【底层原理】存储数据包的一生（下）

2018-01-19 Yikun 码农有道

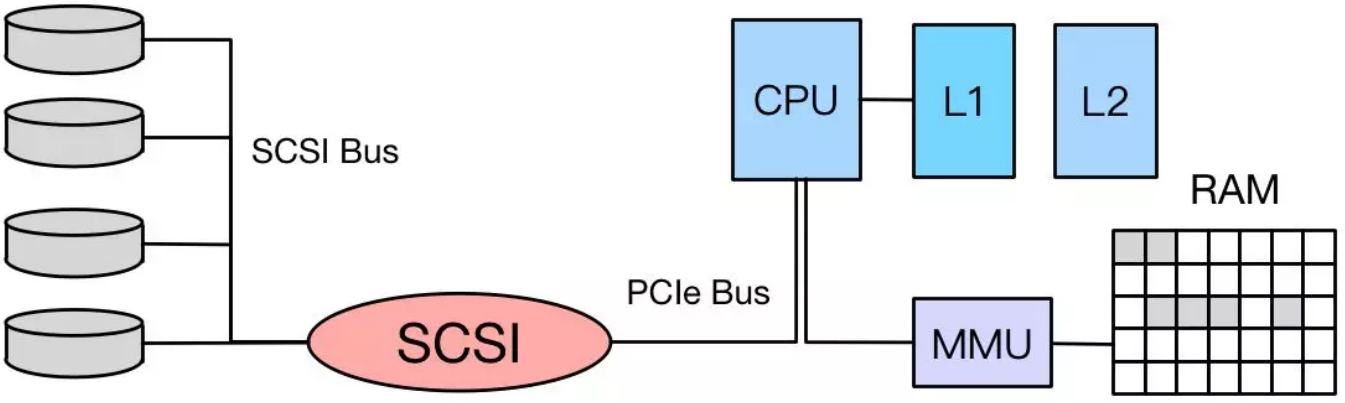
本文来自热心读者投稿，原文链接：http://yikun.github.io/

在前文存储数据包的一生（上）（请戳我）中，主要从整体和Application的角度总结了存储系统。本文将继续从存储介质和文件系统角度来了解存储系统。

Storage角度

块存储的IO流

CPU和内存通过PCI总线（目前通常是PCIe）与存储进行连接，应用会向存储请求一段数据。系统会将数据转换成地址和位置信息，并转换成某种协议的形式。



CPU的指令需要转换或者说是进行适配，以便能够与存储设备进行“交流”，比如SCSI、IDE/ATA

SCSI系统

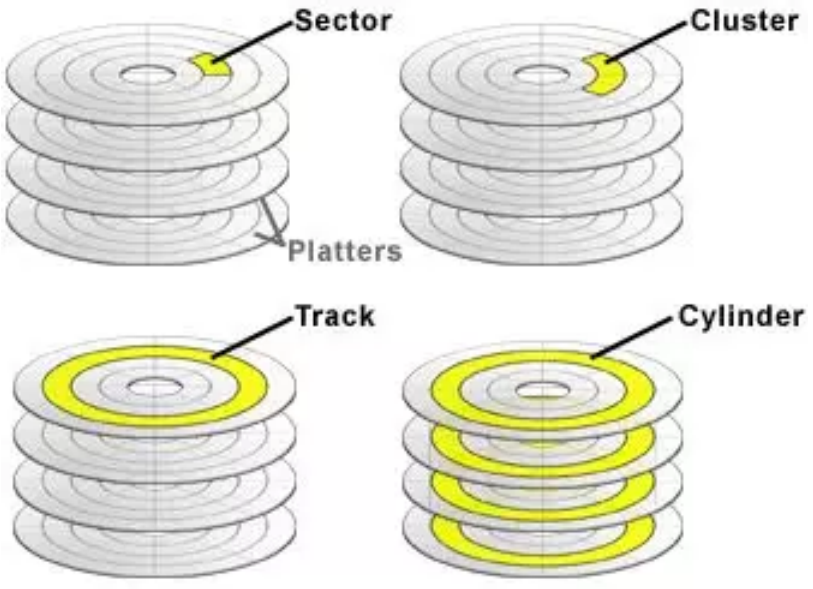
SCSI是一个很普遍、且向后兼容的，通常主要包含的组件有：

Initiator：一些服务用它发起请求，可以视为登陆或者适配的一部分；

Target：物理存储设备，可以是单个磁盘或者磁阵；

Service Delivery Subsystem：Initiator和Target之间的通信（通常是网线）

块设备

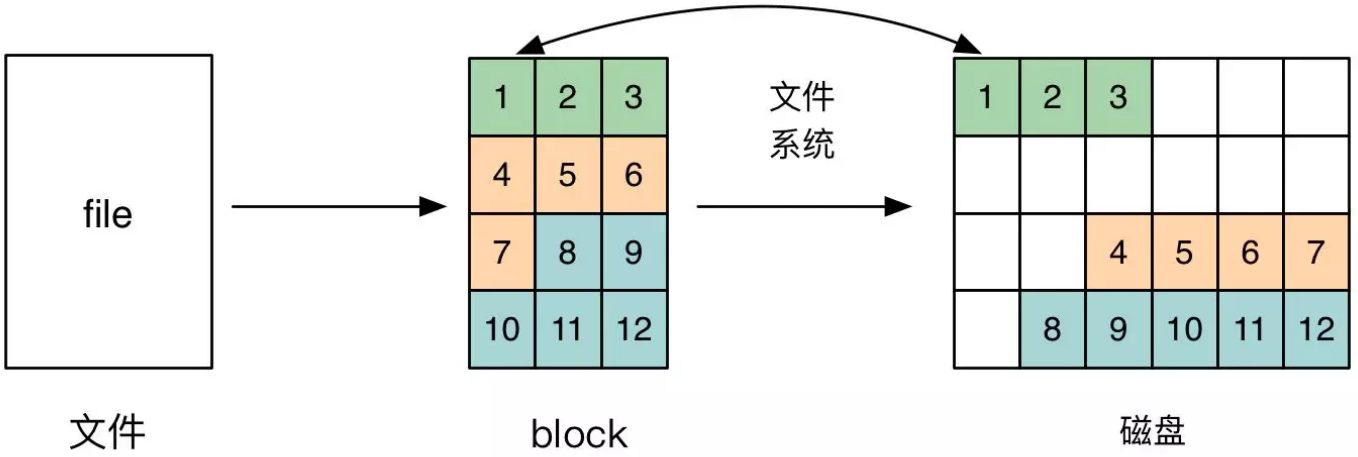


Block：块是存储介质上的物理或者逻辑单元，也是文件系统或者磁盘写入的最小单元，所有的存储（包括文件存储、对象存储）最终都是需要与block进行对话。

