04 | 同城双活：如何实现机房之间的数据同步？

2022-10-31 徐长龙 来自北京

《高并发系统实战课》

讲述：徐长龙

时长 13:18　大小 12.15M

你好，我是徐长龙。今天我们来看看用户中心改造的另一个阶段：构建多机房。

在业务初期，考虑到投入成本，很多公司通常只用一个机房提供服务。但随着业务发展，流量不断增加，我们对服务的响应速度和可用性有了更高的要求，这时候我们就要开始考虑将服务分布在不同的地区来提供更好的服务，这是互联网公司在流量增长阶段的必经之路。

之前我所在的公司，流量连续三年不断增长。一次，机房对外网络突然断开，线上服务全部离线，网络供应商失联。因为没有备用机房，我们经过三天紧急协调，拉起新的线路才恢复了服务。这次事故影响很大，公司损失达千万元。

经过这次惨痛的教训，我们将服务迁移到了大机房，并决定在同城建设双机房提高可用性。这样当一个机房出现问题无法访问时，用户端可以通过 HttpDNS 接口快速切换到无故障机房。

为了保证在一个机房损坏的情况下，另外一个机房能直接接手流量，这两个机房的设备必须是 1:1 采购。但让其中一个机房长时间冷备不工作过于浪费，因此我们期望两个机房能同时对外提供服务，也就是实现同城双机房双活。

