面向分布式业务执行的 BPEL 流程解析方法①

王 雷，齐美石

(中国科学技术大学 自动化系，合肥 230027)

摘 要：当前工业界标准的业务流程描述语言 BPEL 是一种集中式的编制语言，通常业务执行时由集中式引擎控制，易成为系统的性能瓶颈。提出一种 BPEL 业务流程解析方法，以与服务直接相关的活动为基础，将流程划分出若干基本单元，通过递归算法将其它活动归属到相关单元，产生多个子流程，实现对原有流程的分割，使业务流程可以分布式执行。实现了一个分布式业务执行原型系统，验证了解析方法的有效性。

关键词：业务流程执行语言；流程分割；分布式业务执行；

A BPEL Process Parse Method for Decentralizing Execution

WANG Lei, QI Mei-Shi

(Department of Automation, University of Science and Technology of China, Hefei 230027, China)

Abstract: As the current industry standard, BPEL is an orchestration language. The process is under control by centralized engine，will became the bottle neck of System. Give a parse method, partition a BPEL program into several units based on activities associated with services. Classify other activities to these units by recursive algorithm, generating several sub processes. These processes can be executed by several decentralized engines. Make a distributed process execution engine prototype system. Experimental results show the method is effective.

Key words: Business Process Execution Language; Process Partition; Distributed Business Execution

目前工业界标准的业务描述语言为 BPEL(Business Process Execution Language，即业务流程执行语言)[1]，是一种使用 XML 编写的编程语言，广泛应用于与 Web 服务相关的项目开发之中，具有良好的跨平台能力、移植性、厂商无关性等优点。然而作为一种集中式的编制语言，BPEL 流程的执行通常由集中式业务引擎控制，它负责服务的调用、消息的交互、异常处理、流程逻辑控制，易成为流程执行的性能瓶颈。如果将 BPEL 流程分割成多个部分，每个部分由不同的引擎分别执行，可以减少单个引擎的压力、提高整体并发程度、减少系统之间通信量，从而提高系统的整体性能。

对 BPEL 流程的分割解析已经有一些研究工作。文献[2]提出基于依赖图的分割算法，从执行效率的角度考虑 BPEL 分割问题，并没有实现由 BPEL 流程到依赖图转换，且生成的子流程在执行过程中可能会造成某个节点需要监视其他节点的情况。文献[3]提出基于角色的业务流程分解机制，但是其中的“数据分析”模块并没有实现。文献[4]提出一种基于数据流优化的 BPEL 流程分割方法，首先通过“赋值变换”等阶段将流程表示成程序流图，在此基础上进行分割，但是它并没有针对 BPEL 流程各个活动及分割算法作详细说明。

目前主流的 BPEL 执行系统（如 ActiveBPEL 等）通常为集中式引擎，流程的解析、调度、监控、管理等由一个核心引擎控制，在可扩展性、健壮性以及吞量等方面都不能满足实际需求。在分布式执行系统的方面，目前也有些研究工作：如 IBM 公司的基于“持久消息队列”、瑞士苏黎士大学的基于“事件驱动”的分布式工作流系统。

① 基金项目:国家高技术研究发展计划(863)(2008AA01A317)
收稿时间:2011-04-18;收到修改稿时间:2011-05-27

本文提出一种 BPEL 递归解析方法，在 BPEL 流程活动的概念基础上，对 BPEL 流程活动按不同类型分类，将一个完整的 BPEL 流程解析分割成多个子流程，并用倒序方法处理流程中的数据流问题。在研究现有业务流程执行系统的基础上实现了一个分布式业务执行原型系统，实验结果表明，此方法适应于实际运行环境。

1 BPEL流程概述

BPEL 是在 IBM 的 WSFL 和 Microsoft 的 XLANG 基础上发展而来的一种基于 XML 的编程语言，具有灵活性、嵌套组合、关注点分享、会话状态和生命周期管理、可恢复性等诸多优点，是当前 Web 服务组合领域最为成熟和被广泛支持的技术。它将遵守 WSDL 规范描述的 Web 服务组合起来抽象成业务流程，该流程描述了各方参与者的实际行为和消息交互。

