并行计算在多核平台上的实现与应用研究①

秦书茂, 叶海建

(中国农业大学 信息与电气工程学院, 北京 100083)

摘 要: 多核 CPU 在当前已成为 PC 机的常规配置, 为了充分发挥 PC 机的性能, 以提高应用软件的运行速度, 本文针对如何在多核 CPU 上实现并行计算进行了研究, 将其应用到薄层水流流速参数的虚拟正态边界模型计算中. 经实例测试验证, 采用双核、四核并行计算的模型求解速度分别是单核情况下的 1.4 倍、2.4 倍, 核心数越多, 倍数越大.

关键词: 多核; 并行; 薄层水流流速

Application and Realization Research of Parallel Computing on Multi-Core Platform

QIN Shu-Mao, YE Hai-Jian

(College of Information and Electrical Engineering, China Agriculture University, Beijing 100083, China)

Abstract: In the current, Multi-core CPU has become the general configuration of the PC. In order to give full play to the performance of the PC and improve the running speed of application software, how to parallel computing on the multi-core CPU is studied in this paper. The research is applied into the calculation of shallow water flow velocity measurement model with virtual boundary condition. The test result showed that the running speed on dual-core CPU and on quad-core CPU is 1.4 times and 2.3 times faster than that on the single-core CPU, which indicates the more the number of cores, the greater the multiplies.

Key words: multi-core; parallel; shallow flow velocity

目前, 普通 PC 配置双核、四核, 甚至八个核的多核 CPU 已经十分常见. 多核 CPU 是在单个处理器芯片内实现了两个或者更多的“执行核”. 实际上, 这些执行核都是相互独立的处理器, 只是位于同一块芯片而已[1]. 也就是说普通 PC 上实际已经配置了多块“处理器”, 拥有了多个计算单元, 能够进行单机的多核并行计算, 这为在 PC 机上实现并行计算提供硬件基础. 但是, 由于基于多核的并行技术还不为人们所熟知, 因此基于多核并行技术的应用软件还很少见[2].

1 多核并行计算技术

多核并行计算的硬件基础是多核处理器. 多核处理器是由两个或两个以上完整的计算引擎(内核)集成在一个芯片上的处理器. 多核处理器中的每个核心都是一个单独的处理器, 可有独立的高速缓冲存储器, 也可共享高速缓冲存储器, 但内存是每个核心共享的.

串行程序在多核处理器上同一时刻只能在一个核心上运行, 没有充分发挥多核处理器拥有多个核心的潜能. 程序必须采用多线程的方式, 才能够同一时刻在多个核心运行, 实现并行计算. 多核并行计算可以充分利用多核处理器资源, 能够加速应用程序的计算.

目前, 基于这种内存共享的多核并行计算有一个普遍接受的编程模式 OpenMP. OpenMP 编程模式是计算机硬件和软件厂商于 1997 年 10 月联合定义发表的共享内存编程应用程序接口的工业标准协议[3]. 它对多核心的处理器进行了抽象, 编程者不需要关心线程的创建、调度、同步及销毁等, 方便了并行程序的实现, 增强了并行程序可移植性和扩展性.

OpenMP 并不是编程语言, 而是现有语言的扩展, 是应用编程的接口, 支持粗粒度(函数级别)及细粒度

① 收稿时间:2013-05-03;收到修改稿时间:2013-05-27

(循环级别)的并行, 可以在 C/C++或 Fortran 中以编译指令的形式出现[4]. 编程人员通过简单的编译指令就可增量开发并行程序.

用 OpenMP 并行程序采用 fork-join 的方式运行. 程序启动时以单个进程启动, 运行到并行区时 fork 出一定数目的线程. 线程运行并行区代码, 被调度到不同核心上执行. 所有线程执行完后 join 到主线程, 继续往下执行. 程序中可以有多个并行区.

