Redis RDB持久化、AOF持久化

题目标签

学习时长：20分钟

题目难度：中等

知识点标签：缓存、RDB、AOF持久化

题目描述

Redis RDB持久化、AOF持久化

题目解决

1.持久化

1.1 持久化简介

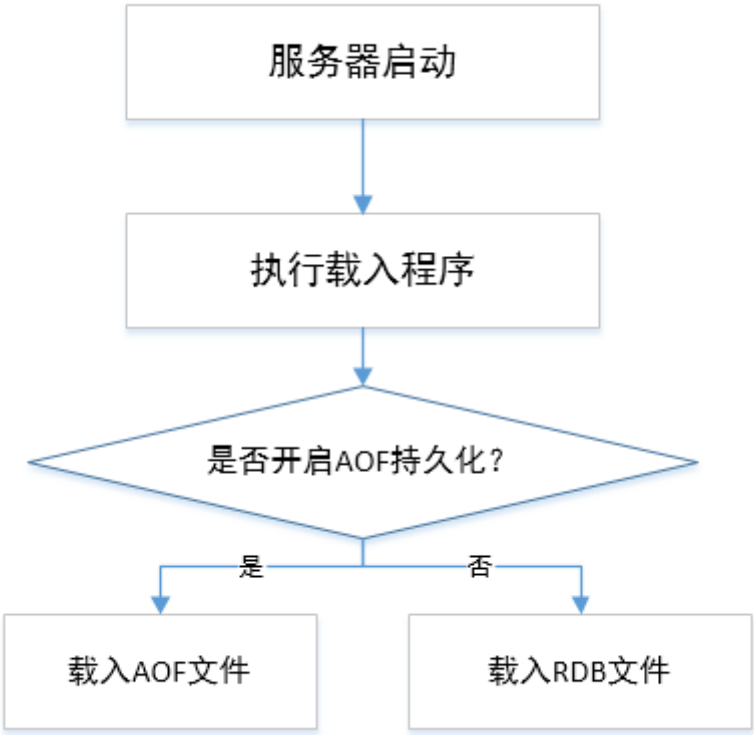
持久化（Persistence），持久化是将程序数据在持久状态和瞬时状态间转换的机制，即把数据（如内存中的对象）保存到可永久保存的存储设备中（如磁盘）。

1.2 redis持久化

redis为内存数据库，为了防止服务器宕机以及服务器进程退出后，服务器数据丢失，Redis提供了持久化功能，即将Redis中内存数据持久化到磁盘中。Redis 提供了不同级别的持久化方式：

- RDB持久化方式：可以在指定的时间间隔能对数据进行快照存储.
- AOF持久化方式：记录每次对服务器写的操作,当服务器重启的时候会重新执行这些命令来恢复原始的数据,AOF命令以redis协议追加保存每次写的操作到文件末尾.Redis还能对AOF文件进行后台重写,使得AOF文件的体积不至于过大.

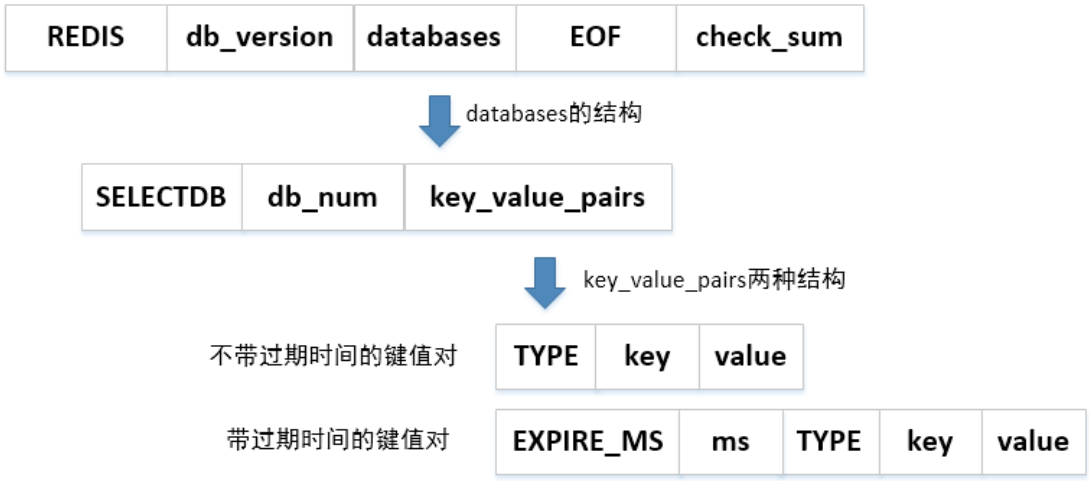
如果服务器开启了AOF持久化功能。服务器会优先使用AOF文件还原数据。只有关闭了AOF持久化功能，服务器才会使用RDB文件还原数据



