一个基于非对称硬件结构的对称式文件系统*

武北虹 邢汉承 黄大海

(东南大学计算机科学与工程系,南京 210018)

摘要 本文介绍基于BJ-1并行计算机非对称硬件结构之上的一种对称式文件系统的设计思想和实现技术.该文件系统在不降低原有效率的前提下为用户提供了方便的界面,并通过文件系统中的管道机制实现了单元间多通道通讯.

关键词 操作系统,文件系统,disk cache,线程,通讯.

随着并行处理技术的日益发展,超高速的并行计算机在国内迅速发展起来.BJ-1是具有4亿次/s浮点运算速度的并行计算机,在硬件结构上采用了共享内存与消息传递相结合的方式.基于此硬件结构,其并行操作系统在实现上将主从式与对称式操作系统[1]结合起来,吸收了MACH[2],UMAX以及UNIX System V release 4.3等成功的操作系统中的先进思想,采用了以微内核(microkernel)为基础的模块结构式设计方法和基于处理机自调度的多线程(multithread)[3]技术,并针对BJ-1本身的结构特点,充分发挥了BJ-1提供的硬件资源,成为一个并行性能高且有利于系统扩充、扩展、移植的并行操作系统,经过长期的考机检验和用户的实际使用,更证实了该并行操作系统的高性能.

在与用户的接口方面,由于保持了与UNIX兼容的系统调用,总体上看,用户使用较为方便.但是,其文件系统是建立在非对称磁盘系统的硬件结构之上的非对称文件系统,这使得用户编程时要熟悉该非对称结构.在用户的实际使用中也发现了这一磁盘结构对用户的不透明性带来的不足之处,因此需要对此做出改进.本文论述的就是一个基于该非对称结构之上的对称式文件系统的设计思想和实现方法,以及通过与该文件系统的结合,对通讯进行的改进工作.

1 BJ-1体系结构及文件系统

如图1所示,BJ-1并行计算机在硬件上采用共享内存(SM)与消息传递(MP)相结合的方式.在本系统中,每个单元机包含多个高性能微处理器(i860),它们之间采用共享存储器的紧密耦合结构共同构成一个处理单元,各个处理单元通过中断互联网络进行同步及通

---

* 作者**武北虹**,女,1970年生,1995年博士毕业于东南大学,讲师,主要研究领域为并行操作系统.**邢汉承**,1938年生,教授,博士导师,主要研究领域为系统软件,人工智能,模式识别,图象处理.**黄大海**,1953年生,副教授,主要研究领域为并行处理,分布式系统.

本文通讯联系人:武北虹,厦门 361005,厦门大学计算机系

本文1994-10-25收到,1995-01-09定稿

讯，实现彼此间的消息传递.host 经过旁路高速总线与各处理单元进行数据传输及通讯控制.

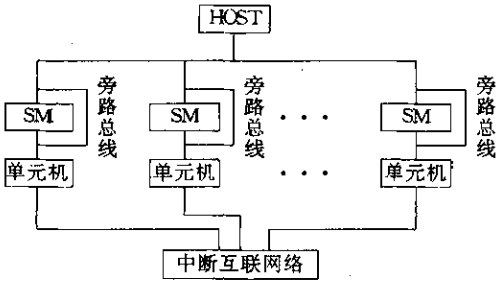

图1 BJ－1并行计算机的体系结构

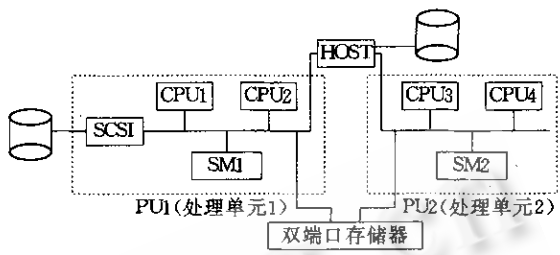

图2 BJ－1并行计算机目前的逻辑结构

限于实际条件，在 BJ－1 并行计算机目前的具体实现中，用了 2 个处理单元，每个处理单元含有 2 个 i860 处理器，2 个处理单元间采用双端口存储器作为消息传递的同步和通讯网络. 其逻辑结构如图 2 所示. 为了提高计算中文件存取速度，除了 host 上的磁盘外，在处理单元 1(PU1)上 CPU1 通过一个 SCSI 接口与一个大容量硬盘相连，这样计算中的信息和中间结果可以直接存储在该盘上，而不必再通过 AT 接口和 ISA 总线到 host 的磁盘，从而提高文件存取速度.

