链滴

Redis（三）持久化与事物机制

作者：wlgzs-sjl
原文链接：https://ld246.com/article/1578905070075
来源网站：链滴

什么是Redis持久化?

  就是将内存数据保存到硬盘。

  Redis 持久化存储 (AOF 与 RDB 两种模式)

RDB持久化

  RDB 持久化把当前进程数据生成快照（.rdb）文件保存到硬盘的过程，有动触发和自动触发；

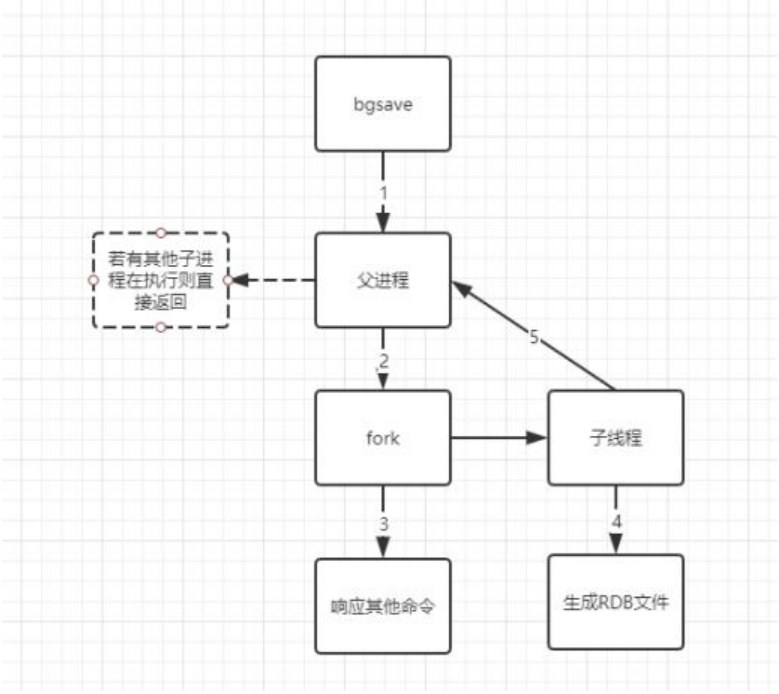
  手动触发有save 和bgsave 两命令：

  save 命令：阻塞当前 Redis，直到RDB 持久化过程完成为止，若内存实例较大，会造成长时间阻塞，线上环境不建议用它；

  bgsave 命令：redis 进程执行fork 操作创建子线程，由子线程完成持久化阻塞时间很短（微秒级），是save 的优化,在执行redis-cli shutdown 关闭redis 服务时，如果没有启AOF 持久化，自动执行 bgsave;

  显然bgsave 是对save 的优化。

  bgsave执行流程：



  而且RDB 是在某个时间 点将数据写入一个临时文件，持久化结束后，用这个临时文件替换上次持久化的文件，达到数据恢复。

  优点：**使用单独子进程来进行持久化，主进程不会进行任何 IO 操作，保证了 redis 的高性能；而且RDB文件存储的是压缩的二进制文件，适用于备份、全量复制，可用于灾难备份，同时RDB文件的加载速度远超于AOF文件；**

  缺点：**RDB 是间隔一段时间进行持久化，如果持久化之间 redis 发生故障，会发生数据丢失。所以这种方式更适合数据要求不严谨的时候，因为RDB无法做到实持久化，而且每次都要创建子进程，频繁创建成本过高；还有一点，RDB文件保存的二进制文件存在老版本不兼容的问题；**

  这里说的这个执行数据写入到临时文件的时间点是可以通过配置来自己确定，通过配置**redis 在 n 秒内如果超过 m 个 key 被修改**这执行一次 RDB 操作。这个操作就类似于在个时间点来保存一次 Redis 的所有数据，一次快照数据。所有这个持久化方法也通常叫做 snapshots。

AOF持久化

  Append-only file，将“操作 + 数据”以格式化指令的方式追加到操作日文件的尾部，在 append 操作返回后(已经写入到文件或者即将写入)，才进行实际的数据变更，“日文件”保存了历史所有的操作过程；当 server 需要数据恢复时，可以直接 replay 此日志文件，即可原所有的操作过程。AOF 相对可靠，它和 mysql 中 bin.log、apache.log、zookeeper 中 txn-log 直异曲同工。AOF 文件内容是字符串，非常容易阅读和解析。

  优点：**可以保持更高的数据完整性，如果设置追加 file 的间是 1s，如果 redis 发生故障，最多会丢失 1s 的数据；且如果日志写入不完整支持 redis-check-ao 来进行日志修复；AOF 文件没被 rewrite 之前（文件过大时会对命令进行合并重写），可以删除其的某些命令（比如误操作的 flushall）。**

  缺点：**AOF 文件比 RDB 文件大，且恢复速度慢。**

  我们可以简单的认为 AOF 就是日志文件，此文件只会记录“变更操作”(例：set/del 等)，如果 server 中持续的大量变更操作，将会导致 AOF 文件非常的庞大，意味着 server 失效后，数据恢复的过程将会很长；事实上，一条数据经过多次变更，将会产生多条 AOF 记录，其

