我们天天都在使用 Redis 内置的命令行工具 redis-cli，久而久之以为它就是一个简单的交互式 Redis 数据结构手工操作程序，但是它背后强大的功能绝大多数同学可能闻所未闻。本节我们一起来挖掘这些鲜为人知的有趣用法。

执行单条命令

平时在访问 Redis 服务器，一般都会使用 redis-cli 进入交互模式，然后一问一答来读写服务器，这种情况下我们使用的是它的「交互模式」。还有另外一种「直接模式」，通过将命令参数直接传递给 redis-cli 来执行指令并获取输出结果。

```
$ redis-cli incrby foo 5
(integer) 5
$ redis-cli incrby foo 5
(integer) 10
```

如果输出的内容较大，还可以将输出重定向到外部文件

```
$ redis-cli info > info.txt
$ wc -l info.txt
     120 info.txt
```

上面的命令指向的服务器是默认服务器地址，如果想指向特定的服务器可以这样

```
// -n 2 表示使用第2个库，相当于 select 2
$ redis-cli -h localhost -p 6379 -n 2 ping
PONG
```

批量执行命令

在平时线上的开发过程中，有时候我们免不了要手工造数据，然后导入 Redis。通常我们会编写脚本程序来做这件事。不过还有另外一种比较便捷的方式，那就是直接使用 redis-cli 来批量执行一系列指令。

```
$ cat cmds.txt
set foo1 bar1
set foo2 bar2
set foo3 bar3
.......
$ cat cmds.txt | redis-cli
OK
OK
OK
...
```

上面的指令使用了 Unix 管道将 cat 指令的标准输出连接到 redis-cli 的标准输入。其实还可以直接使用输入重定向来批量执行指令。

```
$ redis-cli < cmds.txt
OK
OK
OK
...
```

set 多行字符串

如果一个字符串有多行，你希望将它传入 set 指令，redis-cli 要如何做？可以使用 -x 选项，该选项会使用标准输入的内容作为最后一个参数。

```
$ cat str.txt
Ernest Hemingway once wrote,
"The world is a fine place and worth fighting
for."
I agree with the second part.
$ redis-cli -x set foo < str.txt
OK
$ redis-cli get foo
"Ernest Hemingway once wrote,\n\"The world is a
fine place and worth fighting for.\"\nI agree
with the second part.\n"
```

重复执行指令

redis-cli 还支持重复执行指令多次，每条指令执行之间设置一个间隔时间，如此便可以观察某条指令的输出内容随时间变化。

```
// 间隔1s，执行5次，观察qps的变化
$ redis-cli -r 5 -i 1 info | grep ops
instantaneous_ops_per_sec:43469
instantaneous_ops_per_sec:47460
instantaneous_ops_per_sec:47699
instantaneous_ops_per_sec:46434
instantaneous_ops_per_sec:47216
```

如果将次数设置为 -1 那就是重复无数次永远执行下去。如果不提供 -i 参数，那就没有间隔，连续重复执行。在交互模式下也可以重复执行指令，形式上比较怪异，在指令前面增加次数

```
127.0.0.1:6379> 5 ping
PONG
PONG
PONG
PONG
PONG
# 下面的指令很可怕，你的屏幕要愤怒了
127.0.0.1:6379> 10000 info
.......
```

导出 csv

redis-cli 不能一次导出整个库的内容为 csv，但是可以导出单条指令的输出为 csv 格式。

```
$ redis-cli rpush lfoo a b c d e f g
(integer) 7
$ redis-cli --csv lrange lfoo 0 -1
"a","b","c","d","e","f","g"
$ redis-cli hmset hfoo a 1 b 2 c 3 d 4
OK
$ redis-cli --csv hgetall hfoo
"a","1","b","2","c","3","d","4"
```

当然这种导出功能比较弱，仅仅是一堆字符串用逗号分割开来。不过你可以结合命令的批量执行来看看多个指令的导出效果。

```
$ redis-cli --csv -r 5 hgetall hfoo
"a","1","b","2","c","3","d","4"
"a","1","b","2","c","3","d","4"
"a","1","b","2","c","3","d","4"
"a","1","b","2","c","3","d","4"
"a","1","b","2","c","3","d","4"
```

