基于替代和补偿的嵌套 Web Service 事务模型

党德鹏 姜 雪

(北京师范大学信息科学与技术学院 北京 100875)

摘 要 作为一种崭新的并行计算模式,Web Service 以其平台独立性、松耦合及自治性等优势,在涉及复杂业务流程的事务性服务行业具有极大潜力和需求。考虑到 Web Service 环境的限制和特征,传统事务模型无法适应。从 Web Service 事务系统体系结构出发,分析 Web Service 的特点及 Web Service 事务性应用在开放的 Web 环境下的特性,提出了一个基于替代和补偿的嵌套 Web Service 事务模型,阐述了结合替代和补偿的 Web Service 事务结构和处理,给出了形式化描述。

关键词 Web Service,Web Service 事务,事务模型

中图法分类号 TP311.13 **文献标识码** A

Nested Model of Web Service Transactions Based on Alternative and Compensation

DANG De-peng JIANG Xue

(Department of Computer Science and Technology, Beijing Normal University, Beijing 100875, China)

Abstract As a new parallel computing model, Web Service, which has the advantages of platform independence, loose coupling and autonomy, has great potential and demand in service industry involved with complex business processes. Considering the limitations and characteristics of Web Service environment, traditional transaction model cannot be suited. This paper departed from the architecture of the Web service transaction systems, analyzed the characteristics of Web Service and Web Service transactional applications in an open Web environment, and finally proposed a nested model of Web Service transaction combining alternative and compensation. We also illustrated the structure and processing of Web Service transaction combining with alternatives and compensation mechanism and presented a formal representation of this model.

Keywords Web service, Web service transaction, Transaction model

随着互联网和 Web 技术的迅猛发展,Web 应用正在从局部化发展到全球化,从 B2C(business-to-customer)发展到 B2B(business-to-business),从集中式发展到分布式[1]。将 Internet 整合成一个虚拟的、透明的、开放的计算环境,Web Service 这种平台无关的、松耦合的崭新分布式计算模式,在相关领域得到高度重视和应用[2,3]。对于其中大量典型的复杂业务流程,很多情况下特定业务流的实现需要跨越多个企业和组织,并且业务流程间往往具有复杂互操作。为了确保跨企业的业务流程可靠且一致地执行,必须支持事务性[4]。传统事务依赖于紧耦合协议(如原子事务的锁协议),这对于时间和位置均分离的基于 Web Service 应用来说限制太多。Web Service 环境需要更加宽松的、不严格遵守 ACID 属性的事务形式。此外,由于 Web Service 应用环境自身的异构性和网络传输的不稳定性,跨组织的事务性 Web Service 故障率更高。对于复杂业务流程来说,跨组织的长事务的故障恢复需要由各个服务结点来完成,代价很高[5]。如何确保支持复杂业务的 Web Service 事务的有效执行是实现 B2B 等电子服务系统的关键和基础问题。

Web Service 事务的研究已经引起相关领域专家的重视[6-8],但现有研究都是基于传统事务模型,考虑到 Web Service 环境的限制和特征,传统事务模型无法适应。本文从 Web Service 事务系统体系结构出发,根据 Web Service 特点,分析 Web Service 事务的特殊性质与约束,综合 Web Service 事务的替代性和补偿性,提出了一个基于替代和补偿的嵌套 Web Service 事务模型,通过事务的有效替代来尽量降低事务故障率,减少整个事务的重启次数,提高事务执行效率;同时对于无法替代的故障事务考虑事务补偿,可有效保证系统的一致性和正确性。在阐述结合替代和补偿的 Web Service 事务结构和处理模型的基础上,给出了其形式化的表示。

1 Web Service 事务系统体系结构

Web Service 事务系统是一个多层结构,如图 1 所示。该结构共有 3 层,分别为业务事务层、Web Service 层和资源层。其中:

到稿日期:2012-11-08 返修日期:2013-05-08 本文受国家自然科学基金(60940032,61073034),教育部新世纪优秀人才支持计划(NCET-10-0239),国家科技支撑计划重大项目(2006BAK01A07),国家科技支撑计划重点项目(2006BAC18B06)资助。

