

一 Redis简介

1.0 Redis简介

什么是Redis

Redis 是完全开源免费的，遵守BSD协议，是一个高性能(NOSQL)的key-value数据库,Redis是一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库，并提供多种语言的API。

BSD是"Berkeley Software Distribution"的缩写，意思是"伯克利软件发行版"。BSD开源协议是一个给予使用者很大自由的协议。可以自由的使用，修改源代码，也可以将修改后的代码作为开源或者专有软件再发布。BSD代码鼓励代码共享，但需要尊重代码作者的著作权。

BSD由于允许使用者修改和重新发布代码，也允许使用或在BSD代码上开发商业软件发布和销售，因此是对商业集成很友好的协议。

1.1 NoSQL

NoSQL，泛指非关系型的数据库，NoSQL即Not-Only SQL，它可以作为关系型数据库的良好补充。随着互联网web2.0网站的兴起，非关系型的数据库现在成了一个极其热门的新领域，非关系数据库产品的发展非常迅速

互联网发展生命周期

Web.1.0

Web.2.0（可读写时代）--》数据量变的很大、高并发。

互联网阶段

云计算+大数据

人工智能（5G+芯片）

而传统的关系数据库在应付web2.0网站，特别是**超大规模和高并发的SNS类型的web2.0纯动态网站已经显得力不从心，暴露了很多难以克服的问题**，例如：

1、High performance - 对数据库高并发读写的需求

web2.0网站要根据用户个性化信息来实时生成动态页面和提供动态信息，所以基本上无法使用动态页面静态化技术，因此数据库并发负载非常高，往往要达到每秒上万次读写请求。关系数据库应付上万次SQL查询还勉强顶得住，但是应付上万次SQL写数据请求，硬盘IO就已经无法承受了。其实对于普通的BBS网站，往往也存在对高并发写请求的需求，例如网站的实时统计在线用户状态，记录热门帖子的点击次数，投票计数等，因此这是一个相当普遍的需求。

2、Huge Storage - 对海量数据的高效率存储和访问的需求

类似Facebook, twitter, Friendfeed这样的SNS网站，每天用户产生海量的用户动态，以Friendfeed为例，一个月就达到了2.5亿条用户动态，对于关系数据库来说，在一张2.5亿条记录的表里面进行SQL查询，效率是极其低下乃至不可忍受的。再例如大型web网站的用户登录系统，例如腾讯，盛大，动辄数以亿计的帐号，关系数据库也很难应付。

3、High Scalability & High Availability- 对数据库的高可扩展性和高可用性的需求

在基于web的架构当中，数据库是最难进行横向扩展的，当一个应用系统的用户量和访问量与日俱增的时候，你的数据库却没有办法像web server和app server那样简单的通过添加更多的硬件和服务节点来扩展性能和负载能力。对于很多需要提供24小时不间断服务的网站来说，对数据库系统进行升级和扩展是非常痛苦的事情，往往需要停机维护和数据迁移，为什么数据库不能通过不断的添加服务器节点来实现扩展呢？

NoSQL数据库的产生就是为了解决大规模数据集合多重数据种类带来的挑战，尤其是大数据应用难题

1.2 NoSQL的类别

键值(Key-Value)存储数据库

这一类数据库主要会使用到一个哈希表，这个表中有一个特定的键和一个指针指向特定的数据。

Key/value模型对于IT系统来说的优势在于简单、易部署。但是如果DBA只对部分值进行查询或更新的时候，Key/value就显得效率低下了。

相关产品：Tokyo Cabinet/Tyrant、Redis、Voldemort、Berkeley DB

典型应用：内容缓存，主要用于处理大量数据的高访问负载。

数据模型：一系列键值对

优势：快速查询

劣势：存储的数据缺少结构化

列存储数据库

这部分数据库通常是用来应对分布式存储的海量数据。键仍然存在，但是它们的特点是指向了多个列。这些列是由列家族来安排的。

相关产品：Cassandra, HBase, Riak

典型应用：分布式的文件系统

数据模型：以列簇式存储，将同一列数据存在一起

优势：查找速度快，可扩展性强，更容易进行分布式扩展

劣势：功能相对局限

文档型数据库

文档型数据库的灵感是来自于Lotus Notes办公软件的，而且它同第一种键值存储相类似。该类型的数据模型是版本化的文档，半结构化的文档以特定的格式存储，比如JSON。文档型数据库可以看作是键值数据库的升级版，允许之间嵌套键值。而且文档型数据库比键值数据库的查询效率更高。