从支持文件系统的硬件结构来看，上述结构是不对称的. 处理单元 2(PU2)中的处理机无法访问 SCSI 接口及其磁盘，也无法访问与其共享内存 SM2 独立编址的 PU1 的共享内存 SM1. 基于该不对称结构，BJ－1 的文件系统是不对称的. 整个文件系统的数据结构及程序代码全部存放于 SM1 中，并由 PU1 进行驱动 SCSI 及读写盘块等底层操作，而在 PU2 上是看不到该文件系统的. 也就是说，用户程序只有在 PU1 上的子任务才能直接使用该文件系统进行 SCSI 文件访问，而 PU2 上的子任务要想使用 SCSI 盘上的数据必须先通过 PU1 来进行读操作，然后使用操作系统提供的 send，receive 等有关通讯的系统调用，通过双端口存储器将数据传给 PU2；反过来，PU2 上的子任务需要存盘的中间结果也要用 send，receive 先传给 PU1，然后再由 PU1 写入 SCSI 文件. 这就要求高层用户也要对这种不对称的硬件结构非常熟悉，因此用户在使用时很不方便.

2 改进后文件系统的设计思想

鉴于上述情况，需要对 BJ－1 文件系统进行改进. 改进后文件系统的设计目标是在操作系统一级抹平硬件结构的非对称性，使得该非对称性只存在于操作系统中，而对用户是透明的，用户看到的只是一个完全对称的文件系统界面，而不必看到其磁盘结构，用户可以在 PU2 的子任务中与在 PU1 的子任务中完全相同地使用该文件系统进行中间结果和信息的存储，且不降低原有的效率，不减少操作系统的其它功能.

一般的文件系统可分为 4 层. 第 1 层为与用户的接口层，解释用户的键盘命令或将提供给用户的系统调用转变为对底层模块的调用；第 2 层为从逻辑块到物理块的映射层，包括各种文件操作，目录管理以及空白块管理等；第 3 层为 disk cache 管理层，处理各种缓冲区操作；第 4 层为驱动层，负责对硬件接口的驱动以及真正的盘块读写. 从这种结构看，只有最底

层才与硬件设备有关,而上面 3 层仅牵涉到文件系统的内存数据结构,因此在 PU2 中只要能访问与 PU1 相同的内存数据结构,便可与 PU1 完全相同的处理前 3 层操作,这就是对该文件系统进行改进的基本出发点.

基于上述思想,将与文件系统有关的所有内存数据结构放在两个单元中间的双端口存储器中,以便两个单元能够同样地访问,各单元的文件系统代码段存放于各自的 SM 中.双端口存储器中除了系统线程表,文件系统安装表,系统打开文件表,内存 inode 表等固定大小的结构外,其余空间全部划分为 disk cache 块,这样两个单元在前 3 层的操作中是完全相同的,只有到了最底层,PU2 才请求 PU1 的帮助,由 PU1 来完成 SCSI 驱动和盘块读写后告知 PU2.由于在 disk cache 管理中采用了精确 LRU 算法[4],命中率很高,大部分操作均可在前 3 层完成,因此,虽然 PU2 的底层操作要比 PU1 的时间长,但从文件的总体操作来说,PU2 上的效率几乎与 PU1 相同.

3　改进后文件系统的主要实现技术

如前所述,两单元对于前 3 层的处理方法是相同的,本处仅对最底层的实现技术进行阐述.下面以文件系统的底层盘块读操作(bread)为例,说明在这两个单元中的不同实现过程.

(1) 当在 bread 中搜索 disk cache 队列未命中时,调用 scsirdwr 进行底层驱动.

(2) 如果此时是 PU2 的操作,转(5);否则(为 PU1),在 scsirdwr 中驱动 SCSI,向 SCSI 发读取相应盘块命令,然后返回到 bread 中,若此时 SCSI 读尚未完成,则将当前 thread 挂起,调度下一 thread 投入运行.

(3) 当 SCSI 读操作完成后,中断 PU1,CPU1 进行读后数据处理后唤醒第(2)步中挂起的 thread,送入 PU1 的就绪队列.

(4) 当再次调度到该 thread 时,继续在 bread 中执行.

(5) 如果调用 scsirdwr 的为 PU2,则在 scsirdwr 中并不进行真正的 SCSI 驱动,而是将双端口存储器中的某一标志置为 FSHELPME,并在当前 thread 中置好与读该块有关的信息,然后返回到 bread 中.此时检查该块,其读操作显然未完成,故将当前 thread 挂起,调度下一 thread 投入运行.

