JVM内存划分，哪些会发生oom错误?

- 程序计数器，记录当前线程执行的java方法的jvm指令地址。线程私有。不会oom
- jvm虚拟机栈，线程中每次方法调用都创建一个虚拟机栈，栈里保存局部变量表、操作数栈、动态链接、方法正常或异常退出的定义等。线程私有。会发生oom，方法调用层级过多，会报StackOverflowError
- 本地方法栈，与jvm虚拟机栈相似，对应的是本地方法的执行。线程私有。
- 堆，存储所有对象，分为新生代、老年代，容易发生oom。线程共享。
- 方法区，存储元数据，包括类结构信息、运行时常量池、字段、方法代码等
- 直接内存，不在jvm进程控制的内存范围内，但是也可以被jvm所使用
- Code Cache，JIT编译后的代码

分配100M的数组，发生了OOM，但是GC日志显示剩余空间超过100M，这是可能是什么原因?

猜测原因之前，先确认两点：堆剩余超过100M，是新生代+老年代>100M；数组分配内存是需要连续的。
如果新生代和老年代各自的剩余空间小于100M，那么一定会分配失败。这时候看需要考虑扩大内存。
如果老年代内存大于100M，那需要考虑收集器类型。
如果是CMS收集器，可能是内存碎片过多，导致没有大块连续内存可以分配。
如果是G1收集器，可能region分配太小，不足放下100M的数组。