BPEL 的核心概念为活动，活动是指流程中的一条语句或者一个步骤的执行。BPEL 把业务中涉及的服务调用、赋值、逻辑控制等功能抽象为活动，整个业务逻辑由多个不同的活动组成。这些活动又分为基本活动和结构化活动。基本活动描述了流程的一个具体步骤，是 BPEL 流程中的一个原子活动，结构活动规定了一组活动发生的逻辑关系，用于管理流程的数据流和控制流。除了活动之外，BPEL 流程通常还包括合作伙伴链接、相关集、变量、异常处理等元素。合作伙伴链接定义了流程中涉及的 Web 服务的接口和操作；相关集用于关联和标识一个 BPEL 流程实例；变量与 JAVA 等程序语言中的变量类似，用于存储和处理数据；错误和补偿处理机制在流程执行过程出现异常后，捕获异常并作相应处理。

2 BPEL递归分割算法

2.1 活动的定义与划分

BPEL 流程由活动组成，而活动被划分为两大类：基本活动与结构活动。

基本活动包括 Receive、Invoke、Reply、assign 等。<receive>活动通常是一个流程的起点，流程会阻塞在这里直至该活动执行，它从流程的外部伙伴那里接收数据，保存到流程变量中；Assign 活动用于操作数据变量，它可以包含多个子活动，每个子活动的作用相当于程序语言中的一个赋值语句；Invoke 活动指定了流程中涉及到的 Web 服务的接口和操作；Reply 活动发送消息应答 Receive 活动接收的流程请求。

基本活动中 Invoke 是与 Web 服务直接相关的活动，我们把与 Web 服务直接相关的活动称为节点活动，其他基本活动称为附属活动，Receive 活动作为流程的一个起始点，也归为节点活动。

结构活动包括 Sequence、Switch、While、Flow 等等，每个结构活动内部可以包含多个子活动，不同结构活动内子活动的执行逻辑不同。Sequence 活动内的各个子活动按出现次序顺序执行，所有活动完成后 sequence 活动结束；switch 活动内的各个子活动按条件选择执行，这些活动中只有一个会被执行，该活动结束表示 switch 活动即结束，switch 活动有时会被称为 if 活动；flow 活动内的多个子活动是一种并发关系，即这些活动会并发的被执行，只有当所有活动都执行结束后，flow 活动才算结束。

2.2 流程分割原则

为使 BPEL 流程可以被多个引擎分布式执行，一个完整的流程会被分割成多个子流程，这些子流程被分别部署到多个不同的节点上，流程中原有的控制流和数据流关系被破坏，因此流程的正确执行需要在各子流程中添加相关的控制和数据流信息，典型的情形如流程执行过程中某一具有分支结构的子流程后续流程数据流向处理，该子流程中需要有分支的选择条件、下一跳节点地址等信息。为此我们提出流程分割原则和前置条件、后置条件的概念。

为保证分割后的多个子流程能被正确的执行，我们提出以下约定，作为流程分割的原则：

（1）流程的分割以流程中节点活动为依据，子流程的数目为 BPEL 流程中节点活动的个数，节点活动与子流程一一对应，子流程由节点活动命名；

（2）附属活动根据其在整个流程中与节点活动的逻辑关系，被划分到相应节点活动的子流程当中：

a.顺序流程中，附属活动划归到其前一个节点活动所在的子流程当中；

b.分支流程（选择性分支或并行发支），每个分支开始处的附属活动划归到分支流程开始之前节点活动所在的子流程当中；

（3）结构活动如 Flow、Switch 等，把结构活动（不包括其子活动）划归到结构活动开始前的节点活动所在子流程当中；

（4）Switch 等选择性分支，在此结构活动之前的节点活动执行完成后，需要根据条件选择后续分支执行流程，我们称此结构活动为该后置条件，加入该结构活动之前的节点活动所在的子流程中；

