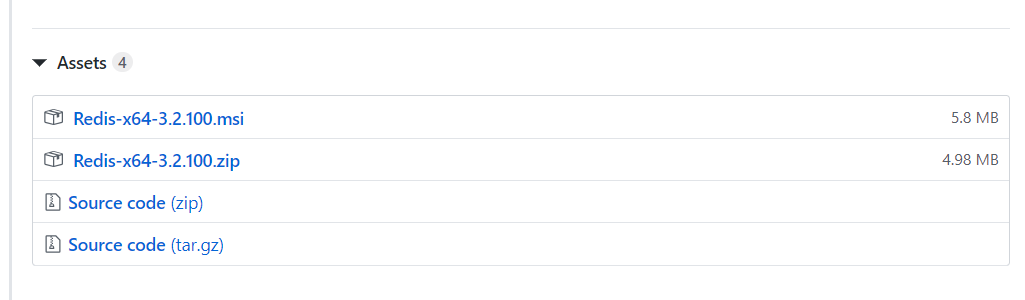
## Window 下安装

**下载地址：**[https://github.com/MSOpenTech/redis/releases](https://github.com/MSOpenTech/redi)。



执行一下命令启动redis服务，后面conf可以省略

redis-server.exe redis.windows.conf

这时候另启一个 cmd 窗口，原来的不要关闭，不然就无法访问服务端了。

切换到 redis 目录下运行:

redis-cli.exe -h 127.0.0.1 -p 6379

## Linux 下安装

**下载地址：**[http://redis.io/download](http://redis.io/downloadhttp://www.runoob.com/redis/_blank)，下载最新稳定版本。

本教程使用的最新文档版本为 2.8.17，下载并安装：

$ wget http://download.redis.io/releases/redis-2.8.17.tar.gz

$ tar xzf redis-2.8.17.tar.gz

$ cd redis-2.8.17

$ make

make完后 redis-2.8.17目录下会出现编译后的redis服务程序redis-server,还有用于测试的客户端程序redis-cli,两个程序位于安装目录 src 目录下：

下面启动redis服务.

$ cd src

$ ./redis-server

注意这种方式启动redis 使用的是默认配置。也可以通过启动参数告诉redis使用指定配置文件使用下面命令启动。

$ cd src

$ ./redis-server ../redis.conf

**redis.conf** 是一个默认的配置文件。我们可以根据需要使用自己的配置文件。

启动redis服务进程后，就可以使用测试客户端程序redis-cli和redis服务交互了。 比如：

$ cd src

$ ./redis-cli

redis> set foo bar

OK

redis> get foo"bar"

查看配置

redis 127.0.0.1:6379> CONFIG GET \*

修改配置

redis 127.0.0.1:6379> CONFIG SET loglevel "notice"

参数说明

redis.conf 配置项说明如下：

1. Redis默认不是以守护进程的方式运行，可以通过该配置项修改，使用yes启用守护进程

**daemonize no**

2. 当Redis以守护进程方式运行时，Redis默认会把pid写入/var/run/redis.pid文件，可以通过pidfile指定

**pidfile /var/run/redis.pid**

3. 指定Redis监听端口，默认端口为6379，作者在自己的一篇博文中解释了为什么选用6379作为默认端口，因为6379在手机按键上MERZ对应的号码，而MERZ取自意大利歌女Alessia Merz的名字

