第34卷 第14期 计 算 机 工 程 2008年7月
Vol.34 No.14 Computer Engineering July 2008
·软件技术与数据库· 文章编号：1000—3428(2008)14—0041—03 文献标识码：A 中图分类号：TP301.6

支持QoS约束的组合服务事务恢复算法

李建华，曾慧琼，陈松乔
（中南大学信息科学与工程学院，长沙 410075）

摘 要：在参与者服务发生异常时，如何有效保证组合服务的服务质量(QoS)是一个亟待解决的问题。该文讨论组合服务在执行过程中动态QoS的计算，建立一个组合服务嵌套事务模型，提出一个组合服务事务恢复算法。该算法能将补偿的范围控制在低层的域内，以减少补偿代价，保证QoS接近最优，同时保证组合服务事务的语义原子性。

关键词：Web服务；组合服务；事务；服务质量

Composite Service Transaction Recovery Algorithm Supporting QoS Constraints

LI Jian-hua, ZENG Hui-qiong, CHEN Song-qiao
(College of Information Science and Engineering, Central South University, Changsha 410075)

【Abstract】It is an urgent problem that how to insure QoS of composite service effectively when exception happens in participant service. This paper discusses the dynamic calculation of QoS when composite service is running, builds an embedded transaction model of composite service and presents a transaction recovery algorithm of composite service. The algorithm can keep the area of compensation in a lower scope, reduce the cost of compensation, and insure QoS approach to the best when transaction recovering and relaxed atomic of transaction-semantic atomic.

【Key words】Web service; composite service; transaction; QoS

Web服务中的服务质量(QoS)问题越来越受关注。保证QoS是Web服务在商业应用中获得成功的关键因素。QoS作为一个可以衡量服务质量的概念，包含了费用、运行时间、可用性、信誉度、可靠性等非功能属性[1]，能体现事务的优劣。采用支持QoS约束的组合服务事务恢复是在事务发生异常时有效保证QoS的重要手段。虽然多数Web服务事务模型都采用了补偿，但没有一种在补偿过程中考虑了QoS，导致组合服务可能因为补偿而费用消耗过高。因此，需要在组合服务事务的恢复过程中采取一种有效的机制来保证QoS。

1 相关工作

尽管高级事务模型(如文献[2]的研究)已经取得了一定的成果，但并不完全适合组合服务事务，原因在于ATM通常面向数据集中的应用，事务结构模式固定，而组合服务事务的失效恢复比ATM复杂。

文献[3]着重讨论了补偿代价和终端用户需求这2个重要方面，并在补偿代价方面提出了一个多层补偿的方法。但该方法在现有的规范和平台上是不可行的，因为无法得到组合服务在不同层次的全局视图。文献[4]在Web服务事务中融入了QoS管理，提出了一个基于QoS的主动两段提交协议，但是该文并没有讨论Web服务事务恢复中的QoS管理。

2 组合服务事务恢复中的QoS指标

2.1 QoS指标分析

Web服务质量模型考虑的QoS指标主要有执行费用、响应时间、成功率、信誉度、可用性[5]。由于信誉度和可用性都涉及组合服务的多次执行，因此组合服务事务在恢复过程只须考虑执行费用P、响应时间T、成功率S 3个指标。

假设组合服务发布的服务质量为QoS_{pvd}；服务用户能接受的极限服务质量为QoS_{ulimit}；服务提供商能接受的极限服务质量为QoS_{plimit}；组合服务执行过程中实际服务质量为QoS_{act}；组合服务执行到具体某个活动时，将来可能消耗的服务质量为QoS_{future}。其中，前三者能在组合服务执行前确定，在服务执行过程中固定不变；QoS_{pvd}可以利用文献[2]中的公式计算得到。极限QoS用于描述在组合服务无法达到预期的QoS_{pvd}时，用户或者服务提供商所能忍受的最差服务质量，在组合服务的事务恢复过程中必须予以考虑。QoS_{act}和QoS_{future}则是动态的，随着业务流程的推进而动态变化，其中，QoS_{act}受极限QoS约束，组合服务事务在执行过程中若无法满足该约束，则终止。

