缓冲区的作用就是将协调低速设备与高速设备。

计算机内核有一个专门IO的缓冲区，用户区有多个进程，每个进程一般会为了减少系统调用的次数，会在IO前申请一块临时内存数组，作为该进程的缓冲区与内核缓冲区进行交互。所以JVM作为一个用户进程，它可以直接申请的一块内存作为临时的用户缓冲区，但是JVM的特殊性，他会在自己申请缓冲区时，用当前占用的内存拿一块作为缓冲区，并返回它的起始地址给系统，然后用它来与内核缓冲区进行交流，但是由于在JVM内部，它里面的地址是虚拟的，只有它自己才清楚什么它的缓冲区的地址，所以系统对这块地址的实际物理地址不知道是不是连续的，所以不一定能正确的直接与内核缓冲区直接交互，所以JVM会再去申请一块连续的内存作为两者之间的交互层，所以所有的数据IO会多拷贝一次，这就显得没有必要，但是JVM的封装性有得就有失。为了缓解这一问题，JVM后来采用一块固定的内存作为缓冲区，但是效果不够好，因为只是将两次拷贝变成了一次，而这一次拷贝也是没有必要的，于是在NIO中为了消除这一问题，就提出了直接缓冲区，它使用内存映射技术，其实说白了就是让IO 的读写不让cpu参与而通过DMA这种独立的读写通道来直接将数据写入内存不用再通过内核的缓冲区拷贝。

**配合这文章你就能彻底知道java缓冲区之间的关系特别是第二篇：**

<https://blog.csdn.net/huaiqu6460/article/details/89677420>

<https://blog.csdn.net/bb_it007/article/details/70920655>

内存映射技术：

<https://blog.csdn.net/y277an/article/details/98000132>

<https://blog.csdn.net/nicewuranran/article/details/52145677>

1.非直接缓冲区：通过allocate（）进行分配的缓冲区，它是将缓冲区建立在JVM内存中。

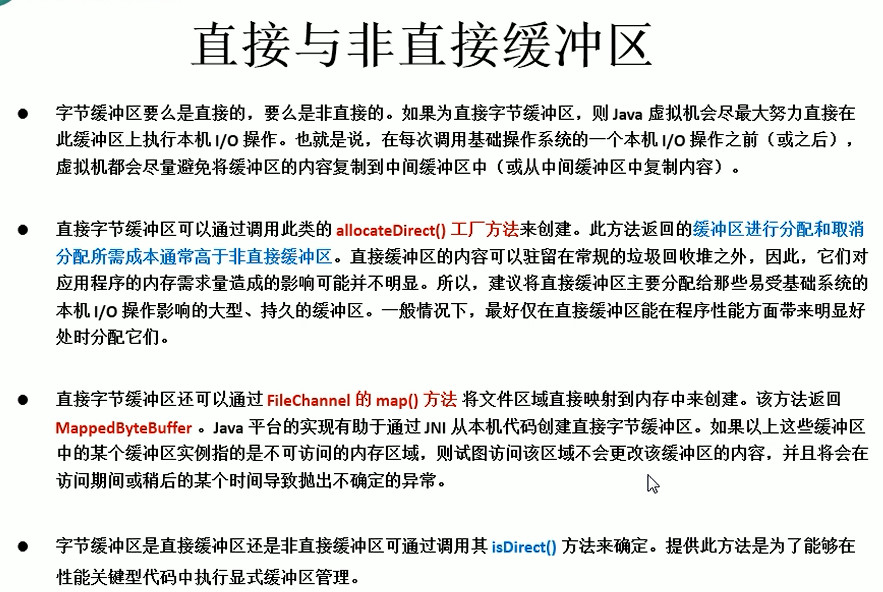
2.直接缓冲区：通过allocateDirect（）方法分配直接缓冲区，它将缓冲区建立在操作系统的物理内存中。在某些情况下是能够提高效率的。

