

第10章 直接内存 (Direct Memory)

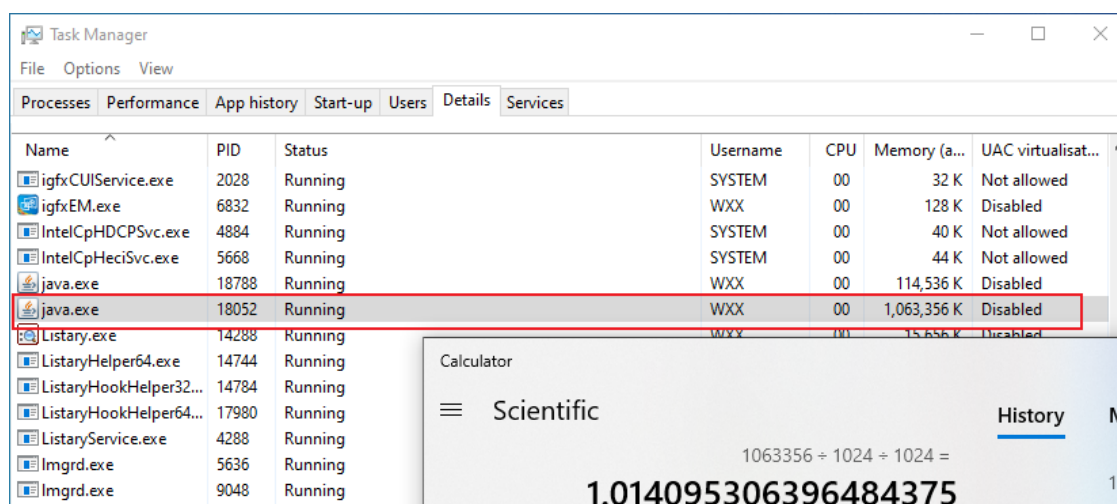
- 不是虚拟机运行时数据区的一部分，也不是《Java虚拟机规范》中定义的内存区域。
- 直接内存是在Java堆外的、直接向系统申请的内存空间。
- 来源于NIO，通过存在堆中的DirectByteBuffer操作Native内存。

```
/**
 * IO                NIO (New IO / Non-Blocking IO)
 * byte[] / char[]   Buffer
 * Stream            Channel
 *
 * 查看直接内存的占用与释放
 */
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();

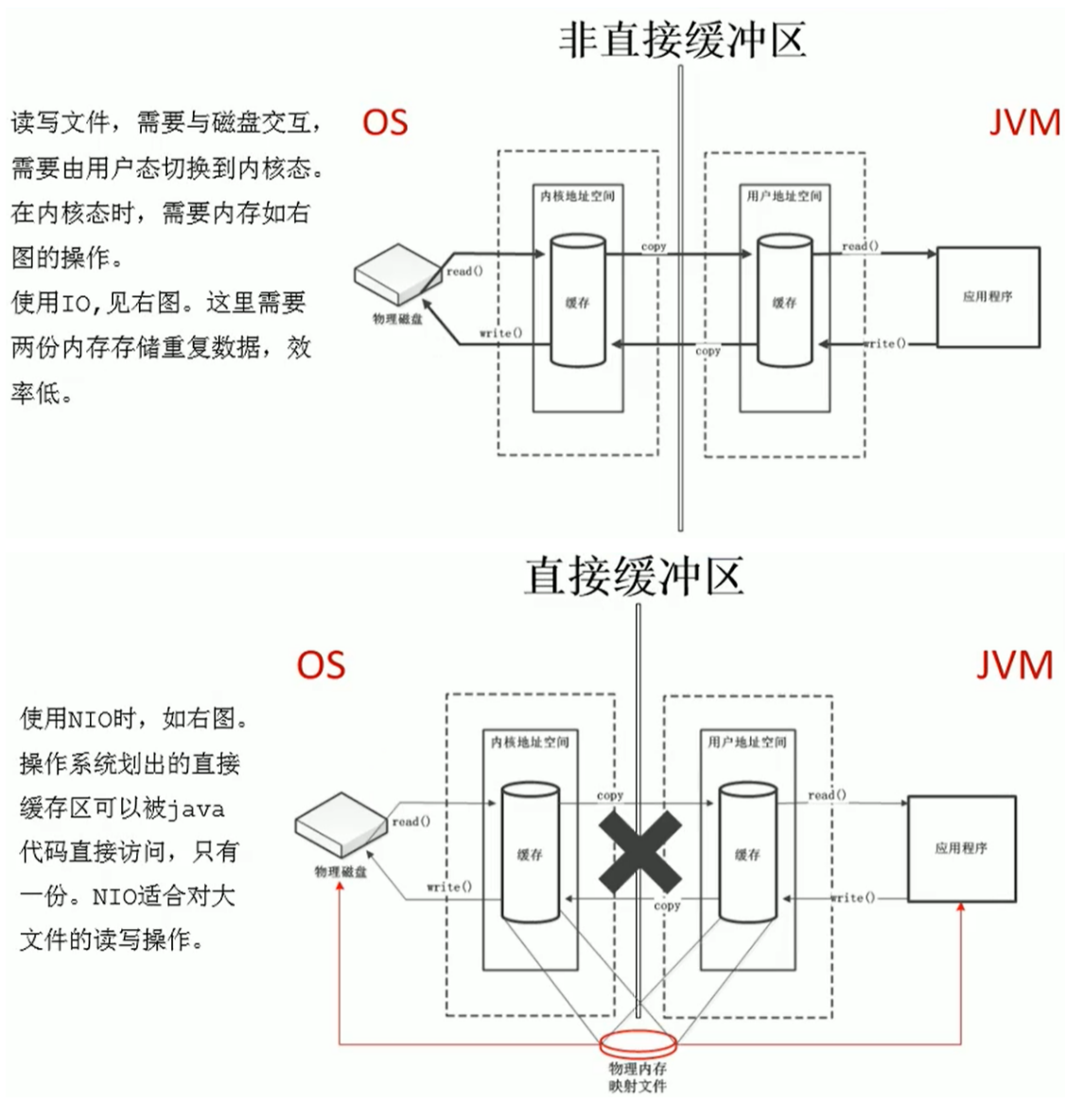
        System.out.println("直接内存开始释放！");
        byteBuffer = null;
        System.gc();
        scanner.next();
    }
}
```



