## Redis主要数据结构

### SDS(Simle dynamic string)-简单动态字符串

Struct sdshdr{

Int len; // 记录buf数组中已使用的字节数量

Int free; // 记录free buf数组中未使用的字节数量

Char buf[]; // 字节数组， 用于保存字符串

}

重点回顾：

1. 大多数情况下， redis使用sds作为字符串表示（如大多数的key,val都是sds）， c字符串常量指挥用在无须对字符串值修改的地方， 如打印系统日志
2. O(1)获取字符串长度(使用len属性)
3. 减少修改字符串长度时所需的内存重分配次数（buf数组中可能有未使用的空间， 惰性空间释放）
4. 杜绝缓存区溢出(结合len和free，在concat前判断)

### List-链表

List(列表键)的底层实现，发布订阅、慢查询、客户端状态等用了链表。 作为常用数据结构， 不再详述。Redis实现的是带头尾指针的双向链表

### Hash-字典

常用数据结构，不详述实现，参考hashmap

渐进式rehash: 避免一次性移动太多键值对导致对服务器的性能影响，分多次、渐进式地迁移。迁移期间字典持有两个hash表， 每次对字典进行操作时，除了执行指定的操作以外，还会顺带将该索引位置的所有键值对迁移到新的hash表上。

### Skiplist-跳跃表

参考原始论文和ppt。

定义：TODO， 核心思想， 搜索时跳过一部分，类似二分查找

延伸提问：

1. 为什么使用跳跃表而不是红黑树？

从原始论文的摘要就可以得到答案： the algorithm for **insertion** and **deletion** in skip lists are much **simpler** and **significantly faster** than equivalent algorithms for balanced trees

### 压缩列表

压缩列表是列表键与hash键的底层实现**之一。**

**当一个列表键只包含少量的列表项，并且每个列表项要么就是小整数值， 要么就是长度比较短的字符串，那么redis就会使用压缩列表来做列表键的底层实现。**

**当一个hash键只包含少量的键值对， 并且每个键值对要么就是小整数值，要么就是长度比较短的字符串， 那么redis 就会使用压缩列表来做hash键的底层实现。**

**查看key的结构： object encoding key or debug object key**

**目的： 节约内存**

## 键的生存时间

EXPIRE

PEXPIRE

EXPIREAT

PEXPIREAT

SETEX

### 实现原理

RedisDb结构的**expires字典保存了数据库中所有键的过期时间**，称这个字典为过期字典。

Key为一个指针， 指向键空间的某个键对象， value是一个longlong类型的整数，保存了键所指向的数据库键的过期时间—一个毫秒精度的UNIX时间戳。

### Redis的过期键删除策略

Redis服务器使用的是惰性删除和定期删除两种策略。

## 持久化

### RDB持久化

也就是snapshot。两个redis命令可以用于生成RDB文件

SAVE：阻塞redis服务器进程，知道RDB文件创建完毕位置， 在服务器进程阻塞期间， 服务器不能处理任何命令请求。

BGSAVE：派生一个子进程负责创建RDB文件， 服务器父进程继续处理请求命令。

可以让服务器每个一段时间或一定条件后自动执行一次BGSAVE命令。

save 900 1 : 900秒内，对数据库进行了至少1次更改则执行BGSAVE

Save 300 10 : 300秒内， 对数据库进行了至少10次更改则执行BGSAVE

多个配置条件只要满足一个即会触发BGSAVE

### AOF持久化（append only file）

三个步骤：

1. 命令追加
2. 文件写入
3. 文件同步
4. **命令追加**：服务器在执行完一个写命令后，会以协议格式将被执行的命令追加到服务器状态的**aof\_buf缓冲区**末尾
5. **文件写入与同步**：在服务器每次结束一个事件循环之前，都会调用flushAppendOnlyFile函数，考虑是否需要将aof\_buf缓存区中的内容写入保存到**AOF文件**中

通过服务器配置的appendfsync选项的值来决定flushAppendOnlyFile函数的行为：

Always:将aof\_buf缓冲区的所有内容写入AOF文件并进行同步

Everysec::将aof\_buf缓冲区的所有内容写入AOF文件， 如果上次同步AOF文件的时间距离现在超过一秒钟，那么再次对AOF文件进行同步， 这个同步操作是由一个线程专门负责执行的。

no:将aof\_buf缓冲区的所有内容写入AOF文件， 但并不对AOF文件进行同步，何时同步由操作系统来决定。

aof文件类似与relicated state machine的log数组，写入aof文件则认为该命令是commited

### 慢查询

## 提问

1. redis bigkey：指key的value较大。 内存使用不均， 慢->阻塞。
2. redis list中元素过多有什么影响？？？ 为什么我觉得没啥？？？ （怕执行LRANGE等类似操作造成同bigkey的影响?）
3. redis的过期策略：redis expired key的删除 ： https://www.jianshu.com/p/9352d20fb2e0
4. redis的淘汰策略： LRU， LFU
5. redis 扩容， 主从模式扩容(内存不可扩展、吞吐量不可扩展)、集群模式扩容
6. 缓存雪崩、缓存穿透、缓存击穿

--https://blog.csdn.net/zeb\_perfect/article/details/54135506

1. sentinel的leader election就是raft的leader election的实现。raft协议
2. skiplist
3. 压缩列表
4. redis rdb, aof。 | aof与wal，为啥使用aof而不是wal
5. redis实现锁 ， redLock
6. 如何保障缓存与数据库的一致性。 | 对于写操作， 先淘汰缓存，再更新数据库
7. redis 集群通过分片来提供复制和故障转移功能。(SLOT)
8. slot的hash函数为CRC16(key) mod 16484, 而CRC16函数的取值范围为65535， 为什么slot的数量是16384而不是65535？| 心跳 ，bitmap ， 对于少于1000 node的集群来说， 16K(16384)已经足够了，SLOT约多，bitmap就需要越大，那么心跳信息也就会越大

参考文章：https://www.programmersought.com/article/33604426038/

redis作者的回答：<https://github.com/antirez/redis/issues/2576>

1. redis lua如何保证事务执行？？（这个选学吧， 没找到过， 应该跟持久化有关）
2. redis copyOnWrite
3. \*redis gossip协议是用来干什么的？ 优先级低