# 直接内存

1. 直接内存不是虚拟机运行时数据区的一部分，也不是《Java虚拟机规范》中定义的内存区域。
2. 直接内存是java堆外内存，不在jvm内存中，直接向系统申请的内存区间。
3. Jdk8后元空间用的就是直接内存
4. 通常，访问直接内存的速度要优于Java堆，因为读写性能高。因此读写频繁的场合会使用直接内存来提高性能。
5. 来源于NIO,通过存在堆中的DirectByteBuffer对象来操作本地内存
6. Java的NIO库允许java程序使用直接内存，用于数据缓冲区。

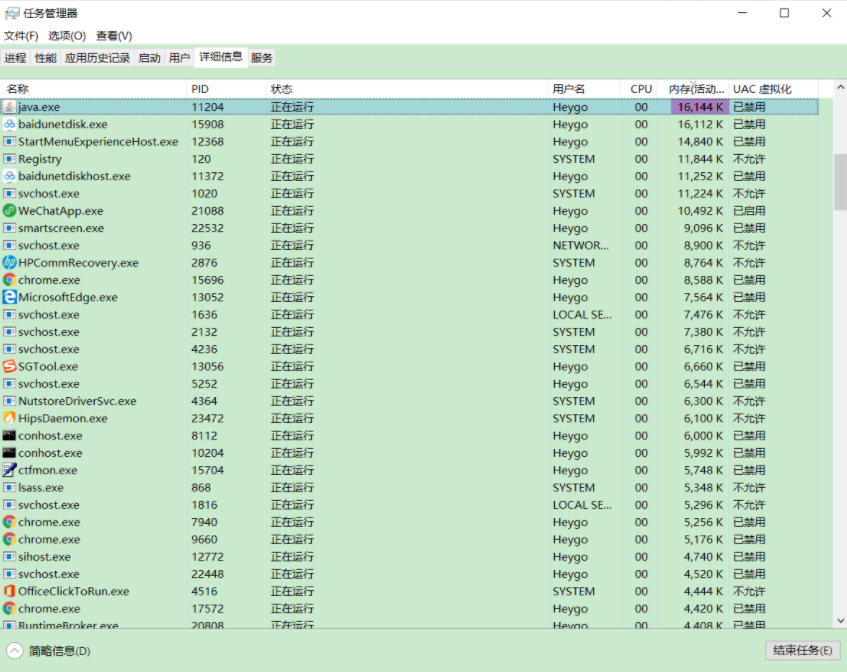
|  |
| --- |
| import java.nio.ByteBuffer; import java.util.Scanner;  */\*\*  \* IO NIO (New IO / Non-Blocking IO)  \* byte[] / char[] Buffer  \* Stream Channel  \*  \* 查看直接内存的占用与释放  \* @author shkstart shkstart@126.com  \* @create 2020 0:22  \*/* public class BufferTest {  private static final int *BUFFER* = 1024 \* 1024 \* 1024;*//1GB* public static void main(String[] args){  *//直接分配本地内存空间* ByteBuffer byteBuffer = ByteBuffer.*allocateDirect*(*BUFFER*);  System.*out*.println("直接内存分配完毕，请求指示！");   Scanner scanner = new Scanner(System.*in*);  scanner.next();   System.*out*.println("直接内存开始释放！");  byteBuffer = null;  System.*gc*();  scanner.next();  } } |

