国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第15卷第1期 1993年1月 Vol. 15 No. 1

巨型机环境下UNIX逻辑盘的实现*

倪继坤

（电子计算机系）

摘 要 本文针对巨型机的体系结构特点，对UNIX逻辑设备概念进行了扩充，引入巨型机逻辑盘的概念，给出了一种逻辑盘的实现。逻辑盘的概念使得文件系统的大小独立于物理盘的容量、支持巨型文件或巨型文件系统。

关键词 操作系统，逻辑设备，UNIX，文件系统，逻辑盘

分类号 TP338.4

UNIX问世二十多年来，其影响深远，应用广泛，成效卓著。由于历史原因，UNIX是一个适合微、小、中、大型单机环境的分时操作系统。核心算法和提供的机制都是面向微、小、中、大型机的应用环境。因此，如果要在巨型机（SC）环境下移值UNIX，就必须结合SC的体系结构和应用特点，对UNIX中有关机制进行修改、增加和扩充、使其适合SC环境。

本文针对SC模型的体系结构，在UNIX逻辑设备概念基础上，引入了逻辑盘的概念，给出了一种适合SC（共享主存的多处理机系统）UNIX环境的逻辑盘实现技术，以支持巨型文件系统和改善I/O瓶颈。

1 SC模型的体系结构

SC是一个复合多处理机系统，它由主机子系统、磁盘子系统、磁带子系统和前端机子系统构成（图1）。

主机子系统由四个共享主存的同构型CPU所组成。每个CPU是一台包含多个向量和标量部件的处理机。

前端机子系统是SC的用户界面。用户经由前端机与系统交互。此外，前端机还负责用户作业，数据集的预处理和事后处理。

磁盘子系统是SC的海量存贮器。它包括磁盘数据通道、磁盘控制器、适配器和磁盘设备。控制器负责盘设备的寻道、驱动、错误处理，以及管理所属设备对于数据通道的分时使用。

磁带子系统由磁带处理部件、磁带控制器、磁带机和数据通道所组成。处理部件负责磁带路径的选择和数据缓冲，管理所属控制器对于数据通道的分时使用。而磁带机的驱动则由磁带控制器负责。

SC的体系结构和大、中、小型机的硬件结构相比，有很大的差异。这种差异反映了SC体系结构的特点。这些特点主要可概括如下[1]：

（1）海量存贮器容量相当大；

（2）扩展存贮器的引入。它的用途有：①用作磁盘I/O的缓冲存贮器；②用作快盘和存贮常用的磁

* 1991年9月6日收稿

盘文件；③用作交换设备；④用作用户作业的临时存贮器、存贮运行过程中常用的用户数据文件；

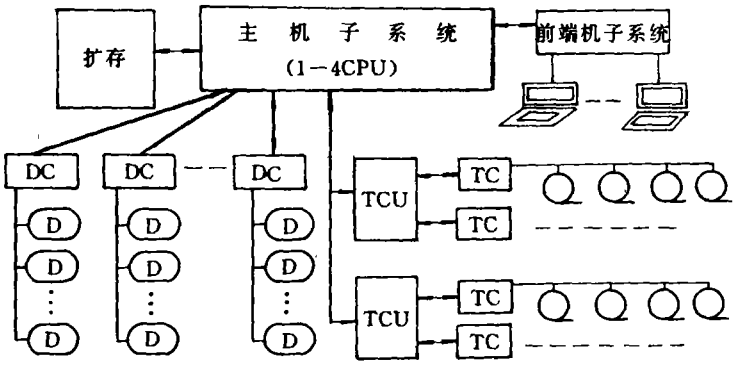

图 1 SC 体系结构

DC：磁盘控制器；TCU：磁带处理部件；D：磁盘设备；TC：磁带控制器

(3) 磁带处理部件和磁盘控制器承担了大部分工作，减轻了设备对 CPU 的干扰、增强了 CPU 与外设的并行处理能力。

2 逻辑盘的概念及其实现

2.1 问题提出

UNIX 系统中，文件系统是最为人们所感兴趣的。它具有很强的灵活的功能和精炼的内容。

虽然，UNIX 文件系统没有严格的层次结构，但是根据文件系统所属核心算法之间的调用关系。我们可以把其分为四个层次（图 2）。这样的划分基本上符合以下的原则：高层中算法调用低层中算法；而低层中算法一般不能调用高层中算法[2]。

