Redis Cluster 搭建与测试（一）

作者：wangsch
原文链接：https://ld246.com/article/1477019413416
来源网站：链滴

<h1 style="text-align: center;"> Redis Cluster搭建与测试（一）</h1>
OS：CentOS 6.5 64位
Redis版本：redis 3.2.4，官网：http://redis.io
<h2> </h2>
<h2>安装</h2>
下载源码包后，解压，看里面的README.md文件，有更具体的编译和安装步骤。我的步骤：</
>
<pre class="brush: bash">cd /opt/src

tar -xvzf redis-3.2.4.tar.gz

cd redis-3.2.4

make

make test

make PREFIX=/opt/redis-3.2.4 install

ln -s /opt/redis-3.2.4/ /opt/redis</pre>

最后两步骤，执行安装的时候，指定特定的安装目录。最后，做一个软连接。这两步骤，主要是之后升级redis版本方便。
注：“make test”可能会报错“You need tcl 8.5...”，解决方法，文章末尾“附：安装tcl”

<h2>测试Redis Cluster</h2>
<h3>创建并启动</h3>
<pre class="brush: bash">cd /opt/src/redis-3.2.4/src

./redis-trib.rb create --replicas 1 127.0.0.1:7000 127.0.0.1:7001 127.0.0.1:7002 127.0.0.1:7003 1
7.0.0.1:7004 127.0.0.1:7005</pre>

成功输出：
[OK] All 16384 slots covered.

7000端口redis日志输出：
<pre class="brush: bash">25979:M 21 Oct 14:10:28.444 # configEpoch set to 1 via CLUSTER S
T-CONFIG-EPOCH
25979:M 21 Oct 14:10:28.544 # IP address for this node updated to 127.0.0.1
25979:M 21 Oct 14:10:32.854 * Slave 127.0.0.1:7003 asks for synchronization
25979:M 21 Oct 14:10:32.854 * Full resync requested by slave 127.0.0.1:7003
25979:M 21 Oct 14:10:32.854 * Starting BGSAVE for SYNC with target: disk
25979:M 21 Oct 14:10:32.855 * Background saving started by pid 26408
26408:C 21 Oct 14:10:32.874 * DB saved on disk
26408:C 21 Oct 14:10:32.875 * RDB: 6 MB of memory used by copy-on-write
25979:M 21 Oct 14:10:32.952 * Background saving terminated with success
25979:M 21 Oct 14:10:32.953 * Synchronization with slave 127.0.0.1:7003 succeeded
25979:M 21 Oct 14:10:33.352 # Cluster state changed: ok</pre>

7003端口redis输出：
<pre class="brush: bash">26003:M 21 Oct 14:10:28.445 # configEpoch set to 4 via CLUSTER S
T-CONFIG-EPOCH

26003:M 21 Oct 14:10:28.645 # IP address for this node updated to 127.0.0.1
26003:S 21 Oct 14:10:32.464 # Cluster state changed: ok
26003:S 21 Oct 14:10:32.854 * Connecting to MASTER 127.0.0.1:7000
26003:S 21 Oct 14:10:32.854 * MASTER <-> SLAVE sync started
26003:S 21 Oct 14:10:32.854 * Non blocking connect for SYNC fired the event.
26003:S 21 Oct 14:10:32.854 * Master replied to PING, replication can continue...
26003:S 21 Oct 14:10:32.854 * Partial resynchronization not possible (no cached master)
26003:S 21 Oct 14:10:32.856 * Full resync from master: 429aaf25a1dad65b8f534d76717ca7db
c1027a6:1
26003:S 21 Oct 14:10:32.953 * MASTER <-> SLAVE sync: receiving 76 bytes from master
26003:S 21 Oct 14:10:32.953 * MASTER <-> SLAVE sync: Flushing old data
26003:S 21 Oct 14:10:32.953 * MASTER <-> SLAVE sync: Loading DB in memory
26003:S 21 Oct 14:10:32.953 * MASTER <-> SLAVE sync: Finished with success
26003:S 21 Oct 14:10:32.954 * Background append only file rewriting started by pid 26409
26003:S 21 Oct 14:10:32.980 * AOF rewrite child asks to stop sending diffs.
26409:C 21 Oct 14:10:32.980 * Parent agreed to stop sending diffs. Finalizing AOF...
26409:C 21 Oct 14:10:32.980 * Concatenating 0.00 MB of AOF diff received from parent.
26409:C 21 Oct 14:10:32.980 * SYNC append only file rewrite performed
26409:C 21 Oct 14:10:32.980 * AOF rewrite: 6 MB of memory used by copy-on-write
26003:S 21 Oct 14:10:33.054 * Background AOF rewrite terminated with success
26003:S 21 Oct 14:10:33.054 * Residual parent diff successfully flushed to the rewritten AOF (0
00 MB)
26003:S 21 Oct 14:10:33.054 * Background AOF rewrite finished successfully</pre>

