一、 Redis常用命令

（一） 键值相关命令

1、 keys 键名

按照键名查找指定的键。支持通配符

127.0.0.1:6379> set hello 1

OK

127.0.0.1:6379> set hallo 1

OK

127.0.0.1:6379> set heeeello 1

OK

127.0.0.1:6379> keys h?llo

1) "hallo"

2) "hello"

127.0.0.1:6379> keys h\*llo

1) "hallo"

2) "heeeello"

3) "hello"

2、 exists 键名

确认一个键是否存在

127.0.0.1:6379> EXISTS name

(integer) 1 //name键存在

127.0.0.1:6379> EXISTS age

(integer) 0 //age键不存在

3、 del 键名

删除一个键

127.0.0.1:6379> del hello

(integer) 1

127.0.0.1:6379> EXISTS hello

(integer) 0

4、 expire 键 秒

设置一个键的过期时间，如果键已经过期，将会被自动删除

127.0.0.1:6379> set age 18

OK

127.0.0.1:6379> EXPIRE age 20

(integer) 1

127.0.0.1:6379> ttl age

(integer) 18

127.0.0.1:6379> ttl age

(integer) -2

127.0.0.1:6379> EXISTS age

(integer) 0

5、 ttl 键

以秒为单位，返回键的剩余生存时间。

当键不存在时，返回值为-2

当键存在，但没有设置剩余生存时间时，返回-1

127.0.0.1:6379> ttl name

(integer) -1

6、 select 数据库号

选择一个数据库。

默认连接的数据库是0，可以支持共16个数据库。

在配置文件中，通过databases 16 关键字定义

127.0.0.1:6379> select 1

OK

127.0.0.1:6379[1]>

7、 move 键 数据库号

将当前数据库的键移动到指定的数据空中

127.0.0.1:6379> set age 18

OK

127.0.0.1:6379> move age 1

(integer) 1

127.0.0.1:6379> get age

(nil)

127.0.0.1:6379> select 1

OK

127.0.0.1:6379[1]> get age

"18"

8、 randomkey

从当前数据库返回一个随机的键。如果当前库没有任何键，则返回nil

9、 rename 旧名 新名

重命名键

127.0.0.1:6379> rename name name\_new

OK

127.0.0.1:6379> get name\_new

"sc"

10、 type 键

返回键类型。

返回值

none (key不存在)

string (字符串)

list (列表)

set (集合)

zset (有序集)

hash (哈希表)

（二） 服务器相关命令

1、 ping

测试服务器是否可以连接

127.0.0.1:6379> ping

PONG //连接正常

127.0.0.1:6379> ping

Could not connect to Redis at 127.0.0.1:6379: Connection refused

//redis被停止，连接拒绝

2、 echo 字符串

在命令行输出字符串

127.0.0.1:6379> echo "test message"

"test message"

3、 quit

退出redis数据库

4、 save

保存所有的数据。很少在生产环境直接使用SAVE 命令，因为它会阻塞所有的客户端的请求，可以使用BGSAVE 命令代替. 如果在BGSAVE命令的保存数据的子进程发生错误的时,用 SAVE命令保存最新的数据是最后的手段

5、 dbsize

返回当前库中键的数量

127.0.0.1:6379> dbsize

(integer) 6

6、 info

获取服务器的详细信息

7、 config get 参数

获取redis服务器配置文件中的参数。支持通配符

127.0.0.1:6379> config get \* //查询配置文件中所有的参数

1) "dbfilename"

2) "dump.rdb"

45) "port"

46) "6379"

99) "save"

100) "900 1 300 10 60 10000"

8、 flushdb

删除当前数据库中所有的数据

127.0.0.1:6379> dbsize

(integer) 6

127.0.0.1:6379> flushdb

OK

127.0.0.1:6379> dbsize

(integer) 0

9、 flushall

删除所有数据库中所有的数据

二、 redis高级应用

1、 给redis服务器设置密码

1）修改redis服务器的配置文件

[root@localhost redis]# vi /usr/local/redis/etc/redis.conf

# requirepass foobared （391行）

#找到这句话，requirepass后面就是登录redis的密码，改为

requirepass flzx\_3QC

2）重启redis

[root@localhost redis]# pkill redis

[root@localhost redis]# bin/redis-server /usr/local/redis/etc/redis.conf

3）连接redis

[root@localhost redis]# /usr/local/redis/bin/redis-cli

127.0.0.1:6379> keys \* //可以正常连接redis

(error) NOAUTH Authentication required. //但因为没有密码，提示操作拒绝

127.0.0.1:6379> auth flzx\_3QC //利用auth命令输入密码(建议字母加数字) 我们测试可以使用密码123

OK

127.0.0.1:6379> keys \* //才可以正常使用

1) "name"

