第29卷 第20期 计 算 机 工 程 2003年11月
Vol.29 №20 Computer Engineering November 2003
·发展趋势/热点技术· 文章编号：1000—3428(2003)20—0006—03 文献标识码：A 中图分类号：TP393.4

基于RPC和基于REST的Web服务交互模型比较分析

许卓明[1,2]，栗 明[1]，董逸生[2]
(1.河海大学计算机及信息工程学院，南京 210098；2.东南大学计算机科学与工程系，南京 210096)

摘 要：传统Web服务采用基于RPC的交互模型，这种模型很难达到Web级的规模可伸缩性，因此制约了Web服务的普遍应用和进一步发展。在对传统Web服务交互模型的特点与不足进行分析、以及传统交互模型与基于REST交互模型进行比较的基础上，该文探讨了Web服务交互模型的发展方向。

关键词：Web服务；交互模型；SOAP；RPC；REST；HTTP

Comparative Analysis of Web Service Interaction Models: RPC-based vs. REST-based

XU Zhuoming [1,2], LI Ming [1], DONG Yisheng [2]
(1.College of Computer and Information Engineering, Hohai University, Nanjing 210098;
2.Dept. of Computer Science and Engineering, Southeast University, Nanjing 210096)

【Abstract】 Traditional Web services adopt RPC-based interaction model, this model is not appropriate for the scalability of the Web, thus restricts universal application and further development of Web services. Based on the analysis of the characteristics and the defects of the traditional interaction model and the comparison between the traditional model and REST(REpresentational State Transfer)-based interaction model, this paper discusses a development direction of Web service interaction model.

【Key words】 Web service; Interaction model; SOAP; RPC; REST; HTTP

当前Web服务中最基础的SOAP[1]协议采用的是基于远程过程调用(RPC)的交互模型，这种交互模型在相对封闭的、小的应用环境中取得了较大成功。然而，在Web这个开放、分布的环境中会产生紧密耦合和接口复杂等问题，难以达到Web级的规模可伸缩性。REST(REpresentational State Transfer)[2]是对当前Web体系结构潜在设计原则的一种描述，也是对Web最成功要素的总结。采用基于REST的交互模型的Web服务将克服基于RPC的交互模型的诸多不足，适应Web级的规模可伸缩性，促使Web服务得到普遍应用和进一步发展。

1 基于RPC交互模型的Web服务的不足

1.1 RPC模型及缺陷

SOAP是Web服务中的基础协议，它采用的交互模型是基于RPC机制的。SOAP设计的初衷之一就是使在Internet上的DCOM、CORBA和EJB等中间件之间相互沟通，而这些中间件在实现企业逻辑时基本都是通过基于RPC或方法的，所以采用RPC来作为Web服务的交互模型是理所当然的[3]。在一个相对封闭的环境中，RPC模型取得了较大成功。因为在封闭环境中，所有用户都是已知的，可以共享一个数据模型，用户直接调用发布接口的API来获得服务，服务器端的开发者也通过接口获得了很大的灵活性。

然而，在Web环境，尤其是当应用达到了Web级的规模可伸缩性，就会出现一些问题。首先，使用RPC的方法会导致调用者和服务之间的紧密耦合。举例来说，在一个订单处理系统中，可能会使用信用卡授权服务，在客户端使用RPC方法调用了服务后，它必须等待，直到服务器端处理完成后，才能进行订单处理的动作。整个订单服务将会持续很长时间，而RPC不支持长时间的连接活动。其次，RPC系统在Internet的规模下产生接口复杂性。接口与实现它的对象或服务之间都有一定的约定，如方法调用的顺序、参数类型等，这样，每个接口都具有自己的语义。随着接口个数的增加，接口的语义有可能以接口个数平方增长，这在Web级的规模上会产生接口复杂性[3]。

1.2 基于RPC的SOAP存在的不足

当前基于RPC机制的SOAP协议存在一些不足，主要表现在：(1)安全性。RPC交互模型下是通过调用方法来访问服务的，所以要访问的服务对象名称藏在方法的参数中，无法在代理服务器一级进行有效的控制。而安全问题是企业计算环境中需要面对的关键问题，也是Web服务达到大规模商业应用的一个障碍。(2)代理和缓存。由于采用RPC模型，服务调用者要访问的资源和调用的方法被封装在SOAP消息中，单从URI和HTTP上无法得到有用的信息，因此在Web环境下，SOAP在支持代理和缓存服务器方面有一定的困难。(3)使用复杂性。由于在SOAP中可以任意定义自己的方法集合，因此要有一个描述和发现机制，使服务调用者获得服务及它提供的方法的语义后，才能根据这些信息进行调用。

