第33卷 第22期 计 算 机 工 程 2007年11月
Vol.33 No.22 Computer Engineering November 2007
·软件技术与数据库· 文章编号：1000—3428(2007)22—0104—03 文献标识码：A 中图分类号：TP311

工作流系统中应用 Agent 改进的高级事务模型

丁月华[1]，李冠宇[1,2]，项 聪[1]，丁伟华[1]
（1. 大连海事大学计算机科学与技术学院，大连 116026；2. 大连理工大学管理学院，大连 116024）

摘 要：工作流高级事务模型因结构过于复杂，或者实现起来过于困难，很难应用到实际的工作流系统中。该文以Saga模型为基础，结合成熟的对象事务服务技术，并应用多agent技术对结合OTS模型的Saga模型进行优化，提出并实现了一种切实可行的高级事务模型Esaga，为Saga模型增加了分布式事务支持，很大程度上弥补了Saga模型的不足。

关键词：高级事务模型；对象事务服务；Saga；agent

Advanced Transaction Model Improved by Agent in Workflow System

DING Yue-hua[1], LI Guan-yu[1,2], XU Cong[1], DING Wei-hua[1]
(1. College of Computer Science and Technology, Dalian Maritime University, Dalian 116026;
2. School of Management, Dalian University of Technology, Dalian 116024)

【Abstract】 Most of advanced transaction models can not be employed in realistic workflow systems, because of their complicated structures or difficulties in implementation. This paper presents Esaga (extended Saga), a new advanced transaction model, which is based on Saga model and mixed with OTS technology and improved by multi-agent. Esaga model, which supports the distributed environment, offset the shortage of Saga in deep extent.

【Key words】 advanced transaction model; object transaction service(OTS); Saga; agent

企业对于信息系统提出的要求在不断地提高，在新的应用需求环境下，以往的高级事务模型暴露了其自身的局限性[1]。首先，工作流模型一般不具备恢复与失败处理的语义，通常是依靠管理人员的手工参与来实现此功能；其次，工作流系统大多支持并且需要分布式的运行环境，但是以往的高级事务模型在分布式支持上存在着明显的不足。

针对上述问题，本文通过对 Saga 模型的局限性进行分析，引入了对象事务服务 OTS，很大程度上弥补了 Saga 模型的不足，同时也弥补了 OTS 模型不支持嵌套事务的不足。本文对多 Agent 进行研究，将多 Agent 系统(MAS)引入到高级事务模型中提出并实现了一种切实可行的高级事务模型 Esaga，使得 Saga 模型能够被真正应用到实际的系统中。

1 Saga 模型及其局限性

Sagas[2]是由普林斯顿大学的H.Garcia-Mo1ina等人提出的一种模型，用于解决长时间事务问题。Sagas 事务模型将长事务分成一系列子事务集合T (T_1,T_2, …,T_n) 和一个相对应的补偿子事务集CT(CT_1,CT_2,…,CTn)，任意一个子事务都具有自己的ACID 特性。当且仅当所有的子事务都进行了提交，一个Saga 事务(在这里，一个长时间的事务被称为一个“Saga”)才提交；如果其中一部分子事务执行失败，必须对已提交的子事务进行补偿。

当然，Saga也存在一些未解决的问题，如其概念至今并没有得到实际系统的实现，另外还有以下问题[3]：

(1)子事务粒度过小。Saga 模型的基本事务单元是传统事务，而且在大多数情况下是一个传统的数据库事务。传统数据库事务理论本身的局限性使得 Saga 模型中的这种基础事务活动的粒度过小。

(2)不支持分布式事务。这一缺陷也是传统事务模型的一个固有缺陷。传统的事务模型大多出现在分布式计算环境普及以前，在当前分布式事务广泛普及的情况下，Saga 模型的底层子事务的应用产生了很大的局限性。

(3)事务边界划分不明确。Saga 模型中对底层子事务的一条明确要求是这些事务彼此之间应该是可见的，也就是应该在逻辑上可以很好地划分开，每一个底层事务都必须能够独立提交，而且结果对于后面的事务应该可见。但是，在工作流系统中，这样的划分很难进行。可能一个活动本身就要求多个原子操作之间保持一定的事务性关系，如果一定要分拆的话，往往很难决定到底哪里是事务的边界。

