背景：

印尼风控生产环境集群有三台机器

c1 ： 4g 只部署风控

c2 ： 4g 只部署风控

c3 ： 8g 部署风控+后台

————————————————————

经过：

周三（8月14日）运维说晚上c2机器报警很多，用jstat -gc pid命令看了下，fullgc次数跟耗时都挺多。决定重启一下看看情况先，重启之后，通过jstat -gcutil 1s 命令 发现过了半分钟左右fullgc又开始了，

新生代跟老年代内存也都被打满。通过jmap -histo:live pid命令（这几个命令在root用户下报错，应该使用启动jvm的用户来执行命令，app用户应该没有配置classpath找不到命令，所以用命令全路径来执行）打印类对象信息

此时日志报错 OutOfMemoryError，摘掉机器下午继续搞。

————————————————————

下午看了jvm的配置堆内存1g，-XX:PermSize=1024m，因为是1.8的jdk所以这个PermSize不起作用，怀疑有这个原因导致fullgc，于是jvm参数加上 -XX:MetaspaceSize=512m 顺便说一下，

这个参数表示堆外内存达到512m时才进行fullgc来清内存，并不表示初始堆外内存就是512m，如果不配置的话 默认是20m，由于mataspace引起的fullgc原因在gc日志里是这样的：Full GC (Metadata GC Threshold)。

同时jvm参数加上 -XX:+HeapDumpOnOutOfMemoryError -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -Xloggc:/xx/gc.log 来打印gc的日志

gc日志截图

251.836: [Full GC (Ergonomics) [PSYoungGen: 24128K->0K(280064K)] [ParOldGen: 633713K->656484K(699392K)] 657841K->656484K(979456K), [Metaspace: 104528K->104528K(1144832K)], 4.4467179 secs] [Times: user=8.61 sys=0.08, real=4.44 secs]

这里Ergonomics表示因为老年代内存不足引起的fullgc，不是堆外内存的原因。并且经过一次gc 老年代的内存用量不降反升（633713K→656484K）

想到c2 jvm配置的内存是1g，经过讨论认为这个内存实在是太小了（其实这里是走了一段弯路）于是加内存到4g，问题还是没有解决。

使用命令jmap -dump:format=b,file=heapdump.phrof pid dump内存 jstack -l pid 看堆栈，线程不是很多，也没有死锁。 关于com.dangdang的线程只有一条。

开始怀疑可能是这次上线的问题导致的，于是回滚到8月5日的版本，问题还是没有解决。

这时候，c1也出现了同样的问题，也就是说现在只有一台机器处于可用状态了。心里慌得一批。

————————————————————

解决：

接着把内存对象比较多的两个类，某个数据库查询死循环了，顺着这个思路我又到运维那执行了多次jstack，找关于dangdang的日志，发现来源都是同一个dao层接口。于是跟黄厚敏讲明情况先注释掉这段，上线后内存满的现象消失了。

```
---是这样一个sql
select count(1)
from t_gocash_core_loan l
JOIN t_gocash_core_customer c ON l.customer_id =  c.id
where c.cell_phone in (${phones})
and l.loan_status in ('CREATED', 'AUTORCPASS', 'APPROVED',
'FUNDED', 'COLLECTION', 'EXTEND')
```

————————————————————

原因：

${phones}拼接的是手机号，这个手机号拼接过程中有出现「087878811727,62817 - 6820 - 416,6282111975344」这种带'-'的数据，sharding-jdbc 1.5.4.1版本 在解析的时候死循环了，并且不断创建SQLIgnoreExpression对象，这样导致了内存打满。

运行了那么长时间都没有问题为啥这次频繁挂掉那么多机器呢？原因是内存打满这个线程也并未完成退出，而这个流程是mq的消息发起的，因为线程没有完成所以没有给mq发送ack，导致mq会把这个消息继续发给下一个消费者，因为没有一个消费者能消费成功（消费的时候就会卡死）所以这个消息会一直打挂接收它的机器。

————————————————————

经验教训：

1，要完全避免${phones}这种带$的语句

2，对用到的第三方中间件要足够了解，了解低版本的bug并及时升级

3，不能因为系统运行稳定了很长时间、中间没有新上线就以为代码没有问题

4，机器监控、日志监控要加全，并且我们能收到报警，不能只依赖运维的通知

5，对dump出来的hprof文件束手无策，没法从远程down下来，少了一种重要信息。