DataBase多活

tianjiqx

2020.12.08

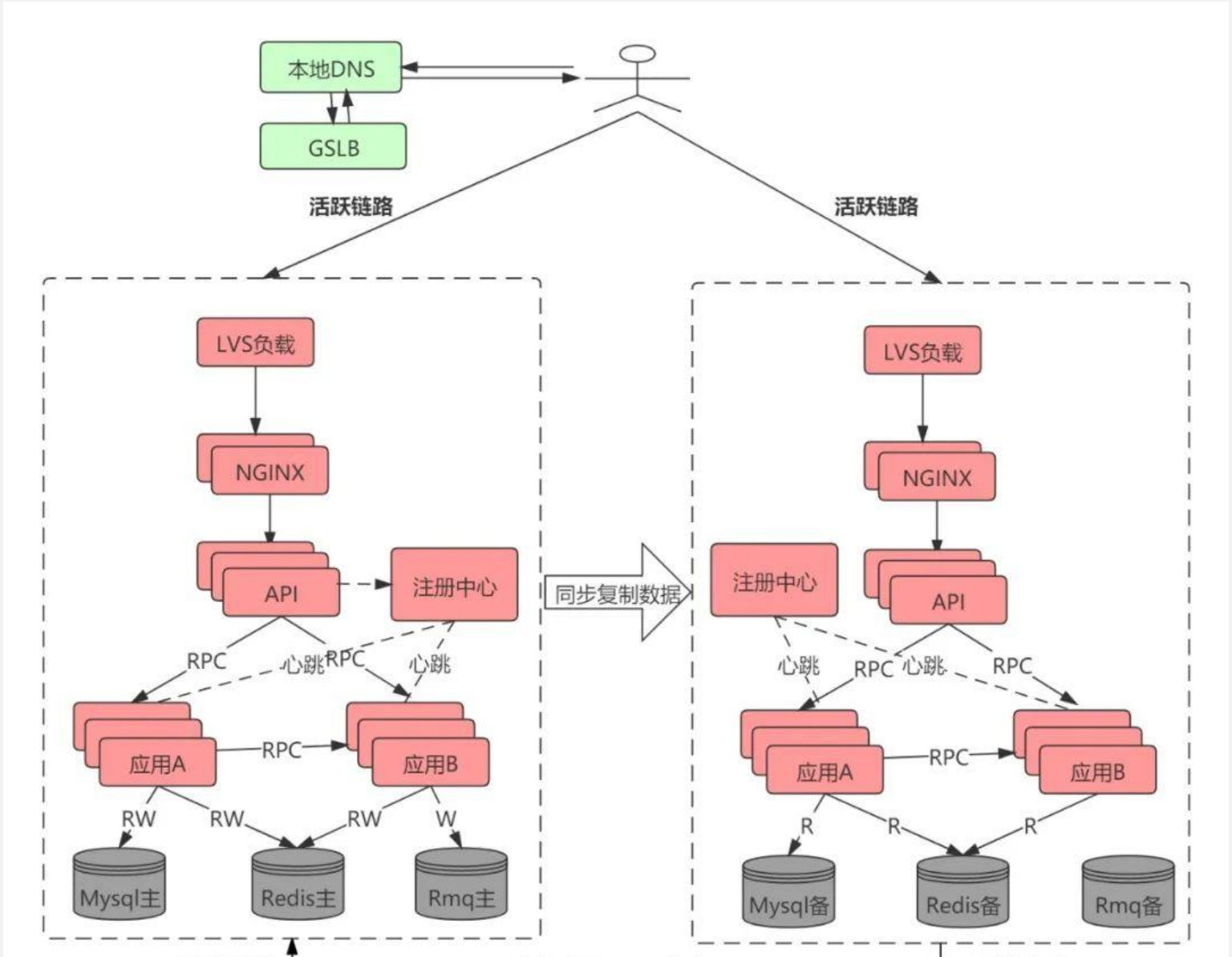
高可用：服务多活

- 背景
  - 应对极端场景机房断电、机房火灾、地震
  - 中心业务连续性，增强抗风险能力
- 多活方案
  - 同城双活
  - 两地三中心
  - 三地五中心
  - 异地多活

同城双活

- 同城双活是在同城或相近区域内建立两个机房。
  - 通信线路质量较好，比较容易实现数据的同步复制
  - 两个机房各承担一部分流量，一般入口流量完全随机，内部RPC调用尽量通过就近路由闭环在同机房
  - 数据仍然是单点写到主机房数据库，然后实时同步到另外一个机房

同城双活


本地DNS
GSLB
活跃链路
活跃链路
LVS负载
NGINX
API
注册中心
同步复制数据
RPC
心跳
RPC
心跳
应用A
RPC
应用B
RW
RW
RW
W
Mysql主
Redis主
Rmq主
LVS负载
NGINX
注册中心
API
心跳
RPC
心跳
RPC
应用A
RPC
应用B
R
R
R
Mysql备
Redis备
Rmq备

同城双活

- 同城双活是在同城或相近区域内建立两个机房。
  - 通信线路质量较好，比较容易实现数据的同步复制
  - 两个机房各承担一部分流量，一般入口流量完全随机，内部RPC调用尽量通过就近路由闭环在同机房
  - 数据仍然是单点写到主机房数据库，然后实时同步到另外一个机房

