网格与Web服务的融合—WSRF和WS-Notification

刘会斌　都志辉
(清华大学计算机系　北京100084)

摘　要　Web服务通知(WS-Notification)和Web服务资源框架(WS-Resource Framework,简称WSRF)规范提供可扩展的发布/预定通知模式和用Web服务为有状态资源(Statefu Resource)建模的能力。本文首先介绍了OGSI和WSRF的关系,WSRF可以简单地看作是OGSI1.0规范中的概念和接口的重写。WSRF通过Web服务资源(WS-Resource)方法来为有状态资源建模,通过WS-Resource方法来说明、执行Web服务和一个或多个指定类型的状态组件间的关系,阐述了通过Web服务接口来设置WS-Resource的属性和生命周期的方法,还讨论了预定/发布模式的通知机制是如何建立在WSRF基础上的,最后对WSRF和WS-Notification进行了简单总结并讨论了它的未来研究方向。

关键词　WSRF,WS-Notification,网格,OGSI,状态建模

The Convergence of Grid and Web Services—WSRF and WS-Notification

LIU Hui-Bin　DU Zhi-Hui
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084)

Abstract　The WS-Notification specification and the WS-Resource Framework (WSRF) provide a scalable publish/subscribe (pub/sub) messaging model and the ability to model stateful resources using Web services. In this paper, the relationhip between OGSI and the WSRF is introduced first. WSRF can be viewed as a straightforward refactoring of the concepts and interfaces developed in the OGSI version 1.0 specification. WSRF tries to model Stateful Resources with Web Services Resource approach to declaring and implementing the association between a Web service and one or more named typed state components. An approach is described for making the properties of a WS-Resource accessible through its Web service interface and for managing a WS-Resource's lifetime. The discussion on how a pub/sub notification mechanism (WS-Notification) can be built on top of the WS Resource framework is also given. Finally, this paper also presents a summary of WSRF and WS-Notification briefly and a simple discussion on the future research.

Keywords　WS-Notification, Grid, OGSI, State modeling

网格[1]作为信息社会的资源与服务平台,根据Ian Foster的观点,它必须同时满足三个条件:在非集中控制的环境中协同使用资源;使用标准、开放和通用的协议和接口;提供高质量的服务。本文介绍网格计算领域的一个最新成果和进展—WSRF[2]和WS-Notification[3]。在WSRF中引入了有状态资源(Stateful Resource)的概念,来对状态进行模型化。WSRF引入了WS-Resource方法(WS-Resource Approach)来表示、访问有状态资源。定义了状态的表示、操作的方法,服务的请求者能发现WS-Resource的类型和对状态进行操作,包括读取、更新和查询它的状态值,管理它的生命周期等。WS-Resource方法有利于服务之间的互操作,为不同服务提供者和服务使用者描述、访问和管理有状态资源提供了一种通用方法。

1　WSRF和WS-Notification与OGSI的关系

OGSA(Open Grid Service Architecture)[4]网格体系结构是由Globus项目组联合IBM公司在2002年6月共同推出的。OGSA是以服务为中心的"服务结构",是Web服务和网格技术融合的产物。在2003年6月,又进一步推出OGSI(Open Grid Service Infrastructure)[5]l来对网格服务的具体实现进行规范。2004年由IBM、Globus联盟和HP共同提出Web服务资源框架和Web服务通知规范是对OGSI的重写和发展,WSRF实现了网格与Web服务的融合。

WSRF和WS-Notification基本上保留所有OGSI的概念,只是在消息和相关语法方面进行了适当改动[6]。OGSI在实践中,Web服务领域发现它有三个方面的不足:(1)OGSI把繁杂的技术内容全部集中在一个规范中,不利于对不同部分的灵活运用;(2)由于OGSI超前地大量使用XML模式(Schema)等技术,不能得到现有Web服务工具环境的有力支持;(3)对于面向对象,将服务(资源)的状态、标识以及生命周期管理等完全封装在Web服务中,受到了Web服务纯化论者的批评和抵制。来自Web服务领域的不同意见阻碍了网格的广泛接受和推广,为了把网格概念特别是要求建立在一个广泛接受和支持的通用基础之上,WSRF和WS-Notification便应运而生。WSRF针对OGSI三个方面的缺点进行了以下改进:(1)WSRF把OGSI功能分成一系列功能规范,这样就可以灵活地使用各个部分;(2)WSRF减少了对XML模式的使用,这样就可以得到目前大多数Web服务工具的直接支持,而且开发者比较熟悉;(3)明确地把服务与该服务所作用的有状态资源区别开来,而不是集成在一起。在功能上,这几个规范和原来的OGSI的不同部分有很直接的对应关系,如表1所示。