(6) 当 PU1 中发生时钟中断时,在时钟中断处理程序中若检查到 FSHELPME 标志,则唤醒 PU1 的系统线程 systemthread,由 systemthread 来依次检查 PU2 的所有 thread(两个单元的所有 thread 数据结构均放在双端口存储器中),当发现有 SCSI 读写请求的 thread 时,才真正进行 SCSI 操作,即 PU1 的 scsirdwr 过程((2)—(4)步).

(7) 在 PU1 执行到第 3 步中,当发现要唤醒的 thread 为 PU2 的 thread 时,PU1 无法将其挂入 PU2 的就绪队列(因为各单元的就绪队列在各自的 SM 中),只置一个 FWAKEUP 标志,在 PU2 的时钟中断处理程序里才将发现有 FWAKEUP 标志的 thread 挂回其就绪队列.

(8) 当 PU2 再次调度到该 thread 时,继续在 bread 中执行.

在上述 PU1 的 SCSI 驱动中,由于硬件上只能由 CPU1 来完成,故在操作系统实现中专门为 CPU1 设置了一个线程调度就绪队列,位于该队列上的就绪线程仅能由 CPU1 来完成.如果进行 SCSI 底层操作的处理器为 CPU2,则调度算法将当前线程变为就绪状态后送

入该CPU1的专用调度队列,当CPU1空闲时调度算法首先让它在其专用队列上选择一个线程投入运行,只有该队列为空时,才去其它就绪队列上调度,这样就保证了具有SCSI操作的线程的相对优先,减少了I/O瓶颈.

4 单元间通讯

分别位于两个单元上的thread之间是通过操作系统提供的系统调用send,receive来通讯的.在BJ-1并行计算机操作系统中,send和receive是利用对两单元间双端口存储器的读写实现的,双端口存储器中除了用于操作系统的一小部分数据区外,其余部分均用于通讯区(原来的方案).当要传送的数据量超过该通讯区大小时,挂起当前thread,释放处理机去调度其它thread,当对方接收数据后,再唤醒该thread继续传送下面的数据.

这种通讯为单通道数据传送方式,适合于大批量的数据传输.但在用户使用中,也有一些少量数据多次传输的情况,如每次循环地计算出一个数组的一行元素并传给对方.由于在这种通讯方式中,每次只能是单通道传输,尽管下一次的数据已算出而且本次传送的数据量并没有占满双端口存储器,也要等到对方将本次传送的数据拿走,才能进行下次传送,这使得这种情况下的通讯效率较低.

在利用上两节中的方法对文件系统改进后,显然该send和receive的功能被破坏(因为通讯区已被用于文件系统内存数据区).对此的解决方法是利用文件系统中的pipe机制,将单元间的通讯结合在改进后的文件系统中,不仅能恢复原来的send,receive功能,而且还实现了多通道数据传送方式,克服了原来单通道通讯的缺点.

这种通讯的基本思想是位于两个单元上的线程通过相同的管道号建立一条管道,发送消息的线程在该管道的一端将信息源源不断地送入其中,接收消息的线程则在另一端取得信息.虽然这种接收消息和发送消息是通过文件读写来实现的,但这种读写只在disk cache一层进行,并不进行真正的磁盘读写,这与UNIX中无名管道[4]有着共同的特点,即管道文件是暂时的.由于双端口存储器的大部分区域用作disk cache区,故这种管道通讯在本质上依然是对双端口存储器的读写.与UNIX中无名管道不同的是用户可以通过不同的管道号建立多条管道,从而实现多通道数据通讯.

进行单元间通讯的线程都必须先利用文件系统提供的系统调用pipe(key,direct)建立一条管道,其中key为管道号,只有管道号相同的线程才能通过该管道通讯,direct为消息传输方向(即用于消息发送还是接收).该文件系统对pipe调用的实现主要是为通讯的线程分配一项如图3所示的pipe结构.如果当前系统pipe结构表中尚未存在管道号为key的pipe结构,则分配图3的结构后返回p_rdfd或p_wrfd(根据direct的值);否则找到该pipe结构后直接返回p_rdfd或p_wrfd.这样用户就象用open操作得到一个普通文件的fd一样,可以同样地利用read和write通过该p_rdfd或p_wrfd对管道进行读写操作(即接收消息或发送消息).

