第 35 卷 第 8 期 计 算 机 工 程 2009 年 4 月
Vol.35 No.8 Computer Engineering April 2009
·软件技术与数据库· 文章编号：1000—3428(2009)08—0062—03 文献标识码：A 中图分类号：TP311.52

基于多核处理器的并行编程模型

伊君翰
(复旦大学并行处理研究所，上海 201203)

摘 要：为解决传统编程模型与并行架构间存在的矛盾，针对多媒体和网络应用程序的特点，提出一种基于多核处理器的并行编程模型，该模型采用节点化的并行程序描述方式，将并行编译器划分到多个核上运行。实验结果表明，这种新的并行编程模型能有效提高程序的执行效率。

关键词：编程模型；并行；多核；编译器

Parallel Programming Model Based on Multi-core Processor

YI Jun-han
(Institute of Parallel Processing, Fudan University, Shanghai 201203)

【Abstract】To solve the contradiction between traditional programming model and parallel structure, aiming at the characters of multimedia and network application program, a parallel programming model based on multi-core processor is proposed, which uses parallel program description modes for nodes to make the parallel compilers run in mult-cores. Experimental results show this novel model can promote the efficiency of program running effectively.

【Key words】programming model; parallel; multi-core; compiler

1 概述

处理器不断发展，由纯粹的频率提升逐渐转到多核运算及并行执行的方向上。与单核处理器相比，多核处理器能够以更低的频率处理更高的工作负载，因此，能在提升处理器性能的情况下降低功耗，减少散热。如果程序员编写的程序没有针对多核的特点来设计，那就不能完全获得多核处理器带来的性能提升。为充分利用多核性能，需要设计一种并行编程模型来编写更高效的多核程序。特别是在多媒体和网络应用中，处理器的利用率显得尤为重要。此类应用计算量大，对实时性要求高。如何在多核环境下更好地发挥处理器的性能优势，满足多媒体和网络应用的需求就成了当前面临的主要问题。为解决这个问题，本文提出一种新的并行编程模型，该模型基于多核架构，将单一程序划分为可并行执行的多个任务分配到多个核上，通过优化调整达到负载平衡，更加有效利用多核处理器的每个核心，从而获得较大的性能提升。

2 背景

2.1 并行编程模型

串行编程模型和并行架构的矛盾日益突出，这就迫切需要根据多核的平台架构来设计一种并行编程模型，使基于新的编程模型生成的程序可以在多核平台上执行效率更高，性能更优越[1]。

并行程序的设计方法主要有以下 3 种：

(1)扩展编译器。通过开发并行化编译器，使其能够发现和表达现有串行语言程序中的并行性，直接编译出并行程序。这种把并行化的工作留给编译器的方法虽然降低了编写并行程序的成本，但由于程序的复杂性，编译器不能识别相当多的可并行代码，因此造成并行力度的不足。它对编译器的分析和划分能力要求很高，而程序的并行成分的识别是比较困难的。

(2)扩展串行编程语言。通过增加函数调用或者编译指令来表示低层语言以获取并行程序。用户能够利用已经存在的库函数创建和结束并行进程或线程，并提供同步与通信的功能函数等。这种方法可以以较小的代价修改源码，并得到较好的并行程序。

(3)创造一个并行编程模型。这也是本文所采取的设计方法。通过创建新的并行模型以及支持该模型的并行编译器，使程序和多核平台架构有很好的结合。

扩展编译器或串行语言的方式都是以串行语言为基础，它们都不能解决串行模型和并行平台之间矛盾这一根本问题，特别在一些特殊的领域，如多媒体和网络应用，传统编程语言本身在这类应用上匹配性就不高，只有通过并行编程模型才能让多核架构的优势得以发挥。

2.2 多媒体和网络应用

在多媒体及网络应用中，如音视频编解码、网络数据的传输和分析等，它们的应用对程序的性能提升有较高要求。这类应用主要以一种“流”(streaming)形式存在，数据量大、数据流方向性基本单一，对实时性要求高[2]。

