# 慢查询

## 什么是慢查询

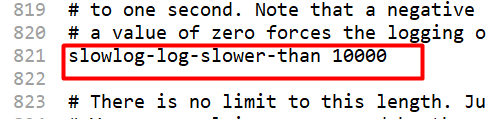
1. Redis慢查询日志功能用于记录执行时长超过给定时长的命令的相关信息。这些相关信息包括慢查询ID、执行时间戳、耗时、命令的类型、参数等。
2. 在实际应用中，开发人员和运维人员可以通过慢查询日志来定位系统的慢操作，然后利用这个功能产生的日志来监视和优化查询速度。
3. 慢查询日志也是存储在内存中的，也可以被持久化。
4. Redis执行一条命令需要经过发送命令、命令排队、执行命令、返回结果等几个步骤，而慢查询日志功能则发生在命令执行的过程中，如果该命令执行超时（超过slowlog-log-slower-than参数设置的时间），就会被记录到慢查询日志中。

## 配置慢查询

慢查询有两个重要的配置参数。

1. slowlog-log-slower-than

slowlog-log-slower-than参数用于指定执行时长超过多少微秒（1s=1000000us）的命令会被记录到慢查询日志中；当值为0时，则表示任何命令都会被记录到慢查询日志中；当值<0时，则表示关闭慢查询功能。例如，如果参数值为1000，那么执行时间超过1000us的命令就会被记录到慢查询日志中。

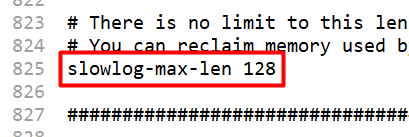


1. slowlog-max-len

slowlog-max-len参数用于指定服务器最多能够保存多少条慢查询日志；一条日志对应着一条命令的相关信息。

Redis服务器使用先进后出（栈）的方式保存多条慢查询日志。当服务器存储的慢查询日志数量等于slowlog-max-len参数的值时，服务器再添加一条新的慢查询日志之前，会先将最旧的那一条慢查询日志删除。

例如，slowlog-max-log参数的值为1000，假如服务器此时已经存储了1000条慢查询日志，如果服务器还将继续存储一条慢查询日志，那么它会先删除最旧的那条日志，再把新日志添加进去。



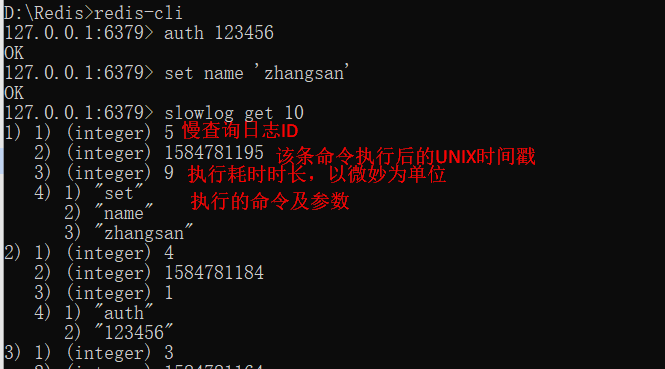
## 慢查询命令

**slowlog get**

slowlog get命令用于获取n条慢查询日志。命令如下：

**slowlog get [n]**

n表示慢查询日志的数量，如果没有设置，则获取全部的慢查询日志。



在每一条慢查询日志中：

第一个值是慢查询日志的唯一标识；

第二个值是该命令执行后的UNIX时间戳；

第三个值是执行该命令所耗费的时长，单位为微妙；

第四个值是一个列表，包括了执行的命令，及设置的参数。

**slowlog len**

slowlog len命令用于获取慢查询日志的数量。命令如下：

**slowlog len**

返回值：返回一个整数，表示慢查询日志的数量。

**slowlog reset**

slowlog reset命令用于清空慢查询日志。命令如下：

**slowlog reset**

执行该命令后，所有的慢查询日志将被删除。

返回值：总是返回OK。

## 慢查询的应用建议

对于慢查询，在实际应用中，应该注意以下几点：

1. slowlog-max-len参数的值不要设置得太小。在记录慢查询日志时，Redis会对长命令进行截断处理，并不会占用大量的内存。默认为128，但建议设置为1000。因为慢查询日志存储在内存中，如果该参数设置得过小，则会导致之前的慢查询日志丢失。
2. slowlog-log-slower-than参数的值不要设置得太大，默认设置为10000us，也就是10ms，表示命令执行时长超过10ms的买那个了就会被记录慢查询日志。在实际应用中也只有1ms或2ms，需要根据Redis的并发量来设置。Redis采用单线程响应命令，对于高并发、高流量的场景，如果命令执行的时长超过1ms，那么Redis最多可以支撑的OPS（每秒查询率）不到1000条。所以，对于高并发、高流量的场景，建议将该参数设置为1ms。
3. 定期对慢查询进行持久化，定期保存备份。

# 流水线（Pipeline）

## 为什么要使用流水线（管道）

Redis采用的是TCP协议来对外提供服务的，也就是说，Redis是基于Request/Response的，是一种一问一答的模式，请求一次响应一次。客户端通过Socket连接发起请求，发送一条命令给服务器，等待服务器应答，进行处理后，返回结果。在这个过程中，每个请求在命令发出后会阻塞等待Redis服务器进行处理，处理完毕后才会将结果返回给客户端；每条命令在发送与接收的过程中都会占用两个网络传输，在业务量非常庞大的情况下，假如处理一个业务需要0.1s，那么在1s内只能处理10个业务。在实际开发中是不能满足需求的，这将严重的影响Redis的性能，进而影响业务的开展。为了解决该问题，引入了流水线（Pipeline），也可以称之为管道。

