面向组合服务动态自适应的事务级主动伺机服务替换算法

印 莹 张 斌 张锡哲

(东北大学信息科学与工程学院 沈阳 110004)

摘 要 动态服务环境的各种异常随时会导致整个业务流程暂时无响应或服务中断,极大影响业务流程的可靠性.已有替换机制大多缺乏事务支持而适应性差,进而不能有效保证系统执行过程中事务服务的原子性和数据一致性,也无法保障替换过程要求的正确性、实时性和高效性.该文以"事务支持"为核心,充分考虑了服务间多关系以及 Web 服务的事务特性,提出一种事务级组合服务主动伺机替换算法.首先,该文给出了事务粒度的获取及替换范围识别算法.然后,提出事务级替换代价/收益 QoS 模型,将事务补偿代价与替换代价有机结合,保证以最少的代价实现服务替换.在此基础上,提出了全新的 QoS 驱动的事务级服务替换算法.为了提高替换的时效性,提出了早期预测模式挖掘算法,对运行的服务实时监控保证其识别失效服务的早期性.实验结果证明,该模型不仅保证了替换过程中业务流程的事务原子性和数据一致性,而且提高了系统的可靠性和时效性.

关键词 早期预测模式;事务 Web 服务;补偿;事务级替换

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2010.02147

An Active and Opportunistic Service Replacement Algorithm Orienting Transactional Composite Service Dynamic Adaptation

YIN Ying ZHANG Bin ZHANG Xi-Zhe

(*Department of Computer Science and Engineering, Northeastern University, Shenyang* 110004)

Abstract Various exceptions leading to the business process no response temporarily or service interruption are common in dynamic service environment, which reduce the reliability of business process greatly. Due to absent of transaction support, most existing replacement mechanisms are of poor adaptabilities. Therefore, the atomicity and the consistency of the transactional service can not be well assured during the system implementation process, and the correctness, the real-time performance and the efficiency of the replacement process can not be guaranteed either. This paper takes the "transaction support" as the core and proposes an active and opportunistic transactional composite service replacement algorithm, which fully takes into account the multiple relationships among Web services and the transactional characteristics of Web services. In this paper, the authors first present a mechanism to obtain the transactional granularity and a method to identify the cascading compensation range, then, propose a transactional replacement cost/benefit QoS model, which completes service replacement with the least cost by integrating transactional compensation service QoS and service replacement QoS. Further, a novel QoS-driven services replacement algorithm with the transactional compensation support is designed. In order to enhance the real-time performance, the authors introduce the early prediction pattern mining to identify

收稿日期:2010-06-08;最终修改稿收到日期:2010-08-31.本课题得到国家"八六三"高技术研究发展计划项目基金(2009AA01Z122)、国家自然科学基金(61073062,60903009,60803026)及中央高校基本科研业务费专项资金(N090304006)资助.**印 莹**,女,1980 年生,博士,主要研究方向为 Web 服务、数据挖掘. E-mail: yy_00000000@163.com. **张 斌**,男,1964 年生,博士,教授,博士生导师,主要研究领域包括 Web 服务、数据挖掘. **张锡哲**,男,1978 年生,博士,副教授,主要研究方向包括数据挖掘、Web 服务、分布式计算.

the failed or unavailable services as earlier as possible by adopting a real-time monitoring to the running services. Experimental results show that the proposed model not only guarantees the atomicity and the data consistency of the process business in the replacement, but also improves the reliability and the real-time performance of the system.

Keywords early prediction pattern; transactional Web service; compensation; transactional replacement

1 引 言

随着面向服务计算体系架构(Service Oriented Architecture, SOA)的进一步推广,基于 Web 服务的业务流程被广泛应用于工业、农业、服务业、商业和教育业等诸多领域,这就要求基于 Web 服务的业务流程具有较高的稳定性和可靠性.然而,在实际应用中,Web 服务驱动的业务流程始终处在一种动态、分布的环境中,各种异常都可能影响服务的质量,甚至会导致整个组合流程的中断或终止.替换是维护组合服务动态自适应运行的一种重要技术手段,受到工业界和学术界的广泛关注.

传统替换机制大多采用备份服务或者服务路径直接替换.这种策略使系统在一定程度上适应了环境的变化[1-8],有效地使系统不被中断而继续运行下去.然而,系统对事务 Web 服务的支持是组合服务可靠运行的基础,这种直接替换策略忽略了失效服务的事务属性.当系统执行过程中缺乏有效的事务服务原子性和一致性保障时,组合服务替换过程所要求的正确性、实时性和高效性将无法得到有效保证.在实际应用中,尤其面临一些商业流程,传统的直接替换模型缺乏对事务的支持而变得不完备,不可靠,也是不可行的[9].因此,如何在实时变化的动态环境中支持事务级服务替换,并设计可靠的事务级组合服务替换算法,提高事务级服务异常处理能力是实现可靠、稳定的动态组合服务的基础,已成为当前组合服务自适应研究中的热点和重点[10-11].

由于 Web 服务事务具有诸如自治性、松耦合、跨越多组织性以及运行周期长等特性[12],使得采用锁定资源的策略,如严格二阶段提交机制(2PC)不再适用,而是有选择地放松事务对 ACID 性质的要求,并引入补偿机制,用以消除已经执行的 Web 服务对系统的影响(比如释放空间等)[9-13]来维护系统中数据一致性.目前已有支持事务的相关标准和协议,如 BPEL4WS[14]、WS-Coordination[15]、WS- Transaction[15]、WSCI[16]和 WS-CDL[17]等.由于业务流程需要支持事务并支持带事务属性约束的非严格事务提交,当运行中的服务需要被替换时,为了维护替换前后事务服务原子性和数据的一致性,需要对属于一个事务中的未执行服务进行级联替换;对属于一个事务中的已执行服务进行级联补偿,以消除该动作对整个业务流程造成的影响,如释放空间等等.由以上分析可知,在服务运行过程中,当已经提交的事务需要被替换时,为了维护事务的原子性和数据的一致性,需要知道事务粒度及补偿范围.

然而,在实际运行中仍然面临着以下困难:一方面,提供商在设计阶段仅仅对单一服务标识了事务属性,而组合服务的执行过程是不同服务间的交互过程,因此,业务流程中的事务粒度无法由设计者决定,同时,依靠设计者指定事务粒度也是不可靠的;另一方面,由于服务的自治性,组合逻辑在设计阶段也无法显式的标记事务范围,因为服务实例在运行过程中是动态绑定的,由于不同 Web 服务之间存在着诸如数据、控制和业务等多依赖关系,这使得组合服务间的事务粒度及嵌套事务粒度是隐藏的[9].因此,在执行过程中如何根据动态绑定的服务实例确定事务粒度和替换范围,是组合服务可靠替换的基础.

下面给出一个例子进一步说明以上提到的问题.图 1 所示的是一个业务流程组合实例.可以看出,业务流程中的 Web 服务事务都是跨越单个或多个 Web 服务的操作来实现的.*ESC* 表示具有相同功能的等价 Web 服务类.

当某个服务操作被执行了一半时,如预定机票服务(tws_1)中的付款操作(op_3)由于该机构网络终端出现故障而失效,为了不中断整个流程,要求替换失效服务,此时,服务 tws_1 中的查询操作(op_1)和预定操作(op_2)已经被成功执行和提交,由于事务 Web 服务 ACID 性质,使得预定操作(op_2)的提交结果(即该票已经被用户预定,其他客户将不能查询到该票信息)仍然驻留在内存中,不能被释放.如

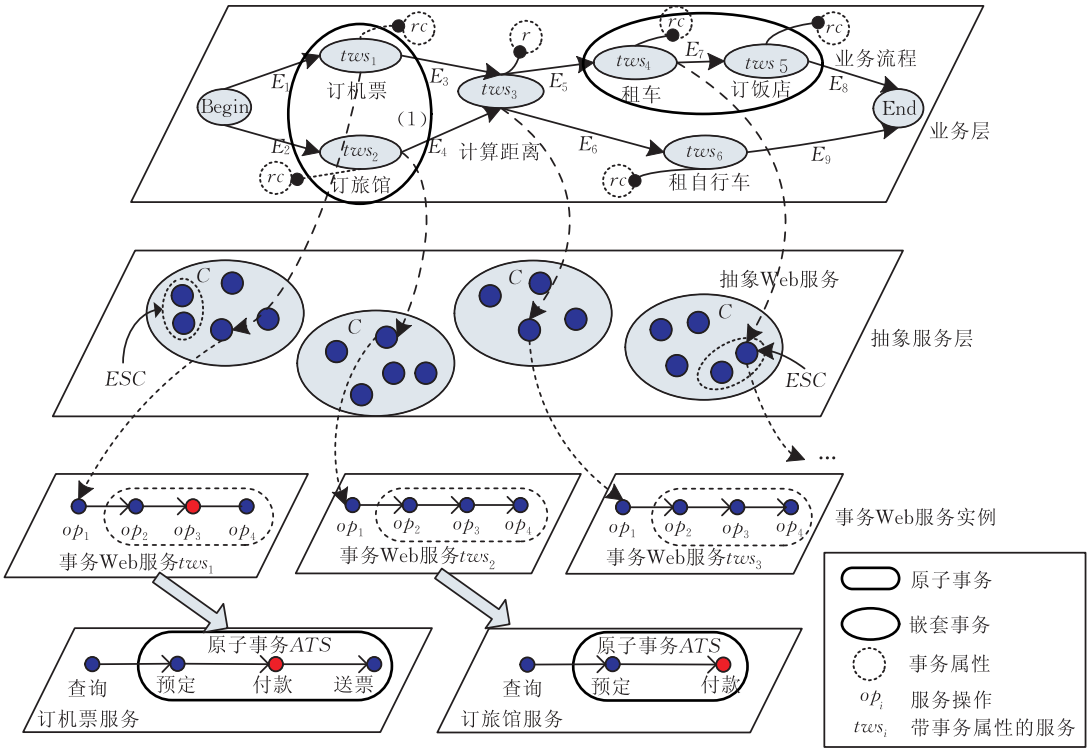


图 1　一个业务流程组合实例

图 1 中的原子事务，如果仅替换单一失效服务操作(op_3)，即选择另一家付款公司直接替换当前付款(op_3)是不可行性，这样就破坏了事务数据的原子性和数据库的一致性. 因此，当原子事务服务中某一操作出现故障时，整个原子事务都要被替换.