相关产品：CouchDB、MongoDB

典型应用：Web应用（与Key-Value类似，Value是结构化的）

数据模型：一系列键值对

优势：数据结构要求不严格

劣势：查询性能不高，而且缺乏统一的查询语法

图形(Graph)数据库

图形结构的数据库同其他行列以及刚性结构的SQL数据库不同，它是使用灵活的图形模型，并且能够扩展到多个服务器上。NoSQL数据库没有标准的查询语言(SQL)，因此进行数据库查询需要制定数据模型。许多NoSQL数据库都有REST式的数据接口或者查询API。

相关数据库：Neo4J、InfoGrid、Infinite Graph

典型应用：社交网络

数据模型：图结构

优势：利用图结构相关算法。

劣势：需要对整个图做计算才能得出结果，不容易做分布式的集群方案。

总结：

因此，我们总结NoSQL数据库在以下的这几种情况下比较适用：

- 1、数据模型比较简单；
- 2、需要灵活性更强的IT系统；
- 3、对数据库性能要求较高；
- 4、不需要高度的数据一致性；
- 5、对于给定key，比较容易映射复杂值的环境

1.3 Redis历史

2008年，意大利的一家创业公司Merzia推出了一款基于MySQL的网站实时统计系统LL00GG，然而没过多久该公司的创始人 Salvatore Sanfilippo便对MySQL的性能感到失望，于是他决定亲自为LL00GG量身定做一个数据库，并于2009年开发完成，这个数据库就是Redis。不过Salvatore Sanfilippo并不满足只将Redis用于LL00GG这一款产品，而是希望更多的人使用它，于是在同一年Salvatore Sanfilippo将Redis开源发布，并开始和Redis的另一名主要的代码贡献者Pieter Noordhuis一起继续着Redis的开发，直到今天。

Salvatore Sanfilippo自己也没有想到，短短的几年时间，Redis就拥有了庞大的用户群体。Hacker News在2012年发布了一份数据库的使用情况调查，结果显示有近12%的公司在使用Redis。国内如新浪微博、街旁网、知乎网，国外如GitHub、Stack Overflow、Flickr等都是Redis的用户。

VMware公司从2010年开始赞助Redis的开发， Salvatore Sanfilippo和Pieter Noordhuis也分别在3月和5月加入VMware，全职开发Redis。

redis的作者，他叫Salvatore Sanfilippo，来自意大利的西西里岛，现在居住在卡塔尼亚。

代码开源。地址是antirez.com，当然也可以去follow他的github，地址是<http://github.com/antirez>。

Redis描述

什么是Redis

Redis 是完全开源免费的，遵守BSD协议，是一个高性能(NOSQL)的key-value数据库,Redis是一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库，并提供多种语言的API。

Redis特点

- **性能极高** – 由于数据是存储在内存中（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支持的类型 String, Hash, List, Set 及 Ordered Set 等数据类型操作。
- **原子性** – 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。
- **高速读写**，redis使用自己实现的分离器，代码量很短，没有使用lock（MySQL），因此效率非常高。

Redis是一个简单的，高效的，分布式的，基于内存的缓存工具。

架设好服务器后，通过网络连接（类似数据库），提供Key-Value式缓存服务。

简单，是Redis突出的特色。

简单可以保证核心功能的稳定和优异。

1.4 Redis的应用场景

根据业内项目实践经验及企业级开发中：

可以用作数据库、缓存、秒杀、计数器、排行榜、热点数据（经常会被查询，但是不经常被修改或者删除的数据）、分布式锁、分布式ID、和消息中间件等大部分功能等性能密切相关场景里

redis常用的场景示例如下：

1、缓存

缓存现在几乎是所有中大型网站都在用的必杀技，合理的利用缓存不仅能够提升网站访问速度，还能大大降低数据库的压力。**Redis**提供了键过期功能，也提供了灵活的键淘汰策略，所以，现在**Redis**用在缓存的场合非常多。