磁盘包括了盘面（Platter）、磁道（Track）、扇区（Sector）。盘面是一个圆，磁道是一个环，扇区是环上的一段，数据的位置影响性能。块由扇区组成，每一个块也有一个唯一的号码，文件系统的一切操作都是由对blocks操作构成。

文件系统

对于应用来说，他们看到的都是文件；而对于存储来说，看到是块。因此需要某种方式将他们联系起来。对于文件来说，应用看到的是一个连续的“空间”，然而实际上，文件是由很多块组成的，而这些块就是磁盘上的块，这些块分布在磁盘的不同区域。

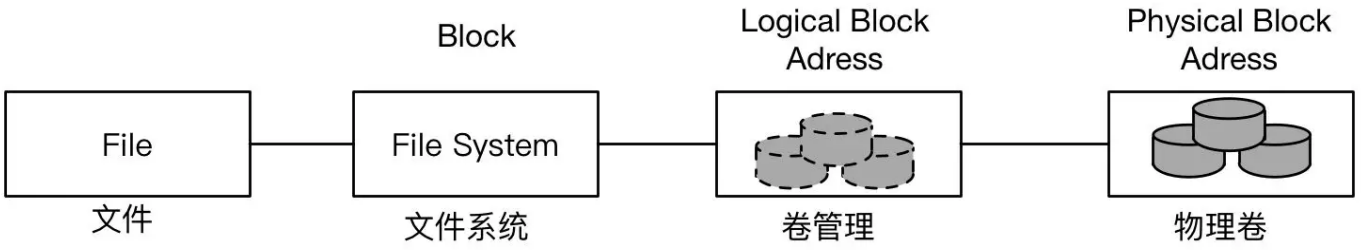


在操作系统中，其内核中会有一个文件系统，文件系统维护着在磁盘上的文件名，文件系统知道这些文件与磁盘上块位置的对应关系，同时还需要管理着磁盘空间、权限、所属、加密、文件缓存等等。

驱动控制：硬盘被驱动控制来操控，接收文件系统的通过某种协议（比如SCSI）下发的一些I/O命令

卷管理：文件系统和设备驱动中间存在着卷管理，卷管理负责抽象出一层“假的”磁盘供操作系统使用.

文件系统和驱动



因此，对于文件系统来说，需要将应用程序所看到的“虚拟地址”翻译到真正的设备地址。例如访问一个文件时，文件系统会先会先在找到对应的逻辑块地址（Logical Block Adress, LBA），然后通过scsi系统进一步访问，对应到磁盘上的物理块地址。

