Redis 学习笔记

文档整理	Falcon.C
官方网站	http://www.linuxtone.org
QQ/Email	389321746 falcon@linuxtone.org
时间	2010.08

Redis 介绍

Redis 本质上一个 Key/Value 数据库，与 Memcached 类似的 NoSQL 型数据库，但是他的数据可以持久化的保存在磁盘上，解决了服务重启后数据不丢失的问题，他的值可以是 string（字符串）、list（列表）、sets（集合）或者是 ordered sets（被排序的集合），所有的数据类型都具有 push/pop、add/remove、执行服务端的并集、交集、两个 sets 集中的差别等等操作，这些操作都是具有原子性的，Redis 还支持各种不同的排序能力

Redis 2.0 更是增加了很多新特性，如：提升了性能、增加了新的数据类型、更少的利用内存（AOF 和 VM）

Redis 支持绝大部分主流的开发语言，如：C、Java、C#、PHP、Perl、Python、Lua、Erlang、Ruby 等等

Redis 性能：

根据 Redis 官方的测试结果：在 50 个并发的情况下请求 10w 次，写的速度是 110000 次/s，读的速度是 81000 次/s

测试环境：

1. 50 个并发，请求 100000 次
2. 读和写大小为 256bytes 的字符串
3. Linux2.6 Xeon X3320 2.5GHz 的服务器上

4. 通过本机的 loopback interface 接口上执行

地址：http://code.google.com/p/redis/wiki/Benchmarks

Redis 的功能：

1、Redis 的 Sharding：Redis 支持客户端的 Sharding 功能，通过一致性 hash 算法实现，当前 Redis 不支持故障冗余，在集群中不能在线增加或删除 Redis

2、Redis 的 master/slave 复制：

1. 一个 master 支持多个 slave
2. Slave 可以接受其他 slave 的连接来替代他连接 master
3. 复制在 master 是非阻塞的，而在 slave 是阻塞的
4. 复制被利用来提供可扩展性，在 slave 端只提供查询功能及数据的冗余

3、Redis 的 Virtual Memory 功能：vm 是 Redis2.0 新增的一个非常稳定和可靠的功能，vm 的引入是为了提高 Redis 的性能，也就是把很少使用的 value 保存到 disk，而 key 保存在内存中。实际上就是如果你有 10w 的 keys 在内存中，而只有仅仅 10%左右的 key 经常使用，那么 Redis 可以通过开启 VM 尝试将不经常使用的 Value 转换到 disk 上保存

4、Redis 的附加档案（AOF）功能：Redis 通过配置的策略将数据集保存到 aof 中，当 Redis 挂掉后能够通过 aof 恢复到挂掉前的状态

Redis 的安装及配置：

下载 Redis：wget http://redis.googlecode.com/files/redis-2.0.0-rc4.tar.gz

```
[falcon@www.fwphp.cn ~/src]$ tar xvzf redis-2.0.0-rc4.tar.gz
[falcon@www.fwphp.cn ~/src]$ cd redis-2.0.0-rc4
[falcon@www.fwphp.cn ~/src/redis-2.0.0-rc4]$ make
[falcon@www.fwphp.cn ~/src/redis-2.0.0-rc4]$ mkdir ~/redis-2.0.0
[falcon@www.fwphp.cn ~/src/redis-2.0.0-rc4]$cp redis-server ~/redis-2.0.0
[falcon@www.fwphp.cn ~/src/redis-2.0.0-rc4]$cp redis-benchmark ~/redis-2.0.0
[falcon@www.fwphp.cn ~/src/redis-2.0.0-rc4]$cp redis-cli ~/redis-2.0.0
[falcon@www.fwphp.cn ~/src/redis-2.0.0-rc4]$cp redis.conf ~/redis-2.0.0
[falcon@www.fwphp.cn ~/src/redis-2.0.0-rc4]$ cd ~/redis-2.0.0/
```

配置 redis.conf 配置文件：

```
#是否作为守护进程运行
daemonize yes
#配置 pid 的存放路径及文件名，默认为当前路径下
pidfile redis.pid
#Redis 默认监听端口
port 6379
#客户端闲置多少秒后，断开连接
timeout 300
#日志显示级别
loglevel verbose
```