设 $P_{\text{limit}}=\min(P_{\text{ulimit}},P_{\text{plimit}})$，$T_{\text{limit}}=\min(T_{\text{ulimit}},T_{\text{plimit}})$，$S_{\text{limit}}=\max(S_{\text{ulimit}},S_{\text{plimit}})$，则有如下约束关系：

$$P_{\text{act}}<P_{\text{limit}},T_{\text{act}}<T_{\text{limit}},S_{\text{act}}>S_{\text{limit}} \tag{1}$$

Web服务的各项QoS指标对整个服务质量的影响并不一致，如服务费用越小越好，而成功率越大越好，因此，必须抵消服务费用等给服务质量带来的负面影响。下面先给出每一个QoS指标相对于极限QoS的离差：

$$\Delta P=\frac{P_{\text{limit}}-P}{P_{\text{limit}}},\Delta T=\frac{T_{\text{limit}}-T}{T_{\text{limit}}},\Delta S=\frac{S-S_{\text{limit}}}{S_{\text{limit}}}$$

由此可以对QoS进行量化：

$$W(QoS)=w_P\times\Delta P+w_T\times\Delta T+w_S\times\Delta S \tag{2}$$

其中，w_P, w_T, w_S为权值，分别表示P, T, S对组合服务用户的

作者简介：李建华(1963－)，男，博士研究生，主研方向：分布式计算，软件工程；曾慧琼，硕士研究生；陈松乔，教授、博士生导师

收稿日期：2007-09-01 **E-mail：**zhq265@163.com

重要性，且$w_P+w_T+w_S=1$。若组合服务的某个参与者服务失败，在满足式(1)的前提下，以$W(QoS_{act})$为参考选取候选服务。

2.2 QoS_{act}及QoS_{future}的计算

2.2.1 QoS_{act}的计算

当组合服务事务执行到某个活动 a 时，可以确定已消耗的费用和时间，则有

$$\begin{cases} P_{act} = P_{past} + P_{future} + P_{cancel} \\ T_{act} = T_{past} + T_{future} + T_{cancel} \\ S_{act} = S_{past} \times S_{future} \end{cases} \tag{3}$$

其中，$S_{past}=1$ 表示组合服务事务已成功执行到a；P_{past}, T_{past}为组合服务执行中已消耗的费用和时间；S_{future}为组合服务业务流程中尚未执行的参与者服务可能的成功率；P_{future}，T_{future}为将来可能消耗的费用和时间；P_{cancel}, T_{cancel}为撤销已执行活动所需消耗的时间和费用，当a为普通活动时，两者的取值都为0，当a为域活动时，则视具体撤销情况而定。

2.2.2 QoS_{future}的计算

由于文献[2]中组合服务QoS各指标的计算公式是基于执行路线的，不能直接用于计算P_{future}，T_{future}，S_{future}，因此必须对业务流程进行预处理。

对于异常活动的QoS_{future}，计算步骤如下：

(1)确定从当前异常活动节点开始的业务流程图G_{future}，分以下 4 种情况讨论：

1)异常活动节点出现在顺序结构中。此时，G_{future}直接从异常活动节点开始直至整个业务流程结束，如图 1 所示。

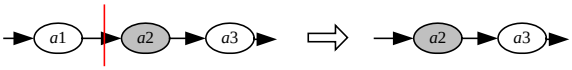

图 1　顺序结构中的G_{future}

2)异常活动节点出现在或分支结构中，如图 2 所示。因为组合服务在执行过程中已经选择了异常活动所在的分支，所以在G_{future}中必须放弃其他分支。

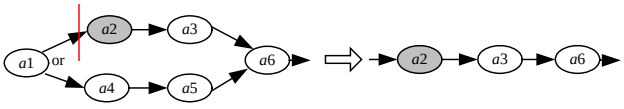

图 2　或分支结构中的G_{future}

3)异常活动节点出现在循环结构中，如图 3 所示。设循环次数为 n，在异常出现时，循环刚好执行了 k-1 次，则原循环结构变为以异常活动为起始节点的 n-k 次循环结构。