2. RDB持久化

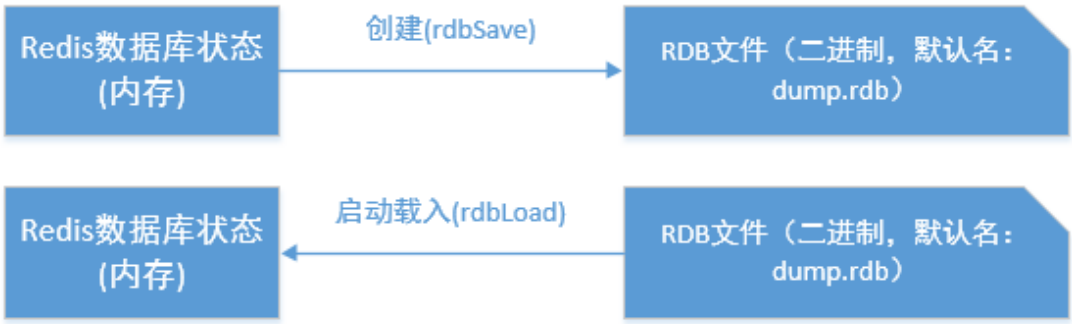
2.1 RDB文件格式

RDB文件是一个经过压缩的二进制文件（默认的文件名：dump.rdb），由多个部分组成，RDB格式：



2.2 RDB文件持久化创建与载入

在 Redis持久化时， RDB 程序将当前内存中的数据库状态保存到磁盘文件中， 在 Redis 重启动时，RDB 程序可以通过载入 RDB 文件来还原数据库的状态。



2.3 工作方式

当 Redis 需要保存 dump.rdb 文件时， 服务器执行以下操作:

- Redis 调用forks。同时拥有父进程和子进程。
- 子进程将数据集写入到一个临时 RDB 文件中。
- 当子进程完成对新 RDB 文件的写入时，Redis 用新 RDB 文件替换原来的 RDB 文件，并删除旧的 RDB 文件。

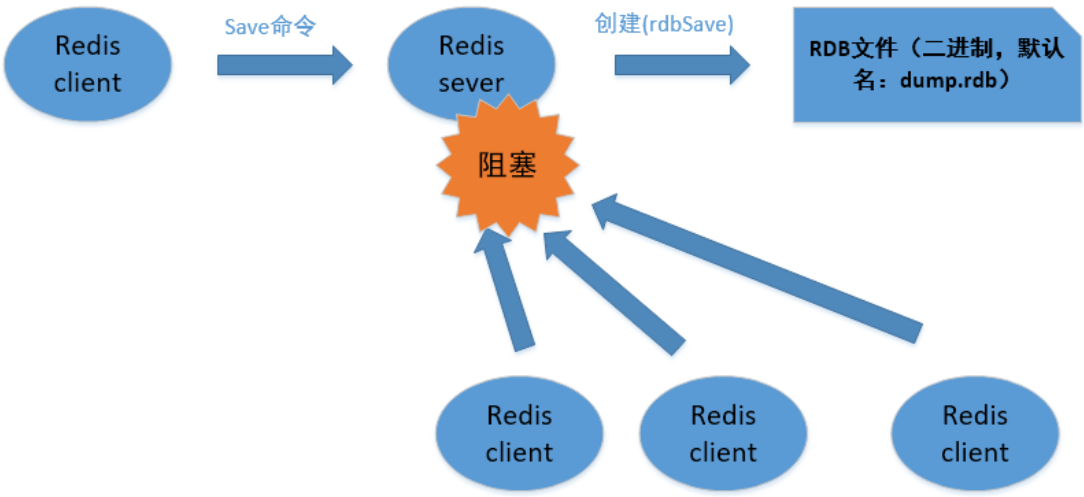
这种工作方式使得 Redis 可以从写时复制（copy-on-write）机制中获益。