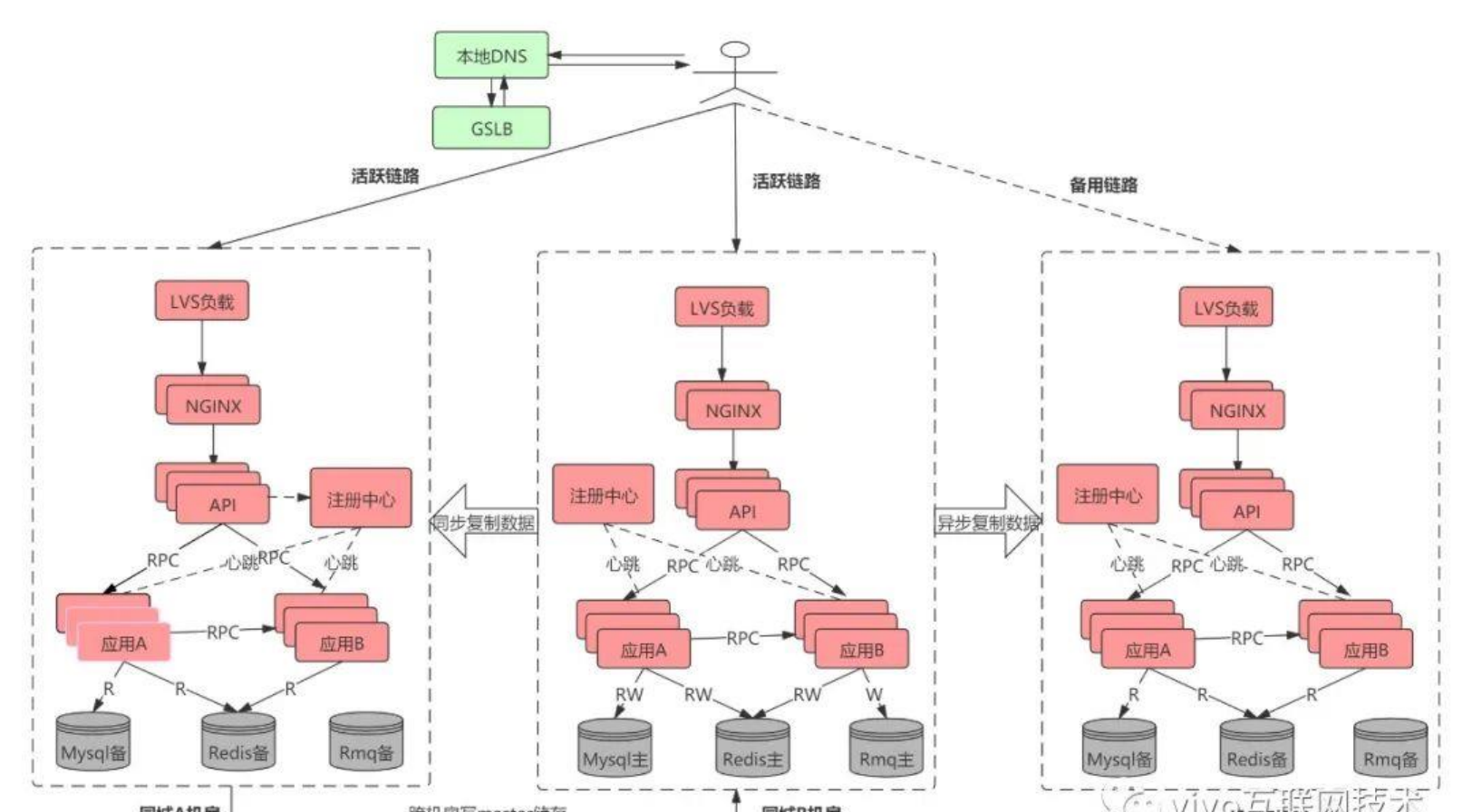
- 同城双活可有效用于防范火灾、建筑物破坏、供电故障、计算机系统及人为破坏引起的机房灾难。

同城双活

- 优势
  - 服务同城双活，数据同城灾备，同城不丢失数据情况下跨机房级别容灾
  - 架构方案较为简单，核心是解决底层数据双活，由于双机房距离近，通信质量好，底层储存例如mysql可以采用同步复制，有效保证双机房数据一致性
- 劣势
  - 数据库**写数据存在跨机房调用**，在复杂业务以及链路下频繁跨机房调用增加响应时间，影响系统性能和用户体验
  - 保证同城市地区容灾，当服务所在的城市或者地区网络整体故障、发生不可抗拒的自然灾害时候有服务故障以及丢失数据风险。对于核心金融业务至少要有跨地区级别的灾备能力
  - 服务规模足够大(例如单体应用超过万台机器)，所有机器链接一个主数据库实例会引起连接不足问题