核心在逻辑上处理文件系统（逻辑设备）而不是磁盘设备。核心把每个文件系统看作由逻辑设备号（主/次设备号）标识的逻辑设备[3]。实际上，逻辑设备是带有 UNIX 文件系统布局的一个磁盘区。磁盘驱动程序使用内部数据结构实现逻辑地址到物理地址的映射，核心与驱动程序间的接口由块设备转接表描述。

逻辑设备概念的引入使得核心算法独立于具体的物理设备。它保证了 UNIX 文件系统的简洁和清晰性，容易理解和修改，而且增强了系统的可移植性。

但是，由于逻辑设备的存贮实体是一个不跨越磁盘设备的连续存贮区，因而，存在以下的不足：

(1) 文件系统的大小完全受限于磁盘设备的容量；

(2) 不支持并发地访问一个逻辑设备中的数据，或一个设备上的多个逻辑设备中的数据。

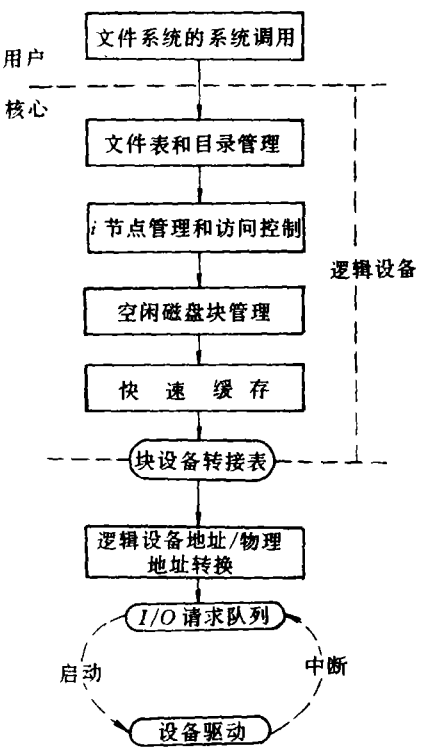

图 2 UNIX 文件系统层次结构

显然，这种适用于微、小、中、大型机环境的逻辑设备概念无法适合于SC应用。首先它不支持巨型文件系统或巨型文件；其次也不利于开发大容量、高速磁盘子系统的潜能，克服SC系统的I/O瓶颈。因此，如果要在SC环境下移植UNIX操作系统，就必须对逻辑设备进行修改、扩充和增强。

2.2 逻辑盘概念

在磁盘设备上，数据块的物理位置由柱面号、磁道号和扇区号说明。如果按磁盘物理地址顺序给设备上所有数据块编号，那么一个容量为n个数据块的磁盘空间可以看作为一个一维n元向量。

$$(b_1, b_2, b_3, \cdots, b_n) \tag{1}$$

其中b_i表示第i个数据块的地址。

对（1）式做一次任意划分，可得：

$$([b_1, b_2, \cdots, b_{l_1}], [b_{l_1+1}, \cdots, b_{l_1+l_2}], \cdots [b_{l_1+l_2+\cdots l_{m-1}} + \cdots, b_{l_1+l_2\cdots+l_m}]) \tag{2}$$

其中$\sum_{i=1}^{m} l_i = n$。

定义 一个磁盘上，由地址连续的磁盘块集构成的一个磁盘分区称作段（用符号S表示）。

根据定义，(2)式中的元素就是段，由此可将（2）式变换为一个一维m元向量、m值取决于划分。

$$(s_1, s_2, \cdots, s_m) \tag{3}$$

因此，任何一个磁盘设备的物理空间也可以看作为以段为元素的一个一维m元向量（称作盘物理空间向量）。

在R个磁盘设备的系统中，如果任意地从其它们的盘物理空间向量中抽取出h个元素，构成如下所示的一个向量（盘逻辑空间向量）：

$$\left(S^{a}_{d_1}, S^{a_2}_{d_2}, \cdots, S^{a_h}_{d_h}\right) \left| \begin{array}{ll} 1 \leqslant di \leqslant R & (1 \leqslant j \neq h) \\ a_j \neq a_i & \text{if} \quad d_j = d_i (j \neq i) \end{array} \right. \tag{4}$$

