阿里云

云数据库

灾备多活产品文档

文档版本：20170717

为了无法计算的价值 | 阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

目录

1. 容灾技术概述

数据库灾备技术是指在一个数据中心发生故障或灾难的情况下，其他正常运行的数据中心可以对数据库业务实现接管，达到互为备份的效果。完备的灾备技术可以实现用户的“故障无感知”。

现在的数据中心承载的业务越来越重要，引入有效的灾备技术，能减少数据中心发生故障时带来的损失。本文主要介绍数据库热备及双活的介绍。

1.1. 热备系统

数据库热备技术是在主备数据中心的基础上实现的，前提是拥有两个一主一备的数据中心。备用数据中心为热备部署，应用业务由主用数据中心响应，当主用数据中心出现故障造成该业务不可用时，需要在规定的 RTO(Recover Time Objective，即故障发生后，信息系统从停顿到恢复正常的时间要求)时间以内，实现数据中心的整体切换。在具体实现上，主备数据中心的两套业务系统网络配置完全一样，备用数据中心路由平时不对外发布。当实现主备数据中心切换时，需要断开主用数据中心路由链路，并连接备用数据中心路由链路，保证同一时间只有一个数据中心在线。

1.2. 双活系统

通过双活技术可以实现主备数据中心均对外提供服务，正常工作时两个数据中心的业务可根据权重做负载分担，没有主备之分，分别响应一部分用户。负载分担权重可以是按地域、数据中心服务能力或对外带宽划分。当其中一个数据中心出现故障时，另一数据中心将承担所有业务。多活的最大特点是不会造成数据中心的资源浪费。数据中心都承载应用业务运行，不至于出现像热备、暖备主用数据中心几乎满载运行，而备用数据中心却很空闲的情况。

2. 系统架构

搭建灾备系统，最重要的环节在于数据库层面，一套稳定的数据库同步技术是至关重要的，阿里云 ApsaraDB 团队旨在为用户提供稳定安全的数据库环境，针对大企业客户的数据库安全需求，我们推出了数据库灾备及数据库双活的解决方案。

容灾系统主要包含以下模块：

- 系统采集模块

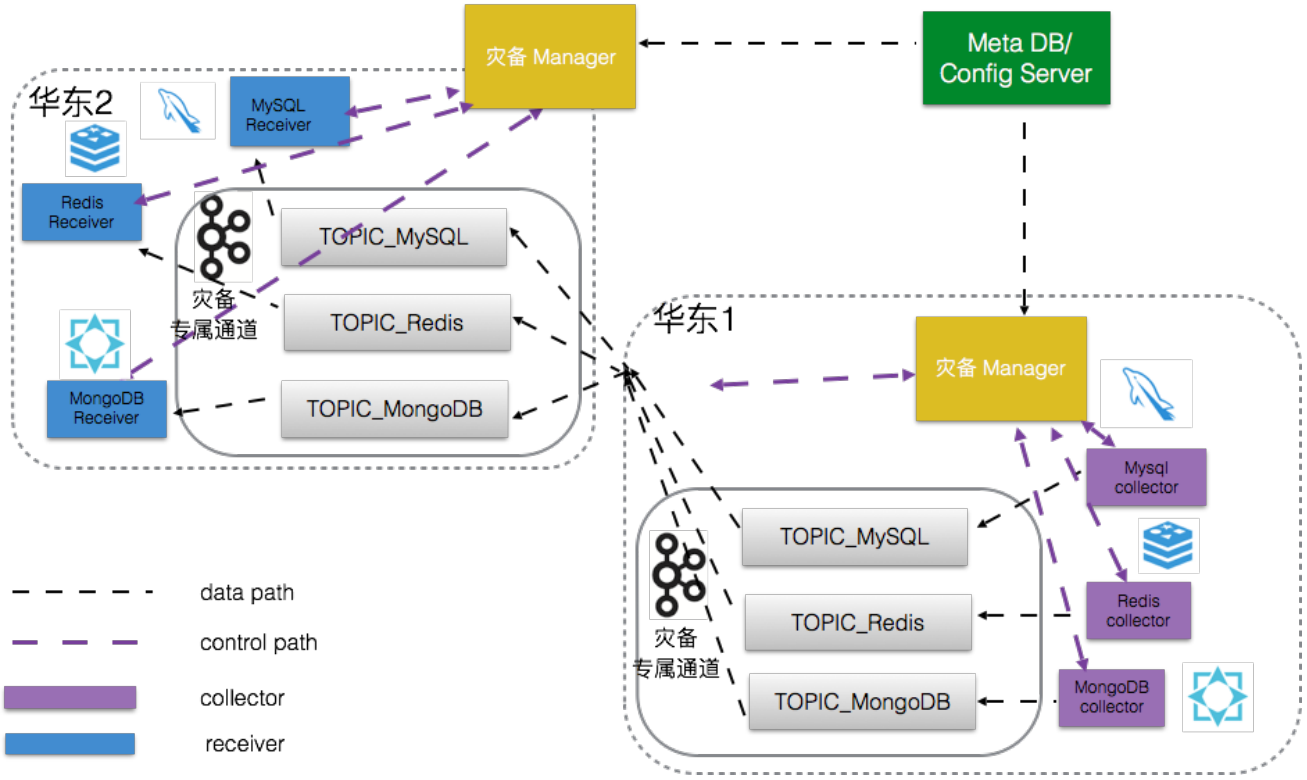
  系统采集模块感知具体的数据库协议，采集 MySQL、Redis 或 MongoDB 的增量日志，并且由灾备 Manager 模块用于监测系统采集模块的健壮及稳定性。

- 数据通道模块

  数据通道模块提供双机高可靠的集群搭建，用于在跨地域及跨机房间传输系统采集模块采集到的增量日志，同时保证数据的顺序性。数据通道模块提供相应的数据持久化能力，可在网络异常修复后，具备快速恢复数据同步的能力。

- 数据回放模块

  目的端的数据回放模块感知具体的数据库协议，用于接收数据通道模块传递的增量数据，根据相应的时间戳顺序，根据相应的协议回放到 MySQL、Redis 或 MongoDB 实例中。

