中间件技术
Middleware Technology

第九章　事务处理中间件

赖永炫　博士
厦门大学　软件学院

大纲

- 事务的概念
- 分布式事务
- 两阶段提交协议
- JavaEE JTA简介
- 小结

事务的概念

- 事务是计算机应用中不可或缺的组件模型，它保证了用户操作的原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）和持久性（Durabilily）。

经典示例

- 信用卡转账：将用户 A 账户中的 500 元人民币转移到用户 B 的账户中，其操作流程如下
  - ✓1. 将 A 账户中的金额减少 500
  - ✓2. 将 B 账户中的金额增加 500
- 这两个操作必须保正 ACID 的事务属性：即要么全部成功，要么全部失败；假若没有事务保障，用户的账号金额将可能发生问题。

- 事务(Transaction)是恢复和并发控制的基本单位。事务是保证数据库从一个一致性的状态永久地变成另外一个一致性状态的根本。
  - 例如在关系数据库中，一个事务可以是一条SQL语句，一组SQL语句或整个程序。如果数据库操作在某一步没有执行或出现异常而导致事务失败，则该事务将被回滚(rollback)，取消先前的操作。

事务的特性ACID

- 原子性（atomicity）
  - ✓ 一个事务是一个不可分割的工作单位，事务中包括的诸操作要么都做，要么都不做。
- 一致性（consistency）
  - ✓ 事务必须是使数据库从一个一致性状态变到另一个一致性状态。一致性与原子性是密切相关的。
- 隔离性（isolation）
  - ✓ 一个事务的执行不能被其他事务干扰。并发执行的各个事务之间不能互相干扰。
- 持久性（durability）
  - ✓ 持久性也称永久性（permanence），指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。

数据库或应用的事务

- 事务是访问并可能更新数据库中各种数据项的一个程序执行单元(unit)。
  - ✓ 事务通常由高级数据库操纵语言或编程语言（如SQL，C++或Java）书写的用户程序的执行所引起，并用形如begin transaction和end transaction语句（或函数调用）来界定。
  - ✓ 事务由事务开始(begin transaction)和事务结束(end transaction)之间执行的全体操作组成。
  - ✓ 只有当事务中的所有操作都正常完成了，整个事务才能被提交到数据库，如果有一项操作没有完成，就必须撤消整个事务。

JDBC的事务

- JDBC(Java Data Base Connectivity)是Java与数据库的接口规范，其包含了大部份基本数据操作功能，也包括了事务处理的功能。JDBC的事务是基于连接(connection)进行管理的。
- 在JDBC中是通过Connection对象进行事务的管理，默认是自动提交事务。
  - ✓ 也可以将自动提交关闭，通过commit方法进行提交，rollback方法进行回滚。如果不提交，则数据不会真正的插入到数据库中。

Connection 对象

- JDBC 事务是用 Connection 对象控制的。Connection 接口(java.sql.Connection)提供了两种事务模式：自动提交和手工提交。
- 事务操作缺省是自动提交。
  - ✓ 一条对数据库的更新表达式代表一项事务操作，操作成功后，系统将自动调用commit()来提交，否则将调用rollback()来回滚。

- ✓ java.sql.Connection 提供了以下控制事务的方法：

- public void setAutoCommit(boolean)
- public boolean getAutoCommit()
- public void commit()
- public void rollback()

```
try {
                    conn =DriverManager.getConnection(
      "jdbc:oracle:thin:@host:1521:SID","username","userpwd");
      conn.setAutoCommit(false);//禁止自动提交，设置回滚点
      stmt = conn.createStatement();
      stmt.executeUpdate("alter table …"); //数据库更新操作1
      stmt.executeUpdate("insert into table …"); //数据库更新操作2
      conn.commit(); //事务提交
}catch(Exception e) {
        e.printStackTrace();
      try {
          conn.rollback(); //操作不成功则回滚
            }catch(Exception e) {
              e.printStackTrace();
            }
          }
```

分布式事务处理

- 分布式事务是指事务的参与者、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。
-
- 分布式事务处理 (TP) 系统旨在协助在分布式环境中跨异类的事务识别资源的事务。

