文章编号:1000-5862(2019)06-0649-06

PAR 平台中并发分布式事务处理机制及其应用研究

刘震伟[1,2],薛锦云[2*],夏 鲸[1,2],汪 雄[1,2]
(1. 江西师范大学计算机信息工程学院,江西 南昌 3300222;2. 江西师范大学国家网络化支撑软件国际科技合作基地,江西 南昌 330022)

摘要:随着大数据、云计算和云服务等新技术的兴起,并发分布式计算作为这些新技术的基础,扮演着越来越重要的角色. 在分布式计算中,数据的一致性难以得到保证,而事务处理技术能够有效解决该问题. 为了提升 PAR 平台在高可靠应用程序的使用范围,在其建模语言 Apla 中融入了并发分布式事务处理机制,使得 Apla 语言不仅支持分布式数据库事务,还支持分布式非数据库事务. 该文对 Apla 语言并发分布式事务处理机制进行了深入地研究,并将其应用在学生管理系统、在线购物系统等实际应用场景中.

关键词:PAR 方法; PAR 平台; 事务处理; 分布式事务
中图分类号:TP 311 文献标志码:A DOI: 10.16357/j.cnki.issn1000-5862.2019.06.15

0 引言

随着大数据、云计算和云服务等新技术的兴起,分布式计算作为这些新技术的基础研究,越来越受到重视. 并发分布式编程面临的巨大挑战之一是数据的一致性难以得到保证[1]. 事务的概念最早由 James Gray 提出并成功地应用于数据库系统中,其特点是具有 ACID 特性[2],这些特性能有效地保证数据库系统中数据的一致性. 如今,事务的概念不再局限于数据库技术,已被应用于分布式计算等技术中[3].

由于并发分布式事务处理的复杂性,在 Java 语言中编写并发分布式应用程序,需要深入地掌握许多底层的实现技术,这无疑增加了开发人员的学习成本,提高了并发分布式应用的开发门槛. 此外,软件开发者使用 Java 等语言编写并发分布式事务处理程序,需要编写大量的底层实现代码,而这往往是重复性劳动且较为复杂,导致软件开发的效率低、程序质量难以保证等问题.

薛锦云等[4]研制成功的 PAR 方法和 PAR 平台(简称 PAR)已融入并发分布式事务处理机制,这一创新性研究成果进一步提升了 PAR 在高可靠性软件开发中的应用范围. 世界著名计算机科学家 Misra 教授也非常希望在他提出的建模语言 Orc 中能实现事务处理机制,但至今尚未成功,这足以说明该原创性成果的重要性、复杂性.

PAR 支持软件开发的全过程,是形式化软件开发中的思想、技术、语言和工具的简称,它由自定义算法设计语言 Radl、抽象程序建模语言 Apla、统一的算法设计方法、一系列的程序自动转换工具等 4 个部分组成[5-7]. Apla 是 PAR 方法中用于从抽象程序模型到可执行程序模型的建模语言,提出该语言的目的是为了更好地实现功能抽象、数据抽象. 使用 Apla 语言编写的程序非常易于形式化推导和验证,而且可由 Apla 到 Java、C++、C#等程序自动转换系统自动地转换成等价可执行的目标语言程序,大幅度提高了应用程序的可靠性和研发效率. 大量实践表明,PAR 不仅适用于开发小型的算法程序,而且适用于开发大型复杂的算法程序[8]. Apla 语言最新融入的并发分布式事务处理机制,是在原关系数据库处理机制、多媒体数据库处理机制以及本地事务处理机制基础上进一步扩展的,使得 Apla 语言的事务处理功能更加强大、完善,能够满足更多应用场景的开发需要. 最新添加的并发分布式事务处理机制和原有的本地事务处理机制自然融合,并对分布式

收稿日期:2019-04-17
基金项目:国家自然科学基金重大国际合作(61020106009),国家自然科学基金面上课题(61272075,61472167),国家自然科学基金(61462041),江西省自然科学基金(20171BAB202008)和江西省教育厅科学技术研究课题(160329)资助项目.
通信作者:薛锦云(1947-),男,江苏海门人,教授,博士生导师,主要从事软件形式化和自动化的研究. E-mail: jinyun@vip.sina.com

事务处理进行扩展,使得 Apla 语言中的分布式事务处理机制能够处理包括分布式数据库事务在内的广义分布式事务.

