

Redis 从入门到实战

一篇文章学会Redis

REmote DIctionary Server(Redis) 是一个开源的，基于key-value键值对的持久化的非关系型数据库存储系统。它支持的数据存储类型包括：字符串(String), 哈希(Map), 列表(list), 集合(sets) 和 有序集合(sorted sets)等。在实际项目中可以用 Redis 做缓存或消息服务器，Redis 也是目前互联网中使用比较广泛的非关系型数据库。

本场 **Chat** 将从以下 6 个方面介绍 **Redis** 的使用：

1. Redis的简介，特点，缺陷和应用场景。
2. Redis的安装，分别介绍Windows与Linux下安装Redis。
3. Redis的配置，介绍redis.conf文件查看、修改和相关参数说明。
4. Redis的五种数据类型与使用，包括key-value，String，Hash，List，Set，sorted set。
5. Java代码如何操作Redis。
6. Redis集群简单介绍，主从、哨兵、集群。

通过本场 Chat 的学习，以达到“从零开始轻松掌握 Redis，并且可以运用到项目中”的目的。

一、Redis的简介

Redis是一个开源的使用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，推荐使用Linux进行部署。

官方网站：<https://redis.io>

1. Redis有以下几个特点

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。

方式一（**RDB**）：根据指定的规则，定时周期性的把内存中更新的数据写入到磁盘里。RDB的方式是通过快照（snapshot）完成，当符合规则时redis会把内存的数据生成一个副本并存储在硬盘中，这个过程称之为“快照”。

方式二（**AOF**）：把修改的操作记录追加到文件里，默认情况redis没有开启AOF方式，可以通过appendonly命令来启用，如：appendonly yes。

两种方式的区别：RDB方式性能较高，但是可能会引起一定程度的数据丢失，AOF方式正好相反。

丰富的数据类型

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。

单进程单线程高性能服务器

启动一个实例只能用一个CPU，所以用Redis可以用多个实例，一个实例用一个CPU以便提高效率。

crash safe 和 recovery slow

redis崩溃后，数据相对安全，但是恢复起来比较缓慢，所以生产环境不建议一个Redis实例数据太多{（20-30）G数据内存对应（96-128）G实际内存}，这种20%-23%的比例比较合适，因为磁盘读到内存的恢复时间也很慢，可以使用ssd磁盘来提高磁盘读取速度。

性能极高

Redis单机qps（每秒的并发）可以达到的速度是110000次/s，写的速度是81000次/s，适合小数据量高速读写访问。

Redis的原子性

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。

Redis支持数据的备份即master-slave模式的数据备份。Redis还支持 publish/subscribe, key过期。

2. Redis缺陷与陷阱

内存管理开销大（不要超过物理内存的3/5）。buffer io 可能会造成系统内存溢出（OOM-Out of Memory）。

3. Redis的应用场景

新浪（sina）使用redis：

1. 应用程序直接访问Redis。
2. Redis当做内存，先访问redis，redis没有数据或访问失败时访问MySQL。
3. 二次开发后实现MySQL和Redis互相同步。

MySQL数据通过RBR解析BINLOG同步到redis中实现关系型数据转变成队列数据。





Redis提供特定数据的读写，通过replication接口同时写入到MySQL。

二、Redis的安装

Window下安装Redis

下载地址：<https://github.com/MSOpenTech/redis/releases>

Downloads

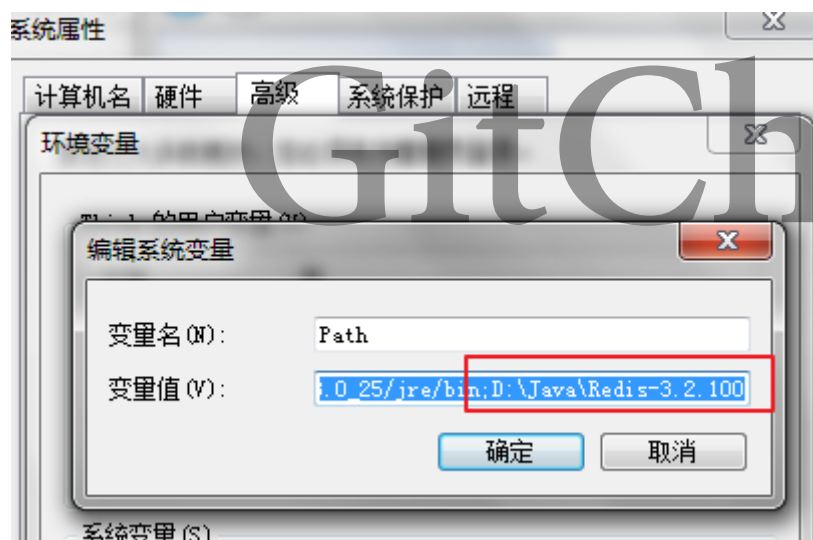
 Redis-x64-3.2.100.msi	5.8 MB
 Redis-x64-3.2.100.zip	4.98 MB
 Source code (zip)	
 Source code (tar.gz)	

Redis 支持 32 位和 64 位，这个需要根据你系统平台的实际情况选择，这里我们下载 Redis-x64-xxx.zip压缩包到你的目录（我的是：D:\Java\Redis-3.2.100），解压后，将文件夹重新命名为Redis-3.2.100，如下图：

