**BIO**

每一个线程对应一个client 每个线程是阻塞的，等待数据

**NIO（linux系统层级的NIO）**

一个线程轮询所有文件描述符，向内核发起系统调用获取数据（系统调用多）

**多路复用**

一个线程将所有文件描述符通过select方法扔给内核，内核遍历文件描述符，返回有数据的文件描述符

**Poll**

和select区别不大，但是文件描述符由链表存储，没有最大连接数限制

**Epoll**

开辟了2块内核空间 a ,b

每来一个客户端，将文件描述符放入a

然后通过硬件中断事件，得知客户端有无读写操作，如果有，则将文件描述符复制到b

然后拿出b里面的文件描述符就行了，复杂度o(1)

关键系统调用 epoll\_create epoll\_ctl epoll\_wait

查看程序的系统调用linux命令 strace

使用epoll的程序 nigix redis

**netty**

NioEventLoop

NioEventLoop底层会根据系统选择select或者epoll。如果是windows系统，则底层使用WindowsSelectorProvider（select）实现多路复用；如果是linux，则使用epoll

零拷贝

Sendfile系统调用 传入两个文件描述符，一个in 一个out

文件直接从磁盘（或其他地方），通过内核，到另一个地方

传统拷贝

程序向内核发起read系统调用，然后磁盘数据从内核拷贝到程序内存空间，然后程序再向内核发起write系统调用，将数据从程序内存空间拷贝到目的地文件描述符

零拷贝条件：不需要对数据进行修饰

**Java NIO**

IO 面向流(Stream，输入流 输出流) 阻塞

NIO 面向Buffer（数据写入buffer,在通道中双向传输） 非阻塞 Selector

**缓冲区**

7种基本类型的buffer（只有boolean没有）

基本操作

put get flip(切换读写模式) rewind（重复读）clear

核心属性

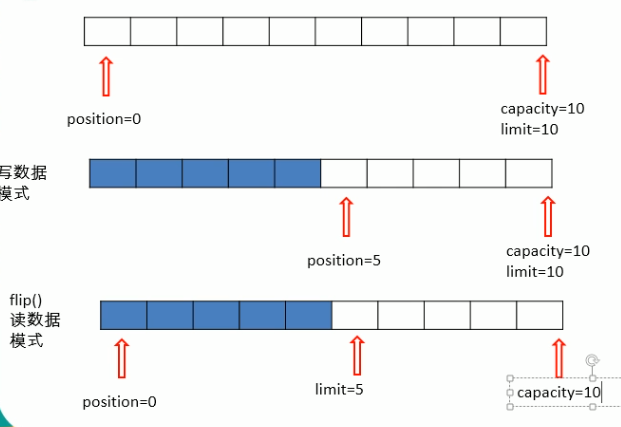
Capacity：容量

limit:界限（limit之后的不能读写）

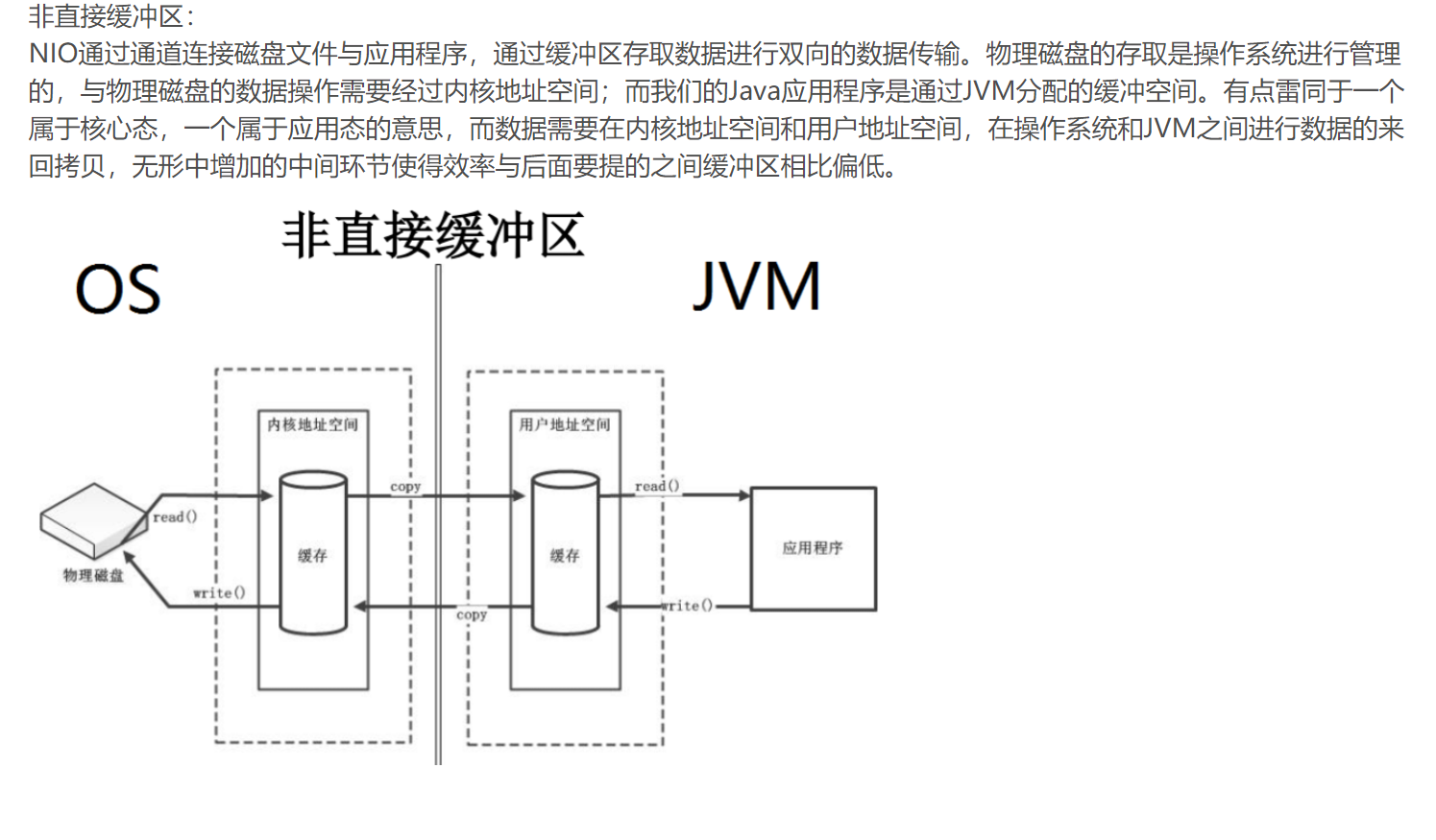