此类应用本身就有很好的并行性，但在传统的串行架构上很难再做进一步优化，给多核的并行化平台带来了机遇。本文没有采用扩展编译器或扩展串行编程语言的方法，而是创建一个新的并行编程模型——ComponentC Streaming Language，多媒体应用程序虽然数据流庞大，但过程控制流

作者简介：伊君翰(1983－)，男，硕士研究生，主研方向：体系结构，编译原理
收稿日期：2008-09-20 **E-mail：**052053026@fudan.edu.cn

却相对简单，因此，编写这样的并行程序对程序员来说十分方便，且并行化程序更有利于编译器的分析处理，使性能得到较大提升。

3 模型实现

根据应用中这种“流”的特点，程序员在编写程序时可以较容易地把其分解成若干个“节点”[3](component)。每个节点内部独立负责对数据的分析处理，而节点之间通过通信通道(channel)传输数据，信道是单方向的，由接口协议控制输入输出规则。由这些就构成了上层应用模型的基础，再加上描述平台信息的底层架构模型就组成了 ComponentC Streaming Language。

3.1 上层应用编程模型

由于传统编程模型的串行化描述方法无法节点化描述程序，单一程序在编译器编译后也只能在单一核上运行，因此本文创造一个基于传统编程语言之上的上层应用编程模型，以节点化的方式描述程序，以信道方式描述节点间的通信。每个节点在本文的并行编译器的预编译下可以生成单独任务的程序，分别以传统编程语言描述，再经过编译器编译就可以被划分到多个核上去并行执行。

上层应用编程模型可以构建于任何传统语言之上，只要可以满足节点化的描述方式，都可以划分到多个核上去并行执行。由于 C 语言在网络和多媒体应用中的普遍性和基础性，因此本模型以 C 语言为基础。

ComponentC 不像传统 C 由单一的 main 函数开始串行执行，而是由多个节点描述而成。ComponentC 的语法如下，主要分为节点声明、信道声明和全局变量声明。

```
Prog → Decl+
Decl → CVarDecl | CTypeDecl | ChannelDecl | ComponentDecl
ChannelDecl → channel<type> ident
ComponentDecl → component ident REP_DIR{ Field*}
REP_DIR → replicate INTEGER | epsilon
Field → CTypeDecl | CompntVarDecl | CompntFnDecl
CompntVarDecl → CVarDecl
CompntFnDecl → CFunctionDecl
%CVarDecl is standard C variable declaration syntax
%CFunctionDecl is standard C function declaration syntax
%INTEGER is an integer
```

3.1.1 节点

在每个节点内，有一个初始化函数 void init()作为配置函数在节点创建时运行，而后运行作为节点主函数的 void work()，替代了 main 函数的作用，它的主要功能是控制节点的工作，从输入信道接收数据，在节点内部运算分析，再将结果输出到输出信道。在节点内还可以声明其他函数，作为节点内部运算函数由 work()函数调用。

3.1.2 信道

为满足单向信道的输入输出协议，对信道设计 2 个操作：channel_put 和 channel_get，构成一个类似堆栈的 FIFO 结构。在并行平台上，将多个运算核心集成在一个处理器上，这跟在一块主板上集成多个处理器的多路 SMP 系统相当相似，不同之处仅在于多核处理器的核心之间相互交换数据并不需要通过前端系统总线，而只需用共享缓存的方式进行，而多核在缓存上的优势也使得多核之间的数据交换速度更快，效率更高。

3.1.3 全局变量

虽然各个节点之间保持基本独立，但是有时还是需要一些全局变量可以在多个节点之间共同使用。在本文设计中，尽量避免这种方式，尽可能使用信道方式来达到更好的并行性。一旦使用全局变量，程序员就必须确保全局变量在不同节点之间使用的同步性，这对于此模型来说是很大的负担。

3.2 底层架构编程模型

多核处理器快速发展使核的数量由两核、四核到 *n* 个核，这就需要本文的编程模型可以在多平台上得以实现。为让底层的架构改变不对上层编程构成影响，本文还设计了底层架构编程模型 Processor Unit Mapping Script(PUMS)。这个模型把这些关于平台结构信息与程序节点划分信息有效地结合起来，使它能针对不同平台编译出相适应的高效程序。