其中d_i是设备号，a_i是设备号为d_i的盘物理空间向量的元素下标。

盘逻辑空间向量表征了一个跨越设备的逻辑上连续，物理上部分连续的磁盘存贮空间。由此，我们可以给出巨型机逻辑盘的定义如下：

定义 一个具有UNIX文件系统布局的盘逻辑空间向量称作巨型机逻辑盘。

图3给出一个由三个物理盘构成的一个逻辑盘实例。

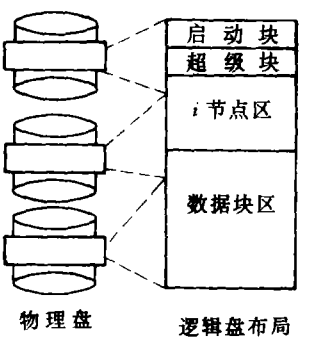

图3 UNIX逻辑盘

与逻辑设备相比较，逻辑盘在以下几方面对逻辑设备进行了扩充和增强。

（1）逻辑盘跨越物理盘，使得文件系统的大小独立于物理盘容量；

（2）在硬件支持的基础上，合理地分配逻辑盘空间，可提高文件系统的并发性；

（3）支持磁盘striping技术（磁盘成组交叉编址技术），提高单一数据流的磁盘传输率；

（4）增强了文件系统的可靠性。

2.3 逻辑盘的实现

为了高效地移植UNIX系统到SC环境，逻辑盘实现应该遵循两个原则：

（1）维持UNIX文件系统的用户界面、核心与磁盘驱动程序的接口；

（2）维持有关文件系统的核心算法的语法和语义，算法之间的调用关系。

巨型机UNIX的文件系统引入了逻辑盘概念之后，它的层次结构如图4所示。

2.3.1 逻辑盘的数据结构

在巨型机（SC）UNIX核心中，使用了三个主要数据结构描述逻辑盘。系统启动时，系统根据应用

需求建立数据结构中的内容。

（1）逻辑盘表

```
        struct ld_table_entry {
char ld_name [24];              / * logial disk name * /
int ld_blkno;                   / * toal blk count in LD * /
int slice_coun;                 / * total slice count in LD * /
struct slice_table_entry * p;   / * index to slice table * /
```

系统中定义的每个逻辑盘分别占用一项。磁盘驱动程序将使用设备特殊文件 i 节点中的次设备号检索逻辑盘表，确定要访问的逻辑盘。

（2）逻辑盘段表

逻辑盘段表描述了系统中定义的所有逻辑盘的段。对每个逻辑盘，分别在表中分配描述它的所有段的一组连续编号的段表项。逻辑盘表项中包含了该逻辑盘的总段数和起始段表项地址。其段表项描述如下：

```
        struct slice_table_entry {
int slice_size;                 / * slice size (blocks) * /
int start_addr;                 / * start cylinder no of slice
                                * /
struct disk_table_entry * p;    / * index to device table * /
```

（3）物理盘表

为了管理磁盘设备，巨型机 UNIX 为磁盘设备设置了物理盘表，每一个磁盘分别占用表中一项。它包含了要访问该盘的路径、设备状态、出错统计信息，以及设备 I/O 队列。显然，对于不同的体系结构，表项中内容会有所不同。对于本文中 SC 模型，其具体描述如下：