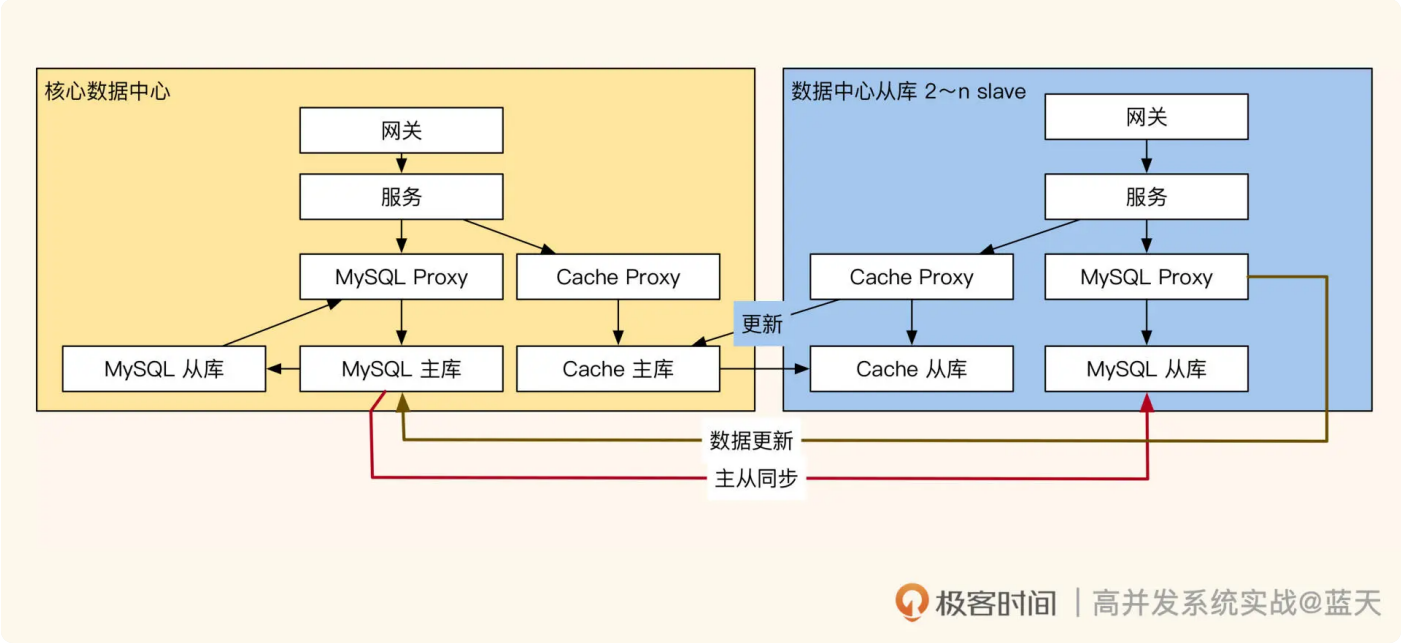
对此，我们碰到的一个关键问题就是，**如何实现同城双活的机房数据库同步？**

核心数据中心设计

因为数据库的主从架构，全网必须只能有一个主库，所以我们只能有一个机房存放更新数据的主库，再由这个机房同步给其他备份机房。虽然机房之间有专线连接，但并不能保证网络完全稳定。如果网络出现故障，我们要想办法确保机房之间能在网络修复后快速恢复数据同步。

有人可能会说，直接采用分布式数据库不就得了。要知道改变现有服务体系，投入到分布式数据库的大流中需要相当长的时间，成本也非常高昂，对大部分公司来说是不切实际的。所以我们要看看怎么对现有系统进行改造，实现同城双活的机房数据库同步，这也是我们这节课的目标。