pipe文件读写过程中的读指针和写指针(分别为p_rdfd和p_wrfd所指打开文件结构的f_offset)的变化与UNIX中相同,不同点在于为管道文件分配的disk cache块对应的设备均为PIPEDEV,这是一个虚设备,在分配内存inode或disk cache块时,并不真正地分配相应的物理盘块.淘汰算法只能淘汰用于普通设备的disk cache块,而不能淘汰用于

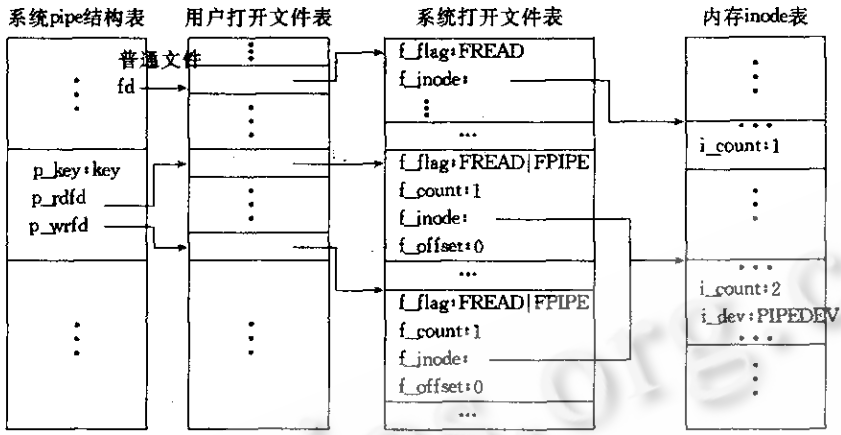

图3 pipe内部结构

PIPEDEV设备的 disk cache 块，这样一 disk cache 块一旦分配给一管道文件后，便不会挪为它用，直到使用该管道的线程全部通过 closepipe 关闭管道后，才将用于该管道的所有 disk cache 块从 PIPEDEV 设备队列中取下，挂入 NODEV 设备队列，以备它用. 在此过程中并没有象 UNIX 无名管道那样限制管道文件的大小，只要有可用的 disk cache 块便可分配给管道传输使用. 当分配不到时，便将当前线程挂起以等待其它管道使用完毕后释放其 diak cache 块.

这种机制使得当进行大批量单通道数据通讯时，所有 disk cache 块(几乎为整个双端口存储器)全部用于通讯，与原来的 send 及 receive 的功能和效果是等价的；同时这种机制也使得在一个少量通讯用不完 disk cache 块时，可以使用其它管道与该通讯同时进行其它数据的传输，即利用多通道来提高通讯效率.

5 结束语

BJ－1 并行计算机有关磁盘硬件结构的非对称性给用户在使用文件时带来很大不便. 本文提出了对其文件系统进行改进的方法，这种方法在操作系统一级将非对称的硬件结构透明化，为用户提供了一个便于使用的对称文件系统；同时利用 pipe 机制将单元间通讯与文件系统结合在一起，实现了多通道通讯，提高了通讯效率.

经过实际测试，PU1 上的文件传输速率仍为 2.5MB/S，与原来的文件系统效率相同，而且 PU2 上的文件传输速率也接近 2.5MB/S；两单元间的大批量数据通讯速率仍为 40MB/S，与原来的通讯效率相同，这就进一步证实了该改进方案在不影响原有效率的前提下，增强了功能，方便了用户. 今后将在用户的使用中不断完善.

参考文献

1 Goscinski A. Distributed operating system, the logic design. Addison Wesley, 1991.

2 Blank D L, Golub D B *et al*. Microkernel operating system architecture. Mach. J. Information Processing, 1991, 14 (4): 442～453.

3 Kai Hwang. Advanced computer architecture. New York, 1993.

4 尤晋元. UNIX操作系统教程. 西安:西北电讯工程学院出版社,1985.

A SYMMETRIC FILE SYSTEM BASED ON NONSYMMETRIC HARDWARE ARCHITECTURE

Wu Beihong Xing Hancheng Huang Dahai

(*Department of Computer Science and Engineering Southeast University Nanjing* 210018)

Abstract This paper introduces a symmetric file system based on the nonsymmetric hardware architecture of BJ—1 parallel computer. Its main design and implementation considerations are discussed in detail. Not reducing the old efficiency, this file system provides a convenient interface for users and implements the multiple channel communications between units by using the pipe method.

Key words Operating system, file system, disk cache, thread, communication.