分布式事务处理 系统

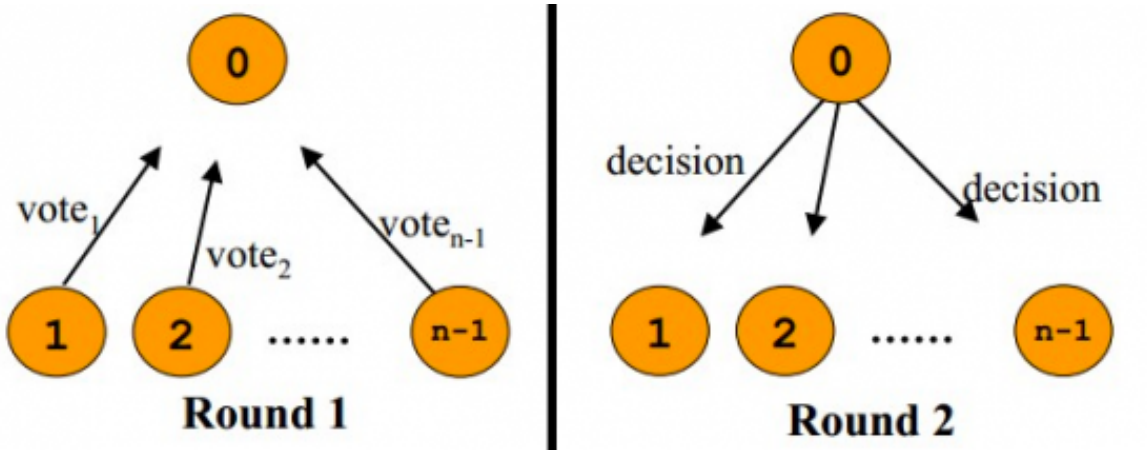
- 在分布式 TP 系统的支持下，应用程序可以将不同的活动合并为一个事务性单元，这些活动包括从“消息队列”队列检索消息、将消息存储在 Microsoft SQL Server 数据库中、将所有现有的消息引用从 Oracle Server 数据库中移除等等。
- 因为分布式事务跨多个数据库资源，故强制 ACID 属性维护所有资源上的数据一致性是很重要的。

两阶段提交

- 在分布式系统中，事务往往包含有多个参与者的活动，单个参与者上的活动是能够保证原子性的，而多个参与者之间原子性的保证则需要通过两阶段提交来实现，两阶段提交是分布式事务实现的关键。

两阶段提交

- 在分布式事务两阶段提交协议中，有一个主事务管理器负责充当分布式事务协调器的角色。事务协调器负责整个事务，并使之与网络中的其他事务管理器（参与者）协同工作
- 协调者可以看做成事务的发起者，同时也是事务的一个参与者。对于一个分布式事务来说，一个事务是涉及到多个参与者的。



第1阶段

- 首先，协调者在自身节点的日志中写入一条的日志记录，然后所有参与者发送消息prepare T，询问这些参与者（包括自身），是否能够提交这个事务；
- 参与者在接受到这个prepare T 消息以后，会根据自身的情况，进行事务的预处理：
  - ✓ 如果参与者能够提交该事务，则会将日志写入磁盘，并返回给协调者一个ready T信息，同时自身进入预提交状态状态；
  - ✓ 如果不能提交该事务，则记录日志，并返回一个not commit T信息给协调者，同时撤销在自身上所做的数据库改；参与者能够推迟发送响应的时间，但最终还是需要发送的。

第2阶段

- 协调者会收集所有参与者的意见，如果收到参与者发来的not commit T信息，则标识着该事务不能提交，协调者会将Abort T 记录到日志中，并向所有参与者发送一个Abort T 信息，让所有参与者撤销在自身上所有的预操作；
- 如果协调者收到所有参与者发来prepare T信息，那么协调者会将Commit T日志写入磁盘，并向所有参与者发送一个Commit T信息，提交该事务。

- 如果协调者迟迟未收到某个参与者发来的信息，则认为该参与者发送了一个VOTE_ABORT信息，从而取消该事务的执行。

第2阶段

- 参与者接收到协调者发来的Abort T信息以后，参与者会终止提交，并将Abort T 记录到日志中；

- 如果参与者收到的是Commit T信息，则会将事务进行提交，并写入记录

正常运转的情况

- 一般情况下，两阶段提交机制都能较好的运行，当在事务进行过程中，有参与者宕机时，它重启以后，可以通过询问其他参与者或者协调者，从而知道这个事务到底提交了没有。

- 前提都是各个参与者在进行每一步操作时，都会事先写入日志。

糟糕的情况

- 如果参与者收不到协调者的Commit 或Abort指令，参与者将处于"状态未知"阶段，参与者将不知道要如何操作。
  - ✓ 比如,如果所有的参与者完成第1阶段的回复后（可能全部yes，可能全部no，可能部分yes部分no），如果协调者在这个时候宕机了,那么所有的参与者将无所适从。可能需要数据库管理员的介入，防止数据库进入一个不一致的状态。

