微信号：

scottlewis

分布式事务框架 Seata-Golang

×

刘晓敏

H3C

目 录

什么是分布式事务问题?

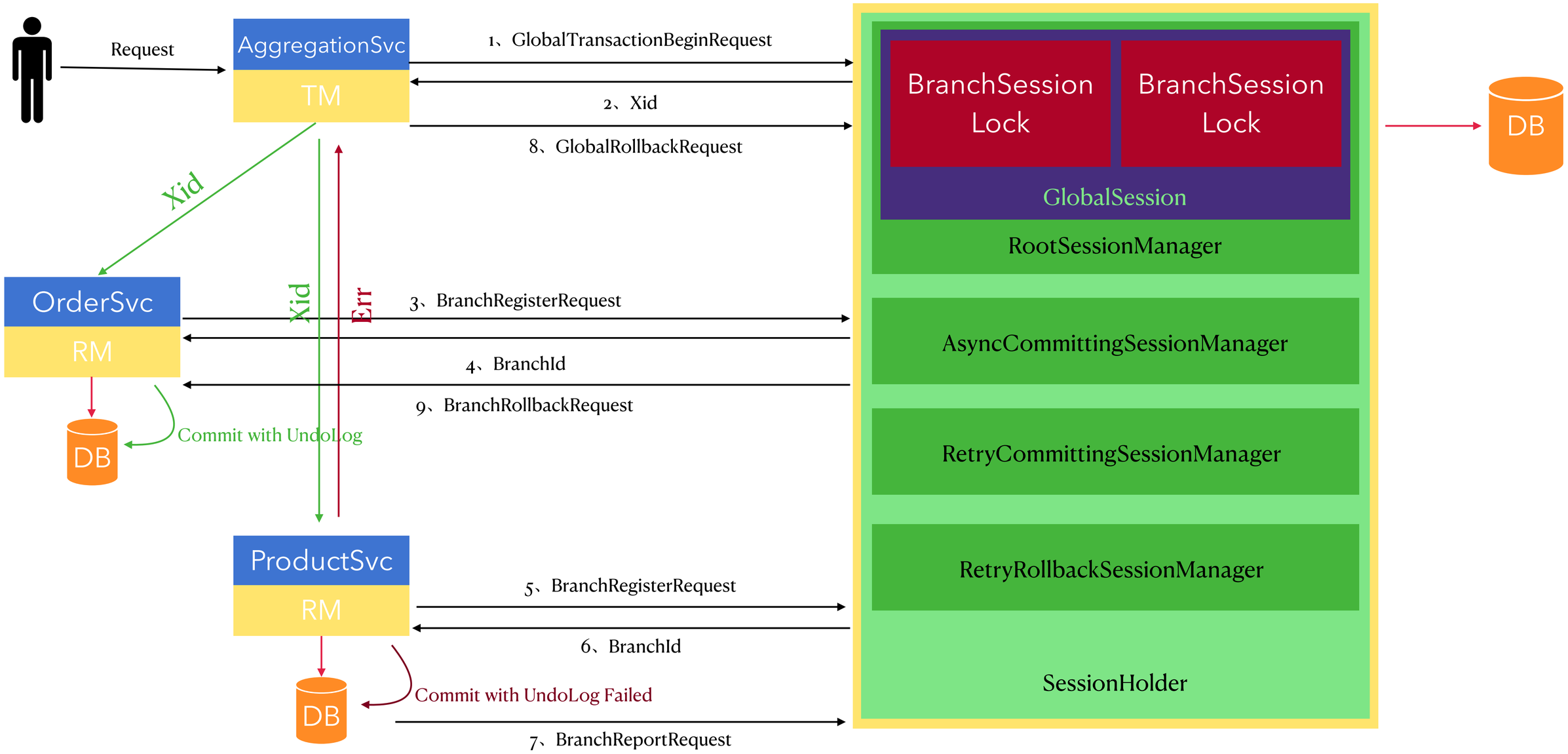
分布式事务就是指事务的参与者、支持事务的服务器、资源服务器以及事务管理器分别位于不同的分布式系统的不同节点之上。简单的说，就是一次大的操作由不同的小操作组成，这些小的操作分布在不同的服务器上，且属于不同的应用，分布式事务需要保证这些小操作要么全部成功，要么全部失败。本质上来说，分布式事务就是为了保证不同数据库的数据一致性。

Demo 演示

Seata 原理

整体机制：

- 一阶段：业务数据和回滚日志记录在同一个本地事务中提交，释放本地锁和连接资源。
- 二阶段：
  - 提交异步化，非常快速地完成。
  - 回滚通过一阶段的回滚日志进行反向补偿。


