# 9.5 块设备驱动程序


西安邮电大学

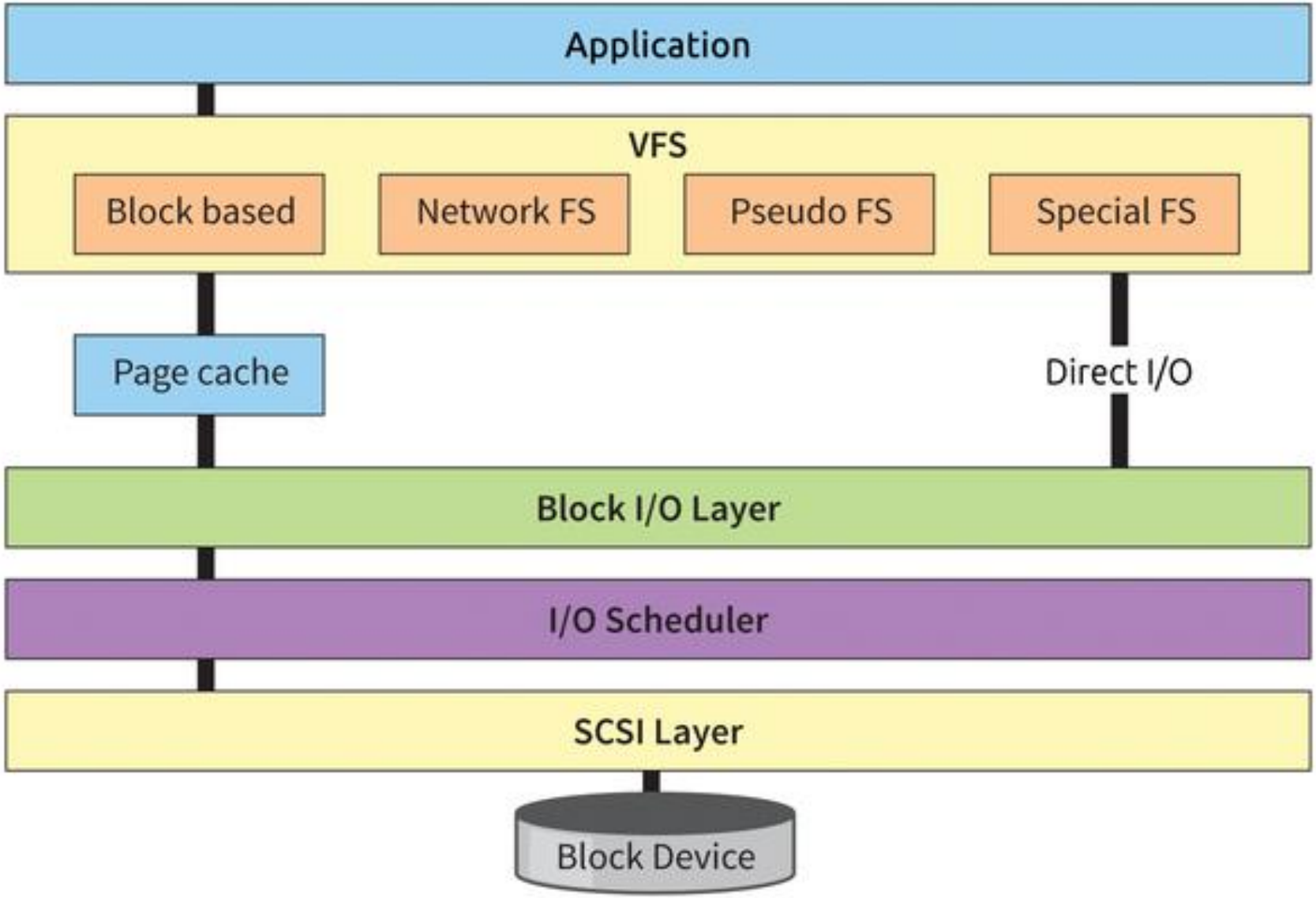
# 块驱动程序

块驱动程序提供了对面向块的设备的访问，这种设备以随机访问的方式传输数据，并且数据总是具有固定大小的块。

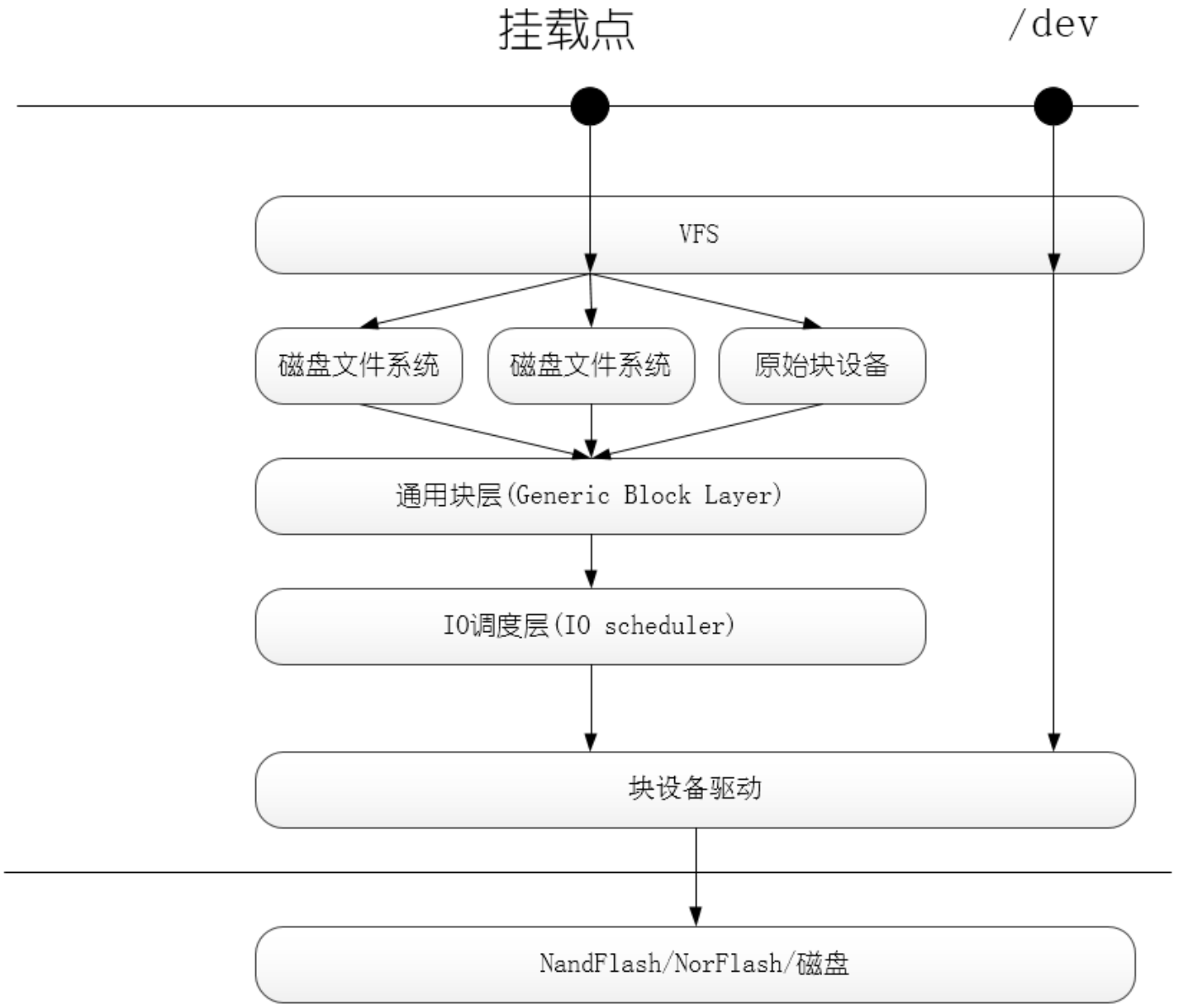
块设备和字符设备的区别：

- 块设备上可以mount文件系统，而字符设备是不可以的；
- 数据经过块设备相比操作字符设备需要多经历一个数据缓冲层（buffer cache ）机制。

# 块驱动程序


Application
VFS
Block based
Network FS
Pseudo FS
Special FS
Page cache
Direct I/O
Block I/O Layer
I/O Scheduler
SCSI Layer
Block Device