两地三中心

- 同城双中心 + 异地灾备中心
  - 当双中心所在城市或者地区出现异常而都无法对外提供服务的时候，异地灾备中心可以用备份数据进行业务的恢复

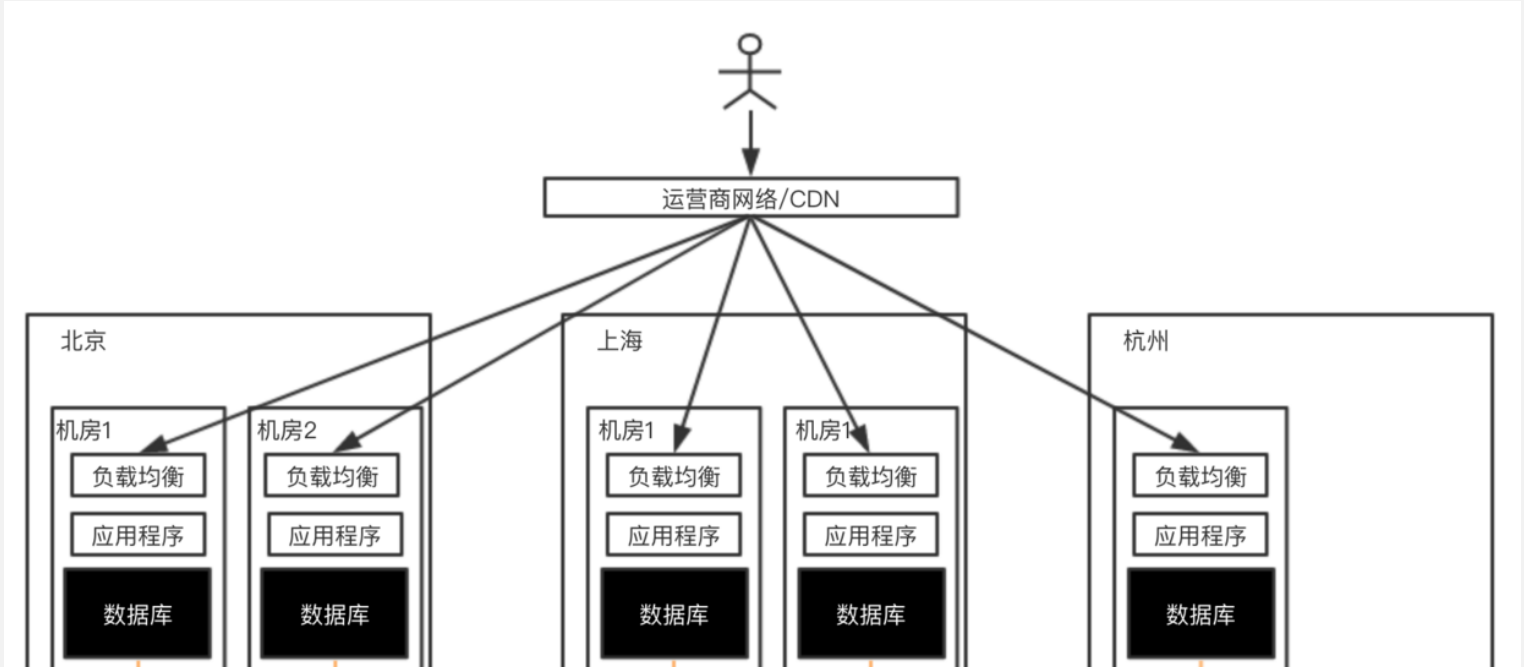


两地三中心

- 优势
  - 服务同城双活，数据同城灾备，同城不丢失数据情况下跨机房级别容灾。
  - 架构方案较为简单，核心是解决底层数据双活，由于双机房距离近，通信质量好，底层储存例如mysql可以采用同步复制，有效保证双机房数据一致性。
  - 灾备中心能防范同城双中心同时出现故障时候利用备份数据进行业务的恢复。
- 劣势
  - 数据库**写数据存在跨机房调用**，在复杂业务以及链路下频繁跨机房调用增加响应时间，影响系统性能和用户体验
  - 服务规模足够大(例如单体应用超过万台机器)，所有机器链接一个主数据库实例会引起连接不足问题
  - 出问题不敢轻易将流量切往异地数据备份中心，异地的备份数据中心是冷的，平时没有流量进入，因此出问题需要较长时间对异地灾备机房进行验证（异步，数据未必完全一致，备份中心利用率不高）

三地五中心

- 2+2+1
  - 不再区分生产数据中心和灾备数据中心，只有数据中心，而且数据中心之间相互备份数据，保证每个数据中心都是全量数据
  - 用户可以在任意一个数据中心上进行读写操作
    - **用户分组**，应用程序、数据库负载均衡、数据库分表等等都需要按用户进行分组
    - **单元化**，同一个用户的请求与操作都在同一个机房内，不跨机房



