**基于codis的redis分布式缓存实现方案**

Codis3.x有如下特点：

1.最新 release 版本为 codis-3.2，codis-server 基于 redis-3.2.8

2.支持 slot 同步迁移、异步迁移和并发迁移，对 key 大小无任何限制，迁移性能大幅度提升

3.相比 2.0：重构了整个集群组件通信方式，codis-proxy 与 zookeeper 实现了解耦，废弃了codis-config 等

4.元数据存储支持 etcd/zookeeper/filesystem 等，可自行扩展支持新的存储，集群正常运行期间，即便元存储故障也不再影响 codis 集群，大大提升 codis-proxy 稳定性

5.对 codis-proxy 进行了大量性能优化,通过控制GC频率、减少对象创建、内存预分配、引入 cgo、jemalloc 等，使其吞吐还是延迟，都已达到 codis 项目中最佳

6.proxy 实现 select 命令，支持多 DB

7.proxy 支持读写分离、优先读同 IP/同 DC 下副本功能

8.基于 redis-sentinel 实现主备自动切换

9.实现动态 pipeline 缓存区（减少内存分配以及所引起的 GC 问题）

10.proxy 支持通过 HTTP 请求实时获取 runtime metrics，便于监控、运维

11.支持通过 influxdb 和 statsd 采集 proxy metrics

12.slot auto rebalance 算法从 2.0 的基于 max memory policy 变更成基于 group 下 slot 数量

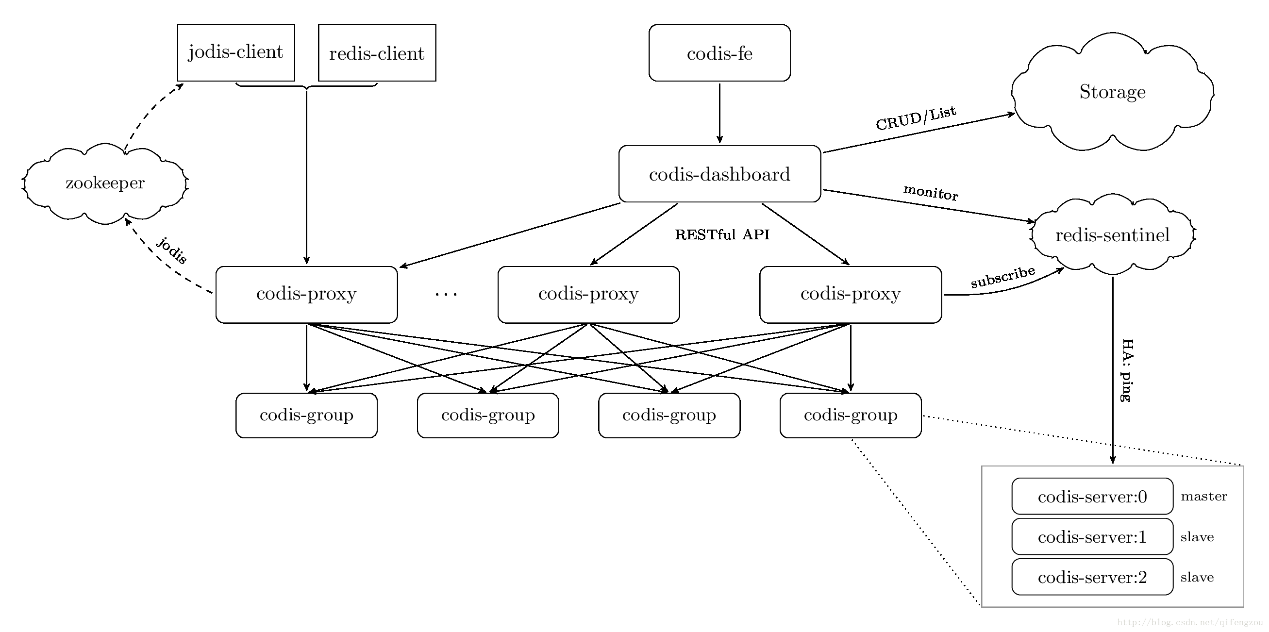
13.提供了更加友好的 dashboard 和 fe 界面，新增了很多按钮、跳转链接、错误状态等，有利于快速发现、处理集群故障