核心数据库中心方案是常见的实现方式，这种方案只适合相距不超过 50 公里的机房。

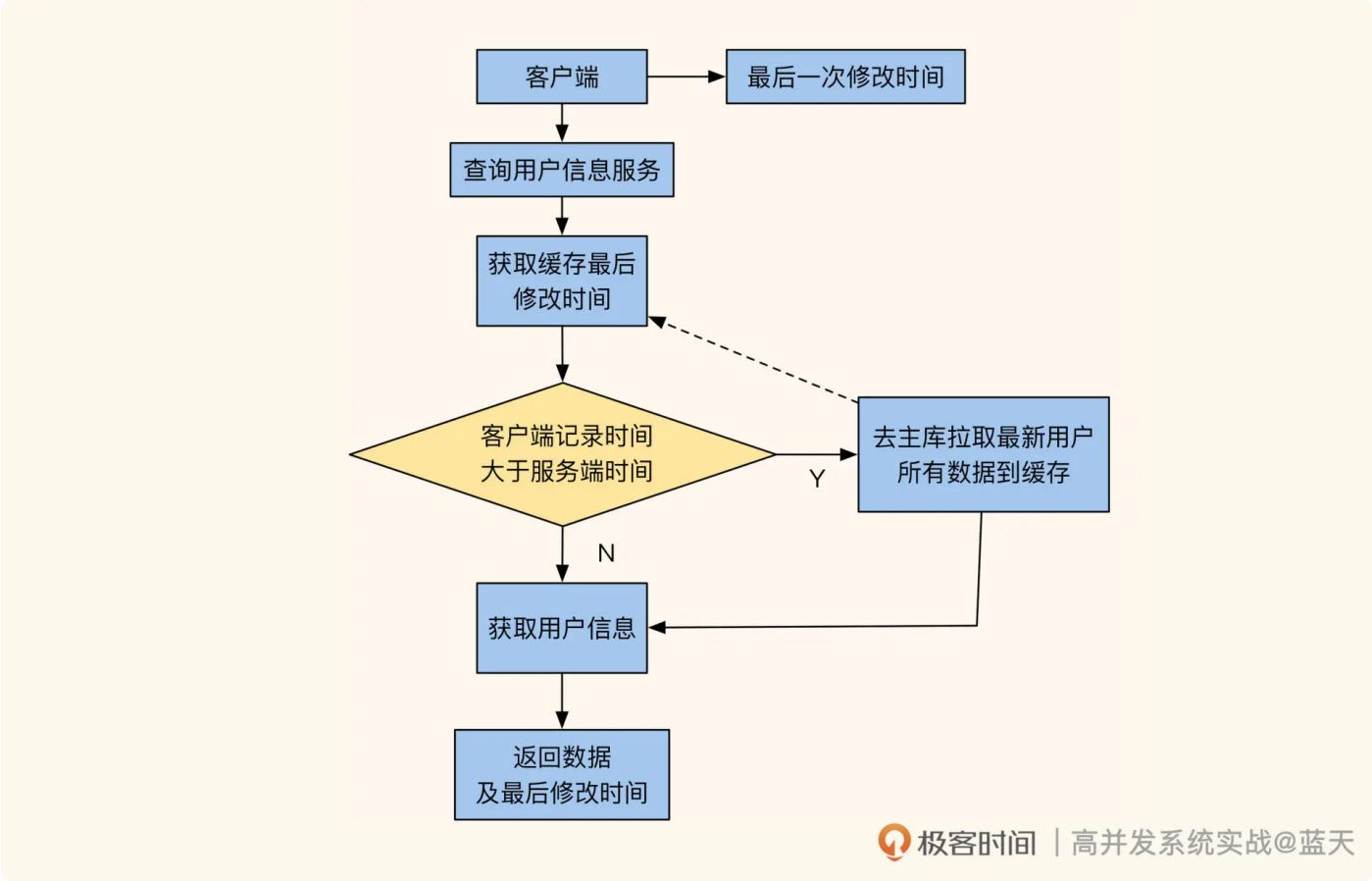


同城双活单向同步

在这个方案中，数据库主库集中在一个机房，其他机房的数据库都是从库。当有数据修改请求时，核心机房的主库先完成修改，然后通过**数据库主从同步**把修改后的数据传给备份机房的从库。由于用户平时访问的信息都是从缓存中获取的，为了降低主从延迟，备份机房会把修改后的数据先更新到本地缓存。

与此同时，客户端会在本地记录下数据修改的最后时间戳（如果没有就取当前时间）。当客户端请求服务端时，服务端会自动对比缓存中对应数据的更新时间，是否小于客户端本地记录的修改时间。

如果缓存更新时间小于客户端内的修改时间，服务端会触发同步指令尝试在从库中查找最新数据；如果没有找到，就把从主库获取的最新数据放到被访问机房的缓存中。这种方式可以避免机房之间用户数据更新不及时的问题。



客户端切换强迫服务端刷新本地缓存逻辑

除此之外，客户端还会通过请求调度接口，让一个用户在短期内只访问一个机房，防止用户在多机房间来回切换的过程中，数据在两个机房同时修改引发更新合并冲突。

总体来看，这是一个相对简单的设计，但缺点也很多。比如如果核心机房离线，其他机房就无法更新，故障期间需要人工切换各个 proxy 内的主从库配置才能恢复服务，并且在故障过后还需要人工介入恢复主从同步。

此外，因为主从同步延迟较大，业务中刚更新的数据要延迟一段时间，才能在备用机房查到，这会导致我们业务需要人工兼顾这种情况，整体实现十分不便。

这里我给你一个常见的网络延迟参考：

- 同机房服务器：0.1 ms
- 同城服务器（100 公里以内）：1ms（10 倍 同机房）
- 北京到上海：38ms（380 倍 同机房）
- 北京到广州：53ms（530 倍 同机房）

注意，上面只是一次 RTT 请求，而机房间的同步是多次顺序地叠加请求。如果要大规模更新数据，主从库的同步延迟还会加大，所以这种双活机房的数据量不能太大，并且业务不能频繁更新数据。

此外还要注意，如果服务有强一致性的要求，所有操作都必须在主库“远程执行”，那么这些操作也会加大主从同步延迟。

除了以上问题外，双机房之间的专线还会偶发故障。我碰到过机房之间专线断开两小时的情况，期间只能临时用公网保持同步，但公网同步十分不稳定，网络延迟一直在 10ms～500ms 之间波动，主从延迟达到了 1 分钟以上。好在用户中心服务主要以**长期缓存**的方式存储数据，业务的主要流程没有出现太大问题，只是用户修改信息太慢了。