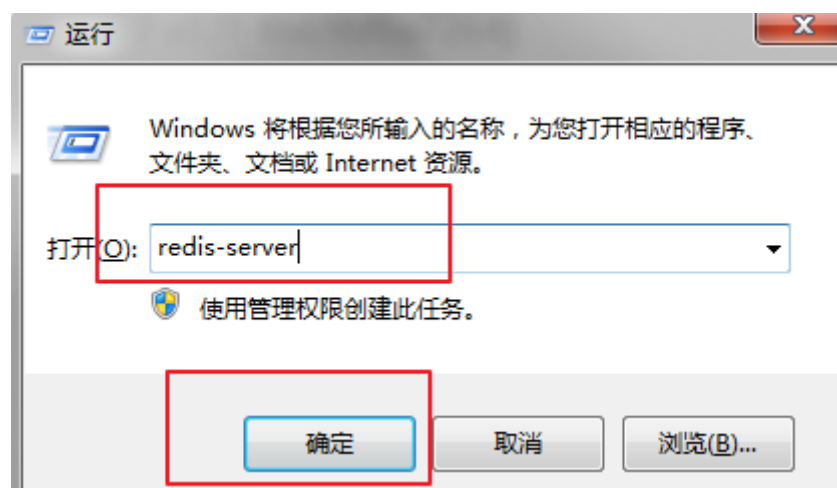
EventLog.dll	2016/7/1 16:27
Redis on Windows Release Notes.do...	2016/7/1 16:07
Redis on Windows.docx	2016/7/1 16:07
redis.windows.conf	2016/7/1 16:07
redis.windows-service.conf	2016/7/1 16:07
redis-benchmark.exe	2016/7/1 16:28
redis-benchmark.pdb	2016/7/1 16:28
redis-check-aof.exe	2016/7/1 16:28
redis-check-aof.pdb	2016/7/1 16:28
redis-cli.exe	2016/7/1 16:28
redis-cli.pdb	2016/7/1 16:28
redis-server.exe	2016/7/1 16:28
redis-server.pdb	2016/7/1 16:28
Windows Service Documentation.docx	2016/7/1 9:17

打开一个cmd窗口，使用cd命令切换目录到 D:\Java\Redis-3.2.100运行 redis-server.exe redis.windows.conf，可以启动redis服务。

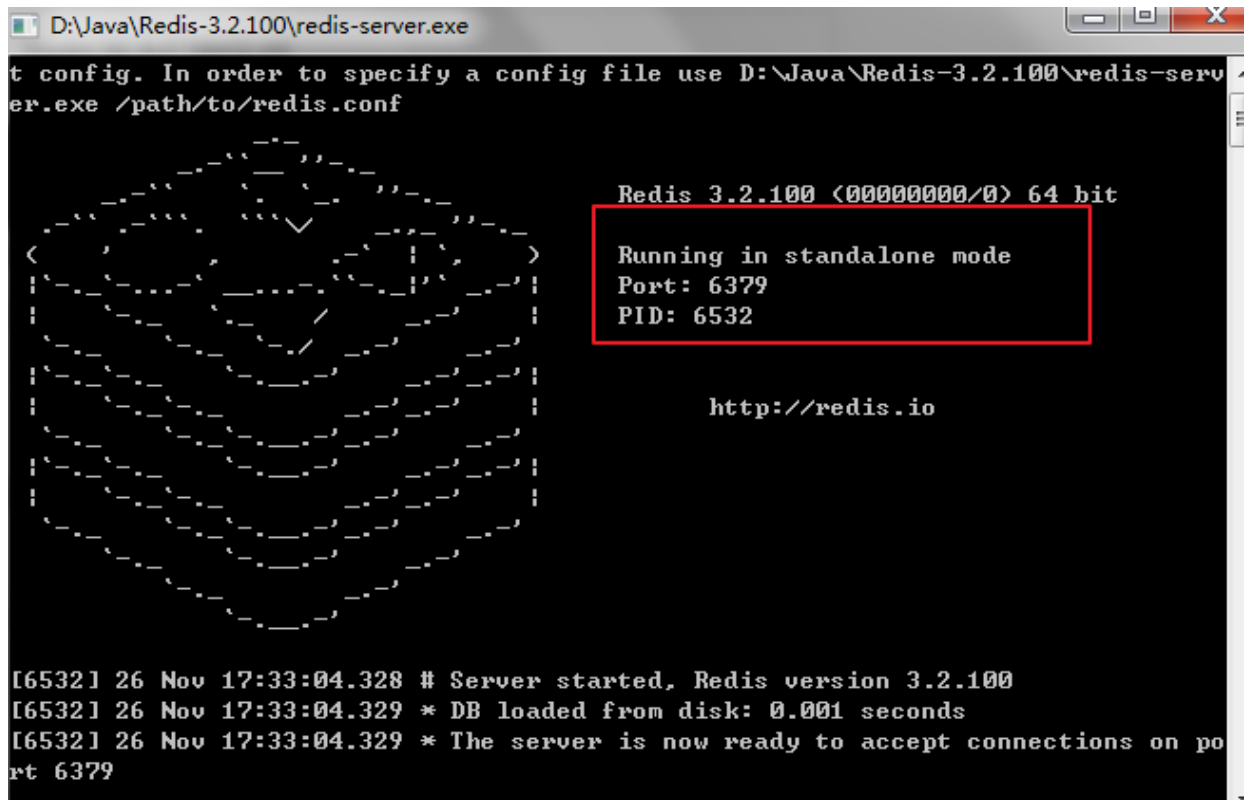
如果想方便的话，可以把 redis 的路径加到系统的环境变量里，这样就省得再输路径了，后面的那个 redis.windows.conf 可以省略，如果省略，会启用默认的。



