第 6 卷第 11 期
1995 年 11 月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 6, No. 11
Nov. 1995

并行文件系统 PFS 的设计与分析*

武北虹　黄大海　邢汉承

(东南大学计算机科学与工程系，南京 210018)

摘要　并行计算机高速处理能力与低速 I/O 设备之间的矛盾目前已成为并行计算机系统中的主要问题之一. 因此必须研制高性能的并行文件系统. 本文介绍的 PFS 是为"八六三计划"中"曙光二号"并行计算机设计的并行文件系统. 该 PFS 分散 I/O 设备的管理到多个处理结点，使文件能交叉地分布在不同 I/O 结点所控制的物理设备上，以实现最大程度的并行访问. 本文阐述了 PFS 的基本设计、实现方法及其性能的粗略分析.

关键词　并行处理，并行文件系统，I/O 结点，目录管理，负载平衡.

随着科学技术的迅猛发展，在气象、石油、理论物理等许多领域中，人们对计算的速度和精度提出了更高的要求. 超高速的并行计算机已成为现代科学技术的制高点，在高科技和基础研究中都发挥着极重要的作用.

并行机操作系统是并行机研究的一个重要组成部分. 在这方面国外已做了许多工作，研制出如 MACH[1]、AMOEBA[2]等一些较成功的并行操作系统. 但它们的文件系统仍然是传统的单一文件系统，即仅由一个处理结点控制所有 I/O 设备，并负责对文件的读写. 在科学计算中，如油田油层分析等，经常要读写一些非常大的数据文件[3]. 当多个计算结点同时要求文件系统进行 I/O 服务时，就会因并行机的高速处理能力与低速 I/O 设备之间非常悬殊的速度差别而导致严重的 I/O"瓶颈"现象，整个系统的性能将明显下降. 因此文件系统的研制是整个并行操作系统中至关重要的部分. "曙光二号"并行计算机具有 25 亿/秒次运算速度，是"八六三计划"课题之一，并行文件系统 PFS 是其中的一个子课题. PFS 已在"曙光二号"模拟环境中运行一段时间并得到良好效果. 本文阐述其设计与实现方法以及基本性能分析.

设计 PFS 的基本出发点是在硬件的支持下，将磁盘管理分散到多个 I/O 处理结点上，并将它们作为一个逻辑上的大磁盘来处理. PFS 将逻辑文件存贮块交叉地分布在不同结点所控制的磁盘上，这样当多个计算结点同时进行 I/O 数据访问时，就可以有多个 I/O 结点并行地为之服务，从而大大地改善 I/O"瓶颈"问题.

* 本文 1994－03－04 收到，1994－06－02 定稿

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. **黄大海**，1953 年生，副教授，主要研究领域为并行处理，系统软件. **邢汉承**，1938 年生，教授，主要研究领域为人工智能，模式识别，系统软件.

本文通讯联系人：武北虹，厦门 361005，厦门大学计算机科学系

PFS设计目标是要为绝大多数的应用程序提供一个各I/O结点负载尽可能均衡、I/O操作并行度尽可能高的文件系统，使得I/O操作对并行机所提供的超高速的影响尽可能小. 与此同时，还应尽可能地方便用户，为用户提供一个良好的界面，使用户只看到一个逻辑上"超高速"的大磁盘.

1 系统结构

"曙光二号"并行机在硬件上采用mesh结构(图1)，其中用i860作为计算结点机，用386/486作为host及I/O结点机，I/O结点机的个数为1至16个，其中一个为master，其余的为slave.

