Seata 在微服务一致性中的探索

季敏

Seata 开源社区发起人

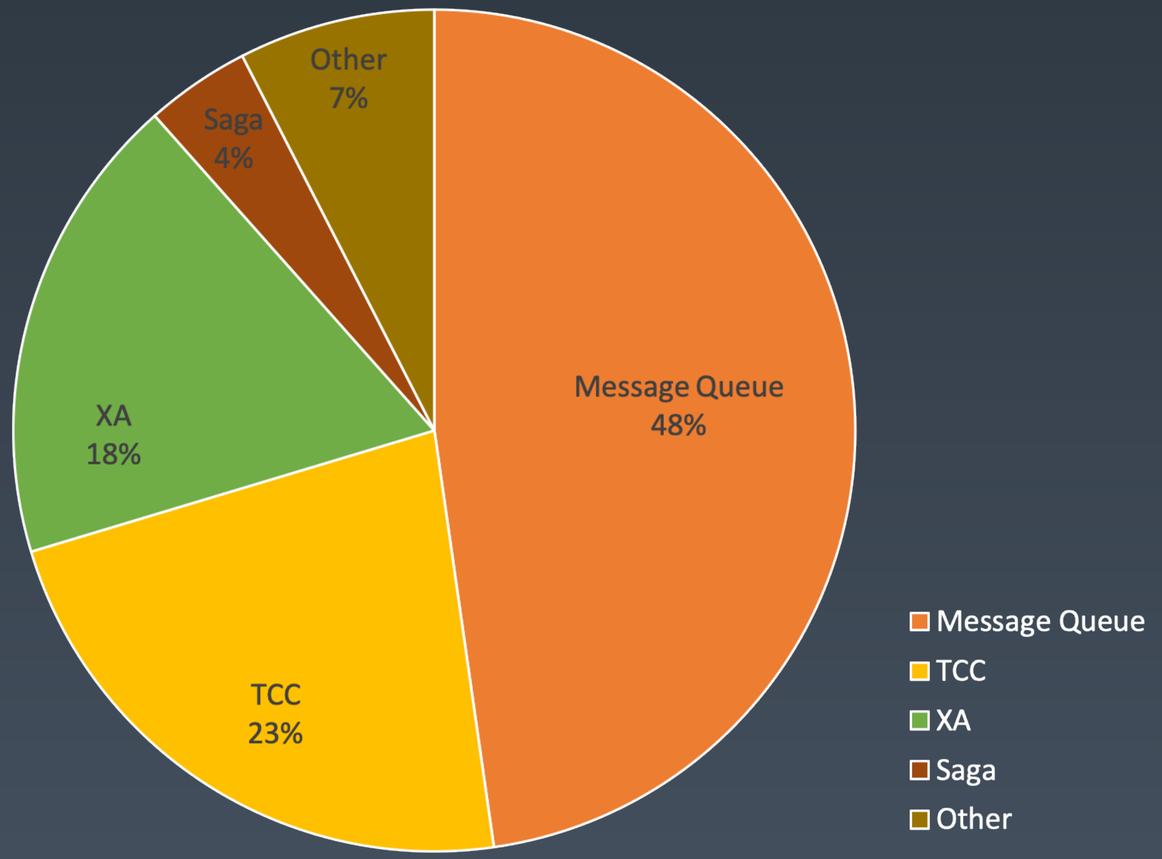
自我介绍

季敏（GitHub ID slievrly），Seata 开源项目负责人。加入阿里巴巴中间件后主要负责 TXC 、GTS 和 Seata 项目的研发，推动了项目在集团内业务的落地，实现了专有云和阿里云的对外技术输出及 Seata 的开源。长期从事于分布式中间件的架构设计与开发工作，在分布式事务领域有着深厚的技术积累。

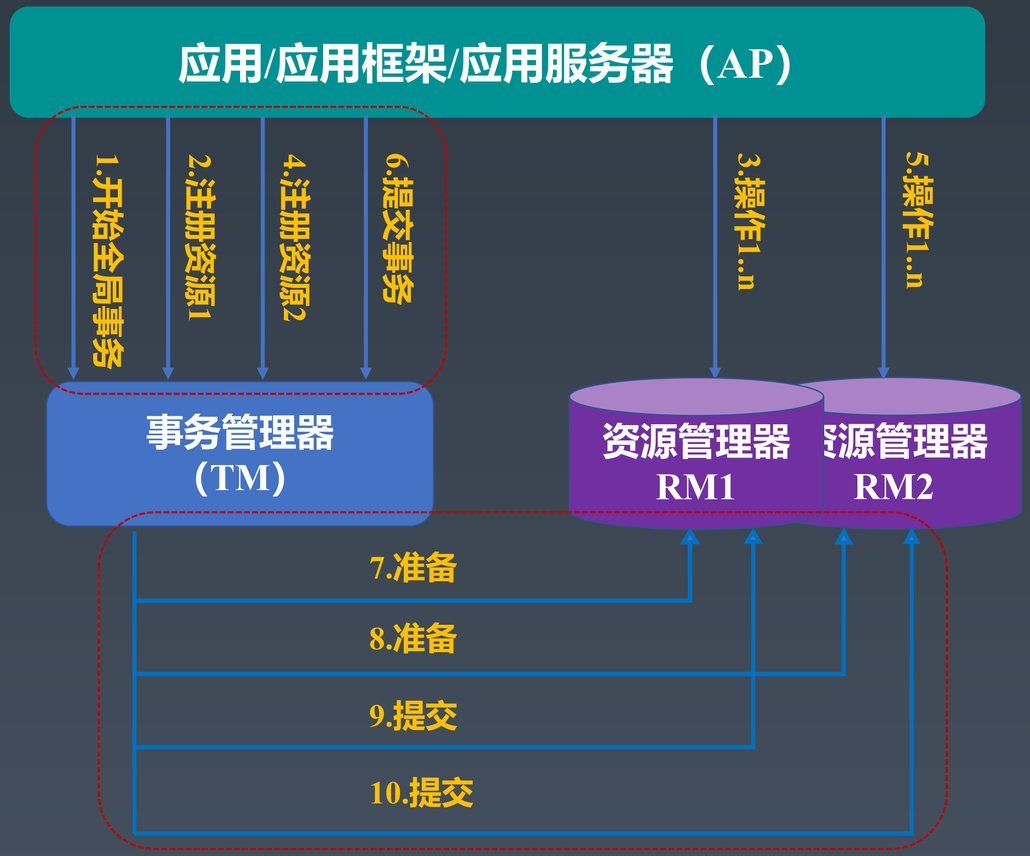
目录

- 常见分布式一致性解决方案介绍
- Seata 原理介绍
- Seata 发展历程与 Roadmap
- 用户案例与推荐阅读

分布式事务现状


Other
7%
Saga
4%
XA
18%
Message Queue
48%
TCC
23%
Message Queue
TCC
XA
Saga
Other

DTP模型--XA



XA是由X/Open组织提出的分布式事务的规范。 XA规范主要定义了(全局)事务管理器(TM)和(局 部)资源管理器(RM)之间的接口。主流的关系型 数据库产品都是实现了XA接口的。