#指定日志输出的文件名，也可指定到标准输出端口
logfile stdout
#设置数据库的数量，默认连接的数据库是 0，可以通过 select N 来连接不同的数据库
databases 16
#保存数据到 disk 的策略
#当有一条 Keys 数据被改变是，900 秒刷新到 disk 一次
save 900 1
#当有 10 条 Keys 数据被改变时，300 秒刷新到 disk 一次
save 300 10
#当有 1w 条 keys 数据被改变时，60 秒刷新到 disk 一次
save 60 10000
#当 dump　.rdb 数据库的时候是否压缩数据对象
rdbcompression yes
#dump 数据库的数据保存的文件名
dbfilename dump.rdb
#Redis 的工作目录
dir /home/falcon/redis-2.0.0/
############　Replication ######################
#Redis 的复制配置
slaveof <masterip> <masterport>
masterauth <master-password>

################ SECURITY ############
requirepass foobared

################ LIMITS ###############
#最大客户端连接数
maxclients 128
#最大内存使用率
maxmemory <bytes>

########## APPEND ONLY MODE #########
#是否开启日志功能
appendonly no
#　刷新日志到 disk 的规则
appendfsync always
appendfsync everysec
appendfsync no
################ VIRTUAL MEMORY ############
#是否开启 VM 功能
vm-enabled no
vm-enabled yes
vm-swap-file logs/redis.swap
vm-max-memory 0

```
vm-page-size 32
vm-pages 134217728
vm-max-threads 4
############## ADVANCED CONFIG ################
glueoutputbuf yes
hash-max-zipmap-entries 64
hash-max-zipmap-value 512
#是否重置 Hash 表
activerehashing yes
```

启动 Redis

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-server redis.conf
```

检测 Redis 是否启动：

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ netstat -an -t
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 0.0.0.0:10022 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:6379 0.0.0.0:* LISTEN
.....
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ps -ef|grep redis-server
falcon 7663 1 0 02:29 ? 00:00:00 ./redis-server redis.conf
```

Redis 的数据类型：

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli -h
usage: redis-cli [-h host] [-p port] [-a authpw] [-r repeat_times] [-n db_num] [-i] cmd arg1 arg2 arg3 ... argN
usage: echo "argN" | redis-cli -c [-h host] [-p port] [-a authpw] [-r repeat_times] [-n db_num] cmd arg1 arg2 ... arg(N-1)
```

Redis 根据 5 种不同的数据类型来操作数据对象：

操作 String 类型的值：

Command	Parameters	Description
SET	*key value*	Set a *key* to a string *value*
GET	*key*	Return the string value of the *key*
GETSET	*key value*	Set a key to a string returning the old value of the key
MGET	*key1 key2 ... keyN*	Multi-get, return the strings values of the keys
SETNX	*key value*	Set a key to a string value if the key does not exist
SETEX	*key time value*	Set+Expire combo command

MSET	*key1 value1 key2 value2 ... keyN valueN*	Set multiple keys to multiple values in a single atomic operation
MSETNX	*key1 value1 key2 value2 ... keyN valueN*	Set multiple keys to multiple values in a single atomic operation if none of the keys already exist
INCR	*key*	Increment the integer value of key
INCRBY	*key integer*	Increment the integer value of *key* by *integer*
DECR	*key*	Decrement the integer value of key
DECRBY	*key integer*	Decrement the integer value of *key* by *integer*
APPEND	*key value*	Append the specified string to the string stored at key
SUBSTR	*key start end*	Return a substring of a larger string

操作方法：

SET 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli -n 0 set uid_001 Falcon.C
OK
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli -n 0 set uid_002 Falcon
OK
```

表示向数据库 0 中插入字符串 key 为 uid_001，value 为 Falcon.C 的字符串

GET 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli -n 0 get uid_001
"Falcon.C"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli get uid_001
"Falcon.C"
```

表示获取数据库为 0，key 为 uid_001 的字符串，因为在不指定数据编号的情况下，默认连接的是 0 数据库，所以可以省略-n 参数

GETSET 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli getset uid_002 "falcom520@gmail.com"
"Falcon"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli get uid_002
falcom520@gmail.com
```

表示返回指定 key 的原始值，并指定一个新值给他

MGET 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli mget uid_001 uid_002
1. "Falcon.C"
2. "falcom520@gmail.com"
```

表示获取多个 key 的值

SETNX 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli setnx uid_001_email "falcom520@126.com"
```

(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli get uid_001_email
"falcom520@126.com"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli setnx uid_001_email "falcom520@126.com"
(integer) 0
表示当一个指定的 key 不存在时，设置这个 key 指定的 value，如果存在，则设置不成功

SETEX 操作
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli setex uid_001_msn 5 "falcom520@126.com"
OK
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli get uid_001_msn
"falcom520@126.com"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli get uid_001_msn
(nil)
表示设置一个 key 指定的 value 保存 5 秒后失效，设置 key/value 的有效期