# 块驱动模型


挂载点
/dev
VFS
磁盘文件系统
磁盘文件系统
原始块设备
通用块层(Generic Block Layer)
IO调度层(IO scheduler)
块设备驱动
NandFlash/NorFlash/磁盘

# 块驱动模型

如图是Linux中的块设备模型示意图，应用层程序有两种方式访问一个块设备：通过/dev目录和文件系统挂载点，前者和字符设备一样，通常用于配置，后者就是我们mount之后通过文件系统直接访问一个块设备了。

- 1. read()系统调用最终会调用一个适当的VFS函数(read()-->sys_read()-->vfs_read())，将文件描述符fd和文件内的偏移量offset传递给它。
- 2. VFS会判断这个系统调用的处理方式，如果访问的内容已经被缓存在缓冲区中，就直接访问，否则从磁盘中读取。
- 3. 为了从物理磁盘中读取，内核依赖映射层mapping layer，即上图中的磁盘文件系统。

# 块驱动模型

1）确定该文件所在文件系统的块的大小，并根据文件块的大小计算所请求数据的长度。本质上，文件被拆成很多块，因此内核需要确定请求数据所在的块

2）映射层调用一个具体的文件系统的函数，这个层的函数会访问文件的磁盘节点，然后根据逻辑块号确定所请求数据在磁盘上的位置。

4. 内核利用通用块层(generic block layer)启动IO操作来传达所请求的数据，通常，一个IO操作只针对磁盘上一组连续的块。

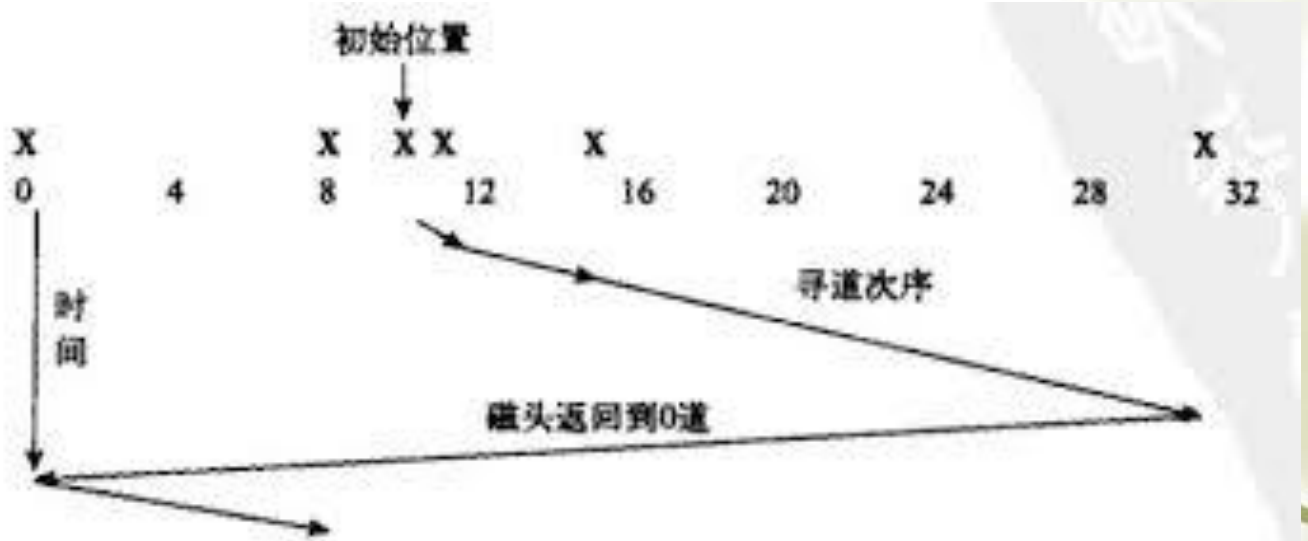
5. IO调度程序根据预先定义的内核策略将待处理的IO进行重排和合并

