Seata AT 模式处理逻辑

设计思路

将一个分布式事务可以理解成一个全局事务，下面挂了若干个分支事务，而一个分支事务是一个满足 ACID 的本地事务，因此我们可以操作分布式事务像操作本地事务一样。

全局事务和分支事务的关系和处理过程



•Transaction Coordinator (TC)：事务协调器，维护全局事务的运行状态，负责协调并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，并最终发起全局提交或全局回滚的决议。

•Resource Manager (RM)：控制分支事务，负责分支注册、状态汇报，并接收事务协调器的指令，驱动分支（本地）事务的提交和回滚。

执行步骤：

1.TM 向 TC 申请开启一个全局事务，TC 创建全局事务后返回全局唯一的 XID，XID 会在全局事务的上下文中传播；

2.RM 向 TC 注册分支事务，该分支事务归属于拥有相同 XID 的全局事务；

3.TM 向 TC 发起全局提交或回滚；

4.TC 调度 XID 下的分支事务完成提交或者回滚。

服务端启动流程



服务端处理逻辑

客户端 [RegisterRMRequest/RegisterTMRequest ] —-> 服务端[ServerMessageListener]—>[ChannelManager ]缓存客户端连接 …..>

TM 向 TC 发起全局提交或回滚 …..>

[DefaultCoordinator]调度 XID 下的分支事务完成提交或者回滚

io.seata.core.rpc.ServerMessageListener

**public** **interface** **ServerMessageListener** {

*//* 处理各种事务，如分支注册、分支提交、分支上报、分支回滚等等

**void** **onTrxMessage**(RpcMessage request, ChannelHandlerContext ctx, ServerMessageSender sender);

*//* 处理 *RM* 客户端的注册连接

**void** **onRegRmMessage**(RpcMessage request, ChannelHandlerContext ctx,

ServerMessageSender sender, RegisterCheckAuthHandler checkAuthHandler);

*//* 处理 *TM* 客户端的注册连接

**void** **onRegTmMessage**(RpcMessage request, ChannelHandlerContext ctx,

ServerMessageSender sender, RegisterCheckAuthHandler checkAuthHandler);

*//* 服务端与客户端保持心跳

**void** **onCheckMessage**(RpcMessage request, ChannelHandlerContext ctx, ServerMessageSender sender)

}

io.seata.core.rpc.ChannelManager

*/\*\**

*\* resourceId -> applicationId -> ip -> port -> RpcContext*

*\* resourceId* 指的是 *RM client* 的数据库地址；

*\* applicationId* 指的是 *RM client* 的服务 *Id*，比如 *springboot* 的*spring.application.name*

*\* ip* 指的是 *RM client* 服务地址；

*\* port* 指的是 *RM client* 服务地址；

*\* RpcContext* 保存了本次注册请求的信息。

*\*/*

**private** **static** **final** ConcurrentMap<String, ConcurrentMap<String, ConcurrentMap<String, ConcurrentMap<Integer,

RpcContext>>>>

RM\_CHANNELS = **new** ConcurrentHashMap<String, ConcurrentMap<String, ConcurrentMap<String, ConcurrentMap<Integer,

RpcContext>>>>();

*/\*\**

*\* appname+ip,port*

*\*/*

**private** **static** **final** ConcurrentMap<String, ConcurrentMap<Integer, RpcContext>> TM\_CHANNELS

= **new** ConcurrentHashMap<String, ConcurrentMap<Integer, RpcContext>>();

TM发起全局事务

io.seata.server.coordinator.DefaultCoordinator#doGlobalBegin

调用io.seata.server.coordinator.DefaultCore#begin

@Override

**public** String **begin**(String applicationId, String transactionServiceGroup, String name, int timeout)

throws TransactionException {

GlobalSession session = GlobalSession.createGlobalSession(applicationId, transactionServiceGroup, name,

timeout);

session.addSessionLifecycleListener(SessionHolder.getRootSessionManager());

session.begin();

//transaction start event

eventBus.post(new GlobalTransactionEvent(session.getTransactionId(), GlobalTransactionEvent.ROLE\_TC,

session.getTransactionName(), session.getBeginTime(), null, session.getStatus()));

LOGGER.info("Successfully begin global transaction xid = {}", session.getXid());

return session.getXid();

}

TM提交全局事务

io.seata.server.coordinator.DefaultCoordinator#doGlobalCommit

调用io.seata.server.coordinator.DefaultCore#commit

调用io.seata.server.coordinator.DefaultCore#doGlobalCommit

循环调用io.seata.server.coordinator.DefaultCoordinator#branchCommit

TM回滚全局事务

io.seata.server.coordinator.DefaultCoordinator#doGlobalRollback

调用io.seata.server.coordinator.DefaultCore#rollback

调用io.seata.server.coordinator.DefaultCore#doGlobalRollback

循环调用io.seata.server.coordinator.DefaultCoordinator#branchRollback

**public interface** **Core** **extends** TransactionManager, ResourceManagerOutbound {

// 设置RM输入接口，提供向RM发送rpc的两个方法（branchCommit，branchRollback）

void **setResourceManagerInbound**(ResourceManagerInbound resourceManagerInbound);

