链滴

Redis 指令总结

作者：wangsch
原文链接：https://ld246.com/article/1475325884609
来源网站：链滴

Redis指令总结
<h2>Redis概述</h2>
Redis目前（2016-10-01）最新稳定版：3.2.4，jedis客户端：2.9.0（支持BITFIELD），测试使：2.8.17

应用场景：实施分析、缓存、消息队列、普通内存存储等

五种数据类型：string、list、set、zset、hash

特性：pipline、pub/sub、transaction、cluster等

整理归纳，主要来自redis.io网站，主要是归纳加强记忆，返回值、注意事或更多详情还要参考官方文档，commands文档：http://redis.io/commands

<h1>Redis数据类型操作</h1>
<h2>Bitmap</h2>
Redis中没有独立的bitmap类型，用字符串替代。bitmap的值，只有两个：0代表不存在，0代存在。bitmap可以用于实时计算，如：每天活跃用户数很适合使用redis bitmap实现。Redis中bitmap的应用，参考一篇比较好的博客：http://blog.getspool.com/211/11/29/fast-easy-realtime-metrics-using-redis-bitmaps/

SETBIT key offset value
设置bitmap指定offset处的值

GETBIT key offset

BITCOUNT key [start end]

BITFIELD key [GET type offset] [SET type offset value] [INCRBY type offset increment] [OVERFLOW WRAP|SAT|FAIL]（since 3.2.0）
普通的bitmap的value只支持0和1，BITFIELD则支持更多的值类型，使“type”参数指定。
The motivation for this command is that the ability to store many small integers as a single large bitmap (or segmented over a few keys to avoid having huge keys) is extremely memory efficient, and opens new use cases for Redis to be applied, especially in the field of real time analytics.

BITOP operation destkey key [key ...]
可以对多个key进行位运算操作，通过“operation”参数致命具体操作，包括：AND OR XOR NOT四种。主要用于对bitmap的运，进行一些实时分析计算。举例：

<pre class="brush: bash">10.44.90.20:6380> setbit active 22 1
(integer) 0
10.44.90.20:6380>
10.44.90.20:6380> setbit play 22 1
(integer) 0
10.44.90.20:6380> setbit active 0 1
(integer) 0
10.44.90.20:6380> BITOP and play-active play active
(integer) 3
10.44.90.20:6380> bitcount play-active
(integer) 1</pre>

说明：如果某天用户活跃，则将用户放到active这个bitmap中。如果某天用户至少播放了一首歌则放到play这个bitmap中。active和play两个bitmap进行“AND”运算，结果用play-active标识。结果是：既算作活跃用户，而且至少播放了一首歌的用户组成的新的bitmap。显然，用户2符合条件，用户0不符合条件，所以，在最后“bitcount play-active”指令，返回值为。

BITPOS key bit [start] [end]
从左到右数，返回某个bitmap，第一个0或者1所在的索引位置，索引从0开始，“key包位数 - 1”结束。start和end代表有效的字节范围，不在这个字节范围内，即使出现，也会被略。如：BITPOS mykey 0 2 5，这个命令代表，从mykey中，找第一个为0的位，寻找范围是第三个节到第六个字节，其他字节出现0不算。

start和end均可以省略，而对于寻找0或者1，start和end有不同的影响。具体参考：http://redis.io/commands/bitpos

<h2>string</h2>

<h2>list</h2>
list可以实现普通列表、栈和队列等功能，并且提供阻塞模式，可以基于阻塞模式list实现消息队服务。

LPUSH

LPOP

RPOP

BLPOP key [key ...] timeout
（阻塞多个key规则比较复杂，如果使用请仔细阅读命令文档）

<h2>hash</h2>
HSET key field

HGET key field

HGETALL key

HMSET key field [key field ...]

HMGET key field [field ...]

HKEYS key

HVALS key

...