党德鹏(1971—),男,博士后,教授,主要研究方向为事务信息系统及应用、数据管理、分析与服务软件,E-mail:ddepeng@bnu.edu.cn;姜 雪(1988—),女,硕士生,主要研究方向为数据库应用和服务计算。

(1)业务事务层的工作单元是 Web Service 事务(WST)。

对于来自服务请求者的业务请求,一个业务流程的处理对应着业务事务层的一个 Web Service 事务的执行。每个 Web Service 事务由一系列 Web Service 子事务(S-WST)协同实现,其中 Web Service 子事务间的执行顺序和依赖关系由业务逻辑的需求而定。每个 Web Service 子事务既可以是 Web Service 长事务(如图 1 中虚线箭头所示),也可以是由独立的 Web Service 构成的原子事务。

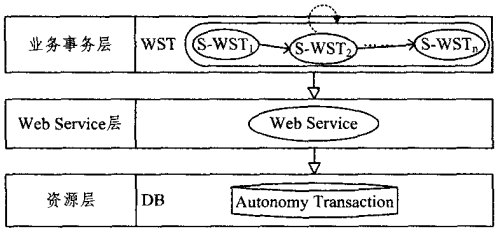

图 1 Web Service 事务系统体系结构

(2)Web Service 层的工作单元是 Web Service(WS)。

Web Service 是一个平台独立的、松耦合的、自包含的 Web 应用程序,可使用开放的 XML 标准描述、发布、发现、协调和配置这些应用程序,用于开发分布式的互操作的应用程序[10]。在实际应用中,许多复杂的业务流程往往涉及多个相互协作的 Web Service 应用。它的分布式工作环境包含 3 种角色:服务提供者、服务注册和服务请求者。服务注册机构维护一个已发布服务和搜索服务的列表;服务提供者在服务注册中发布服务,并获取服务的绑定信息;服务请求者将这些绑定信息绑定到服务提供者,进而使用所提供的服务。服务提供者使用 WSDL 文档描述他们本地业务流程的 Web 接口,通过 UDDI 库发布成 Web Service,并且服务请求者与提供者间通过 SOAP 协议进行相互间的信息交换。

Web Service 是构成 Web Service 事务的基本单位。一个 Web Service 事务由多个功能独立的 Web Service 按照既定的业务流程顺序实现。Web Service 由服务提供者实现。Web Service 层为业务事务层提供了 Web Service 的调用接口。对 Web Service 的并发控制也在这一层实现。

(3)资源层是数据库层,它的工作单元是数据库级的原子事务。

每个 Web Service 操作都对应着数据库的事务,即通过数据库事务对资源的读写操作来实现 Web Service 层的服务功能。

2 Web Service 事务的特性

与传统应用程序相比,Web Service 工作环境具有分布透明性、平台无关性和高度自治性等显著特点。因为在分布式的网络环境中,数据源和服务提供者是分散在各个服务站点中的,服务请求者不必关心服务提供者的数据源格式、某一服务请求需调用哪些服务以及如何被执行等问题。所以 Web Service 具备分布式的服务响应和松散耦合的特征,对用户具有分布透明性。而且 Web Service 使用易理解的标准 Web 协议描述组件接口,使得不同软件平台的差异得以屏蔽,具有平台互通性。同时,Web Service 的服务提供者拥有对服务的控制权,决定服务的执行过程与资源的使用[8]。

具有复杂业务流程的 Web Service 事务需要由多个组织提供的 Web Service 来协同实现,而 Web Service 的上述特点决定了 Web Service 事务已不能满足传统原子事务严格的 ACID 特性。Web Service 事务具备以下特征:

(1)运行周期长:一个 Web Service 事务的执行时间一般很长,通常以小时或者天为单位[9]。因为事务在进行业务处理的过程中,可能由于网络原因导致延迟,或者存在用户交互,这些因素都决定了 Web Service 事务是长事务,所以 Web Service 事务不得不放松为保证分布式事务串行化而设定的严格的隔离性需求。

