1. AOF持久化
2. 概述

除了RDB持久化功能，Redis还提供了AOF持久化功能，通过保存Redis服务器所执行的写命令来记录数据库状态。被写入AOF文件的所有命令都是以Redis的命令请求协议格式保存的，服务器在启动时，可以通过载入和执行AOF文件中保存的命令来还原服务器关闭之前的数据库状态。

1. AOF持久化的实现

实现分为命令追加，文件写入，文件同步三个步骤。

1. 命令追加

当AOF持久化功能处于打开状态时，服务器在执行完一个写命令之后，会以协议格式将被执行的写命令追加到服务器状态的aof\_buf缓冲区的末尾。

1. 写入与同步

Redis的服务器进程近视一个事件循环，这个循环中的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，而时间事件负责执行像serverCron函数这样需要定时运行的函数。服务器在每次结束一个事件循环之前，都会调用flushAppendOnlyFile函数，考虑是否需要将aof\_buf缓冲区中的内容写入和被保存到AOF文件中，判断依据是服务器配置的appendfsync选项，该选项可能有三个值，默认是everysec：

1. always

将aof\_buf缓冲区中的所有内容写入并同步到AOF文件。效率最慢，但是最安全，最多丢失一个时间循环中的命令数据。

1. everysec

将aof\_buf缓冲区中的所有内容写入到AOF文件中，如果上次同步AOF文件的时间距离现在超过1s，那么再次对AOF文件进行同步。最多丢失一秒钟的命令数据。

1. no

将aof\_buf缓冲区中的所有内容写入到AOF文件中，但不对AOF文件进行同步。会丢失上次同步AOF文件之后的所有写命令数据。

1. AOF文件的载入和还原

Redis读取AOF文件并还原数据库状态的步骤如下：

1. 创建一个不带网络连接的伪客户端，因为Redis的命令只能在客户端上下文中执行，而载入AOF文件时所使用的命令直接来源于AOF文件而不是网络连接，所以服务器使用了一个没有网络连接的伪客户端。
2. 从AOF文件中分析并读取一条写命令
3. 使用伪客户端执行被读出的写命令
4. 重复上述步骤，直到所有命令都被读取出来。
5. AOF重写
6. 概述

随着服务器运行时间流逝，AOF文件中的内容会越来越多，文件的体积也会越来越大，如果不控制的话，会对Redis服务器以及宿主机造成影响。因此，为了解决AOF文件体积膨胀的问题，Redis提供了AOF文件重写功能，服务器会创建一个新的AOF文件来替代现有的AOF文件，两个AOF文件保存的数据库状态相同，但新的AOF文件中不会包含浪费空间的冗余命令，所以体积更小。

1. 实现

实际上，AOF重写不需要对现有的AOF文件进行任何读取，分析，写入操作，它是通过直接读取当前的数据库状态实现的。如之前的AOF文件中记录了对一个键的5条写入命令，那么新的AOF文件直接记录最新的一个写入命令即可，这就减少了AOF文件的体积。为了避免在执行命令时造成客户端输入缓冲区溢出，重写程序会先检查键所包含的元素数量，如果过多将使用多条命令来记录键的值。

1. AOF后台重写

AOF重写程序aof\_rewrite函数可以完成创建一个新的AOF文件的任务，但是这个函数会进行大量的写入操作，所以调用这个函数的线程会被长时间阻塞。由于Redis服务器采用单线程来处理命令请求，所以将AOF重写程序放到子进程中执行，使得父进程继续处理命令请求。但是这样在进行重写的过程中，父进程对数据库状态的修改可能就会导致与重写后的AOF文件状态不一致。

为了解决这个问题，Redis服务器设置了一个AOF重写缓冲区，在服务器创建子进程之后开始使用，当服务器执行完一个写命令之后，它会同时将这个写命令发送给AOF缓冲区和AOF重写缓冲区，AOF缓冲区中的内容会被定期写入和同步到AOF文件中，而当子进程完成AOF重写工作后，它会向父进程发送一个信号，父进程在接收到信号后，会调用一个信号处理函数，将AOF重写缓冲区中的内容写入到新的AOF文件中，这时的新AOF文件数据库状态与当前的数据库是一致的。对新的AOF文件改名，覆盖原AOF文件。

在整个AOF后台重写过程中，只有信号处理函数会对服务器进程阻塞，其他时候，AOF后台重写都不会阻塞父进程，将AOF重写对服务器性能的影响降到最低。这也是BGREWRITEAOF命令的实现原理。