基于 BPEL4WS 的网格服务组合体系结构及其分析*)

蒋哲远　韩江洪　王　钊
(合肥工业大学 计算机与信息学院　合肥 230009)

摘　要　开放网格服务基础结构 OGSI(Open Grid Services Infrastrstructure)把 Web 服务工作流引入到网格任务描述中,给出了几种 Web 服务与网格技术相融合机制,但并没有界定如何进行网格服务组合。而 BPEL4WS(Business Process Execution Language for Web Services)是描述 Web 服务业务工作流的工业标准。通过对 BPEL4WS 和 OGSI 在生命周期管理、Web 服务实例化和状态交互管理等方面异同的深度分析,提出了一种兼容 OGIS 并使用 BPEL4WS 来合成网格服务的高层体系结构。介绍了一个电力网电能损耗理论计算的实际应用原型系统,表明该文提出的体系结构可应用于网格服务的建模和构造。

关键词　BPEL4WS,网格,服务组合,体系结构,OGS

BPEL4WS-Based Grid Service Composition Architecture and its Analysis

JIANG Zhe-Yuan　HAN Jiang-Hong　WANG Zhao
(School of Computer and Information, Hefei University of Technology, Hefei 230009)

Abstract　The Open Grid Services Infrastrstructure(OGSI) incorporates Web services technologies to facilitate resource sharing, but it does not address overarching concerns of grid service composition. BPEL4WS(Business Process Execution Language for Web Services) is an industrial process modeling language standard for composing Web services. This paper provides an analysis of BPEL4WS and OGSI in terms of their similarities and differences focusing on life cycle management, Web service instantiation and stateful interaction. Based on the analysis a high-level architecture that provides saparate ties for composing and resolving Grid services is presented. The proposed architecture has been applied to a theoretical energy loss calculation system for power grid, which consequently shows in the OGSI framework orchestrating Grid services with BPEL4WS is feasible and effective.

Keywords　BPEL4WS(Business Process Execution Language for Web Services), Grid, Service composition, Architecture, OGSI(Open Grid Services Infrastrstructure)

1　引言

Web 服务作为一种新的面向 Internet 环境的自描述、自包含的分布式计算应用范式,具有跨平台、跨语言、松散耦合等优良特性,正成为学术界和工业界日益关注的焦点[1]。它为 Web 环境下实现企业间业务的动态交互提供了一个全新的解决方案,同时在应用程序之间的通信和系统互操作能力等方面也出现了一些问题。Web 服务标准是建立在现有的标准技术和协议之上,包括简单对象访问协议(Simple Object Access Protocol,SOAP)[2] Web 服务描述语言(Web Services Description Language,WSDL)[3],统一描述、发现和集成(Universal Discovery,Description,and Integration,UDDI)[4],以及可扩展标记语言(Extensible Markup Language,XML)等。一个 Web 服务通常被看作为一个独立的智能实体,可在 Internet 上被广播和传输。

为了满足特别需要,Web 服务通常还需要按照一定的粒度将多个 Web 服务根据特定的应用背景和需求进行合理的组合,实现完整的业务逻辑。服务组合的基础是业务流程定义。早期 IBM 和 Microsoft 都提出了各自的流程定义语言 WSFL 和 XLANG。为了统一业务流程的描述,2002 年 8 月,相关组织提出了 Web 服务的业务流程执行语言 BPEL4WS(Business Process Execution Language for Web Services)规范,它融合了 WSFL 和 XLANG,提供一个管理 Web 服务间的复杂交互的标准业务流程集成模型,支持具有状态和长期交互的多伙伴之间的对等(Peer-to-Peer,P2P)同步和异步消息序列交换计算模式。BPEL4WS 引擎,例如 BPWS4J 和 Collaxa,为 Web 服务的组合和执行提供了一个实时运行环境。

