GC总结:

1) JAVA 8 默认采用并行GC ，在调整GC内存变大的情况下，该种策略会降低GC次数，但是每次GC处理时间会增长数倍。

2) 串行化Serial GC ，单线程处理，效率较低。

3) CMSGC:在串行GC基础上采用了多线程处理，YOUNG GC部分的处理时间较慢。初始化标记的各类对象耗时短，初始化标记和最终标记发生时需要暂停业务线程，其这两者中间预处理标记、预处理标记可取消等阶段都是并发执行的，在最终化标记之后进行并发清除和并发重置两个阶段。

4) G1GC: G1初始化失败(大对象的出现导致内存不足)，依次执行初始标记、并发的根扫描、并行的标记、再次标记、并行的清理等阶段，由于是CMS的升级版，所以经历的阶段和CMS比较类似。

堆栈内日志分析总结 :

分为JVM内部线程和业务线程

1. 其中JVM内部线程分为VM线程、定时任务线程、GC线程、编译器线程、信号分发线程。
2. 包装类型占用内存较大， 占用16个字节(JAVA一个对象的栈用内存)，尽量用原生类型节约内存使用。

3)多维数组和较短的字符字符串都会额外浪费多余的内存。

4)内存溢出解决办法: 1)增加PermGen/Metaspace大小 2)高版本JVM调大 -XX:+CMSClassUnloadingEnabled

5)线程内存堆栈溢出解决办法:1)调整代码 2)降低XSS参数 3)调整系统线程最大参数threads-max。