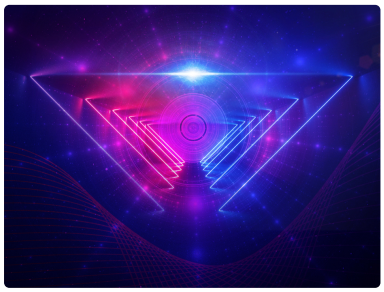


图文 34 生产经验：Linux操作系统的存储系统软件层原理剖析以及IO调度优化原理

手机观看

详情 评论

生产经验：Linux操作系统的存储系统软件层原理剖析以及IO调度优化原理

- **如何提问**：每篇文章都有评论区，大家可以尽情留言提问，我会逐一答疑
- **如何加群**：购买狸猫技术窝专栏的小伙伴都可以加入狸猫技术交流群，一个非常纯粹的技术交流的地方

具体加群方式，请参见目录菜单下的文档：**《MySQL专栏付费用户如何加群》**（购买后可见）

接着上一篇文章的讲解，我们继续来讲解MySQL数据库在执行底层磁盘读写IO操作的原理，这其实就涉及到了Linux操作系统的磁盘IO原理了，不管是MySQL执行磁盘随机读写，还是磁盘顺序读写，其实在底层的Linux层面，原理几乎都是一致的。

同时我们还会针对这块内容，连带讲解一下生产环境中，针对MySQL数据库的IO调度优化的建议。

大家都知道，所谓的操作系统，无论是Linux也好，还是Windows也好，说白了他们自己本身就是软件系统，之所以需要操作系统，是因为我们不可能直接去操作CPU、内存、磁盘这些硬件，所以必须要用操作系统来管理CPU、内存、磁盘、网卡这些硬件设备。

操作系统除了管理硬件设备以外，还会提供一个操作界面给我们，比如Windows之所以在全世界大获成功，其实就是他提供了一个比较简便易用的可视化的界面，让我们可以普通人都能操作台式电脑或者笔记本电脑内部的内存、CPU、磁盘和网卡。

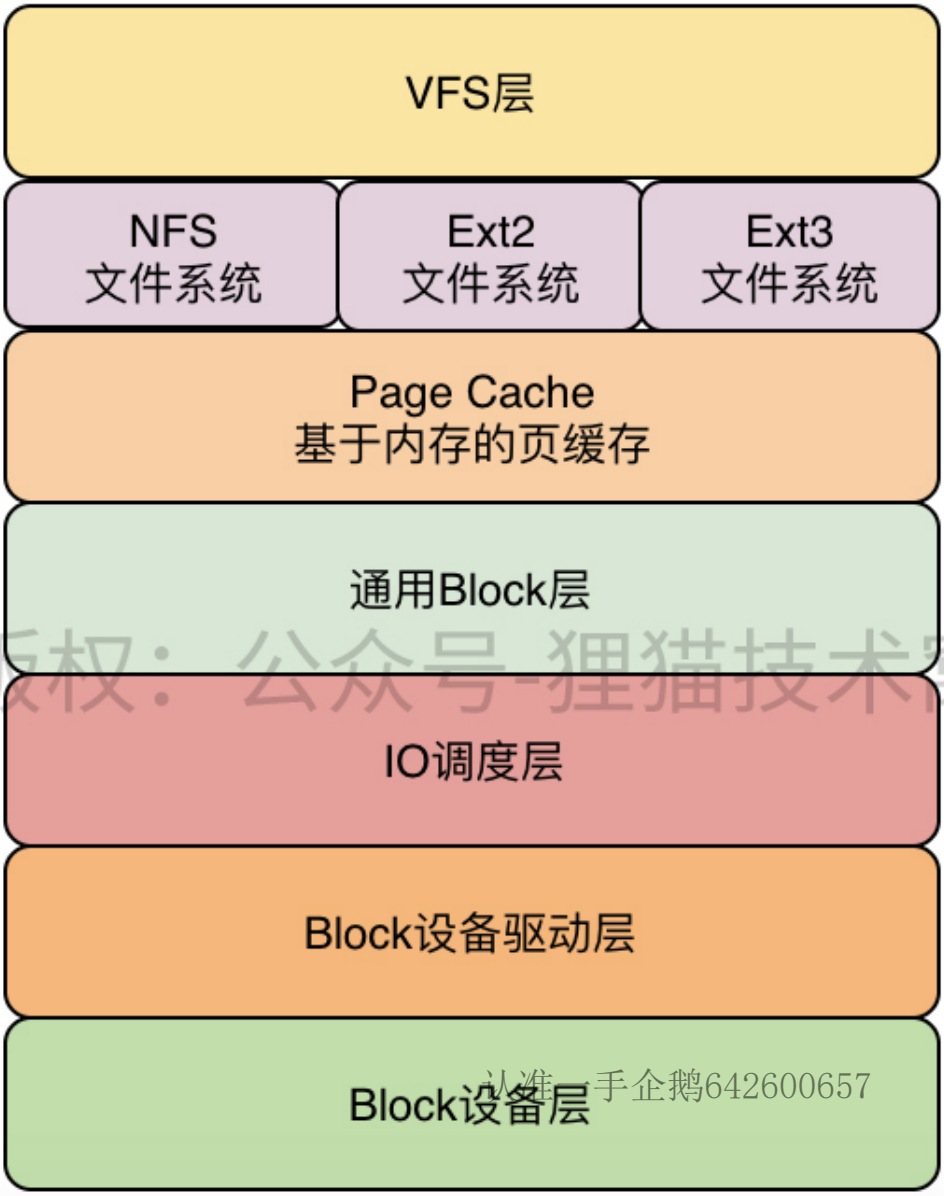
我们只要打开windows操作系统的电脑，就可以随意编辑文件，上网，聊天，使用各种软件，这些软件运行的时候本质底层都是在使用计算机的CPU、内存、磁盘和网卡，比如基于CPU执行你的文件编辑的操作，基于内存缓冲你对文件的编辑，基于磁盘存储你在文件里输入的内容，基于网卡去进行网络通信，让你进行QQ聊天什么的。

