Codis性能测式报告

[一、报告简述： 1](#_Toc470257136)

[基于容器的Codis架构： 2](#_Toc470257137)

[测式规模： 2](#_Toc470257138)

[二、用户认证(安全纬度) 2](#_Toc470257139)

[三、压测端数据（性能纬度）： 2](#_Toc470257140)

[benchMark to Codis QPS: 2](#_Toc470257141)

[Jetty to Codis QPS: 3](#_Toc470257142)

[四、codis端负载数据（性能纬度）： 3](#_Toc470257143)

[CPU\MEMORY\网络: 3](#_Toc470257144)

[codis 连接数： 4](#_Toc470257145)

[codis 日志： 4](#_Toc470257146)

[五、Jetty端负载数据： 4](#_Toc470257147)

[日志异常数： 4](#_Toc470257148)

[Jetty负载： 4](#_Toc470257149)

[六、故障演练（质量纬度）： 5](#_Toc470257150)

[在线扩容/缩容： 5](#_Toc470257151)

[在线模拟codis-proxy宕机 5](#_Toc470257152)

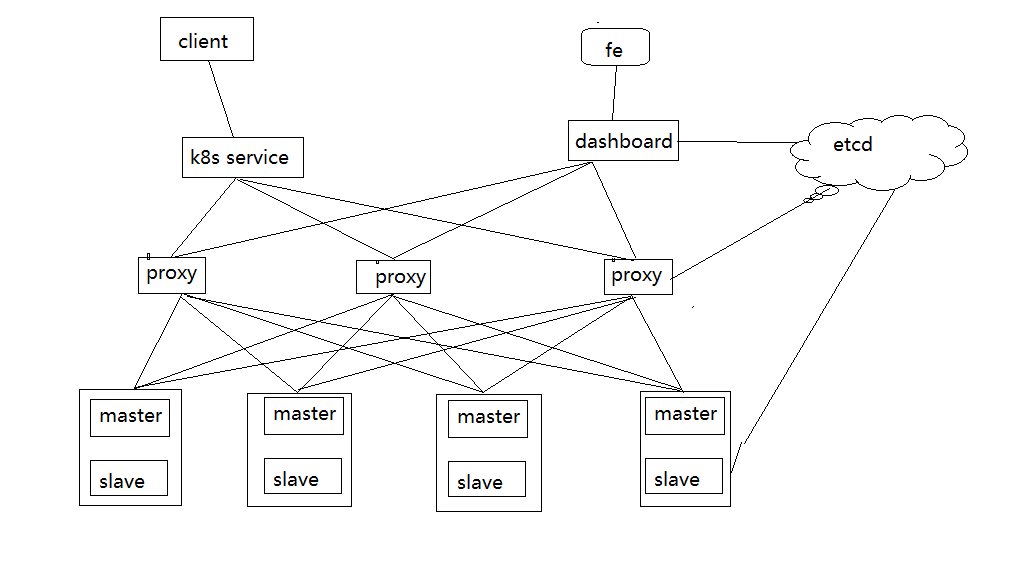
[在线模拟codis-sever宕机 6](#_Toc470257153)

[在线模拟flushall与keys \* 6](#_Toc470257154)

# 一、报告简述：

经过3周时间的测式，发现问题，修复问题，升级版本，现在codis测式工作基本完成，核心功能与性能没有问题，有些操作与维护的方式还需要优化，从结果来看，可以逐步投入线上环境使用。

## 基于容器的Codis架构：



codis基于redis研发，与传统redis的区别是扩展成了分布式架构，用户可以将codis看作一个无限大的redis缓存，可以平滑的扩容，缩容，按需分配容量，以达到节省资源，提高效率的目的，而且依托于docker容器平台的优势，量级更轻，升级，部署，自动化作业等方面都有所提升。

## 测式规模：

性能测式的节点规模为，N客户端🡪1 Codis-Proxy 🡪1 Group (1master+1slave)

质量测式的节点规模为，N客户端🡪2 Codis-Proxy 🡪2 Group (2master+2slave)

