

分布式事务之TX-LCN

一、简介

1.1 CAP定律

这个定理的内容是指的是在一个分布式系统中、Consistency（一致性）、Availability（可用性）、Partition tolerance（分区容错性），三者不可得兼。

一致性（C）

在分布式系统中的所有数据备份，在同一时刻是否同样的值。（等同于所有节点访问同一份最新的数据副本）

可用性（A）

在集群中一部分节点故障后，集群整体是否还能响应客户端的读写请求。（对数据更新具备高可用性）

分区容错性（P）

以实际效果而言，分区相当于对通信的时限要求。系统如果不能在时限内达成数据一致性，就意味着发生了分区的情况，必须就当前操作在C和A之间做出选择。

1.2 BASE理论

BASE是Basically Available（基本可用）、Soft state（软状态）和 Eventually consistent（最终一致性）三个短语的缩写。BASE理论是对CAP中一致性和可用性权衡的结果，其来源于对大规模互联网系统分布式实践的总结，是基于CAP定理逐步演化而来的。BASE理论的核心思想是：即使无法做到强一致性，但每个应用都可以根据自身业务特点，采用适当的方式来使系统达到最终一致性。

基本可用

基本可用是指分布式系统在出现不可预知故障的时候，允许损失部分可用性----注意，这绝不等价于系统不可用。比如：

（1）响应时间上的损失。正常情况下，一个在线搜索引擎需要在0.5秒之内返回给用户相应的查询结果，但由于出现故障，查询结果的响应时间增加了1~2秒

（2）系统功能上的损失：正常情况下，在一个电子商务网站上进行购物的时候，消费者几乎能够顺利完成每一笔订单，但是在一些节日大促购物高峰的时候，由于消费者的购物行为激增，为了保护购物系统的稳定性，部分消费者可能会被引导到一个降级页面

软状态

软状态指允许系统中的数据存在中间状态，并认为该中间状态的存在不会影响系统的整体可用性，即允许系统在不同节点的数据副本之间进行数据同步的过程存在延时

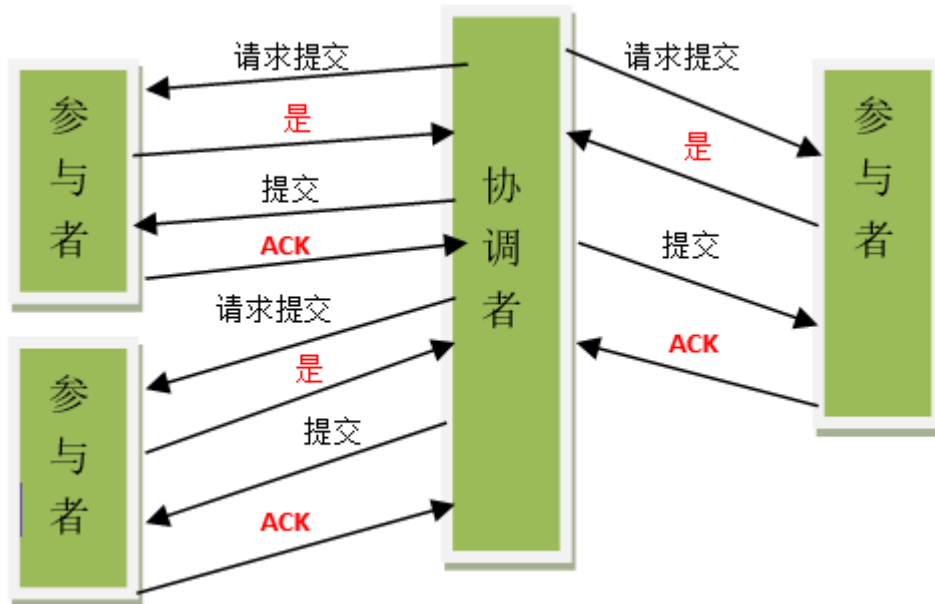
最终一致性

最终一致性强调的是所有的数据副本，在经过一段时间的同步之后，最终都能够达到一个一致的状态。因此，最终一致性的本质是需要系统保证最终数据能够达到一致，而不需要实时保证系统数据的强一致性。

