Redis 源码解析

- 主要特点 (29)
- 持久化 (51)
- 事务 (57)
- 分布式 (54)
- 使用场景 (6)
- 数据结构与对象 (78)
- 单机数据库实现
  - 多数据库结构
    - Redis服务器中的所有数据库，保存在 redis.h / redisServer 结构的 db 数组中
    - db 数组的每一项都是 redis.h / redisDb 结构
    - redisSerfer / dbnum 由服务器配置的 database 选项决定，默认 16
  - 多数据库切换
    - 使用 select index 命令可以切换数据库 [0...15]
    - Redis内部有一个客户端状态(redisClient) 结构的 db 属性，记录了当前客户端的目标数据库号
    - 这个属性是一个指向 RedisDb 结构的指针
    - select 的实现原理就是通过修改 RedisClient.db 指针来切换数据库的
  - 数据库键空间
    - RedisDb 结构的 dict 属性保存着数据库中的所有键值对
    - 所有对数据库的操作，其实都是对这个 dict 的操作
  - 数据库键过期字典
    - RedisDb 结构的 expires 字典保存了数据中所有键的过期时间
    - expires dict 的键是一个指针，指向键空间的某个键对象
    - expires dict 的值是一个long long类型的整数，保存了该键的过期时间，一个毫秒精度的UNIX时间戳
  - 命令请求的执行过程
    - 1. 当用户在客户端键入一个命令请求时，客户端将命令请求转换成协议格式，通过连接到服务器的socket，发送给服务器
    - 2. 当客户端与服务器之间的连接套接字因为客户端的写入而变得可读时，服务器将调用命令请求处理器来处理
    - 3. 读取socket中协议格式的命令请求，并将其保存到客户端状态的输入缓冲区里面（querybuf）
    - 4. 对输入缓冲区的命令请求进行分析，将命令参数与参数数量保存到客户端状态的 argv属性和 argc属性
    - 5. 调用命令执行器，执行指令
    - 6. 命令执行器根据redisClient的 argv[0] 参数，在命令表(command table)中查找参数所指定的命令，将找到的命令保存到redisClient的 cmd 属性
      - 命令表是一个字典
      - 键是命令名称(set, get, del)
      - 值是 redisCommand 结构，记录了命令的实现信息
    - 7. 命令执行器在真正执行之前，还需要做一些预备操作，从而确保命令可以正确、顺利地执行
      - 检查 redisClient.cmd 是否指向 NULL，如果是返回错误
      - 检查 cmd 属性所需的参数是否合法
      - 检查客户端是否已经通过了身份验证
      - 如果服务器开启了 maxmemory 功能，那么在执行命令之前，先检查服务器的内存占用情况，并在有需要的时候进行内存回收
      - 如果服务器上一次执行 BGSAVE 命令时出错，而且即将要执行的是个写命令，服务器将拒绝执行这个命令，返回错误
      - ....
    - 8. 执行命令，并将产生的回复信息，保存到客户端输出缓冲区 redisClient.buf 属性中
      - 并将客户端socket关联到命令回复处理器
      - 当客户端socket变为可写状态时，服务器就会执行命令回复处理器，将保存在客户端输出缓冲区中的命令回复发送给客户端
    - 9. 后续工作
      - 如果服务器开启了慢查询日志功能，那么慢查询日志模块会检查是否需要为刚刚执行完的命令请求添加一条新的慢查询日志
      - 根据刚刚执行命令所耗费的时长，更新被执行命令的redisCommand结构的milliseconds 属性，并将命令的 redisCommand.calls 的值加一
      - 如果服务器开启了AOF持久化，那么AOF持久化模块会将刚刚执行的命令请求写入到AOF缓冲区里面
      - 如果有其他服务器正在复制当前服务器，那么服务器会将刚刚执行的命令传播给所有从服务器
    - 10. 执行完成，服务器继续从文件事件处理器中取出并处理下一个命令请求
  - serverCron函数
    - 默认100ms执行一次，该函数负责管理服务器的资源，维护状态
    - Redis服务器需要频繁获取系统的当前时间，而获取系统的当前时间需要执行系统调用，为了减少系统调用的执行次数，服务器状态结构(redisServer)中的unixtime属性和mstime属性被用作当前时间的缓存
    - redisServer.lruclock 属性保存了服务器的LRU时钟，用于计算键的空转时长，也是时间缓存的一种，默认每10s更新一次
      - 每个Redis对象都有 lru 属性，保存了对象最后一次被访问的时间
      - 键对应的值对象空转时长 = redisServer.lruclock - redisObject.lru
      - $ info server
    - trackOperationsPerSecond 函数
      - serverCron 函数中的 trackOperationsPerSecond 函数会以每100ms一次的频率执行
      - 功能是以抽样计算的方式，估算并记录服务器在最近一秒处理的命令请求数量
      - $ info status　　instantaneous_ops_per_sec: 6
    - redisServer.stat_peak_memory 属性
      - 记录了服务器的内存峰值大小
      - $ info memory
    - clientsCron 函数
      - serverCron 函数每次执行都会调用 clientsCron 函数
      - 检查客户端连接是否已经超时
      - 检查客户端输入缓冲区大小，如果超过了一定长度，就会释放当前输入缓冲区，并重新创建一个默认大小的输入缓冲区，防止客户端的输入缓冲区占用过多内存
    - databaseCron 函数　　数据库检查，删除过期键
    - 执行之前被推迟的 BGREWRITEAOF 命令
    - 将AOF缓冲区中的内容写入AOF文件
  - 初始化服务器过程
    - 1. 初始化服务器状态结构
      - 创建一个 struct redisServer 类型的实例变量 server 作为服务器状态对象，设置各个属性默认值
      - 实现：redis.c / initServerConfig 函数
    - 2. 载入配置选项　　redis.conf
      - databases 32
      - rdbcompression no
    - 3. 初始化其他服务器数据结构
      - server.clients 链表
      - server.db 数组
      - server.pubsub_channels 字典
      - server.lua
      - server.slowlog 属性
      - 设置进程信号处理器
      - 创建共享对象
        - “OK”
        - 1-10000 字符串对象
      - 打开或创建 AOF 文件
      - .....
    - 4. 还原数据库状态　　载入RDB / AOF
    - 5. 执行事件循环　　开启时间循环主程序，等待请求
- 事件模型 (30)

XMind