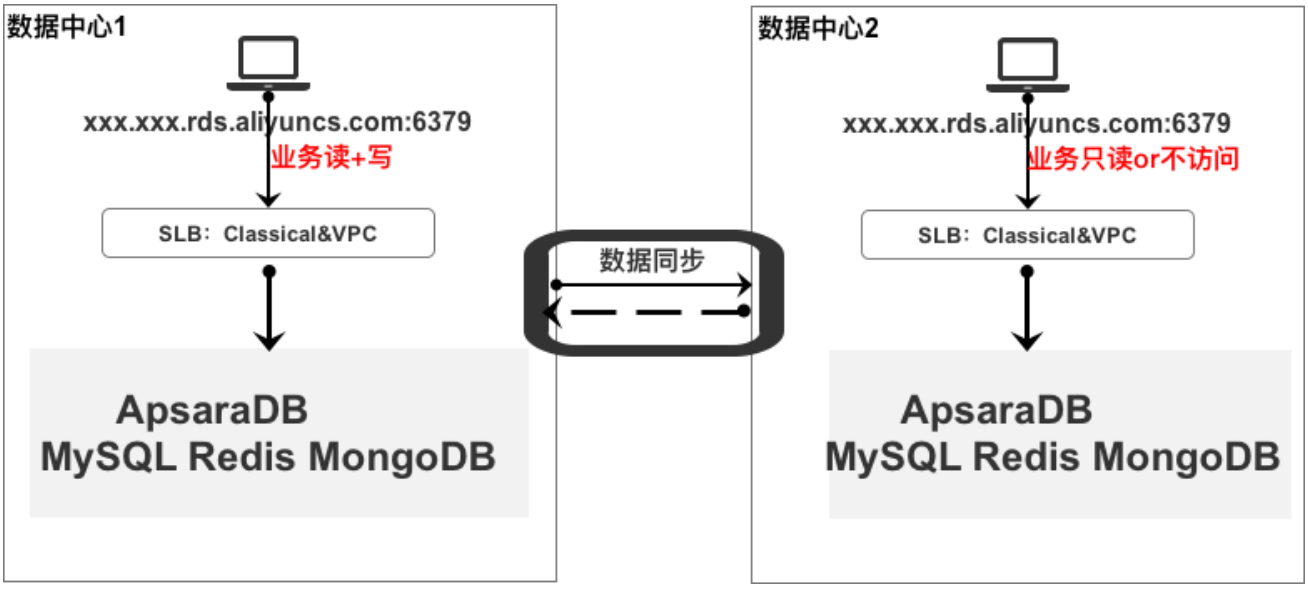


图示模块说明：

- Collector：系统采集模块，负责采集 MySQL、Redis 或 MongoDB 等数据库的增量日志；
- Receiver：数据回放模块，负责回放具体的数据库协议到 MySQL、Redis 或 MongoDB 中；
- 灾备 Manager：Manage 用于监控 Collector 及 Receiver，当其出现业务 down 机的时候可以迅速拉起；
- ConfigServer：用于进行整体灾备系统的配置服务模块，采用双机高可用架构。

2.1. 热备方案

针对数据库的灾备实现，ApsaraDB 团队输出了数据库灾备解决方案，用户通过控制台可以在已有数据库实例上生成一个灾备实例，系统会自动为两个实例创建同步关系，保持数据库的更改可以同步到异地的数据库中。



ApsaraDB 的数据库灾备解决方案帮助用户实现数据底层链路的同步问题，在业务层面有如下限制：

1 主数据中心的数据库支持读写操作，备数据中心的数据库不提供访问或者只提供读请求。

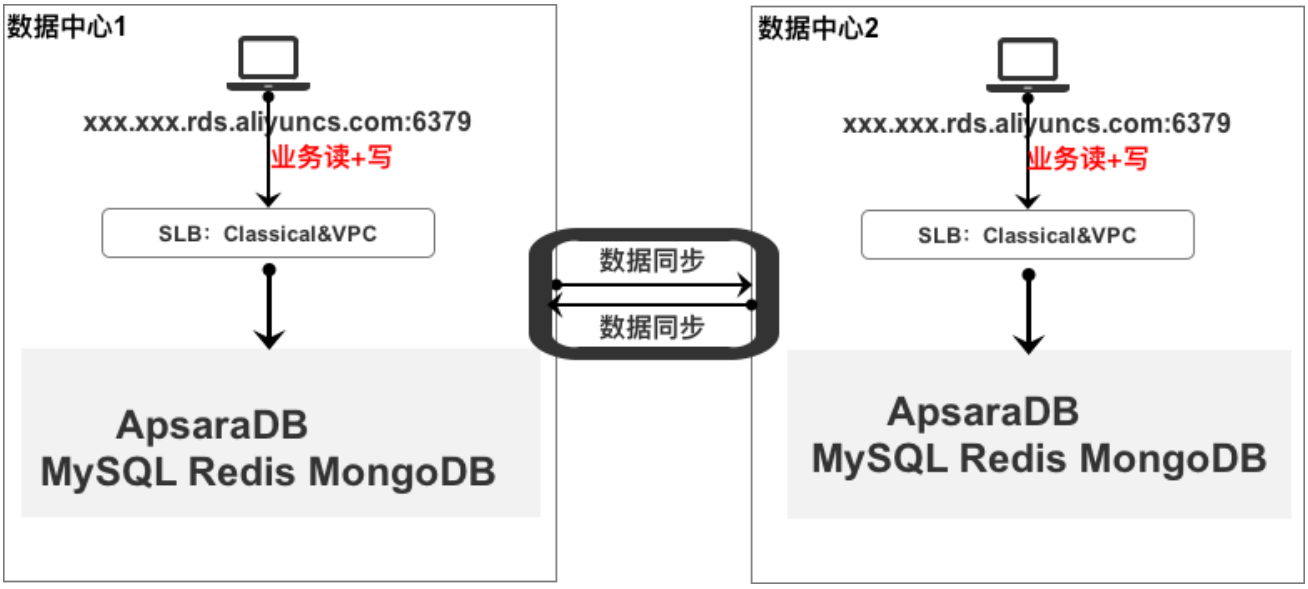
2 当业务判断主数据中心出现故障或者实例不可访问，且确认备数据中心的实例没有

同步延迟后，自动将所有读写请求全部切换至备数据中心。

3 主数据中心的数据库恢复后，不提供访问或者只提供读请求，仅充当灾备实例。

2.2. 双活方案

针对数据库的双活实现，ApsaraDB 团队输出了数据库双活解决方案，用户通过控制台可以在已有数据库实例上生成一个双活实例，系统会自动为两个实例创建同步关系，保持数据库的更改可以同步到异地的数据库中。