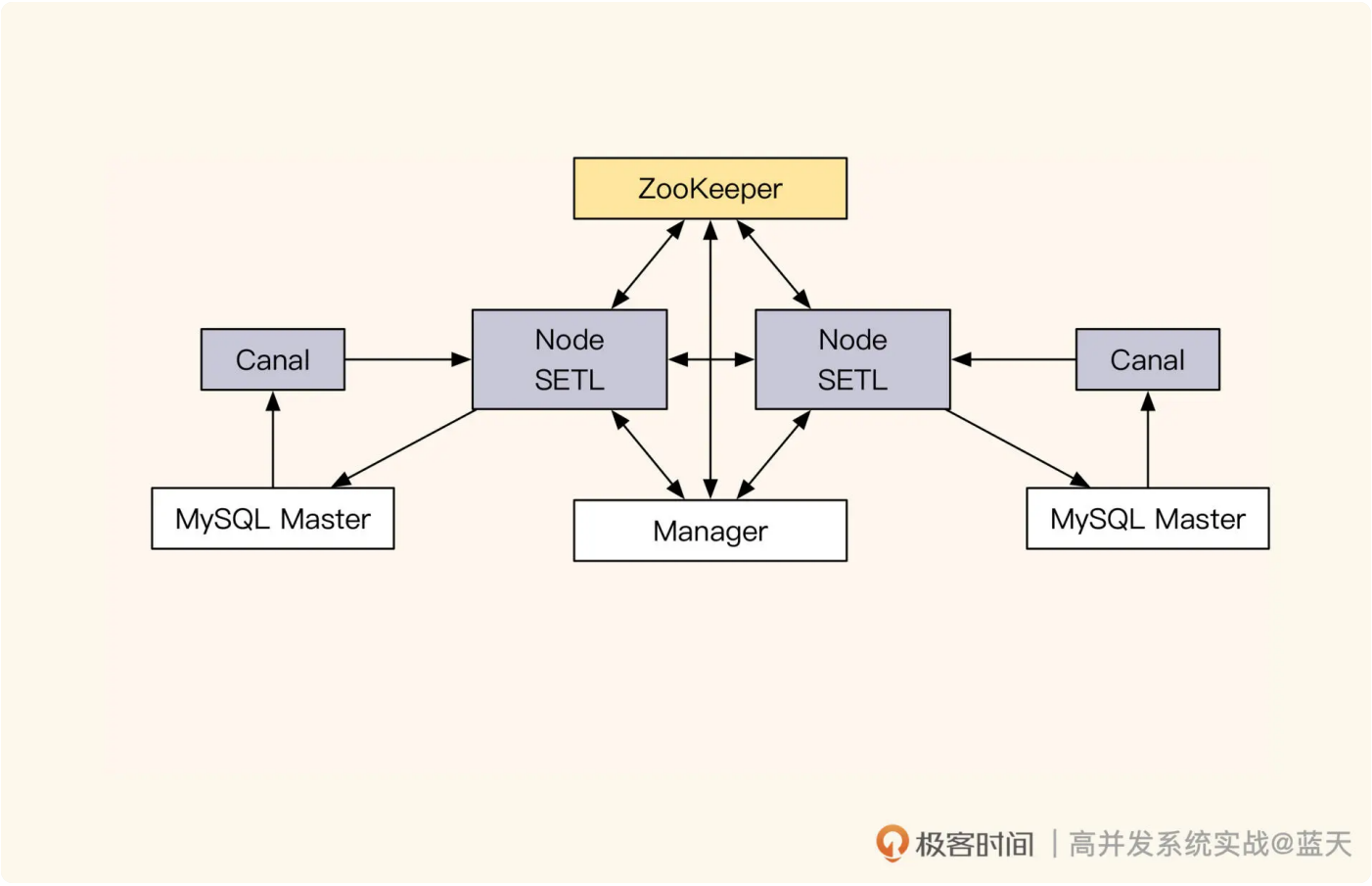
有时候，双机房还会偶发主从同步断开，对此建议做告警处理。一旦出现这种情况，就发送通知到故障警报群，由 DBA 人工修复处理。

另外，我还碰到过主从不同步期间，有用户注册自增 ID 出现重复，导致主键冲突这种情况。这里我推荐将自增 ID 更换为“由 SnowFlake 算法计算出的 ID”，这样可以减少机房不同步导致的主键冲突问题。