鉴于 Web 服务技术的成熟和流行,开放网格服务体系结构(Open Grid Services Architecture,OGSA)把 Globus 标准与面向商业应用的 Web 服务结合起来,把网格计算从科学与工程计算应用扩展到更广泛的以分布式系统服务集成为主要特征的商业应用领域,并试图通过某种特别机制,把 Web 服务技术同现有的网格标准技术,例如实例管理、信息通知机制、网格资源间的有状态交互等相集成,从而导致了网格服务概念的引入。开放网格服务基础结构(Open Grid Services Infrastructure,OGSI)是构建 OGSA 的基础设施,它的核心就是网格服务规范,该规范在 Web 服务的基础上定义了网格服务

*)基金项目:国家"八六三"高技术研究发展计划基金项目(2002AA415280);合肥工业大学科学研究发展基金项目(040501F)。蒋哲远　博士研究生,主要研究领域为信息系统软件理论与环境和分布式系统;韩江洪　教授,博士生导师,主要研究领域为智能控制技术和分布式系统;王钊　副教授,主要研究领域为电子商务和软件工程。

的标准接口和行为。GT3(Globus Toolkit 3)[5]开发平台是OGSI的第一个完整实现,该平台建立在SOAP,WSDL和WS-Inspection等Web服务技术的基础上,用来支持分布式状态管理、轻量级检查和发现以及异步通知。但是,OGSI并没有明确定义如何发现特定领域的网格服务和组合已存在的网格服务。

本文在对BPEL4WS和OGSI在生命周期管理、Web服务实例化和状态交互等方面的异同进行深度分析的基础上,提出了一种使用BPEL4WS定义网格服务的业务工作流的高级体系结构。为了证明所提出的参考体系结构模型是可行的,开发了一个电力网电能损耗理论计算的实际应用原型系统,用来演示如何集成BPEL4WS和OGSI完成协同工作,发现它们之间交互操作存在的主要问题。

2 相关技术和背景知识

2.1 Web服务

Web服务是通过标准的基于XML的SOAP协议提供的一种分布式软件服务,它使用WSDL文档对其接口进行非常详细的说明,以使用户能够创建客户端应用程序与它们进行通信,并按照UDDI规范进行注册,以便潜在用户能够轻易地找到这些服务。

为了完成在松散耦合环境下的对象访问,以及在基本对象访问之上的事务、工作流、安全机制等,Web服务需要有一系列的标准协议规范来支撑。Web服务的核心协议栈主要包括数据网络层、消息传输层、服务描述层,以及服务发布和发现层等。

从OGSI的角度来看,最有用的Web服务标准是WSDL。OGSI充分利用了由W3C开发的WSDL标准[2],该标准采用不依赖任何特定编程语言和实现方法的XML来描述软件组件或服务。WSDL为服务描述定义了服务接口,以及如何将接口映像到协议消息和具体端口地址的实现细节。它由如下元素组成:portType(一个端口的抽象描述);message(对具有一定规则定义的交互数据的抽象描述);operation(描述单个服务所支持的行为);port(指定绑定具体地址);service(一组相关端口地址与特定绑定的集合)。

Web服务的优势是为异构环境中企业应用的交互或协作提供一个平台。不过,Web服务主要应用于企业间的无状态模式和只有相对简单交互的场合,缺乏对复杂交互和永久状态交互的复杂应用提供支持。OGSI试图利用网格服务来弥补Web服务的某些不足。

2.2 网格服务

OGSI规范采用Web服务的WSDL和XML schema来定义一个扩展构件模型[7]。规范的目的是解决复杂分布式应用中出现的常见问题,例如分布式系统中的永久状态管理。为了达到上述目标,OGSI定义了网格服务的概念。一个网格服务是一个(潜在临时)服务,该服务为达到生命周期管理、特征发现和消息通知等目的,需遵从一组约定(由WSDL接口、扩展和行为来表达)[8]。OGSI规范继承了Web服务的互操作特性,同时包括如下特性:

(1)生命周期。赋予客户可根据依赖或管理该服务的组件的需要,创建和撤消一个服务实例能力,从而在不需要大规模分布式垃圾收集清理程序的情况下,防止网格服务无限地消耗资源。

(2)状态管理。OGSI采用serviceData元素暴露服务的状态数据,从而使得服务的请求者可以对服务实例的状态数据进行查询、更新和当这些数据发生变化时收到相应的事件通知,实现有状态网格服务管理。serviceData的概念与JavaBean类似。因此,每个服务的内部状态元素,都可以通过另外定义的一组Set和Get操作实现从外部访问这些状态数据。