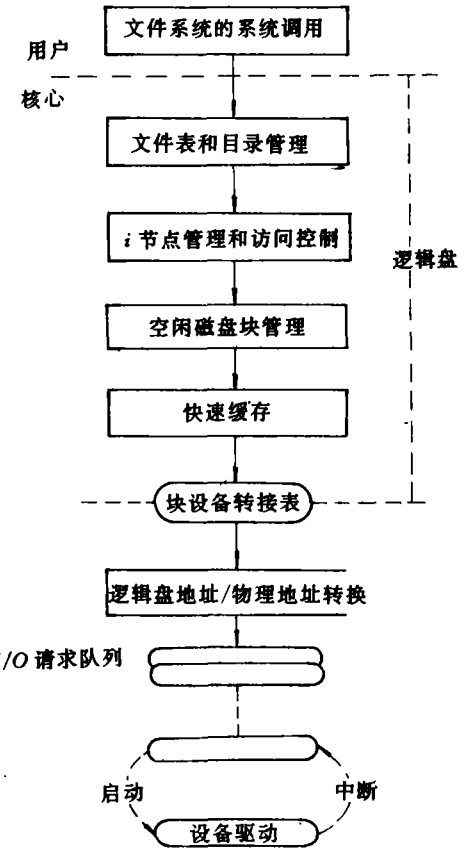

图 4　巨型机 UNIX 文件系统层次结构

```
    struct disk—table—entry {
    int d—flags;                          / * flags * /
    struct buf * b—forw;                  / * pointer to first buffer * /
    struct buf * b—back;                  / * pointer to last buffer * /
    struct buf * b—actf;                  / * pointer to first buffer in I/O queue * /
    struct buf * b—actl;                  / * pointer to last buffer in I/O queue * /
    char d—name;                          / * device name * /
    char d—active;                        / * device busy * /
    char d—errcnt;                        / * error count * /
    struct eblock * io—erec;              / * pointer to error count block * /
    int io—chan;                          / * number of channel attached by DC * /
    int io—adano;                         / * number of adaptor attached by device * /
    int io—dno;                           / * device number * /
    struct iostat * io—stp;               / * pointer to statiscal block of unit I/O * /
    time—t io—start;                      / * reserve for driver program * /
    int io—s1;                            / * reserve for driver program * /
    int io—s2;
```

上述数据结构由磁盘驱动程序使用。它们与核心算法使用的数据结构之间的关系如图 5 所描述。

2.3.2 磁盘驱动程序接口

核心与磁盘驱动程序的接口由块设备转接表描述（表1）。

图中，每种类型的设备占用一个表项，指示核心找到驱动程序的适当函数。表1仅给出了三种块设备类型的驱动程序接口。尽管存贮器是一种动态存贮设备，系统支持将扩存或主存贮器当作逻辑盘（文件系统）处理。这样，就可以将扩存定义为系统的“交换设备。”

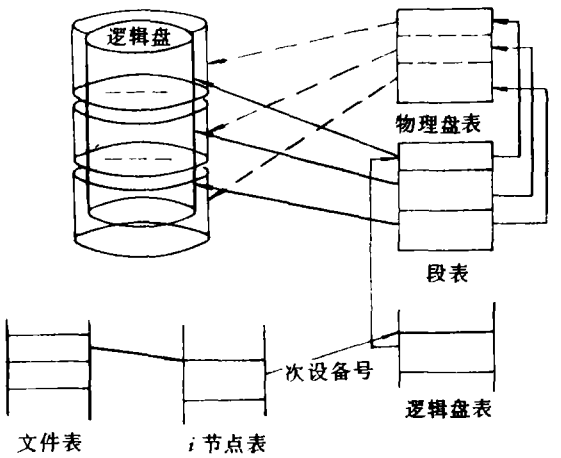

图5 数据结构间关系

图6给出了访问逻辑盘的控制流程。用户进程使用文件系统的系统调用进入核心，核心沿着从用户文件描述字到核心文件表及i节点的指针，依次执行rdwr，readi、bread核心函数[4]。它们从i节点中抽取出主/次设备号。用主设备号检索块设备转接表，而将次设备号作为参数（逻辑盘表的索引）调用表中定义的相应驱动程序函数。

ldstrategy函数（见图7）首先用次设备号检索逻辑盘表，获得逻辑盘的起始段表项地址。用缓冲区头中的逻辑块号和段表项中内容计算出磁盘块号（该逻辑盘中某个物理盘的块号），并将其放入缓冲区头中。最后，将该I/O请求挂入物理盘表项中的设备队列上。如果需要则启动设备。

由于盘逻辑空间向量构成的逻辑盘空间仅逻辑上连续，物理上部分连续（逻辑设备物理上全部连续），因此，一旦“段溢出（数据跨越段）”，这就需要将一个请求分解为多个子请求。子请求的个数取决于这次请求中溢出的次数。