至于说linux操作系统，其实也是类似的，只不过一般我们用linux操作系统，他是不给我们提供可视化界面的，只有命令行的界面，我们需要输入各种各样的命令去执行文件编辑、系统部署和运行，本质linux操作系统在底层其实也是利用CPU、内存、磁盘和网卡这些硬件在工作。

所以，简单来说，我们今天要讲解的就是Linux操作系统的存储系统，Linux利用这套存储系统去管理我们的机器上的机械硬盘、□SSD固态硬盘，这些存储设备，可以在里面读取数据，或者是写入数据。

理解了这个，你就理解了MySQL执行的数据页随机读写，redo log日志文件顺序读写的磁盘IO操作，在Linux的存储系统中是如何执行的。

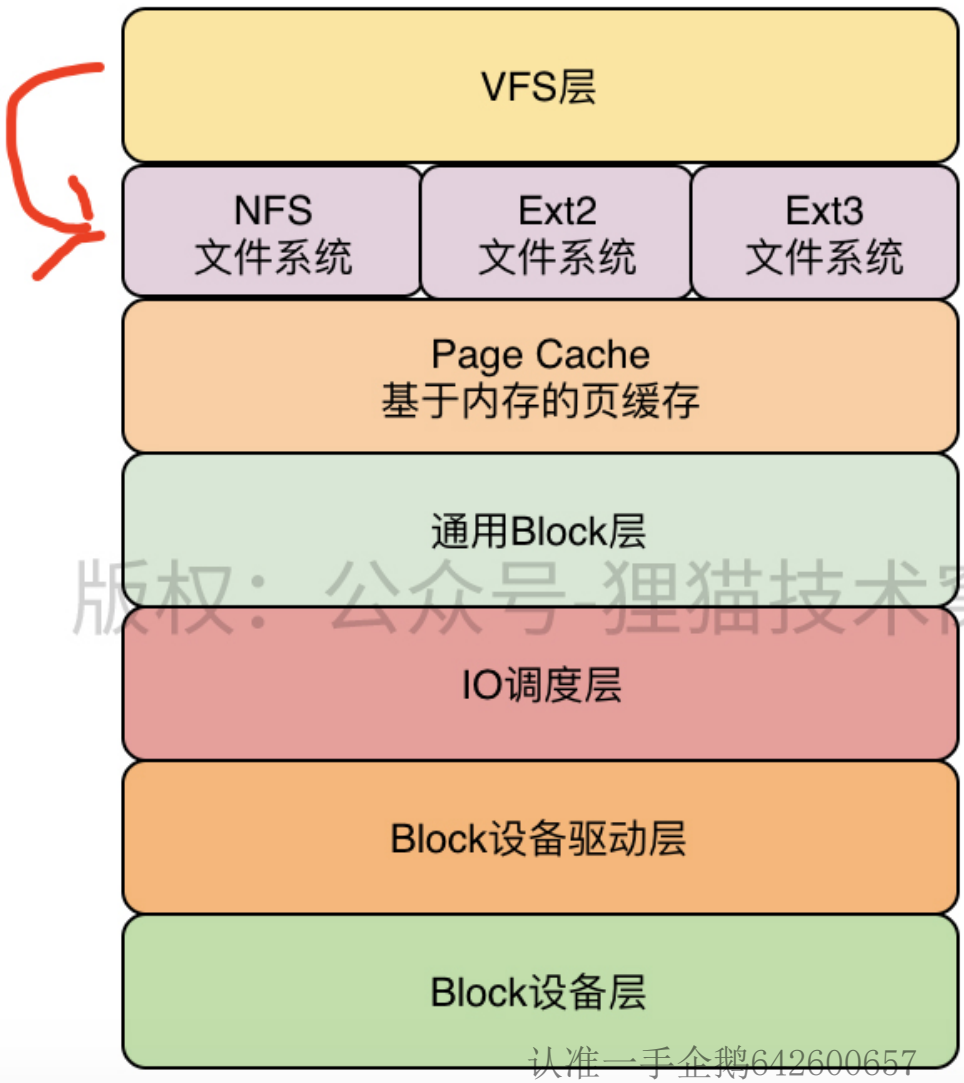
简单来说，Linux的存储系统分为VFS层、文件系统层、Page Cache缓存层、通用Block层、IO调度层、Block设备驱动层、Block设备层，如下图：