(3)引用。OGSI使用网格服务句柄(Grid Service Handles,GSH)命名和管理网格服务实例。一个用户想访问一个网格服务实例,就必须通过句柄解析器(HandleResolver)服务映射一个GSH到一个具体的网格服务引用(Grid Service Reference,GSR)。这是因为一个GSH仅包含很少信息,例如只是一个统一资源标识符(URI)形式的名字,没有直接地携带访问网格服务实例所需要的足够信息。相比较,GSR中包含访问对应服务实例所需要的所有信息。有了一个服务实例的有效GSR,用户就可以访问相应的服务实例。

(4)服务组。OGSI允许客户使用ServiceData来建立索引,并根据特定目的对网格服务进行分组管理。一个网格服务能定义和同组中其它成员服务之间的关系,也能根据需要加入或离开一个特别服务组。

(5)继承。OGSI需要WSDL 1.2所支持的特性,而WSDL 1.2还正在制定中,因此OGSI对WSDL 1.1进行了扩展,对WSDL1.1中的portType定义了新的模式,关联新的域名gwsdl。gwsdl:portType在几个方面扩展了WSDL 1.1中所定义的portType:为了合成和扩展portType而定义了扩展属性;采用了开放扩展模型,以便允许声明服务数据元素为portType的一部分。

(6)通知。网格服务的状态信息会随着系统的运行而变化,它们之间的许多交互要求动态地监控状态变化。OGSI的通知就是把一种传统的发布和订阅范式应用于这种监控。网格服务支持一个NotificationSource接口,以允许其他网格服务(NotificationSink)订阅进行状态变更。

总之,OGSI规范试图为大规模分布式应用中网格服务可管理提供一个环境,同时也提供合成服务高级机制平台。

2.3 BPEL4WS

BPEL4WS是建立在WSDL所定义的服务模型基础之上的一个定义复杂和不确定性Web服务工作流工业标准规范,其核心是WSDL描述服务间的对等交互概念,流程及其伙伴都被建模成WSDL服务[9]。由于相互关联业务的处理常取决于数据,因此BPEL4WS为数据依赖行为建模提供了一系列活动。为了解决业务处理过程中经常碰到的不确定性情况,BPEL4WS提供了条件和暂停构造过程。BPEL4WS的最重要特性是支持不同伙伴之间业务流程协调,同时也支持对业务流程中的各种不同粒度工作单元执行结果的处理。BPEL4WS支持嵌套结构,并允许嵌套结构中的每个工作单元都拥有自己独立数据需求,从而为业务进程之间的长期交互建模成为可能。

BPEL4WS是建立在WSDL 1.1,XML Schema 1.0,XPath 1.0和WS-Addressing等这些基于XML的规范之上。BPEL4WS通过伙伴为业务流程中的交互服务建模。每一个伙伴拥有一个惟一的名字,其它服务能通过名字和该伙伴进行交互。每个伙伴和一个WSDL文档相关联,该文档描述了一个服务所包含的抽象信息。为了合成Web服务,BPEL4WS流程模型允许开发者通过预先定义的活动集来指定伙伴间的关系。

BPEL4WS流程模型本身是一个流程图，这类似于用来表达算法的流程图。流程的每一步称为一个活动。流程业务处理过程一般是从 receive 活动开始，该活动阻塞业务流程，等待来自客户和触法器的调用请求消息；终点是 reply 活动，该活动发送消息，作为对 receive 接收到的消息响应。invoke 活动调用由伙伴 Web 服务在 portType 上提供的操作。assign 活动把数据从一个地方复制到另一个地方。

BPEL4WS中控制流程类似传统的结构化过程控制构造块，可以将原语活动组合成更复杂的算法。例如，使用 sequence 活动定义包含一个或以上的有序执行活动块；switch 活动从一组分支中只选择一个执行分支；while 指定反复执行一个活动，直到某个条件被满足；flow 活动指定一个或多个并行执行的活动。此外，BPEL4WS 还有较强的异常处理和事务处理能力，支持异常的抛出机制和捕获机制。