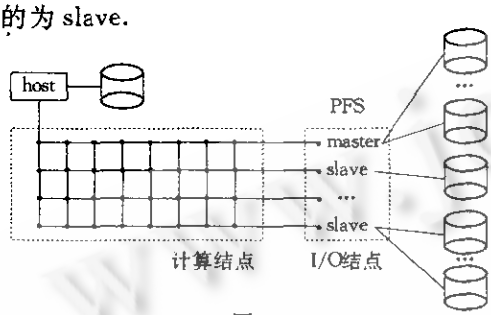

图1

PFS的设计包括3个部分：

1. host结点

host上装载UNIX操作系统，它将整个mesh网作为它的一个"外部设备"看待. host上的程序可通过访问"/dev/mesh/node_number"设备和相应结点通讯. PFS为host上用户提供了若干操作命令，如pcd，pls，pmkdir，prm，pcp，ptar等，它们的操作方式及功能与UNIX命令cd，ls，mkdir，rm，cp，tar等一样，但它们对PFS文件系统进行操作. host上用户程序也可通过popen，pclose，pread，pwrite等调用去读写PFS的文件. PFS以一个文件卷/PFS呈现在host终端用户和用户程序面前，同时还提供了一个pshell，在该pshell下，可使用UNIX原有的命令对/pfs下文件进行操作. pshell在发现操作参数含有/pfs时自动去调用相应的PFS操作命令.

2. 计算结点

在计算结点上装有一个简单的操作系统内核OS860，它负责管理内存和通讯工作. 并行文件系统PFS所提供的系统调用命令是以库的形式与用户程序连接，并随用户程序一起被加载到计算结点上运行. PFS提供了与UNIX完全一致的系统调用命令. 读写命令能确定物理存贮块的位置，直接向相应的I/O结点机发出读写物理块请求；含有文件路径名的命令，如open，creat等，交给I/O master结点机处理；其余命令则就地处理.

3. I/O master结点

在master结点上运行着PFS的核心程序master_server，它负责管理整个PFS的目录，根据一定的负载均衡算法分配整个PFS中的空白块以及为host或计算结点上程序提供读写存贮块的服务. master_server分为4层：第1层为接口层，它解释来自host或计算结点的I/O请求，调用下层功能块去完成该请求，再将执行结果返回给请求结点. 第2层为文件操作层，它负责查找、创建、修改或删除目录，打开或创建一个文件，为文件分配空白块或回收文件中的空白块. 它还负责对整个PFS的config进行管理，决定文件存贮块的分布，以保证I/O操作的负载平衡. 第3层为disk cache管理层，负责管理本结点机所带磁盘的存贮块在内存中的映象. PFS采用write_back技术，最大限度地减少实际I/O次数. 第4层为驱动层，负责对磁盘存贮块进行读写.

4. I/O slave 结点

在 slave 结点上装载 slave_server，它是 master_server 的子集，仅处理磁盘存贮块读写和空白块申请、释放这 4 种 I/O 请求. 它也分为四层，下两层与 master_server 完全一样，上两层少了目录管理、config 管理、全局空白块分配的负载均衡策略的功能.

2　主要实现技术及基本性能分析

PFS 设计包括硬盘分区、目录管理、负载平衡策略、disk cache 管理、基本块读写以及多线程(multithread)管理. 限于篇幅，我们仅对下列主要实现技术进行阐述与分析.

2.1　目录管理

PFS 基于这样的前提：并行机所要处理的文件通常都非常大，为 1MB～1GB，并以顺序访问为主[3]. 因此，我们将一个文件逻辑块大小取为 4K. 为了方便 cache 管理和存贮块读写，将目录块、文件说明块、文件索引块也取为 4K. PFS 的逻辑块号为 32bit，前 4 个 bit 为 I/O结点号，中间 4 个 bit 为结点内盘号，最后 24 个 bit 为盘内块号，因此 PFS 允许 1～16 个 I/O 结点，每个结点可带 1～16 个磁盘，每个盘的最大容量为 64GB.