MSET 操作
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli mset uid0001 "0001" uid0002 "0002" uid0003 "0003"
OK
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli mget uid0001 uid0002 uid0003
1. "0001"
2. "0002"
3. "0003"
表示多键值对的数据保存，在保证原子操作性的情况下

MSETNX 操作
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli msetnx uid0003 "0003" uid0004 "0004" uid0005 "0005"
(integer) 0
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli msetnx uid0004 "0004" uid0005 "0005"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli mget uid0001 uid0002 uid0003 uid0004 uid0005
1. "0001"
2. "0002"
3. "0003"
4. "0004"
5. "0005"
表示在单原子操作性的情况下，keys 不存在的前提下插入多个 values 值，如果存在其中一个 keys 则插入失败

INCR 操作
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli incr uid

```
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli incr uid
(integer) 2
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli incr uid
(integer) 3
```

表示对给定 key 的 value 进行递增的操作

INCRBY 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli incrby uid 5
(integer) 8
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli incrby uid 5
(integer) 13
```

表示对给定 key 的 value 进行指定步长的递增操作

DECR 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli decr uid
(integer) 12
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli decr uid
(integer) 11
```

表示对给定的 key 的 value 进行递减操作

DECRBY 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli decrby uid 3
(integer) 8
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli decrby uid 3
(integer) 5
```

表示对给定 key 的 value 做指定步长的递减操作

APPEND 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli append content "01234"
(integer) 5
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli get content
"01234"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli append content "56789"
(integer) 10
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli get content
"0123456789"
```

表示追加一个 value 到指定的 key 中，如果 key 不存在，则新建 key

SUBSTR 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli substr content 0 4
"01234"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli substr content 5 10
"56789"
```

表示返回指定 key 的 value 的部分字符串

操作 lists 类型的值：（列表）

Command	Parameters	Description
RPUSH	*key value*	Append an element to the tail of the List value at key
LPUSH	*key value*	Append an element to the head of the List value at key
LLEN	*key*	Return the length of the List value at key
LRANGE	*key start end*	Return a range of elements from the List at key
LTRIM	*key start end*	Trim the list at key to the specified range of elements
LINDEX	*key index*	Return the element at index position from the List at key
LSET	*key index value*	Set a new value as the element at index position of the List at key
LREM	*key count value*	Remove the first-N, last-N, or all the elements matching value from the List at key
LPOP	*key*	Return and remove (atomically) the first element of the List at key
RPOP	*key*	Return and remove (atomically) the last element of the List at key
BLPOP	*key1 key2 ... keyN*	*timeout* Blocking LPOP
BRPOP	*key1 key2 ... keyN*	*timeout* Blocking RPOP
RPOPLPUSH	*srckey dstkey*	Return and remove (atomically) the last element of the source List stored at *srckey* and push the same element to the destination List stored at *dstkey*

RPUSH 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli rpush list_001 0000001
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli rpush list_001 0000002
(integer) 2
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli rpush list_001 0000003
(integer) 3
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli lrange list_001 0 3
1. "0000001"
2. "0000002"
3. "0000003"
```

表示向指定 key 的 list 的后面（右边）追加指定的 value

LPUSH 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli lpush list_001 000099
(integer) 4
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli lpush list_001 000098
(integer) 5
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli lpush list_001 000097
(integer) 6
```

```
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_001 0 8
1. "000097"
2. "000098"
3. "000099"
4. "0000001"
5. "0000002"
6. "0000003"
```

表示向指定 key 的 list 的前面（左边）追加指定的 value

LLEN 操作

```
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli llen list_001
(integer) 6
```

表示返回指定 key list 的长度

LRANGE 操作

```
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_001 2 4
1. "000099"
2. "0000001"
3. "0000002"
```

表示返回指定 key list 里面的位置的范围 value

LTRIM 操作

```
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli ltrim list_001 0 2
OK
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_001 0 4
1. "000097"
2. "000098"
3. "000099"
```

表示删除指定 key 的值范围以外的数据

LINDEX 操作

```
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_001 0 9
1. "000097"
2. "000098"
3. "000099"
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lindex list_001 2
"000099"
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lindex list_001 1
"000098"
```

表示返回指定 key list 里面索引的值

LSET 操作

```
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_001 0 9
1. "000097"
```

```
2. "000098"
3. "000099"
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lset list_001 0 "100097"
OK
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_001 0 9
1. "100097"
2. "000098"
3. "000099"
```

表示给指定 key 的 list 里面指定索引的值修改为一个新值

LREM 操作