1 事务处理概述

事务具有 A(Atomicity)、C(Consistency)、I(Isolation)、D(Durability)特性,这 4 种特性使得事务中的多个操作作为一个整体,这些操作要么全部执行、要么全部不执行. 不同事务对同一数据进行操作互不影响,因此可以有效地保证数据的一致性. 事务的类型可以分为本地事务和分布式事务[9-10].

1.1 本地事务

Java 语言的 JDBC 事务专门用作处理本地事务[11-12],JDBC 事务相对比较简单,其并发控制、事务恢复均由数据库厂商实现. 用户仅需要在 begin、commit/rollback 语句块中自定义事务的相关操作即可,事务的处理交给资源管理器实现,目前主流的关系型数据库产品均已实现资源管理器. 本地事务的执行过程如图 1 所示.

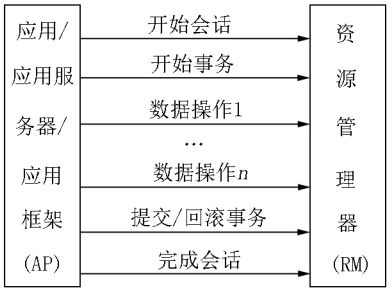


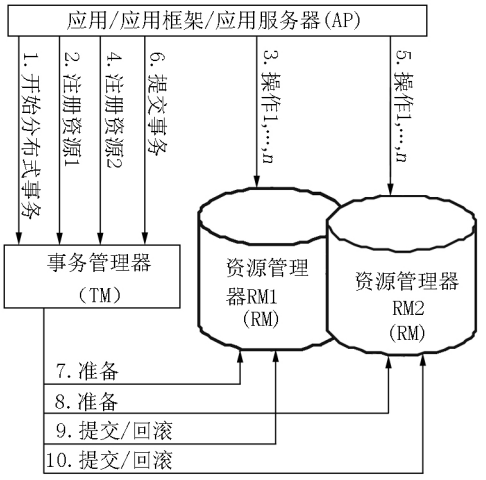
图 1 本地事务执行过程

本地事务的局限性在于资源管理器中的数据必须来源于同一个数据源,换言之,本地事务只能对单个数据库进行操作. 然而,随着互联网的发展,应用系统的数据规模越来越庞大. 为了避免单个数据库的性能达到瓶颈,往往需要根据业务将数据保存在多个数据库中. 显然,本地事务无法解决多个数据源的数据一致性问题,因此对于需要访问多个数据源的应用程序,就要使用分布式事务处理技术.

1.2 分布式事务

分布式事务可以对多个数据源的数据进行操作,较本地事务而言更为复杂. 为了保证多个数据源数据的一致性,X/Open 组织定义了分布式事务处理 XA 规范,该规范规定了分布式事务处理模型[13-20],即 DTP(Distributed Transaction Processing Reference Model),DTP 模型包括应用程序(AP)、事务管理器(TM)、资源管理器(RM)3 个部分. 在 DTP 模型中通过 TM 负责统一的事务管理,RM 负责数据源数据的连接,AP 用作用户自定义事务的处理逻辑.

分布式事务处理的过程分为以下几步:(i)开始分布式事务;(ii)注册资源;(iii)执行事务操作;(iv)提交事务;(v)根据(ii)阶段提交协议,发起提交/回滚操作. 分布式事务处理过程如图 2 所示.



2→3、4→5 的执行过程具有先后顺序,其它过程不存在先后顺序.

图 2 分布式事务执行过程

2 Apla 语言分布式事务处理机制

Apla 语言在已经支持本地事务处理的基础上,增加了对分布式事务处理的支持. Apla 语言的分布式事务处理机制遵守 JTA 规范,并使用了开源的分布式事务管理器 Atomikos.

2.1 分布式事务的并发控制

当多个事务并发执行时,为了保证其执行结果的正确性,需要采用合理的并发控制机制,使它们的执行效果是串行等价的. 互斥锁是实现事务串行化的一种简单办法,其并发控制机制是:任何一个事务在对数据对象进行操作前,都需要为数据对象加上互斥锁;若所访问的数据对象已经被其他事务加了互斥锁,则事务处于挂起状态,直到数据对象的互斥锁被释放.