目录和文件的数据结构如图 2 所示，一个目录包含 32byte 的目录说明和 127 个目录项，每个目录项 32byte，其中 28byte 存放文件或目录名，4byte 存放相应文件 head 或目录块号. 根目录放在 master 上启动盘的指定位置，启动时读入内存. 一个文件的数据结构包括文件 head、文件索引块和文件存贮块. 文件 head 包含 32byte 的文件说明和 1016 个块号，前 965 个块号直接存放文件存贮块，后 50 个为一级索引块块号，每个索引块中存放 1024 个文件存贮块号，最后一个为二级索引块号. 目录和文件 head 块存贮在 master 上的磁盘中，其它块分散在各个 I/O 结点所管理的磁盘上.

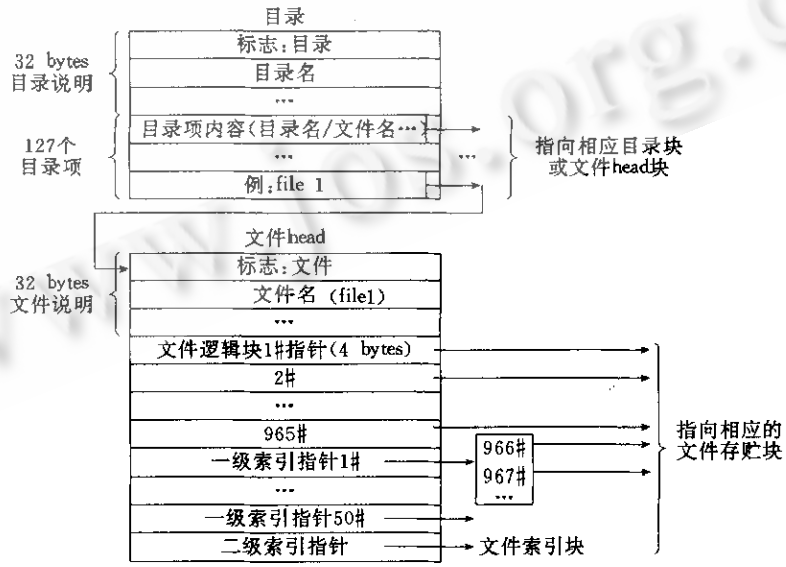

图2

当 host 或某个计算结点要求打开一文件时，其请求被送到 master 结点. master 从根

目录或相应进程当前目录开始逐层查找,在找到相应文件 head 后,将整个 head 传送给请求结点并在 master 中记录该请求结点已打开此文件.以后 host 或计算结点可根据 head 中块号直接与数据块或索引块所在的 I/O 结点进行通讯,读写相应数据存贮块.由于文件很大,打开操作与读写操作相比可忽略不计,因此只要文件存贮块能均匀地分散在各个 I/O 结点的磁盘上,就能实现文件的并行读写操作.

PFS 空白盘块管理用类似 UNIX 中的空白块管理方法[4],其实现过程和优点不再重述.

2.2 config 管理及负载分配策略的性能估算

config 表中登记着当前系统中各个 I/O 结点上磁盘的试用情况,包括总容量、已用块数和未用块数.当分配或回收空白块时都要对此表进行维护.config 表主要用于均衡分配空白块,使 I/O 结点操作达到平衡,从而实现最大程度的并行.

考虑到并行机的大多数应用程序都是顺序读写文件,因此 PFS 空白块分配策略是把文件的逻辑块依次循环地分配到各个 I/O 结点上,即第 1 块分到 I/O 1 上,…,第 n 块分到I/O n 上,第 n+1 块再分到 I/O 1 上,…,第 2n 块分到 I/O n 上,…,这样当对文件从头到尾顺序访问时,各 I/O 结点的并行度在理论上可达到最大.下面以对一大文件的顺序读操作为例,通过与单一文件系统比较,对此并行策略进行理论上的粗略分析.

设:读一盘块的时间为 td,与 I/O 结点通讯一次的时间为 tc,传输一块数据的时间为 Tc,内存操作时间与磁盘操作时间相比忽略不计.文件的大小为 m 块(m 一般为 256~256k),上述并行文件系统中,I/O 结点个数为 n($1\leqslant n\leqslant 16$),则 $m\gg n$,为讨论方便起见,设 m 为 n 的整数倍.

