说说你知道的redis规范?

1. key的设计

可管理性和可读性:

以业务名(数据库名)为前缀, 防止key冲突 用冒号隔开

如: 数据库名:表名:id local:user:12

简洁性:

保证语义的前提下, 尽量缩短key, key很多时也会占用很多的内存

如: user:{uid}:friend:message:{mid} 简化为 u:{uid}:f:msg:{mid}

不要包含特殊字符:

如: 包括空格、换行符、单双引号、转移符等

1. value的设计

拒绝bigkey (防止网卡流量、慢查询):

string最好控制在10kb

hash、list、set、zest元素不要超过5000

如果删除bigkey不要直接del, 最好用scan进行渐进式的删除, 同时也要防止bigkey自动过期, 自动过期会触发del操作, 造成阻塞

选择合适的数据类型:

反例: set user:1:name 张三 set user:1:age 18

正例: hmset user:1 name 张三 age 18

控制key的生命周期:

使用expire设置过期时间 (随机打散过期时间, 防止缓存雪崩)

不过期的数据使用idletime

1. 命令的使用

时间复杂度为O(n)的命令需要关注n的数量:

例如: hgetall lrange smembers zrange sinter 这些命令时间复杂度都是O(n), 在使用的时

候一定要指定n的大小, 否则也有可能造成缓存雪崩, 如果需要遍历, 可以使用 scan 命令

遍历

禁用命令:

生产环境下禁用 FLUSHALL FLUSHDB CONFIG KEYS, 这些命令需要通过配置文件禁用掉, 如: rename-command KESY ""

合理使用select :

redis多库会比较弱, 使用分区很多客户端支持较差, redis本身就是单线程的, 同时使用多

个库还是单线程的, 会干扰性能

使用批量处理会提高性能:

如 mget, mset, pipeline 批量插入会提高性能, 但是一次最多不要超过500个, 和数据的 实际大小有关

mget 和 mset是原生操作, 所以他是原子性的, pipeline是非原生的命令, 不是原子性的

pipeline可以打包不同的命令, 原生命令做不到, pipeline需要客户端和服务端的同时支持

不要使用过多的事务:

redis的事务比较弱, 不支持回滚操作, 在集群中官方要求一次事务操作的key必须在一个 槽上

集群上使用LUA脚本有特殊要求:

所有key都应该由 KEYS 数组来传递, redis.call/pcall 里面调用的redis命令, key的位置,

必须是KEYS array, 否则直接返回error

所有的key, 必须在同一个slot上, 否则直接返回error

生产环境下不要长时间使用monitor命令:

monitor 会开辟一个进程, 实时打印出redis的命令, 所以会占用资源, 除了排查故障时, 其他时候建议不要使用

1. 客户端的使用

避免多个应用使用一个redis实例:

拆分不同的业务, 公共数据服务化处理

使用连接池:

可以有效的控制连接, 同时提高效率

熔断功能:

高并发情况下客户端添加熔断功能, 可以使用netflix hystrix

合理的加密:

设置合理的密码, 有必要时需要设置SSL (阿里)

淘汰策略:

根据自身业务, 配置好maxmemory, 设置好过期时间, 根据自己业务的需求, 设置淘汰策

略, 默认策略是volatile-lru, 但是可能出现OOM错误 (只读不可写)