同时，由于业务合作，使得订旅馆的服务(tws_2)提供 8 折优惠的房价预定，该用户已经支付旅馆的订金. 也就是说该 Web 服务(tws_1)的处理结果已经被其它事务(tws_2)所使用，造成与 tws_1 级联的事务 tws_2 随之失效. 在这种情况下直接用传统的 Web 服务替换模型也是不可行的，系统虽然没有显式地指出事务(tws_1)和事务(tws_2)是级联事务，但是由于业务的合作，事务(tws_1)和事务(tws_2)已经被绑定，它们属于一个嵌套事务. 如果一个事务(tws_1)需要被替换，那么另一个事务(tws_2)也要随之被撤销，因为已经执行的服务对整个流程造成的影响滞留在内存中，直接替换会造成服务器上数据的不一致，会导致系统中断. 因此，当嵌套事务中的服务出现故障时，整个嵌套事务都要被替换.

综上所述，已有替换模型缺乏对事务的支持而变得不完备和不可靠；替换选取算法单一、适应性不好，不仅灵活性受到限制，而且也不能客观、全面地反映补偿支持的替换服务质量. 主要问题体现在以下几个方面，如：(1) 缺乏对事务的支持而无法面对业务流程的组合服务替换过程中数据一致性问题；(2) 缺乏对事务粒度识别、替换范围确定以及相关联的补偿范围确定，使得已有的替换算法无法进行正确的替换；(3) 缺乏考虑补偿代价对整体替换代价质量影响的 QoS 模型，而无法得到更准确的替换代价/收益模型；(4) 缺乏事务级的早期预测机制，以减少替换的中断时间来降低替换代价的问题. 因此，如何快速识别事务粒度及替换范围来维护替换前后数据的一致性问题是实现正确替换的首要问题；另外，设计一个能够反映补偿支持的服务替换代价/收益模型成为替换过程的重要问题；最后，如何快速发现失效服务并响应替换，减少中断时间，提高替换时效性是事务级替换要解决的另一个重要问题.

为了解决上述问题，本文以“事务支持”为核心，充分考虑服务间多关系以及 Web 服务的事务特性，提出了支持事务补偿机制的主动伺机服务替换算法. 首先，给出事务粒度和级联替换范围的识别算法，然后，将事务补偿代价与替换选取代价有机结合，建立全面评价补偿支持的替换服务 QoS 代价/收益模型. 当服务不能完成任务时，利用支持补偿的替换代价/收益 QoS 模型保证服务以最少的代价实现替换. 进一步，为了提高替换的时效性，提出基于早期预测模式的替换服务启动机制，对运行的服务

进行实时监控保证其识别失效服务的早期性.综上所述,整个服务替换过程分为3个阶段:事务级替换范围识别、事务级替换代价/收益模型的建立、高效替换算法3个阶段.进一步,为了减少替换的中断时间,提出了基于早期预测模式的替换触发机制.整个基本过程如图2所示.

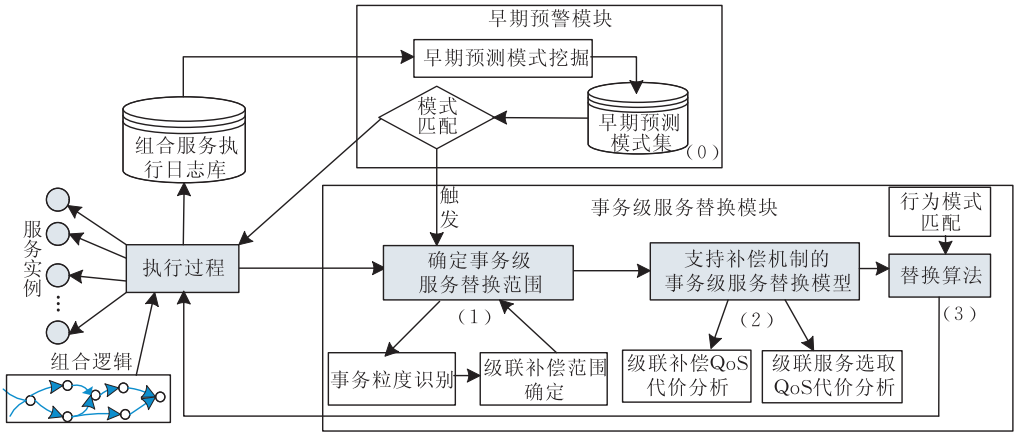


图2 支持事务机制的服务替换过程

第1阶段为事务级服务替换范围确定阶段.该阶段确定级联替换的服务范围.当组合服务在动态执行过程中需要被替换时,为了维护替换前后的业务流程数据一致性,实现正确的替换,需要知道失效服务对应的事务粒度及替换范围,进一步确定回滚的级联补偿范围,并给出补偿路线生成算法.然而,由于服务的自治性,使得跨多组织运行的组合服务间事务粒度是隐藏的,依靠设计者指定事务粒度是不可靠的,也是不完备的,因此,基于服务间多关系识别事务粒度及级联补偿范围是解决该问题的难点.此阶段如图2中标号(1)所示.本文的第3节将介绍这部分内容.

第2阶段为建立事务级服务替换QoS代价/收益模型.事务级替换代价包含补偿代价和替换代价,该阶段通过分析补偿服务QoS、替换服务QoS和补偿的时效问题,建立了灵活的服务替换QoS代价/收益模型.该模型全面、客观、综合地评价了支持补偿机制的替换服务QoS.因此,在保证语义原子性的前提下,使整体QoS接近最优是解决该问题的难点.此阶段如图2中标号(2)所示.本文的第4节介绍这部分内容.

第3阶段为支持事务补偿机制的服务替换算法.该阶段通过子图匹配的服务行为模式挖掘算法实现高效的服务替换.该部分如图2中标号(3)所示,基于已有的前期基础.本文的第4节将给出该部分的简要介绍.

另外,为了提升替换的时效性,尽可能减少服务的中断时间,本文提出了基于早期预测模式的替换启动机制.该机制保证在服务执行过程中,当执行的服务序列与挖掘出的早期预测模式匹配时,说明将来某时刻会有异常发生,提前触发替换模块.该策略能大大减少服务中断时间,保障时效性的前提下,实现开销较少的替换.该部分如图2中标号(0)所示.本文的第5节介绍这部分内容.

本文的创新点如下:(1)提出了基于多关系依赖的事务粒度识别和替换范围确定算法.在分析服务间多关系的基础上,实现跨组织服务的事务粒度识别,为事务级组合服务自适应运行提供可靠支撑;(2)提出了支持事务补偿的替换代价/收益模型,在保证语义原子性的前提下,使整体QoS接近最优;(3)提出了早期预测模式的算法,在保证时效性的前提下,实现开销较少的替换.

本文第2节给出基本定义;第3节讨论事务级替换范围识别算法;第4节介绍支持事务补偿机制的QoS代价/收益模型;第5节给出基于早期预测的替换触发机制,以提高替换的时效性;第6节给出实验结果和分析;第7节介绍相关工作;最后,第8节总结全文并提出未来工作.

2 基本定义

一个复杂的业务流程往往是由多个Web服务聚合而成,而Web服务是由多属性信息描述的,包括功能描述(XML类型)、语义描述、质量信息描述、事务属性描述等.本文研究的目标是维护事务级服务替换过程中的可靠替换问题,因此前提是假设每个Web服务具有事务属性,是可以补偿或重复执行的.根据服务的事务行为,给出带事务属性事务

Web 服务基本定义和带事务属性的组合服务业务流程的逻辑模型.

定义 1.　具有事务属性的 Web 服务. 具有事务属性的 Web 服务可以用 3 元组表示：$TWS=\{F,TP,QoS\}$，其中：F 表示服务 TWS 的功能，TWS 是通过一系列服务操作$\{op_1,op_2,\cdots,op_n\}$来实现. TP 代表服务 TWS 具有的事务行为，$TP\in\{r,p,c,rp,rc\}$. 其中，r 代表该服务操作可以被重复执行，p 代表该服务操作一旦执行不允许被撤销和补偿，c 代表该服务操作可以被补偿，rp 代表该操作同时具有 r 行为和 p 行为，rc 代表该服务操作同时具有 r 行为和 c 行为；QoS 代表服务的非功能属性，$QoS=\{p,et,av,au\}$. 其中，p 代表执行服务 TWS 所需的费用，et 代表执行服务 TWS 所需的时间，包括数据传输时间和服务运行时间，此时传输时间可以忽略不计，av 代表服务 TWS 的可用性，au 表示服务 TWS 执行的成功率，et,av,su 可由通过统计历史执行记录获得.

例如，图 1 所示的业务流程中，服务 $tws_1=\{$订机票，rc，5，5ms，0.9，0.8$\}$，表示该服务 tws_1 的功能的预定机票；它的事务属性是可补偿、可重复被执行的；它的质量信息如下：执行该服务的费用是 5 美元，花费的时间是 5ms，该服务可访问的概率是 0.9，该服务被成功执行的概率是 0.8.

定义 2.　带事务属性的组合服务. 一个组合服务业务流程的逻辑模型可以用 6 元组表示：$TCS=\langle\sum TWS,E,\sum CR,\sum DR,\sum BR,\sum TB\rangle$. 其中：

(1) $\sum TWS$ 是业务流程中的一系列带事务属性的 Web 服务，简记为任务 $\sum TWS=\{tws_1,tws_2,\cdots,tws_n\}$，$tws_i\in TWS(1\leq i\leq n)$ 表示业务流程中的任务(带事务属性的 Web 服务)，任务由相应 Web 服务操作的执行来实现.

