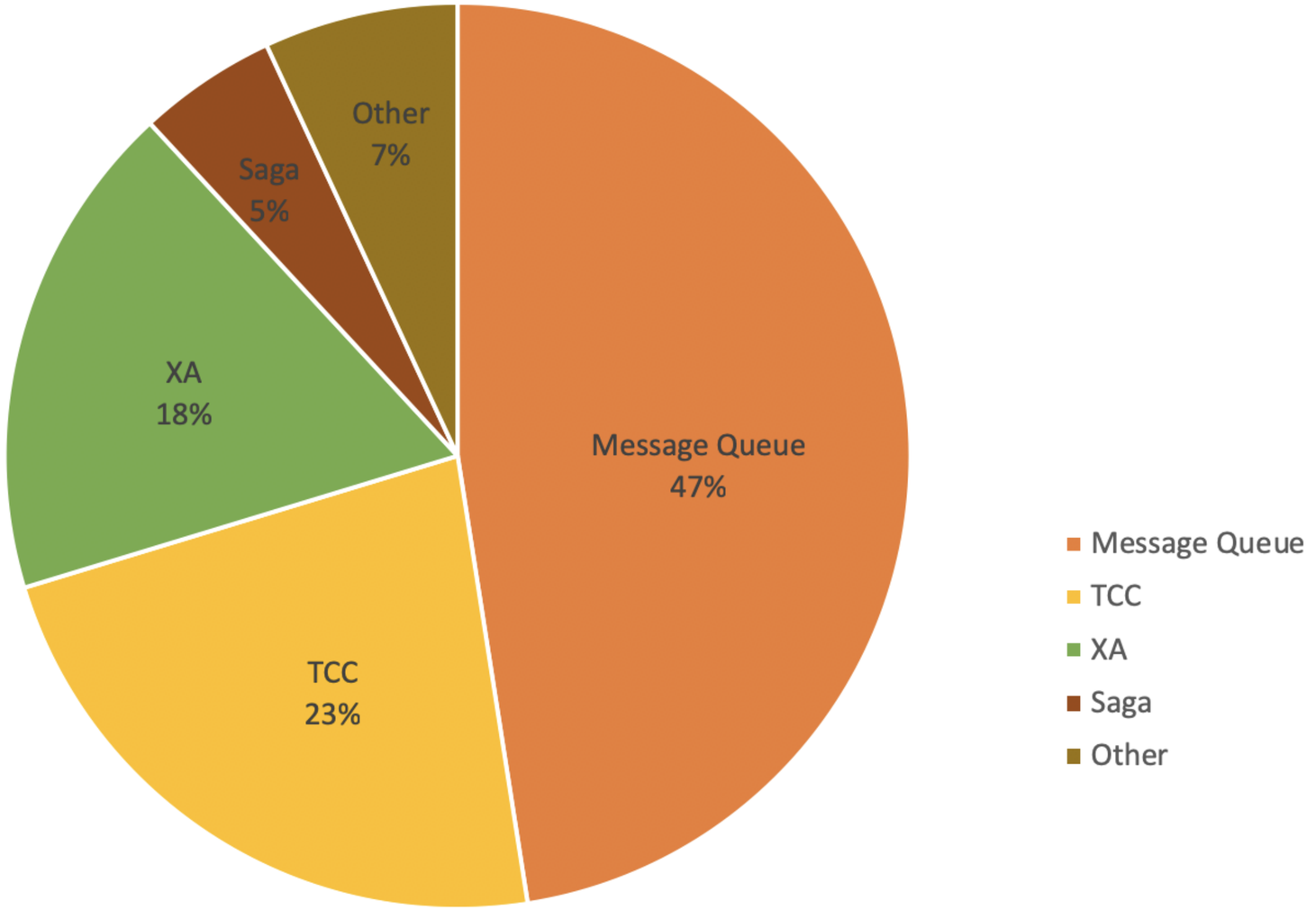
分布式事务哪家强？Seata来帮忙

季敏@slievrly　　Seata 开源项目发起人

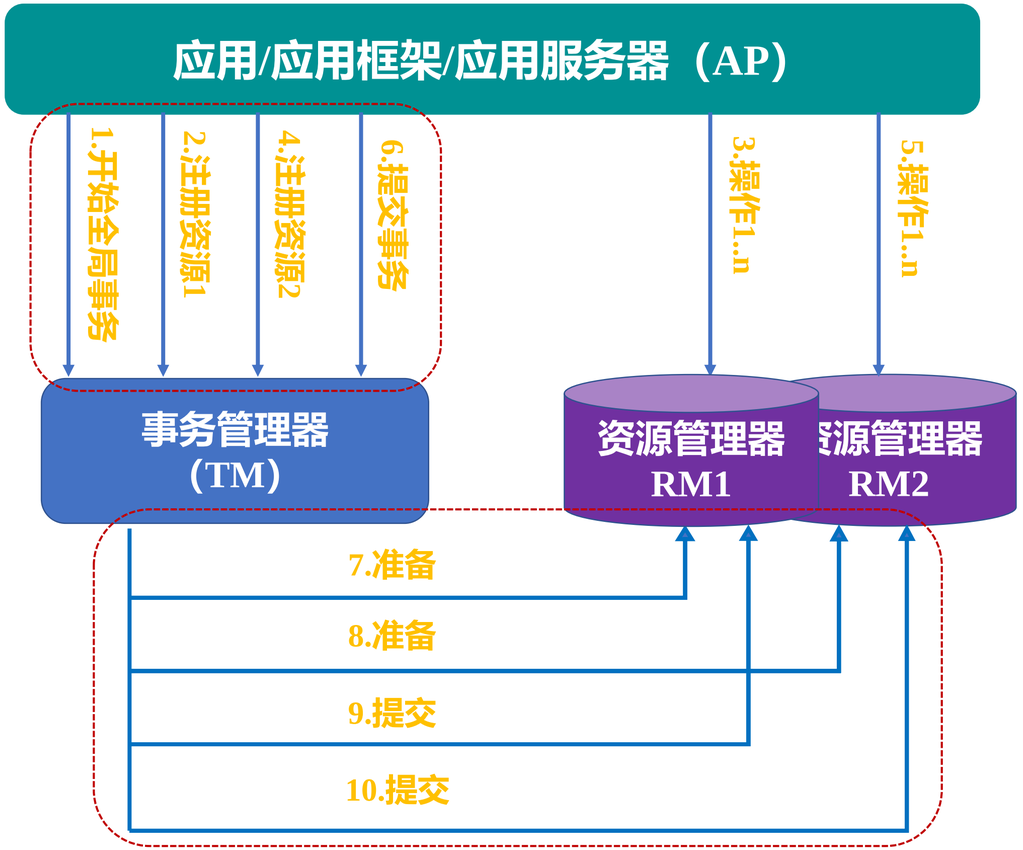
Agenda

- 常见分布式一致性解决方案介绍
- Seata 原理介绍
- What’ s New in Seata 0.7.1
- Seata 与 Dubbo 的集成

分布式事务市场现状


Other
7%
Saga
5%
XA
18%
Message Queue
47%
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。