**port 6379**

4. 绑定的主机地址

**bind 127.0.0.1**

5.当 客户端闲置多长时间后关闭连接，如果指定为0，表示关闭该功能

**timeout 300**

6. 指定日志记录级别，Redis总共支持四个级别：debug、verbose、notice、warning，默认为verbose

**loglevel verbose**

7. 日志记录方式，默认为标准输出，如果配置Redis为守护进程方式运行，而这里又配置为日志记录方式为标准输出，则日志将会发送给/dev/null

**logfile stdout**

8. 设置数据库的数量，默认数据库为0，可以使用SELECT <dbid>命令在连接上指定数据库id

**databases 16**

9. 指定在多长时间内，有多少次更新操作，就将数据同步到数据文件，可以多个条件配合

**save <seconds> <changes>**

    Redis默认配置文件中提供了三个条件：

**save 900 1**

**save 300 10**

**save 60 10000**

    分别表示900秒（15分钟）内有1个更改，300秒（5分钟）内有10个更改以及60秒内有10000个更改。

10. 指定存储至本地数据库时是否压缩数据，默认为yes，Redis采用LZF压缩，如果为了节省CPU时间，可以关闭该选项，但会导致数据库文件变的巨大

**rdbcompression yes**

11. 指定本地数据库文件名，默认值为dump.rdb

**dbfilename dump.rdb**

12. 指定本地数据库存放目录

**dir ./**

13. 设置当本机为slav服务时，设置master服务的IP地址及端口，在Redis启动时，它会自动从master进行数据同步

**slaveof <masterip> <masterport>**

14. 当master服务设置了密码保护时，slav服务连接master的密码

**masterauth <master-password>**

15. 设置Redis连接密码，如果配置了连接密码，客户端在连接Redis时需要通过AUTH <password>命令提供密码，默认关闭

**requirepass foobared**

16. 设置同一时间最大客户端连接数，默认无限制，Redis可以同时打开的客户端连接数为Redis进程可以打开的最大文件描述符数，如果设置 maxclients 0，表示不作限制。当客户端连接数到达限制时，Redis会关闭新的连接并向客户端返回max number of clients reached错误信息

**maxclients 128**

17. 指定Redis最大内存限制，Redis在启动时会把数据加载到内存中，达到最大内存后，Redis会先尝试清除已到期或即将到期的Key，当此方法处理 后，仍然到达最大内存设置，将无法再进行写入操作，但仍然可以进行读取操作。Redis新的vm机制，会把Key存放内存，Value会存放在swap区

**maxmemory <bytes>**

18. 指定是否在每次更新操作后进行日志记录，Redis在默认情况下是异步的把数据写入磁盘，如果不开启，可能会在断电时导致一段时间内的数据丢失。因为 redis本身同步数据文件是按上面save条件来同步的，所以有的数据会在一段时间内只存在于内存中。默认为no

**appendonly no**

19. 指定更新日志文件名，默认为appendonly.aof

**appendfilename appendonly.aof**

20. 指定更新日志条件，共有3个可选值：   
    **no**：表示等操作系统进行数据缓存同步到磁盘（快）   
    **always**：表示每次更新操作后手动调用fsync()将数据写到磁盘（慢，安全）   
    **everysec**：表示每秒同步一次（折中，默认值）

**appendfsync everysec**

21. 指定是否启用虚拟内存机制，默认值为no，简单的介绍一下，VM机制将数据分页存放，由Redis将访问量较少的页即冷数据swap到磁盘上，访问多的页面由磁盘自动换出到内存中（在后面的文章我会仔细分析Redis的VM机制）

**vm-enabled no**

22. 虚拟内存文件路径，默认值为/tmp/redis.swap，不可多个Redis实例共享

**vm-swap-file /tmp/redis.swap**

23. 将所有大于vm-max-memory的数据存入虚拟内存,无论vm-max-memory设置多小,所有索引数据都是内存存储的(Redis的索引数据 就是keys),也就是说,当vm-max-memory设置为0的时候,其实是所有value都存在于磁盘。默认值为0

**vm-max-memory 0**

24. Redis swap文件分成了很多的page，一个对象可以保存在多个page上面，但一个page上不能被多个对象共享，vm-page-size是要根据存储的 数据大小来设定的，作者建议如果存储很多小对象，page大小最好设置为32或者64bytes；如果存储很大大对象，则可以使用更大的page，如果不 确定，就使用默认值