(2) E 是有向边的集合，$E_{ij}=(E_i,E_j)\in E$ 表示任务 tws_i 和 tws_j 之间的执行顺序.

(3) $\sum CR$ 表示任务间控制逻辑关系的集合，$\sum CR=\{$Sequence，And-Join，And-Split，Or-Join，Or-Split$\}$，其中，Sequence 表示两个任务间是顺序关系；And-Join 表示两个任务间是并列合并关系；And-Split 表示两个任务间是并列分离关系；Or-Join 表示两个任务间是选择合并关系；Or-Split 表示两个任务间是选择分离关系；

(4) $\sum DR$ 表示任务(tws_i 和 tws_j)间数据逻辑关系集合，$\sum DR=\{0,1\}$，1 表示任务(tws_i 和 tws_j)间存在数据逻辑关系，0 表示任务(tws_i 和 tws_j)间不存在数据逻辑关系.

(5) $\sum BR$ 表示任务(tws_i 和 tws_j)之间的业务逻辑关系集合，$\sum BR=\{0,1\}$，1 表示任务(tws_i 和 tws_j)间存在业务逻辑关系，0 表示任务(tws_i 和 tws_j)间不存在业务逻辑关系.

(6) $\sum TR$ 表示任务的事务属性集，$\sum TR=\{c,r,p,cr,pr\}$，详细属性的行为含义同定义 1 所示.

(7) 映射函数 $s:t\rightarrow States$ 表示 TCS 中任务的状态类型集，$States=\{$initial，active，failed，completed，aborted，cancelled$\}$. 业务流程执行前所有的任务状态为 initial.

例如，图 1 所示的是一个带事务属性的业务流程组合实例，表示的是一次旅行的业务流程. 由订机票、订旅馆、计算路径距离，选择租车或租自行车的业务等组成. 可以看出，业务流程中的 Web 服务事务都是跨越单个或多个 Web 服务的操作来实现的. ESC 表示具有相同功能的等价 Web 服务类. 虚线的圆圈表示服务的事务属性. 箭头表示服务间的控制逻辑关系，数据逻辑关系和业务逻辑关系在每次调用的服务实例中被确定. 每个服务都有(6)中状态，初始态、激活态，失效态、完成态、中断态、取消态. 默认的状态是初始状态.

定义 3.　原子事务服务. 原子事务服务(Atomic Transactional Service)是由一系列事务 Web 服务操作序列组成，简记为 $ATS=\{op_1,op_2,\cdots,op_n\}$，该服务执行序列全部被执行或一个都不被执行. 其中，op_i 代表原子事务服务操作(Web 服务操作)，即事务组合 Web 服务的最小粒度是 Web 服务操作.

例如，图 1 所示的预定机票的服务(tws_1)中预定操作(op_2)和付款操作(op_3)属于一个原子事务，即，预定操作(op_2)和付款操作(op_3)要满足同时被执行或一个都不被执行. 当预定操作(op_2)成功执行时，付款操作(op_3)也要求被成功执行，否则如果付款操作(op_3)由于网络中的故障被要求替换时，必须首先撤销预定操作(op_2)，即从数据库中取消该用户的预定，才能进行替换. 否则，其他客户将不能查询到该票信息，破坏了数据的一致性.

定义 4.　嵌套事务服务. 给定两个原子事务服务(Atomic Transaction Service) ATS_1 和 ATS_2，如果 ATS_1 和 ATS_2 之间存在某种定义的关系 $R\in\{$数据关系、业务关系、补偿关系$\}$，即满足 $R(BT_1,BT_2)=1$，本文称这两个原子事务组成了一个粒度大的事务，用 NTS 表示.

例如，图 1 所示的预定机票的服务(tws_1)和预

定旅馆的服务(tws_2)属于一个嵌套事务.即当用户预订某航空公司的机票,由于业务合作,使得订旅馆的服务(tws_2)提供 8 折优惠的房价预定.当用户临时更改旅行计划时,因为预定机票的服务(tws_1)和预定旅馆的服务(tws_2)属于一个嵌套事务,则意味着同时补偿这两个服务之后,才能实现新的旅行预订.

由定义 4 进一步推广,$R(R(BT_i,BT_j),BT_k)=1$ 可以得到粒度大的事务.进一步,依次可以识别出所有的嵌套事务.基于所得的事务及嵌套事务,可以构造一个事务树,如图 3 所示.

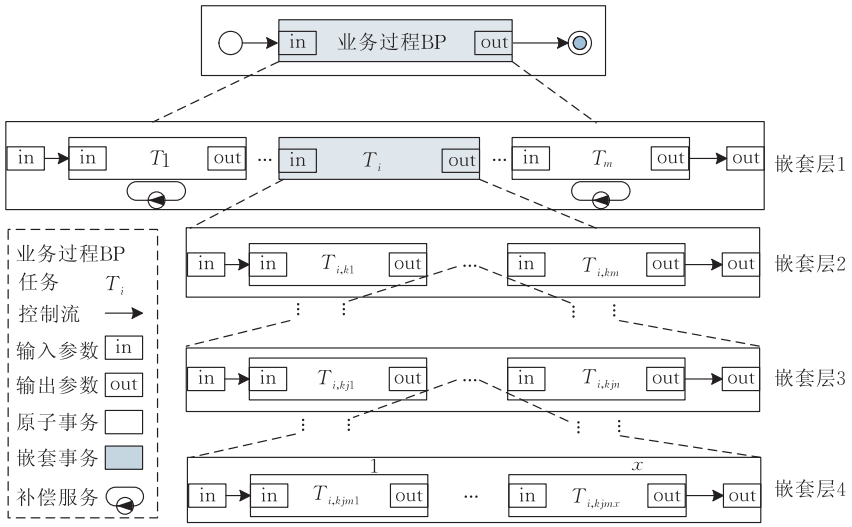


图 3 事务及嵌套事务识别图

3 事务级替换范围识别算法

组合服务间补偿范围的获取是由服务间的补偿依赖关系决定的.由于业务流程中的伙伴服务来自不同地域、不同企业,因此,服务提供者无法知道服务参与者的事务行为.此外,Web 服务工作在不稳定的 Internet 之上,业务流程可能需要运行很长时间(几小时、几天等)来执行复杂的业务逻辑.由于 Web 服务之间的数据、控制和业务等多依赖关系,使得业务流程中的事务依赖并不是由提供商决定.因此,为了便于运行过程中的主动监测和可靠性维护,必须首先得到原子事务的范畴.由于 Web 服务本身的自治性,完成一个功能的多个 Web 服务可能属于一个原子事务,也可能不属于一个原子事务.这就需要根据服务间存在的多关系来识别原子事务.

3.1 事务粒度

为了获取原子事务服务,需要事先获取组件服务间存在的依赖关系.基于事务 Web 服务在执行过程中的状态和组件服务间存在的多关系,包括控制关系、数据关系和事务关系,可以获取任意直接相连的服务之间的补偿依赖关系.并基于此,得到间接相连的服务间的补偿依赖关系,最终得到原子事务.

定义 5. 直接补偿依赖关系.业务流程 BPM 在执行过程中,某个任务 tws_i 发生中断,如果任务 tws_i 和任务 tws_j 之间存在以下关系,称任务 tws_i 和任务 tws_j 之间存在补偿依赖关系,简记 $CD(tws_i,tws_j)$.

(1) 当任务 tws_i 和任务 tws_j 之间是顺序关系时,即 $CR(tws_i,tws_j)=$"Sequence",如果满足 $BR(tws_i,tws_j)=1$ 或 $DR(tws_i,tws_j)=1$,我们说任务 tws_i 和任务 tws_j 之间存在补偿依赖关系:$tws_i \prec_{\text{compensation}} tws_j$,即 $CD(tws_i,tws_j)=1$;

(2) 当任务 tws_i 和任务 tws_j 之间是并列合并关系时,即 $CR(tws_i,tws_j)=$"And-Joint",如果满足 $BR(tws_i,tws_j)=1$,则我们说任务 tws_i 和任务 tws_j 之间存在补偿依赖关系:$tws_i \prec_{\text{compensation}} tws_j$,即 $CD(tws_i,tws_j)=1$;

(3) 当任务 tws_i 和任务 tws_j 之间是并列分离关系时,即 $CR(tws_i,tws_j)=$"And-Split",如果满足 $BR(tws_i,tws_j)=1$,并且 $s(tws_j)=$"completed",则认为任务 tws_i 和任务 tws_j 之间存在补偿依赖关系:$tws_i \prec_{\text{compensation}} tws_j$,即 $CD(tws_i,tws_j)=1$;

(4) 当任务 tws_i 和任务 tws_j 之间是选择连接或选择分离关系时,即 $CR(tws_i,tws_j)=$"Or-Joint"或 $CR(tws_i,tws_j)=$"Or-Split",则说任务 tws_i 和任务 tws_j 之间不存在补偿依赖关系,即 $CD(tws_i,tws_j)=0$.

由此可见,直接相连的任务之间存在明显的补

偿依赖关系.间接相连的任务之间也可能存在补偿依赖关系.这种补偿依赖关系是隐含的.例如任务 A 和任务 B 存在业务合作关系,任务 B 和任务 C 存在业务合作关系.基于事务的原子性和数据一致性,如果任务 C 失效,任务 A 就会级联失效.获得这种间接依赖关系,就能很好地识别级联失效的服务范围.确定级联失效的服务范围也是支持补偿替换模型中的关键问题.下面给出间接补偿依赖关系的定义.

