第 25 卷第 3 期　　四川理工学院学报(自然科学版)　　Vol. 25　No. 3
2012 年 6 月　　Journal of Sichuan University of Science & Engineering(Natural Science Edition)　　Jun. 2012
文章编号:1673-1549(2012)03-0037-06　　DOI:10.3969/j.issn.1673-1549.2012.03.010

基于 B+树的 BPEL 流程异常处理机制研究

吴吉红, 高　辉

(辽宁大学信息学院, 沈阳　110036)

摘　要:Web 服务组合具有松耦合、自治性的特点,BPEL 规范中的异常处理机制也不完善,于是 BPEL 流程中的异常处理方法是研究的一个主要问题。针对该问题,在流程运行阶段处理异常的一些通用的策略的基础上,提出了基于 B+树的等价服务替换算法,分别阐述了 BPEL 中同步和异步调用 Web 服务产生异常的情况,最后构架了一个异常处理的系统框架原型。

关键词:B+树;BPEL;异常处理;QoS;等价服务替换

中图分类号:TP311　　**文献标识码**:A

引　言

在不断开发软件的过程中,开发人员越来越清楚地认识到合理的业务逻辑以及完整的处理手段对整个软件系统起着至关重要的作用。BPEL (Business Process Execution Language),即业务流程执行语言,是一种使用 XML 编写的编程语言,用于自动化业务流程,广泛使用于 Web 服务相关的项目开发中。BPEL 实现的 Web 服务组合具有松耦合,服务可重用性,递归组合方式等特性,异常和错误发生的概率将大大提高。然而传统业务流程执行语言,例如 BPEL2.0 只定义了异常的捕获方法和补偿机制,异常的处理措施由业务开发者自己定义,从而也将异常处理的复杂性推给了业务开发者[1]。

面对 BPEL2.0 规范缺乏异常处理的现状,为了增强系统的鲁棒性 ,提升用户的信心,学者们采用了多种方法来对服务组合中的异常进行处理。由于 Web 服务组合本身自治性、松耦合的特点,使得异常处理非常困难,所以早期学者们都集中在业务流程的设计阶段。

文献[2]介绍了一种 ECA 规则驱动的 BPEL 流程异常处理机制,它将 BPEL 流程和异常处理设计逻辑分开,使用户可以自动的将其设定的异常处理逻辑规则嵌入到 BPEL 流程中。该方法是在流程的设计阶段实现的,很难应对运行时出现的新的异常情况。并且它要求用户根据 ECA 规则制定异常处理策略,这就要求用户知道其异常处理逻辑的含义,对用户的理解能力和专业功底提出了要求,如果考虑不周,很容易产生异常和错误,导致流程无法继续运行。

文献[3-4]考虑到在流程运行过程中可能出现的一系列问题,提出了一种基于补偿业务生成图的组合服务异常处理方法。基于流程中任务之间补偿依赖关系,讨论了补偿业务生成图的自动生成问题,当流程出现异常的时候,采取向前处理或者向后恢复的策略。在向后恢复时,保证补偿的实现;在向前处理中对补偿服务不存在的情况采用的等价服务替换,对补偿代价过高的服务转让,提高了灵活性。但是并没有涉及对于那些暂时转让不出去的服务要怎样处理,在寻找等价服务的时候,只是按照类型寻找,没有考虑其它的要求。

由于存在上述不足,学者们开始从多种角度对异常进行分类处理,提出了忽略、重试、等价替换、补偿恢复等主流的异常解决方法。文献[5]提出了一个 MPEHS (多策略异常处理系统),它通过在业务层和系统层之间加了一个截流服务的中间件来进行异常处理。MPEHS 系统包括一个拦截子系统和异常处理策略库,通过拦截系统的服务调用和底层服务调用结果的返回实现系统级别的异常处理。

在对上述文献研究,以及总结流程运行阶段处理异常的一些通用的策略基础上,本文首先将给出一个基于 B+树的等价服务替换算法,算法将综合考虑用户的

收稿日期:2012-03-17

基金项目:辽宁省科技厅博士启动基金(20091031);辽宁省教育厅科学研究一般项目(L2011004)