**vm-page-size 32**

25. 设置swap文件中的page数量，由于页表（一种表示页面空闲或使用的bitmap）是在放在内存中的，，在磁盘上每8个pages将消耗1byte的内存。

**vm-pages 134217728**

26. 设置访问swap文件的线程数,最好不要超过机器的核数,如果设置为0,那么所有对swap文件的操作都是串行的，可能会造成比较长时间的延迟。默认值为4

**vm-max-threads 4**

27. 设置在向客户端应答时，是否把较小的包合并为一个包发送，默认为开启

**glueoutputbuf yes**

28. 指定在超过一定的数量或者最大的元素超过某一临界值时，采用一种特殊的哈希算法

**hash-max-zipmap-entries 64**

**hash-max-zipmap-value 512**

29. 指定是否激活重置哈希，默认为开启（后面在介绍Redis的哈希算法时具体介绍）

**activerehashing yes**

30. 指定包含其它的配置文件，可以在同一主机上多个Redis实例之间使用同一份配置文件，而同时各个实例又拥有自己的特定配置文件

**include /path/to/local.conf**

Redis 简介

Redis 是完全开源免费的，遵守BSD协议，是一个高性能的key-value数据库。

Redis 与其他 key - value 缓存产品有以下三个特点：

Redis支持数据的持久化，可以将内存中的数据保存在磁盘中，重启的时候可以再次加载进行使用。

Redis不仅仅支持简单的key-value类型的数据，同时还提供list，set，zset，hash等数据结构的存储。

Redis支持数据的备份，即master-slave模式的数据备份。

Redis 优势

性能极高 – Redis能读的速度是110000次/s,写的速度是81000次/s 。

丰富的数据类型 – Redis支持二进制案例的 Strings, Lists, Hashes, Sets 及 Ordered Sets 数据类型操作。

原子 – Redis的所有操作都是原子性的，意思就是要么成功执行要么失败完全不执行。单个操作是原子性的。多个操作也支持事务，即原子性，通过MULTI和EXEC指令包起来。

丰富的特性 – Redis还支持 publish/subscribe, 通知, key 过期等等特性

Redis是一个字典结构的存储服务器，而实际上一个Redis实例提供了多个用来存储数据的字典，客户端可以指定将数据存储在哪个字典中。

每个数据库对外都是一个从0开始的递增数字命名，Redis默认支持16个数据库（可以通过配置文件支持更多，无上限），可以通过配置databases来修改这一数字。

179.redis 是什么？都有哪些使用场景？

Redis全称为：Remote Dictionary Server（远程数据服务），是一个基于内存的高性能key-value数据库。

1）、会话缓存（Session Cache）

最常用的一种使用Redis的情景是会话缓存（session cache）。用Redis缓存会话比其他存储（如Memcached）的优势在于：Redis提供持久化。当维护一个不是严格要求一致性的缓存时，如果用户的购物车信息全部丢失，大部分人都会不高兴的，现在，他们还会这样吗？ 幸运的是，随着 Redis 这些年的改进，很容易找到怎么恰当的使用Redis来缓存会话的文档。甚至广为人知的商业平台Magento也提供Redis的插件。

（2）、全页缓存（FPC）

除基本的会话token之外，Redis还提供很简便的FPC平台。回到一致性问题，即使重启了Redis实例，因为有磁盘的持久化，用户也不会看到页面加载速度的下降，这是一个极大改进，类似PHP本地FPC。 再次以Magento为例，Magento提供一个插件来使用Redis作为全页缓存后端。 此外，对WordPress的用户来说，Pantheon有一个非常好的插件 wp-redis，这个插件能帮助你以最快速度加载你曾浏览过的页面。

（3）、队列

Reids在内存存储引擎领域的一大优点是提供 list 和 set 操作，这使得Redis能作为一个很好的消息队列平台来使用。Redis作为队列使用的操作，就类似于本地程序语言（如Python）对 list 的 push/pop 操作。 如果你快速的在Google中搜索“Redis queues”，你马上就能找到大量的开源项目，这些项目的目的就是利用Redis创建非常好的后端工具，以满足各种队列需求。例如，Celery有一个后台就是使用Redis作为broker，你可以从这里去查看。

（4），排行榜/计数器

Redis在内存中对数字进行递增或递减的操作实现的非常好。集合（Set）和有序集合（Sorted Set）也使得我们在执行这些操作的时候变的非常简单，Redis只是正好提供了这两种数据结构。所以，我们要从排序集合中获取到排名最靠前的10个用户–我们称之为“user\_scores”，我们只需要像下面一样执行即可： 当然，这是假定你是根据你用户的分数做递增的排序。如果你想返回用户及用户的分数，你需要这样执行： ZRANGE user\_scores 0 10 WITHSCORES Agora Games就是一个很好的例子，用Ruby实现的，它的排行榜就是使用Redis来存储数据的，你可以在这里看到。

