面向过程的任务并行化设计方法①

蒋少丙, 刘书刚

(华北电力大学 控制与计算机工程学院, 保定 071000)

摘 要: 为有效实现迭代问题的并行化, 提出了面向过程的任务并行化设计方法. 该方法的主要思想是对任务求解的单次迭代过程进行并行化设计. 将面向过程的思想运用到 K-means 聚类算法的并行设计过程中, 并通过 OpenMP 编程模型来验证该方法的有效性. 通过实验结果分析得知, 面向过程的任务并行化执行相较于传统的串行执行在效率上有很大的优势, 可以运用到迭代问题的并行化设计过程中.

关键词: 多核; 面向过程; 并行化设计; K-means 聚类算法; OpenMP

Task Parallelization Process-Oriented Design Methods

JIANG Shao-Bing, LIU Shu-Gang

(Control and Computer Engineering, North China Electric Power University, Baoding 071000, China)

Abstract: To realize the parallelization of iterative problem,the author proposes a task parallelization process-oriented design method. The main idea of this method is that a single iteration process of solving problem should be designed in parallel. Then the authors apply the process-oriented thinking to the K-means clustering algorithm, use OpenMP programming model to verify the validity of the method. The analysis of the experimental results show that process-oriented task parallelism has a great advantage in efficiency compared to serial execution, and can be applied to the process of the parallel design iteration problem.

Key words: multicore; process-oriented; parallel design; K-means clustering algorithm; OpenMP

在多核处理器系统普及的今天, 传统的串行软件设计已经不能充分利用多核处理器的良好性能, 也不能发挥出多核处理器系统在执行任务方面的巨大优势. 因此, 如何在软件设计中充分利用多核处理器的硬件优势, 成为了计算机领域研究的热点. 这种情况下, 多核并行计算应运而生. 多核处理器简单来说是在一个处理器中集成两个或者两个以上的处理器核心. 多核并行计算(简称并行计算)是指在多核处理器的硬件环境中, 同时对多个任务进行处理[1,2]. 运用并行计算技术可以充分利用多核资源, 发挥多核处理器的巨大优势.

当前, 国内外的并行计算主要应用在三个方面[3]: 一是高性能计算, 如: 数据挖掘、金融分析、大气模拟等. 卢浩, 王少华等人将并行计算应用到了水文分析中, 提出了基于 OpenMP(Open Multi-Processing, 开放多处理)框架的并行化水文分析算法[4]; 二是图形处理, 如: 图像渲染、图像压缩、图像处理[5]等. 周勇在数据流挖掘中对 GPU 并行计算结构进行了研究, 提出了基于 GPU 的数据流通用并行处理模型[6], Svetislav Momcilovic 与 Leonel Sousa 研究了在非共享存储模式下基于多核处理器架构的视频图像处理[7]; 三是游戏, 如: 游戏环境的模拟、人物的设定、动作和声音的协调等. 著名的网络游戏, 如魔兽争霸、使命召唤, 早于 2006 年就在 Intel 公司的配合下, 针对多核处理器进行了优化. 由此可见, 并行计算已经在很多领域得到了广泛的应用, 并取得了良好的效果.

① 收稿时间:2015-03-25;收到修改稿时间:2015-05-04

但在并行计算的应用过程中, 主要是从问题的整体上来规划并行计算的过程, 很少将研究的重点放在面向解决问题的过程上, 尤其是对于迭代求解的问题. 因此, 研究面向过程的任务并行化执行的一般方法有助于快速求解迭代问题. 面向过程的任务并行化设计方法综合考虑了任务执行中遇到的各种情况, 针对任务执行的不同阶段应该设计出不同的并行处理方法. 因此, 针对任务执行的不同阶段, 如何利用该过程的特殊性提出并行化设计就是面向过程的任务并行化设计的研究内容.

1　并行编程模型及一般过程

1.1 并行编程模型简介

并行编程模型是将并行算法转换成程序的规则或标准, 它用于定义基本并行程序的结构和语法内容. 目前主要的并行编程模型有两类[8]: (1)基于共享变量的共享存储并行编程, 主要的代表有: OpenMP、POSIX Threads 等; (2)基于消息传递的并行编程, 主要有: MPI、HPF 等. 目前, 应用比较多的编程模型主要有: OpenMP、POSIX Threads 和 MPI.

其中, OpenMP 是共享存储系统上的一个编程模型, OpenMP 的编程是以编译指令显式地要求编译器进行编译, 从而实现并行化.