6. 块设备驱动程序向磁盘控制器硬件接口发送适当的指令，进行实际的数据操作

# I/O调度

I/O调度器的总体目标是希望让磁头能够总是往一个方向移动，移动到底了再往反方向走，这恰恰就是现实生活中的电梯模型，所以IO调 度器也被叫做电梯(elevator)调度，而相应的算法也就被叫做电梯算法。

我们知道，磁盘的读写是通过机械性的移动磁头来实现的，理论上磁盘设备满足块设备的随机读写的要求，但是出于节约磁盘，提高效率的考虑，我们希望当磁头处于某一个位置的时候，一起将最近需要写在附近的数据写入，而不是这写一下，那写一下然后再回来，IO调度就是将上层发下来的IO请求的顺序进行重新排序以及对多个请求进行合并，这样就可以实现上述的提高效率、节约磁盘的目的。Linux内核中提供了下面的几种电梯算法来实现IO调度。



# I/O调度算法

| 算法名 | 描述 |
| --- | --- |
| No-op I/O scheduler | 先来先服务算法：只实现了简单的FIFO，只进行最简单的合并，比较适合基于Flash的存储 |
| Anticipatory I/O scheduler | 预测调度算法：推迟IO请求(大约几个微秒)，以期能对他们进行排序，获得更高效率 |
| Deadline I/O scheduler | 最后期限调度算法：试图把每次请求的延迟降到最低，同时也会对BIO重新排序，特别适用于读取较多的场合，比如数据库 |
| CFQ I/O scheduler | 公平调度算法：为系统内所有的任务分配均匀的IO带宽，提供一个公平的工作环境，在多媒体环境中，能保证音视频及时从磁盘中读取数据，是当前内核默认的调度器。 |

