**一、介绍**  
  
           今天继续redis-cli使用的介绍，上一篇文章写了一部分，写到第9个小节，今天就来完成第二部分。话不多说，开始我们今天的讲解。如果要想看第一篇文章，地址如下：[http://www.cnblogs.com/PatrickLiu/p/8508975.html](http://www.cnblogs.com/PatrickLiu/p/8508975.html" \t "http://www.cnblogs.com/PatrickLiu/p/_blank)  
  
**二、使用详解**  
  
          上篇文章写到第9个小节，今天直接按着以上的序号，继续来写  
  
          **10、特殊的操作模式**  
  
                    到目前为止，我们看到了redis-cli的两种主要模式。  
  
                         1、命令行执行Redis命令。  
  
                         2、交互式的“REPL-like”用法。  
  
        　　      然而，CLI执行与Redis相关的其他辅助任务，这些任务将在下一节中介绍：  
  
                         1、监控工具显示有关Redis服务器的连续统计信息。  
  
                         2、扫描Redis数据库查找非常大的key。  
  
                         3、与模式匹配的key空间扫描仪。  
  
                         4、作为Pub/Sub客户订阅频道。  
  
                         5、监视Redis实例中执行的命令。  
  
                         6、以不同方式检查Redis服务器的延迟。  
  
                         7、检查本地计算机的调度程序延迟。  
  
                         8、从远程Redis服务器传输RDB备份到本地。  
  
                         9、扮演Redis从节点的角色，展现从节点所接受的东西。  
  
                        10、模拟LRU工作负载以显示有关按键命中的统计信息。  
  
                        11、Lua调试器的客户端。  
  
  
                  **10.1、连续统计模式**  
  
                             这可能是redis-cli的最不常用的功能之一，并且对于实时监控Redis实例来说是非常有用。要启用此模式，使用--stat选项。 在这种模式下，CLI的行为非常清晰的：

[IMG_256](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

$ redis-cli -h 192.168.127.130 -p 6379 --stat

------- data ------ --------------------- load -------------------- - child -

keys mem clients blocked requests connections

506 1015.00K 1 0 24 (+0) 7

506 1015.00K 1 0 25 (+1) 7

506 3.40M 51 0 60461 (+60436) 57

506 3.40M 51 0 146425 (+85964) 107

507 3.40M 51 0 233844 (+87419) 157

507 3.40M 51 0 321715 (+87871) 207

508 3.40M 51 0 408642 (+86927) 257

508 3.40M 51 0 497038 (+88396) 257

[IMG_257](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                          在这种模式下，每秒都会打印一条新的数据行，其中包含有用信息和旧数据点之间的差异。 您可以轻松了解内存使用情况，客户端的链接等情况。  
  
                          在这种情况下，-i <interval>选项的作用就是修改输出新数据行的频率。 默认值是一秒。  
  
    **10.2、大键扫描**  
  
                            在这种特殊模式下，redis-cli可用作key空间容量大小的分析器。 它扫描占据比较大空间的key的数据集合，并能提供有关数据集组成的数据类型的信息。 该模式使用--bigkeys 选项启用，并生成十分详细的输出：

[IMG_258](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

[root@linux redis]# redis-cli -h 192.168.127.130 -p 6379 --bigkeys

# Scanning the entire keyspace to find biggest keys as well as average sizes per key type.

# 扫描整个键的空间以查找最大键以及每种键类型的平均大小。

# You can use -i 0.1 to sleep 0.1 sec per 100 SCAN commands (not usually needed).

# 您可以使用-i 0.1来每100次SCAN命令休息0.1秒（通常不需要）。

[00.00%] Biggest string found so far 'ss' with 1 bytes

[00.00%] Biggest string found so far 'foo1' with 25 bytes

-------- summary -------

Sampled 5 keys in the keyspace!