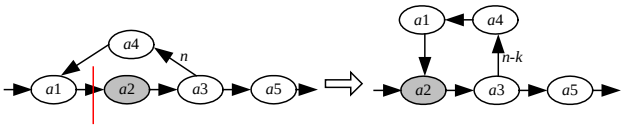

图 3　循环结构中的G_{future}

4)异常活动节点出现在与分支结构中。考虑到其他分支也在执行，必须将异常活动节点和其他分支中未完成的活动节点组成新的与分支结构。如图 4 所示，加入一个空活动节点 *start* 以保证业务流程的单入口特点，该活动不执行任何动作，其 P=0, T=0, S=1。

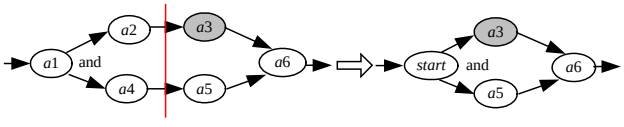

图 4　与分支结构中的G_{future}

(2)确定了G_{future}后，将图中的或分支结构、循环结构以及与分支结构封装为特殊的活动as，并计算这个活动可能消耗的QoS指标值。

1)或分支结构如图 5 所示。有

$$P^{as}_{future} = \sum_{i=1}^{n} pr_i \times P^{ai}_{future},\ T^{as}_{future} = \sum_{i=1}^{n} pr_i \times T^{ai}_{future},\ S^{as}_{future} = \prod_{i=1}^{n} pr_i \times S^{ai}_{future}$$

其中， pr_i 为各分支的概率。

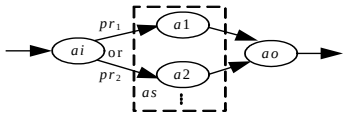

图 5　或分支结构

2)循环结构如图 6 所示。有

$$P^{as}_{future} = \sum_{i=1}^{n} P^{a1}_{future},\ T^{as}_{future} = \sum_{i=1}^{n} T^{a1}_{future},\ S^{as}_{future} = \prod_{i=1}^{n} S^{a1}_{future}$$

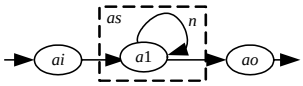

图 6　循环结构

3)与分支结构如图 7 所示。有

$$P^{as}_{future} = \sum_{i=1}^{n} P^{ai}_{future},\ T^{as}_{future} = \max(T^{a1}_{future}, T^{a2}_{future}, \cdots, T^{an}_{future})$$

$$S^{as}_{future} = \prod_{i=1}^{n} S^{ai}_{future}$$

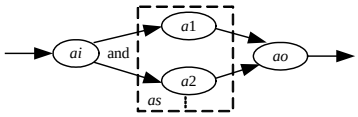

图 7　与分支结构

(3)经过上述处理后，G_{future}中不再有特殊结构，用$a1$~an标注，结合文献[5]的思想，给出G_{future}各指标的计算公式：

$$P_{future} = \sum_{i=1}^{n} P^{ai}_{future},\ T_{future} = \sum_{i=1}^{n} T^{ai}_{future},\ S_{future} = \prod_{i=1}^{n} S^{ai}_{future}$$

3　基于域的组合服务嵌套事务模型

3.1　域的划分

由于服务提供者无法知道服务参与者的事务行为，因此组合服务事务是平坦的。本文从服务提供商的角度考虑，结合补偿的特点，将平坦的组合服务事务模型扩展为嵌套的事务模型。

定义 1　域是多个连续活动或子域的有意义的封装，可看作一个特殊的活动。用 s 表示一个域，则 s=[$a1$, $a2$,⋯,an]，其中，ai 表示一个活动或子域。

