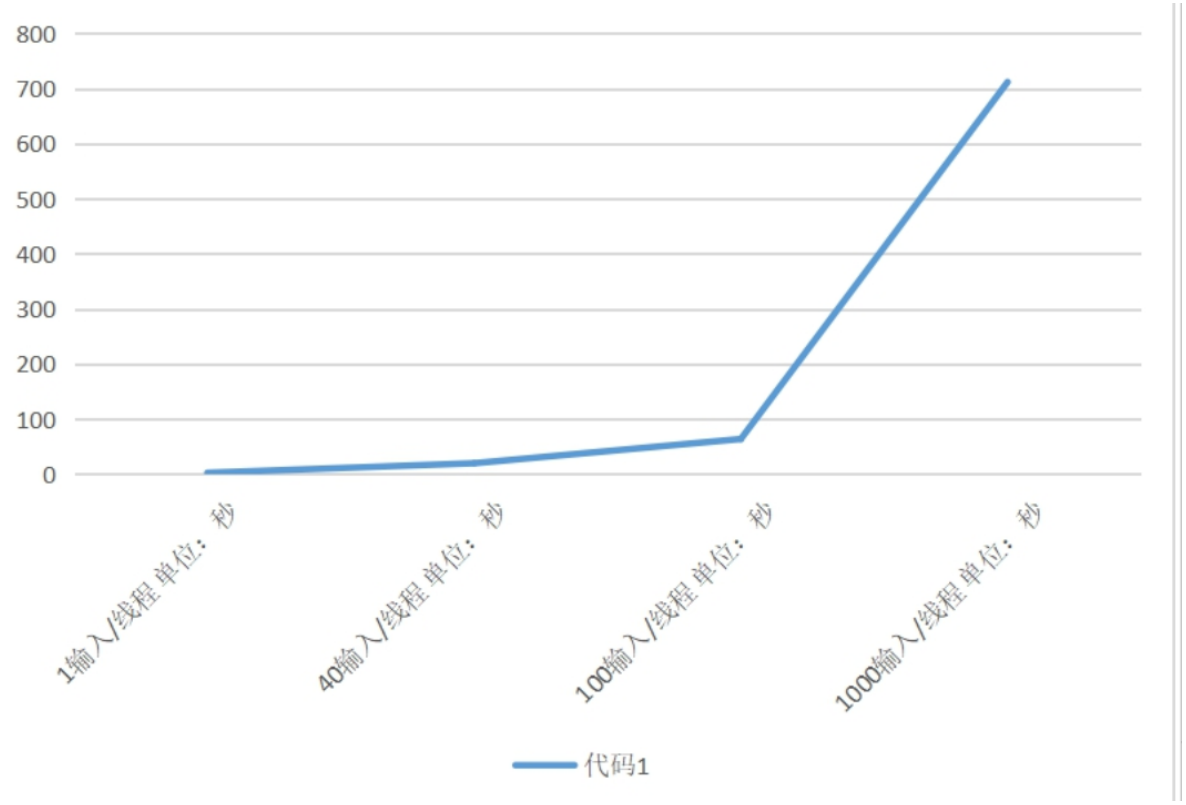
增加了一个用于管理多线程的类，使得每进行一次解，增加一个线程

输入/线程	1输入/线程	40输入/线程	100输入/线程	1000输入/线程
时间	0.01035	0.001864	0.74146	12.1

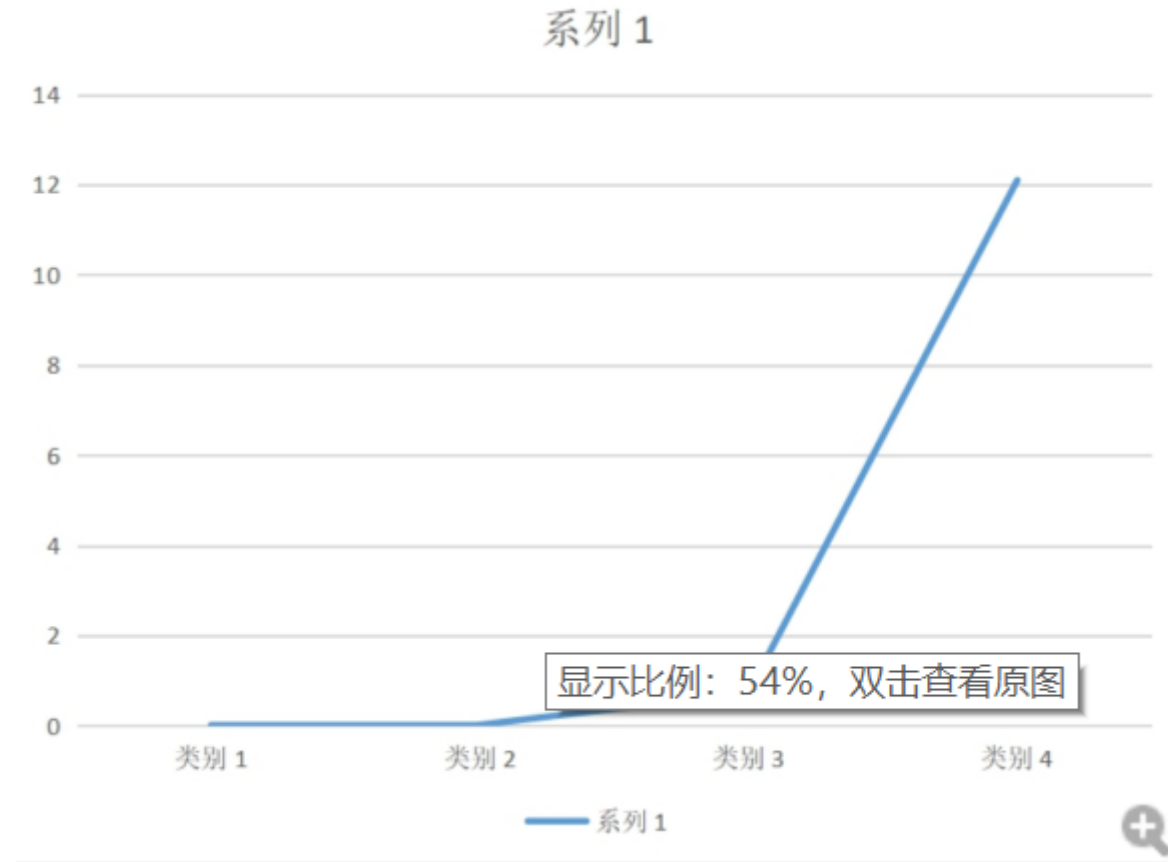
对比前一组代码建表

	1输入/线程	40输入/线程	100输入/线程	1000输入/线程
代码1	2.42074	19.8059	63.7892	711.784
代码2	0.01035	0.01864	0.74146	12.1

Code2:



Code3:



可以看出，由于每个解都设置单独的线程进行运算，效率大幅提升