（5）结构活动 Sequence，表示顺序逻辑关系，默认不在子流程中表示；

（6）为保留原有流程的控制和数据流关系，每个子流程必须保留有下一个子流程的索引（我们统一为下一个子流程中节点活动的名称），对于分支流程，其前一个节点活动所在的子流程保留所有分支的索引信息。

（7）结构活动结束处，对于并行分支执行结束后，后续节点活动的必须要等到所有分支都执行完并传输来相应数据后方可激活，所以此处需要添加相应的结构信息，我们称为子流程的前置条件。

图 1 的例子为一学位申请查询业务流程图，该业务根据学生的姓名等信息，判定该学生是否可以正常毕业，它涉及到的活动有：

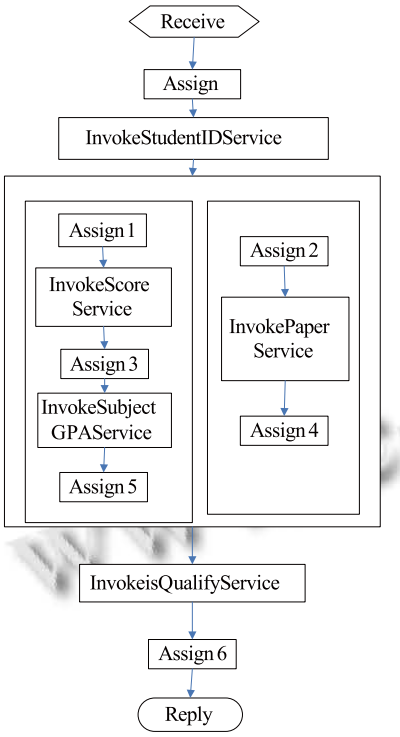

图 1　业务流程图

节点活动：receiveInput、StudentIDService、Scorce-Service 、 SubjectGPAService 、 PaperGPAService 、 isQualifyService；

附属活动：Assign，Reply；

结构活动：flow、sequence

流程中有六个节点活动，所以子流程数目为六个，分别与六个节点活动一一对应；

Assign 作为顺序逻辑中的附属活动，划归到 Receive 节点活动所在的子流程之中，Assign3、Assign4、Assign5、Assign6 与 Assign 类同，Reply 活动划归到 InvokeisQualifyService 节点活动所在的子流程之中；

Assign1 和 Assign2 作为并行分支开始处的附属活动，划归到分支开始前节点活动所在的子流程当中，即划归到 InvokeStudentIDService 节点活动所在的子流程之中；

结构活动 flow，划归到 InvokeStudentIDService 节点活动所在的子流程之中，作为其后置条件；

结构活动 Flow 结束处，节点活动 InvokeisQualifyService 必须要等到所有分支都执行结束并传输来参数后方可激活，此处添加 flow 结束标志和分支数目作为其前置条件。

2.3 递归分割算法

BPEL 递归解析分割算法根据 2.2 中提出的分割原则，将一个完整的 BPEL 流程分割成多个以节点活动为基础组成的子流程之中。子流程仍由 XML 语言描述，其算法描述如下：

