持久化

RDB（Redis Database）

- 概念 — 将数据生成快照保存在硬盘中
- 原理 — 使用操作系统地多进程 Copy On Write 机制
- 触发方式
  - 自动触发 — "save m n"（m 秒内 n 次修改后触发）
  - 手动触发 — bgsave：fork 子进程异步进行快照操作
- 优缺点
  - 优点
    - 适用于备份，全量复制
    - 恢复数据比 AOF 快
  - 缺点
    - 需要 fork 子进程，无法实时持久化，易出现数据丢失
    - 新老版本 RDB 二进制文件格式不兼容

AOF（Append Only File）

- 概念 — 以日志的方式记录写命令，解决持久化的实时性
- 原理 — 基于 redis 通讯协议，将命令以纯文本的方式写入到文件中
- 同步方式
  - AOF_FSYNC_NO：不同步 — 命令写入 aof_buf 后调用操作系统的 write 同步到操作系统页缓存
  - AOF_FSYNC_ALWAYS：每次写都同步 — 命令写入 aof_buf 后调用操作系统的 fsync 同步到 AOF 文件
  - AOF_FSYNC_EVERYSEC：每秒同步（默认） — 命令写入 aof_buf 后调用 write，专门线程每秒执行 fsync
- 配置 — appendonly yes & appendfsync everysec
- 文件重写
  - 触发方式
    - 自动触发 — bgrewriteaof
    - 手动触发 — 根据 auto-aof-rewrite-min-size 和 auto-aof-rewrite-percentage 确定
  - 原理 — 开辟子进程遍历内存并转换成一系列指令，然后序列化到新的 AOF 文件，之后将期间发生的增量日志追加到新的 AOF 文件，完毕后替代旧的 AOF 文件
  - 重写文件减小
    - 超时数据不再写入
    - 旧文件无效命令不再写入
    - 合并多条命令
- 重启加载 — 判断是否开启且存在 AOF 文件，若不存在就加载 RDB 文件
- 优缺点
  - 优点
    - 适合做灾难性的误删除紧急恢复
    - 实时性高，保护数据不丢失
  - 缺点
    - 恢复数据比 RDB 慢
    - 支持的写 QPS 比 RDB 支持的写 QPS 低

混合持久化

- 优点 — 快速加载同时避免丢失过多的数据
- 缺点 — aof 里面的 rdb 部分可读性差
- 启用 — aof-use-rdb-preamble=yes

优化

- fork 操作优化
  - 使用物理机或高效支持 fork 操作的虚拟化技术
  - 控制 Redis 最大内存，建议 10G
  - 合理配置 Linux 内存分配策略
  - 降低 fork 操作频率
    - 放宽 AOF 触发时机
    - 避免不必要的全量复制
- 子进程优化
  - CPU — 不和 CPU 密集型应用一起部署
  - 内存 — 由于写时复制的存在，尽量保证同一时刻只有一个子进程在操作，不在大量写入时重写
  - 硬盘
    - 不和硬盘高负载的应用一起部署，如消息队列
    - 开启 no-appendfsync-on-rewrite
    - 分盘存储 AOF 文件
- 单机多 Redis 实例优化 — 轮训确保同一时刻只有一个实例做 AOF 重写

追加阻塞

- 原因 — 主线程需要比对上次同步 AOF 时间，小于 2s 返回，大于阻塞
- 后果 — everysec 配置最多可能丢失 2s 数据