# sentinel排障记

生产环境的redis sentinel出错退出，重新批量加入redis集群的master节点，在批量增加的过程中，运行到async.c的如下位置产生assert的中断错误



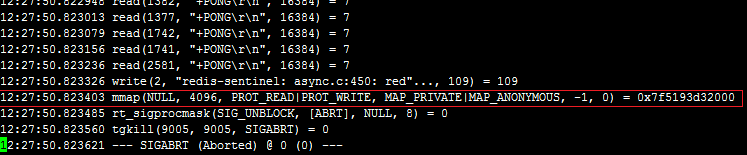
从该部分代码初步分析，sentinel服务在提供服务的过程中，出现状态不一致的现象，sentinel此时既不处于监控状态，又不处于订阅状态

## 尝试1

首先在网络上查找该错误，有人提供的解决方案为，批量增加sentinel节点的间隔时间增大。原有批量增加的时间间隔为1s，修改5秒，10秒，不论如何修改间隔时间，都会发生错误。

## 尝试2

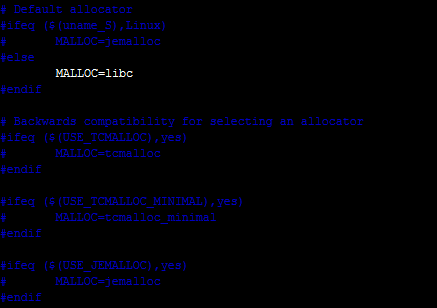
对进程进行strace的系统调用跟踪，程序崩溃前得到如下内容



多次跟踪后，都出现红框的系统调用

在程序中进行查找后，定位到redis的chunk\_mmap.c中存在该函数的调用初步怀疑是jemalloc导致的。

首先尝试修改redis的编译选项，使用libc做为内存的（修改Makefile的截图）

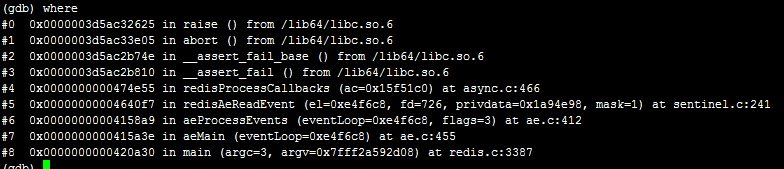


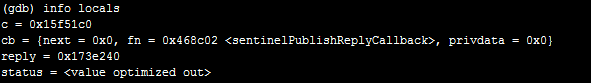
编译后，完成失效sentinel机器中一台机器的成功部署，但是在第二台机器上出现问题。尝试对jemalloc组件进行比较，确认是否为该组件不一致导致的问题，但所有机器上的组件都版本一直。所以，该方案在单台机器上成功，但还是结果还是失败

## 尝试3

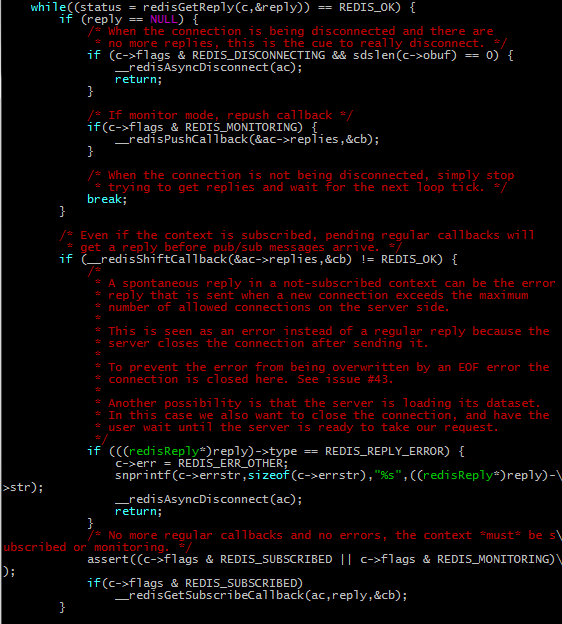
对crash程序进行core dump，记录assert fail时间点的程序运行状态，包括堆栈/变量/线程等信息，结合源码进行现场分析。