// 接受TM的globalCommit,此处为具体globalCommit逻辑

void **doGlobalCommit**(GlobalSession globalSession, boolean retrying) throws TransactionException;

// 接受TM的globalRollback,此处为具体globalRollback逻辑

void **doGlobalRollback**(GlobalSession globalSession, boolean retrying) throws TransactionException;

}

io.seata.server.coordinator.DefaultCoordinator

@Override

**public** BranchStatus **branchCommit**(BranchType branchType, String xid, long branchId, String resourceId,

String applicationData) throws TransactionException {

try {

BranchCommitRequest request = new BranchCommitRequest();

request.setXid(xid);

request.setBranchId(branchId);

request.setResourceId(resourceId);

request.setApplicationData(applicationData);

request.setBranchType(branchType);

GlobalSession globalSession = SessionHolder.findGlobalSession(xid);

if (globalSession == null) {

return BranchStatus.PhaseTwo\_Committed;

}

if (BranchType.SAGA.equals(branchType)) {

Map<String, Channel> channels = ChannelManager.getRmChannels();

if (channels == null || channels.size() == 0) {

LOGGER.error("Failed to commit SAGA global[" + globalSession.getXid() + ", RM channels is empty.");

return BranchStatus.PhaseTwo\_CommitFailed\_Retryable;

}

String sagaResourceId = globalSession.getApplicationId() + "#" + globalSession

.getTransactionServiceGroup();

Channel sagaChannel = channels.get(sagaResourceId);

if (sagaChannel == null) {

LOGGER.error("Failed to commit SAGA global[" + globalSession.getXid()

+ ", cannot find channel by resourceId[" + sagaResourceId + "]");

return BranchStatus.PhaseTwo\_CommitFailed\_Retryable;

}

BranchCommitResponse response = (BranchCommitResponse)messageSender.sendSyncRequest(sagaChannel,

request);

return response.getBranchStatus();

} else {

BranchSession branchSession = globalSession.getBranch(branchId);

if (null != branchSession) {

BranchCommitResponse response = (BranchCommitResponse)messageSender.sendSyncRequest(resourceId,

branchSession.getClientId(), request);

return response.getBranchStatus();

} else {

return BranchStatus.PhaseTwo\_Committed;

}

}

} catch (IOException | TimeoutException e) {

throw new BranchTransactionException(FailedToSendBranchCommitRequest,

String.format("Send branch commit failed, xid = %s branchId = %s", xid, branchId), e);

}

}

@Override

**public** BranchStatus **branchRollback**(BranchType branchType, String xid, long branchId, String resourceId,

String applicationData) throws TransactionException {

try {

BranchRollbackRequest request = new BranchRollbackRequest();

request.setXid(xid);

request.setBranchId(branchId);

request.setResourceId(resourceId);

request.setApplicationData(applicationData);

request.setBranchType(branchType);

GlobalSession globalSession = SessionHolder.findGlobalSession(xid);

if (globalSession == null) {

return BranchStatus.PhaseTwo\_Rollbacked;

}

if (BranchType.SAGA.equals(branchType)) {

Map<String, Channel> channels = ChannelManager.getRmChannels();

if (channels == null || channels.size() == 0) {

LOGGER.error(

"Failed to rollback SAGA global[" + globalSession.getXid() + ", RM channels is empty.");

return BranchStatus.PhaseTwo\_RollbackFailed\_Retryable;

}

String sagaResourceId = globalSession.getApplicationId() + "#" + globalSession

.getTransactionServiceGroup();

Channel sagaChannel = channels.get(sagaResourceId);

if (sagaChannel == null) {

LOGGER.error("Failed to rollback SAGA global[" + globalSession.getXid()

+ ", cannot find channel by resourceId[" + sagaResourceId + "]");

return BranchStatus.PhaseTwo\_RollbackFailed\_Retryable;

}

BranchRollbackResponse response = (BranchRollbackResponse)messageSender.sendSyncRequest(sagaChannel,

request);

return response.getBranchStatus();

} else {

BranchSession branchSession = globalSession.getBranch(branchId);

BranchRollbackResponse response = (BranchRollbackResponse)messageSender.sendSyncRequest(resourceId,

branchSession.getClientId(), request);

return response.getBranchStatus();

}

} catch (IOException | TimeoutException e) {

throw new BranchTransactionException(FailedToSendBranchRollbackRequest,

String.format("Send branch rollback failed, xid = %s branchId = %s", xid, branchId), e);

}

}

参考

<http://seata.io/zh-cn/blog/seata-at-mode-start.html>

<https://mp.weixin.qq.com/s/Pypkm5C9aLPJHYwcM6tAtA>

<https://gitlab.wxbpdapp.com/root/seata/blob/master/core/src/main/java/io/seata/core/rpc/ChannelManager.java>

<http://seata.io/zh-cn/blog/seata-sourcecode-server-bootstrap.html>

<https://www.codeleading.com/article/34553983905/>

<https://gitlab.wxbpdapp.com/root/seata/blob/master/server/src/main/java/io/seata/server/coordinator/DefaultCore.java>