分布式事务产生背景

业务服务化拆分

在业务发展初期，“一块大饼”的单业务系统架构，能满足基本的业务需求。但是随着业务的快速发展，系统的访问量和业务复杂程度都在快速增长，单系统架构逐渐成为业务发展瓶颈，解决业务系统的高耦合、可伸缩问题的需求越来越强烈。

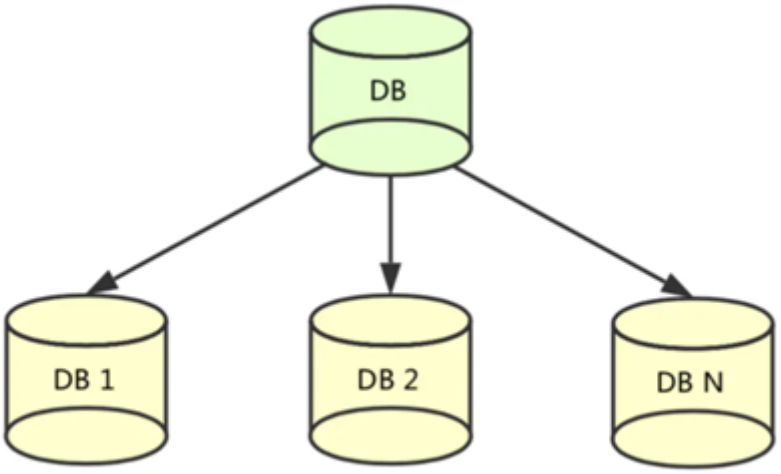
按照面向服务架构（SOA）的设计原则，将单业务系统拆分成多个业务系统，降低了各系统之间的耦合度，使不同的业务系统专注于自身业务，更有利于业务的发展和系统容量的伸缩。

业务系统按照服务拆分之后，一个完整的业务往往需要调用多个服务，如何保证多个服务间的数据一致性成为一个难题。

数据库拆分

业务数据库起初是单库单表，但随着业务数据规模的快速发展，数据量越来越大，单库单表逐渐成为瓶颈。所以我们对数据库进行了水平拆分，将原单库单表拆分成数据库分片。

如下图所示，分库分表之后，原来在一个数据库上就能完成的写操作，可能就会跨多个数据库，这就产生了跨数据库事务问题。



水平拆分成N个库

Seata 是什么

Seata 是一款开源的分布式事务解决方案，致力于提供高性能和简单易用的分布式事务服务。Seata 将为用户提供了 AT、TCC、SAGA 和 XA 事务模式，为用户打造一站式的分布

