

TBB 多核编程及其混合编程模型的研究

胡 斌,袁道华

(四川大学 计算机学院,四川 成都 610065)

摘 要:多核处理器越来越普及,如何通过软件技术最大提升 CPU 每个核心的使用率,成为热点问题。引入多核并行编程模型 Threading Building Blocks,并与 raw threads、OpenMP 进行各方面详细比较,分析了其优劣。并研究了 TBB 结合 MPI 在 SMP 集群系统上实现高效的混合并行计算应用的方法。最终发现 TBB 在多核编程方面有显著的优势。TBB 和 MPI 的结合,又为多核处理器结点集群提供了并行层次化结构,大大优化集群的性能。

关键词:TBB;多核处理器;多核并行编程;raw threads;OpenMP;对称多处理器集群;MPI

中图分类号:TP311

文献标识码:A

文章编号:1673-629X(2009)02-0098-04

TBB Multi-Core Programming and Research on Its Hybrid Paradigms

HU Bin, YUAN Dao-hua

(College of Computer, Sichuan University, Chengdu 610065, China)

Abstract: Nowadays multi-core processors are becoming more and more common. How to make every core in a CPU work most effectively in order to enhance performance through software multi-threading has become a hotspot. So Threading Building Blocks was introduced. Compared the advantages and disadvantages of TBB to raw threads and OpenMP. Also researched on the computing applications of mixing TBB with MPI on SMP cluster. As a result, found TBB had many significant advantages on multi-core programming. Mixing TBB with MPI provides a layered structure for computing applications on multi-core cluster, and it highly optimized the performance of clusters.

Key words: Threading Building Blocks; multi-core processor; multi-core parallel programming; raw threads; OpenMP; SMP Cluster; MPI

0 引 言

在摩尔定律和芯片功耗的限制下,单纯通过提升 CPU 的时钟频率来提升 CPU 的运算速度已经显得步履蹒跚。同时 CPU 速度提升和内存频率的提高不同步的问题也成为制约计算机性能提升的一个瓶颈问题。这些问题促成了多核处理器的研究和诞生。随着 Intel, AMD 多核产品的投放市场,多核处理器正在逐步普及。

但通过应用发现传统的在单核计算机上运行的大多数程序并没有多少性能提升。解决的方式之一是通过多处理器编译器把顺序程序编译成并行程序。但结果并不能令人满意,加速比一般只有 10%~30%^[1],原因是在很多时候只有其中的一个核处于工作状态。

这种机械的转换并不能很好提升程序性能。因此,如何根据实际情况通过软件技术最大提升 CPU 每个核心的使用率,以达到提升应用程序性能的目的,成为越来越倍受关注的一个问题。开发人员需要更多地去考虑负载均衡、锁竞争、伸缩性等等复杂的问题,因此编程和调试的难度也大大增加。为了简化这些问题,Intel 针对多核平台,推出了 Threading Building Blocks,一个基于 C++ 模板库的并行编程模型。

1 Threading Building Blocks

1.1 TBB 简介

Threading Building Blocks 是由 Intel 针对多核平台开发的一组开源的 C++ 的模板库,基于 GPLv2 开源证书,支持可伸缩的并行编程。它不需要特别的语言或者编译器的支持,因此可以广泛地被用于任何处理器、任何操作系统以及任何编译器。STL 之父, Alexander Stepanov 对此评价说:“TBB……会成为 C++

收稿日期:2008-05-16

作者简介:胡 斌(1983-),男,四川成都人,硕士研究生,研究方向为分布式并行处理与网络计算;袁道华,教授,研究方向为分布式并行计算、网络计算、移动计算。

+模板库中的一个并行的标准。”TBB 可以在 Windows, Linux 和 MacX 上运行,支持 Intel C++、VC 7/8 和 gcc 编译器,这都是由它的特性决定的,因为它不是对现有语言的扩展(不像 OpenMP),而是在现有语言基础上对并行开发概念的封装和实现,底层依然依赖于 raw thread 的支持。

