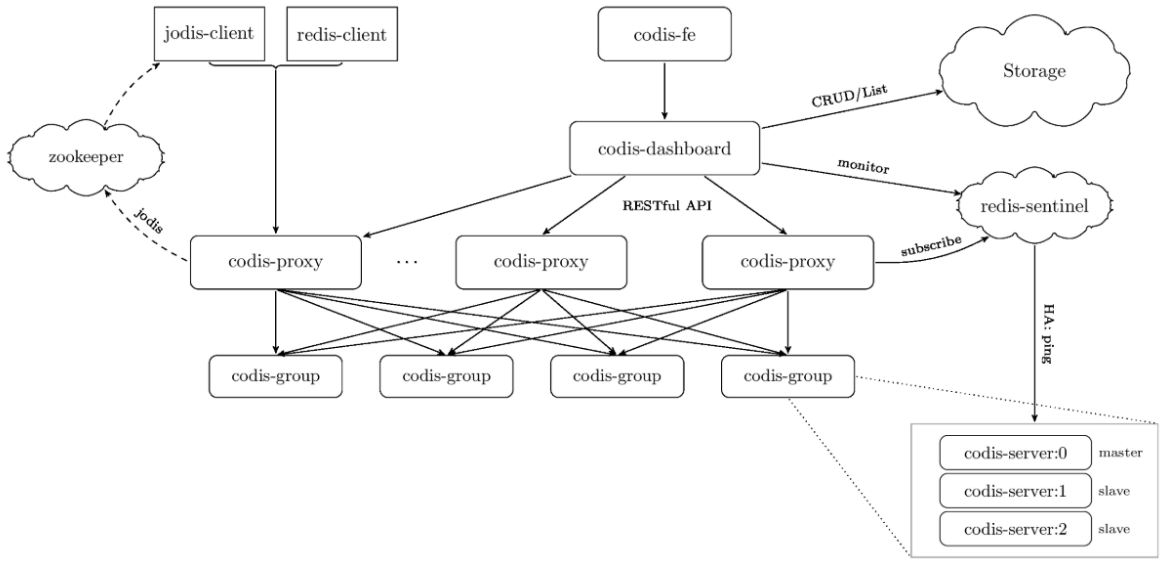
Redis加餐：Proxy端分区—Codis集群

proxy端分区

在客户端和服务器端引入一个代理或代理集群，客户端将命令发送到代理上，由代理根据算法，将命令路由到相应的服务器上。常见的代理有Codis（豌豆荚）和TwemProxy（Twitter）。

部署架构

Codis由豌豆荚于2014年11月开源，基于Go和C开发，是近期涌现的、国人开发的优秀开源软件之一。



Codis 3.x 由以下组件组成：

- **Codis Server**：基于 redis-3.2.8 分支开发。增加了额外的数据结构，以支持 slot 有关的操作以及数据迁移指令。
- **Codis Proxy**：客户端连接的 Redis 代理服务, 实现了 Redis 协议。 除部分命令不支持以外(不支持的命令列表)，表现的和原生的 Redis 没有区别（就像 Twemproxy）。
  - 对于同一个业务集群而言，可以同时部署多个 codis-proxy 实例；
  - 不同 codis-proxy 之间由 codis-dashboard 保证状态同步。
- **Codis Dashboard**：集群管理工具，支持 codis-proxy、codis-server 的添加、删除，以及据迁移等操作。在集群状态发生改变时，codis-dashboard 维护集群下所有 codis-proxy 的状态的一致性。
  - 对于同一个业务集群而言，同一个时刻 codis-dashboard 只能有 0个或者1个；
  - 所有对集群的修改都必须通过 codis-dashboard 完成。
- **Codis Admin**：集群管理的命令行工具。
  - 可用于控制 codis-proxy、codis-dashboard 状态以及访问外部存储。
- **Codis FE**：集群管理界面。
  - 多个集群实例共享可以共享同一个前端展示页面；
  - 通过配置文件管理后端 codis-dashboard 列表，配置文件可自动更新。
- **Storage**：为集群状态提供外部存储。
  - 提供 Namespace 概念，不同集群的会按照不同 product name 进行组织；

