msup

微服务单元化架构实践

讲师简介

张乐

腾讯云技术专家

目前就职于腾讯云中间件团队，曾就职于携程、蚂蚁。

专注于中间件微服务领域研发工作，同时也是开源爱好者，参与过的开源产品包括：

- 开源配置中心 Apollo
- Spring Cloud Tencent
- 服务注册治理中心 Polaris

目录

1 两地三中心架构演进

2 单元化架构实践

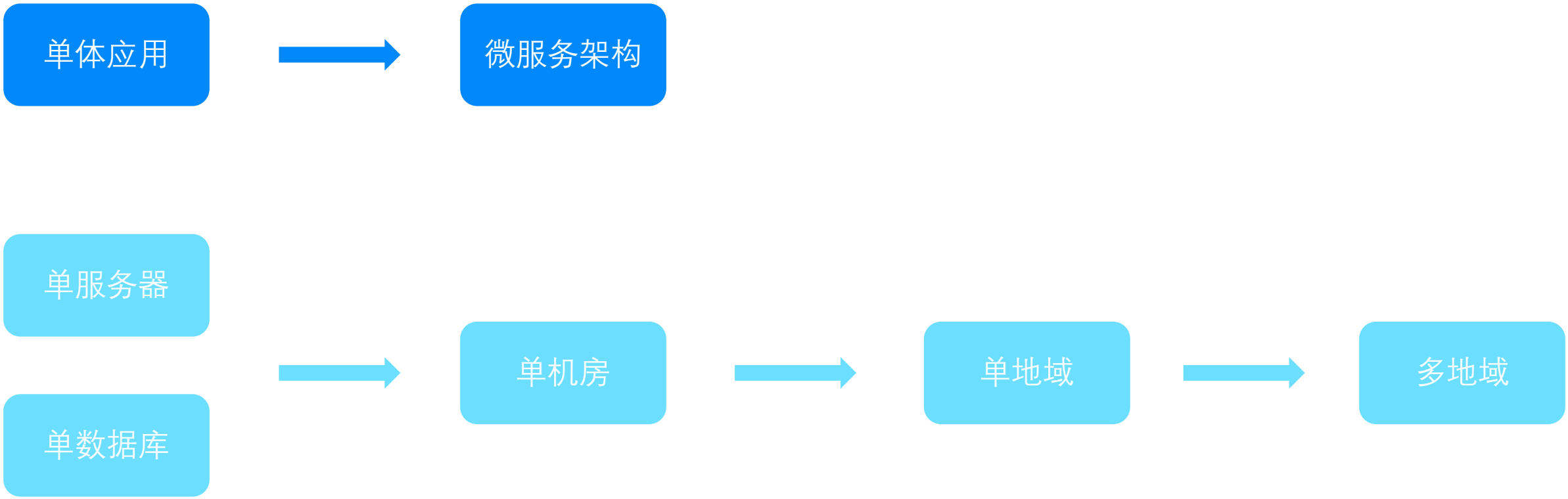
3 中间件在单元化架构中作用

01

两地三中心架构演进

01
02
03

演进



转账场景


转账： ￥50
uid：10086
uid：10000


9月2日 下午17:15
￥50.00
微信转账
9月2日 下午17:24
已收钱
￥50.00
微信转账

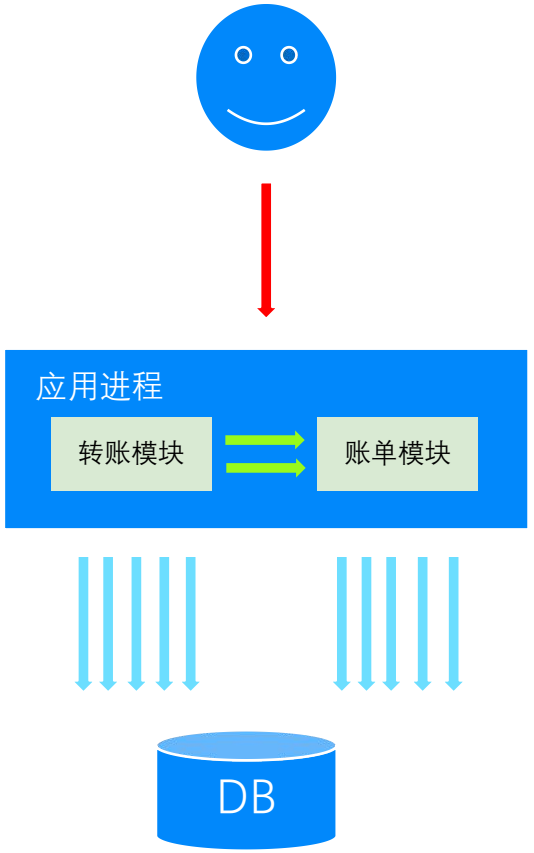
单体应用

一次请求链路

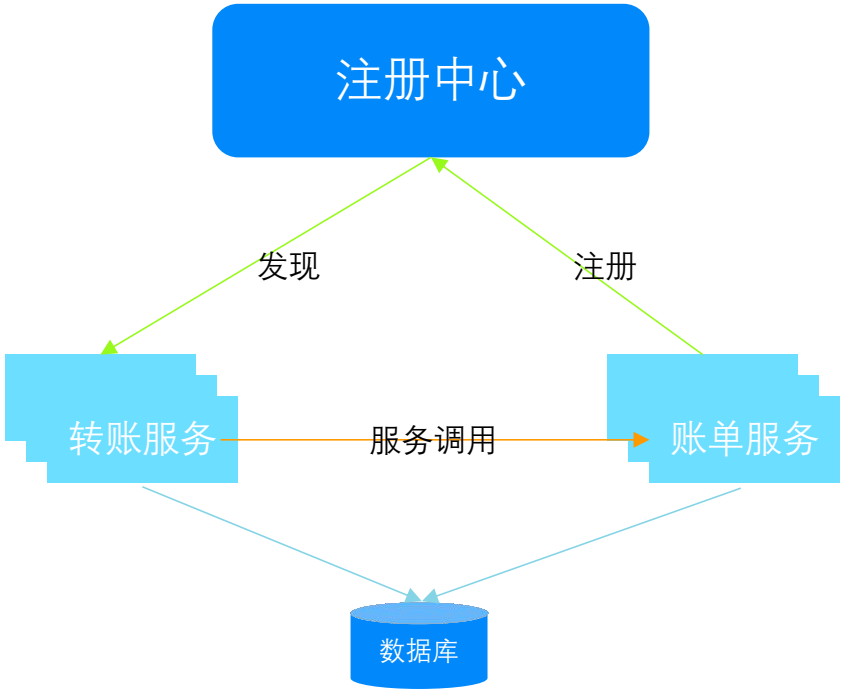
- 异地请求 1 次 （数 10ms）
- 进程内调用 N 次 （无消耗）
- 数据库访问 10 次 （忽略不计）

