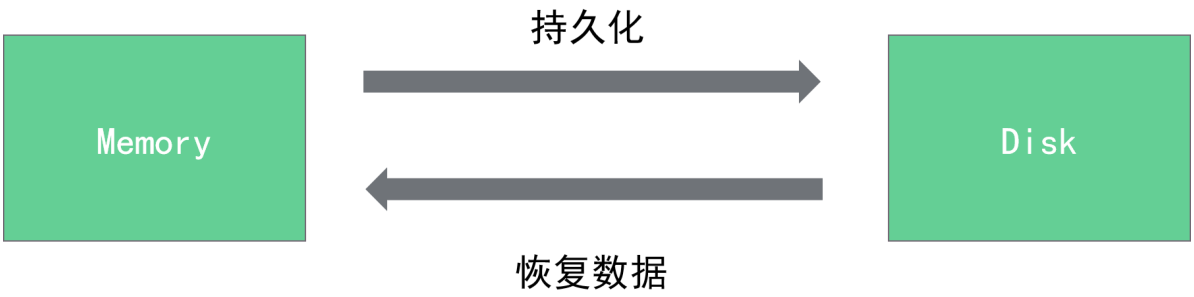
Redis持久化机制之AOF和RDB模式

持久化介绍

Redis的数据都存放在内存中，如果没有配置持久化，redis重启后数据就全丢失了，于是需要开启redis的持久化功能，将数据保存到磁盘上，当redis重启后，可以从磁盘中恢复数据。



RDB 持久化

客户端直接通过命令BGSAVE或者SAVE来创建一个内存快照

BGSAVE 调用fork来创建一个子进程，子进程负责将快照写入磁盘，而父进程仍然继续处理命令。

SAVE 执行SAVE命令过程中，不再响应其他命令。

在redis.conf中调整save配置选项，当在规定的时间内，Redis发生了写操作的个数满足条件会触发发生BGSAVE命令

```
# 900秒之内至少一次写操作
save 900 1
# 300秒之内至少发生10次写操作
save 300 10
# 60秒之内发生至少10000次
save 60 10000
```

优缺点

优点	缺点
对性能影响最小	同步时丢失数据
RDB文件进行数据恢复比使用AOF要快很多	如果数据集非常大且CPU不够强（比如单核CPU），Redis在fork子进程时可能会消耗相对较长的时间，影响Redis对外提供服务的能力。

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化方式能够在指定的时间间隔对你的数据进行快照存储

记录每次服务受到的写

BGREWRITEAOF命令可以触发日志重写或自动重写，废除对同一个Key历史的无用命令，重建当前数据集所需的最短命令序列。

意外中断，如果最后的命令只写了一部分，恢复时则会跳过它，执行后面完整的命令。

开启AOF持久化

```
appendonly yes
```

AOF策略调整

```
#每次有数据修改发生时都会写入AOF文件，非常安全非常慢
appendfsync always
#每秒钟同步一次，该策略为AOF的缺省策略，够快可能会丢失1秒的数据
appendfsync everysec
#不主动fsync,由操作系统决定，更快，更不安全的方法
appendfsync no
```

AOF优点和缺点

优点	缺点
最安全	文件体积大
容灾	性能消耗比RDB高
易读，可修改	数据恢复速度比RDB慢

Redis丢失数据的可能性

持久化丢失的可能

RDB方式

快照产生的策略，天生就不保证数据安全

AOF持久化策略

默认每秒同步一次磁盘，可能会有1秒的数据丢失

每次修改都同步，数据安全可保证，但Redis高性能的特性全无

主从复制丢失的可能

异步复制，存在一定的时间窗口数据丢失

网络、服务器问题，存在一定数据的丢失

总结：持久化和主从都可能出现数据丢失**