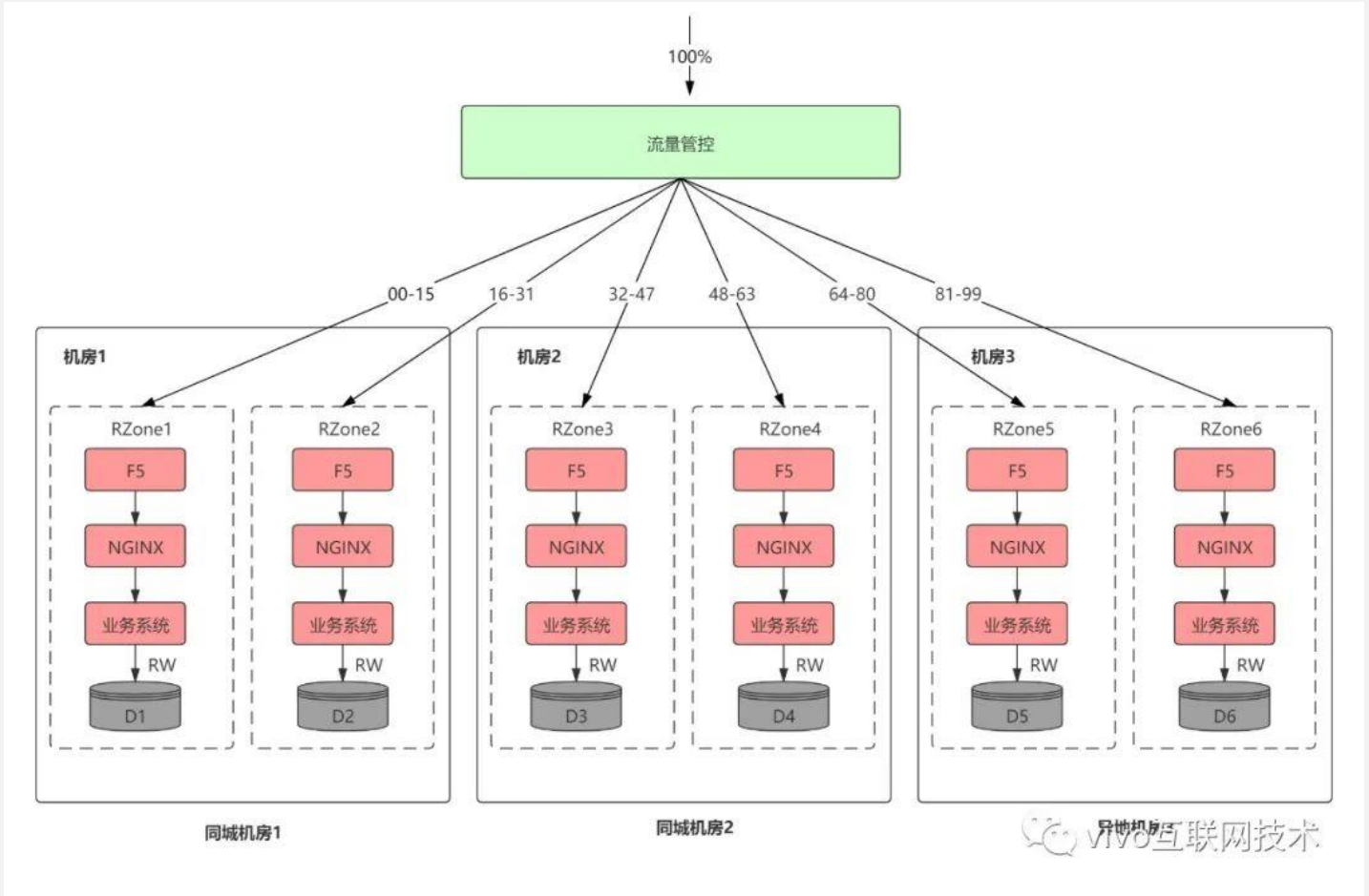
异地多活

- 异地多活指分布在异地的多个站点同时对外提供服务的业务场景

- 挑战
  - 延时，异地多个单元对同一行记录进行修改
  - 维度划分，避免跨机房调用，单元内数据读写封闭
  - 某个单元内访问其他单元数据需要能正确路由到对应的单元(A转账B，A，B不同单元)
  - 数据同步，单元数据同步到其他中心

异地多活

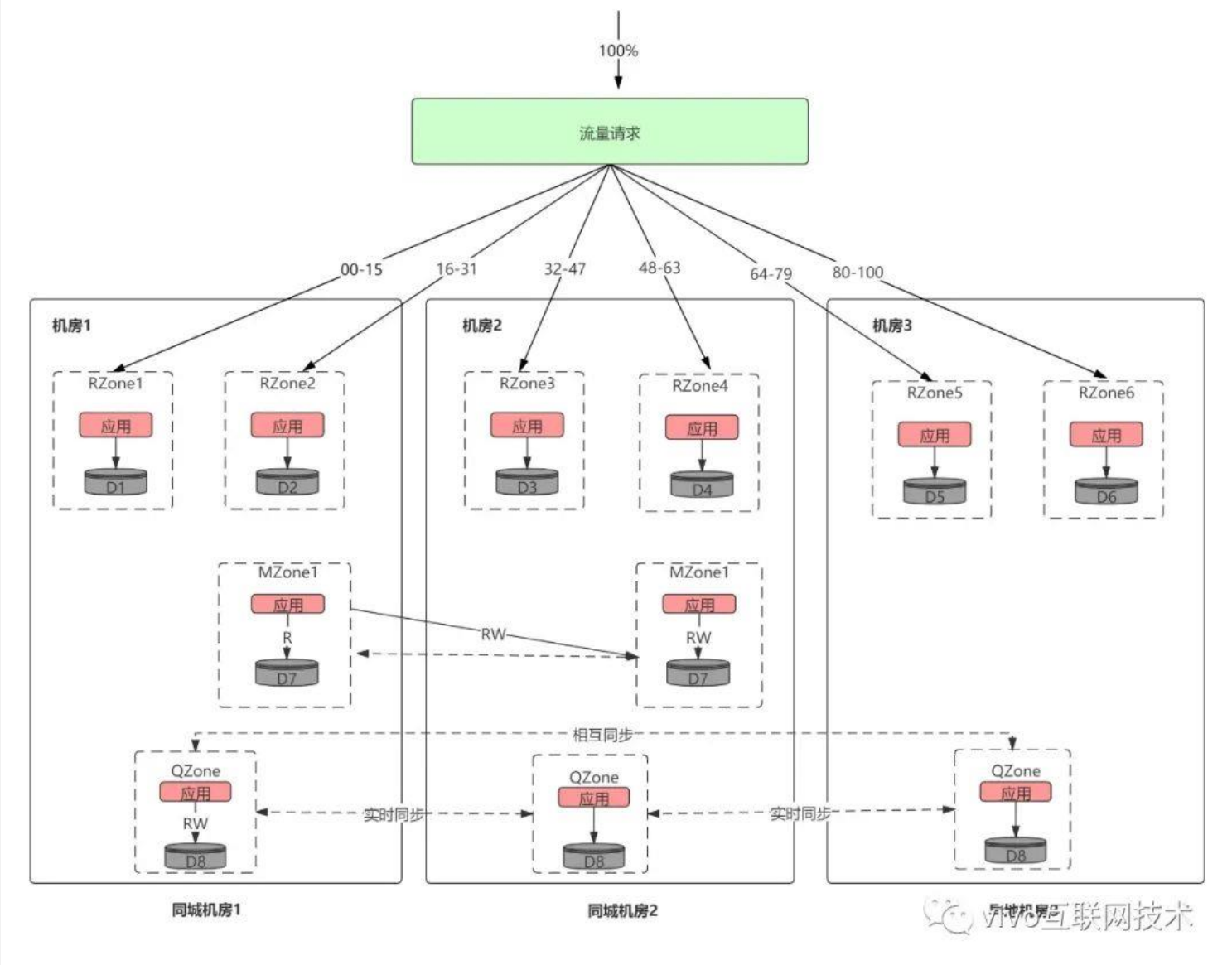
- 单元Rzone
  - 一个能完成所有业务操作的自包含集合（服务和数据），如买家相关操作



