

Save 秒钟 写操作次数：

RDB是整个内存的压缩过的Snapshot，RDB的数据结构，可以配置复合的快照触发条件，

默认

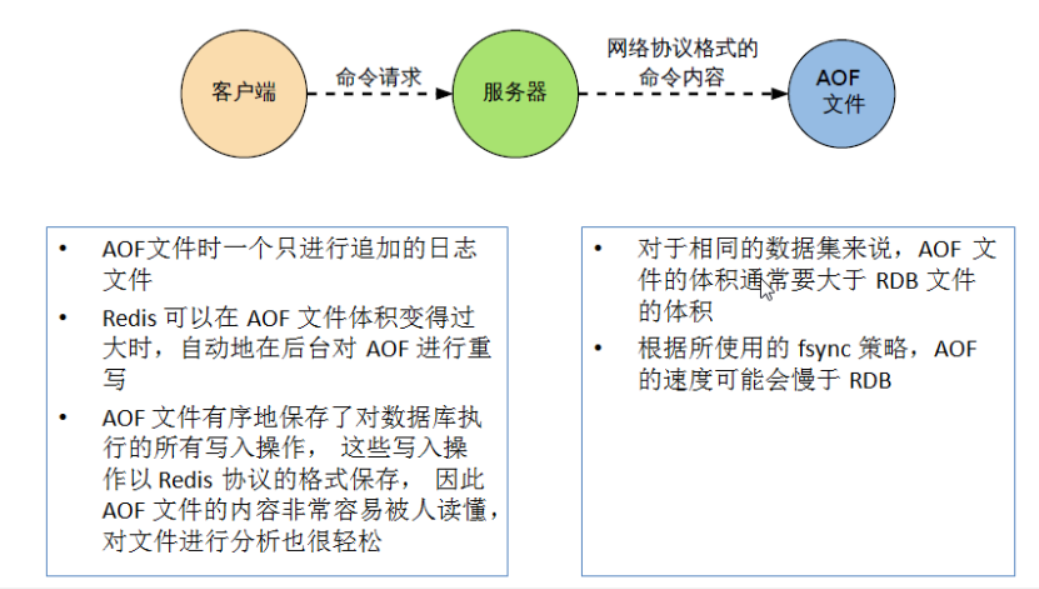
是1分钟内改了1万次，

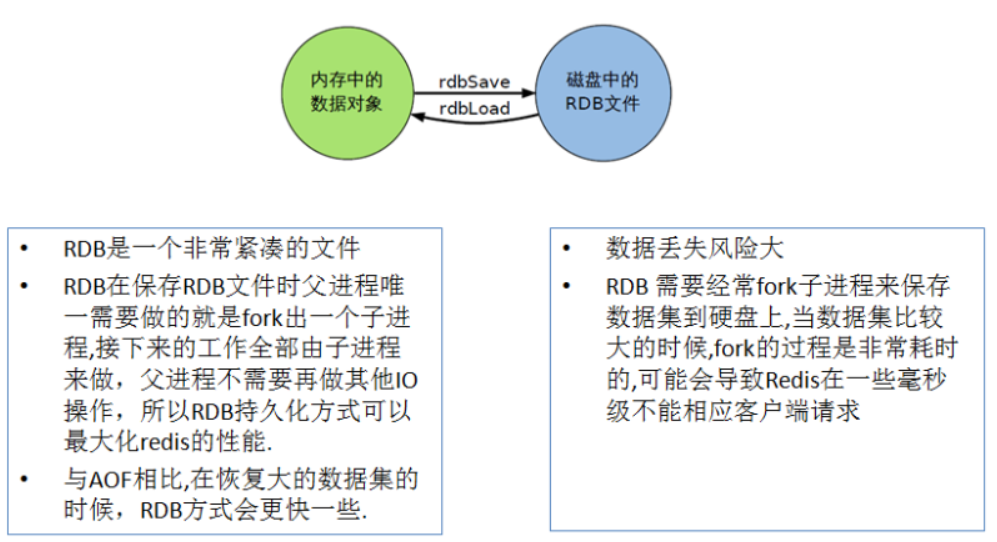
或5分钟内改了10次，

或15分钟内改了1次。

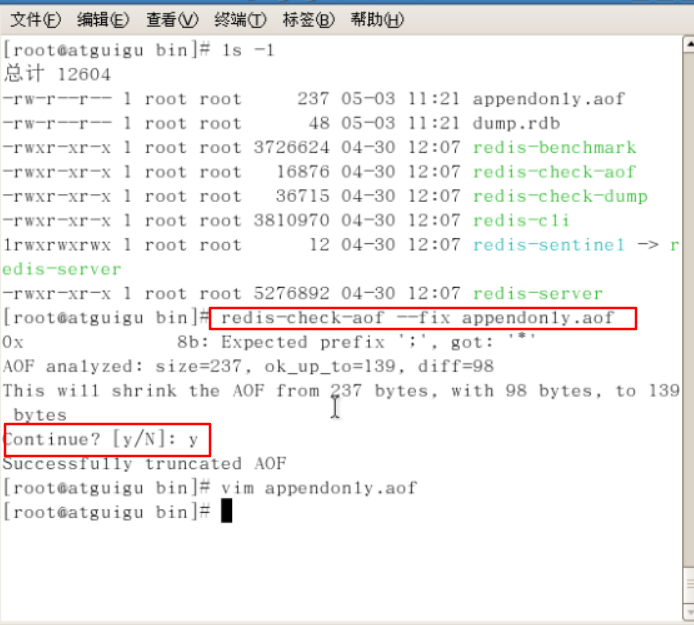
禁用：

如果想禁用RDB持久化的策略，只要不设置任何save指令，或者给save传入一个空字符串参数也可以。





如果appendonly.aof 文件被胡乱的修改，这使用下面的语句进行修复：



# 1.什么是redis持久化：

Rdb 和 aof

总结：

RDB持久化方式能够在指定的时间间隔能对你的数据进行快照存储。

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。

只做缓存：如果你只希望你的数据在服务器运行的时候存在,你也可以不使用任何持久化方式.。

**同时开启两种持久化方式：**

在这种情况下,当redis重启的时候会优先载入AOF文件来恢复原始的数据,

因为在通常情况下AOF文件保存的数据集要比RDB文件保存的数据集要完整.

RDB的数据不实时，同时使用两者时服务器重启也只会找AOF文件。那要不要只使用AOF呢？

作者建议不要，因为RDB更适合用于备份数据库(AOF在不断变化不好备份)，

快速重启，而且不会有AOF可能潜在的bug，留着作为一个万一的手段。

**性能建议：**

因为RDB文件只用作后备用途，建议只在Slave上持久化RDB文件，而且只要15分钟备份一次就够了，只保留save 900 1这条规则。

如果Enalbe AOF，好处是在最恶劣情况下也只会丢失不超过两秒数据，启动脚本较简单只load自己的AOF文件就可以了。代价一是带来了持续的IO，二是AOF rewrite的最后将rewrite过程中产生的新数据写到新文件造成的阻塞几乎是不可避免的。只要硬盘许可，应该尽量减少AOF rewrite的频率，AOF重写的基础大小默认值64M太小了，可以设到5G以上。默认超过原大小100%大小时重写可以改到适当的数值。

如果不Enable AOF ，仅靠Master-Slave Replication 实现高可用性也可以。能省掉一大笔IO也减少了rewrite时带来的系统波动。代价是如果Master/Slave同时倒掉，会丢失十几分钟的数据，启动脚本也要比较两个Master/Slave中的RDB文件，载入较新的那个。新浪微博就选用了这种架构