式解决方案。AT模式是阿里首推的模式,阿里云上有商用版本的GTS（Global Transaction service 全局事务服务）

官网：https://seata.io/zh-cn/index.html

源码: https://github.com/seata/seata

Seata的三大角色

在 Seata 的架构中，一共有三个角色：

TC (Transaction Coordinator) - 事务协调者

维护全局和分支事务的状态，驱动全局事务提交或回滚。

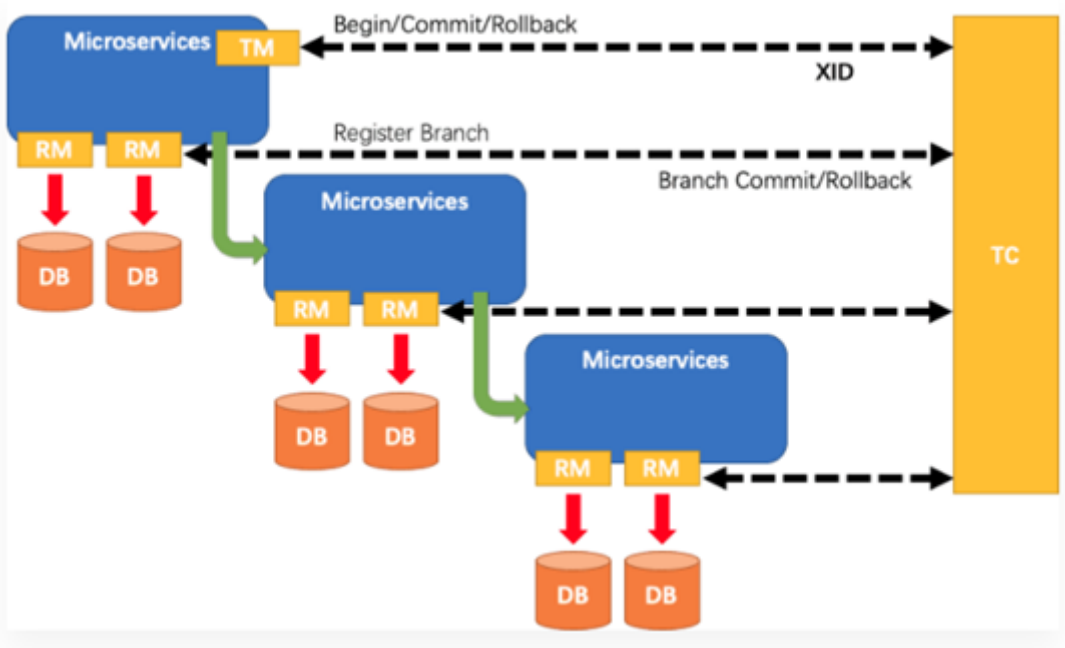
TM (Transaction Manager) - 事务管理器

定义全局事务的范围：开始全局事务、提交或回滚全局事务。

RM (Resource Manager) - 资源管理器

管理分支事务处理的资源，与TC交谈以注册分支事务和报告分支事务的状态，并驱动分支事务提交或回滚。

其中，TC 为单独部署的 Server 服务端，TM 和 RM 为嵌入到应用中的 Client 客户端。



快速开始：

Seata Server（TC）环境搭建

https://seata.io/zh-cn/docs/ops/deploy-guide-beginner.html

Server端存储模式（store.mode）支持三种：

l file：单机模式，全局事务会话信息内存中读写并持久化本地文件root.data，性能较高

l db：高可用模式，全局事务会话信息通过db共享，相应性能差些

l redis：Seata-Server 1.3及以上版本支持,性能较高,存在事务信息丢失风险,请提前配置适合当前场景的redis持久化配置

资源目录：https://github.com/seata/seata/tree/1.4.0/script

l client

存放client端sql脚本，参数配置

l config-center

各个配置中心参数导入脚本，config.txt(包含server和client，原名nacos-config.txt)为通用参数文件

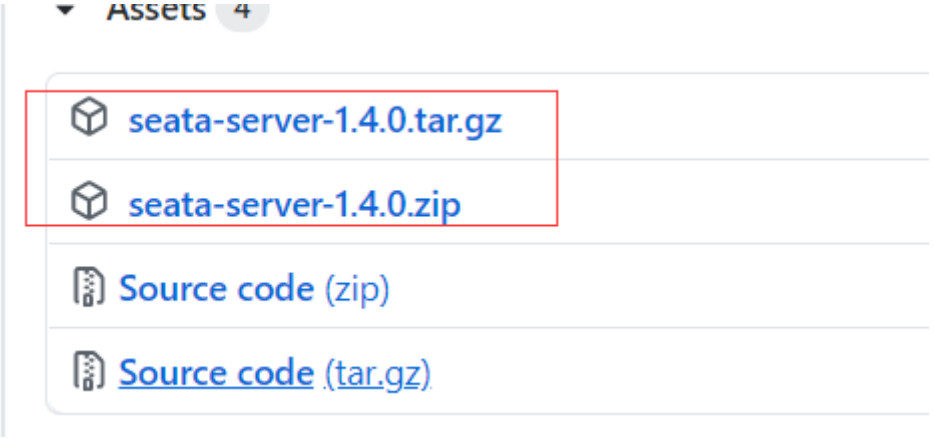
l server

server端数据库脚本及各个容器配置

db存储模式+Nacos(注册&配置中心)部署

步骤一：下载安装包

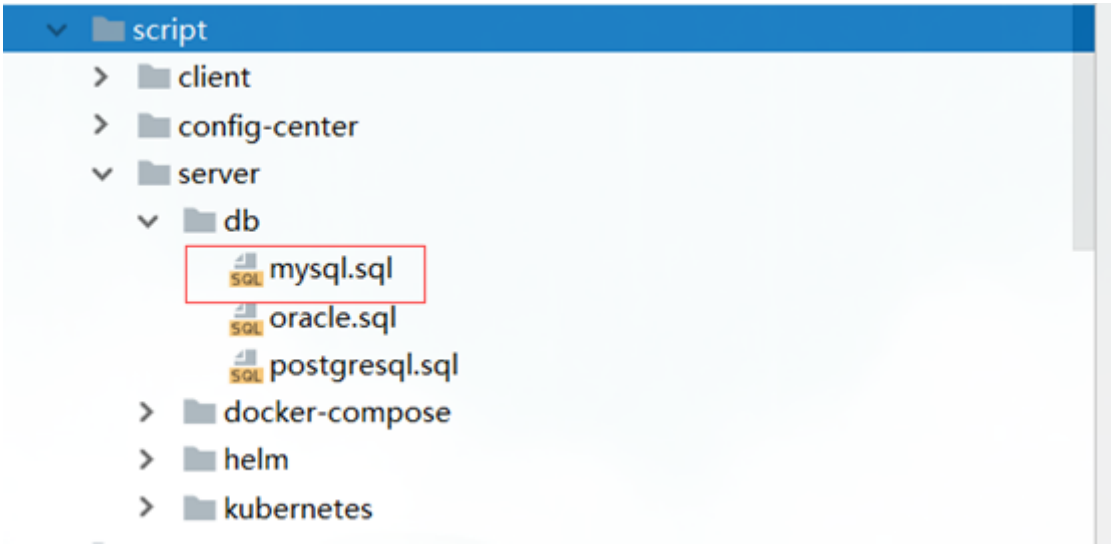
https://github.com/seata/seata/releases



步骤二：建表(仅db模式)

全局事务会话信息由3块内容构成，全局事务-->分支事务-->全局锁，对应表global_table、branch_table、lock_table

创建数据库seata，执行sql脚本，文件在script/server/db/mysql.sql（seata源码）中



步骤三：修改store.mode

启动包: seata-->conf-->file.conf，修改store.mode="db"

源码: 根目录-->seata-server-->resources-->file.conf，修改store.mode="db"

```
store {
  ## store mode: file、db、redis
    mode = "db"
```

