Redis配置与集群搭建

# 安装环境与版本

测试时候使用的是CentOS6.5的三虚拟机，最后真实搭建是在线上的服务器上面。三服务器的ip： 192.168.17.128 、192.168.17.129、192.168.17.130

共创建9个节点，3主6从，一个主节点对应2个从节点，创建节点分别为：

192.168.17.128 ： 7000

192.168.17.128 ： 7001

192.168.17.128 ： 7002

192.168.17.129 ： 7000

192.168.17.129 ： 7001

192.168.17.129 ： 7002

192.168.17.130 ： 7000

192.168.17.130 ： 7001

192.168.17.130 ： 7002

redis版本：redis-3.0.0

ruby版本：2.1.1p76

# 简单介绍

Redis是一款高性能的非关系型数据库，使用key-value存储数据的方式。主要特点如下：

数据存储在内存中，性能极高；

支持持久化，向磁盘中进行保存；

支持数据库的主从备份，即master-slave模式的数据备份；

支持多种数据存储方式（String、List、Set、ZSet、Hash）

集群也有如下特点：

如果一个主节点挂掉，并且这个时候没有从节点可以接替的，就会导致整个集群down掉；

如果有超过（经测试是包含）半数的主节点挂掉，就不论有没有从节点，整个集群就会down掉；

一旦集群down掉，所有的操作都不能使用

在本项目中将Redis作为缓存使用，存储SSO中所有用户的信息。在搭建时候，会将三个主节点分配在三台服务器上面，如果有一台服务器宕机，使用sentinel可以在不影响用户使用的情况下完成故障的转移。但如果再有第二次机器的宕机，可能会导致集群down掉。如果必须，会去重启服务器，一旦重启服务器，redis集群中存储的SSO中的用户的信息就会没有用处，可以使用提供的脚本去完成集群的重建。

# 集群搭建过程

## 软件准备

redis安装包：redis-3.0.0.tar.gz

ruby安装包：ruby-2.1.1.tar.gz

## 安装

1. 上传redis包并解压：

解压后的包需要一直使用，因此，最好将解压的redis目录放在一个软件放置的目录。我放置的目录是 /usr/local/redis-3.0.0

tar –zxvf redis-3.0.0.tar.gz

1. 编译并安装

进入解压后的redis目录：cd /usr/local/redis-3.0.0

执行 make && make install 即可安装，默认安装的位置为 /usr/local/bin。

如果想修改安装位置，使用 make PREFIX=安装位置路径 install的方式进行编译安装，如：想把redis安装在/usr/install/redis目录中，就在redis-3.0.0目录中执行 make PREFIX=/usr/install/redis install 即可。

我使用的是默认的路径。

tips：安装的路径会被配置进环境变量中

1. 复制 redis-trib.rb文件

redis自身提供了redis-trib.rb工具来完成redis集群的搭建，redis-trib.rb有多个方法，可以用来进行redis集群节点的管理，在后面会用到create创建集群的方法，另外还有check、fix、info等集群的信息查看与修改的方法。

将redis-trib.rb文件复制到可直接执行的路径下，就是环境变量里面，这里直接就使用redis的安装目录即可。redis-trib.rb文件在redis的解压目录的src目录下有。

cp src/redis-trib.rb /usr/local/bin

## 集群搭建

1. 创建集群以及节点目录

在第一台服务器上面（192.168.17.128）的redis-3.0.0目录下创建redis\_cluster目录，用来存储所有的集群节点信息。 在redis-3.0.0目录下执行：mkdir redis\_cluster

进入redis\_cluster ：cd redis\_cluster

依次创建3个节点目录，命名为 7000 7001 7002

mkdir 7000 7001 7002

1. 将配置文件放进节点目录中

将redis-3.0.0目录下的redis.conf依次copy放入几个节点目录中。

在redis-3.0.0目录中操作：

cp redis.conf redis\_cluster/7000

cp redis.conf redis\_cluster/7001

cp redis.conf redis\_cluster/7002

1. 修改配置文件（\*\*\*）

修改的为每个节点目录中对应的redis.conf文件，先以一个节点中的进行配置。

建议采用如下方式进行配置：先在一个服务器的一个节点的配置文件中完成所有要修改的信息，然后依次将该配置文件复制到其他节点对应的目录中，然后将所有的7000替换为对应的节点的名称（可以使用Notepad++的ftp插件进行编辑）。

同样，在配置第二个服务器的时候采用同样的方式，修改服务器ip以及端口对应的名称即可。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*配置信息\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

