文章编号:1673 - 0062(2013)01 - 0064 - 05

基于 OpenMP 的多核系统并行程序设计方法研究

龚向坚,邹腊梅,胡义香

(南华大学 计算机科学与技术学院,湖南 衡阳 421001)

摘　要:随着多核处理器被广泛的应用于桌面系统,如何充分利用多核处理器的计算能力,挖掘应用程序的并行性,以充分发挥多核系统的强大计算能力,成为近几年国内外计算机领域研究的热点,多核系统并行程序设计也成为所有专业软件开发人员必须了解和掌握的一项技术. 本文阐述了一种以 OpenMP 为基础的多核并行程序设计方法,研究并提出了两种符合多核系统并行程序设计特征的编程模式:条件同步模式和线程分组竞争模式. 实验证明这些方法简单高效.

关键词:多核系统;并行计算;OpenMP;编程模式

中图分类号:TP391.1　　**文献标识码**:B

The Research of Multi-core System Parallel Program Design Methods Based on OpenMP

GONG Xiang-jian, ZOU La-mei, HU Yi-xiang

(School of Computer Science and Technology, University of South China, Hengyang, Hunan 421001, China)

Abstract: With the development of multi-core processor widely used in desktop system, how to make full use of multi-core computing power, mining application in parallel, to give full play to the powerful computing ability of multi-core system, becomes the hot point of the domestic and foreign computer field in recent years. Multi-core system parallel programming has become a technology that all professional software developers must understand and master. This paper describes a OpenMP based on multi-core parallel program design methods, and puts forward two kinds of nuclear system with multiple parallel program design feature programming mode: condition synchronization mode and thread grouping

收稿日期:2012 - 11 - 22

基金项目:2011 年湖南省教育厅科研课题基金资助项目(11C1097);2011 年衡阳市科技局课题基金资助项目(2011KG61);2011 年南华大学高等教育研究与改革立项基金资助项目(2011XJG025);2012 年南华大学高等教育研究与改革立项基金资助项目(2012XJG15)

作者简介:龚向坚(1977 -),男,湖南娄底人,南华大学计算机科学与技术学院讲师,硕士. 主要研究方向:计算机网路、并行计算、计算机仿真.

competition mode. Experiments prove that the method is simple and efficient.

key words: multi-core system; parallel computing; OpenMP; programming mode

多核处理器(Multi-Core Processor)[1-2],是指在一个处理器芯片上集成多个处理器核心,其中每个核心都是一个独立的微处理器,这些处理器核心可以并行的执行多个线程[3]. 而多核系统是指使用多核处理器的计算机系统,与对称多处理器系统不同的是,多个处理器核心之间共享着很多系统资源,比如最后一级缓存、系统总线以及主存储器等.

随着多核处理器被广泛的应用于桌面系统,如何利用多核处理器的计算能力,提高应用程序在多核系统上的运行效率,以充分发挥多核微机系统的强大计算能力日益引起人们越来越多的关注. 要充分发挥多核微机系统的计算能力,必须在系统中运行并行程序或者同时运行多个程序,因而在多核系统并行程序设计中需要重点关注的关键问题有两个:1)设计适合多核系统运行的并行程序;2)优化多核系统中单个程序并行运行和多个程序同时并行运行的效率.

本文针对以上两个关键问题阐述相应的简单而有效的解决方法:提出两种多核并行编程模式以保证单个并行程序在多核系统上运行的性能优化和多个程序共享并行运行的多核系统性能优化,并使用 OpenMP 来实现多核并行程序设计.

1　基于多核系统并行特征的编程模式

1.1　多核系统并行特征分析

在多核系统中各个内核共享内存,每个内核对内存的分配、访问、释放都需要由内存管理器集中控制处理,这其中必然要牵涉到锁的请求和释放等问题,需要使用相应的技术来对访存动作进行同步,而获取/释放锁总是会影响系统的性能,因此如何减少访存时对锁的竞争,如何提高获取/释放锁的速度,是提高多核系统性能的关键所在;同时多个线程在多核系统中并行运行,它们彼此之间会互相竞争共享系统资源,这也会明显降低系统性能[4-7]. 因此,如何有效的使用多核系统中的共享资源、解决线程间竞争成为当前多核系统研究领域的一个关键问题.

通过对多核系统的工作原理和并行运行特征进行深入研究和分析,提出了两种符合多核系统特征的并行程序设计模式:线程分组竞争模式和条件同步模式,下面将对它们分别进行详细阐述.

1.2　线程分组竞争模式

所谓线程分组竞争就是将线程分成若干个组,每个组的线程间存在锁竞争,但是不同组之间的线程不存在锁竞争.