Total key length in bytes is 20 (avg len 4.00)

Biggest string found 'foo1' has 25 bytes

5 strings with 35 bytes (100.00% of keys, avg size 7.00)

0 lists with 0 items (00.00% of keys, avg size 0.00)

0 sets with 0 members (00.00% of keys, avg size 0.00)

0 hashs with 0 fields (00.00% of keys, avg size 0.00)

0 zsets with 0 members (00.00% of keys, avg size 0.00)

[IMG_259](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                            在输出的第一部分中，报告每个大于前一个较大键（相同类型）的新键。 摘要部分提供有关Redis实例内数据的一般统计信息。  
  
                            该程序使用 SCAN 命令，因此它可以在不影响客户端操作的情况下在繁忙的服务器上执行，不过也可以使用-i选项来限制所请求的每100个键的扫描过程的秒数。 例如，-i 0.1会减慢程序的执行速度，但也会大幅减轻服务器上的负载。  
  
                            请注意，摘要还会以更清晰的形式反映每次发现的最大键。 如果针对一个非常大的数据集运行，最初的输出只是提供一些有趣的信息ASAP。  
  
  
**10.3、获取键的列表**  
  
                              还可以扫描密钥空间，再次以不阻塞Redis服务器的方式（当您使用诸如 KEYS \* 之类的命令时会发生这种情况），并打印所有键的名称，或者使用特定模式进行过滤。 此模式与 --bigkeys 选项一样，使用SCAN命令，如果数据集正在发生更改，键就可能会多次反映更改，但如果从迭代开始以来就存在该键，那么该键也不会丢失。由于它使用这个选项的命令叫做--scan。

[IMG_260](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

[root@linux redis]# redis-cli -h 192.168.127.130 -p 6379 --scan | more -8

name

age

aaa

myset

myhash

address

myzset

rlist

[IMG_261](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                             请注意，使用 head -8 仅用于打印输出所有数据的前8行。  
  
                             scan命令可以配合 --pattern 选项使用模式匹配进行扫描

[root@linux redis]# redis-cli -h 192.168.127.130 -p 6379 --scan --pattern 'a\*'

age

aaa

address

                            根据键的名称，通过使用wc命令可以使管道输出针对特定种类对象的计数：

[IMG_262](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

[root@linux redis]# redis-cli -h 192.168.127.130 -p 6379 --scan --pattern 'a\*'

age

aaa

address

[root@linux redis]# redis-cli -h 192.168.127.130 -p 6379 --scan --pattern 'a\*' | wc -l

3

[IMG_263](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                               wc -l 这个选项的 -l,横杠后面是英文字母 L 的小写，不是数字 1。  
  
  
**10.4、发布/订阅模式**  
  
                              只需使用PUBLISH命令，CLI就能够在 Redis Pub/Sub通道中发布消息。这是预期的，因为PUBLISH命令与其他任何命令非常相似，使用简单。订阅频道为了接收消息使用了特殊的方法 - 在这种情况下，我们需要阻止和等待消息，此方法是作为redis-cli中的特殊模式实现的。 与其他特殊模式不同，此模式不是通过使用特殊选项启用的，而是通过使用SUBSCRIBE或PSUBSCRIBE命令启用的，无论是交互模式还是非交互模式：

[IMG_264](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

//非系统级通用通道

[root@linux redis]# redis-cli -h 192.168.127.130 -p 6379 psubscribe '\*'

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

1) "psubscribe"

2) "\*"

3) (integer) 1

//单一通道

[root@linux redis]# redis-cli -h 192.168.127.130 -p 6379 psubscribe mychannel

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

1) "psubscribe"

2) "\*"

3) (integer) 1

[IMG_265](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                             ’\*’ 带有单引号的星号表示非系统发布的消息通道，可以接受来自任何用户定义通道的信息，当然也可以输入具体名称的通道，比如：mychannel，我们针对具体名称的通道发布信息，必须制定通道名称，否则无效。  
  
                             \* 单独星号，没有单引号包含的，会显示系统当前所有发布的通道，如下：

[IMG_266](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

[root@linux redis]# redis-cli -h 192.168.127.130 -p 6379 psubscribe \*

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

1) "psubscribe"

2) "datas"

3) (integer) 1

1) "psubscribe"

2) "logs"

3) (integer) 2

1) "psubscribe"

2) "redis-benchmark"

3) (integer) 3

1) "psubscribe"

2) "redis-cli"

3) (integer) 4

1) "psubscribe"

2) "redis.conf"

3) (integer) 5

1) "psubscribe"

2) "redis-sentinel"

3) (integer) 6

1) "psubscribe"

2) "redis-server"

3) (integer) 7

1) "psubscribe"

2) "redis-trib.rb"

3) (integer) 8

1) "psubscribe"