```
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lpush list_001 000099
(integer) 4
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lpush list_001 000099
(integer) 5
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lpush list_001 000099
(integer) 6
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_001 0 9
1. "000099"
2. "000099"
3. "000099"
4. "100097"
5. "000098"
6. "000099"
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrem list_001 2 000099
(integer) 2
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_001 0 9
1. "000099"
2. "100097"
3. "000098"
4. "000099"
```

表示删除指定 key 的 list 里面值为 value 的指定个数

LPOP 操作

```
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_001 0 9
1. "000099"
2. "100097"
3. "000098"
4. "000099"
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lpop list_001
"000099"
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_001 0 9
1. "100097"
2. "000098"
```

3. "000099"
表示删除指定 key 的 list 里面最前面（左边）的值，并返回该值

RPOP 操作

```
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_001 0 9
1. "100097"
2. "000098"
3. "000099"
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli rpop list_001
"000099"
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_001 0 9
1. "100097"
2. "000098"
```

表示删除指定 key 的 list 里面最后面（右边）的值，并返回该值

BLPOP 和 BRPOP 操作
在阻塞的模式下执行 LPOP 和 RPOP 操作

RPOPLPUSH 操作

```
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_001 0 9
1. "100097"
2. "000098"
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli rpoplpush list_001 list_999
"000098"
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_001 0 9
1. "100097"
[falcon@www.fwphp.cn  ~/redis-2.0.0]$ ./redis-cli lrange list_999 0 6
1. "000098"
```

表示将原 key 的 list 后面（右边）的值删掉，并保存到指定的目的 key 中，并返回该值

操作 sets 类型的值：（sets 集合）

Command	Parameters	Description
SADD	*key member*	Add the specified member to the Set value at key
SREM	*key member*	Remove the specified member from the Set value at key
SPOP	*key*	Remove and return (pop) a random element from the Set value at key
SMOVE	*srckey dstkey member*	Move the specified member from one Set to another atomically
SCARD	*key*	Return the number of elements (the cardinality) of the Set at key
SISMEMBER	*key member*	Test if the specified value is a member of the Set at key
SINTER	*key1 key2 ... keyN*	Return the intersection between the Sets stored at key1, key2, ..., keyN

SINTERSTORE	*dstkey key1 key2 ... keyN*	Compute the intersection between the Sets stored at key1, key2, ..., keyN, and store the resulting Set at dstkey
SUNION	*key1 key2 ... keyN*	Return the union between the Sets stored at key1, key2, ..., keyN
SUNIONSTORE	*dstkey key1 key2 ... keyN*	Compute the union between the Sets stored at key1, key2, ..., keyN, and store the resulting Set at dstkey
SDIFF	*key1 key2 ... keyN*	Return the difference between the Set stored at key1 and all the Sets key2, ..., keyN
SDIFFSTORE	*dstkey key1 key2 ... keyN*	Compute the difference between the Set key1 and all the Sets key2, ..., keyN, and store the resulting Set at dstkey
SMEMBERS	*key*	Return all the members of the Set value at key
SRANDMEMBER	*key*	Return a random member of the Set value at key

SADD 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sadd s_001 "Falcon.C"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sadd s_001 "Falcon"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "Falcon"
2. "Falcon.C"
```

表示向指定 key 的集合中添加成员

SREM 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "Falcon"
2. "Falcon.C"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli srem s_001 Falcon
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "Falcon.C"
```

表示删除指定 key 的指定 Value 成员值

SPOP 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "Falcon.C"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sadd s_001 "www.linuxtone.org"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sadd s_001 "bbs.linuxtone.org"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sadd s_001 "uc.linuxtone.org"
```

```
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "www.linuxtone.org"
2. "Falcon.C"
3. "bbs.linuxtone.org"
4. "uc.linuxtone.org"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli spop s_001
"www.linuxtone.org"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "Falcon.C"
2. "bbs.linuxtone.org"
3. "uc.linuxtone.org"
```

表示从指定 key 的 set 集中随机删除一个成员 value 并返回

SMOVE 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "Falcon.C"
2. "bbs.linuxtone.org"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smove s_001 s_002 bbs.linuxtone.org
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "Falcon.C"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_002
1. "bbs.linuxtone.org"
```

表示从一个指定的 key 中移动一个指定的 value 成员到另一个指定的 key 中，这些操作是具有原子性的

SCARD 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli scard s_001
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli scard s_002
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "Falcon.C"
```