POSIX Threads 是一个可移植的多线程库, 提供了很多直接控制的多线程应用程序的开发接口, 主要包括: 线程的生成、线程的终止、线程的同步以及有关的一些辅助操作.

MPI 是一个消息传递接口的标准, 主要用于开发基于消息传递的并行程序, 实现一个通用的基于分布式存储的并行计算系统.

分析这几种并行编程模型, OpenMP 更适合用来求解迭代问题, 因为只须在求解迭代问题的程序中加入编译指令就可实现程序的并行化. OpenMP 编程模型相对于 PThreads 来说, 实现起来比较简单, 对串行程序的改动少. 更重要的是如果系统的编译器不支持 OpenMP 的编译指令, 编译器就会自动忽略编译指令, 具有良好的可移植性. 因此, 本文使用 OpenMP 并行编程模型来实现迭代问题的求解.

1.2 并行计算设计的一般过程

当前最主流的并行计算设计的过程是 PCAM 设计流程[9], 在多核处理器的条件下, 同样可以得到应用[10]. PCAM 设计流程主要分成以下四步: 问题划分(Partitioning)、任务通信(Communication)、任务组合(Agglomeration)、处理核映射(Mapping), 如图 1 所示.

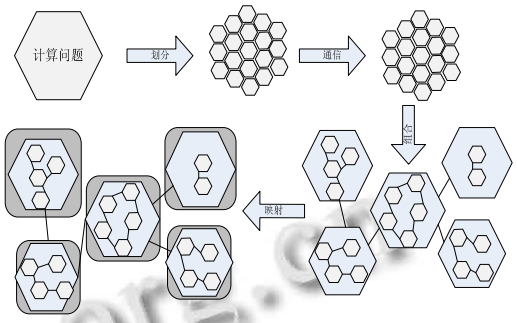

图 1　并行设计的一般流程

(1)问题划分: 将整个问题划分成一些小的任务, 目的是使任务之间尽可能独立以达到可以并行执行的目的, 其方法是先进行问题中的数据分解, 然后进行功能的划分, 使计算过程中的数据相关性降到最低.

(2)任务通信: 确定划分的任务之间的关联性, 以确保任务的正确执行和任务划分的合理性. 在共享存储系统中, 任务之间的关联性主要是通过共享变量的访问来体现, 因此要控制好共享变量的访问次序.

(3)任务组合: 按照性能要求和任务执行的代价来重新评价前两个阶段所得到的结果, 如有必要, 就适当组合任务来提高性能或者减少执行代价.

(4)处理核映射: 将组合好的任务分配到处理器核心上, 将可以并行执行的任务放到不同的处理核心上以达到并行化的目的, 或者将通信量比较大的任务组合映射到同一核心上来降低通信代价, 最终的目标是最大并行化, 最小化执行代价.

这四个步骤中, 前两步主要是从任务的整体来考虑执行的并行性, 后两步关注的是执行过程, 提高局部的性能. 面向过程的并行化任务执行方法就是将 PCAM 流程中后两步的设计放大到整个问题解决过程中, 下面将通过 K-means 聚类方法的实现来具体介绍这个过程.

2　K-means聚类算法的并行化实现

面向过程的并行化任务执行方法的核心是针对任务的不同阶段进行并行优化. K-means 聚类算法是基于迭代来求解的分类问题. 由于每一次的迭代会依赖上一次的问题求解结果, 因此迭代的次数并不确定.

针对此类问题, 运用面向过程的并行化任务执行方法可以对每次迭代过程中的局部进行并行化实现, 从而实现整体任务快速执行的目的.

2.1 K-means 聚类方法的流程

K-means 聚类方法是经典的基于距离的聚类方法, 采用距离的大小评价相似性, 即距离越小, 相似性越大. 该算法的基本思想是: 从 N 个数据对象中任意选取 k 个数据作为初始的聚类中心, 而对于剩余的数据对象, 则计算它们与这些聚类中心的相似度(距离), 分别将它们归类到与其最相似的类别中, 然后重新计算每个聚类的中心; 然后不断重复这一过程直到满足收敛条件为止, 计算流程如图 2 所示.