3 网格服务组合的参考体系结构

BPEL4WS 原被设计用来合成 Web 服务，但由于如下原因，它不能被用来组合网格服务。正如前面所描述的那样，OGSI 引进 GSH 和 GSR 是为了引用 Web 服务实例，但这并不为 BPEL4WS 规范所支持。另外，由于 BPEL4WS 的当前版本一直是基于 WSDL 1.1，因此 BPEL4WS 不识别某些采用 WSDL 1.2 新特性的网格服务接口描述。

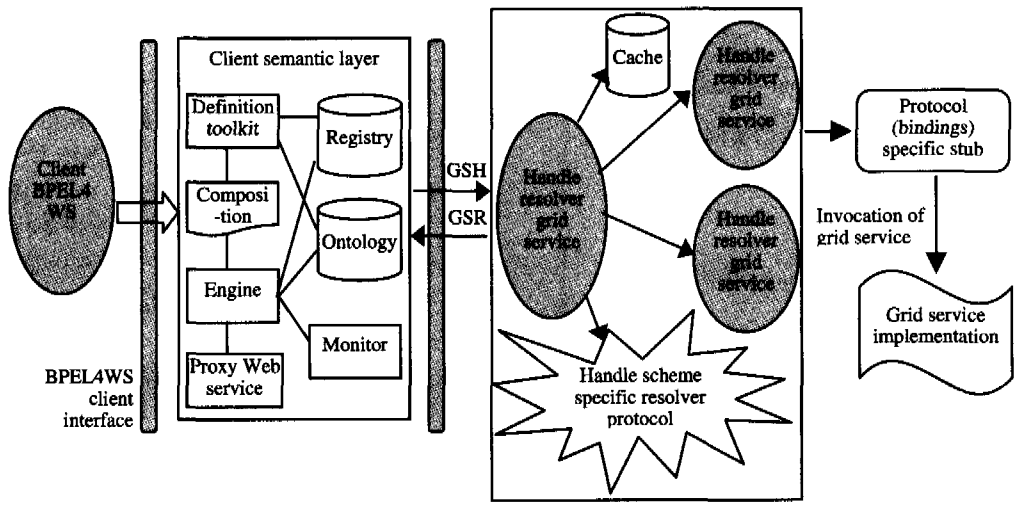

图 1 基于 BPEL4WS 的网格服务组合体系结构

为了解决上述问题，本文以 OGSI 为基础，提出了图 1 所示的一个参考体系结构，可以让用户在客户端使用 BPEL4WS 来合成网格服务。在所给出的参考体系结构中，OGSI 的服务实例、服务内部状态维护和生命期协商管理等机制被引入到网格服务组合语义层，网格服务的请求者可以通过 OGSI 所定义的编程模型创建服务组合实例、维护服务组合实例的生命期，提供 OGSI 规范的兼容实现。网格服务组合语义层主要包括定义工具、执行引擎、监控管理工具、领域本体库以及网格服务的全局注册库等功能部分。网格服务客户被包装成代理 Web 服务容器中的 Web 服务。所有定义网格服务的接口也被重新用 Java beans 定义为 WSDL 文档中具有一个公开网格服务实例属性的 XML 复杂类型。

为了创建新的网格服务实例，从已存在的 BPEL4WS 描述中合成一个新的网格服务。在 Web 服务中还需要定义和实现一个新的网格服务操作，其一系列的激活操作执行过程可描述如下：BPEL4WS 用户启动网格服务组合语义层的客户应用；客户应用根据 BPEL4WS 中的工作流描述，调用代理 Web 服务；代理 Web 服务通过嵌入的网格服务操作调用 GSH 来创建一个网格服务实例的标准过程，并把它的返回值映射到一个 GSR；当完成一个网格服务实例的创建后，网格服务操作将获得一个引用，且把它保存在预先定义的公共网格服务实例属性中。

