大数据库系统

3.8 Redis运维与哨兵模式

3.7 Redis运维

- 主要内容

3.7.1 Redis运维相关命令简介

3.7.2 Redis哨兵模式的配置及使用

3.7.1 Redis运维相关命令简介

- Redis运维相关命令

服务器配置准备：

Master服务器127.0.0.1:6379的配置文件：redis6379.conf（注释掉登录密码）

```
pidfile /var/run/redis.pid
port 6379
dbfilename dump6379.rdb
dir /var/rdb/
appendonly yes
appendfilename /var/rdb/appendonly6379.aof
```

Master服务器127.0.0.1:6380的配置文件：redis6380.conf （注释掉登录密码验证）

```
daemonize yes
pidfile /var/run/redis6380.pid
port 6380
dbfilename dump6380.rdb
dir /var/rdb/
appendonly yes
appendfilename /var/rdb/appendonly6380.aof
```

3.7.1 Redis运维相关命令简介

- TIME命令

格式：TIME

获取当前服务器的时间

返回包含两个字符串的列表，第一个字符串是当前时间（UNIX时间戳），第二个字符串是在当前这一秒内逝去的微秒数

UNIX时间戳是从1970年1月1日（UTC/GMT的午夜）开始所经过的秒数

```
redis 127.0.0.1:6379> time
1) "1380812899"
2) "881270"
```

现在的Unix时间戳(Unix timestamp)是： 1618016978 开始 停止 刷新

Unix时间戳（Unix timestamp） 1618016973 秒 转换

时间（年/月/日 时:分:秒） 转换成Unix时间戳 秒

时间 2021 年 月 日 时 分 秒 转换Unix时间戳 秒

3.7.1 Redis运维相关命令简介

- DBSIZE命令

格式：DBSIZE

统计当前数据库中键的数量

```
redis 127.0.0.1:6379> set title www.zixue.it
OK
redis 127.0.0.1:6379> keys *
1) "title"
redis 127.0.0.1:6379> dbsize
(integer) 1
```

注意：不是查看服务器有多少个key，而是当前数据库有多少个key

```
redis 127.0.0.1:6379> select 2
OK
redis 127.0.0.1:6379[2]> dbsize
(integer) 0
redis 127.0.0.1:6379[2]>
```

3.7.1 Redis运维相关命令简介

◆ BGREWRITEAOF命令

格式：BGREWRITEAOF

执行一个AOF文件重写操作。该命令执行后，会创建一个当前AOF文件的优化版本，执行失败时，AOF文件数据并不会丢失

- 例：1、终端1开启6379服务器，添加键值title，并修改title

```
redis 127.0.0.1:6379[2]> set title www.baidu.com
OK
redis 127.0.0.1:6379[2]> set title www.so.com
OK
redis 127.0.0.1:6379[2]>
```

终端1

- 2、在终端2查看aof文件内容和大小

终端2

```
[root@localhost rdb]# ll
total 4
-rw-r--r-- 1 root root 174 Oct  3 23:10 appendonly6379.aof
```

- 3、回到6379服务器输入命令重写AOF文件

终端1

```
redis 127.0.0.1:6379[2]> bgrewriteaof
Background append only file rewriting started
redis 127.0.0.1:6379[2]>
```

- 4、重新回到终端2查看aof文件内容和大小

终端2

```
[root@localhost rdb]# ll
total 4
-rw-r--r-- 1 root root 130 Oct  3 23:11 appendonly6379.aof
```

- 发现文件的大小变小了

```
[root@localhost rdb]# more appendonly6379.aof
*2
$6
SELECT
$1
0
*3
$3
SET
$5
title
$12
www.zixue.it
*2
$6
SELECT
$1
2
*3
$3
SET
$5
title
$10
www.so.com
```

```
[root@localhost rdb]# more appendonly6379.aof
*2
$6
SELECT
$1
0
*3
$3
set
$5
title
$12
www.zixue.it
*2
$6
SELECT
$1
2
*3
$3
set
$5
title
$13
www.baidu.com
*3
$3
set
$5
title
$10
www.so.com
```

- 相比于之前的AOF文件，发现后面www.baidu.com那部分没有了，重写成功

3.7.1 Redis运维相关命令简介

- SAVE命令

格式：SAVE

保存RDB数据到磁盘中

SAVE命令具体执行的是一个同步保存操作，它以RDB文件的形式将当前Redis的所有内存数据快照保存到磁盘中

不建议使用SAVE命令来保存数据，因为它在执行后会阻塞所有客户端

3.7.1 Redis运维相关命令简介

- BGSAVE命令

格式：BGSAVE

在Redis服务后端采用异步的方式将RDB快照数据保存到当前数据库的磁盘中

执行原理：Redis启动一个新的子进程，原来的Redis进程（父进程）继续执行客户端请求操作，而子进程则负责将数据保存到磁盘中，然后退出

```
redis 127.0.0.1:6379> bgsave
Background saving started
redis 127.0.0.1:6379>
```