据上述假设,在单一文件系统中,无论对 m 块的读操作由多少个计算结点完成,其操作时间 $T1=tc+m\times td+Tc$.

与此相比,在上述并行文件系统中,设由 x 个计算结点同时交叉完成对 m 块的顺序读操作,则其操作时间

$$T2=\begin{cases} tc+m/n*td+Tc*\lceil n/x\rceil & (1\leqslant x\leqslant n)\\ tc+m/n*td+Tc & (x>n)\end{cases}$$

故 $tc+m/n*td+Tc\leqslant T2\leqslant tc+m/n*td+n*Tc$,而 $td\gg Tc$,$td\gg tc$,$m/n\geqslant n$,故 $T2\approx T1/n$.

由此可见,在理想情况下,此并行文件系统的并行度在理论上可达到最大.当然在应用中需根据实际情况对此策略进行适当调整,使其并行度尽可能大.

例如,由于 master 除了做存贮块存取的工作外,还要做许多额外的工作,因此需要减轻 master 的负载,如每从 slave 上分配两块时,仅从 master 上分配一块,这样就可使 master 的一半时间用于与 slave 并行进行数据读写,另一半时间用于目录管理等额外工作.

又如,为了进一步提高效率,将所有目录块以及所有文件 head 块全部放在 master 上,这样只要与 master 一次通讯就可找到文件 head.最后还应考虑各盘上的未用空间大小,使得空间多的盘多提供一些,空间少的盘少提供一些.

2.3 PFS 基本读写过程

在单机文件系统中,文件的读写过程是由一个进程从头做到底,而在 PFS 中却要由

host、计算结点与 I/O 结点间通过 mesh 网彼此多次通讯来共同完成. 图 3 及图 4(a)、(b)给出了 open 及 write 基本工作过程示意.