XA接口是双向的系统接口，在事务管理器 （TM）以及一个或多个资源管理器（RM）之 间形成通信桥梁。

XA之所以需要引入事务管理器是因为，在分布 式系统中，从理论上讲两台机器理论上无法达 到一致的状态，需要引入一个单点进行协调。

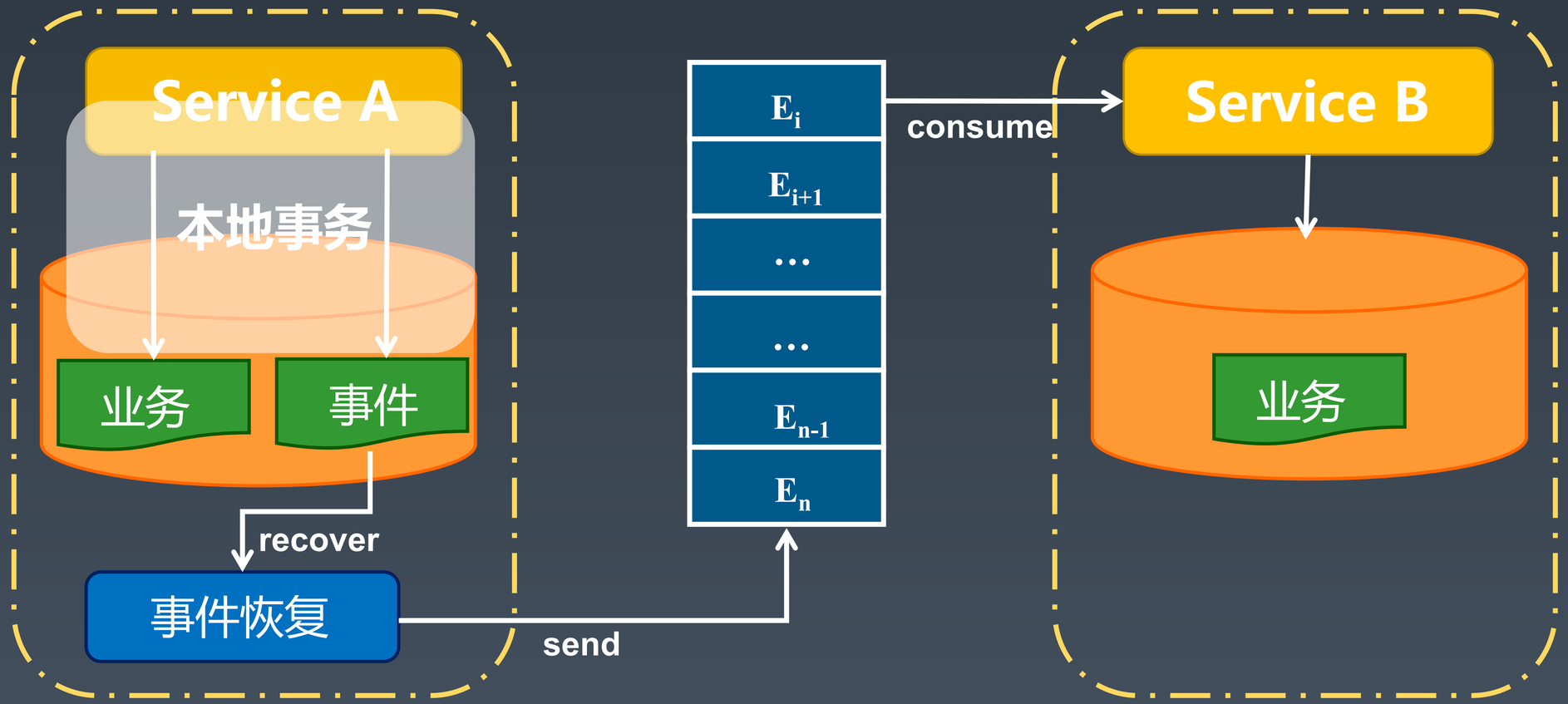
由全局事务管理器管理和协调的事务，可以跨 越多个资源（如数据库或JMS队列）和进程。

柔性事务-异步确保型

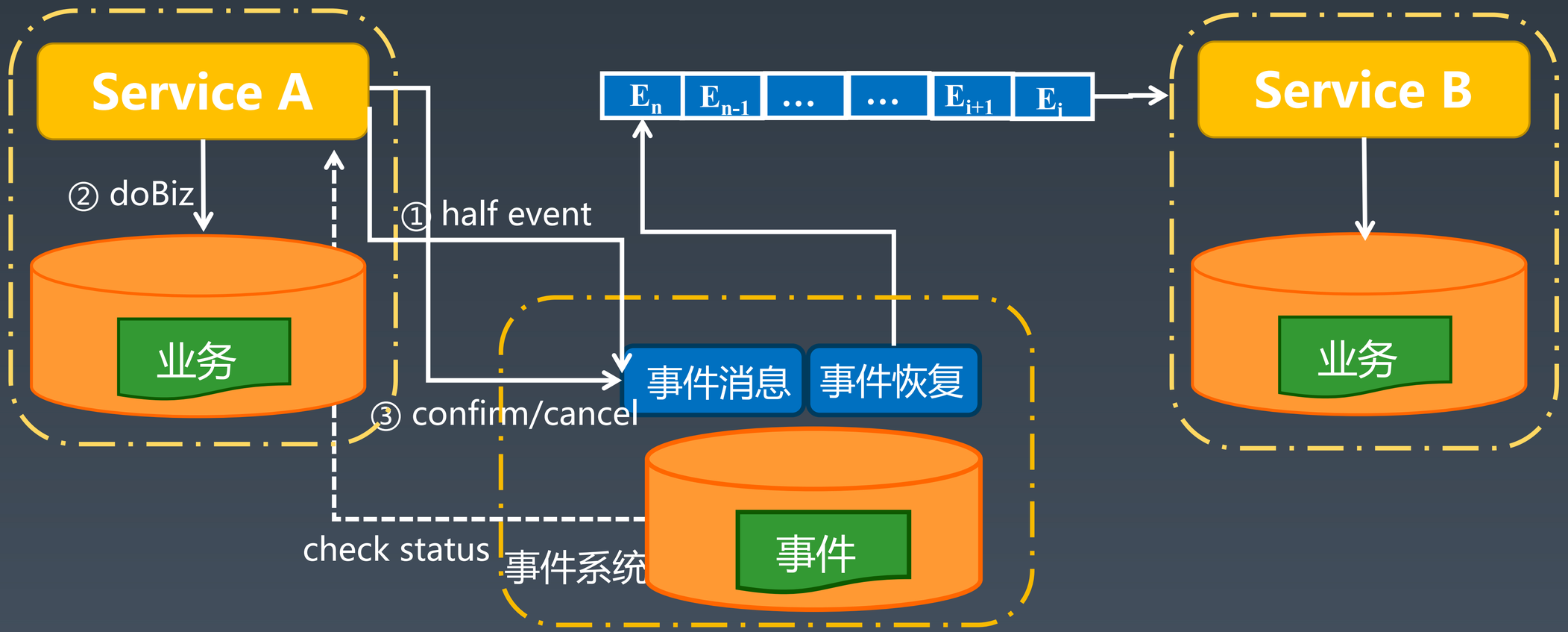
```
@Transactional(rollbackFor = Exception.class)
    public void msgAndDbTrans() {
        try {
            boolean result = dao.update(model);//①
            //②
            if (result) {
                mq.send(model);//③
            }
            //④
        } catch (Exception exx) {
            throw exx;
        }
    }
```

是否能保证
一致性?