2、排行榜

很多网站都有排行榜应用的，如京东的月度销量榜单、商品按时间的上新排行榜等。**Redis**提供的有序集合数据类型功能实现各种复杂的排行榜应用。

3、计数器

什么是计数器，如电商网站商品的浏览量、视频网站视频的播放数等。为了保证数据实时性，每次浏览都得给+1，并发量高时如果每次都请求数据库操作无疑是种挑战和压力。**Redis**提供的**incr**命令来实现计数器功能，内存操作，性能非常好，非常适用于这些计数场景。

4、分布式会话

集群模式下，在应用不多的情况下一般使用容器自带的**session**复制功能就能满足，当应用增多相对复杂的系统中，一般都会搭建以**Redis**等内存数据库为中心的**session**服务，**session**不再由容器管理，而是由**session**服务及内存数据库管理。

5、分布式锁

在很多互联网公司中都使用了分布式技术，分布式技术带来的技术挑战是对同一个资源的并发访问，如全局ID、减库存、秒杀等场景，并发量不大的场景可以使用数据库的悲观锁、乐观锁来实现，但在并发量高的场合中，利用数据库锁来控制资源的并发访问是不太理想的，大大影响了数据库的性能。可以利用**Redis**的**setnx**功能来编写分布式的锁，如果设置返回1说明获取锁成功，否则获取锁失败，实际应用中要考虑的细节要更多。

6、社交网络

点赞、踩、关注/被关注、共同好友等是社交网站的基本功能，社交网站的访问量通常来说比较大，而且传统的关系数据库类型不适合存储这种类型的数据。**Redis**提供的哈希、集合等数据结构能很方便的实现这些功能。

7、最新列表

Redis列表结构，**LPUSH**可以在列表头部插入一个内容ID作为关键字，**LTRIM**用来限制列表的数量，这样列表永远为N个ID，无需查询最新的列表，直接根据ID去到对应的内容页即可。

8、消息系统

消息队列是大型网站必备中间件，如**ActiveMQ**、**RabbitMQ**、**Kafka**等流行的消息队列中间件，主要用于业务解耦、流量削峰及异步处理实时性低的业务。**Redis**提供了发布/订阅及阻塞队列功能，能实现一个简单的消息队列系统。另外，这个不能和专业的消息中间件相比。

1.5 Redis总结

1.5.1 Redis 优势

- **性能极高** – 由于数据是存储在内存中（**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**支持的类型 String, Hash, List, Set 及 Ordered Set 等数据类型操作。
- **原子性** – **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。
- **高速读写** **redis**使用自己实现的分离器，代码量很短，没有使用lock（MySQL），因此效率非常高。
- **持久化** **Redis**直接将数据存储在内存中，要将数据保存到磁盘上，**Redis**可以使用两种方式实现持久化过程。定时快照（snapshot）：每隔一段时间将整个数据库写到磁盘上，每次均是写全部数据，**代价非常高**。第二种方式基于语句追加（aof）：只追踪变化的数据，但是追加的log可能过大，同时所有的操作均重新执行一遍，**回复速度慢**。

1.5.2 Redis 缺点

- 耗内存，占用内存过高。
- 在线扩容，Redis难以支持在线扩容，尤其在集群场景里，当存储容量达到上限后，在线扩容会非常困难。

二 Redis安装

Redis安装目录介绍	目录位置
官网下载目录：	/opt
普通Redis安装目录：	/usr/local/redis
Docker安装目录：	/usr/local/docker/redis
Redis集群方式：	RedisCluster 6台服务器

2.1安装前准备

Redis官网

官方网站：<http://redis.io/> 官方下载：<http://redis.io/download> 可以根据需要下载不同版本（域名后缀io属于国家域名，是british/Indian Ocean territory，即英属印度洋领地）