(2)分布式:Web Service 事务通常由在不同服务提供者端执行的子事务组成,所以在事务的执行和提交阶段,要考虑 Web Service 事务的分布式管理策略。

(3)嵌套性:工作流业务应用通常依赖于协作活动,通过嵌套简单的流程来定义复合流程,因此这些嵌套流程的事务性执行需要支持嵌套事务。Web Service 事务的执行依赖于 Web Service 子事务间的协作,而每个 Web Service 子事务执行过程中也可能请求其他 Web Service 事务,所以 Web Service 事务具有可嵌套性。

(4)替代性:构成 Web Service 事务的一系列 Web Service 可能由不同的服务提供者提供,每个组织会有其独立的业务规则和平台环境[10],这种异构性恰好为 Web Service 事务的替代提供了支持。针对不同组织发布的功能相同、性能相近或相异的 Web Service,当某服务出现故障时,Web Service 事务可选择来自其他组织的 Web Service 来满足等价的功能需求。

(5)补偿性:由于 Web Service 应用环境自身的异构性和网络传输的不稳定性,Web Service 事务潜在的故障很多。为了保证系统的正确性和一致性,Web Service 事务是可补偿的。传统的数据库事务的撤销与恢复策略在 Web Service 环境下已行不通,因为不可以使服务请求者实现服务提供者端的补偿逻辑,而且分散的补偿逻辑不利于策略的应用。

考虑到传统事务在 Web Service 环境中的诸多约束,以及 Web Service 事务的自身特点,传统事务模型无法适应开放的 Web Service 环境。本文提出基于替代和补偿的嵌套 Web Service 事务模型。替代事务为正在执行的事务提供了功能等价的备选执行路径,降低了事务由于故障而重启的概率,提高了事务的执行效率;补偿事务通过撤销部分提交的事务对系统的影响,确保了整个系统的一致性;另外事务嵌套也同时保证了 Web Service 的可集成性和灵活性。

3 基于替代和补偿的嵌套事务模型

Web Service 事务模型是研究 Web Service 事务的基础。本文从 Web Service 事务系统体系结构出发,分析传统事务在 Web Service 环境下的制约和限制,提出了基于替代和补偿的嵌套 Web Service 事务模型 ACNM(Alternative Compensation Nested Model)。Web Service 事务的替代使得一个事务可有多条可执行路径。当事务执行失败而且没有替代事务,或者替代事务也失败时,为了消除已提交的部分事务的影响,事务的补偿便显得尤为重要。另外,一个复杂的 Web Service 事务也可由其他 Web Service 事务即 Web Service 子事务来完成,所以,Web Service 事务也是嵌套的事务。

3.1 Web Service 事务结构模型

定义 1 Web Service 事务(WST)是一系列 Web Service

子事务的有序组合。Web Service 子事务也属于 Web Service 事务，它由一个或者多个 Web Service 实现。换而言之，Web Service 事务是一系列 Web Service 的有序组合。Web Service 事务也称为业务事务。

定义 2 Web Service 的替代(WSA)是指具有与某 Web Service 功能相同或者相近的其他 Web Service 的集合。

定义 3 Web Service 事务的替代(WSAT)是一个 Web Service 事务，该事务具有与被替代的 Web Service 事务功能等价的执行路径。业务事务的功能等价的执行路径是依靠 Web Service 的替代来实现。一般地，为保证 Web Service 事务的成功运行，一个 Web Service 事务具有多种功能等价的执行途径。当构成业务事务的 Web Service 出现故障时，只要该 Web Service 的替代服务正确执行，便可保证整个 Web Service 事务的继续执行。