可以看到，核心数据库的中心方案虽然实现了同城双机房双活，但是人力投入很大。DBA 需要手动维护同步，主从同步断开后恢复起来也十分麻烦，耗时耗力，而且研发人员需要时刻关注主从不同步的情况，整体维护起来十分不便，所以我在这里推荐另外一个解决方案：数据库同步工具 Otter。

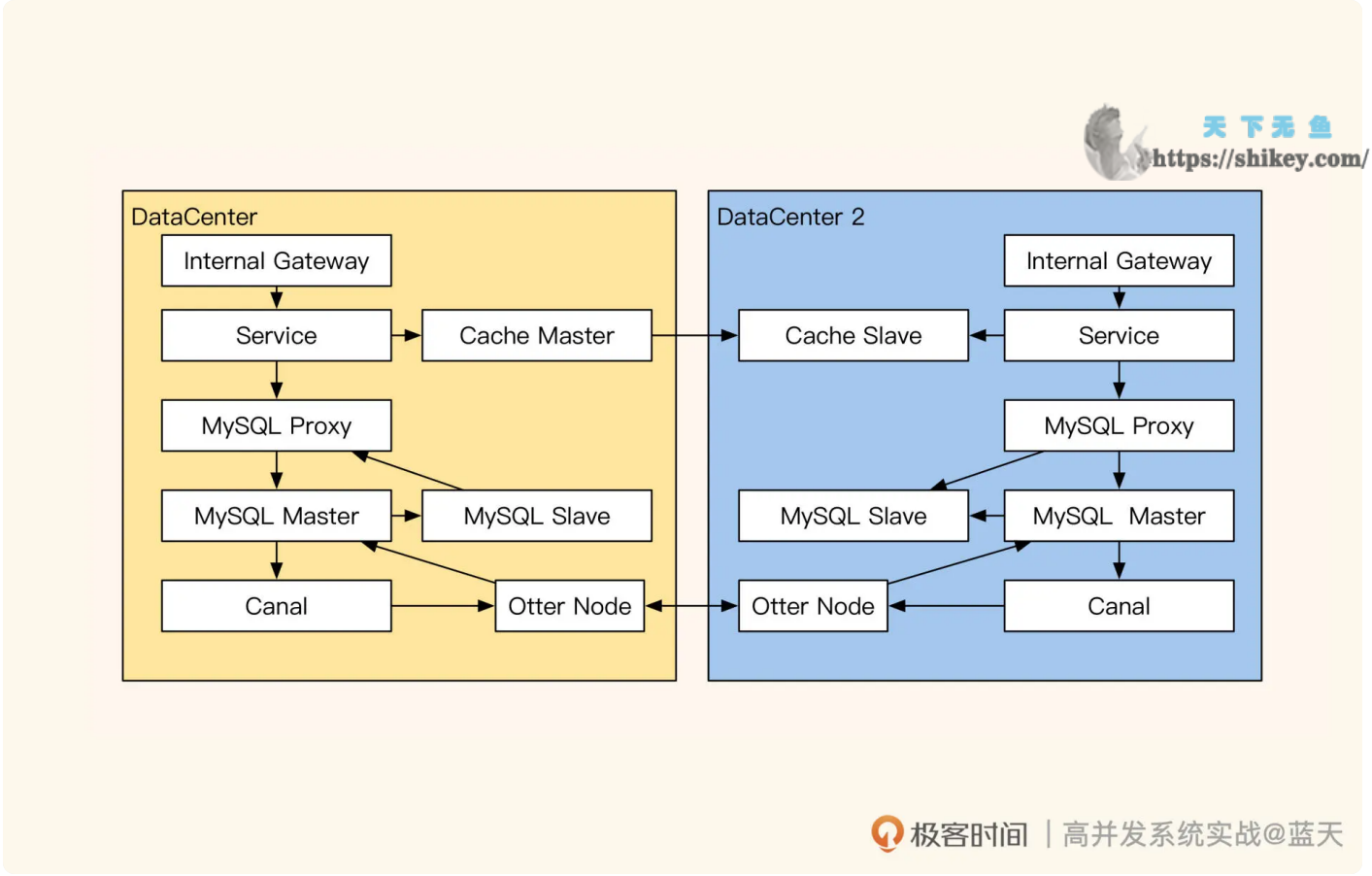
跨机房同步神器：Otter

Otter 是阿里开发的数据库同步工具，它可以快速实现跨机房、跨城市、跨国家的数据同步。如下图所示，其核心实现是通过 Canal 监控主库 MySQL 的 Row binlog，将数据更新并行同步给其他机房的 MySQL。