2.4 创建方式

SAVE

同步操作，在执行该命令时，服务器会被阻塞，拒绝客户端发送的命令请求

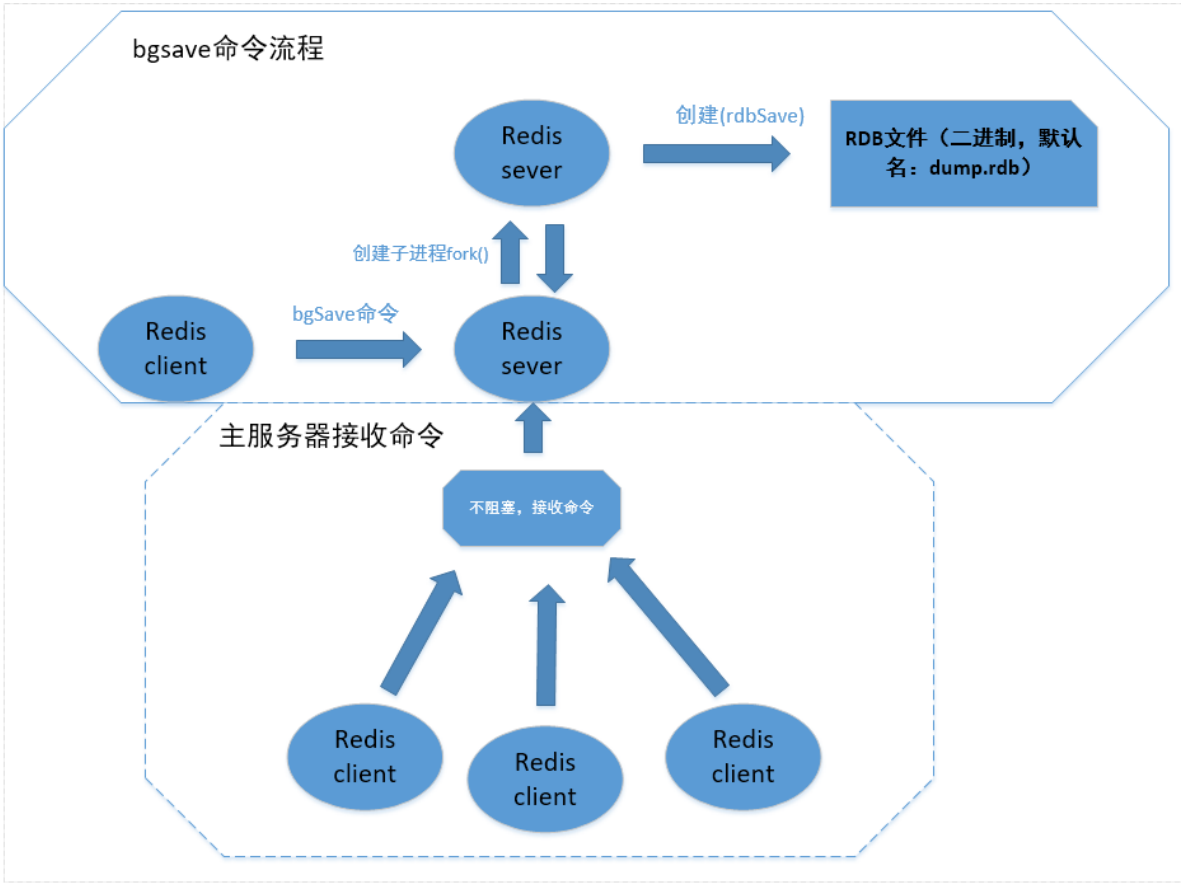
```
    redis> save
1
```



BGSAVE

异步操作，在执行该命令时，子进程执行保存工作，服务器还可以继续让主线程处理客户端发送的命令请求

```
 redis>bgsave
1
```



自动创建

由于BGSAVE命令可不阻塞服务器进程下执行，可以让用户自定义save属性，让服务器每个一段时间自动执行一次BGSAVE命令（即通过配置文件对 Redis 进行设置， 让它在“ N 秒内数据集至少有 M 个改动”这一条件被满足时， 自动进行数据集保存操作）。

```
比如：
/*服务器在900秒之内，对数据库进行了至少1次修改*/
Save  900   1
/*服务器在300秒之内，对数据库进行了至少10次修改*/
Save  300   10
/*服务器在60秒之内，对数据库进行了至少10000次修改*/
Save  60     10000
1234567
```

只要满足其中一个条件就会执行BGSAVE命令

2.5 RDB 默认配置

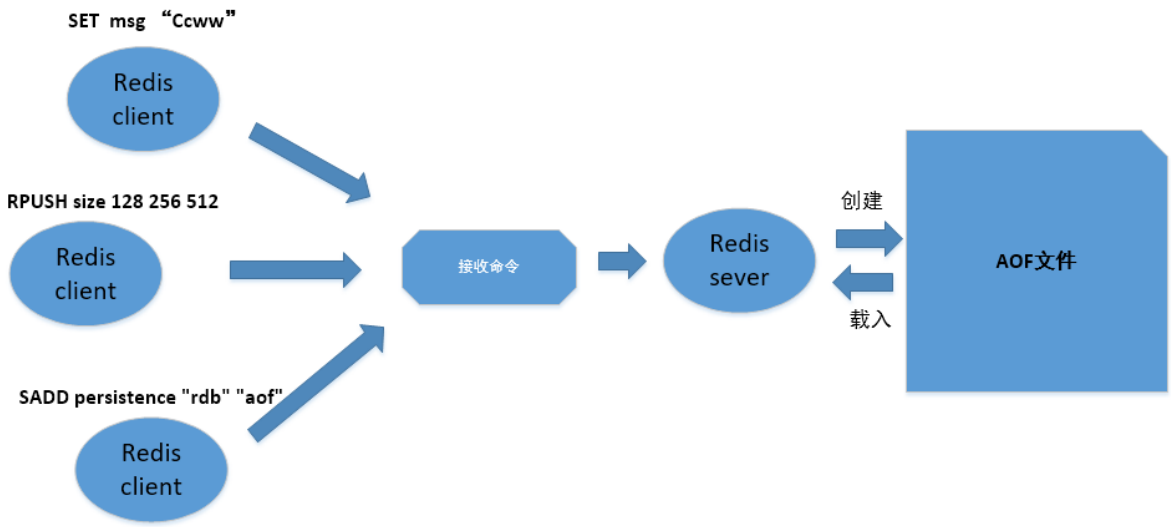
```
################################ SNAPSHOTTING  ################################
#
# Save the DB on disk:
#在给定的秒数和给定的对数据库的写操作数下，自动持久化操作。
#   save <seconds> <changes>
#
save 900 1
save 300 10
save 60 10000

#bgsave发生错误时是否停止写入，一般为yes
stop-writes-on-bgsave-error yes
```

```
#持久化时是否使用LZF压缩字符串对象?
rdbcompression yes

#是否对rdb文件进行校验和检验，通常为yes
rdbchecksum yes

# RDB持久化文件名
dbfilename dump.rdb

#持久化文件存储目录
dir ./
12345678910111213141516171819202122232 4
```

3. AOF持久化

3.1 AOF持久化简介

AOF持久化是通过保存Redis服务器所执行的写命令来记录数据库状态



AOF持久化功能实现：

1. append命令追加：当AOF持久化功能处于打开状态时，服务器执行完一个写命令会协议格式被执行的命令追加服务器状态的aof_buf缓冲区的末尾。

```
 reids>SET KET VAULE
 //协议格式
 \r\n$3\r\nSET\r\n$3\r\nKEY\r\n$5\r\nVAULE\r\n
123
```