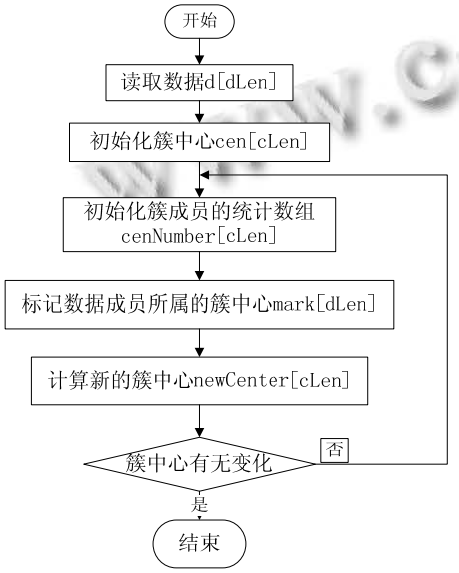

图 2 K-means 聚类算法流程

2.2 K-means 聚类方法的并行化设计

K-means 聚类方法是通过迭代来进行求解的, 结束的条件是簇的中心不再发生变化, 因此在程序运行之前, 我们并不知道程序要迭代多少次, 所以从整体上来进行问题的划分或任务的分解就显得有点力不从心. 但是我们可以深入到程序运行的过程中, 去研究任务执行的具体过程, 如果我们并行化执行任务的每一个阶段, 理论上可以使得整体的任务得到快速执行.

利用面向过程的并行化设计思路对 K-means 聚类方法中的关键过程进行并行化设计:

(1)初始化簇中心. 数据对象 *d*[*dLen*]的前 *cLen* 个元素赋值给初始簇中心 *cen*[*cLen*], 这样就可以将初始化的任务均分给不同的线程, 每个线程的任务量为 *cLen/threadCount*, 赋值的对象不同提供了并行执行的可能.

(2)类别划分, 将剩余的数据对象根据相似度划分到不同的聚类中. 体现在程序中就是首先计算数据对象到 *cen*[*cLen*]的距离, 然后根据距离的大小, 设置标记数组 *mark*[*dLen*]的值. *mark*[*dLen*]来表示数据对象属于哪个聚类. 其中设置 *mark*[*dLen*]的值属于临界区, 对于 *mark*[*dLen*]的访问应该互斥访问. 并行化设计体现在计算数据对象到 *cen*[*cLen*]的距离, 每个线程执行 *dLen/threadCount* 个数据对象的计算, 数据对象的不同保证了并行执行.

(3)计算新的簇中心. 这部分的并行实现相对比较困难, 体现在程序中是首先通过读取标记数组 *mark*[*dLen*]得到数据对象属于哪一类, 然后累加到相应的类别 *sum*[*cLen*]中. *sum*[*cLen*]的累加属于临界区, 设置成互斥访问后和串行执行没有任何区别. 我们可以通过变量拆分的方法来实现并行, 具体的过程是: 设置一个数组 *section*[*threadNum*]来表示每个线程的累加和, 这样每个线程访问的是私有变量的累加. 所有线程执行完毕后, 全部累加到 *sum*[*cLen*]中. 并行执行是通过不同的变量累加来实现的. 通过增加变量来实现执行时间上的降低, 典型的以空间换取时间的做法. 注意, 在计算完 *section*[*threadNum*]的值后, 需要设置线程同步的标志, 确保所有线程已经执行完毕.

单次的迭代过程的任务并行执行过程, 如图 3 所示.

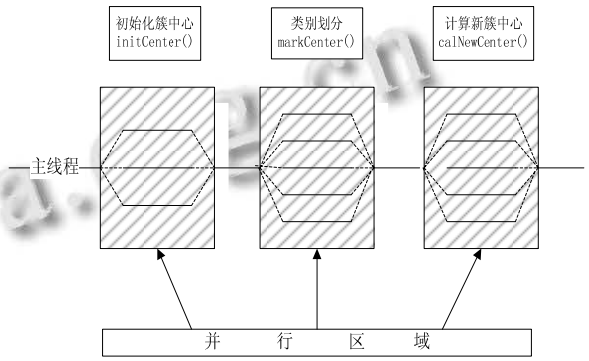

图 3 单次迭代并行执行过程

2.3 实验结果

面向过程的并行化 K-means 聚类方法根据 OpenMP 编程模型, 使用 C 作为编程语言实现, 程序运行平台是 Intel(R) Pentium(R) D CPU, 主频是 2.8GHz、512M 内存, 操作系统为 CentOS release 6.4, 编译环境是 GCC 4.4.7. K-means 聚类方法串行和并行运行的时间如表 1 所示.

表 1 串/并行运行的时间(单位: ms)

数据(组)	1000	2000	3000	4000	5000	6000	7000	8000	9000
串行	67.47	69.21	136.01	207.06	291.13	478.62	483.2	551.29	426.78
并行	107.26	79.28	147.04	179.89	240.01	374.51	357.92	396.33	301.19
迭代次数	31	16	21	24	27	37	32	32	22

实验说明: 数据对象是包含 5 个属性的结构体, 结构体中的属性是通过生成 100 以内的随机数进行赋值. 每组数据运行 10 次, 取其平均值. 初始的簇的类别数为 5, 即将数据对象归为 5 类.

