湖南大學

HUNAN UNIVERSITY

云计算技术
性能报告

题　目：	“Super-fast” Sudoku Solving
	Lab 1
组 长 姓 名：	龙思宇-201708010823
学 生 姓 名：	刘梦迪-201708010826
学 生 姓 名：	高文静-201708010827
学 生 姓 名：	吕　喆-201708010830
完 成 日 期：	2020/3/31

目 录

一、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。

接受一个输入文件名，并且输入文件大小小于 100MB。

```
longsiyu@ubuntu:~/Lab1( )$ ./sudoku_solve
./test10
693784512487512936125963874932651487568247391741398625319475268856129743274836159
793684512486512937125973846932751684578246391641398725319465278857129463264837159
673894512912735486845612973798261354526473891134589267469128735287356149351947628
679835412123694758548217936416723895892561374735489621287956143961342587354178269
346795812258431697971862543129576438835214769764389251517948326493627185682153974
598463712742851639316729845175632498869145273423978156934287561681594327257316984
364978512152436978879125634738651429691247385245389167923764851486512793517893246
649835712358217964172649385916784523834521679725963148287356491591472836463198257
367485912425391867189726354873254196651973428294168573718649235946532781532817649
378694512564218397291753684643125978712869453859437261435971826186542739927386145
```

1.2 程序输出

实验中把数独的解按与输入相对应的顺序写入到 stdout 文件中。

1.3 Sudoku 算法

实验共提供了 4 种不同的 Sudoku 求解算法：BASIC,DANCE,MINA 和 MINAC。其中，DANCE 算法速度最快，BASIC 算法速度最慢。实验中选用的是最慢的 BASIC 算法。

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）。

在用户输入文件路径进行运算后，在输出结果最后会获取运行时间，单位 ms。

1.5 实验环境

Linu 内核版本为 3.13.0-32-generic；2GB 内存；CPU 型号为 Intel(R) Core(TM) i5-7200U CPU @ 2.5GHz 2.7GHz，共有 1 个核心 CPU；不使用超线程技术。

1.6 代码实现版本

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（指定线程数量为 N）。

全局指定 5 个线程（N=5），主线程负责文件的顺序读入，每读入一个数独记成一个任务（将一个数独存放在全局二维数组里），空闲的线程领取完成任务（recvAJob 函数），线程任务完成后若有待解决的数独则再去领取完成，在全部数独求解完后将解写入全局的 bool 变量 shutdown 记录任务全做完，退出 5 个子线程。

二、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 3 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

单线程程序只能利用 1 个 CPU 核心，而多线程程序能使 CPU 的多个核心并行运作，因此，多线程能够充分发挥多核 CPU 的优势。在一定范围内，加速比会随着线程数的增加而增长，即时间开销越少、效率越高。当线程数超过 CPU 核心数时，性能会有所下降。

实验将提供？个大小为？.？MB、内含？K 个字节的数独题目，然后分别使用单线程版本的？和多线程版本的？分别对该文件进行求解测试，记录求解所需的时间开销并计算加速比

如图，具体化不同线程数对性能的影响，两条折线：？和？分别表示 sodoku_solve 随着线程数量的变化所需要的时间开销。从图中可知，……

分析加速比达到？的原因

2.2 程序性能最优时的线程数量

线程数量如 总线程与 CPU 核心数的大小关系，会导致同一个程序在不同的线程数量时有不同的表现。对总线程数小于 CPU 核心数，线程增加，开销减小。实验将测试不同的线程数量求解数独对程序性能的影响，其中 sudoku_solve 线程数从 1 开始逐步增加，测量时间开销。

~~2.3 不同代码实现性能比较~~

对于实现同一功能的程序，可以有多种不同的代码实现，不同的代码实现在时间开销上不一定会相同。实验中使用的 Code2 比 Code1 多了一些额外的代码段，因此 Code2 的时间开销要略大与 Code1，并且随着问题规模的增大，差距也会愈加明显。 考虑到代码可读性、可扩展性或其他因素，有时会在代码实现上增加一些额外的代码段。当这些额外的代码段被调用的次数足够多时，其所造成的时间开销会逐渐显现出来。

实验将使用 2 份不同的代码进行性能比较：Code1 和 Code2。实验提供 8 个不同大小的文件，每个文件分别有数独题： 1 K,2 K,4 K,8 K,16 K,32 K,64 K,128 K,256 K,512 K 和 1024 K。分别用 Code1 和 Code2 对这些文件进行求解（此处 Code1 和 Code2 都是使用单个数独求解线程进行求解），并测量时间开销。

图 2-2 显示数独题量从 1 K 增长到 1024 K 时，Code1 与 Code2 之间的时间开销差距逐渐拉大。在 1024 K 时，Code2 比 Code1 多花费了约 10 S 的时间。因为在代码实现时，Code2 比 Code1 多增加了一些额外的代码，必然会引入一些额外的开销。在这些额外增加的代码中，有些代码段在程序运行期间调用次数不多，有些代码段则是进行 1 次数独求解就要调用 1 次，当数独求解量达到 1024 K 之多时，其所造成的开销就会凸显出来。因而会造成随着数独题量的增多，Code1 和 Code2 在时间花销上差距逐渐明显的现象。

~~2.4 不同硬件环境性能比较~~

硬件环境如 CPU 主频（其它如物理 CPU 个数、物理核心数等）的不同，会导致同一个程序在不同的硬件环境中有不同的表现。对单个 CPU 物理核心，其主频越高，运行速度越快。 实验将使用 Code2 分别在 ENV1 和 ENV2 中对大小为 84.0MB、具有

1024 K 个数独题的文件进行求解，其中 sudoku_solve 线程数从 1 开始逐步增加，测量时间开销。

图 2-3 为 Code2 在不同的硬件环境 ENV1 和 ENV2 中分别调整不同的 sudoku_solve 线程数的测试结果。从图 2-3 可以看出，当 sudoku_solve 线程数在 1～4 时，ENV2 的时间花销略小于 ENV1，因为 ENV2 的 CPU 主频为 2.6 GHZ，略大于 ENV1 的主频 2.0 GHZ，即对单个物理核心而言，ENV2 的运行速度比 ENV1 快。当 sudoku_solve 线程数量在 5～8 时，ENV2 的时间开销要略大于 ENV1，这是因为 ENV2 总共只有 4 个物理核心，通过超线程技术模拟出 8 个逻辑核心，其单个逻辑核心的性能还是略逊于单个物理核心，但总体上看增加超线程还是会使性能略有提升。在 ENV2 中，继续增加 sudoku_solve 线程数量使得线程数超过逻辑核心数（线程数大于 8），其性能会因为线程的调度而出现下降的趋势；同理，在 ENV1 中，继续增加 sudoku_solve 线程数量使得线程数超过物理核心数（线程数大于 16），其性能也会因线程调度而有所下降。