如果内存数据量大，BGSAVE可以后台做RDB存储，可以继续手头的工作

3.7.1 Redis运维相关命令简介

- LASTSAVE命令

格式：LASTSAVE

用于获取最近一次Redis成功将数据保存到磁盘上的时间，时间格式是UNIX时间戳

如果不能确定BGSAVE命令是否已经成功执行，此时可以使用LASTSAVE命令来查看相关信息

```
redis 127.0.0.1:6379> save
OK
redis 127.0.0.1:6379> lastsave
(integer) 1380813172
redis 127.0.0.1:6379>
```

假如未开启AOF，数据库崩溃了，可以将当前时间-LASTSAVE得到的时间，就能推出丢失了多长时间段的数据

3.7.1 Redis运维相关命令简介

```
redis 127.0.0.1:6379[2]> set title www.baidu.com
OK
redis 127.0.0.1:6379[2]> set title www.so.com
OK
redis 127.0.0.1:6379[2]>
```

```
redis 127.0.0.1:6379[2]> select 0
OK
redis 127.0.0.1:6379> flushall
OK
redis 127.0.0.1:6379> select 2
OK
redis 127.0.0.1:6379[2]> keys *
(empty list or set)
redis 127.0.0.1:6379[2]>
```

- FLUSHALL 命令

  格式： FLUSHALL

  清空所有数据库所有键

```
redis 127.0.0.1:6379> keys *
1) "title"
redis 127.0.0.1:6379> flushdb
OK
redis 127.0.0.1:6379> select 2
OK
redis 127.0.0.1:6379[2]> keys *
1) "title"
redis 127.0.0.1:6379[2]>
```

- FLUSHDB

  格式： Flushdb

  清空当前数据库所有键

- 慎用！

3.7.1 Redis运维相关命令简介

- INFO命令

格式：INFO [section]

用于查看Redis服务器的各种信息及统计相关数值

参数section的设置可以让INFO命令只返回某一部分的信息

如果INFO命令不带任何参数，则默认以default作为参数

例：使用info查看6379服务器的信息，不加参数

```
redis 127.0.0.1:6379[2]> keys *
(empty list or set)
redis 127.0.0.1:6379[2]> select 0
OK
redis 127.0.0.1:6379> info
```

INFO server

```
# Server
redis_version:2.6.16
redis_git_sha1:00000000
redis_git_dirty:0
redis_mode:standalone
os:Linux 2.6.32-358.el6.x86_64 x86_64
arch_bits:64
multiplexing_api:epoll
gcc_version:4.4.7
process_id:24908
run_id:bbdd5e5d5176582cd30d5d7444d92d08d9c14f9d
tcp_port:6379
uptime_in_seconds:587
uptime_in_days:0
hz:10
lru_clock:1766460
```

INFO clients

```
# Clients
connected_clients:1
client_longest_output_list:0
client_biggest_input_buf:0
blocked_clients:0
```

INFO memory

```
# Memory
used_memory:857664
used_memory_human:837.56K
used_memory_rss:7577600
used_memory_peak:856992
used_memory_peak_human:836.91K
used_memory_lua:31744
mem_fragmentation_ratio:8.84
mem_allocator:jemalloc-3.2.0
```

INFO persistence

```
# Persistence
loading:0
rdb_changes_since_last_save:3
rdb_bgsave_in_progress:0
rdb_last_save_time:1380813350
rdb_last_bgsave_status:ok
rdb_last_bgsave_time_sec:0
rdb_current_bgsave_time_sec:-1
aof_enabled:1
aof_rewrite_in_progress:0
aof_rewrite_scheduled:0
aof_last_rewrite_time_sec:0
aof_current_rewrite_time_sec:-1
aof_last_bgrewrite_status:ok
aof_current_size:202
aof_base_size:130
aof_pending_rewrite:0
aof_buffer_length:0
aof_rewrite_buffer_length:0
aof_pending_bio_fsync:0
aof_delayed_fsync:0
```

```
# Stats
total_connections_received:1
total_commands_processed:22
instantaneous_ops_per_sec:0
rejected_connections:0
expired_keys:0
evicted_keys:0
keyspace_hits:0
keyspace_misses:0
pubsub_channels:0
pubsub_patterns:0
latest_fork_usec:8757

# Replication
role:master
connected_slaves:0

# CPU
used_cpu_sys:5.26
used_cpu_user:4.64
used_cpu_sys_children:0.02
used_cpu_user_children:0.01

# Keyspace
```

INFO [section]

- server部分：该部分主要说明Redis的服务器信息
- clients部分：该部分记录了已连接客户端的信息
- memory部分：该部分记录了Redis服务器的内存相关信息。
- persistence部分：该部分记录了与持久化（RDB持久化和AOF持久化）相关的信息。
- stats部分：该部分记录了相关的统计信息。
- replication部分：该部分记录了Redis数据库主从复制信息。
- cpu部分：该部分记录了CPU的计算量统计信息。
- Command stats部分：该部分记录了Redis各种命令的执行统计信息，如执行命令消耗的CPU时间、执行次数等。
- cluster部分：该部分记录了与Redis集群相关的信息。
- keyspace部分：该部分记录了与Redis数据库相关的统计信息，如键的数量。