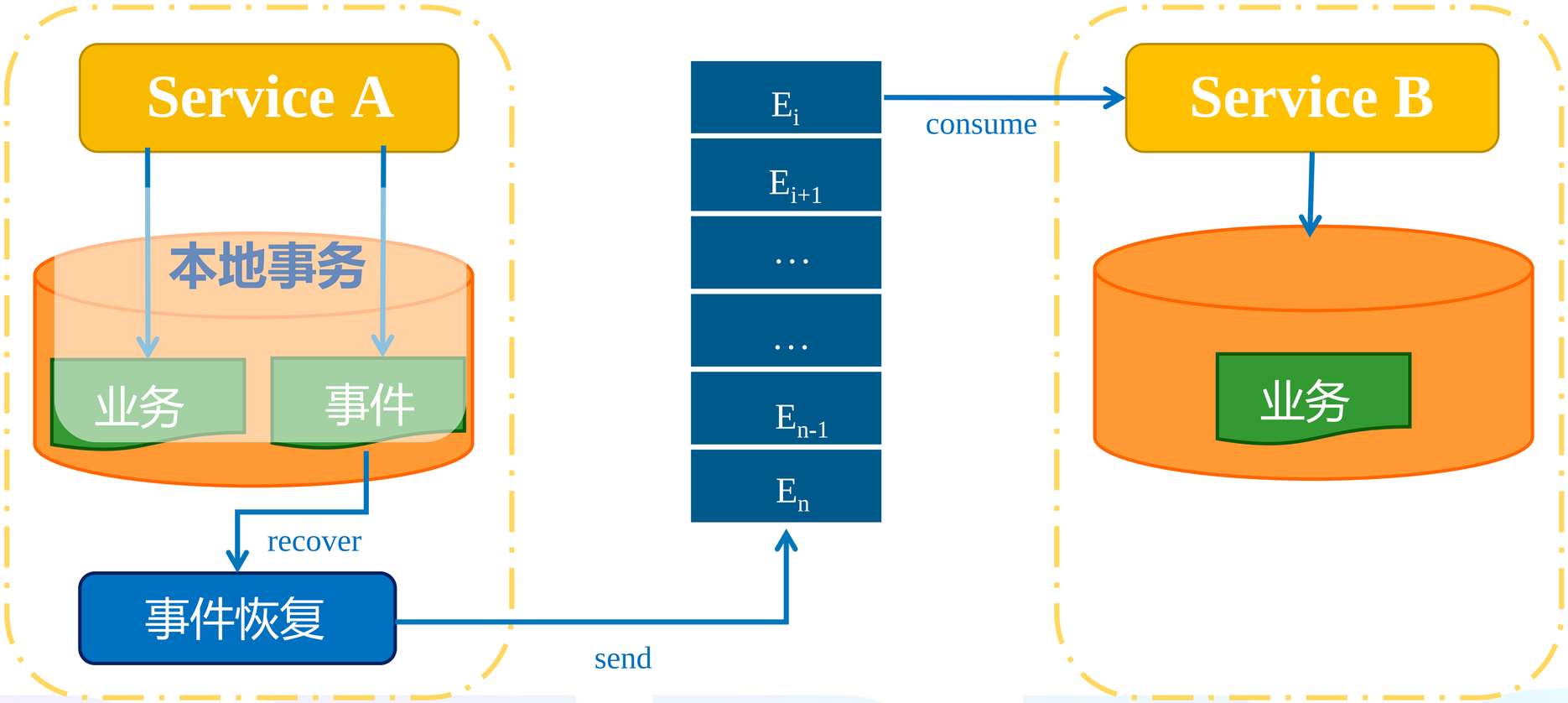
XA之所以需要引入事务管理器是因为，在分布 式系统中，从理论上讲两台机器理论上无法达 到一致的状态，需要引入一个单点进行协调。

