# Redis主从同步

从服务器会主动向主服务器建立连接，然后通过psync命令将主服务器数据同步到本地。

初次同步是完整同步，主返回+FULLRESYNC，主服务器创建并发送RDB文件发送给从服务器，并将同步过程中收到的客户端操作存入缓冲区中。

后续复制是增量同步，主返回+CONTINUE，主服务器根据从服务器的复制偏移量来判定需要将哪些数据同步给从服务器。

如果主从断线之后重连，则判断从服务器的复制偏移量是否在主服务器的复制缓冲区范围内。在范围内是增量同步，不在范围内则全量同步。

当主从服务器命令同步完成，进入命令传播阶段，从服务器会定时向主服务器发送REPLCONF ACK命令用于心跳检测。命令会携带从服务器当前的复制偏移量，主服务器根据该偏移量即可判断是否出现消息传播丢失的情况。

## 设置主从关系

redis通过SLAVEOF命令或者slaveof配置来确定服务器之间的主从关系。

### 关键函数

#### slaveofCommand

入参c：用于接收用户命令的redisClient数据结构。

出参：无。

1. 集群模式不能使用slaveof命令。集群模式下主从关系在创建集群的时候就已经设置好了，后续主服务器如果异常下线，发生主从切换，则是由集群自动选举决定，不需要人为设置。
2. 对于slave no one命令的处理。收到这个命令表示当前从服务器升级为主服务器，此时调用replicationUnsetMaster清除本地保留的master信息以及停止与master之间的通信。
3. 解析命令参数，如果指定的master和本地保存的master相同，则不做任何操作返回+OK。
4. 调用replicationSetMaster设置master信息。

#### replicationSetMaster

入参ip：master的ip。

入参port：master的port。

出参：无。

1. master的ip和port保存在sever结构下的masterhost和masterport字段中。
2. 如果本机之前连结过其他主服务器，则要清除用于连接主服务器的redisClient结构，断开与主服务器的连接，清除缓冲区等。
3. 设置sever的复制状态repl\_state字段为REDIS\_REPL\_CONNECT。主服务器复制偏移量master\_repl\_offset为0。复制连接断开时长repl\_down\_since为0。
4. 该函数并没有执行连接主服务器的操作，只是设置了连接状态。这个状态在replicationCron函数中会触发向主服务器发起连接。

## 主从服务器通信

从服务器主动向主服务器发起连接。replicationCron函数处理了主从服务器之间连接状态，比如连接的发起，超时断开等等。主从服务器之间通信命令有如下几种：

REPLCONF：从服务器向主服务器通告自己的监听端口。

REPLCONF ACK：

PING：

PONG：

PSYNC：从服务器通过该消息向主服务器请求数据同步。会携带主服务器的runid和从服务器复制偏移量。runid用于主服务器判定从服务器之前是否也是与自己连接。如果从服务器之前没有连接过任何主服务器，则携带的runid为?，偏移量为-1。主服务器在应答消息里会携带自己的runid给从服务器保存。

### 关键函数

#### replicationCron

入参：无。

出参：无。

1. 处理连接超时的情况。关闭当前的连接，将repl\_state设置为REDIS\_REPL\_CONNECT。这样在下一次调用replicationCron函数的时候会重新向主服务器发起连接。
2. RDB文件传输超时。关闭传输用的连接，并删除本地临时文件。
3. 已连接上主服务器，但是超时没有收到主服务器发送的数据或者PING消息。释放主服务器。
4. repl\_state == REDIS\_REPL\_CONNECT，尝试连接主服务器，对应到replicationSetMaster说的流程。调用connectWithMaster向主服务器发起连接，创建非阻塞socket，设置fd对应的读写事件。同时将repl\_state设置为REDIS\_REPL\_CONNECTING。
5. 定期向主服务器发送REPLCONF ACK命令。
6. 主服务器定时向所有建立连接的从服务器发送PING命令。
7. 释放所有断线的从服务器。注意这里的REDIS\_REPL\_ONLINE状态，只有当主从之间完成全量同步或增量同步之后，才会设置这个状态。
8. 如果本机已经没有从服务器，释放backlog。