查看堆栈信息和变量信息





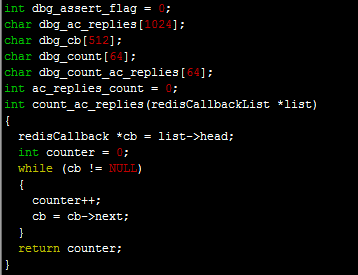
对应代码为

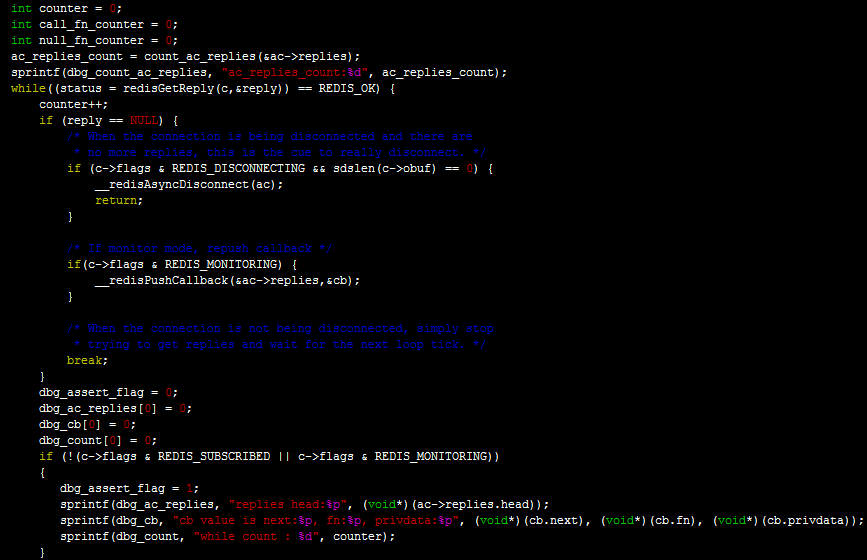


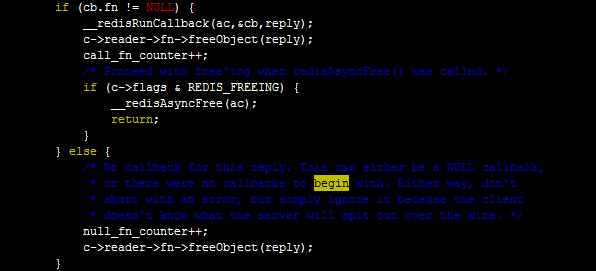
通过对该段代码的分析得知，问题的关键在于\_\_redisShiftCallback该函数返回的内容不正确，导致执行该部分代码，此时从上下文信息无法确定进入该段代码的