定义 6.　间接补偿依赖关系.给定任务 tws_i、tws_j 和 tws_k 之间的控制关系,如果任务 tws_i 和 tws_j 之间、任务 tws_j 和 tws_k 之间存在补偿依赖关系,那么间接相连任务 tws_i 和任务 tws_k 之间存在以下依赖关系,即 $ICD(tws_i,tws_k)=1$:

(1) 如果 $CR(tws_i,tws_j,tws_k)=$"Sequence",并且 $CD(tws_i,tws_j)=1$,$CD(tws_j,tws_k)=1$,当任务 tws_k 发生中断时,根据直接补偿依赖关可以推出:$(tws_i \prec_{\text{compensate}} tws_j)\wedge(tws_j \prec_{\text{compensate}} tws_k)\Rightarrow tws_i \prec_{\text{compensate}} tws_k$;即 $ICD(tws_i,tws_k)=1$;

(2) 如果 $CR(tws_i, tws_j)=$"And-Joint",$CR((tws_j\wedge tws_j),tws_k)=$"Sequence",并且 $CD(tws_i,tws_j)=1$,$CD(tws_j,tws_k)=1$,当任务 tws_k 发生中断时,根据直接补偿依赖关可以推出:$(tws_k \prec_{\text{compensate}} tws_j)\wedge(tws_j \prec_{\text{compensate}} tws_i)\Rightarrow tws_k \prec_{\text{compensate}} tws_i$;即 $ICD(tws_i, tws_k)=1$;

(3) 如果 $CR(tws_i, tws_j)=$"Or-Joint",$CR((tws_i\vee tws_j),tws_k)=$"Sequence"或 $CR(tws_i, tws_j)=$"Or-Split",$CR((tws_j\vee tws_j),tws_k)=$"Sequence",因为任务 tws_i 和任务 tws_j 是互斥关系,这种关系与定义 5 的第 4 种情况相同.

(4) 如果 $CR(tws_i, tws_j)=$"And-Split",$CR(tws_k,(tws_i\wedge tws_j))=$"Sequence",并且 $CD(tws_i, tws_j)=1$,$CD(tws_j,tws_k)=1$;任务 tws_i 与任务 tws_k 之间存在以下两种补偿依赖关系:

(a) 当 $s(tws_i)=$"completed"时,根据 $(tws_k \prec_{\text{compensate}} tws_j)\wedge(tws_j \prec_{\text{compensate}} tws_i)\Rightarrow tws_k \prec_{\text{compensate}} tws_i$;即 $ICD(tws_k,tws_i)=1$;

(b) 当 $s(tws_i)=$"initial",不需要进行补偿.

以上间接补偿依赖关系可以用图 4 描述.图 4(a)是定义 6 的第 1 种情况,表示在 Sequence 关系下,任务 tws_k 失效会导致间接任务 tws_i 失效.图 4(b)是定义 6 的第 2 种情况,表示在 And-Joint 关系下,任务 tws_k 的失效会导致间接任务 tws_i 失效.图 4(c)和图 4(e)是定义 6 的第 3 种情况,表示在 Or-Joint 或 Or-Split 关系下,这种关系与定义 5 的第 4 种情况.图 4(d)是定义 6 的第 3 种情况等等.综上可知,所有任务之间的补偿依赖关系都可以由上面的基本服务补偿依赖关系和级联服务补偿依赖关系得到.

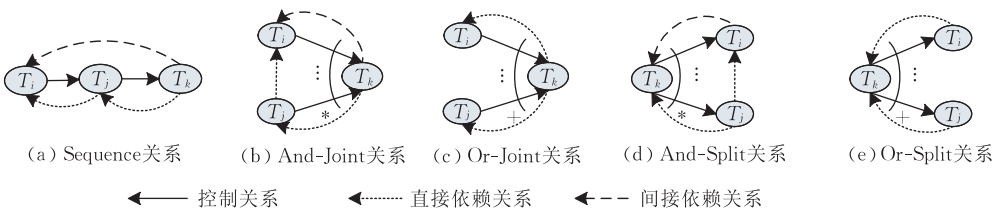


图 4　间接补偿依赖关系图

算法 1.　事务粒度识别算法 TGIA.

输入:服务操作集 $OP=\{\langle op_1,op_2,\cdots,op_n\rangle\}$,多关系库 $R=\{BR,CR,DR\}$

输出:事务粒度集 TG

算法描述:

1. 初始化:$TG=\varnothing$,$TG_i=\varnothing$,$tg=\varnothing$,$i=1$
2. while $i<n$ do
3. 　对任意相邻的操作对 $op_k=\{op_i,op_j\}$ //利用定义 5 和定义 6
4. 　if $R\{op_i,op_j\}\in\{BR, DR\}$ then //两个操作间存在业务关系或数据关系
5. 　　$tg=\{op_j,op_i\}$ //$op_j \prec op_i$
6. 　　$TG_k=TG_k\cup tg$ //基本粒度添加到 TG_k 中
7. 　　$OP=OP-TG_k$, $i++$
8. 　end if
9. 　return $\sum TG_k$ //输出所有原子事务
10. end while
11. 重复步 2～9
12. return TG //输出所有粒度的事务

基于以上定义,可以得到事务粒度识别算法.算法的过程如算法 1 所示,由于篇幅限制,该算法将不做详细描述.有了基本事务,就可以识别出所有嵌套事务,因为嵌套事务可以由原子事务得到,嵌套事务可以看做是多个原子事务的有意义封装,可以看成一个特殊的活动.基于该算法可以识别出图 1 业务流程中某具体实例的事务服务粒度.$BP=\{NTS_1, ATS_1, \langle NTS_2\rangle\}=\{\langle tws_1, tws_2\rangle, tws_3, \langle tws_4, tws_5\rangle\}$,$tws_1$ 和 tws_2 属于嵌套事务(NTS_1),tws_4 和 tws_5 属于另一个嵌套事务(NTS_2).如图 5 所示的是

一个层次的嵌套事务，用短虚线包围的部分代表原子事务，如 a,b,c,d,e；用长虚线包围的部分代表嵌套事务，如 A,B 所示. 进一步，可以推导出所有的嵌套事务.

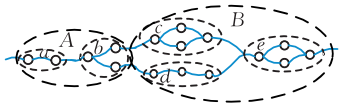


图 5 原子(嵌套)事务

3.2 补偿服务路线生成算法

补偿路线是异常结点及其事务依赖关系的集合，是异常处理器进行补偿时的执行路线. 然而，服务执行的动态性增加了根据存在的依赖关系生成补偿服务路线的困难，所以必须根据每个结点的工作日志，记录其执行时的前驱和后继结点. 考虑到事务服务之间的多种控制模式，如顺序模式(Sequence)、并列分支(And-Split)模式，并列合并(And-Joint)模式、选择分类(Or-Split)模式、选择合并(Or-Joint)模式等以及数据关系、业务关系等多关系，识别各种控制模式下服务与其直接前驱和直接后继服务间的补偿依赖关系，并由此识别出间接相连服务间的补偿依赖关系，并建立异常结点及其依赖的事务集的补偿路线，采用 XML 来保存记录并交换数据，详细过程如算法 2 所示.

算法 2. 补偿服务生成算法 CSGA.

输入：执行实例 $EE=\{\langle s_1,t_1\rangle,\langle s_2,t_2\rangle,\cdots,\langle s_i,t_i\rangle\}$，多关系库$\{BR,CR,DR\}$

输出：补偿服务生成图 CSG_1

算法描述：

1. 初始化：$CSG_1=\varnothing$，$csg=\varnothing$
2. while $EE\neq\varnothing$ do
3. for each $s_k=\{s_i,s_j\}\in$ {And-Joint}或{And-Split} //两个服务是并行关系
4. if $R\{s_i,s_j\}\in\{BR,DR\}$ then //两个并行的服务间存在业务关系或数据关系
5. if $t_i\geq t_j$ then
6. $csg=s_j\prec s_i$
7. end if
8. else $csg=s_i\prec s_j$
9. $CSG_1\leftarrow csg$，$EE=EE-\{s_i,s_j\}+s_k$ //将补偿关系插入到补偿生成图中，并标记
10. end if
11. for each $s_t=\{s_i,s_j\}\in$ {Sequence}
12. if $R\{s_i,s_j\}\in\{BR,DR\}$ then
13. $csg=s_j\prec s_i$
14. $CSG_1\leftarrow csg$，$EE=EE-\{s_i,s_j\}+s_t$ //将补偿关系插入到补偿生成图中
15. end if
16. return CSG_1 //输出补偿生成图

算法 2 给出了补偿服务生成算法的伪码描述过程. 首先，给出服务执行计划，$EP=\{\langle s_1,t_1\rangle,\langle s_2,t_2\rangle,\cdots,\langle s_i,t_i\rangle\}$ 和多关系库$\{BR,CR,DR\}$，其中，s_i 表示服务实例，t_i 表示该服务实例的预计开始执行时间. 首先初始化补偿服务图为空(第 1 行)；对执行实例中的并行服务结点根据预计的执行时间和服务之间的多关系，识别所有并行执行服务的补偿关系，加入到补偿图中，并在原始执行计划对已经生成的补偿路径的服务做标记(步 3～10)；进一步，对执行实例中的顺序服务结点根据执行顺序和服务之间的多关系，识别所有顺序执行服务的补偿关系，加入到补偿图中，并在原始执行计划对已经生成的补偿路径的服务做标记(步 11～15)；递归整个过程，获得所有的服务补偿路线. 本研究主要针对顺序执行和并发执行的服务构建补偿服务路径，其它的如循环等执行结构，也可以由此推导出. 基于得到的补偿路线，以下将详细介绍如何实现支持事务补偿机制的 QoS 代价/收益分析.

4 支持事务补偿机制的 QoS 代价/收益模型

如前所述，当组合服务发生异常需要进行替换选取时，为了实现正确替换，需要采用补偿机制消除已执行结果造成的影响，即需要考虑到服务所属的事务以便更好支持服务替换. 为了实现开销较小的替换，需要对候选服务进行替换代价分析. 替换代价主要包含最小回滚补偿的代价和替换重选取的代价.

4.1 事务级补偿代价

本节首先给出补偿服务 QoS 的基本定义，进一步给出事务级服务替换模型.