#### replicationFeedSlaves

#### syncWithMaster

入参el：没用。

入参fd：与主服务器通信的fd。

入参privdata：没用。

入参mask：没用。

出参：无。

1. 该函数是从服务器与主服务器发起异步连接的回调函数。相当于建连已经完成，从服务器开始发送命令。
2. repl\_state == REDIS\_REPL\_CONNECTION，在第一次同步之前向主服务器发送一个PING命令。目的是确保主服务器能够正常访问，因为第一次同步是全量同步耗时很长，因此要先确保主服务器能够正常访问。发送完成后修改repl\_state为REDIS\_REPL\_RECEIVE\_PONG，函数返回，等待PONG命令到达。
3. 接收上一个PING命令对应的主服务器应答，可能是PONG也可能是NOAUTH。但就只有这两种情况。
4. 向主服务器进行身份验证，这个操作是同步的。
5. 通过REPLCONF命令向主服务器通告从服务器的监听端口。也是同步操作。
6. 调用slaveTryParitialResynchronization向主服务器发送PSYNC。根据返回值判断如果主服务器不支持PSYNC，则发送SYNC。
7. 创建临时文件用来保存主服务器发送来的RDB文件。设置读取RDB数据的回调函数为readSyncBulkPayload。
8. 修改repl\_state为REDIS\_REPL\_TRANSFER。

#### slaveTryPartialResynchronization

入参fd：主从服务器socket连接的fd。

出参：PSYNC\_FULLRESYNC进行全量同步。PSYNC\_CONTINUE进行增量同步。PSYNC\_NOT\_SUPPORTED主服务器不支持PSYNC命令。

1. 如果缓存的主服务器存在，则使用缓存的主服务器runid和对应的复制偏移量。如果没有，PSYNC ? -1。缓存的主服务器是在该函数内PSYNC成功之后设置的。
2. server的cached\_master字段和master字段实际上指向的是同一块内存。
3. 向主服务器同步发送PSYNC命令。
4. 主服务器应答+FULLRESYNC，表示要全量同步。后面的处理是先将主服务器的runid保存到server的repl\_master\_runid字段。
5. 主服务器应答+CONTINUE，表示要增量同步。将缓存的master设置为当前master。但实际返回到外层函数之后并没有做任何处理。因为此时从服务器只需要等待主服务器将自己缺失的数据发来即可。
6. 主服务器应答-ERR，表示不支持PSYNC命令。
7. 收到上面3个应答都会清除当前缓存的master，即cached\_master。

#### readSyncBulkPayload

入参el：没用。

入参fd：主从服务器socket连接的fd。

入参privdata：没用。

入参mask：没用。

1. 如果不知道RDB的大小，那么先读取一次数据，从中解析出RDB文件的大小。
2. 接下来就是读取RDB文件的内容，事件驱动会不断的调用这个函数直到RDB数据传输完毕。
3. 读取的过程中会定期将数据写入磁盘，防止数据过大。
4. RDB文件读取完成之后，先清空当前数据库中的所有数据，然后调用rdbLoad来解析RDB文件。这个过程在《Redis持久化》中说明。
5. 创建一个client赋值给server的master字段。这个client就用来处理主服务器后续同步过来的各个命令。然后设置repl\_state = REDIS\_REPL\_CONNECTED，设置主服务器的runid等等。

#### syncCommand

入参：接收命令的redisClient数据结构。

出参：无。

该函数时SYNC和PSYNC命令对应的处理函数，两个命令传入的参数数量不一样，PSYNC命令传入3个参数。这里主要介绍PSYNC命令的流程。

1. 判断命令为psync，调用masterTryPartialResynchronization尝试进行增量同步。
2. 如果不能增量同步，就进行全量同步。如果当前有BGSAVE的命令在执行，判断主服务器的所有从服务器是不是至少有一个在等待主服务器完成BGSAVE操作。如果有的话，说明当前的从服务器也可以使用这个BGSAVE所生成的RDB文件。如果没有只能等待下一次的BGSAVE。
3. 如果当前没有BGSAVE在执行，那么直接开始一个新的BGSAVE操作。
4. 将redisClient结构加入server的slaves链表中。如果是第一个slave则初始化backlog。这里可以看出在收到PSYNC命令主服务器才会记录从服务器，而不是在收到REPLCONF命令时候记录。

#### replconfCommand