<h2>set</h2>
SADD key member [member ...]
向set中添加元素，如果元素已经存在，则忽略，返回成功添加的元素数量（不包括已经存在的）
/p>

SCARD key
set中元素的数量

SISMEMBER key member
指定元素是否包含在key对应的set中。

SMEMBERS key
返回key对应的set中所有的元素，和只带有一个key的SINTER命令有相同的结果。

SMOVE source dest member
以原子方式，将source对应的set中的元素，移动到由dest代表的set中，元素通过member参数
定。如果member在source对应的set不存在，或者source代表的key不存在，不做任何操作。如果de
t代表的set中已经存在member，则只将member从source代表的set中移除。返回1如果成功，返回
，如果操作无效。

SPOP key [count]
随机返回一个或多个元素，并从key对应的set中移除。说明：count参数将在3.2正式支持，目前
非稳定分支支持。和SRANDMEMBER命令类似，不过后者不会移除元素。指定count参数后，具体
为和返回元素的规则见文档：http://redis.io/comma
ds/spop

SRANDMEMBER key [count]
只有一个key参数时，和SPOP类似，但是不移除元素。另外，对于count，如果正数返回非重复
素。如果负数，返回结果可以包括重复元素。指定count参数后，具体行为和返回元素的规则见文档
http://redis.io/commands/srandmember
strong>

SREM key member [member ...]
从key代表的set中移除指定的一个或多个元素，不存在的元素被忽略，返回成功移除的元素个数

SCAN key cursor [MATCH pattern] [COUNT count]
参考：http://redis.io/commands/scan</
>

SINTER key [key ...]
取多个set中元素的交集，即：元素必须在所有的set中都存在。最终返回交集中的元素数组。</
>

SINTERSTORE dest key [key ...]
和“SINTER”命令相同，不同点是，将最终结果保存到dest参数指定的key中，形
一个新的set。

SUNION key [key ...]
取多个key指定的set的并集，返回并集中的所有元素组成的数组。

SUNIONSTORE dest key [key ...]
和SUNION相同，但是会将结果存储到dest参数指定的key中，形成新的set，如果dest指定的ke
存在将会被覆盖，最终返回dest中元素个数。

SDIFF key [key ...]
第一个key代表的set和后续多个key代表的set之间的差集，即：在第一个set中存在，而在后续
个set中都不存在的元素。差集中元素作为数组返回。

<h2>zset</h2>
ZADD key [NX|XX|CH] [INCR] score member [score member ...]

ZADD options(Redis 3.0.2 or greater)
XX:
NX:
CH:
INCR:

同分的多个element，按照字典序排序。字典排序，从左到右比较elemen
的字节，会把string作为字节数组。

ZCARD key

ZSCORE key member

ZSCAN key cursor [MATCH pattern] [COUNT count]

ZCOUNT key min max

ZLEXCOUNT key min max
当key对应的sorted set所有元素的score都相同时，获取元素值在min和max之间的元素个数。<
p>

ZINCRBY key incr member
增加指定元素的分值，返回最终的分值。

ZRANK key member

ZREVRANK key member

ZREM key member [member ...]

ZREMRANGEBYLEX key min max

ZREMRANGEBYRANK key start stop

ZREMRANGEBYSCORE key min max

ZRANGE key start stop [WITHSCORES]

ZREVRANGE key start stop [WITHSCORES]

ZRANGEBYLEX key min max [LIMIT offset count]
当key对应的sorted set所有元素的score都相同时，获取元素值在min和max之间的元素组成的组。如果sorted set包含不同的score，返回值不确定。

可以作为二级索引应用。http://redis.io/topics/indexes
<pre class="brush: bash">10.44.90.20:6379>
10.44.90.20:6379>
10.44.90.20:6379> zadd shist 0 bia
(integer) 1
10.44.90.20:6379> zadd shist 0 ba
(integer) 1
10.44.90.20:6379> zadd shist 0 hello
(integer) 1
10.44.90.20:6379> zadd shist 0 jack
(integer) 1
10.44.90.20:6379> zrange shist 0 -1
1) "ba"
2) "bia"
3) "bit"
4) "hello"
5) "jack"
10.44.90.20:6379>
10.44.90.20:6379> ZRANGEBYLEX shist "[bit" "[bit\xff"
1) "bit"
10.44.90.20:6379> zadd shist 0 bita
(integer) 1
10.44.90.20:6379> zadd shist 0 bitb
(integer) 1
10.44.90.20:6379> zadd shist 0 bitbafda
(integer) 1
10.44.90.20:6379>
10.44.90.20:6379> ZRANGEBYLEX shist "[bit" "[bit\xff"
1) "bit"
2) "bita"
3) "bitb"
4) "bitbafda"</pre>
这是官网关于二级索引（Secondary Indexing）的一个例子，可以用来做“搜索提示”。“shist”这个key存储用户的搜索历史，每搜索一个关键字都会“zadd”把关键字作为成员添加进zset，后续根据前缀搜索条目。需要注意，zadd时，成员的分数必须都致，比如都是“0”，才能使用ZRANGEBYLEX。