ApsaraDB 的数据库双活解决方案帮助用户实现数据底层链路的同步问题，可以保证同一条更新记录不会循环复制，在业务层面有如下限制：

1 业务系统需要控制好数据分区策略，保证单元化写入数据库的数据不会重复。可以根据业务，或者地域的用户分群保障将数据分配到不同的数据中心进行写入或更新。保障两个数据中心不会对同一条记录同时进行写操作。

2 ApsaraDB 数据库双活解决方案可以保持双向同步，保障每个数据中心都是全量数据。当业务判断其中有一个数据中心或实例故障，且确认没有同步延迟之后，自动将所有读写请求转发到另一个数据中心。

3 当故障的数据中心业务恢复后，ApsaraDB 数据库双活系统会自动将故障这段时间的增量日志同步到之前故障的实例上，确认没有数据延迟后，业务可以进行读写切换回到最初的双活模式。

3. 功能特性

用户可以根据 ApsaraDB 构建灾备或者双活的解决方案，我们在云上灾备的控制台支持初始化、监控、管理和操作的基本功能，主要针对两个实例复制关系的管理。

针对单个实例的管理（变配，参数调整，监控，报警）等功能继续在各自的产品控制台集成。

3.1. 一键操作

用户需要构建灾备或者双活的解决方案的时候，我们支持一键的初始化操作。基于一个源实例（ApsaraDB for MySQL/ ApsaraDB for Redis/ ApsaraDB for MongoDB）可以为您初始化好一个灾备实例，用户可以根据自己的业务情况，作为灾备实例或者多活的解决方案。

3.2. 监控报警

目前灾备通道的报警正在对接中，预计 8 月初支持。

具体的实例报警信息可以直接登录到 RDS 及 MongoDB 控制台设置即可。

数据库的灾备的通道提供 6 项监控指标，分别为：

- 同步流量、同步速度、同步延迟、同步状态、同步中最大语句、同步中平均语句；
- 同步流量：监控记录同步方向中，每秒的同步流量大小，单位 MB/S；
- 同步速度：监控记录同步方向中，每秒的同步语句个数，单位 TPS；
- 同步延迟：监控记录同步方向中，目的端与源端的数据延迟差，单位 ms；
- 同步状态：监控记录同步方向中，同步的状态，目前只有同步正常及异常的两种状态值；
- 同步中的最大语句：监控记录同步方向中，同步通道中最大语句的大小，用户可基于该监控值判断同步延迟的原因；
- 同步中平均语句：监控记录同步方向中，同步通道中平均语句的大小，用户可基于该监控值判断同步延迟的原因。

以上监控均提供源实例与灾备实例的双向通道监控值，用户可根据业务情况部署为灾备或者双活解决方案来有效使用监控，灾备实例只关注单向的同步监控即可。

3.3. 灾备通道管理

灾备通道的管理主要提供的是灾备实例关系展示，基本信息展示、灾备监控、报警等，构建灾备或者多活的实例管理还是继续在 RDS or MongoDB 控制台进行管理，如实例的监控、报警，规格的变更配置等。

4. 性能指标

4.1. 同步延迟

ApsaraDB 灾备或者双活解决方案，为用户提供数据库同步基础设施，整体的同步延时在 5s 以内。

5. 开通与计费

5.1. 开通方式

用户需要构建灾备或者双活的解决方案的时候，我们支持一键的初始化操作。基于一个源实例（ApsaraDB for MySQL/ ApsaraDB for Redis/ ApsaraDB for MongoDB）可以为您初始化好一个灾备实例，用户可以根据自己的业务情况，作为灾备实例或者多活的解决方案。

在初始化操作前，需要填写一些初始表单。

1 需要选择云数据库 MySQL、Redis 或者 MongoDB（空实例或者在线运行系统均可以）。

2 选择构建灾备或者双活的地域或者可用区。

3 选择灾备实例的网络类型、付费类型等信息，确认下单进行初始化创建。

系统收到请求后，会进行灾备或者双活系统的初始化操作，系统会分几步操作为您将灾备实例创建好，并建立好数据同步关系，以下步骤是系统的操作流程，对用户透明。

步骤1 系统的准备工作，数据同步通道的初始化流程操作。耗时较少，一般为 1-2 分钟。

步骤2 对源实例进行一次全量备份操作，该操作的耗时时间依赖于数据量大小而定，如果是一个接近于空库的实例，一般需要 7 分钟左右的时间。

步骤3 依赖于上一步生成的备份文件，克隆一个新的实例（实例信息依赖于用户表单提交所选择的地域、网络信息、付费类型等），该步骤的耗时同样依赖于源实例的数据量大小，如果源实例是一个接近于空库的实例，该步骤一般需要 20 分钟左右。

步骤4 数据同步通道建立及追赶增量同步，该步骤主要是建立同步关系，并将在备份文件生成后源实例的增量数据追加，该步骤主要依赖于源实例的 TPS，如果源实例是一个未上线的系统，该步骤一般需要 2 分钟左右。

完成以上四步操作之后，灾备实例就已经为您搭建完毕，并且可以开始部署您的应用系统，构建灾备或者双活的数据库解决方案了。

5.2. 计费说明

用户只需要对两个搭建灾备或者多活的实例进行付费。

灾备或者多活进行数据传输目前对用户完全免费。

例如用户在华东 1 及华东 2 各有一个云数据库 MongoDB 配置为 1 核 2G，500G 硬盘。那么用户只需要付出这两个配置的价格即可。

6. 使用说明

6.1. 使用限制

已经搭建灾备或者多活的实例以下操作需要格外谨慎，由于同步挂载的原因，请尽量减少以下操作，如需一定要做以下操作，请务必与阿里云工程师联系，加强监控，采取相应的方法后，再进行操作。避免同步链路出现问题。