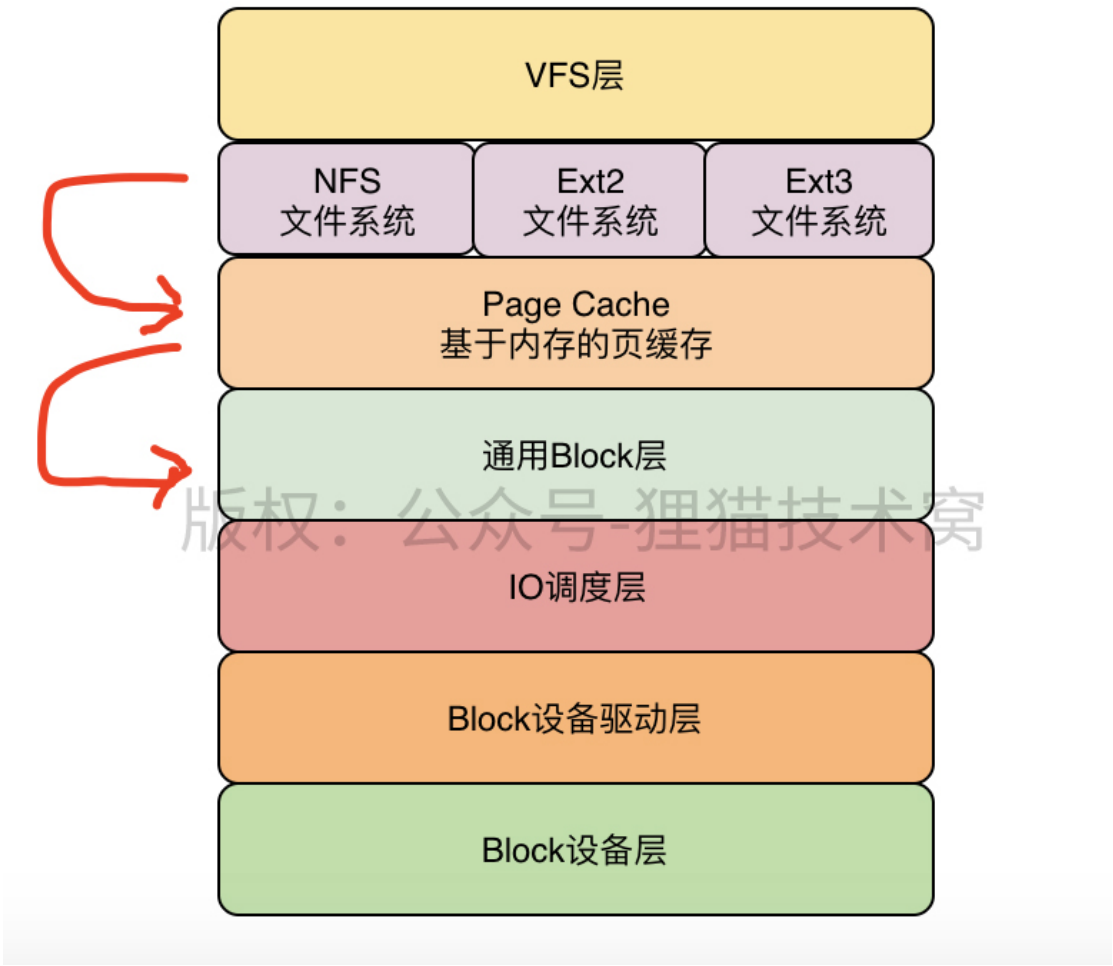
当MySQL发起一次数据页的随机读写，或者是一次redo log日志文件的顺序读写的时候，实际上会把磁盘IO请求交给Linux操作系统的VFS层