例如，组合服务中存在 2 个连续活动：订火车票 $a1$，订酒店 $a2$，若网上有可以同时完成这 2 个功能的服务，则可以将它们封装为一个域：s=[$a1$,$a2$]。

组合服务的业务流程图为有向图，用$G_B(V_B,E_B)$表示。给定一个活动集X，可以得到X在G_B的一个有向子图$G_X(V_X,E_X)$，其中，$V_X=X\subseteq V_B$；E_X为V_X在G_B中所有的有向边的集合；$E_X\subseteq E_B$。同理，给定一个域s，根据域中的活动集，可以得到s在G_B中的子图$G_s(V_s,E_s)$。

定义 2　给定一个活动集X，称X中的活动是连续的，当且仅当G_X是连通的。

定义 3　在G_B中给定子图G_s, $G_B\neq G_s$，存在$a_{ss}\in V_s$, $v_B\in V_B$, $v_B\notin V_s$，且$(a_{ss}, v_B)\in E_B$，则称a_{ss}为G_s的始活动。存在a_{se}

$\in V_s$, $v_B \in V_B$, $v_B \notin V_s$，且$(v_B, a_{se})\in E_B$，则称a_{se}为G_s的终活动。当$G_B=G_s$时，G_s的始活动为G_B的起始活动，G_s的终活动为G_B的终止活动。

规则 1 域的划分规则

(1)域之间不能交叉重叠，但可以嵌套。

(2)域必须是单入口的，即G_s只有一个始活动，但可以有多个终活动。

(3)域必须具有一定的意义，即组合服务能对这个域的异常进行有效处理，如忽略、补偿。

(4)域中的活动必须是连续的，只有连续的活动或者域才能划分为新的域。

(5)整个业务流程是一个域s_B，处于最顶层。

给定一个业务流程，并根据规则 1 进行域的划分，如图 8 所示。图 8 中共有 5 个域：$s11$=[$a2$,$a3$], $s22$=[$s11$,$a4$], $s23$=[$a5$,$a6$], $s34$=[$a1$,$s22$,$s23$], $s45$=[*start*,$s34$,$a7$, *end*]。

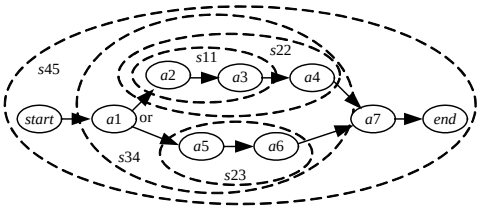

图 8　域的划分示例图

根据规则 1 以及域之间的嵌套关系，可以构造一棵事务树*TT*。树的节点为业务流程中的活动(或域)子事务。其中，活动子事务以事务树叶子节点的形式出现，而s_B构成的事务则是事务树的根。

定义 4 组合服务嵌套事务模型可以看作是一个五元组：*CWTM*=(*TT*,*IS*,*RS*,*CP*,*CO*)，其中，*TT* 为事务树；*IS* 为初始的参与者服务集合；*RS* 为事务树中顶点的恢复策略集；*CP* 为候选服务集；*CO* 为已提交活动的撤销操作集。

定义 5 若存在已完成的活动，且在业务流程中处于异常活动的父域之后，则称这种情况为越界，如图 9 所示。

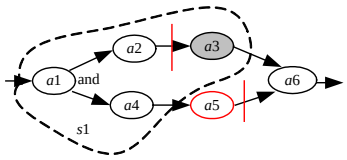

图 9　越界示例图

图 9 中 $a3$ 发生异常，$a5$ 刚好执行完毕，显然，$a5$ 对于 $s1$ 越界。对于越界的活动，异常活动的父域无法消除产生的影响。但越界是一种特殊情况，本文不予考虑。