使用互斥锁实现事务并发控制,虽然较为简单,但存在的问题是:锁的粒度性过大,使得事务的并发处理性能降低. 表 1 给出了读操作和写操作之间的冲突关系,从表 1 可知并发事务若不执行写操作而只执行读操作,则不会发生冲突. 好的锁机制能够支持多个并发事务同时读取某个对象,或者允许一个事务来写对象.

表 1　读操作和写操作的冲突关系

事务操作	是否冲突
读操作 + 读操作	否
读操作 + 写操作	是
写操作 + 写操作	是

在 Apla 语言的事务并发控制机制中,采用读写锁来降低锁的粒度,提升并发事务的处理性能. 读写锁机制分为读锁和写锁 2 种,它们之间的加锁相容性如表 2 所示.

表 2　读锁和写锁之间的相容性

对象已有锁	新请求锁	
	读锁	写锁
无	√	√
读锁	√	×
写锁	×	×

1) 若事务 T 已经对某个对象加了读锁,则其它并发事务 U 在事务 T 释放锁之前,只能对该对象加读锁;

2) 若事务 T 已经对某个对象加了写锁,则其它并发事务 U 在事务 T 释放锁之前,不能对该对象加读锁或写锁.

2.2　分布式事务的扩展

事务的思想最初被应用于数据库系统中,用以保证数据库数据的一致性. Apla 语言对分布式事务进行了扩展,使其不仅支持分布式数据库事务,还能支持分布式非数据库事务.

JTA(Java Transaction API) 事务,包括事务(Transaction) 、事务管理器(TransactionManager) 、资源管理器(XAResource) 3 个部分,其中 Transaction 允许用户根据业务的需求,自定义事务的开始与结束边界,TransactionManager 用于对用户所提交的多个分布式事务进行管理,XAResource 用于对分布式事务所操作的数据进行统一管理. 分布式数据库事务与分布式非数据库事务的区别就在 XAResource 部分,对于分布式数据库事务,一些主流的数据库系统产品(如 Oracle、SQL Server、MySQL 等) 已经实现了 XAResource,因此用户可以直接使用. 对于分布式非数据库事务,则需要用户自己提供对 XAResource 部分的实现,主要包括资源回滚、提交等操作. 用户在实现 XAResource 时,可以选择使用第 3 方提供的实现,也可以结合业务的需求自定义相应接口的实现. TransactionManager 部分已经在 Apla 构件库中,通过第 3 方开源库 Atomikos 分布式事务管理器来实现.

2.3　分布式事务相关操作

为了简化用户的使用,并且保持 Apla 语言的抽象性. 在 Apla 语言中,通过内置 DistTransaction 类来实现对分布式事务的基本操作,通过内置 TransactionThread 类来处理分布式事务的并发特性.

DistTransaction 类提供了对分布式事务的声明、定义以及对事务的提交、回滚等操作,这些操作对应的 Apla 语法如下:

(i) 创建分布式事务.

var transName: DistTransaction;

该语句用于创建一个名为 transName 的分布式事务对象,在完成分布式事务的创建后,即可定义 transName 的分布式事务相关操作.

(ii) 开始分布式事务.

transName. begin;

该语句用于开启一个分布式事务的定义,在调用该语句后,用户可以定义分布式事务的相关操作.

(iii) 提交分布式事务.

transName. commit;

该语句用于提交用户定义的分布式事务操作,当执行过程中出现异常时,会将异常信息输出到内置的 exception 对象中.

(iv) 回滚分布式事务.

transName. rollback;

该语句用于回滚用户定义的分布式事务操作.

(v) 结束分布式事务.

transName. end;

该语句用于结束分布式事务的定义,在 begin 和 end 语句之间定义的处理操作构成一个分布式事务,该语句在执行结束时会自动释放分布式资源管理器的连接对象等资源.

TransactionThread 类主要用于事务并发执行的多线程调度,这些操作对应到 Apla 中的语法如下:

(i) 创建事务线程对象.

var threadName: TransactionThread;

该语句用于创建一个 TransactionThread 类的对象,在创建该对象后,可以并发执行分布式事务的多线程.