特点

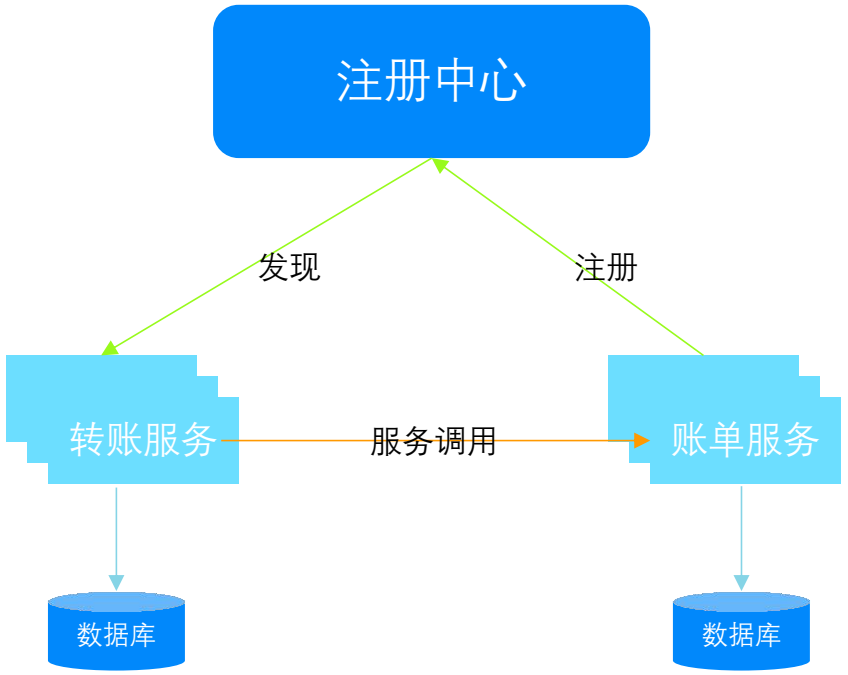
- 网络开销小，延迟低
- 架构简单
- 模块之间互相影响
- 代码开发耦合



微服务拆分



过渡态：数据库未拆分



终态：拆分数据库

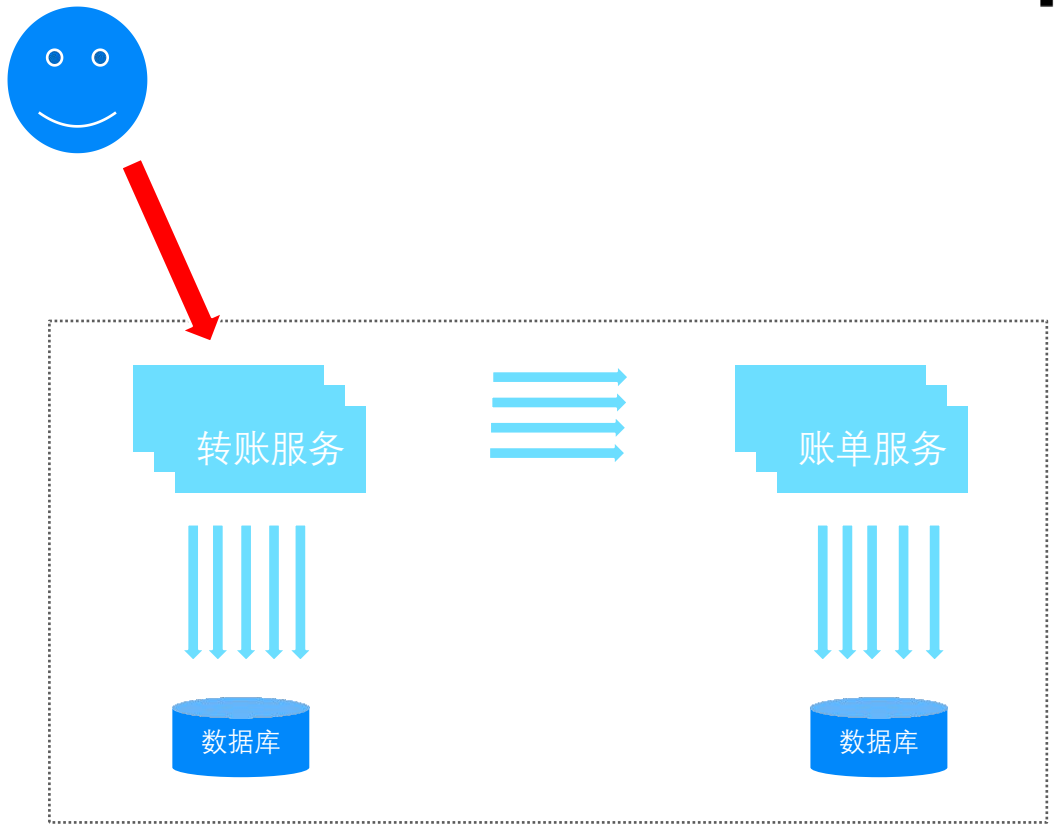
单机房微服务

一次请求链路

- 异地请求 1 次 （数 10ms）
- 同机房 RPC 调用 N 次 （忽略不计）
- 数据库访问 10 次 （忽略不计）

特点

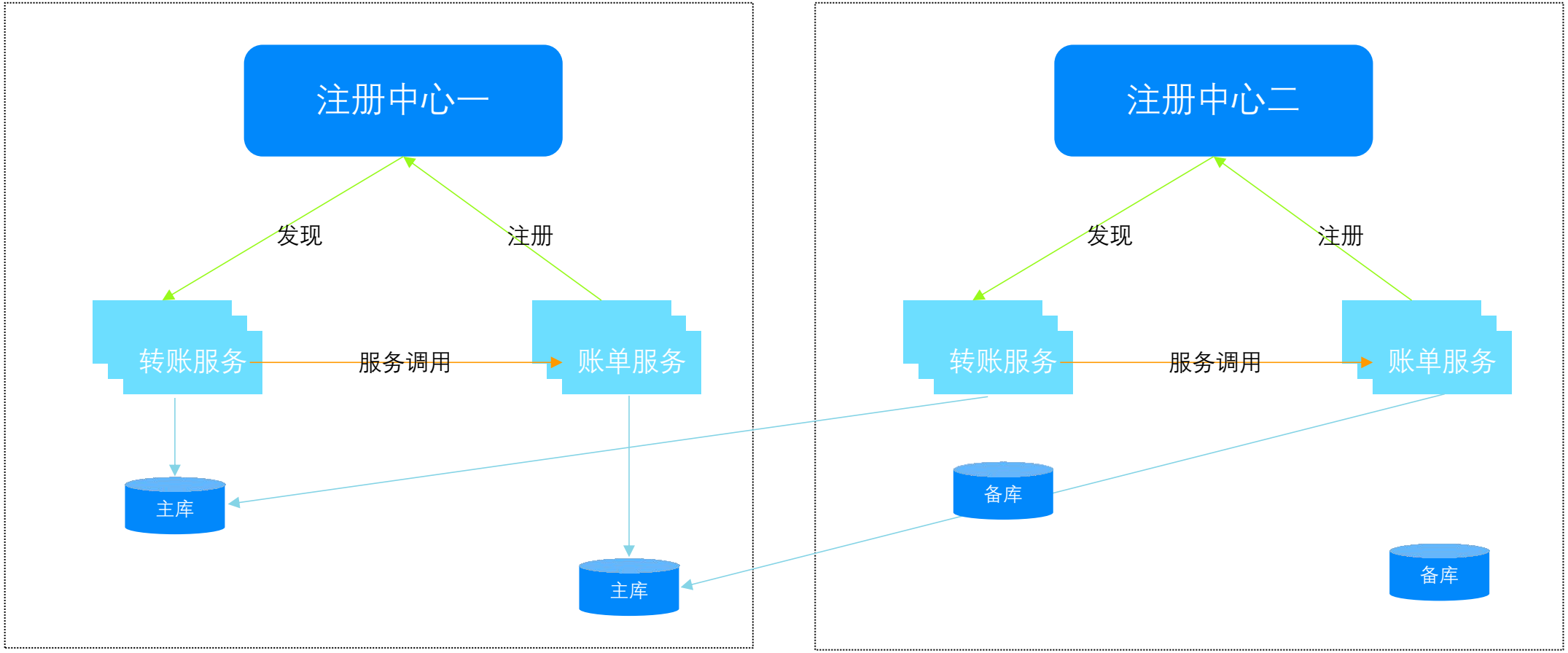
- 业务按模块拆分成微服务，业务解耦
- 架构变得复杂，引入一系列中间件，
  例如：服务注册发现、链路追踪
- 单机房网络开销小，延迟低