刘会斌　硕士研究生,主要研究方向为网格计算。都志辉　副教授,研究方向为并行计算、网格计算。

表1 OGSI 与 WSRF 和 WS-Notification 功能的对比

OGSI	WSRF 和 WS-NOTIFICATION
Grid Service Reference	WS-Addressing Endpoint Reference
Grid Service Handle	WS-Addressing Endpoint Reference
Handle Resolver portType	WS-RenewableReferences
Service data defn & access	WS-ResourceProperties
GridService lifetime mgmt	WS-ResourceLifeCycle
Notification portTypes	WS-Notification
Factory portType	Treated as a pattern
ServiceGroup portTypes	WS-ServiceGroup
Base fault type	WS-BaseFaults

从表1可以清楚地看出，OGSI 的功能，在 WSRF 和 WS-Notification 中都有相应的体现。WSRF 和 WS-Notification 对 OGSI 的改动只是语法上的，而且很小，因此，基于 OGSI 开发的系统可以比较容易地移植到 WSRF 平台上。

WSRF 和 WS-Notification 重写 OGSI 之后，将取代原来 OGSI 的位置，作为一种新的基础设施，为基于 Web 服务的 OGSA 提供更广泛而强大的支持。WSRF 和 WS-Notification 实现了网格与 Web 服务的融合，这是一个重要的成果，它既可以充分利用已有 Web 服务领域的各种成果，又吸纳了网格技术，可以支持网格的需求，为网格和 Web 领域的发展建立了一个共同基础。

2 WSRF 和 WS-Notification 的概念和结构

“状态”是许多应用中都存在的一个重要概念，WSRF 把有状态的实体统称有状态资源，有状态资源具有特定生命周期和一组状态数据，可被一个或多个 Web 服务访问。生命周期是指资源从创建到消除这段时间。状态数据用 XML 文档来描述，通过对 XML 文档操作来对有状态资源的状态进行设置。对有状态资源进行更新、消除或其它操作时，状态就会发生改变，需要将状态变化通知给相关的资源，有状态资源通过直接方式或代理方式来通知信息。下面对有状态资源寻址、生命周期管理、状态设置和相互通信的过程、原理作简要分析。

2.1 WS-Resource 和 Web 服务寻址(WS-Addressing)

Web 服务[7]就是可被 URI (Universal Resource Identifier)识别的软件应用，它的接口用 XML (eXtensible Markup Language)来定义，并且可以通过基于 Internet 协议直接支持与其他基于 XML 消息的软件应用交互。

WS-Resource 是由 Web 服务和有状态资源合成的组件，并反映了它们之间的关系。WS-Resource 有以下两个特点：(1)组件状态用 XML 文档来描述，用 XML 文档定义它和 Web 服务的接口类型。(2)用“隐式模式”寻址和访问有状态资源，通过端点引用来寻址。“隐式”指对客户端来说，不需要了解有状态资源标识符(标识符代表有状态资源的身份信息，用来识别有状态资源)的内容，有状态资源标识符只是对被访问的 Web 服务有意义的，由 Web 服务以一种特殊方式去识别在执行请求信息过程中使用的 WS-Resource。“模式”是指它们之间关系是用现有的常规 Web 服务技术(例如，XML、WSDL)来约束的。图2说明了通过 WS-Resource 创建有状态资源的过程[8]。