异地多活

- **非单元化应用和数据**
  - 如商家相关操作
  - 延时不铭感但是对数据一致性非常铭感，同城双活方式部署，其他应用可能跨区域调用该应用，MZone应用
  - 对数据调用延时铭感但是可以容忍数据短时间不一致，保持一个机房一份全量数据，机房之间以增量的方式实时同步，Qzone
    - 算法、风控、配置

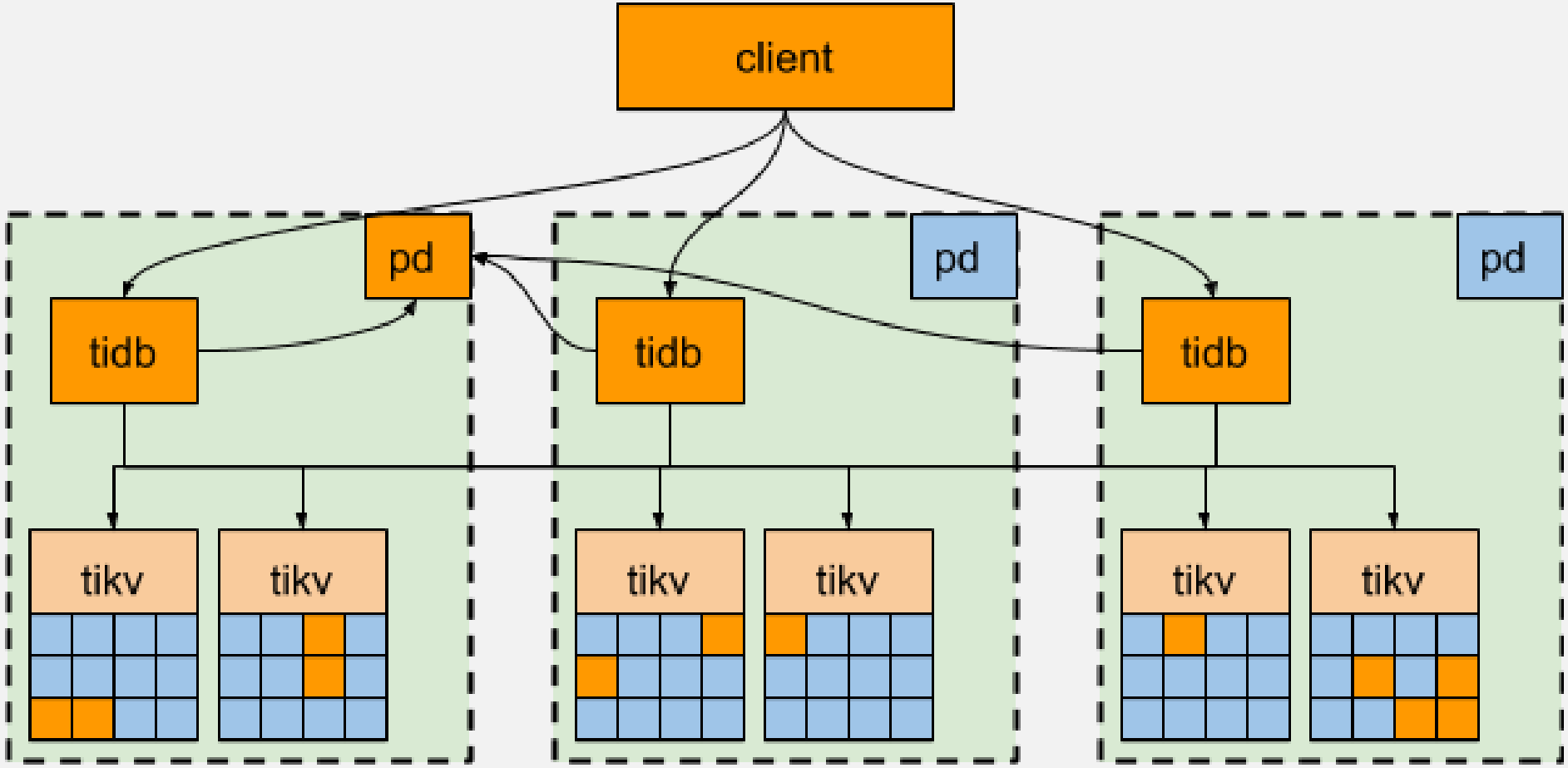
异地多活


100%
流量请求
00-15
16-31
32-47
48-63
64-79
80-100
机房1
机房2
机房3
RZone1
RZone2
RZone3
RZone4
RZone5
RZone6
应用
D1
D2
D3
D4
D5
D6
MZone1
R
RW
D7
相互同步
QZone
D8
实时同步
同城机房1
同城机房2
vivo互联网技术

异地多活

- 优势
  - 容灾能力大幅度提高，服务异地多活，数据异地多活
  - 理论上系统服务可以水平扩展，异地多机房突破大幅度提升整体容量，理论上不会有性能担忧
