Redis 持久化

版本：Redis 3.0.4

目录

简述

- 持久化：内存中的数据，写入磁盘
- 为什么要持久化?
    - 数据在内存中，一旦断电，数据丢失；
    - 数据持久化之后，断电重启，数据仍然存在；
- 两种策略：
    - RDB：全量
    - AOF：增量
- 两种触发方式：
    - 手动触发持久化
    - 自动持久化：设置触发条件

RDB 持久化

- RDB 持久化，是什么?
- 基本过程?
- 有哪些潜在问题?
- 配置举例

RDB 持久化

- RDB 持久化：rdb 文件，压缩的二进制文件
- 2 种方式：SAVE、BGSAVE
  - SAVE：阻塞 Redis 服务器进程
  - BGSAVE：启用子进程

- 创建 RDB 文件：
  - 手动：SAVE、BGSAVE
  - 自动：持久化策略（执行 BGSAVE ）
- 加载 RDB 文件
  - 没有专门的加载 RDB 文件的命令
  - 服务器启动时，检测到 RDB 文件存在，就加载
  - 加载 RDB 文件时，Redis 的服务进程阻塞

RDB 自动持久化

- 自动持久化：
  - 策略？满足下面场景：
    - 时间：2s 一次？2min 一次?
    - 写的次数：100 个key？1000 个key?
    - 权衡：效率 vs. 数据丢失风险
  - 基本策略：
    - 高频率写、短时间
    - 低频率写、长时间
  - 开启方式：
    - 服务器配置 save 选项，触发BGSAVE 定期执行
    - 满足任何一个 save 选项，都会触发 BGSAVE 执行
  - 举例：
    - save 60 100 ： 60s 内，100 次修改，就触发 BGSAVE

```
save 900 1

save 300 10

save 60 10000
```

900秒内，至少执行了1次写操作

300秒内，至少执行了10次写操作

60秒内，至少执行了10000次写操作。

- 疑问：
    - 如何开启\关闭 RDB? 实质，关闭 RDB 自动持久化。
        - save “” // 覆盖前文配置的 RDB 自动持久化策略
    - 如何配置 RDB 自动持久化策略?
        - save <seconds> <changes> // 指定时间内，修改的 key 的个数
- RDB 默认的自动持久化策略?
    - save 900 1
    - save 300 10
    - save 60 10000
- 如何查看一个服务器的自动持久化策略?
    - info : 查看配置文件；
    - 查看当前 redis 版本对应的默认配置