client “CLUSTER NODES”命令输出：
<pre class="brush: bash">127.0.0.1:7000> CLUSTER NODES
8dc1c8679a46ba85750a5d388937932689e4963f 127.0.0.1:7002 master - 0 1477030603379 3
onnected 10923-16383
be619fa382c8471eaeb6222d735ed05e6cab5e7a 127.0.0.1:7004 slave 688a1ac1bae1a24327cd
6ac4b9efc69888885c9 0 1477030602877 5 connected
088390b994557c8a94cab38cafded4ec60fce724 127.0.0.1:7005 slave 8dc1c8679a46ba85750a
d388937932689e4963f 0 1477030604380 6 connected
688a1ac1bae1a24327cdd6ac4b9efc69888885c9 127.0.0.1:7001 master - 0 1477030603379 2 c
nnected 5461-10922
86ca55c465645c184fd1dbc48336e0aae5620948 127.0.0.1:7000 myself,master - 0 0 1 connect
d 0-5460
eb5000f25af13145a04ee8ec8b9ecfe8426ae833 127.0.0.1:7003 slave 86ca55c465645c184fd1d
c48336e0aae5620948 0 1477030602877 4 connected</pre>

<h3>执行一些数据操作</h3>
<pre class="brush: bash">./redis-cli -c -h 127.0.0.1 -p 7001
127.0.0.1:7001>
127.0.0.1:7001>
127.0.0.1:7001> set foo bar
-> Redirected to slot [12182] located at 127.0.0.1:7002
OK
127.0.0.1:7002> dbsize
(integer) 1
127.0.0.1:7002>
127.0.0.1:7002> get foo
"bar"
127.0.0.1:7002>
127.0.0.1:7002> set key1 value1
-> Redirected to slot [9189] located at 127.0.0.1:7001

OK
127.0.0.1:7001> dbsize
(integer) 1
127.0.0.1:7001> get key1
"value1"</pre>
可以看出，操作被自动重定向

<h3>测试failover</h3>
逐步关闭redis cluster中的redis实例，看看是什么效果。

1. 关闭7000的redis实例
7001
<pre class="brush: bash">25986:M 21 Oct 14:28:16.408 * Marking node 86ca55c465645c184f
1dbc48336e0aae5620948 as failing (quorum reached).
25986:M 21 Oct 14:28:16.408 # Cluster state changed: fail
25986:M 21 Oct 14:28:17.217 # Failover auth granted to eb5000f25af13145a04ee8ec8b9ecfe8
26ae833 for epoch 7
25986:M 21 Oct 14:28:17.256 # Cluster state changed: ok</pre>

7002
<pre class="brush: bash">25996:M 21 Oct 14:28:16.408 * Marking node 86ca55c465645c184f
1dbc48336e0aae5620948 as failing (quorum reached).
25996:M 21 Oct 14:28:16.408 # Cluster state changed: fail
25996:M 21 Oct 14:28:17.217 # Failover auth granted to eb5000f25af13145a04ee8ec8b9ecfe8
26ae833 for epoch 7
25996:M 21 Oct 14:28:17.256 # Cluster state changed: ok</pre>