position:当前正在读写的位置

0<=mark<=position<=limit<=capacity

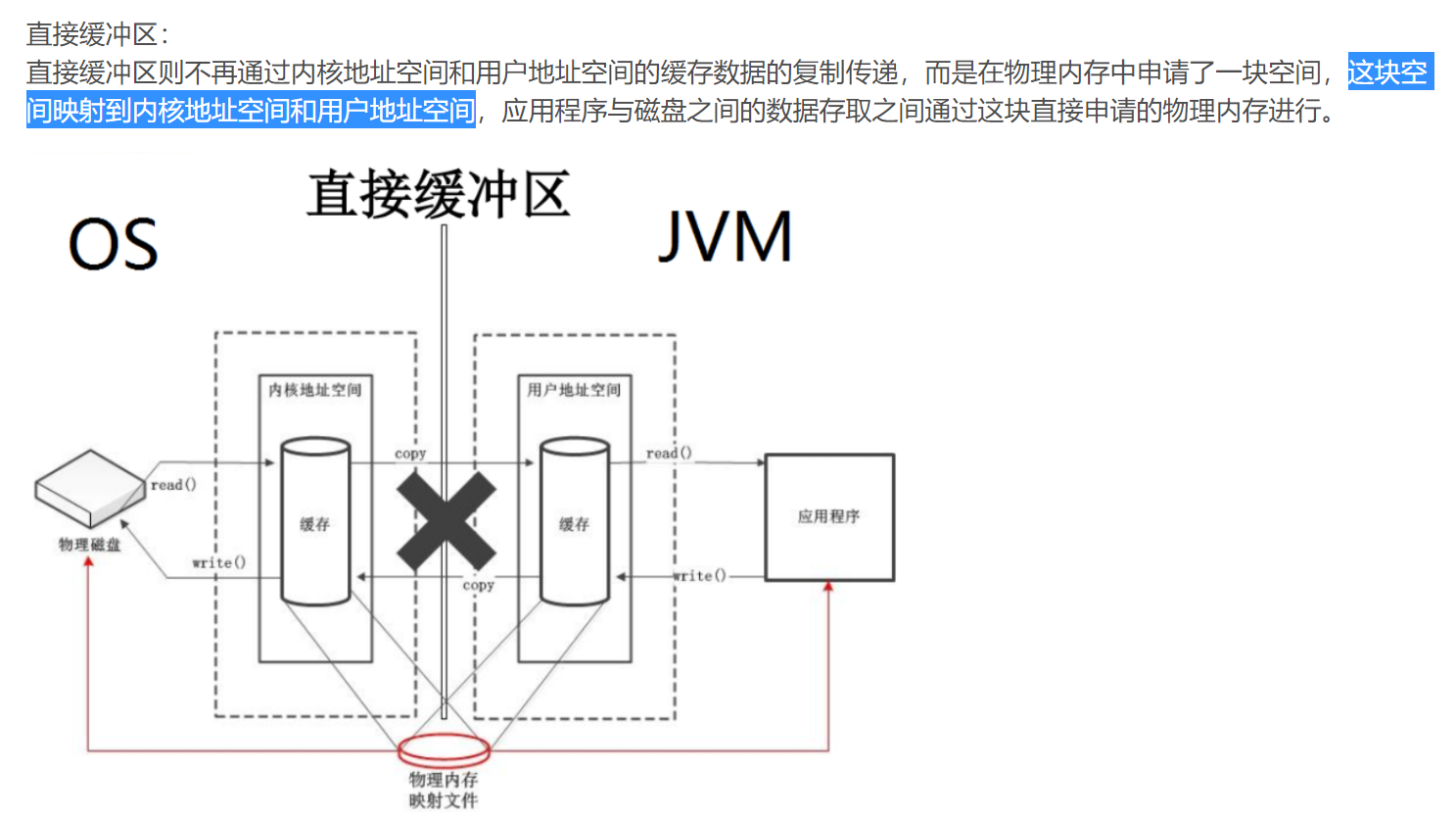
mark 标记 记录当时position的位置，之后可通过reset操作回到mark的位置



非直接缓冲区



直接缓冲区



通道(Channel)

IO源和目标之间的连接。Channel需要通过Buffer才能和数据交互

**非阻塞式网络通信**

NIO通过Selector监听所有client，将读写请求准备完成的交给服务端

Selector是对select poll epoll的封装，具体采用什么取决于操作系统。Linux默认epoll

流程

获取通道

切换到非阻塞模式

通道绑定到某个端口连接

获取选择器

将通道注册到选择器，监听Accept事件

轮询选择器上已经准备就绪的事件

获取准备就绪的客户端连接通道

通道切到换非阻塞模式

将通道注册到选择器上，监听读取事件

如果读取事件准备就绪。获取读数据通道，读取数据

**管道(Pipe)**

多个线程之间，通过·buffer读写数据