分析 RDB 文件

- SAVE 命令，触发生成 dump.db 文件

- od -cx <file.rdb>： 将文件以八进制输出

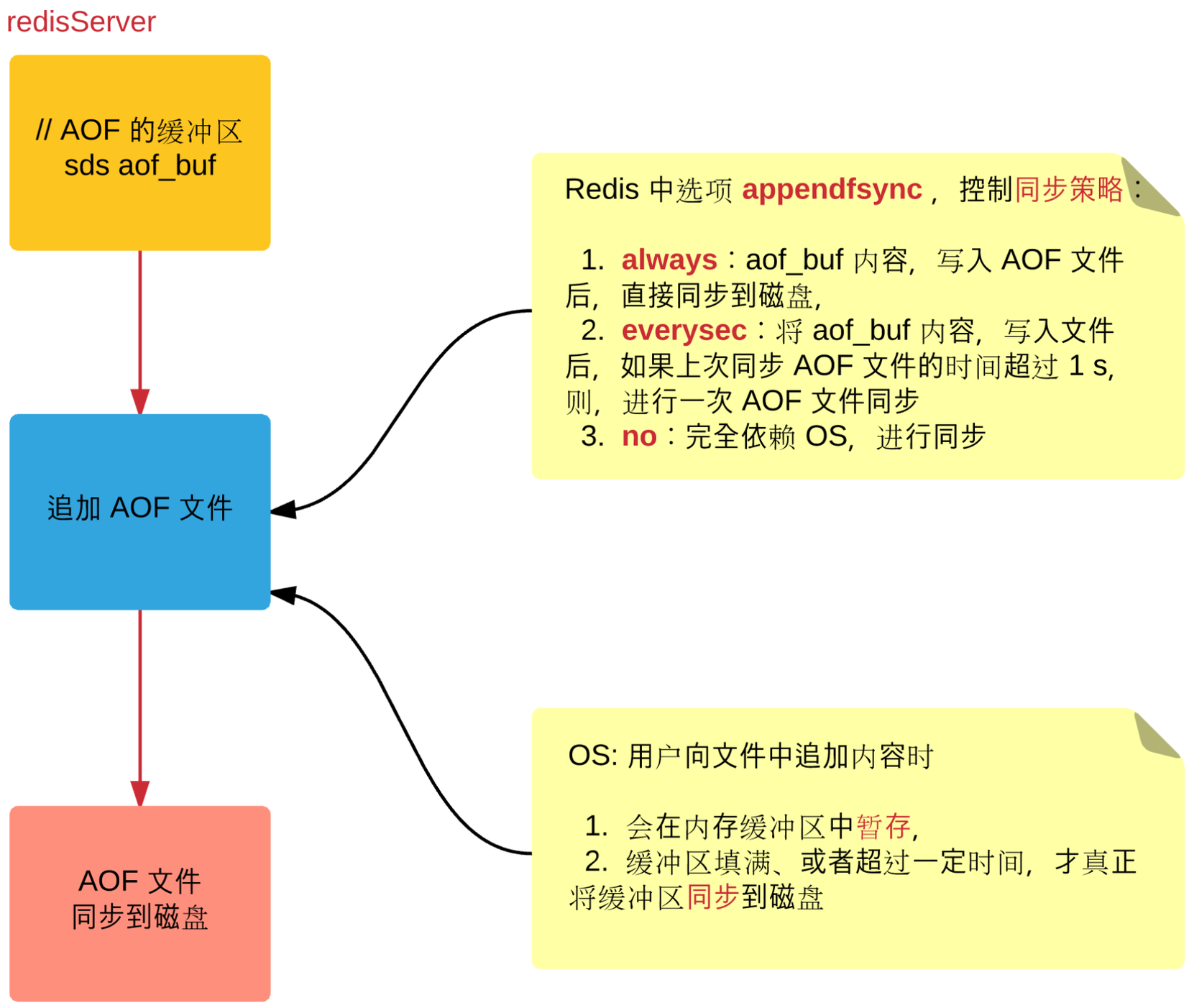
```
[redis@utopia 7003]$ od -cx dump.rdb
0000000   R   E   D   I   S   0   0   0   6 376  \0  \0 004   g   u   o
           4552    4944    3053    3030    fe36    0000    6704    6f75
0000020   1 005   v   a   l   u   e 377 026 255  \a   _ 253   7 362 344
           0531    6176    756c    ff65    ad16    5f07    37ab    e4f2
0000040
```

- redis-check-dump: 校验 RDB 文件

```
[redis@utopia 7003]$ redis-check-dump dump.rdb
==== Processed 3 valid opcodes (in 15 bytes) ==============================
CRC64 checksum is OK
```

AOF 持久化

- AOF 持久化：Append Only File
- AOF 持久化，基本过程：
    - 命令追加：redisServer 中 sds aof_buf 缓冲区
    - 文件写入：命令，写入文件
    - 文件同步：文件内容，刷新到磁盘
- AOF 文件，默认同步策略：
    - appendfsync everysec


redisServer
// AOF 的缓冲区
sds aof_buf
追加 AOF 文件
AOF 文件
同步到磁盘
Redis 中选项 appendfsync，控制同步策略：
1. always：aof_buf 内容，写入 AOF 文件后，直接同步到磁盘，
2. everysec：将 aof_buf 内容，写入文件后，如果上次同步 AOF 文件的时间超过 1 s，则，进行一次 AOF 文件同步
3. no：完全依赖 OS，进行同步
OS: 用户向文件中追加内容时
1. 会在内存缓冲区中暂存，
2. 缓冲区填满、或者超过一定时间，才真正将缓冲区同步到磁盘