- 目前仅提供了 Zookeeper、Etcd、Fs 三种实现，但是提供了抽象的 interface 可自行扩展。

集群搭建

下载软件

```
#下载golang1.8.3  不好下载可以从本地上传 rz
wget https://storage.googleapis.com/golang/go1.8.3.linux-amd64.tar.gz
#解压
tar zxvf go1.8.3.linux-amd64.tar.gz
#下载jdk
#采用rz上传 jdk-linux-x64.tar.gz
tar -zxvf jdk-linux-x64.tar.gz
#下载zookeeper
wget https://mirrors.tuna.tsinghua.edu.cn/apache/zookeeper/zookeeper-
3.4.14/zookeeper-3.4.14.tar.gz
#解压
tar -zxvf  zookeeper-3.4.14.tar.gz
```

设置配置文件

```
vim /etc/profile
#golang
export GOROOT=/usr/go
#codis编译路径
export GOPATH=/usr/codis
#java
export JAVA_HOME=/usr/jdk1.8.0_131
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib:$CLASSPATH
export JAVA_PATH=${JAVA_HOME}/bin:${JRE_HOME}/bin
#zookeeper
export ZOOKEEPER_HOME=/usr/zookeeper-3.4.14/
#path
export PATH=$PATH:${GOROOT}/bin:${JAVA_PATH}:${ZOOKEEPER_HOME}/bin

source /etc/profile
```

设置编译环境

```
#编译目录
mkdir -p $GOPATH/src/github.com/CodisLabs

# 在编译目录下下载codis codis-release3.2.zip
cd $GOPATH/src/github.com/CodisLabs
unzip  codis-release3.2.zip
mv codis-release3.2 codis
cd codis
```

编译 Codis 源码

```
 yum install -y gcc git autoconf
 make MALLOC=libc
```

Codis-Proxy配置（zookeeper）

```
#复制zoo.cfg
cp $ZOOKEEPER_HOME/conf/zoo_sample.cfg $ZOOKEEPER_HOME/conf/zoo.cfg
#配置zoo.cfg
vim  $ZOOKEEPER_HOME/conf/zoo.cfg
#修改
dataDir=/data/zookeeper/data
autopurge.snapRetainCount=500
autopurge.purgeInterval=24
#一台zk可以不配置
#server.1=codis-1:2888:3888
#创建数据目录
mkdir -p /data/zookeeper/data
#创建myid
echo "1" > /data/zookeeper/data/myid

#编写codis的配置文件
vim /usr/codis/config.ini

#zookeeper的地址，如果是zookeeper集群，可以这么写:
zk=hostname1:2181,hostname2:2181,hostname3:2181,hostname4:2181,hostname5:2181,如果是etcd，则写成http://hostname1:port,http://hostname2:port,http://hostname3:port
zk=localhost:2181
#产品名称，这个codis集群的名字，可以认为是命名空间，不同命名空间的codis没有交集
product=test
#proxy会读取，用于标记proxy的名字，针对多个proxy的情况，可以使用不同的config.ini，只需要更改 proxy_id 即可
proxy_id=proxy_1
#检测状态时间间隔
net_timeout=5
#dashboard 服务的地址，CLI 的所有命令都依赖于 dashboard 的 RESTful API，所以必须启动
dashboard_addr=localhost:18087
#如果用etcd，则将zookeeper替换为etcd
coordinator=zookeeper
```

Codis-Server配置（Redis）

启动codis中的redis-3.2.8实例

利用admin/codis-server-admin.sh 启动

如果配置多个需要修改config/redis.conf (redis.conf 默认6379)