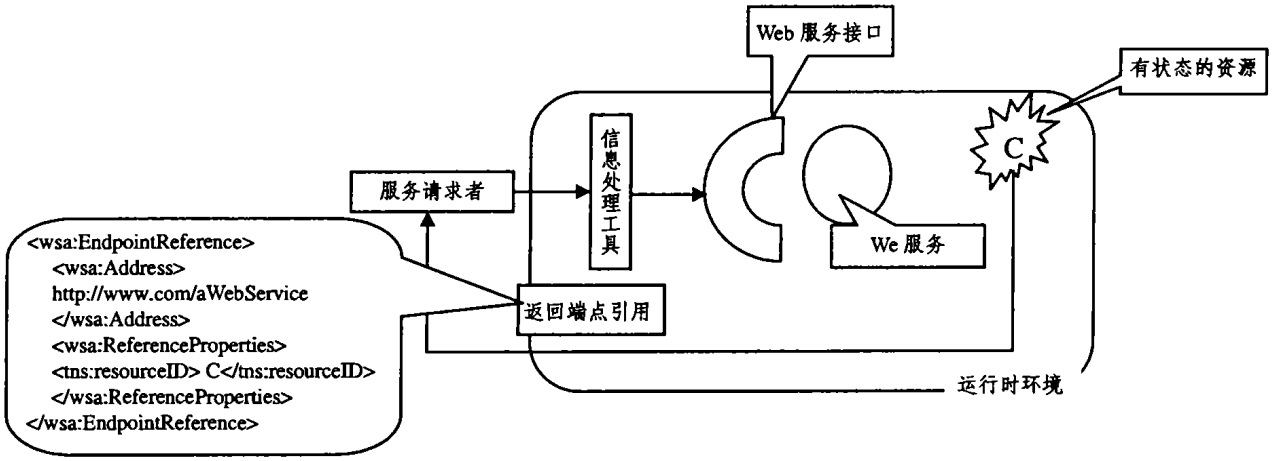

图1 具有身份标识的端点引用

服务请求者发送请求信息给 Web 服务，由信息处理工具接受请求信息，信息处理工具支持一个或多个网络传输协议，信息处理工具要翻译这些信息并进行封装，封装后的信息能被 Web 服务识别。然后，传送给 Web 服务接口，Web 服务接收请求信息并创建了一个名为“C”的有状态资源，Web 服务返回给服务请求者一个新创建的 WS-Resource 端点引用，因为这个 WS-Resource 由新创建有状态资源和 Web 服务合成的，所以，称创建有状态资源的 Web 服务为 WS-Resource 制造者。服务请求者通过端点引用来寻址有状态资源，端点引用包含了地址组件和引用属性组件这两个组件。地址组件(Address component)，说明有状态资源位置(在 HTTP 协议中，通常是 URL)。引用属性组件(ReferenceProperties component)，描述有状态资源的属性。引用属性包括有状态资源标识符(Identifier)子组件，下图说明端点引用、标识符、地址组件、引用属性组件间的关系。

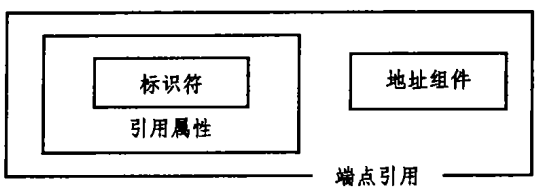

图2 端点引用的结构

包含有状态资源标识符的端点引用是有资格的端点引用(WS-Resource qualified endpoint reference)。当服务请求者要访问有状态资源，它所发送请求信息必须包含有资格的端

点引用。有资格的端点引用“隐含”在请求信息中，“隐含”指引用属性组件包含在 SOAP 的信息头(header)中，而不是在它本体信息中，被访问的 Web 服务从 SOAP 中析取有状态资源标识符，通过标识符来识别有状态资源。

2.2 WS-Resource 生命周期

当一个请求执行完成后，与之相联系的有状态资源应该被消除，以释放相关的资源，有状态资源并不是无限期存在的，它具有一定的有效期。WS-Resource 的生命周期[9]是指 WS-Resource 从创建到消除这段时间。WSRF 和 OGSI 的生命周期的管理本质上是相同的，都规定了实体创建、分配标识、消除三方面内容。表2列出了 OGSI 和 WSRF 有关生命周期主要操作的对比。

