第18卷 第4期
2008年4月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol.18 No.4
Apr. 2008

基于多核集群系统的并行编程模型的研究

胡晨骏，王晓蔚
（东南大学 计算机科学与工程学院，江苏 南京 210096）

摘 要：并行计算技术是计算机技术发展的重要方向之一。当前并行程序模型主要有消息传递模型和共享存储模型两种。随着处理器多核技术的发展，在一枚多核处理器中集成两个或多个完整的计算引擎（内核），并充分利用多核计算机的特性，发挥多核计算机的性能成为一个很重要的研究方向。介绍一种新的MPI实现机制，这种机制集成了共享存储模型和消息通信模型的优点，在节点内使用共享存储模型，在节点间使用消息传递模型，并且通过自动生成线程级的任务来获得更好的性能。

关键词：并行编程模型；消息传递模型；共享存储模型；MPI；线程
中图分类号：TP311　　**文献标识码**：A　　**文章编号**：1673－629X(2008)04－0070－04

Research of Parallel Programming Model Based on Multi－Core Cluster System

HU Chen-jun, WANG Xiao-wei
(School of Computer Science and Technology, Southeast University, Nanjing 210096, China)

Abstract: Parallel computing is one of the most important techniques of computing. Contemporary parallel programming model can be roughly divided into message－passing model and shared－memory model. With the development of multi－core processor, this integrated two or more intact computing engine (kernel) in one processor. There arouses a very important researching direction as how to effectively utilize the feature of the multi－core to achieve optimal performance. Introduce a new implementation of MPI, which could integrates the features of both the shared－memory model and the message－passing model, that is, use shared－memory model within one node while use message－passing model among nodes. And more over, better performance is achieved through automatic thread－level task spawning mechanism.

Key words: parallel programming model; message－passing model; shared－memory model; MPI; thread

0 引 言

当前，处理器技术发展很快，从超线程到现在的多核技术。多核技术是指在一枚处理器中集成两个或多个完整的计算引擎（内核），通过划分任务，使用多线程来充分利用多个执行内核，并可在特定的时间内执行更多任务。英特尔的技术总监拉特纳曾在2006年说，到2010年，单一处理器可能会集成有数十个内核，然而在2007年英特尔已经开发了有80个内核的处理器，这款处理器每秒能够执行一万亿次运算，虽然这只是个研究项目，但是它展示了处理器技术的一个发展方向。

为了更有效地利用多核处理器，就需要开发新的编程模型并要对现有的MPI进行优化[1]。现在，商品化的集群系统由多个共享存储的多核、SMP计算节点构成。它是一种共享存储与分布存储集成的混合结构，这就决定了在多核、SMP计算节点上适合做基于共享存储（线程）的程序设计，而在节点之间更适合采用消息传递的程序设计。如使用MPI和OpenMP[2]的混合编程模式[3]，或PThread[4]和MPI[5]的混合编程模式，这种混合编程模式会使本来已经十分复杂的并行算法设计任务变得更加困难，而且要求并行程序的设计者对这两种编程模型都很熟悉，这就极大地增加了并行程序设计的难度，提高了对并行程序设计者的要求。如何将这两者的优点很好地结合在一起，并且尽量避免混合编程模式所带来的并行程序开发的难度，将是一个很重要的研究方向。文中介绍了一种新的MPI设计思路，这一思路可以结合消息传递模型和共享存储模型，在节点间使用消息传递模型，在节点内由调度程序生成多个并行执行的线程，在线程间的数据通信可以通过共享内存的数据拷贝直接进行，从而

收稿日期：2007－07－11
作者简介：胡晨骏（1978－），男，江苏仪征人，硕士研究生，研究方向计算机体系结构；王晓蔚，副教授，研究方向为计算机体系结构。

充分利用了单一计算节点的性能,同时降低并行程序设计的难度。

1 并行程序设计模型

1.1 消息传递模型

消息传递模型的底层是一组处理器,每个处理器有自己的内存且只能直接访问本地的指令和数据。同时一个互联网络支持各处理器之间进行消息传递。消息传递模型程序开始时,用户将指定并发的进程数,通常在程序执行过程中,活动的进程数将保持不变。每个进程执行着同一个程序,但是,由于每一个进程都有一个唯一的ID号,在程序展开之后不同的进程可以执行不同的操作。一个进程或执行针对其局部变量的操作,或与其它进程进行通信,两个过程可交替进行,进程之间通过传递消息实现数据共享和进程同步。MPI是当今最为流行的用于并行编程的消息传递库标准。