定义 7. 补偿服务 QoS. 给定补偿服务集 $CTWS=\{ctws_1,ctws_2,\cdots,ctws_n\}$，对于任意服务 $ctws_i$，其补偿服务 QoS 记为 $CQoS(ctws_i)=\{Q_{\text{price}}(ctws_i)\times t_i,Q_{\text{time}}(ctws_i)\}$，其中 $ctws_i$ 代表服务 tws_i 所对应的补偿服务，$Q_{\text{price}}(cwss_i)\times t_i$ 代表在时刻 t_i 执行补偿服务 $ctws_i$ 的花费代价，$Q_{\text{time}}(ctws_i)$ 代表执行补偿服务 $ctws_i$ 的花费时间.

跟据不同的业务领域，其补偿代价是不同的. 通常，触发补偿服务的当前时间与补偿所需得代价是有一定的映射关系的[13]. 本文以订机票服务(BTicket)为例，退订机票补偿服务(CBTicket)的补偿代价为与执行退票服务的时间有如下映射关系：

$Q_{\text{price}}(\text{CBTicket})=\{f_i(Q_{\text{price}}(\text{CBTicket})),t_i\}$，即当普通用户提前半个月退定机票时，其补偿代价为20%的机票价钱；提前3天退定机票时，其补偿代价为50%的机票价钱等等. 另外，考虑到用户身份，比如对VIP用户、会员用户和普通用户的退票费用是不同的. 表1列举了针对不同类用户、分别于不同时间(30天前、15天前、3天前、0天前)补偿服务的代价映射关系，说明补偿服务QoS计算是随着业务领域的不同而不同的.

表1　预定机票的补偿服务例子

用户识别	补偿代价			
	30天	15天	3天	0天
VIP用户	无	无	无	20%
会员用户	无	10%	20%	50%
普通用户	无	20%	50%	100%

在真实生活中，当某个服务需要补偿时，为了保持事务数据的一致性，可能引发间接管理的服务随之补偿，考虑到服务间的各种控制关系，定义了级联服务QoS.

定义8.　级联补偿服务QoS. 某服务引发需要补偿的级联服务记为$CTS=\{\sum CTWS,\sum CR,\sum DR\}$，$\sum CTWS$表示受失效服务影响的级联补偿服务集，$\sum CTWS=\{ctws_1,ctws_2,\cdots,ctws_n\}$，根据补偿服务间的各种控制关系($\sum CR$)和数据依赖关系($\sum DR$)，可以得到级联补偿服务QoS，记为$C_{\text{QoS}}$，如下所示

$$C_{\text{QoS}}(\sum CTWS)=\{\sum Q_{cp}^t,\sum Q_{ct}\}.$$

其中，$\sum Q_{cp}^t$表示执行级联补偿服务在t时刻执行补偿所需的代价之和，$\sum Q_{ct}$代表执行级联补偿服务所需的执行时间之和. 本文列举了几种控制关系下的级联补偿服务计算公式，参见表2所示.

表2　级联补偿服务代价分析

	Sequence	And-Joint	Or-Joint	And-Split	Or-Split
价格	$\sum_{i=1}^{n}Q_{\text{price}}(ctws_i)\times t_i$	$\sum_{i=1}^{n}Q_{\text{price}}(ctws_i)\times t_i$	$\sum_{i=1}^{n}Q_{\text{price}}(ctws_i)\times t_i$	$\sum_{i=1}^{n}(Q_{\text{price}}(ctws_i)\times s(tws_i))\times t_i$	
时间	$\sum_{i=1}^{n}Q_{\text{time}}(ctws_i)$	$\max_{i=1}^{n}(Q_{\text{time}}(ctws_i))$	$\sum_{i=1}^{n}Q_{\text{time}}(ctws_i)$	$\max_{i=1}^{n}(Q_{\text{time}}(ctws_i))$	$\sum_{i=1}^{n}Q_{\text{time}}(ctws_i)$

4.2　支持补偿的替换代价/收益模型

不同于以往替换服务选取模型，该模型不仅支持事务补偿，而且支持事务补偿的替换重选取，全面、综合地分析了替换服务QoS，在保证系统正确运行的前提下，使得整体替换代价最小，即效用函数(1)的值最大.

$$Score_k(tws_i)=Max[Q_m^t(CTWS)+Q_n^k(TWS)] \tag{1}$$

式(1)是支持补偿的替换服务代价函数. $Q_m^t(CTWS)$代表t时刻的补偿代价，$Q_n^k(TWS)$代表选择第k条替换路径的替换代价，m,n分别表示补偿服务的参数个数和替换服务的参数个数.

为了统一不同质量的量纲，在计算替换代价之前需要对补偿服务和事务服务作标准化处理，针对增量型的质量，其标准化公式如式(2)所示，针对减量型的质量，其标准化如式(3)所示

$$\begin{cases}V_x(TWS)=\dfrac{QoS^x(tws_i)-QoS^x_{\min}(tws_i)}{QoS^x_{\max}(tws_i)-QoS^x_{\min}(tws_i)},\\ \qquad\qquad x\in price,time\\ QoS^x_{\max}(tws_i)-QoS^x_{\min}(tws_i)\neq 0\end{cases} \tag{2}$$

$$\begin{cases}V_x(TWS)=\dfrac{QoS^x_{\max}(tws_i)-QoS^x(tws_i)}{QoS^x_{\max}(tws_i)-QoS^x_{\min}(tws_i)},\\ \qquad\qquad x\in available,\cdots\\ QoS^x_{\max}(tws_i)-QoS^x_{\min}(tws_i)\neq 0\end{cases} \tag{3}$$

由此，进一步得到t时刻选择第k条替换路径的总体替换代价计算式(4)：

$$Score_k(tws_i)=Max\left\{\sum_{\alpha=1}^{m}W_\alpha V_\alpha^t(CTWS)+\left(\sum_{\beta=1}^{n}W_\beta V_\beta(TWS)\right)^k\right\}=$$

$$Max\left\{\frac{\sum_{\alpha=1}^{m}\left(W_\alpha\times\sum_{i=1}^{rl}V_\alpha^t(ctws_i)\right)+\sum_{\beta=1}^{n}\left(W_\beta\times\sum_{j=1}^{rl+fl}V_\beta(tws_j)\right)^k}{\sum_{\alpha=1}^{m}w_\alpha+\sum_{\beta=1}^{n}w_\beta}\right\},$$

$$\sum_{i=1}^{rl}V_\alpha^t(ctws_i)\leq RLA^\alpha;\sum_{i=1}^{rl+fl}V_\beta(tws_i)\leq RLA^\beta \tag{4}$$

注：补偿代价QoS和替换服务选取QoS计算参数是不同的，本文将补偿服务系数计为W_α；补偿服务代价包括执行费用和执行时间. 替换服务系数计为W_β，替换服务代价计算函数包括执行费用、执行时间、

执行成功率、可靠性和可用性等. $\sum_{\alpha=1}^{m} W_\alpha V_\alpha^t (CTWS)$ 代表补偿质量，$\left(\sum_{\beta=1}^{n} W_\beta V_\beta (TWS)\right)^k$ 表示第 k 条替换路径的服务质量，rl 表示级联回滚的补偿服务个数，$rl+fl$ 表示替换重选取的服务个数，补偿服务代价和替换重选取代价分别满足用户定义的约束条件 RLA^α 和 RLA^β. 可以看出，rl 越小，需要回滚补偿的服务越少，所需的补偿代价就越少，则目标函数就是使整体替换代价最小，包括最小补偿代价和最小替换代价，即式(1)质量函数 $Score_k(tws_i)$ 最大. 下面通过实验对本文提出替换模型性能进行评估分析.

4.3 面向自适应的事务级服务替换算法

本小节将给出事务支持的服务替换算法. 为了验证该算法的正确性，首先给出事务支持的服务替换过程正确性证明. 当选择的服务能满足用户的功能需求时，替换过程正确性即是替换过程不破坏事务服务的原子性和数据一致性，详见定理 1.

定理 1. 支持事务补偿机制的服务替换不破坏被替换服务的事务原子性和数据一致性，即能够保证替换过程的正确性.

证明. 因为替换过程的正确性是通过事务服务的原子性维护和数据的一致性维护体现的. 因此，只需证明被替换服务仍然保持原子性和数据一致性即可.

给定具有补偿属性的事务服务 tws，包含三个操作 op_a、op_b 和 op_c；在运行过程中，op_a 和 op_b 已经成功提交，当执行到 op_c 时，op_c 发生故障，准备使用候选 $op_{c'}$ 替换 op_c. 基于事务服务的状态(见定义 2)和事务服务具有的行为属性约束(见定义 1)，在替换之前，需要完成以下操作.

(1)维护事务服务的原子性(要么全做，要么一个都不做). 该过程是通过执行 op_a 和 op_b 的逆操作(补偿操作 op_a^c 和 op_b^c)实现的，该操作使服务恢复到执行前的状态，维护了服务的原子性.

(2)同时，补偿操作(op_a^c 和 op_b^c)撤销了提交服务对数据库造成的影响，使数据恢复到初始状态，维护了数据的一致性(事务在完成时，必须使所有的数据都保持一致状态. 事务未能完成时，必须使所有的数据都恢复到初始状态，以保持所有数据的完整性).

综上，若直接替换服务，会造成已经完成服务的提交结果依然驻留在服务器中，破坏了数据一致性和原子性. 由此得出，支持事务的服务替换能保证替换前后数据一致性和原子性，进而保证了替换过程的正确性. 证毕.

进一步，本文给出了支持事务补偿的服务替换算法. 支持事务补偿的服务替换算法由三部分组成：首先，判断失效服务是否需要补偿；如果需要补偿，则根据服务间的控制关系、数据关系、业务关系等多关系，识别出受失效服务影响的最小补偿波及范围；最后，根据匹配的接口服务行为模式确定最小替换范围并建立支持补偿的替换代价分析，重选取最优的替换服务. 详细过程参见算法 3.