表示返回指定 key 的 set 集的 value 成员个数

SISMEMBER 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "Falcon.C"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sismember s_001 Falcon
(integer) 0
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sismember s_001 Falcon.C
(integer) 1
```

表示判断指定的 key 的成员是否存在于 sets 集中

SINTER 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sadd s_001 "000001"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sadd s_001 "000002"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "Falcon.C"
2. "000001"
3. "000002"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_002
1. "bbs.linuxtone.org"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sadd s_002 "000001"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sadd s_002 "000002"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_002
1. "000001"
2. "bbs.linuxtone.org"
3. "000002"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sinter s_001 s_002
1. "000001"
2. "000002"
```

表示对指定的 key 的 sets 集执行交集操作，返回指定 sets 集合中相同的 value 成员

SINTERSTORE 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sadd s_001 "000003"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sadd s_001 "00000099"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "000003"
2. "Falcon.C"
3. "000001"
4. "000002"
5. "00000099"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sadd s_002 "000003"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sadd s_002 "00000099"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_002
1. "000003"
2. "000001"
3. "bbs.linuxtone.org"
```

4. "000002"
5. "00000099"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_002
1. "000003"
2. "000001"
3. "bbs.linuxtone.org"
4. "000002"
5. "00000099"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sinterstore s_003 s_001 s_002
(integer) 4
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_003
1. "000003"
2. "000001"
3. "00000099"
4. "000002"

表示将指定的 key 的 sets 集做交集，并将结果保存到指定的 key 中

SUNION 操作

[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "000003"
2. "Falcon.C"
3. "000001"
4. "000002"
5. "00000099"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_002
1. "000003"
2. "bbs.linuxtone.org"
3. "000001"
4. "000002"
5. "00000099"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sunion s_001 s_002
1. "000003"
2. "Falcon.C"
3. "000001"
4. "bbs.linuxtone.org"
5. "000002"
6. "00000099"

表示对指定的 key 的 sets 集合做并集

SUNIONSTORE 操作

[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "000003"
2. "Falcon.C"
3. "000001"

4. "000002"
5. "00000099"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_002
1. "000003"
2. "bbs.linuxtone.org"
3. "000001"
4. "000002"
5. "00000099"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sunionstore s_004 s_001 s_002
(integer) 6
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_004
1. "000003"
2. "Falcon.C"
3. "000001"
4. "bbs.linuxtone.org"
5. "000002"
6. "00000099"

表示对指定的 key 的 sets 集做并集，并将结果保存到指定的 key 中

SDIFF 操作

[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_001
1. "000003"
2. "Falcon.C"
3. "000001"
4. "000002"
5. "00000099"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_002
1. "000003"
2. "bbs.linuxtone.org"
3. "000001"
4. "000002"
5. "00000099"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sdiff s_001 s_002 s_003 s_004
(empty list or set)
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sdiff s_001 s_002
1. "Falcon.C"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli sdiff s_001 s_003
1. "Falcon.C"

表示对给定的第一个 key 的 sets 集合与其他的 key 的 sets 集合的 value 进行对比，并返回不同的 value 的成员

SDIFFSTORE 操作

Sdiffstore 与 sdiff 操作一样，只是把不同的 sets 集合成员保存到一个给定的 key 的 sets 集合

中

SMEMBERS 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_004
1. "000003"
2. "Falcon.C"
3. "000001"
4. "bbs.linuxtone.org"
5. "000002"
6. "00000099"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_003
1. "000003"
2. "000001"
3. "00000099"
4. "000002"
```

