# Redis介绍

redis即远程字典服务，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。

## 特点：

1开源

2支持多种数据结构

3基于键值的存储服务系统

4高性能，功能丰富

5 速度快

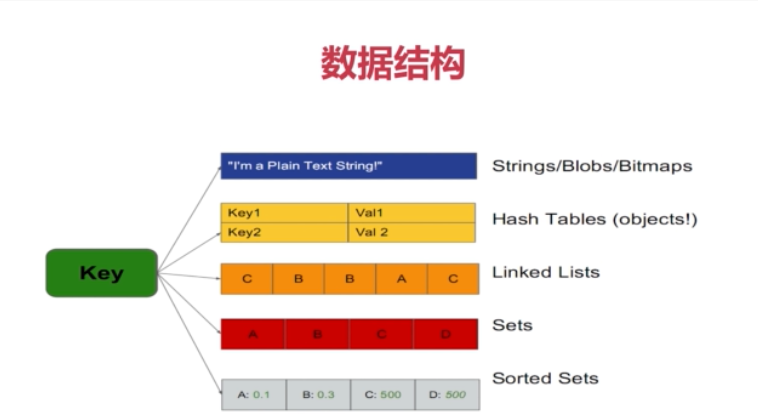
6持久化

7支持多种编程语言

8 主从复制

9 高可用分布式

## redis的数据结构



## redis典型应用场景

1 缓存系统

2 计数器

3 消息队列系统

4 排行榜

5 社交网络

6 实时系统

## redis可执行文件说明

redis-server 启动redis服务器

redis-cli 连接redis命令行客户端

redis-benchmark redis性能测试

redis-check-aof aof文件修复工具

redis-check-dump RDP文件检查工具

redis-sentinel sentine服务器

## redis启动方法

1最简启动

redis-server

2动态参数启动

redis-server --port 6380

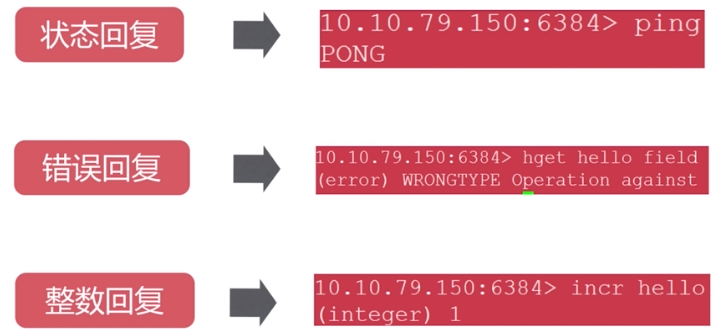
3配置文件启动

redis-server redis-serverPath

## redis连接方法

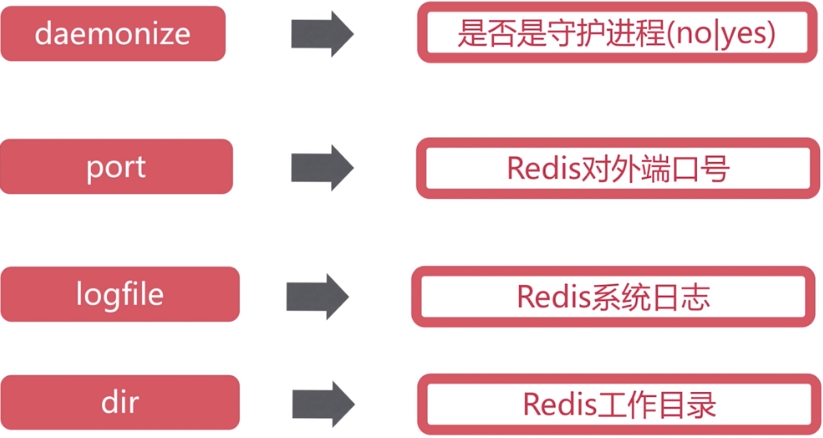
redis-cli -h IP -p port