算法 3. 支持事务补偿的服务替换算法.

输入：执行计划 EP，组合服务图 CSG，失效服务结点 tws_i

输出：选取的替换服务

算法描述：

```
1. for 对失效结点 tws_i
2. 标记与之相连的结点和边
3. if tws_i需要补偿 then
4. 找到失效服务对应的所有前驱服务 PreTWS
5. 根据 PreTWS 中服务的直接补偿依赖 CD//定义 5
6. 计算间接补偿依赖 ICD//定义 6
7. if 对应的需要补偿的服务不空 then
8. 调用 CSGA 算法以确定失效服务的补偿路线图
                                    //级联回滚范围
9. endif
10. 以接口匹配的服务集为起点，
11. 构建组合服务中间图
12. 计算支持补偿的替换代价分析函数 Score_k(tws_i)
13. 重选取最优的替换路径进行替换
14. endif
15. endfor
```

以上给出了算法的伪码描述过程. 首先，识别失效结点并标记与失效结点相连接的结点和边(第 1～2 行)；第二，判断失效服务是否需要补偿，如果需要补偿，通过定义 5 和定义 6 得到最小需要补偿的级联服务范围(第 3～9 行)；第三，构建对应的中间子图，并基于本文的前期工作[18]，得到满足条件的行为匹配替换服务集合，最后计算替换代价分析函数，得到最优的满足条件的替换服务(第 10～13 行). 由于空间限制，本节省略掉部分细节的描述. 下面给出实验分析.

5 基于早期预测模式的提前替换触发机制

为了减少故障服务的中断时间，提出了基于早期预测模式的替换启动机制，该机制通过模式匹配能主动伺机地启动服务替换选取. 事务属性是非常重要的一类 QoS 属性. 然而，已有工作大都基于常

见的服务 QoS，没有考虑事务属性. 已有的预测方法[19-21]也大都基于常见 QoS 的监控，缺乏事务级 QoS 监控. 因此，已有方法没有很好地应用到这一 QoS 属性. 更重要的是，已有的预测方法在服务执行的生命周期都启动预测，却忽视了预测点的选取问题. 实际应用中，在所有服务点都启动预测是代价大的，也是不可行的. 大大浪费了资源. 不同于已有预测方法[19-21]，本文提出的方法通过异常模式自学习机制保证当最早出现的异常模式下才启动预测并进行替换触发，不必在所有执行结点都启动预测，在保障时效性的前提下，大大减少了全程启动预测的代价和服务中断的时间. 下面首先给出早期预测模式的相关定义.

定义 9.　早期预测模式. 给定异常服务序列集合 ES，$P=\{s_1,s_2,\cdots,s_l\}\subseteq ES$ 代表一个或一系列序列时，我们称 P 为模式或模式集合，记为 Pattern，简称模式 P，Fe 是模式 P 的特征集合(如运行时间、运行代价、成功率、延迟时间等)，当模式 P 存在前缀集 p，记为 $p=minprefix(P)$，若满足 $U(p,Fe)>\delta$，δ 是用户给定的阈值. 我们说模式 p 是 P 的最早预测模式.

最早预测模式通过简单的模式匹配机制就可以识别服务执行序列从安全区进入临界区，进行预警. 安全区即服务在该环境下能很好地满足执行用户需求，运行良好；临界区表示服务执行序列在将来时刻会偏离正常的服务质量. 当满足条件 $U(p,Fe)>\delta$ 时，启动预测机制，进行替换服务判定，进而降低全程启动预测导致的大代价问题.

为了得到早期预测模式，需要对组合服务执行序列进行自学习. 首先，对给出的序列模式集进行数据预处理，将序列数据转换成可挖掘的序列数据，对该序列数据进行频繁模式挖掘，频繁模式挖掘方法不是本文的研究重点，可以参照项目组前期基础[22]，挖掘出的频繁模式作为抽取的特征模式添加到预测模式集中. 本文着重如何实现早期准确的预测模式挖掘. 为了得到早期预测模式，本文对所有的频繁模式，记录其早出现在对应序列模式的位置，计算权重支持度，如定义 10 所示.

定义 10.　早期模式置信度. 给定子模式 $p\rightarrow fail$，$fail$ 表示运行失败，子模式 p 对应的序列为 P_i，对所有的包含该模式的集合 $SDB(p)$，记录该模式对应的每个记录长度和模式的最早出现的位置，记为 pre，它的权重支持度计算式(5)如下

$$supp(SDB(p))=\sum_{i=1}^{n}\frac{\dfrac{length(p)-pre(p)+1}{length(P_i)}}{num(P_i)}supp(p),\ p\in P_i \tag{5}$$

例如：给定两个模式集合库分别为 $SDB_1(s_2,s_3)=\{\langle s_1,s_2,s_3\rangle;\langle s_2,s_3,s_4\rangle;\langle s_0,s_1,s_2,s_3\rangle\}$ 和 $SDB_2(s_2,s_3)=\{\langle s_0,s_1,s_2,s_3\rangle;\langle s_2,s_3,s_4,s_5\rangle;\langle s_0,s_1,s_2,s_3,s_4\rangle\}$，如果用传统的频繁模式挖掘，可得到模式 s_2,s_3 在两个模式库中的支持度都是 1，而由式(5)可知，模式 s_2,s_3 在 SDB_1 中的权重支持度为 0.72，模式 s_2,s_3 在 SDB_2 中的权重支持度为 0.63，这说明模式 s_2,s_3 在 SDB_1 的早期性更明显. 因此，权重支持度能很好地反应模式的早期性.

进一步，对抽取出的所有特征模式，利用熵理论知识，计算对应数据库的熵值 $E(SDB)$ 和包含该特征模式的熵值 $E(SDB_p)$，计算信息量. 最后，得到综合的效用函数，如下式(6)所示：

$$U(p,Fe)=(E(SDB)-E(SDB_p))^{\omega}supp(SDB(p)) \tag{6}$$

在预测过程中，对所有的模式特征进行分类学习，当组合服务执行序列与预测模式中的部分模式匹配时，进行预测并提前触发替换选取服务机制，当服务出现故障时，直接进行替换，减少或避免了由于出错而导致的服务中断时间. 由于空间限制，忽略了该算法的详细描述.

6　实验分析

本文提出了事务级主动伺机服务替换算法. 为了评估提出算法的效率和可靠性，设计了相关模拟环境，分别从算法的性能和可靠性方面对相关算法进行分析比较. 首先模拟生成网络环境，采用 BRITE[23] 网络拓扑生成工具产生网络拓扑图，根据该网络拓扑图采用 NS-2 网络模拟软件计算服务节点间的消息传输时间；服务数据采用人工生成的组合服务，每个服务包含服务名称、服务类别、服务所在网络节点、服务操作、补偿服务以及补偿目标代价(补偿服务采用随机添加)，参数接口等基本信息，生成 JAX-WS 服务，采用数据接口与语义匹配的方式随机组合服务，得到若干服务组合序列. 采用压力测试工具模拟 Web 上多个用户的并行执行情况从而得到真实的服务执行日志. 实验参数为采集 Web 服务 400 个，属于 50 个服务类，随机分布在 400 个网

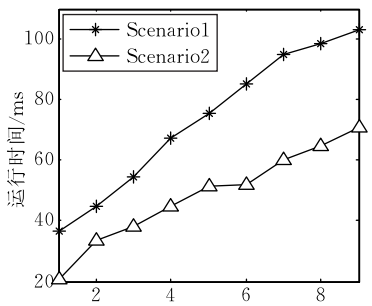
络节点中;日志采集时间长度为 10 周,系统按照给定的执行频度随机选取服务组合模型进行执行.

6.1 性能分析

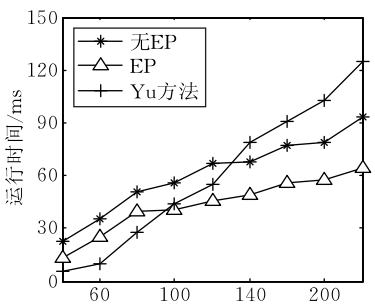
性能分别包括算法的效率分析和替换的时效性分析.其中,算法效率指的是算法对业务流程的响应时间;替换时效性指的是当有服务需要被替换时,算法响应服务替换的最早时间.本节分别对算法的效率和时效性进行比较分析,分别评估了三个算法:支持事务补偿的主动伺机替换算法(EP)、支持事务补偿但忽略早期预测的替换算法(无 EP)和 Yu 的算法的性能进行分析.实验分析结果如图 6 所示.图 6(a)和图 6(b)分别为对不同回滚长度和不同组合服务数目的响应时间的比较分析;图 6(c)为随不同故障位置点的最早启动替换响应时间的比较分析.

图 6(a)显示的是主动伺机替换算法(EP)在不同回滚范围下的执行时间比较.图 6(a)中的 Scenario1 和 Scenario2 分别代表服务总数为 50 的组合服务和服务总数为 100 的组合服务,可以看出,在相同数据集大小的条件下,回滚长度越大,替换时间越多,但是替换时间波动几乎呈线性变化.这说明算法的可伸缩性好.

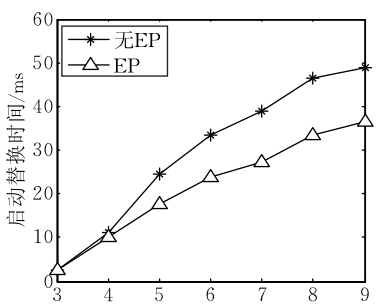
图 6(b)显示的是在数据集(服务结点数分别为 40～200)变化的情况下,针对本文提出的支持早期预测的服务替换算法(EP)、不支持早期预测的服务替换算法(无 EP)和 Yu[6]算法中替换时间比较分析图.设置补偿支持替换算法的回滚长度为 4.可以看出,当运行的服务较少时,补偿支持替换算法的运行时间稍长,当运行的服务较多时,本文提出算法的运行时间明显优于 Yu 提出的算法的运行时间,说明本文提出的算法具有较好的可应用性.