异步确保型—本地事件表


Service A
本地事务
业务
事件
recover
事件恢复
send
E_i
E_{i+1}
…
…
E_{n-1}
E_n
consume
Service B
业务

异步确保型—外部事件表


Service A
② doBiz
业务
① half event
③ confirm/cancel
check status
E_n E_{n-1} … … E_{i+1} E_i
事件消息
事件恢复
事件系统
事件
Service B
业务

常用分布式事务-总结

- 刚性事务:

  标准分布式事务（2PC/3PC）

  缺点：性能差、资源占用时间长、需要实现XA接口

- 柔性事务：

  异步确保型

  缺点：侵入性高、依赖可靠事件服务、实时性差、消费失败状态不可逆

  纯补偿性

  缺点：侵入性高、开发成本高、一致性差

  TCC

  缺点：侵入性高、开发成本高

  最大努力通知

  缺点：侵入性高、开发成本高、实时性差

Seata 原理介绍

What is Seata

- Seata: Simple Extensible Autonomous Transaction Architecture

  微服务架构下，易用、高效的分布式事务解决方案

- 技术积累:

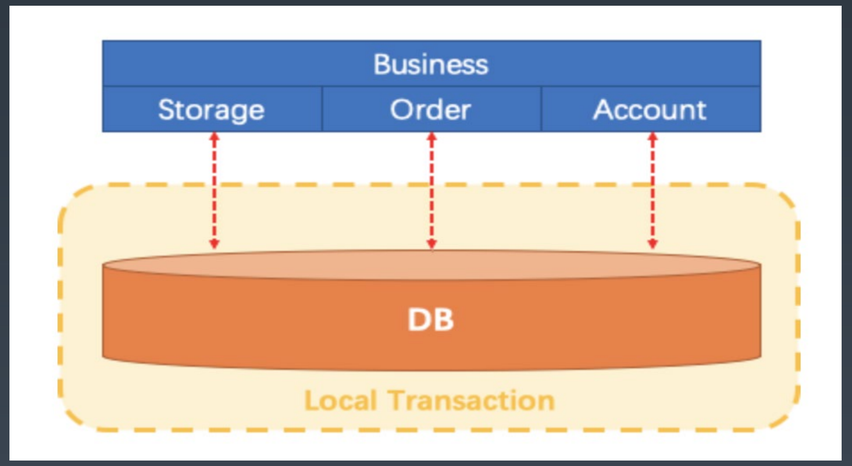
  TXC

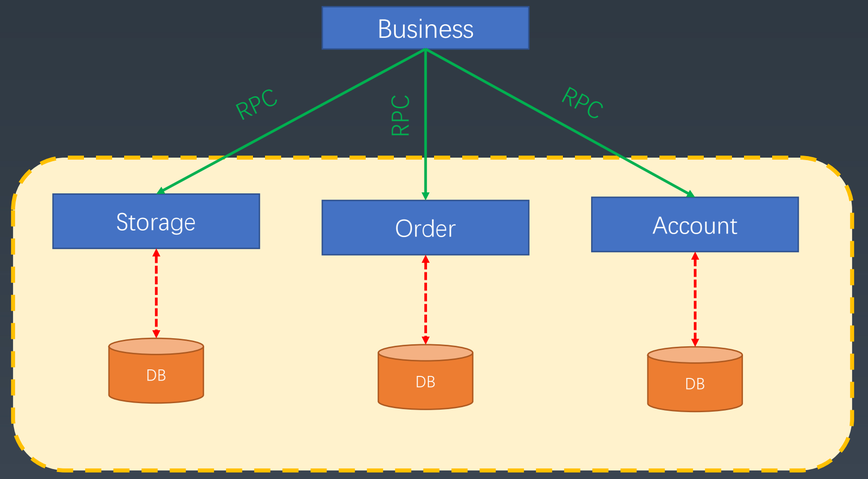
  GTS

  XTS

- 愿景: 像使用本地事务一样使用分布式事务

Problem VS Solution






```
@Transactional(timeout = 300000)
public void purchaseLocal(String userId,
                          String commodityCode,
                          int count) {
    storageLocalService.deduct(
        commodityCode, count);
    int needPay = orderLocalService.create(
        userId, commodityCode, count);
    accountLocalService.debit(userId,needPay);

}
```

```
@GlobalTransactional(timeout = 300000)
public void purchaseRemote(String userId,
                           String commodityCode,
                           int count) {
    storageRemoteService.deduct(
        commodityCode, count);
    int needPay = orderRemoteService.create(
        userId, commodityCode, count);
    accountRemoteService.debit(userId,needPay);
}
```

Solution

- RPC:

  Dubbo、Spring Cloud、Motan 和 自定义 RPC 框架

- 资源:

  MySQL、Oracle、PostgreSQL、H2 和 RDS系列 等数据库

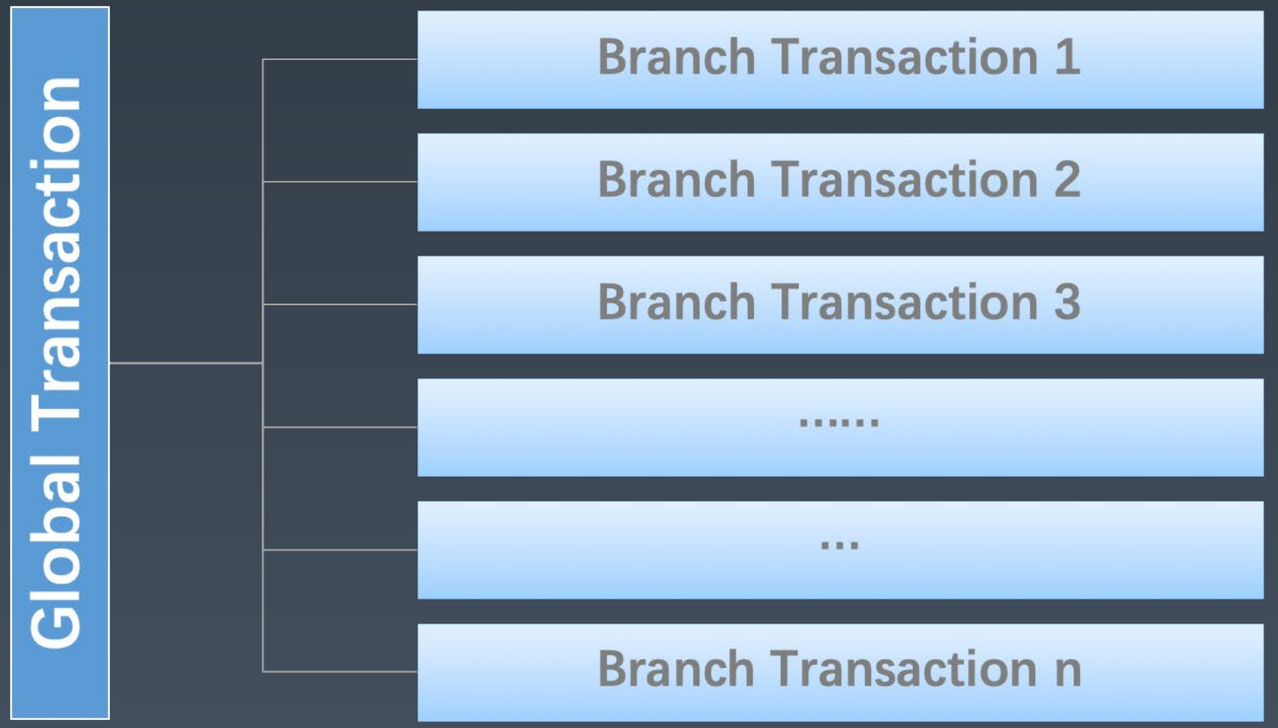
  MQ、NoSQL

  XA

  用户自定义类型资源

Seata 原理介绍

一个分布式（全局）事务是由若干本地（分支）事务组成。



Seata 原理介绍

◆Transaction Coordinator(TC)：

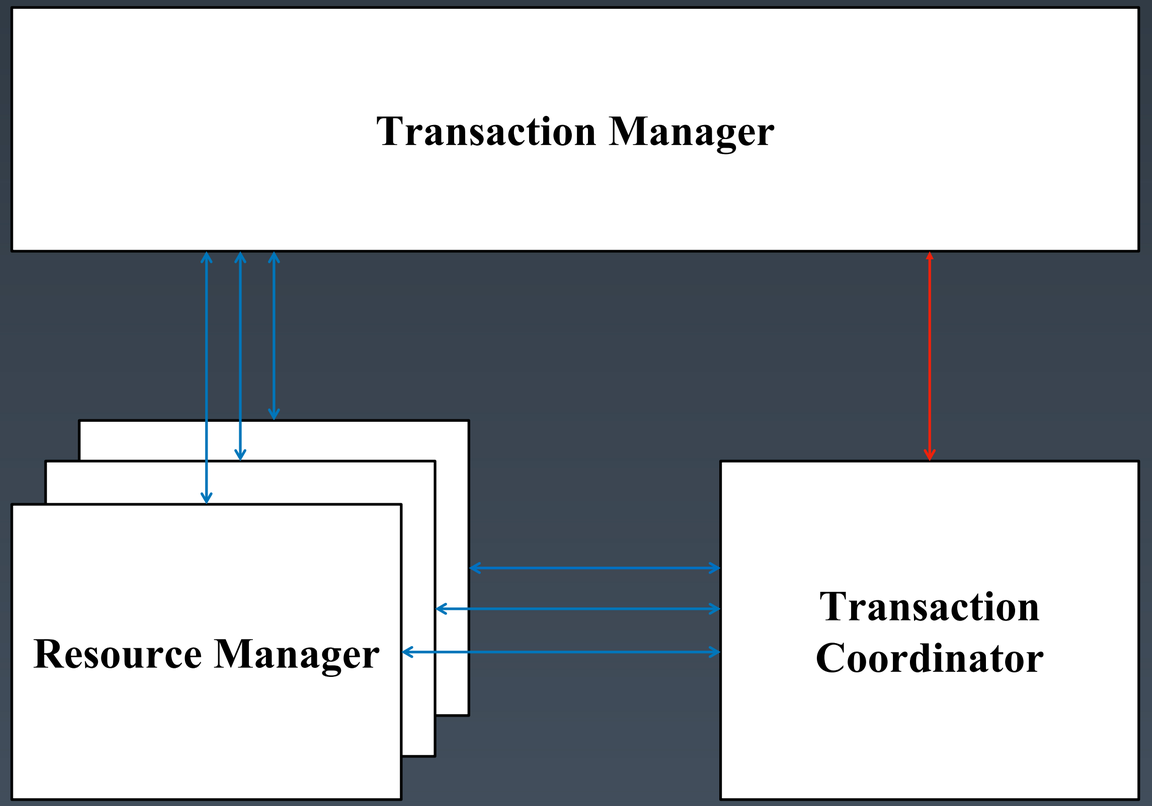