2 REST及其优势

2.1 REST的产生和定义

REST是Roy Fielding在他的博士论文[5]中发明的一个新名词，它代表REpresentational State Transfer，是对Web体系结构设计原则的一种描述。REST的目的是决定如何使一个良好定义的Web程序向前推进：一个程序可以通过选择一个带有链接的Web页面上的超链(状态变换)，使得另一个Web

基金项目：国家自然科学基金项目(6017036)
作者简介：许卓明(1965－)，男，博士生、副教授，主研方向为数据库、Web服务及语义Web技术；栗 明，硕士生；董逸生，教授、博导
收稿日期：2003-01-07 **E-mail**：zmxu@acm.org

页面(代表程序的下一个状态)返回到用户，使程序进一步运行。如图1，客户请求某航空公司747飞机的服务(使用逻辑URI)，返回结果页面中包含有航班次和时刻表两个链接，这两个链接是对服务端提供资源的表示(representation)，而不是直接对资源的操作动作。得到返回结果后，客户选择一个链接来决定下一步动作，这样可以做到客户维护自己的程序状态。

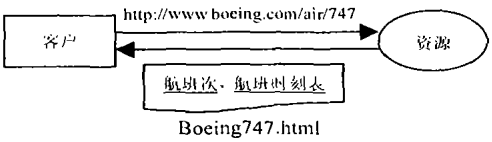

图1 REST状态迁移图

REST不是一个协议，它是当前Web体系结构的风格，是对当前Web体系结构设计原则的一种抽象和描述。从这个意义上讲，Web是REST系统的一个实例。REST描述了如何设计和开发分布式系统。对REST的应用只能是理解它，把它的原则应用到Web服务的设计中。

2.2 REST的目标及设计原则

REST的目标[5]是：

(1)组件相互作用的规模性；

(2)接口的一般性；

(3)组件的独立分发；

(4)支持中间媒介。

REST系统中的组件必须遵守下列约束[5]：

(1)所有的资源都是通过URI来标识；

(2)组件的相互操作是无状态的；

(3)资源的操作要通过对它的描述进行；

(4)自描述的消息；

(5)超媒体是程序状态的驱动器；

(6)REST强调中间媒介的作用(包括代理服务器、缓存服务器和网关)。

因为REST是对当今Web体系结构设计原则的一种描述，所以REST的目标和原则是对当今Web中已成功应用的要素的总结。从我们正在应用的Web上讲，Web上的每个资源通过一个URI来标识，可以通过简洁通用的接口(如HTTP的GET和POST)来操作Web上的资源。资源使用者与资源之间有代理服务器、缓存服务器来解决安全及性能等问题。REST系统中的组件必须是自描述的，这样，客户可根据这些自描述信息来维护自己的程序状态。

2.3 REST的主要思想

在REST系统中，所有资源都有一个URI，包括Web服务也可以用URI来标识。用以资源标识的URI最好是逻辑URI，而不是物理URI(如图1)。使用逻辑URI的好处是对服务器端的资源修改不影响客户的使用。逻辑和物理URI举例如下：

一个逻辑URI：http://www.boeing.com/air/747

一个物理URI：http://www.boeing.com/air/747.html

使用HTTP的GET、POST、PUT和DELETE 4个动作作为资源的通用接口，用户通过它们访问资源。在此，HTTP作为一个程序协议来使用，而不是只作为传送一个SOAP消息的传输协议来使用。

EST在对请求的响应数据中包含链接，这一点是非常重要的(如图1)。在响应数据中保持对其他资源的链接可以使客户程序是自推进的，因为响应信息中就包含了程序的下一步动作，这样就可以使客户程序维护自己的状态，而且以资源为中心的Web服务在本质上是易于集成的。

以一个订单服务为例。由于Web服务可以通过一个URI来访问，因此调用服务就十分简单。例如：我们需要得到某处的订单列表，假如有一个Web服务GetList实现此功能，那么使用http://www…/GetList/即可得到订单列表，客户只需这样调用，而至于服务器端如何实现，对客户来说是透明的，返回结果如图2所示。返回响应数据中包含对各个订单的链接，客户通过选择合适的链接，可以将对该链接指向的资源的描述迁移到客户，实现客户状态的自维护，这也是REST的关键特征。如果想以XML的形式返回，只需在URL后面加上参数，但附带参数的设计要尽量简单。

```
<?xml version="1.0" encoding="UTF-8"?>
<items xmlns="…">
  <item>http://www…/po_1</item>
  <item>http://www…/po_2</item>
</items>
```

图2 返回结果的XML文档

如果要提交订单，只需在客户端创建一个符合订单模式(比如：PO.xsd)的实例，通过HTTP的POST方法提交，而返回的响应数据中包含对此订单的URL.