2) "sentinel.conf"

3) (integer) 9

[IMG_267](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                                 阅读消息，消息显示我们输入了 Pub/Sub 模式。 当其他客户端在某个频道发布某条消息时（例如，您可以使用redis-cli PUBLISH mychannel mymessage），Pub/Sub模式中的CLI将显示如下内容：

[IMG_268](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

[root@linux redis]# redis-cli -h 192.168.127.130 -p 6379 publish mychannel mymessage

(integer) 1

显示内容：

1) "pmessage"

2) "\*"

3) "mychannel"

4) "mymessage"

[IMG_269](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                                  这对调试 发布/订阅 的问题非常有用。要退出发布/订阅模式只需处理CTRL-C。  
  
  
**10.5、监视在Redis中执行的命令**  
  
                                与 Pub/Sub 模式类似，使用MONITOR模式后，将自动输入监控模式。它将打印Redis实例收到的所有命令：

[root@linux redis]# redis-cli -h 192.168.127.130 -p 6379 monitor

OK

1520321617.017015 [0 192.168.127.130:34984] "publish" "mych" "mymessage"

1520321654.339150 [0 192.168.127.130:34986] "set" "sex" "1"

                                请注意，可以使用管道输出，因此您可以使用诸如grep等工具监视特定模式。  
  
  
                 **10.6、 监视Redis实例的延迟**  
  
                             Redis经常用于延迟非常严重的环境中。延迟涉及应用程序中的多个动态的部分，从客户端库到网络堆栈，再到Redis实例本身。  
  
                             CLI有多种功能用于研究Redis实例的延迟并了解延迟的最大值，平均值和分布。  
  
                             基本的延迟检查工具是 --latency 选项。 使用此选项，CLI运行一个循环，将PING命令发送到Redis实例，并测量获得答复的时间。这种情况每秒发生100次，统计信息在控制台中实时更新：

[root@linux redis]# redis-cli -h 192.168.127.130 -p 6379 --latency

min: 0, max: 3, avg: 0.28 (1051 samples)

                              统计数据以毫秒计数。通常情况下，由于系统内核调度程序运行redis-cli本身所导致的延迟，所以一个非常快的实例的平均延迟往往被高估了一点，所以0.19以上的平均延迟可能是0.01或更少。然而，这通常不是一个大问题，因为我们对几毫秒或更长时间的事件才感兴趣。  
  
                              有时候，研究平均延迟期的最大值和平均值如何随时间发展是有用的。--latency-history选项用于此目的：它的工作方式与--latency完全相同，但每15秒（默认情况下）一个全新的采样会话从头开始：

[IMG_270](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

[root@linux ~]# redis-cli -h 192.168.127.130 -p 6379 --latency-history

min: 0, max: 6, avg: 0.35 (1230 samples) -- 15.00 seconds range

min: 0, max: 3, avg: 0.34 (1277 samples) -- 15.01 seconds range

min: 0, max: 6, avg: 0.30 (1272 samples) -- 15.00 seconds range

min: 0, max: 2, avg: 0.33 (1289 samples) -- 15.00 seconds range

min: 0, max: 4, avg: 0.36 (1312 samples) -- 15.01 seconds range

min: 0, max: 1, avg: 0.24 (67 samples)^C

[IMG_271](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                             您可以使用-i <interval>选项更改采样会话的时间间隔步长。  
  
                             最先进的延迟研究工具，对于没有经验的用户来说也有点难解释明白，因此使用彩色终端显示一系列延迟是一种能力。您将看到一个彩色输出，指示不同样本的百分比，以及不同的ASCII字符表示不同的延迟数字。 使用 --latency-dist 选项启用此模式：

[IMG_272](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

[root@linux ~]# redis-cli -h 192.168.127.130 -p 6379 --latency-dist

---------------------------------------------

. - \* # .01 .125 .25 .5 milliseconds

1,2,3,...,9 from 1 to 9 milliseconds

A,B,C,D,E 10,20,30,40,50 milliseconds

F,G,H,I,J .1,.2,.3,.4,.5 seconds

K,L,M,N,O,P,Q,? 1,2,4,8,16,30,60,>60 seconds

From 0 to 100%:

---------------------------------------------

.-\*#123456789ABCDEFGHIJKLMNOPQ?