事务协调器，维护全局事务的运行状态，驱动全局事务的提交或回滚。

◆Transaction Manager(TM)：

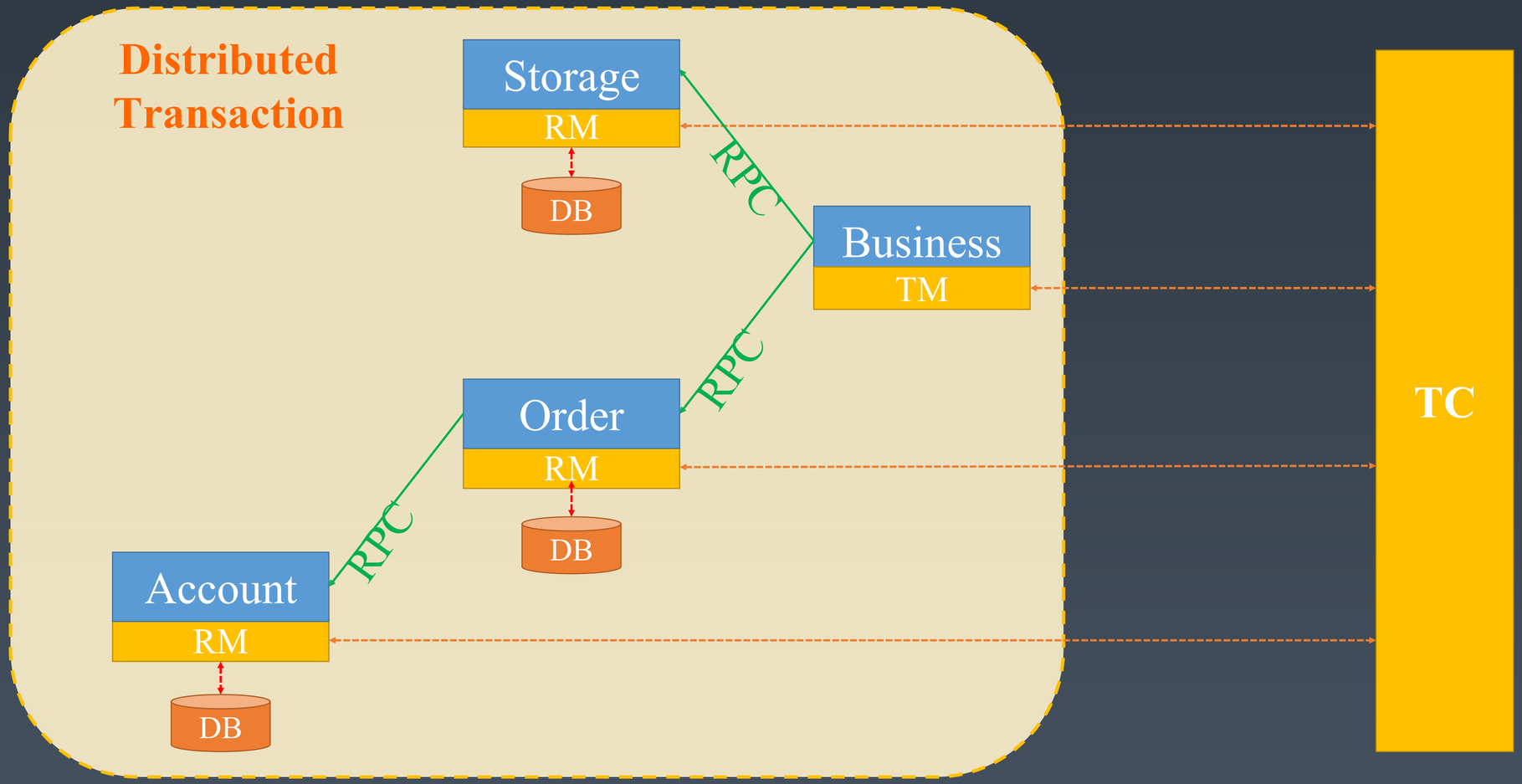
控制全局事务的边界，负责开启一个全局事务，并最终负责发起全局提交或全局回滚。

◆Resource Manager(RM)：

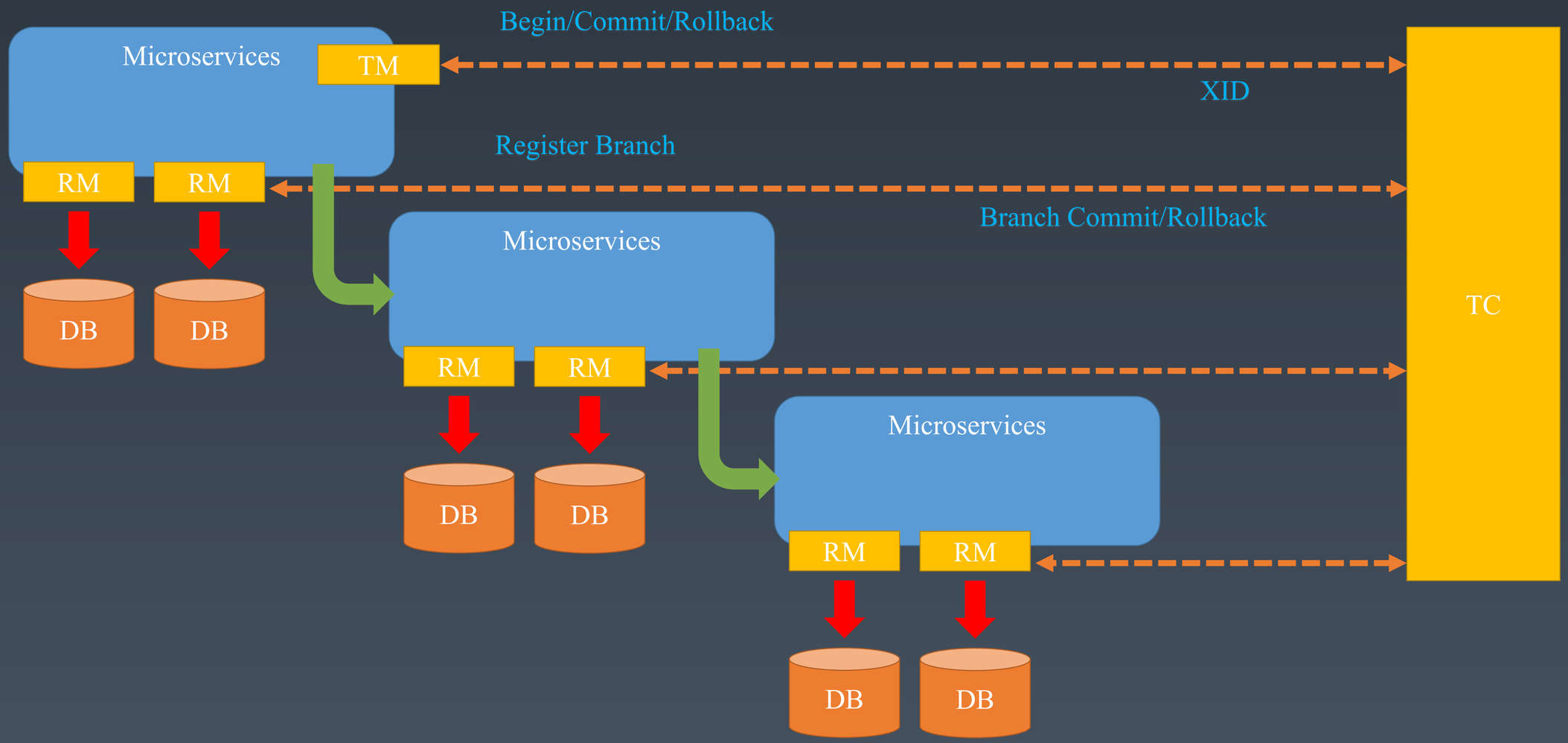
控制分支事务，负责分支事务的注册、状态汇报，并驱动分支（本地）事务的提交和回滚。