2.4 REST的交互模型简述

从以上对REST的介绍中，我们知道REST的交互模型就是当前Web中正在应用的交互模型。每一个资源都用URI来标识，客户使用HTTP协议来访问资源，通过HTTP中GET、POST等接口对资源做不同的操作。在客户与请求服务器之间可通过HTML或XML进行数据传输。这种交互模型同样可以适用于Web服务，每一个Web服务可以通过URI来标识，我们通过HTTP所提供的通用简洁的接口来访问服务(因为服务也可看成一种资源)，服务调用者通过带有资源描述的响应结果来维护自身的状态，使客户程序向前进行。

3 两种交互模型的比较

3.1 地址模型

REST体系结构利用当前Web的地址模型(URI、DNS)，这样做的最大优点是现有的地址模型是标准化的[6]。对所有的Web资源(包括Web服务)都赋一个URI，这样可以使用标准的发现和引用资源的方法使用Web服务。而RPC模型的SOAP定义了自己的地址模型(如图3)，每一个Web服务都有一个进入点，通过URI标识，这个进入点也就是SOAP消息的处理器，服务资源的请求定向到这个处理器，而不是直接定向到所请求的资源,因此,资源与URI不是一一对应的,而是通过一个定制的、具体服务的地址(如图3的URL1)来标识。

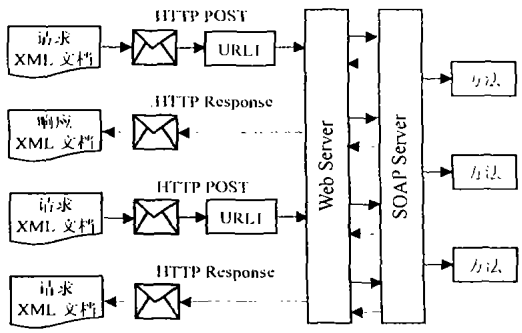

图3 RPC模型的SOAP请求-响应过程

RPC模型的SOAP的地址模型会有以下问题：代理服务

器很难很好地工作，因为URI不是指向具体资源的，而是SOAP消息服务器；其次，基于URI的一些技术(XSLT等)不能很好地工作；最后，集成工作也需要定制的逻辑。

3.2 通用接口和互操作性

REST强调使用标准的、通用的操作，也就是HTTP提供的GET、PUT、POST和DELETE[6]，通过这4个接口对基于URI标识的资源作统一的处理。采用RPC机制的SOAP不提供通用操作，每一个服务可以定义它自己的操作集(即定义自己的方法调用)[1]。正是因为这种定义的任意性，要想使得服务被别人使用，就得有一种描述和发现机制(在实际中对应WSDL[1]和UDDI[1])。

RPC模型的SOAP不提供通用接口意味着要求客户端必须知道服务描述和发现机制的知识；客户在使用服务前要预先知道服务的语义。这种定义的任意性也会降低服务间的互操作性。

REST模型的通用接口符合以往成功的经验。例如在数据库中，可以通过SELECT、INSERT、UPDATE和DELETE等几个简单的词汇在数据库中建立一个非常复杂应用逻辑；在文件系统中，可以通过读和写来操作各种数据。所以，使用通用简洁的接口来处理复杂的逻辑的这个原则同样可应用在Web上。Web可以被看成一个巨大的信息库，Web服务由于可以被唯一的URI来标识，同样可以被看成是一种Web资源。

3.3 中间媒介

REST模型很重要的目标就是中间媒介的兼容性。REST系统中所有的动作和要访问的资源都可以从HTTP和URI中得到[6]，这使得代理服务器、缓存服务器和网关很好地协调工作。而RPC模型的SOAP要访问的资源仅从URI无法得知，要调用的方法也无法从HTTP中得知，它们都隐藏在SOAP消息中[6]。