在多核编程中,锁竞争导致的 CPU 饥饿现象是引起多核 CPU 性能无法发挥的最重要原因之一,如何消除锁竞争导致的 CPU 饥饿现象成了迫切需要解决的问题. 线程分组竞争模式,它使用锁来进行保护共享数据,但是又避免了锁竞争时出现 CPU 饥饿现象. 我们通过一个简单的例题来对线程分组竞争模式的思想和原理进行说明. 图 1 以添加操作和删除操作作为示例来显示 2 个分组线程竞争的情况.

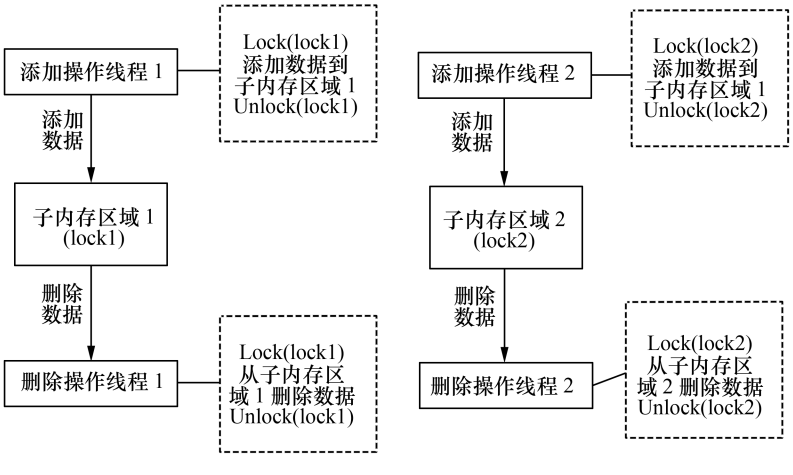

图 1　两个分组线程竞争模型

Fig. 1　The competition model of two groups threads

图 1 中显示了两个分组的线程竞争情况,共有四个线程分成两组进行竞争,添加操作线程 1 和删除操作线程 1 之间存在锁竞争情况,添加操作线程 2 和删除操作线程 2 之间存在锁竞争情况,但是添加操作线程 1(或删除操作线程 1)和添加操作线程 2 和删除操作线程 2 之间不存在锁竞争. 在这种分组锁竞争模式下,任意一组线程中,至少有一个线程在执行,因此如果有 N 组线程的话,那么至少有 N 个线程在执行,如果 N 大于等于 CPU 的核数,那么任意一个 CPU 核上都有线程一直在运行,可以充分保证 CPU 不会产生饥饿现象.

1.3 条件同步模式

条件同步模式的基本思想是在并行算法设计时减少使用同步的次数,比如原来要使用同步 1 000 次,现在改为在满足一定条件下才使用同步,只需要 10 次,那么同步的开销平摊下来就被减少了 100 倍,使得并行算法的效率大大提高.

在多线程编程中,当对共享资源进行操作时,需要使用同步(通常是锁或原子操作)来进行保护,以避免数据竞争问题. 不幸的是,同步操作的开销非常大,比如对一个整数变量进行加法操作,那么同步操作的开销是加法操作的上百倍以上. 如果能设计出更快的锁或更快的原子操作来,那么这种开销自然就减少了. 以目前的技术来看,最快速的原子操作耗时也是普通加法操作的上百倍,所以从这方面着手是非常困难的.

那么能不能从软件算法的角度来减少同步操作的开销呢? 答案是肯定的,基本思想是减少使用同步的次数,比如原来要使用同步 1 000 次,现在改为在满足一定条件下才使用同步,只需要 10 次,那么同步的开销平摊下来就被减少了 100 倍,效率大大提高了. 我们通过一个简单的例题来对条件同步模式的思想和原理进行说明.

在有锁保护的共享队列中,在队列的进队和出队操作时,通常都是使用锁来进行保护的,一个典型的使用锁保护的出队操作伪代码如下:

```
template class <T>
Locked_DeQueue(T &data)
{       Lock();
    DeQueue(data);  //调用串行的出队操作
    Unlock();
}
```

在使用上面的 Locked_DeQueue( ) 函数时,每调用一次,就会发生一次锁操作. 事实上,并不是每次都需要加锁操作的,比如队列为空时,这时实际上是不需要进行出队操作的,完全可以采取一定的方法避免锁操作,但是采用上面的 Locked_DeQueue( ) 函数无法避免锁操作,这就需要对上面的函数进行改进.

