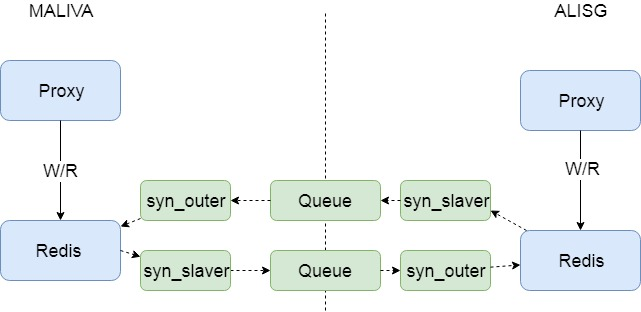
**Redis 异地多机房同步长期方案**

**背景**

开发 Redis服务的异地双机房同步长期方案, 可扩展到异地多机房。取名为长期方案是用来区别现有的当前的基于proxy双写的同步方案（详见[文档](https://docs.bytedance.net/doc/Tsy1Tl0NhhZ6CD0dceC4pg)）

**架构**

* 基于Alchemy架构（Alchemy架构详见[文档](https://docs.bytedance.net/doc/rvcaRgyo2p7Bi4Xm8xTVPf)）。初期适配目前的redis cluster的架构，后续Alchemy架构稳定后支持Alchemy架构。
* 每个机房是一个独立的集群，每个分片双副本部署(一主一从)；
* 实现弱网复制：对aof协议改造，增加命令的唯一自增id信息，支持内存缓冲区查找失败时从文件中读取增量数据；（redis 内核基于阿里云[Apsaracache改造](https://github.com/alibaba/ApsaraCache)）
* 命令的唯一性保证：目标实例中消费同步过来的命令，并保存当前已消费消息的命令id。在消费命令之前，检查当前id（id1）是否大于当前实例已消费的id（id2），如果id1 <= id2，将该命令丢弃不处理；
* 防回环：在aofbinlog中加入src serverid字段标记来源；syn\_slaver 对同步的数据判断是否本server 产生的命令，如果不是则丢弃不同步；



组件：

syn\_slaver: 作为伪从，获取所有aof binlog，过滤掉非 source redis 产生的命令（每个redis实例部署一个）；

syn\_outer 为集群共用，在 TCE上部署（每个集群一个）；

**增量同步**

aof 协议中加入 opid 字段，syn\_slaver 保存当前处理的opid；当数据中断后，slave通过请求指定的opid，传输 opid 之后的数据，不再全量同步；

syn\_slaver 连接集群中的 slave实例进行同步；当 redis master 挂掉后，slave 被提升为主，重连到 新slave 上指定 offset 继续同步（如果 request offset > slave offset, 按slave offset 继续同步）；

**同步正确性保证**

oplog按产生顺序同步到对端。

回环控制，解决双向同步的循环同步问题。

在对端Redis实例上，所有oplog 仅被执行1次。

**数据一致性**

flushdb/flushall命令不会同步到另一端，如需清空实例，需在两端都调用flush命令。只在一端调用flush，会造成两端数据不一致。

同时操作一个key，可能会造成数据不一致（主要是数据交换和写入时序不一致）

**数据不一致的处理**

数据不一致后的处理方式分为：全同步和冲突检测；

**全同步**

以一方为准，触发全同步；针对已存在key，全同步的策略：

* string，直接覆盖
* set，zset、hash等带sub field的数据结构：可选配置直接使用新集合覆盖或者解析子key后覆盖 （默认解析子key覆盖）

**冲突检测（初期不实现）**

对于复杂场景（eg：针对不同的key的处理方法不同），提供全量的冲突检测

方法：

在同一时间点，触发集群所有rdb dump。之后对两集群的 rdb数据进行分析，识别不一致的key；（双机房同城灾备也有此需求）

不一致的key的处理：

* 提供给业务方，业务来修复
* 确定使用一边数据：从 A 集群读出来，写到 B 集群（开发工具实现）