配置环境变量后，可以直接在cmd窗口，启动redis服务，如下：



启动成功显示：



```
t config. In order to specify a config file use D:\Java\Redis-3.2.100\redis-serv
er.exe /path/to/redis.conf

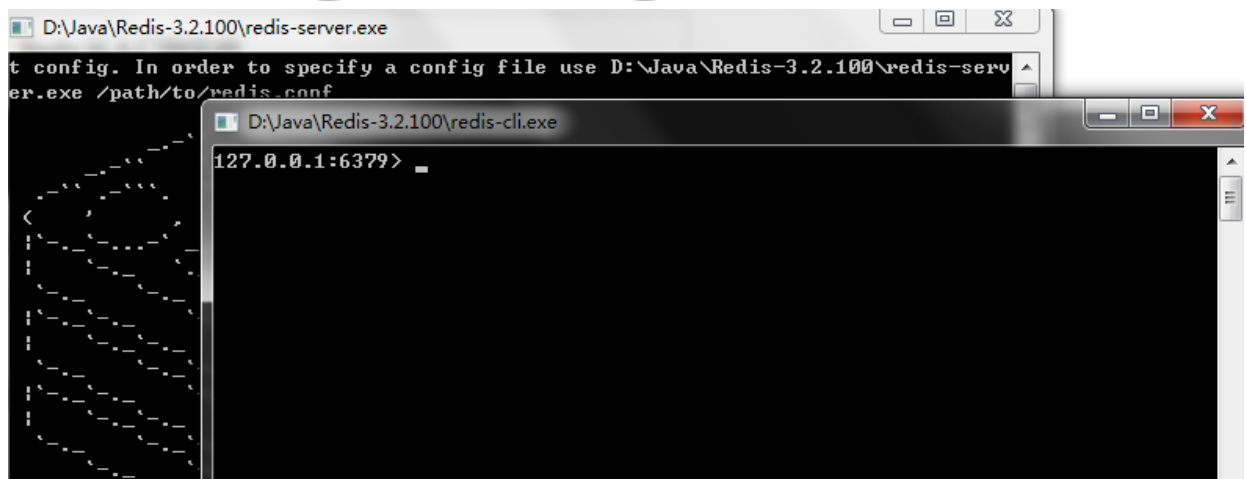
Redis 3.2.100 (00000000/0) 64 bit
Running in standalone mode
Port: 6379
PID: 6532

http://redis.io

[6532] 26 Nov 17:33:04.328 # Server started, Redis version 3.2.100
[6532] 26 Nov 17:33:04.329 * DB loaded from disk: 0.001 seconds
[6532] 26 Nov 17:33:04.329 * The server is now ready to accept connections on po
rt 6379
```

下面让我们来看一下，在window环境下使用Redis。

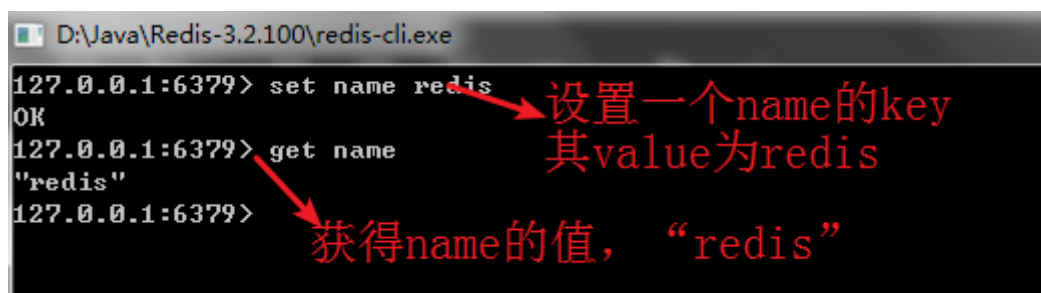
这时候另启一个cmd窗口，原来的不要关闭，不然就无法访问服务端了。因为配置了环境变量，所以可以直接在运行窗口，输入“redis-cli”打开客户端，如下图：



```
D:\Java\Redis-3.2.100\redis-server.exe
t config. In order to specify a config file use D:\Java\Redis-3.2.100\redis-serv
er.exe /path/to/redis.conf

D:\Java\Redis-3.2.100\redis-cli.exe
127.0.0.1:6379>
```

在客户端界面，使用redis，过程如下图：



```
D:\Java\Redis-3.2.100\redis-cli.exe
127.0.0.1:6379> set name redis
OK
127.0.0.1:6379> get name
"redis"
127.0.0.1:6379>
```

设置一个name的key
其value为redis

获得name的值，“redis”

到这里，windows下安装Redis已经完成。

Linux下安装Redis

下载地址: <https://redis.io/download>

详细命令:

- \$ wget <http://download.redis.io/releases/redis-4.0.2.tar.gz>
- \$ tar -zxvf redis-4.0.2.tar.gz -C /usr/local/
- \$ cd redis-4.0.2
- make

注意: 直接make如果报错, 如下图:

```
make[1]: Entering directory `/usr/local/redis-4.0.2/src'
CC adlist.o
在包含自 adlist.c: 34 的文件中:
zmalloc.h:50:31: 错误: jemalloc/jemalloc.h: 没有那个文件或目录
zmalloc.h:55:2: 错误: #error "Newer version of jemalloc required"
make[1]: *** [adlist.o] 错误 1
make[1]: Leaving directory `/usr/local/redis-4.0.2/src'
make: *** [all] 错误 2
```

则执行make MALLOC=libc, 编译成功后, 如下图:

```
Hint: It's a good idea to run 'make test' ;)

make[1]: Leaving directory `/usr/local/redis-4.0.2/src'
```

Linux下启动redis服务, 命令: “./redis-server”, 如下图:

```
defrag.o  hyperloglog.c  module.o  rax.h  redis-trib.rb  sha1.c  t_set.o
src]# ./redis-server
01:12:36.141 # 0000000000000000 Redis is starting 0000000000000000
01:12:36.141 # Redis version=4.0.2, bits=64, commit=00000000, modified=0, pid=3970, just started
01:12:36.141 # Warning: no config file specified, using the default config. In order to specify a config file use ./redis-s
redis.conf
01:12:36.143 * Increased maximum number of open files to 10032 (it was originally set to 1024).

Redis 4.0.2 (00000000/0) 64 bit
Running in standalone mode
Port: 6379
PID: 3970
```