注意：目前的实现还并没有在控制台上直接禁止以下操作，所以请务必谨慎。

MySQL 限制的功能如下：

- 变更实例规格
- 切换实例的主备关系
- 迁移实例可用区
- 修改实例访问模式
- 基于备份集恢复实例
- 数据导入
- 切换网络类型

Redis 限制的功能如下：

- 变更实例规格
- 修改实例访问模式
- 基于备份集恢复实例
- 数据导入
- 切换网络类型
- 实例类型变配（如主从版->集群版）

MongoDB 限制的功能如下：

- 基于备份集恢复实例
- 数据导入
- 切换网络类型

- 变更实例配置

6.2. 产品支持情况

6.2.1. 支持的产品系列

支持产品	支持的产品形态
ApsaraDB for MySQL	双机热备版
	三节点企业版
ApsaraDB for Redis	标准版-双节点
ApsaraDB for MongoDB	三节点副本集版

6.2.2. 支持的地域

MySQL

地域	可用区
华东 1	华东 1 可用区 B
	华东 1 可用区 D
	华东 1 可用区 E
	多可用区 1（可用区 B+可用区 C）
	多可用区 2（可用区 E+可用区 F）
华东 2	华东 2 可用区 A
	华东 2 可用区 B
	多可用区 1（可用区 B+可用区 C）
华北 2	华北 2 可用区 A
	华北 2 可用区 B
	华北 2 可用区 C
	华北 2 可用区 D
	多可用区 1（可用区 A+可用区 B）

Redis

地域	可用区
华东 1	华东 1 可用区 B
	华东 1 可用区 D

	华东 1 可用区 E
	华东 1 可用区 F
华东 2	华东 2 可用区 A
	华东 2 可用区 B
华北 2	华北 2 可用区 A
	华北 2 可用区 B
	华北 2 可用区 C
	华北 2 可用区 D

MongoDB

地域	可用区
华东 1	华东 1 可用区 B
华东 2	华东 2 可用区 B
华北 2	华北 2 可用区 A
	华北 2 可用区 B
	华北 2 可用区 C

6.2.3. 架构方案

- 支持跨地域的两个实例实现灾备或者多活的解决方案。
- 支持同地域的不同可用区的两个实例实现灾备或者多活的解决方案。
- 支持同地域相同可用区的两个实例实现灾备或者多活的解决方案。

6.2.4. 数据源能力

- 当前支持 MySQL、Redis、MongoDB。
- 支持 MySQL 到 MySQL 的数据同步。
- 支持 Redis 到 Redis 的数据同步。
- 支持 MongoDB 到 MongoDB 数据同步。
- 不支持异构数据源的同步。

6.3. 避免数据冲突

构建健壮的灾备系统或者双活系统，一定需要业务上与数据库系统做好配合。ApsaraDB 灾备系统会在系统层面保证数据流通的稳定性，但是整体数据库防止冲突的设计要靠用户的业务系统一起配合。本节主要介绍双活的设计介绍业务设计。灾备场景在业务切流时候避免冲突的方式和双活是一样的。

6.3.1. MySQL

无论什么时候切流，用户都可以插入新的（在原实例也没有写入过的）数据，而对原实例数据的更新、删除操作选择禁止是最安全的。如果期望做更新、删除操作，同时避免原实例恢复后的数据冲突，业务需要通过当前通道状态和自身业务确认不会对没有同步的数据做更新、删除操作。

以下举例常见数据冲突场景及解决办法。

6.3.1.1. 使用过程中避免数据冲突

1 两个实例在使用过程中分别写没有外键依赖的不同的库和表。

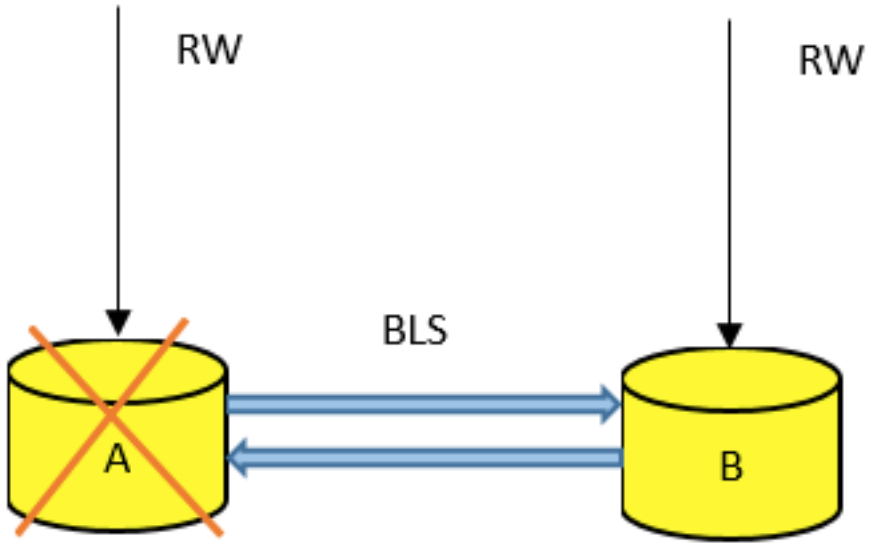
例如：两个业务使用不同的库和表，按照业务来区分到实例的路由。

2 两个实例在业务使用过程中如果需要写一个表，可以对主键中部分列添加路由规则，确保主键、唯一键同一个值的增删改操作都落在同一个实例上。

例如：在电商类似业务中按照用户的 ID 做 range 或者 hash 划分，使同一个用户的数据更新都落到一个单元。

6.3.1.2. 实例不可用，业务切流时候如何避免数据冲突

当实例不可用的时候，例如图中的 A 实例不可服务，这个时候通道中可能依然会有数据没有应用完，A 实例的部分 binlog 也可能还未写入到通道中。如果业务在 B 实例对原本路由到 A 的数据做了操作，很可能在 A 实例恢复后就会报出大量数据冲突。就会费很大的精力去恢复数据。



如何避免操作

1 单元数据禁止更新。

切流后，避免 A 实例恢复后数据冲突的最安全方法是：单元数据禁止更新。禁止对原本写入 A 实例的数据做更新、删除操作，只能够插入新数据（也没有在 A 实例写入过的数据）。

例如：A、B 两个实例负责存储订单信息，不同的用户订单路由到 A、B 两个实例。当 A 实例不可服务的时候，业务将 A 实例上用户的相关请求转到 B 实例。但是，这个时候禁止更新和删除操作，不去改变原有订单状态，只能够新建订单。当 A 实例恢复的时候，再切流会 A 实例，完整的恢复业务。