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。

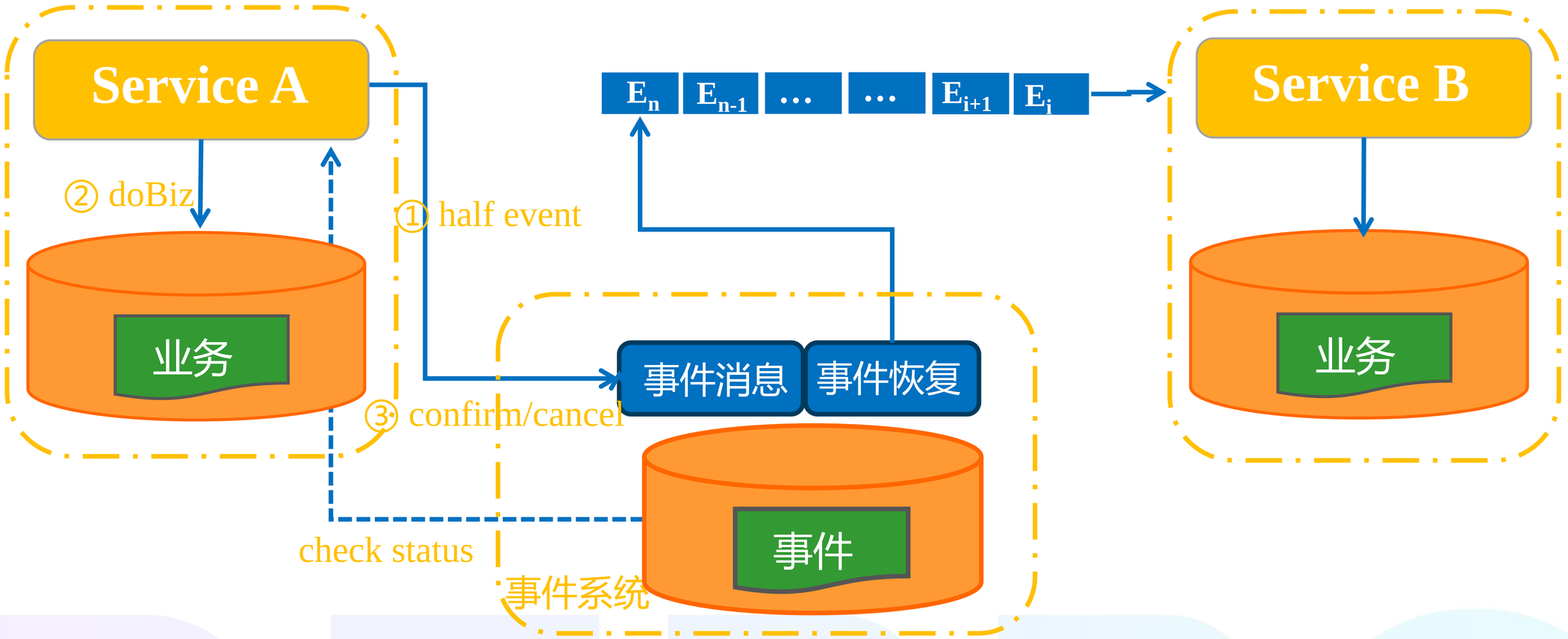
柔性事务-异步确保型

```
@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;
        }
    }
```