只要保存当前的状态，历史的操作记录是可以抛弃的；因为 AOF 持久化模式还伴生了“AOF rewrite
。

  AOF 的特性决定了它相对比较安全，如果你期望数据更少的丢失，那么可采用 AOF 模式。如果 AOF 文件正在被写入时突然 server 失效，有可能导致文件的最后一次记录是完整，你可以通过手工或者程序的方式去检测并修正不完整的记录，以便通过 aof 文件恢复能够正常同时需要提醒，如果你的 redis 持久化手段中有 aof，那么在 server 故障失效后再次启动前，需要检 aof 文件的完整性。

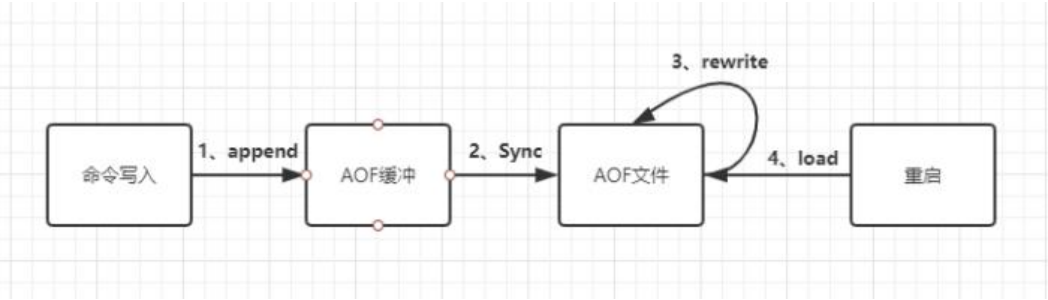
  AOF 是文件操作，对于变更操作比较密集的 server，那么必将造成磁盘 IO 负荷加重；此外 linux 对文件操作采取了“延迟写入”手段，即并非每次 write 操作都会触发实际磁操作，而是进入了 buffer 中，当 buffer 数据达到阀值时触发实际写入(也有其他时机)，这是 linux 文件系统的优化，但是这却有可能带来隐患，如果 buffer 没有刷新到磁盘，此时物理机器失效(比如电)，那么有可能导致最后一条或者多条 AOF记录的丢失。通过上述配置文件，可以得知 redis 提供了 3 种 AOF 记录同步选项：

  always：每一条 AOF记录都立即同步到文件，这是最安全的方式，也以为多的磁盘操作和阻塞延迟，是 IO 开支较大。

  everysec：每秒同步一次，性能和安全都比较中庸的方式，也是 redis 推荐方式。如果遇到物理服务器故障，有可能导致最近一秒内 AOF记录丢失(可能为部分丢失)。

  no：redis 并不直接调用文件同步，而是交给操作系统来处理，操作系统可根据 buffer 填充情况 / 通道空闲时间等择机触发同步；这是一种普通的文件操作方式。性能较好，物理服务器故障时，数据丢失量会因 OS 配置有关。

AOF持久化流程：命令写入->文件同步->文件重写->重启加载



如何从AOF 恢复?

1. 设置 appendonly yes;
2. 将appendonly.aof 放到dir 参数指定的目录;
3. 启动 Redis，Redis 会自动加载appendonly.aof 文件。

redis 重启时恢复加载AOF 与RDB 顺序及流程:

1. 当AOF 和RDB 文件同时存在时，优先加载
2. 若关闭了 AOF，加载RDB 文件
3. 加载AOF/RDB 成功，redis 重启成功
4. AOF/RDB 存在错误，redis 启动失败并打印错误信息

AOF与RDB区别

1. AOF 更加安全，可以将数据更加及时的同步到文件中，但是 AOF 需要较多的磁盘 IO 开支，AOF 件尺寸较大，文件内容恢复数度相对较慢。

2. RDB安全性较差，它是“正常时期”数据备份以及 master-slave 数据同步的最佳手段，文件尺寸小，恢复数度较快。

  可以通过配置文件来指定它们中的一种，或者同时使用它们(不建议同时使用，或者全部禁用，在架构良好的环境中，master 通常使用 AOF，slave 使用 RDB，主要原因是 master 需要首先确保数据完整性，它作为数据备份的第一选择；slave 提供只读服务(目前 slave 只能提供取服务)，它的主要目的就是快速响应客户端 read 请求；但是如果你的 redis 运行在网络稳定性差 / 理环境糟糕情况下，建议你 master 和 slave 均采取 AOF，这个在 master 和 slave 角色切换时，可减少“人工数据备份” / “人工引导数据恢复”的时间成本；如果你的环境一切非常良好，且服务需接收密集性的 write 操作，那么建议 master 采取 RDB，而 slave 采用 AOF。

Redis事务机制

  Redis 事务可以一次执行多个命令， 并且带有以下两个重要的保证：

  事务是一个单独的隔离操作：事务中的所有命令都会序列化、按顺序地执行事务在执行的过程中，不会被其他客户端发送来的命令请求所打断。

  事务是一个原子操作：事务中的命令要么全部被执行，要么全部都不执行。

  一个事务从开始到执行会经历以下三个阶段：

    开始事务。

    命令入队。

    执行事务。

Redis 事务命令

   DISCARD -- 取消事务，放弃执行事务块内的所有命令。

   EXEC -- 执行所有事务块内的命令。

   MULTI -- 标记一个事务块的开始。

   UNWATCH -- 取消 WATCH 命令对所有 key 的监视。

   WATCH key [key ...] -- 监视一个(或多个) key ，如果在事务执行之前这个或这些) key 被其他命令所改动，那么事务将被打断。