输入：BPEL 流程；

输出：若干 BPEL 子流程；

算法过程：

读入整个流程；

创建活动集合，依次从流程中取出活动，调用 3）-4），遍历所有活动，返回流程活动集合，转 5）；

如果其为基本活动，建模，放入活动集合之中；

如果其为结构活动，对其中的每个分支流程，递归调用 2）-4），完成后对结构活动建模，放入活动集合之中；

遍历活动集合，调用 6）-7）；

如果其为节点活动，则新建一子流程文档，将其写入子流程之中，并依次取后续活动：

a.如果其为附属活动，则将其写入子流程之中；

b.如果其为节点活动，结束；

c.如果其为结构活动，对其每个分支，依次取出后续活动，视情况调用 a、b 或 c，并加入后置条件；

如果其为结构活动，对每个分支，递归调用 5)-7)；并在结构活动最后一个子流程中加入前置条件；

输出所有子流程；

2.4 数据流倒序处理

集中式环境下所有的数据流由集中式引擎控制，该引擎存储该业务流程相关的全部数据变量。分布式引擎执行系统中各个引擎只负责处理一部分子流程，原流程中该子流程外的数据变量对其不可见。对于单个引擎来说，处理数据流的方法有两种：保存和处理所有数据变量，转发所有数据给后续引擎而不管这些数据是否已经废弃；根据后续流程需要有选择性的保存、转发数据。前者会造成不必要的通信开销、内存消耗和处理时间，造成整体系统性能下降。我们在业务流程递归解析分割算法中，引入数据流倒序处理机制，为第二种方法提供支持。其算法描述如下：

初始化：后续流程所需变量集合 post=null，当前子流程所需要变量集合 current = null; 子流程集合 E={分割后所得子流程}；

While(E 非空) {

取出 E 中最后一个子流程，根据子流程描述，赋值当前子流程所需要的变量 current；

将 post 加入该子流程描述中，作为当前子流程后续服务所需要变量；

post = post + current - 当前子流程执行过程中产生的变量；

}

经过数据流倒序处理，各子流程包含了原流程各阶段所需要数据变量信息，在分布式环境下各引擎可以根据实际情况选择性的保存、转发数据。

3 验证原型系统

3.1 原型系统结构

为验证本文提出的 BPEL 解析方法，我们实现了一个分布式业务流程执行原型系统，该原型系统由业务流程生成平台、BPEL 流程解析分割模块、分布式业务流程部署模块、分布式引擎模块四部分组成，其结构如图 2 所示，其中：

业务生成平台，以 eclipse + BPEL 插件为软件基础，用以生成 BPEL 流程；

BPEL 流程解析分割模块使用本文提出的分割算法将 BPEL 流程分割成多个子流程，包括流程分割、BPEL 活动建模、WSDL 解析等子模块，WSDL 解析子模块用于提取和处理 BPEL 流程中相关的 Web 服务的接口、操作和消息类型等信息；

分布式业务流程部署模块负责将 BPEL 流程解析分割模块解析后得到的若干子流程部署到不同的服务器节点；

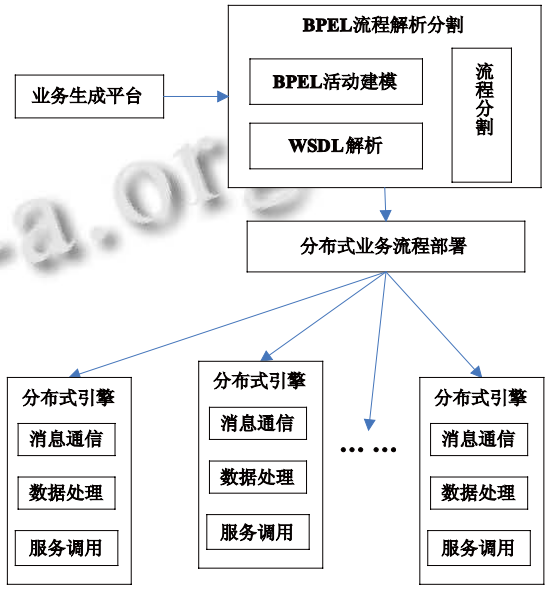

图 2 原型系统结构图

分布式引擎模块驻留在不同的服务节点用于控制各个 BPEL 子流程的执行，它负责接收上一跳引擎转发的流程数据参数、提取数据输入进行服务调用、处理服务调用返回结果、转发流程数据给下一跳引擎和执行子流程中定义的相关活动。