## 尝试4

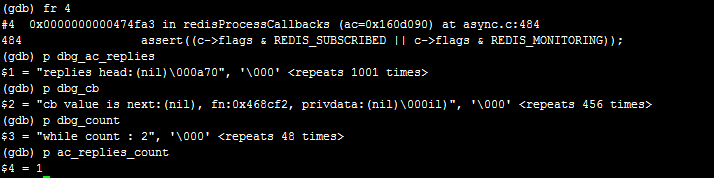
通过对尝试增加对上下文的记录信息，进行调试。代码如下



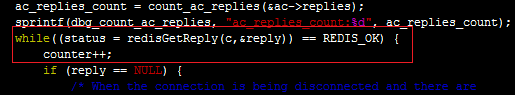




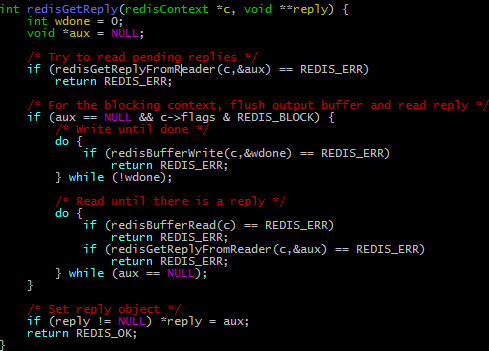
从core文件得到如下跟踪信息



根据这些变量可以得知，处理事件的回调函数只有，而函数redisGetReply却产生了两个reply消息，导致在获取第二个callback函数的时候出现错误

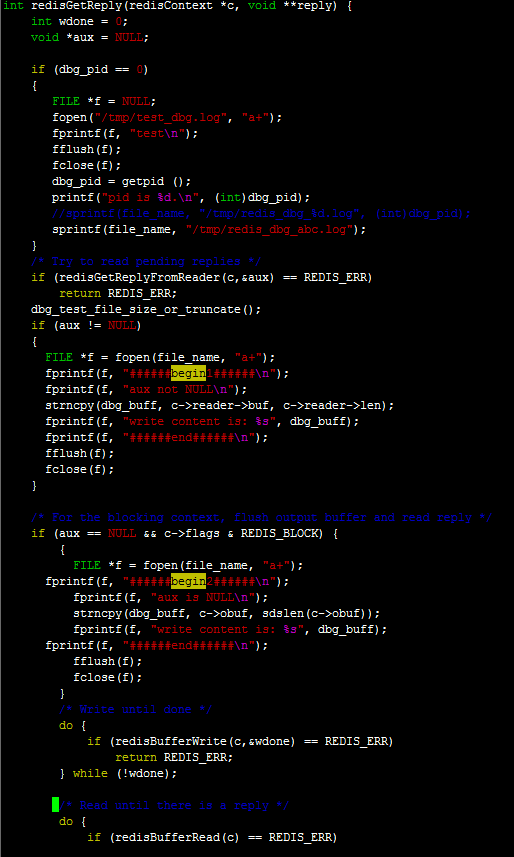


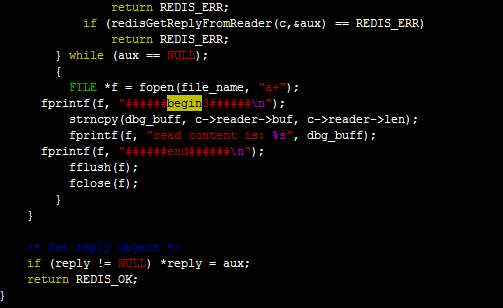
其源码如下



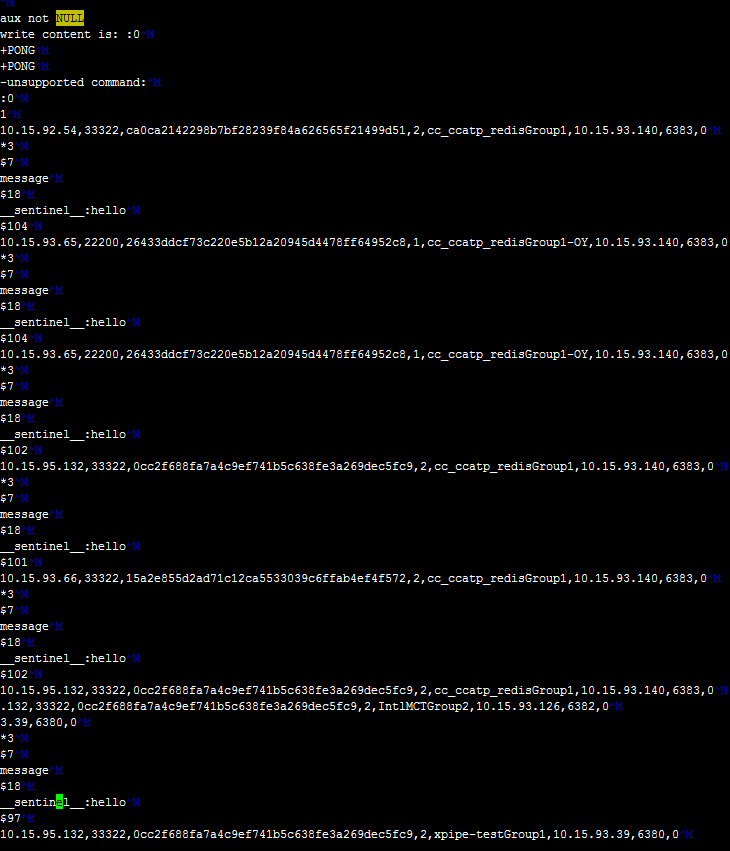
## 尝试5

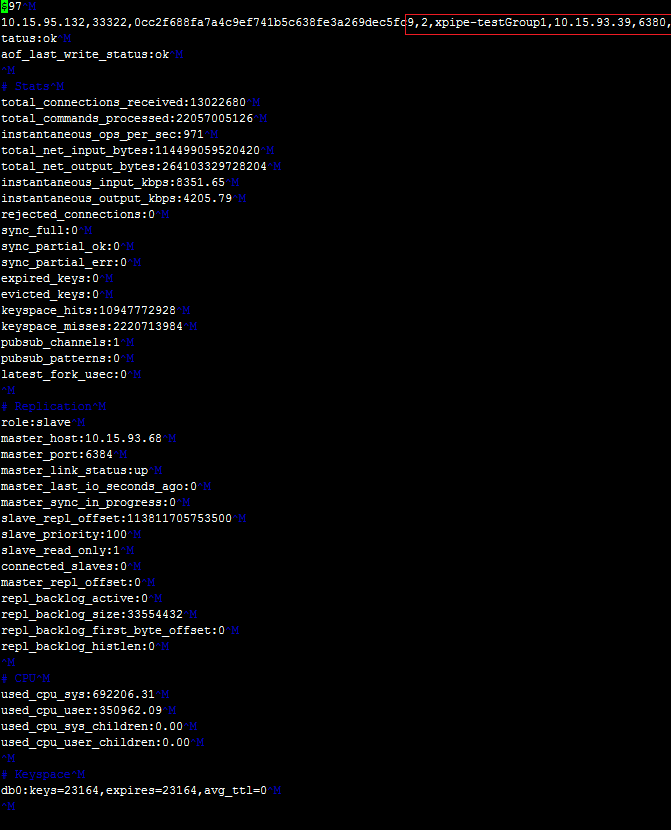
针对redisGetReply进行跟踪和分析，由源码得知该函数的功能是用于接收和发送消息，因此针对每次进行的接收和发送消息进行跟踪，并打印到文件中





出错前打印的文件的输出内容为





红框为trace中最后一个redis集群，由此怀疑为该集群出现问题，从添加监控集群列表中去除该集群，其他所有集群完成sentinel哨兵的添加。

经查看，该集群为3.0版本的，经过修改的redis服务器，消息命令的不一致导致该问题产生