ZREVRANGEBYLEX key min max [LIMIT offset count]

ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT offset count]
分数从低到高，获取分数在[min, max]区间的元素。LIMIT类似于MySQL “SELECT LIMIT offset, count”，返回需要的数据之前，需要从offset之前的元素“穿过”，所会有些时间消耗。WITHSCORES将在结果中，将具体分数也一同返回。
对于min和max参数，可以使用-inf和+inf代表最小和最大值。默认情况下，min和max是被包在结果中的，如果在min或者max前加上“(”左括号，min或max将被排除，不会出现结果中。

ZREVRANGEBYSCORE key min max [WITHSCORES] [LIMIT offset count

ZINTERSTORE dest numkeys key [key ...] [WEIGHTS weight [weight ...]] [AGGREATE SUM|MIN|MAX]
取多个sorted set元素的交集，组成新的sorted set，结果储存在dest指定的sorted set中，如果est存在将被覆盖。key参数指明输入的sorted set，只有在所有sorted set都存在的元素</srong>，才会出现在dest中。dest中元素的score，默认情况是所有sorted set中元素score的和。然，WEIGHTS和AGGREGATE选项，将影响dest中元素的score，参考“ZUNIONSTORE&rdquo命令。

ZUNIONSTORE dest numkeys key [key ...] [WEIGHTS weight [weight ...]] [GGREGATE SUM|MIN|MAX]
取多个sorted set元素的并集，组成新的sorted set，结果储存在dest指定的sorted set中，如果est存在将被覆盖。key参数指明输入的sorted set，所有元素都会出现在dest。dest中元素的score，默认情况是所有sorted set中元素score的和。

WEIGHTS：在对score进行聚合（AGGREGATE选项指明聚合方式）之前，core将会被乘以“weight”倍。
AGGREGATE：score的聚合方式，默认是SUM求和，MIN求最小值，MA求最大值。

<h1>Redis管理</h1>
<h2>Client</h2>
客户端管理指令，可以进行查看客户端连接、关闭某客户端连接等操作。

CLIENT LIST

CLIENT KILL

CLIENT SETNAME connection-name

CLIENT GETNAME

CLIENT PAUSE(since 2.9.50)

CLIENT REPLY ON|OFF|SKIP(since 3.2)

<h2>config配置</h2>
在Redis运行时，修改配置，并可以重写到配置文件中。
CONFIG GET glob-pattern

CONFIG SET param value

CONFIG REWRITE

CONFIG RESETSTAT

SHUTDOWN [NOSAVE|SAVE]
关闭当前redis实例，具体行为：

关闭所有客户端
执行SAVE命令
Flush AOF文件，如果AOF开启
退出服务器

<h2>DB操作相关</h2>
Redis中也有DB的概念，不同的DB之间存储的key彼此独立，相互不影响。每个db由数字标识
连接redis时，会默认连接到DB 0

<h4>SELECT index</h4>
选择数据库，索引从0开始，新的连接总是默认连接到数据库0上。

DBSIZE
DB中key的数量

FLUSHDB
删除当前DB的所有key（危险）

FLUSHALL
删除redis实例中的所有key（十分危险）

<h2>持久化相关</h2>
官方文档：http://redis.
o/topics/persistence
Redis作者博客，讲解深层实现原理：<a href="http://oldblog.antirez.com/post/redis-persis
ence-demystified.html">http://oldblog.antirez.com/post/redis-persiste
ce-demystified.html