## 什么是流水线（管道）

1. 所谓的流水线（Pipeline）技术，其实就是一次性把所有的命令发送给服务器，这样就可以避免频繁地发送、接收命令所带来的网络延迟，减少I/O调用次数；
2. 服务器在接收到一堆命令后，会依次执行，然后把结果打包，一次性返回给客户端；
3. 使用流水线（Pipeline）技术的好处是节约了网络带宽，缩短了访问时间，减少了服务器I/O调用次数，提升了Redis的性能。
4. 但是，要控制流水线（Pipeline）的大小，也就是它每次最多发送多少条命令的限制，过多的使用将会消耗Redis的内存，并且流水线（Pipeline）一次只能运行在一个Redis节点上。

## 如何使用流水线

我们可以通过一个小测试来体验流水线（Pipeline）所带来的好处。如：

不使用流水线（Pipeline）技术，执行10000条命令需要耗费的时间；

使用流水线（Pipeline）技术，执行10000条命令需要耗费的时间；

将两者所耗费的时间进行对比，可以得出一个结论：使用流水线（Pipeline）技术所耗费的时间比不使用流水线（Pipeline）技术所耗费的时间少。

# 地理位置的应用

## 介绍

1. 在Redis3.2版本以后引入了GEO（地理位置）功能，主要用于存储地理位置的经度、纬度信息；以及计算两个地理位置之间的距离，也可以用于计算某个地理范围内的位置信息。
2. GEO的用于广泛，最常见的就是类似于微信的“摇一摇”和“附近的人”功能，以及美团外卖、百度外卖等通过本地的地理位置坐标自动识别周围餐馆等。

## 存储地理位置

geoadd命令用于存储一个或多个地理位置信息。命令如下：

**geoadd location-set longitude latitude name**

**[location-set longitude latitude name……]**

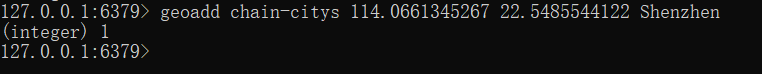
location-set为存储地理位置的集合；

longitude、latitude、name分别表示地理位置的经度、纬度、名称，可以有多对；

其中，经度范围为90°~90°，纬度范围为180°~180°；

返回值：返回成功添加的地理位置数量。

如下：



## 获取地理位置的经纬度信息

geopos命令用于获取一个或多个地理位置的经纬度信息。命令如下：

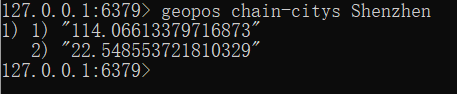
**getpos location-set name [name……]**

location-set表示存放地理位置信息的集合；

name表示地理位置名称，可以有多个；

返回值：以列表的形式返回经纬度信息；

如下：



## 计算两地之间的距离

geodist命令用于计算两个地理位置之间的距离。geodist命令如下：

**getodist location-set name1 name2 [unit]**

location-set表示存放地理位置信息的集合；

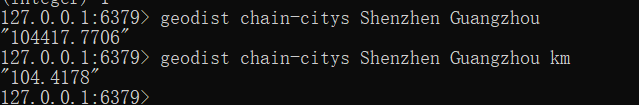
name1表示地理位置1；

name2表示地理位置2；

util用于指定计算距离的单位，可以是m（米），km（千米），mi（英里）以及ft（英尺），如不指定，则默认为m（米）；

返回值：返回一个数值，表示距离，单位为util指定的单位。

如下：



## 获取指定位置指定范围内的位置信息

除了计算两地之间的距离外，还可以以某个位置为中心点，查找在给定范围内的其他地理位置信息。为此，Redis提供了georadius和georadiusbymember命令。

1. georadius

georadius命令以给定的经纬度为中心点，查找指定范围内的其他地理位置。命令如下：

**georadius location-set longitude latitude radius m|km|ft|mi**

**[withcoord] [withdist] [withhash] [asc|desc] [count count]**

location-set表示存放地理位置的集合；

radius表示范围半径；

m|km|ft|mi：表示距离单位；

withcoord表示将地理位置的经纬度信息一起返回；

withdist表示在返回位置的同时，将中心点与位置元素之间的距离一同返回，单位保持一致；

withhash表示返回位置元素经过原始geohash编码的有序集合的分值，是一个52位有符号的整数，并不常有；

asc表示返回位置信息的排列顺序，由近到远；

desc表示返回位置信息的排列顺序，由远到近；

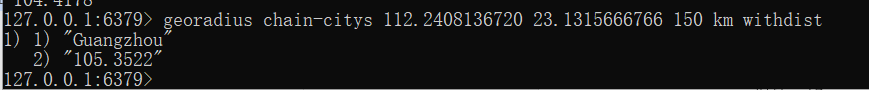
count表示返回位置信息的结果数量；

返回值：

返回在指定中心点和给定范围内的地理位置；

如果存储的地理位置集合中没有在以中心点为原点的指定范围内，就会返回空集合。

如下：



1. georadiusbymember

georadiusbymember命令以已保存的地理位置为中心点，查找指定范围内的其他地理位置。其他参数与georadius命令相同。如下：