表示返回指定 key 的所有 sets 集合的成员

SRANDMEMBER 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli smembers s_003
1. "000003"
2. "000001"
3. "00000099"
4. "000002"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli srandmember s_003
"000001"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli srandmember s_003
"000002"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli srandmember s_003
"000002"
```

表示返回一个给定 key 的 sets 集合中随机的一个成员

操作 zsets 类型的值：（排序后的 sets 集合）

Command	Parameters	Description
ZADD	*key score member*	Add the specified member to the Sorted Set value at key or update the score if it already exist
ZREM	*key member*	Remove the specified member from the Sorted Set value at key
ZINCRBY	*key increment member*	If the member already exists increment its score by *increment*, otherwise add the member setting *increment* as score
ZRANK	*key member*	Return the rank (or index) or *member* in the sorted set at *key*, with scores being ordered from low to high

ZREVRANK	*key member*	Return the rank (or index) or *member* in the sorted set at *key*, with scores being ordered from high to low
ZRANGE	*key start end*	Return a range of elements from the sorted set at key
ZREVRANGE	*key start end*	Return a range of elements from the sorted set at key, exactly like ZRANGE, but the sorted set is ordered in traversed in reverse order, from the greatest to the smallest score
ZRANGEBYSCORE	*key min max*	Return all the elements with score >= min and score <= max (a range query) from the sorted set
ZCOUNT	*key min max*	Return the number of elements with score >= min and score <= max in the sorted set
ZCARD	*key*	Return the cardinality (number of elements) of the sorted set at key
ZSCORE	*key element*	Return the score associated with the specified element of the sorted set at key
ZREMRANGEBYRANK	*key min max*	Remove all the elements with rank >= min and rank <= max from the sorted set
ZREMRANGEBYSCORE	*key min max*	Remove all the elements with score >= min and score <= max from the sorted set

ZUNIONSTORE / ZINTERSTORE *dstkey N key1 ... keyN* WEIGHTS *w1 ... wN* AGGREGATE SUM|MIN|MAX Perform a union or intersection over a number of sorted sets with optional weight and aggregate

ZADD 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zadd z_001 1 "Falcon.C"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zadd z_001 1 "Falcon"
(integer) 0
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zadd z_001 1 "LinuxTone"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 4
1. "Falcon"
2. "Falcon.C"
3. "LinuxTone"
```

表示通过给定的积分顺序插入成员值到指定的 key 的顺序 sets 集合中，如果成员存在则插入失败

ZREM 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 4
1. "Falcon"
2. "Falcon.C"
3. "LinuxTone"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrem z_001 Falcon
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 4
```

1. "Falcon.C"
2. "LinuxTone"

表示从给定的 key 的顺序 sets 集合中删除指定的成员

ZINCRBY 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 4
1. "Falcon.C"
2. "LinuxTone"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zincrby z_001 2 Falcon.C
"3"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 8
1. "LinuxTone"
2. "Falcon.C"
```

表示给指定的 key 的成员的排序积分进行给定的递增值的递增，如果积分为零，则将递增值作为积分排序

ZRANK 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 8
1. "LinuxTone"
2. "0"
3. "Falcon.C"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrank z_001 Falcon.C
(integer) 2
```

表示获取当前指定 key 的成员在排序 sets 集合中的排名，从 0 开始计数（正序）

ZREVRANK 操作

与 ZRANK 一样，从 0 开始计数（倒序）

ZRANGE 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 8
1. "LinuxTone"
2. "0"
3. "Falcon.C"
```

表示通过开始值和结束值来获取指定 key 的排序 sets 集合中的成员范围

ZREVRANGE 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 2
1. "LinuxTone"
2. "0"
3. "Falcon.C"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 1
1. "LinuxTone"
2. "0"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrevrange z_001 0 1
```

1. "Falcon.C"
2. "0"

表示通过对指定 key 的排序 sets 集合倒序后在获取指定范围的成员集

ZRANGEBYSCORE 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zadd z_001 1 "linuxtone.org"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zadd z_001 2 "qq.com"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zadd z_001 3 "google.com"
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 10
1. "linuxtone.org"
2. "LinuxTone"
3. "qq.com"
4. "google.com"
5. "0"
6. "Falcon.C"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrangebyscore z_001 2 4
1. "LinuxTone"
2. "qq.com"
3. "google.com"
```

表示获取指定 key 的积分范围的排序 sets 集合的成员

ZCOUNT 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 10
1. "linuxtone.org"
2. "LinuxTone"
3. "qq.com"
4. "google.com"
5. "0"
6. "Falcon.C"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zcount z_001 2 4
(integer) 3
```

表示获取指定 key 的积分范围的排序 sets 集合的成员数量

ZCARD 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 10
1. "linuxtone.org"
2. "LinuxTone"
3. "qq.com"
4. "google.com"
5. "0"
6. "Falcon.C"
```

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zcard z_001
(integer) 6
```

表示获取指定 key 的排序 sets 集合的成员基数

ZSCORE 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 10
1. "linuxtone.org"
2. "LinuxTone"
3. "qq.com"
4. "google.com"
5. "0"
6. "Falcon.C"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zscore z_001 0
"5"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zscore z_001 qq.com
"2"
```