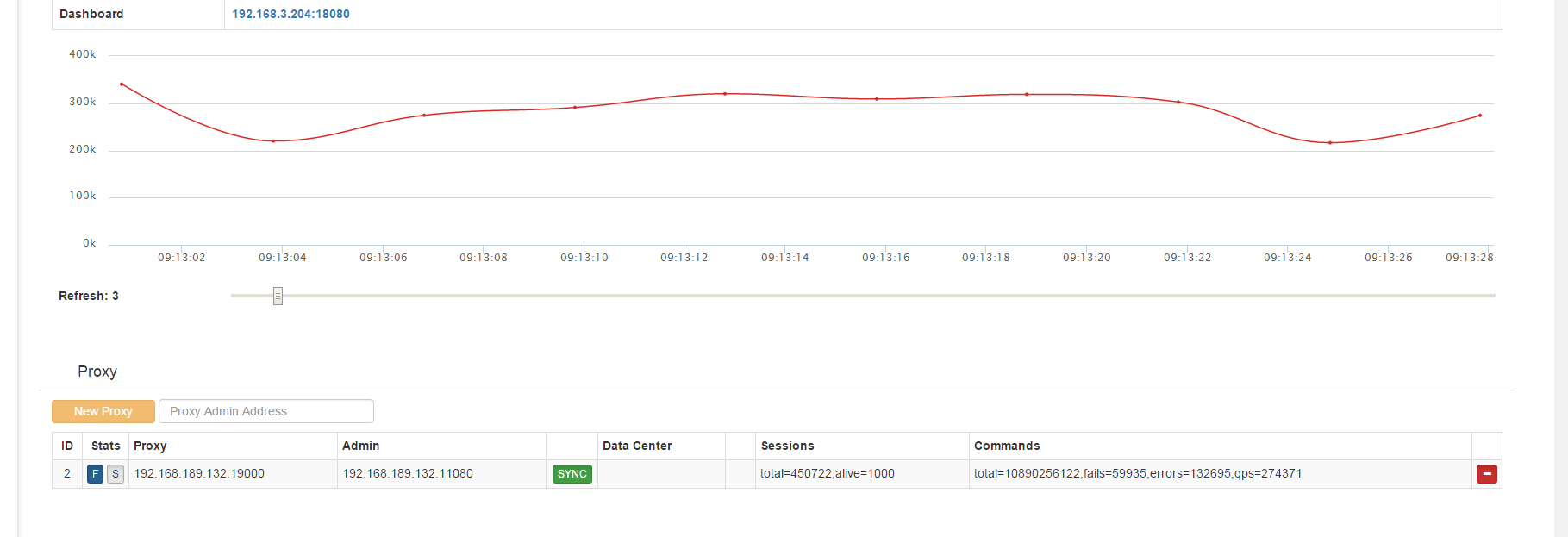
# 二、用户认证(安全纬度)

