# redis是什么？

Redis是一个开源的使用ANSI [C语言](https://baike.baidu.com/item/C%E8%AF%AD%E8%A8%80)编写、支持网络、可基于内存亦可持久化的日志型、Key-Value[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)，并提供多种语言的API。从2010年3月15日起，Redis的开发工作由VMware主持。从2013年5月开始，Redis的开发由Pivotal赞助。

redis是一个key-value[存储系统](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E7%B3%BB%E7%BB%9F)。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、list([链表](https://baike.baidu.com/item/%E9%93%BE%E8%A1%A8))、set(集合)、zset(sorted set --有序集合)和hash（哈希类型）。这些[数据类型](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E7%B1%BB%E5%9E%8B)都支持push/pop、add/remove及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis支持各种不同方式的排序。与memcached一样，为了保证效率，数据都是缓存在内存中。区别的是redis会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了master-slave(主从)同步。

Redis 是一个高性能的key-value数据库。 redis的出现，很大程度补偿了[memcached](https://baike.baidu.com/item/memcached)这类key/value存储的不足，在部 分场合可以对关系数据库起到很好的补充作用。它提供了Java，C/C++，C#，PHP，JavaScript，Perl，Object-C，Python，Ruby，Erlang等客户端，使用很方便。[1]

Redis支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得Redis可执行单层树复制。存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助。

redis的官网地址，非常好记，是redis.io。（特意查了一下，域名后缀io属于国家域名，是british Indian Ocean territory，即英属印度洋领地）

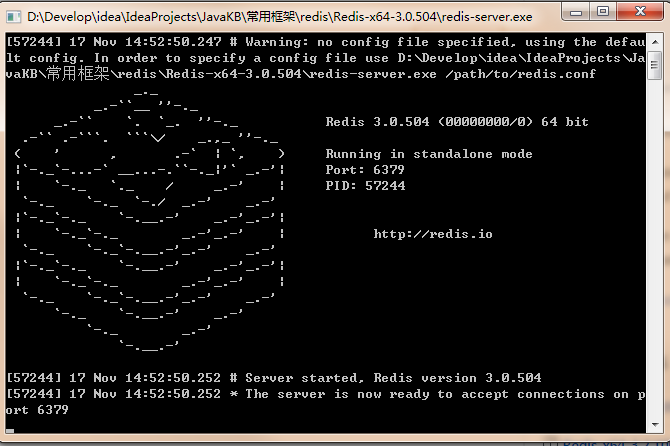
目前，Vmware在资助着redis项目的开发和维护。

# redis的安装

## window环境

Redis 没有官方的Windows版本，但是微软开源技术团队（Microsoft Open Tech group）开发和维护着这个 Win64 的版本。更多信息请参考[这里](https://github.com/MSOpenTech/redis)。

在<https://github.com/MicrosoftArchive/redis/releases> 下载最新的windwos版redis,解压运行redis-server.exe.



## linux环境

Redis是c语言开发的。

安装redis需要c语言的编译环境。如果没有gcc需要在线安装。Yum install gcc-c++

安装步骤：

第一步：redis的源码包上传到linux系统。

第二步：解压缩redis。

第三步：编译。make

第四步：安装。make install PREFIX=/usr/local/redis

## 2.3连接redis

### redis的启动：

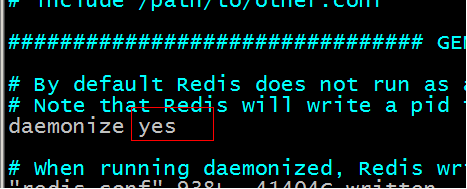
**前端启动**：[root@localhost bin]# ./redis-server