支持两阶段提交的协议

- 为了实现分布式事务，必须使用一种协议在分布式事务的各个参与者之间传递事务上下文信息，IIOP（Internet Inter-ORB Protocol，互联网内部对象请求代理协议）便是这种协议。
  - ✓ 不同开发商开发的事务参与者必须支持一种标准协议，才能实现分布式的事务。
  - ✓ 两阶段提交保证了分布式事务的原子性，这些子事务要么都做，要么都不做。
- 数据库的一致性是由数据库的完整性约束实现的，持久性则是通过 commit日志来实现的，不是由两阶段提交来保证的。

EJB事务体系结构

- Java事务API（JTA：Java Transaction API）和它的同胞Java事务服务（JTS：Java Transaction Service），为Java EE平台提供了分布式事务服务（distributed transaction）。
- JTS也定义了一套规范，它约定了各个程序角色之间如何传递事务上下文，它源自CORBA的OTS规范，基于IIOP（一种软件交互协议）。

EJB事务体系结构

- JTA约定的这些角色要进行事务上下文的交互，JTS约定了应该怎样去进行交互。

- EJB有两种使用事务的方式。
  - ✓通过容器管理的事务，叫CMT，
  - ✓通过bean管理的事务，叫BMT。

JTA的概念

- JTA（Java Transaction API）是一种高层的、与实现无关的、与协议无关的API，应用程序和应用服务器可以使用JTA实现事务管理。
- JTA允许应用程序执行分布式事务处理，在两个或多个网络计算机资源上访问并且更新数据。
- JTA为 Java EE 平台提供了分布式事务服务。

JTA的概念

- 一般JTA事务都用于EJB中，主要用于分布式的多个数据源的事务控制。
  - ✓ 一般的应用服务器（如Weblogic、JBoss、Websphere等服务器）都有自己的事务管理器用来管理JTA事务。
- JTA也是用于管理事务的一套API，与JDBC相比，JTA主要用于管理分布式多个数据源的事务操作，而JDBC主要用于管理单个数据源的事务操作。

- 在JTA中，一个分布式事务包括一个事务管理器（transaction manager）和一个或多个资源管理器(resource manager)。
- 资源管理器是任意类型的支持XA协议的持久化数据存储，事务管理器承担着所有事务参与单元的协调与控制。

XA协议

- XA协议由Tuxedo首先提出的，并交给X/Open组织，作为资源管理器（数据库）与事务管理器的接口标准。
- 目前，Oracle、Informix、DB2和Sybase等各大数据库厂家都提供对XA的支持。

- XA协议采用两阶段提交方式来管理分布式事务。XA接口提供资源管理器与事务管理器之间进行通信的标准接口。

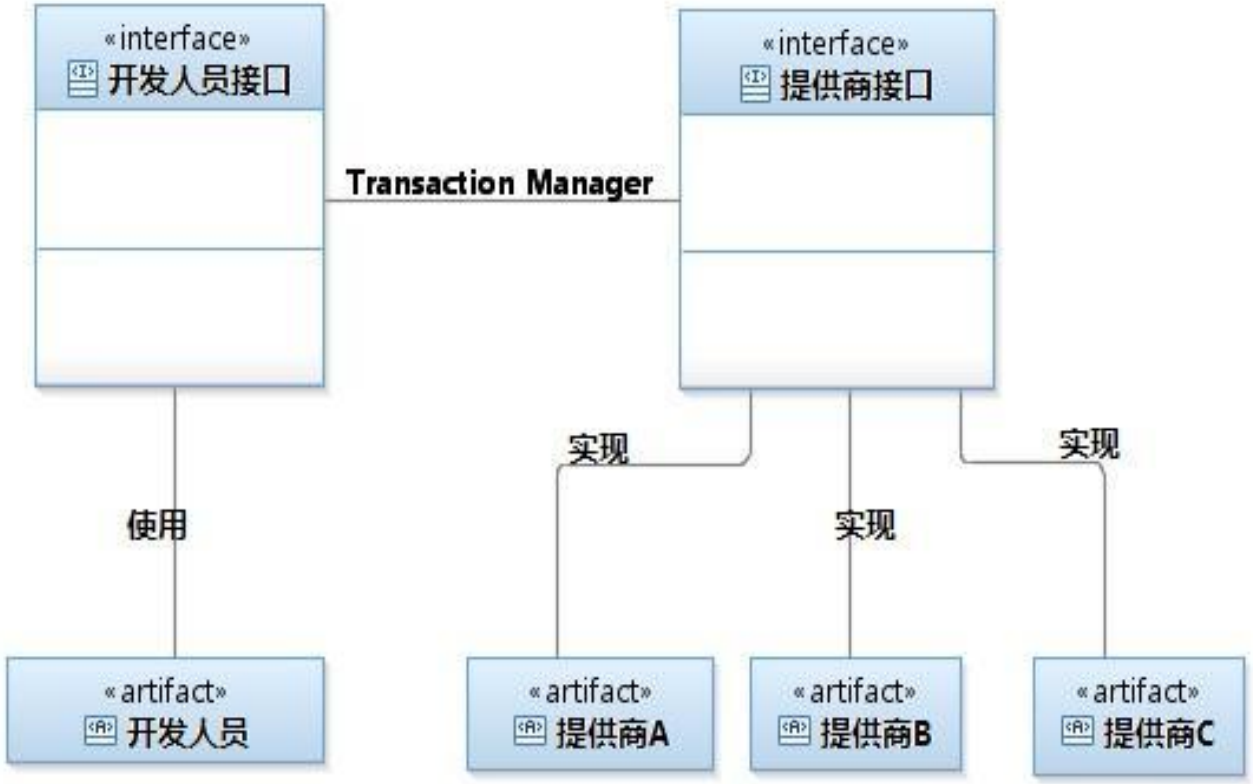
- JTA 事务有效的屏蔽了底层事务资源，使应用可以以透明的方式渗入到事务处理中。