codis提供密码认证功能，全程都是基于密码认证的模式做测式，没出现相关异常，能够解决现有redis无认证的安全问题。

# 三、压测端数据（性能纬度）：

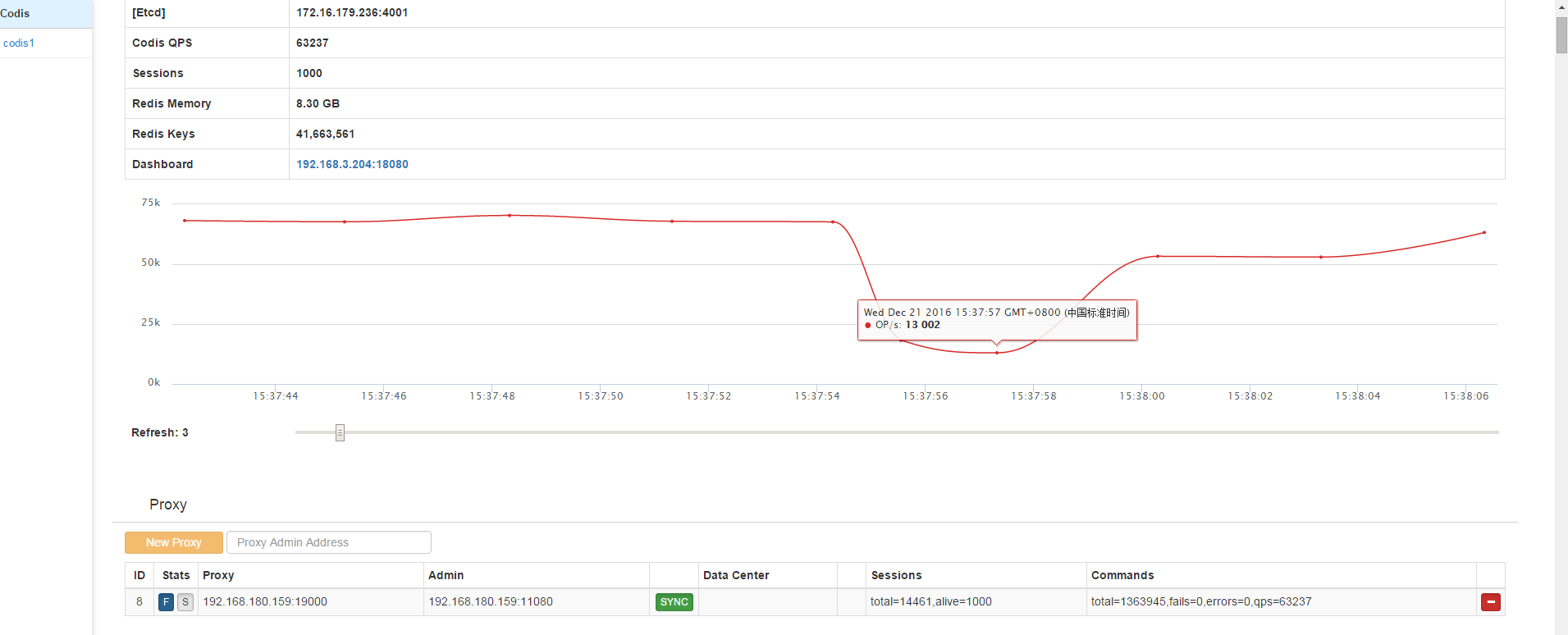
## benchMark to Codis QPS:

平均: **300K**



## Jetty to Codis QPS:

平均至少：**70K**

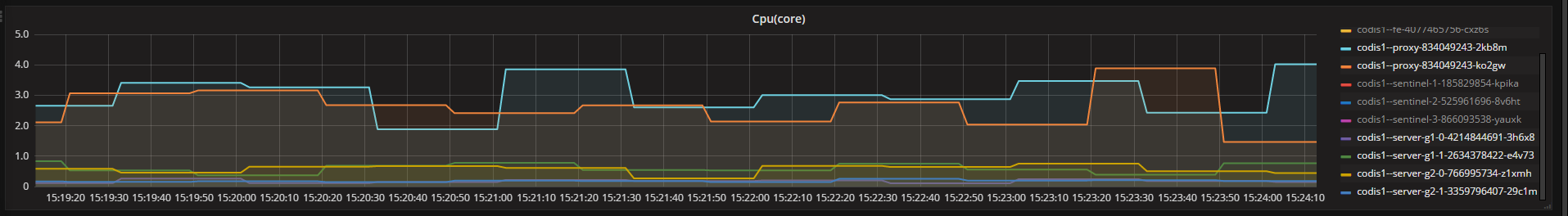


redis 性能很高，从benchmark作基准测式也可以看得出来，由于JETTY的资源有限，而且jetty自身的性能无法与redis相比，所以我们没法模拟太高的业务流量来测QPS，只是突破一下传统Twemproxy模式的40K QPS，数据表明codis集群比Twemproxy性能高很多。