机房一

同城双机房架构一（不建议）

1. 通过注册中心隔离，RPC流量机房内收敛。无法跨机房服务调用。只能做到机房级容灾 2. 数据库出现跨机房调用

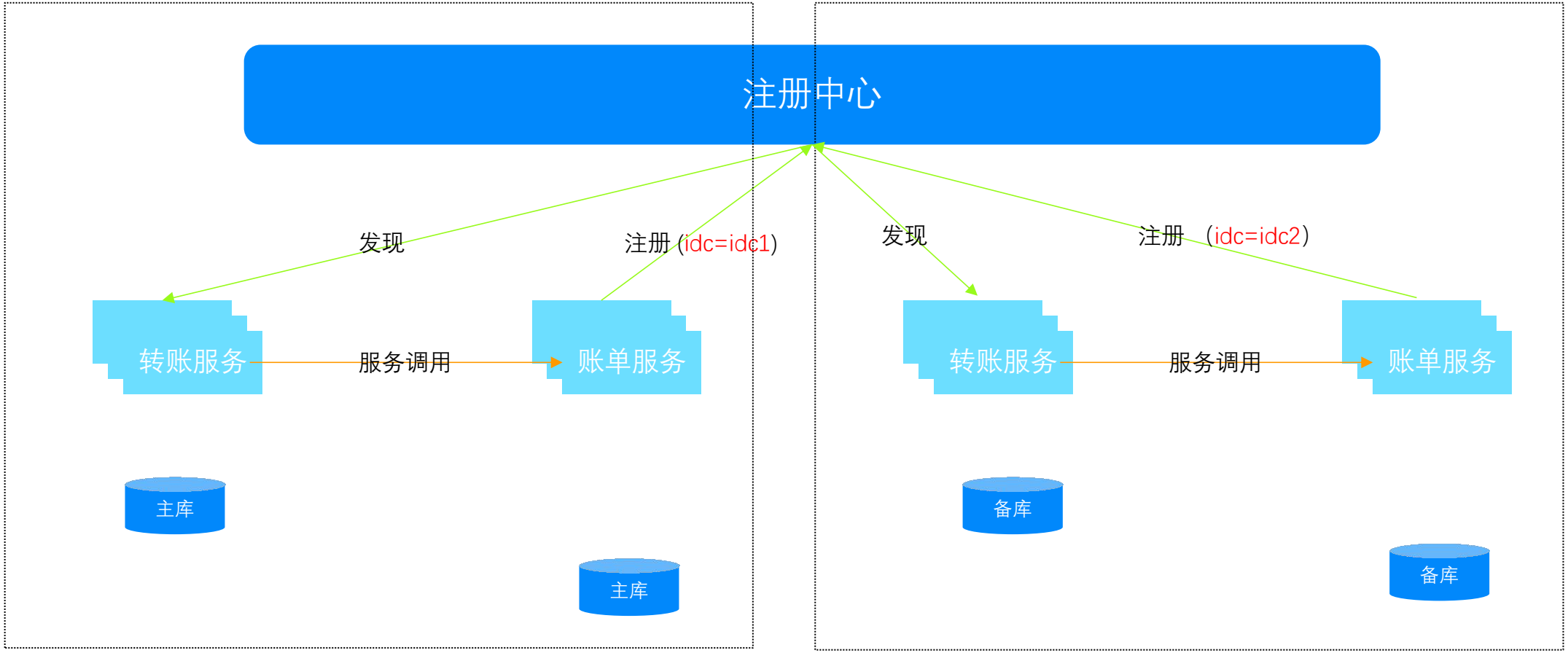


机房一

机房二

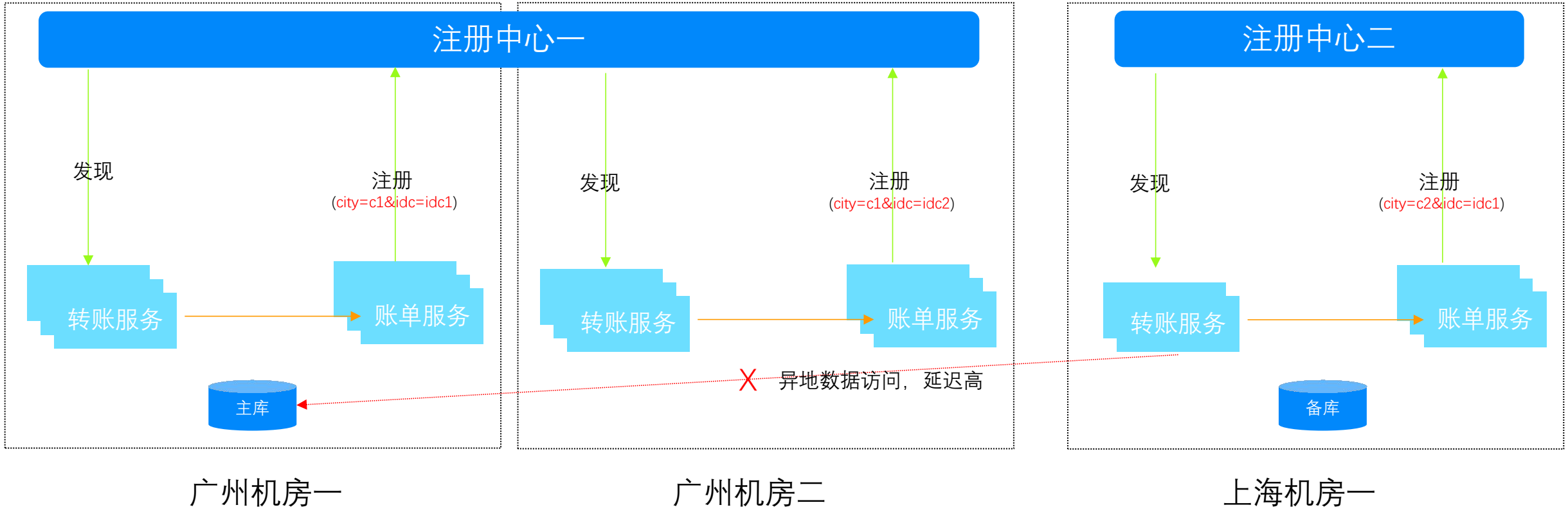
同城双机房架构二（建议）

1. 通过服务路由组件，灵活控制 RPC 流量 2. 可以做到服务级机房容灾



两地三中心架构（不建议）

1. 由于数据库跨城访问，无法异地多活 2. 资源利用率低 3. 数据库备节点数据一致性无法保证 4. 容灾切换时，备机房可用性无法保证
5. 实际场景中，异地机房常用于跑离线任务，主机房腾出资源给在线业务。



小结

架构形态	好处	问题	适合场景
单体架构	1. 简单 2. 资源消耗少 3. 无网络开销	1. 故障不隔离 2. 维护困难	适合绝大部分规模小的业务
微服务架构	1. 业务解耦 2. 可维护性高	1. 各个维度变得复杂，例如：基础设施、排查问题、研发测试环境	有基础架构维护能力且业务具有一定规模的中大型业务场景
单机房架构	1. 成本低 2. 架构简单	1. 不具备机房级容灾能力 2. 单机房容量优先，无法一直扩展业务	业务规模小，容灾时效性不高
同城双机房架构一 （物理隔离注册中心）	1. 机房内流量收敛 2. 具备机房级容灾能力	1. 无法跨机房服务调用，不具备服务级容灾能力 2. 跨机房数据访问	不建议
同城双机房架构三 （一个逻辑注册中心+本机房优先）	1. 具备机房、服务级容灾能力 2. 正常情况下，同机房流量收敛 3. 可跨机房服务调用	1. 跨机房数据访问	强烈建议，可水平扩容机房
两地三中心架构	1. 具备城市级容灾能力	1. 无法异地多活，资源利用率低 2. 容灾切换时，可用性无法保证 3. 数据一致性无法保证	不建议

02

单元化架构实践

01
02
03

核心问题

msup

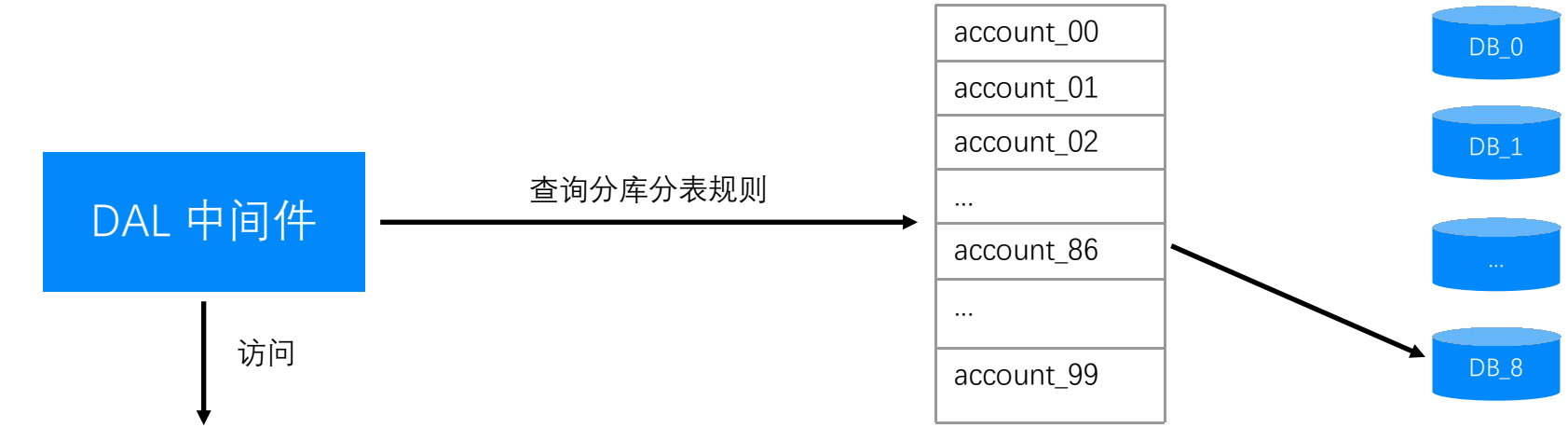
如何实现异地多机房在线业务多活呢?