.-\*#123456789ABCDEFGHIJKLMNOPQ?

.-\*#123456789ABCDEFGHIJKLMNOPQ?

.-\*#123456789ABCDEFGHIJKLMNOPQ?

.-\*#123456789ABCDEFGHIJKLMNOPQ?

.-\*#123456789ABCDEFGHIJKLMNOPQ?

.-\*#123456789ABCDEFGHIJKLMNOPQ?

.-\*#123456789ABCDEFGHIJKLMNOPQ?

.-\*#123456789ABCDEFGHIJKLMNOPQ?

.-\*#123456789ABCDEFGHIJKLMNOPQ?

[IMG_273](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                              在redis-cli中还有另一个非常不寻常的延迟工具。它不会检查Redis实例的延迟，而是检查运行redis-cli的计算机的延迟。你可能会问什么延迟？ 内核调度程序固有的延迟，管理虚拟化实例的程序的延迟等等。  
  
                              我们称之为内部延迟，因为它对大多数程序员来说是不透明的。 如果您的Redis实例延迟不佳，任何微不足道的事情都有可能是造成延迟的罪魁祸首，那么通过在运行Redis服务器的系统中直接在此特殊模式下运行redis-cli，可以检查系统的最佳性能。  
  
                              通过测量内部延迟，您知道这是基准，Redis无法超越您的系统。为了在此模式下运行CLI，请使用--intrinsic-latency <test-time>。 测试的时间以秒为单位，并指定redis-cli多少秒可以检查一次当前正在运行的系统的延迟。

[IMG_274](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

[root@linux ~]# redis-cli -h 192.168.127.130 -p 6379 --intrinsic-latency 5

Max latency so far: 1 microseconds.

Max latency so far: 88 microseconds.

Max latency so far: 120 microseconds.

Max latency so far: 950 microseconds.

Max latency so far: 1192 microseconds.

Max latency so far: 1830 microseconds.

Max latency so far: 2107 microseconds.

32993317 total runs (avg latency: 0.1515 microseconds / 151.55 nanoseconds per run).

Worst run took 13903x longer than the average latency.

[IMG_275](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                             重要提示：必须在要运行Redis服务器的计算机上执行此命令，而不是在不同的主机上执行此命令。 它甚至不连接到Redis实例，只在本地执行测试。  
  
                             在上述情况下，我的系统不可能比最糟延迟2107微秒的情况更好，所以我可以期望某些查询在不到1毫秒的时间内运行。  
  
  
                  **10.7、远程备份RDB文件**  
  
                              在Redis复制的第一次同步期间，主设备和从设备以RDB文件的形式交换整个数据集。redis-cli利用此功能来提供远程备份功能，该功能允许将RDB文件从任何Redis实例传输到运行redis-cli的本地计算机。要使用此模式，请使用--rdb <dest-filename>选项调用CLI：

[root@linux ~]# redis-cli -h 192.168.127.130 -p 6379 --rdb /tmp/dump.rdb

SYNC sent to master, writing 534 bytes to '/tmp/dump.rdb'

Transfer finished with success.

                           这是确保您拥有Redis实例的灾难恢复RDB备份文件的简单而有效的方法。 但是，在脚本或cron作业中使用此选项时，请确保检查命令的返回值。如果它不为零，则发生错误，如下例所示：

[root@linux ~]# redis-cli -h 192.168.127.130 -p 6379 --rdb /tmp/dump.rdb

SYNC sent to master, writing 534 bytes to '/tmp/dump.rdb'

Transfer finished with success.

[root@linux ~]# echo $?

0

      **10.8、从模式**  
  
                           CLI的从属模式是一种高级功能，可用于Redis开发人员和调试操作。它允许检查主站发送到复制流中的从站以便将写入传播到其副本。选项名称简单--slave。示例代码如下：

[root@linux ~]# redis-cli -h 192.168.127.129 -p 6379 --slave

SYNC with master, discarding 535 bytes of bulk transfer...

SYNC done. Logging commands from master.