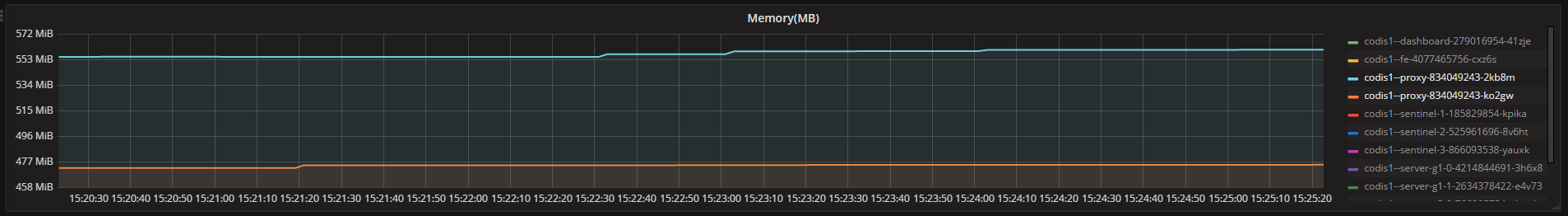
# 四、codis端负载数据（性能纬度）：

## CPU\MEMORY\网络:

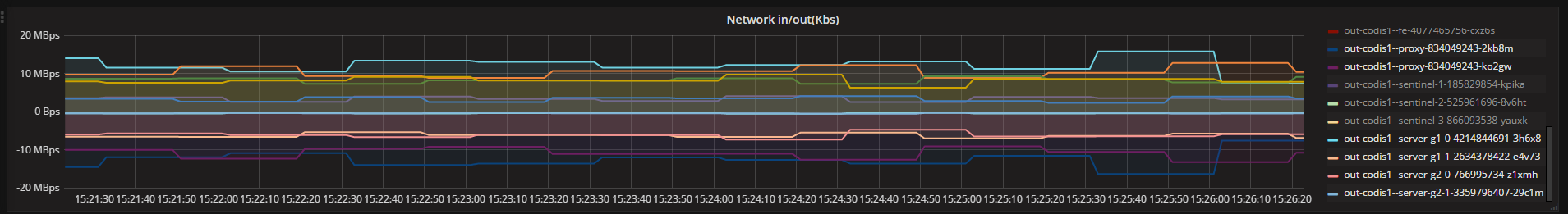
proxy and server CPU：



proxy memory:



proxy and server network:



以上结果是codis顶峰QPS时期的截图：

codis-server：1核 CPU可以满足，建议配置2核，内存建议1G起配。

codis-proxy： 4核CPU可以满足，建议配置8核，内存600M可满足，建议配置为1G。

## codis 连接数：

压测端至codisProxy的连接数2000左右。

## codis 日志：

压测过程无异常日志。

# 五、Jetty端负载数据：

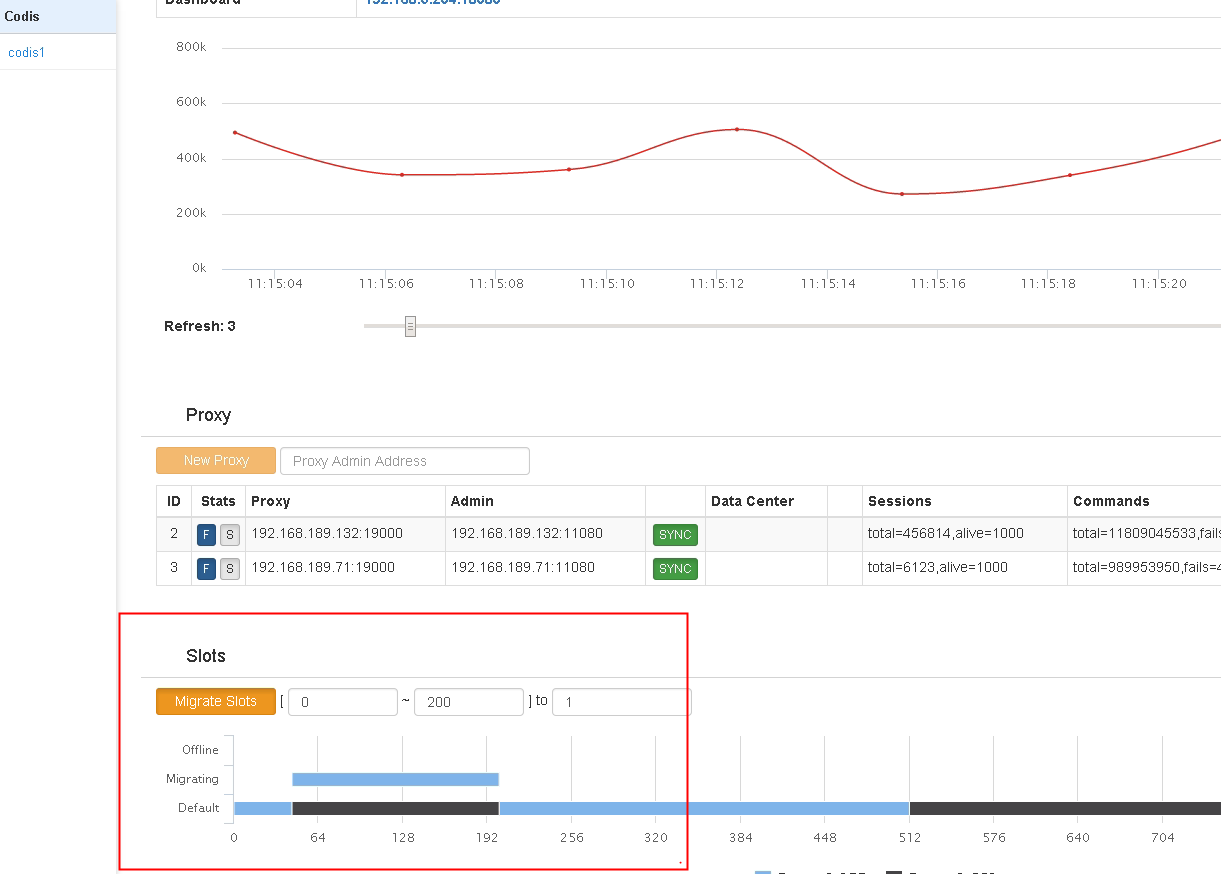
## 日志异常数：

## Jetty负载：

以上结果是codis顶峰QPS时期的截图：

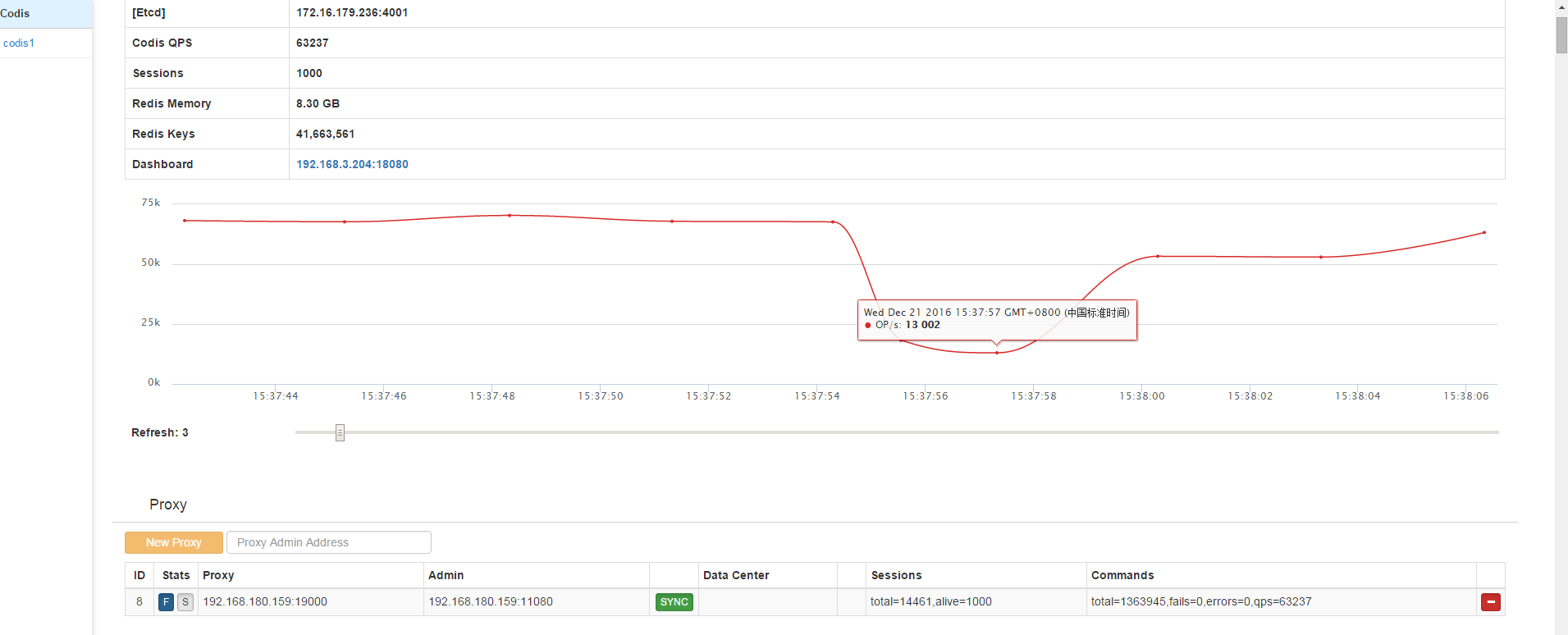
# 六、故障演练（质量纬度）：

## 在线扩容/缩容：



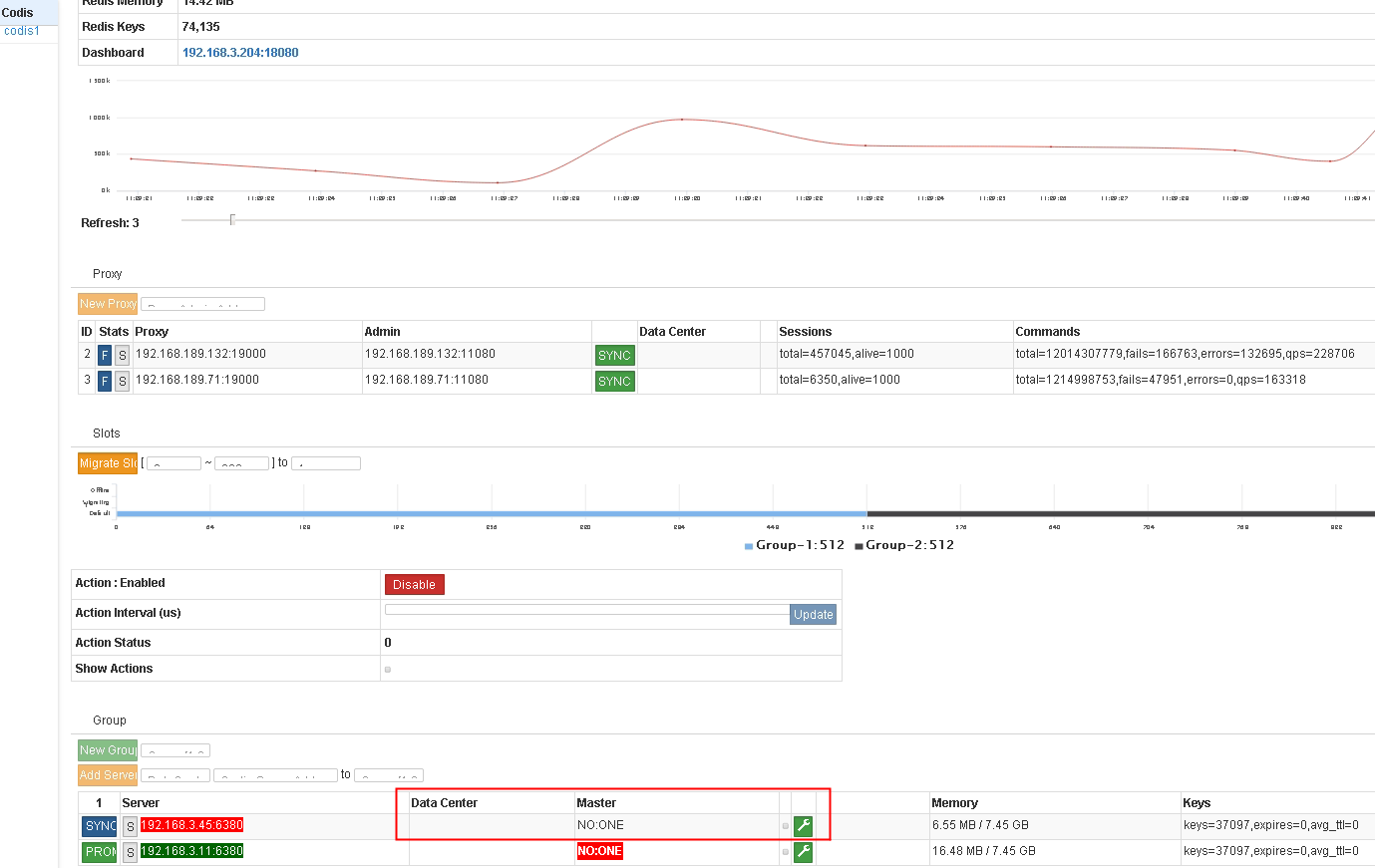