核心问题

1. 异地数据库访问延迟问题
2. 有限的数据库连接资源，支撑不了无限水平扩容

分库分表

Select * from account where userId = "10086";

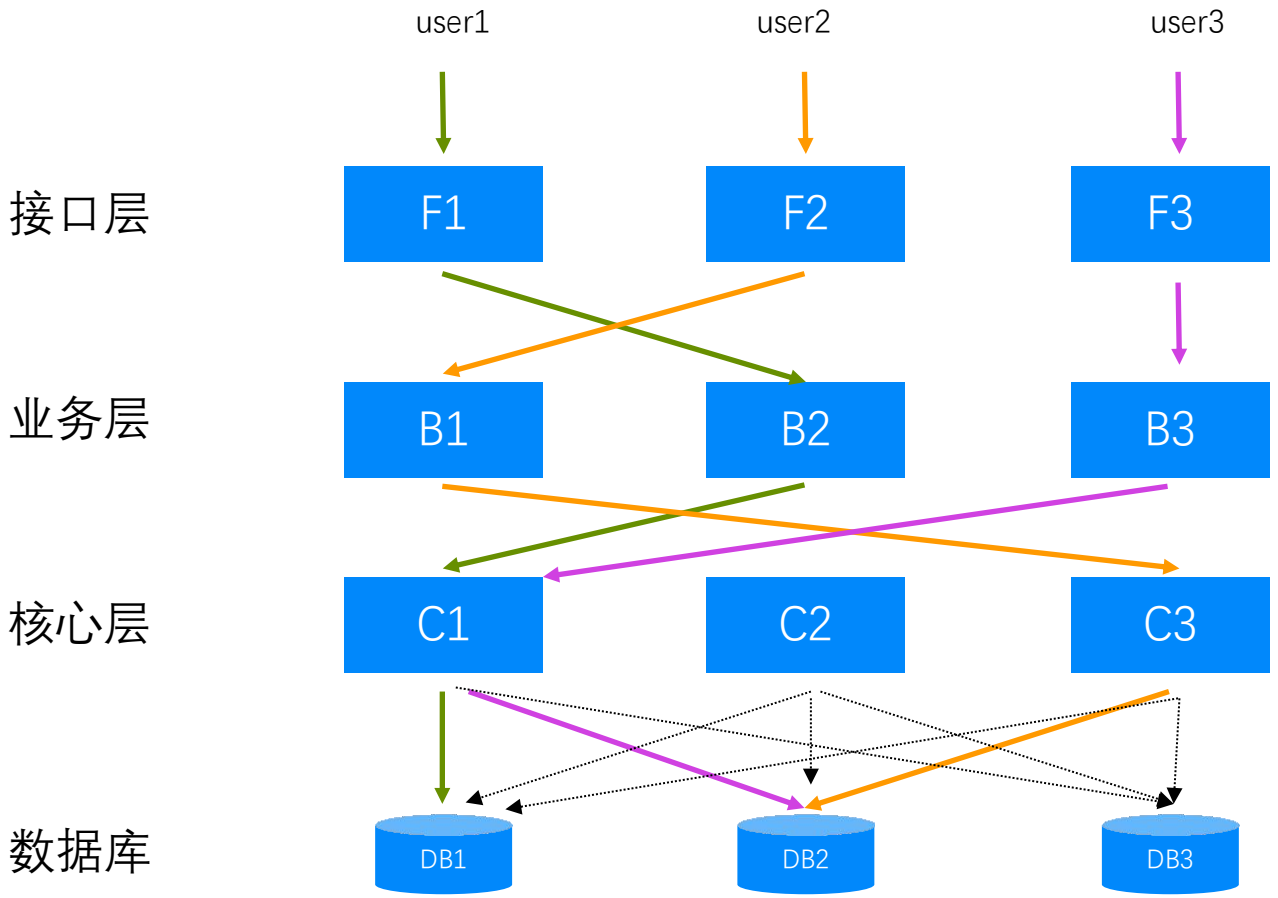


DB_8#account_86 表

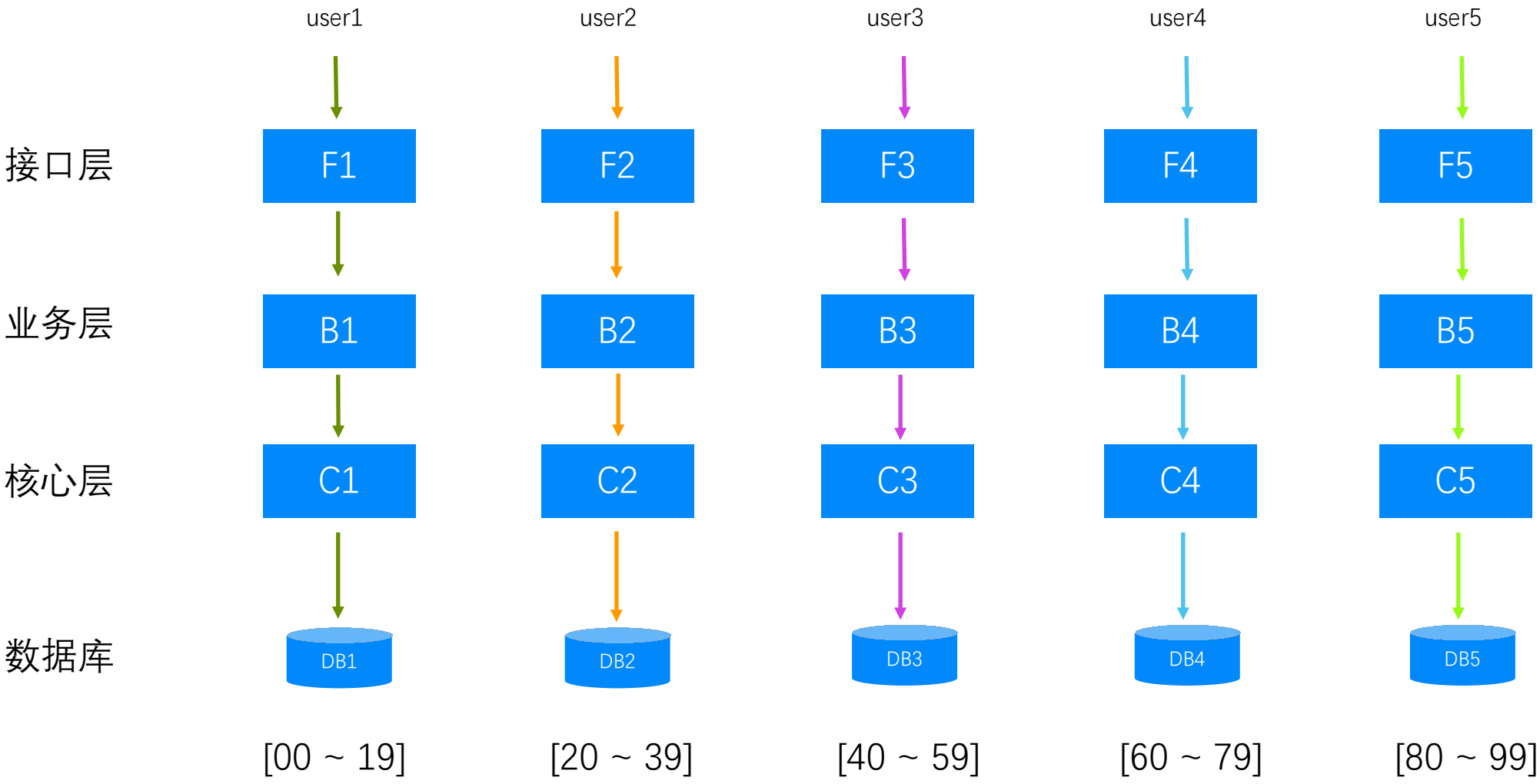
userId	balance
11086	10000
10086	5000
98786	100

十库百表

数据库连接瓶颈


user1
user2
user3
接口层
F1
F2
F3
业务层
B1
B2
B3
核心层
C1
C2
C3
数据库
DB1
DB2
DB3

“单元化”模型


user1
user2
user3
user4
user5
接口层
F1
F2
F3
F4
F5
业务层
B1
B2
B3
B4
B5
核心层
C1
C2
C3
C4
C5
数据库
DB1
DB2
DB3
DB4
DB5
[00 ~ 19]
[20 ~ 39]
[40 ~ 59]
[60 ~ 79]
[80 ~ 99]