作者简介:吴吉红(1986-),女,辽宁本溪人,硕士生,主要从事云计算方面的研究,(E-mail) wjh86619@126.com

QoS(Quality of Service)指标要求,保证 BPEL 流程的满意度。随后将分别详细介绍 BPEL 业务流程同步调用 Web 服务和异步调用 Web 服务产生异常的解决策略。文章最后将提出一个关于异常处理的系统架构原型。

1 BPEL 异常处理策略

1.1 异常产生的原因

服务组合本身具有低耦合、自治性的特点,可靠性大大降低,使得 Web 服务组合很容易出现异常,以下为异常发生的几种常见原因[6]:

(1)BPEL 流程组合过程中 Web 服务本身发生异常。

(2)服务调用过程中,网络错误或者超时,导致异常。

(3)BPEL 流程异常,比如 BPEL 中的某个活动异常。

(4)BPEL 执行引擎产生错误。

现在 BPEL 执行引擎(例如 Active BPEL 和 ODE 等)很多,每个执行引擎都有自己的异常处理方法。如果在服务组合过程中,由 BPEL 执行引擎引起的异常,由引擎处理。而由网络错误或超时引起的异常,看作是 Web 服务本身的错误所引起的,处理方式与服务本身异常的处理方式相同。所以只需研究两种异常的解决策略:Web 服务本身异常的处理方式;BPEL 业务流程异常的处理方式[7]。

1.2 异常处理策略

对于上述 4 种异常类型[8],处理策略概括如下:

(1)忽略(Ignore):对后续 Web 服务执行没有影响的服务,如果产生异常,可以采用忽略操作,返回一些默认值,提高流程的响应比。

(2)重试(Retry):为了保持流程原始的特征,最好的方式就是使原来产生异常的 Web 服务重新执行一次,规定最大次数 N,考虑到响应时间,N 值不宜过大,过小起不到重试的作用。当重试次数超过 N,则宣告此异常处理策略失败。

(3)等价服务替换(Replace):使原 BPEL 流程继续运行,最常用的方法就搜索等价的 Web 服务进行替换。等价服务替换仓库中存储常被调用的等价服务,如果仓库中没有,则动态搜索 UDDI(用户注册中心)来寻找。

(4)服务转让(Transfer):如果 BPEL 流程需要终止,为了降低代价,增强资源的利用率,已经成功执行的服务进行转让,给有相同需要的用户。

(5)补偿(Abort):若某个 Web 服务没有正常执行,为了保持一致性,采用对该服务进行补偿。

(6)语法等价的 BPEL 流程替换(GE_BPEL_Replace):语法等价的 BPEL 流程指功能相同,调用接口和原流程的接口匹配的流程。如果是同步流程,由异常产生的位置接着执行。如果是异步流程,由于并不能清楚的知道哪些服务成功执行,哪些服务并未成功执行,只能重新执行这个流程,终止原流程,转让成功服务,终止失败服务。由于是语法等价的 BPEL 流程,所以已经成功执行的 Web 服务都能成功的转让出去。

(7)语义等价的 BPEL 流程替换(ME_BPEL_Replace):语义等价 BPEL 流程是功能相同,调用接口不同的流程。同步流程,由异常信息决定流程执行的位置。异步流程,终止原流程,执行新流程,转让成功服务,终止失败服务。

(8)终止业务流程(Terminate):若以上的异常解决策略全部失效的话,只能采用终止业务流程,给用户返回一个异常信息。在终止业务流程的过程中,转让已经成功执行的服务,补偿未转让成功或未成功执行的服务。

2 等价服务替换仓库的内部实现

2.1 QoS 属性分析

一个 Web 服务除了具有其执行功能外,还具有其它一些诸如:响应时间(response time),安全(security),代价(cost),可用性(availability)或成功率(successful execution rate)等非功能性属性,这些属性集称为 QoS(Quality of Service)指标。在等价服务替换的时候,对用户来说,要求替换的服务具有较高的 QoS 指标,否则可能会产生不符合用户要求的负面效果[9]。