Linux下使用redis服务, 命令: “./redis-cli”如下图:

```
defrag.o  hyperloglog.c  module.o
src]# ./redis-cli
79> get *
```

ping命令检查是否成功启动redis服务:

```
127.0.0.1:6379> ping
PONG
127.0.0.1:6379> █
```

如果需要在远程 redis 服务上执行命令, 同样我们使用的也是 redis-cli 命令: redis-cli -h host -p port -a password

命令: `redis-cli -h 127.0.0.1 -p 6379 -a "mypassword"`

表示: 连接到主机为 127.0.0.1, 端口为6379, 密码为 mypassword的redis服务上。

至此, Redis的安装介绍完毕。

三、Redis的配置

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

```
00-RELEASENOTES  CONTRIBUTING  deps  Makefile  README.md  runtest  runtest-sentinel  src  utils
BUGS             COPYING      INSTALL  MANIFESTO  redis.conf  runtest-cluster  sentinel.conf  tests
```

1. redis.conf文件使用说明

你可以通过 `CONFIG get` 命令查看或设置配置项, 使用 `config get *` 获取所有配置项。

```
127.0.0.1:6379> config get dbfilename
1) "dbfilename"
2) "dump.rdb"
127.0.0.1:6379>
```

上面的命令是通过 `config get dbfilename` 来获取 `redis.conf` 文件中 `dbfilename` 的配置项的值。

你可以通过修改 `redis.conf` 文件或使用 `CONFIG set` 命令来修改配置。

下面通过命令 `config set` 来修改日志记录级别。

```
127.0.0.1:6379> config get loglevel
1) "loglevel"
2) "notice"
127.0.0.1:6379> config set loglevel debug
OK
127.0.0.1:6379> config get loglevel
1) "loglevel"
2) "debug"
```

loglevel的级别开始为notice
loglevel的级别修改为debug
修改成功

Redis默认不是以守护进程的方式运行, 可以通过修改 `daemonize` 配置项, 使用 `yes` 启用守护进程, 以达到redis在后台运行, 如下图:

```
##### GENERAL #####

# By default Redis does not run as a daemon. Use 'yes' if you need it.
# Note that Redis will write a pid file in /var/run/redis.pid when daemonized.
daemonize no

# If you run Redis from upstart or systemd, Redis can interact with your
# supervision tree. Options:
```

修改为yes, 则开启后台运行redis服务

2. redis.conf文件参数说明* 标黄的要熟悉*

序号	redis.conf参数	说明
1	daemonize no	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
2	pidfile /var/run/redis.pid	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
3	port 6379	指定Redis监听端口，默认端口为6379
4	bind 127.0.0.1	绑定的主机地址
5	timeout 300	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
6	loglevel verbose	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
7	logfile stdout	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，则日志将会发送给/dev/null
8	databases 16	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
9	save <seconds> <changes>	指定在多长时间，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	rdbcompression yes	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
11	dbfilename dump.rdb	指定本地数据库文件名，默认值为dump.rdb
12	dir ./	指定本地数据库存放目录
13	slaveof <masterip> <masterport>	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
14	masterauth <master-password>	当master服务设置了密码保护时，slav服务连接master的密码
15	requirepass foobared	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
16	maxclients 128	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
17	maxmemory <bytes>	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
18	appendonly no	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
19	appendfilename appendonly.aof	指定更新日志文件名，默认为appendonly.aof
20	appendfsync everysec	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
21	vm-enabled no	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将访问量较少的页面冷数据swap到磁盘上，访问多的页面由磁盘自动换出到内存中
22	vm-swap-file /tmp/redis.swap	虚拟内存文件路径，默认值为/tmp/redis.swap，不可多个Redis实例共享
23	vm-max-memory 0	将所有大于vm-max-memory的数据存入虚拟内存，无论vm-max-memory设置多少，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
24	vm-page-size 32	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；如果存储很大对象，则可以使用更大的page，如果不 确定，就使用默认值
25	vm-pages 134217728	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。
26	vm-max-threads 4	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
27	glueoutputbuf yes	设置在向客户端应答时，是否把较小的包含并为一个包发送，默认为开启
28	hash-max-zipmap-entries 64hash-max-	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
29	activeremhashing yes	指定是否激活重置哈希，默认为开启
30	include /path/to/local.conf	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

四、Redis的五种数据类型与使用

1. Redis的key-value介绍

Redis key值是二进制安全的，下面说一下key的几条规则：

太长的键值不好

例如1024字节的键值就不好，不仅因为消耗内存，在数据中查找这类键值的成本也很高。

太短的键值通常也不好

键值的设置应该像变量命名一样，能标识出它的含义，比如“u:666:pwd”就不如“user:666:password”更易阅读。

最好坚持一种模式

例如：“object-type:id:field”，“user:666:password”。设置编码为666的用户的密码为666888，则可以使用如下命令：

```
set user:666:password 666888
```

key建议

10到20个字符。

value建议

String 不要超过2K。set元素不要超过5000，如果内容长度太多，可以根据内容长度规划不同的实例端口

Redis的key-value使用说明

Redis键命令的基本语法：COMMAND KEY_NAME。

下面截图是使用Redis键命令的示例：

```

127.0.0.1:6379> ping
PONG
127.0.0.1:6379> set name redis → 设置键名称为name, 值为redis
OK
127.0.0.1:6379> get name → 获取name的值
"redis"
127.0.0.1:6379> exists name → 判断name是否存在, 返回1, 表示存在
(integer) 1
127.0.0.1:6379> exists NAME → 判断NAME是否存在, 返回0, 表示没有
(integer) 0
127.0.0.1:6379> expire name 3600 → 为name设置过期时间, 3600秒即一小时
(integer) 1
127.0.0.1:6379> keys * → 获取所有的key
1) "name"
127.0.0.1:6379> rename name NAME → 重命名name的key为NAME, 成功返回OK
OK
127.0.0.1:6379> get name → 再次获取name的值就没有了
(nil)
127.0.0.1:6379> get NAME → 获取修改后的NAME得值为redis
"redis"
127.0.0.1:6379> type NAME → 查看NAME这个key的值的类型, 返回string
string
127.0.0.1:6379> DEL NAME → 删除key为NAME的键
(integer) 1
127.0.0.1:6379> keys *
(empty list or set)
127.0.0.1:6379>