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
事件系统
Service B
业务

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

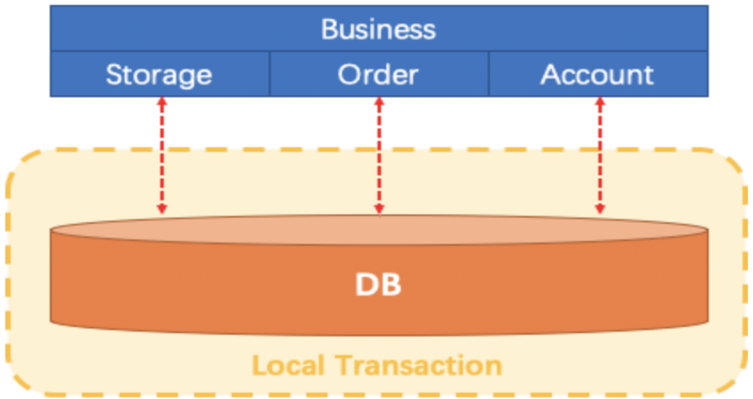
  TCC

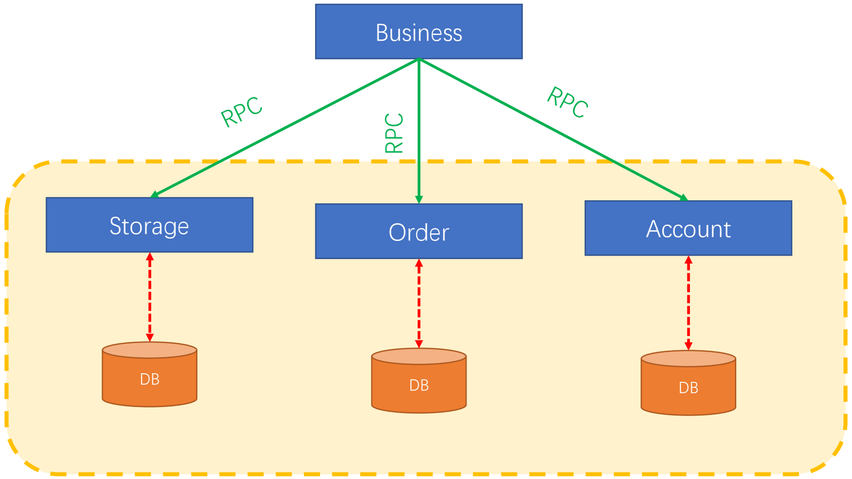
  缺点：侵入性高、开发成本高

  最大努力通知

  缺点：侵入性高、开发成本高、实时性差

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

Seata 原理