(a) 替换时间与回滚长度的关系



(b) 不同算法的运行时间



(c) 不同故障点的启动替换时间

图 6 算法性能比较分析

为了体现早期预测的优势,提出了启动替换时间随故障点位置变化的曲线图,如图 6(c)所示.即在故障点变化的情况下,对本文提出的早期预测的服务替换算法(EP)和不支持早期预测的服务算法(无 EP)的替换启动时间比较图.当故障点接近起始点时,有 EP(有早期预测)和无 EP(早期预测)算法的启动预测时间很接近,因为预测点和运行的点距离很近,当故障点与起始点相距较远时,有 EP(早期预测)算法的启动时间明显优于没有 EP(早期预测)的替换启动时间.实验说明提前启动预测,能提前启动替换,可以大大减少服务的中断时间,说明该早期预测的替换触发策略是有效的.

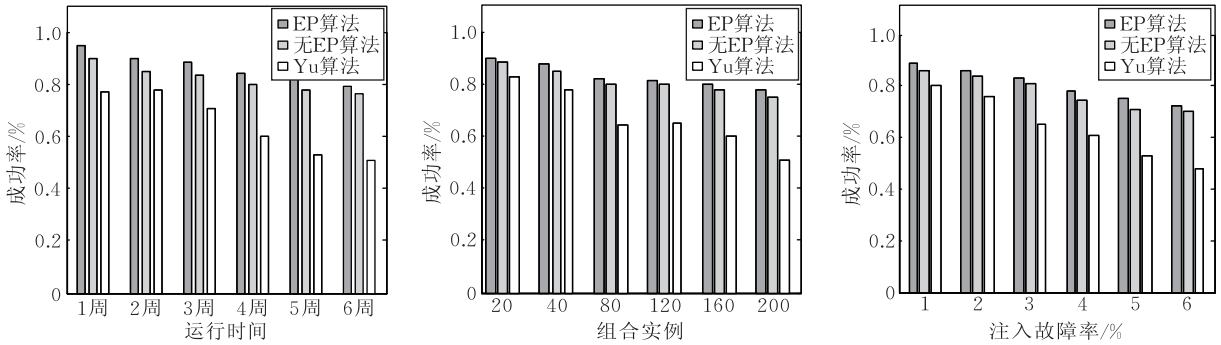
6.2 可靠性分析

算法的可靠性指的是成功执行的业务流程数目(包括替换之后成功执行的业务流程)与总任务(业务流程数目)的比值.本小节分别使用支持事务补偿的主动伺机替换算法(EP)、支持事务补偿但忽略早期预测的替换算法(无 EP)和 Yu 的算法对可靠性进行分析.实验分析结果如图 7 所示.图 7(a)数据集中包含 100 个服务,大约 15000 条记录的组合服务实例,考察一段时间(从 1 周～6 周)的可靠性;图 7(b)给出多个数据集(服务结点数变化范围从20～200)运行 6 周的可靠性分析.图 7(c)显示的是设置注入故障服务率时,运行一段时间的平均可靠性分析.

可以看出,支持补偿的主动伺机服务替换算法的替换成功率优于不支持早期预测的替换算法成功率,更优于已有的替换算法,说明该算法比单纯的服务替换算法更加健壮和可靠,不支持补偿的服务替换算法当需要补偿时则变得不可用,大大降低了可应用性.

6.3 一致性分析

本节针对事务数据一致性方面,提出了针对失效服务数据一致性的评价方法及实验.如前文所述,服务在其生存周期所处的状态集合为:States={initial, active, failed, completed, aborted, cancelled}.在服务的执行过程中,监控服务的运行状态并记录



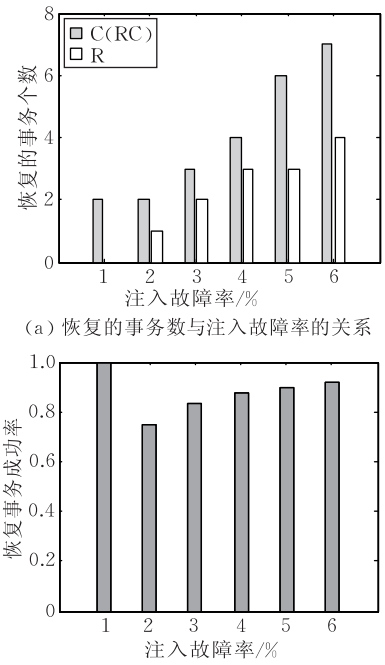
(a) 成功率与运行周期的关系　(b) 成功率与组合实例的关系　(c) 成功率与故障率的关系

图 7　算法可靠性分析

状态信息，如 1 表示初始态(initial)，2 表示运行态(active)，3 表示失败态(failed)，4 表示完成态(completed)，5 表示中止态(aborted)，6 表示取消态(cancelled). 因为事务可能是单个原子服务，也可能是多个组件服务的集合. 因此，已知服务运行状态可以进一步获取不同时刻对应事务的运行状态信息.

另外，根据具有事务属性(可补偿，可重做等)的失效服务的状态变化，考察该服务以及对应事务是否能通过补偿机制正确回滚到初始态(initial)或通过重做机制达到完成态(completed)来考察事务服务的数据一致性问题. 具体实现过程如下：对运行中的服务通过注入故障，监控其状态并记录状态变化情况. 当系统监控到某个失效服务时，首先，确定失效服务所属事务及在该事务中的位置；其次，获取失效服务的事务属性，是否可以重做或补偿. 如果失效服务具有可重做属性(或重做补偿属性)，则重新激活该服务并运行，最终成功完成或者因超时取消；如果失效服务具有可补偿属性，则逆向回滚事务中以该服务为起点的所有已完成服务. 实验结果如图 8(a)和图 8(b)所示. 其中 C(RC)表示的是通过补偿机制(或先重做后补偿机制)恢复数据一致性的事务个数；R 表示通过重做机制完成的事务个数. 当事务能通过补偿机制返回初始态(1)或通过重做机制达到完成态(4)时，说明该算法能很好地保证替换前后的数据一致性. 由于空间限制，忽略了详细算法.

图 8 考察的是服务结点数为 200 的组合服务在不同故障率(1%～6%)下一致性评估结果. 图 8(a)显示的是在不同故障率下恢复的事务个数；图 8(b)显示的是恢复事务的成功率. 通过实验可以看出，由于组合服务是在动态绑定的时候才确定事务范围，因而，可能会出现某个事务粒度的划分不符合某专



(a) 恢复的事务数与注入故障率的关系

(b) 成功率与故障率的关系

图 8　一致性保障评估

业背景下的业务逻辑. 当故障率低时，事务都能通过补偿操作正确恢复到初始状态；当故障率增大时，恢复的事务个数也趋于较稳定的趋势(75%～91%). 说明该方法通过监控失效服务的状态转换过程，可以对事务服务通过补偿或重做机制恢复到原始状态或完成状态，能很好地维护事务服务的数据一致性.

综上，可以得出，本文提出的基于早期预测的主动伺机服务替换算法在性能方面和可靠性方面优于已有的算法，在一致性评估方面也获得较好的效果，进一步证明了该算法的可靠性、有效性和正确性.

7　相关工作讨论

目前的替换模型有两种方式：面向功能的替

换[1-4]和面向组合服务 QoS 的替换[4-7]. 文献[2]认为具有相同接口参数的服务可以提供相似的功能,从而通过对接口参数语法和语义相似性的判断,发现可替换的服务;文献[3]提出了一个服务复制方法,在网络负载较大导致服务不可用的情况下,用复制服务替换原服务,提高服务的可用性并保证组合服务的 QoS;文献[4]提出了发现重新配置的服务组合方法,为组合流程中的每个服务根据语义等特性配置新的服务集,在组合流程中的服务发生失效的情况下用新配置的服务替换它;Yu 等人[5-6]提出基于两种失效服务替换算法,第 1 种方式是基于已有后备路径服务,一旦该服务的前躯服务失效,立刻转到预先定义的后备路径. 第 2 种方法是重构建一个跨越失效服务替换路径来替换. 文献[7-8]提出了以 QoS 全局优化为目标的组合服务替换方法,其核心是在发现某一服务发生 QoS 失效的情况下,中断组合服务的执行、对组合服务中所有未执行的服务进行重选取并替换未执行的服务,以保证组合服务的全局 QoS. 文献[21]提出监控 QoS,当发现组合服务 QoS 偏离目标太大时,启动替换服务重选取机制,选择最优的 QoS 进行替换. 以上方法对面向功能的服务替换和面向 QoS 的服务替换提出了有效的替换策略,在一定程度上保证了服务的可靠运行. 然而,这些方法大都基于传统 QoS(如时间、代价、可靠性、信誉度等)的服务替换,都缺乏对事务 Web 服务的支持[4-8]. 事务 Web 服务是保证组合服务可靠运行的根本保障[11-12],当服务出现故障时,尤其是面临一些复杂业务流程故障时,已有的直接替换方法[4-8]虽然在功能语义和 QoS 角度都能满足替换要求,却因忽视服务的事务属性,使得已经成功提交的数据结果依然驻留在服务器内存中,没有被释放,造成替换前后服务器数据的不一致. 在这种情况下,为了维护服务器数据一致性,系统就会提示运行错误,替换可能被中断,应用受到限制.