这里仅考虑可用性、代价、响应时间三个指标,分别采用 $Q1$、$Q2$、$Q3$ 来表示。由服务类型设置相应的权重,每个等价服务类内,类型相同,可以将权重设成相同大小,用 P 值来评价服务的 QoS 属性。P 值大的 Web 服务,优先选择。P 值由计算公式如下:

$$P = \sum_{i=1}^{3} Qi * wi \tag{1}$$

其中 Qi 是每个指标的具体属性值,wi 是属性所占的权重。

2.2 B+树的特点与等价服务类构建的内在联系

B+树是 B 树的一种特殊形式,它具有很多非常适合实现等价服务替换仓库存储的特点[10]:

(1)B+树的高度很少超过 3 或 4,是一种高扇出率的结构,正好适合存储同一功能的等价 Web 服务,使得单独访问每个 Web 服务非常方便。

(2)每棵 B+树中,叶子结点中的关键字信息为每个 Web 服务的 P 值,指针指向具体的 Web 服务。在底层的叶子结点之间有指针连接,所以对 B+树的查找可以从最小关键字起顺序查找。将关键字由大到小的排列,更容易找到 QoS 值大的 Web 服务。

(3)所有的非终端结点可以看成是索引部分,结点是只含其子树(根结点)中最大(或最小)关键字。根据这个特点,可以从根结点进行随机查找,使 BPEL 流程能够快速的找到 QoS 属性值大的 Web 服务。

(4)随着插入和删除关键字,动态调整 B + 树的结构,使得 QoS 属性权值高的 Web 服务很容易的加入到等价的 Web 服务类中,属性权值低或很少被访问的 Web 服务很容易从等价服务类中删除。

2.3　等价服务替换仓库的具体实现方式

BPEL 流程中等价服务替换仓库是按照以下方式实现[11]:

(1)按照功能将 Web 服务进行分类,在每个服务等价类中,按照 P 值将其排列成一棵 B + 树。

(2)每个服务设置一个访问标志 Si ,来解决服务访问瓶颈问题。$Si = 0$ 服务空闲,可以调用;$Si = 1$ 服务正在被使用,用户可以选择排队等待,或者跳过这个服务而选择次优的服务。等待会相应延长较优服务的响应时间,而且这个服务也可能会发生异常,从而极大的影响其性能。所以在访问时间冲突的情况下,这里选择次优的等价服务。

(3)每个等价服务替换类中存在一个探针,用于定期访问 UDDI,以查询是否有 P 值更高的服务存在,如果有,则加入到 B + 树中。定期维护 B + 树,很少访问或是 P 值很低的服务,从树中删除,得到一棵能够满足用户要求的较优的 B + 树。

(4)当等价服务仓库中没有功能等价的 Web 服务类时,动态搜索 UDDI。为了缩短响应时间,找到较优的服务,替换异常服务,同时将该服务存储到等价服务仓库中,分配一个试探指针,形成一个新类。以后随着试探指针的探寻,逐渐扩大该类,根据 P 值构成一棵 B + 树,从而形成一个完备的等价服务类。

下面根据旅游业务流程中购买火车票的 Web 服务发生异常,采用服务替换策略解决的情况为例,说明等价服务类内 Web 服务的 QoS 属性权值计算以及 Web 服务的存储方式。这里设定服务权值为 0.4,0.1,0.3。购买火车票服务的 QoS 属性值见表 1:

表 1　购买火车票服务的 QoS 属性

节点编号	Q1:可用性	Q2:代价	Q3:响应时间
等价服务 S1	5	5	5
等价服务 S2	5	5	5
等价服务 S3	4	4	4
等价服务 S4	4	4	4
等价服务 S5	5	5	5
等价服务 S6	5	5	5
等价服务 S7	5	5	5
等价服务 S8	4	4	4
等价服务 S9	5	5	5
等价服务 S10	5	5	5