Web Service 事务的替代来源于 Web 运行环境的松散性和不稳定性。在实际应用中，由于 Web Service 事务运行时间长、运行故障多，一个 Web Service 事务的实现需要具有多种备选方案。Web Service 事务维护了在其覆盖范围里发布注册的 Web Service 列表，列表包含了该 Web Service 的功能信息、性能信息、调用接口和替代服务索引。索引指向的 Web Service 与该 Web Service 功能相等，可相互替代。当业务事务因为某一环节的 Web Service 的失败而无法继续执行时，Web Service 事务可以通过索引选择故障服务的替代服务重新执行，这样选择新的服务而不重复执行原有服务，即可改变原有业务事务的执行路径。这种做法可以避免当子事务由于资源竞争被高优先级事务抢占，或者因为其它原因运行失败而需要重启时，如果依然重复被抢占子事务的原执行路径，那么其实导致重启的因素并没有消除，则势必会造成再次重启；如果选择另一条可替代的路径，那么就可能避开障碍而获得成功，从而提高该事务的成功率。

定义 4 Web Service 事务的补偿(WSCT)也是一个 Web Service 事务，它用来撤销已提交的 Web Service 事务或者部分 Web Service 事务的影响。构成补偿事务的 Web Service 与构成已提交事务的各个 Web Service 功能相反，以达到消除子事务结果的目的。基于替代和补偿的嵌套 Web Service 事务模型 ACNM 如图 2 所示。

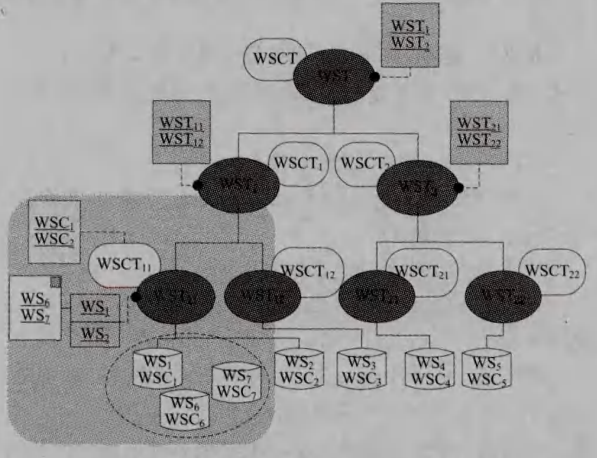

图 2 基于替代和补偿的嵌套 Web Service 事务树模型

该模型把顶层事务与所有子事务看作一个事务树。该模型有两类节点，树的非叶子节点代表事务，树的叶子节点是 Web Service 服务，节点边代表相关事务间的父子关系。在该模型中，树的根节点代表整个业务事务，如图中的事务 WST，它负责整个事务执行的过程控制与协调，以及补偿事务的触发。

每一个事务节点维护了其需要调用的子事务列表，如图 2 中虚线连接的方框所示，子事务 WST_1 的列表中 WST_{11} 和 WST_{12} 列表项表示 WST_1 将由 WST_{11} 和 WST_{12} 两个子事务实现。所有事务功能的实现最终都是由叶子节点的 Web Service 来完成。事务节点 WST_{11} 维护了一个服务列表，该服务列表记录了完成事务 WST_{11} 所需要请求的服务 WS_1 和 WS_2 的功能描述、性能信息、调用接口以及替代服务索引等等。

另外，每个 Web Service 节点都拥有自身服务的替代服务集，如图中灰色虚线圆圈内 WS_1 的两个替代服务 WS_6 和 WS_7。多数情况下，替代服务都分散在不同的站点中。对于 WS_1 的替代服务 WS_6，WS_1 的父节点 WST_{11} 搜索 WS_6 的注册机构，根据 WS_6 的功能描述(与 WS_1 相同)，将 WS_6 视为 WS_1 的一个替代服务，并将 WS_6 的信息添加到父节点 WST_{11} 服务列表中 WS_1 列表项的替代服务索引中，如图 2 中的折角矩形所示。这样保证了在 WS_1 故障时，事务节点 WST_{11} 能够通过索引直接请求其他替代服务重新执行。正是由于 Web Service 的替代服务的存在，才使得事务的执行具有不同的路径，如事务 WST 可能的执行路径可以是 $WS_1 \rightarrow WS_2 \rightarrow WS_3 \rightarrow WS_4 \rightarrow WS_5$，也可以为 $WS_6 \rightarrow WS_2 \rightarrow WS_3 \rightarrow WS_4 \rightarrow WS_5$，后者即为前者的替代事务。