◆**Transaction Coordinator(TC)：**

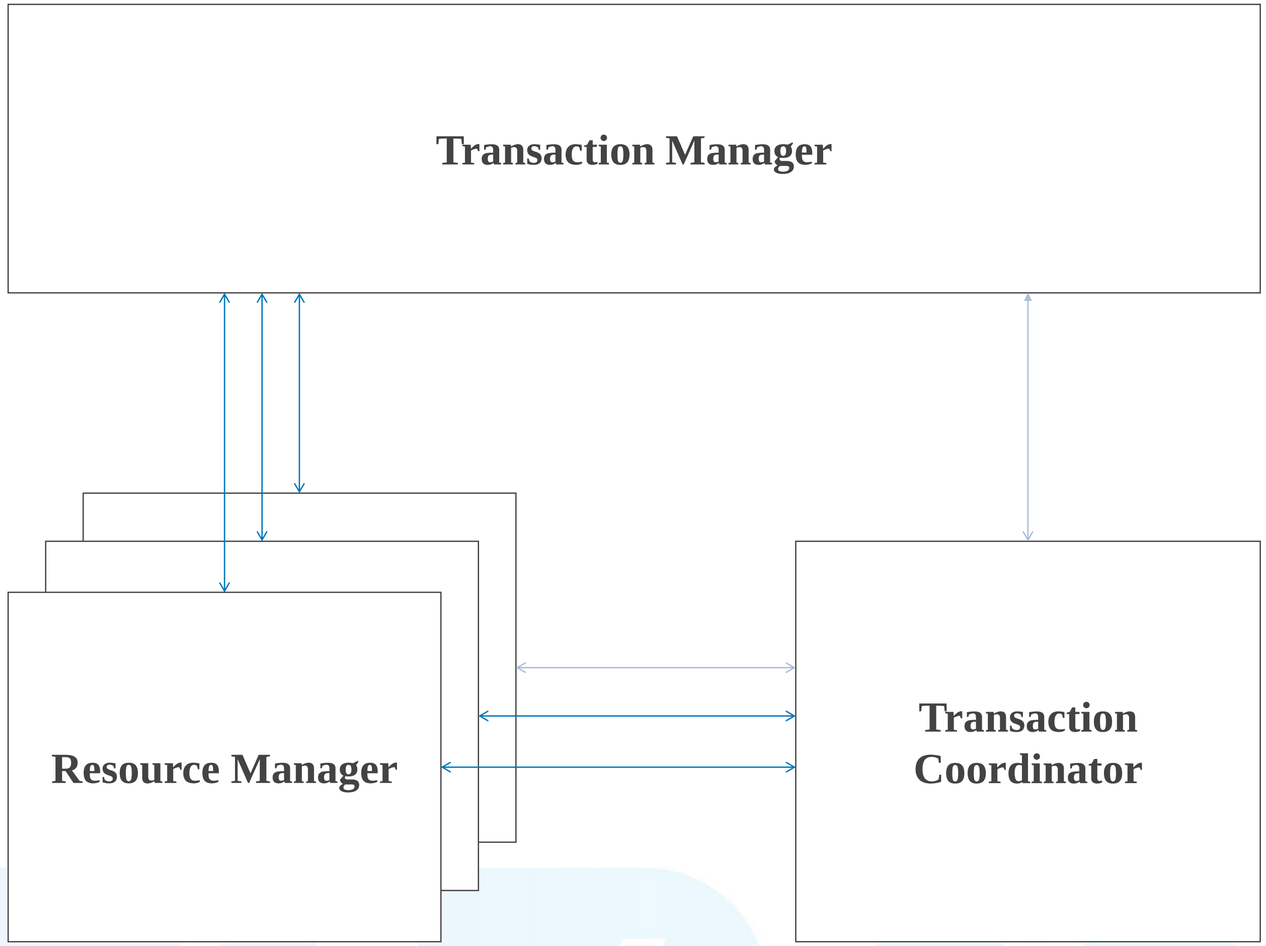
事务协调器，维护全局事务的运行状态，驱动全局事务的提交或回滚。

◆**Transaction Manager(TM)：**

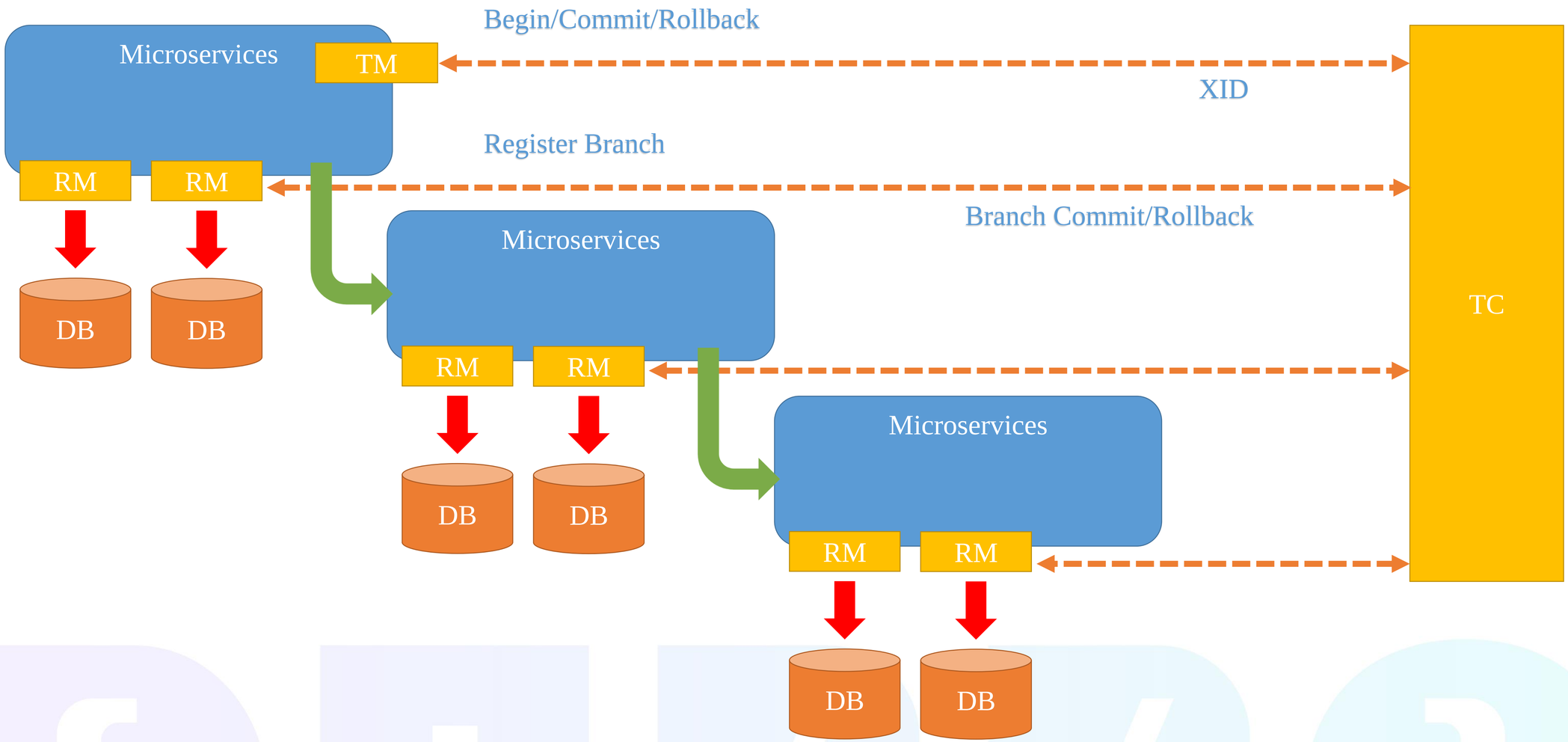
控制全局事务的边界，负责开启一个全局事务，并最终负责发起全局提交或全局回滚。

◆**Resource Manager(RM)：**

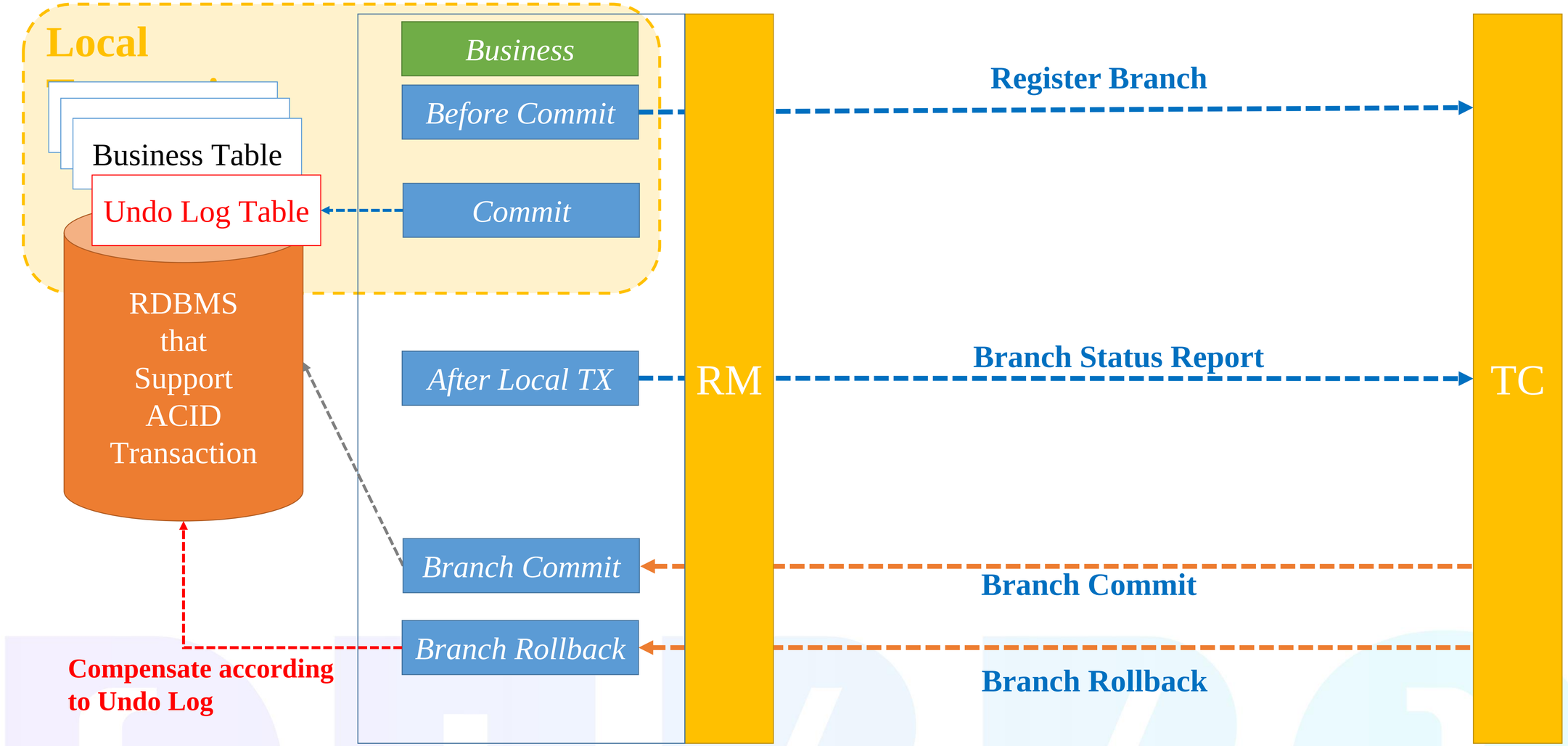
控制分支事务，负责分支事务的注册、状态汇报，并驱动分支（本地）事务的提交和回滚。