Name	PID	Status	Username	CPU	Memory (a...)	UAC virtualisat...
igfxCUIService.exe	2028	Running	SYSTEM	00	32 K	Not allowed
igfxEM.exe	6832	Running	WXX	00	128 K	Disabled
IntelCpHDCPSvc.exe	4884	Running	SYSTEM	00	40 K	Not allowed
IntelCpHeciSvc.exe	5668	Running	SYSTEM	00	44 K	Not allowed
java.exe	18788	Running	WXX	00	114,536 K	Disabled
java.exe	18052	Running	WXX	00	1,063,356 K	Disabled
Listary.exe	14288	Running	WXX	00	15,656 K	Disabled
ListaryHelper64.exe	14744	Running				
ListaryHookHelper32...	14784	Running				
ListaryHookHelper64...	17980	Running				
ListaryService.exe	4288	Running				
Imgrd.exe	5636	Running				
Imgrd.exe	9048	Running				

- 通常，访问直接内存的速度会优于Java堆。即读写性能高。

- 因此出于性能考虑，读写频繁的场所可能会考虑使用直接内存。
- Java的NIO库允许Java程序使用直接内存，用于数据缓存区



- 也可能导致OutOfMemoryError异常
- 由于直接内存存在Java堆外，因此它的大小不会直接受限于-Xmx指定的最大堆大小，但是系统内存是优先的，Java堆和直接内存的总和依然受限于操作系统能给的最大内存。

```
/**
 * 本地内存的OOM: OutOfMemoryError: Direct buffer memory
 */
public class BufferTest2 {
    private static final int BUFFER = 1024 * 1024 * 20; //20MB

    public static void main(String[] args) {
        ArrayList<ByteBuffer> list = new ArrayList<>();

        int count = 0;
        try {
            while(true){
```

```

        ByteBuffer byteBuffer =
ByteBuffer.allocateDirect(BUFFER);
        list.add(byteBuffer);
        count++;
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
} finally {
    System.out.println(count);
}
}
}

```

BufferTest2 ×

```

D:\Java\jdk1.8.0_231\bin\java.exe ...
180
Exception in thread "main" java.lang.OutOfMemoryError: Direct buffer memory
    at java.nio.Bits.reserveMemory(Bits.java:694)
    at java.nio.DirectByteBuffer.<init>(DirectByteBuffer.java:123)
    at java.nio.ByteBuffer.allocateDirect(ByteBuffer.java:311)
    at com.atguigu.java.BufferTest2.main(BufferTest2.java:21)