表示获取指定 key 的排序 sets 集合中成员的积分

ZREMRANGEBYRANK 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 10
1. "linuxtone.org"
2. "LinuxTone"
3. "qq.com"
4. "google.com"
5. "0"
6. "Falcon.C"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zremrangebyrank z_001 4 5
(integer) 2
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 10
1. "linuxtone.org"
2. "LinuxTone"
3. "qq.com"
4. "google.com"
```

表示删除指定 key 的排序 sets 集合中成员的排名范围的成员（通过排名范围删除成员）

ZREMRANGEBYSCORE 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 10
1. "linuxtone.org"
2. "LinuxTone"
3. "qq.com"
4. "google.com"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zscore z_001 linuxtone.org
"1"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zscore z_001 LinuxTone
```

"2"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zscore z_001 qq.com
"2"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zscore z_001 google.com
"3"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zremrangebyscore z_001 2 2
(integer) 2
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli zrange z_001 0 10
1. "linuxtone.org"
2. "google.com"
表示删除通过指定 key 的排序 sets 集合中给定范围积分的成员（通过积分范围删除成员）

ZUNIONSTORE/ZINTERSTORE 操作
表示通过指定的 keys 做交际或者并集，并将结果保存到指定的结果集中

操作 hash 类型的值：

Command	Parameters	Description
HSET	*key field value*	Set the hash field to the specified value. Creates the hash if needed.
HGET	*key field*	Retrieve the value of the specified hash field.
HMGET	*key field1 ... fieldN*	Get the hash values associated to the specified fields.
HMSET	*key field1 value1 ... fieldN valueN*	Set the hash fields to their respective values.
HINCRBY	*key field integer*	Increment the integer value of the hash at *key* on *field* with *integer*.
HEXISTS	*key field*	Test for existence of a specified field in a hash
HDEL	*key field*	Remove the specified field from a hash
HLEN	*key*	Return the number of items in a hash.
HKEYS	*key*	Return all the fields in a hash.
HVALS	*key*	Return all the values in a hash.
HGETALL	*key*	Return all the fields and associated values in a hash.

HSET 操作
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hset h_uid uid001 'Falcon.C'
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hset h_uid uid002 'NetSeek'
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hset h_uid uid003 'LinuxTone'
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hkeys h_uid
1. "uid001"

2. "uid002"
3. "uid003"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hvals h_uid
1. "Falcon.C"
2. "NetSeek"
3. "LinuxTone"
表示给指定的 hash 字段设置值，如果不存在则创建

HGET 操作
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hget h_uid uid001
"Falcon.C"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hget h_uid uid002
"NetSeek"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hget h_uid uid003
"LinuxTone"
表示获取指定 hash 字段的值

HMGET 操作
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hmget h_uid uid001 uid002 uid003
1. "Falcon.C"
2. "NetSeek"
3. "LinuxTone"
表示批量获取指定 hash 字段的值

HMSET 操作
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hmset h_uid uid004 'linuxtone.org' uid005 'qq.com'
OK
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hkeys h_uid
1. "uid001"
2. "uid002"
3. "uid003"
4. "uid004"
5. "uid005"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hvals h_uid
1. "Falcon.C"
2. "NetSeek"
3. "LinuxTone"
4. "linuxtone.org"
5. "qq.com"
表示批量设置 hash 字段的值

HINCRBY 操作
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hincrby h_uid_incr uid 1

```
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hincrby h_uid_incr uid 1
(integer) 2
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hincrby h_uid_incr uid 3
(integer) 5
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hvals h_uid_incr
1. "5"
```

表示对指定的 hash 字段的值进行递增操作

HEXISTS 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hexists h_uid uid001
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hexists h_uid uid0011
(integer) 0
```

表示判断指定的 hash 字段是否存在

HDEL 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hset h_uid uid 1
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hvals h_uid
1. "Falcon.C"
2. "NetSeek"
3. "LinuxTone"
4. "linuxtone.org"
5. "qq.com"
6. "1"
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hdel h_uid uid
(integer) 1
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hvals h_uid
1. "Falcon.C"
2. "NetSeek"
3. "LinuxTone"
4. "linuxtone.org"
5. "qq.com"
```

表示通过指定的 hash 字段删除 hash 值

HLEN 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hlen h_uid
(integer) 5
```

表示返回 hash 长度

LKEYS 操作

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hkeys h_uid
1. "uid001"
```

2. "uid002"
3. "uid003"
4. "uid004"
5. "uid005"

表示返回指定 hash 的所有 key

HVALS 操作

[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hvals h_uid
1. "Falcon.C"
2. "NetSeek"
3. "LinuxTone"
4. "linuxtone.org"
5. "qq.com"

表示返回指定 hash 的所有 value

HGETALL 操作

[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli hgetall h_uid
1. "uid001"
2. "Falcon.C"
3. "uid002"
4. "NetSeek"
5. "uid003"
6. "LinuxTone"
7. "uid004"
8. "linuxtone.org"
9. "uid005"
10. "qq.com"