此外，为确保补偿操作的正确执行，Web Service 在本地都有相应的补偿服务，补偿服务由服务提供者自行开发和维护，即 Web Service 与其补偿服务是一一对应的。如图 2 中服务 WS_1 与其补偿服务 WSC_1 在同一服务站点中。同样，每个事务节点也对应着补偿事务，如图中事务 WST_{11} 的补偿事务 $WSCT_{11}$。补偿事务 $WSCT_{11}$ 维护了补偿服务列表，列表中包含 WSC_1，WSC_2 的调用接口信息。一旦遇到故障需要补偿，事务节点 WST_{11} 通过当前服务标识搜索补偿服务列表申请补偿操作。当服务 WS_1 故障而需要请求服务 WS_6 替代执行时，事务节点 WST_{11} 会将 $WSCT_{11}$ 中 WSC_1 替换成 WSC_6。这样，若事务 WST_{11} 需要补偿，则它的补偿事务 $WSCT_{11}$ 可以直接调用 WSC_6 和 WSC_2 进行补偿。有些 Web Service 之间互为补偿服务，如网上购票系统中的买票服务和退票服务互为补偿。在这种情况下，发布买票服务和退票服务的机构可不必发布其相应的补偿服务(退票服务和买票服务)，请求买票服务的父节点可直接将退票服务的信息绑定到自身的补偿事务列表中，以备补偿时调用。

ACNM 事务树模型需要重新考虑事务的原子性，并且放松了传统事务的隔离性。

ACNM 事务的原子性分为强原子性和弱原子性两种。强原子性约束的事务要求要么全部提交，要么全部撤销到事务开始前的状态；而弱原子性约束的事务则支持阶段性提交。事务满足何种原子性取决于具体的实际应用。如图 2 所示的模型中，假定事务 WST_1 满足弱原子性，那么其子事务 WST_{11} 一旦完成，便可通知父事务 WST_1 申请提交，若父事务 WST_1 允许提交，WST_{11} 便提交成功，而不必等待子事务 WST_{12} 一起提交。所以对于事务 WST_1 而言，提交过程中产

生了中间状态。而模型中的叶子节点事务是独立的 Web Service 应用，它满足强原子性。若父事务因为某种原因提前中止，那么正在提交的子事务停止提交，已提交的子事务要进行补偿。

传统原子事务的隔离性要求是对同一时间访问相同资源的事务进行串行化执行，即保证多个事务不能同时修改相同数据，而且保证事务操作产生的变化直到变化被提交或终止时才能对另一个事务可见。数据库采用了锁的机制来实施隔离。但此机制对于 Web Service 事务不完全适用，因为长时间的事务通常在执行期间要访问许多数据项，这些数据项直到事务提交才能被释放，而 Web Service 事务的运行时间很长，会导致访问相同数据项的短事务需要等待相当长的时间（因为数据项保持被锁定状态）。所以，Web Service 事务放宽了隔离性要求，它允许并发的事务间共享中间结果。如图 2 的事务模型中，事务 WST_1 满足弱原子性，若其子事务 WST_{11} 执行完成并提交，那么 WST_{11} 对应的资源锁定予以解除，并且 WST_{11} 生成的新的数据结果对其他事务可见。而不必等到事务 WST_1 提交完成才共享结果数据。

3.2 Web Service 事务处理模型

由于 Web Service 事务是长事务，请求的 Web Service 分散在各个服务站点，为了节省资源以及提高执行效率，整个业务事务不能等到最后一个子事务执行结束才提交，因此 Web Service 事务可以分阶段进行提交，即每个子事务执行完成便可提交。当某个阶段中的子事务执行失败，首先考虑该子事务的替代事务，若替代事务可以成功执行，那么该阶段的子事务便可顺利提交；否则继续重新执行可用的替代事务。若所有替代事务均无法成功提交而导致事务无法继续执行，就需要启动补偿事务对整个 Web Service 事务的失败点之前的各个阶段的子事务进行补偿，以保证整个系统的数据一致性。图 3 描述了考虑替代和补偿的 Web Service 事务处理模型。