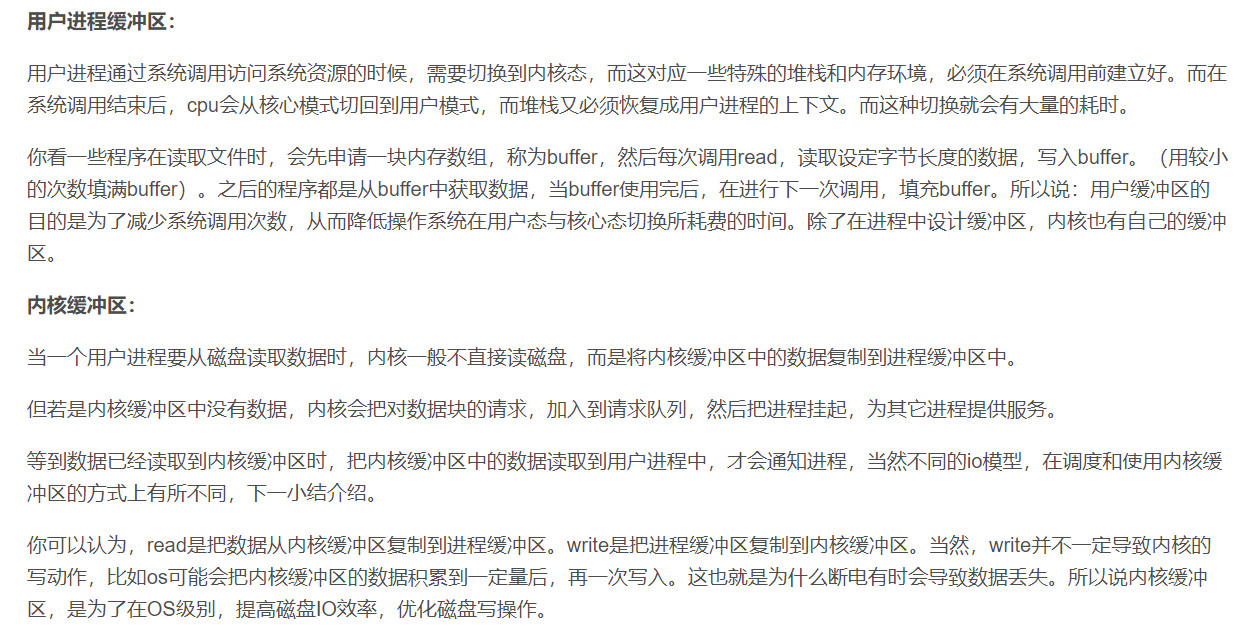
（有点本地方法味道在里面，虽然实际上JVM会通过操作系统进行资源申请，但是前者是会受到JVM的限制）

问题：JVM执行程序申请资源如何与操作系统进行交互，在执行程序过程中他们之间的关系究竟是怎样的？

问题：allocateDirect（）如何进行缓冲区的分配，而JVM又将如何与它进行联系？

问题：为什么allocateDirect（）能够提高效率？





每个Java程序的运行会生成一个JVM进程，这个进程由main进程主导，可以在里面开启多个子进程，这个JVM进程就相当于用户进程。

所以直接缓冲区相当于内核缓冲区，非直接缓冲区相当于用户缓冲区，但是由于是JVM进行管理，所以相当于JVM申请得到一块用户缓冲区资源，然后根据程序的需求进行分配管理。

可见其实JVM管理内存应该是一个动态过程，因为进程非是静态概念，一切非写死，资源不一定一次分配完全。所以他与OS之间必然需要交互。比如程序要申请直接缓冲区。（有点责任链模式的味道在里面）

**基于上述，我们可以理一下JVM执行过程：**

关于JVM的启动可以看看这个地址：

[***https://www.cnblogs.com/muffe/p/3540001.html***](https://www.cnblogs.com/muffe/p/3540001.html)

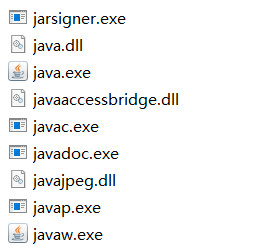
编写好一个程序后，通过JavaC命令进行将源文件编译为字节码文件，

然后通过Java 字节码文件名.class来执行程序，而这个程序的入口就是main方法，

没有main，它就会报错，所以一般程序编写规范，只是将main()作为一个程序入口，正真的逻辑都封装到其他类中。

而JavaC与Java这些操作指令都怎么被找到并执行的，首先这些指令都是存在于jdk的/bin目录下，所以我们在安装jdk时需要配置JAVA\_HOME环境变量，并且需要将其的/bin添加到PATH路径中，这样不仅仅当你要执行命令时，OS就能自动加载相应的路径中的程序到内存并开始执行。