1.2 TBB 的多核并行编程

Threading Building Blocks 的编程模式是使用模板作为通常的并行迭代模型。这些丰富的 C++ 模板类使得程序员不需要很专业地去学习同步、负载平衡、缓存优化等等的问题,而能够轻松地实现自动调度的并行程序,使得 CPU 的多个核心处于高效运转之中。TBB 提升了程序的伸缩性,并且完全支持嵌套的并行编程。程序员可以创建自己的并行组件,以构建大型的并行程序^[2]。同时由于 TBB 是芯片制造商 Intel 开发的,对于多核平台将有更好的支持。图 1 就是一个最简单的 TBB 的示例。

```
#include "tbb/parallel_for.h"
#include "tbb/blocked_range.h"
using namespace tbb;
int main() {
    task_scheduler_init init;
    string str = "hello world";
    parallel_for(blocked_range<size_t>(0, str.size(), 100),
        printf("%s\n", str));
    return 0;
}
```

图 1 TBB 示例程序

首先 TBB 提供了 parallel_for、parallel_reduce、parallel_scan, TBB 实现了与 OpenMP 类似的并行循环指令,甚至在很多方面还提高了并行循环指令的功能。例如其中的 parallel_scan 的设计就是考虑到并行循环中如果遇到有串行相关性的情况^[3]。对于流和串, TBB 专门提供了 parallel_while、pipeline 和 parallel_sort。Pipeline 可以设置一个或者多个过滤器来处理数据。Parallel_sort 可以对串进行排序,使用方法和 std::sort 相同,不同的只是并行处理。

TBB 还提供了一些并发性的容器 concurrent_queue、concurrent_vector、concurrent_hashmap, 这些容器与传统的数据结构相比都是线程安全的,可以允许多个线程同时进行操作,而又不需要程序员再来自己实现。在多线程程序中,内存的分配成为一个瓶颈,原因是每个线程都会从一个全局堆中竞争全局锁来进行内存的分配和回收。还有就并发程序的假共享问题也是值得关注的。TBB 为此提供了伸缩性的内存分配策略, scalable_allocator、cache_aligned_allocator、

aligned_space 将使并行程序在内存分配上的效率大为提高。当然 TBB 也和其他并行编程 API 一样提供了 mutex 锁。实现原子操作的 atomic 类也是 TBB 的一个特点。TBB 还提供了线程安全的计时机制。

还有一点是 TBB 的重中之重,任务调度机制。由于线程接近硬件层,系统开销和任务相比比较大, TBB 将任务和线程结合,引入了任务调度机制 task_scheduler 和 task, 减少了系统开销。TBB 隐含地初始化了一个全局线程池,自动地通过调度和任务机制来管理线程池。所有的这些功能使程序员能够更容易地编写出高性能的代码。TBB 核心功能见表 1。

表 1 TBB 核心功能

类	作用
parallel_for<Range, Body>	并行循环
parallel_reduce<Range, Body>	并行规约
parallel_scan<Range, Body>	并行扫描, 针对循环中有串行相关性的情况
parallel_while<Range, Body>	针对流和串的并行操作
pipeline	管道类, 可添加一个或多个过滤器
parallel_sort< RandomAccessIterator, Compare>	并行排序
concurrent_hash_map< Key, T, HashCompare>	并发哈希表, 线程安全
concurrent_queue< T>	并发队列, 线程安全
concurrent_vector	并发向量, 线程安全
scalable_allocator	伸缩性的内存分配器, 由处理器数量决定
cache_aligned_allocator	在缓存分界线分配数组, 避免假共享
aligned_space	为数组占据足够空间, 常用于本地变量和域
mutex	锁
atomic< T>	原子操作
tick_count	计时器
task_scheduler_init	任务调度者初始化, 可以 OpenMP 结合
task	任务类
task_list	任务列表