```

Redis的keys命令（标黄的要熟悉）

序号	Redis keys 命令	描述
1	DEL key	该命令用于在 key 存在时删除 key
2	DUMP key	序列化给定 key，并返回被序列化的值
3	EXISTS key	检查给定 key 是否存在
4	EXPIRE key seconds	为给定 key 设置过期时间
5	EXPIREAT key timestamp	EXPIREAT 命令接受的时间参数是 UNIX 时间戳(unix timestamp)
6	PEXPIRE key milliseconds	设置 key 的过期时间以毫秒计
7	PEXPIREAT key milliseconds-timestamp	设置 key 过期时间的时间戳(unix timestamp) 以毫秒计
8	KEYS pattern	查找所有符合给定模式(pattern)的 key
9	MOVE key db	将当前数据库的 key 移动到给定的数据库 db 当中
10	PERSIST key	移除 key 的过期时间, key 将持久保持
11	PTTL key	以毫秒为单位返回 key 的剩余的过期时间
12	TTL key	以秒为单位, 返回给定 key 的剩余生存时间(TTL, time to live)
13	RANDOMKEY	从当前数据库中随机返回一个 key
14	RENAME key newkey	修改 key 的名称
15	RENAMENX key newkey	仅当 newkey 不存在时, 将 key 改名为 newkey
16	TYPE key	返回 key 所储存的值的类型

2. Redis的字符串(String)类型介绍**

这是Redis最简单而且最常用的数据类型之一。如果只使用这种数据类型，那么Redis就是一个持久化的memcached服务器（memcached的数据仅保存在内存中，服务器重启后，数据将丢失）。

在Redis中，我们通常用set设置一对key-value键值，然后用get来获取字符串的值。value值可以是任何类型的字符串（包括二进制数据），但值的长度不能超过1G。

String类型也可以用来存储数字，并支持对数字的加减操作。

Redis的字符串命令使用说明

```

127.0.0.1:6379> set name redis
OK
127.0.0.1:6379> get name
"redis"
127.0.0.1:6379> strlen name
(integer) 5
127.0.0.1:6379> set age 5
OK
127.0.0.1:6379> get age
"5"
127.0.0.1:6379> incr age
(integer) 6
127.0.0.1:6379> incrby age 2
(integer) 8
127.0.0.1:6379> decr age
(integer) 7
127.0.0.1:6379> decr age 3
(error) ERR wrong number of arguments for 'decr' command
127.0.0.1:6379> decrby age 3
(integer) 4
127.0.0.1:6379> append name success
(integer) 12
127.0.0.1:6379> get name
"redissuccess"
127.0.0.1:6379>

```

返回name的值redis的长度为5

将age的值加1变为6

将age的值加2变为8

将age的值减1变为7

将age的值减3, 需要使用decrby命令, decr只能默认减1

将age的值减3变为4

在已存在的name值redis后追加success字符串

Redis的字符串命令（标黄的要熟悉）

序号	Redis 字符串命令	描述
1	SET key value	设置指定 key 的值
2	GET key	获取指定 key 的值
3	GETRANGE key start end	返回 key 中字符串值的子字符
4	SETSET key value	将给定 key 的值设为 value，并返回 key 的旧值(old value)
5	GETBIT key offset	对 key 所储存的字符串值，获取指定偏移量上的位(bit)
6	MGET key1 [key2...]	获取所有(一个或多个)给定 key 的值
7	SETBIT key offset value	对 key 所储存的字符串值，设置或清除指定偏移量上的位(bit)
8	SETEX key seconds value	将值 value 关联到 key，并将 key 的过期时间设为 seconds（以秒为单位）
9	SETNX key value	只有在 key 不存在时设置 key 的值
10	SETRANGE key offset value	用 value 参数覆写给定 key 所储存的字符串值，从偏移量 offset 开始
11	STRLEN key	返回 key 所储存的字符串值的长度
12	MSET key value [key value ...]	同时设置一个或多个 key-value 对
13	MSETNX key value [key value ...]	同时设置一个或多个 key-value 对，当且仅当所有给定 key 都不存在
14	PSETEX key milliseconds value	它以毫秒为单位设置 key 的生存时间，而不是像 SETEX 命令那样，以秒为单位
15	INCR key	将 key 中储存的数字值增一
16	INCRBY key increment	将 key 所储存的值加上给定的增量值(increment)
17	INCRBYFLOAT key increment	将 key 所储存的值加上给定的浮点增量值(increment)
18	DECR key	将 key 中储存的数字值减一
19	DECRBY key decrement	key 所储存的值减去给定的减量值(decrement)
20	APPEND key value	如果 key 已经存在并且是一个字符串， APPEND 命令将 value 追加到 key 原来的值的末尾