事务 Web 服务能够支持系统的可靠运行[11-13],补偿机制通常被用于事务 Web 服务中[12-13],用来消除已经执行的 Web 服务对系统的影响(比如释放空间),维护原子性. 越来越多广大研究学者关注事务服务的研究. 尤其在商业流程中,有了事务的支持,一旦组合 Web 服务中的某个服务出现故障而使整个流程无法运行时,系统就可以通过补偿操作[13-16]来维护事务的原子性和数据的一致性,使系统恢复到正确的状态. 如文献[12-13]定义了不同的补偿类型,研究如何调度服务来降低全局事务的补偿代价或设计了新的事务规范来更好地支持补偿操作. 文献[24]则提出使用 ATS 来表达放松的原子性,从而允许设计者定义不同级别的事务;在此基础上,文献[25]能够针对不同组合服务的可接受终止状态推出各个组件服务的事务属性,自动生成协调规则来协调服务的执行;文献[26]提出基于 Petri 网的可靠组合服务协调方法,根据服务的事务属性及服务组合的失效处理机制建立服务组合的失效处理模型,并构建组合协调策略;以上 Web 服务事务模型大都集中在如何更好地支持事务,如采取补偿机制撤销驻留在内存中的数据,维护事务一致性,但仍然存在着诸如忽略服务的事务支持等带来的不能保证替换正确性的一系列问题.

由此可见,已有的服务替换方法之所以不能很好地支持事务级服务替换,最大区别是没有充分考虑到事务 Web 的服务特性,忽视事务属性会造成替换前后由于服务器数据的不一致而导致的替换失败. 另外已有研究大都假设事务粒度已知,然而,实际应用中,服务是被动态绑定的,已有方法不能充分地反映真实事务粒度. 本文补充了这个不足,考虑了服务的事务属性,在保证补偿代价最小的前提下,选取优质的替换服务,实现可靠的组合服务替换是本文要解决的关键问题.

8 结 论

本文针对已有替换算法无法解决替换中事务服务的数据不一致导致替换失效的情况,提出了一种支持补偿的替换模型,该模型通过补偿机制能很好地处理在服务替换中由服务失效引起数据不一致的情况. 考虑服务间的多种控制依赖关系,快速识别事务粒度和失效服务的替换范围,并提出了全新的以事务为粒度支持补偿的替换 QoS 代价/收益模型. 进一步,基于提出的早期预测模式,能更高效地实现主动伺机替换算法. 一系列实验也进一步证明了该算法的高可靠性和有效性.

下一步,将研究如何设计更高效的事务级替换算法来处理更加复杂的应用环境,比如不仅考虑单任务失效,还应该考虑业务流程中更加复杂的情况,比如多个任务同时失效的事务级服务替换,来进一步提高动态自适应能力.

参 考 文 献

[1] Ge J, Hu H, Lu J. Service discovery and substitution according to inheritance of behavior with invariant analysis//Proceedings of the 8th IEEE/ACIS International Conference on Computer and Information Science. Shanghai, China, 2009: 971-976

[2] Taher Y, Benslimane D, Fauvet M C. Towards an approach for Web services substitution//Proceedings of the 10th International Database Engineering and Applications Symposium. Delhi, India, 2006: 166-173

[3] Zhang G, Liu L, Seshadri S, Bamba B, Wang Y. Scalable and reliable location services through decentralized replication//Proceedings of the 7th IEEE International Conference on Web Services. Los Angeles, USA, 2009: 632-638

[4] Zaremba M, Migdal J, Hauswirth M. Discovery of optimized Web service configurations using a hybrid semantic and statistical approach//Proceedings of the 7th IEEE International Conference on Web Services. Los Angeles, USA, 2009: 149-156

[5] Yu T, Lin K. Adaptive algorithms for finding replacement services in autonomic distributed business processes//Proceedings of the 7th International Symposium on Autonomous Decentralized Systems. Chengdu, China, 2005: 427-434

[6] Yu T, Zhang Y, Lin K. Efficient algorithms for Web services selection with end-to-end QoS constraints. ACM Transactions on the Web, 2007, 1(1): 139-164

[7] Wang P, Chao K. On optimal decision for QoS-aware composite service selection. Expert Systems with Applications, 2010, 37(1): 440-449

[8] Cao H, Feng X, Sun Y, Zhang Z, Wu Q. A service selection model with multiple QoS constraints on the MMKP//Proceedings of the 2007 IFIP International Conference on Network and Parallel Computing Workshops. Washington, DC, USA, 2007: 584-589

[9] Alrifai M, Dolog P, Balke W T, Nejdl W. Distributed management of concurrent Web service transactions. IEEE Transactions on Services Computing, 2009, 2(4): 289-302

[10] Zhang L, Zhang J, Hong C. Services Computing. Beijing: Tsinghua University Press, 2007

[11] Moreews F. A framework for building reliable distributed bioinformatics service repositories//Proceedings of the 7th IEEE International Conference on Web Services. Los Angeles, USA, 2009: 1018-1019

[12] Zhu Rui, Guo Chang-Guo, Wang Huai-Min. A scheduling algorithm for long duration transaction based on cost of compensation. Journal of Software, 2009, 20(3): 744-753(in Chinese)
(朱锐，郭长国，王怀民. 一种基于补偿代价的长事务调度算法. 软件学报，2009，20(3)：744-753)

[13] Michael S, Peter D, Wolfgang N. An environment for flexible advanced compensations of Web service transactions. ACM Transactions on the Web, 2008, 2(2): 1-36

[14] Andrews T, Curbera F, Dholakia H et al. Business process execution language for Web services version 1.1. Available at http://www.ibm.com/developerworks/library/specification/ws-bpel/,Feb,2007

[15] Luis F C, George C, Max F et al. Web services transactions specifications. Available at http://www.ibm.com/developerworks/library/specification/ws-tx/, Aug, 2005

[16] Arkin A, Askary S, Fordin S et al. Web service choreography interface (WSCI) 1.0, Available at http://www.w3.org/TR/2002/NOTE-wsci-20020808/, Aug, 2002

[17] Kavantzas N, Burdett D, Ritzinger G. Web services choreography description language (WS-CDL) 1.0, Available at http://www.w3.org/TR/2004/WD-ws- cdl-10-20041217/, Dec, 2004

[18] Yin Y, Zhang X, Zhang B. An efficient service substitution algorithm based on temporal composite behavior graph//Proceedings of the 6th Web Information Systems and Applications Conference. Xuzhou, China, 2009: 126-131

[19] Shao L, Zhang J, Wei Y, Zhao J, Xie B, Mei H. Personalized QoS prediction for Web services via collaborative filtering//Proceedings of the 5th IEEE International Conference on Web Services. Salt Lake City, Utah, USA, 2007: 439-446

[20] Shao Ling-Shuang, Zhou Li, Zhao Jun-Feng, Xie Bing, Mei Hong. Web Service QoS prediction approach. Journal of Software, 2009, 20(8): 2062-2073(in Chinese)
(邵凌霜，周立，赵俊峰，谢冰，梅宏. 一种 Web Service 的服务质量预测方法. 软件学报，2009，20(8)：2062-2073)

[21] Dai Y, Yang L, Zhang B. QoS-driven self-healing Web service composition based on performance prediction. Journal of Computer Science and Technology, 2009, 24(2): 250-261

[22] Zhang Ming-Wei, Wei Wei-Jie, Zhang Bin, Zhang Xi-Zhe, Zhu Zhi-Liang. Research on service selection approach based on composite service execution Information. Chinese Journal of Computers, 2008, 31(8): 1398-1411(in Chinese)
(张明卫，魏伟杰，张斌，张锡哲，朱志良. 基于组合服务执行信息的服务选取方法研究. 计算机学报，2008，31(8)：1398-1411)

[23] Tyan Hung-Ying. Design, realiation and evaluation of a component-based compositional software architecture for network simulation[Ph. D. dissertation]. Ohio State University, San Diego, CA, USA, 2002

[24] Bhiri S, Perrin O, Godart C. Ensuring required failure atomicity of composite Web services//Proceedings of the 14th International Conference on World Wide Web. Chiba, Japan, 2005: 138-147

[25] Montagut F, Molva R. Augmenting Web services composition with transactional requirements//Proceedings of the 4th International Conference on Web Services. Washington, DC, USA, 2006: 91-98

[26] Fan Gui-Sheng, Liu Dong-Mei, Chen Li-Qiong, Yu Hui-Qun. A coordination strategy for reliable service composition and its analysis. Chinese Journal of Computers, 2008, 31(8): 1445-1457(in Chinese)
(范贵生，刘冬梅，陈丽琼，虞慧群. 可靠服务组合的协调策略与分析. 计算机学报，2008，31(8)：1445-1457)

YIN Ying, born in 1980, Ph. D., lecturer. Her main research interests include service-oriented computing and data mining.

ZHANG Bin, born in 1964, Ph. D., professor, Ph. D. supervisor. His main research interests include service-oriented computing, data mining and information integration.

ZHANG Xi-Zhe, born in 1978, Ph. D., associate professor. His main research interests include service-oriented computing and data mining.

Background

Composite Web services always run in the highly dynamic and variational internet environment, so that it is often inevitable that one component service fails or becomes overload. Therefore, how to guarantee composite Web services reliably and correctly execute has become one of the major challenges in the area of Service-Oriented Computing (SOC). Specially, advanced transactional support is required to ensure the execution qualities of composite services. However, most existing service-replacing methods consider only service function or traditional QoS, such as time, available, successful and so on. The pure replacement strategy is incomplete, unreliable and infeasible, since it is absent of transaction support, especially for business process. This paper takes the "transaction support" as the core, proposes a novel QoS-driven service replacement approach with transactional support, gives a replacement trigger mechanism based on early prediction pattern, which identifies the failed or unavailable services as earlier as possible to enhance the real-time performance, and describes all the related algorithms.

This research is supported by the National High Technology Research and Development Program of China under grant No. 2009AA01Z122, the Fundamental Research Funds for the Central Universities under grants No. N090304006, and the National Natural Science Foundation of China under grants No. 61073062, 60903009, 60803026. One mission of these projects is to study the issues on adaptation and evolvement of composite service. Additionally, these projects are also interested in the issues on "service behavior pattern discovery" and "intelligence aggregation for service computing". Several related papers have been published or accepted by some journals or international conferences. This work focuses on active and opportunistic transactional service replacement for reliable execution which is the important part of the above projects.