                           该命令首先丢弃第一个同步的RDB文件，然后以CSV格式记录每个收到的命令。  
  
                           如果您认为某些命令未在您的从站中正确复制，这也是检查发生了什么事情的好方法，对于改进错误报告也是有用的信息。  
  
  
       **10.9、执行LRU模拟**  
  
                           Redis通常用作LRU驱逐的缓存。根据键（key）的数量和为缓存分配的内存量（通过maxmemory指令指定），缓存命中和未命中的数量将会改变。有时，模拟命中率对正确配置缓存非常有用。  
  
                          CLI有一个特殊模式，它在请求模式中使用80-20％幂律分布来执行对GET和SET操作的模拟。这意味着20％的键将被80％的时间用来请求，这是缓存场景中的普遍存在的定律。  
  
                          从理论上来讲，基于给定的请求分布和Redis内存开销，可以用数学公式分析并计算命中率。 但是，Redis可以配置为不同的LRU设置（样本数量），并且LRU的实现（在Redis中近似）在不同版本之间也会有很大的变化。类似地，每个键的内存容量在各个版本之间也可能会有所不同。这就是为什么创建这个工具的原因：它的主要动机是测试Redis的LRU实现的质量，但现在也可用于测试给定版本的行为与您为部署考虑的设置的关系。  
  
                          为了使用此模式，您需要指定测试中的键的数量。您还需要为maxmemory设置一个有意义值的作为第一次尝试。  
  
                         重要注意事项：在Redis配置中配置maxmemory设置至关重要：如果没有最大内存使用量上限，则由于所有键均可存储在内存中，因此命中率最终将为100％。 或者，如果您指定的键太多而没有最大内存，则最终将使用所有计算机RAM。 还需要配置适当的maxmemory策略，大部分时间是allkeys-lru。  
  
  
                          在以下示例中，我配置了最大内存限制是100MB，并使用1000万个键对LRU进行了模拟。  
  
                         警告：测试使用流水线并会给服务器带来压力，请勿将其用于生产实例。

[IMG_276](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

[root@linux redis]# redis-cli -h 192.168.127.130 -p 6379 --lru-test 10000000

156000 Gets/sec | Hits: 4552 (2.92%) | Misses: 151448 (97.08%)

153750 Gets/sec | Hits: 12906 (8.39%) | Misses: 140844 (91.61%)

159250 Gets/sec | Hits: 21811 (13.70%) | Misses: 137439 (86.30%)

151000 Gets/sec | Hits: 27615 (18.29%) | Misses: 123385 (81.71%)

145000 Gets/sec | Hits: 32791 (22.61%) | Misses: 112209 (77.39%)

157750 Gets/sec | Hits: 42178 (26.74%) | Misses: 115572 (73.26%)

154500 Gets/sec | Hits: 47418 (30.69%) | Misses: 107082 (69.31%)

151250 Gets/sec | Hits: 51636 (34.14%) | Misses: 99614 (65.86%)

[IMG_277](http://www.cnblogs.com/PatrickLiu/p/javascript:void(0);)

                         该程序每秒显示统计信息。 如您所见，在第一秒钟内缓存开始被填充。 丢失率稍后稳定在我们可以预期的实际数字中：

120750 Gets/sec | Hits: 48774 (40.39%) | Misses: 71976 (59.61%)

122500 Gets/sec | Hits: 49052 (40.04%) | Misses: 73448 (59.96%)

127000 Gets/sec | Hits: 50870 (40.06%) | Misses: 76130 (59.94%)

124250 Gets/sec | Hits: 50147 (40.36%) | Misses: 74103 (59.64%)

                         对于我们的用例来说，59％的丢失率可能是不可接受的。所以我们知道100MB内存是不够的。让我们试试500MB字节。几分钟后，我们会看到输出稳定到以下数字：

140000 Gets/sec | Hits: 135376 (96.70%) | Misses: 4624 (3.30%)

141250 Gets/sec | Hits: 136523 (96.65%) | Misses: 4727 (3.35%)

140250 Gets/sec | Hits: 135457 (96.58%) | Misses: 4793 (3.42%)

140500 Gets/sec | Hits: 135947 (96.76%) | Misses: 4553 (3.24%)

                        因此我们知道在500MB的情况下，我们的键数量支持足够多（1000万）和分布也很合理（80-20方式）。  
  
  
**三、总结**  
  
          好了，今天就写到这里了，终于把redis-cli的使用细节写完了，翻译起来也挺耗时间的，有的时候可能翻译的不准确，也请大家指出。继续努力，不能松懈。如果想看原文，地址如下：https://redis.io/topics/rediscli。