14.新增 SLOTSSCAN 指令，便于获取集群各个 slot 下的所有 key

15.codis-proxy 与 codis-dashbaord 支持 docker 部署

**1 大体架构**

**1.1 总体架构**



Codis 3.x 由以下组件组成：

1.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。具体的修改可以参考文档 redis 的修改。

2.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。

3.Redis sentinel：Redis官方推荐的高可用性(HA)解决方案。它可以实现对Redis的监控、通知、自动故障转移。如果Master不能工作，则会自动启动故障转移进程，将其中的一个Slave提升为Master，其他的Slave重新设置新的Master服务。

4.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。

5.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。

6.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。

7.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、filesystem三种实现，但是提供了抽象的 interface 可自行扩展。

**1.2 部署规划**

| **序号** | **IP** | **主机名** | **部署程序** |
| --- | --- | --- | --- |
| 01 | 192.168.1.11 | WebServer-11 | codis-server:(6379&6380) |
| 02 | 192.168.1.12 | WebServer-12 | codis-server:(6379&6380) |
| 03 | 192.168.1.13 | WebServer-13 | codis-server:(6379&6380) |
| 04 | 192.168.1.14 | WebServer-14 | codis-server:(6379&6380) |
| 05 | 192.168.1.15 | WebServer-15 | codis-server:(6379&6380) |
| 06 | 192.168.1.21 | WebServer-21 | codis-proxy:19000 |
| 07 | 192.168.1.22 | WebServer-22 | codis-proxy:19000 |
| 08 | 192.168.1.31 | WebServer-31 | codis-dashborad:18080、codis-fe:18090 |
| 09 | 192.168.1.41 | WebServer-41 | redis-sentinel:26379 |
| 10 | 192.168.1.42 | WebServer-42 | redis-sentinel:26379 |
| 11 | 192.168.1.43 | WebServer-43 | redis-sentinel:26379 |
| 12 | 192.168.1.51 | WebServer-51 | zookeeper:2181 |
| 13 | 192.168.1.52 | WebServer-52 | zookeeper:2181 |
| 14 | 192.168.1.53 | WebServer-53 | zookeeper:2181 |

**2 部署流程**

**2.1 安装ZK**

**作用**：用于存放数据路由表。   
**描述**：zookeeper简称zk。在生产环境中，zk部署越多，其可靠性越高。由于zk集群是以宕机个数过半才会让整个集群宕机，因此，**奇数**个zk更佳。   
**部署**：按照1.2中的部署规划，将在如下几台机器上部署该程序。

| **序号** | **IP** | **主机名** | **部署程序** |
| --- | --- | --- | --- |
| 01 | 192.168.1.51 | WebServer-51 | zookeeper:2181 |
| 02 | 192.168.1.52 | WebServer-52 | zookeeper:2181 |
| 03 | 192.168.1.53 | WebServer-53 | zookeeper:2181 |

**步骤**：   
***1. 下载源码***   
下载源代码，并解压至/usr/local/zookeeper目录。

下载路径：<http://zookeeper.apache.org/releases.html#download>。   
注意事项：请勿yum -y install zookeeper。可能出现各种莫名其妙得问题，以致zk始终无法正常启动。

***2. 安装java包***

#yum -y install java # 安装java包

***3. 配置程序***   
第一步：添加域名   
编辑/etc/hosts文件，并添加以下配置。[注：在192.168.1.51~53上都**完全一样**]

#vim /etc/hosts #添加域名

192.168.1.51 zookeeper-node1   
192.168.1.52 zookeeper-node2   
192.168.1.53 zookeeper-node3

第二步：修改zk配置   
编辑配置文件，并添加以下配置：[注：在192.168.1.51~53上都**完全一样**]

#cd /usr/local/zookeeper #安装目录   
#vim ./conf/zoo.cfg #编辑配置

maxClientCnxns=50 #最大连接数设置. 注：可不配置.   
tickTime=2000 #一个周期(tick)的时长(单位：毫秒). 注：可用默认值   
initLimit=10 #初始化同步阶段最多耗费tick个数. 注：可用默认值   
syncLimit=5 #等待应答的最大间隔tick个数. 注：可用默认值   
dataDir=/data/zookeeper/ #数据存储目录. 注：勿放在/tmp目录   
clientPort=2181 #帧听端口. 注：可用默认值   
server.1=zookeeper-node1:2888:3888   
server.2=zookeeper-node2:2888:3888   
server.3=zookeeper-node3:2888:3888