单元化设计原则

1. 核心业务单元化，单元内收敛
2. 保证核心业务单元分片均衡（UID）
3. 面向逻辑分区设计，而不是物理部署

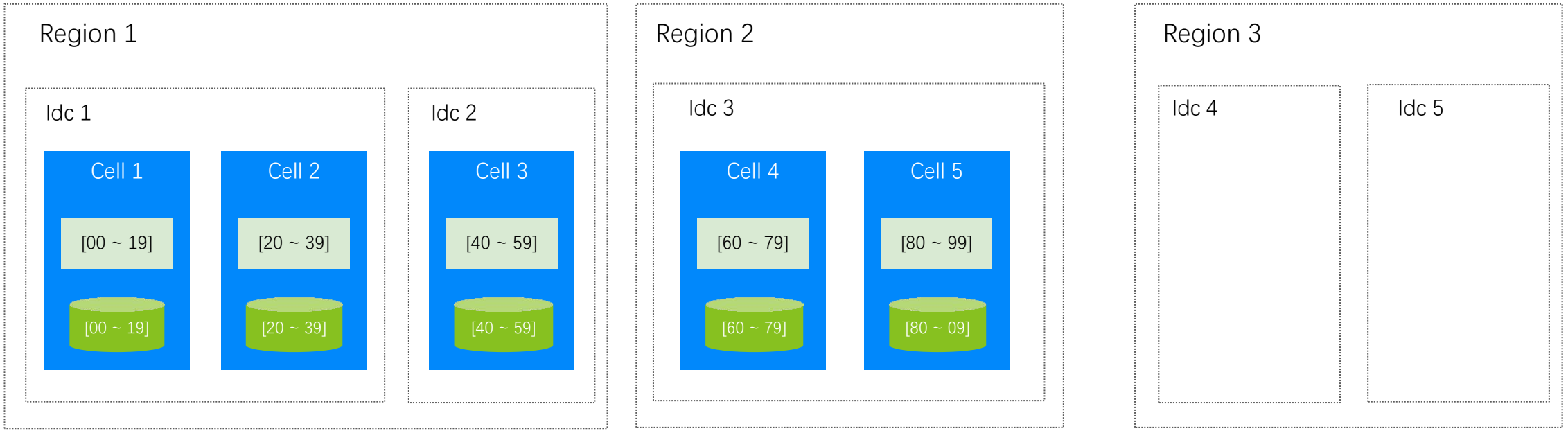
逻辑单元与物理数据中心模型

两地三中心

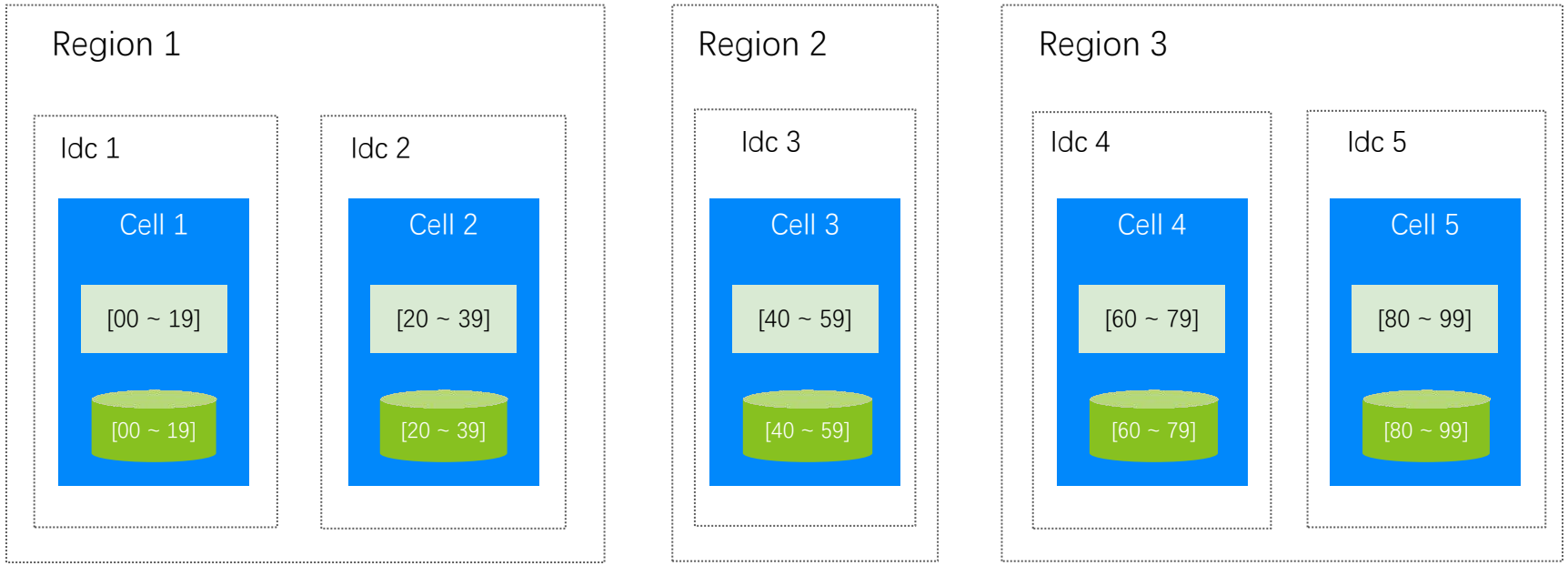
单元路由表

分片	Cell	Region	Idc
[00 ~ 19]	C1	R1	I1
[20 ~ 39]	C2	R1	I1
[40 ~ 59]	C3	R1	I2
[60 ~ 79]	C4	R2	I3
[80 ~ 99]	C5	R2	I3

向三地五中心演进


Region 1
Idc 1
Cell 1
[00 ~ 19]
[00 ~ 19]
Cell 2
[20 ~ 39]
[20 ~ 39]
Idc 2
Cell 3
[40 ~ 59]
[40 ~ 59]
Region 2
Idc 3
Cell 4
[60 ~ 79]
[60 ~ 79]
Cell 5
[80 ~ 99]
[80 ~ 09]
Region 3
Idc 4
Idc 5

