解密ShardingSphere与ServiceComb联合打造的分布式事务解决方案

京东数科-潘娟
panjuan@apache.org

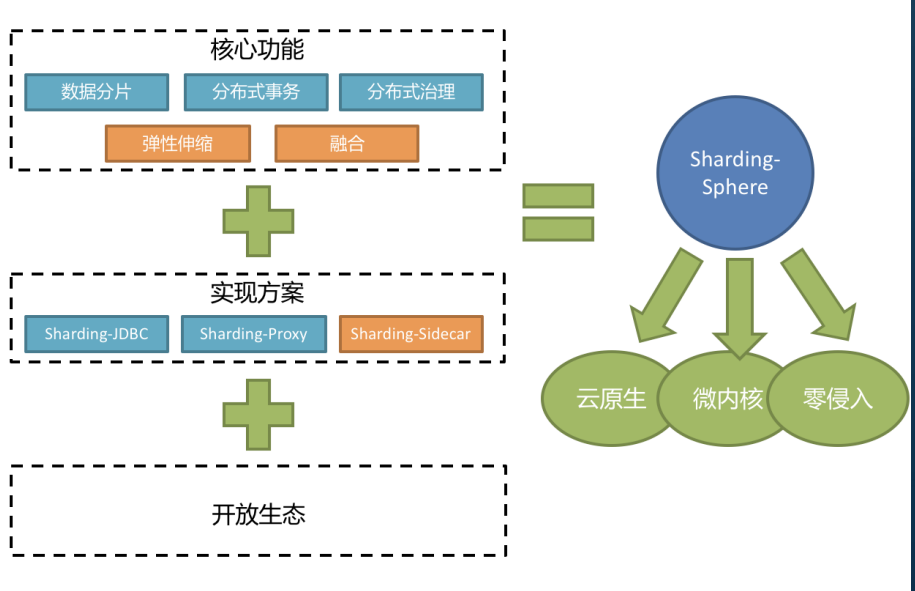
servicecomb.apache.org
github.com/apache?q=servicecomb
www.huaweicloud.com