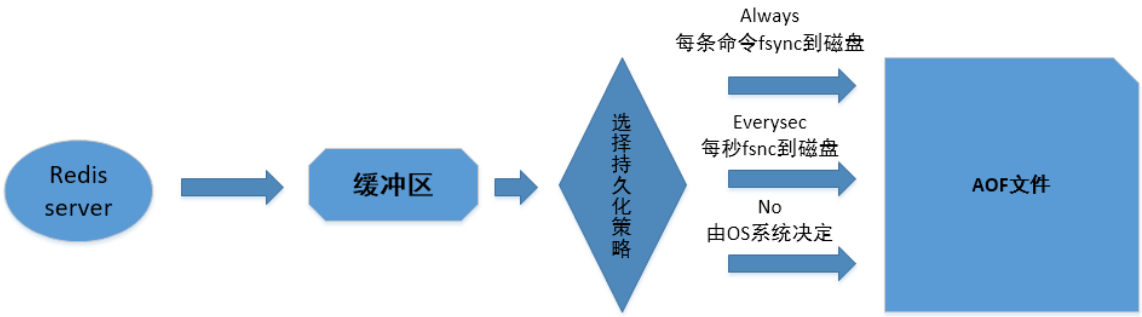
2. 文件写入和同步sync：Redis的服务器进程是一个事件循环，这个文件事件负责接收客户端的命令请求以及向客户端发送命令回复。当执行了append命令追加后，服务器会调用flushAppendOnlyFile函数是否需要将AOF缓冲区的内容写入和保存到AOF文件

```
 redis> SET msg "Ccww"
 redis> SADD persistence "rdb" "aof"
 redis> RPUSH size 128 256 512
123
```

3.2 AOF持久化策略

AOF持久化策略(即缓冲区内容写入和同步sync到AOF中)，可以通过配置appendfsync属性来选择AOF持久化策略：

- always：将aof_buf缓冲区中的所有内容写入并同步到AOF文件，每次有新命令追加到 AOF 文件时就执行一次 fsync。
- everysec（默认）：如果上次同步AOF的时间距离现在超过一秒，先将aof_buf缓冲区中的所有内容写入到AOF文件，再次对AOF文件进行同步，且同步操作由一个专门线程负责执行。
- no：将aof_buf缓冲区中的所有内容写入到AOF文件，但并不对AOF文件进行同步，何时同步由操作系统(OS)决定。



AOF持久化策略的效率与安全性：

- Always:效率最慢的，但安全性是最安全的，即使出现故障宕机，持久化也只会丢失一个事件 循环的命令数据
- everysec：兼顾速度和安全性，出现宕机也只是丢失一秒钟的命令数据
- No:写入最快，但综合起来单次同步是时间是最长的，且出现宕机时会丢失上传同步AOF文件之后的所有命令数据。

3.3 AOF重写

由于AOF持久化会把执行的写命令追加到AOF文件中，所以随着时间写入命令会不断增加， AOF文件的体积也会变得越来越大。AOF文件体积大对Reids服务器，甚至宿主服务器造成影响。

为了解决AOF文件体积膨胀的问题，Redis提供了AOF文件重写（rewrite）功能：

- 生成一个不保存任何浪费空间的冗余命令新的AOF文件，且新旧AOF文件保存数据库状态一样的
- 新的AOF文件是通过读取数据库中的键值对来实现的，程序无须对现有的AOF文件进行读入，分析，或者写入操作。
- 为防止缓冲区溢出，重写处理list，hash，set以及Zset时，超过设置常量数量时会多条相同命令记录一个集合。
- Redis 2.4 可以通过配置自动触发 AOF 重写，触发参数 `auto-aof-rewrite-percentage` **(触发AOF文件执行重写的增长率)** 以及 `auto-aof-rewrite-min-size` **(触发AOF文件执行重写的最小尺寸)**