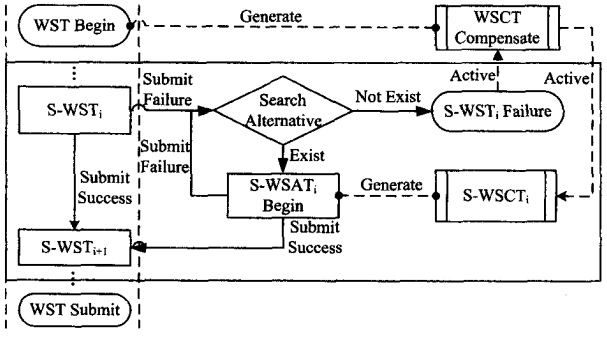

图 3 考虑替代和补偿的 Web Service 事务处理模型

其中，$S\text{-}WST_i$ 表示 Web Service 事务 WST 的第 i 个子事务，$S\text{-}WSCT_i$ 表示子事务 $S\text{-}WST_i$ 的补偿事务，$S\text{-}WSAT_i$ 表示子事务 $S\text{-}WST_i$ 的可用的替代事务。

Web Service 事务 WST 从开始执行到提交，要按照业务流程依次执行每个子事务 $S\text{-}WST_i$，只有所有子事务均成功提交，事务 WST 才可以最终提交。考虑子事务间的业务依赖性和资源冲突，子事务 $S\text{-}WST_i$ 执行提交操作的前提是前 $i-1$ 个子事务已成功提交，即组成前 $i-1$ 个子事务的 Web Service 均可成功执行。

若子事务 $S\text{-}WST_i$ 提交失败，那么选择与 $S\text{-}WST_i$ 功能等价的替代事务重新执行。如果 $S\text{-}WST_i$ 的替代事务存在，并能成功提交，那么继续提交下一个子事务 $S\text{-}WST_{i+1}$（为提高事务的执行效率，一般允许上一个子事务提交完成前，随后的子事务在没有资源冲突和逻辑依赖的情况下可以提前开始）。如果替代事务也执行失败，那么继续寻找可用的替代事务。如果替代事务不存在，那么子事务 $S\text{-}WST_i$ 失败。此时会通知整个业务事务 WST 节点激活事务树中各层事务节点的补偿事务 $WSCT_i$，如图 3 中虚线部分所示，WST 补偿事务会按照与相关子事务提交顺序相反的顺序执行前 $i-1$ 个子事务的补偿事务进行事务恢复。

3.3 正确性及规范描述

基于替代和补偿的嵌套 Web Service 事务的正确性及规范描述如下。

定义 5 一个 Web Service 事务是一个五元组：

$WST=(S\text{-}WST, WS, AT, CT, \angle_t)$

其中，$S\text{-}WST$ 表示 Web Service 子事务集，WS 表示 Web Service 集，AT 表示 WST 的替代事务集，CT 表示 WST 的补偿事务集，$\angle_t$ 表示 WST 中子事务的执行顺序。WST、$S\text{-}WST$ 和 WS 的关系如下：

$WST ::= \langle S\text{-}WST\rangle\langle WST\rangle$

$S\text{-}WST ::= \langle WS\rangle\langle S\text{-}WST\rangle$

$WS ::= \langle WS\rangle\langle WS\rangle|\varepsilon$

WS 是一个五元组：

$WS=(ST, WSA, WSC, RR, RC)$

其中，ST 表示基本服务集，WSA 表示 WS 的替代服务集，WSC 表示 WS 的补偿服务集，RR 是 WS 的系统资源需求集，RC 是 WS 的资源约束集。ST、WSA 和 WSC 的关系如下：