一种最简单的方法就是先判断队列是否为空,如果不为空才使用锁保护进行出队操作,从而避免同步操作. 伪代码如下:

```
template class <T>
Locked_DeQueue_a(T &data)
{       If ( ! IsEmpty( ) )
    {           Lock();
        DeQueue(data);  //调用串行的出队操作
        Unlock();
    }
}
```

通过一个不使用锁操作的 IsEmpty( ) 函数,使得上面的 Locked_DeQueue_a( ) 函数能够通过先判断是否为空来减少同步操作,从而提高系统的执行效率.

上面讲的条件同步模式非常适用于具有状态机性质的场合,只有在发生状态切换(例如队列中空或非空的状态的切换)时才使用同步,通过对状态变量(例如 EmptyFlag)的操作来替代其他非状态变量(例如队首指针和队尾指针)的操作,减少同步的使用,从而提高多核系统的效率.

2 基于 OpenMP 的多核并行程序设计

2.1 OpenMP 简介

OpenMP 是一种面向共享存储器的多处理器多线程并行编程语言,线程间通过共享变量传递数据结果[8]. OpenMP 标准形成于 1997 年,它是一种 API,用于编写可移植的多线程应用程序[9]. OpenMP 程序设计模型提供了一组与平台无关的编译指令、指导命令、函数调用和环境变量,可以显式地指导编译器如何以及何时利用应用程序中的并行性. OpenMP 通过对原有的串行代码插入一些指导性的注释,并进行必要的修改,可以快速的实现并行编程,而这些注释的解析由编译器所完成. 目前,C,C ++,Fortran 语言都支持 OpenMP,所有 OpenMP 的并行化都是通过使用嵌入到 C,C ++ 或 Fortran 源代码中的编译制导语句来达到的.

OpenMP 使用 Fork-Join(派生/连接)并行执行模型. 一个 OpenMP 程序从一个单个线程开始执行,在程序某点需要并行时程序派生(Fork)出一些额外(可能为零个)线程组成线程组,被派生出来的线程称为组的从属线程,并行区域中的代码在不同的线程中并行执行,程序执行到并行区域末尾,线程将会等待直到整个线程组到达,然后将它们连接(Join)在一起. 在该点处线程组中的从属线程终止而初始主线程继续执行直到下一个并行区域到来. 一个程序中可以定义任意数目的并行块,因此,在一个程序的执行中可 Fork-Join(派生/连接)若干次. 其工作原理和过程如图 2 所示.

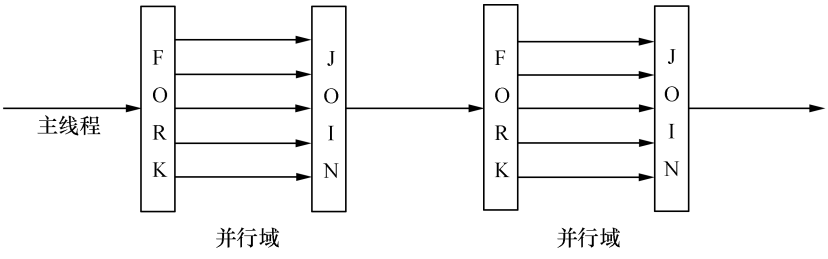

图 2　OpenMP 的 Fork-Join 并行执行模型

Fig. 2　The Fork-Join parallel execution model of OpenMP

这就是标准的并行模式 fork/join 式并行模式,标准并行模式执行代码的基本思想是,程序开始时只有一个主线程,程序中的串行部分都由主线程执行,并行的部分是通过派生其他线程来执行,但是如果并行部分没有结束时是不会执行串行部分的.

2.2　OpenMp 的编译指导语句

OpenMP 是一个编译器指令和库函数的集合,在 OpenMP 中编译指导语句是用来表示开始并行运算的特定注释,在编译器编译程序时,编译指导语句能够被并行编译程序识别,串行编译程序则忽略这些语句. 并行编译程序跟据这些指导语句将相关代码转换成在并行计算机中运行的代码. 一条编译指导语句由 directive(命令/指令)和 clause list(子句列表)组成. OpenMP 的编译指导语句格式为:

pragma　omp　directive-name [clause [[,] clause]...]new-line

Structured-block

OpenMP 的所有编译指导语句以#pragma omp 开始,其中 directive 部分就包含 OpenMP 的主要命令,包括 parallel、for、parallel　for、section、sections、single、master、critical、flush、ordered、barrier 和 atomic. 这些指令用来分配任务,或者用来同步. 后面的可选子句 clause 给出了相应的编译指导语句的参数,子句可以影响到编译指导语句的具体行为,每一个编译指导语句都有一系列适合它的子句,其中有 6 个指令(master,critical,flush,ordered、atomic,barrier)不能跟相应的子句. new-line 为换行符,表示一条编译指导语句的终止.