Replication主从复制

```
# Replication
role:master
connected_slaves:0
```

这里因为6379和6380均为主服务器，所以这里6379的Replication信息显示从服务器数为0

1、增加服务器6381作为6379的从服务器

```
[root@localhost redis]# vim redis6381.conf
daemonize yes
port 6381
pidfile /var/run/redis6381.pid
appendonly no
slaveof localhost 6379
```

并关闭登陆密码验证选项，关闭RDB，关闭AOF，保存退出

2、启动6381服务器

```
hadoop@fisher-VirtualBox:/usr/local/redis$ ./src/redis-server ./redis6381.conf
```

3、客户端连接6379服务器，输入info，观察6379的replication

```
# Replication
role:master
connected_slaves:1
slave0:127.0.0.1,6381,online
```

显示有1个从服务器，并且给出了从服务器的ip和端口号

3.7.1 Redis运维相关命令简介

4、客户端连入6381服务器，查看该服务器的Replication

role:slave　　表示该服务器角色是从服务器

master_host:192.168.1.128　　该服务器的主服务器ip

master_port:6379　　该服务器主服务器端口号

master_link_status:up　　主服务器状态：在线

```
# Replication
role:slave
master_host:127.0.0.1
master_port:6379
master_link_status:up
master_last_io_seconds_ago:4
master_sync_in_progress:0
```

3.7.1 Redis运维相关命令简介

Persistence 持久化

RDB文件状态监控

- rdb_changes_since_last_save 上次RDB保存以后改变的key次数，太多可以考虑手动RDB下
- db_bgsave_in_progress 表示当前是否在进行bgsave操作
- rdb_last_save_time 上次保存RDB文件的时间戳
- rdb_last_bgsave_time_sec 上次保存的耗时
- rdb_last_bgsave_status 上次保存的状态
- rdb_current_bgsave_time_sec 目前保存RDB文件已花费的时间

```
# Persistence
loading:0
rdb_changes_since_last_save:3
rdb_bgsave_in_progress:0
rdb_last_save_time:1380813350
rdb_last_bgsave_status:ok
rdb_last_bgsave_time_sec:0
rdb_current_bgsave_time_sec:-1
aof_enabled:1
aof_rewrite_in_progress:0
aof_rewrite_scheduled:0
aof_last_rewrite_time_sec:0
aof_current_rewrite_time_sec:-1
aof_last_bgrewrite_status:ok
aof_current_size:202
aof_base_size:130
aof_pending_rewrite:0
aof_buffer_length:0
aof_rewrite_buffer_length:0
aof_pending_bio_fsync:0
aof_delayed_fsync:0
```

3.7.1 Redis运维相关命令简介

Persistence 持久化

AOF文件状态监控

- aof_enabled AOF是否启用
- aof_rewrite_in_progress 表示当前是否在进行重写AOF文件操作
- aof_last_rewrite_time_sec 上次重写的时间戳
- aof_current_rewrite_time_sec:-1 目前重写的时间
- aof_last_bgrewrite_status:ok 上次后台重写状态
- aof_last_write_status:ok 上次命令写入状态

```
# Persistence
loading:0
rdb_changes_since_last_save:3
rdb_bgsave_in_progress:0
rdb_last_save_time:1380813350
rdb_last_bgsave_status:ok
rdb_last_bgsave_time_sec:0
rdb_current_bgsave_time_sec:-1
aof_enabled:1
aof_rewrite_in_progress:0
aof_rewrite_scheduled:0
aof_last_rewrite_time_sec:0
aof_current_rewrite_time_sec:-1
aof_last_bgrewrite_status:ok
aof_current_size:202
aof_base_size:130
aof_pending_rewrite:0
aof_buffer_length:0
aof_rewrite_buffer_length:0
aof_pending_bio_fsync:0
aof_delayed_fsync:0
```

3.7.1 Redis运维相关命令简介

- CONFIG GET命令

格式：CONFIG GET parameter

获取运行中的Redis服务器的配置参数

参数parameter是命令搜索关键字，用于查找所有匹配的配置参数，查找返回的参数和值以键值对的形式排列

例：3.7.1 Redis运维相关命令简介查看该服务器是否需要密码

```
redis 127.0.0.1:6379> config get requirepass
1) "requirepass"
2) ""
redis 127.0.0.1:6379>
```

3.7.1 Redis运维相关命令简介

- CONFIG SET命令

  格式：CONFIG SET parameter value

  用于修改Redis服务器的配置，修改之后不需要重新启动服务器也能生效

  能被CONFIG SET命令所修改的Redis配置参数都可以在配置文件redis.conf中找到

3.7.1 Redis运维相关命令简介

- 慢查询日志（慢日志）
  - Redis慢查询日志功能用于记录服务器在执行命令时，执行时间超过给定时长的命令的相关信息，这些相关信息包括慢查询ID、发生时间戳、耗时、命令的详细信息等
  - 开发人员和运维人员可以通过慢查询日志来定位系统的慢操作，然后利用这个功能产生的日志来监视和优化查询速度