# 如何指定或者改变调度算法

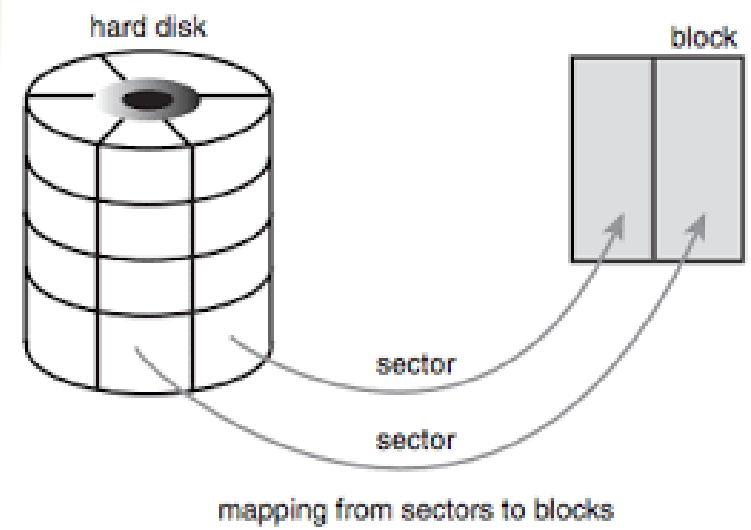
内核传参的方式指定使用的调度算法

kernel elevator=deadline

使用命令改变内核调度算法

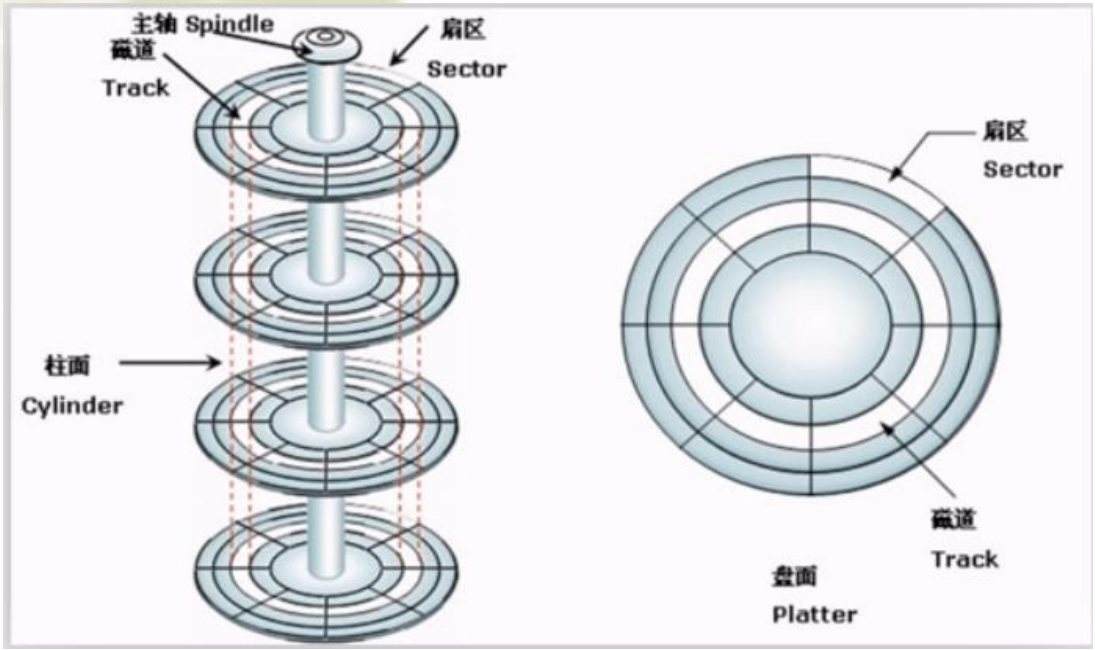
echo SCHEDULER >/sys/block/DEVICE/queue/scheduler

# 块-文件系统的读写单位




文件系统的读写单位是块，一个块的大小是2的n次方个扇区，比如1k，2k，4k，4M 等，如ext4 文件系统的block缺省是4k。block是VFS和文件系统传送数据的基本单位。