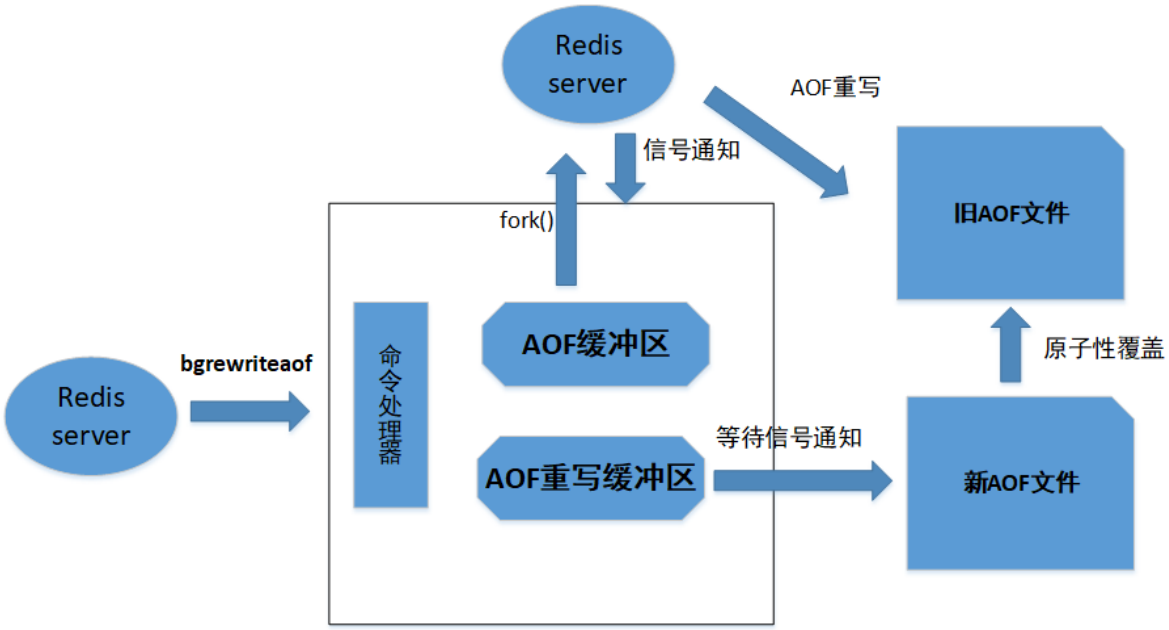
AOF重写的作用：

- 减少磁盘占用量
- 加速数据恢复

Redis服务器使用单个线程来处理命令请求，服务器大量调用aof_rewrite函数，在AOF重写期间，则无法处理client发来的命令请求，所以AOF重写程序放在子进程执行，好处：

1. 子进程进行AOF重写期间，服务器进程可以继续处理命令请求
2. 子进程带有服务器进程的数据副本，保证了数据的安全性。

AOF重写使用子进程会造成数据库与重写后的AOF保存的数据不一致，为了解决这种数据不一致，redis使用了AOF重写缓冲区
实现：



BGREWRITEAOF命令实现原理（只有信号处理函数执行时才对服务器进程造成阻塞）：

- 执行命令，同时将命令追加到AOF缓冲区和AOF重写缓冲区
- 当AOF子进程重写完成后，发送一个信号给父进程，父进程将执行AOF重写缓冲区中的所有内容写入到新AOF文件中，新AOF文件保存的数据库状态将和服务器当前的数据库状态一致。
- 对新的AOF文件进行改名，原子性地覆盖现有AOF文件，完成新旧两个AOF文件替换处理完成。

3.4 AOF持久化默认参数

```
################################ APPEND ONLY MODE ################################

#开启AOF持久化方式
appendonly no

#AOF持久化文件名
appendfilename "appendonly.aof"
#每秒把缓冲区的数据fsync到磁盘
appendfsync everysec
# appendfsync no
#是否在执行重写时不同步数据到AOF文件
no-appendfsync-on-rewrite no

# 触发AOF文件执行重写的增长率
auto-aof-rewrite-percentage 100
#触发AOF文件执行重写的最小size
auto-aof-rewrite-min-size 64mb

#redis在恢复时，会忽略最后一条可能存在问题的指令
aof-load-truncated yes

#是否打开混合开关
aof-use-rdb-preamble yes
12345678910111213141516171819202122223
```

4 持久化方式总结与抉择

4.1 RDB优缺点

RDB的优点

- RDB是一个非常紧凑的文件,它保存了某个时间点得数据集,非常适用于数据集的备份,比如你可以在每个小时报保存一下过去24小时内的数据,同时每天保存过去30天的数据,这样即使出了问题你也可以根据需求恢复到不同版本的数据集.
- 基于RDB文件紧凑性,便于复制数据到一个远端数据中心，非常适用于灾难恢复.
- RDB在保存RDB文件时父进程唯一需要做的就是fork出一个子进程,接下来的工作全部由子进程来做，父进程不需要再做其他IO操作，所以RDB持久化方式可以最大化redis的性能.
- 与AOF相比,在恢复大的数据集的时候，RDB方式会更快一些.