以一主一从为例

主机，端口6379

```
#拷贝生成redis-6379.conf
cp redis.conf redis-6379.conf
vim redis-6379.conf

pidfile /tmp/redis_6379.pid
logfile "/tmp/redis_6379.log"
dbfilename dump_6379.rdb
appendfilename "appendonly_6379.aof"
```

从机，端口6380

```
#拷贝生成redis-6380.conf
cp redis.conf redis-6380.conf
vim redis-6380.conf

port 6380
slaveof 127.0.0.1 6379
pidfile /tmp/redis_6380.pid
logfile "/tmp/redis_6380.log"
dbfilename dump_6380.rdb
appendfilename "appendonly_6380.aof"
```

修改codis-server-admin.sh

```
#拷贝生成codis-server-admin-6379.sh
cp codis-server-admin.sh codis-server-admin-6379.sh

vim codis-server-admin-6379.sh

CODIS_SERVER_PID_FILE=/tmp/redis_6379.pid

CODIS_SERVER_LOG_FILE=/tmp/redis_6379.log
CODIS_SERVER_DAEMON_FILE=$CODIS_LOG_DIR/codis-server-6379.out

CODIS_SERVER_CONF_FILE=$CODIS_CONF_DIR/redis-6379.conf

#拷贝生成codis-server-admin-6380.sh
cp codis-server-admin.sh codis-server-admin-6380.sh

vim codis-server-admin-6380.sh

CODIS_SERVER_PID_FILE=/tmp/redis_6380.pid

CODIS_SERVER_LOG_FILE=/tmp/redis_6380.log
CODIS_SERVER_DAEMON_FILE=$CODIS_LOG_DIR/codis-server-6380.out

CODIS_SERVER_CONF_FILE=$CODIS_CONF_DIR/redis-6380.conf
```

redis-sentinel配置

```
#修改sentinel.conf
protected-mode no
daemonize yes
```

注: 不需要配置监控信息，在将sentinel加入到codis集群后，同步到要监控的集群里。

启动admin下的组件

```
#启动zookeeper
zkServer.sh start
#启动Codis Dashboard
./admin/codis-dashboard-admin.sh start
#启动Codis Proxy
./admin/codis-proxy-admin.sh start
#启动Codis Server
./admin/codis-server-admin-6379.sh start
./admin/codis-server-admin-6380.sh start
#启动 Codis Fe
./admin/codis-fe-admin.sh start
#启动sentinel
./bin/redis-sentinel  ../config/sentinel.conf
```