Redis安装

Redis是C语言开发，安装Redis需要先将官网下载的源码进行编译，编译依赖gcc环境，如果没有gcc环境，需要安装gcc

```

-rw-r--r--. 1 root root 1711660 7月 24 21:59 redis-4.0.1.tar.gz
drwxr-xr-x. 13 root root 273 7月 25 21:55 tl
drwxr-xr-x. 2 root root 6 7月 23 22:47 Templates
drwxr-xr-x. 3 root root 19 7月 30 00:07 usr
drwxr-xr-x. 2 root root 6 7月 23 22:47 Videos
[root@localhost ~]# cd redis-4.0.1/
[root@localhost redis-4.0.1]# make
cd src && make all
make[1]: Entering directory `/root/redis-4.0.1/src'
CC Makefile.dep
make[1]: Leaving directory `/root/redis-4.0.1/src'
make[1]: Entering directory `/root/redis-4.0.1/src'
CC adlist.o
/bin/sh: cc: command not found
make[1]: *** [adlist.o] Error 127
make[1]: Leaving directory `/root/redis-4.0.1/src'
make: *** [all] Error 2
```


安装gcc

gcc的安装很简单，首先要确保root登录，其次就是Linux要能连外网

```
yum -y install gcc automake autoconf libtool make
```

注意：运行yum时出现/var/run/yum.pid已被锁定,PID为xxxx的另一个程序正在运行的问题解决

```
rm -f /var/run/yum.pid
```

2.2安装Redis

下载redis二进制安装包

```
wget http://download.redis.io/releases/redis-5.0.0.tar.gz
```

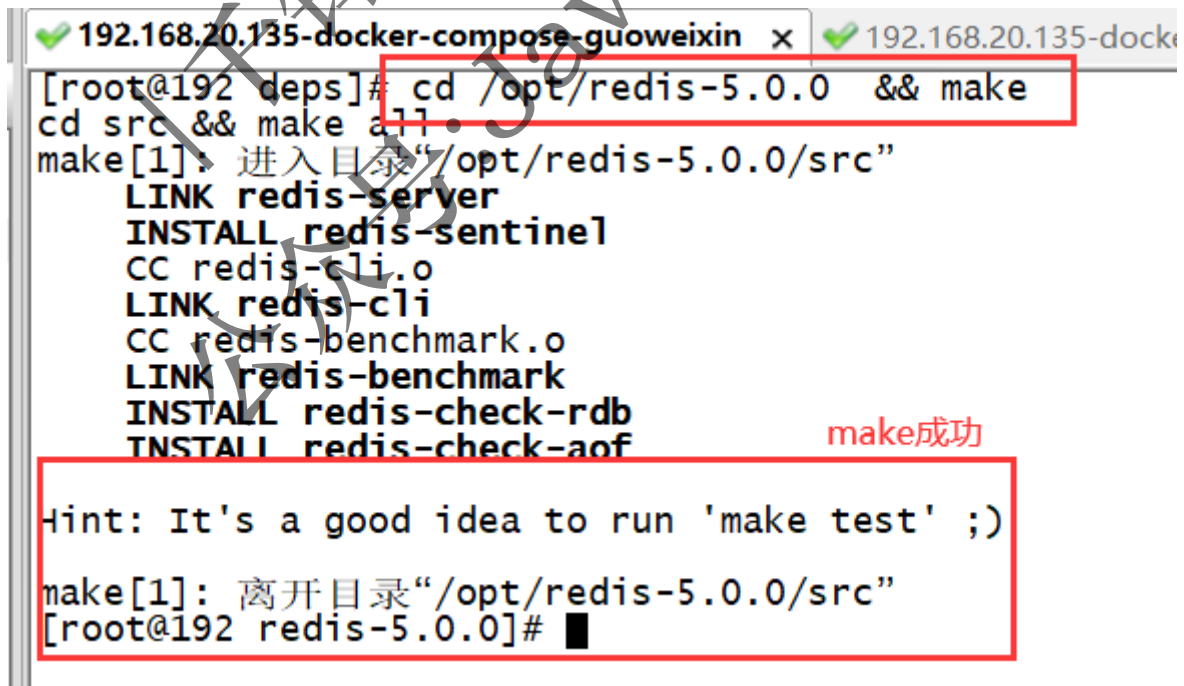
解压到/opt目录下

```
tar zxvf redis-5.0.0.tar.gz -C /opt
```