下面都是192.168.17.128服务器上的7000的节点中的配置

a、daemonize 是否作为后台守护进程运行，修改为yes

daemonize yes

b、pidfile 作为守护进程时候，将pid默认写到/var/run/redis.pid里面，修改为每个节点对应的名称

pidfile /var/run/redis\_7000.pid（7001节点就将7000改为7001）

c、port 监听的端口，修改为每个节点对应的

port 7000（其他节点修改同上）

d、bind 绑定的服务器ip，默认使用的是127.0.0.1，但是这个节点是需要被其他服务器的节点进行访问的，需要修改为自身对应的ip，当然也可以添加两个，一个127.0.0.1，一个自身的ip

bind 192.168.17.128 （192.168.17.129服务器中配置的话就修改为192.168.17.129）

e、cluster-enabled 开启集群

cluster-enabled yes 将注释去掉即可

f、cluster-config-file 集群自己维护的一个配置文件，只用指定，在第一次启动的时候自己生成

cluster-config-file nodes\_7000.conf（其他节点修改同上）

g、appendonly no 是否启用aof持久化的方式

appendonly yes 使用了就改为yes即可

h、appendfilename "appendonly.aof" 本地保存的aof文件的名称，如果没有设置下面要说的的dir的值，这个值一定要改，不然会造成多个节点的aof本地文件都是同一个，导致不可用（建议修改，方便分辨），文件会生成在dir指定的目录下面

appendfilename "appendonly\_7000.aof"（其他节点修改同上）

i、dir " 日志生成的目录，dump与aof生成的位置都是在这个目录中的，建议修改到每个节点

dir "/usr/local/redis-3.0.0/redis\_cluster/7000"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一个修改完成之后，将其他几个节点的配置文件参照这个进行修改，将对应的节点的信息（7000）都替换为每一个的节点信息。

1. 在另外两台服务器上面进行同样的操作

在另外两台服务器（192.168.17.129、192.168.17.130）上，同样执行安装、创建集群节点目录、修改配置文件的操作。

tip：

redis-trib.rb文件，也就是创建集群要使用的文件只用在一个服务器上面创建即可，也就是说复制redis-trib.rb这个操作只用在一个服务器上面操作即可。

在配置的时候，配置文件的内容基本跟第一台服务器上面的配置一样， 注意bind的时候修改为该服务器的ip。

1. 安装ruby（可以只在一台服务器上操作，建议所有机器全部安装）

在使用redis提供的redis-trib.rb工具的时候，需要有ruby的环境，这里就不过多介绍ruby。

将ruby包ruby-2.1.1.tar.gz上传并解压，进入解压后的目录。先执行配置：./configure –prefix=/usr/local/ruby 后面跟的prefix就是安装的目录。然后执行make && make install

执行完成之后就安装好了ruby，同Java类似，配置环境变量，可以直接去修改/root/.bash\_profile文件，在export PATH 下面加上export PATH="$PATH:/usr/local/ruby/bin"，之后使用source /root/.bash\_profile刷新一下。

环境变量配置好了之后可以使用 ruby –version进行验证。



1. 安装redis所需的gem文件（可以只在一台服务器上操作，建议所有机器全部安装）

gem将相当于ruby的软件管理工具，这里需要用到的是redis，可以直接联网进行安装即可。如果没有网，可以使用提供的redis-3.0.0.gem进行安装，安装的时候进入redis-3.0.0.gem所在的目录，gem install redis-3.0.0.gem即可。

1. 启动所有节点

启动节点的方式，就是调用redis-3.0.0目录中的src目录中的redis-server文件，后面跟上启动要跟随的配置文件。如：./src/redis-server redis\_cluster/7000/redis.conf即可启动端口为7000的节点。

这样当然很麻烦，可以写为脚本去执行，在redis-3.0.0目录中创建文件start-all.sh，编辑里面内容为：

./src/redis-server redis\_cluster/7000/redis.conf

./src/redis-server redis\_cluster/7001/redis.conf

./src/redis-server redis\_cluster/7002/redis.conf

，之后给该文件赋予可执行的权限 chmod u+x start-all.sh，这样要启动所有节点的时候，只用在redis-3.0.0目录中执行 ./start-all.sh即可。

在另一台服务器中以同样的方式可以启动所有节点。

启动成功之后，可以使用ps –ef | grep redis 方法去查看redis的启动情况，结果如下：



1. 创建集群（只在一台服务器上操作）

所有东西都准备好了之后，节点也已经全部启动之后，就可以去创建集群了。可以执行redis-trib.rb工具的create方法完成集群的创建。

redis-trib.rb create --replicas 2 192.168.17.128:7000 192.168.17.129:7000 192.168.17.130:7000 192.168.17.128:7001 192.168.17.129:7001 192.168.17.130:7001 192.168.17.128:7002 192.168.17.129:7002 192.168.17.130:7002