1.2 共享存储模型

共享存储的程序设计模型大多是由PVP(parallel vector processor)和SMP平台提供,共享存储的程序大多是在特定的多处理器平台上用平台专用的语言写成。在这种模型上,数据处在单一地址空间,分为共享和私有两种,数据通信通过共享存储来完成。平台独立的共享存储并行程序设计的标准模型有Pthread和OpenMP。当前最重要的共享存储标准是OpenMP。

2 MPI新的实现机制

MPI是基于消息传递模型的一套并行开发库,它适用于基于分布式存储的集群系统,而现在的集群系统很多都是由SMP和多核处理器计算节点组成的,那么在并行程序设计时就要考虑节点内和节点间的两极并行:在节点间可以用基于消息传递模型的并行程序设计模型,如MPI;在节点内可以用共享存储的程序设计模型,如线程。

线程是程序流程中的单一控制流,同一进程的多个线程,可以共享同一进程中的全局地址空间,这样线程间的数据通信都是隐式的,可以直接通过共享内存间的数据拷贝来实现,所以线程间的数据通信的效率很高,而且可以有效地避免网络通信带来的延迟,同时基于线程的编程方式可以更好地发挥多核处理器的性能。

在由多核或SMP计算节点组成的集群的环境下自然是以MPI+线程的模型来获得更好的效率,但是这样做所带来的代价是提高了并行编程的复杂性,会给并行算法的设计者带来很大的负担,他们必须掌握消息传递模型和共享存储模型的并行程序设计的思想和方法,不仅要熟悉MPI的编程方法,还要掌握多线程或OpenMP的开发方法。现有的基于MPI开发的并行程序很多都不是MPI+线程组合的方式来设计开发的,如果重新设计就会造成资源的极大浪费。

2.1 MPI的新机制

以上原因促使我们去考虑这样的问题,即对现有的MPI模型进行改进,使得MPI能结合共享存储模型和消息传递模型的优点,让并行程序的设计者不需要过多地关心共享存储模型的开发机制,只要掌握消息传递模型的开发机制。利用MPI的库函数和调度程序去实现在多核的计算节点内自动地调度生成多个并行执行的线程,这样就可以充分利用多核计算节点的运算能力,避免消息传递模型和共享存储模型混合编程所带来的难度。

如图1所示,在新的MPI机制下,当调度运行一个MPI程序时,主节点将这个MPI程序分配到各个计算节点上去,各个计算节点的子调度程序根据传输过来的MPI程序的PE结构或ELF结构,调度安排在一个计算节点内生成多个该MPI程序的线程级的子任务,这些线程属于同一进程空间。并行程序的开发者无需去关心这些线程是如何创建以及它们之间是如何通信和同步的。这些工作都可以交给修改后的MPI的库函数和调度程序来完成,他们只要像以前那样去开发MPI程序,而不用去关心节点内的共享存储空间的线程编程问题。

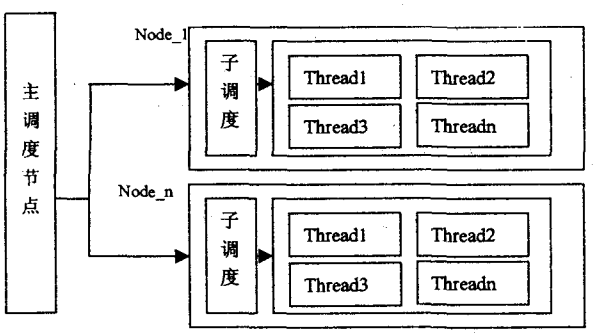

图1　新的调度机制

2.2 MPI新机制下的数据通信

在计算节点内,因为多个线程同属于同一个进程空间,所以在节点内的数据通信可以通过共享内存直接进行数据复制,要实现这一功能就需要对现有MPI通信机制进行修改。

首先对现有MPI的通信机制[5]作一个简单的介绍,在MPI中有通信域这样一个定义,一个通信域是一组可以相互通信的进程的集合,有关通信域的信息存储在类型为MPI_Comm的变量中,这些变量被称为通信器。这些通信器作为参数用在所有进行消息传递的MPI程序中,并唯一标识参与消息传递操作的所

