1.DirectByteBuffer

直接缓冲,直接操作本地空间数据，不再java内存空间内

见**Buffer**中的address属性

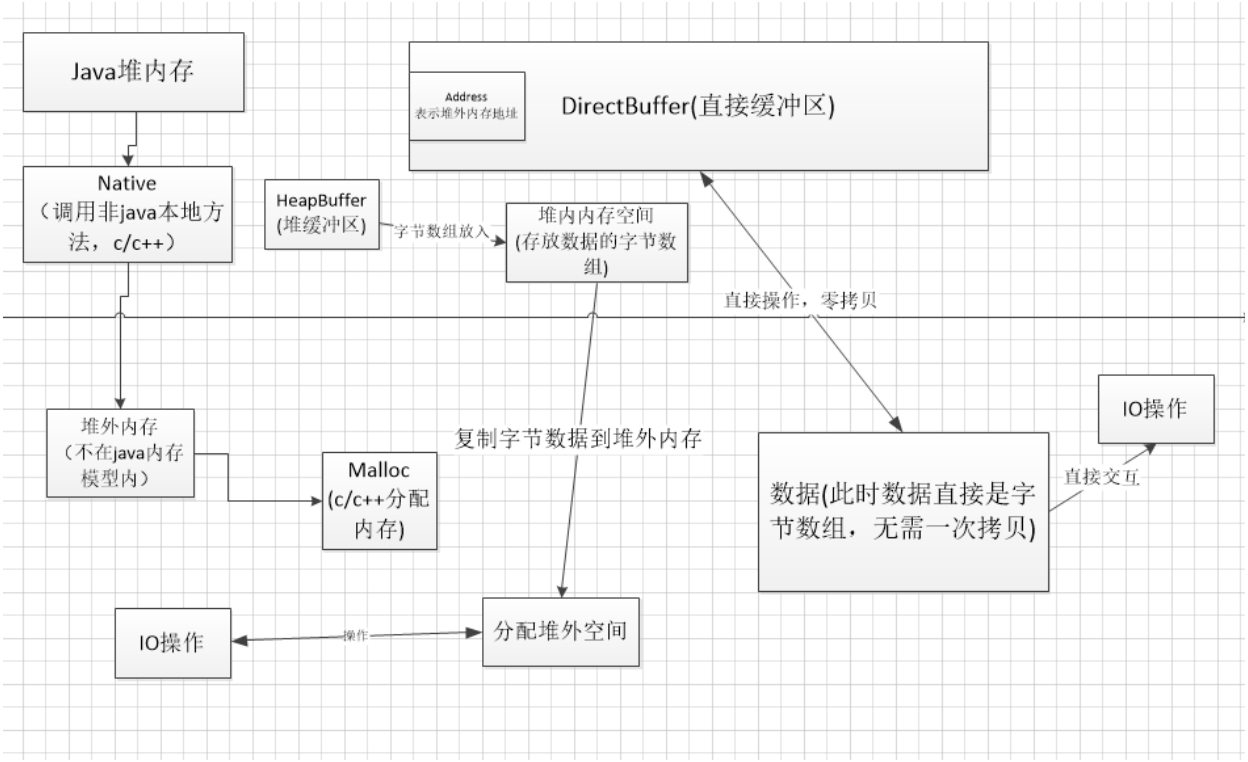
```
// Used only by direct buffers
// NOTE: hoisted here for speed in JNI GetDirectBufferAddress
long address;
```

只能被DirectBuffer使用

之所以放在Buffer中，是为了JNI调用时提升速率

address表示堆外内存的数据地址

HeapBuffer在JVM模型内，堆内空间，DirectBuffer在也在堆内空间，但是address属性表示堆外空间地址。



对于操作系统来说，IO操作并不是直接操。

HeapBuffer(堆缓冲区)

比如HeapBuffer实在堆内空间存放数据（字节数组），操作系统需要在堆外空间在开辟一块空间，然后将堆内内存数据拷贝到堆外内存数据。然后在再通过IO操作操作数据，

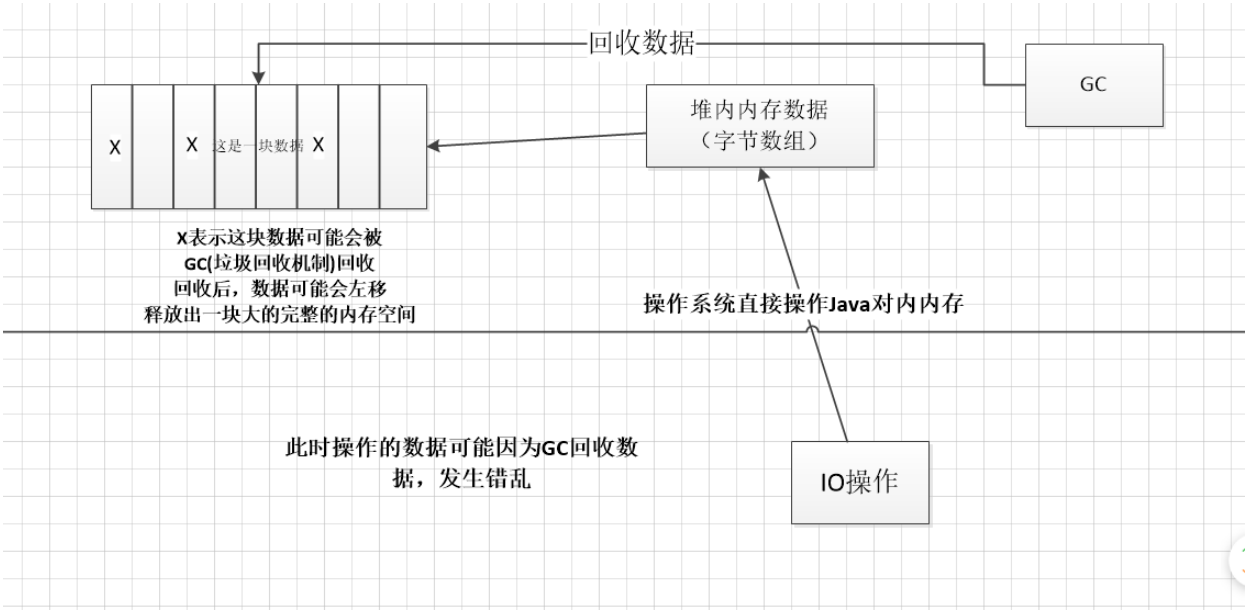
DirectBuffer(直接缓冲区)

DirectBuffer通过Address(表示堆外内存地址)属性直接操作堆外内存数据（存放数据直接存放在堆外而不是堆内内存复制到堆外内存），此时IO操作直接操作堆外数据

直接内存模型(零拷贝)：零拷贝，address维护堆外内存地址，address销毁，堆外内存释放

非直接(间接)内存模型：需要拷贝到间接缓冲区

严格来说，操作系统可以操作所有内存(堆外堆内都行)，通过原生操作方法操作java堆内内存，可能出现问题，如图



直接操作java堆内内存，gc调用时候数据会错误。

解决方法：

1.关闭GC，即不回收(不可取)

2.堆内内存数据拷贝到堆外内存(**拷贝过程中不会调用GC**)，拷贝过程需要开销

3.零拷贝，java直接操作堆外内存通过native修饰的本地方法