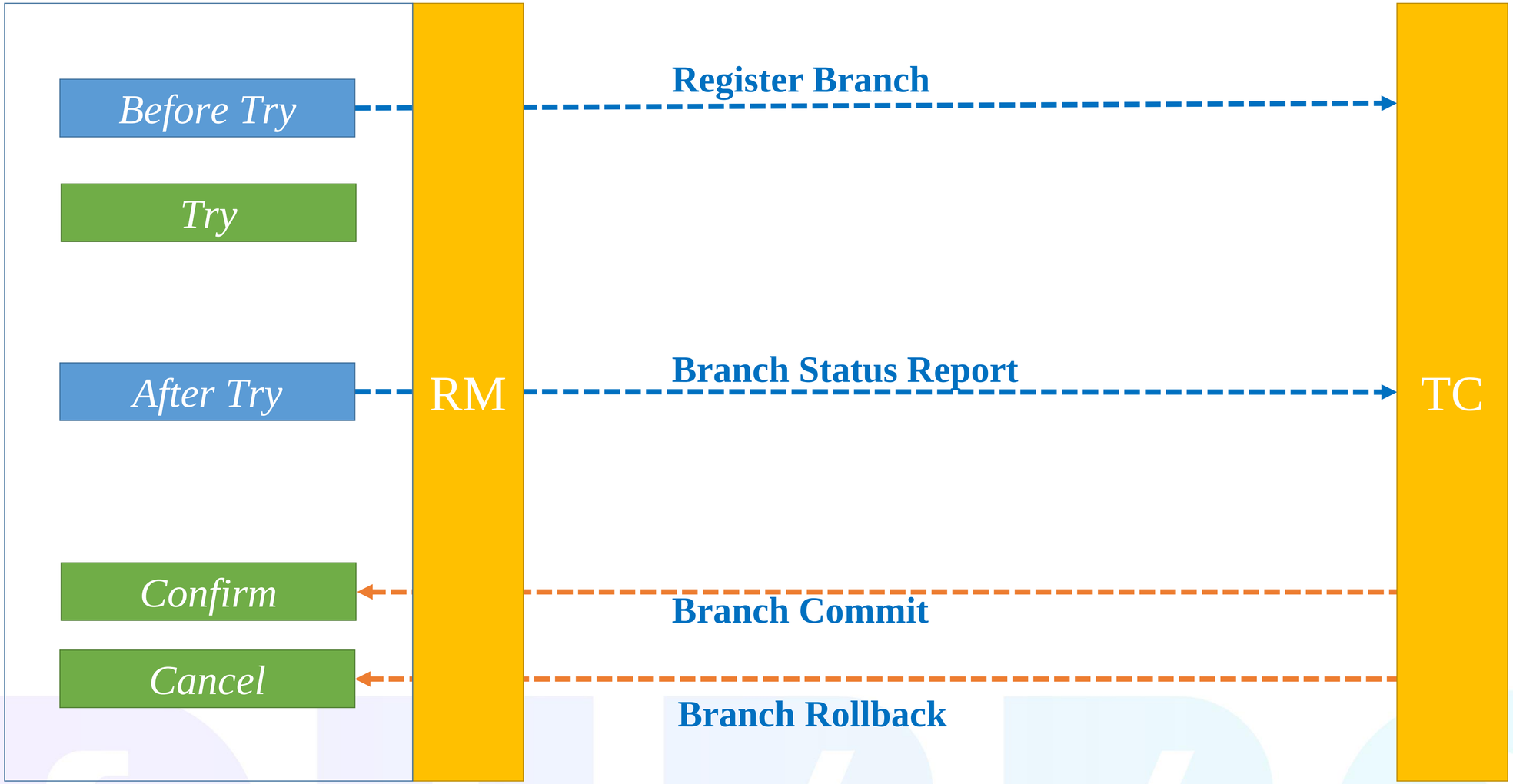
Seata 原理


Begin/Commit/Rollback
Microservices
TM
XID
Register Branch
RM
RM
Branch Commit/Rollback
DB
DB
Microservices
TC
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
Branch Commit
Branch Rollback
RM
Register Branch
Branch Status Report
Branch Commit
Branch Rollback
TC
Compensate according to Undo Log

Seata TCC


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

What’s New in Seata 0.7.1

- Protobuf 支持
- Mertric 支持
- 协议重设计

Seata 协议

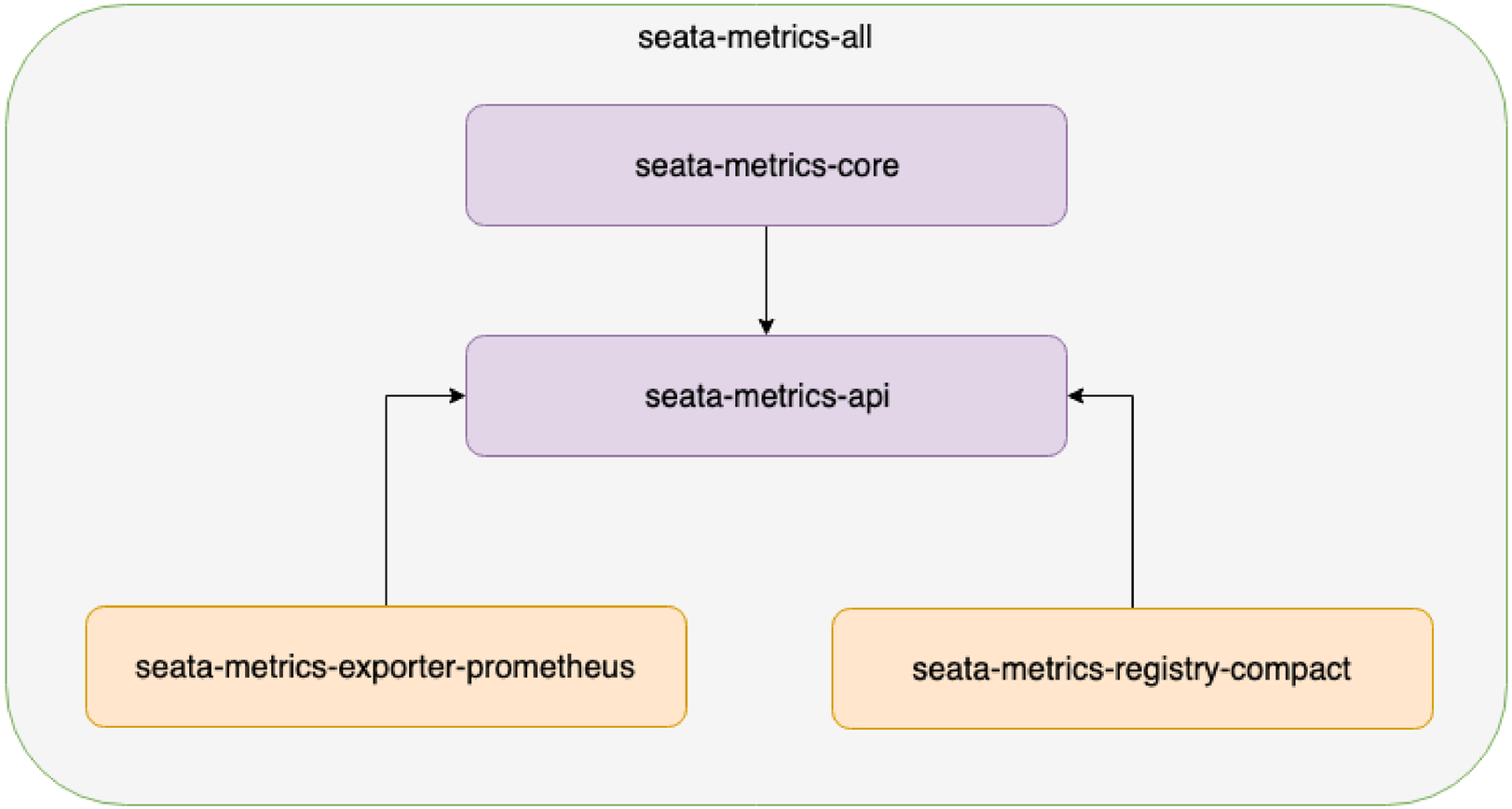
0.1.0 ~ 0.6.1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Magic		Flag		Type Code		Request ID								Body…	
Body															
……															
……															