有进程,并且每个进程可以属于不同的通信域。每个MPI的进程可以通过调用MPI_Comm_Rank函数来确定它在某个通信域中的标号。MPI在进行消息传递时是通过通信域和进程在通信域中的标号来决定通信的源和目标的,而在新机制下每个任务不再是以进程为单位,而是以线程为单位,为了实现这种同一节点内的线程间的数据的高效传输,可以由以下数据结构组成的列表来标识属于同一个节点内的所有任务线程。

```
Struct{
    Mpi_Comm comm_domain; //线程所属的通信域
    Int rank; //线程在通信域中的标号
    Thread_handle thread_id;//线程在操作系统中的编号
}
```

Comm_domain表示线程所属的通信域,Rank表示线程在通信域中的标号,thread_id表示线程在操作系统中的编号,该列表可以在MPI任务线程在调用MPI_Comm_Rank函数时在列表中插入一条记录,其中thread_id在Windows平台可以通过GetCurrentThreadId()函数来获得,在Unix和Linux平台中可以通过Pthread_self()来获得。当MPI并行程序调用MPI_Comm_Send函数发送数据时,首先通过目标的通信域和目标在通信域中的标号来遍历该列表,如果发现目标就在本计算节点的内部,就获得目标线程的编号,将要传送的数据直接通过共享内存的拷贝传送目标线程,如果没有发现,则表示目标不在本计算节点内,则仍然采用MPI原有的数据传送机制,如图2所示。这样在计算节点内部的线程间的数据传输就直接通过共享内存来完成,避免了由网络传输带来的延时。

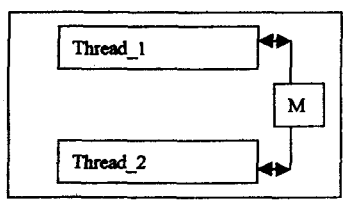

图2 节点内的数据通信

3 关键技术

如何将一个进程作为一个计算节点内的线程来运行,以及线程间的同步是这一新机制的关键技术,在Linux和Unix平台下,可执行的程序和链接库都是用ELF文件结构来描述的,在Windows平台下可执行程序是由PE文件结构来描述的。在这些文件中存储了运行程序所需要的运行代码,以及相应的数据(常量、变量)、资源等信息。下面主要介绍在Windows平台下如何通过导入一个可执行程序的PE文件结构,将一个进程作为一个线程来运行。Unix和Linux平台的实现原理与之基本接近,并且在文献[6]中已经做了详细的描述。

3.1 PE文件结构

在Windows平台中PE(Portable Executable)文件中存储了操作系统要运行一个可执行程序的所需要的信息,它主要由MS_Dos信息,NT信息,Section Headers和Section images组成,对PE文件结构的详细介绍大家可以参考文献[7]。