## redis客户端返回值





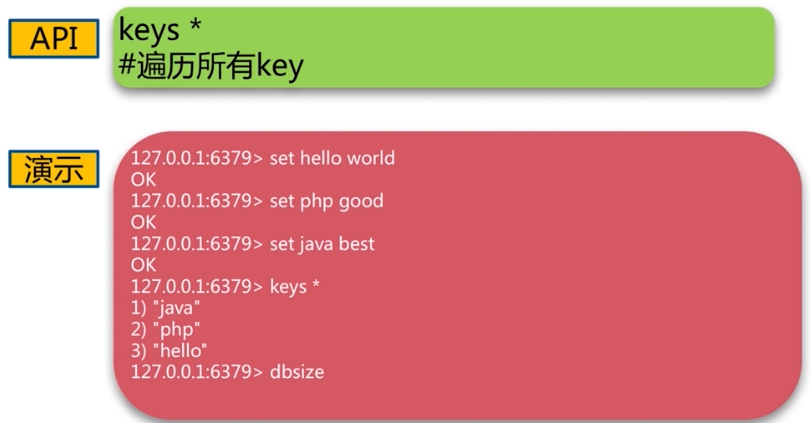
## redis常用配置



# RedisAPi

# 通用命令

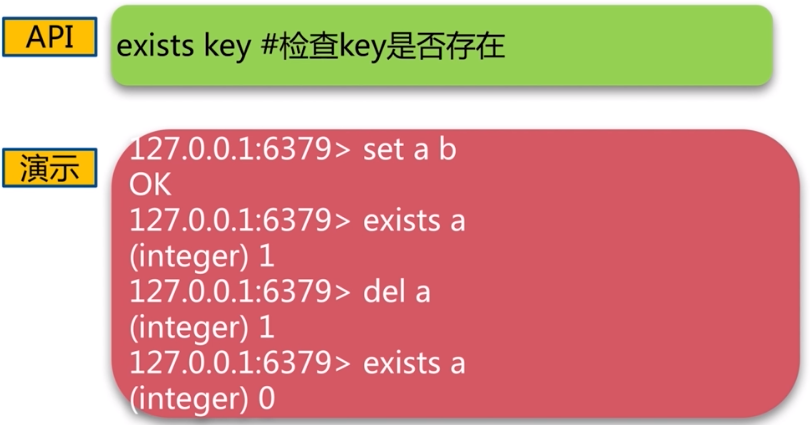
1 keys 计算数据中的所有键



2 dbsize 计算数据库大小



3 exists key 判断key是否存在



4 del key 删除key 支持删除多个



5 expire key seconds 设置key的失效时间





6 type key 查询key数据类型

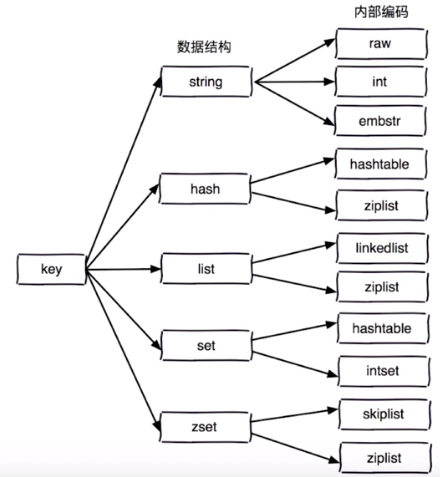


## 时间复杂度

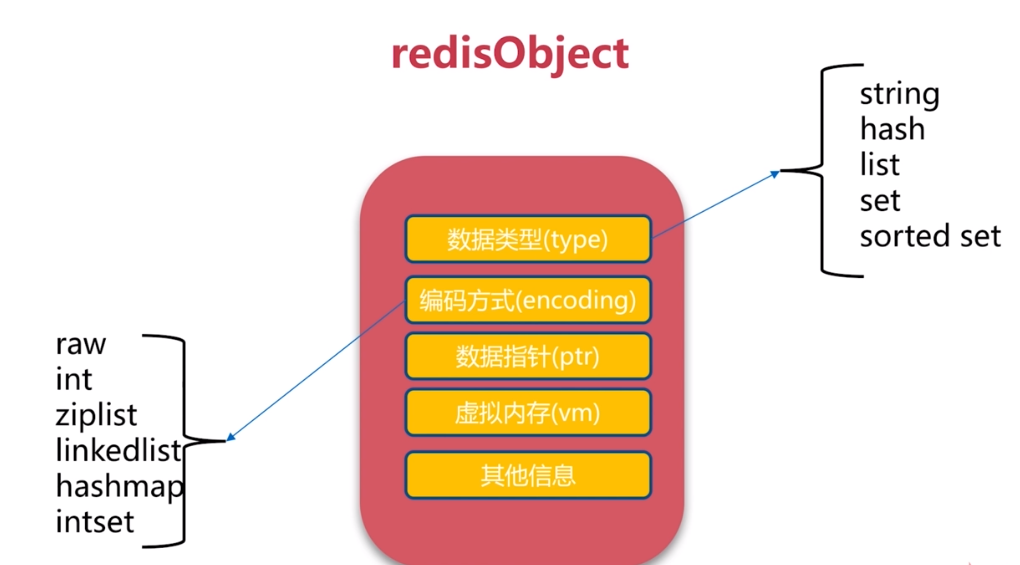
除了O(n)级别的命令其他命令都可以在任何环境中使用



## 数据结构和内部编码

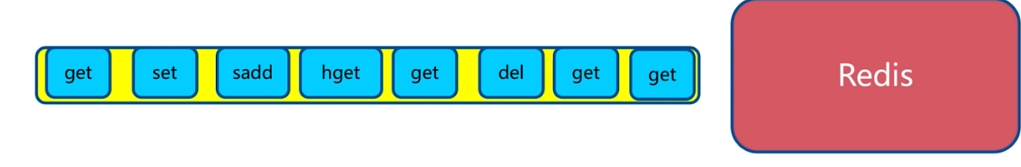


redisObject



## 单线程

redis是单线程架构每条命令都是在上一个命令执行之后执行的，串行的执行命令



### 单线程快的原因

1、完全基于内存，绝大部分请求是纯粹的内存操作，非常快速。数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)；

2、数据结构简单，对数据操作也简单，Redis中的数据结构是专门进行设计的；