  - 将用户流量切分到多个机房和地区去，有效能减少机房和地区级别的故障影响范围
- 劣势
  - 架构非常复杂，部署和运维成本很高，需要对依赖的中间件、储存做多方面能力改造
  - 对业务系统有一定的侵入性，由于单元化影响服务调用或者写入数据要路由到对应的单元，业务系统需要设置路由标识(例如uid)。
  - 无法完全避免跨单元、跨地区调用服务，例如转账业务。我们要做的是尽力避免跨地区的服务调用

TiDB：同城多数据中心部署

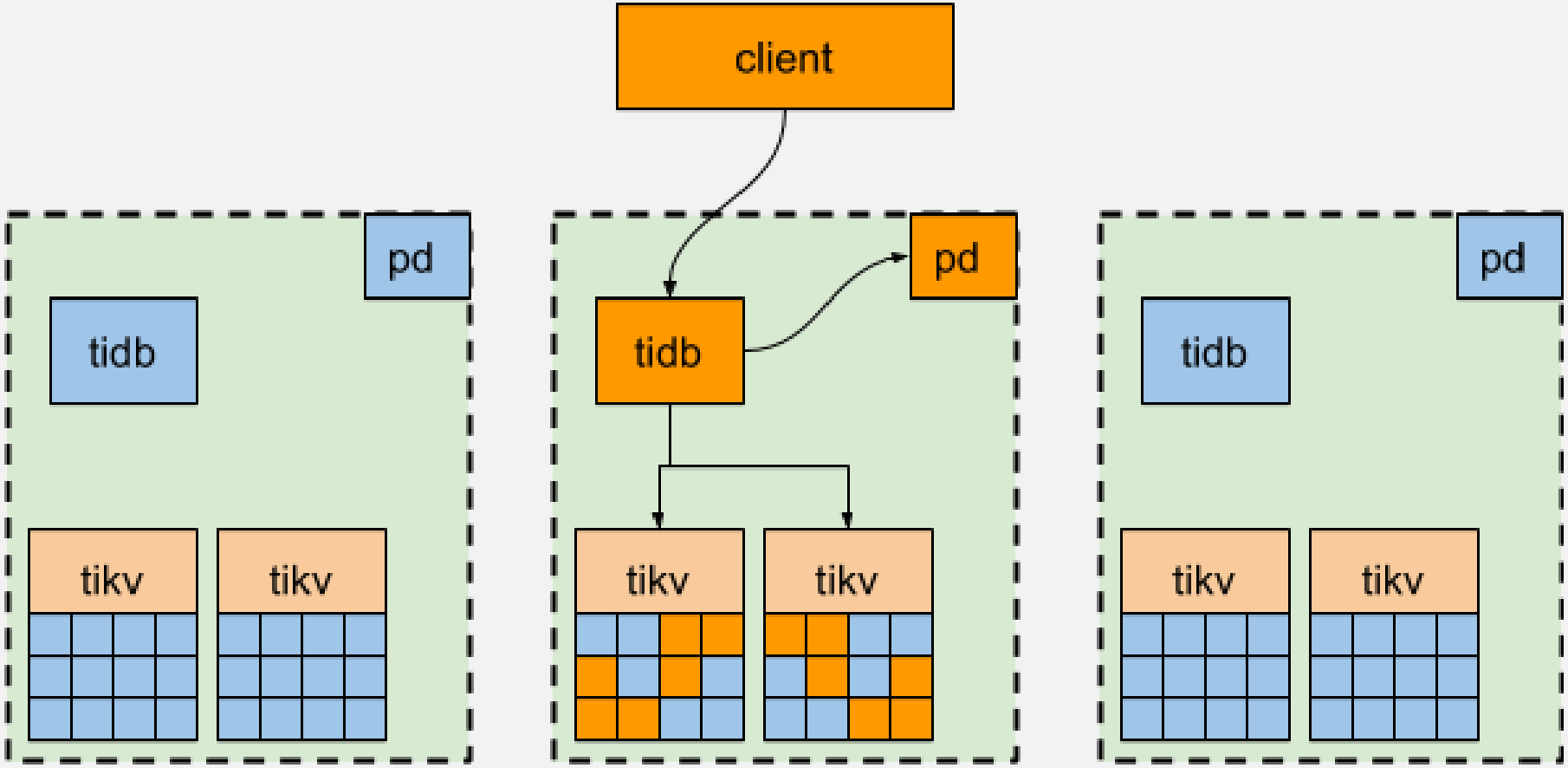


TiDB：同城多数据中心部署

- 同城多数据中心方案提供的保障是，任意一个数据中心故障时，集群能自动恢复服务，不需要人工介入，并能保证数据一致性。
- 优点：
  - 所有数据的副本分布在三个数据中心，具备高可用和容灾能力
  - 任何一个数据中心失效后，不会产生任何数据丢失 (RPO = 0)
  - 任何一个数据中心失效后，其他两个数据中心会自动发起 leader election，并在合理长的时间内（通常情况 20s 以内）自动恢复服务