(ii) 执行分布式事务.

threadName. run(方法名/过程名,事务是否只读 [,参数列表…]);

该方法将传入的方法/过程,放入一个事务线程中,根据事务是否只读,自动对事务加读锁/写锁,并对事务线程进行调度执行,其中参数列表为可变参数,用于接收方法/过程对应的参数.

为了对 Apla 语言的分布式事务处理机制进行扩展,使其不仅支持分布式数据库事务,还支持分布式非数据库事务,设置了 executeMethod 方法,其定义如下:

executeMethod(分布式资源管理器的连接对象,

操作名称 [,参数列表…]);
该语句会连接传入的分布式资源管理器,并执行指定的相应操作.

3 Apla 并发分布式事务的应用

Apla 语言中最新融入的并发分布式事务处理机制虽然已经通过了部分测试用例,但还未在实际应用场景中使用,为了扩充 Apla 并发分布式事务处理机制的应用实例,下面分析其在学生管理系统和在线购物系统中的应用.

3.1 学生管理系统

学生管理系统作为一个综合性的信息管理系统,为学生的学习和生活提供了众多的便利. 学生管理系统的功能通常包括学生信息管理、课程信息管理、学生成绩管理、教师信息管理和学生缴费等功能模块. 下面以在线缴费功能为例,通过 Apla 语言分布式事务处理技术分析其实现方法.

(i) 在线缴费事务处理过程.

在学生管理系统中,学生个人信息、课程信息、教师信息和成绩信息等与学生相关的信息存储在 MySQL 数据库的学生(student)数据库中. 学生的银行账户信息(如银行卡号、账户余额、交易记录等信息)则存储在 SqlServer 数据库的银行(bank)数据库中,这 2 个数据库部署在网络的不同节点上. 当用户进行缴费操作时,先检查学生的银行账户是否有足够的金额,若有足够的金额则从账户上扣除缴费金额,并在学校银行账户中增加对应的金额,此过程相当于银行转账操作. 缴费完成后,还需要更新用户的缴费状态. 这几个处理步骤构成一个分布式事务.

(ii) Apla 分布式事务处理代码.

使用 Apla 语言的分布式事务处理机制实现该功能的核心代码为

```
program PayTuitionFee;
var
urls,users,passwords: array [0..1,string];
//创建 3 个事务线程对象,3 个学生同时缴费//
th: TransactionThread;
procedure pay(money: real; account: string);
var
dt: DistTransaction;
//创建一个分布式事务对象//
begin
dt.begin(urls);
distUpdate(urls [0]," Accounts"," money = money - " + money," account = "" + account + "");
distUpdate(urls [0]," Accounts"," money = money + " + money," account = 201500");
distUpdate(cons [1]," Bill"," payStatus = 1"," account = "" + account + "");
if(exception) →dt.rollback;
 I →dt.commit;
fi;
dt.end;
end;
begin
//远程的学生信息数据库的连接信息//
urls [0]: = "jdbc:sqlserver://192.168.1.24:1433;DatabaseName = bank";
users [0]: = "admin";
passwords [0]: = "admin";
//远程的银行数据库的连接信息//
urls [1]: = " jdbc: mysql://192.168.1.25:3306/student ";
users [1]: = "root";
passwords [1]: = "root";
connect(urls,users,passwords);
//3 个事务线程执行分布式事务//
th.run(" pay",false,5000,"201501");
th.run(" pay",false,5000,"201502");
th.run(" pay",false,5000,"201503");
end.
```

(iii) 程序运行结果.

```
"获得写锁 pool-1-thread-1
帐户 201501 的余额为:3 000
分布式事务执行了 [回滚] 操作!
释放写锁 pool-1-thread-1
获得写锁 pool-1-thread-2
帐户 201502 的余额为:5 000
分布式事务执行了 [提交] 操作!
释放写锁 pool-1-thread-2
获得写锁 pool-1-thread-3
帐户 201503 的余额为:8 000
分布式事务执行了 [提交] 操作!
释放写锁 pool-1-thread-3".
```

(iv) 运行结果分析.

账户 201501 的余额为 3 000,少于扣款额 5 000,因此事务执行回滚操作. 账户 201502 的余额为 5 000,能够支付款项,事务成功提交. 账户 201503 的余额为 8 000,大于所要支付的款项,事务提交成功.