inode

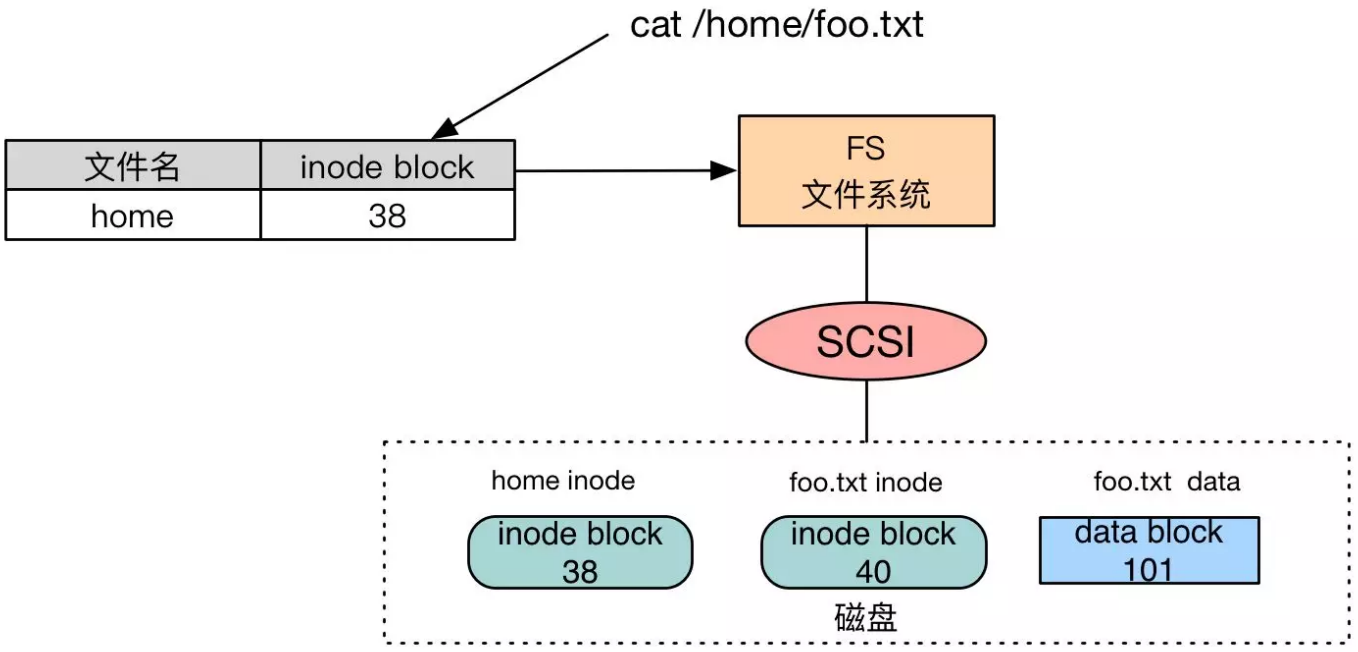
Inode是文件的原数据，用来记录文件的所述、权限、block信息等。一个Inode会对应一个文件或目录，通过inode就可以将文件与某些block对应起来。

也就是说通过inode信息，就可以完全的找到一个文件，包括这个文件所对应的data block。

对于目录而言，data block包含了目录的内容，即该目录下文件及其这些文件对应inode的列表。

对于文件而言，data block则包含了文件的实际内容。

举个例子



当执行`cat /home/foo.txt`命令时，会对foo.txt文件进行访问，经历了以下步骤：

1：获取”/“内容；

2：查找”home”的node信息（inode 38）；

3：获取”home”目录内容；

4：查找”foo.txt”的inode信息（inode 40）；

5：根据inode获得block numbers；

6：文件系统将block number转换为真正的block number（data block 101）；

7：通过SCSI Controller读取block内容。

至此，我们可以看到存储数据是这样走自己的路的：

1：应用程序说我要获取/home/foo.txt的内容

2：操作系统依次检查L1、L2、RAM中是否存在，若均不存在则在文件系统中查找

3：文件系统先检查Unified Buffer Cache中是否存在该文件，如果不存在则去目录中查找“home”文件

4：**(访问磁盘)**读取”home”目录的内容，并把inode信息放到buffer cache以便下次访问，最后根据目录内容查询到foo.txt的inode

5：**(访问磁盘)**读取foo.txt的inode，获取到data block号；

6：**(访问磁盘)**文件系统读取data block，最终将数据返回给应用，并且把数据存储在L1、L2、RAM和UBC中，加速下次访问。

我们可以看到，访问/home/foo.txt有3次磁盘访问：第1次是读取home的inode，获取foo.txt的inode号；第2次是读取foo.txt的inode，获取data block号；第3次是读取data block，获取真正的内容

然而，我们对于机械硬盘来说，每次需要磁盘转到正确的地址才能访问到内容，尤其是这些data block若未按顺序存储，就需要“下一圈”的时候再访问，这样会很耗时，也就是说访问磁盘的时间和数据在磁盘上的位置非常相关。所以Flash技术(如flash-based SSD)就腾空出世了，可以做到任意数据的随机访问，就大大减少了数据访问时间。

推荐阅读：
人人都可以做深度学习应用：入门篇（下）

深入理解系统中log机制（下）

【C++札记】了解 typename 的双重意义

【C++札记】C++对象模型之内存布局（2）

【C++札记】C++对象模型之内存布局（1）

专注服务器后台技术栈知识总结分享

欢迎关注交流共同进步

码农有道 coding

码农有道，为您提供通俗易懂的技术文章，让技术变的更简单！