09 | 外存管理与文件系统

2019-05-14 许式伟

许式伟的架构课 进入课程 >

讲述：姚迪迈
时长 09:34 大小 8.77M

你好，我是七牛云许式伟。

在 “07 | 软件运行机制及内存管理”中，我们已经聊了内存管理，这一节我们聊聊外置存储设备的管理。

外存的分类

计算机有非常多样化的外置存储设备，比如：磁带、光盘、硬盘、U 盘、SSD 等等。外置存储设备的种类是不可穷尽的。随着科技的发展，新的存储设备会不断涌现，有着更低的单位能耗（存储量 / 每日能源消耗成本），更低的单位存储成本（存储量 / 可存储的时间 / 设备价格），或者更高的访问性能。

但不管这些存储设备内部如何存储数据的原理怎么变，改变的主要是质量，而不是它的功能。对操作系统来说，管理它们的方式是非常一致的。这些外置存储设备依据其功能特性不同，简单可以分为如下三类。

顺序读写型。如：磁带。

随机只读型。更准确说是单次完整写入多次读取型，也就是每次写数据都是整个存储介质一次性完整写入数据。如：光盘（含可擦写光盘）。

随机读写型。如：软盘、硬盘、U 盘、SSD 等等。

顺序读写型的外置存储（如磁带）我们日常并不常见，它的主要应用场景是归档，也就是数据备份。今天我们略过不提。

随机只读型的外置存储（如光盘）我们日常有较多应用，常见的应用场景是资料分发和归档。资料发布的内容很广泛，比如：软件、娱乐媒体包括电影、MTV、音乐等等。

随机读写型的外置存储最为常见，我们今天在所有“能够称得上叫电脑”的设备上，无论是 PC、笔记本、手机，还是手表、汽车，随处可见它们的身影。

外存的数据格式

外置存储和内存最大的区别是什么？

毫无疑问，外置存储是持久存储，它的目的是用来存储资料的。而内存是临时存储，虽然是存储，但是它实质上是为 CPU 的计算服务的。

那么，怎么让很多的软件进程同时使用这些外置存储设备，而不会乱呢？直接基于物理的存储地址进行读写肯定是行不通的，过上几个月你自己可能都不记得什么数据写到哪里了。

所以和内存管理不同，外部存储的管理，我们希望这些写到存储中的数据是“自描述”的某种数据格式，我们可以随时查看之前写了哪些内容，都什么时候写的。

这就是文件系统的来源。

文件系统把存储设备中的数据组织成为了一棵树。节点可以是目录（也叫“文件夹”），也可以是文件。

树的根节点为目录，我们叫根目录。如果是目录，那么它还可以有子节点，子节点同样可以是子目录或文件。文件则是叶节点，保存我们希望存储的资料。

每个节点，无论是目录还是文件，都有自己的名字、创建时间、最后编辑时间、最后访问时间等信息。有些文件系统还会提供最近一段时间的操作日志。这些信息有助于提醒我们有什么内容，以前都做过什么。

尽管几乎所有文件系统的接口是非常一致的，但文件系统的实现却有很多。对于随机只读型的外置存储（如光盘），常见的文件系统有如下这些。

文件系统名称	格式制定者	介绍
ISO-9660	国际标准化组织（ISO）	1985年颁布的通用光盘文件系统，被广泛支持。它只支持 8.3 格式的文件名，不支持长文件名。
UDF	国际标准化组织（ISO）	1996年制定的通用光盘文件系统。它支持多版本（用追加写实现），必要时可以将数据恢复到老版本。
Joliet	微软（Microsoft）	它是 ISO-9660 的扩展集。它兼容 ISO-9660 光盘文件系统，支持长文件名（最大64个字符）和中文文件名。

由于这类存储设备的写特征是批量写，一次把所有的数据写完，所以它的数据格式通常偏向于读优化（存储系统一般都有读写操作，所谓读优化是指在数据结构和算法设计时尽可能考虑让读操作更高效）。整个文件系统的元数据和文件数据都会非常紧凑，比如文件数据不必支持分块等等。

对于随机读写型的存储（如硬盘），常见的文件系统有如下这些。

文件系统名称	格式制定者	元数据组织	日志（journal）
FAT32	微软（Microsoft）	FAT	不支持
NTFS	微软（Microsoft）	B树	支持
HFS/HFS+	苹果（Apple）	B树	支持
EXT3/EXT4	开源（Linux）	H树（哈希B树）	支持
BTRFS	甲骨文（Oracle）	B树	不支持
ReiserFS	Namesys	B+树	支持
JFS2	IBM	B+树	支持
XFS	Silicon Graphicsis	B+树	支持