创建DirectByteBuffer直接内存对象，该对象使用直接内存。参数是字节数组的长度，单位是字节。

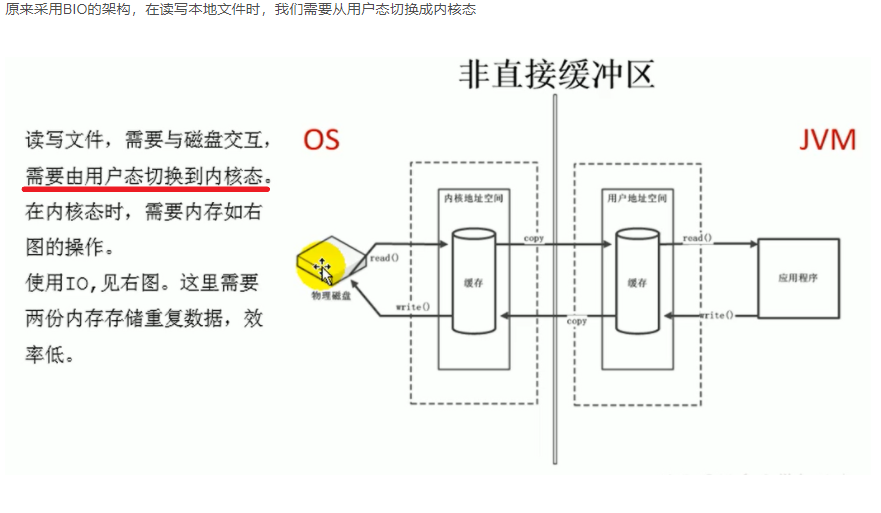
代码里的意思是创建一个大小为1GB的DirectByteBuffer对象，然后通过丢失引用，垃圾回收，回收此直接内存对象。会发现Java程序占用内存小了很多



释放后，Java程序的内存占用明显减少

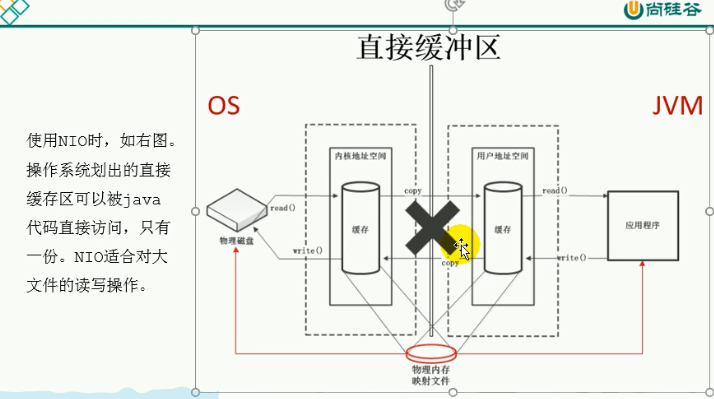


# 为什么直接内存比堆内存要高效？



Java程序要读物理磁盘的文件，应用程序

NIO 直接操作物理磁盘，省去了中间商赚差价



# 直接内存也会导致OutOfMemoryError异常

直接内存在堆外，因此其大小不会受限于虚拟机内存的大小。

但是系统内存是 有限的,java堆和直接内存的总和依然受限于操作系统可以给出的最大内存。

直接内存溢出异常：

java.lang.OutOfMemoryError: Direct buffer memory

JVM内存可视化工具是无法看见直接内存的情况的

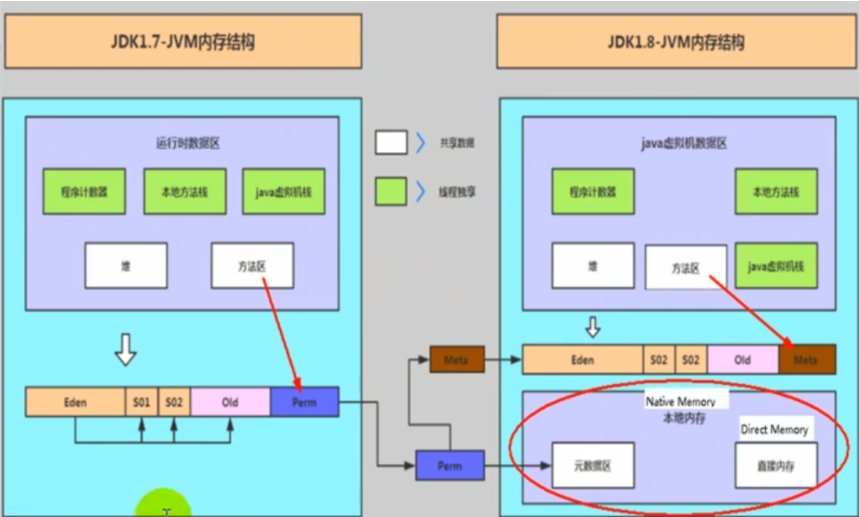
## 直接内存的缺点：

1. 分配回收成本高：直接内存的回收
2. 不受JVM内存回收管理

## 直接内存的大小设置

* 直接内存的大小可以通过MaxDirectMemorySize来设置。

如果不指定，默认和堆的最大值-Xmx参数一致。



Jdk1.8之后JVM内存结构可以理解为：java运行时数据区 + 本地内存。