3、采用单线程，避免了不必要的上下文切换和竞争条件，也不存在多进程或者多线程导致的切换而消耗 CPU，不用去考虑各种锁的问题，不存在加锁释放锁操作，没有因为可能出现死锁而导致的性能消耗；

4、使用多路I/O复用模型，非阻塞IO；

5、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移动和请求；

以上几点都比较好理解，下边我们针对多路 I/O 复用模型进行简单的探讨：

（1）多路 I/O 复用模型

多路I/O复用模型是利用 select、poll、epoll 可以同时监察多个流的 I/O 事件的能力，在空闲的时候，会把当前线程阻塞掉，当有一个或多个流有 I/O 事件时，就从阻塞态中唤醒，于是程序就会轮询一遍所有的流（epoll 是只轮询那些真正发出了事件的流），并且只依次顺序的处理就绪的流，这种做法就避免了大量的无用操作。

这里“多路”指的是多个网络连接，“复用”指的是复用同一个线程。采用多路 I/O 复用技术可以让单个线程高效的处理多个连接请求（尽量减少网络 IO 的时间消耗），且 Redis 在内存中操作数据的速度非常快，也就是说内存内的操作不会成为影响Redis性能的瓶颈，主要由以上几点造就了 Redis 具有很高的吞吐量。

### 单线程执行命令的注意事项：

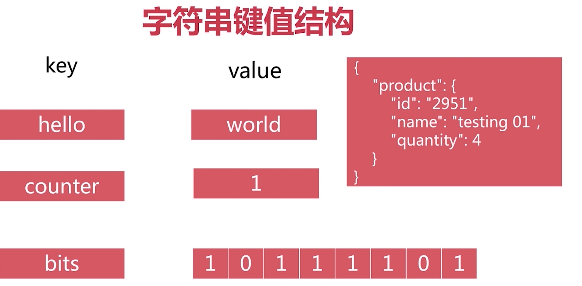
1一次之运行一条命令

2拒绝长（慢）命令

# 数据结构

## 字符串

redis在存储字符串类型的数据时会直接存储字符串或者存储数字类型的时候会存为int类型转换，或者存为二进制数据，字符串的限制不能大于512MB不过在平常使用中最好不要超过100k