或

[root@localhost redis]# /usr/local/redis/bin/redis-cli -a flzx\_3QC

#在登录的同时指定密码

#注意历史命令中会明文保存此密码

127.0.0.1:6379> keys \*

1) "name"

2、持久化

Redis 提供了不同级别的持久化方式:

1）RDB持久化方式能够在指定的时间间隔能对你的数据进行快照存储。是默认的持久化方式。这种方式是将内存中数据以快照的方式写入到二进制文件中，默认的文件名为dump.rdb。这种持久化方式被称为快照 snapshotting（快照）。

142row

save 900 1

#900秒内，最少有1个键被改动。则自动保存一次数据集

save 300 10

#300秒内，最少有10个键被改动。则自动保存一次数据集

save 60 10000

#60秒内，最少有10000个键被改动。则自动保存一次数据集

实验：验证dump.rdb数据保存文件

[root@localhost ~]# ls

anaconda-ks.cfg dump.rdb install.log install.log.syslog

#root目录下有dump.rdb文件

[root@localhost ~]# /usr/local/redis/bin/redis-server \

/usr/local/redis/etc/redis.conf

#在root目录中启动redis

[root@localhost ~]# /usr/local/redis/bin/redis-cli

127.0.0.1:6379> auth 123

OK

127.0.0.1:6379> keys \*

1) "name2"

2) "name"

3) "name1"

#0库中有键

[root@localhost ~]# cd /usr/local/redis/

[root@localhost redis]# pkill -9 redis

[root@localhost redis]# /usr/local/redis/bin/redis-server \

/usr/local/redis/etc/redis.conf

#在/usr/local/redis/库中重启redis，

[root@localhost redis]# ls

[root@localhost redis]# /usr/local/redis/bin/redis-cli

127.0.0.1:6379> keys \*

(empty list or set)

#0库中没有键

127.0.0.1:6379> save

OK

#保存

127.0.0.1:6379> quit

[root@localhost redis]# ls

bin dump.rdb etc

#在redis目录中也生成dump.rdb文件

结论：

[root@localhost redis]# vi /usr/local/redis/etc/redis.conf

dir ./

#定义了dump.rdb数据库文件保存在当前位置。所以每次重启redis服务的所在位置不同，导致生成新的dump.rdb文件

dir /usr/local/redis/

#将数据库保存目录写为绝对路径（注意只能是目录）

重启服务

2） 使用AOF 会让你的Redis更加耐久: 你可以使用不同的持久化策略：无备份,每秒备份,每次写的时候备份。使用默认的每秒备份策略,Redis的性能依然很好(备份是由后台线程进行处理的,主线程会尽力处理客户端请求),一旦出现故障，你最多丢失1秒的数据。

appendonly no

#默认不使用AOF持久化（450行）

appendonly yes

#开启AOF持久化

# appendfsync always #每次有写操作，就马上写入磁盘。效率最慢

appendfsync everysec #默认，每秒钟写入磁盘一次。

# appendfsync no #不进行AOF备份，将数据交给操作系统处理。最快，最不 安全

重启

3、主从备份

Redis主从复制特点：

a.Master可以拥有多个slave

b.多个slave可以连接同一个master外，还可以连接到其它slave

c.主从复制不会阻塞master，在同步数据时，master可以继续处理client请求

d.提高系统的伸缩性

Redis主从复制过程：

a.Slave与master建立连接，发送sync同步命令

b.Master会启动一个后台进程，将数据库快照保存到文件中，同时master主进程会开始收集新的写命令并缓存。

c.后台完成保存后，就将此文件发送给slave

d.Slave将此文件保存到硬盘上

1） 不同服务器配置主从

A）克隆一台linux作为从服务器

克隆机需要进行如下操作：

① vi /etc/sysconfig/network-scripts/ifcfg-eth0

删除MAC地址行

② rm -rf /etc/udev/rules.d/70-persistent-net.rules

删除网卡和MAC地址绑定文件

③ 注意关闭防火墙和SELinux

关闭seliuxn永久

vi /etc/selinux/config

SELINUX=disabled

④ 重启动系统

B）在从服务器上配置

[root@localhost ~]# vi /usr/local/redis/etc/redis.conf

# slaveof <masterip> <masterport>

#把此句开启，并指定主服务器ip和端口 （196行）

masterauth flzx\_3QC

#设定主服务器密码

C）重启从服务器上redis

2） 同一台服务器实现主从配置

这里我们以本机配置 1台Master + 1台Slave 为例子,其中:

Master IP:127.0.0.1 PORT:6379

Slave1 IP:127.0.0.1 PORT:63791

A） 复制出从服务器目录

[root@localhost ~]# cp -r /usr/local/redis/ /usr/local/redis-slave1

B） 修改redis-slave1配置文件

[root@localhost ~]# vi /usr/local/redis-slave1/etc/redis.conf

pidfile /usr/local/redis-slave1/redis.pid

#指定pid文件

port 63791

#指定端口号

dir /usr/local/redis-slave1/

#指定服务器目录

slaveof 127.0.0.1 6379

#指定主服务器IP和端口

masterauth flzx\_3QC

#指定主服务器密码

C） 启动服务

/usr/local/redis-slave1/bin/redis-server /usr/local/redis-slave1/etc/redis.conf

#启动从服务器，并调用从服务器配置文件

[root@localhost ~]# netstat -tlun

tcp 0 0 :::6379 :::\* LISTEN

tcp 0 0 :::63791 :::\* LISTEN

#验证两个端口是否都启动

D）验证

从客户端开启:/usr/local/redis/bin/redis-cli -p 63791 -a 123

主客户端开启

set 新值

keys \* 查看是否已同步

4、集群（3.0.0新增功能）

需求：三个主节点，三个从节点对应节点与端口关系如下

127.0.0.1:7000

127.0.0.1:7001

127.0.0.1:7002

127.0.0.1:7003

127.0.0.1:7004

127.0.0.1:7005

1. 创建集群(cluster)目录

mkdir -p /usr/local/cluster

cd /usr/local/cluster

mkdir 7000

mkdir 7001

mkdir 7002

mkdir 7003

mkdir 7004

mkdir 7005

1. 修改配置文件

cp /root/redis-3.0.0/redis.conf /usr/local/cluster/

vi redis.conf

######修改配置文件下面选项

port 7000

daemonize yes

cluster-enabled yes

cluster-config-file nodes.conf

cluster-node-timeout 5000

appendonly yes

######修改完将此文件拷贝至每个子目录下

cp /usr/local/cluster/redis.conf /usr/local/cluster/7000

cp /usr/local/cluster/redis.conf /usr/local/cluster/7001

cp /usr/local/cluster/redis.conf /usr/local/cluster/7002

cp /usr/local/cluster/redis.conf /usr/local/cluster/7003

cp /usr/local/cluster/redis.conf /usr/local/cluster/7004

cp /usr/local/cluster/redis.conf /usr/local/cluster/7005

注意：拷贝完后，修改每个文件的端口号，与目录名对应。

1. 启动6个redis服务

cd /usr/local/cluster/7000/

/usr/local/redis/bin/redis-server redis.conf

cd /usr/local/cluster/7001/

/usr/local/redis/bin/redis-server redis.conf

cd /usr/local/cluster/7002/

/usr/local/redis/bin/redis-server redis.conf

cd /usr/local/cluster/7003/

/usr/local/redis/bin/redis-server redis.conf

cd /usr/local/cluster/7004/

/usr/local/redis/bin/redis-server redis.conf

cd /usr/local/cluster/7005/

/usr/local/redis/bin/redis-server redis.conf

查看redis的启动情况

ps -ef|grep redis

1. 执行redis命令创建集群

cd /root/redis-3.0.0/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 127.0.0.1:7004 127.0.0.1:7005

切换至src目录执行创建集群命令

排错，ruby脚本无ruby环境。

cp CentOS-Base.repo /etc/yum.repos.d/

yum -y install ruby rubygems

gem install redis

再次执行步骤4

交互输入yes

1. 登陆

redis-cli -c -p 7000

7000 和 7003 为主从

7001 和 7004 为主从

7002 和 7005 为主从

主与主之间随机分配主

查看集群节点: cluster nodes

5、事务处理

简单事务控制（批处理）

Redis对事务的支持目前还比较简单。

Redis只能保证一个client发起的事务中的命令可以连续的执行，

而中间不会插入其他client的命令。当一个client在一个连接中发出multi命令时，

这个连接会进入一个事务上下文，该连接后续的命令不会立即执行，

而是先放到一个队列中，当执行exec命令时，

redis会顺序的执行队列中的所有命令。

redis 127.0.0.1:6379> multi

redis 127.0.0.1:6379> set age 10

redis 127.0.0.1:6379> set age 20

redis 127.0.0.1:6379> exec #执行前面的命令

redis 127.0.0.1:6379> discard #放弃

乐观锁复杂事务控制

大多数是基于数据版本（version）的记录机制实现的。何谓数据版本？即为数据增加一个版本标识，在基于数据库表的版本解决方案中，一般是通过为数据库表添加一个“version”字段来实现读取出数据时，将此版本号一同读出，之后更新时，对此版本号加1。