- 开启、关闭 AOF：
    - appendonly no/yes
    - 默认：appendonly no
    - Note：appendonly no ，
        - 不会追加生成 AOF 文件，
        - 重启时，也不会载入 AOF文件
        - 但执行：bgrewriteaof 仍然会重写 AOF 文件
- Redis 的 选项 appendfsync，控制同步策略：
    - always
    - everysec
    - no
- 为什么不使用：always?
- 效率 vs. 数据丢失风险

AOF 载入

- AOF 文件中保存的是：key-value 的写命令
- 加载 AOF 文件的过程：

1. 创建伪客户端（fake client）：不带网络连接，Server 侧的 redisClient，因为命令来至 AOF 文件
2. 从 AOF 文件读取一条命令
3. 伪客户端执行命令
4. 重复步骤1、步骤2

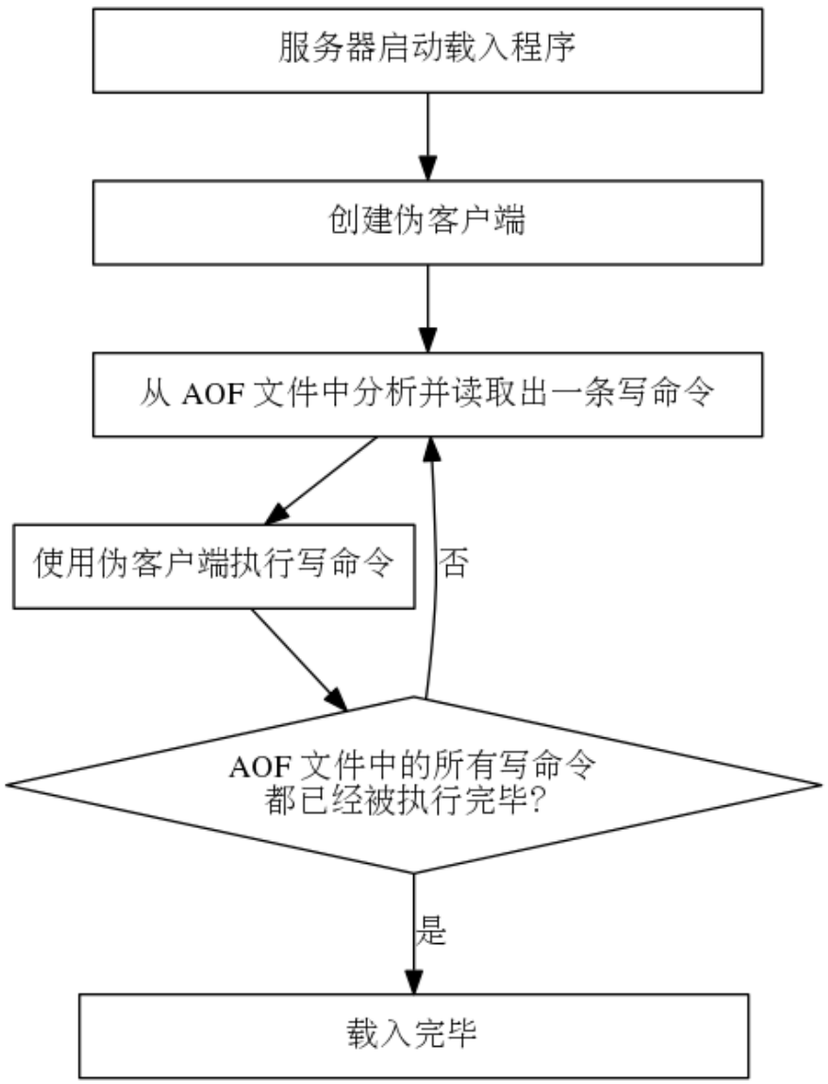
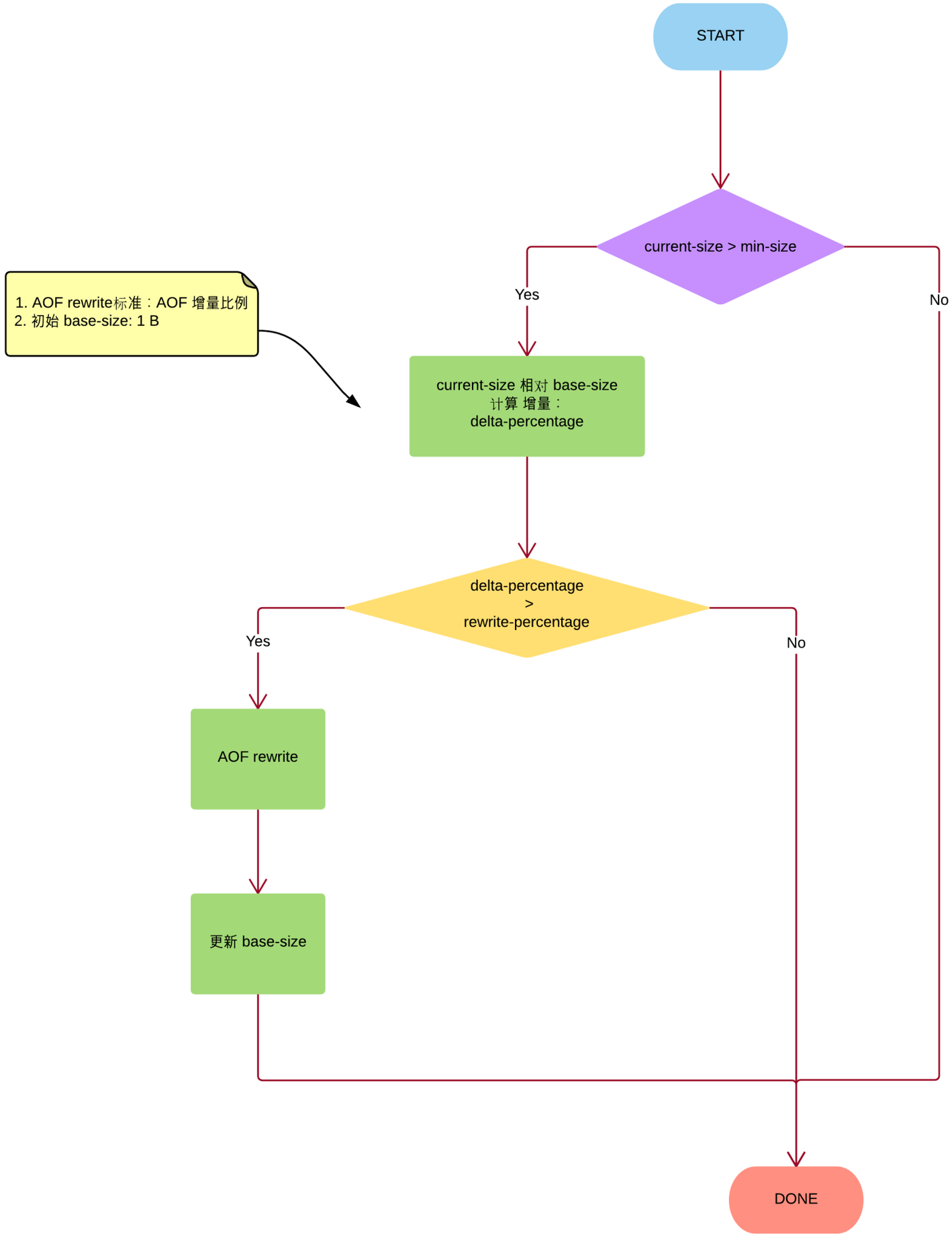


图 11-2　AOF 文件载入过程

AOF 重写

- 为什么需要 AOF 重写?
  - AOF，增量式持久化，文件越来越大，载入效率低
    - 大量的冗余命令，例如，set key value
- AOF 重写，原理:
  - 不对现有 AOF 文件读取、分析，而直接读取数据库的当前状态
  - 遍历每个数据库，遍历每个 key 生成对应的写命令