2 TBB 和其他并行编程模型的比较

2.1 Raw Threads

Windows Threads 和 POSIX threads 都属于原线程 (Raw Threads^[4]) 的范畴, 它们是最低层次的并行编程 API。不同点是 Windows threads 应用于 Windows 平台, 有 Win32 这样一个完整的库支持, 相对较为成熟, 而 POSIX threads 主要用于 UNIX/Linux 平台, 但编程难度相当大。

虽然底层的编程提供了很好的适应性, 但在程序设计、开发、调试以及维护上开销都大大增加。对程序员来说使用原线程, 对于基本的协同以及数据共享要

做到正确高效都是一项复杂而枯燥的工作。代码往往很依赖于一些和操作系统相关的具体线程细节问题,迁移起来会比较困难。例如如何最好地管理线程池以及负载均衡锁竞争等问题都是编程过程中最让人头痛的。TBB 相比 Raw Threads 在编程难度上大为降低,同时对于普通程序员来说用 Raw Threads 编写的程序也很难达到 TBB 程序的自动调度的高效率。

2.2 OpenMP

OpenMP 能够为编写多线程应用程序提供一种简单的方法,无须程序员进行复杂的线程创建、同步、负载均衡和销毁工作。OpenMP 形成于 1997 年,用于编写可移植的多线程应用程序。起初只是一个 Fortran 标准,后来又发展到 C/C++,目前在 Intel C++ Compiler 和 Microsoft VS2005 等编译器上都得到了广泛支持^[5]。

OpenMP 采用了共享存储中标准的并行模式 fork-join,当程序开始执行时只有主线程存在,主线程执行程序的串行部分,通过派生出其他的线程来执行其他的并行部分。当重新执行程序的串行部分时,这些线程将终止。如图 2 所示。

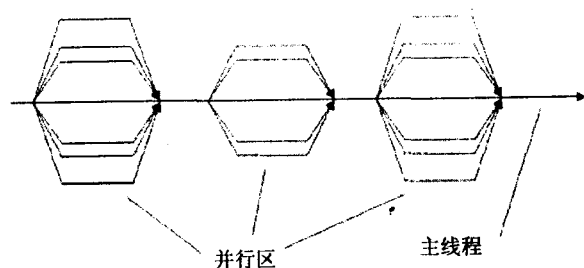


图 2 OpenMP 执行模型

OpenMP 程序设计模型提供了一组与平台无关的编译指导(pragmas)、指导命令(directive)、函数调用和环境变量,通过现实地指导编译器如何将程序中的有关代码编译成为并行代码^[6]。编程语言和一般的 Fortran、C 没有什么区别,只是在程序中嵌入一些标签,来指导编译。例如对于很多循环,如果每一次循环之间没有相关性,则可在之前加入一条 #pragmas omp parallel for 将其转化为并行代码。这样,开发人员就不需要关心具体的实现,只需要关心并行的算法问题。

2.3 Raw Threads, OpenMP 和 TBB 的对比

从表 2 不难看出 Raw Threads 是最为底层次的,提供的功能最少^[7]。OpenMP 和 TBB 都是比较高级的并行,虽然 OpenMP 编程在某些方面比 TBB 更为简单,但 TBB 具有很多 OpenMP 所没有的高级功能,和更为丰富的函数库。例如动态线程调度机制、伸缩性的内存分配器、内嵌式并行程序、并发的数据结构等。

基于这些方面的原因 TBB 在多核并行编程上将

比 OpenMP 有更好的表现。

表 2 Raw Threads、OpenMP、TBB 对比

API	Raw Threads	OpenMP	Intel(r) TBB
任务级并行	-	+	+
数据分解支持	-	+	+
复杂并行模式(非循环)	-	-	+
广泛应用的基本并行模式	-	-	+
伸缩性的内嵌式并行支持	-	-	+
内建的负载均衡机制	-	+	+
密切的支持	-	+	+
静态调度	-	+	-
并发数据结构	-	-	+
伸缩性的内存分配	-	-	+
输入输出主导的任务	-	-	+
用户级同步模式	-	+	+
无需编译器支持	+	-	+
跨操作系统支持	-	+	+