RDB的缺点

- 如果你希望在redis意外停止工作（例如电源中断）的情况下丢失的数据最少的话，那么RDB不适合你.虽然你可以配置不同的save时间点(例如每隔5分钟并且对数据集有100个写的操作),是Redis要完整的保存整个数据集是一个比较繁重的工作,你通常会每隔5分钟或者更久做一次完整的保存,万一在Redis意外宕机,你可能会丢失几分钟的数据.
- RDB 需要经常fork子进程来保存数据集到硬盘上,当数据集比较大的时候,fork的过程是非常耗时的,可能会导致Redis在一些毫秒级内不能响应客户端的请求.如果数据集巨大并且CPU性能不是很好的情况下,这种情况会持续1秒,AOF也需要fork,但是你可以调节重写日志文件的频率来提高数据集的耐久度.

4.2 AOF的优缺点

AOF的优点:

- 使用AOF 会让你的Redis更加耐久:使用不同的fsync策略：无fsync,每秒fsync,每次写的时候fsync.使用默认的每秒fsync策略,Redis的性能依然很好(fsync是由后台线程进行处理的,主线程会尽力处理客户端请求),一旦出现故障，你最多丢失1秒的数据.
- AOF文件是一个只进行追加的日志文件,所以不需要写入seek,即使由于某些原因(磁盘空间已满，写的过程中宕机等等)未执行完整的写入命令,你也可使用redis-check-aof工具修复问题.
- Redis可以在AOF文件体积变得过大时，自动对 AOF 进行重写： 重写后的新 AOF 文件包含了恢复当前数据集所需的最小命令集合。 整个重写操作是绝对安全的，因为 Redis 在创建新 AOF 文件的过程中，会继续将命令追加到现有的 AOF 文件里面，即使重写过程中发生停机，现有的 AOF 文件也不会丢失。 而一旦新 AOF 文件创建完毕，Redis 就会从旧 AOF 文件切换到新 AOF 文件，并开始对新 AOF 文件进行追加操作。
- AOF 文件有序地保存了对数据库执行的所有写入操作， 这些写入操作以 Redis 协议的格式保存，因此 AOF 文件的内容非常容易被人读懂， 对文件进行分析（parse）也很轻松。 导出（export） AOF 文件也非常简单（例如， 如果你不小心执行了 FLUSHALL 命令， 但只要 AOF 文件未被重写， 那么只要停止服务器， 移除 AOF 文件末尾的 FLUSHALL 命令， 并重启 Redis ， 就可以将数据集恢复到 FLUSHALL 执行之前的状态）。

AOF 缺点:

- 对于相同的数据集来说，AOF 文件的体积通常要大于 RDB 文件的体积。
- 根据所使用的 fsync 策略，AOF 的速度可能会慢于 RDB 。 在一般情况下， 每秒 fsync 的性能依然非常高， 而关闭 fsync 可以让 AOF 的速度和 RDB 一样快， 即使在高负荷之下也是如此。 不过在处理巨大的写入载入时，RDB 可以提供更有保证的最大延迟时间（latency）。

4.3 如何选择使用哪种持久化方式?

一般来说， 如果想达到足以媲美 PostgreSQL 的数据安全性， 你应该同时使用两种持久化功能。

如果你非常关心你的数据， 但仍然可以承受数分钟以内的数据丢失， 那么你可以只使用 RDB 持久化。

有很多用户都只使用 AOF 持久化， 但我们并不推荐这种方式： 因为定时生成 RDB 快照（snapshot）非常便于进行数据库备份， 并且 RDB 恢复数据集的速度也要比 AOF 恢复的速度要快， 除此之外， 使用 RDB 还可以避免之前提到的 AOF 程序的 bug 。