平滑扩容与与缩容，对线上业务完全无影响。

## 在线模拟codis-proxy宕机



QPS会瞬间下降一半，几秒之内，LVS会将请求转发至另一台Proxy，连接数恢复。

## 在线模拟codis-sever宕机



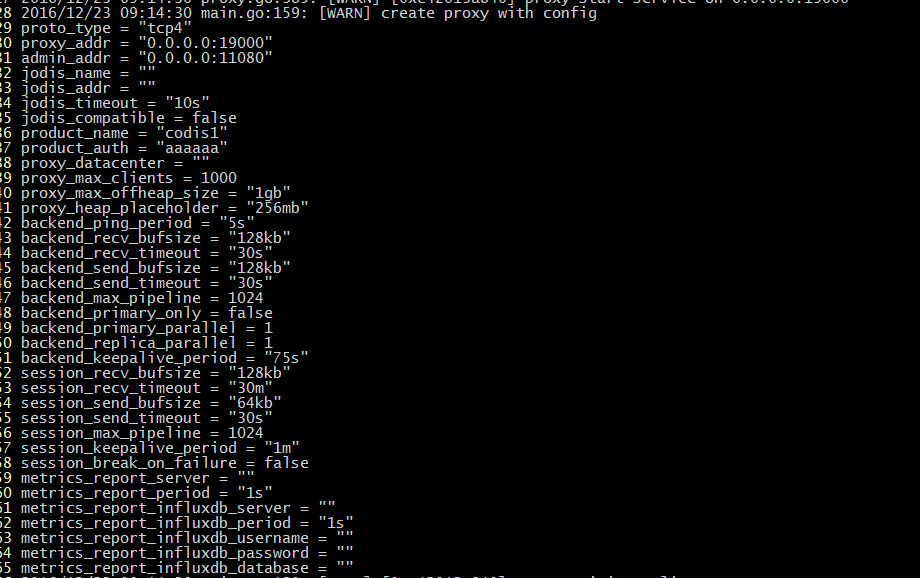
cedis-server故障能迅速转移，只不过恢复slave还需要手工介入，这个后期需要优化。

## 在线模拟flushall与keys \*

和twemproxy模式一样，这种操作在内存大于3G的时候会阻塞codis-server至无法服务，然后导致主从切换，有了高可用，这个不影响线上使用，需要flushall的时候，手动在界面切换两次集合。

问题以及解决方案：

1. codis-proxy默认的一些参数设置



默认单个proxy的连接数数1000，如果需要可以适当的调整proxy的数量。

1. 当数据的访问量过大的时候，有时候会出现slave无法挂载到master而出现slave反复同步的情况，这个由于master的输出缓冲区的设置过小导致的，因此我们需要去适当的修改输出缓冲区的大小，以后默认设置为1G

# 输出缓冲配置格式：client-output-buffer-limit <class> <hard limit> <soft limit> <soft seconds>

下面是Master上的slave客户端，默认缓冲区大小限制:当缓冲区使用超过256mb,Master会尽快杀掉它；  
当缓冲区使用大于64mb,且小于256mb的soft limit值时，并持续时间达60秒，也会被Master尽快杀掉。  
client-output-buffer-limit slave 256mb 64mb 60