3.7.1 Redis运维相关命令简介

- 慢日志配置

  在Redis的配置文件redis.conf中

- slowlog-log-slower-than

  慢查询的预设阈值

  指定执行时间超过多少微秒的命令请求会被记录到日志中（1s = 1000000μs）

  例：如果这个参数的值为1000，那么执行时间超过1000μs的命令就会被记录到慢查询日志中

```
# The following time is expressed in microseconds, so 1000000 is equivalent
# to one second. Note that a negative number disables the slow log, while
# a value of zero forces the logging of every command.
slowlog-log-slower-than 10000
```

3.7.1 Redis运维相关命令简介

- slowlog-max-len

  慢查询日志的最大长度

  用于指定服务器最多保存多少条慢查询日志

  先进先出的方式保存多条慢查询日志

  当服务器存储的慢查询日志数量等于slowlog-max-len参数的值时，服务器在添加一条新的慢查询日志之前，会先将最旧的一条慢查询日志删除

```
# There is no limit to this length. Just be aware that it will consume memory.
# You can reclaim memory used by the slow log with SLOWLOG RESET.
slowlog-max-len 128
```

3.7.1 Redis运维相关命令简介

- SLOWLOG 命令

格式：SLOWLOG GET [n]：指定获取n条慢查询日志

SLOWLOG LEN：　获取慢查询日志的长度

SLOWLOG RESET：清空慢查询日志

- 例1：config get查看慢查询的预设阈值

```
redis 127.0.0.1:6379> config get slowlog-log-slower-than
1) "slowlog-log-slower-than"
2) "10000"
```

- 例2：将慢查询的预设阈值改为100微秒

```
redis 127.0.0.1:6379> config set slowlog-log-slower-than 100
OK
redis 127.0.0.1:6379> config get slowlog-log-slower-than
1) "slowlog-log-slower-than"
2) "100"
redis 127.0.0.1:6379>
```

通过config set修改成功，随便执行几条命令flushall、bgsave、save等

- 例3：查看慢查询日志的最大长度

```
redis 127.0.0.1:6379> config get slowlog-max-len
1) "slowlog-max-len"
2) "128"
```

- 例4：查询慢日志，全部内容

```
redis 127.0.0.1:6379> slowlog get
1) 1) (integer) 4
   2) (integer) 1380813688
   3) (integer) 11023
   4) 1) "SYNC"
2) 1) (integer) 3
   2) (integer) 1380813350
   3) (integer) 46939
   4) 1) "flushall"
3) 1) (integer) 2
   2) (integer) 1380813247
   3) (integer) 27669
   4) 1) "bgsave"
4) 1) (integer) 1
   2) (integer) 1380813172
   3) (integer) 32845
   4) 1) "save"
5) 1) (integer) 0
   2) (integer) 1380813097
   3) (integer) 16067
   4) 1) "bgrewriteaof"
```

例5：查询慢日志，前三条

```
redis 127.0.0.1:6379> slowlog get 3
1) 1) (integer) 4
   2) (integer) 1380813688
   3) (integer) 11023
   4) 1) "SYNC"
2) 1) (integer) 3
   2) (integer) 1380813350
   3) (integer) 46939
   4) 1) "flushall"
3) 1) (integer) 2
   2) (integer) 1380813247
   3) (integer) 27669
   4) 1) "bgsave"
```

3.7.1 Redis运维相关命令简介

◆ SHUTDOWN命令

格式：SHUTDOWN [SAVE|NOSAVE]

关闭服务器

SHUTDOWN或SHUTDOWN SAVE会作如下操作：

- ✓直接关闭Redis服务器。
- ✓关闭（停止）所有客户端。
- ✓在AOF选项被打开的情况下，执行SHUTDOWN命令将会更新AOF文件
- ✓执行SHUTDOWN命令的同时，即使没有触发RDB保存状态，也会执行SAVE命令

SHUTDOWN NOSAVE命令的作用与SHUTDOWN SAVE命令的作用刚好相反，它会阻止Redis数据库执行保存操作

3.7.1 Redis运维相关命令简介

- 例：1、使用shutdown关闭6379服务器

```
127.0.0.1:6379> shutdown
not connected>
```

发现执行shutdown后没有反应，但所有命令均没有反应

- 2、客户端连接6379服务器

```
hadoop@fisher-VirtualBox:/usr/local/redis$ ./src/redis-cli
Could not connect to Redis at 127.0.0.1:6379: Connection refused
```

提示连接被拒绝

- 问：如果不小心运行了flushall，清空了所有数据库，但有开持久化该怎么办?
- 答：首先，立即 shutdown nosave 关闭服务器，不可以shutdown或者shutdown save；接着，手工编辑aof文件, 去掉文件中的 “flushall ”相关行, 然后开启服务器，就可以导入回原来数据