**后台启动**：

把/root/redis-3.0.0/redis.conf复制到/usr/local/redis/bin目录下

[root@localhost redis-3.0.0]# cp redis.conf /usr/local/redis/bin/

修改配置文件：[root@localhost bin]# ./redis-server redis.conf



**查看redis进程**：

[root@localhost bin]# ps aux|grep redis

root 5190 0.1 0.3 33936 1712 ? Ssl 18:23 0:00 ./redis-server \*:6379

root 5196 0.0 0.1 4356 728 pts/0 S+ 18:24 0:00 grep redis

### Redis-cli

[root@localhost bin]# ./redis-cli

默认连接localhost运行在6379端口的redis服务。

[root@localhost bin]# ./redis-cli -h 192.168.25.153 -p 6379

-h：连接的服务器的地址

-p：服务的端口号

# Redis的数据类型

String：key-value（做缓存）

Hash：key-fields-values（做缓存） 使用字段的形式来表示对象

List：有顺序可重复

Set：无顺序，不能重复

SortedSet（zset）：有顺序，不能重复

[官方文档--数据类型链接](http://www.redis.cn/topics/data-types.html)

## 3.1疑问

3.1.1 Hash是什么类型？

Redis Hashes是字符串字段和字符串值之间的映射，所以它们是完美的表示对象（eg:一个有名，姓，年龄等属性的用户）的数据类型。

|  |
| --- |
| HMSET user:1000 username antirez password P1pp0 age 34  HGETALL user:1000  HSET user:1000 password 12345  HGETALL user:1000 |

一个拥有少量（100个左右）字段的hash需要 很少的空间来存储，所有你可以在一个小型的 Redis实例中存储上百万的对象。

尽管Hashes主要用来表示对象，但它们也能够存储许多元素，所以你也可以用Hashes来完成许多其他的任务。

一个hash最多可以包含232-1 个key-value键值对（超过40亿）。

3.1.2 sorted set的原理

每个有序集合 的成员都关联着一个评分，这个评分用于把有序集 合中的成员按最低分到最高分排列。使用有序集合，你可以非常快地（**O(log(N))**）完成添加，删除和更新元素的操作。 因为元素是在插入时就排好序的，所以很快地通过评分(score)或者 位次(position)获得一个范围的元素。

# Redis内存优化方案

[官方文档-内存优化方案](http://www.redis.cn/topics/memory-optimization.html)

## 4.1 小的聚合类型数据的特殊编码处理

Redis2.2版本及以后，存储集合数据的时候会采用内存压缩技术，以使用更少的内存存储更多的数据。如Hashes,Lists,Sets和Sorted Sets，当这些集合中的所有数都小于一个给定的元素，并且集合中元素数量小于某个值时，存储的数据会被以一种非常节省内存的方式进行编码，使用这种编码理论上至少会节省10倍以上内存（平均节省5倍以上内存）。并且这种编码技术对用户和redis api透明。因为使用这种编码是用CPU换内存，所以我们提供了更改阈值的方法，只需在redis.conf里面进行修改即可.

|  |
| --- |
| hash-max-zipmap-entries 64 (2.6以上使用hash-max-ziplist-entries)  hash-max-zipmap-value 512 (2.6以上使用hash-max-ziplist-value)  list-max-ziplist-entries 512  list-max-ziplist-value 64  zset-max-ziplist-entries 128  zset-max-ziplist-value 64  set-max-intset-entries 512 |

（集合中）如果某个值超过了配置文件中设置的最大值，redis将自动把把它（集合）转换为正常的散列表。这种操作对于比较小的数值是非常快的，但是，如果你为了使用这种编码技术而把配置进行了更改，你最好做一下基准测试（和正常的不采用编码做一下对比）.

## 4.2 使用32位的redis

使用32位的redis，对于每一个key,将使用更少的内存，因为32位程序，指针占用的字节数更少。但是32的redis整个实例使用的内存将被限制在4G以下。使用make 32bit命令编译生成32位的redis。RDB和AOF文件是不区分32位和64位的（包括字节顺序）,所以你可以使用64位的reidis恢复32位的RDB备份文件，相反亦然.

## 4.3 尽可能使用散列表（hashes）

小散列表（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.

# Redis对事务的处理

[官方文档--事务](http://www.redis.cn/topics/transactions.html)