Otter主要部署结构

因为我们要实现同城双机房双活，所以这里我们用 Otter 来实现同城双主（注意：双主不通用，不推荐一致要求高的业务使用），这样双活机房可以双向同步：



同城双活双向同步方案

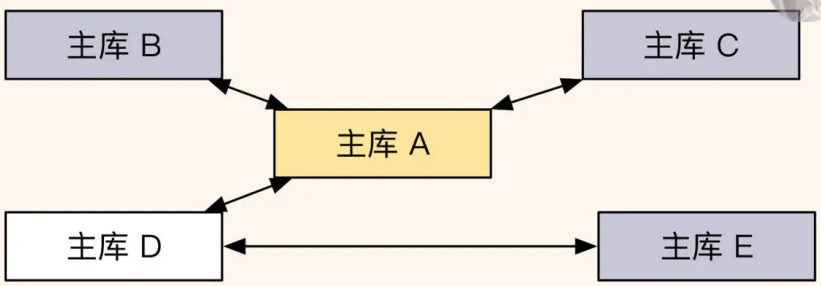
如上图，每个机房内都有自己的主库和从库，缓存可以是跨机房主从，也可以是本地主从，这取决于业务形态。Otter 通过 Canal 将机房内主库的数据变更同步到 Otter Node 内，然后经由 Otter 的 SETL 整理后，再同步到对面机房的 Node 节点中，从而实现双机房之间的数据同步。

讲到这里不得不说一下，Otter 是怎么解决两个机房同时修改同一条数据所造成的冲突的。

在 Otter 中数据冲突有两种：一种是行冲突，另一种是字段冲突。**行冲突**可以通过对比数据修改时间来解决，或者是在冲突时回源查询覆盖目标库；对于**字段冲突**，我们可以根据修改时间覆盖或把多个修改动作合并，比如 a 机房 -1，b 机房 -1，合并后就是 -2，以此来实现数据的最终一致性。

但是请注意，**这种合并方式并不适合库存一类的数据管理，因为这样会出现超卖现象。如果有类似需求，建议用长期缓存解决**。

Otter 不仅能支持双主机房，还可以支持多机房同步，比如星形双向同步、级联同步（如下图）等。但是这几种方式并不实用，因为排查问题比较困难，而且当唯一决策库出现问题时，恢复起来很麻烦。所以若非必要，不推荐用这类复杂的结构。



星状同步

极联同步

另外，我还要强调一点，我们讲的**双活双向同步方案只适合同城**。一般来说，50～100 公里以内的机房同步都属于同城内。

超过这个距离的话，建议只做数据同步备份，因为**同步延迟过高**，业务需要在每一步关注延迟的代价过大。如果我们的业务对一致性的要求极高，那么建议在设计时，把这种一致性要求限制在同一个机房内，其他数据库只用于保存结果状态。

那为什么机房间的距离必须是 100 公里以内呢？你看看 Otter 对于不同距离的同步性能和延迟参考，应该就能理解了。

具体表格如下所示：

同步距离	操作数据量	Otter 主从延迟
同城机房同步 1 ms RTT 50km以内	100tps	延迟 0.1 s
	5000tps	延迟 1 s
中美异地机房同步 (200 ms RTT)	100tps	延迟 2 s
	5000tps	延迟 10 s

