内存

消耗

- 子进程消耗
  - 与父进程共享内存页，修改时使用"copy-on-write"技术
  - Transparent Huge Pages(THP)
    - copy-on-write 期间复制内存页从 4KB 变成 2MB
    - fork 子进程的速度变慢
    - 高并发下开启容易造成内存溢出，建议关闭
- 消耗统计
  - info_memory：查看内存使用情况命令
  - used_memory：Redis 分配器分配的内存
  - used_memory_rss：操作系统的角度显示 Redis 占用的内存
  - mem_fragmentation_ration：内存碎片率
    - 正常比率：1.03
    - >1：多出的内存是碎片
    - <1：Redis 把部分内存交换到硬盘中
- 消耗划分
  - userd_memory_rss
    - used_memory
      - Redis 进程自身内存
      - 数据内存
      - 缓冲内存
        - 客户端缓冲
          - 普通客户端：普通 TCP 连接（注意大量慢连接会使内存飙升）
          - 从客户端：为主从复制建立的连接（主从网络不好或从节点过多占用内存大）
          - 订阅客户端：pubsub 功能建立的连接（消息产生的速度远大于消费的速度造成输出缓冲区溢出）
        - 复制积压缓冲区：部分复制功能，默认 1M，调大可以有效避免全量复制（如 100M）
        - AOF 缓冲区：Redis 重写期间保存写入的命令，通常占用小
    - 内存碎片
      - 默认内存分配器 jemalloc
        - 小：8byte,16byte...
        - 大：4KB,8KB...
        - 巨大：4MB,8MB
      - 产生原因
        - 数据长短差异过大，频繁做更新操作，如 append,setrange
        - 数据长短差异大，大量过期键删除
      - 解决
        - 尽量数据对齐
        - 安全重启

设置

- 设置内存上限
  - 缓存场景，当超过 maxmemory 时触发删除策略释放内存
  - 避免超过物理内存大小
- 动态设置内存上限
  - config set maxmemory
  - 通过调小收缩内存会造成数据丢失和阻塞问题

回收机制

- 删除过期键
  - 定时删除
    - 10s 运行一次
    - 快慢模式回收
  - 惰性删除
    - 避免对每个键维护删除机制
    - 查询时过期的键删除，返回 null
    - 无法删除过期但不查询的键

溢出回收策略

- noeviction —— 默认，不删除键，拒绝写入
- volatile-lru —— 根据 LRU 算法删除设置了过期时间的键
- volatile-ttl —— 删除最近将要过期的键
- volatile-random —— 随机删除设置了过期时间的键
- allkeys-lru —— 根据 LRU 算法删除所有键
- allkeys-random —— 随机删除所有键
- 小结
  - volatile-xxx 策略只淘汰设置了过期时间的键
  - allkeys-xxx 策略淘汰所有的键
  - 用 Redis 做缓存，使用 allkeys-xxx 策略
  - 用 Redis 做数据库，使用 volatile-xxx 策略