# 扇区-磁盘读写单位



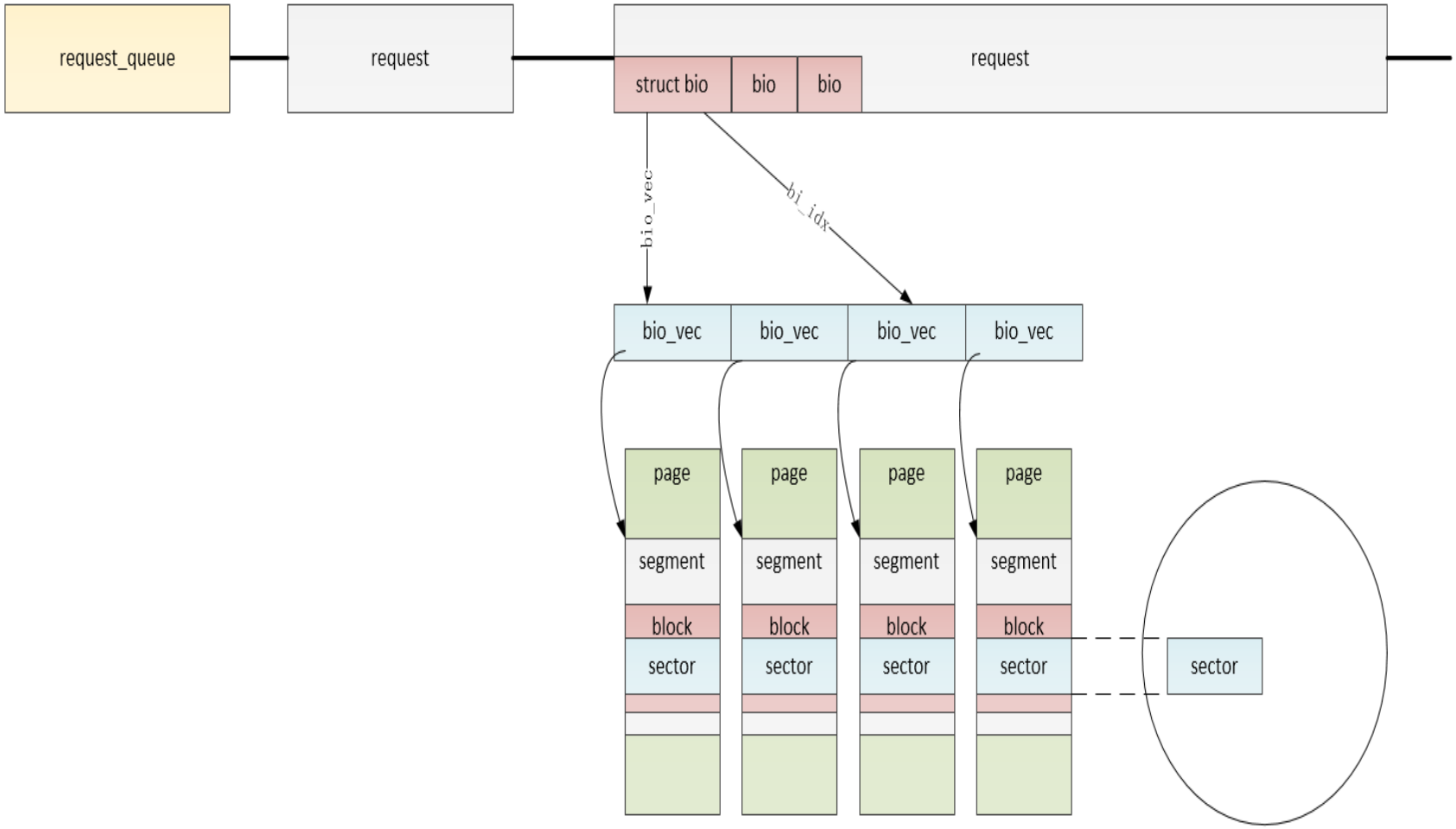

扇区（sector）：扇区是硬件（磁盘）上的最小的操作单位，是文件系统和块设备（硬件、磁盘）之间传送数据的单位。一般一个扇区的大小是512字节。如果实际的设备的扇区不是512字节，而是4096字节(比如 SSD)，那么只需要将多个内核扇区对应一个设备扇区即可。

# 块设备驱动的核心结构

| 核心结构 | 描述 |
| --- | --- |
| gendisk | 磁盘描述符，是对一个物理磁盘或分区的描述 |
| block_device_operations | 描述磁盘的操作方法集，它之于gendisk，类似于file_operations之于cdev |
| request_queue | 针对一个gendisk（磁盘）对象的所有请求的队列，是相应gendisk对象的一个域 |
| request | 表示经过IO调度之后的针对一个gendisk(磁盘)的一个"请求"，是request_queue的一个节点。多个request构成了一个request_queue |
| bio | 表示应用程序对一个gendisk(磁盘)原始的访问请求，一个bio由多个bio_vec组成，多个bio经过IO调度和合并之后可以形成一个request。 |
| bio_vec | 描述的应用层准备读写一个gendisk(磁盘)时需要使用的内存页page的一部分，也就是"段segment"，多个bio_vec和bio_iter形成一个bio |
| bvec_iter | 用于记录当前bio_vec被处理的情况，用于遍历bio。 |