Request
AggregationSvc
TM
1、GlobalTransactionBeginRequest
2、Xid
8、GlobalRollbackRequest
Xid
OrderSvc
RM
Xid
Err
3、BranchRegisterRequest
4、BranchId
9、BranchRollbackRequest
DB
Commit with UndoLog
ProductSvc
RM
5、BranchRegisterRequest
6、BranchId
DB
Commit with UndoLog Failed
7、BranchReportRequest
BranchSession Lock
BranchSession Lock
GlobalSession
RootSessionManager
AsyncCommittingSessionManager
RetryCommittingSessionManager
RetryRollbackSessionManager
SessionHolder
DB

Mysql Driver 原理

在每个 mysqlConn 对象中，都创建了一个 connCtx 对象，在调用 BeginTx(ctx context.Context, opts driver.TxOptions) (driver.Tx, error) 方法的时候，会根据 ctx 中是否具有 xid 的值来初始化 connCtx 对象。

```
val := ctx.Value(XID)
if val != nil {
        if xid, ok := val.(string); ok {
                mc.ctx = &connCtx{
                        xid:              xid,
                        lockKeys:         make([]string, 0),
                        sqlUndoItemsBuffer: make([]*sqlUndoLog, 0),
                }
        }
}
```

Mysql Driver 原理

mysqlConn 对象的 PrepareContext(ctx context.Context, query string) (driver.Stmt, error) 方法会返回一个 mysqlStmt 对象，在 mysqlStmt 执行 Exec(args []driver.Value) (driver.Result, error) 方法时，会拦截执行的 sql 语句，解析 sql 语句后，获得 sql 语句执行的前后镜像构造成 undoLog 填充到 connCtx 的 sqlUndoItemsBuffer 中。

UndoLog


```
{
    "branchId": 641789253,
    "undoItems": [{
        "afterImage": {
            "rows": [{
                "fields": [{
                    "name": "id",
                    "type": 4,
                    "value": 1
                }, {
                    "name": "name",
                    "type": 12,
                    "value": "GTS"
                }, {
                    "name": "since",
                    "type": 12,
                    "value": "2014"
                }]
            }],
            "tableName": "product"
        },
        "beforeImage": {
            "rows": [{
                "fields": [{
                    "name": "id",
                    "type": 4,
                    "value": 1
                }, {
                    "name": "name",
                    "type": 12,
                    "value": "TXC"
                }, {
                    "name": "since",
                    "type": 12,
                    "value": "2014"
                }]
            }],
            "tableName": "product"
        },
        "sqlType": "UPDATE"
    }],
    "xid": "xid:xxx"
}
```


```go
// PK is at last one.
// INSERT INTO a (x, y, z, pk) VALUES (?, ?, ?, ?)
// UPDATE a SET x=?, y=?, z=? WHERE pk = ?
// DELETE FROM a WHERE pk = ?
//todo 后镜数据和当前数据比较，判断是否可以回滚数据
stmt,err := tx.Prepare(undoSql)
if err != nil : err ↗
defer stmt.Close()

for _,row := range undoRows.Rows {
    var args = make([]interface{},0)
    var pkValue interface{}

    for _, field := range row.Fields {
        if field.KeyType == _struct.PRIMARY_KEY {
            pkValue = field.Value
        } else {
            if executor.sqlUndoLog.SqlType != sqlparser.SQLType_INSERT
                args = append(args, field.Value)
            }
        }
    }
    args = append(args,pkValue)
    _,err = stmt.Exec(args...)
    if err != nil : err ↗
}
```

Mysql Driver 原理

在 mysqlTx Commit 或者 Rollback 时，会根据 mysqlConn 的 connCtx 是否有值来决定是否和 tc 交互，报告分支事务的执行状态。如果执行 Commit，connCtx 有值则把 sqlUndoItemsBuffer 中的 undoLog 和业务数据一起提交到数据库，然后报告 tc 事务分支提交的状态（成功还是失败），否则执行正常的提交。如果执行 Rollback，connCtx 有值则回滚然后向 tc 报告分支执行失败，tc 会根据这个状态回滚整个全局事务，connCtx 没有值则只需正常回滚。

Mysql driver 接入演示

TODO & QA

注册中心配置中心支持

PostgreSQL 支持

GRPC 支持

https://github.com/opentrx/seata-golang

https://github.com/opentrx/mysql

https://github.com/opentrx/seata-go-samples

https://github.com/apache/dubbo-getty

https://github.com/apache/dubbo-go

seata-golang 社区

115人