2 确保落后数据不会更新，完整切流。

业务方在确保灾备实例落后主实例数据量很小，且近期修改过的数据不会再被更新的情况下，可以等通道数据应用完后，做完整的切流，将 A 实例的写操作切换到 B 实例。

6.3.1.3. 当实例可用，只是客户无法访问的时候,如何避免冲突

这种情况下，确保无法访问的时间点前的数据都已经同步到另一个实例，进行相应验证后，即可完整切流。

6.3.2. Redis

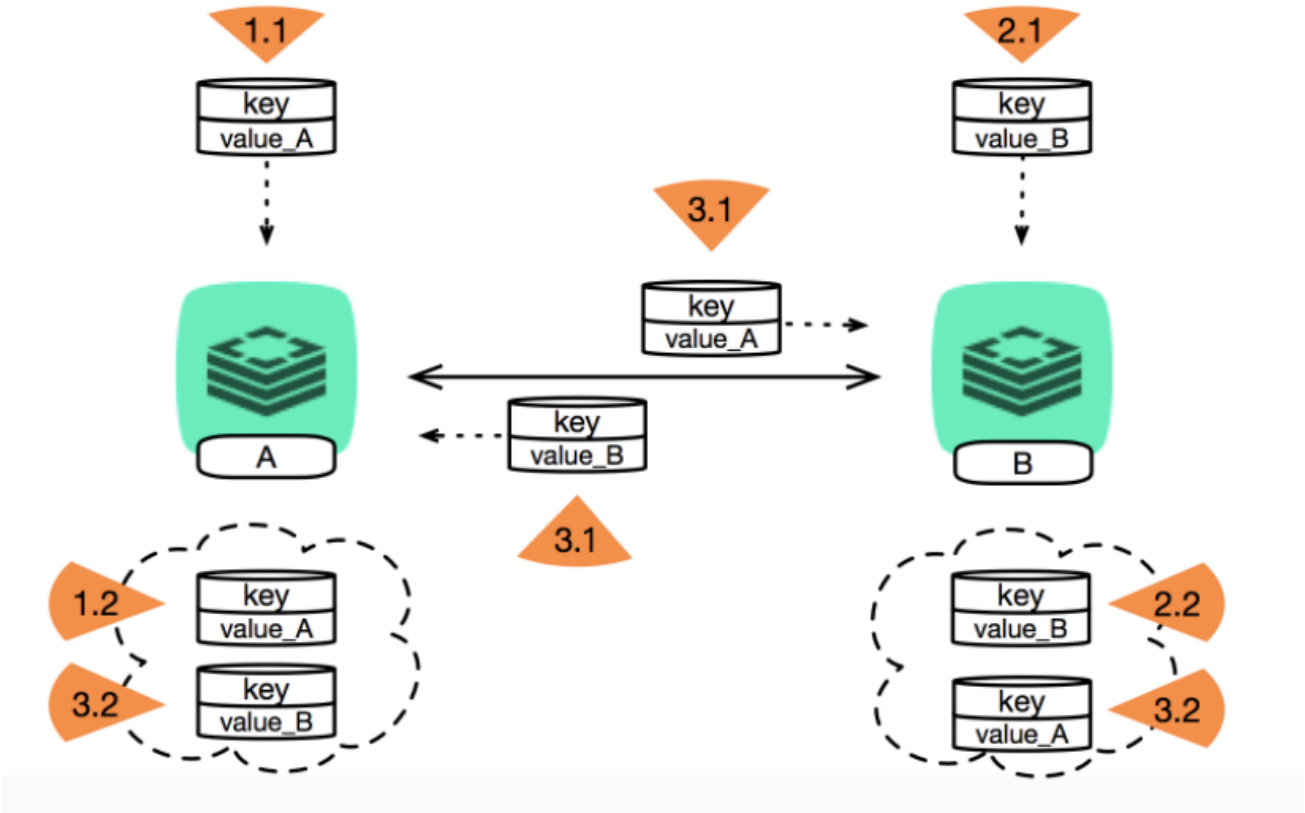
Redis 在数据库同步中，不会存在由于一个 key 的数据冲突而造成整个数据同步断掉的场