web界面管理

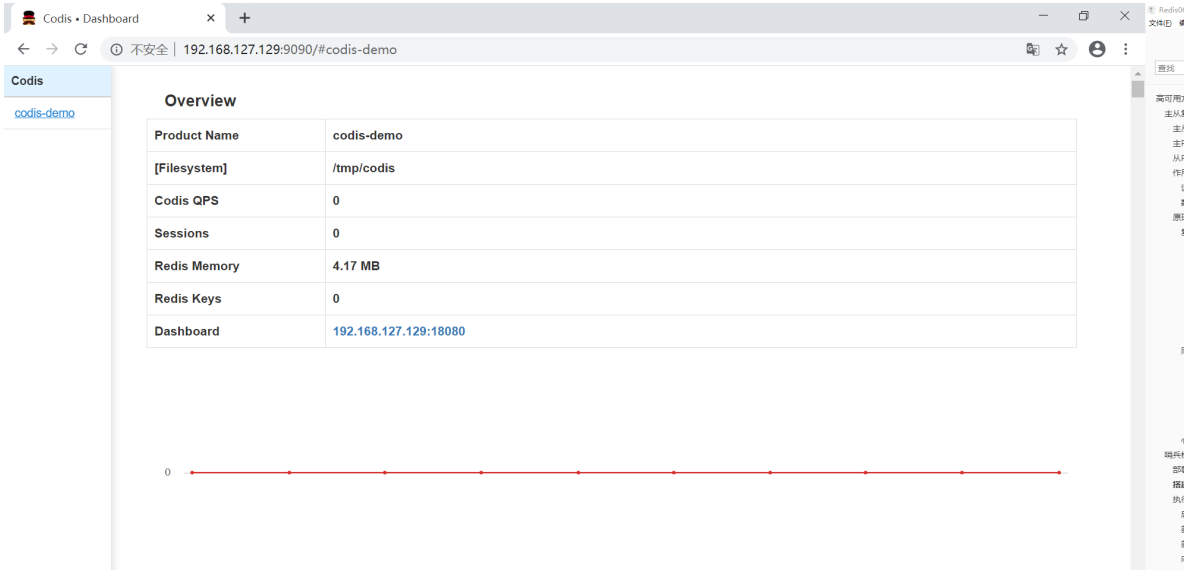
查看启动状态

```
[root@localhost bin]# ps -ef |grep codis
root       6719      1  0 01:00 pts/0    00:00:11
/usr/codis/src/github.com/CodisLabs/codis/admin/../bin/codis-server *:6379
root       6893      1  0 01:06 pts/0    00:00:39
/usr/codis/src/github.com/CodisLabs/codis/admin/../bin/codis-dashboard --
config=/usr/codis/src/github.com/CodisLabs/codis/admin/../config/dashboard.toml
--log=/usr/codis/src/github.com/CodisLabs/codis/admin/../log/codis-dashboard.log
--log-level=INFO --
pidfile=/usr/codis/src/github.com/CodisLabs/codis/admin/../bin/codis-
dashboard.pid
root       6911      1  0 01:06 pts/0    00:00:16
/usr/codis/src/github.com/CodisLabs/codis/admin/../bin/codis-proxy --
config=/usr/codis/src/github.com/CodisLabs/codis/admin/../config/proxy.toml --
dashboard=127.0.0.1:18080 --
log=/usr/codis/src/github.com/CodisLabs/codis/admin/../log/codis-proxy.log --
log-level=INFO --ncpu=4 --
pidfile=/usr/codis/src/github.com/CodisLabs/codis/admin/../bin/codis-proxy.pid
root       6924      1  0 01:06 pts/0    00:00:02
/usr/codis/src/github.com/CodisLabs/codis/admin/../bin/codis-fe --assets-
dir=/usr/codis/src/github.com/CodisLabs/codis/admin/../bin/assets --
filesystem=/tmp/codis --
log=/usr/codis/src/github.com/CodisLabs/codis/admin/../log/codis-fe.log --
pidfile=/usr/codis/src/github.com/CodisLabs/codis/admin/../bin/codis-fe.pid --
log-level=INFO --listen=0.0.0.0:9090
root       7116      1  0 01:19 pts/0    00:00:09
/usr/codis/src/github.com/CodisLabs/codis/admin/../bin/codis-server *:6380
root       8392   2200  0 03:06 pts/0    00:00:00 grep --color=auto codis

[root@localhost bin]# ps -ef |grep redis
root       7404      1  0 01:40 ?        00:00:13 ./redis-sentinel *:26379
[sentinel]
root       8477   2200  0 03:07 pts/0    00:00:00 grep --color=auto redis
```

web管理

输入：http://192.168.127.129:9090/



组添加

New Group

Group

New Group 1

Add Server Data Center Codis Server Address to Group [1,9999]

主机添加

New Group Group [1,9999]

Add Server Data Center 127.0.0.1:6379 to 1

从机添加

Group

New Group Group [1,9999]

Add Server Data Center 127.0.0.1:6380 to 1

Group

New Group Group [1,9999]

Add Server Data Center 127.0.0.1:6380 to 1

GROUPS: SYNC ALL REPLICA(S): ENABLE ALL REPLICA(S): DISABLE ALL

1	Server	Data Center	Master			Memory	Keys
SYNC	S 127.0.0.1:6379 [HA]		NO:ONE			3.92 MB / INF.	NA
PROMOTE	S 127.0.0.1:6380		127.0.0.1:6379:up			2.65 MB / INF.	NA