此时，将提交数据的版本号与数据库表对应记录的当前版本号进行比对，如果提交的数据版

本号大于数据库表当前版本号，则予以更新，否则认为是过期数据。

watch age #监视age

multi

set age 20 如果此时有另一进程修改了age，则下一步exec执行将失败

exec

get age

watch 命令会监视给定的key,当exec 时候如果监视的key 从调用watch 后发生过变化，则整个事务会失败。也可以调用watch 多次监视多个key.这样就可以对指定的key 加乐观锁了。

注意watch 的key 是对整个连接有效的，事务也一样。如果连接断开，监视和事务都会被自动清除。当然了exec,discard,unwatch 命令都会清除连接中的所有监视。

redis 只能保证事务的每个命令连续执行，但是如果事务中的一个命令失败了，并不回滚其他命令

6、发布及订阅消息

发布订阅(pub/sub)是一种消息通信模式

一个client 可以订阅多个channel(通道、频道),也可以向多个channel发送消息。

subscribe tv1 订阅一个频道

Reading messages... (press Ctrl-C to quit)

1) "subscribe"

2) "tv1"

3) (integer) 1

subscribe tv1 tv2 订阅两个频道

开启另一个客户端发布内容：

publish tv1 test111111

publish tv2 test222222

也可以用psubscribe tv\*的方式批量订阅以tv 开头的频道的内容

7、虚拟内存的使用

redis 没有使用操作系统提供的虚拟内存机制而是自己在实现了自己的虚拟内存机制

对于大多数数据库而言，最为理想的运行方式就是将所有的数据都加载到内存中，而之后的查询操作则可以完全基于内存数据完成。然而在现实中这样的场景却并不普遍，更多的情况则是只有部分数据可以被加载到内存中。  
    在Redis中，有一个非常重要的概念，即keys一般不会被交换，所以如果你的数据库中有大量的keys，其中每个key仅仅关联很小的value，那么这种场景就不是非常适合使用虚拟内存。如果恰恰相反，数据库中只是包含少量的keys，而每一个key所关联的value却非常大，那么这种场景对于使用虚存就再合适不过了。  
    在实际的应用中，为了能让虚存更为充分的发挥作用以帮助我们提高系统的运行效率，我们可以将带有很多较小值的Keys合并为带有少量较大值的Keys。其中最主要的方法就是将原有的Key/Value模式改为基于Hash的模式，这样可以让很多原来的Keys成为Hash中的属性。

vm-enabled yes #开启vm功能

vm-swap-file /tmp/redis.swap #交换出来的value保存的文件路径

vm-max-memory 1000000 #redis使用的最大内存上限

vm-page-size 32 #每个页面的大小32字节

vm-pages 134217728 #最多使用多少页面

vm-max-threads 4 #用于执行value对象换入患处的工作线程数量

8、配置

daemonize 如果需要在后台运行，把该项改为yes

pidfile 配置多个pid的地址 默认在/var/run/redis.pid

bind 绑定ip，设置后只接受来自该ip的请求

port 监听端口，默认为6379

timeout 设置客户端连接时的超时时间，单位为秒

loglevel 分为4级，debug、verbose、notice、warning

logfile 配置log文件地址

databases 设置数据库的个数，默认使用的数据库为0

save 设置redis进行数据库镜像的频率

rdbcompression 在进行镜像备份时，是否进行压缩

Dbfilename 镜像备份文件的文件名

Dir 数据库镜像备份的文件放置路径

Slaveof 设置数据库为其他数据库的从数据库

Masterauth 主数据库连接需要的密码验证

Requirepass 设置登录时需要使用的密码

Maxclients 限制同时连接的客户数量

Maxmemory 设置redis能够使用的最大内存

Appendonly 开启append only模式

Appendfsync 设置对appendonly.aof文件同步的频率

vm-enabled 是否开启虚拟内存支持

vm-swap-file 设置虚拟内存的交换文件路径

vm-max-memory 设置redis使用的最大物理内存大小

vm-page-size 设置虚拟内存的页大小

vm-pages 设置交换文件的总的page数量

vm-max-threads 设置VM IO同时使用的线程数量

Glueoutputbuf 把小的输出缓存存放在一起

hash-max-zipmap-entries 设置hash的临界值

Activerehashing 重新hash