3 混合编程模型的研究

TBB 除了有这些强大的功能之外,还具有良好的扩展性能。由于是使用标准的 C++ 模板类作为 API 的核心,使得 TBB 和 MPI、OpenMP 等结合起来都为简单。对于搭建更高效的 SMP 机群系统,以及编写内嵌式的并行程序都有极大好处。

3.1 TBB 与 MPI 的结合

3.1.1 机群与 MPI

SMP 机群系统是当今搭建高性能计算平台最为应用广泛的。对于那些单台主机无法完成的大型计算任务,搭建机群系统是最好的解决办法。因为它具有价格便宜、构建简单、伸缩性强等优势,许多知名企业和科研单位都广泛地应用了 SMP 机群系统。

MPI 是由学术界、政府和工业协会共同开发的一个消息传递模型的标准,目前是分布式存储系统上应用最为广泛的编程模型。MPI 不是一个专门的编程语言,而是一项标准,同时 MPI 提供了 FORTRAN 和 C/C++ 等多种语言的绑定。MPI 具有可移植性好、功能强大、效率高等优点,适合于粗粒度的并行,支持多种操作系统,如 UNIX/Linux、Windows 等。目前是 SMP 集群系统中最可信赖的平台。MPI 包含了多种通信函数供编程人员选择。MPI 程序可以实现计算与通信的重叠,提高并行程序的性能。

MPI 编程模式优势在于用户可完全控制对数据的划分和进程的同步,进而优化数据局部性和工作流。但它也有许多不足之处:消息传递和全局操作的开销非常大;细粒度任务会引发大量的通信;动态负载均衡

困难;将串行程序转化为并行程序需要对代码作大量改动,编程和调试的难度大^[8]。

3.1.2 TBB与MPI在机群上的结合

TBB在这个范畴上属于共享存储的模型。考虑到SMP集群系统具有层次存储结构的特点,为了充分利用SMP集群层次存储结构的特点,可以考虑将两种编程模型相结合的TBB/MPI层次混合编程模型。该模型具有两层:上层通过MPI的通信在节点之间并行,下层通过TBB在节点内部并行。实现的基本原理是:首先对问题进行MPI分解,将任务划分成通信不密集的几个部分,每个部分分配到一个节点进行,节点间通过消息传递来通信。在节点内部通过TBB的线程池及任务调度机制实现节点内部的并行计算。

随着多核处理器在个人PC和服务器的普及,将TBB与MPI的消息传递机制结合起来,不仅可以发挥出TBB在单台主机上多核并行计算的优势,还能有效地将大型计算任务分配到各个节点上分别处理。TBB与MPI的混合编程模型提供了节点间和节点内两级并行机制,其贡献在于结合了进程级的粗粒度并行和循环级的细粒度并行。TBB/MPI混合编程模型见图3。

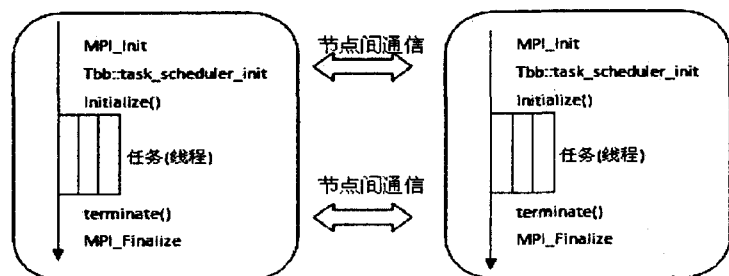


图3 TBB/MPI混合编程模型

3.1.3 优缺点分析

(1)提高了每个节点内的计算效率。通过在节点内使用TBB进行编程将使程序在节点内部运行时更好地发挥多核并行计算的优势。同时线程也比MPI进程耗费资源少得多,比单纯使用MPI时的运算效率大为提高。