3. Redis的哈希(Hash)类型介绍

Redis hash 是一个string类型的field和value的映射表，hash特别适合用于存储对象，能够存储key对多个属性的数据。

Redis的哈希(Hash)命令使用说明

```
127.0.0.1:6379> hset user name redis
(integer) 1
127.0.0.1:6379> hset user age 10
(integer) 1
127.0.0.1:6379> hset user sex male
(integer) 1
127.0.0.1:6379> hget user
(error) ERR wrong number of arguments for 'hget' command
127.0.0.1:6379> hget user name
"redis"
127.0.0.1:6379> hvals user
1) "redis"
2) "10"
3) "male"
127.0.0.1:6379> hgetall user
1) "name"
2) "redis"
3) "age"
4) "10"
5) "sex"
6) "male"
127.0.0.1:6379> hlen user
(integer) 3
127.0.0.1:6379> hkeys key
(empty list or set)
127.0.0.1:6379> hkeys user
1) "name"
2) "age"
3) "sex"
127.0.0.1:6379> █
```

设置哈希表中的字段的值
user中name为redis
age为10, sex为male

获取哈希表中name字段的值redis

获取哈希表中所有值

获取哈希表中指定key的所有字段及值

获取哈希表中字段的数量

获取哈希表中的字段

Redis的哈希(Hash)命令（标黄的要熟悉）

序号	Redis hash 命令	描述
1	HDEL key field1 [field2]	删除一个或多个哈希表字段
2	HEXISTS key field	查看哈希表 key 中，指定的字段是否存在
3	HGET key field	获取存储在哈希表中指定字段的值
4	HGETALL key	获取在哈希表中指定 key 的所有字段和值
5	HINCRBY key field increment	为哈希表 key 中的指定字段的整数值加上增量 increment
6	HINCRBYFLOAT key field increment	为哈希表 key 中的指定字段的浮点数值加上增量 increment
7	HKEYS key	获取所有哈希表中的字段
8	HLEN key	获取哈希表中字段的数量
9	HMGET key field1 [field2]	获取所有给定字段的值
10	HMSET key field1 value1 [field2 value2]	同时将多个 field-value (域-值)对设置到哈希表 key 中
11	HSET key field value	将哈希表 key 中的字段 field 的值设为 value
12	HSETNX key field value	只有在字段 field 不存在时，设置哈希表字段的值
13	HVALS key	获取哈希表中所有值
14	HSCAN key cursor [MATCH pattern] [COUNT count]	迭代哈希表中的键值对

4. Redis的列表(List)类型介绍

列表就是有序元素的序列，1,2,3,4,5,6就是一个列表。用数组实现的List和用Linked List实现的List，在属性方面大不相同。

Redis Lists基于Linked List实现。这意味着即使在一个list中有数百万个元素，在头部或尾部添加一个元素的操作，其时间复杂度也是常数级别的。用LPUSH命令在十个元素的list头部添加新元素，和在千万元素list头部添加新元素的速度相同。

Redis列表是简单的字符串列表，按照插入顺序排序。你可以添加一个元素到列表的头部（左边）或者尾部（右边）。

查看帮助，命令：help @list。

```

127.0.0.1:6379> clear
127.0.0.1:6379> help @list
查看帮助

BLPOP key [key ...] timeout
summary: Remove and get the first element in a list, or block until one is available
since: 2.0.0

BRPOP key [key ...] timeout
summary: Remove and get the last element in a list, or block until one is available
since: 2.0.0

BRPOPLPUSH source destination timeout
summary: Pop a value from a list, push it to another list and return it; or block until one is available
since: 2.2.0

LINDEX key index
summary: Get an element from a list by its index
since: 1.0.0

LINSERT key BEFORE|AFTER pivot value
summary: Insert an element before or after another element in a list
since: 2.2.0

LLEN key
summary: Get the length of a list
since: 1.0.0

LPOP key
summary: Remove and get the first element in a list
since: 1.0.0

```

Redis的列表(List)命令使用说明

```

127.0.0.1:6379> lpush users zhangsan
(integer) 1
127.0.0.1:6379> lpush users lisi wangwu
(integer) 3
127.0.0.1:6379> lpush users zhaoliu
(integer) 4
127.0.0.1:6379> llen users
(integer) 4
127.0.0.1:6379> lrange users 1 3
(error) ERR unknown command 'lrange'
127.0.0.1:6379> lrange users 1 3
1) "wangwu"
2) "lisi"
3) "zhangsan"
127.0.0.1:6379> lpop users
"zhaoliu"
127.0.0.1:6379> llen users
(integer) 3
127.0.0.1:6379> lrange user 0 3
(error) WRONGTYPE Operation against a key holding the wrong kind of value
127.0.0.1:6379> lrange users 0 3
1) "wangwu"
2) "lisi"
3) "zhangsan"
127.0.0.1:6379> lrange users 0 4
1) "wangwu"
2) "lisi"
3) "zhangsan"
127.0.0.1:6379> lrem users 1
(error) ERR wrong number of arguments for 'lrem' command
127.0.0.1:6379> lrem users 1 lisi
(integer) 1
127.0.0.1:6379> lrange users 0 4
1) "wangwu"
2) "zhangsan"
127.0.0.1:6379>

```

加入4个元素

返回list的长度为4

获取指定范围的元素

移除一个元素，剩下3个了

移除指定个数的值

Redis的列表(List)命令（标黄的要熟悉）

序号	Redis 列表命令	描述
1	BLPOP key1 [key2] timeout	移出并获取列表的第一个元素， 如果列表没有元素会阻塞列表直到等待超时或发现可弹出元素为止。
2	BRPOP key1 [key2] timeout	移出并获取列表的最后一个元素， 如果列表没有元素会阻塞列表直到等待超时或发现可弹出元素为止。
3	BRPOPLPUSH source destination timeout	从列表中弹出一个值，将弹出的元素插入到另外一个列表中并返回它； 如果列表没有元素会阻塞列表直到等待超时或发现可弹出元素为止。
4	LINDEX key index	通过索引获取列表中的元素
5	LINSERT key BEFORE AFTER pivot value	在列表的元素前或者后插入元素
6	LLEN key	获取列表长度
7	LPOP key	移出并获取列表的第一个元素
8	LPUSH key value1 [value2]	将一个或多个值插入到列表头部
9	LPUSHX key value	将一个值插入到已存在的列表头部
10	LRANGE key start stop	获取列表指定范围内的元素
11	LREM key count value	移除列表元素
12	LSET key index value	通过索引设置列表元素的值
13	LTRIM key start stop	对一个列表进行修剪(trim)，就是说，让列表只保留指定区间内的元素，不在指定区间之内的元素都将被删除。
14	RPOP key	移除并获取列表最后一个元素
15	RPOPLPUSH source destination	移除列表的最后一个元素，并将该元素添加到另一个列表并返回
16	RPUSH key value1 [value2]	在列表中添加一个或多个值
17	RPUSHX key value	为已存在的列表添加值