从文件系统格式的设计角度来说，它和架构关联性不大，更多的是数据结构与算法的问题；而且，不是基于内存的数据结构，而是基于外存的数据结构，这两者非常不同。

尽管文件系统的种类非常多，但是它们的设计思路其实基本相似。大部分现代文件系统都基于日志（journal）来改善文件系统的防灾难能力（比如突然断电，或不正常的 unmount 行为），基于 B 树或 B+ 树组织元数据。

古老的 DOS 引入的 FAT 文件系统（典型代表为 FAT32）是个例外，它直接把目录当作一个特殊的文件，里面依次列出了这个目录里的所有子节点的元信息。

这个结构简单是简单了，但是缺点非常明显，如果目录树深、目录里的子节点数量多，都会大幅降低文件系统的性能。

对于随机读写型的存储设备，操作系统往往还支持对其进行分区，尤其是在这个存储设备的容量非常大的情况下。分区是一个非常简单而容易理解的行为，本质上只是把一个存储设备模拟成多个存储设备来使用而已。

一般来说，拿到一块存储设备，我们往往**第一步是对其进行分区**（当然也可以省略这一步，把整个设备看做一个分区）。

第二步是对每个分区进行格式化。所谓格式化就是给这个分区生成文件系统的初始状态。格式化最重要的是标记分区的文件系统格式（用来告诉别人这个分区是数据是怎么组织的），并且生成文件系统的根目录。

第三步是把该分区挂载（mount）到操作系统管理的文件系统名字空间中。完成挂载后，该分区的文件系统管理程序就工作起来了，我们可以对这个文件系统进行目录和文件的读取、创建、删除、修改等操作。

外存的使用接口

怎么使用这些外置存储设备？

最简单的方式是用操作系统提供的命令行工具。例如：

目录相关：ls, mkdir, mv, cp, rmdir 等。

文件相关：cat, vi, mv, cp, rm 等。

当然，最原始的方式还是我们上一节介绍的 “系统调用”。但大部分编程语言对此都有相应的封装，例如 Go 语言中的相关功能如下所示。

目录相关：os.Mkdir, os.Rename, os.Remove 等。

文件相关：os.Open/Create/OpenFile, os.Rename, os.Remove 等。

有意思的是，在早期，操作系统试图将所有的输入输出设备的接口都统一以 “文件” 来抽象它。

最典型的代表就是标准输入（stdin）和标准输出（stdout）这两个虚拟的文件，分别代表了键盘和显示器。在 UNIX 系里面有个 “一切皆文件” 的口号，便由此而来。

但事实证明 UNIX 错了。输入输出设备太多样化了，所谓的 “一切皆文件” 不过是象牙塔式的理想。就拿键盘和显示器来说，图形界面时代到来，所谓标准输入和标准输出就被推翻了，编程接口产生颠覆性的变化。

有了文件系统的使用接口，进程就可以互不影响地去使用这些外置存储设备。除非这些进程要操作的文件或目录的路径产生冲突（所谓路径，是指从根目录到该节点的访问序列。例如路径 /a/b/c 是从根目录访问子目录 a，再访问子子目录 b，最后访问节点 c），一般情况下它们并不需要感知到其他进程的存在。

路径冲突是可以避免的，只要我们对路径取名进行一些基础的名字空间约定，但有时候也会故意利用这种路径的冲突，来实现进程间的通讯。

操作系统提供了一些冲突检查的机制。例如 “检查文件是否存在，不存在就创建它”，这个语义在保证原子性的前提下，就可以用于做进程间的互斥。例如，我们希望一个软件不要运行多个进程实例，就可以基于这个机制来实现。

虚拟内存的支持

前面我们在 “07 | 软件运行机制及内存管理” 一节中提到，在物理内存不足的时候，操作系统会利用外存把一些很久没有使用的内存页的数据，保存到外存以进行淘汰。

在 UNIX 系的操作系统中，操作系统为此分配了一个磁盘分区叫 swap 分区，专门用于内存页的保存和恢复。在 Windows 操作系统中则通过一个具有隐藏属性的 .swp 文件来实现。