- 但是与本地事务相比，XA 协议的系统开销大，在系统开发过程中应慎重考虑是否确实需要分布式事务。
- 若确实需要分布式事务以协调多个事务资源，则应实现和配置所支持 XA 协议的事务资源，如 JMS、JDBC 数据库连接池等。

JTA的实现架构

- JTA 的事务管理器和资源管理器理解为两个方面：
  - ✓面向开发人员的使用接口（事务管理器）
  - ✓面向服务提供商的实现接口（资源管理器）。

- 开发人员通过使用接口在信息系统中实现分布式事务；而实现接口则用来规范提供商（如数据库连接提供商）所提供的事务服务，它约定了事务的资源管理功能，使得 JTA 可以在异构事务资源之间执行协同沟通。



开发人员使用开发人员接口，实现应用程序对全局事务的支持；各提供商（数据库，JMS 等）依据提供商接口的规范提供事务资源管理功能；事务管理器（TransactionManager）将应用对分布式事务的使用映射到实际的事务资源并在事务资源间进行协调与控制。

面向开发人员的使用接口

- begin()- 开始一个分布式事务，（在后台 TransactionManager 会创建一个 Transaction 事务对象并把此对象通过 ThreadLocale 关联到当前线程上)
- commit()- 提交事务（在后台 TransactionManager 会从当前线程下取出事务对象并把此对象所代表的事务提交）
- rollback()- 回滚事务（在后台 TransactionManager 会从当前线程下取出事务对象并把此对象所代表的事务回滚）
- getStatus()- 返回关联到当前线程的分布式事务的状态 (Status 对象里边定义了所有的事务状态，感兴趣的读者可以参考 API 文档)
- setRollbackOnly()- 标识关联到当前线程的分布式事务将被回滚

面向提供商的实现接口

- commit()- 协调不同的事务资源共同完成事务的提交
- rollback()- 协调不同的事务资源共同完成事务的回滚
- setRollbackOnly()- 标识关联到当前线程的分布式事务将被回滚
- getStatus()- 返回关联到当前线程的分布式事务的状态
- enListResource(XAResource xaRes, int flag)- 将事务资源加入到当前的事务中（在上述示例中，在对数据库 A 操作时 其所代表的事务资源将被关联到当前事务中，同样，在对数据库 B 操作时其所代表的事务资源也将被关联到当前事务中）
- delistResourc(XAResource xaRes, int flag)- 将事务资源从当前事务中删除
- registerSynchronization(Synchronization sync)- 回调接口，Hibernate 等 ORM 工具都有自己的事务控制机制来保证事务，但同时它们还需要一种回调机制以便在事务完成时得到通知从而触发一些处理工作，如清除缓存等。通过此接口将回调程序注入到事务中，当事务成功提交后，回调程序将被激活。

实现框架

- JTA的实现框架有GeronimoTM/Jencks、SimpleJTA、 Atomikos、JOTM以及JBossTS等。

本章小结

- 事务的概念
- 分布式事务
- 两阶段提交协议
- JavaEE JTA简介
- 小结