(4)补偿事务过于复杂。引起这一问题的直接原因在于 Saga 模型的子事务粒度太小， 因为每一个子事务都必须提供一个补偿节点。但是工作流系统中的最小工作单元是活动，根据每个活动来进行补偿是一个非常自然的选择，在逻辑上也比较容易理解。但是因为前 3 个原因，这种自然的选择不得不被打破，所以补偿活动必须降低到每一个原子操作中。

2 OTS 模型的优缺点

对象事务服务(object transaction service, OTS)是 OMG

基金项目：国家自然科学基金资助项目(60172043)；辽宁省自然科学基金资助项目(002055)
作者简介：丁月华(1981－)，女，硕士研究生，主研方向：智能软件；李冠宇，博士、教授；项 聪、丁伟华，硕士研究生
收稿日期：2006-12-25 **E-mail：**dyh@newmail.ftpm.edu.cn

定义的基于 CORBA 的事务处理服务。其框架如图 1 所示。

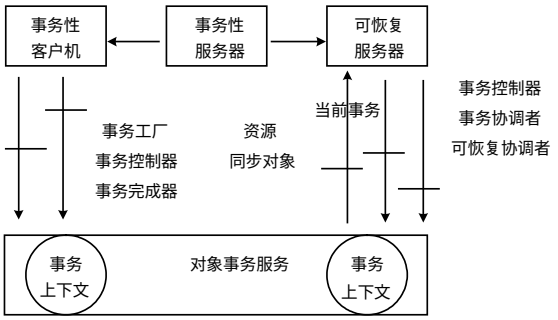

图 1　OTS 结构

它将事务概念引入到分布式对象计算中。能有效地保证分布事务的原子性和永久性，对构建高可靠性应用，特别是要求并发访问共享数据的分布式应用起着关键作用。对象事务服务结合了分布式对象技术和事务管理技术，一方面具有事务管理器的高可靠性，同时具有 CORBA 中互操作性、可扩展性、可维护性等优点，提供了良好的分布式事务处理解决方案。

但是，OTS 技术也有一定的局限性，主要体现在事务模型本身上。OTS 事务模型作为一种扁平事务模型，并不支持嵌套事务。它认为常规的事务没有内部结构(即不存在子事务)，在事务进行过程中，任意一个动作的失败，都要求对整个事务进行回滚。

这种事务模型在集中式系统中可以工作得很好，但是在分布式环境下却存在较大的缺陷。这是因为，在分布式环境下，网络环境的复杂性大大增加了操作出错的几率，网络中任意一个节点的临时故障(甚至只是延迟)都会导致整个事务的回滚。这种情况在一定程度上限制了 OTS 事务在分布式事务中的应用能力，大量的事务处理逻辑，需要应用程序开发人员自己来进行控制，体现在业务逻辑之中。

所以，提供一种能够支持嵌套事务的运行环境有着非常重要的意义。这种支持即可以在 OTS 模型内部实现，也可以在 OTS 模型之上实现。

3　多 Agent 技术在分布式事务处理中的优势

多智能体系统[4]是当今人工智能中的前沿学科，是分布式人工智能研究的一个重要分支，其目标是将大的复杂系统(软硬件系统)建造成小的、彼此相互通信及协调的、易于管理的系统。多智能体的研究涉及智能体的知识、目标、技能、规划以及如何使智能体协调行动解决问题等。

在异构数据集成领域，多 agent 组成的系统不仅具有良好的适用性、可维护性，而且能为实现信息共享、协调和冲突管理创造条件。

在分布式事务处理模型中引入多agent 具有如下意义[5]：

(1)agent 的自治性使得 agent 可以根据目标、环境等情况，对自己的行为作出规划。尤其是当事务不能正常提交时，agent 可以根据环境情况，自主决定是否可以重试，从而减少失败的次数。

(2)agent 的交互性使得 agent 可以感知环境的变化，并通过自身的行为来改变环境。

(3)agent 的协作性使得 agent 可以相互协作来提高整个系统的并行性。

(4)agent 的可通信性使得某个 agent 可以直接与其他 agent 进行信息交换，避免了所有 agent 都因等待协调者而造成时间的延误。

(5)agent 的移动性使得 agent 可以减轻网络负载。分布式数据库中通常依赖于通信协议，协议在完成给定任务的过程中涉及到多次交互行为，并将其传输到目的地去进行本地交互；另外，远程节点在进行大量数据处理时，这些数据也不需要在网络上传输，而是将 agent 移动过去，在远程节点上直接处理完成相应的事务。