3.2　在线购物系统

在线购物系统在电子商务领域中起着举足轻重的作用,其功能通常包括商品浏览、加入购物车、在线下单等. 下面以在线购物下单为例,给出使用 Apla 语言并发分布式事务的实现方法.

(i) 在线下单事务处理过程.

在线购物系统中,商品的库存信息存储在 SQLServer 数据库的仓库(warehouse)数据库中,订单相关的信息存储在 MySQL 数据库的购物(shopping)数据库中,这 2 个数据库部署在网络的不同节点上. 当用户下单时,先检查下单商品的库存是否充足,若库存充足则在 warehouse 数据库中减去对应商品的库存数量,在 shopping 数据库中生成商品的订单信息. 整个处理过程构成一个分布式事务.

(ii) Apla 分布式事务处理代码.

使用 Apla 语言的分布式事务处理机制实现该功能的核心代码为

```
program CommitOrder;
var
urls,users,passwords: array [0..1,string];
//5 个用户同时下单//
th: TransactionThread;
procedure order(comId: integer);
var dt: DistTransaction;
//创建分布式事务对象//
begin
dt.begin(urls);
distUpdate(cons [0],"store","comNum = comNum - 1","comId = " + comId + "4");
distInsert(cons [1],"order","comId = " + comId);
if(exception) →dt.rollback;
 [ →dt.commit;
fi;
dt.end;
end;
begin
//远程的仓库数据库的连接信息//
urls [0]: = "jdbc: sqlserver: //192.168.1.24: 1433; DatabaseName = warehouse";
users [0]: = "admin";
passwords [1]: = "admin";
//远程的购物数据库的连接信息//
urls [1]: = "jdbc: mysql: //192.168.1.25: 3306/shopping";
users [1]: = "root";
passwords [1]: = "root";
connect(urls,users,passwords);
th.run("order",false,1);
//事务线程执行分布式事务//
th.run("order",false,2);
//事务线程执行分布式事务//
th.run("order",false,3);
//事务线程执行分布式事务//
th.run("order",false,4);
//事务线程执行分布式事务//
th.run("order",false,5);
//事务线程执行分布式事务//
end.
```

(iii) 程序运行结果.

```
"获得写锁 pool-1-thread-1
商品 ID: 100 的库存为: 1
下单数量为 2
分布式事务执行了 [回滚] 操作!
释放写锁 pool-1-thread-1
获得写锁 pool-1-thread-2
商吕 ID: 101 的库存为: 0
下单数量为 1
分布式事务执行了 [回滚] 操作!
释放写锁 pool-1-thread-2
获得写锁 pool-1-thread-3
商品 ID: 102 的库存为: 2
下单数量为 3
分布式事务执行了 [回滚] 操作!
释放写锁 pool-1-thread-3
获得写锁 pool-1-thread-4
商品 ID: 103 的库存为: 2
下单数量为 1
分布式事务执行了 [提交] 操作!
释放写锁 pool-1-thread-4
获得写锁 pool-1-thread-5
商品 ID: 104 的库存为: 1
下单数量为 1
分布式事务执行了 [提交] 操作!
释放写锁 pool-1-thread-5".
```

(iv) 运行结果分析.

当商品数量无法满足下单商品的数量时,事务执行回滚操作,不会在 shopping 数据库中产生订单信息. 当商品数量能够满足下单商品的数量时,事务执行提交操作,在 shopping 数据库中产生订单信息.

4 结语

本文首先研究 PAR 平台中建模语言 Apla 的并发分布式事务处理机制,并分析其对分布式非数据库事务的支持,然后利用该语言机制构建学生管理系统、在线购物系统. 通过这 2 个应用实例,展现了利用 Apla 语言编写的分布式事务程序具有结构简洁、开发快捷等优点,体现了 PAR 平台中并发分布式事务处理机制的易用性、实用性.

5 参考文献

[1] 王之元,杨学军,周云. 大规模 MPI 并行计算的可扩展三模冗余容错机制 [J]. 软件学报,2012,23(4):1022-1035.

[2] Gray I, Reuter A. Transaction processing: concepts and techniques [M]. New York: Academic Press, Morgan Kaufmann, 1992.