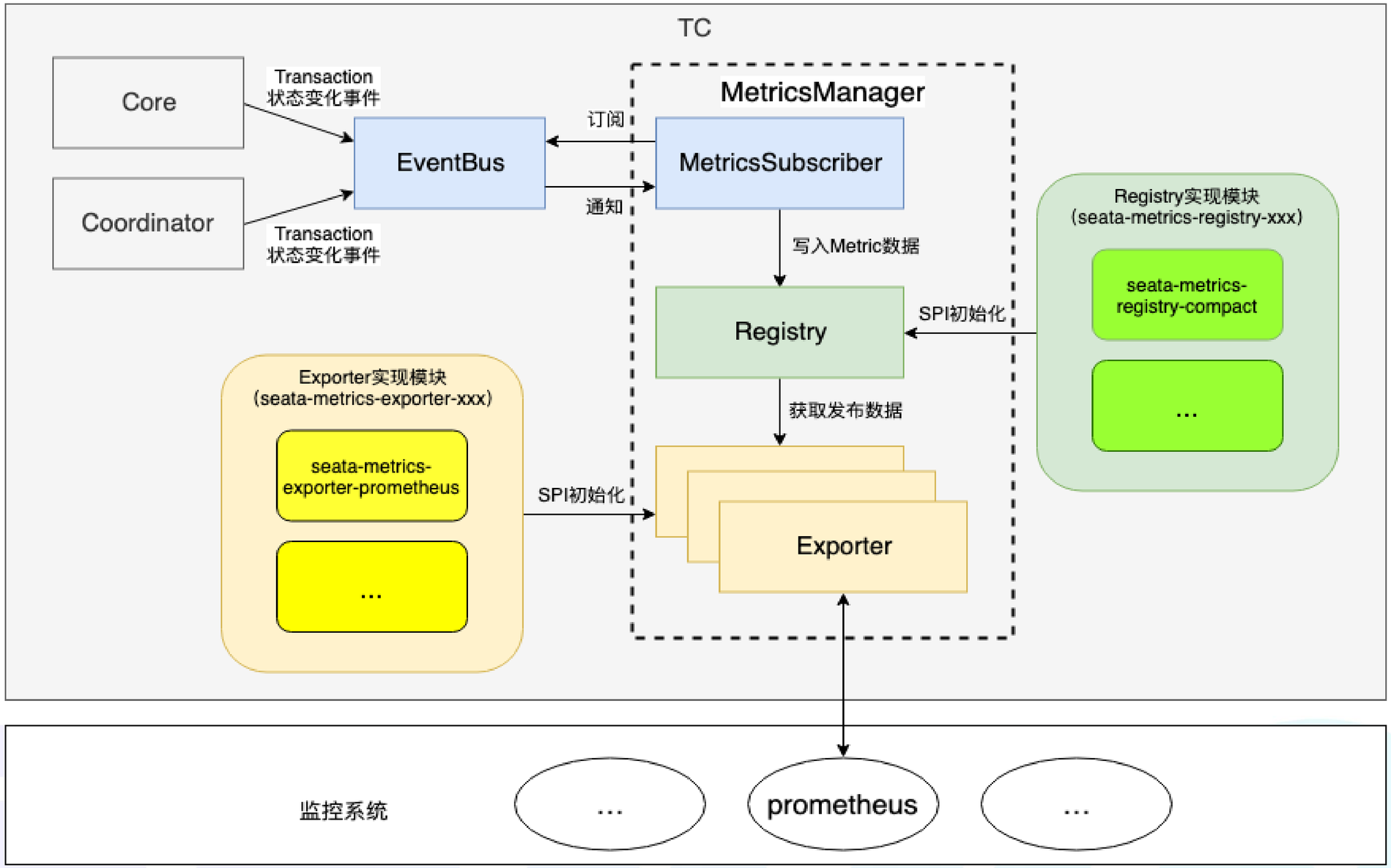
0.7.1 +

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Magic		version	Full length				Header length		Msg Type	Serializer	Compress	Request ID			
Header Map [Optional]															
Body															
……															
……															