2 薄层水流流速参数计算

薄层水流流速参数是定量分析研究土壤侵蚀的重要参数, 其水流测速虚拟正态边界数学模型复杂[5], 计算量大, 串行求解耗时. 数学模型如式(1)所示:

$$C(x,t)=\int_0^t C_0\frac{x}{2(t-\tau)\sqrt{\pi D_H(t-\tau)}}\times\exp\left(-\frac{(x-u(t-\tau))^2}{4D_H(t-\tau)}\right)g(\tau)d\tau \quad (1)$$

$$g(t)=A\exp\left(\frac{(t-D)^2}{2B^2}\right) \quad (2)$$

模型描述了薄层水流中注入的电解质溶液浓度 $C(x,t)$ 在相对注入点 x 处随时间 t 的浓度变化. A、B、D、C_0、D_H、u 是待求解的参数. 在距离电解质溶液注入点 x, 测量 t_1、t_2、$\cdots t_{n-1}$、t_n 时间盐溶液对应的浓度 c_1、c_2、$\cdots c_{n-1}$、c_n. 然后求解参数 A、B、D、C_0、D_H、u 使得正态虚拟模型函数拟合盐溶液在 x 处随时间变化的浓度值. 拟合函数参数求解采用最小二乘法. 最小二乘法拟合就是使计算的函数值与测得的浓度值误差平方和最小. 如式(3)所示:

$$\min(F(t))=\min\left(\sum_{i=1}^{n}(C(x,t_i)-c_i)^2\right) \quad (3)$$

采用模式搜索算法求解这个最小值. 模式搜索算法又叫 Hooke-Jeeves 方法, 是一种无约束最优化的直接方法[6]. 它由轴向移动和模式移动两个步骤组成. 其总体思路是: 由起始点, 进行轴向移动和模式移动. 轴向移动依次沿坐标轴方向移动, 用来确定新的起始点和有利下降方向; 模式移动则是沿相邻两探测点的连线进行移动, 试图使函数值下降更快.

算法中除轴向移动及模式移动有计算工作外, 其余都是简单的赋值及比较操作. 模式移动的计算工作是简单的加减乘. 轴向移动包括正向移动、负向移动、保持不动三个方向的试探. 这三个试探方向的每次试探都会进行拟合函数值的计算. 由于拟合函数复杂, 轴向移动承担的计算工作量远远大于其他步骤, 是整个计算性能的瓶颈.

3 多核并行计算应用

并行计算中对应用程序分解问题的主要方式有三种: 任务分解、数据分解及数据流分解[7]. 结合前文对应用实例的计算性能瓶颈分析, 该应用实例比较适合任务分解来进行并行计算. 任务分解是对应用程序根据其执行的功能进行分解的过程, 是一种能够简单实现并行执行的方法. 本文对模式搜索算法轴向移动的正向移动、负向移动、保持不动三个探测任务分解, 即由原来轴向移动中串行的三次探测计算改为并行执行. 并行执行后三次探测计算只需原来的一次探测计算的时间. 并行执行的轴向移动流程图如图 1 所示.

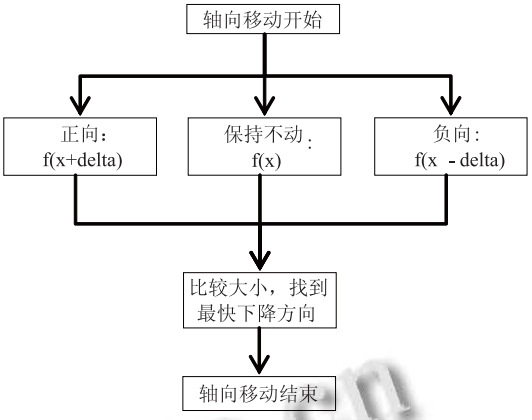

图 1　轴向移动并行算法流程图

本文对轴向移动三次探测计算的并行分解采用 sections 语句实现. 伪代码如下:

```
#pragma omp parallel sections
{//进入并行区域, fork 出线程
#pragma omp section
    {
        f_zero=f(x);//保持不动方向拟合函数值计算
    }
#pragma omp section
    {
        f_plus= f(x+delta);//正向探测方向拟合函数值计算
    }
#pragma omp section
```

```
        {
            f_minus= f(x-delta);//负向探测方向拟合函数值计算
        }
    }//并行区结束, 所有线程 join 到主线程
```

最后, 对各个试探方向求得的函数值比较得出最佳的下降方向, 结束轴向移动步骤. 采用 sections 及 section 语句比较容易实现一个参数求解的任务分解. 如果采用 for 语句, 则能够实现多个参数求解任务的分解, 能获得更好的性能.