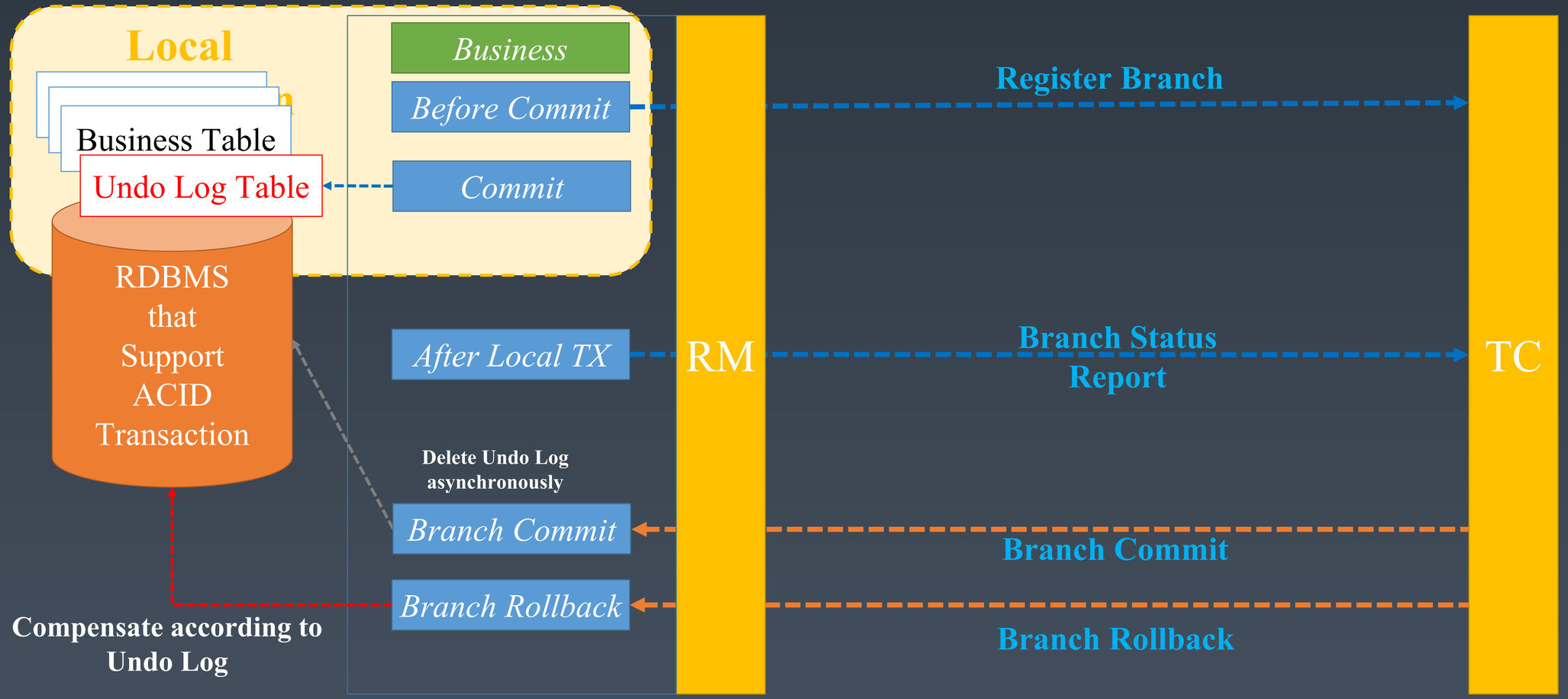
Seata 原理介绍


Distributed
Transaction
Storage
RM
DB
Business
TM
Order
RM
DB
Account
RM
DB
RPC
RPC
RPC
TC

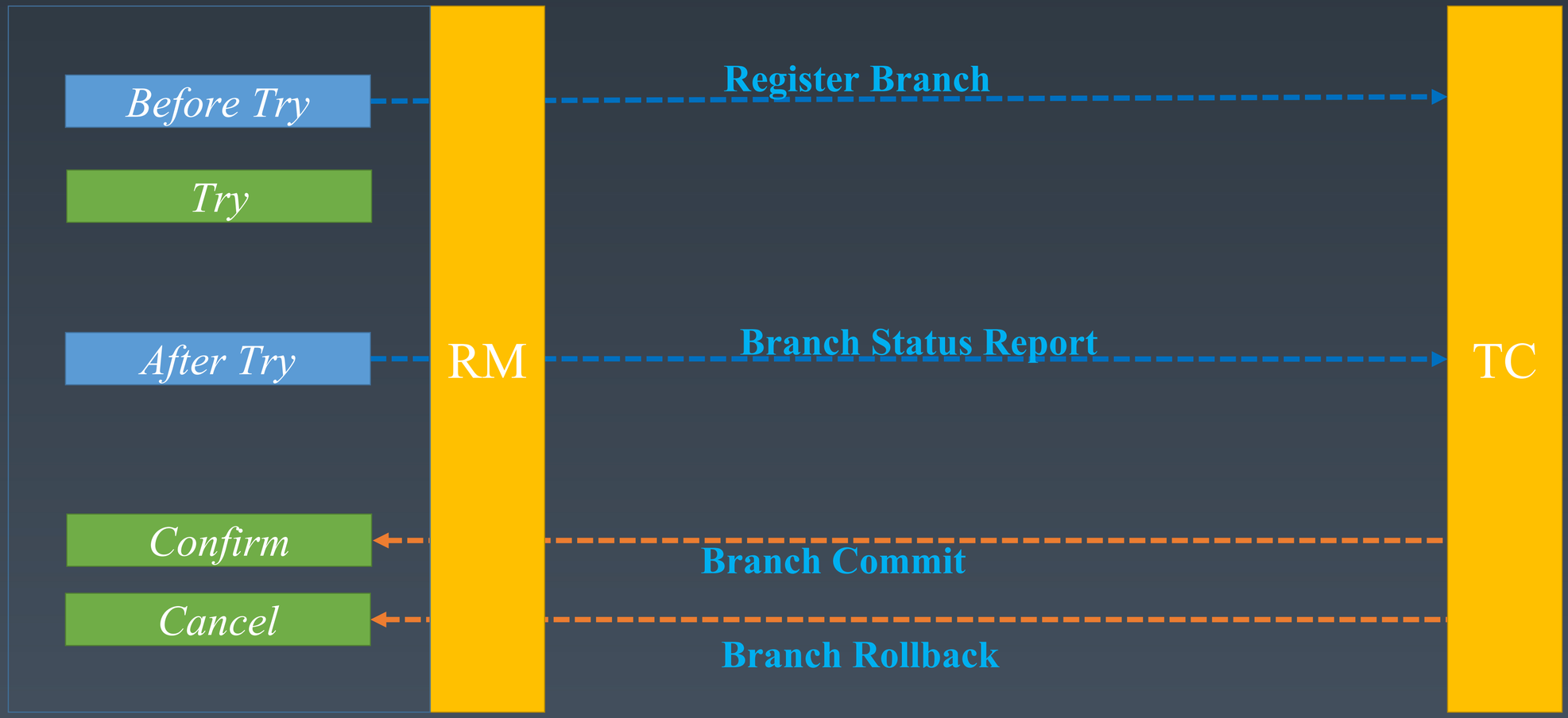
Seata 原理介绍


Begin/Commit/Rollback
Microservices
TM
XID
Register Branch
RM
RM
Branch Commit/Rollback
TC
DB
DB
Microservices
RM
RM
DB
DB
Microservices
RM
RM
DB
DB

Seata-AT


Local
Business Table
Undo Log Table
RDBMS that Support ACID Transaction
Business
Before Commit
Commit
After Local TX
Delete Undo Log asynchronously
Branch Commit
Branch Rollback
Compensate according to Undo Log
RM
Register Branch
Branch Status Report
Branch Commit
Branch Rollback
TC

Seata-TCC


Before Try
Try
After Try
Confirm
Cancel
RM
Register Branch
Branch Status Report
Branch Commit
Branch Rollback
TC

Seata 发展历程与Roadmap

阿里巴巴分布式事务

2014.11

TXC v1.0 （Taobao Transaction Constructor）

阿里系业务分布式事务解决方案，主要应用于集团电商、新零售和物流等微服务化场景

2015.06

TXC 专有云输出

树立了大量的企业、政府、金融等行业的标杆用户

2015.12

TXC v2.0

支持MT（Manual Transaction）模式，支持用户自定义补偿

2017.02

GTS 公测 (Global Transaction Service)