说明：server.A=B：C：D：其中 A 是一个数字，表示这个是第几号服务器；B 是这个服务器的 ip 地址；C 表示的是这个服务器与集群中的 Leader 服务器交换信息的端口；D 表示的是万一集群中的 Leader 服务器挂了，需要一个端口来重新进行选举，选出一个新的 Leader，而这个端口就是用来执行选举时服务器相互通信的端口。如果是伪集群的配置方式，由于 B 都是一样，所以不同的 Zookeeper 实例通信端口号不能一样，所以要给它们分配不同的端口号。

第三步：其他处理   
创建第二步中的dataDir目录，并设置当前zk的结点ID。[注：在192.168.1.51~53上ID值**各不相同**]

#mkdir -p /data/zookeeper #创建zk数据目录(datadir)   
#echo “1” > /data/zookeeper/myid #生成ID，这里需要注意， myid对应的zoo.cfg的server.ID.比如zookeeper-node2对应的myid应该是2.   
/usr/lib/zookeeper/bin/Server.sh start #服务启动

***4. 启动程序***   
需要在192.168.1.51~53上依次执行以下命令。

#cd /usr/local/zookeeper/ #安装目录

| **序号** | **IP** | **主机名** | **部署程序** |
| --- | --- | --- | --- |
| 01 | 192.168.1.11 | WebServer-11 | codis-server:(6379&6380) |
| 02 | 192.168.1.12 | WebServer-12 | codis-server:(6379&6380) |
| 03 | 192.168.1.13 | WebServer-13 | codis-server:(6379&6380) |
| 04 | 192.168.1.14 | WebServer-14 | codis-server:(6379&6380) |
| 05 | 192.168.1.15 | WebServer-15 | codis-server:(6379&6380) |
| 06 | 192.168.1.21 | WebServer-21 | codis-proxy:19000 |
| 07 | 192.168.1.22 | WebServer-22 | codis-proxy:19000 |
| 08 | 192.168.1.31 | WebServer-31 | codis-dashborad:18080、codis-fe:18090 |
| 09 | 192.168.1.41 | WebServer-41 | redis-sentinel:6379 |
| 10 | 192.168.1.42 | WebServer-42 | redis-sentinel:6379 |
| 11 | 192.168.1.43 | WebServer-43 | redis-sentinel:6379 |

#./bin/Server.sh start #服务启动

**2.2 编译Codis**

请在以下机器列表上按步骤[2.2.1~2.2.3]安装codis环境：

**步骤**：   
***1. 安装go环境***   
先从官网(<https://golang.org/dl/>)下载golang安装包，并将其解压，再拷贝至/usr/local/go/中，最后配置如下环境变量。

#vim $HOME/.bashrc

export GOROOT=/usr/local/go # 安装路径   
export GOPATH=$HOME/godir # 工作路径   
export PATH=$PATH:$GOPATH/bin:$GOROOT/bin # 命令搜索路径

***2. 下载codis源码***   
需要在如下每台机器上下载codis源码，源码下载命令为:

#go get github.com/CodisLabs/codis.git -b release3.2

***3. 编译codis源码***

#cd $GOPATH/src/github.com/CodisLabs/ # 源码目录

#make # 执行编译

#ls ./bin/ # 查看结果   
codis-admin codis-dashboard codis-fe codis-ha codis-proxy codis-server redis-benchmark redis-cli

完成编译后，将会在bin目录中看到codis-admin、codis-dashboard、codis-fe、codis-ha、codis-proxy、codis-server六个可执行文件。另外，bin/assert文件夹是codis-dashboard的http服务需要的前端资源，其需要和codis-dashboard放置在同一个文件夹中。   
补充：在目录./extern/redis-3.2.8/src/中可以找到redis-sentinel可执行文件，其将会用于集群主从的切换。

***4. 拷贝codis程序***

#sudo mkdir -p /usr/local/codis/bin

#cp -fr $GOPATH/github.com/CodisLabs/codis/bin/\* /usr/local/codis/bin/   
#cp -fr $GOPATH/github.com/CodisLabs/codis/conf/\* /usr/local/codis/conf/