串行的轴向移动部分要两次到三次函数计算. 并行的轴向移动部分创建三个线程同时进行函数计算. 在四核 CPU 上, 仅一次单核 CPU 计算时间就能得出三次函数计算结果. 整个计算步骤中轴向移动占用的时间是压倒性的比例, 其他步骤的计算时间可忽略. 因此轴向移动的性能可作为整个计算性能的指标. 并行执行在最坏(即串行轴向移动仅两次函数计算)、最好(即串行轴向移动进行三次函数计算)情况下分别是串行速度的 2 倍、3 倍.

4 实验验证与分析

本文对给定的 A、B、D、$C0$、DH、u 初始值均为 1, 分别对两组实验中 x 为 0.3 米、0.6 米、0.9 米处的测量数据进行了串行执行及并行实验. 记录了串行运行时间及并行分别在四核、二核上运行时间, 并采用加速比(串行执行时间/并行执行时间)作为性能衡量标准.

测试所用机器为同一台 PC 机. CPU 为 Intel Core2 Quad CPU, 四个核心分别是 CPU0、CPU1、CPU2、CPU3; 测试系统环境均为 Windows 7; 编译器均选择 VS 2010 提供的编译器; OpenMP 版本 2.0.

为增强可比性、测试准确性, 首先设置所有可控制进程在 CPU0 上运行并终止不必要进程; 其次禁止对串行、并行多核进程及线程优先级作动态调整; 最后串行执行绑定到 CPU3, 双核执行绑定到 CPU2、CPU3, 四核执行绑定到 CPU0、CPU1、CPU2、CPU3. 然后分别测试, 测试结果及加速比计算如表 1 所示.

在四核上, 并行执行在最优情况下是串行执行性能的 3 倍, 在最差情况下应是串行执行性能的 2 倍.

在两核上, 并行执行的三个线程要分两次运行. 因此, 并行执行最优情况下是串行执行性能的 1.5 倍, 最差情况下甚至没有串行性能好(这是由于并行创建线程开销所致).

表 1 串行、并行实验结果及加速比计算

组别	x	串行(ms)	并行(ms)		加速比	
			两核	四核	两核	四核
第一组	0.3	88421	62650	36301	1.411	2.436
	0.6	89170	62993	36488	1.415	2.444
	0.9	84085	59515	34476	1.412	2.439
第二组	0.3	94568	65426	37675	1.445	2.510
	0.6	561120	387881	223347	1.447	2.512
	0.9	166875	114848	66192	1.453	2.521

由表 1 看出, 两核心、四核心上并行后的性能是串行性能的 1.4 倍、2.4 倍. 核心越多, 倍数越大.

5 结论

多核并行计算是计算机的重要发展方向之一, 是复杂数学模型计算的利器. 本文针对在多核 CPU 上并行计算实现进行了研究, 并将其应用到实际数学模型计算的实例中. 经验证, 多核并行计算能充分挖掘多核处理器的潜能, 核心数越多, 加速倍数越大.

参考文献

1 Akhter S, Roberts J. Multi-Core Programming. Hillsboro, Intel Press. 2006: 7–12.

2 Rattner J. Multi-core to the masses. Parallel Architectures and Compilation Techniques. Washington DC. IEEE Computer Society. 2005.3.

3 Dagum L, Menon R. OpenMP: an industry standard API for shared-memory programming. Computational Science & Engineering, IEEE, 1998, 5(1): 46–55.

4 Chandra R, Menon R, Dagum L. Parallel programming in OpenMP. San Francisco, Morgan Kaufmann. 2000: 15–40.

5 Shi XN, Zhang F, Lei TW. Measuring shallow water flow velocity with virtual boundary condition signal in the electrolyte tracer method. Journal of Hydrology, 2012, 452–453: 172–179.

6 Hooke R, Jeeves TA. “Direct Search” solution of numerical and statistical problems. Journal of the Association for Computing Machinery, 1961, 8(2): 212–229.

7 Mattson TG, Sanders BA, Massingill BL. Patterns for parallel programming. Boston. Addison-Wesley Professional. 2004: 31–60.