由于 GSR 是被存储在一个公共全局变量中，所有网格服务的实例都可访问到它。一旦 BPEL4WS 希望调用某个网格服务中具体操作时，首先是调用 BPEL4WS 引擎，例如 BPWS4J 或 Collaxa，来激活存储在 Web 服务容器中的代理 Web 服务。代理 Web 服务然后使用存储在公共属性区中的网格服务引用，去告诉网格服务容器来调用对应网格服务。网格服务把返回结果发送给发出请求的网格服务客户，网格服务客户又把它传给代理 Web 服务。BPEL4WS 引擎获得返回结果后，把它传到下一个服务中。

使用该体系结构的优点之一是当网格服务被重新部署到不同地方时，BPEL4WS 描述和相关联的 WSDL 文档受到的影响能减小到最小程度。

4 参考体系结构的原型系统实现

为了验证所提出的体系结构是可行的，我们在 J2EE 平台上开发了面向虚拟企业的电力网电能损耗分布式理论计算[10]网格服务合成原型系统。该系统基于传统的模型-视图-控制(Model-View-Control，MVC)[11]设计模式，把存储数据和展示数据的功能模块区别开来，以便降低它们之间的耦合度，具有很好的可扩展性和可复用性。电力网电能损耗的计算结果展示(采用 JSP 技术，提供客户视觉上的界面)、计算调度(使用 Servlet 实现，接收用户从计算调度控制界面发送的命令，调用相应的 JavaBeans 组件进行计算业务逻辑)和可视化数据建模(采用 JavaBean 封装所有计算业务逻辑和数据的处理)分别被实现成图 2 所示的三个网格服务。正如图 2 所显示的那样，客户通过浏览器，可看到电力网电能损耗计算的各种计算分析结果报表，并能控制计算参数的实时采集。

原型系统的处理过程如下：首先，计算控制器(computing controller)服务接收来自浏览器客户的请求，并决定应该调用哪个计算模型来处理，然后计算模型(computing model)服务用具体计算业务逻辑来处理客户的请求并返回计算数据，最后计算控制器服务用相应的视窗格式化模型返回的数据，并

通过计算视图(computing view)服务呈现给客户。

原型系统的 Web 服务(对应一个网格服务)被描述为合作伙伴。服务之间的可能事件被定义为一个 XML 复杂类型,且用 Java beans 实现。因此,一旦 BPWS4J 启动 Web 服务,该服务调用相应地 Java beans。这些 Java beans 再调用网格服务中的对应操作。使用 BPEL4WS 描述上述 3 个网格服务组合为一个新的复合网格服务的部分代码如下:

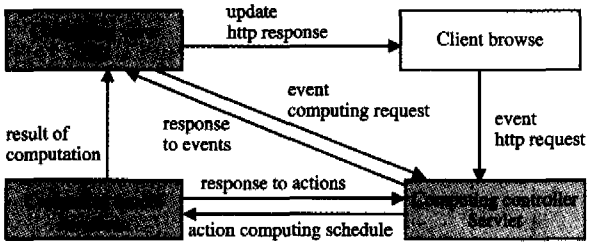

图 2 网格服务组合的实例

```
〈process name="EnergyLossesComputationProcess"
……
〈partnerLinks〉
    〈partnerLink name="computing"
        partnerLinkType="lns:ComputingViewLinkType"
        myRole="viewerService"/〉
    〈partnerLink name="scheduing"
        partnerLinkType="lns:ComputingControlLinkType"
        myRole="controllerService"/〉
    〈partnerLink name="reporting"
        partnerLinkType="lns:ComputingModelLinkType"
        partnerRole="modellerService"/〉
〈/partners〉
〈variables〉
    〈variable name="computingReport"
            messageType="lns:computingReportMessage"/〉
    〈variable name="computingRequest"
            messageType="lns:computingRequestMessage"/〉
    〈variable name="computingControl"
            messageType="lns:computingControlMessage"/〉
    〈variable name="computingInfo"
            messageType="lns:computingInfoMessage"/〉
〈/variables〉
〈sequence〉
    〈receive partnerLink="computing"
        portType="lns:ComputationViewPT"
        operation="sendComputationInfo"
        variable="computingRequest"
        createInstance="yes"〉
    〈/receive〉
    〈assign name="assign"〉
        〈copy〉
          〈from variable = "computingRequest" part = "computingInfo" /〉
          〈to variable="ComputingControl" part = "computingInfo"/〉
        〈/copy〉
    〈/assign〉
    〈invoke partnerLink="scheduling"
        portType="lns:SchedulingPT"
        operation="sendComputingSchedule"
        inputVariable="computingControl"
        outputVariable="computingInfo"〉
    〈/invoke〉
    ……
    〈/sequence〉
〈/process〉
```