**2.3 Codis-server**

**作用**：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。具体的修改可以参考文档 redis 的修改。

**部署**：按照1.2中的部署规划，将在如下机器上部署该程序。

| **序号** | **IP** | **主机名** | **部署程序** |
| --- | --- | --- | --- |
| 01 | 192.168.1.11 | WebServer-11 | codis-server:(6379&6380) |
| 02 | 192.168.1.12 | WebServer-12 | codis-server:(6379&6380) |
| 03 | 192.168.1.13 | WebServer-13 | codis-server:(6379&6380) |
| 04 | 192.168.1.14 | WebServer-14 | codis-server:(6379&6380) |
| 05 | 192.168.1.15 | WebServer-15 | codis-server:(6379&6380) |

**步骤：**   
***1. 修改主配置***

#cd /usr/local/codis/conf/

#cp redis.conf redis-6379.conf # 主配置

#vim redis-6379.conf # 修改配置

daemonize yes   
pidfile /usr/loca/codis/proc/redis-6379.pid # 进程ID文件路径   
port 6379 # 绑定端口   
timeout 86400   
tcp-keepalive 60   
loglevel notice   
logfile /usr/local/codis/log/redis-6379.log # 日志文件路径   
databases 16   
save “”   
save 900 1   
save 300 10   
save 60 10000   
stop-writes-on-bgsave-error no   
rdbcompression yes   
dbfilename dump-6379.rdb # dump文件   
dir /usr/local/codis/data/redis\_data\_6379 # dump路径   
masterauth "123456" # Master密码（从主同步密码）   
slave-serve-stale-data yes   
repl-disable-tcp-nodelay no   
slave-priority 100   
requirepass "123456" # 鉴权密码（客户端连接密码）   
maxmemory 10gb   
maxmemory-policy allkeys-lru   
appendonly no   
appendfsync everysec   
no-appendfsync-on-rewrite yes   
auto-aof-rewrite-percentage 100   
auto-aof-rewrite-min-size 64mblua-time-limit 5000   
slowlog-log-slower-than 10000   
slowlog-max-len 128   
hash-max-ziplist-entries 512   
hash-max-ziplist-value 64   
list-max-ziplist-entries 512   
list-max-ziplist-value 64   
set-max-intset-entries 512   
zset-max-ziplist-entries 128   
zset-max-ziplist-value 64   
client-output-buffer-limit normal 0 0 0   
client-output-buffer-limit slave 0 0 0   
client-output-buffer-limit pubsub 0 0 0   
hz 10   
aof-rewrite-incremental-fsync yes   
repl-backlog-size 33554432

***2. 修改从配置***

#cd /usr/local/codis/conf/

#cp redis.conf redis-6380.conf # 主配置

#vim redis-6380.conf # 修改配置

daemonize yes   
pidfile /usr/loca/codis/proc/redis-6380.pid # 进程ID文件路径   
port 6380 # 绑定端口   
timeout 86400   
tcp-keepalive 60   
loglevel notice   
logfile /usr/local/codis/log/redis-6380.log # 日志文件路径   
databases 16   
save “” # 如果不希望存储到磁盘，则可以使用井号注销save配置行   
save 900 1 # 如果不希望存储到磁盘，则可以使用井号注销save配置行   
save 300 10 # 如果不希望存储到磁盘，则可以使用井号注销save配置行   
save 60 10000 # 如果不希望存储到磁盘，则可以使用井号注销save配置行   
stop-writes-on-bgsave-error no   
rdbcompression yes   
dbfilename dump-6380.rdb # dump文件   
dir /usr/local/codis/data/redis\_data\_6380 # dump路径   
masterauth "123456" # Master密码（适合主从集群）   
slave-serve-stale-data yes   
repl-disable-tcp-nodelay no   
slave-priority 100   
requirepass "123456" # 鉴权密码（客户端连接密码）   
maxmemory 10gb   
maxmemory-policy allkeys-lru   
appendonly no   
appendfsync everysec   
no-appendfsync-on-rewrite yes   
auto-aof-rewrite-percentage 100   
auto-aof-rewrite-min-size 64mblua-time-limit 5000   
slowlog-log-slower-than 10000   
slowlog-max-len 128   
hash-max-ziplist-entries 512   
hash-max-ziplist-value 64   
list-max-ziplist-entries 512   
list-max-ziplist-value 64   
set-max-intset-entries 512   
zset-max-ziplist-entries 128   
zset-max-ziplist-value 64   
client-output-buffer-limit normal 0 0 0   
client-output-buffer-limit slave 0 0 0   
client-output-buffer-limit pubsub 0 0 0   
hz 10   
aof-rewrite-incremental-fsync yes   
repl-backlog-size 33554432