PUMS 需要给出平台信息，包括核的数量以及每个核上运行的进程数量。这样，编译器就可以将每个节点划分到具体的核的某一个进程上去执行。本文的实现是通过调用 sched_setaffinity 这个函数完成的，这个函数可以通过 CPU 的 pid 将程序和核绑定起来，通过分解程序节点，将每个节点转化为绑定到某个核上运行的程序，以实现多核的划分。

3.3 模型框架

本文编译器是个 source to source 的并行编译器 Agassiz(复旦大学与明尼苏达大学合作开发的并行化编译器)。ComponentC 和 PUMS 经过 Agassiz 的识别可以将程序中的每个节点转化为传统的 C 程序，并划分到具体核上的进程去执行，然后用传统编译器 GCC 编译成可执行代码，运行于多核平台，模型框架如图 1 所示。

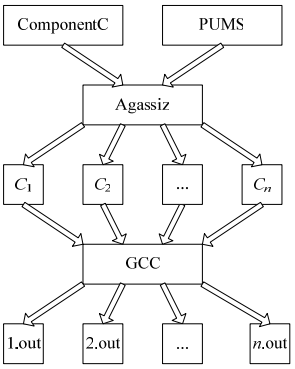

图 1 模型框架

4 性能优化

在多核处理器中，若要很好地发挥出多个核心的性能优势，就必须保证分配到各个核心上的任务有一个很好的负载平衡[4]。

4.1 节点复制

随着处理器中核的数量增多，程序中节点的数量会小于核的数量，那么本文的划分还是会造成有些核在工作而有些在空闲的情况。基于目前主流的多核处理器都是双核或四核，虽然每个核上都在运行程序节点，但是其上的负载并不相同，运行速率的不匹配会导致节点之间流水线的延迟，从而降低多核的并行效率。因此，本文对 ComponentC 的语法加入了复制选项 replicate。当某一可复制节点的输入信道的接收速率大于其输出信道产生速率，即说明这个节点需要被复制，增加其输出的速率。

图 2 给出了节点复制模型，程序员在描述 ComponentC 节点的同时告诉编译器，哪些节点是可以复制的。可复制的节点必须满足同步无关的要求，即复制后的节点共用输入输出信道，且并不会产生时序上的冲突，这就要求信道传送的

数据之间是不相关的，经过节点的计算后以不同于输入的顺序输出不会对程序产生影响。

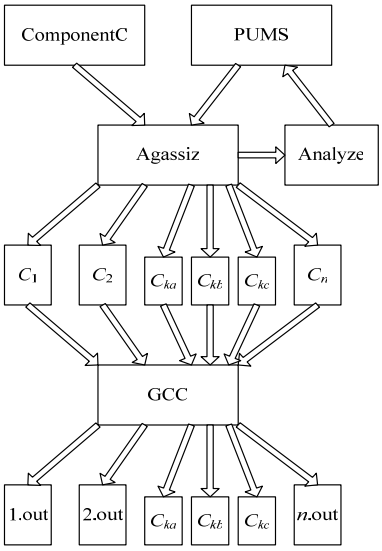

图 2　节点复制模型

当 Agassiz 分析 ComponentC 和 PUMS 时，根据复制信息将某些节点复制，如节点 *Ck* 的输入信道的接收速率大约是其输出信道产生速率的 3 倍，*Ck* 被复制成 3 个子节点 C_{ka}, C_{kb}, C_{kc}，分配到多核平台上运行。在处理器没有空闲核的情况下，就会分别以进程的形式绑定在同一个核上运行，这样的程序就可以更有效地在多核上执行。

4.2　节点合并

反之，当 ComponentC 中的节点过多时，太多的进程在同一个核上的执行会使效率降低。这就需要合并相邻的节点。这时着重考虑的是紧密联系节点间的信道。

假设节点 *p* 和 *q*，*p* 的输出信道只有输出到 *q*，而 *q* 的输入信道都来自于 *p*。这时就可以认为 *p* 和 *q* 是紧密联系的节点，当需要的时候可以将节点合并。