- 缺点：
  - 性能受网络延迟影响。具体影响如下：
    - 对于写入的场景，所有写入的数据需要同步复制到至少 2 个数据中心，由于 TiDB 写入过程使用两阶段提交，故写入延迟至少需要 2 倍数据中心间的延迟。
    - 对于读请求来说，如果数据 leader 与发起读取的 TiDB 节点不在同一个数据中心，也会受网络延迟影响。
    - TiDB 中的每个事务都需要向 PD leader 获取 TSO，当 TiDB 与 PD leader 不在同一个数据中心时，它上面运行的事务也会因此受网络延迟影响，每个有写入的事务会获取两次 TSO。

TiDB：同城多数据中心部署



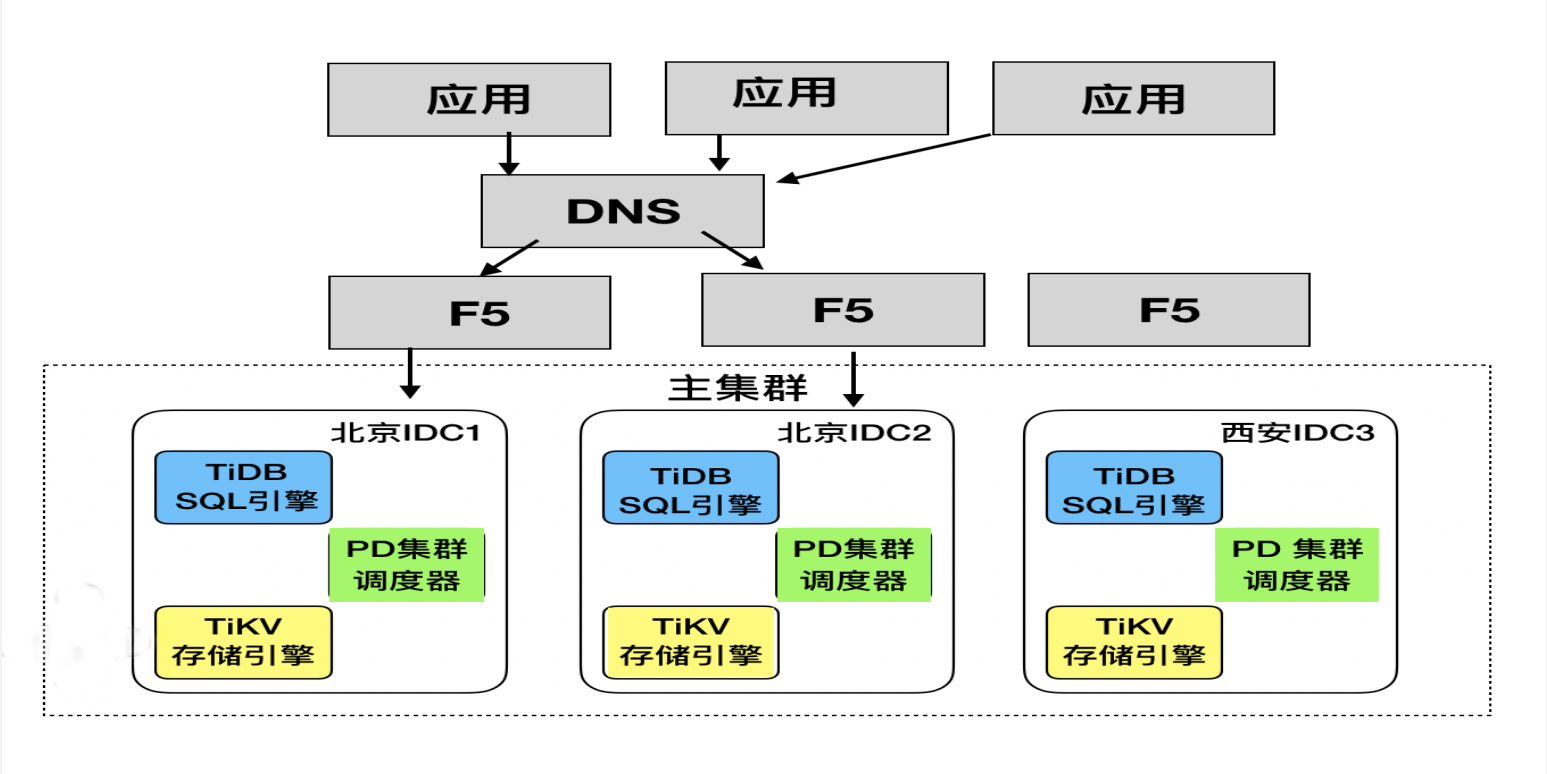
TiDB：同城多数据中心部署

- 可以将业务流量全部派发到一个数据中心，并通过调度策略把 Region leader 和 PD leader 都迁移到同一个数据中心
  - 出现灾备时，leader集体迁移到另一个数据中心