### 使用场景

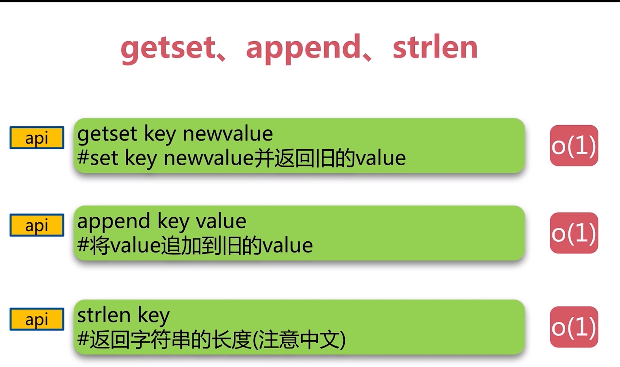














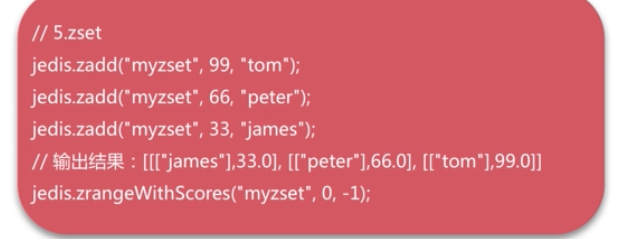
# redis在java中的使用

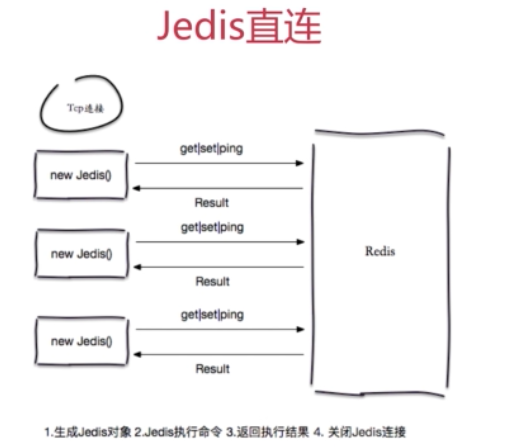
## 连接代码

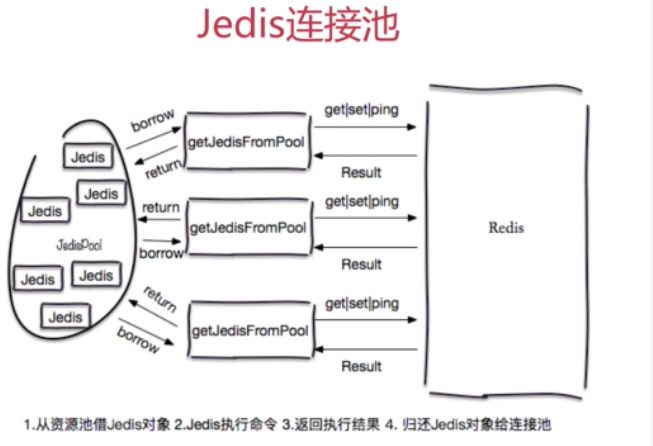
Jedis jedis = **new** Jedis(**"192.168.11.129"**,6379);  
jedis.auth(**"Redis!123"**);  
jedis.set(**"db"**,**"size"**);  
System.***out***.println(jedis.get(**"hello"**));