***3. 启动程序***

#cd /usr/local/codis/bin/   
#./codis-server ../conf/redis-6379.conf & #启动主程序   
#./codis-server ../conf/redis-6380.conf & #启动从程序

所有机器完成以上五个步骤后，便完成了对codis-server的启动。

**2.4 Codis-dashboard**

**作用**：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。

1.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；   
2.所有对集群的修改都必须通过 codis-dashboard 完成。

**部署**：按照1.2中的部署规划，将在如下机器上部署该程序。

| **序号** | **IP** | **主机名** | **部署程序** |
| --- | --- | --- | --- |
| 08 | 192.168.1.31 | WebServer-31 | codis-dashborad:18080、codis-fe:18090 |

**步骤：**   
***1. 修改配置***

#cd /usr/local/codis/bin/   
#./codis-dashboard - -default-conifg | tee ../conf/dashboard.conf #生成配置   
#vim ../conf/dashboard.conf # 修改配置

coordinator\_name = "zookeeper" # 外部存储类型   
coordinator\_addr = "192.168.1.51:2181,192.168.1.52:2181,192.168.1.53:2181" # 外部存储IP列表

product\_name = "chatroom" # 项目名称   
product\_auth = “123456” # 集群密码（注意:需要与redis配置中的requirepass保持一致）

admin\_addr = "0.0.0.0:18080" # RESTful API 端口

为了防止出现dashboard监控页面中OPS始终为0的现象，需要将各proxy的IP和主机名写到hosts文件中。

#vim /etc/hosts # 添加域名

192.168.1.21 WebSocket-21   
192.168.1.22 WebSocket22

***2. 启动程序***

#cd ./bin/   
#nohup ./codis-dashboard - -ncpu=24 - -config=/usr/local/codis/conf/dashboard.conf - -log=/usr/local/codis/log/dashboard.log - -log-level=WARN & #启动程序(注意：使用绝对路径）

参数描述如下：

| **序号** | **参数** | **描述** | **备注** |
| --- | --- | --- | --- |
| 01 | - -ncpu | 最大使用CPU个数 |  |
| 02 | - -config | 指定配置路径和文件 | 使用绝对路径 |
| 03 | - -log | 指定日志输出路径和文件 | 使用绝对路径 |
| 02 | - -log-level | 指定日志级别 | 取值：INFO、WARN、DEBUG、ERROR，推荐使用WRAN. |

完成以上2个步骤后，便完成了对codis-config的安装。   
***3. 关闭程序***   
如果想关闭dashboard服务，可执行：

#./codis-admin - -dashboard=192.168.1.31:18080 –auth=123456 - -shutdown

**2.5 Codis-proxy**

**作用**：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）。

1.对于同一个业务集群而言，可以同时部署多个 codis-proxy 实例；   
2.不同 codis-proxy 之间由 codis-dashboard 保证状态同步。

**部署**：按照2.1中的用途规划，将在如下几台机器上部署该程序。

| **序号** | **IP** | **主机名** | **部署程序** |
| --- | --- | --- | --- |
| 01 | 192.168.1.21 | WebServer-21 | codis-proxy |
| 02 | 192.168.1.22 | WebServer-22 | codis-proxy |

**步骤：**   
***1. 修改配置***

#cd /usr/local/codis/bin/

#./codis-proxy - -default-config | tee ../conf/proxy.conf # 生成配置

#vim ../proxy.conf #修改配置

product\_name = "chatroom" # 设置项目名   
product\_auth = "123456" # 设置登录dashboard的密码（注意：与redis中requirepass一致）

session\_auth = "56789" # Redis客户端的登录密码（注意：与redis中requirepass不一致）   
# Set bind address for admin(rpc), tcp only.   
admin\_addr = "0.0.0.0:11080"   
# Set bind address for proxy, proto\_type can be “tcp”,”tcp4”, “tcp6”, “unix”   
or “unixpacket”.   
proto\_type = “tcp4”   
proxy\_addr = "0.0.0.0:19000" #绑定端口（Redis客户端连接此端口）   
# 外部存储   
jodis\_name = "zookeeper" # 外部存储类型   
jodis\_addr = “192.168.1.51:2181,192.168.1.52:2181,192.168.1.53:2181” # 外部存储列表   
jodis\_timeout = “20s”   
#会话设置   
session\_recv\_timeout = “0s” #如果不为０可能导致应用程序出现”write: broken pipe”的问题

备注：其他参数使用默认配置….