3.7.1 Redis运维相关命令简介

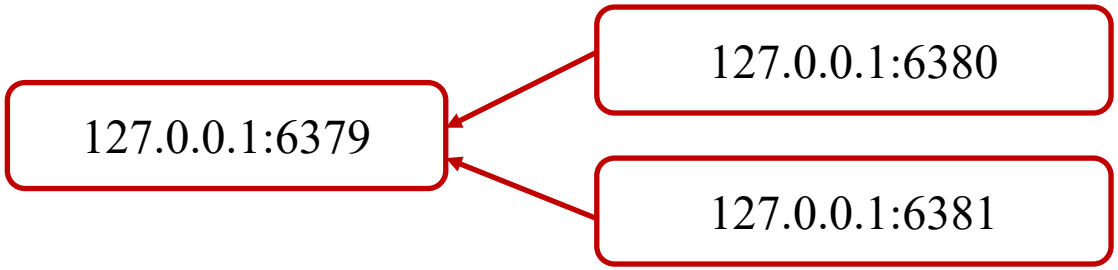
1、假定共3个服务器：

A：Master服务器：127.0.0.1:6379，对应的配置文件为redis6379.conf

B： Master的Slave：127.0.0.1:6380，对应的配置文件为redis6380.conf

C： Master的Slave：127.0.0.1:6381，对应的配置文件为redis6381.conf

星型结构：




```
redis 127.0.0.1:6379> info Replication
# Replication
role:master
connected_slaves:2
slave0:127.0.0.1,6381,online
slave1:127.0.0.1,6380,online
redis 127.0.0.1:6379>
```

假定Master6379服务器出现宕机，接下来该做什么？

答：将6380变成master服务器，再将6381变成6380的从服务器

2、关闭6379服务器，并查看6380和6381的主从状态

```
redis 127.0.0.1:6379> shutdown
redis 127.0.0.1:6379>
```

```
redis 127.0.0.1:6380> info Replication
# Replication
role:slave
master_host:localhost
master_port:6379
master_link_status:down
master_last_io_seconds_ago:-1
master_sync_in_progress:0
master_link_down_since_seconds:3
slave_priority:100
slave_read_only:1
connected_slaves:0
```

```
redis 127.0.0.1:6381> info Replication
# Replication
role:slave
master_host:localhost
master_port:6379
master_link_status:down
master_last_io_seconds_ago:-1
master_sync_in_progress:0
master_link_down_since_seconds:22
slave_priority:100
slave_read_only:1
connected_slaves:0
```

看到6380和6381的master_link_status状态变成了down

3、此时，使用命令行使6380变为master服务器，并关闭只读

```
redis 127.0.0.1:6380> slaveof no one
OK
redis 127.0.0.1:6380> info Replication
# Replication
role:master
connected_slaves:0
redis 127.0.0.1:6380> config set slave-read-only no
OK
```

4、将6381设置成6380的从服务器

```
redis 127.0.0.1:6381> slaveof localhost 6380
OK
redis 127.0.0.1:6381> info Replication
# Replication
role:slave
master_host:localhost
master_port:6380
master_link_status:up
master_last_io_seconds_ago:2
master_sync_in_progress:0
slave_priority:100
slave_read_only:1
connected_slaves:0
redis 127.0.0.1:6381>
```

查看6380的主从状态

```
redis 127.0.0.1:6380> info Replication
# Replication
role:master
connected_slaves:1
slave0:127.0.0.1:6381,online
redis 127.0.0.1:6380>
```

3.7.2 Redis哨兵模式的配置及使用

在实际生产中，如果一旦发生master服务器宕机，则应当以最短的时间更改几个从服务器的主从状态（使用命令行的方式），继续对外服务，否则将造成损失（运维24小时、半夜随时待命）

有没有更好的方式能减轻运维人员的工作负担?

