基于数据库复制的技术架构讨论

作者：88250
原文链接：https://ld246.com/article/1465278948107
来源网站：链滴

<h3 id="背景">背景</h3>
这里的数据库复制指的是将 主 业务数据库实例上的库通过同步机制（比如 MySQL binlog）_准_实时（比如最大延迟为 3s）复制到其他数据库实例上，这些实例库只做查询，做数据写入。
这套架构设计的主要优势：

各业务应用能够方便地在自己的 DB 实例上进行业务查询，比如通过 join 主业务库
在不明确业务边界、没有梳理好业务对应技术模块时可以最小成本进行变更或扩展
实现读写分离，提升性能

<h3 id="一些问题">一些问题</h3>
实际在实施过程中主要遇到两个问题：

不可能实时完成数据同步，将造成业务上面的不一致，比如调用主库服务更新数据后，在业务库不能实时查询到已更新的数据
很难保证高可用（在使用阿里 DTS 时出现过多次问题，自己做主从可能会好一些）

为了满足业务发展，复制库的数量会逐步增多（比如新开一个产品可能就需要多复制一套库），上两个问题可能会导致严重的故障， [CAP] 不能兼得。
<h3 id="服务化">服务化</h3>
基于数据库复制架构的核心理念是将数据源暴露给应用，开发者直接针对数据源进行开发，是一非常直接的方式。
但随着业务的逐渐清晰，一些业务逻辑是可以抽取形成服务并提供给其他应用使用的。这一点业有着非常多的实践，比如 [SOA] 、 [ESB] 等。这种架构的核心理念是基于 服务 的，应用开发者是面向服务进行开发，而不是面向数据源，用近些年的流行语来说，这是 服务 。
一个良好的演进路径应该是：

应用内部服务化：各点针对数据源编程 -> 类似工具函数的复用 -> 抽象函数为本地服务 -gt; 以本地服务组合来完成业务事务
这个路线基本所有应用框架都已会覆盖，即使编程语言不同，架构范式基本一致，都是 MVC + DB 地事务

应用分布式服务化：将应用进行拆分 -> 形成服务分类（业务/技术）和分级（1/2/3） -> 通过网络协议暴露各个服务 -> 解决分布式事务问题 -> 以分布式服务组合完成业务事务
这个路线会随着编程语言不同而略有不同，架构范式基本都是注册中心 + 服务链路 RPC + 分布式事，目前 Java 系可用的框架比较多，比如 [Dubbo] 及其衍生框架，需要考虑异构技术之间的整合

数据库方面基本路线是随着服务的拆分而拆分，主要难点在于消除 join，或者只在某些不可避免场景少量使用。
<hr>
以上都是我瞎扯的，实在编不下去了，欢迎大家分享真正的经验 :-p