景，由于数据的冲突会造成同步双方的数据不一致问题。

1.业务上，请务必要避免在两端同时写入一个 key，两端同时写入一个 key，会造成两端数据不一致，下面会有举例说明

2.flushdb/flushall 命令不会同步到另一端，需要清空实例的话，要在两端都调用 flush 命令，只在一端 flush，会造成两端数据不一致

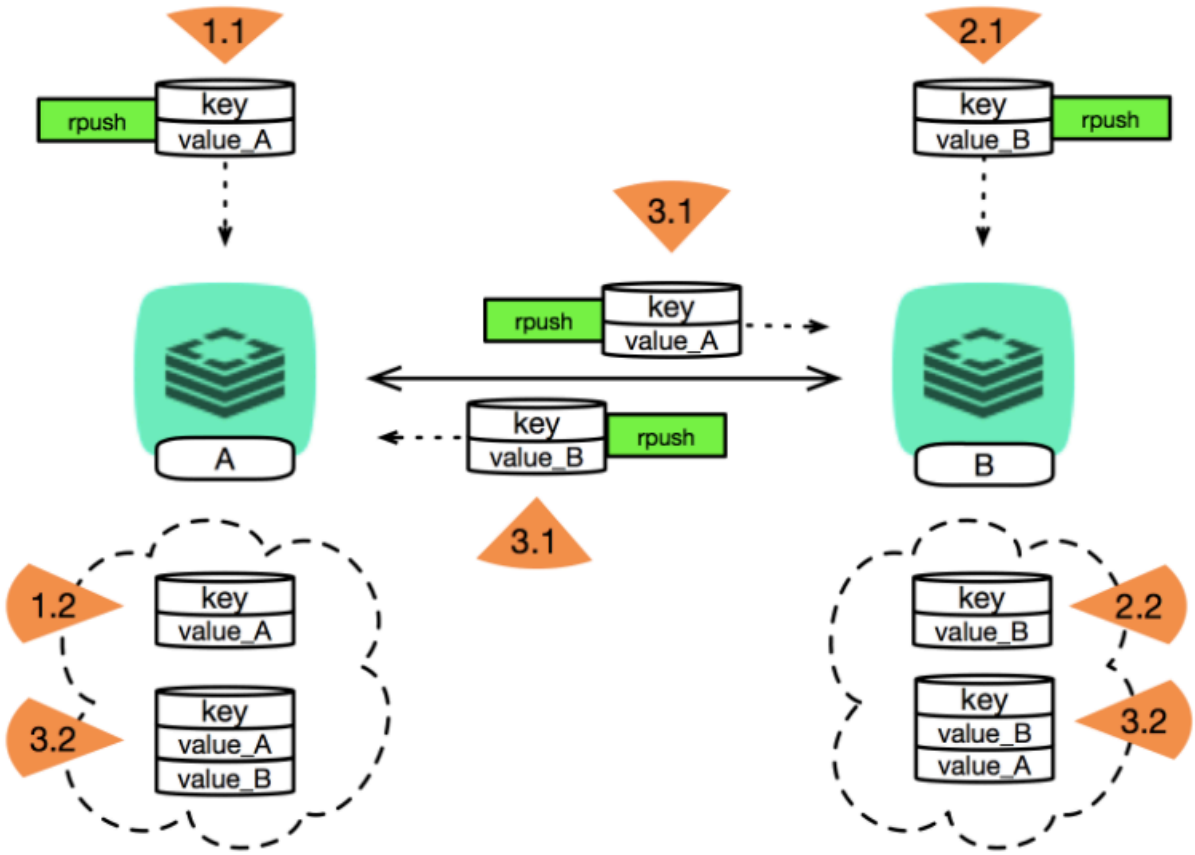
6.3.2.1. 两端同时写入一个 key，造成数据不一致



- 1.1 时刻 A 收到(key->value_A)，在 1.2 时刻 A 记录 db(key->value_A)
- 2.1 时刻 B 收到(key->value_B)，在 2.2 时刻 B 记录 db(key->value_B)
- 3.1 时刻 A 向 B 同步(key->value_A)，同时 B 向 A 同步(key->value_B)，在 3.2 时刻 A 记录(key->value_B), B 记录(key->value_A)

此时，A 和 B 两个数据库的 value 交换了

6.3.2.2. 乱序/丢失数据

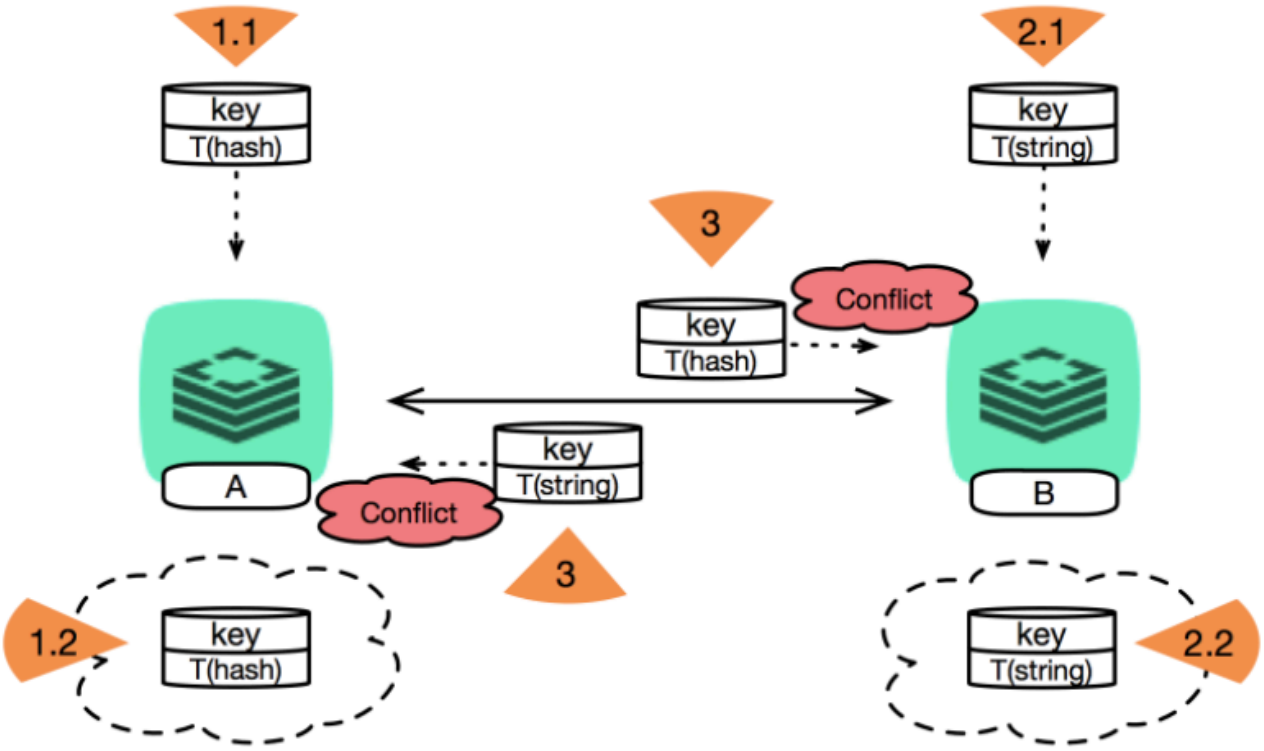


- 1.1 时刻 A 收到(rpush key->value_A)，在 1.2 时刻 A 记录 db(key->value_A)
- 2.1 时刻 A 收到(rpush key->value_B)，在 2.2 时刻 B 记录 db(key->value_B)
- 3.1 时刻 A 向 B 同步(rpush key->value_A)，同时 B 向 A 同步(rpush key->value_B)
- 3.2 时刻 A 记录(rpush key->value_A,value_B), B 记录(rpush key->value_B,value_A)