（5）、发布/订阅

最后（但肯定不是最不重要的）是Redis的发布/订阅功能。发布/订阅的使用场景确实非常多。我已看见人们在社交网络连接中使用，还可作为基于发布/订阅的脚本触发器，甚至用Redis的发布/订阅功能来建立聊天系统！

180.redis 有哪些功能？

181.redis 和 memecache 有什么区别？

(1)、存储方式 Memecache把数据全部存在内存之中，断电后会挂掉，数据不能超过内存大小。 Redis有部份存在硬盘上，这样能保证数据的持久性。

(2)、数据支持类型 Memcache对数据类型支持相对简单。 Redis有复杂的数据类型。

(3)、使用底层模型不同 它们之间底层实现方式 以及与客户端之间通信的应用协议不一样。 Redis直接自己构建了VM 机制 ，因为一般的系统调用系统函数的话，会浪费一定的时间去

182.redis 为什么是单线程的？

官方FAQ表示，因为Redis是基于内存的操作，CPU不是Redis的瓶颈，Redis的瓶颈最有可能是机器内存的大小或者网络带宽。既然单线程容易实现，而且CPU不会成为瓶颈，那就顺理成章地采用单线程的方案了（毕竟采用多线程会有很多麻烦！）。

183.什么是缓存穿透？怎么解决？

缓存穿透

一般的缓存系统，都是按照key去缓存查询，如果不存在对应的value，就应该去后端系统查找（比如DB）。一些恶意的请求会故意查询不存在的key,请求量很大，就会对后端系统造成很大的压力。这就叫做缓存穿透。

如何避免？

1：对查询结果为空的情况也进行缓存，缓存时间设置短一点，或者该key对应的数据insert了之后清理缓存。

2：对一定不存在的key进行过滤。可以把所有的可能存在的key放到一个大的Bitmap中，查询时通过该bitmap过滤。

缓存雪崩

当缓存服务器重启或者大量缓存集中在某一个时间段失效，这样在失效的时候，会给后端系统带来很大压力。导致系统崩溃。

如何避免？

1：在缓存失效后，通过加锁或者队列来控制读数据库写缓存的线程数量。比如对某个key只允许一个线程查询数据和写缓存，其他线程等待。

2：做二级缓存，A1为原始缓存，A2为拷贝缓存，A1失效时，可以访问A2，A1缓存失效时间设置为短期，A2设置为长期

3：不同的key，设置不同的过期时间，让缓存失效的时间点尽量均匀。

184.redis 支持的数据类型有哪些？



185.redis 支持的 java 客户端都有哪些？

Redisson、Jedis、lettuce等等，官方推荐使用Redisson。

186.jedis 和 redisson 有哪些区别？

Jedis 和 Redisson 都是Java中对Redis操作的封装。Jedis 只是简单的封装了 Redis 的API库，可以看作是Redis客户端，它的方法和Redis 的命令很类似。Redisson 不仅封装了 redis ，还封装了对更多数据结构的支持，以及锁等功能，相比于Jedis 更加大。但Jedis相比于Redisson 更原生一些，更灵活。

Redis官方对Java 语言的封装框架推荐的有十多种（Redis 官网），主要是Jedis 、Redisson。

Jedis使用阻塞的I/O，且其方法调用都是同步的，程序流需要等到sockets处理完I/O才能执行，不支持异步。Jedis客户端实例不是线程安全的，所以需要通过连接池来使用Jedis。

Redisson使用非阻塞的I/O和基于Netty框架的事件驱动的通信层，其方法调用是异步的。Redisson的API是线程安全的，所以可以操作单个Redisson连接来完成各种操作。