看到这里读者应该明白 --csv 参数的效果就是对输出做了一次转换，用逗号分割，仅此而已。

执行 lua 脚本

在 lua 脚本小节，我们使用 eval 指令来执行脚本字符串，每次都是将脚本内容压缩成单行字符串再调用 eval 指令，这非常繁琐，而且可读性很差。redis-cli 考虑到了这点，它可以直接执行脚本文件。

```
127.0.0.1:6379> eval "return redis.pcall('mset',
KEYS[1], ARGV[1], KEYS[2], ARGV[2])" 2 foo1 foo2
bar1 bar2
OK
127.0.0.1:6379> eval "return redis.pcall('mget',
KEYS[1], KEYS[2])" 2 foo1 foo2
1) "bar1"
2) "bar2"
```

下面我们以脚本的形式来执行上面的指令，参数形式有所不同，KEY 和 ARGV 之间需要使用逗号分割，并且不需要提供 KEY 的数量参数

```
$ cat mset.txt
return redis.pcall('mset', KEYS[1], ARGV[1],
KEYS[2], ARGV[2])
$ cat mget.txt
return redis.pcall('mget', KEYS[1], KEYS[2])
$ redis-cli --eval mset.txt foo1 foo2 , bar1 bar2
OK
$ redis-cli --eval mget.txt foo1 foo2
1) "bar1"
2) "bar2"
```

如果你的 lua 脚本太长，--eval 将大有用处。

监控服务器状态

我们可以使用 --stat 参数来实时监控服务器的状态，间隔 1s 实时输出一次。

```
$ redis-cli --stat
------- data ------ -------------------- load --
------------------ - child -
keys       mem      clients blocked requests
connections
2          6.66M    100     0       11591628 (+0)
335
2          6.66M    100     0       11653169
(+61541)   335
2          6.66M    100     0       11706550
(+53381)   335
2          6.54M    100     0       11758831
(+52281)   335
2          6.66M    100     0       11803132
(+44301)   335
2          6.66M    100     0       11854183
(+51051)   335
```

如果你觉得间隔太长或是太短，可以使用 -i 参数调整输出间隔。

扫描大 KEY

这个功能太实用了，我已经在线上试过无数次了。每次遇到 Redis 偶然卡顿问题，第一个想到的就是实例中是否存在大 KEY，大 KEY 的内存扩容以及释放都会导致主线程卡顿。如果知道里面有没有大 KEY，可以自己写程序扫描，不过这太繁琐了。redis-cli 提供了 --bigkeys 参数可以很快扫出内存里的大 KEY，使用 -i 参数控制扫描间隔，避免扫描指令导致服务器的 ops 陡增报警。

```
$ ./redis-cli --bigkeys -i 0.01
# Scanning the entire keyspace to find biggest
keys as well as
# average sizes per key type.  You can use -i 0.1
to sleep 0.1 sec
# per 100 SCAN commands (not usually needed).

[00.00%] Biggest zset   found so far
'hist:aht:main:async_finish:20180425:17' with
1440 members
[00.00%] Biggest zset   found so far
'hist:qps:async:authorize:20170311:27' with 2465
members
[00.00%] Biggest hash   found so far
'job:counters:6ya9ypu6ckcl' with 3 fields
[00.01%] Biggest string found so far
'rt:aht:main:device_online:68:{-4}' with 4 bytes
[00.01%] Biggest zset   found so far
'machine:load:20180709' with 2879 members
[00.02%] Biggest string found so far
'6y6fze8kj7cy:{-7}' with 90 bytes
```

redis-cli 对于每一种对象类型都会记录长度最大的 KEY，对于每一种对象类型，刷新一次最高记录就会立即输出一次。它能保证输出长度为 Top1 的 KEY，但是 Top2、Top3等 KEY 是无法保证可以扫描出来的。一般的处理方法是多扫描几次，或者是消灭了 Top1 的 KEY 之后再扫描确认还有没有次大的 KEY。

采样服务器指令

现在线上有一台 Redis 服务器的 OPS 太高，有很多业务模块都在使用这个 Redis，如何才能判断出来是哪个业务导致了 OPS 异常的高。这时可以对线上服务器的指令进行采样，观察采样的指令大致就可以分析出 OPS 占比高的业务点。这时就要使用 monitor 指令，它会将服务器瞬间执行的指令全部显示出来。不过使用的时候要注意即使使用 ctrl+c 中断，否则你的显示器会噼里啪啦太多的指令瞬间让你眼花缭乱。

```
$ redis-cli --host 192.168.x.x --port 6379
monitor
1539853410.458483 [0 10.100.90.62:34365] "GET"
"6yax3eb6etq8:{-7}"
1539853410.459212 [0 10.100.90.61:56659] "PFADD"
"growth:dau:20181018" "2klxkimass8w"
1539853410.462938 [0 10.100.90.62:20681] "GET"
"6yax3eb6etq8:{-7}"
1539853410.467231 [0 10.100.90.61:40277] "PFADD"
"growth:dau:20181018" "2kei0to86ps1"
1539853410.470319 [0 10.100.90.62:34365] "GET"
"6yax3eb6etq8:{-7}"
1539853410.473927 [0 10.100.90.61:58128] "GET"
"6yax3eb6etq8:{-7}"
1539853410.475712 [0 10.100.90.61:40277] "PFADD"
"growth:dau:20181018" "2km8sqhlefpc"
1539853410.477053 [0 10.100.90.62:61292] "GET"
"6yax3eb6etq8:{-7}"
```