自动均衡slot

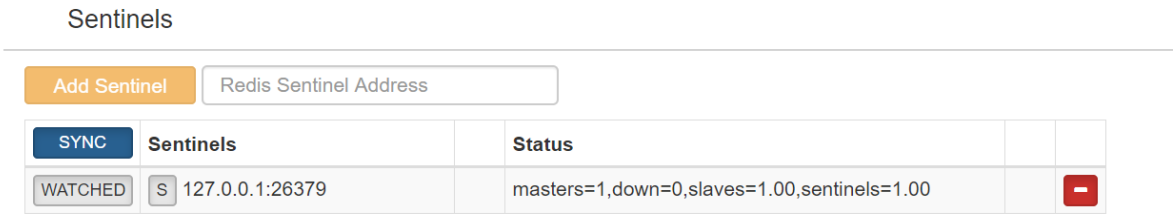
Auto-Rebalance Rebalance All Slots

哨兵添加

Sentinels

Add Sentinel 127.0.0.1:26379

点击“SYNC”后，codis将哨兵自动添加到集群中



可以通过codis客户端查看集群状态

```
[root@localhost bin]# ./redis-cli -p 19000
127.0.0.1:19000> info
# Server
redis_version:3.2.11
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:f7113f5fae59990d
redis_mode:standalone
os:Linux 3.10.0-229.el7.x86_64 x86_64
arch_bits:64
multiplexing_api:epoll
gcc_version:4.8.5
process_id:6719
run_id:1092e2ee66a31c04e680ce64f7ef43c415515ee2
tcp_port:6379
uptime_in_seconds:8608
uptime_in_days:0
hz:10
lru_clock:14812711
executable:/usr/codis/src/github.com/CodisLabs/codis/admin/../bin/codis-server
config_file:/usr/codis/src/github.com/CodisLabs/codis/admin/../config/redis-
6379.conf

# Clients
connected_clients:19
client_longest_output_list:0
client_biggest_input_buf:0
blocked_clients:0

# Memory
used_memory:4140984
used_memory_human:3.95M
used_memory_rss:5636096
used_memory_rss_human:5.38M
used_memory_peak:4698040
used_memory_peak_human:4.48M
total_system_memory:1027235840
total_system_memory_human:979.65M
used_memory_lua:37888
used_memory_lua_human:37.00K
maxmemory:0
maxmemory_human:0B
maxmemory_policy:noeviction
mem_fragmentation_ratio:1.36
```

```
mem_allocator:libc

# Persistence
loading:0
rdb_changes_since_last_save:0
rdb_bgsave_in_progress:0
rdb_last_save_time:1591863565
rdb_last_bgsave_status:ok
rdb_last_bgsave_time_sec:0
rdb_current_bgsave_time_sec:-1
aof_enabled:0
aof_rewrite_in_progress:0
aof_rewrite_scheduled:0
aof_last_rewrite_time_sec:-1
aof_current_rewrite_time_sec:-1
aof_last_bgrewrite_status:ok
aof_last_write_status:ok

# Stats
total_connections_received:56
total_commands_processed:51157
instantaneous_ops_per_sec:15
total_net_input_bytes:1448589
total_net_output_bytes:20140160
instantaneous_input_kbps:0.35
instantaneous_output_kbps:11.05
rejected_connections:0
sync_full:1
sync_partial_ok:0
sync_partial_err:0
expired_keys:0
evicted_keys:0
keyspace_hits:0
keyspace_misses:0
pubsub_channels:1
pubsub_patterns:0
latest_fork_usec:700
migrate_cached_sockets:0

# Replication
role:master
connected_slaves:1
slave0:ip=127.0.0.1,port=6380,state=online,offset=408626,lag=0
master_repl_offset:408626
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:2
repl_backlog_histlen:408625

# CPU
used_cpu_sys:6.45
used_cpu_user:6.72
used_cpu_sys_children:0.00
used_cpu_user_children:0.00

# Cluster
cluster_enabled:0
```

```
# Keyspace
```