5 分析与讨论

原型系统的实验验证了参考体系结构中使用 BPEL4WS 合成网格服务是可行的,显示了 BPEL4WS 和网格服务在生命周期管理、状态交互等方面存在许多类似的特性,同时在以下几个方面也存在很大差异:

(1)在 BPEL4WS 中,Web 服务实例之间的协调是由数据驱动的,相互间的联系方法与数据库系统中通过索引键建立关联方法相类似。因此,开发者必须通过 WSDL 的 portType 来定义关联集。另一方面,OGSI 使用网格服务实例引用来协调网格服务实例,每个网格服务实例拥有惟一的引用。既然 GT3 主要是通过一种基于 Java RMI 的 Web 服务规范 JAX-RPC 实现,实例之间的联系管理类似 Java 中多实例的处理。因此,GT3 不能把它的网格服务实例引用输出到 BPEL4WS 中。同样,BPEL4WS 也不能拥有网格服务实例的引用。结果,GT3 所附加的函数,例如网格服务的组管理以及生命周期管理等,不能直接被 BPEL4WS 使用。BPEL4WS 不支持在构造过程中删除 Web 服务实例,只能通过终止整个进程来实现。

(2)GT3 和 BPEL4WS 实现序列化的方法不同。网格服务是序列化网格服务实例,而 BPEL4WS 是序列进程中的变量。因此,BPEL4WS 不能创建网格服务来序列网格实例。BPEL4WS 和 GT3 都使用 WSDL 获得服务信息,并使用它发送消息和请求响应。然而,它们所依据的版本是不同的。GT3 采用 WSDL 1.2,并重新命名为 gwsdl,而 BPEL4WS 是基于 WSDL 1.1 的。BPEL4WS 不能解析 WSDL 1.2 新增加的特性,这个问题也许要等到 WSDL 1.2 变成一个万维网的正式规范后才可能容易地解决。GT3 所支持的最重要特性是通知机制,这与 Java 中观察者和观察机制有点类似,允许服务在状态发生变化时推或送信息。然而,BPEL4WS 中并没有一个机制能映射这种机制。

基于上述分析,可得出如下结论:

(1)BPEL4WS 和 OGSI 有不同的关注焦点。如何挖掘相互间的潜能,设计一个使之能被充分地集成在一起的规格或系统,并不是一件轻松的工作。

(2)BPEL4WS 并不是合成 Web 服务的惟一规范。语义 Web 社区已经提出了 DAML-S[12]来描述 Web 服务的语义和 Web 服务的组合机制。然而,它们并没有提供像 BPWS4J 或 Collaxa 这样复杂的引擎来支持 DAML-S 规范。文[13]中使用了 DAML-S 来定义 Web 服务语义,并可把该语义描述转换成 BPEL4WS。因此,BPWS4J 也能为执行 Web 服务提供相应的实时环境。其它目前合成 Web 服务的主要方法有使用线性逻辑(linear logic)原理进行语义 Web 服务的自动组合[14]、使用 π-calculus 和 Petri nets 等对 Web 服务组合进行更抽象描述[6]。

结束语 本文提出了一种使用 BPEL4WS 合成网格服务的高层体系结构模型。为了验证该体系结构的可行性,文中引入了一个基于 MVC 设计模式的电力网线损理论计算实例。实验结果显示,使用 BPEL4WS 对网格服务建模或合成是适当的。基于实验结果,我们对 OGSI 和 BPEL4WS 不相匹配的地方做了详细分析,而这些问题是在网格服务中使用 BPEL4WS 时所必须要面对的。未来的工作是对所提出的体系结构进行必要的扩充,以便能够进一步处理文中所提到的问题。