编译

```
cd /opt/redis-5.0.0 && make MALLOC=libc //或 make MALLOC=libc
```

成功如下图：



The screenshot shows a terminal window with two tabs. The active tab is titled '192.168.20.135-docker-compose-guoweixin'. The terminal output shows the following commands and results:

```
[root@192 deps]# cd /opt/redis-5.0.0 && make
cd src && make all
make[1]: 进入目录“/opt/redis-5.0.0/src”
LINK redis-server
INSTALL redis-sentinel
CC redis-cli.o
LINK redis-cli
CC redis-benchmark.o
LINK redis-benchmark
INSTALL redis-check-rdb
INSTALL redis-check-aof
Hint: It's a good idea to run 'make test' ;)
make[1]: 离开目录“/opt/redis-5.0.0/src”
[root@192 redis-5.0.0]#
```

The text 'make成功' (make success) is written in red in the terminal output.

注意：如果安装出现如下图：

```

[root@192 redis-5.0.0]# make MALLOC=libc
cd src && make all
make[1]: 进入目录“/opt/redis-5.0.0/src”
LINK redis-server
cc: 错误: ../deps/hiredis/libhiredis.a: 没有那个文件或目录
cc: 错误: ../deps/lua/src/liblua.a: 没有那个文件或目录
make[1]: *** [redis-server] 错误 1
make[1]: 离开目录“/opt/redis-5.0.0/src”
make: *** [all] 错误 2
[root@192 redis-5.0.0]# make
cd src && make all
make[1]: 进入目录“/opt/redis-5.0.0/src”
LINK redis-server

```

则进入redis下的deps下的运行如下命令，再重新运行 `cd /opt/redis-5.0.0 && make` 即可

```

cd /opt/redis-5.0.0/deps
make lua hiredis linenoise

```

指定安装位置

```
make PREFIX=/usr/local/redis install
```

(安装编译后的文件) 安装到指定目录:

注意: PREFIX必须大写、同时会自动为我们创建redis目录，并将结果安装此目录

查看安装后的文件

```

[root@192 redis]# cd bin
[root@192 bin]# ll
总用量 12876
-rwxr-xr-x. 1 root root 354080 12月 25 23:53 redis-benchmark
-rwxr-xr-x. 1 root root 4017680 12月 25 23:53 redis-check-aof
-rwxr-xr-x. 1 root root 4017680 12月 25 23:53 redis-check-rdb
-rwxr-xr-x. 1 root root 771448 12月 25 23:53 redis-cli
lrwxrwxrwx. 1 root root 12 12月 25 23:53 redis-sentinel -> redis-server
-rwxr-xr-x. 1 root root 4017680 12月 25 23:53 redis-server
[root@192 bin]#

```

三 Redis启动

启动Redis服务端

进入对应的安装目录

```
cd /usr/local/redis
```

执行命令:

```
./bin/redis-server
```




启动Redis客户端

在redis的安装目录中有redis的客户端，即redis-cli（Redis Command Line Interface），它是Redis自带的基于命令行的Redis客户端

进入Redis服务端（Clone Session克隆一个窗口） 进入对应的安装目录

```
cd /usr/local/redis
```

执行命令：

```
./bin/redis-cli
```

启动Redis 客户端命令语法：

```
redis-cli -h IP地址 -p 端口 //默认IP本机 端口6379
```

退出客户端命令： Ctrl+C

检测是否服务端启动

启动 redis 客户端，打开终端并输入命令 **redis-cli**。该命令会连接本地的 redis 服务。

```
$redis-cli
redis 127.0.0.1:6379>
redis 127.0.0.1:6379> PING
PONG
```

在以上实例中我们连接到本地的redis 服务并执行 **PING** 命令，该命令用于检测 redis 服务是否启动

四 Redis配置详解

Redis默认定义了很多默认配置。但在实际开发中，一般我们都会通过手动配置完成。回到安装目录下找到解压文件中的reids.conf

Redis 的配置文件位于 Redis 安装目录下，文件名为 redis.conf

配置Redis

命令：解压目录下的redis.conf 配置文件复制 到安装文件的目录下

```
cp /opt/redis-5.0.0/redis.conf /usr/local/redis/
```