- AOF 重写，两种方式：
    - 主进程
    - 子进程：BGREWRITEAOF
- 子进程 AOF 重写
    - 问题：Redis 仍然对外提供服务，AOF 重写期间，会有新的写命令执行
    - AOF 重写缓冲区，执行完 AOF 重写后，会将 AOF 重写缓冲区内命令，追加到 AOF 文件
- AOF 重写，自动触发策略：
    - auto-aof-rewrite-percentage 100 ： AOF 文件的增量比例，current-size 相对 base-size 的增量
    - Note：第一次的 base-size 为 1 B
    - auto-aof-rewrite-min-size 64mb：AOF 的 current-size 太小时，重写意义不大，设定最低阈值


START
current-size > min-size
Yes
No
1. AOF rewrite标准：AOF 增量比例
2. 初始 base-size: 1 B
current-size 相对 base-size
计算 增量：
delta-percentage
delta-percentage
>
rewrite-percentage
Yes
No
AOF rewrite
更新 base-size
DONE

- 疑问：AOF 重写期间，是否还会追加 AOF 文件?
  - 写命令，会同时追加到 AOF 缓冲区、AOF 重写缓冲区。

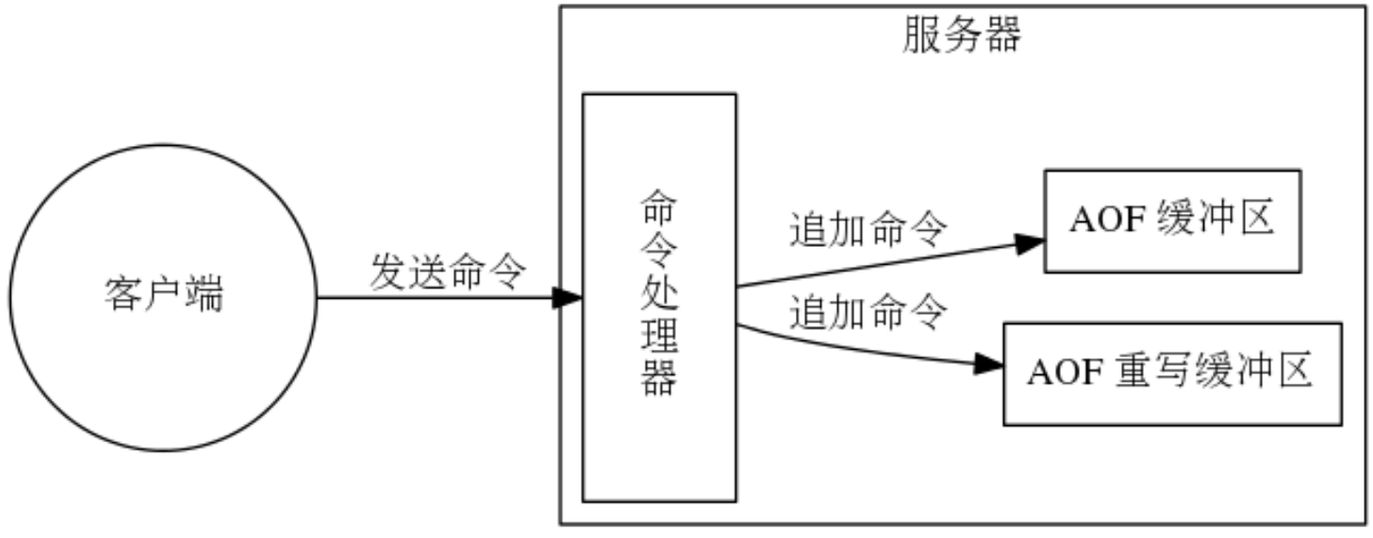


图 11-4　服务器同时将命令发送给 AOF 文件和 AOF 重写缓冲区

常见问题

- 问题1：RDB、AOF 同时存在时，优先使用哪个?
  - AOF 文件更新频率更高，系统优先加载AOF 文件
- RDB：全量持久化
- AOF：增量持久化