在缺页发生比较频繁时，内存页的数据经常性发生保存和恢复，这会发生大量的磁盘 IO 操作，非常占用 CPU 时间，这时候我们通常能够非常明显感觉到计算机变得很慢。

在计算机变慢，并且计算机的硬盘灯不停闪烁的时候，我们基本可以确定是物理内存严重不足，不能满足运行中的软件的内存需要。

结语

回顾一下我们今天的内容。整体来说，外存管理从架构角度来说比较简单，复杂性主要集中在外存数据格式，也就是文件系统的设计上。

文件系统的实作非常多。如果你希望进一步研究某个文件系统的具体实现细节，我这里推荐一个由七牛云开源的 BPL 语言（Binary Processing Language，二进制处理语言）。地址如下：

https://github.com/qiniu/bpl

顾名思义，BPL 语言主要用于分析二进制数据格式。应用场景包括：文件格式分析（含磁盘分区格式，因为一个磁盘分区可以把它理解为一个大文件）、网络协议分析。

我们在后面的介绍文本处理相关的章节，还会专门拿出 BPL 语言进行讨论。

如果你对今天的内容有什么思考与解读，欢迎给我留言，我们一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。感谢你的收听，我们下期再见。

上一篇　08 | 操作系统内核与编程接口

下一篇　10 | 输入和输出设备: 交互的演进

精选留言 (31)

写留言

Enthusiasm
2019-05-14

7

关于外存管理，有个问题从之前就困扰我：磁盘的IO是由CPU完成的吗？但之前见到的说法是“CPU只能操作内存”。既然今天又提到了这个问题，文中提到“大量的磁盘 IO 操作，非常占用 CPU 时间”，那这两种说法是否矛盾？还想知道磁盘中的数据是怎么被加载到内存上来的呢？另外，更多的文章是说，“CPU的速度远远大于磁盘IO，CPU经常需要‘等待’磁盘IO”，这明显也是一种将CPU和外存割舍开的一种说法，而且按这种说...

展开

作者回复: 所有外设cpu都统一基于数据交换（io）的方式操作。cpu并不知道数据的含义，但是设备的使用方和设备知道。这种情况下你可以简单理解cpu只是一根网线，但是很重要的一点是它让设备使用方和设备可以交互。cpu并不负责磁盘io，但是它要等它结束以接收数据。这方面当然也有一些新技术出现改善这一点，可以想一想可能的优化路径，这里不表。

DaDo Wang
2019-05-15

4

想到了HDFS(Hadoop Distributed File System)，文中说的文件系统和HDFS应该不属于一个层吧？个人理解，HDFS应该全是更上层的应用软件层文件系统，它在外置存储的文件系统上，做了对分布式的文件进行管理的功能还请老师解答～😁

作者回复: 不是一个层面，后面服务端开发部分会讨论分布式文件系统

82
2019-05-14

4

多个进程去访问修改相同的外存地址文件时，谁来控制并发修改是操作系统还是外设驱动程序?
如果提高外存的访问速度是否可以减少缺页的处理时间，进而一定程度缓解卡顿的情况?

作者回复: 1、操作系统；2、对的，以前用机械硬盘作为swap，有时候慢比较显著；现在电脑基本上都用固态硬盘做swap，慢的感觉基本上完全消失了。

陈光
2019-05-14

4

老师，能否简单介绍一下基于内存的数据结构和基于外存的数据结构有何不同？我们平时所说的“数据结构和算法”是不是偏向于内存？另外，“路径冲突”是不是指多个进程同时访问同一个文件？

展开

作者回复: 1、外存的数据结构的特征是io是很费时的操作，所以外存数据结构+算法的优化方向是减少io次数，这个和内存很不一样。
2、平常数据结构大部分是内存；但一般数据结构书最后有几章会谈到外存数据结构+算法。
3、是的。

庆增 3
2019-05-14

再深入一点就好了，比如文件系统是如何恢复数据的。

展开

作者回复: 日志里面记录的是元数据的变更历史，所以可以恢复的是元数据破坏。如果文件内容坏了就没办法了

开放(深度... 2
2019-05-27

太简单了感觉，基本没有怎么说清楚，第一硬盘存储其中一个文件他是怎么存放的数据块和元数据是怎么结构，inode是什么，还有数据和元数据的索引表，甚至作为文件系统ext3的多层表索引和12个直接链接，一个单层，和多层索引等等都没说，文件系统的整体架构，特别是到底怎么优化等等也没说，还有虚拟内存，具体程序段怎么映射到物理内存，空余内存怎么管理，虚链表，对应的两难性能问题怎么解决?