Grow with Intelligence

目录

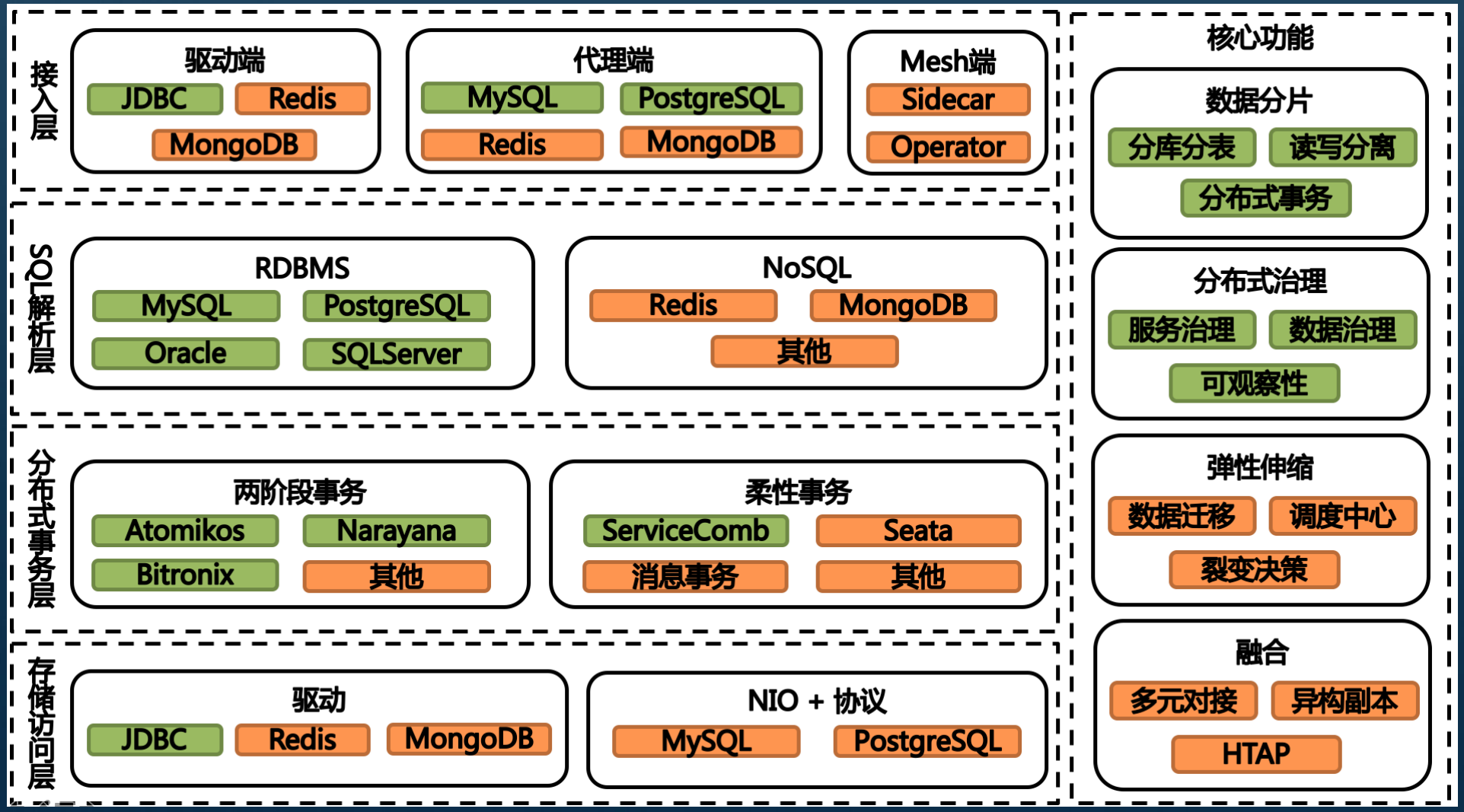
CONTENTS

Apache ShardingSphere简介

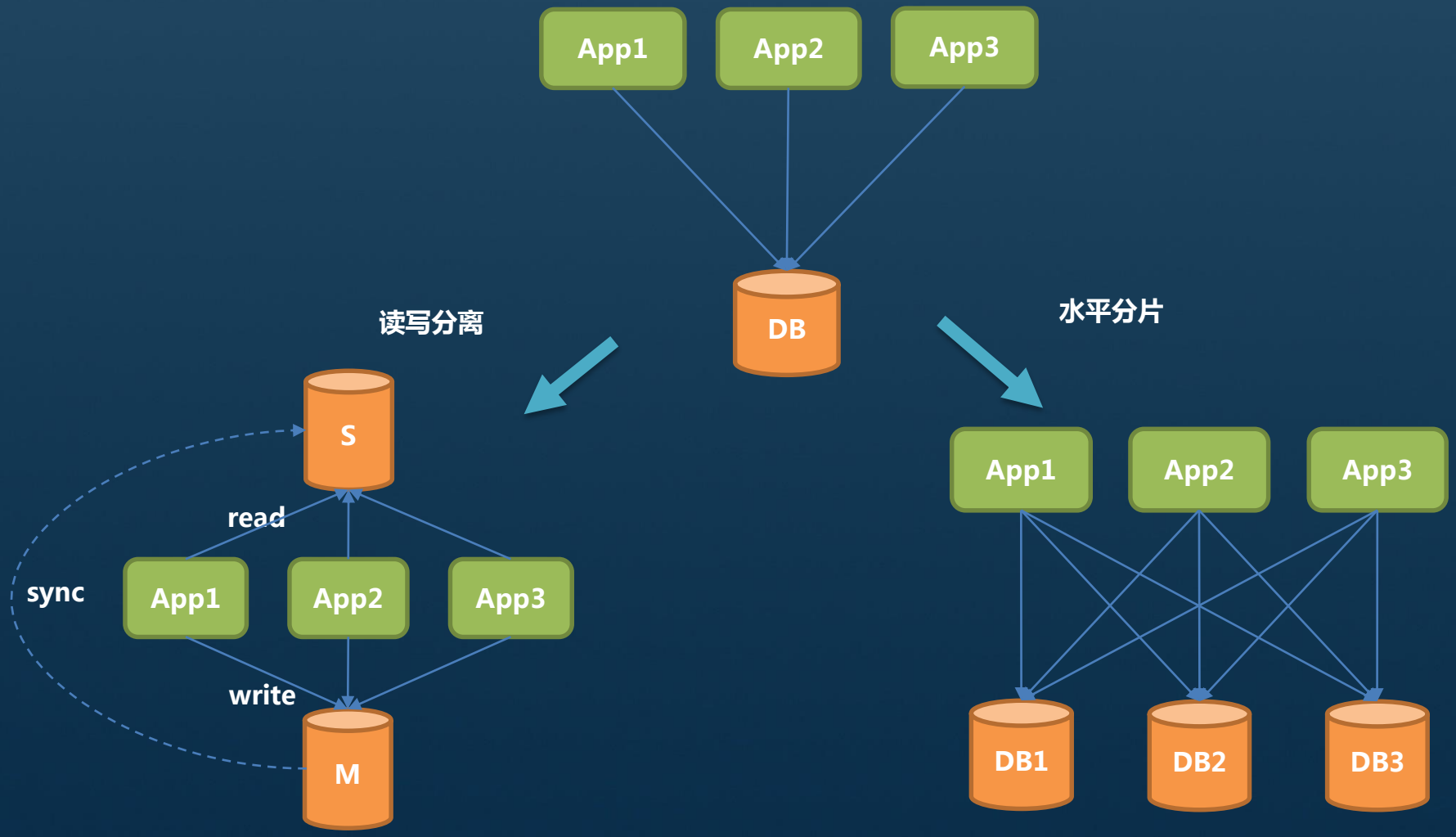


- ◆完全开源
- ◆Apache的首个分布式数据库中间件
- ◆gitHub近8000star，近百家公司的成功落地案例
- ◆核心功能：数据分片&分布式事务&数据库治理
- ◆多接入端选择
- ◆京东主导，多公司&社区参与推动