3.2 线程的创建过程

有了对PE文件结构的了解,下面就介绍如何将一个可执行程序的PE文件结构导入并作为一个线程运行的过程。

3.2.1 基本流程

图3为线程创建基本流程。

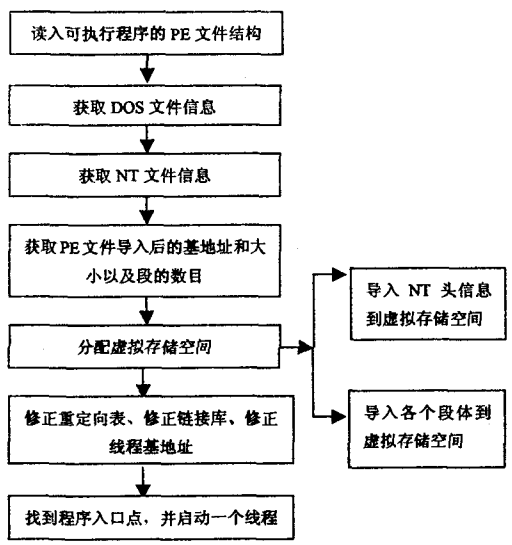

图3 线程创建流程

3.2.2 关键代码

```
//分别获取DOS信息和NT信息
pimage_dos_header=(PIMAGE_DOS_HEADER)pFileData;
pimage_nt_headers=(PIMAGE_NT_HEADERS)(pFileData+pimage_dos_header->e_lfanew);
//获取虚拟空间的基地址和大小
pImageBase=pimage_nt_headers->OptionalHeader.ImageBase;
dwImageSize=pimage_nt_headers->OptionalHeader.SizeOfImage;
//分配虚拟地址空间
pImageBase2=VirtualAlloc(pImageBase, dwImageSize);
//导入PE头信息
memcpy(pImageBase2,pFileData,
  pimage_nt_headers->OptionalHeader.SizeOfHeaders);
  pimage_section_header= pimage_nt_headers+sizeof(IM-
```

```
AGE_NT_HEADERS);
    //导入各个段信息
    for(i = 0;i < pimage_nt_headers - > FileHeader. NumberOf-
Sections;i + + )
    memcpy(pImageBase2 + pimage_section_header - > Virtual-
Address,
    pFileData + pimage_section_header - > PointerToRawData,
    pimage_section_header - > SizeOfRawData);
    //修正重定向信息
    if(pImageBase2! = pImageBase) reloc_fixup(pImageBase2);
    //修正链接库信息
    Dll_fixup(pImageBase2);
    //创建线程
    ThreadId = run_thread(AddressOfEntryPoint);
```

通过上面的步骤,可以将一个可执行程序的 PE 文件导入并作为一个线程来运行。计算节点内的子调度程序,根据主节点指定的节点内的线程数目,启动多个线程,这些线程在同一个进程空间中并行运行。

4 结束语

给出了将 MPI 程序调度成线程执行的方法,这种方法可以充分利用 SMP、多核计算节点的运算能力,而且可以避免消息传递模型和共享存储模型混合编程所带来的难度。在实际实现时还有一个主要的难点就是线程间的数据同步和任务同步的问题,这是以后要解决的主要问题。

参考文献:

[1] 陈文光.多核处理器的软件挑战[EB/OL].第三届全国高性能算法软件研究开发研讨会.2006.http://www.samss.org.cn/2006 - meeting/05.pdf.

[2] OpenMP Standards Board. OpenMP: a Proposed Industry Standard API for Shared Memory Programming [EB/OL]. 1997. http://www.openmp.org/openmp/mp - documents/paper/paper.Html.

[3] Quinn M J. MPI 与 OpenMP 并行程序设计(C 语言版)[M].陈文光,武永未,等译.北京:清华大学出版社,2001:212 - 245.

[4] IEEE. POSIX P1003.4a: Threads Extension for Portable Operating Systems[M]. Piscataway, NJ: IEEE Press, 1994.

[5] 都志辉.高性能计算之并行编程技术——MPI 并行程序设计[M].北京:清华大学出版社,2001.

[6] 查　礼,刘玉树.一种多线程计算程序的机群移植方法[J].计算机学报,2002,25(3):307 - 312.

[7] Pietrek M. An In - Depth Look into the Win32 Portable Executable File Format, part 1/2[J]. MSDN Magazine, 2002, 17(2):17 - 20.

(上接第 69 页)

对于上例,进行边界调整后为:

[我们][[要][加强]][[社会主义 精神文明 的][建设]]。

最后将短语边界依次两两配对,并根据短语结构信息进行匹配,若匹配成功标注上相应的短语标记。

下面给出边界匹配与标注算法的主要实现流程:

(1)读入一个边界调整后的汉语句子;

(2)读入该句一个边界;

(3)若该边界为右边界,则

自该边界向右检索第一个边界;

若所检索边界为右边界,则转上一步;

否则,调用短语边界分布信息,进行短语匹配,

若匹配成功,则标记该短语;

否则,若该右边界是首次匹配,则按相应错误情形处理;

(4)重复(2)、(3),循环处理该汉语句子的所有边界。

如:上例中最后标注的结果为:

np/(我们)vp/(要加强)np/(社会主义精神文明的建设)

4 结束语

对于短语的划分与标记,采用的是规则和统计相结合的方法。首先根据互信息来进行短语边界的划分,然后根据短语的组成特点来进行括号匹配和短语标记。此方法提高了短语的识别率和准确率,为进一步的汉语分析打下了很好的基础。

参考文献:

[1] 刘开瑛,郭炳炎.自然语言处理[M].北京:科学出版社,1991.

[2] 周　明,黄昌宁.面向语料库标注的汉语依存体系的探讨[J].中文信息学报,1994(3):35 - 52.

[3] 张国煊.基于语料库的汉语边界划分的研究—计算语言学进展与应用[M].北京:清华大学出版社,1995:94 - 99.

[4] 周　强.汉语短语的自动划分和标注[J].中文信息学报,1997(1):1 - 10.

[5] 周　强,俞士纹.汉语短语标注标注集的确定[J].中文信息学报,1996(4):1 - 11.

[6] Fano R M. Transmission of Information: A Statistical Theory of Communication[M]. [s.l.]: MIT Press, 1961.