考虑到文章篇幅，其它算法与一般UNIX中的完全相同，因此，本文中仅给出ldstrategy算法。

表1 块设备转接表

设　备	open	close	strategy
nulldev	nulldev	nulldev	nulldev
磁　盘	ldopen	ldclose	ldstrategy
扩　存	emopen	emclose	emstrategy
主　存	mmopen	mmclose	mmstrategy

```
algorithm ldstrategy
input bp;                    /* pointer to buffer header */
{
    if (i/0 count not multiple of 512 words)
return (error);
while (count != 0)

locate the correct slice;
calculate physical address in disk for
    logical block number in buffer header;
if (slice overflow)

    calculate the number of blocks transfered for
  this slice;
calculate the start logical block number for the next slice;
decrement the i/o count with the number fr words
```

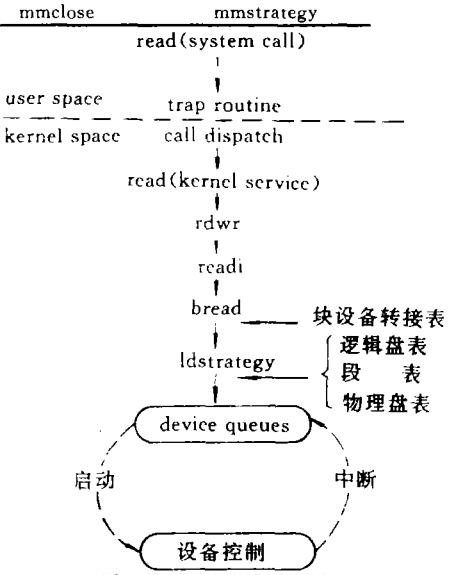

图6 read系统调用控制

```
for this slice;
calculate the buffer addr offset for the next slice;
mark as partial request;
increment the slice table index;
}
else {
decrement i/o count /* count=0 */
get the number of blocks for this i/o;
get the buffer addr for this i/0;
}
load the parameters into the buffer header;
link this part of request to device queue;
}
return;
}
```

图 7 ldstrategy 算法

3 结束语

逻辑盘概念及其实现打破了逻辑设备在物理盘之间空间的独立性。一个逻辑盘可以跨多个盘，使得UNIX 巨型文件系统的大小独立于物理盘的容量。

在巨型机环境下移植 UNIX，就文件系统方面还有许多技术问题有待于分析和研究。例如逻辑盘并发性研究，磁盘 striping 技术，逻辑盘空间的连续分配，cache 的可变长分配等。

遵循 3.2 节中的两个基本原则，对核心部分尽量少改动，是研究实现方法的主要立足点。作者深信，本文的实现方法并不是唯一的，会有不足之处。因此，作者希望同行们批评、指教。

在本文撰写过程中，陈立杰教授提出了许多有益的建议，在此表示衷心的感谢。

参 考 文 献

1 吴涛等。巨型机文件机制的研究。计算机外部设备，1991.

2 汤子赢等编. 计算机操作系统，西北电讯工程学院出版社，1984. 299－300

3 Bach M J. The Design of the UNIX Operating System. Prentice/Hall International, INC, 1987. 22－24

4 Prabhat K. Andleigh. UNIX[R] System Archetechure. Prentice/Hall International, INC. 1990: 131－167

The UNIX Logical Disk Implementation for Supercomputer Environment

Ni Jikun

(Department of Computer Science)

Abstract

The paper extends the concept of logical device of UNIX and introduces the supercomputer logical disk. The method for implementing the ligical disk is described according to the given architecture of supercomputer system then. The concept of logical disk makes it possible that the size of file system is independent of the capacity of the physical disk and that the logical disk supports the super file or super file system.

Key words operating system, logical device, UXIX, file system, logical disk

---

(上接第 43 页)

Abstract

Branch-delay is one of the RISC(Reduced Instruction Set Computer)technology characters. According to our simulation to scalar programs on Cray-1-Like supercomputer, about 10～20%CPU time is occupied by branch inst. In this paper, the implementation method of branch-delay technology in i860 microprocessor is deeply analyzed. The method of introducing branch-delay technology into Cray-1-Like prototype is studied. Simulation results and theoretical analysis show that reducing one cp execute time for each branch inst will increase system pertormance by 4～5 percent.

Key words supercomputer, branch Inst, Branch-delay, RISC