BGREWRITEAOF</
>
手动触发，开启AOF日
重写进程（redis 2.4后，重写进程由redis自动触发）。意义在于，优化AOF日志
生成更小更紧凑的RDB文件。

BGSAVE
手动触发，开启一个子进程，在后台异步地将Redis实例中的数据持久化
磁盘中。

SAVE
手动触发，阻塞所有客户端请求，将Redis实例中数据持久化到磁盘中。很少使用在生产环境，
非遇到Redis不能在后台执行持久化操作，可以作为最后的补救手段。

LASTSAVE

<h2>数据迁移</h2>
DUMP

RESTORE key tl serialized-value [REPLACE]

MOVE key db

MIGRATE host port key "" dest-db timeout [COPY] [REPLACE] [KEYS key [key ...]

以 原子方式将指定的key从一个Redis实例迁移到目标Redis实例，如果成功，保证key将从原始R
dis实例删除，并在目标Redis实例中存 在。COPY和REPLACE3.0及以上可用，指定COPY选项时，ke
不会在原始Redis实例删除。指定REPLACE时，如果key在目标 Redis实例存在则替换。KEYS选项在3.
.6及可用，将“key”指定为空字符串，同时使用KEYS，可以一次指定多个key。

具体迁移规则及原子性，参考：http://redis.io
commands/migrate

<h2>监控</h2>
INFO section
查看当前redis实例状态，支持多个section。具体每个section字段含义，参考：<a href="http:/
redis.io/command/info">http://redis.io/command/info

<h1>MISC</h1>
HELP
HELP可以说是学习和使用Redis第一个要学的命令，具体格式：
<pre class="brush: bash">10.44.90.20:6380> help
redis-cli 2.8.17
Type: "help @<group>" to get a list of commands in <group>
     "help <command>" for help on <command>
     "help <tab>" to get a list of possible help topics
     "quit" to exit</pre>

可以列出命令的格式和帮助信息，是必不可少的参考资料。

<h2>COMMAND</h2>
工具指令，用于查看redis各个命令的信息，包括：支持的命令、命令的属性等

COMMAND
以数组格式输出Redis支持的所有命令的详细信息
<pre class="brush: bash">10.44.90.20:6380> command
  1) 1) "zrem"
     2) (integer) -3
     3) 1) write
        2) fast
     4) (integer) 1
     5) (integer) 1
     6) (integer) 1

```
  2) 1) "msetnx"
     2) (integer) -3
     3) 1) write
        2) denyoom
     4) (integer) 1
     5) (integer) -1
     6) (integer) 2
  ...
```

</pre>
具体含义参考：http://redis.io/commands command

COMMAND COUNT
Redis支持的命令个数，对于redis 2.8.17，输出是：157

COMMAND GETKEYS redis-command

COMMAND INFO command-name [command-name ...]

<h2>Cluster相关</h2>
TODO

<h1>通用</h1>

EXPIRE key seconds

EXPIREAT key timestamp
以unix timestamp方式（1970-1-1 00：00以来的秒数），指定过期时间

TTL key
key对应的value剩余的存活时间，秒数。

PTTL key
和TTL相同，不同的是，返回毫秒

PERSIST key
去除key对应的value的expire时间

OBJECT subcommand [arg [arg ...]]
查看key对应的value在Redis内部的对象表示方式，Redis是否为节省空间而对value类型做了优等。

subcommand

REFCOUNT key   与key对应的value的引用数
ENCODING key   与key对应的value，在Redis内部的类型表示</l
>
IDLETIME key   key的空闲时间，即：距上次被读写的空闲秒数

TYPE key

SCAN cursor [MATCH glob-pattern] [COUNT count]
SCAN基于游标方式，迭代当前redis实例，当前DB中的key。避免了当key特别多时，KEYS命令时间阻塞服务的缺陷。相关的命令SSCAN、HSCAN、ZSCAN同属SCAN系列。

官方文档：
http://redis.io/commands/scan

实现原理：
http://www.bubuko.com/infodetail-826677.html
http://antirez.com/news/63
https://github.com/antirez/redis/pull/579