(2)减少了节点间通信。通过在外层MPI层进行粗粒度并行,节点内部进行细粒度并行,实现了多层混合粒度的编程。减少了节点间的通信次数,更有利于提高性能。

(3)实现了通信和计算的重叠。大多数MPI的实现如MPICH和LAM,都是使用单线程实现的。这样虽然可以减小同步的上下文切换所要花费的开销,但通信时对前面计算结果的等待仍然会大大降低效率。节点内的并行可以通过其中一个线程通信,其他线程计算,实现通信和计算的重叠,以提高效率。

3.1.4 需要注意的问题

(1)MPI负载均衡问题。本来在SMP机群中负载均衡是由消息传递平台所提供的,混合粒度编程很多时候就需要人工地来分配粗粒度的任务,因此这成为了一个难点。

(2)混合模型要求MPI必须是线程安全的。

(3)避免在TBB的并行区域内调用MPI的通信函数,只能在主线程中进行MPI的消息传递通讯。

(4)数据的完整性,在使用接收数据和发送数据时,都要保证相关的计算已经完成,否则很容易造成计算结果的错误。

(5)点对点通信时必须注意正确的发送和接收。

3.2 TBB与OpenMP的结合

TBB还提供了与OpenMP相结合的功能。由于两者是在同一层次上的并行,一般并不推荐使用。对于特殊的情况来说,使用也是允许的。示例如下:

```
void OpenMP_Call_TBB( int n ) {
    #pragma omp parallel
    {
        task_scheduler_init init;
        #pragma omp for
        for( int i=0; i<n; ++i ) {
            TBB code here
        }
    }
}
```

4 结束语

在并行程序开发中,负载平衡、锁竞争、伸缩性等问题一直都是一个难点。有效的并行程序设计决定于并行体系结构、并行算法和并行编程模型等多方面因素。文中引入了一种新的并行编程模型TBB,并将TBB与其他的并行编程模型进行了比较。TBB将有利于并行多核编程中各方面问题的解决,同时大大降低了编程难度。文中后半部分重点研究了如何将TBB与MPI相结合,搭建SMP集群系统上的混合并行计算应用。并对混合编程的优缺点及需要注意的问题进行了详细分析。

参考文献:

- [1] Reinders J. Programming For Parallelism[EB/OL]. 2007. <http://www.cajcd.edu.cn/pub/wml.txt/980810-2.html>.
- [2] Reinders J. Intel Threading Building Blocks[M]. [s.l.]: O'REILLY 出版社, 2007.
- [3] Intel 公司. Intel Threading Building Blocks reference manual [EB/OL]. 2007. <http://threadingbuildingblocks.org/>.

(下转第104页)

自由摄影机,查看注视摄影机方向的区域

.....

```
c.type = #target //相机的方向一直对着目标物体
```

```
c.target.wirecolor = wcol
```

```
c.fov = 25.5 //经过测试得出的焦距,这样出来图像正好占满整个画面
```

2.3 系统实现

最终系统共分为四部分:时间输出、输出文件大小、渲染选项、输入输出设置。

其中时间输出部分有 single(单帧)和 Active Time Segment(活动时间段)两个选项,如选择 single 则输出是 jpg 图像,如果选择 Active Time Segment,可以在下面范围中填入渲染的帧的范围,例如添 0,100 则它表示渲染的是从第 0 帧到第 100 帧的动画,输出格式为 .avi 的文件。

输出文件大小模块选项组可以控制最后渲染图像的大小和比例。可以在下拉式列表框中直接选取预先设置的工业标准,也可以直接指定图像的宽度和高度,这些设置将影响渲染图像的纵横比。