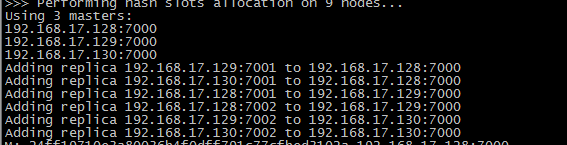
create：方法名，就是创建集群

2 ： 一个主节点有多少个从节点，2就是一主两从

ip:port ：所有要加入集群的节点的ip加上端口

关于顺序，我试了几次发现是不能自己去规定的，redis内部会自己去根据ip以及端口去分配主从节点，并且redis分配的基本都能保证主从分在不同的服务器上。

执行上面的脚本，没有问题的话会出现如下：



在上面的信息中可以看到主从对应关系：

Using 3 masters:

192.168.17.128:7000

192.168.17.129:7000

192.168.17.130:7000

Adding replica 192.168.17.129:7001 to 192.168.17.128:7000

Adding replica 192.168.17.130:7001 to 192.168.17.128:7000

Adding replica 192.168.17.128:7001 to 192.168.17.129:7000

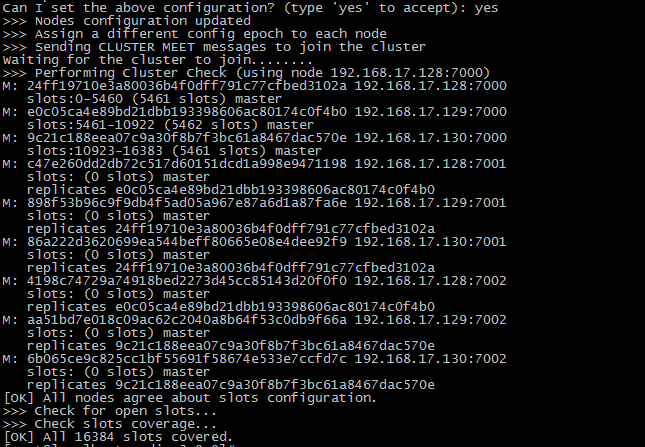
Adding replica 192.168.17.128:7002 to 192.168.17.129:7000

Adding replica 192.168.17.129:7002 to 192.168.17.130:7000

Adding replica 192.168.17.130:7002 to 192.168.17.130:7000

adding replica 后面跟的就是从节点，对应的主节点就是to后面的。

输入yes，成功的话就会出现如下的界面：



1. 测试集群

感觉要测试的几点：

1. 在一个节点中存值，能否在另外的节点中取出来
2. 主节点挂了之后，从节点能否自动更改为主节点，其他节点还可以取到值
3. 一台机器挂了后，用户能否还可以取到值

连接某个节点：

./src/redis-cli –c –p 7000 –h 192.168.17.128

-c：说明使用cluster集群模式登录

-p：后面跟上要连接的端口

-h：后面跟上要连接的服务器ip，如果bind后面跟的有127.0.0.1就可以不使用-h

关闭某个节点：

./src/redis-cli –c –p 7000 –h 192.168.17.128 shutdown

就是在连接的后面加上shutdown

# Sentinel监控程序

## 介绍

redis3.0之后的集群可以在检测到一台主节点断了之后，切换到相应的从节点，但是，经过多次的测试，发现在切换之前，第一次都会取不到值（在Java代码中进行的测试），会抛出Can not get a resource from pool的异常，之后再访问就可以。对此，redis提供的有自带的一个Sentinel工具，就是哨兵。

可以完成对redis集群的监控、通知、故障自动转移的功能。

sentinel是作为独立于redis的程序，不包含任何的存储的数据信息。

sentinel是会去修改自身的配置文件的，因此，也通过这点做到了持久化，每次有变化就去修改配置文件，每次启动都会去读取配置文件的状态。

## 启动

启动方式有两种：

如果在redis-3.0.0的目录的src目录中（就是redis-server、redis-cli所在的目录）有redis-sentinel的程序的话，可以直接使用 ./src/redis-sentinel sentinel.conf的方式。就是调用redis-sentinel 后面跟上sentinel.conf配置文件。（提供的包中可以使用这个）

如果在redis-3.0.0目录的src目录中没有redis-sentinel的程序，就可以使用redis-server 后面跟sentinel.conf文件 然后跟上 --sentinel 来表明启动sentinel。