根据上述 Qi 值由公式(1)可得表 2。

表 2　每个等价服务对应的 P 值

服务	$S1$	$S2$	$S3$	$S4$	$S5$	$S6$	$S7$	$S8$	$S9$	$S10$
P	3.1	3.3	2.6	3.2	3	2.8	3.4	2.9	3.6	2.7

根据 P 值建立一棵等价服务的类 B + 树,如图 1 所示。

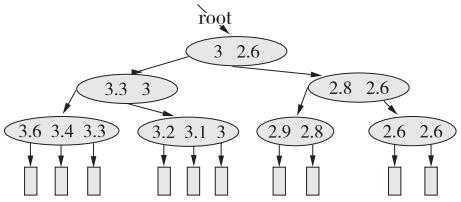

图 1　购买火车票服务等价类的 B + 树

这棵 B + 树是按照 P 值由大到小排列的,从底层迅速的查找到 P 值最高的服务进行替换。假设此时所有服务都空闲,这里采用服务 9 进行替换。

以下是探针在 UDDI 中查找到 P 值更高的服务,Web 服务 11,假设 P 值为是 3.7,其插入过程如图 2 所示。

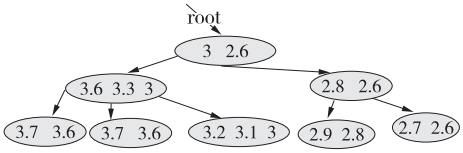

图 2　新服务添加之后的 B + 树

当需要从 B + 树中删除那些 P 值很低或者经常并不被访问的 Web 服务时,这里我们将 P 值为 2.6 的 web 服务 Q3 删除,其处理过程如图 3 所示。

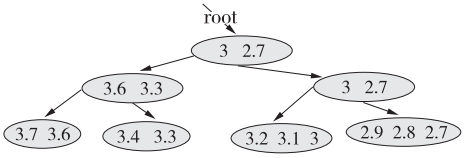

图 3　P 值低或者经常不被访问的服务删除之后的 B + 树

3　形式化定义及异常处理算法

3.1　异常处理的相关形式化定义

为了便于异常分析和处理[12],将服务分为四类,定义如下:

定义 1　(决定类服务)对后续的服务组合有决定性影响作用的服务,称为决定类 Web 服务,Define(determine_webs ws)。

定义 2　(信息类服务)只搜索用户需要知道的信息的服务,称为信息类 Web 服务,Define (information_webs ws)。

定义 3　(比较类服务)从两个或多个功能相同的

服务中选取一个服务,其中每一个服务称为一个比较类 Web 服务,Define(compare_webs ws)。

定义 4 (动作类服务)单独完成一项特定任务的服务,称为动作类的服务,Define(action_webs ws)。

定义 5 (同步服务)BPEL 流程在相对较短的时间内返回结果,并且只有在 response 到达后,整个服务组合才能继续进行,这样的服务定义为同步 Web 服务,Define(synchronization _webs ws)。

定义 6 (异步服务) BPEL 流程允许在相对较长的时间内返回结果,不要求该服务的 response 到达流程后,流程才能继续运行,BPEL 流程和该服务可同时进行,只要在流程需要的时刻,返回一个回调给流程即可,称为异步 Web 服务,Define(asynchronism _webs ws)。

其中定义 1、2、3 和 4 中的 ws 为某个 Web 服务在系统中的唯一标识。

定义 7 (同步的 BPEL 流程) 当 BPEL 流程中没有异步 Web 服务,即全都是同步 Web 服务的时候,此时流程为同步的 BPEL 流程,规定如下:

If (all ws ∈ synchronization _webs)

bpels ∈ synchronization _BPEL

定义 8 (BPEL 流程都是异步)BPEL 流程中只要调用一个异步的 Web 服务,整个 BPEL 流程都是异步,定义为:

If (any ws ∈ asynchronism _webs)

bpels ∈ asynchronism_BPEL

其中定义 7 和 8 中的 bpels 是系统中 bpel 流程的唯一标识。

定义 9 (捕获异常种类的动作)当异常发生时,要根据发生的原因确定异常的种类,于是定义一个捕获异常种类的动作:

Define(exception ws_fail)

Define(exception bpel_fail)

其中 ws_fail 是发生异常的 Web 服务的唯一标识,bpel_fail 是发生异常的 BPEL 流程的唯一标识,exception 是异常的种类。

定义 10 (异常的触发条件)针对异常的种类选取相应的异常处理策略,为这些异常处理策略定义一定的触发条件:

Defrule trigger_rule

trigger_condition

= > policy

其中 trigger_condition 是定义 1、2、3 和 4 中的 4 种类型的 Web 服务异常的逻辑组合,而 policy 指前文提到的异常处理策略。

3.2 异常处理的规则

为了更加方便、合理的处理异常,下面将 4 类服务进行同步或异步的归类。

规则 1 对于决定类服务,由于它将会影响到后续的 Web 服务调用,规定为同步服务,如下:

If (ws ∈ determine_webs)

ws ∈ synchronization_webs

规则 2 对于信息类服务,对后续的 Web 服务将不会产生任何影响,为了提高响应比,采用异步服务调用的方式。规定如下:

If (ws ∈ information_webs)

ws ∈ asynchronism_webs

规则 3 对于比较类服务,至少有两个功能相同的 Web 服务存在,考虑到 BPEL 流程的整体性能采用异步的方式调用这类服务,让这两个服务同时运行。规定如下:

If (ws ∈ compare_webs)

ws ∈ asynchronism_webs

规则 4 对于动作类服务,由于动作本身的性质不同,或对后续动作有影响,或没有。若有,则采取同步调用方式,若无,则采用异步调用方式。规定如下:

If(ws ∈ action_webs)

{

if(ws affects other webs)

ws ∈ synchronization_webs

else if (ws doesn't affect other webs)

ws ∈ asynchronism_webs

}

3.3 服务异常和流程异常的处理算法

服务异常和流程异常的具体调度的算法如下:

catch(exception ws_fail)

catch(exception bpel_fail)

if(ws_fail ∈ synchronization_webs)

{

Defrule retry_rule

if (retry_rule == false)

Defrule replace_rule

else if (retry_rule == false and replace_rule == false)

Defrule GE_BPEL_Replace

else if (retry_rule == false and replace_rule == false

and GE_BPEL_Replace == false)

Defrule ME_BPEL_Replace

else

{

Defrule terminate

```
if ( ws 成功执行)
Defrule transfer
else if ( ws 未成功执行 or ws 未转让成功)
Defrule abort
}
}
else if ( ws_fail ∈ asynchronism _webs)
{
Defrule ignore_rule
if( ignore_rule  == false)
Defrule retry_rule
if ( retry_rule  == false)
Defrule replace_rule
else if ( retry_rule  == false and replace_rule  == false)
{
Defrule GE_BPEL_Replace
if ( ws 成功执行)
Defrule transfer
else if ( ws 未成功执行 or ws 未转让成功)
Defrule abort
}
else if ( retry_rule  == false and replace_rule  == false
and GE_BPEL_Replace  == false)
if ( ws 成功执行)
Defrule transfer
else if ( ws 未成功执行 or ws 未转让成功)
Defrule abort
Defrule ME_BPEL_Replace
else
{
Defrule terminate
if ( ws 成功执行)
Defrule transfer
else if ( ws 未成功执行 or ws 未转让成功)
Defrule abort
}
}
else if ( bpel_fail ∈ synchronization_bpel)
{
Defrule GE_BPEL_Replace
if ( GE_BPEL_Replace  == false)
Defrule ME_BPEL_Replace
else
{
Defrule terminate
if ( ws 成功执行)
Defrule transfer
else if ( ws 未成功执行 or ws 未转让成功)
Defrule abort
}
}
else if ( bpel_fail ∈ asynchronism_bpel)
{
Defrule GE_BPEL_Replace
if ( ws 成功执行)
Defrule transfer
else if ( ws 未成功执行 or ws 未转让成功)
Defrule abort
else if ( GE_BPEL_Replace  == false)
{
if ( ws 成功执行)
Defrule transfer
else if ( ws 未成功执行 or ws 未转让成功)
Defrule abort
Defrule ME_BPEL_Replace
}
else
{
Defrule terminate
if ( ws 成功执行)
Defrule transfer
else if ( ws 未成功执行 or ws 未转让成功)
Defrule abort
}
}
```

4　异常处理原型系统的设计

这里采用 jdk1. 5. 0_07, apache-tomcat-6. 0. 29 和 eclipse-jee-ganymede-SR2-win32 来实现流程的自动运行。根据上述异常处理策略的描述,提出一个解决异常的原型系统[13],如图 4 所示。