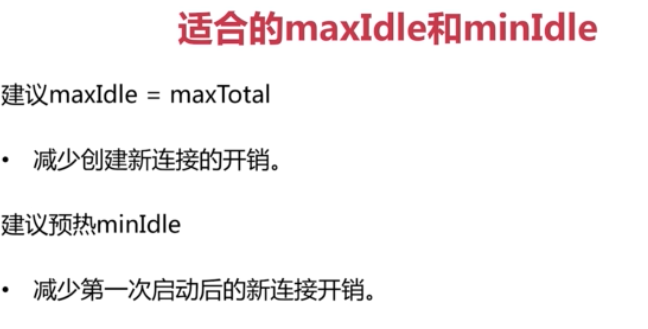


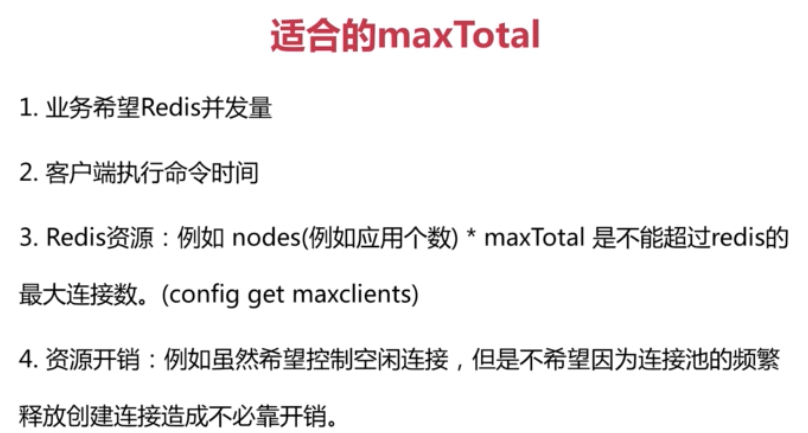
### jedispool连接池

GenericObjectPoolConfig poolConfig = **new** GenericObjectPoolConfig();  
JedisPool jedisPool = **new** JedisPool(poolConfig,**"192.168.11.129"**,6379);  
Jedis jedis = **null**;  
**try** {  
 jedis = jedisPool.getResource();  
 jedis.auth(**"Redis!123"**);  
 jedis.set(**"he"**,**"dd"**);  
} **catch** (Exception e) {  
 e.printStackTrace();  
} **finally** {  
 **if**(**null**!=jedis){  
 System.***out***.println(jedis.get(**"he"**));  
 jedis.close();  
 }  
}



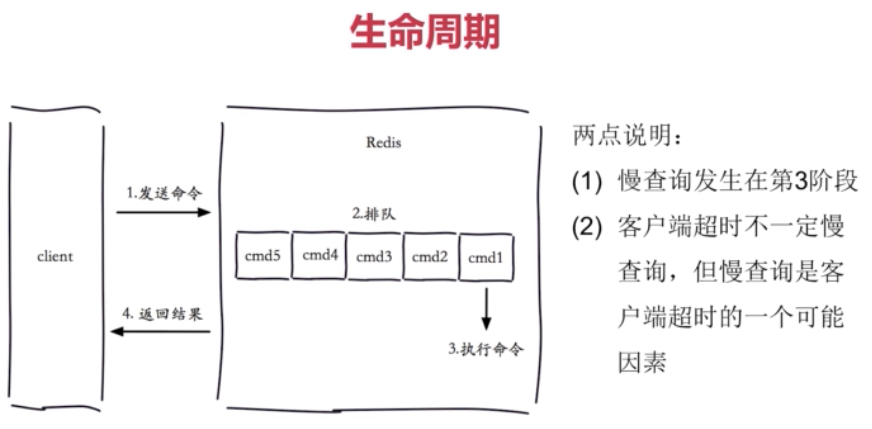


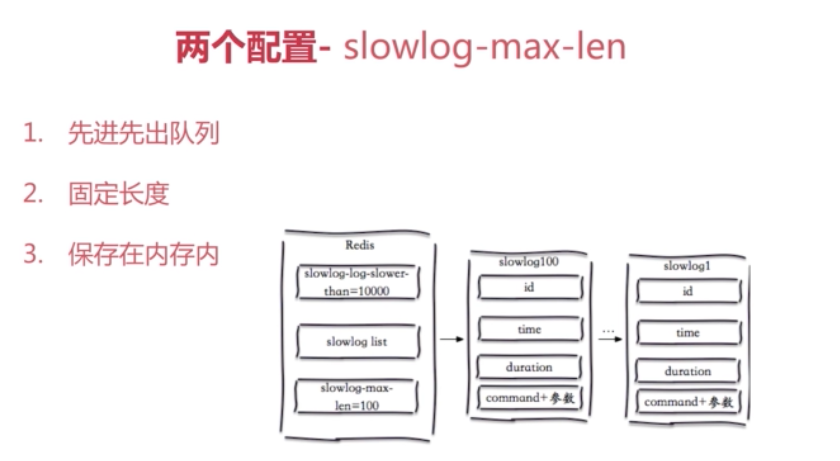




# 慢查询

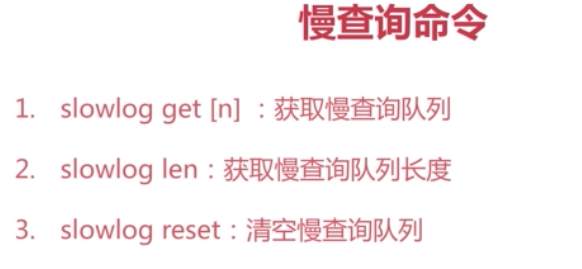
redis的生命周期











运维经验

slowlog-max-len 不要设置过小，通常设置1000左右

slowlog-log-slower-than 不要设置过大，默认10ms，通常设置1ms

理解命令的生命周期

定期持久化慢查询

# pipeline流水线