2.3　基于 OpenMP 的多核并行程序设计

OpenMP 主要是为共享式存储计算机上的并行程序设计使用的,而通过前面对多核系统并行特征的分析我们知道多核系统正是多个处理器核心之间共享主存储器的并行处理系统,使用 OpenMP 来进行多核并行程序设计既简单易行又能够有效地提高效率大大缩短程序运行时间[10-12].

3　实例验证及结果分析

3.1　多核并行程序设计实例

如何将 for 循环里的语句变成并行执行从而提高效率,是由传统串行程序设计向并行程序设计转化中最典型的一个问题. 以此为例,一个简单的测试程序来介绍一下如何使用 OpenMP 进行并行程序设计:

```
void test( )
{    int a = 0;
   clock_t t1 = clock( );
   for (int i = 0;i < 100000000;i + + )
   {              a = i +1;
   }
   clock_t t2 = clock( );
```

```
    printf("Time = %d/n",t2 - t1);
}
int main(int argc,char * argv[])
{   clock_t t1 = clock();
#pragma omp parallel for
    for (int j = 0;j < 2;j + + )
{       test();
    }
    clock_t t2 = clock();
    printf("Total time = %d/n",t2 - t1);
    test();
    return 0;
}
```

在 test()函数中,执行了 1 亿次循环,主要是用来执行一个长时间的操作. 在 main()函数里,先在一个循环里调用 test()函数,循环执行 2 次 test()函数,之后单独调用执行了一次 test()函数.

3.2 实验结果及分析

在主频为 2.66 GHz 的 INTEL 双核 CPU 上运行测试以上程序代码,得到如下的运行结果:

Time = 269

Time = 269

Total time = 269

Time = 269

可以看到在 for 循环里的两次 test()函数调用都花费了 269 ms,但是打印出的总时间却只花费了 269 ms,后面那个单独执行的 test()函数花费的时间也是 269 ms,可见使用 OpenMP 来实现多核并行计算后使得程序的执行效率比原来提高了整整一倍.

4 小 结

本文阐述了一种以 OpenMP 为基础的多核并行程序设计方法,研究并提出了两种符合多核系统并行程序设计特征的编程模式:条件同步模式和线程分组竞争模式. 实验证明这些方法能够很好地帮助我们设计出更加符合多核系统特征的并行程序,并提高相应程序在多核系统中的运行效率,从而充分挖掘出多核系统强大的计算能力.

参考文献:

[1] Olukotun K, Nayfeh B A, Hammond L, et al. The case for a single-chipmultiprocessor [J]. ACM SIGPLAN Notices. ,1996,31(9):2-11.

[2] Cesare Ferri. SoC-TM: IntegratedHW/SWSupportfor transactiona memory programmingon embedded MPSoCs [J]. Published by ACM 2011:39-48.

[3] 周淑贤. 基于 OpenMP 的多核程序设计[J]. 科技信息,2010(9):78-79.

[4] Feng H, Li E, Chen Y, et al. Parallelization and characterization of SIFT on multi-core systems[C]. IEEE International Symposium on Workload Characterization,2008. Seattle, USA,2008.

[5] Tam D, Azimi R, Stumm M. Thread clustering: sharing-aware scheduling on smp-cmp-smt multiprocessors[C]// The 2nd ACM SiGOPS/EuroSys European Conference on Computer Systems. Lisbon, Portugal,2007:47-58.

[6] 周伟明. 多核计算与程序设计[M]. 武汉:华中科技大学出版社,2009.

[7] 陈国良,安虹,陈峻,等. 并行算法实践[M]. 北京:高等教育出版社,2004.

[8] Lee R, Ding X, Chen F, et al. MCC-DB: minimizing cache conflicts in multi-core processors for databases [J]. Proceedings of the VLDB Endowment. 2009,2(1): 373-384.

[9] 蒋弘山,田金兰,张素琴,等. OpenMP 中隐式数据并行编译策略[J]. 清华学报,2004,44(1):54-57.

[10] Eigenmann R, Voss M J. OpenMP shared memory parallel programming[C]. Berlin: Springer,2001.

[11] 张平,李清宝,赵荣彩. OpenMP 并行程序的编译器优化[J]. 计算机工程,2006,32(24):37-40.

[12] 刘凯,寇正. OpenMP 在并行计算中的应用[J]. 微型机与应用,2003(12):12-14.