加速比是指在相同的数据规模和运行环境下, 串行执行时间与并行执行时间的比值, 即加速比 $S=Ts/Tp$, 其中 Ts 为串行执行时间, Tp 为并行执行时间. 加速比的统计情况如图 4 所示.

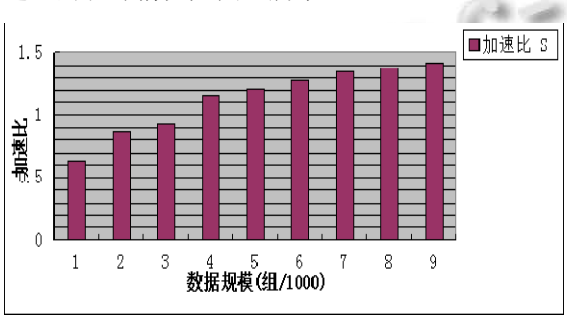

图 4 加速比统计情况图示

2.4 结果分析以及总结

从图中可以看出, 当数据规模较小时, 并行计算的优势没有得到发挥. 原因是线程的创建、同步、终止等开销与迭代的次数有关, 迭代的次数越多, 线程开销越大, 在并行运行的总时间中占的比重也就越大, 这就影响了加速比. 但是迭代次数与数据规模无关, 与初始簇中心的设置有关, 随着数据规模的增大, 线程开销所消耗的时间相对有所减少, 加速比不断的增大, 但是增长率较小, 且加速比的值总体不是很大, 主要的原因如下:

(1)数据规模的不断增大, 迭代次数的不同使得线程的开销相对有所减少, 所以加速比有增大的趋势.

(2)虽然数据规模不断增大, 但是程序执行过程中的数据依赖关系没有改变, 所以加速比的值没有出现大幅增加.

综合以上可知: 总体性能上并行计算的执行效率相对于串行执行有优势, 但是由于线程开销, 当数据规模小于某个特定值时, 并行计算反而不如串行执行的效率高. 所以面向过程的并行化设计方法的实现应该从以下的因素考虑:

(1)细化任务执行的过程, 并行化某一阶段任务的执行, 达到局部加速的效果.

(2)考虑到线程的开销, 对于任务量偏小的阶段, 我们应该舍弃并行化执行, 否则会适得其反, 例如初始化标记数组 mark[cLen].

(3)任务的并行化执行应该保证任务执行的正确性, 所以如何处理数据依赖是必须要考虑的.

(4)并行执行肯定会有线程开销, 只有当数据的规模达到一定值时, 并行执行节省的时间大于线程开销消耗的时间, 并行执行的效果才会体现出来.

3 结语

本文分析了并行计算中应用比较多的并行编程模型和并行计算设计的 PCAM 过程, 在 K-means 聚类算法中应用了面向过程的并行化设计方法, 得到了当数据规模达到一定值时, 并行计算相对串行执行有一定优势的结论, 验证了面向过程的并行化设计方法可以提高任务的运行效率.

参考文献

1 张林波,迟学斌,莫则尧,等.并行计算导论.北京:清华大学出版社,2006.

2 Gorder PF. Muticor processors for science and engineering. Computing in Science & Engineering, 2007, 9(2): 3–7.

3 陈小兰.Linux 应用程序多核并行化方法研究与实现.成都:西南交通大学,2010.

4 卢浩,王少华,等.基于 OpenMP 的并行化水文分析算法的研究与实现.测绘与空间地理信息,2013.8:7–9.

5 Zhang Y, Mueller F. GStream: A general-purpose data streaming framework on GPU clusters(ICPP). 2011 International Conference on IEEE. 2011. 245–254.

6 周勇.基于并行计算的数据流处理方法研究[博士学位论文].大连:大连理工大学,2013.

7 Momcilovit S, Sousa L. Modeling and evaluating non-shared memory CELL/BE type multi-core architectures for local image and video processing. Journal of Signal Processing Systems, 2011, 62: 301–18.

8 王磊.并行计算技术综述.信息技术, 2012,(10):112–116.

9 陈国良.并行计算—结构算法编程(修订版).北京:高等教育出版社,2003.

10 Gembris D, Neeb M, Gipp M. Correlation analysis on GPU systems using NVIDIA's CUDA. Journal of Real-Time Image Processing, 2011, 6(4): 275–280.