举例来讲，如果一个公司发布了3个资源：资源1、资源2、资源3，客户通过代理服务器来访问这3个资源，现在公司决定不允许访问资源3，它就可以在代理服务器上进行设置。这样，当客户访问时，代理服务器在它的URI上进行判断控制。这在PRC模型的SOAP中很难控制(如图4)，因为URI指向一个SOAP消息服务器，请求访问的资源信息隐藏在SOAP消息中，代理服务器仅在URI上得不到访问资源信息，所以很难控制资源3的使用，除非能理解每个SOAP消息的语义。同样的，在REST系统中的代理服务器还可以通过HTTP的动作(GET、POST)来进行控制。

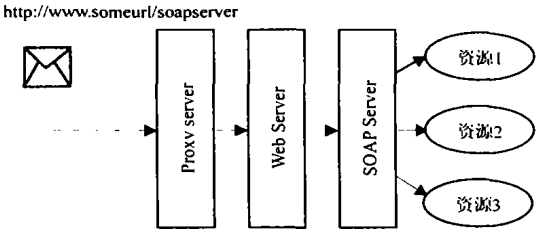

图4 RPC模型的SOAP代理服务器工作图

3.4 安全性

REST使用了简单有效的安全模型[6]。REST中很容易隐藏某个资源，只需不发布它的URI；而在资源上也很容易使用一些安全策略，比如可以在每个URI针对4个通用接口设置权限；再者，以资源为中心的Web服务是防火墙友好的，因为GET的意思就是GET，PUT的意思就是PUT，管理员可以通过堵塞非GET请求把资源设置为只读的，而现在的基于RPC模型的SOAP一律工作在HTTP的POST上。使用REST，可以利用现有的安全模型，大大简化了安全问题的难度。

使用SOAP RPC模型，要访问的对象名称藏在方法的参数中，因此需要创建新的安全模型。现在的商业Web服务架构有Microsoft的Passport、Sun的Liberty Project等，但这些服务架构都还不成熟，有待于进一步发展。安全性已成为Web服务商业应用的最大障碍[4]。

4 REST应用现状

通过上文对两种交互模型的比较分析，可以知道：在Web级的规模上，基于RPC模型构建的分布式系统会产生诸如紧密耦合和接口复杂等问题；基于REST交互模型的Web服务可适应Web级的规模可伸缩性、解决现有采用RPC模型所存在的一些问题。目前，工业界的Web服务基本上采用基于RPC的交互模型。而W3C正在发展Web服务的体系结构[7]，其中REST与Web服务体系结构之间的关系也正在W3C的TAG(Technical Architecture Group)热烈讨论。因此，基于REST交互模型的Web服务还没有在工业界大规模推广应用，但是，一些探索性应用开发已着手进行，如：Paul Prescod已经利用REST的设计原则改善了Google提供的Web搜索服务的性能[8]；在Amazon和eBay也根据REST原则实现了用HTTP＋URI＋XML的服务[6]。

5 结束语

本文通过对基于RPC和基于REST交互模型的Web服务之间的比较分析，说明了在Web级规模上，RPC交互模型的不足、REST交互模型的优势。REST是Web服务交互模型的发展方向，其必然会对Web服务的体系结构产生重要的影响[3]。基于REST的交互模型的Web服务必将推进当前Web服务技术的大规模应用和进一步发展，使Web服务真正成为基于Web的服务。

参考文献

1 Curbera F.Unraveling the Web Service Web－An Introduction to SOAP,WSDL,and UDDI.IEEE Internet Computing,2002,6(2):86-93
2 Fielding R T.Principled Design of the Modern Web Architecture. Proc. of the 2000 Intl.Conf. on Software Engineering (ICSE 2000),Limerick, Ireland,2000-06:407-416
3 Vinoski S.Web Service Interaction Models - Part II: Putting the Web into Web Service. IEEE Internet Computing,2002,6(4):90-92
4 Clark D. Next-generation Web Services. IEEE Internet Computing, 2002,6(2):12-14
5 Fielding R T.Architectural Styles and the Design of Network-based Software Architecture. Doctorial Dissertation, Dept. of Computer Science, Univ. of California, Irvine, 2000
6 Prescod P.REST and the Real World.Online Article,O'Reilly & Associates, Inc.,Online Available at: http://www.xml.com/pub/a/ws/2002/02/20/rest.html,2002-02-20
7 Austin D,Barbir A,Garg S.Web Services Architecture Requirements. W3C Working Draft,Online Available at: http://www.w3.org/TR/2002/WD-wsa-reqs-20021114,2002-11-14
8 Prescod P.Googles Gaffe. Online Article, O'Reilly & Associates, Inc., Online Available at: http://www.xml.com/pub/a/ws/2002/04/24/google.html,2002-04-24