redis梳理

# redis的数据类型

string ,list, set,hash,zset

https://mp.weixin.qq.com/s/V81hP9QI\_boMjnpryxRDhA

# redis的内部结构:

# redis的应用场景

1.缓存热点数据,

2.计数器:统计点击数,由于单线程,可以避免并发问题,毫秒级性能

持久化

3.队列:相当于消息系统,数据一致性要求高的用mq

redis添加数据到对列是返回元素在对列的第几位,判断用户是第几个访问这种业务.

4.位操作(大数据处理)

用于数据量上亿的场景,例如几亿用户系统的签到,去重登陆次数统计,某用户是否在线状态.

腾讯10亿用户,需要几毫秒查询某个用户是否在线,这里用到的是位操作,setbit,getbet,bitcount

原理:redis构建一个足够长的数组,每个数组元祖只能是0和1两个值,这个数组的下标,表示用户id,数组通过下标和元素值该构建记忆系统.

5.分布式锁和单线程机制

验证前端的重复请求,通过redis类进行过滤,将ip 参数 接口等hash作为可以存储redis(幂等性请求),设置多长时间有效,下次请求时,现在redis中检索有没有key,进而验证是不是有效时间的重复提交.

秒杀系统:基于redis是单线程的,防止出现数据库爆破.

全局增量id生成,类似于秒杀.

6.最新列表

在总数量很大的情况下,尽量不要limit10,尝试redis的lpush命令构建list,内存清掉了,用mysql查询并初始化list到redis中.

7.排行榜

谁得分高,排名往上,zadd

# redis持久化数据

服务器重启后重你以前的数据,防止系统出现故障将数据备份到一个远程的位置.

对需要大量计算而得到的数据,放置在redis服务器,进行持久化,

存储机制:

快照(rdb):俗称的备份,定期对数据进行备份,将redis服务器中的数据持久化到硬盘.

只追加文件(aof):会在写命令时,将执行的命令复制到硬盘中,后期恢复时,只需要重新执行这个命令就可以.

快照:60秒有1000次写入的时候执行快照的创建

创建快照失败时是否任然继续写命令.

是否对快照文件进行压缩,

快照文件.rbd 的命名

快照文件保存的位置

aof持久化配置"

是否开启aof持久化

多久执行一次将写入内容同步到硬盘上

对aof进行压缩时是否执行同步操作

多久执行一次aof压缩

多大执行一次aof压缩

aof文件保存的位置

可以单独使用,也可以同时使用

快照持久化:只适合于数据不经常修改或者丢失部分数据影响不大的场景.

创建快照方式:

# Redis 集群方案与实现

官方redis集群推荐方案:Redis-Cluster

现有问题:高可用问题保证redis的持续高可用问题

容量问题,单实例redis实例内存无法无限扩充

并发问题:redis号称单实例10万并发,但是也是有尽头的.

优势:去中心化集群最大可增加1000节点,性能岁节点增加线性扩展.

后续可自行增加或摘除节点,移动分槽等.简单易上手.

名词:master:主节点

slave:分节点

slot:槽一共有16384个槽,分布在集群的所有主节点中

6739位客户端通讯端口

16379端口为集群总线端口

集群内部划分为16384个数据槽,分布在三个主redis中 ,

从redis没有分槽,不会参与集群投票,也不会加快读取数据,仅仅作为主机的备份.

平分16384槽,不会有重复数据

# redis为什么是单线程的

完全基于内存,cpu不是Redis的瓶颈,redis的瓶颈是机器内存的大小,或者网络带宽.

单进程多线程:MySql,Memcached,Oracle(Windows版)

多进程:Oracle(Linux版)

memcached:单进程多线程

redis为什么这么快:

a.完全基于内存,hashmap的查找和操作的时间复杂度低.

b.数据结构简单,操作也简单

c.采用单线程,避免了上下文切换和竞争条件,也不存在多进程和多线程的切换而切换cpu,不考虑锁的问题,不会出现死锁导致性能消耗.

d.使用多路I/o复用模型,非阻塞IO

e.使用底层不同,他们之间底层实现方式及客户端之间的应用协议不一样,redis直接自己构建了vm机制,因为一般的系统调用系统函数导入时候,会浪费一定时间区移动和请求.

# redis常见的回收策略

lru()最近最少使用,

Volatile-lru:从已设置过期时间的数据集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中任意选择数据淘汰

allkeys-lru:从数据集中挑选最近最少使用的数据淘汰

allkeys-random:从数据集中任意选择数据淘汰

no-enviction:禁止驱逐数据