1.3 分布式事务实现方式

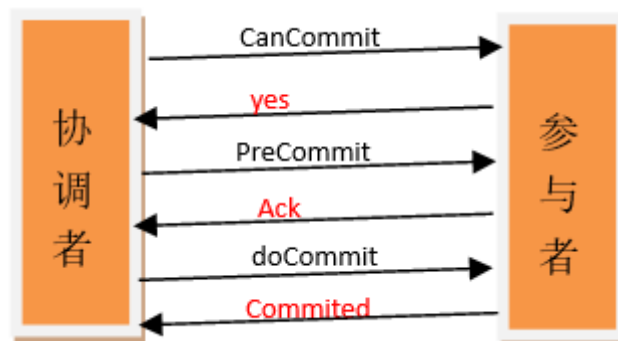
2PC

所谓的两个阶段是指：第一阶段：准备阶段(投票阶段) 和第二阶段：提交阶段（执行阶段）



3PC

三阶段提交协议 (Three-phase commit protocol) , 是二阶段提交 (2PC) 的改进版本。



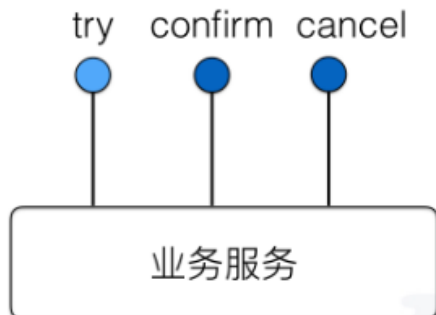
与两阶段提交不同的是，三阶段提交有两个改动点。

1. 引入超时机制。同时在协调者和参与者中都引入超时机制。
2. 在第一阶段和第二阶段中插入一个准备阶段。保证了在最后提交阶段之前各参与节点的状态是一致的。

也就是说，除了引入超时机制之外，3PC 把 2PC 的准备阶段再次一分为二，这样三阶段提交就有 CanCommit、PreCommit、DoCommit 三个阶段。

TCC

TCC



- Try: 尝试执行业务
 - 完成所有业务检查，预留必须的业务资源
- Confirm: 确认执行业务
 - 真正执行业务，不做业务检查
- Cancel: 取消执行业务
 - 释放Try阶段预留的业务资源