  当操作类型依赖 value 原始值时，在*交换*的情况下可能会出现乱序甚至丢失的情况。

  类似的操作还有 incr 系列, lpush, lpop, append, sort(store), del, hdel 等命令。

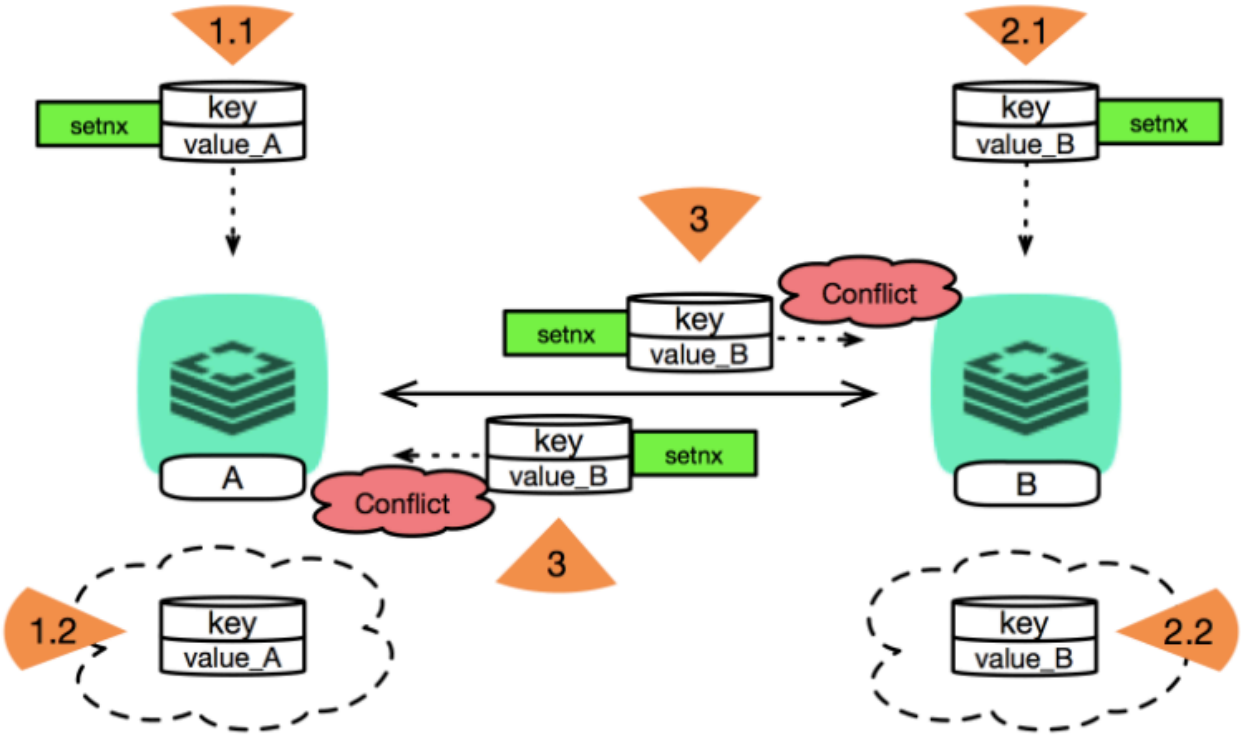
6.3.2.3. 类型不一致



- 1.1 时刻 A 收到(key->T(hash)), 1.2 时刻 A 记录(key->T(hash))
- 2.1 时刻 B 收到(key->T(string)), 2.2 时刻 B 记录(key->T(string))

  之后时刻，A 向 B 同步(key->T(hash))，或者 B 向 A 同步(key->T(string))，都将形成冲突

6.3.2.4. 写入条件不满足



- 1.1 时刻 A 收到(setnx key->value_A), 1.2 时刻记录(key->value_A)
- 2.1 时刻 A 收到(setnx key->value_B), 2.2 时刻记录(key->value_B)