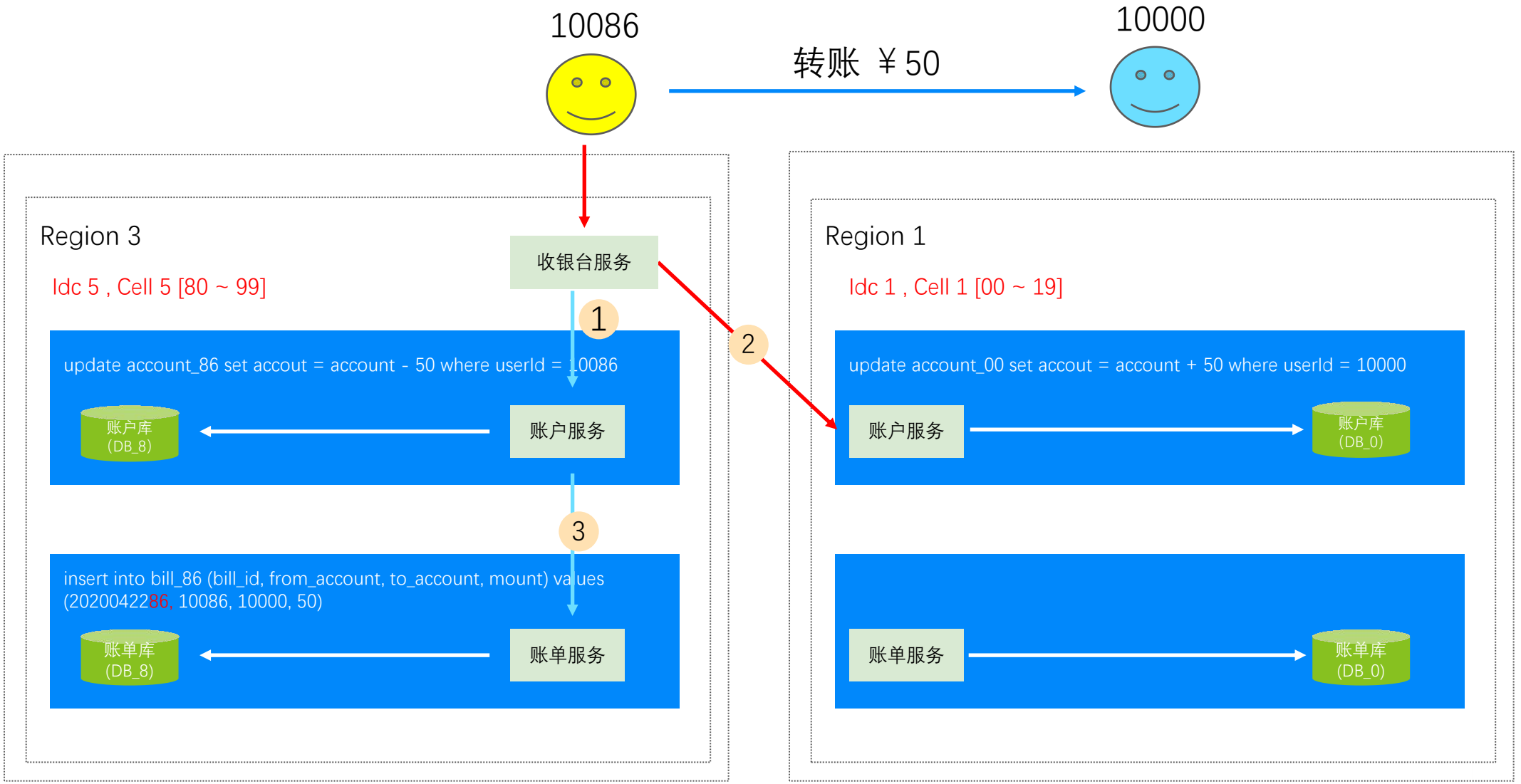
三地五中心终态



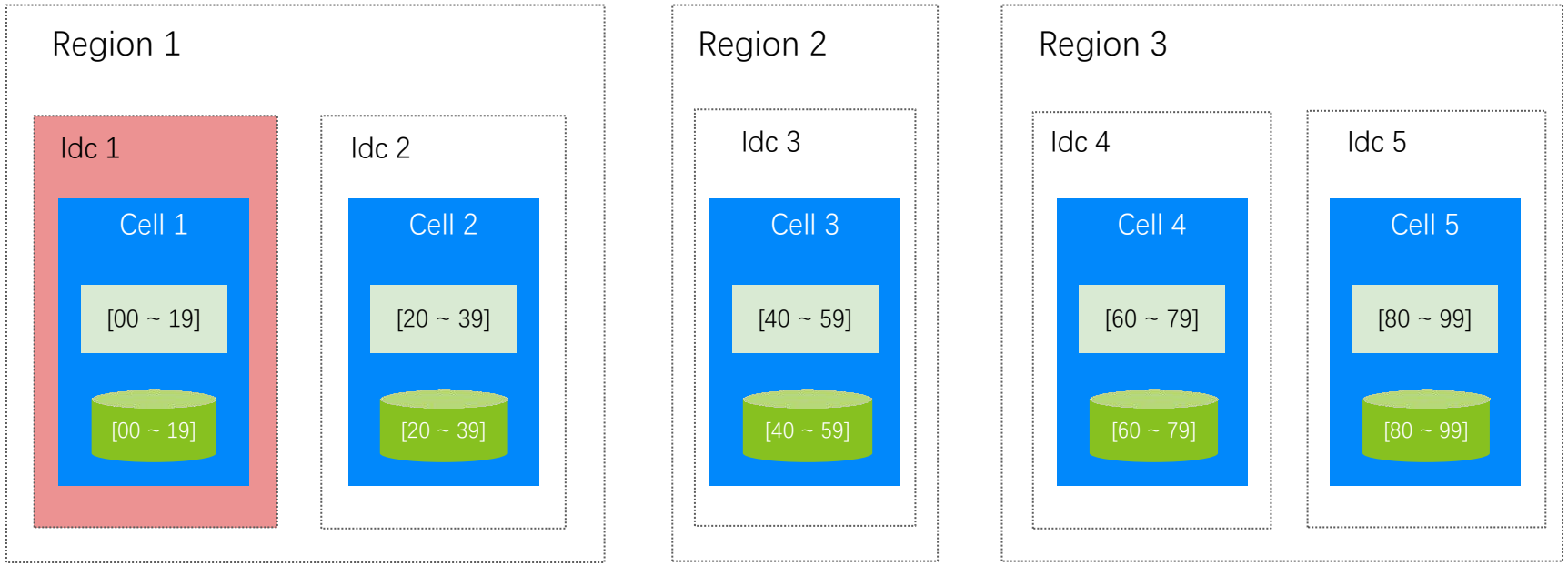
单元路由表

分片	Cell	Region	Idc
[00 ~ 19]	C1	R1	I1
[20 ~ 39]	C2	R1	I2
[40 ~ 59]	C3	R2	I3
[60 ~ 79]	C4	R3	I4
[80 ~ 99]	C5	R3	I5

跨区调用


10086
转账 ￥50
10000
Region 3
Idc 5 , Cell 5 [80 ~ 99]
收银台服务
1
update account_86 set accout = account - 50 where userId = 10086
账户库
(DB_8)
账户服务
2
3
insert into bill_86 (bill_id, from_account, to_account, mount) values
(2020042286, 10086, 10000, 50)
账单库
(DB_8)
账单服务
Region 1
Idc 1 , Cell 1 [00 ~ 19]
update account_00 set accout = account + 50 where userId = 10000
账户服务
账户库
(DB_0)
账单服务
账单库
(DB_0)

单元级容灾



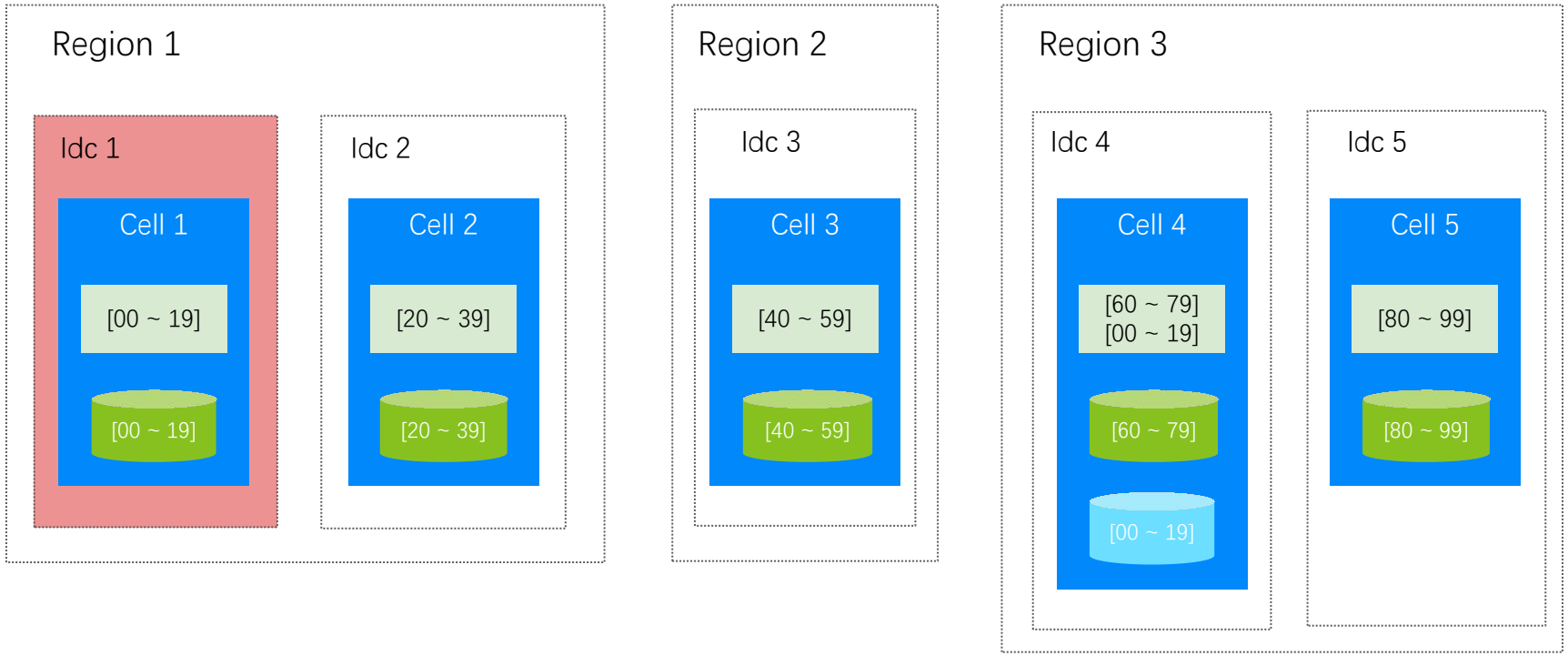
单元路由表

分片	Cell	Region	Idc
[00 ~ 19]	C1	R1	I1
[20 ~ 39]	C2	R1	I2
[40 ~ 59]	C3	R2	I3
[60 ~ 79]	C4	R3	I4
[80 ~ 99]	C5	R3	I5