[3] 陈小芳,丁柯,金蓓弘. 事务处理技术综述 [J]. 计算机科学,2003,30(5):12-16.

[4] 薛锦云,李云清,杨庆红,等. 程序设计方法学 [M]. 北京: 高等教育出版社,2001.

[5] 石海鹤,薛锦云. 基于 PAR 的算法形式化开发 [J]. 计算机学报,2009,32(5):982-991.

[6] 王昌晶,薛锦云. PAR 平台从规约出发的算法推导与自动生成 [J]. 计算机工程与应用,2007,43(2):41-42,59.

[7] 王昌晶. PAR 平台中结构化需求语言研究 [EB/OL]. [2007-10-19]. http://www.docin.com/p-242033622.html.

[8] 李英龙. PAR 方法中关系数据库机制的描述与实现 [D]. 南昌: 江西师范大学,2006.

[9] 罗摩克里希纳,格尔基. 数据库管理系统原理与设计 [M]. 3 版. 周立柱,张志强,李超,等译. 北京: 清华大学出版社,2004: 390-391.

[10] 埃尔玛斯利,纳瓦特赫. 数据库系统基础 [M]. 6 版. 李翔鹰,刘镔,邱海艳,等译. 北京: 清华大学出版社,2011.

[11] Cheung S, Matena V. Java transaction API 1.01 specification [EB/OL]. [2018-10-19]. https://docs.oracle.com/cd/A91202_01/901_doc/java.901/a90211/jdbctran.htm.

[12] Cheung S. Java transaction service 1.0 specification [EB/OL]. [2018-10-19]. https://docs.oracle.com/cd/A91202_01/901_doc/java.901/a90211/jdbctran.htm.

[13] Ozsu M Tamer, Valduriez Patrick. 分布式数据库系统原理 [M]. 3 版. 周立柱,范举,吴昊,等译. 北京: 清华大学出版社,2014: 257-262.

[14] X/Open Company. Distributed transaction processing: reference model [EB/OL]. [2018-10-19]. https://www.researchgate.net/publication/200030484_Distributed_Transaction_Processing_Reference_Model.

[15] Moss J E B. Nested transactions: an approach to reliable distributed computing [M]. Massachusetts: MIT Press, 1985.

[16] Weikum G. Principles and realization strategies of mutil-level transaction management [EB/OL]. [2018-10-19]. https://dl.acm.org/citation.cfm?id=103145.

[17] Pu Calton, Kaiser G E, Hutchinson N. Split-transactions for open-ended activities [EB/OL]. [2018-10-19]. http://www.vldb.org/dblp/db/conf/vldb/PuKH88.html.

[18] Hetor G M, Salem K. Sagas [J]. Acm Sigmod Record, 1987,16(3):249-259.

[19] Bukhres O, Elmagarmid A, Kuhn E. Implementation of the flex transaction model [J]. IEEE Data Engineering, 1993, 16(2):28-32.

[20] 夏鲸. 并发分布式事务处理机制在 PAR 平台中的设计与实现 [D]. 南昌: 江西师范大学,2018.

The Research on Concurrent Distributed Transaction Processing Mechanism in PAR Platform and Its Application

LIU Zhenwei[1,2], XUE Jinyun[2*], XIA Jing[1,2], WANG Xiong[1,2]

(1. College of Computer and Information Engineering, Jiangxi Normal University, Nanchang Jiangxi 330022, China;
2. Networked Supporting Software International S&T Coopertation Base of China, Jiangxi Normal University, Nanchang Jiangxi 330022, China)

Abstract: With the emergence of new technologies such as big data, cloud computing, and cloud services, concurrent distributed computing is playing a more and more important role as the foundation of these new technologies. In distributed computing, data consistency is difficult to guarantee, and transaction processing technology can solve this problem effectively. In order to enhance the application of PAR platform in high reliable applications, the concurrent distributed processing mechanism has been integrated into its modeling language Apla, which Apla language supports distributed database transaction and non-distrubuted database transaction. The mechanism is deeply studied and used in constructing student management system and online shopping system.

Key words: PAR method; PAR platform; transaction processing; distributed transaction (责任编辑:冉小晓)