***2.启动程序***

#nohup ./codis-proxy - -ncpu=24 - -config=../conf/proxy.conf - -log=../log/proxy.log - -log-level=WRAN &

程序codis-proxy启动后，仍然处于waiting状态，虽然侦听了proxy\_addr端口，但是不会accept连接请求。只有将codis-proxy加入到集群并完成集群状态的同步，才能将状态改为online。最终才能accept连接请求。

**2.6 Redis-sentinel**

**作用**：Redis官方推荐的高可用性(HA)解决方案。它可以实现对Redis的监控、通知、自动故障转移。如果Master不能工作，则会自动启动故障转移进程，将其中的一个Slave提升为Master，其他的Slave重新设置新的Master服务。   
**部署**：按照1.2中的部署规划，将在如下机器上部署该程序。

| **序号** | **IP** | **主机名** | **部署程序** |
| --- | --- | --- | --- |
| 09 | 192.168.1.41 | WebServer-41 | redis-sentinel:26379 |
| 10 | 192.168.1.42 | WebServer-42 | redis-sentinel:26379 |
| 11 | 192.168.1.43 | WebServer-43 | redis-sentinel:26379 |

**步骤**：   
***1. 拷贝程序***

#cp -fr $GOPATH/github.com/CodisLabs/codis/extern/redis-3.2.8/src/redis-sentinel /usr/local/codis/bin/

***2. 拷贝配置***

#cp -fr $GOPATH/github.com/CodisLabs/codis/extern/redis-3.2.8/sentinel.conf /usr/local/codis/conf/

***3. 修改配置***

#cd /usr/local/codis/   
#vim ./conf/sentinel.conf

bind 0.0.0.0   
protected-mode no   
port 26379   
dir “/usr/local/codis/data/

备注：其他结点的配置与此一致。

***4. 启动程序***

cd /usr/local/codis/bin/   
nohup ./redis-sentinel ../conf/sentinel.conf &

**2.7 Codis-fe**

**作用**：集群管理界面。

1.多个集群实例共享可以共享同一个前端展示页面；   
2.通过配置文件管理后端codis-dashboard列表，配置文件可自动更新。

**部署**：按照2.1中的用途规划，将在如下机器上部署该程序。

| **序号** | **IP** | **主机名** | **部署程序** |
| --- | --- | --- | --- |
| 01 | 192.168.1.31 | WebServer-31 | codis-config:18080、codis-fe:18090 |

**安装**：   
***1. 生成配置***

#./codis-amdin - -dashboard-list - -zookeeper=192.168.1.51:2181 | tee ../conf/codis.json

[   
{   
“name”:”chatroom”,   
“dashboard”:192.168.1.31:18087”   
}   
]

***2. 启动程序***

#nohup ./codis-fe - -ncpu=4 - -log=../log/fe.log - -log-level=WARN - -dashboard-list=../conf/codis.josn –listen=0.0.0.0:18090 &

打开浏览器，输入192.168.1.31:18090便可看到codis集群的监控界面。如下图所示： 

**2.8 加入集群**

**步骤**：

***1. Codis-proxy***

在proxy栏的输入框中输入相应ip和端口可将proxy添加到集群中，如下图所示：

***2. Codis-server***

***3. 配置Slots***

***4. Sentinel***

至此，codis集群的搭建已经完成。可通过redis-cli操作codis集群存取数据。

**3 注意事项**

***1.密码配置***   
为了提高存储集群的安全性，就需要为集群配置数据存取密码和集群管理密码。

| **序号** | **密码类型** | **描述** |
| --- | --- | --- |
| 01 | 数据存取密码 | 可通过修改proxy配置中session\_auth调整数据存取密码。  当使用redis-cli存取数据时，auth密码需要与session\_auth一致。 |
| 02 | 集群管理密码 | 在proxy配置中的product\_auth、codis-dashbroad配置中  的product\_auth需要与codis-server配置中的requirepass  保持一致。 |