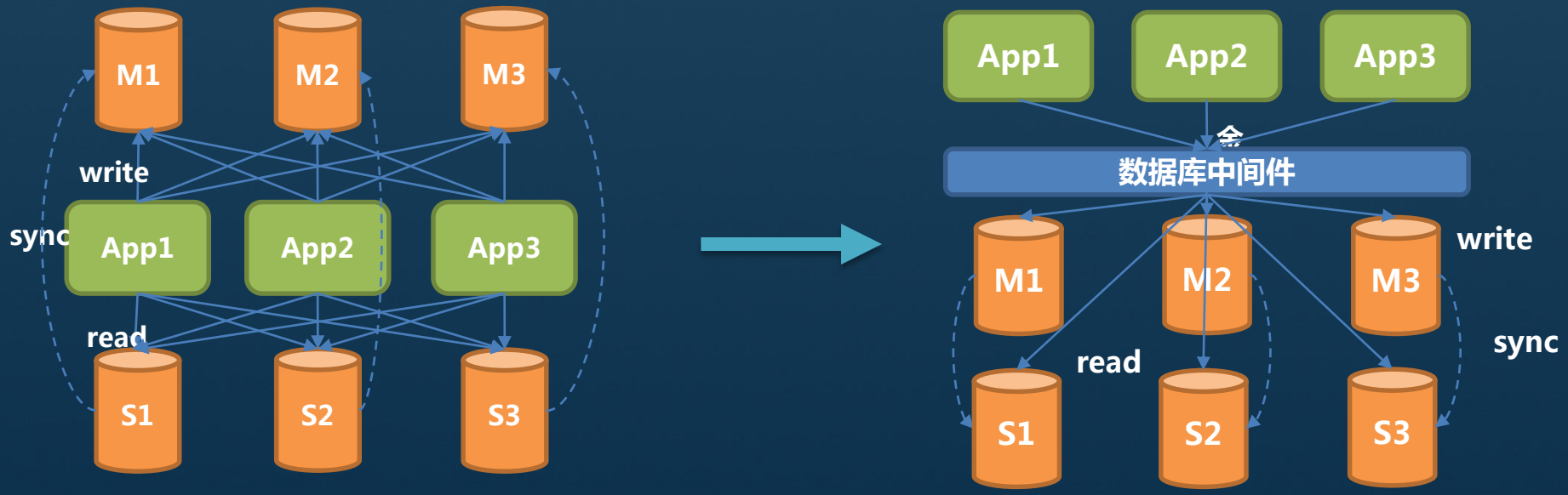
Apache ShardingSphere生态



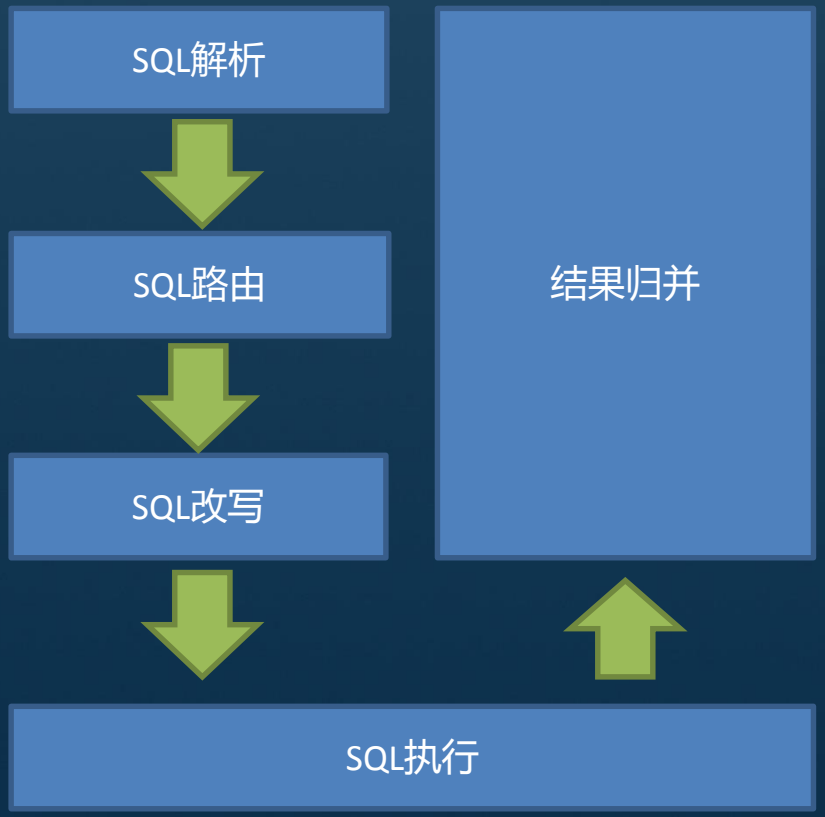
数据分片


App1
App2
App3
DB
读写分离
水平分片
S
read
sync
App1
App2
App3
write
M
App1
App2
App3
DB1
DB2
DB3