单元&城市容灾

msup

单元间有容灾互备关系，例如：Cell1 的备单元是 Cell4，那么就会在 Cell4 部署备数据库



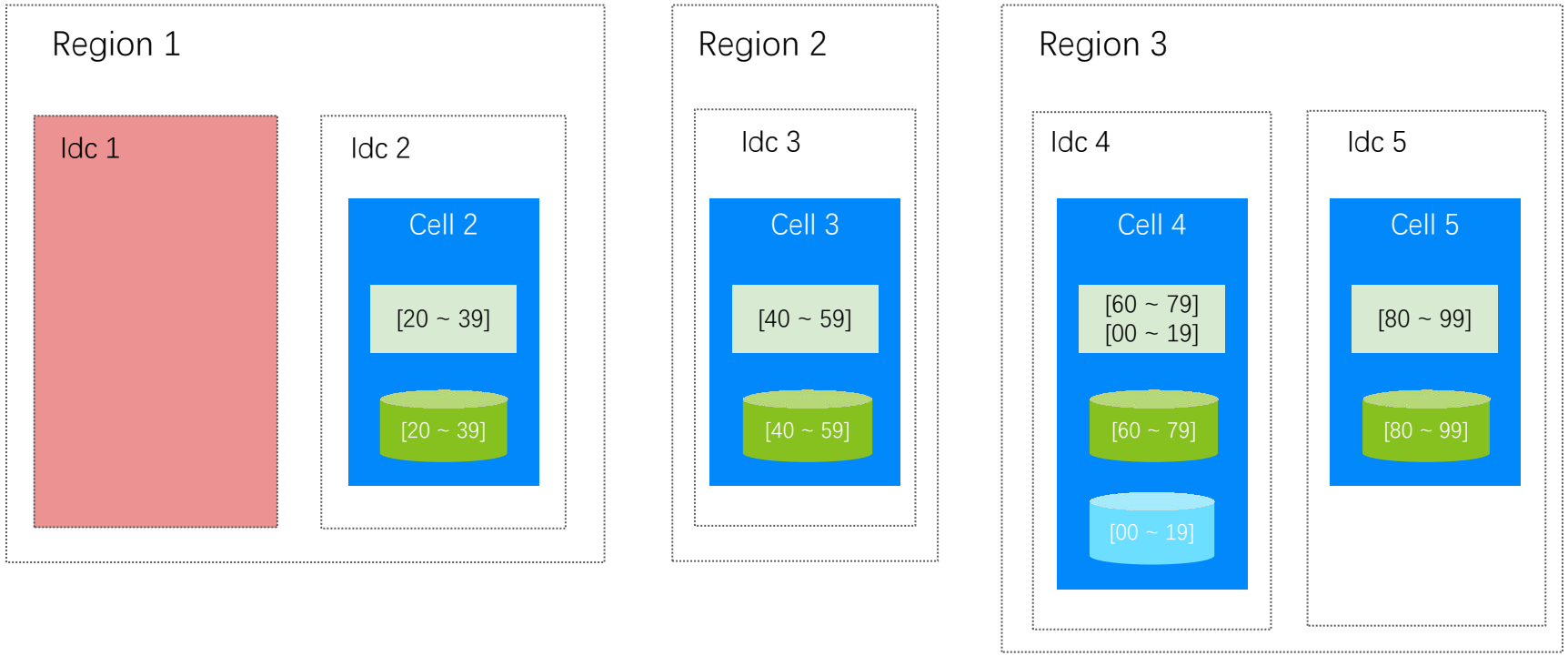
单元路由表

分片	Cell	Region	Idc
[00 ~ 19]	C4	R3	I4
[20 ~ 39]	C2	R1	I2
[40 ~ 59]	C3	R2	I3
[60 ~ 79]	C4	R3	I4
[80 ~ 99]	C5	R3	I5

单元&城市容灾

msup

单元间有容灾互备关系，例如：Cell1 的备单元是 Cell4



单元路由表

分片	Cell	Region	Idc
[00 ~ 19]	C4	R3	I4
[20 ~ 39]	C2	R1	I2
[40 ~ 59]	C3	R2	I3
[60 ~ 79]	C4	R3	I4
[80 ~ 99]	C5	R3	I5

小结

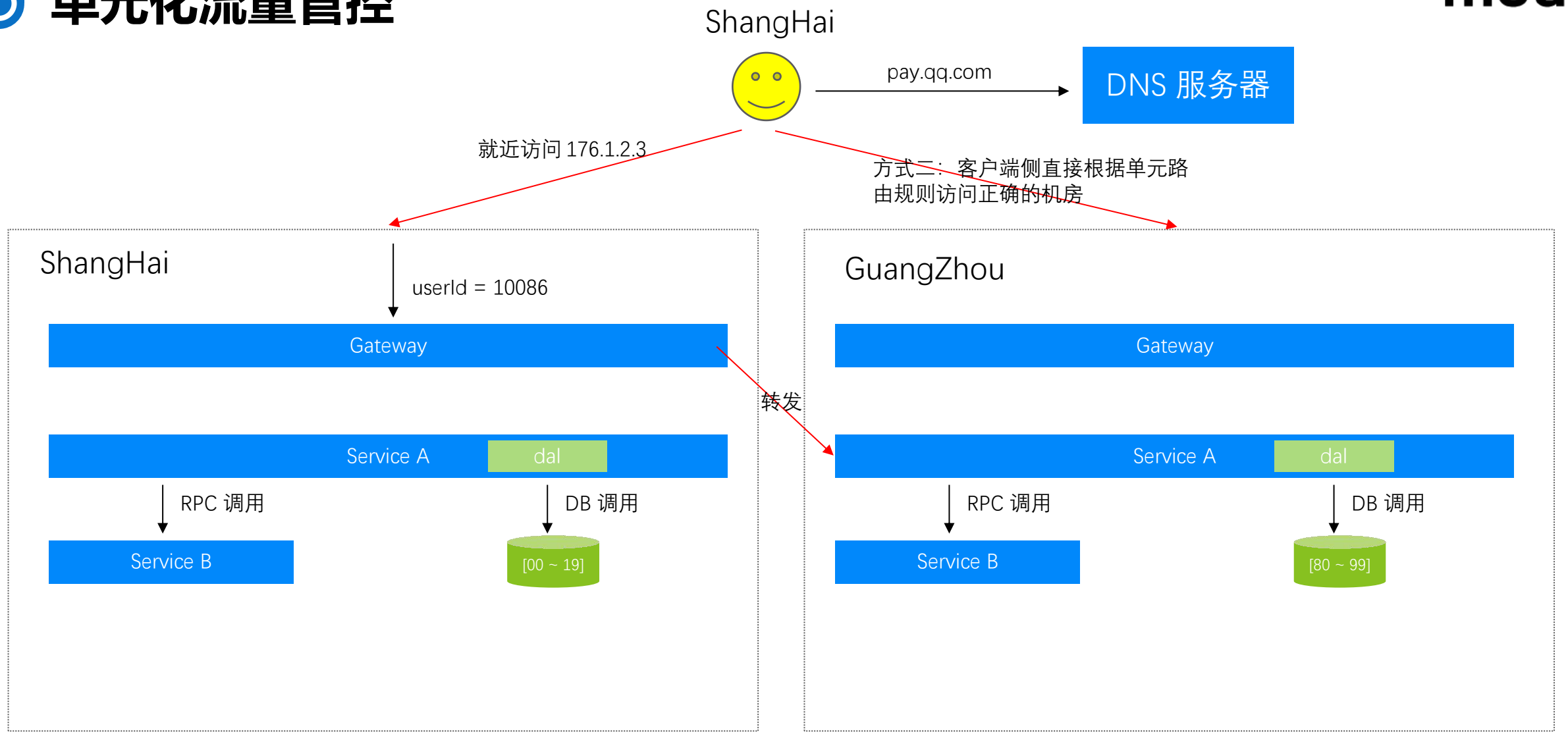
1. 核心业务单元化，单元内收敛
2. 保证核心业务单元分片均衡（UID）
3. 面向逻辑分区设计，而不是物理部署
4. 核心业务尽早分百表，一步到位