为了提高跨机房数据同步的效率，Otter 对用于主从同步的操作日志做了合并，把同一条数据的多次修改合并成了一条日志，同时对网络传输和同步策略做了滑窗并行优化。

对比 MySQL 的同步，Otter 有 5 倍的性能提升。通过上面的表格可以看到，通过 Otter 实现的数据同步并发性能好、延迟低，只要我们将用户一段时间内的请求都控制在一个机房内不频繁切换，那么相同数据的修改冲突就会少很多。

用 Otter 实现双向同步时，我们的业务不需要做太多改造就能适应双主双活机房。具体来说，业务只需要操作本地主库，把“自增主键”换成“snowflake 算法生成的主键”、“唯一索引互斥”换成“分布式互斥锁”，即可满足大部分需求。

但是要注意，采用同城双活双向同步方案时，数据更新不能过于频繁，否则会出现更大的同步延迟。当业务操作的数据量不大时，才会有更好的效果。

说到这里，我们再讲一讲 Otter 的故障切换。目前 Otter 提供了简单的主从故障切换功能，在 Manager 中点击“切换”，即可实现 Canal 和数据库的主从同步方式切换。如果是同城双活，那关于数据库操作的原有代码我们不需要做更改，因为这个同步是双向的。

当一个机房出现故障时，先将故障机房的用户流量引到正常运转的机房，待故障修复后再恢复数据同步即可，不用切换业务代码的 MySQL 主从库 IP。切记，**如果双活机房有一个出现故障了，其他城市的机房只能用于备份或临时独立运行，不要跨城市做双活，因为同步延迟过高会导致业务数据损坏的后果。**

最后，我再啰嗦一下使用 Otter 的注意事项：第一，为了保证数据的完整性，变更表结构时，我们一般会先从从库修改表结构，因此在设置 Otter 同步时，建议将 pipeline 同步设置为忽略 DDL 同步错误；第二，数据库表新增字段时，只能在表结尾新增，不能删除老字段，并且建议先把新增字段同步到目标库，然后再同步到主库，因为只有这样才不会丢数据；第三，双向同步的表在新增字段时不要有默认值，同时 Otter 不支持没有主键的表同步。

总结

机房之间的数据同步一直是行业里的痛，因为高昂的实现代价，如果不能做到双活，总是会有一个 1:1 机器数量的机房在空跑，而且发生故障时，没有人能保证冷备机房可以马上对外服务。

但是双活模式的维护成本也不低，机房之间的数据同步常常会因为网络延迟或数据冲突而停止，最终导致两个机房的数据不一致。好在 Otter 对数据同步做了很多措施，能在大多数情况下保证数据的完整性，并且降低了同城双活的实现难度。

即使如此，在业务的运转过程中，我们仍然需要人工梳理业务，避免多个机房同时修改同一条数据。对此，我们可以通过 HttpDNS 调度，让一个用户在某一段时间内只在一个机房内活跃，这样可以降低数据冲突的情况。

而对于修改频繁、争抢较高的服务，一般都会在机房本地做整体事务执行，杜绝跨机房同时修改导致同步错误的发生。

相信未来随着行业的发展，多活机房的同步会有更好的解决方案，今天的内容就讲到这里，期待你在留言区与我互动交流！

思考题

如果 Otter 同步的链路是环形的，那么如何保证数据不会一直循环同步下去？

赞 4　提建议

上一篇　03丨Token：如何降低用户身份鉴权的流量压力？

下一篇　05丨共识Raft：如何保证多机房数据的一致性？

精选留言 (9)

写留言

张申傲

2022-10-31 来自北京

印象中 MySQL 的 binlog 中会记录一个 server id，用于唯一标识一个集群中的 MySQL 实例。做数据同步时，如果解析发现 binlog 中的 server id 和自己相同，说明是当前实例生成的数据变更，就不会再执行同步了。这个机制应该可以打破循环同步。

作者回复: 你好，很高兴再次收到你的留言，我理解你这是一个很好的方式，我查相关资料的时候，突然意识到，这个方式也是有瑕疵的，比如，串联同步，级联同步，这会被中转服务覆盖掉