步骤四：修改数据库连接

启动包: seata-->conf-->file.conf，修改store.db相关属性。

源码: 根目录-->seata-server-->resources-->file.conf，修改store.db相关属性。

```
  ## database store property
  db {
    ## the implement of javax.sql.DataSource, such as
DruidDataSource(druid)/BasicDataSource(dbcp)/HikariD
(hikari) etc.
    datasource = "druid"
    ## mysql/oracle/postgresql/h2/oceanbase etc.
    dbType = "mysql"
    driverClassName = "com.mysql.jdbc.Driver"
    url = "jdbc:mysql://127.0.0.1:3306/seata"
    user = "root"
    password = "root"
    minConn = 5
    maxConn = 100
```

步骤五：配置Nacos注册中心

将Seata Server注册到Nacos，修改conf目录下的registry.conf配置

```
registry {
  # file 、nacos 、eureka、redis、zk、consu
  type = "nacos"
  loadBalance = "RandomLoadBalance"
  loadBalanceVirtualNodes = 10

  nacos {
    application = "seata-server"
    serverAddr = "127.0.0.1:8848"
    group = "SEATA_GROUP"
    namespace = ""
    cluster = "default"
    username = ""
    password = ""
  }
```

然后启动注册中心Nacos Server

#进入Nacos安装目录，linux单机启动 bin/startup.sh -m standalone # windows单机启动 bin/startup.bat

步骤六：配置Nacos配置中心

```
config {
  # file、nacos 、apollo、zk、consul、etcd3
  type = "nacos"

  nacos {
    serverAddr = "127.0.0.1:8848"
    namespace = ""
    group = "SEATA_GROUP"
    username = ""
    password = ""
  }
```

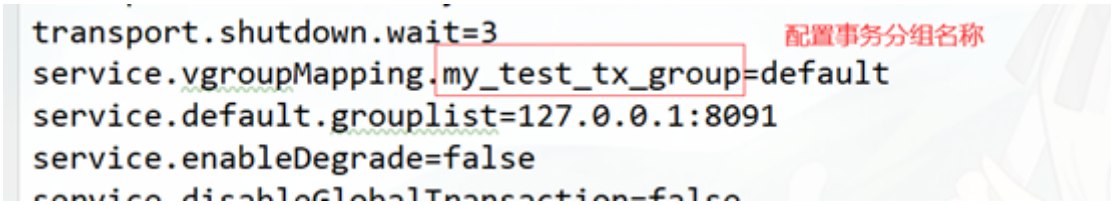
注意：如果配置了seata server使用nacos作为配置中心，则配置信息会从nacos读取，file.conf可以不用配置。 客户端配置registry.conf使用nacos时也要注意group要和

seata server中的group一致，默认group是"DEFAULT_GROUP"