3.7.2 Redis哨兵模式的配置及使用

- Sentinel（哨兵）模式

哨兵模式是由一个或多个哨兵组成的哨兵系统

a. 监控：

监控任意多台主服务器和从服务器是否发生故障

b. 通知：

当主服务器发生故障时，立即向管理员或者相关的应用程序发送通知

c. 自动故障转移：

投票选举的方式从主服务器下属的从服务器中根据优先级选举出一台新的主服务器，让新的主服务器代替之前的主服务器继续处理命令请求及完成相关工作

无须手工操作，达到了高可用、热部署的目的

3.7.2 Redis哨兵模式的配置及使用

- Sentinel的配置文件

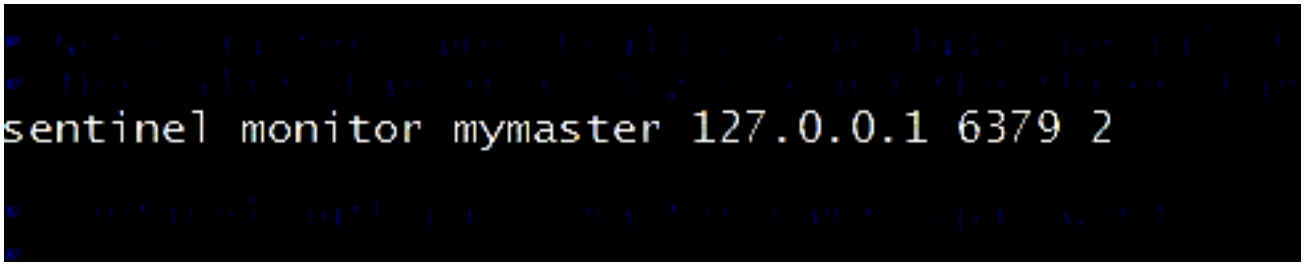
在/usr/local/redis/目录下有个“sentinel.conf”文件，修改这个配置文件

```
hadoop@fisher-VirtualBox:/usr/local/redis$ ls
00-RELEASENOTES  CONTRIBUTING  Makefile         redis.conf          src
6379.log         COPYING       MANIFESTO        runtest             tests
6380.log         deps          README.md        runtest-cluster     utils
6381.log         dump6380.rdb  redis6379.conf   runtest-moduleapi
appendonly.aof   dump.rdb      redis6380.conf   runtest-sentinel
BUGS             INSTALL       redis6381.conf   sentinel.conf
```

3.7.2 Redis哨兵模式的配置及使用

- sentinel monitor mymaster 127.0.0.1 6379 2

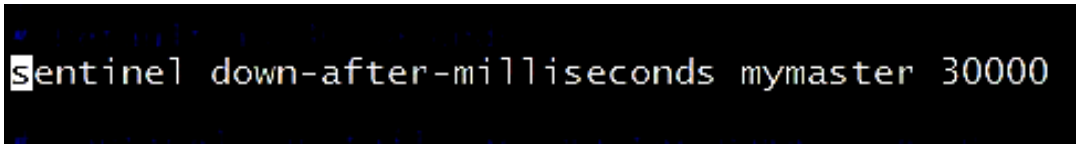
Sentinel监控一台名为mymaster（这个名字自己定，一旦改了，后面几个配置项参数也要改）的主服务器，这台主服务器的IP地址为127.0.0.1，端口号为6379，当发现这台主服务器连接超时，至少需要2个Sentinel同意，才能认定主服务器已经离线



3.7.2 Redis哨兵模式的配置及使用

- sentinel down-after-milliseconds mymaster 30000

该选项指定了Sentinel将服务器标记为主观下线所需要的毫秒数，默认为30s



主观下线并不一定代表这台服务器已经离线，只有当多个Sentinel都将这台服务器标记为主观下线之后，这台服务器就会被标记为客观下线（已经离线）

如果服务器被标记为客观下线，就会触发自动故障转移机制

3.7.2 Redis哨兵模式的配置及使用

- sentinel parallel-syncs mymaster 1

该选项指定了在执行故障转移时，最多可以有多少台从服务器同时对新的主服务器进行复制

```
sentinel parallel-syncs mymaster 1
```

当master离线后，防止突然大量的从服务器一起连接至新master，导致新master一直导RDB和AOF给大量从服务器，负荷剧增

参数的默认值为1，这个值越小，在进行故障转移时所需要的时间就越长

3.7.2 Redis哨兵模式的配置及使用

- sentinel auth-pass
  - 如果被监控的服务器设置有密码，则需要给这个配置项设定好服务器的密码，否则不能动态切换
  - 注意：需要检查该配置项的位置，要放在sentinel monitor mymaster 配置项之后，否则会报错

```
# time while performing the synchronization with the master.
sentinel monitor mymaster 127.0.0.1 6380 1
sentinel auth-pass mymaster passwd
```

3.7.2 Redis哨兵模式的配置及使用

◆ 开启哨兵模式

开启哨兵模式命令：

```
Sentinel mode:
        ./redis-server /etc/sentinel.conf --sentinel
```

- 1、关闭6379的redis服务，修改sentinel.conf文件配置如下，存储后退出；

```
sentinel monitor mymaster 127.0.0.1 6379 1
sentinel down-after-milliseconds mymaster 30000
sentinel parallel-syncs mymaster 1
sentinel auth-pass mymaster passwd
```