分片原理

Codis 将所有的 key 默认划分为 1024 个槽位(slot)，它首先对客户端传过来的 key 进行 crc32 运算计算哈希值，再将 hash 后的整数值对 1024 这个整数进行取模得到一个余数，这个余数就是对应 key 的槽位。

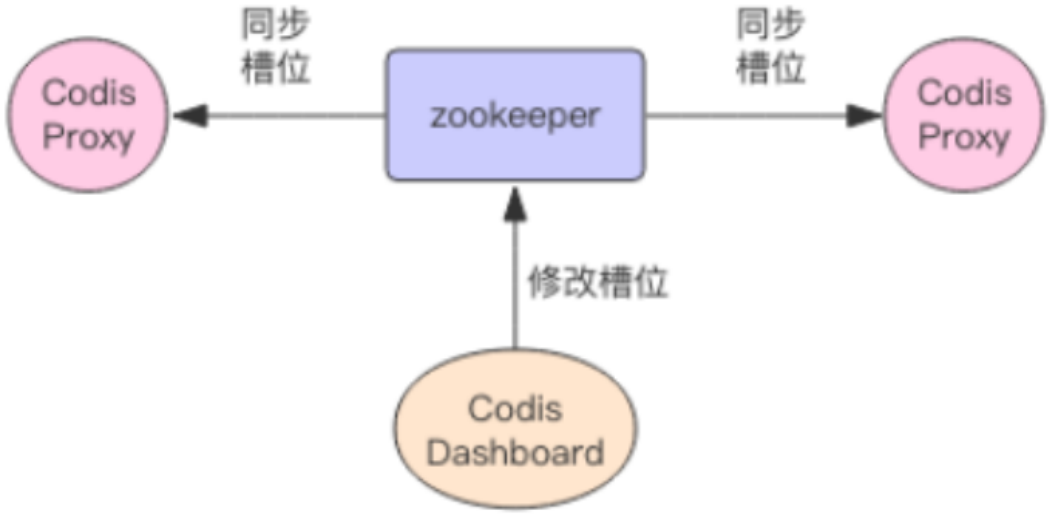
Codis的槽位和分组的映射关系就保存在codis proxy当中。

实例之间槽位同步

codis proxy存在单点问题，需要做集群。

不同proxy之间需要同步映射关系

在Codis中使用的是Zookeeper（Etcd）来保存映射关系



Codis 将槽位关系存储在 zk 中，并且提供了一个 Dashboard 可以用来观察和修改槽位关系，当槽位关系变化时，Codis Proxy 会监听到变化并重新同步槽位关系，从而实现多个 Codis Proxy 之间共享相同的槽位关系配置。

扩容&自动均衡

新增一组redis（一主一从）

Codis-Server配置（Redis）

启动codis中的redis-3.2.8实例

利用admin/codis-server-admin.sh 启动

如果配置多个需要修改config/redis.conf (redis.conf 默认6379)

以一主一从为例

主机，端口6381

```
#拷贝生成redis-6381.conf
cp redis.conf redis-6381.conf
vim redis-6381.conf

pidfile /tmp/redis_6381.pid
logfile "/tmp/redis_6381.log"
dbfilename dump_6381.rdb
appendfilename "appendonly_6381.aof"
```

从机，端口6382

```
#拷贝生成redis-6382.conf
cp redis.conf redis-6382.conf
vim redis-6382.conf

port 6382
slaveof 127.0.0.1 6381
pidfile /tmp/redis_6382.pid
logfile "/tmp/redis_6382.log"
dbfilename dump_6382.rdb
appendfilename "appendonly_6382.aof"
```