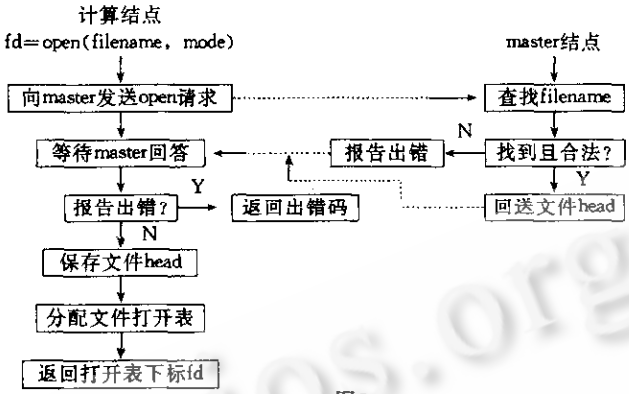

图3

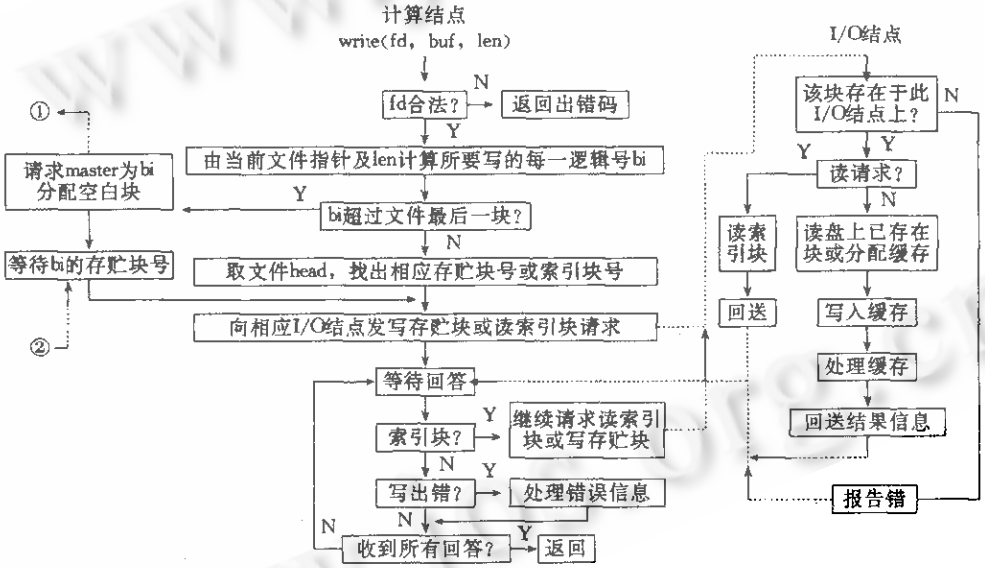

图4(a)

3　结束语

PFS 已在“曙光二号”的模拟环境中运行取得了良好效果. 当然此方案还有一些问题需做进一步探讨. 例如,master 结点的安全性问题将关系到整个 PFS 的安全性,可通过将目录的多个副本分布于不同 I/O 结点上或采用双磁盘映象等技术来提高其可靠性. 又如,若考虑多次非顺序的文件操作,需在计算结点上引入二级 cache,并进一步考虑 disk cache 多副本间的数据一致性问题.

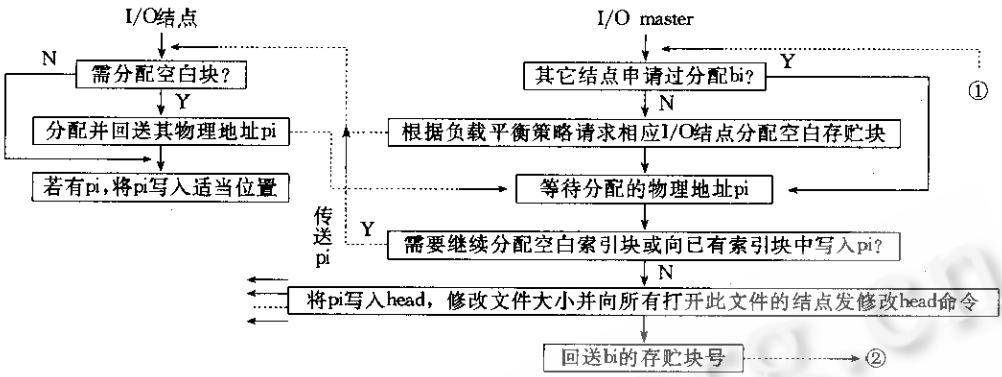

图4(b)

参考文献

1 Accetta *et al*. Mach: a new kernel foundation for UNIX development. Proc. Summer 1986. USENIX Conf., 1986. 93—112.

2 Mullender. Amoeba: a distributed operating system for 1990s. IEEE Trans. on Computer, 1990, **23(5)**: 44—53.

3 French J C, Pratt T W, Das M. Performance measurement of a parallel input/output system for the intel ipsc/2 Hypercube. ACM Performance Evaluation Review, 1991, **18(4)**: 178—187.

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.

DESIGN AND ANALYSIS OF THE PARALLEL FILE SYSTEM, PFS

Wu Beihong Huang Dahai Xing Hancheng

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

Abstract At present, the contradictory between the high speed process ability and the low speed I/O devices has become one of the main problems in multicomputers. So it is necessary to develop parallel file system of high performance. PFS is a parallel file system designed for a multicomputer, one of the 863—plans. Its goal is to decentralize the management of I/O devices into multiple process nodes so that files can be distributed into different physical devices controlled by different process nodes, and accessed as parallel as possible. This paper states the basic design, implementation methods and approximate performance analysis about PFS.

Key words Parallel processing, parallel file system, I/O node, directory management, load balance.