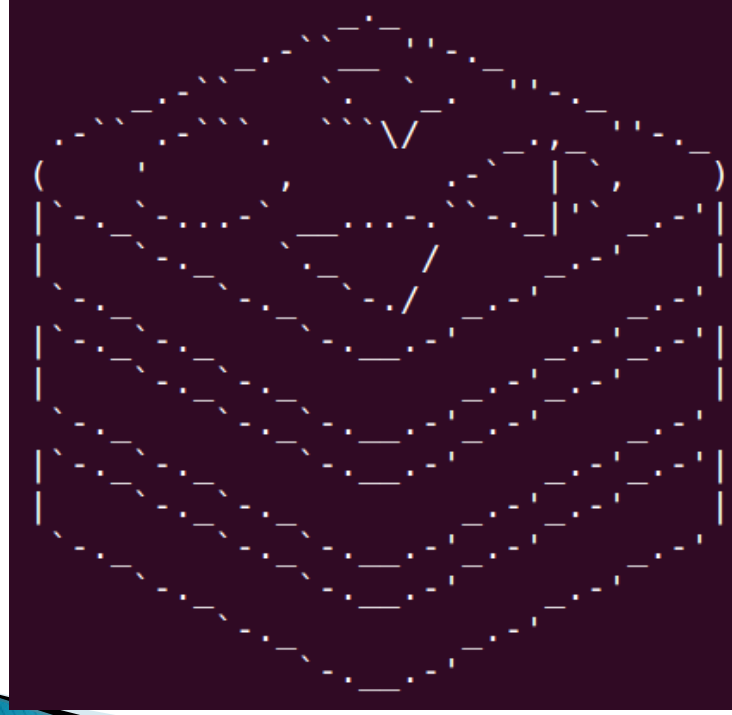
- （被监控服务器没密码可以不加这句）
- 2、重新配置6379为主服务器，6380与6381作为6379的从服务器；

```
hadoop@fisher-VirtualBox:/usr/local/redis$ ./src/redis-server ./redis6379.conf
hadoop@fisher-VirtualBox:/usr/local/redis$ ./src/redis-server ./redis6380.conf
hadoop@fisher-VirtualBox:/usr/local/redis$ ./src/redis-server ./redis6381.conf
```

- 4、根据刚才配置好的sentinel.conf，开启哨兵模式进行监控

./src/redis-server ./sentinel.conf --sentinel

```
hadoop@fisher-VirtualBox:~$ redis-server /usr/local/redis/sentinel.conf --sentinel
2177:X 09 Apr 2021 08:30:22.377 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
2177:X 09 Apr 2021 08:30:22.377 # Redis version=5.0.5, bits=64, commit=00000000, modified=0, pid=2177, just started
2177:X 09 Apr 2021 08:30:22.378 # Configuration loaded
2177:X 09 Apr 2021 08:30:22.378 * Increased maximum number of open files to 10032 (it was originally set to 1024).
```



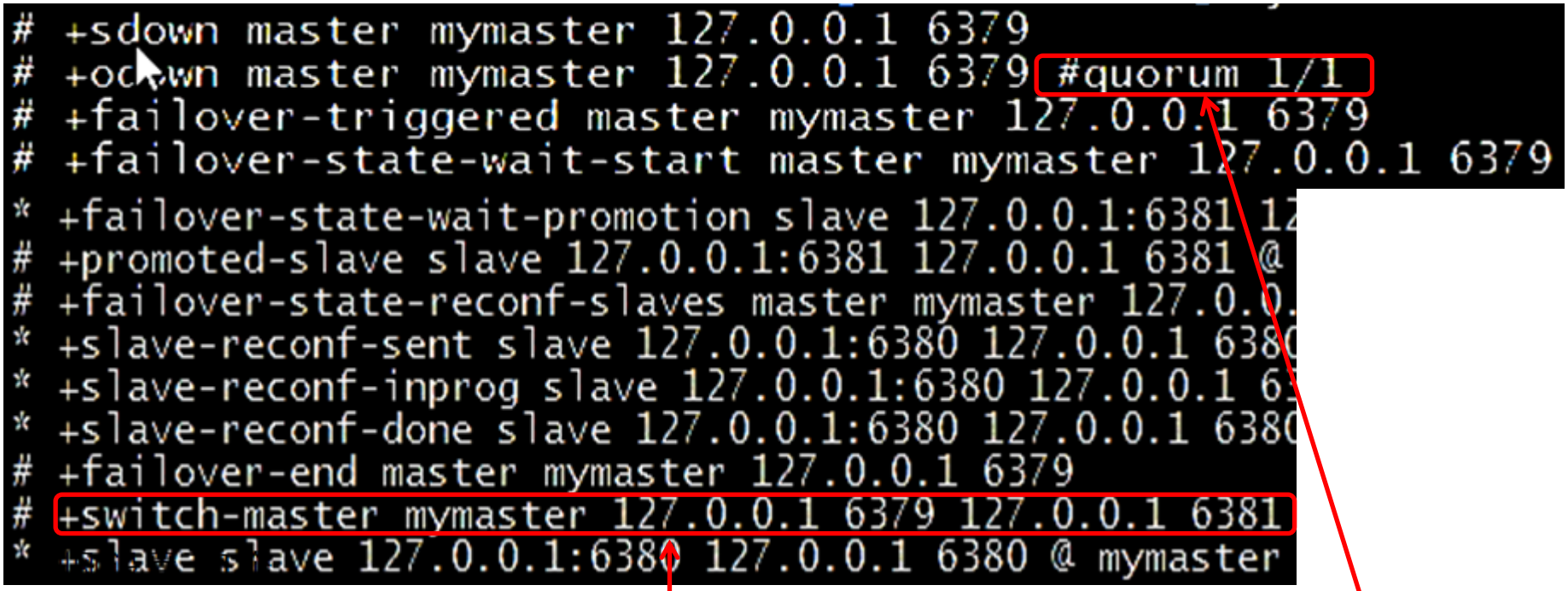
```
Redis 5.0.5 (00000000/0) 64 bit

Running in sentinel mode
Port: 26379
PID: 2177

http://redis.io
```