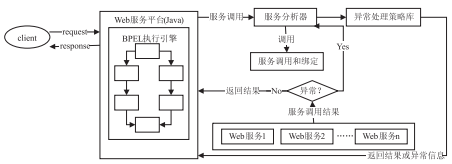

图 4　异常处理的原型系统

服务分析器:对调用的服务进行分析,把服务归为

以上四类,方便异常的处理。异常处理策略库:包含所有异常处理的策略,根据异常的种类,采用相应的异常处理策略,具体处理方式如图 5 所示。

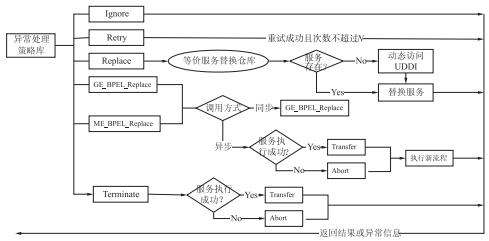

图 5 异常处理策略系统内部结构图

本异常处理的系统原型是独立于 BPEL 执行引擎的,是一个中间层的检测系统,使异常的处理更加灵活。这个原型系统可以解决以上提到的各种异常情况,对不同的异常种类,触发相应的异常处理策略,保持流程的一致性和可靠。

5 结束语

现有的异常处理方法还不够详尽,而且没有区别处理异步和同步服务调用过程中的异常,在等价服务替换的时候也没有考虑到 QoS 属性。针对以上问题,总结了在流程运行阶段处理异常的一些通用的策略,并提出了基于 B + 树的等价服务替换算法,考虑了用户的 QoS 要求,保证了 BPEL 流程的满意度。在此基础上构建了一个异常处理的系统框架原型,此原型系统能解决 Web 服务环境下异常处理的各种情况,符合应用的要求。

参 考 文 献:

[1] 刘明涛.BCSEP 中异常处理机制的研究与实现[D].沈阳:东北大学,2006.

[2] 刘 海,刘 安,李 青,等.一种 ECA 规则驱动的 BPEL 流程异常处理和分析机制[J].小型微型计算机系统 2010,31(7):1363-1370.

[3] 尚宗敏,崔立真,王海洋,等.基于补偿业务生成图的组合服务异常处理方法研究[J].计算机学报,2008,31(8):1478-1490.

[4] 朱颢东,蔡乐才.一种基于熵的不一致规则的处理算法[J].四川理工学院学报:自然科学版,2008,21(4):96-98.

[5] 徐彰杰.基于 BPEL 的 Web 服务组合异常处理方法[D].西安:西北大学,2010.

[6] 张 华,王 茜.面向服务工作流补偿机制的研究与实现[J].东南大学学报:自然科学版,2009,39(1):40-46.

[7] 周如民,陈 平,鲍 亮,等.基于动态代理的 BPEL 恢复机制[J].计算机应用研究,2009,26(5):1770-1773,1784.

[8] 尚宗敬.智能流程异常处理的若干关键技术研究[D].山东:山东大学,2009.

[9] 温嘉佳.Web 服务组合及其相关技术的研究[D].北京:北京邮电大学,2006.6.

[10] 李卓伟,郭松涛.一种新的基于 B + 树结构的 XML 元素的索引方法[J].计算机工程与应用,2007,43(14):162-165.

[11] Liu Hai,Li Qing.Enhancing Web Services Conversation with Exception Contexts for Handling Exceptions of Composite Services[C].The 9th IEEE International conference on E-Commerce Technology. Japan,2007,23-26.

[12] Friedrich G,Fugini M,Mussi E,et al.Exception Handling for Repair in Service-Based Processes[C]. The 2th IEEE International conference on E-Commerce Technology.Austria,2007,23-26.2010,(36):198-215.

[13] Lerner B S,Christov S,Osterweil L J,et al. Exception Handling Patterns for Process Modeling[C]. The 2th IEEE International conference on E-Commerce Technology.USA,2007,23-26,2010,(36):198-215.

Exception Handling Mechanism Based on the BPEL Process of the B + Tree

WU Ji-hong, *GAO Hui*

(College of Information, Liaoning University, Shenyang 110036, China)

Abstract: Web service composition has the characteristics of loose coupling and autonomy, the BPEL specification of exception handling mechanism is not perfect, so BPEL process exception handling is a major problem of the study. To solve the problem, on the basis of some common strategies to handle the exception in the operational phase of the process, replacement algorithm based on the equivalent of B + tree is proposed. synchronous and asynchronous web service in the BPEL which produces an anomalous situation is described respectively. Finally an exception handling framework prototype is proposed.

Key words: B + tree; BPEL; exception handling; QoS; equivalent service replacement