数据分片



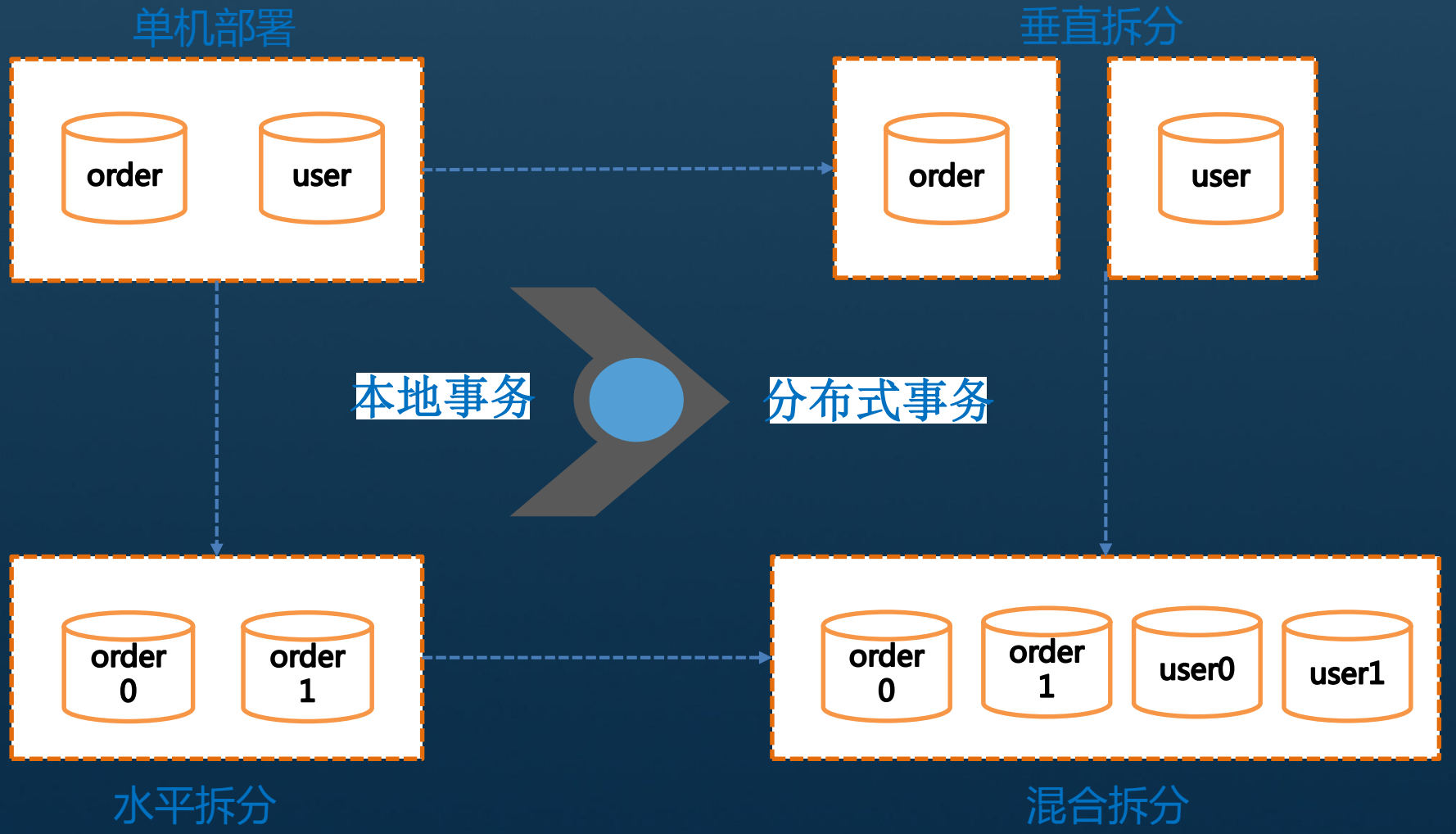
数据分片



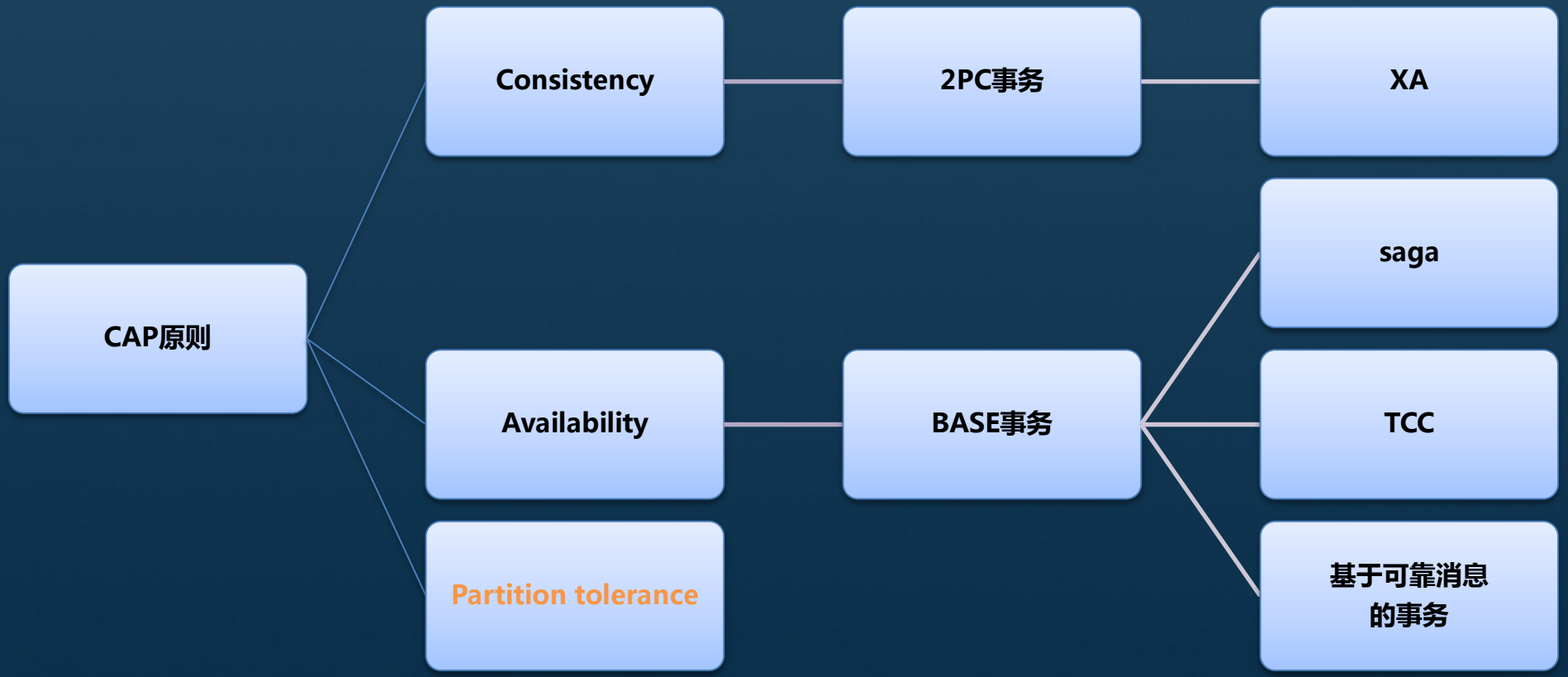
分布式治理

数据治理	可观察性	服务治理
• 权限 • 数据脱敏 • SQL防火墙 & SQL审核	• APM • 链路追踪 • 调用拓扑图 • 监控报警	• 配置动态化 • 高可用 • 过载保护 & 熔断 & 禁用 & 失效转移