3

千锤百炼领悟之极限

2022-11-13 来自北京

如果老师可以在每一期的结尾给出上一期思考题的答案就好了。

作者回复: 你好，答案后期会公布，前期多看大家留言讨论，多查，多独立思考，会有更好的帮助

1

一步

2022-11-10 来自北京

binglog 中有偏移量，可以根据这个偏移量来判断是否已经同步过

作者回复: 你好，一步，如果同步的一个节点超过两个个，这个方式就会存在问题～

Mr.Tree
2022-11-10 来自北京

用 SnowFlake 算法计算一个更新版本ID，数据同步时，从库的更新版本ID不同则同步更新数据

作者回复: 你好，是个方法，如何解决对于多个事务需要对比的内容有些多

移横为固
2022-11-04 来自北京

没有接触过otter：
如果otter有节点id可以传递，类似于binlog的serverId，可以在循环过程中判断是否自己的id，那样就可以中断循环。
还想到一种方法：把otter的节点中的收集binglog功能和同步功能进行分离。新增一个otter节点，连接所有机房；而机房中的各个otter节点收集binglog，提取同步指令，汇总到新增的节点，由新增的节点发送到各个机房。这样其实把循环更新转换为由中心节点更新。

作者回复: 你好，移横为固，很高兴收到你的回复，第一个方式我觉得是可行的～另外那个方法需要解决多节点同时修改的冲突时候会比较麻烦～

共 2 条评论 >

peter
2022-11-02 来自北京

请问：客户端访问指定机房，是客户端还是服务端决策的？如果是服务端，是怎么实现的？

作者回复: 你好，peter，这里是通过统一调度服务去做的，常见方式是24小时内用户最近访问的ip距离哪个机房比较近，刚进app时会问调度服务，会被调度服务指定最近一段时间都请求一个机房，通过这样保证用户修改的数据都集中在一个机房，这样缓存和修改有一定时间内在一个机房集中，这样能减少多机房交叉更新同步导致冲突，同时这种调度普遍喜欢用httpdns去做，因为比较好整体替换管理。

Sky
2022-11-01 来自北京

你好，请教几个问题。
1，除了数据库，其实缓存也有数据同步的问题，而且缓存的访问量会比数据库高很多，双机房对缓存怎么处理？
2，如果一个城市的网络主干断了，同城双机房也不行，这种情况也不是没有出现过，异地双活应该怎么做？

作者回复: 你好，老朋友，很高兴再次见到你
第一个问题确实如此，目前是通过保证用户一段时间内只能访问一个机房来减少延迟，并且客户端保存最后更新时间，用这个方式来刺激服务端主动更新过期的缓存

第二个问题，城市主干线断了，可以让其他城市机房接管之前的双活机房工作，但是必须距离很近的两个机房才可以。核心在于同步延迟过大，会出现刚插入的数据查不到的情况，会让业务很难维护。如果有决心做双活，业务层是必须能够有类似raft的支撑才可以

共 4 条评论 >

Geek_fc9db1
2022-10-31 来自北京

老师你好，oceanbase的三地五中心异地多活容灾方案怎么样

作者回复: 你好，感谢你的留言，ob这个服务有些贵，笑，所以我一直没进行测试，印象中这个服务刚出来的时候偏订单服务，所以后续没有深入分析，后续我研究下

共 2 条评论 >

PunkHoo
2022-10-31 来自北京

徐老师你好，文中 "如果双活机房有一个出现故障了，其他城市的机房只能用于备份或临时独立运行，不要跨城市做双活，因为同步延迟过高会导致业务数据损坏的后果。" 这里面如果其他城市的机房在切换前没来得及同步"主库"数据，切换后的数据是否如何保证一致性呢？

作者回复: 你好，PunkHoo，很高兴收到你的留言，一般来说但凡碰到彻底失联的机房情况的切换都是有损的，对于损失只是大小问题，对于这种单主库情况，最理想情况就是损失几条数据，如果是多主情况下会好一点。我碰到的情况多数是需要人工对比修复一下，不过我们的数据常见都有create_time，update_time相对的修复难度会小很多。
同时，高频率更新的数据是不推荐用这个方式做多机房的。

共 3 条评论 >