核心配置文件意义：

高并发、高可用、高性能。

高性能：

技术学习有三步：

1、初级：为什么、是什么，怎么用。

2、中级：应用场景、解决哪些问题、应用的方法和技巧。

3、高级：nginx高级（核心配置文件）、tomcat高级（核心配置文件）、MySQL（核心配置文件）

Redis高级：优化。

4、Redis高级运维。企业级开发中。（架构师需要关心点）---> 学完Redis所有命令和结构类型应用场、各种实践。

redis.conf

前10个

****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(默认只允许127.0.0.1Redis发起访问)
`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. 指定在多长时间里，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（压缩算法）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
```

中间10个

**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slave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，slave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存放在内存，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（折衷，默认值）
- `appendfsync everysec`

结尾10个

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）
- `activeresharding 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中的内存维护策略

redis作为优秀的中间缓存件，时常会存储大量的数据，即使采取了集群部署来动态扩容，也应该即时的整理内存，维持系统性能。

在redis中有两种解决方案

一 为数据设置超时时间

设置过期时间

`expire key time`(以秒为单位)--这是最常用的方式
`setex(String key, int seconds, String value)`--字符串独有的方式

- 除了字符串自己独有设置过期时间的方法外，其他方法都需要依靠`expire`方法来设置时间
- 如果没有设置时间，那缓存就是永不过期
- 如果设置了过期时间，之后又想让缓存永不过期，使用`persist key`

二 采用LRU算法动态将不用的数据删除

内存管理的一种页面置换算法，对于在内存中但又不用数据块（内存块）叫做LRU，操作系统会根据哪些数据属于LRU而将其移出内存而腾出空间来加载另外的数据。

- 1.volatile-lru：设定超时时间的数据中,删除最不常使用的数据.
- 2.allkeys-lru：查询所有的key中最近最不常使用的数据进行删除，这是应用最广泛的策略.
- 3.volatile-random：在已经设定了超时的数据中随机删除.
- 4.allkeys-random：查询所有的key,之后随机删除.
- 5.volatile-ttl：查询全部设定超时时间的数据,之后排序,将马上将要过期的数据进行删除操作.
- 6.noeviction：如果设置为该属性,则不会进行删除操作,如果内存溢出则报错返回.
- 7.volatile-lfu：从所有配置了过期时间的键中驱逐使用频率最少的键
- 8.allkeys-lfu：从所有键中驱逐使用频率最少的键

自定义配置Redis

进入对应的安装目录 `/usr/local/redis` 修改 `redis.conf` 配置文件 `vim redis.conf` (进入命令模式 通过/内容 查找相应字符串)

`daemonize no` 修改为 `daemonize yes` 守护进程启动
`bind 127.0.0.1` 注释掉 允许除本机外的机器访问Redis服务
`requirepass` 设置密码 设定数据库密码（保证服务安全/有些情况下不设定密码是无法进行远程连接访问的）

Redis采用的是单进程多线程的模式。当`redis.conf`中选项`daemonize`设置成`yes`时，代表开启守护进程模式。在该模式下，redis会在后台运行，并将进程pid号写入至`redis.conf`选项`pidfile`设置的文件中，此时redis将一直运行，除非手动kill该进程。但当`daemonize`选项设置成`no`时，当前界面将进入redis的命令行界面，`exit`强制退出或者关闭连接工具(`putty`,`xshell`等)都会导致redis进程退出。服务端开发的大部分应用都是采用后台运行的模式

requirepass设置密码。因为redis速度相当快，所以一台比较好的服务器下，一个外部用户在一秒内可以进行15W次密码尝试，这意味着你需要设定非常强大的密码来防止暴力破解。

可以通过 redis 的配置文件设置密码参数，这样客户端连接到 redis 服务就需要密码验证，这样可以让你的 redis 服务更安全

五 Redis启动

服务端启动：

```
./bin/redis-server ./redis.conf
```

客户端登录：用redis-cli 密码登陆（redis-cli -a password）