  之后时刻，A 向 B 同步(setnx key->value_A)，或者 B 向 A 同步(setnx key->value_B)，都将形成冲突

  同样可能造成问题的还有 set(nx | xx), hsetex 等

6.3.3. MongoDB

对 Oplog 执行插入、更新、删除操作时，如果发生了主键、唯一键的冲突异常，则认为是数据冲突。

冲突可能存在以下情况：

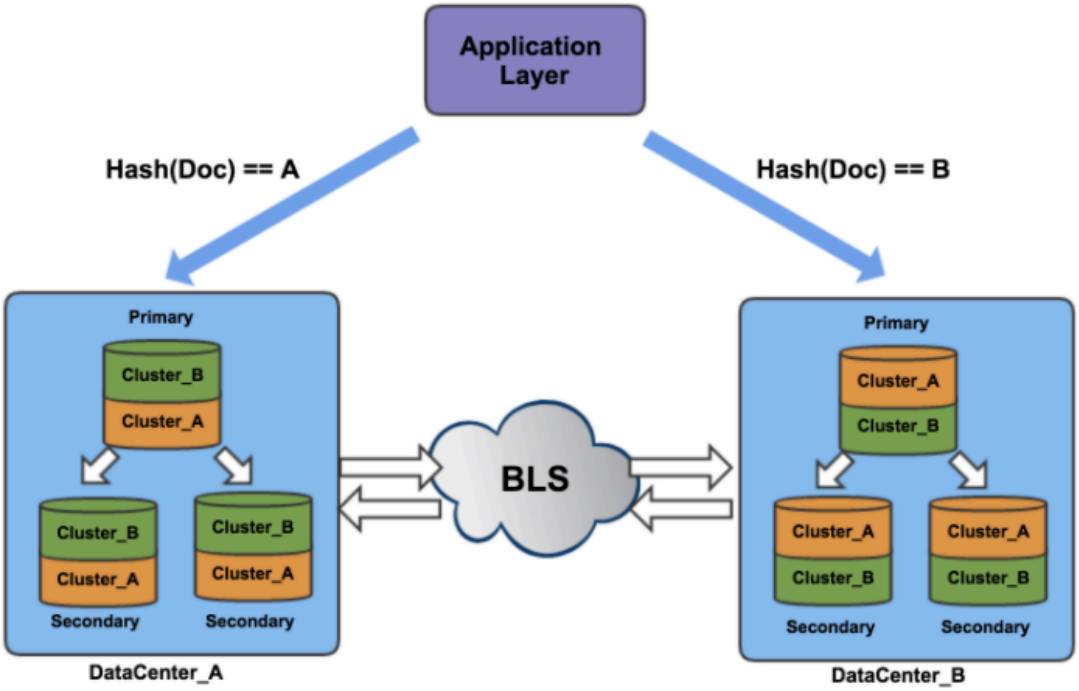
1 源实例读出的插入、更新操作，由于目标实例的路由策略、切换策略导致冲突；

2 源实例读出的插入、更新操作，由于目标实例 schema、index 不同，导致冲突；

3 源实例读出的插入、更新操作，由于源和目标实例均为定义唯一索引，导致潜在的冲突（原本应该一条数据，出现两条）。

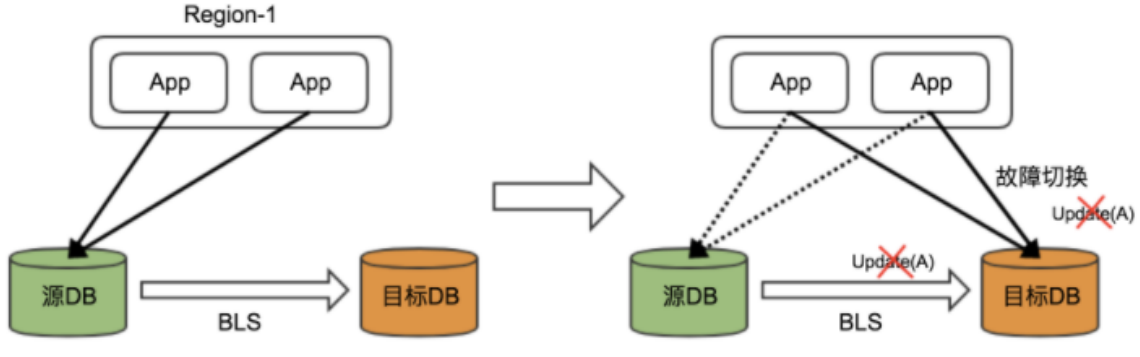
说明：上述情况 2 和 3 不属于本次数据冲突描述范围。

6.3.3.1. 多活实例应用 hashing(可通过主键_id、表名、库名进行 hashing)



应用方需要定制数据路由策略，并保证多写节点间的数据不可冲突。某实例出现不可服务时，切换路由规则保证数据正常访问。此种潜在的数据冲突是多写场景，应由业务方负责管理。如果存在写冲突，则按数据冲突处理。

6.3.3.2. 多活切换(灾备切换)



单向链路 (多向链路同理)

发生实例切换时，写入冲突属于内部冲突，需要数据库灾备系统和应用方一起配合。数据库灾备系统提供接口查询当前同步状态：

1 源实例、灾备系统通道正常，但源实例不可访问(网络错误)：

此场景下，应用通过状态查询接口，保证数据全部同步完成。

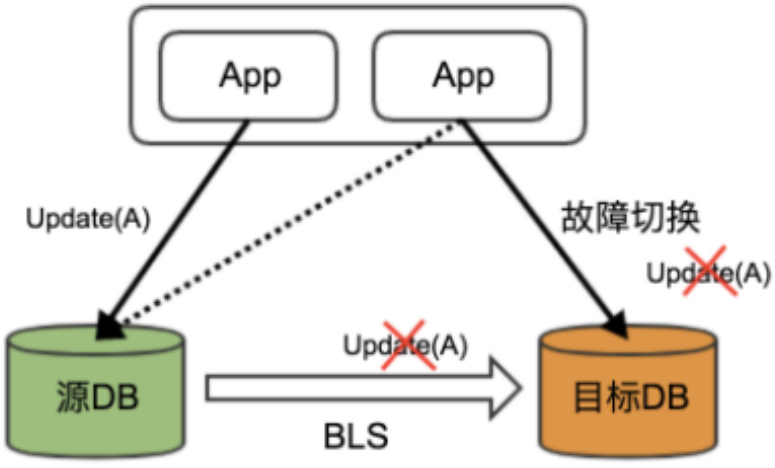
2 通道正常，但源实例不可用，也不可访问(网络错误)：

此场景下，部分 oplog 还未取到，是否切换取决于应用的可用窗口(如 100ms)，会触发异常，停止同步。

3 数据库灾备通道，源实例，都不可用切不可访问：

不建议应用切换，需要等待数据库灾备通道可用(不需要一定可同步)，BLS 停止同步。

6.3.3.3. 应用切换导致数据冲突



应用切换如果存在不完整的情况，数据库灾备同步也会存在异常。

6.4. 如何解决数据冲突

6.4.1. MySQL

目前数据冲突造成同步异常停止，需要与后台工程师联系，我们会根据两个实例的冲

突情况做相应人工操作，主要根据以下两个原则处理：

- 可修复时候：在不记录 binlog 下，修正数据，然后跳过冲突的事务。
- 很难修复时候：在大量数据不一致，难以修正时候，可以选择以某一个实例数据为准创建新的灾备实例。这个时候未同步的数据将丢失。

6.4.2. Redis

Redis 在数据库同步中，不会存在由于一个 key 的数据冲突而造成整个数据同步断掉的场景，同步过程中的数据冲突会造成同步双方的数据不一致问题。目前存在两种处理场景

1. 数据强制覆盖：

   当 Redis key 在数据层面在不一致，会采用强制覆盖的策略。例如，redis 实例 A <--->B 双活部署，同时对一个 key 进行了操作，A 上 set key valueA，B 上 set key valueB。结果 A 上 key 的值是 valueB，B 上 key 的值是 valueA。没有达到最终一致性。

2. 数据忽略：

   如果 Redis 实例 A，key 的 value 是 int 类型，如果数据同步接收到对这个 key 的 value 值设置为 string 类型，就会造成数据类型的冲突，此时 ApsaraDB 灾备功能的策略是忽略掉该条的同步语句，两个实例存在数据不一致。

6.4.3. MongoDB

处理办法

强制更新：按时间序，后者覆盖前者。如果存在唯一索引的偏序关系，则跳过。

优劣势

- 优势：简单，不需要额外存储。适合只依赖_id 使用，无多_id 依赖场景，并且容忍小部分数据写脏。
- 劣势：数据覆盖，乱序，可能导致写入和查询结果不一致。

潜在问题

1 单条数据可能出现历史版本；

2 多个数据之间的偏序关系无法保证；

3 写入的数据，与查询结果不一致（也不保证最终一致）。