5. Redis的集合(Set)类型介绍

Redis 的 Set 是 String 类型的无序集合。集合成员是唯一的，这就意味着集合中不能出现重复的数据。

Redis的集合(Set)命令使用说明

```
127.0.0.1:6379> help sadd
SADD key member [member ...]
summary: Add one or more members to a set
since: 1.0.0
group: set

127.0.0.1:6379> sadd persons aaa
(integer) 1
127.0.0.1:6379> sadd persons bbb ccc
(integer) 2
127.0.0.1:6379> aadd persons ddd
(error) ERR unknown command 'aadd'
127.0.0.1:6379> sadd persons ddd
(integer) 1
127.0.0.1:6379> scard persons
(integer) 4
127.0.0.1:6379> sismember persons bbb
(integer) 1
127.0.0.1:6379> sismember persons fff
(integer) 0
127.0.0.1:6379> smembers persons
1) "bbb"
2) "aaa"
3) "ccc"
4) "ddd"
127.0.0.1:6379>
```

查看某个命令的帮助

添加元素aaa到集合persons中

获取集合persons的元素个数

判断集合中有没有bbb
1代表有，0代表没有

返回集合中的所有元素

Redis的集合(Set)命令（标黄的要熟悉）

序号	Redis 集合命令	描述
1	SADD key member1 [member2]	向集合添加一个或多个成员
2	SCARD key	获取集合的成员数
3	SDIFF key1 [key2]	返回给定所有集合的差集
4	SDIFFSTORE destination key1 [key2]	返回给定所有集合的差集并存储在 destination 中
5	SINTER key1 [key2]	返回给定所有集合的交集
6	SINTERSTORE destination key1 [key2]	返回给定所有集合的交集并存储在 destination 中
7	SISMEMBER key member	判断 member 元素是否是集合 key 的成员
8	SMEMBERS key	返回集合中的所有成员
9	SMOVE source destination member	将 member 元素从 source 集合移动到 destination 集合
10	SPOP key	移除并返回集合中的一个随机元素
11	SRANDMEMBER key [count]	返回集合中一个或多个随机数
12	SREM key member1 [member2]	移除集合中一个或多个成员
13	SUNION key1 [key2]	返回所有给定集合的并集
14	SUNIONSTORE destination key1 [key2]	所有给定集合的并集存储在 destination 集合中
15	SSCAN key cursor [MATCH pattern] [COUNT count]	迭代集合中的元素

6. Redis的有序集合(sorted set)类型介绍

Redis 有序集合和集合一样也是string类型元素的集合，且不允许重复的成员。不同的是每个元素都会关联一个double类型的分数，redis正是通过分数来为集合中的成员进行从小到大的排序。

Redis的有序集合(sorted set)命令使用说明

```

127.0.0.1:6379> zadd words 1 xiaomin
(integer) 1
127.0.0.1:6379> zadd words 2 xiaohong
(integer) 1
127.0.0.1:6379> zadd words 3 xiaozhao
(integer) 1
127.0.0.1:6379> zadd words 4 xiaoliu
(integer) 1
127.0.0.1:6379> zadd words 5 dawang
(integer) 1
127.0.0.1:6379> zrange words 0 5
1) "xiaomin"
2) "xiaohong"
3) "xiaozhao"
4) "xiaoliu"
5) "dawang"
127.0.0.1:6379> zscore words xiaozhao
"3"
127.0.0.1:6379> zscore words dawang
(error) ERR unknown command 'zscore'
127.0.0.1:6379> zscore words dawang
"5"
127.0.0.1:6379> zrangebyscore words 3
(error) ERR wrong number of arguments for 'zrangebyscore' command
127.0.0.1:6379> zrangebyscore words 2 4
1) "xiaohong"
2) "xiaozhao"
3) "xiaoliu"
127.0.0.1:6379>