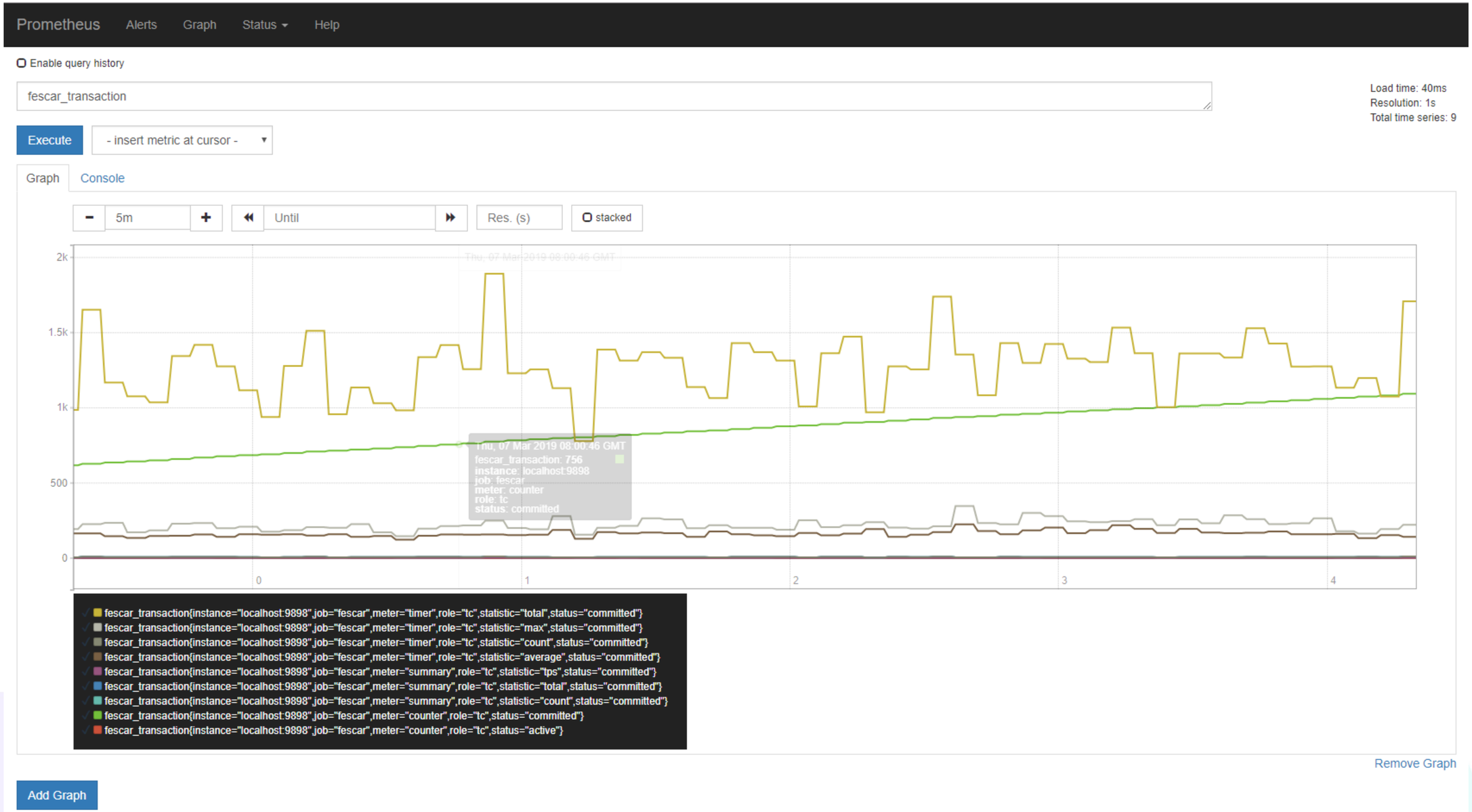
Seata Mertric



Seata Mertric



Seata Mertric



Seata Mertric

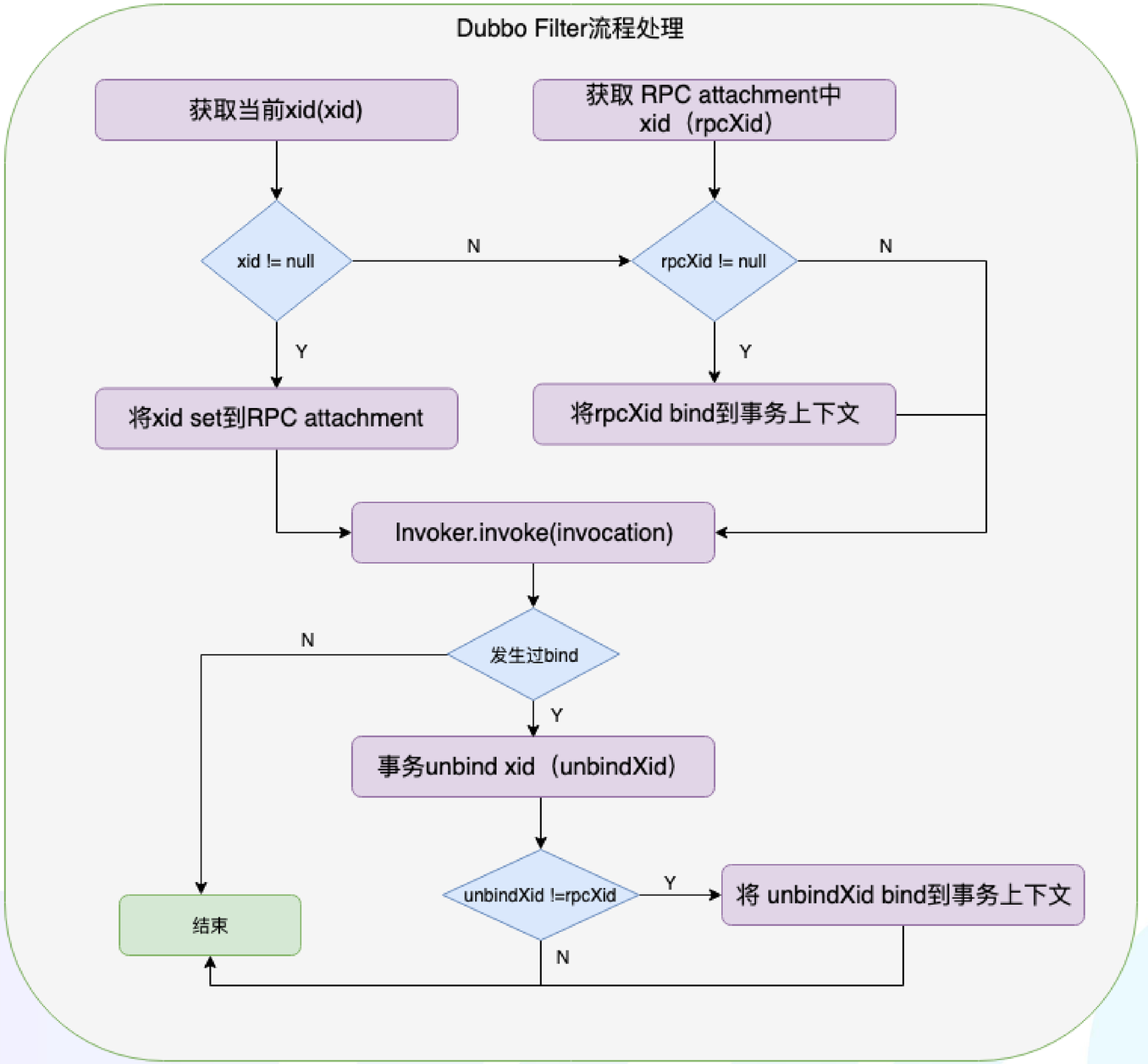
Seata 与 Dubbo 的集成

Dubbo集成

与RPC框架集成需要实现的关键点：

- 事务上下文传递与清理
- 绑定和解绑事务
- 服务调用链路的异常处理

Dubbo集成


Dubbo Filter流程处理
获取当前xid(xid)
获取 RPC attachment中
xid（rpcXid）
xid != null
N
rpcXid != null
N
Y
Y
将xid set到RPC attachment
将rpcXid bind到事务上下文
Invoker.invoke(invocation)
N
发生过bind
Y
事务unbind xid（unbindXid）
unbindXid !=rpcXid
Y
将 unbindXid bind到事务上下文
结束
N

Join Us

欢迎钉钉扫描二维码进群

欢迎关注阿里巴巴中间件微信公众号

GitHub：https://github.com/seata/seata

Thank you !