3.2　事务恢复策略

通用的处理策略有忽略 in、重试 rt、补偿 cp、关键 ct 等。关于事务异常处理策略的文章比较多，本文不再详述。在组合服务事务的恢复过程中只须保证服务的语义原子性即可。

定义 6 组合服务事务满足语义原子性，当且仅当它满足以下 2 个条件之一：(1)成功完成；(2)有效撤销，成功终止。

规则 2 继承规则

若父域是可忽略(或可重试)的，则子域一定可忽略(或可重试)。

4　组合服务事务恢复算法

4.1　算法描述

本算法基于上述组合服务嵌套事务模型，并且有如下前提条件：(1)组合服务所选取的初始服务参与者能保证组合服务达到 QoS 最优。(2)撤销事务能正确执行。(3)不会出现越界的情况。

4.1.1　相关子算法

(1)域的撤销事务的生成。已经提交的子事务必须是可以撤销的。一旦子事务提交完成，则其撤销事务(即撤销操作或撤销子事务序列构成的事务)产生并生效。当组合服务中一个具体的活动提交完成时，其撤销操作构成撤销事务。当某个域 *s* 提交完成时，按照 *s* 中活动(包括子域)执行轨迹的逆序将轨迹中其撤销事务串起来构成 *s* 的撤销事务。撤销事务由父域启动执行。

(2)计算QoS_{act}的子算法calQoS。QoS_{act}是动态变化的，在未执行的活动节点上选择不同的服务参与者，值不同。当某个活动节点失效时，可以按照 2.2 节中给出的计算步骤和公式，对其中每一个候选服务计算QoS_{act}。

(3)判断是否满足QoS约束的子算法isSat。在选择候选服务时，此算法用于判断选候选服务能否保证整个组合服务的QoS_{act}满足极限QoS约束。首先调用calQoS计算QoS_{act}各指标值，再根据式(1)判断该候选服务是否满足约束条件。

(4)对QoS_{act}进行量化的子算法estQoS。主要根据式(2)对QoS_{act}进行量化，以便补偿时能选取最好的候选服务。

4.1.2　组合服务事务恢复算法

输入：异常活动ID，QoS_{limit}

基本处理步骤：

(1) cActID:=ID。

(2)若异常活动为普通活动，则放弃发生异常的服务参与者。

(3)若可以忽略则忽略该活动，返回。

(4)若可以重试，则调用 isSat 判断重试能否满足 QoS 约束条件，满足则重试，返回。

(5)若异常活动可以替换，则：

1)对失效活动的每一个候选服务调用isSat，判断是否满足QoS约束条件，若满足则调用estQoS量化QoS_{act}。

2)检查父域是否可以补偿，若可以，则对父域的每个候选服务调用isSat，判断是否满足QoS约束条件，若满足，则调用estQoS量化QoS_{act}。

3)若上述 2 步的量化结果不为空，则比较量化结果：

①若失效活动中某候选服务的$W(QoS_{act})$最大，则调用该候选服务执行，返回；

②若父域中某个候选服务的$W(QoS_{act})$最大，则父域启动撤销事务执行，置cActID:=父域ID，同时启动候选服务执行，并返回。

(6)将异常抛给父域，置 cActID:=父域 ID。

(7)若异常活动为域，则执行该域的撤销事务。

(8)若父域ID不是顶层域s_B的ID，则转步骤(2)，否则终止组合服务事务。

本算法将失效范围尽可能限制在低层的父域内，因为越往上走，要撤销的活动越多，消耗的撤销费用和时间也越多。

4.2　语义原子性的证明

因为本算法存在 2 个出口：(1)在异常处理策略的作用下，满足约束条件时，重试或者调用候选服务，使得失效活动(域)能够恢复，组合服务能继续向前推进；(2)在极限QoS无法约束或无处理策略时，异常被抛给父域，直至顶层域s_B结束，

(下转第 46 页)