获取/seata/script/config-center/config.txt，修改配置信息

配置事务分组， 要与客户端配置的事务分组一致

（客户端properties配置：spring.cloud.alibaba.seata.tx-service-group=my_test_tx_group）



配置参数同步到Nacos

shell:

sh ${SEATAPATH}/script/config-center/nacos/nacos-config.sh -h localhost -p 8848 -g SEATA_GROUP

```
MINGW64:/d/workspace/seata/script/config-center/nacos
Set server.maxCommitRetryTimeout=-1 successfully
Set server.maxRollbackRetryTimeout=-1 successfully
Set server.rollbackRetryTimeoutUnlockEnable=false successfully
Set client.undo.dataValidation=true successfully
Set client.undo.logSerialization=jackson successfully
Set client.undo.onlyCareUpdateColumns=true successfully
Set server.undo.logSaveDays=7 successfully
Set server.undo.logDeletePeriod=86400000 successfully
Set client.undo.logTable=undo_log successfully
Set client.log.exceptionRate=100 successfully
Set transport.serialization=seata successfully
Set transport.compressor=none successfully
Set metrics.enabled=false successfully
Set metrics.registryType=compact successfully
Set metrics.exporterList=prometheus successfully
Set metrics.exporterPrometheusPort=9898 successfully
=========================================================================
 Complete initialization parameters,  total-count:80 ,  failure-count:0
=========================================================================
 Init nacos config finished, please start seata-server.
```

步骤七：启动Seata Server

l 源码启动: 执行server模块下io.seata.server.Server.java的main方法

l 命令启动: bin/seata-server.sh -h 127.0.0.1 -p 8091 -m db -n 1 -e test

支持的启动参数

参数	全写	作用	备注
-h	--host	指定在注册中心注册的 IP	不指定时获取当前的 IP，外部访问部署在云环境和容器中的 server 建议指定
-p	--port	指定 server 启动的端口	默认为 8091
-m	--storeMode	事务日志存储方式	支持 `file` , `db` , `redis` ，默认为 `file` 注:redis需seata-server 1.3版本及以上
-n	--serverNode	用于指定seata-server节点ID	如 `1` , `2` , `3` ..., 默认为 `1`
-e	--seataEnv	指定 seata-server 运行环境	如 `dev` , `test` 等, 服务启动时会使用 `registry-dev.conf` 这样的配置

启动Seata Server

bin/seata-server.sh

启动成功，默认端口8091

```
2021-01-05 16:22:54.727  INFO --- [           main] io.seata.config.FileConfiguration        : T
he configuration file used is registry.conf
2021-01-05 16:22:54.754  INFO --- [           main] io.seata.config.FileConfiguration        : T
he configuration file used is file.conf
2021-01-05 16:22:55.281  INFO --- [           main] com.alibaba.druid.pool.DruidDataSource   : {
dataSource-1} inited
2021-01-05 16:22:55.422  INFO --- [           main] i.s.core.rpc.netty.NettyServerBootstrap  : S
erver started, listen port: 8091
```