分布式事务的需求体现


单机部署
order
user
垂直拆分
order
user
本地事务
分布式事务
order 0
order 1
水平拆分
order 0
order 1
user0
user1
混合拆分

分布式事务概述


CAP原则
Consistency
2PC事务
XA
Availability
BASE事务
saga
TCC
基于可靠消息
的事务
Partition tolerance

事务对比

	本地事务	两阶段事务	柔性事务
业务改造	无	无	实现相关接口
一致性	不支持	强一致	最终一致
隔离性	不支持	原生支持	服务保证
并发性能	无影响	严重衰退	略微衰退
适合场景	业务方处理不一致	短事务/低并发	长事务/高并发
事务层级	数据库层	数据库层	服务层

Saga概述

◆论文出处：https://www.cs.cornell.edu/andru/cs711/2002fa/reading/sagas.pdf

◆一个长事务可以拆分成相对独立的若干子事务

$LLT = T_1 + T_2 + \ldots + T_j + \ldots + T_n (j < \mathrm{n})$

◆除了最后一个子事务外，每个子事务都对应着一个补偿子事务

$T_1 + T_2 + \ldots + T_j + \ldots + T_n$

$C_1 + C_2 + \ldots + C_j + \ldots + C_{n-1} \ (j < \mathrm{n} - 1 < \mathrm{n})$

◆SAVEPOINT用于持久化各个子任务的执行状态

◆补偿行为：Forward Recovery/Backward Recovery/混合

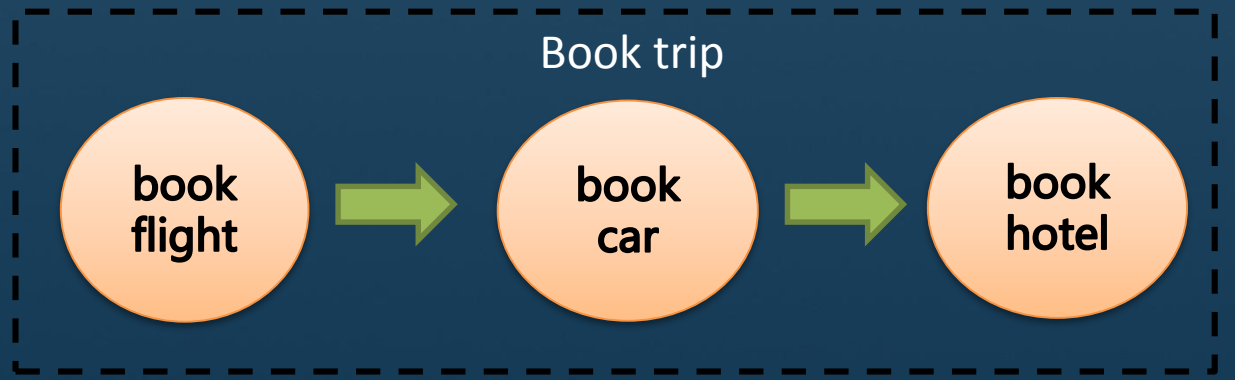
◆执行路径

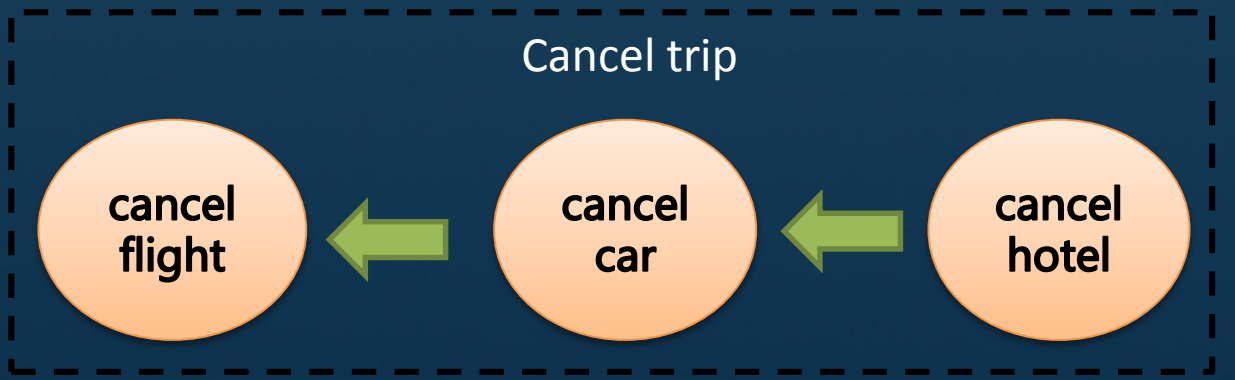
$T_1 + T_2 + \ldots + T_j + \ldots + T_n (j < \mathrm{n})$

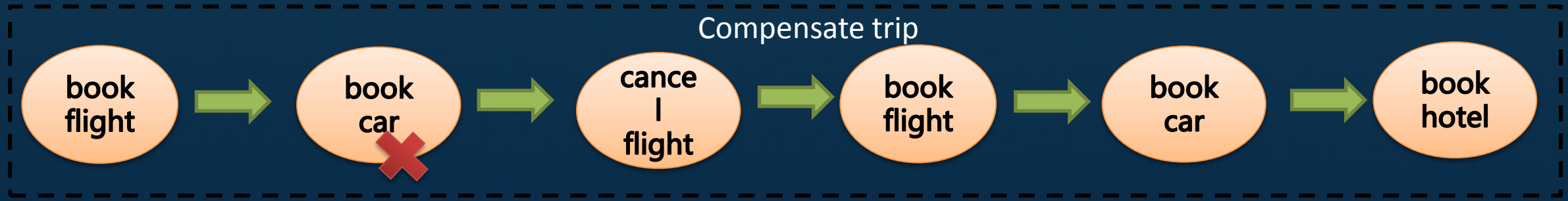
$T_1 + T_2 + \ldots + T_j + C_j + C_{j-1} \ldots + C_1 (j < \mathrm{n})$