# 核心结构之间的关系图

上面描述的这些核心数据结构，用图表示出来，如图所示：

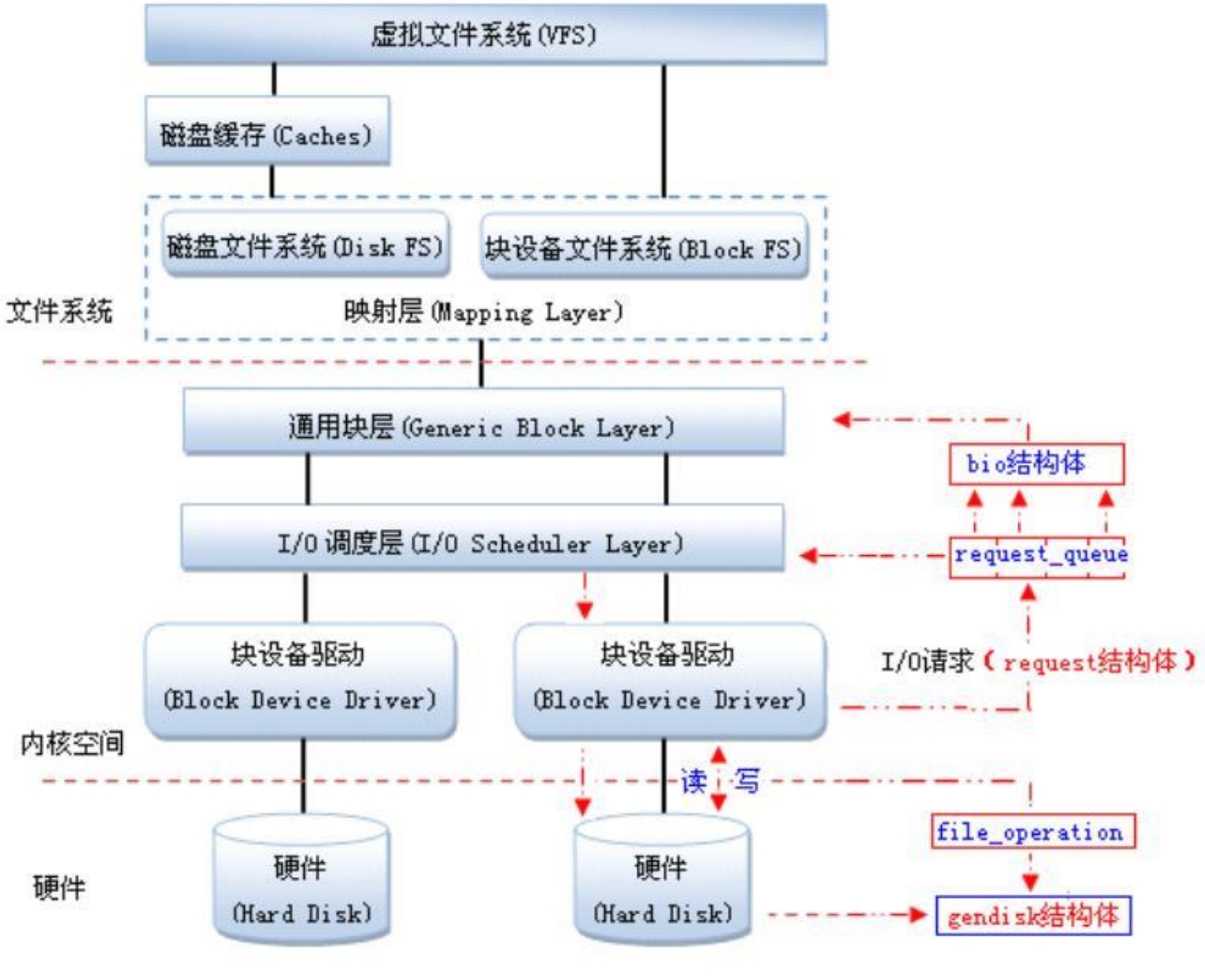


# 核心结构和核心方法详述

在网上看到一篇非常详尽的块设备驱动相关知识介绍。
https://www.cnblogs.com/xiaojiang1025/p/6500557.html

- 上面所讲内容大部分来自本篇，因此，建议大家仔细阅读，并动手实践，块设备编程基本就可以入门了。

# 块设备驱动程序小结


虚拟文件系统(VFS)
磁盘缓存(Caches)
磁盘文件系统(Disk FS)
块设备文件系统(Block FS)
文件系统
映射层(Mapping Layer)
通用块层(Generic Block Layer)
bio结构体
I/O 调度层(I/O Scheduler Layer)
request_queue
块设备驱动
(Block Device Driver)
块设备驱动
(Block Device Driver)
I/O请求(request结构体)
内核空间
读
写
file_operation
硬件
硬件
(Hard Disk)
硬件
(Hard Disk)
gendisk结构体

# 小结-从文件系统到驱动程序

1. VFS 层：VFS是对各种具体文件系统的一种封装，是用户程序访问文件提供统一的接口。