$$WSA=\{wsa \mid wsa\in ST \wedge (\forall st\in ST, wsa \xleftrightarrow[Alternate]{Function\ Equal} st)\}$$

$$WSC=\{wsc \mid \forall st\in ST, wsc \xrightarrow{compensate} st\}$$

WSA 与 AT，WSC 与 CT 的关系如下：

$AT ::= \langle WSA\rangle\langle AT\rangle$

$CT ::= \langle WSC\rangle\langle CT\rangle$

定义 6 把 Web Service 事务的替代操作定义为 $\lambda[t]$，其中 t 表示需要替换的子事务，$|\lambda[t]|$ 表示 t 的替换事务的个数，整个操作的返回结果是子事务 t 的可用替代事务集。

定义 7 把 Web Service 事务的补偿操作定义为 $C[t]$，其中 t 表示需要补偿的子事务，整个操作的返回结果是成功或者失败。

定义 8 把 Web Service 事务提交操作定义为 $Exec[t]$，其中 t 表示提交的事务，整个操作的返回结果是成功或者失败。

Web Service 事务提交成功表示为：

$\forall t\in WST, (Exec[t]=\text{true}) \vee (|\lambda[t]|\geqslant 1 \wedge (\exists at\in \lambda[t], Exec[at]=\text{true}))$

Web Service 事务补偿成功表示为：

$\exists t\in WST, (Exec[t]=\text{false}) \wedge (|\lambda[t]|=0 \vee |\lambda[t]|\geqslant 1 \wedge \forall at\in\lambda[t], Exec[at]=\text{false}) \wedge (C[t]=\text{true})$

结束语 Web Service 作为一种崭新的分布式计算模式，具有平台无关性、自治性和松耦合等特点，是应用到应用的交互。这种面向服务的编程模式，已越来越多地应用到涉及复杂业务流程的事务性服务行业。鉴于 Web Service 的事务性要求以及网络环境的不稳定性，为提高业务事务的成功率，需要同时兼顾事务的替代和补偿。本文提出了基于替代和补偿

的嵌套 Web Service 事务模型，并给出了其正确性描述。该模型具有很多优点，很适合开放的 Web 环境。

(1)Web Service 事务是长事务，不具有严格的原子性，整个 Web Service 事务不能等到整个事务结束才提交。该模型允许采用分阶段进行提交，在不影响业务逻辑和保证资源约束的前提下，优先提交已完成的子事务，提高了整个事务的执行效率。

(2)采用先替代后补偿的方法，当某阶段中子事务出现失败，首先考虑该子事务的替代事务，若替代事务可以成功执行，那么该阶段子事务可以顺利提交；否则，启动补偿事务对整个 Web Service 事务的失败点之前的各个阶段的子事务进行补偿，保证了整个系统的数据的完整性和一致性。

(3)为高性能地处理 Web Service 事务、灵活地维护 Web 数据的一致性以及并发地控制 Web Service 事务奠定了基础。

参 考 文 献

[1] 岳昆，王晓玲，周傲英. Web 服务核心支撑技术：研究综述[J]. 软件学报，2004，15(3)：428-442

[2] 王剑辉，吴永明. 基于细胞膜模型的 Web Service 事务管理[J]. 计算机应用与软件，2007，24(1)：87-89

[3] 郭玉彬，奚建清. Web service 的事务协调框架研究与实现[J]. 计算机工程与应用，2009，45(36)：22-25

[4] Papazoglou M P. Web 服务：原理和技术[M]. 龚玲，张云涛，译. 北京：机械工业出版社，2009：5-15

[5] Liu Cheng-fei, Zhao Xiao-hui. Towards flexible compensation for business transactions in Web service environment [J]. Service Oriented Computing and Applications, 2008, 2: 79-91