3.7.2 Redis哨兵模式的配置及使用

- 5、将6379服务器关闭，观察sentinel界面



投票：刚刚设置了1个sentinel同意即可

将6381设置为master服务器

3.7.2 Redis哨兵模式的配置及使用

- 6、观察6380和6381的主从状态

```
redis 127.0.0.1:6380> info Replication
# Replication
role:slave
master_host:127.0.0.1
master_port:6381
master_link_status:up
master_last_io_seconds_ago:2
master_sync_in_progress:0
slave_priority:100
slave_read_only:1
connected_slaves:0
```

```
redis 127.0.0.1:6381> info Replication
# Replication
role:master
connected_slaves:1
slave0:127.0.0.1,6380,online
```

成功在6379关闭后，自动将6381作为新的主服务器，6380作为6381的从服务器

如何能自己指定哪个服务器作新的master?

比如6379离线后，主服务器定为6380，方便脚本等保持一致

答：修改服务器redis.conf里的slave-priority优先级

3.7.2 Redis哨兵模式的配置及使用

- slave-priority （低版本）
- replica-priority （高版本）

设置slave的优先级，默认为100

```
#
# By default the priority is 100.
replica-priority 100
```

当master出现故障（宕机）不能使用时，Sentinel会根据slave的优先级选举一个新的master。 replica-priority的值设置得越小，就越有可能被选中成为master。

注：当replica-priority的值为0时，表示这个slave永远不可能被选中

3.7.2 Redis哨兵模式的配置及使用

- 1、将redis6380.conf中优先级设置为10

```
# By default the priority is 100.
replica-priority 10
```

- 2、开启6379、6380、6381和sentinel
- ./src/redis-server ./sentinel.conf --sentinel

```
redis 127.0.0.1:6379> info Replication
# Replication
role:master
connected_slaves:2
slave0:127.0.0.1,6380,online
slave1:127.0.0.1,6381,online
```

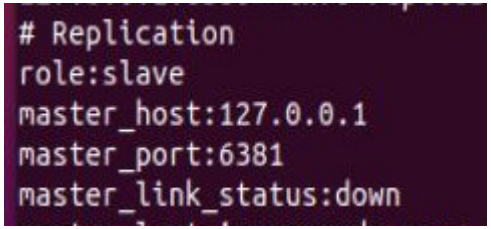
- 3、再关闭6379服务器

```
* +failover-state-wait-promotion slave 127.0.0.1:6380 127.0.0.1 6380 @
# +promoted-slave slave 127.0.0.1:6380 127.0.0.1 6380 @ mymaster 127.0.
# +failover-state-reconf-slaves master mymaster 127.0.0.1 6379
* +slave-reconf-sent slave 127.0.0.1:6381 127.0.0.1 6381 @ mymaster 127
* +slave-reconf-inprog slave 127.0.0.1:6381 127.0.0.1 6381 @ mymaster 1
* +slave-reconf-done slave 127.0.0.1:6381 127.0.0.1 6381 @ mymaster 127
# +failover-end master mymaster 127.0.0.1 6379
# +switch-master mymaster 127.0.0.1 6379 127.0.0.1 6380
* +slave slave 127.0.0.1:6381 127.0.0.1 6381 @ mymaster 127.0.0.1 6380
```

- 成功将6380设置为主服务器，并将6381作为6380的从服务器

◆ 注意

通常情况，info replication中，role正确，而status为down的情况，通常为密码验证未通过



作为备选主服务器的从服务器在平时也需要设置与主服务器一样的密码以防止故障转移时其他从服务器连不上（无密码情况下，有masterauth也会认为密码错误）。

例如：6379作为6380和6381的主服务器密码为passwd，6380平时为从服务器，也作为故障转移后的备选主服务器，它也需要设置密码passwd，同时两边都需要设置masterauth，这样6379重新连上后，会作为6380的从服务，从而通过密码验证

总结

- Redis运维相关

  相关命令

  - TIME
  - DBSIZE
  - BGREWRITEAOF
  - SAVE
  - BGSAVE
  - LASTSAVE
  - Flushall
  - Flushdb
  - INFO
    - info中的部分项目含义
  - CONFIG GET/SET
  - SHUTDOWN

总结

- Redis运维相关

  慢日志

  - slowlog-log-slower-than
  - slowlog-max-len
  - SLOWLOG 命令

- Sentinel模式

  Sentinel模式的具体工作

  Sentinel的配置

  - sentinel monitor
  - sentinel down-after-milliseconds
  - sentinel parallel-syncs
  - 开启Sentinel模式
    - slave-prio