**一、介绍**  
  
          有时候，Redis实例需要在很短的时间内加载大量先前存在或用户生成的数据，以便尽可能快地创建数百万个键。这就是所谓的批量插入，本文档的目标是提供有关如何以尽可能快的速度向Redis提供数据的信息。如果想查看英文原文，地址如下：[https://redis.io/topics/mass-insert](https://redis.io/topics/mass-insert" \t "http://www.cnblogs.com/PatrickLiu/p/_blank)  
  
  
**二、操作详解**

           话不多说，直接进入主题了。  
  
**1、使用协议，卢克（Use the protocol, Luke）**  
  
                 使用普通Redis客户端的方式执行批量插入的操作并不是一个很好的办法，原因如下：发送一个命令的方式很慢，因为您必须为每个命令都会有往返的时间消耗。虽然可以使用管道模式来操作，但为了批量插入多条记录，您需要在读取回复的同时编写新命令，以确保尽可能快地插入。  
  
                 另外，只有一小部分客户端支持非阻塞 I/O 操作，而且并不是所有的客户端都能够以最大化吞吐量这种有效的方式来解析这些回复。 由于以上这些原因，将大量数据导入Redis的首选方式是生成包含Redis协议的文本文件（原始格式），以便调用插入所需数据所需的命令。  
  
                例如，如果我需要生成一个大型数据集，其中包含数十亿个键：“keyN - > ValueN”，我将创建一个包含如下Redis协议格式的命令的文件：

SET Key0 Value0

SET Key1 Value1

...

SET KeyN ValueN

                 一旦创建了该文件，剩下的操作就是尽可能快地将其提供给Redis。在过去，做法是使用如下的netcat的命令：

(cat data.txt; sleep 10) | nc localhost 6379 > /dev/null

                然而，这并不是一个非常可靠的方式来执行批量导入，因为 netcat 命令并不会真正知道所有数据何时传输完毕，并且也无法检查发生的错误。在Redis的2.6或更高版本中，redis-cli实用程序支持称为管道的新模式，该模式就是为了执行批量插入而存在的。  
  
                使用管道模式，运行的命令如下所示：

cat data.txt | redis-cli --pipe

                这将产生类似于这样的输出：

All data transferred. Waiting for the last reply...

Last reply received from server.

errors: 0, replies: 1000000

               redis-cli实用程序还将确保只将从Redis实例收到的错误重定向到标准输出。  
  
  
**2、生成Redis协议（Generating Redis Protocol）**  
  
                  Redis协议生成和解析非常简单，如果想了解协议的详情，英文原地址点击《[这里](https://redis.io/topics/protocol" \t "http://www.cnblogs.com/PatrickLiu/p/_blank)》，我翻译的文章的地址点击《[Redis进阶实践之十七 Redis协议的规范](https://www.cnblogs.com/PatrickLiu/p/8567453.html" \t "http://www.cnblogs.com/PatrickLiu/p/_blank)》。然而，为了生成用于大容量插入协议的目标，您不需要了解协议的每个细节，只需要按照以下方式书写每个命令：

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

\*<args><cr><lf>

$<len><cr><lf>

<arg0><cr><lf>

<arg1><cr><lf>

...

<argN><cr><lf>

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

                其中<cr>表示“\r”（或ASCII字符13），<lf>表示“\n”（或ASCII字符10）。  
  
                例如，命令 SET key value 由以下协议表示：

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

\*3<cr><lf>

$3<cr><lf>

SET<cr><lf>

$3<cr><lf>

key<cr><lf>

$5<cr><lf>

value<cr><lf>

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

              或者表示为引用的字符串：

"\*3\r\n$3\r\nSET\r\n$3\r\nkey\r\n$5\r\nvalue\r\n"

             为批量插入而生成的文件只不过是由以上述方式表示的一个接一个的命令组成的。  
  
             以下Ruby函数生成有效的协议：

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

def gen\_redis\_proto(\*cmd)

proto = ""

proto << "\*"+cmd.length.to\_s+"\r\n"

cmd.each{|arg|

proto << "$"+arg.to\_s.bytesize.to\_s+"\r\n"

proto << arg.to\_s+"\r\n"