表2 OGSI 和 WSRF 生命周期操作管理的对应关系

功能	OGSI	WSRF
创建新实体	Factory portType operation “createService”	Factory pattern definition
定位新实体	Grid Service Reference Grid Service Handle	WS-Addressing Endpoint Reference with reference properties.
立即消除	GridService portType operation “destroy.”	ResourceLifetime portType operation “Destroy.”
定时消除	GridService portType “requestTerminationAfter” “requestTerminationBfore”	ResourceLifetime portType “SetTerminationTime” “TerminationTime”

WS-Resource 是通过 WS-Resource 制造者(factory)来创建的，WS-Resource 制造者是有能力创建 WS-Resource 的 Web 服务。WS-Resource 制造者能够创建一个有状态资源；给新创建有状态资源分配标识；建立新创建有状态资源和 Web 服务间的联系。OGSI 中创建有状态 Web 服务和 WSRF 中创建 WS-Resource 是同一个概念。

对服务请求者来说，新创建的实体只是在一段时间是有用的，过了一段时间，应该消除这个 WS-Resource，与之相联系的有状态资源也就被消除了。有两种消除方式：一种是立即消除，由服务请求者发送消除资源请求，消除请求信息包含要消除的 WS-Resource 有资格的端点引用，端点引用属性组件包含被消除的 WS-Resource 标识符，Web 服务根据端点引用属性组件去寻址要消除 WS-Resource，如果成功消除了，回应消除成功的信息，否则返回错误信息，宣告消除失败。另一种是定时消除，WS-Resource 被分配了一个生命周期，它有一个终止时间，过了终止时间，由资源提供者消除 WS-Resource，资源的提供者是指负责对有状态资源进行操作的组件。可通过改变终止时间来改变 WS-Resource 的生命周期。服务请求者通过“SetTerminationTime”信息来请求改变 WS-Resource 的终止时间，假如 Web 服务接收这个请求，改变了 WS-Resource 的终止时间，从而改变了 WS-Resource 的生命周期。如果改变终止时间失败，则返回错误信息，宣告设置失败。WSRF 支持定时消除的“SetTerminationTime”操作和 OGSI 的“RequestTerminationAfter”操作实现的功能是相同的。

2.3 WS-Resource 状态属性管理

服务的请求者需要了解有状态资源的状态，对有状态资源的状态进行检查、设置等操作，包括对状态读取、修改、查询等。OGSI 和 WSRF 本质上用相同的方法来管理资源状态，主要区别是采用了不同的语法[10]。

OGSI 通过声明服务数据元素来设置状态类型，服务数据元素是接口组成部分。WSRF 中，WS-Resource 的状态用 WS-Resource 属性文档来定义，资源属性文档用 XML 模式来描述。WS-Resource 属性文档集中了资源属性组件，描述了它和 Web 服务接口的关系。

WSRF 中资源属性元素和 OGSI 中的服务数据元素是几乎等同的。唯一的区别是资源属性元素用简单的 XML 全局元素说明，而 OGSI 服务数据元素说明通过 OGSI 特殊的语法来映射到 XML 元素说明。

OGSI 通过一系列的扩展操作实现对状态的读取、查询、修改等操作，WSRF 通过修改资源属性文档来设置 WS-Resource 状态。WSRF 定义了特殊操作来获取和设置属性值。例如，单个元素值的获取，多属性值获取/修改等。服务请求者可以修改一个 Web 服务多项属性值。通过发送设置属性请求来对资源属性文档元素进行插入、更新、删除操作，从而达到改变 WS-Resource 状态的目的。假如所有的设置都被成功地执行，回应设置成功的信息。否则，返回出错信息。

2.4 有状态资源的信息交换

当有状态资源被创建、消除或状态就发生改变，都需要把这些变化通知给相关的资源，资源之间的通信是通过“发布/订阅”式的消息通信模式来进行的，直接发布，另一种通过代理发布。它规范发布、订阅、提交通知的信息的交换，变化和事件可以直接或通过代理来发布。直接模式，通知产生者(NotificationProducer)直接收预定信息和发送通知信息。代理模式，由通知代理(NotificationBroker)负责产生分发信息给一个或多个通知使用者(Notification Consumer)[3]。