$T_1 + T_2 + \ldots + T_j + C_{j-1} + T_{j-1} + T_j \ldots + T_n (j < \mathrm{n})$

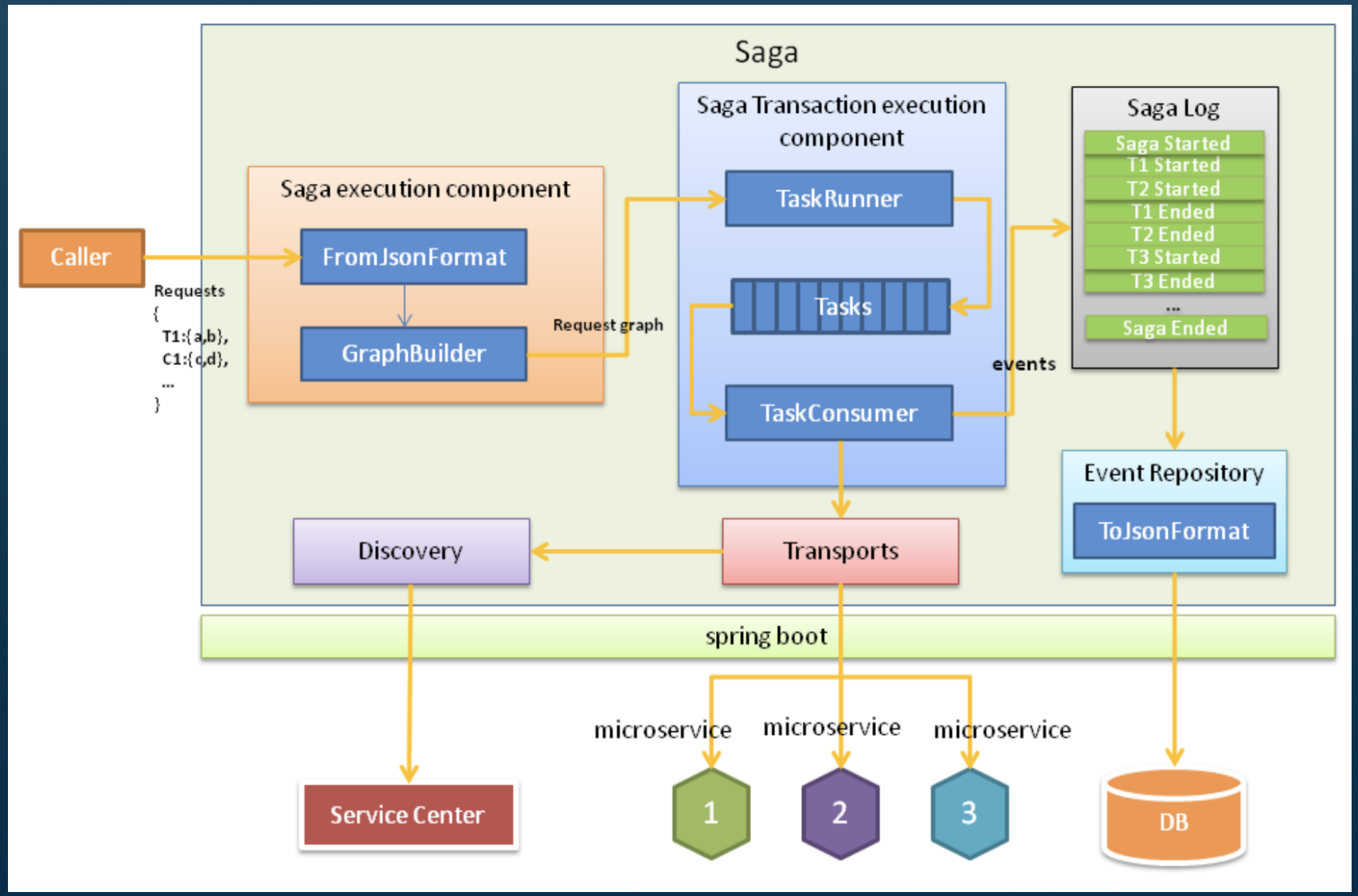
Saga概述



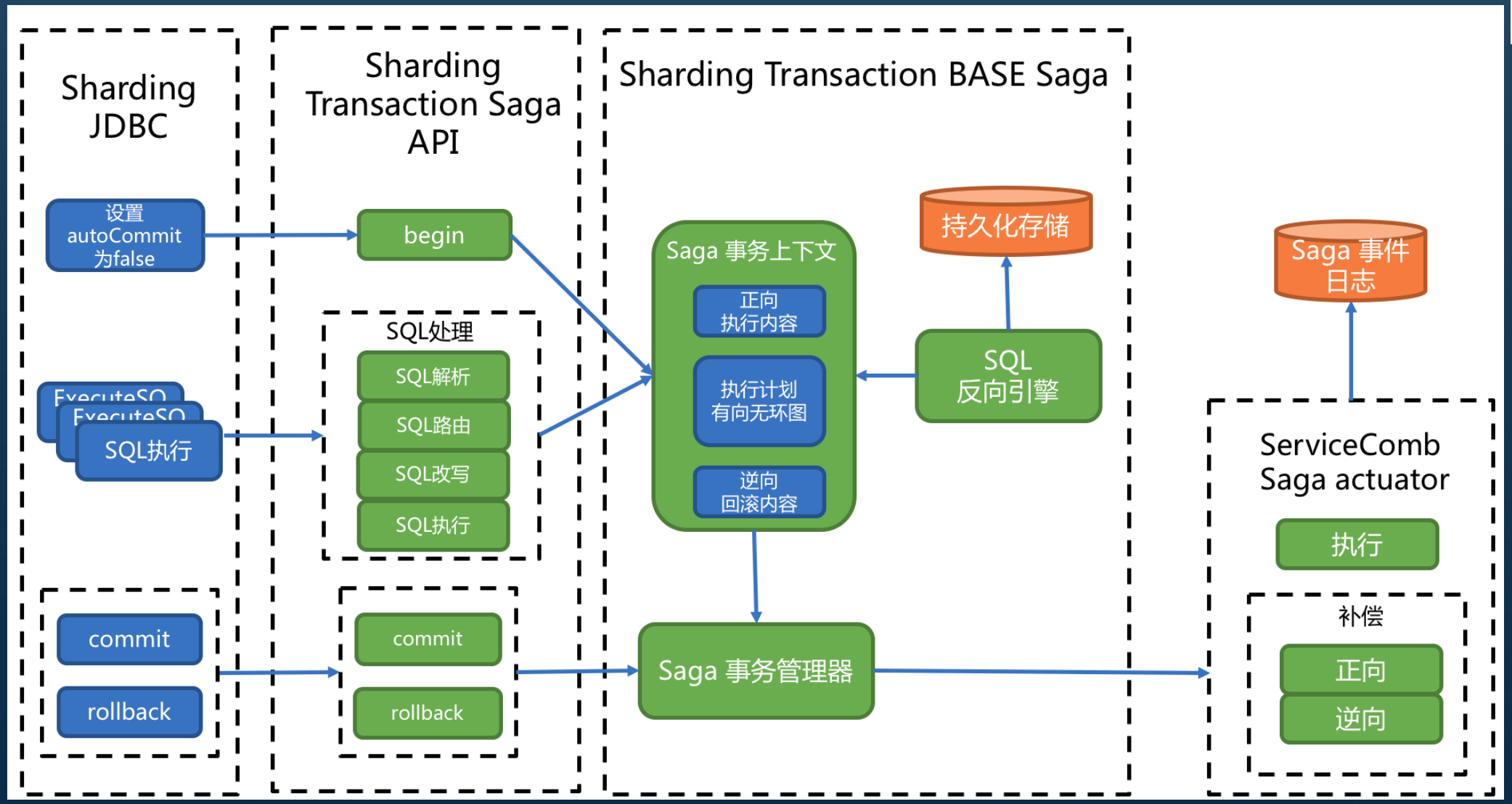




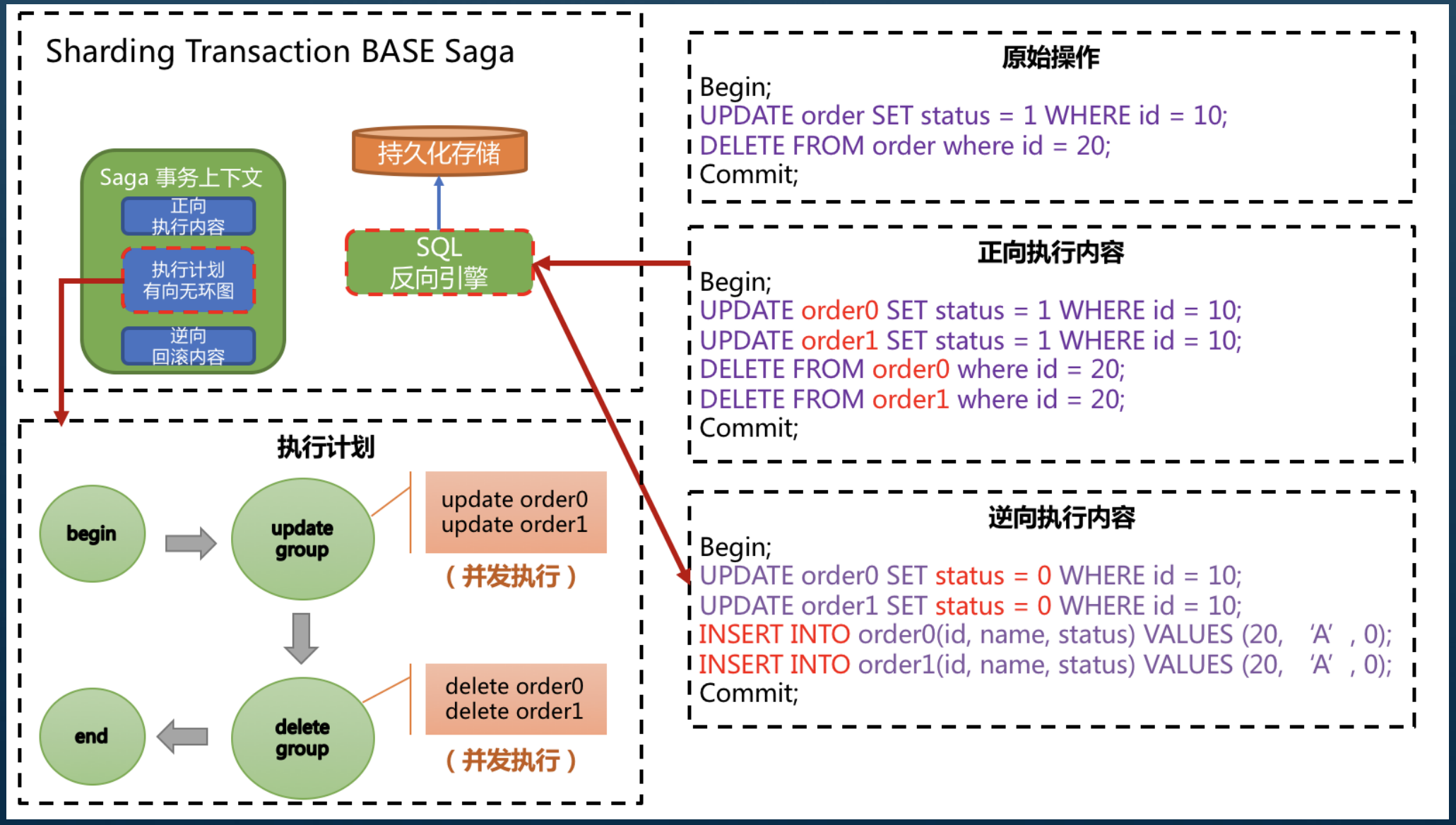
ServiceComb-saga



分布式事务解决方案

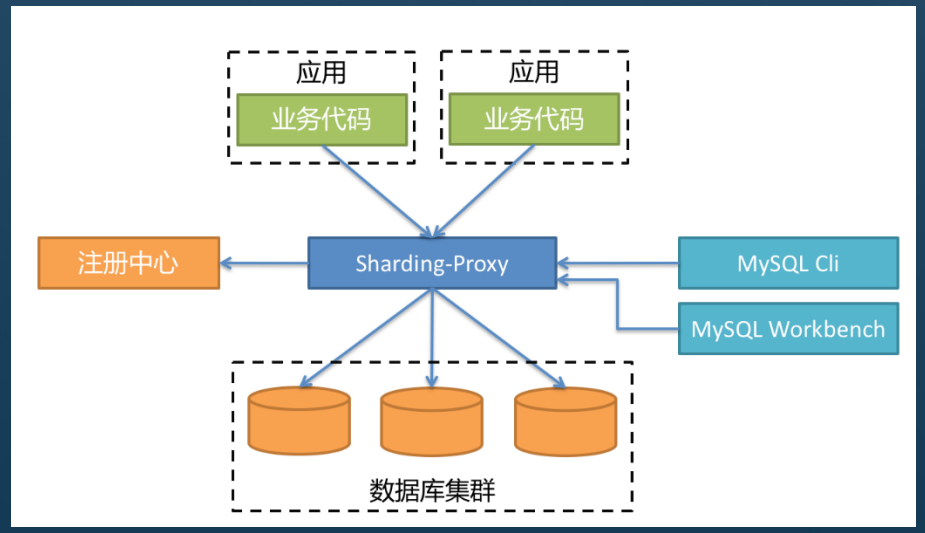


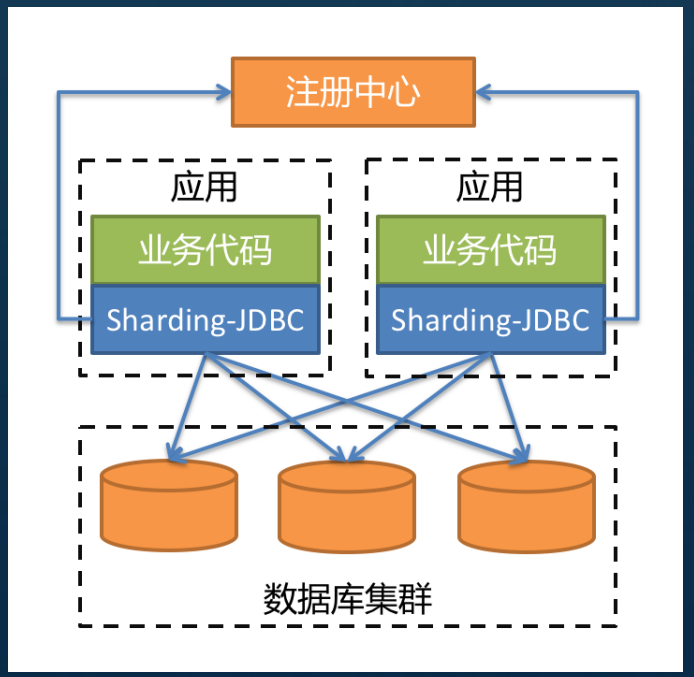
分布式事务解决方案


Sharding Transaction BASE Saga
Saga 事务上下文
正向
执行内容
执行计划
有向无环图
逆向
回滚内容
持久化存储
SQL
反向引擎
执行计划
begin
update
group
update order0
update order1
（并发执行）
end
delete
group
delete order0
delete order1
（并发执行）
原始操作
Begin;
UPDATE order SET status = 1 WHERE id = 10;
DELETE FROM order where id = 20;
Commit;
正向执行内容
Begin;
UPDATE order0 SET status = 1 WHERE id = 10;
UPDATE order1 SET status = 1 WHERE id = 10;
DELETE FROM order0 where id = 20;
DELETE FROM order1 where id = 20;
Commit;
逆向执行内容
Begin;
UPDATE order0 SET status = 0 WHERE id = 10;
UPDATE order1 SET status = 0 WHERE id = 10;
INSERT INTO order0(id, name, status) VALUES (20, 'A', 0);
INSERT INTO order1(id, name, status) VALUES (20, 'A', 0);
Commit;

接入端选择

	JDBC	Proxy
数据库	任意	单一
连接数	高	低
异构语言	仅Java	任意
性能	损耗低	损耗略高
无中心化	是	否
静态入口	无	有





欢迎加入我们

https://github.com/apache/incubator-shardingsphere

https://shardingsphere.apache.org/

Thank You.

Grow with Intelligence

servicecomb.apache.org
github.com/apache?q=servicecomb
www.huaweicloud.com