修改codis-server-admin.sh

```
#拷贝生成codis-server-admin-6381.sh
cp codis-server-admin.sh codis-server-admin-6381.sh

vim codis-server-admin-6381.sh

CODIS_SERVER_PID_FILE=/tmp/redis_6381.pid

CODIS_SERVER_LOG_FILE=/tmp/redis_6381.log
CODIS_SERVER_DAEMON_FILE=$CODIS_LOG_DIR/codis-server-6381.out

CODIS_SERVER_CONF_FILE=$CODIS_CONF_DIR/redis-6381.conf

#拷贝生成codis-server-admin-6382.sh
cp codis-server-admin.sh codis-server-admin-6382.sh

vim codis-server-admin-6382.sh

CODIS_SERVER_PID_FILE=/tmp/redis_6382.pid

CODIS_SERVER_LOG_FILE=/tmp/redis_6382.log
```

```
CODIS_SERVER_DAEMON_FILE=$CODIS_LOG_DIR/codis-server-6382.out

CODIS_SERVER_CONF_FILE=$CODIS_CONF_DIR/redis-6382.conf
```

启动Codis Server

```
#启动Codis Server
./admin/codis-server-admin-6381.sh start
./admin/codis-server-admin-6382.sh start
```

通过web管理

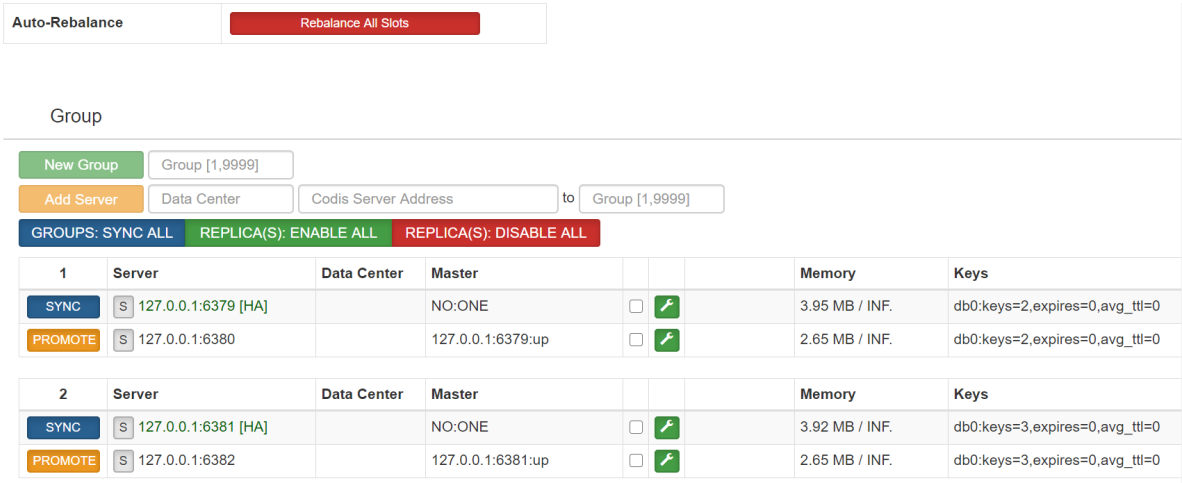
添加新组、添加主机、添加从机后

点击“Rebalance All Slots”

通过codis客户端添加数据

```
[root@localhost bin]# ./redis-cli -p 19000
127.0.0.1:19000> set cn3 zhaoyun3
OK
127.0.0.1:19000> set name:001 zhaoyun3
OK
127.0.0.1:19000> set name:002 zhaoyun3
OK
```

可以看到数据分区



优点&缺点

优点

- 对客户端透明,与codis交互方式和redis本身交互一样
- 支持在线数据迁移,迁移过程对客户端透明有简单的管理和监控界面
- 支持高可用,无论是redis数据存储还是代理节点
- 自动进行数据的均衡分配
- 最大支持1024个redis实例,存储容量海量
- 高性能

缺点

- 采用自有的redis分支,不能与原版的redis保持同步

- 如果codis的proxy只有一个的情况下, redis的性能会下降20%左右
- 某些命令不支持