2.4.1 *直接通知的操作顺序* 通知的产生者通常是 Web 服务，支持一个或者多个标题(Topic)，预定者(Subscriber)通过通知产生者来获取标题列表。例如，预定者想了解“goingOffLine”的信息，发送标题为“goingOffLine”预定请求信息给通知产生者，通知产生者 Web 服务产生了一个确定的状态，包含这个标题的通知信息(NotificationMessage)就产生了。有关“goingOffLine”标题的通知信息被发送给通知使用者(Notification Consumer)，图3说明了直接方式发布通知过程[3]。

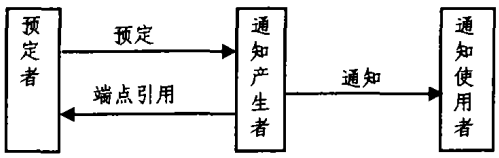

图3 直接方式发布通知过程

服务请求者，同时也是预定者，发送标题请求信息给通知产生者，指示通知使用者的地址、预定的标题和其他的相关预定信息。通知产生者创建一个预定 WS-Resource，这个预定 WS-Resource 建立了预定者和通知产生者的联系，这个预定 WS-Resource 有资格的端点引用被返回预定者，然后，通知产生者发布了一个符合这个标题的通知信息，通知产生者发送通知信息给通知使用者。

2.4.2 *代理模式下通知的操作顺序* 在有代理的情况下，支持多个实体用同一标题发布通知。例如，当预定者 A 和通知使用者 B 之间选用“goingOffLine”标题建立了预定，由通知使用者 B 接收有关“goingOffLine”标题的通知。通知发布者 C 可以选择“goingOffLine”这个标题来发布通知，C 发

布的标题为"goingOffLine"的通知被发送给通知使用者B。过了一段时间，通知发布者D也选择"goingOffLine"这个标题来发布通知，因为这个标题符合预定者A和通知使用者之间的预定，所以，D发布的标题"goingOffLine"的通知也被发送给通知使用者B。图4说明了代理模式下发布通知过程[3]。

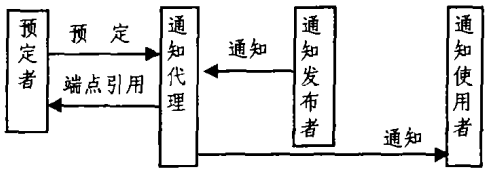

图4 代理方式发布通知过程

在有代理的情况下，通知代理起到一个中间桥梁的作用，代替了直接通知模式下的通知产生者，使用通知代理的目的是用来代替和最终通知使用者的互操作。首先，预定者发送标题请求信息给通知代理，通知代理创建一个预定资源，并返回这个预定资源端点引用给预定者。通知发布者向通知代理请求发布，然后，通知代理传递通知信息给任何符合这个预定的通知使用者。交换通知信息有两个作用：发布通知信息给代理和发布通知信息给通知使用者。

3 WSRF规范和Web服务通知规范(WS-Notification)

对有状态资源操作、管理在WSRF子规范中都有详细的说明，WSRF规范包括五个子技术规范[2]。Web服务资源生命周期(WS-ResourceLifetime)定义了WS-Resource消除管理机制，服务请求者可以立即消除或定时消除WS-Resource；Web服务资源属性(WS-Resource Properties)用XML文档定义一个WS-Resource，提供了管理WS-Resource属性的机制；Web服务更新引用(WS-RenewableReferences)说明了WS-Resource如何通过某种通知策略重新找回失效的端点；Web服务服务组(WS-ServiceGroup)定义了描述、管理不同种类的Web服务的方法；Web服务的基础错误(WS-Base Faults)定义了在Web服务信息交换发生错误时，可使用的XML错误类型。除此之外，还发布了Web服务通知(WS-Notification)规范，该规范定义了一种用发布/预定模式去预定和通知事件的方法。Web服务资源生命周期、Web服务资源属性、Web服务通知这3个规范的草稿已于2004年1月发布，不久，其他的3个技术规范也将发布。