渲染操作模块是一些常用的一些渲染选项,包括 Video Color Check(视频颜色检查),Render Hidden Geometry(渲染隐藏的对象),Super Black(超黑),Force 2-Sided(强制双面),Atmospherics(大气),Render to Fields(渲染到场景)。

输入输出模块是对输入输出的一些设置,包括选择输出路径,选择相机,使用设备,选择要被渲染的文件等常用设置。

渲染文件时首先将渲染参数设置后,在输入输出模块中选定要渲染的文件,最后点击 Render 按钮,3DSMAX 就开始自动进行渲染。例如当输入文件夹中存放的是图 2 中所示的 5 张点云图像时候,得到的渲染的图像如图 4 所示。



图 4 自动渲染的头像

同理如果设置头像的旋转度数,使得头像在任何方向进行旋转,当对输出的文件命名就可以输出这个

头像旋转时候的图像。图 4 所示头像为每次 Z 轴转动 10 度时, Y 轴转动 -45 度的图像。渲染输出图像如图 5 所示。

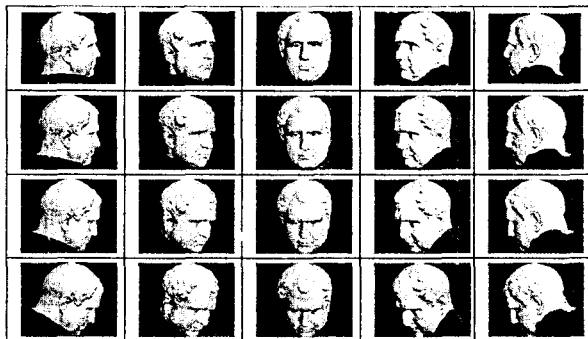


图 5 自动输出旋转的头像

3 结束语

为了解决物体识别领域中大量 3D 图像到二维图像的转换问题,利用 3DSMAX 的二次开发工具 MAXScript 语言设计并实现了自动批量渲染系统,以实现图像的批量渲染功能。此系统可以自动导入、渲染以及输出文件,并且可以修改设置常用的渲染参数。通过点云模型渲染测试,其渲染效果形象逼真,渲染速度快、效率高,可用于物体识别实验的前期准备工作,并可适用于多种类型的三维模型渲染,具有很高的推广价值。

参考文献:

- [1] Jain A K, Flynn P J. 3D Object recognition systems[M]. Amsterdam: Elsevier Science Publishers, 1993.
- [2] Webster M, MacLain. Figural after-effects in the perception of faces[J]. Psychonomisc Bulletin and Review, 1999, 6(4): 647-653.
- [3] Zhao L, Chubb C. The size-turning of the face-distortion after-effect[J]. Vision Research, 2001, 41: 2979-2994.
- [4] 龚蕾, 赵力, 李萍. 高精密度点云数据三维人头模型的变形[J]. 计算机应用, 2006, 26(6): 1383-1384.
- [5] 高志清, 陈云龙. 3DS MAXScript 动画制作基础[M]. 北京: 人民邮电出版社, 2000.

(上接第 101 页)

- [4] Akhter S, Robert J. 多核程序设计技术——通过软件多线程提升性能[M]. 北京: 电子工业出版社, 2007.
- [5] Akhter S, Roberts J. 多核编程[M]. 北京: 电子工业出版社, 2007.
- [6] Intel 公司. Intel Threading Building Blocks tutorial[EB/OL]. 2007. <http://threadingbuildingblocks.org/>.
- [7] 单莹, 吴建平, 王正华. 基于 SMP 集群的多层次并行编程模型与并行优化技术[J]. 计算机应用研究, 2006(10): 254-260.
- [8] 冯云, 周淑秋. MPI + OpenMP 混合并行编程模型应用研究[J]. 计算机系统应用, 2006(2): 86-89.