参 考 文 献

1 Papazoglou M P. Service-oriented computing: concepts, characteristics and directions. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering. Roma, Italy: IEEE, 2003. 3~12

2 Christensen E. Web services description language (WSDL) version 1.1. www.w3.org/TR/wsdl, 2001

3 Gudgin M. SOAP version 1.2 part 1: Messaging framework. http://www.w3.org/TR/soap12-part1, 2003

(下转第 131 页)

定义 4 如果 $P_1 \subseteq P_2$ 并且 $P_2 \subseteq P_1$，则 P_1 和 P_2 等价，记作 $P_1 \equiv P_2$。

4.2 路径表达式的最小化技术

本文的主要目的是把用户书写的复杂的比较长的路经表达式，根据排他性包含约束缩短成等价的较短的路径表达式，以此减少结构连接次数。这样一方面减小了结构连接的代价，另一方面在查询计划生成阶段也大大地减少可选的连接计划，从而优化了 XML 路径表达式的查询。定理 2 及其推论描述了我们的路径表达式的缩短策略。

定理 2 如果 $E_1 \alpha E_3$，那么 $(E_1/E_2/E_3) \equiv (E_1//E_3)$ 一定成立。

证明：使用反证法很容易证明这个定理的正确性。假定 $E_1/E_2/E_3$ 与 $E_1//E_3$ 不等价，则有 $(E_1/E_2/E_3) \not\subset (E_1//E_3)$，或者 $(E_1//E_3) \not\subset (E_1/E_2/E_3)$ 成立。我们首先证明第一个假设不成立：

假设 $(E_1/E_2/E_3) \not\subset (E_1//E_3)$，则必有 $(\exists e_i)(e_i \in Q(T_d, (E_1/E_2/E_3)) \rightarrow e_i \notin Q(T_d, (E_1//E_3)))$，即存在另一条从 E_1 开始访问 E_2 元素实例的路径，这与 $E_1 \alpha E_3$ 相互矛盾，假设不成立。$(E_1/E_2/E_3) \subseteq (E_1//E_3)$ 得证。同理可以证明 $(E_1//E_3) \subseteq (E_1/E_2/E_3)$，因此 $(E_1/E_2/E_3) \equiv (E_1//E_3)$ 成立。

[证毕]

推论 如果 $E_1 \alpha E_n$，那么 $(C_1E_1C_2E_2C_3 \cdots C_nE_n) \equiv (C_1E_1//E_n)$（$C_i$ 是路径操作符'/'或'//'）。

证明方法同定理 2(略)。

4.3 路径表达式最小化算法

在进行路径缩短时要首先对路径表达式进行预处理。预处理的工作有两个：①保证路径的合法性，即路径上相邻元素之间满足包含关系。对不合法的路径表达式不予处理。②将以"/"开头的路径表达式转换为以"//"开头的路径表达式。如将路径表达式/pub /papers/paper/sections/section/title 转换为//papers/paper/sections/section/title，由于在任意一个模式图中根元素都排他包含其直接子元素，因此这样的转换是等价的。

算法 1 利用排他性包含约束缩短路径表达式

```
输入：路径表达式(C1E1C2E2C3…CnEn)(C1="//")
输出：被缩短的路径表达式
BEGIN
{i=1;j=3;
  WHILE (j<=n)
    {
      在包含关系表中查找 EiαEj;
      IF(已找到)THEN  {从 Ei 到 Ej 的路径改写为 Ei//Ej;
        i=j;j=j+3;}
      ELSE IF(Ci="//"and j>3)THEN
        { E'=Ej-1的孩子节点;
          在包含关系表中找 EiαE';
          IF (已找到) THEN
            { 将从 Ei 到 Ej-1的路径改写为 Ei//Ej-1;
              i=j;j=j+2;}
            ELSE j=j+1;
        }
    }
}
```

