浙江大学实验报告

专业：计算机科学与技术
姓名：吴同
学号：3170104848
日期：2020 年 5 月 18 日

课程名称：并行计算与多核编程　指导老师：楼学庆　电子邮件：wutongcs@zju.edu.cn
实验名称：系统性能分析　实验类型：综合型　联系电话：18888922355

一、 实验目的

- 使用性能分析工具对程序进行性能分析
- 用 TBB 库对程序进行并行化改造
- 对改造前后的程序性能进行比较分析

二、 实验原理

1. VTune

Intel VTune 是一个用于分析和优化程序性能的可视化性能分析器。作为 Intel 为开发者提供的专门针对寻找软硬件性能瓶颈的一款分析工具，它能确定程序的 hotspot，找到限制性能提升的瓶颈因素，从而有助于开发者提出优化方案。VTune 性能分析器能通过以下的手段发现和定位程序中的性能问题：从当前系统中收集性能数据；从系统到源代码不同的层次上，以不同的交互形式来组织和展示数据；发现可能存在的性能问题，并提出改进建议。

VTune 主要包括三个小工具：Performance Analyzer 用于性能分析，找到软件的 hotspot。Intel Threading Checker 用于查找线程错误，如资源竞争、死锁等问题。Intel Threading Profiler 用于线程性能检测，使得负载平衡。

2. TBB

TBB 是英特尔发布的一个 C++ 模版库，其并行化级别在更高的抽象层，是基于任务的编程，而不是基于线程。任务是比线程更高级别的并行抽象。在使用 TBB 进行开发时，开发者无须关心具体的线程细节，例如负载均衡、优化调度等。TBB 的底层组件已经实现了这些工作，可以实现自动调度。

TBB 共包含了 6 个模块：Algorithms、Containers、Memory Allocation、Synchronization、Timing、Task Scheduling。在使用 TBB 编程时，需要定义自己的任务，这些任务是从 **tbb::task** 中派生的，并使用 **tbb/task.h** 进行声明。当 Intel TBB 任务调度程序选择运行一些任务时，会调用该任务的 **execute** 方法。**execute** 方法会返回一个指向 task 指针，告诉调度程序将要运行的下一个任务。如果返回 **NULL**，那么调度程序可以自由选择下一个任务。

三、 实验过程

1. 编写串行程序

本实验以矩阵乘法的程序为例进行性能分析。首先编写普通的串行化矩阵乘法：

```
template <typename T>
Matrix<T> matrixMultiplySerial(Matrix<T>& mat1, Matrix<T>& mat2)
{
   T** result = new T*[mat1.width];
   for (int i = 0; i < mat1.width; ++i)
   {
      result[i] = new T[mat2.height];
   }
   for (int i = 0; i < mat1.width; ++i)
   {
      for (int j = 0; j < mat2.height(); ++j)
      {
         for (int k = 0; k < mat1.height(); ++k)
         {
            result[i][j] += mat1.at(i,k) * mat2.at(k,j);
         }
      }
   }
   return *new Matrix<T>(result, mat1.width, mat2.height);
}
```

2. 进行性能分析

按照以下步骤使用 VTune 进行性能分析：

- 启动 Intel VTune 性能分析器
- 选择 New Project，建立新工程，
- 选择 sampling Wizard，建立采样向导
- 选择文件类型，在 Application To Launch 对话框中选择可执行文件
- 查看 Hotspot 热点图，分析具体函数的 Clocktick 和 CPI
- 查看调用图，查看程序的函数及其依赖关系

程序的并行化改造方向有三：一是数据分解，对程序的数据输入、数据处理和数据输出进行并行化，考虑数据分解方式的并行；二是循环并行化，对数据处理的无耦合操作采用适当的粒度划分以进行并行化。三是任务并行，对于完全独立的功能可采用任务并行的改造，若任务间有依赖关系，则进行流水线的改造。

3. 并行化改造

本实验主要对矩阵乘法的循环进行并行化改造。改造后的代码如下：

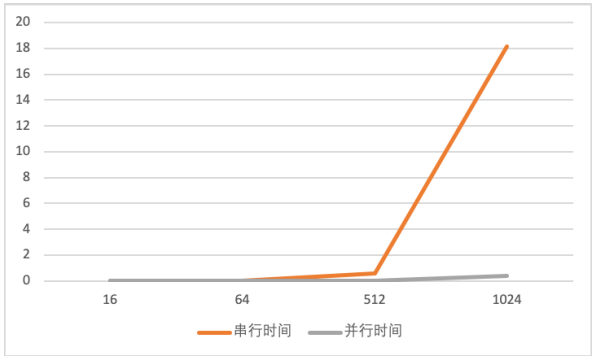
```
template <typename T>
Matrix<T> matrixMultiplyParallel(Matrix<T>& mat1, Matrix<T>& mat2) {
   T** result = new T*[mat1.width];
   for (int i = 0; i < mat1.width; ++i)
   {
      result[i] = new T[mat2.height];
   }
   tbb::parallel_for(size_t(0), size_t((mat1.width)),
                [&] (size_t i)
                {
                   tbb::parallel_for(size_t(0), size_t((mat2.height)),
                    [&] (size_t j)
                    {
                       for (int k = 0; k < mat1.height; ++k)
                       {
                          result[i][j] += mat1.at(i,k) * mat2.at(k,j);
                       }
                    });
                });
   return *new Matrix<T>(result, mat1.width, mat2.height);
}
```

四、 实验结果

程序的测试结果如下：

数据规模	串行时间	并行时间	加速比
16	0.00166	0.00004	41.5
64	0.001731	0.000134	12.9
512	0.595137	0.043276	13.8
1024	18.1694	0.427469	42.5

运行时间的折线图如下：



五、 分析讨论

在进行本次实验的过程中，我发现针对并行程序性能分析和改造的资料相对较少。结合在平常使用各种软件时“一核有难，七核围观”的情况，可以看出虽然处理器早已进入多核化时代，但目前程序的多核开发现状仍然很不乐观。从本次实验的经历就可感受出，要想真正发挥出多核心的优势，仅仅掌握多线程编程的知识是不够的。进行多核编程，要求对计算机体系结构有着非常深入的认识，这样才能准确找出制约程序性能提升的瓶颈，并提出有效的解决方案。

英特尔以处理器制造厂商的身份开发的许多工具对于程序的性能分析和优化非常有用。然而，这些工具目前并为得到广泛使用。一方面是因为软件有特定的硬件平台要求，只能适用于英特尔的处理器，如果是 AMD 的处理器则会起到反向优化的效果。这对于软件开发所追求的跨平台性极其不利。另一方面，系统性能分析所要求的知识水平较高，并且投入产出比不如业务开发高，很难跟得上互联网市场的快节奏要求。

我在这次实验中，对程序性能分析和多核改造有了初步的了解，但尚不能有深刻的体会和理解。在未来还要多阅读资料，多动手实践，以求深入理解计算机系统。