1.4 刚性事务和柔性事务

ACID刚性事务

基于Base理论实现的分布式事务，柔性事务，XA事务

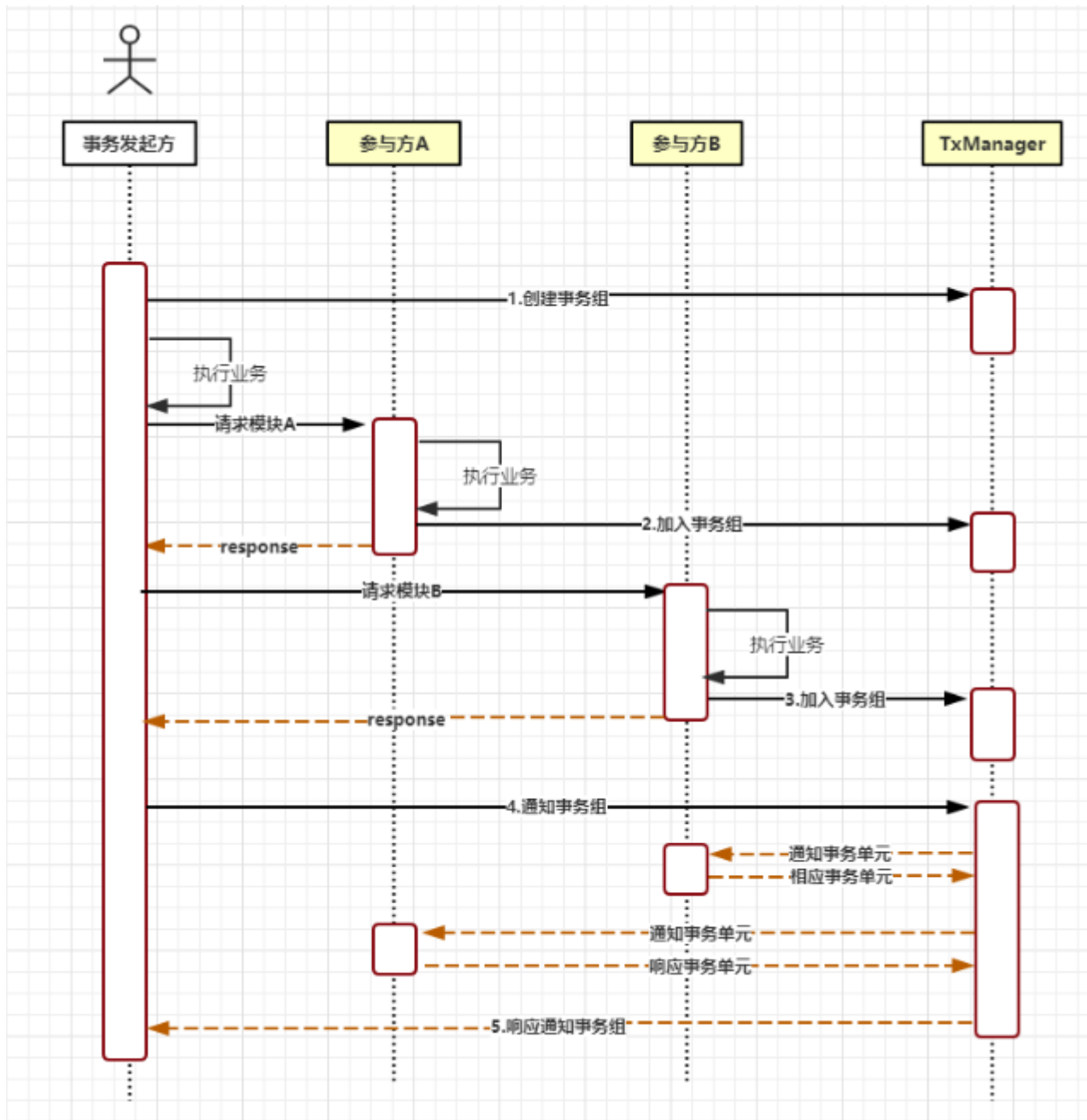
目前主流的分布式事务框架：

TX-LCN

SAGA 华为 贡献给了Apache

1.5 TX-LCN

TX-LCN:分布式事务搬运工, TX-LCN由两大模块组成, TxClient、TxManager, TxClient作为模块的依赖框架, 提供TX-LCN的标准支持, TxManager作为分布式事务的控制处理器。事务发起方或者参与反都由TxClient端来控制。



核心步骤

- 创建事务组

是指在事务发起方开始执行业务代码之前先调用TxManager创建事务组对象，然后拿到事务标识GroupId的过程。

- 加入事务组

添加事务组是指参与方在执行完业务方法以后，将该模块的事务信息通知给TxManager的操作。

- 通知事务组

是指在发起方执行完业务代码以后，将发起方执行结果状态通知给TxManager,TxManager将根据事务最终状态和事务组的信息来通知相应的参与模块提交或回滚事务，并返回结果给事务发起方。

TX-LCN是基于Consul注册中心实现的，所以需要先配置注册中心

二、初体验

<http://www.txlcn.org/zh-cn/docs/start.html>

2.1 搭建并运行Tx-Manager项目

1、下载源码

<https://github.com/codingapi/tx-lcn/releases/tag/5.0.2.RELEASE>

2、导入到IDE

3、配置

数据库

执行一下脚本

```
create database txmanager1901; use txmanager1901; CREATE TABLE t_tx_exception ( id bigint(20) NOT NULL AUTO_INCREMENT, group_id varchar(64) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT NULL, unit_id varchar(32) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT NULL, mod_id varchar(128) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT NULL, transaction_state tinyint(4) NULL DEFAULT NULL, registrar tinyint(4) NULL DEFAULT NULL, remark varchar(4096) NULL DEFAULT NULL, ex_state tinyint(4) NULL DEFAULT NULL COMMENT '0 未解决 1已解决', create_time datetime(0) NULL DEFAULT NULL, PRIMARY KEY (id) USING BTREE ) ENGINE = InnoDB AUTO_INCREMENT = 1 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci ROW_FORMAT = Dynamic;
```

txlcn-tm项目的配置文件

修改数据库的相关配置

、、

```
spring.application.name=TransactionManager
server.port=7970
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/txmanager1901?characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.database-platform=org.hibernate.dialect.MySQL57InnoDBDialect
spring.jpa.hibernate.ddl-auto=update
spring.redis.host=39.105.189.141
spring.redis.password=qfjava
spring.redis.port=6379
```

4、运行并测试

运行txlcn-tm项目的开关类，在浏览器中访问下面的地址：

<http://localhost:7970/admin/index.html>



TxManager系统后台

首页

异常记录

系统日志

配置信息

名称	值
TM主机IP	127.0.0.1
TM事务消息端口	8070
TM心跳时间	300000 ms
已注册的TC	0 查看详情
事务并发处理等级	20
分布式事务时间	8000 ms
事务异常通知	disabled
允许系统日志	false
TM版本号	5.0.2.RELEASE

2.2 运行默认应用案例

1、下载示例代码

<https://github.com/codingapi/txlcn-demo>

2、导入程序

3、配置数据库

```
create table t_demo (id bigint(20) NOT NULL AUTO_INCREMENT, kid varchar(45) DEFAULT NULL, group_id varchar(64) DEFAULT NULL, demo_field varchar(255) DEFAULT NULL, app_name varchar(128) DEFAULT NULL, create_time datetime DEFAULT NULL, PRIMARY KEY (id)) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC;
```