Jedis仅支持基本的数据类型如：String、Hash、List、Set、Sorted Set。

Redisson不仅提供了一系列的分布式Java常用对象，基本可以与Java的基本数据结构通用，还提供了许多分布式服务，其中包括（BitSet, Set, Multimap, SortedSet, Map, List, Queue, BlockingQueue, Deque, BlockingDeque, Semaphore, Lock, AtomicLong, CountDownLatch, Publish / Subscribe, Bloom filter, Remote service, Spring cache, Executor service, Live Object service, Scheduler service）。

187.怎么保证缓存和数据库数据的一致性？

1：读的顺序是先读缓存，后读数据库

2：写的顺序是先写数据库，然后写缓存

3：每次更新了相关的数据，都要把该缓存清理掉

4：为了避免极端条件下造成的缓存与数据库之间的数据不一致，缓存需要设置一个失效时间。时间到了，缓存自动被清理，达到缓存和数据库数据的“最终一致性”

188.redis 持久化有几种方式？

RDB做镜像全量持久化，AOF做增量持久化

RDB持久化也分两种：SAVE和BGSAVE。

•SAVE是阻塞式的RDB持久化，当执行这个命令时redis的主进程把内存里的数据库状态写入到RDB文件中，直到该文件创建完毕的这段时间内redis将不能处理任何命令请求；

•BGSAVE属于非阻塞式的持久化，它会创建一个子进程专门去把内存中的数据库状态写入RDB文件里，同时主进程还可以处理来自客户端的命令请求。<br><br>但子进程基本是复制的父进程，这等于两个相同大小的redis进程在系统上运行，会造成内存使用率的大幅增加。

AOF的持久化是通过命令追加、文件写入和文件同步三个步骤实现的

当reids开启AOF后（redis备份方式默认是RDB），

服务端每执行一次写操作（如set、sadd、rpush）就会把该条命令追加到一个单独的AOF缓冲区的末尾，这就是命令追加；然后把AOF缓冲区的内容写入AOF文件里。

189.redis 怎么实现分布式锁？

先拿setnx来争抢锁，抢到之后，再用expire给锁加一个过期时间防止锁忘记了释放。

190.redis 分布式锁有什么缺陷？

etnx之后执行expire之前进程意外crash造成死锁，其实Redis官方也考虑到了这个问题，在Redis2.8 之后，官方执行setnx 和 expire命令一起使用了。如下：

SET lock\_key lock\_value NX PX 30000

其中：

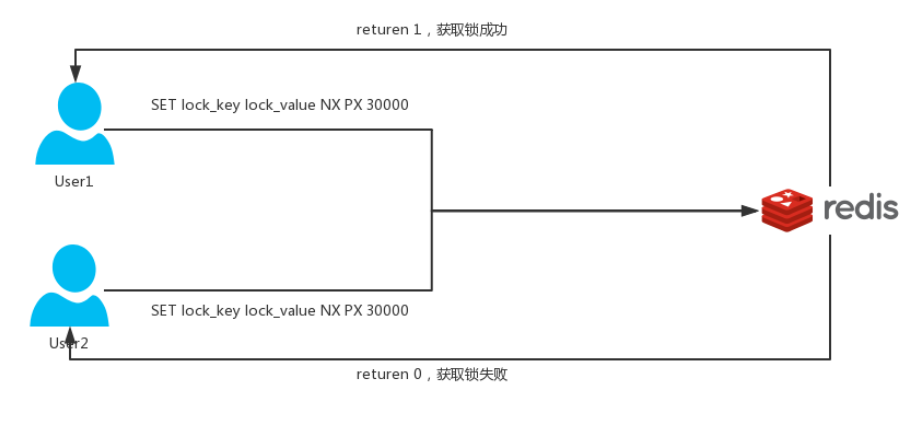
1.lock\_key：即锁名称，这个名称应是公开的，在分布式环境中，对于某一确定的公共资源，所有争用方（客户端）都应该知道对应锁的名字。对于 Redis 而言，lock\_name 就是 key-value 中的 key，具有唯一性。

lock\_value:是由客户端生成的一个随机字符串，它要保证在足够长的一段时间内在所有客户端的所有获取锁的请求中都是唯一的，用于唯一标识锁的持有者。

NX 表示只有当 lock\_key(key) 不存在的时候才能 SET 成功，从而保证只有一个客户端能获得锁，而其它客户端在锁被释放之前都无法获得锁。

PX 30000 表示这个锁节点有一个 30 秒的自动过期时间（目的是为了防止持有锁的客户端故障后，无法主动释放锁而导致死锁，因此要求锁的持有者必须在过期时间之内执行完相关操作并释放锁）。