```

添加有序集合

查看集合

获取xiaozhao的分数score

获取范围

通过zrange命令进行正向排序，通过zrevrange命令进行方向排序。

Redis的有序集合(sorted set)命令（标黄的要熟悉）

序号	Redis 有序集合(sorted set)命令	描述
1	ZADD key score1 member1 [score2 member2]	向有序集合添加一个或多个成员，或者更新已存在成员的分数
2	ZCARD key	获取有序集合的成员数
3	ZCOUNT key min max	计算在有序集合中指定区间分数的成员数
4	ZINCRBY key increment member	有序集合中对指定成员的分数加上增量 increment
5	ZINTERSTORE destination numkeys key [key ...]	计算给定的一个或多个有序集的交集并将结果集存储在新的有序集合 key 中
6	ZLEXCOUNT key min max	在有序集合中计算指定字典区间内成员数量
7	ZRANGE key start stop [WITHSCORES]	通过索引区间返回有序集合或指定区间内的成员
8	ZRANGEBYLEX key min max [LIMIT offset count]	通过字典区间返回有序集合的成员
9	ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT]	通过分数返回有序集合指定区间内的成员
10	ZRANK key member	返回有序集合中指定成员的索引
11	ZREM key member [member ...]	移除有序集合中的一个或多个成员
12	ZREMRANGEBYLEX key min max	移除有序集合中给定的字典区间的所有成员
13	ZREMRANGEBYRANK key start stop	移除有序集合中给定的排名区间的所有成员
14	ZREMRANGEBYSCORE key min max	移除有序集合中给定的分数区间的所有成员
15	ZREVRANGE key start stop [WITHSCORES]	返回有序集中指定区间内的成员，通过索引，分数从高到底
16	ZREVRANGEBYSCORE key max min [WITHSCORES]	返回有序集中指定分数区间内的成员，分数从高到低排序
17	ZREVRANK key member	返回有序集合中指定成员的排名，有序集成员按分数值递减(从大到小)排序
18	ZSCORE key member	返回有序集中，成员的分数值
19	ZUNIONSTORE destination numkeys key [key ...]	计算给定的一个或多个有序集的并集，并存储在新的 key 中
20	ZSCAN key cursor [MATCH pattern] [COUNT count]	迭代有序集合中的元素（包括元素成员和元素分值）

到此，Redis的五种数据类型介绍完毕，大家可以自行练习一下，多练几次就熟悉了。

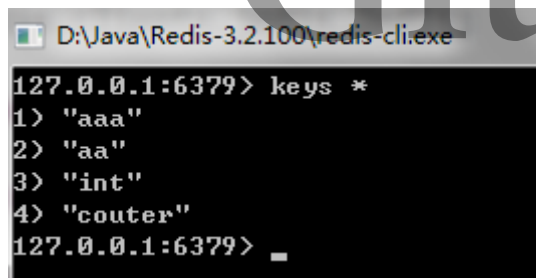
一般使用String类型的较多，多用于缓存服务器。

五、Java代码如何操作Redis

Java代码操作Redis，特别简单。首先你需要下载驱动包，下载 jedis.jar。

下载地址：<https://mvnrepository.com/artifact/redis.clients/jedis>

在你的classpath 中包含该驱动包。连接java代码操作前的本地redis如下：



```

D:\Java\Redis-3.2.100\redis-cli.exe
127.0.0.1:6379> keys *
1) "aaa"
2) "aa"
3) "int"
4) "couter"
127.0.0.1:6379> _

```

用java代码连接本地redis，并设置一个key为“redis”的键，其值为“I love redis!”。Java代码如下：


```

package redis;

import redis.clients.jedis.Jedis;

public class TestRedis {

    private static final Jedis jedis = new Jedis("127.0.0.1", 6379);

    public static void main(String[] args) {
        if (jedis != null) {
            String connection = jedis.ping();
            System.out.println(connection);
            if ("pong".equals(connection)) {
                System.out.println("Redis连接成功！");
                // 设置字符串数据
                jedis.set("redis", "I love redis!");
                // 获取存储的数据并输出
                System.out.println("键\"redis\"存储的字符串为：" +
jedis.get("redis"));
            } else {
                System.out.println("Redis连接失败！");
            }
        }
    }
}

```

GitChat

代码运行结果如下：

```

PONG
Redis连接成功！
键"redis"存储的字符串为：I love redis!

```

通过Java代码操作后，观察Redis的数据，如下：

```

127.0.0.1:6379> keys *
1) "aaa"
2) "aa"
3) "int"
4) "couter"
127.0.0.1:6379> keys *
1) "aaa"
2) "aa"
3) "int"
4) "couter"
5) "redis"
127.0.0.1:6379> get redis
"I love redis!"
127.0.0.1:6379>

```

获取到了java代码写入的redis的值

六、Redis集群简单介绍

Redis有3种集群策略

1. 主从

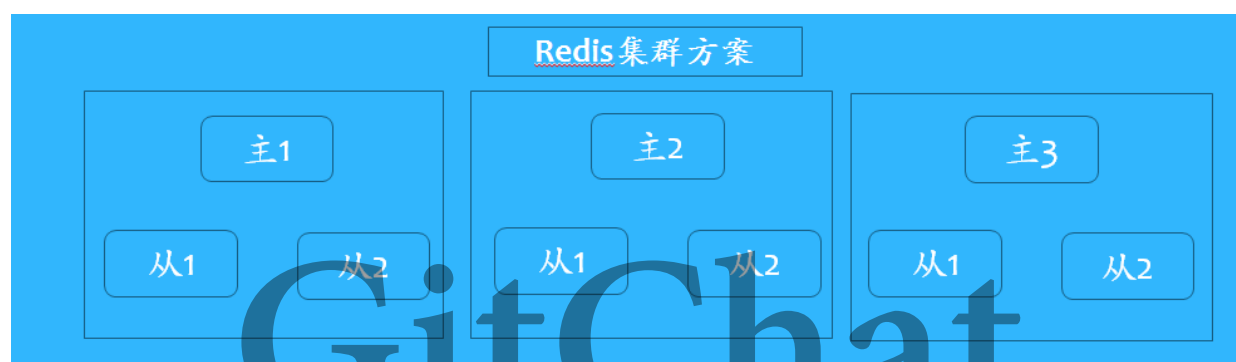
1台机器可写作为主，另外2台可读，作为从，类似于MySQL的主从复制，不过Redis没有BINLOG机制。

2. 哨兵

增加一台机器作为哨兵，监控3台主从机器，当主节点挂机的时候，机器内部进行选举，从集群中从节点里指定一台机器升级为主节点，从而实现高可用。当主节点恢复的时候，加入到从节点中继续提供服务。

3. 集群

Redis3.0以后增加了集群的概念，可以实现多主多从结构，实现真正的高可用。



到此，Redis的Chat就结束了，文章有点长，希望对你有用，咱们下次再见，谢谢。