7003是7000的slave，控制台输出：
<pre class="brush: bash">26003:S 21 Oct 14:28:10.288 # Connection with master lost.
26003:S 21 Oct 14:28:10.289 * Caching the disconnected master state.
26003:S 21 Oct 14:28:10.891 * Connecting to MASTER 127.0.0.1:7000
26003:S 21 Oct 14:28:10.891 * MASTER <-> SLAVE sync started
26003:S 21 Oct 14:28:10.891 # Error condition on socket for SYNC: Connection refused
26003:S 21 Oct 14:28:11.893 * Connecting to MASTER 127.0.0.1:7000
26003:S 21 Oct 14:28:11.893 * MASTER <-> SLAVE sync started
26003:S 21 Oct 14:28:11.893 # Error condition on socket for SYNC: Connection refused
26003:S 21 Oct 14:28:12.896 * Connecting to MASTER 127.0.0.1:7000
26003:S 21 Oct 14:28:12.896 * MASTER <-> SLAVE sync started
26003:S 21 Oct 14:28:12.896 # Error condition on socket for SYNC: Connection refused
26003:S 21 Oct 14:28:13.898 * Connecting to MASTER 127.0.0.1:7000
26003:S 21 Oct 14:28:13.898 * MASTER <-> SLAVE sync started
26003:S 21 Oct 14:28:13.898 # Error condition on socket for SYNC: Connection refused
26003:S 21 Oct 14:28:14.903 * Connecting to MASTER 127.0.0.1:7000
26003:S 21 Oct 14:28:14.903 * MASTER <-> SLAVE sync started
26003:S 21 Oct 14:28:14.903 # Error condition on socket for SYNC: Connection refused
26003:S 21 Oct 14:28:15.905 * Connecting to MASTER 127.0.0.1:7000
26003:S 21 Oct 14:28:15.905 * MASTER <-> SLAVE sync started
26003:S 21 Oct 14:28:15.905 # Error condition on socket for SYNC: Connection refused
26003:S 21 Oct 14:28:16.307 * Marking node 86ca55c465645c184fd1dbc48336e0aae5620948
s failing (quorum reached).
26003:S 21 Oct 14:28:16.308 # Cluster state changed: fail
26003:S 21 Oct 14:28:16.407 # Start of election delayed for 713 milliseconds (rank #0, offset 1
46).
26003:S 21 Oct 14:28:16.909 * Connecting to MASTER 127.0.0.1:7000

26003:S 21 Oct 14:28:16.909 * MASTER <-> SLAVE sync started
26003:S 21 Oct 14:28:16.909 # Error condition on socket for SYNC: Connection refused
26003:S 21 Oct 14:28:17.210 # Starting a failover election for epoch 7.
26003:S 21 Oct 14:28:17.217 # Failover election won: I'm the new master.
26003:S 21 Oct 14:28:17.217 # configEpoch set to 7 after successful failover
26003:M 21 Oct 14:28:17.217 * Discarding previously cached master state.
26003:M 21 Oct 14:28:17.217 # Cluster state changed: ok</pre>

7004
<pre class="brush: bash">26010:S 21 Oct 14:28:16.307 * Marking node 86ca55c465645c184f
1dbc48336e0aae5620948 as failing (quorum reached).
26010:S 21 Oct 14:28:16.307 # Cluster state changed: fail
26010:S 21 Oct 14:28:17.257 # Cluster state changed: ok</pre>

7005
<pre class="brush: bash">26020:S 21 Oct 14:28:16.407 * Marking node 86ca55c465645c184f
1dbc48336e0aae5620948 as failing (quorum reached).
26020:S 21 Oct 14:28:16.407 # Cluster state changed: fail
26020:S 21 Oct 14:28:17.258 # Cluster state changed: ok</pre>

client “CLUSTER NODES”命令输出：
<pre class="brush: bash">127.0.0.1:7002> CLUSTER NODES
be619fa382c8471eaeb6222d735ed05e6cab5e7a 127.0.0.1:7004 slave 688a1ac1bae1a24327cd
6ac4b9efc69888885c9 0 1477031314864 5 connected
eb5000f25af13145a04ee8ec8b9ecfe8426ae833 127.0.0.1:7003 master - 0 1477031314363 7 c
nnected 0-5460
86ca55c465645c184fd1dbc48336e0aae5620948 127.0.0.1:7000 master,fail - 1477031290388 1
77031289787 1 disconnected
088390b994557c8a94cab38cafded4ec60fce724 127.0.0.1:7005 slave 8dc1c8679a46ba85750a
d388937932689e4963f 0 1477031313361 6 connected
688a1ac1bae1a24327cdd6ac4b9efc69888885c9 127.0.0.1:7001 master - 0 1477031315366 2 c
nnected 5461-10922
8dc1c8679a46ba85750a5d388937932689e4963f 127.0.0.1:7002 myself,master - 0 0 3 connect
d 10923-16383 </pre>