(6)多 agent 的分布式结构使得整个系统具有鲁棒性和可靠性。当系统中一个 agent 发生故障时，整个系统不会崩溃，甚至性能也不会显著下降。

因此，MAS 不仅能解决大规模的复杂问题，同时可以节省开始费用，提高系统性能。综上所述将 MAS 引入到高级事务模型中。

4　应用多 Agent 改进的高级事务模型

针对上述问题，本文提出了扩展的 Saga 模型——Esaga 模型。Esaga 模型的主要思想是通过引入 OTS 事务，引入分布式支持，明确子事务边界的划分，减少补偿活动，同时引入多 agent 进行优化。图 2 和图 3 对比描述了 Esaga 模型与 Saga 模型的异同。

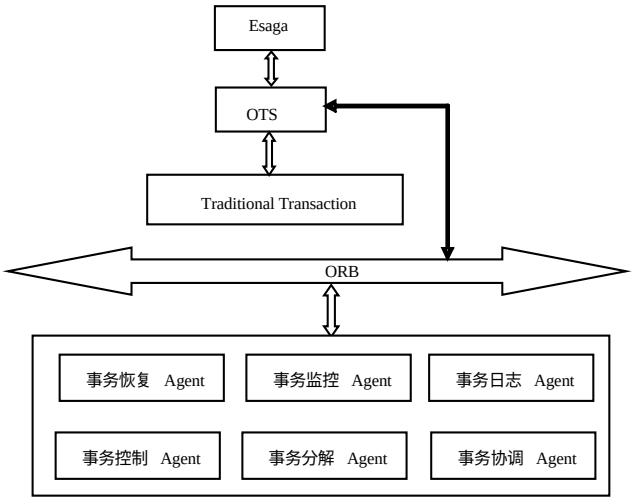

图 2　改进的高级事务模型 Esaga

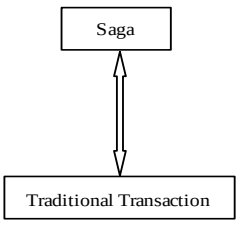

图 3　事务模型 Saga

(1)OTS 事务层的加入，提高了子事务的粒度。因为 OTS 事务本身也是建立在传统事务模型的基础上的，一个 OTS 事务由一组传统事务组成。OTS 中的事务管理器，能够很好地保证这些传统事务作为一个粒度更大的原子事务进行处理，即符合所谓的“ All-or-Nothing ”(全做或者全不做)原则。

(2)OTS 事务模型提供了对分布式事务的良好的支持。通过定义事务上下文以及传递的接口，使得一个事务能够由多个事务管理器中共同协作完成，而且 OTS 接口很好地屏蔽了其中的细节处理，对于客户端而言，处理的仍然是一个原子事务。引入 OTS 事务能够自然地扩展原来的 Saga 模型，使

其增加对分布式的支持，这一扩展对于现代信息系统尤为重要。

(3)OTS 事务的引入缓解了 Saga 模型中的补偿问题。可以将补偿活动提高到针对每一个 OTS 事务，而不是传统事务的层次上来。这将大大减少补偿活动的数目，而且作为工作流系统中的最小复用单位，针对活动的补偿显然更有意义

(4)结合 Agent 技术和按照 Agent 在事务处理中的不同角色分工，事务管理服务具体可以抽象出 6 种不同角色的事务处理 Agent：

1)事务监控 Agent。事务监控 Agent 负责监听 ORB 上的请求创建事务信息，当监听到有请求信息后，试着创建一个有唯一事务标识的新事务，如果在规定的时间内事务创建成功，则启动相应的事务协调 Agent 和事务分解 Agent，然后把新创建的事务加入到事务响应队列中，否则把事务创建失败信息返回给应用程序。

2)事务分解 Agent。事务分解 Agent 负责全局事务的映射工作，判断事务是局部事务还是分布式事务。事务如果是一个分布式事务，事务分解 Agent 则把分布式事务映射到多个分布式结点上去，进而保证分布式事务的正确调度执行。

3)事务协调 Agent。在实现事务两阶段提交协调过程中，协调器要不断地向参加者轮番发送“提交准备”信息、同时要轮番查询参加者的投票表决信息，这样一来就大大地增加了作出提交决定所花的时间，因此两阶段提交协议的性能会受到一定程度的影响。为了提高两阶段提交协议的性能，协调器用移动 Agent 来实现。

4)事务控制 Agent。在异构数据集成系统中，不同模式结构的分布式数据资源，允许多个用户同时访问和使用。因此在同一时刻可能存在多个并行运行的事务，要保证一个对象被并行事务计算和访问时的数据一致性，需要用到事务控制 Agent。事务控制 Agent 主要负责协调并行运行的事务，保证并行事务对数据资源访问时数据的一致性。