这一层的作用，就是根据你是对哪个目录中的文件执行的磁盘IO操作，把IO请求交给具体的文件系统。

举个例子，在linux中，有的目录比如/xx1/xx2里的文件其实是由NFS文件系统管理的，有的目录比如/xx3/xx4里的文件其实是由Ext3文件系统管理的，那么这个时候VFS层需要根据你是对哪个目录下的文件发起的读写IO请求，把请求转交给对应的文件系统，如下图所示。



接着文件系统会先在Page Cache这个基于内存的缓存里找你要的数据在不在里面，如果有就基于内存缓存来执行读写，如果没有就继续往下一层走，此时这个请求会交给通用Block层，在这一层会把你对文件的IO请求转换为Block IO请求，如下图所示。



接着IO请求转换为Block IO请求之后，会把这个Block IO请求交给IO调度层，在这一层里默认是用CFQ公平调度算法的

也就是说，可能假设此时你数据库发起了多个SQL语句同时在执行IO操作。

有一个SQL语句可能非常简单，比如update xxx set xx1=xx2 where id=1，他其实可能就只要更新磁盘上的一个block里的数据就可以了

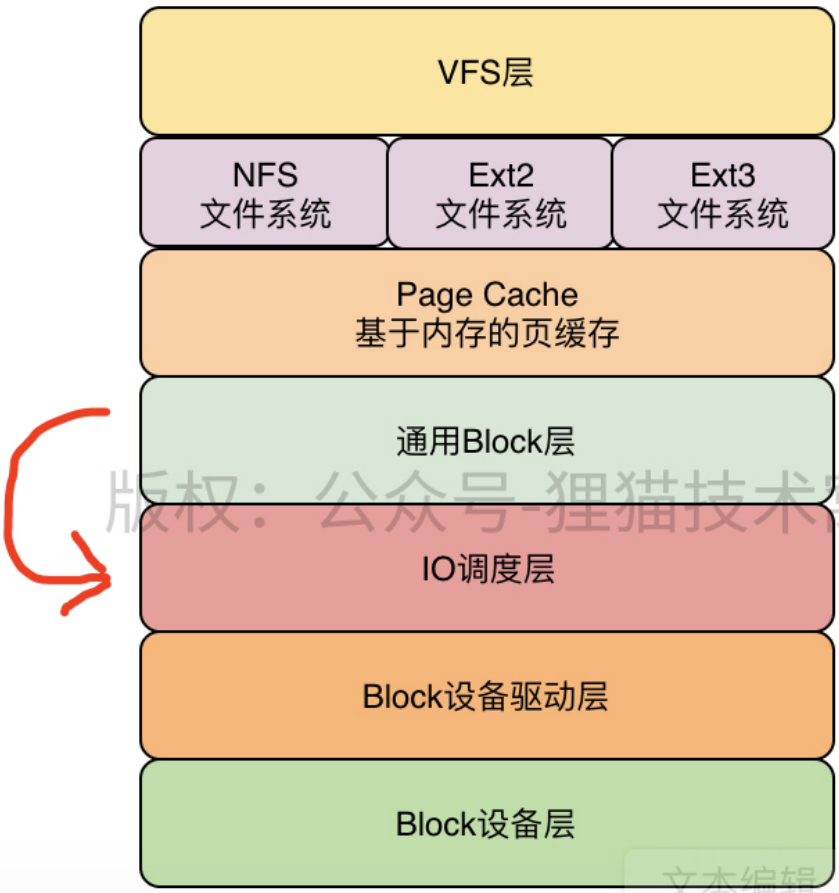
但是有的SQL语句，比如说select * from xx where xx1 like "%xx%"可能需要IO读取磁盘上的大量数据。

那么此时如果基于公平调度算法，就会导致他先执行第二个SQL语句的读取大量数据的IO操作，耗时很久，然后第一个仅仅更新少量数据的SQL语句的IO操作，就一直在等待他，得不到执行的机会。

所以在这里，其实一般建议MySQL的生产环境，需要调整为deadline IO调度算法，他的核心思想就是，任何一个IO操作都不能一直不停的等待，在指定时间范围内，都必须让他去执行。

所以基于deadline算法，上面第一个SQL语句的更新少量数据的IO操作可能在等待一会儿之后，就会得到执行的机会，这也是一个生产环境的IO调度优化经验。

我们看下图，此时IO请求被转交给了IO调度层。



最后IO完成调度之后，就会决定哪个IO请求先执行，哪个IO请求后执行，此时可以执行的IO请求就会交给Block设备驱动层，然后最后经过驱动把IO请求发送给真正的存储硬件，也就是Block设备层，如下图所示。