结论 WSRF和WS-Notification提出的目的是为商业应用、网格资源和系统管理提供公共的、标准的基础设施。WSRF引入有状态资源的概念，规定通过对一个由Web服务地址和资源标识组成的端点来引用来访问Web资源，该规范提出一种用WS-Resource方式来创建、管理有状态资源方法；通过对Web服务属性文档操作来对状态进行检查和设置；通过信息交换来管理有状态资源的生命周期。WS-Notification提出了基于Web服务有状态资源间通信方式。变化和事件能够以一种标准的方式发布，然后再直接或通过代理将通知发送出去。WSRF仅仅是一个开始，未来发展趋势是建立在WSRF上的Web服务分布式管理(WS Distributed Management，简称WSDM)框架。

参考文献

1 Foster I. What is the Grid? A Three Point Checklist. July 2002. http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf
2 The Web Services Resource Framework. http://www-106.ibm.com/developerworks/library/WS-Resource/ws-wsrf.pdf
3 WS-Notification. http://www-106.ibm.com/developerworks/library/ws-pubsub/WS-PubSub.pdf
4 Foster I, Kesselman C, Nick J, Tuecke S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Jan. 2002. http://www.Globus.org/research/papers/ogsa.pdf
5 Open Grid Services Infrastructure (OGSI) V1.0 http://XML.coverpages.org/OGSI-SpecificationV110.pdf
6 Czajkowski K, Ferguson D, Foster I, Frey J, Graham S, Snelling D, Tuecke S. OGSI-Refactor. From Open Grid Services Infrastructure to Web Services Resource Framework: Refactoring and Evolution, 2004. http://xml.coverpages.org/OGSI-to-WSRFv11-20040305.pdf
7 都志辉，陈渝，刘鹏. 网格计算. 北京：清华大学出版社，2002
8 Modeling Sateful Resource in Web Services. http://www-106.ibm.com/developerworks/library/WS-Resource/ws-modelingresources.pdf
9 WS-ResourceLifetime. http://www-106.ibm.com/developerworks/library/wsresource/WS-Resourcelifetime.pdf
10 WS-ResourceProperties. http://www-106.ibm.com/developerworks/library/WS-Resource/WS-Resourceproperties.pdf

(上接第65页)

pam_authenticate的执行顺序图见图4。(1)首先，向认证服务进程发出调用PAM的pam_authenticate的申请，即而，认证服务进程调用pam_authenticate，pam_authenticate会调用具体的认证模块完成认证；(2)认证服务进程的conversation函数会在认证模块与PBASAuth之间传递认证数据，将认证模块的提示信息传给PBASAuth，从客户端取得用户名、密码回传给认证模块，客户端的输入、输出功能是由PBASConv完成的；(3)如果认证成功，认证服务进程最后组成访问许可证返回给客户端。图5为pam_authenticate的顺序图，与pam_authenticate的唯一不同是，pam_acct_mgmt执行过程中不需要用户参与，在客户端是由PBASModClient的talk_to_mod方法取得客户端MAC，不需要调用PBASConv。

结论 PAM是Linux下一个非常成熟的认证框架，本文提出的身份认证系统基于PAM开发，不仅继承了PAM的灵活性、可扩展性、稳定性，更可以利用现有的大量PAM认证模块，以减少开发认证协议的复杂度，节约成本。另外，PAM源代码的开放性，更使我们进一步的深入开发成为可能。

参考文献

1 冯登国. 计算机通信网络安全. 清华大学出版社，2001
2 韩波. 深入Linux PAM体系结构. http://www-900.ibm.com/developerWorks/cn/linux/l-pam/index.shtml，2002
3 Samar V, Schemers R. Unified Login With PluggableAuthentication Modules (PAM). RFC 86.0, Oct. 1995
4 Morgan A G. The Linux-PAM System Administrator's Guide. http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam.html，2002
5 Morgan A G. The Linux-PAM System Administrator's Guide. http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam_appl.html，2002
6 Morgan A G. The Linux-PAM System Administrator's Guide. http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam_modules.html，2002
7 Morgan A G. Pluggable Authentication Modules (PAM). http://www.kernel.org/pub/linux/libs/pam/pre/doc/current-draft.txt，Dec. 2001