表示返回指定 hash 的所有字段及关联的值

公共操作命令部分：（不受数据类型的影响）

Command	Parameters	Description
SORT	*key* BY *pattern* LIMIT *start end* GET *pattern* ASC|DESC ALPHA	Sort a Set or a List accordingly to the specified parameters
数据持久化控制命令		
SAVE	-	Synchronously save the DB on disk
BGSAVE	-	Asynchronously save the DB on disk
LASTSAVE	-	Return the UNIX time stamp of the last successfully saving of the dataset on disk
SHUTDOWN	-	Synchronously save the DB on disk, then shutdown the server
BGREWRITEAOF	-	Rewrite the append only file in background when it gets too big

远程服务器控制命令

INFO	-	Provide information and statistics about the server
MONITOR	-	Dump all the received requests in real time
SLAVEOF	-	Change the replication settings
CONFIG	-	Configure a Redis server at runtime

Redis 的 master/slave 复制：

Redis 的 master/slave 数据复制方式可以是一主一从或者是一主多从的方式，Redis 在 master 是非阻塞模式，也就是说在 slave 执行数据同步的时候，master 是可以接受客户端的请求的，并不影响同步数据的一致性，然而在 slave 端是阻塞模式的，slave 在同步 master 数据时，并不能够响应客户端的查询

Redis 的 master/slave 模式下，master 提供数据读写服务，而 slave 只提供读服务

Redis 的 master/slave 的配置方式是在 slave 主机的 Redis 目录下的 redis.conf 配置文件中添加：

```
slaveof master_ip master_port
```

例如：

我们配置我们的 slave 为：redis-slave.conf

```
daemonize yes
pidfile redis-slave.pid
port 6380
timeout 300
loglevel verbose
logfile stdout
databases 16
save 900 1
save 300 10
save 60 10000
rdbcompression yes
dbfilename dump-slave.rdb
dir /home/falcon/redis-2.0.0/
slaveof 127.0.0.1 6379
appendonly no
appendfsync everysec
vm-enabled no
vm-swap-file logs/redis-slave.swap
vm-max-memory 0
vm-page-size 32
```

```
vm-pages 134217728
vm-max-threads 4
glueoutputbuf yes
hash-max-zipmap-entries 64
hash-max-zipmap-value 512
activerehashing yes
```

启动 slave：

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-server redis-slave.conf
```

查看状态信息：

```
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli -p 6380 info
redis_version:1.3.17
redis_git_sha1:00000000
redis_git_dirty:0
arch_bits:32
multiplexing_api:epoll
process_id:10772
uptime_in_seconds:249
uptime_in_days:0
connected_clients:2
connected_slaves:0
blocked_clients:0
used_memory:1756868
used_memory_human:1.68M
changes_since_last_save:0
bgsave_in_progress:0
last_save_time:1281654285
bgrewriteaof_in_progress:0
total_connections_received:13
total_commands_processed:9
expired_keys:0
hash_max_zipmap_entries:64
hash_max_zipmap_value:512
pubsub_channels:0
pubsub_patterns:0
vm_enabled:0
role:slave
master_host:127.0.0.1
master_port:6379
master_link_status:up
master_last_io_seconds_ago:248
db0:keys=23,expires=0
[falcon@www.fwphp.cn ~/redis-2.0.0]$ ./redis-cli -p 6379 info
```

redis_version:1.3.17
redis_git_sha1:00000000
redis_git_dirty:0
arch_bits:32
multiplexing_api:epoll
process_id:7663
uptime_in_seconds:16787
uptime_in_days:0
connected_clients:1
connected_slaves:1
blocked_clients:0
used_memory:1757232
used_memory_human:1.68M
changes_since_last_save:0
bgsave_in_progress:0
last_save_time:1281654286
bgrewriteaof_in_progress:0
total_connections_received:835
total_commands_processed:55953
expired_keys:1
hash_max_zipmap_entries:64
hash_max_zipmap_value:512
pubsub_channels:0
pubsub_patterns:0
vm_enabled:0
role:master
db0:keys=23,expires=0

解释:
1、 红色标注为 master/slave 端口和运行模式
2、 蓝色标注为 master/slave 数据是否同步，目前显示为 keys 23 个，0 个过期

扩展思维：

Redis 可以做一主一从，也可以做一主多从，更可以做一主一从，在从下面挂从，大家可以根据需求做这样的试验