阿里云服务对外公测

2018.05

GTS 商业化

阿里云商用版本正式对外提供商用服务

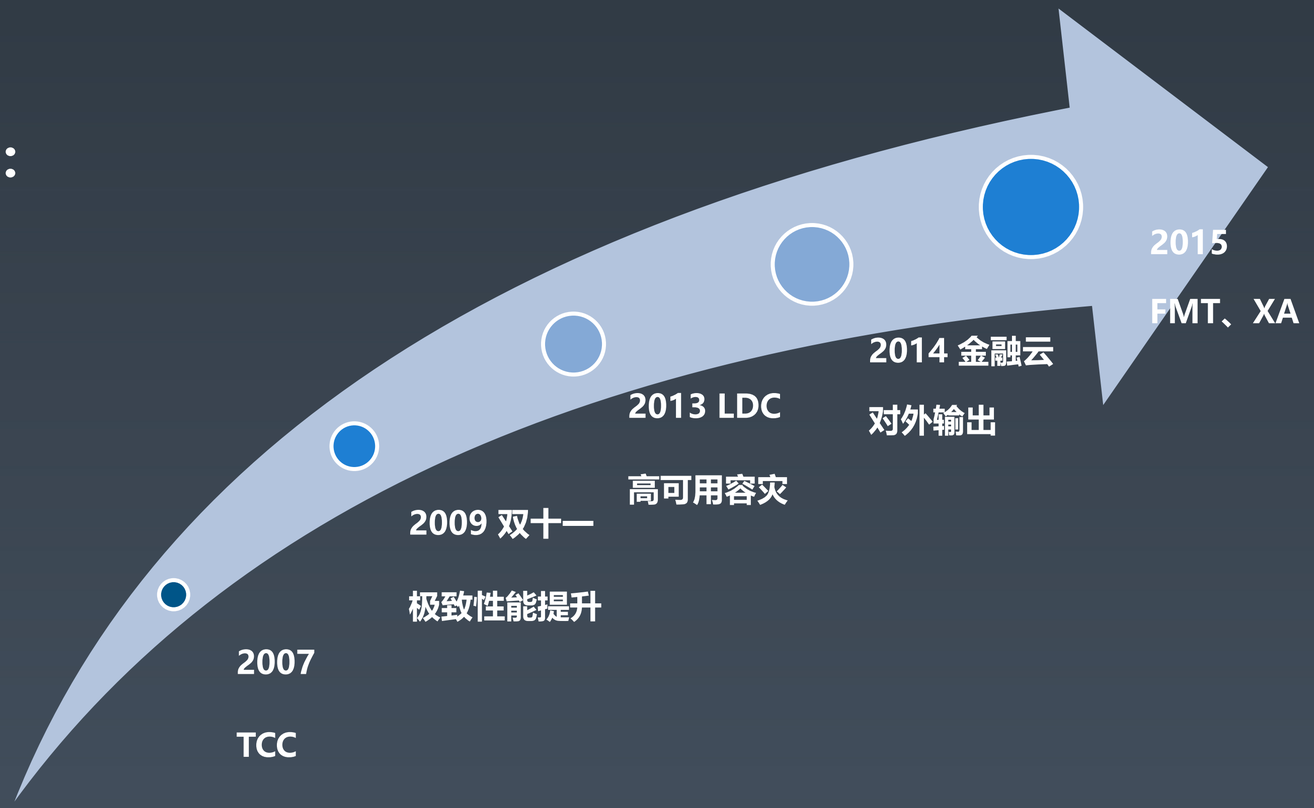
2019.01

Alibaba Fescar

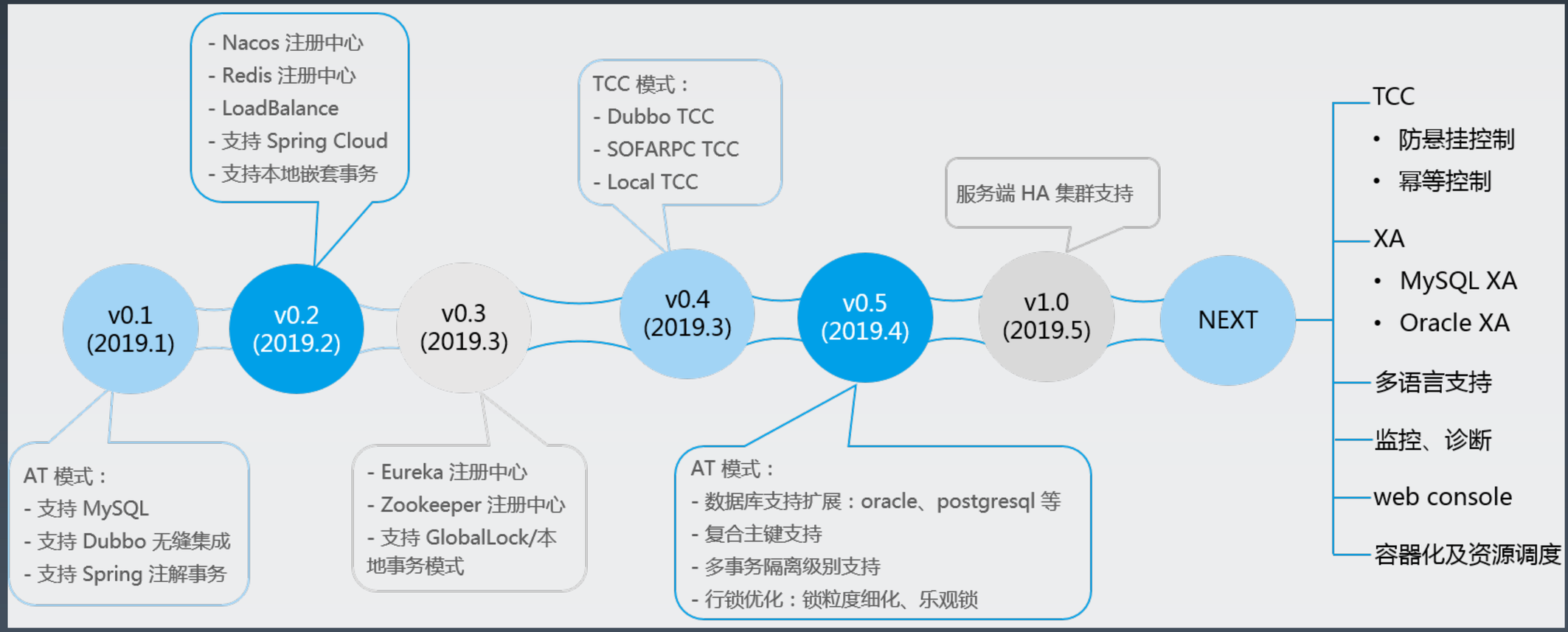
开源版本正式发布

蚂蚁金服分布式事务

- ❖三种模式，丰富的应用场景：
  - ➢TCC模式
  - ➢FMT模式
  - ➢XA模式
- ❖极致性能提升
- ❖高可用容灾
- ❖无侵入解决方案

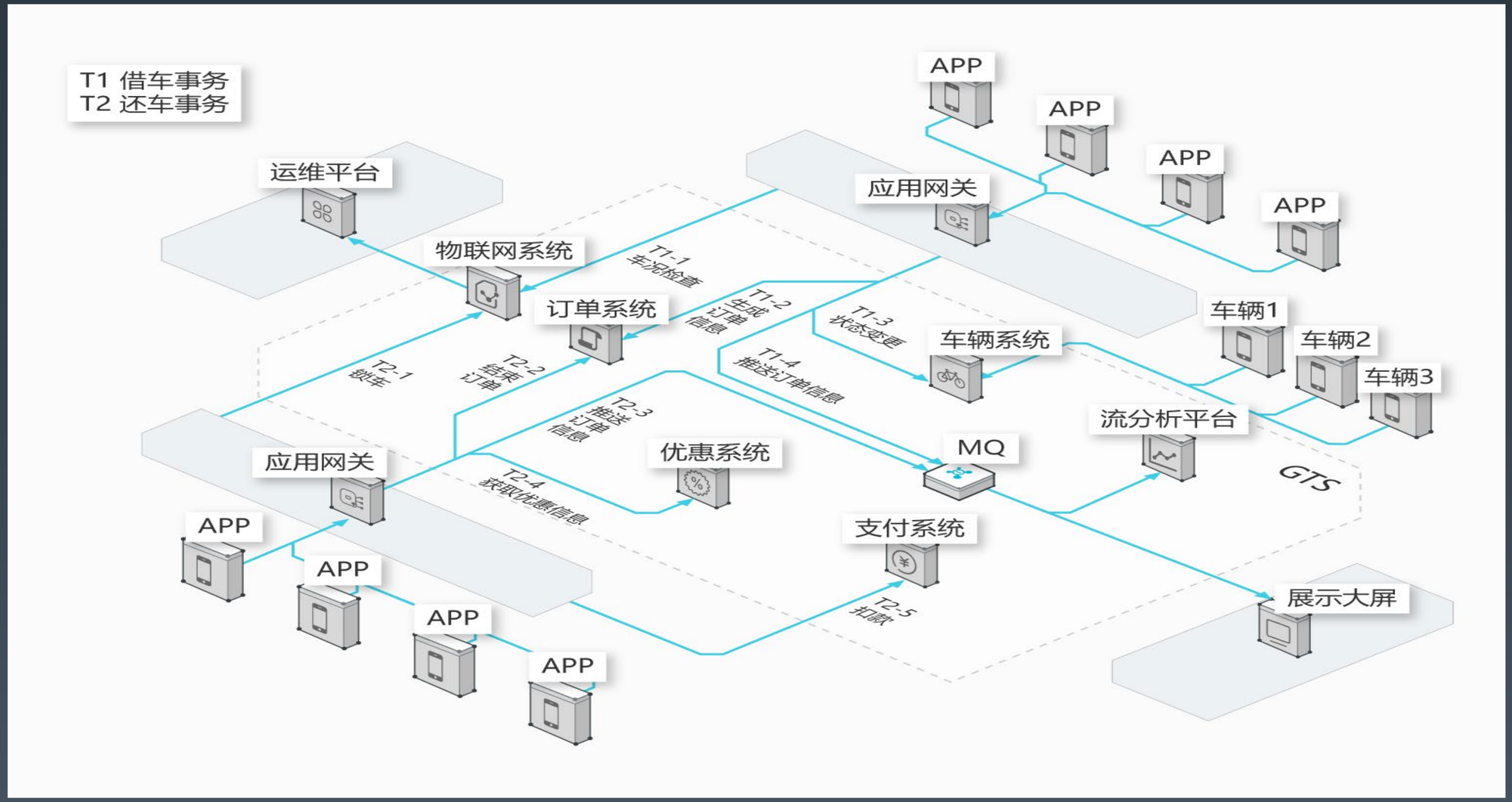


阿里巴巴 + 蚂蚁金服 社区共建

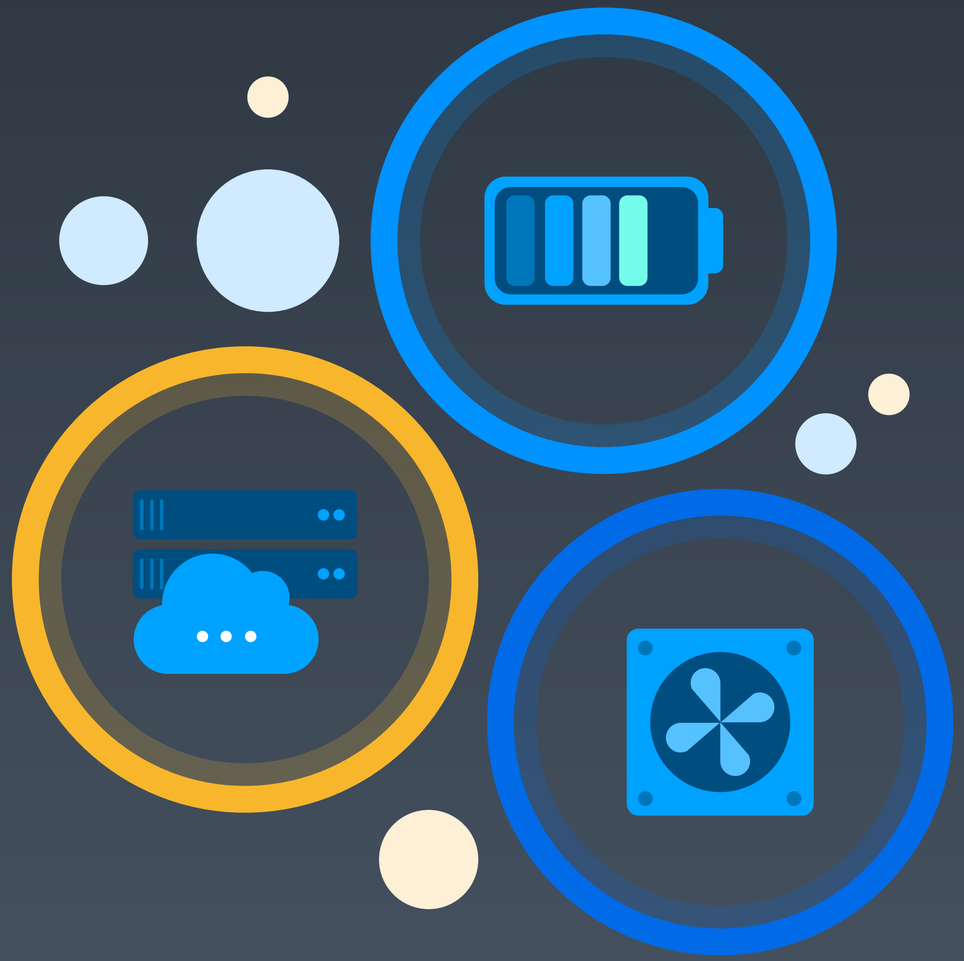


用户案例与推荐阅读

用户案例-GTS


T1 借车事务
T2 还车事务
APP
APP
APP
APP
运维平台
应用网关
物联网系统
T1-1
车况检查
订单系统
T1-2
生成
订单
信息
T1-3
状态变更
车辆系统
车辆1
车辆2
车辆3
T1-4
推送订单信息
T2-1
锁车
T2-2
结束
订单
T2-3
推送
订单
信息
流分析平台
MQ
优惠系统
应用网关
T2-4
获取优惠信息
GTS
APP
APP
APP
APP
支付系统
T2-5
扣款
展示大屏

实例演示



环境要求

JDK 1.8+

MySQL 5.6+

Nacos 0.8+

演示内容

模拟创建订单、库存和账

Dubbo微服务间的一致性

https://github.com/seata/seata-workshop

推荐阅读

1.《 The XA Specification》
2.《 Designing data-intensive applications》-- Martin Kleppmann
3.《数据库事务处理的艺术:事务管理与并发控制数据库技术 》--李海翔等
4.《数据库中间件详解 | 珍藏版》
5.《微服务架构下，解决数据一致性问题的实践》
6.《集成源码深度剖析：Fescar x Spring Cloud》
7.《分布式事务中间件Fescar-全局写排它锁解读》
8.《开发者说：深度剖析开源分布式事务方案 Seata 的事务协调器》
9.《源码｜详解分布式事务之 Seata-Client 原理及流程》
10.《详细解读 Github 上发布仅一个月就获得4k+ star 的分布式事务解决方案》

THANKS!

QCon 10th