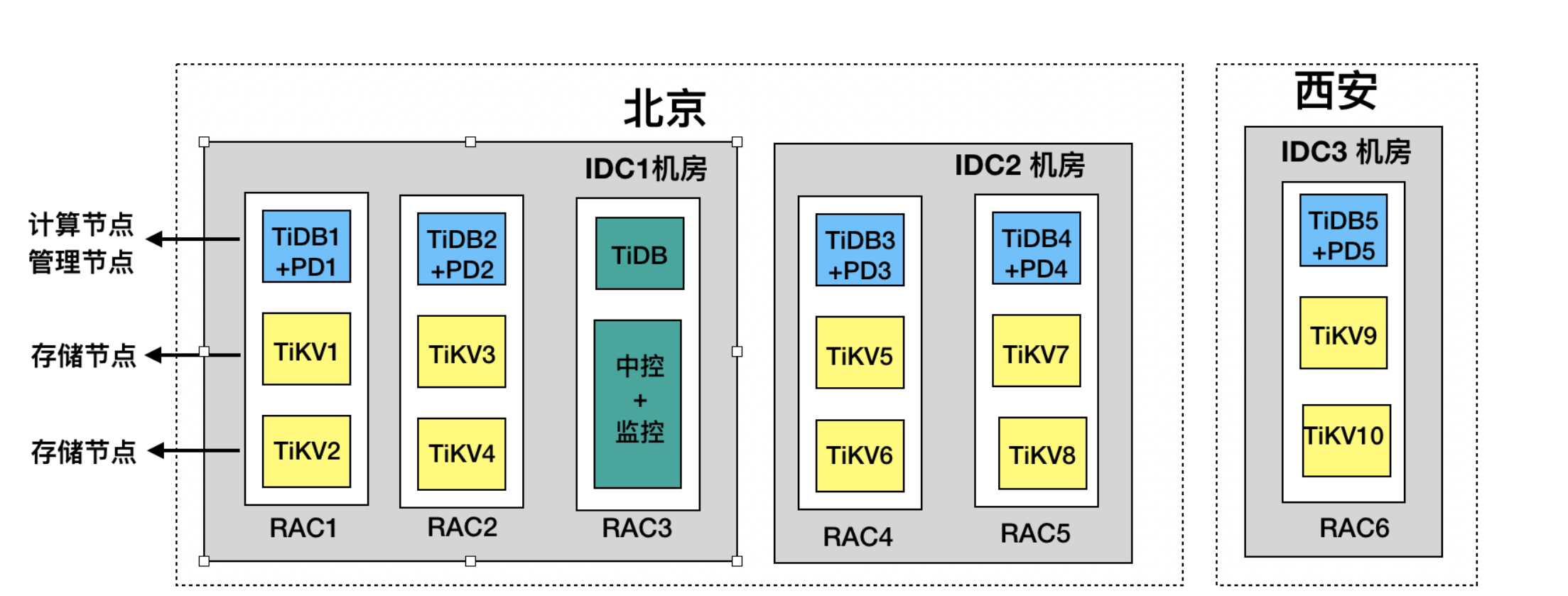
- 优点
  - 集群 TSO 获取能力以及读取性能有所提升
- 缺点：
  - 写入场景仍受数据中心网络延迟影响，这是因为遵循 Raft 多数派协议，所有写入的数据需要同步复制到至少 2 个数据中心
  - TiDB Server 数据中心级别单点
  - 业务流量纯走单数据中心，性能受限于单数据中心网络带宽压力
  - TSO 获取能力以及读取性能受限于业务流量数据中心集群 PD、TiKV 组件是否正常，否则仍受跨数据中心网络交互影响

TiDB：两地三中心部署

- 生产数据中心、同城灾备中心、异地灾备中心的高可用容灾方案
  - 业务流量同时派发到同城两个数据中心，并通过控制 Region Leader 和 PD Leader 分布实现同城数据中心共同负载业务流量
  - Region Leader 尽量只出现在同城的两个数据中心



TiDB：两地三中心部署



TiDB：两地三中心部署

- 优点
  - Region Leader 都在同城低延迟机房(<3ms)，数据写入速度更优
  - 两中心可同时对外提供服务，资源利用率更高
  - 可保证任一数据中心失效后，服务可用并且不发生数据丢失
- 缺点
  - 因为数据一致性是基于 Raft 算法实现，当同城两个数据中心同时失效时，因为异地灾备中心只剩下一份副本，不满足 Raft 算法大多数副本存活的要求。最终将导致集群暂时不可用，需要从一副本恢复集群，只会丢失少部分还没同步的热数据。这种情况出现的概率是比较小的
  - 由于使用到了网络专线(20ms)，导致该架构下网络设施成本较高
  - 两地三中心需设置 **5 副本**，数据冗余度增加，增加空间成本

参考

- 《数据密集型应用设计》
- 《MySQL高可用》第2版
- 同城双活与异地多活架构分析 https://zhuanlan.zhihu.com/p/237782153
- 三地五中心 https://www.jianshu.com/p/aff048130bed
- **TIDB同城多数据中心部署** https://docs.pingcap.com/zh/tidb/dev/multi-data-centers-in-one-city-deployment
- **多活架构思考总结**http://blogxin.cn/2019/03/02/multi-datacenter/