}

proto

end

puts gen\_redis\_proto("SET","mykey","Hello World!").inspect

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

             使用上述功能，可以使用此程序轻松生成上例中的键值对：

(0...1000).each{|n|

STDOUT.write(gen\_redis\_proto("SET","Key#{n}","Value#{n}"))

}

              我们可以在redis-cli的管道中直接运行程序，以执行我们的第一次海量导入会话。

$ ruby proto.rb | redis-cli -h 192.168.127.130 -p 6379 --pipe

All data transferred. Waiting for the last reply...

Last reply received from server.

errors: 0, replies: 1000

 **3、管道模式如何在引擎下工作（How the pipe mode works under the hoods）**  
  
                 redis-cli管道模式的速度和netcat一样快，与此同时，仍然能够明白服务器最后一次发送回复的时间。  
  
                 这是通过以下方式获得的：  
  
       **3.1、redis-cli --pipe Redis客户端会尽可能快的向服务器发送数据。**  
  
**3.2、同时，会尽可能快的读取并解析数据文件中的内容。**  
  
**3.3、一旦从标准输入设备读取数据完毕，它将会发送一个带有20个字节的字符串的特殊的ECHO命令到服务器：我们确信这是最新发送的命令，如果我们收到作为批量回复的相同的20个字节的消息，我们确信可以做“答复匹配检查”。**  
  
**3.4、这个特殊的最终命令一经发送，Redis服务器端将接收到回复和这20个字节的回复消息做匹配。如果匹配，它可以成功退出，表示插入完毕。**  
  
  
               使用这个技巧，我们不需要解析我们发送给服务器的协议，以了解我们发送了多少条命令，仅仅是一个答复而已。  
  
               但是，在解析回复时，我们会对所有解析的回复进行计数，以便在最后我们能够告诉用户传输到服务器的命令的数量在这次批量插入的会话中。  
  
  
    **4、示例代码操作**  
  
  
         **4.1、准备数据文件，格式是文本文件，名称是：redis\_commands.txt。**  
  
                   我在Windows环境下生成了一个txt文件，一条数据一行，代码如下：

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

SET Key0 Value0

SET Key1 Value1

SET Key2 Value2

SET Key3 Value3

SET Key4 Value4

SET Key5 Value5

SET Key6 Value6

SET Key7 Value7

SET Key8 Value8

SET Key9 Value9

SET Key10 Value10

...

SET KeyN ValueN

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

                   我生成了500万的数据，因为这个文本文件我是在Windows环境下生成的，所以需要格式转换。  
  
  
**4.2、如果使用Windows环境下生成的文件，需要进行格式转换，如果是在Linux环境下生成的文件就不需要格式转换，如果文本文件比较大，执行转换时间会有几秒，等待即可。**  
  
                  执行格式转换

[root@linux ~]# unix2dos redis\_commands.txt

unix2dos:converting file redis\_commands.txt to DOS format ...

                  以上代码进行格式转换完毕  
  
                  需要说明一点，unix2dos这个命令需要先安装，如果没有安装，会提示：command not found。  
  
                  执行以下命令安装：

[root@linux ~]# yum install unix2dos

        **4.3、进行数据批量插入**

[root@linux ~]# cat redis\_commands.txt | redis-cli -h 192.168.127.130 -p 6379 [-a "password"] -n 0 --pipe

All data transferred.Waiting for the last reply...

Last reply received from server.

errors:0，replies：10000000

                  批量插入数据成功，一千万的数据大概要花费50几秒左右。  
  
  
**三、总结**  
  
           好的，今天就写到这里，大批量数据插入的就是这么容易。只要理解了，其实也不是很难，技术就是一层窗户纸，一捅就破，但是没人捅就比较麻烦。下一篇文章，我们将写一些关于redis协议格式的文章，如果要涉及大批量数据插入，就会涉及到redis规范协议的问题。