5)事务日志 Agent。事务日志 Agent 记录事务执行过程中发生的所有事件，为事务恢复提供依据。事务日志 Agent 把事务开始、事务提交、事务回滚等每一个操作都登记为一条日志记录，存放在日志文件中。日志登记次序严格按照并行事务操作执行的时间次序，同时遵循先写日志文件，再执行操作的规则。

6)事务恢复 Agent。事务恢复 Agent 主要是为了在事务异常情况下采取一定的事务恢复策略来保证事务的原子性。

5 实例分析

在工作流系统中，高级事务模型 Esaga 长事务处理实例如图 4 所示。

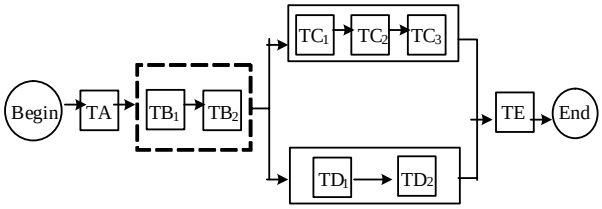

图 4 长事务处理实例

根事务 T 由叶事务 TA，TE 和子事务 TB,TC,TD 组成。假设TA，TB 顺利提交后同时执行TC和TD，而TE需TC与TD同时提交后才能开始执行，但执行TD_2时事务被取消。

开始执行 TA 时，当事务监控 agent 监听到有请求信息后，试着创建一个有唯一事务标识的新事务TA。创建其补偿子事务CTA，并由日志agent记录所有事务执行过程中发生的所有事件，设为A_TA (提交后均须写日志表，下同)，在规定的时间内事务创建成功后，启动相应的事务协调agent 和事务分解agent，然后把新创建的事务加入到事务响应队列中(agent的调度，下同)。执行TB，创建其补偿子事务CTB， 开始执行TB_1，创建其补偿子事务CTB_1，并开始执行TB_2， 创建其补偿子事务CTB_2。

顺利执行后提交事务后，此时因为TB所包含的事务已全部提交，所以创建其补偿子事务CTB替换补偿子事务CTB_1和CTB_2 。此后同时执行TC与TD 。

执行TC，创建其补偿子事务CTC，依次执行TC_1,TC_2，TC_3，同理将分别创建其补偿子事务CTC_1,CTC_2,CTC_3，全部提交后将补偿子事务CTC_1,CTC_2,CTC_3更新为补偿子事务CTC。

执行TD，创建其补偿子事务CTD，开始执行TD1，创建其补偿子事务CTD_1，并开始执行TD_2，创建其补偿子事务CTD_2，此时事务TD_2被取消，TD_2 状态为Running，长事务T对应空间数据库中补偿事务如图 5 所示。

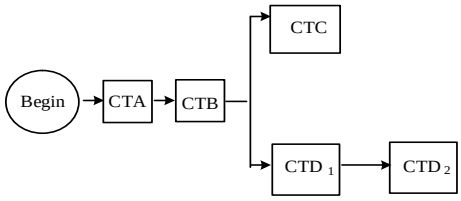

图 5 补偿事务状态

监控Agent监听到有请求信息后，把事务创建失败信息返回给应用程序。TD将停止其正常活动序列中剩余子活动的执行，依次执行补偿子事务CTD_2 ,CTD_1,CTC,CTB,CTA。

6 结束语

实验分析表明，由 Saga 模型和 OTS 模型组成，并由 agent 优化的高级事务模型 Esaga，具备较强的长事务处理能力，为工作流系统中的长事务处理提供有力的技术支持。当然，Esaga 模型还需要进一步完善：当系统有成千上万、甚至更多并发事务请求时，如何均衡、维护和管理事务请求进程是一个必须解决的问题；还包括 agent 的推理算法设计、多 agent 协作和移动 agent 稳定性等等问题。

参考文献

1 Sheth A. On Transactional Workflows[J]. IEEE Data Engineering Bulletin, 1993, 16(2): 37-40.

2 刘传杰. 基于 Saga 的高级事务模型在工作流系统中的应用研究[D]. 西安: 西安电子科技大学, 2004.

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

4 刘发军, 李冠宇. 基于角色的分布式事务处理模型设计[J]. 微机发展, 2005, 15(6): 25-27.

5 史忠植. 智能主体及其应用[M]. 北京: 科学出版社, 2000: 50-79.