[6] 许峰，徐碧云，黄皓，等. Web 服务事务中的补偿机制研究与实现[J]. 计算机科学，2006，33(7)：242-244

[7] 张晓雯，黄永忠，李占峻. 基于 Web Service 的工作流补偿机制[J]. 计算机工程，2009，35(24)：99-102

[8] 唐飞龙，李明禄，曹健. 一个 Web 服务事务处理模型：结构、算法和事务补偿[J]. 电子学报，2003，31(12A)：2074-2078

[9] Alrifai M, Dolog P, Balke W-T, et al. Distributed Management of Concurrent Web Service Transactions [J]. IEEE Transactions on Services Computing, 2009, 2(4): 289-302

[10] 申德荣，于戈，张蓉. Web 服务合成中的异构问题[J]. 东北大学学报，2004，25(3)：220-222

(上接第 162 页)

[11] Zhang Yuan-fang, Gill C, Lu Chen-yang. Real-time Performance and Middleware for Multiprocessor and Multicore Linux Platforms[C]// 15th IEEE Internaltional Conference on Embedded and Real-time Computing Systems and Applications. Beijing, China, 2009: 437-446

[12] Baruah S. An improved global EDF schedulability test for uniform multiprocessors[C]// 16th IEEE Internaltional Conference on Embedded and Real-Time Computing Systems and Applications. Macau SAR, China, 2010: 184-192

[13] Li Ning, Kinebuchi Y, Nakajima T. Enhancing Security of Embedded Linux on a Multi-core Processor[C]// 17th IEEE Internaltional Conference on Embedded and Real-time Computing Systems and Applications. Toyama, Japan, 2011: 117-121

[14] Lin T-H, Kinebuchi Y, Shimada H, et al. Hardware-assisted Reliability Enhancement for Embedded Multi-core Virtualization Design[C]// 17th IEEE Internaltional Conference on Embedded and Real-time Computing Systems and Applications. Toyama, Japan, 2011: 101-105

[15] Yoon M-K, Kim J-E, Sha Lui. Optimizing Tunable WCET with Shared Resource Allocation and Arbitrationin Hard Real-time Multicore Systems[C]// 32th IEEE Real-Time Systems Symposium. Vienna, Austra, 2011: 227-238

[16] Nógueira A, Calha M. Predictability and efficiency in contemporary Hard RTOS for multiprocessor systems[C]// 17th IEEE Internaltional Conference on Embedded and Real-time Computing Systems and Applications. Toyama, Japan, 2011: 3-8

(上接第 184 页)

化查询扩展算法，并在 CSE 中实现了该方法。研究的关键在于规则库中规则的形式、匹配的方式和整个扩展算法的设计。最后利用实验证明了该扩展方法比 Google 具有更好的查准率。

这里的扩展仅考虑了环境信息，为了更好地理解用户意图，生成符合用户需求的查询，未来的工作将在基于环境信息的查询扩展研究的基础上，进一步融合用户的其他上下文信息，如历史记录、用户浏览偏好以及用户社交网络等的查询扩展与查询推荐技术。

参 考 文 献

[1] Chirita P-A, Firan C S, Nejdl W. Personalized Query Expansion for the Web[C]// SIGIR 2007 Proceeding. 2007: 7-14

[2] 吕碧波，赵军. 基于相关文档池建模的查询扩展[J]. 中文信息学报，2006，20(3)：78-83

[3] 王秉卿，张奇，吴立德，等. 机器学习的查询扩展在博客检索中的应用[J]. 中文信息学报，2008，22(6)：98-102，109

[4] Anderson N. Putting Search in Context: Using Dynamically-Weighted Information Fusion to Improve Search Results [C]// Eighth International Conference on Information Technology. New Generations, 2011: 66-71

[5] Gui Feng, Adjouadi M, Rishe N. A Contextualized and Personalized Approach for Mobile Search[C]// International Conference on Advanced Information Networking and Applications Workshops. 2009: 966-971

[6] Ahmadian N, Nematbakhsh M A, Vahdat-Nejad H. A Context Aware Approach to Semantic Query Expansion[C]// International Conference on Innovations in Information Technology. 2011: 57-60

[7] 陈翀，彭波，闫宏飞，等. 一种词汇共现算法及共现词对检索系统排序的影响[J]. 清华大学学报：自然科学版，2005，45(S1)：1857-1860