## 配置

配置文件就是sentinel.conf，就在redis3.0.0的目录中

配置信息：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. port 监听端口，自定义即可，默认为26379

port 27000

1. daemonize 是否作为守护进程，后台启动，这个有的配置文件中没有，可以自己添加上去。

daemonize yes

1. logfile “/var/log/sentinel\_log.log” 日志文件，修改了上面的后台启动，这个最好修改，可以去使用tail –f 监听日志信息。

logfile “/var/log/sentinel\_log\_27000.log”

1. dir “/tmp” 运行的临时文件地址，这个可以保持默认即可

================下面是配置监听的一个主节点信息=============

sentinel monitor master7000 192.168.17.128 7000 1

sentinel down-after-milliseconds master7000 5000

sentinel parallel-syncs master7000 1

sentinel failover-timeout master7000 180000

# sentinel notification-script <master-name> <script-path>

# sentinel client-reconfig-script <master-name> <script-path>

==========================================================

解释一个主节点的信息：

sentinel monitor 后面跟的是 监听的主节点名（自定义，不能重复） 然后跟上监听的主节点的ip以及端口号 最后跟的数字1是说明至少有多少个sentinel监听程序认为这个主节点断了才去发起故障转移的处理。

sentinel down-after-milliseconds 该sentinel主观判断这个主节点断线的时间，单位是毫秒。sentinel会定时向指定的主节点发送ping命令，如果超过了指定的这个时间没有收到回复的信息，就认为该主节点主观下线。

sentinel parallel-syncs 发生故障转移的时候最多可以有多少个从节点同时对新的主节点进行同步。

sentinel failover-timeout 故障转移的超时时间，超过了这个时间还没完成故障转移的操作，就认为该次转移操作失败。单位是毫秒。

下面的两个注释掉的，是sentinel完成的通知的功能，发生节点下线等信息的时候会去调用对应的指定的脚本，可以在脚本中进行相应的操作。

上面的是一个主节点的配置信息，sentinel中只用去配置主节点的信息即可，它可以自己去通过集群去获取相应的从节点信息，主节点可以配置多个，如下：

sentinel monitor master7000 192.168.17.128 7000 1

sentinel down-after-milliseconds master7000 5000

sentinel parallel-syncs master7000 1

sentinel failover-timeout master7000 180000

# sentinel notification-script <master-name> <script-path>

# sentinel client-reconfig-script <master-name> <script-path>

sentinel monitor master7001 192.168.17.129 7000 1

sentinel down-after-milliseconds master7001 5000

sentinel parallel-syncs master7001 1

sentinel failover-timeout master7001 180000

# sentinel notification-script <master-name> <script-path>

# sentinel client-reconfig-script <master-name> <script-path>

sentinel monitor master7002 192.168.17.130 7000 1

sentinel down-after-milliseconds master7002 5000

sentinel parallel-syncs master7002 1

sentinel failover-timeout master7002 180000

# sentinel notification-script <master-name> <script-path>

# sentinel client-reconfig-script <master-name> <script-path>

# 集群重建

## 说明

集群重建会清空所有的数据并且不可恢复。

重建集群要做的事：

解除集群的相互关联的信息

清空每个节点中的数据。

## 具体操作

1. 将所有的节点全部关闭，可以使用脚本的方式，类似于start-all.sh脚本
2. 将redis\_cluster/下面的每一个节点目录中的文件（除了redis.conf）删除（删除appendonly.aof和dump.rdb文件可以清空数据信息、 删除nodes-7000.conf可以解除节点与集群关联的信息）
3. 在其他服务器上进行同样的操作，保证9个节点全部都完成了上面的操作，也就是说保证这个时候，所有节点处于关闭状态、所有服务器redis\_cluster目录下的每一个节点目录中只有一个redis.conf文件
4. 依次将所有的节点全部启动
5. 重新在安装有ruby环境的机器上执行redis-trib.rb工具完成集群的创建：

redis-trib.rb create --replicas 2 192.168.17.128:7000 192.168.17.129:7000 192.168.17.130:7000 192.168.17.128:7001 192.168.17.129:7001 192.168.17.130:7001 192.168.17.128:7002 192.168.17.129:7002 192.168.17.130:7002

按提示输入yes，即可。

# 常用命令

均在redis-3.0.0目录中进行的操作

启动节点服务：

./src/redis-server redis\_cluster/7000/redis.conf

连接节点：

./src/redis-cli -c -p 7000 -h 192.168.17.128

关闭节点：

./src/redis-cli -c -p 7000 -h 192.168.17.128 shutdown

查看节点集群信息：

./src/redis-cli -c -p 7000 -h 192.168.17.128 cluster nodes

查看单个节点的信息：

./src/redis-cli -c -p 7000 -h 192.168.17.128 info

只查看单个节点的主从信息：

./src/redis-cli -c -p 7000 -h 192.168.17.128 info Replication