4、修改配置文件

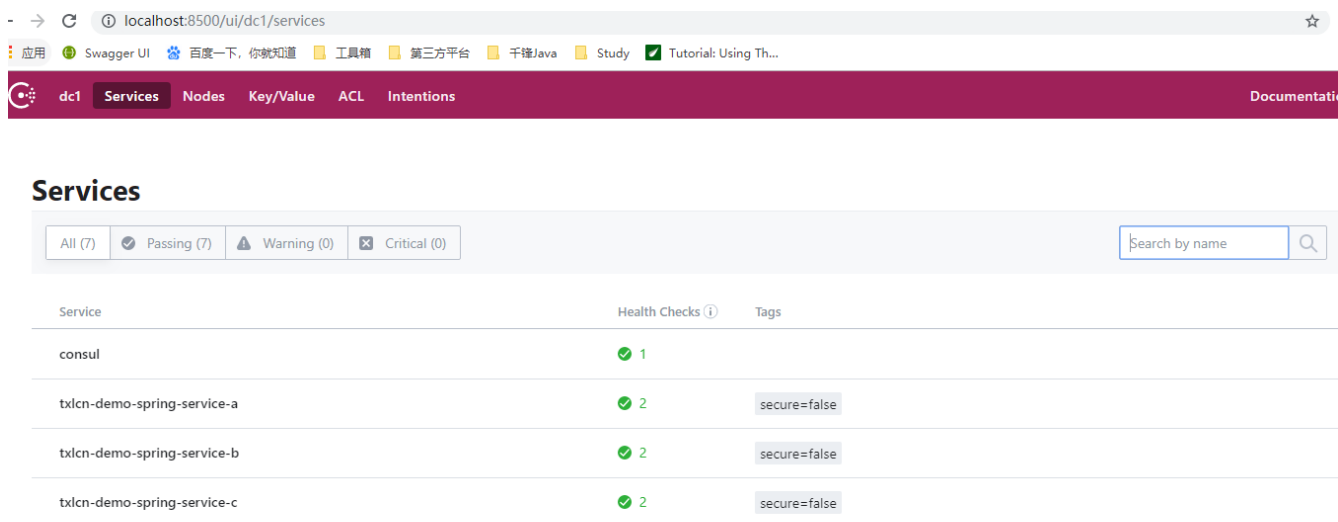
5、分别启动三个服务

Service-A

Service-B

Service-C

在注册中心可以看到发布的服务



The screenshot shows the Consul UI at localhost:8500/ui/dc1/services. The 'Services' tab is selected. The table below lists the services and their health status.

Service	Health Checks	Tags
consul	1 (Passing)	
txlcn-demo-spring-service-a	2 (Passing)	secure=false
txlcn-demo-spring-service-b	2 (Passing)	secure=false
txlcn-demo-spring-service-c	2 (Passing)	secure=false

6、测试事务是否生效

浏览器访问：

请求A服务的路径

<http://localhost:12011/txlcn?value=www&ex=11>

记得核对数据库的数据

如果成功：t_demo表中的数据 会新增2条

如果失败：没有数据新增

三、TX-LCN核心

3.1 事务模式

LCN:

LCN模式是通过代理Connection的方式实现对本地事务的操作，然后在由TxManager统一协调控制事务。当本地事务提交回滚或者关闭连接时将会执行假操作，该代理的连接将由LCN连接池管理

TCC:

TCC事务机制相对于传统事务机制（X/Open XA Two-Phase-Commit），其特征在于它不依赖资源管理器(RM)对XA的支持，而是通过对（由业务系统提供的）业务逻辑的调度来实现分布式事务。主要由三步操作，Try: 尝试执行业务、Confirm:确认执行业务、Cancel: 取消执行业务。

TXC:

TXC模式命名来源于淘宝，实现原理是在执行SQL之前，先查询SQL的影响数据，然后保存执行的SQL快走信息和创建锁。当需要回滚的时候就采用这些记录数据回滚数据库，目前锁实现依赖redis分布式锁（setnx）控制。

3.2 事务注解

1、开关类上启用TxLCN

```
@EnableDistributedTransaction
```

2、对应的业务逻辑中使用一下注解：

@LcnTransaction 采用LCN事务模式

@TxcTransaction 采用TXC事务模式

@TccTransaction 采用TCC事务模式

四、结合Tx-LCN实现分布式事务

模拟：下单

订单服务---订单表

库存服务---库存表

消息服务---消息表

4.1 订单服务

4.2 库存服务

4.3 消息服务

4.4 测试分布式事务

源码地址：