03

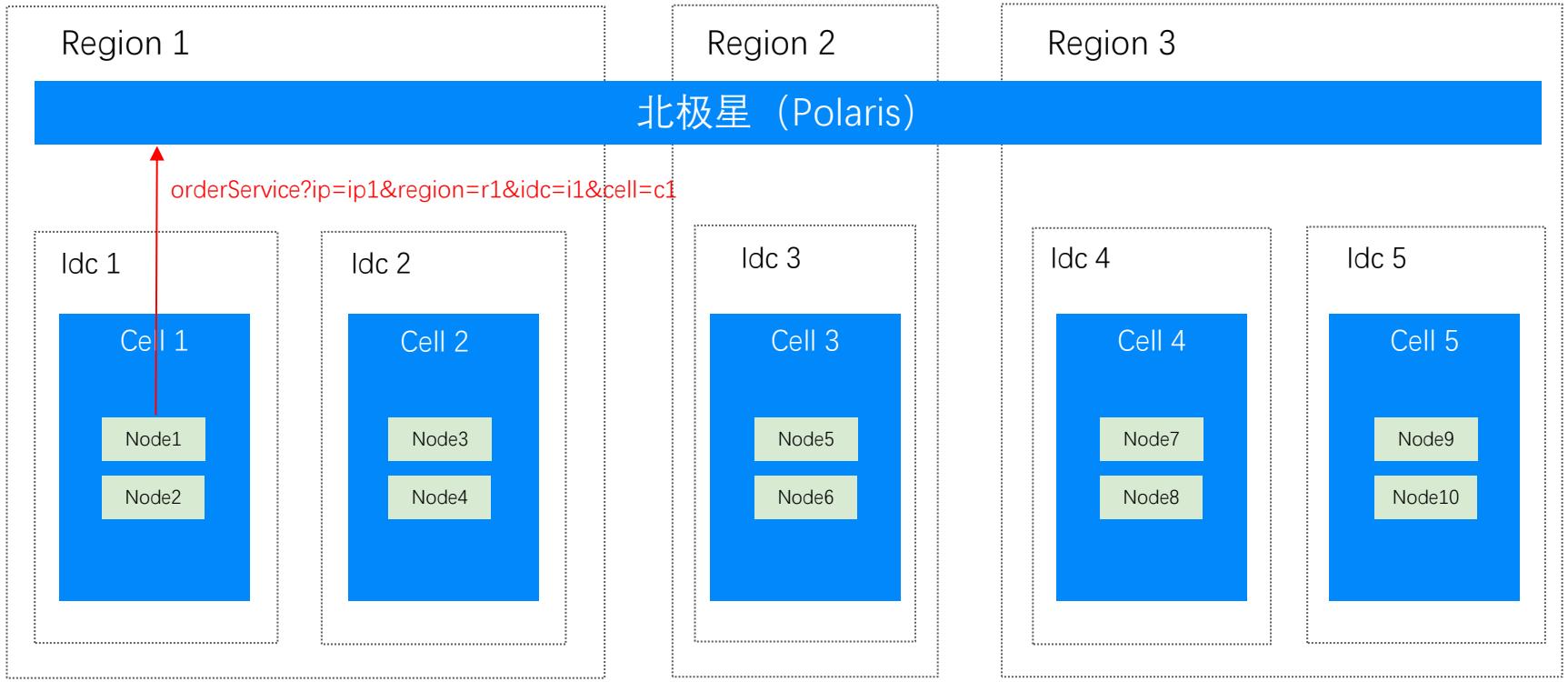
中间件在单元化中的作用

01
02
03

单元化流量管控


ShangHai
pay.qq.com
DNS 服务器
就近访问 176.1.2.3
方式二：客户端侧直接根据单元路
由规则访问正确的机房
ShangHai
userId = 10086
Gateway
Service A
dal
RPC 调用
DB 调用
Service B
[00 ~ 19]
转发
GuangZhou
Gateway
Service A
dal
RPC 调用
DB 调用
Service B
[80 ~ 99]

全局注册中心


Region 1
Region 2
Region 3
北极星（Polaris）
orderService?ip=ip1&region=r1&idc=i1&cell=c1
Idc 1
Idc 2
Idc 3
Idc 4
Idc 5
Cell 1
Cell 2
Cell 3
Cell 4
Cell 5
Node1
Node2
Node3
Node4
Node5
Node6
Node7
Node8
Node9
Node10

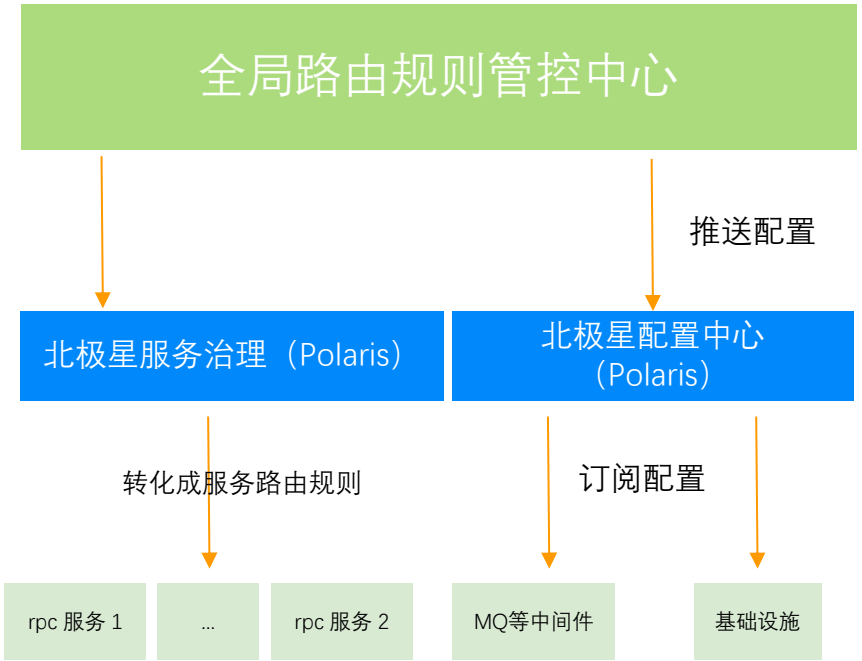
RPC 框架如何知道 UserId

- 如果是 Http 接口，统一规定 X-UserId Header 字段
- 如果是 trpc、dubbo 接口
  - 方案一：第一个参数固定为 UserId 字段
  - 方案二：通过在每个方法上，增加注解，自定义 UserIdResolver 解析器

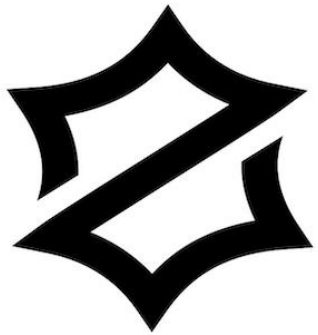
单元化路由规则

核心单元化路由规则

分片	Cell	Region	Idc
[00 ~ 19]	C1	R1	I1
[20 ~ 39]	C2	R1	I2
[40 ~ 59]	C3	R2	I3
[60 ~ 79]	C4	R3	I4
[80 ~ 99]	C5	R3	I5



Polaris



北极星

一个支持多语言、多框架的云原生服务发现和治理中心

Github

官网主页

亮点一：注册中心和服务治理整合在一起

本质上注册中心和部分服务治理能力解决的是流量去哪里的问题

1. 注册中心提供了全量的目标服务地址
2. 负载均衡、服务路由、服务降级则从全量的目标地址中挑一个地址

亮点二：无状态服务

北极星自身服务为无状态服务，计算存储分离。从而具备高扩展性、极低的运维成本，完美适配云原生体系。

亮点三：打通 k8s 集群

通过 k8s controller 同步 k8s 集群服务和 Endpoint 数据，做到跨多 k8s 集群，虚拟机和 Pod 互相通信

亮点四：提供了配置中心的能力

北极星也提供了动态配置管理的能力

小结

1. 网关在入口处转发请求到正确的单元
2. 全局注册中心，通过标签区分单元
3. 全局统一路由规则管理，配置中心下发路由规则到所有应用
4. RPC 框架支持单元化路由，对业务透明
5. DAL 层框架兜底，确保数据一致性

总结

1. 适合公司自身业务发展的架构才是好的架构
2. 单元化架构核心是按用户维度全链路分片
3. 单元化架构实际逻辑比较复杂，每个公司按照自己公司的情况可能有不同的落地方案

msup®

Thanks

关注msup公众号

获取更多AI落地实践

麦思博(msup)有限公司是一家面向技术型企业的培训咨询机构，携手2000余位中外客座导师，服务于技术团队的能力提升、软件工程效能和产品创新迭代，超过3000余家企业续约学习，是科技领域占有率第1的客座导师品牌，msup以整合全球领先经验实践为己任，为中国产业快速发展提供智库。