```
redis-cli -h host -p port -a password //redis-cli -h IP地址 -p 端口 -a 密码
```

六 Redis关闭

第一种关闭方式：

（断电、非正常关闭。容易数据丢失） 查询redis进程id

```
PID ps -ef | grep -i redis
```

kill对 查询的id进行强制关闭

```
kill -9 PID
```

第二种关闭方式

（正常关闭、数据保存）

关闭redis服务，通过客户端进行shutdown

shutdown命令会终止服务器上的所有客户端连接，并终止服务器。

如果redis设置了密码，需要先在客户端通过密码登录，再进行shutdown即可关闭服务端

```
root redis]# ./bin/redis-cli -a guoweixin
379> shutdown
```

通过 `ps -ef | grep -i redis` 查看当前进程：

```
root redis]# ps -ef | grep -i redis
root      9289      1   0 22:23 ?        00:00:00 ./bin/redis-server 127.0.0.1:6379
root      9294    7553   0 22:23 pts/1    00:00:00 ./bin/redis-cli
root      9317    7361   0 22:25 pts/0    00:00:00 grep --color=auto -i redis
root redis]# ./bin/redis-cli shutdown
root redis]# ps -ef | grep -i redis
root      9294    7553   0 22:23 pts/1    00:00:00 ./bin/redis-cli
root      9329    7361   0 22:26 pts/0    00:00:00 grep --color=auto -i redis
```


七 远程连接

远程连接比较流行的软件：`RedisDesktopManager`

默认不允许远程连接，需要修改一下信息才可以进行修改，

```
bind 127.0.0.1 注释掉 允许除本机外的机器访问Redis服务
requirepass 设置密码 设定数据库密码（有些情况下不设定密码是无法进行远程连接访问的）
```

八 Docker 安装Redis

安装单机版Redis

1、搜索redis

```
docker search redis
```

2、下载镜像

```
docker pull redis:4.0.1
```

3、创建并运行容器

```
#默认配置文件方式 docker run -d --name redis6379 -p 6379:6379 redis:4.0.1 --requirepass "guoweixin"
```

为方便后续对配置文件进行高级修改。此处用自定义配置文件

```
#获取 redis.conf 配置文件，用数据卷方式
#1 官网下载redis安装包，解压。进入目录 找到redis.conf。并复制到指定目录下
mkdir -p /usr/local/docker/redis
cp redis.conf /usr/local/docker/redis
```

```
#2 修改redis.conf中相应属性值
# 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 #修改为 daemonize yes 守护进程启动
# 你可以绑定单一接口，如果没有绑定，所有接口都会监听到来的连接
# bind 127.0.0.1
# 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
# 设置Redis连接密码，如果配置了连接密码，默认关闭
requirepass guoweixin
```

创建容器

```
docker run -p 6379:6379 --name redis6379 -v
/usr/local/docker/redis/redis.conf:/etc/redis/redis.conf -v
/usr/local/docker/redis/data:/data -d redis redis-server
```

4、测试Redis 进入客户端

使用redis镜像执行redis-cli命令连接到刚启动的容器

```
docker exec -it redis6379 redis-cli //开启客户端功能
docker exec -ti redis6379 redis-cli -h 127.0.0.1 -p 6379
//如果有密码, 使用-a参数
docker exec -ti redis6379 redis-cli -h 127.0.0.1 -p 6379 -a guoweixin

# config set requirepass guoweixin 可通过此种方法设置密码
```

千锋教育Java教研院 关注公众号【Java架构栈】下载所有课程代码课件及工具 让技术回归本该有的纯静! |千锋教育|千锋Java|公众号:Java架构栈 作者:Wilson

Centos防火墙端口

开放8080端口 (如下命令只针对Centos7以上)

查看已经开放的端口:

```
firewall-cmd --list-ports
```

开启端口:

```
firewall-cmd --zone=public --add-port=6379/tcp --permanent
```

重启防火墙:

```
firewall-cmd --reload #重启
firewall systemctl stop firewalld.service #停止
firewall systemctl disable firewalld.service #禁止firewall开机启动
```

千锋教育Java教研院 关注公众号【Java架构栈】下载所有课程代码课件及工具 让技术回归本该有的纯静! 作者:Wilson