当我们使用Java 命令执行程序时，（宋红康版JVM中说过JVM启动前时因为引导类加载器加载了指定类，**注意所谓的启动并不是站在用户角度说看见程序的图标出现就叫启动了，此启动非彼启动**）**所以我们可以推测JVM的加载过程和OS的装载过程及其相似，**但是起始装载的原因肯定还是Java这个指令引起的，Java指令对应的源程序



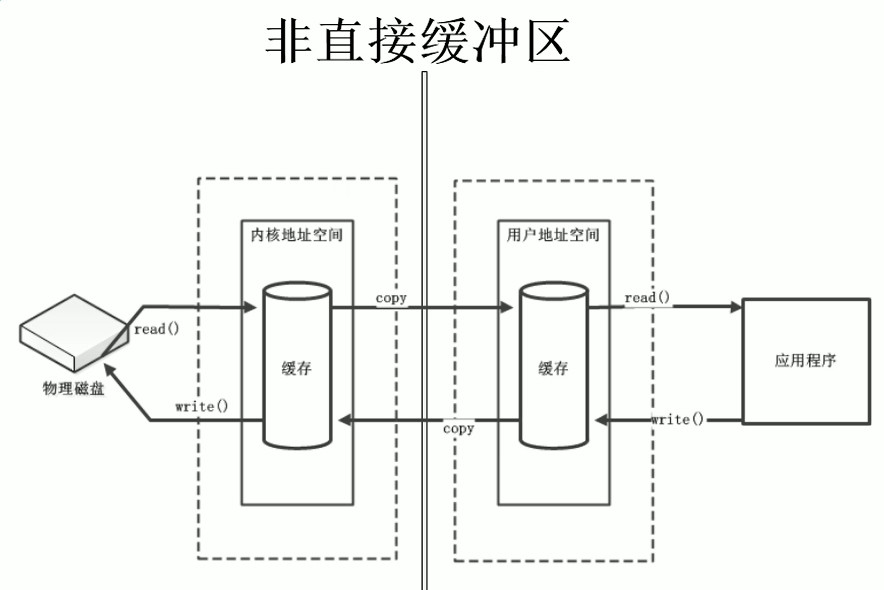
会根据你的指令参数去执行一些程序，这中间就会寻找JVM的路径，并将其加载进内存，**但是JVM作为一个程序，一个程序想要正常执行，就必须要构建好其运行环境，每个程序都是如此（即便是OS这种特殊程序），**所以这时候就显得我们的JAVA\_HOME的作用尤为重要了，JVM需要的有些东西就在这里面，JVM装载入内存开始执行并逐步构建自己能启动的环境这必须要依赖于它所需要的资源，这些资源由操作系统提供（而从广义上说操作系统的资源就是硬件资源，中间其实应该涉及到微指令微程序这些东西），当我们的JVM启动



后，OS上就开始了一个JVM进程，这时候JVM就能将你要执行的字节码文件先通过类加载器加载进运行数据区，开始执行你的程序，这期间程序有各种要求，JVM能满足的资源就满足，不能满足的就和OS进行交互。（从某种意义上说JVM具有一定OS管理特征，它本质上是一个程序，但又有资源和运行的独立性，但是又不能完全独立，这是每个程序必然的缺点即便它是虚拟机。）

非直接缓冲区：

（）



这样来回拷贝数据及其降低效率于消耗没必要的性能，但是这也是为了保证了数据的安全性。

为了性能：在JavaNIO中提出了一种物理内存映射文件，也叫做直接缓冲区。

优点：快速

缺点：不安全（将数据写到这个映射文件后，里卖弄得数据就不归我们管了，数据什么时候会被写入到磁盘中完全由操作系统决定，所以要分配和销毁消耗很大，如果要销毁只有当垃圾回收器释放该内存引用才会被销毁）、消耗大。

（**判断是否是直接缓冲区**）