展开

作者回复: 文件系统主要我觉得从架构上理解比较容易，所以没有特别交代太多。可能后面 review 的时候会适当重构一下这一篇。

慕-小坏 2
2019-05-15

请问，分区的格式化是由操作系统的文件系统管理程序完成的吗？比如Linux 支持ext3和ext4两种格式那么操作系统会内置两种文件系统管理程序吗？文件系统管理程序从某种角度可以理解为是一种“驱动”吗？谢谢！

展开

作者回复: 分区格式化其实一个普通的用户态应用程序就可以完成。文件系统管理程序不是驱动程序。

lckfa李钊

2019-05-15

2

看回复 ,很多人和我一样,对"Unix的一切皆是文件不是最佳实践"这个论点,有困惑.如果在图形界面时代的一切皆文件这样的架构设计不再适宜,为何类unix的设计者不改变这种设计,或者说,从架构的角度说,这样的设计定型了,是不是就不好改了?本课的主题其实是外设的统一接口是文件系统,那么把外设都当做文件进行抽象化处理,不出很合适么?

展开

作者回复: 这种统一给我的感觉就像很多面向对象的类库，所有的类统一从Object类继承一样，是一个过度设计。

ฆ醉里挑灯...

2019-05-14

2

缺页产生是因为长时间没有用的内存地址淘汰所以需要写入swap中进行保存，频繁的产生是说很多内存地址长时间没有用么？那这是怎么产生的呀

作者回复: 频繁缺页说明活跃的内存页超过物理内存大小，导致一个页刚刚换进来又要换出去，就比较折腾

靠人品去赢

2019-05-14

2

这个外存的数据格式NTFS我也是装系统的时候见到过，没有去考虑过这是一个什么东西。这个格式是有日志的，想问一下数据恢复可不可以借助日志，可以的话为什么数据恢复，机械盘可以固态盘就很大概率凉凉?

展开

作者回复: 固态盘为何凉凉没有了解过

大糖果
2019-05-14

2

老师好，有个问题，就是关于Windows自带的文件搜索，我们都知道那东西很慢，但是有一款everything的软件却可以做到很快，微软的技术是不用质疑的，他们为什么不把这个文件搜索做快点呢？还是这样的软件会有别的损耗？

展开

作者回复: 确实不一样，一个有建立搜索的索引，有额外的存储成本；一个是遍历（挨家挨户问过去）。

宝宝疯
2019-05-14

2

简单明了，如果能再深入些就更好了👍

展开

作者回复: 希望增加什么方面的内容?

cnanlnanc
2019-05-20

1

迫不及待地想听听有关架构重构的知识，毕竟这些才是几乎每个程序员每天都会面临的问题吧

作者回复: 重构的确是个有意思的话题，后面我们会谈到重构的一些常见思路

瞧，这个人
2019-05-17

1

怎么越来越像计算机组成原理了。希望更多的软件架构知识

展开

作者回复: 第一章会更偏基础体系，结合体系谈架构

Ryan
2019-05-16

1

对外置存储进行分区和格式化，是操作系统落实的吗？

展开

作者回复: 对

傲娇的小宝
2019-05-15

1

因为很多东西没学过，或者没这么深入，感觉自己要学习的还很多很多。

展开

作者回复: 能够在头脑中把整体串起来就可以了，然后对某些东西有兴趣想进一步了解的，后面可以继续研究

刘文坛
2019-05-14

1

linux把键盘、显示器看做设备文件，虽然不使用stdin stdout，仍然是以文件方式来操作硬件设备，还是没理解“一切皆文件”为什么很多时候不是最佳实践，或者说“一切皆文件”有什么显著弊端？

展开

作者回复: 底层大家都是io这肯定没错，但是我理解“一切皆文件”肯定是指使用范式上的，不然没有意义

史鹏飞
2019-05-14

1

许老师，共享内存的机制，是在内存上开辟的空间吗？服务器断电重启共享内存的数据是不是就丢失了？

作者回复: 是的

Dimple
2019-05-14

1

06-09这四篇看完，又加深了很多认识，感谢许老师的课程

展开

刘文坛
2019-05-14

1

老师您好，对于“一切皆文件”很多时候不是最佳实践，能举个例子吗？不是很理解

作者回复: 键盘和显示器就是很好的例子。stdin和stdout虽然能够用，但是实际上除了古老的命令行程序谁也不用他们。