3.2 实验

使用网上公共服务和自己创建的 Web 服务创建流程，5 台联想万全服务器中部署相关服务并作为分布式引擎驻留节点。

实验所用如图 3 所示的人脸识别业务流程，流程包含认证、人脸特征识别、检索、生成列表等四个服务，涉及 Receive、Assign、Invoke、IF、Sequence、Reply 等 BPEL 活动。各服务的功能为：认证服务根据用户输入的 ID，密码等信息查找数据库判断用户是否合法；人脸特征识别服务提取人脸特征信息；检索服务根据人脸特征识别服务提取的特征信息检索数据库，返回结果与该特征相符合的数据；生成列表服务对检索服务返回的结果按一定判定标准排序，返回列表。

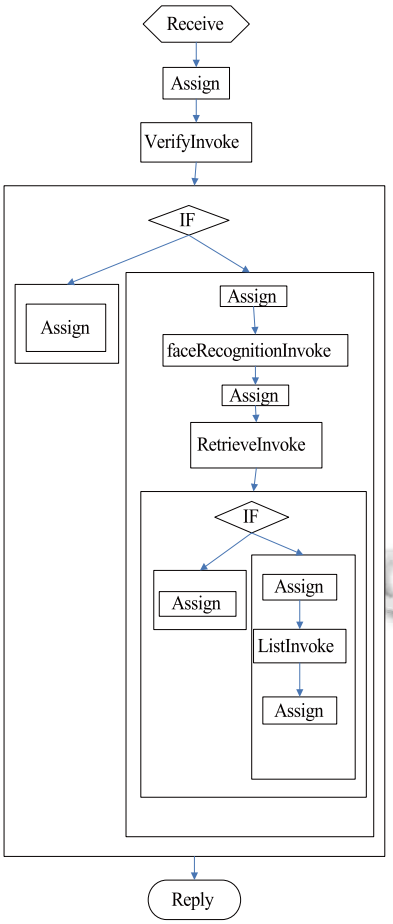

图 3 人脸识别业务流程

整个流程的执行过程：流程等待接收用户请求，用户输入 ID、密码信息后，调用认证服务对用户进行认证，若认证失败，则流程结束；否则继续调用人脸特征识别服务，提取特征信息，调用检索服务根据人脸特征返回结果列表，若失败，则流程结束；否则调用生成列表服务对结果进行排序，流程结束，返回流程执行结果。

我们对人脸识别流程进行解析，将解析后的子流程部署到各个服务器上，同时在各服务器上部署分布式引擎。实验结果表明通过解析分割产生的子流程可以通过各分布式引擎成功执行。

4 结论

本文提出一种面向分布式业务执行的业务流程解析方法，按活动类型将 BPEL 流程分割成若干 BPEL 子流程，为保留各子流程之间的控制逻辑关系与数据流关系，提出流程分割原则与数据流倒序处理算法，为 BPEL 流程的分布式执行提供支持。原型系统运行结果表明，此方法分割后得到的子流程，可以在实际环境中运行。下一步的工作是针对具有 While 等复杂结构活动、错误和补偿机制的 BPEL 流程提出完善的分割方法，分析分布式环境下 BPEL 流程的执行效率。

参考文献

1 Web Service Business Process Execution Language Version 2.0. http://www.oasis-open.org/committees/download.php/23964/wsbpel-v2.0-primer.htm.9M ay 2007.

2 Nanda MG, Chandra S, Sarkar V. Decentralizing execution of composite Web Services. Proceeding of the 19th ACM Conference on Object-oriented Programming, Systems, Languages, and Applications. Vancouver: ACM, 2004:170−187.

3 Khalaf R, Leymann F. Eole-based decomposition of business process using BPEL. Process of the IEEE International conference on Web Services(IC-WS). Chicago: IEEE Computer Society, 2006:770−780.

4 翟岩龙,宿红毅,张晗,战守义.基于数据流优化的BPEL流程分割方法.华南理工大学学报,2009,4.