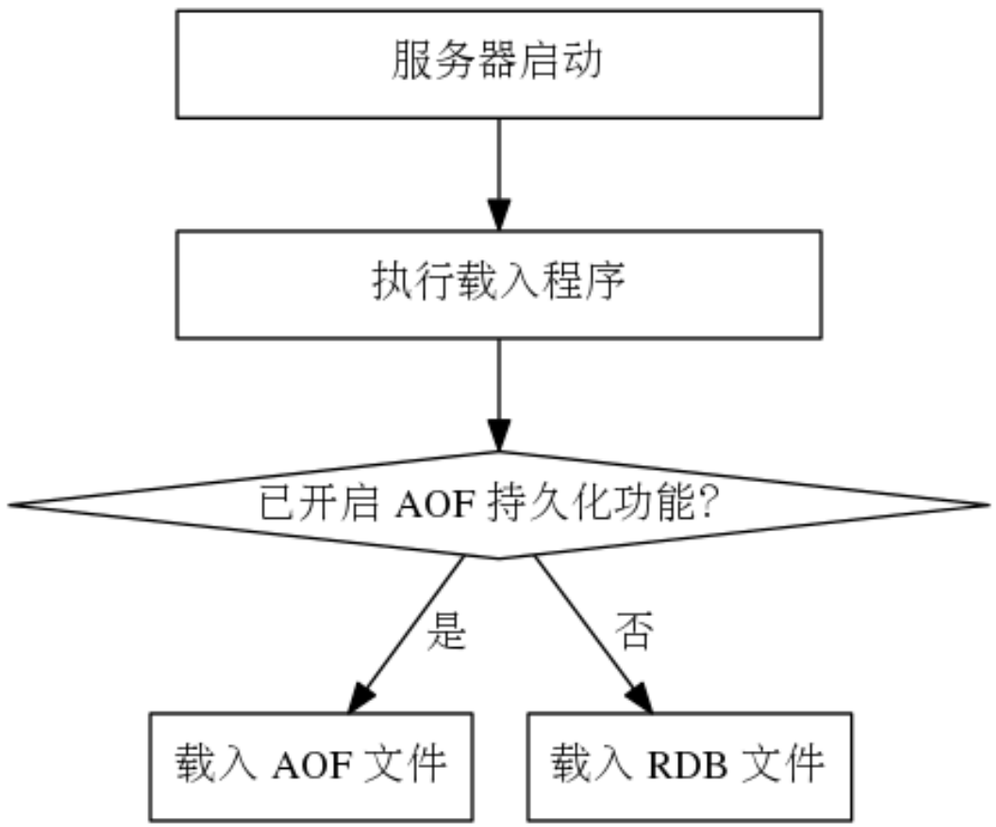


图 10-4　服务器载入文件时的判断流程

- 问题2：RDB、AOF 时，过期 key 的处理?
    - RDB：
        - 保存RDB时，不存过期key
        - 载入RDB时，master 不载入，slave载入
    - AOF：
        - 写入AOF，不存过期 key
        - 惰性删除、者定期删除时，新增DEL记录，
        - AOF重写时，不存过期key

- SAVE、BGSAVE 的区别
    - SAVE 阻塞 Redis 服务进程
    - BGSAVE 启用子进程，不阻塞服务进程，执行 BGSAVE 期间有如下约束：
        - 不会执行SAVE 命令：
            - Background save already in progress
        - 不会执行BGSAVE命令：
            - Background save already in progress
        - 不会与BGREWRITEAOF 同时执行：
            - 在执行 BGREWRITEAOF 时，如果执行 BGSAVE，则：Can’t BGSAVE while AOF log rewriting is in progress
            - 在执行 BGSAVE 时，如果执行 BGREWRITEAOF，则：BGREWRITEAOF 会延迟到 BGSAVE 执行结束之后，再执行
- SAVE、BGSAVE、BGREWRITEAOF 不同时执行的原因：
    - SAVE、BGSAVE，避免产生资源竞争
    - BGSAVE、BGREWRITEAOF，两个操作都会产生大量的写磁盘操作，效率问题