诊断服务器时延

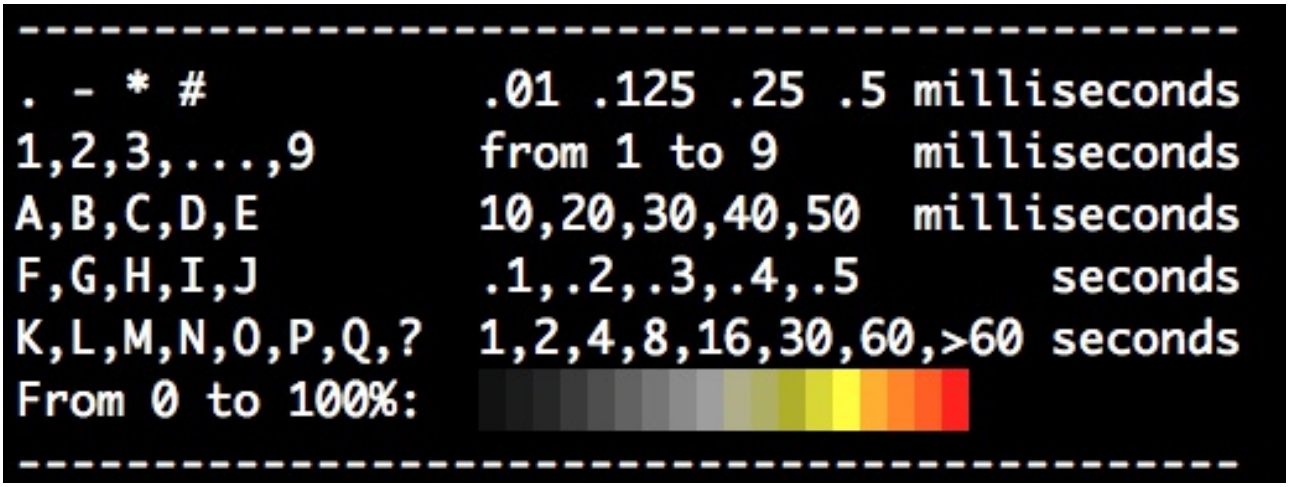
平时我们诊断两台机器的时延一般是使用 Unix 的 ping 指令。Redis 也提供了时延诊断指令，不过它的原理不太一样，它是诊断当前机器和 Redis 服务器之间的指令(PING指令)时延，它不仅仅是物

理网络的时延，还和当前的 Redis 主线程是否忙碌有关。如果你发现 Unix 的 ping 指令时延很小，而 Redis 的时延很大，那说明 Redis 服务器在执行指令时有微弱卡顿。

```
$ redis-cli --host 192.168.x.x --port 6379 --
latency
min: 0, max: 5, avg: 0.08 (305 samples)
```

时延单位是 ms。redis-cli 还能显示时延的分布情况，而且是图形化输出。

```
$ redis-cli --latency-dist
```



这个图形的含义作者没有描述，读者们可以尝试破解一下。

远程 rdb 备份

执行下面的命令就可以将远程的 Redis 实例备份到本地机器，远程服务器会执行一次bgsave操作，然后将 rdb 文件传输到客户端。远程 rdb 备份让我们有一种“秀才不出门，全知天下事”的感觉。

```
$ ./redis-cli --host 192.168.x.x --port 6379 --
rdb ./user.rdb
SYNC sent to master, writing 2501265095 bytes to
'./user.rdb'
Transfer finished with success.
```

模拟从库

如果你想观察主从服务器之间都同步了那些数据，可以使用 redis-cli 模拟从库。

```
$ ./redis-cli --host 192.168.x.x --port 6379 --
slave
SYNC with master, discarding 51778306 bytes of
bulk transfer...
SYNC done. Logging commands from master.
...
```

从库连上主库的第一件事是全量同步，所以看到上面的指令卡顿这很正常，待首次全量同步完成后，就会输出增量的 aof 日志。