2. 再退出一个slave，redis实例7004
剩下的7001、7002、7003、7005均会收到类似消息：
<pre class="brush: bash">26020:S 21 Oct 14:34:09.473 * FAIL message received from 8dc1c8
79a46ba85750a5d388937932689e4963f about be619fa382c8471eaeb6222d735ed05e6cab5e
a</pre>

3. 再退出一个master，redis实例7001
剩下的7002、7003、7005均会收到类似消息：
<pre class="brush: bash">25996:M 21 Oct 14:37:00.001 * Marking node 688a1ac1bae1a2432
cdd6ac4b9efc69888885c9 as failing (quorum reached).
25996:M 21 Oct 14:37:00.001 # Cluster state changed: fail</pre>

过去一段时间，保持在“Cluster state changed: fail”这个状态不再变动
此刻，redis cluster已经完全不可用。client执行命令：
<pre class="brush: bash">127.0.0.1:7002> set a b
(error) CLUSTERDOWN The cluster is down</pre>

QUES：Redis Cluster如何平滑的关闭和重启，是否有这个需求？

<h2>附</h2>
<h3>安装tcl</h3>
编译redis后，推荐“make test”测试编译结果，然而：
[root@wangsch-serv redis-3.2.4]# make test
cd src && make test
make[1]: Entering directory `/opt/src/redis-3.2.4/src'</em><br /><em>You need tcl 8.5 or newer in order to run the Redis test</em><br /><em>make[1]: *** [test] 错误 1</em><br /><em>make[1]: Leaving directory `/opt/src/redis-3.2.4/src'
make: *** [test] 错误 2
"You need tcl 8.5 or newer in order to run the Redis test"，说明要执行这个测试，还要依赖tcl，要下载最新版并编译安装。官网：http://www.tcl.tk，在https://sourceforge.net/projects/tcl/下载
最新版源码包下载到：/opt/src/tcl8.6.6-src.tar.gz，解压后进到解压目录的unix子目录，然后接编译安装三板斧。
<pre class="brush: bash">cd tcl8.6.6/unix

./configure && make & make install</pre>

再次执行“make test”，成功：
All tests passed without errors!
Cleanup: may take some time... OK

<h3>安装redis为daemon</h3>
可选步骤，可以通过“service”命令管理redis。进入到redis解压目录的utils目录执行：
<pre class="brush: java">./install_server.sh
Welcome to the redis service installer
This script will help you easily set up a running redis server

Please select the redis port for this instance: [6379]
Selecting default: 6379
Please select the redis config file name [/etc/redis/6379.conf] /opt/redis-conf/6379
Please select the redis log file name [/var/log/redis_6379.log] /data/log/redis/6379
Please select the data directory for this instance [/var/lib/redis/6379] /data/redis/6379
Please select the redis executable path [/usr/local/bin/redis-server] /opt/redis/bin/redis-server
Selected config:
Port   : 6379
Config file   : /opt/redis-conf/6379
Log file   : /data/log/redis/6379
Data dir   : /data/redis/6379
Executable   : /opt/redis/bin/redis-server
Cli Executable : /usr/local/bin/redis-cli
Is this ok? Then press ENTER to go on or Ctrl-C to abort.</pre>

Copied /tmp/6379.conf => /etc/init.d/redis_6379

Installing service...

Successfully added to chkconfig!

Successfully added to runlevels 345!

Starting Redis server...

Installation successful!</pre>

上面的步骤，采用交互方式，指定端口号、数据目录等配置项。最后点击确认后，生成脚本文件“/etc/init.d/redis_<portnumber>”，portnumber为redis实例端口号。
<pre class="brush: bash">service redis_6379 status
Redis is running (25080)</pre>
 可以看到redis进程已经启动，支持“start”、“stop”、“stop”和“status”四种指令。