2. 缓存（Cache）层：当用户发起文件访问请求的时候，首先回到磁盘缓存中查找文件是否被缓存了，如果在缓存,则直接从缓存中读取。如果数据不在缓存中，就必须要到具体的文件系统中读取数据。

3. 映射层（ Mapping Layer）

（1） 首先确定文件系统的块大小，然后计算所请求的 数据包含多少个块。

(2) 调用具体文件系统的函数来访问文件的inode结构，确定所请求的数据在磁盘上的地址。

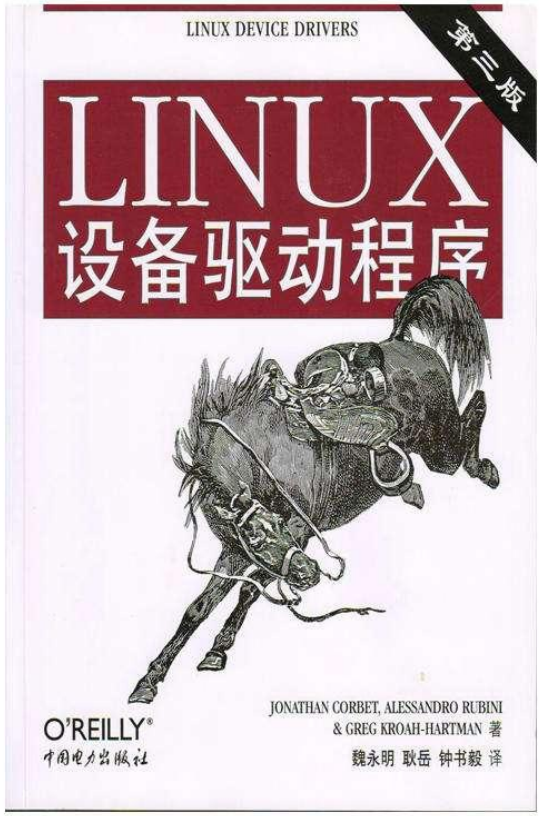
# 小结-从文件系统到驱动程序

4. 通用块层：Linux内核把块设备看做是由若干个扇区组成的数据空间，上层的读写请求在通用块层被构造成一个或多个bio结构。

5. I/O 调度层： I/O调度层负责采用某种算法（如：电梯调度算法）将I/O操作进行排序。

6. 块设备驱动:最底层，由块设备驱动根据排序好的请求，对硬件进行数据访问。

# 参考文献




1. 《Linux 驱动开发》

2. 网上有大量详尽的块驱动开发资料，读者可自行查阅,推荐一篇

https://my.oschina.net/fileoptions/blog/951759

3. 文中的大多数图片来自google搜索，版权归原作者所有

# 带着疑问上路

从虚拟文件系统到硬件，为什么要对块设备进行分层管理？

# 谢谢大家！

THANK YOU