4.3　节点拆分

节点拆分与合并相反，但不等同于复制。节点拆分是将一个节点分为 2 个，并且实现节点间的信道，这是为了多核上的流水操作能够提高程序的性能而需要的方法，它的实现过程和节点合并是完全相逆的，在应用中，本文更提倡程序员描述更多的节点，这样在编译器分析时，可选择性地合并。

5　数据分析

在完成模型的实现后，本文在 Intel 双核服务器上测试多媒体和网络应用。其中，IPfragment 是指将网络数据分段打包再分发出去的操作，本文将这一实例用 ComponentC 模型描述，共有 2 个节点 packet_source 和 fragment，且 fragment 是可以复制的节点。

在实际测试中，本文采用多种分配方案。由于环境限制，只有含 2 个核的处理器，因此本文的复制只能以多个进程的方式在双核上运行，表 1 为具体的实验结果。表中，cpuM[*x*]代表核 M 上运行了 *x* 个进程，主要有 3 种分配方案。cpuA[0]cpuB[*x*]是将一个核空闲而将程序全部运行于一个核上，*x* 代表 packet_source 进程和复制 *x*-1 次的 fragment 进程(其中，cpuA[0]cpuB[1]代表只有一个节点，没有拆分 packet_source 和 fragment 的原始程序)；cpuA[1]cpuB[*x*]是将 packet_source 绑定在一个核上，而将复制 *x* 次的 fragment 绑定到另一个核上；cpuA[2]cpuB[2]和 cpuA[3]cpuB[3]是平均分配两个核上的进程，将 packet_source 和复制的 fragment 分配到 2 个核上。

表 1　实验数据

运行次数 *x*	cpuA[0]cpuB[*x*]的运行时间/s	cpuA[1]cpuB[*x*]的运行时间/s	cpuA[*x*]cpuB[*x*]的运行时间/s
1	17.43	17.36	—
2	17.49	16.68	17.20
3	17.76	16.58	17.32
4	17.93	16.63	—
5	18.04	16.87	—

当只在一个核上运行程序时，进程越多效果越差，合并 2 个节点是最好的选择，这说明要发挥多核的优势就不能在一个核上运行大量程序而使另外一个核空闲。而分到 2 个核上的情况就使效率明显提高，但划分的方式是有选择的，复制的次数不是越多越好。本文发现将 packet_source 绑定到一个核上，而将 fragment 复制 3 次绑定到另一个核上的效果是最好的。一方面，这由平台上的多核 CPU 性能决定；另一方面，也是由程序中这 2 个节点的计算复杂程度不同决定。根据计算，packet_source 所计算的结果放入输出信道的速率大约是 fragment 从输入信道取出数据速率的 3 倍。因此，这种划分方式是最优的。

6　结束语

本文根据多媒体和网络应用的特点，创建一个并行编程模型，该模型以分块思想描述程序，在核的数量越来越多的情况下，能有效提高处理器的利用率。当然，本文模型的研究只停留在框架设计阶段，要完成针对不同应用的处理是今后的研究方向。

参考文献

[1] Kenjiro T, Kenji K, Toshio E, et al. A Parallel Programming Model for Accommodating Dynamically Joining/Leaving Resources[C]// Proc. of ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. San Diego, USA: [s. n.], 2003.

[2] Jayanth G, Mendel R. Stream Programming on General-purpose Processors[C]//Proc. of the 38th Annual Int'l Symp. on Microarchitecture. Barcelona, Spain: [s. n.], 2005.

[3] Robert E, Richard S, Mycroft A. Task Partitioning for Multi-core Network Processors[C]//Proc. of Int'l Conf. on Compiler Construction. Edinburgh, UK: [s. n.], 2005.

[4] Jeffrey D, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters[C]//Proc. of the 6th Symposium on Operating Systems Design and Implementation. San Francisco, USA: [s. n.], 2004.

编辑　陈　文