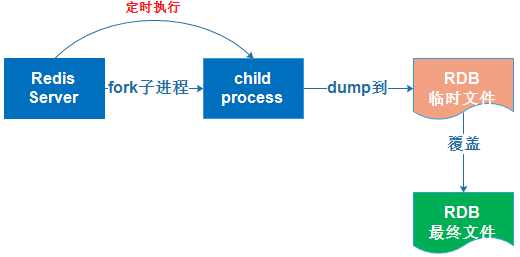
Redis

# Redis持久化RDB和AOF

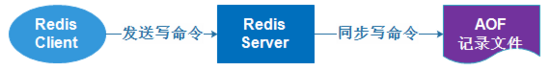
## RDB

RDB持久化是指在指定的时间间隔内将内存中的数据集快照写入磁盘，实际操作过程是fork一个子进程，先将数据集写入临时文件，写入成功后，再替换之前的文件，用二进制压缩存储。



## AOF

AOF持久化以日志的形式记录服务器所处理的每一个写、删除操作，查询操作不会记录，以文本的方式记录，可以打开文件看到详细的操作记录。



## 二者优缺点

RDB存在哪些优势呢？

1). 一旦采用该方式，那么你的整个Redis数据库将只包含一个文件，这对于文件备份而言是非常完美的。比如，你可能打算每个小时归档一次最近24小时的数据，同时还要每天归档一次最近30天的数据。通过这样的备份策略，一旦系统出现灾难性故障，我们可以非常容易的进行恢复。

2). 对于灾难恢复而言，RDB是非常不错的选择。因为我们可以非常轻松的将一个单独的文件压缩后再转移到其它存储介质上。

3). 性能最大化。对于Redis的服务进程而言，在开始持久化时，它唯一需要做的只是fork出子进程，之后再由子进程完成这些持久化的工作，这样就可以极大的避免服务进程执行IO操作了。

4). 相比于AOF机制，如果数据集很大，RDB的启动效率会更高。

RDB又存在哪些劣势呢？

1). 如果你想保证数据的高可用性，即最大限度的避免数据丢失，那么RDB将不是一个很好的选择。因为系统一旦在定时持久化之前出现宕机现象，此前没有来得及写入磁盘的数据都将丢失。

2). 由于RDB是通过fork子进程来协助完成数据持久化工作的，因此，如果当数据集较大时，可能会导致整个服务器停止服务几百毫秒，甚至是1秒钟。

AOF的优势有哪些呢？

1). 该机制可以带来更高的数据安全性，即数据持久性。Redis中提供了3中同步策略，即每秒同步、每修改同步和不同步。事实上，每秒同步也是异步完成的，其效率也是非常高的，所差的是一旦系统出现宕机现象，那么这一秒钟之内修改的数据将会丢失。而每修改同步，我们可以将其视为同步持久化，即每次发生的数据变化都会被立即记录到磁盘中。可以预见，这种方式在效率上是最低的。至于无同步，无需多言，我想大家都能正确的理解它。

2). 由于该机制对日志文件的写入操作采用的是append模式，因此在写入过程中即使出现宕机现象，也不会破坏日志文件中已经存在的内容。然而如果我们本次操作只是写入了一半数据就出现了系统崩溃问题，不用担心，在Redis下一次启动之前，我们可以通过redis-check-aof工具来帮助我们解决数据一致性的问题。

3). 如果日志过大，Redis可以自动启用rewrite机制。即Redis以append模式不断的将修改数据写入到老的磁盘文件中，同时Redis还会创建一个新的文件用于记录此期间有哪些修改命令被执行。因此在进行rewrite切换时可以更好的保证数据安全性。

4). AOF包含一个格式清晰、易于理解的日志文件用于记录所有的修改操作。事实上，我们也可以通过该文件完成数据的重建。

AOF的劣势有哪些呢？

1). 对于相同数量的数据集而言，AOF文件通常要大于RDB文件。RDB 在恢复大数据集时的速度比 AOF 的恢复速度要快。

2). 根据同步策略的不同，AOF在运行效率上往往会慢于RDB。总之，每秒同步策略的效率是比较高的，同步禁用策略的效率和RDB一样高效。

二者选择的标准，就是看系统是愿意牺牲一些性能，换取更高的缓存一致性（aof），还是愿意写操作频繁的时候，不启用备份来换取更高的性能，待手动运行save的时候，再做备份（rdb）。rdb这个就更有些 eventually consistent的意思了。不过生产环境其实更多都是二者结合使用的。

# Redis数据类型及其操作

FlushDB删除一个数据库的数据

FlushALL 删除全部

## [String类型操作](https://www.cnblogs.com/zumengjie/p/12527008.html#a1)

get

set

incr key 递增数字，返回递增结果

incrby key num 递增指定步数返回递增结果

decr key 递减数字返回递减结果

decrby key num 递减指定步数返回递减结果

getset key value取值，如果没有返回null，并且在去set一下，然后后面就会有值了

setnx key value 如果key不存在，才会添加返回1，否则不添加返回0。

append key value 根据key追加value，如果key不存在就创建。返回追加后的长度

strlen key 计算value的长度

mset key value key1 value1 一次设置多个

mget key1 key2 一次得到多个

## Hash类型

hash类型用于存储对象类型，比如存一个person对象，如果一个对象需要频繁的修改某些属性，则可以存hash类型。当然也可以存string，不过需要将string变成json，而每一次修改属性需要将value全部取出

hset key field value 设置key和单个字段

hget key field

hmset key field value field value

mget key field field

hgetall key 得到所有属性值

hdel key field field 删除一个或多个字段

hsetnx key field value 同上

hexists key field 获取字段是否存在

hkeys key 获取key所有的字段

hvals key 获取key的所有字段值

hlen key 获取key的字段数量

hincrby key field increment 某个字段加一

# List类型操作

[左边，右边]

Lpush key value value 返回数量

Lrange key start stop 获取list中的元素，从指定位置开始，指定位置结束，start从0开始，stop为-1标识直到最后一个

Rpush key value value 从左边添加元素

Lpop从左边移除一个元素，并且返回被移除的元素

Rpop key 从右边移除一个元素，并且返回被移除的元素

Llen key 获取总个数

Lrem key count value 删除指定个数的value，比如list中有三个A (lrem list 2 value)则会删除两个A，当count大于0从左边开始删除，当count小于0从右边开始删除，当count=0时删除所有A

Lindex key index 获取指定下标的元素

Lset key index value 修改指定下标的值

Ltrim key start end只保留start 到end 的其他元素删除

Linsert key before|after pivot value 寻找指定value然后在此value的after或者before添加指定的value

Rpoplpush list1 list2 把list1中rpop的元素lpush到list2中

# Set

Set类型元素不可重复无序

Sadd key value value 添加指定元素

Smembers key 获取所有元素

Srem key value 删除指定元素

Sismember key value判断元素是否存在

scard key 获取元素个数

spop key 随机弹出一个元素，主义弹出是返回并且删除

sdiff key1 key2 取差集，在A中有在B中没有

sinter key1 key2取交集，在A中有在B中也有

sunion key1 key2 并集，取AB中所有的元素，但是不重复

# Zset

在set的基础上增加了一个分数