Seata Client快速开始

来听直播，课上讲解，课上干货很多，这儿先给大家讲讲AT模式（monkey老师课上重点讲）

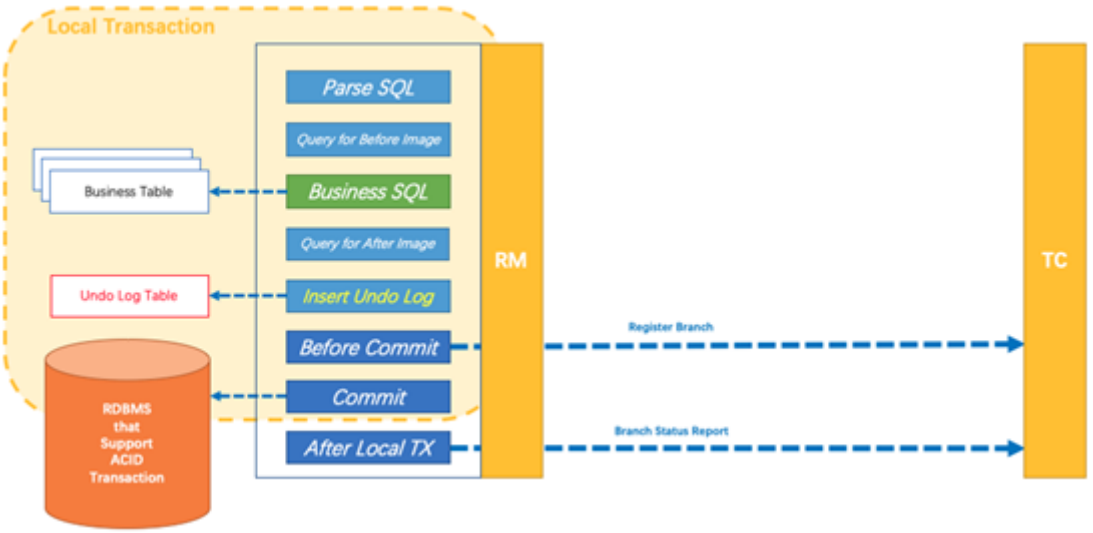
AT模式：

AT模式的核心是对业务无侵入，是一种改进后的两阶段提交，其设计思路如图

第一阶段

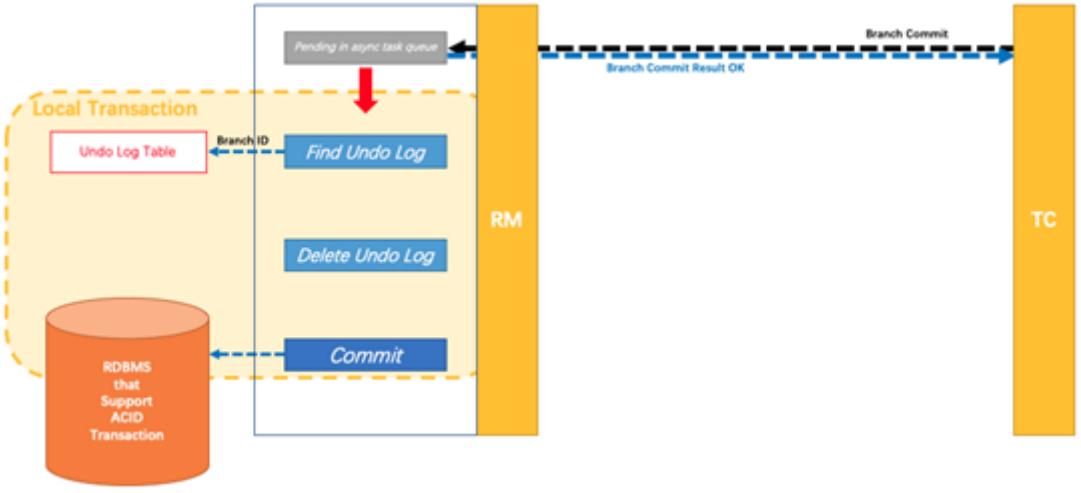
业务数据和回滚日志记录在同一个本地事务中提交，释放本地锁和连接资源。核心在于对业务sql进行解析，转换成undolog，并同时入库，这是怎么做的呢？先抛出一个概念DataSourceProxy代理数据源，通过名字大家大概也能基本猜到是什么个操作，后面做具体分析

参考官方文档： https://seata.io/zh-cn/docs/dev/mode/at-mode.html

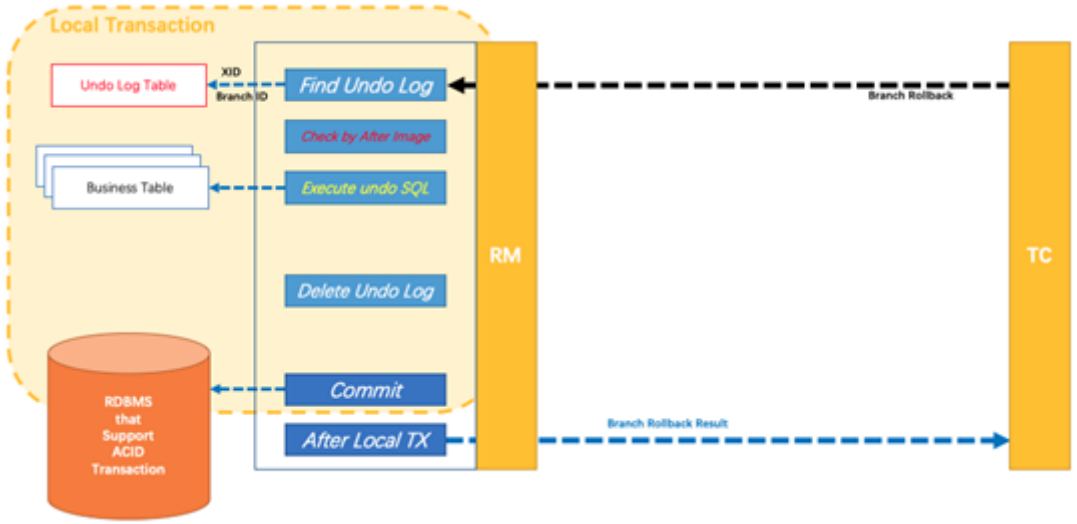


第二阶段

分布式事务操作成功，则TC通知RM异步删除undolog



分布式事务操作失败，TM向TC发送回滚请求，RM 收到协调器TC发来的回滚请求，通过 XID 和 Branch ID 找到相应的回滚日志记录，通过回滚记录生成反向的更新 SQL 并执行，以完成分支的回滚。



整体执行流程