```

- 缺点
 - 分配回收成本高
 - 不受JVM内存回收管理
- 直接内存大小可以通过MaxDirectMemorySize设置

```

/**
 * 绕过DirectByteBuffer，直接分配本地内存
 * -Xmx20m -XX:MaxDirectMemorySize=10m
 *
 * @author shkstart shkstart@126.com
 * @create 2020 0:36
 */
public class MaxDirectMemorySizeTest {
    private static final long _1MB = 1024 * 1024;

    public static void main(String[] args) throws
IllegalAccessError {
        Field unsafeField = Unsafe.class.getDeclaredFields()[0];
        unsafeField.setAccessible(true);
        Unsafe unsafe = (Unsafe) unsafeField.get(null);
        while (true) {
            unsafe.allocateMemory(_1MB);
        }
    }
}

```

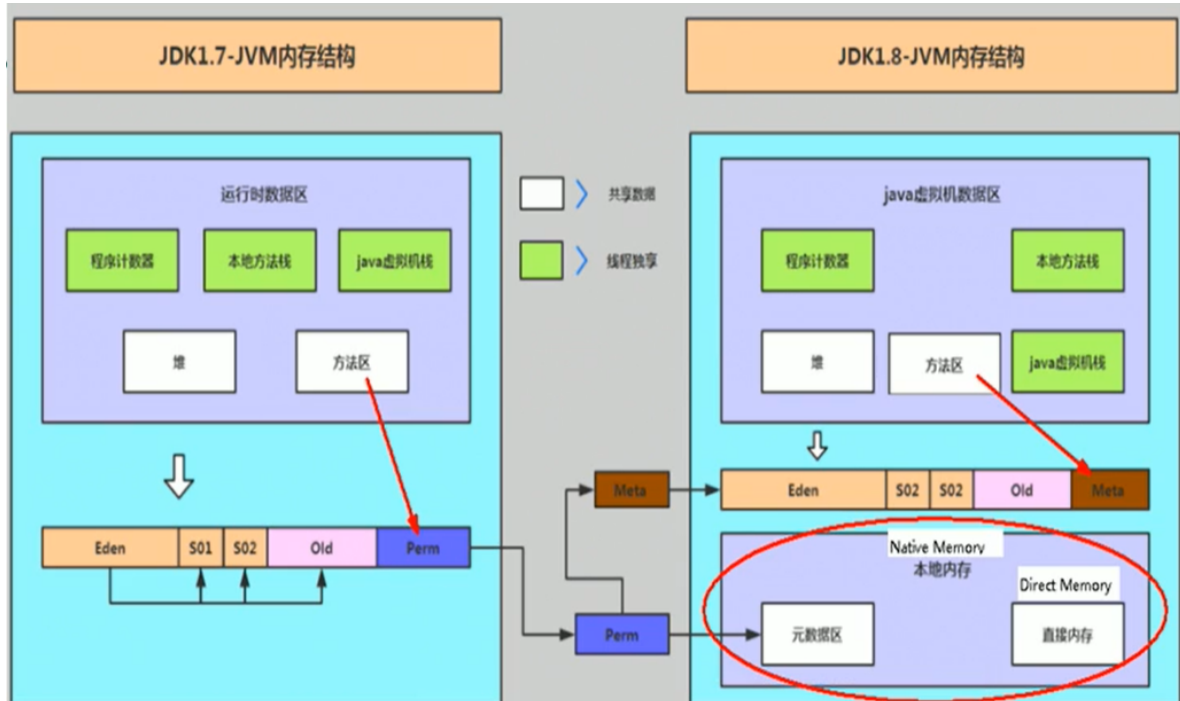
```
D:\Java\jdk1.8.0_231\bin\java.exe ...
```

```
Exception in thread "main" java.lang.OutOfMemoryError
```

```
at sun.misc.Unsafe.allocateMemory(Native Method)
```

```
at com.atguigu.java.MaxDirectMemorySizeTest.main(MaxDirectMemorySizeTest.java:22)
```

- 如果不指定，默认与堆的最大值-Xmx参数值一致



简单理解：**java process memory = java heap + native memory**