总结 本文的工作可以看成是对路径缩短策略的改进和补充。对于路径表达式/pub/papers/paper/sections /section/title，使用文献[8]中的路径缩短策略可以缩短为//paper/sections/section/title，而使用本文的方法却可以缩短为//papers//section/title，计算它只需要两次结构连接运算，并且在文档中元素类型 papers 的实例要比 paper 的元素实例少得多，从而有效地优化了路径表达式查询。使用文[8]中的方法可以将/pub/books /book//author/name 缩短为//book//author/name，而使用本文的方法可以缩短为//books//author/name，虽然缩短后的路径长度相同，但 books 元素实例比 book 的元素实例少得多。通过以上分析可见本文提出的路径表达式的优化策略是有效的和现实可行的。今后我们将进一步探讨 XML 的完整性约束在 XML 查询优化中的作用。

参 考 文 献

1 Goldman R, Widom J. DataGuides: Enabling query formulation and optimization in semistructured databases[C]. In: Proc. of the 20th Int. Conf. on VLDB, 1997,8:436～445

2 Fernandez M, Suciu D. Optimizing regular path expressions using graph schemas[C]. In: Proc. of the 14th Int. Conf. on Data Engineering, 1998,2:14～23

3 Li Q Z, Moon B. Indexing and quering XML data for Regular Path Expressions[C]. In: Proc. of the 27th Int. Conf. on VLDB. Roma, Italy, 2001

4 Wood P T. Minimizing simple xpath expressions[C]. In: Proc. of the 4th Int. Workshop on the Web and Database (WebDB), santa Barbara, California, USA, 2001,5:21～24

5 Amer-Yahia S, Cho S, Lakshmanan L K S, Srivastava D. Minimization of tree pattern queries[C]. In: Proc. of the 2001 ACM SIGMOD Conf. on Management of Data, Santa Barbara, California, USA, 2001,5:21～24

6 Flesca S, Furfaro F, Masciari E. On the minimization of Xpath queries[C]. In: Proc. of the 29th Int. Conf. on VLDB, Berlin, Germany, 2003

7 Chan C Y, Fan W F, Zeng Y M. Taming XPath Queries by Minimizing Wildcard Steps[C]. In: Proc. of the 30th Int. Conf. on VLDB. Toronto, Canada, 2004

8 Wang G R, Sun B, Lv J H, Yu G. Query Processing and Optimization for Regular Path Expressions[J]. Journal of Computer Science & Technology, 2004,19(2):224～238

9 Al-Khalifa S, Jagadish H V, Koudas N, Patel J M, Srivastava D, Wu Y Q. Structural Joins: A Primitive for Efficient XML Query Pattern Matching[C]. In: Proc. of the 18th Int. Conf. on Data Engineering. Los Alamitos: IEEE Press, 2002. 141～152

(上接第 120 页)

4 UDDI Org. Universal description, discovery, and integration(UDDI). http://www.uddi.org

5 The globus alliance. Globus toolkits 3.0 documentation. http://www-unix.globus.org/toolkit/docs/3.0/index.html,2004

6 Milanovic N, Malek M. Current solutions for Web service composition. IEEE Internet Computing, 2004,8(6):51～59

7 Foster I, Kesselman C, Nick J M, et al. Grid Services for Distributed System Integration. IEEE Computer, 2002,35(6):37～46

8 Tuecke S, Czajkowski K, Foster I, et al. Open grid services infrastructure(OGSI) version 1.0. http:// www.ggf.org/ogsi-wg, 2002

9 Andrews T, Curbera F, Dholakia H, et al. Business process execution language for Web services version 1.1. ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf,2003

10 国家电力公司. 电力网电能损耗计算导则 DL/T 686-1999[M]. 北京：中国电力出版社，2001

11 Gamma E, Helm R, Johnson R, et al. Design patterns: Elements of reusable object-oriented software, Boston: Addison Wesley, 1998

12 DAML Services Coalition. DAML-S: Semantic markup for Web services. http://www.daml.org/services/daml-s/0.9/daml-s.html,2003

13 Liu S, Khalaf R, Curbera F. From DAML-S processes to BPEL4WS. In: Proceedings of the 14th International Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications. Boston, MA: IEEE, 2004. 77～84

14 Rao J H, Kungas P, Matskin M. Logic-based Web services composition: from service description to process model. In: Proceedings of the IEEE International Conference on Web Services. San Diego, California: IEEE, 2004. 446～453