具体操作如下图：



<https://www.cnblogs.com/justuntil/p/10458211.html>

191.redis 如何做内存优化？

尽可能使用散列表（hashes），散列表（是说散列表里面存储的数少）使用的内存非常小，所以你应该尽可能的将你的数据模型抽象到一个散列表里面。比如你的web系统中有一个用户对象，不要为这个用户的名称，姓氏，邮箱，密码设置单独的key,而是应该把这个用户的所有信息存储到一张散列表里面.

192.redis 淘汰策略有哪些？

我们set key的时候，都可以给一个expire time，就是过期时间，指定这个key比如说只能存活1个小时，我们自己可以指定缓存到期就失效。

　　如果假设你设置一个一批key只能存活1个小时，那么接下来1小时后，redis是怎么对这批key进行删除的？

　　答案是：定期删除+惰性删除

　　所谓定期删除，指的是redis默认是每隔100ms就随机抽取一些设置了过期时间的key，检查其是否过期，如果过期就删除。

　　注意，这里可不是每隔100ms就遍历所有的设置过期时间的key，那样就是一场性能上的灾难。

　　实际上redis是每隔100ms随机抽取一些key来检查和删除的。

　　但是，定期删除可能会导致很多过期key到了时间并没有被删除掉，所以就得靠惰性删除了。

　　这就是说，在你获取某个key的时候，redis会检查一下 ，这个key如果设置了过期时间那么是否过期了？如果过期了此时就会删除，不会给你返回任何东西。

　　并不是key到时间就被删除掉，而是你查询这个key的时候，redis再懒惰的检查一下

　　通过上述两种手段结合起来，保证过期的key一定会被干掉。

　　但是实际上这还是有问题的，如果定期删除漏掉了很多过期key，然后你也没及时去查，也就没走惰性删除，此时会怎么样？

　　如果大量过期key堆积在内存里，导致redis内存块耗尽了，怎么办？

　　答案是：走内存淘汰机制。

volatile-lru：从已设置过期时间的数据集（server.db[i].expires）中挑选最近最少使用的数据淘汰

volatile-ttl：从已设置过期时间的数据集（server.db[i].expires）中挑选将要过期的数据淘汰

volatile-random：从已设置过期时间的数据集（server.db[i].expires）中任意选择数据淘汰

allkeys-lru：从数据集（server.db[i].dict）中挑选最近最少使用的数据淘汰

allkeys-random：从数据集（server.db[i].dict）中任意选择数据淘汰

no-enviction（驱逐）：禁止驱逐数据

注意这里的6种机制，volatile和allkeys规定了是对已设置过期时间的数据集淘汰数据还是从全部数据集淘汰数据，后面的lru、ttl以及random是三种不同的淘汰策略，再加上一种no-enviction永不回收的策略。

使用策略规则：

1、如果数据呈现幂律分布，也就是一部分数据访问频率高，一部分数据访问频率低，则使用allkeys-lru

2、如果数据呈现平等分布，也就是所有的数据访问频率都相同，则使用allkeys-random

193.redis 常见的性能问题有哪些？该如何解决？

1.Master写内存快照，save命令调度rdbSave函数，会阻塞主线程的工作，当快照比较大时对性能影响是非常大的，会间断性暂停服务，所以Master最好不要写内存快照。

2.Master AOF持久化，如果不重写AOF文件，这个持久化方式对性能的影响是最小的，但是AOF文件会不断增大，AOF文件过大会影响Master重启的恢复速度。

3.Master调用BGREWRITEAOF重写AOF文件，AOF在重写的时候会占大量的CPU和内存资源，导致服务load过高，出现短暂服务暂停现象。

https://blog.csdn.net/wchengsheng/article/details/79925654