基于XML的RPC技术分析*)

罗 玲 白晓颖
(清华大学计算机科学与技术系 北京100084)

摘 要 RPC(Remote Procedure Call,远程过程调用)是支持分布式应用系统之间通讯的一种重要机制。基于XML(eXtansible MarkupLanguage,可扩展标识语言)的RPC技术采用工业界的信息交换标准XML的消息格式封装RPC。由于XML技术提供了一种开放的、有语义的讯息机制,基于XML的RPC技术可有效支持网络环境下异构平台上应用系统的互操作。本文在分析RPC技术的主要机制的基础上,对XML的RPC技术和传统的RPC技术进行了比较,并讨论了当前两种主要的基于XML的RPC协议XML-RPC和SOAP RPC。文章最后分析了一个简单的在Java SOAP RPC协议上实现的应用实例。

关键词 RPC(远程过程调用),XML(可扩展置标语言),SOAP(简单对象访问协议)

A Review of the XML-Based RPC Technology

LUO Ling BAI Xiao-Ying
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084)

Abstract RPC(Remote Procedure Call)is an important mechanism for application communications in a distributed environment. XML-based RPC encapsulates RPC messages using XML, the industry standards for information exchanging. As XML is a self-description open standard, XML-based RPC can greatly improve the mutral-understanding and interoperability among applications on heterogeneous platforms. This paper reviews the basic RPC mechanism and compares XML-based RPC with traditional RPC technique. The paper also introduces and analyzes two typical realizations of XML-based RPC: XML-RPC and SOAP RPC. A simple example using Java SOAP RPC protocol is presented to illustrate the approach.

Keywords RPC (Remote Procedure Call), XML (eXtensible MarkupLanguage), SOAP (Simple Object Access Protocol)

1. 概述

RPC(Remote Procedure Call,远程过程调用)是支持分布式应用系统之间通讯的一种重要机制。RPC技术采用软件代理的模式,实现位于不同地址空间的执行程序之间的通讯。RPC技术的提出,使得协同工作的应用程序可同时分布于多台主机上,方便地实现了跨越多个操作系统、多种网络协议的应用程序的开发,提高了网络上主机资源的共享[1]。

网络的普及与网络技术的发展对于分布式异构环境中应用系统之间的互操作提出了新的挑战。基于XML的RPC技术采用XML描述远程过程调用的接口和参数,利用XML的可伸缩性来包装和交换RPC调用的相关信息。由于XML具有平台/环境无关性、可扩展性好、自描述性、结构化和标准化等特点,与传统RPC技术相比,基于XML的RPC技术具有更为通用的编码形式,支持更加灵活的用户自定义数据类型的映射,能够更有效地实现跨越不同软/硬件平台的应用系统之间的互理解和互操作,为解决由于语言、系统软件、协议、数据等差异所造成的高代价的应用系统集成奠定基础。

目前,基于XML的RPC技术主要有两个代表:Userland Software Inc.的XML-RPC和W3C联盟的SOAP RPC。XML-RPC是最早推出的使用HTTP做为传输协议、XML做为编码规则的远程过程调用,简单有效是其最大特点。SOAP RPC是新一代Web应用架构Web Service的基础和核心协议,具有可定制的特点,支持用户自定义数据的传输,并得到了包括IBM,Microsoft在内的大型软件公司的支持,是W3C联盟力推的工业界标准。

本文首先简要介绍RPC技术,概述了RPC技术的历史和发展状况、实现机制、RPC应用程序的运行过程。然后,文章着重讨论了基于XML的RPC技术,介绍了XML技术的特点,对基于XML的RPC技术与和传统的不基于XML的RPC进行了比较,并分析比较了两种主要的基于XML的RPC技术:XML-RPC和SOAP RPC。最后文章通过一个Web Services的例子,说明了其中SOAP RPC的实现,是Web应用中基于XML的远程调用机制使用的体现。

2. RPC技术简介

远程过程调用概念的提出至少可以追溯到1976年,全面的实现出现在70年代后期和80年代早期[1]。传统的RPC实现主要是OSF(Open Software Foundation)提出的DCE(Distributed Computing Environment)RPC和Sunsoft,Inc.开发的ONC(Open Network Computing)RPC。现在,在不同的应用平台上,如Windows(3.1,NT,95),Macintosh,26 variants of UNIX,OS/2,NetWare,and VMS等,都有了开发

*)国家科技攻关项目:奥运会信息系统集成测试总体方案及集成测试管理平台的预研,编号:041250001。

RPC 应用程序的工具[2]。

简单地说，RPC 就是过程的实现(类似于函数体)在远端的机器上，过程的声明(函数原型)在本地机器上，通过本地机器的过程声明调用过程。一个典型的 RPC 通讯过程如图1所示。

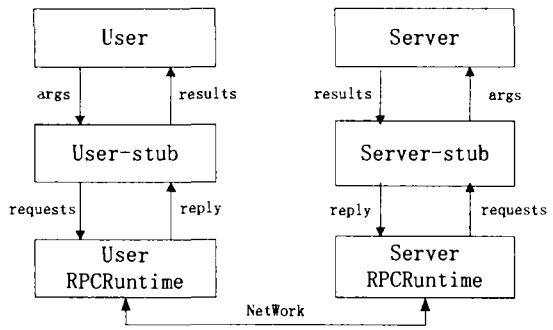

图1 RPC 通讯过程

调用程序 User 通过远程服务的本地代理 User-Stub 向远端的过程(Server)发出请求信息。User-Stub 对过程调用和参数信息打包封装，并通过 User RPCRuntime 与特定的网络传输协议绑定，将封装后的请求传递给被调用程序 Server。Server-Stub 解析 User 的请求，接受并封装过程调用的响应结果，通过 Server RPCRuntime 与网络传输协议绑定，回传结果信息。User-Stub 接受并解析结果，回传给 User。

消息映射和消息处理是 RPC 技术中两个重要方面。消息映射包括数据类型的映射和函数方法的映射，是在调用程序和被调用程序之间，将过程调用的函数方法相互绑定、将参数的数据类型正确对应的过程，是异构平台上程序间互理解的基础。消息处理包括消息封装、消息传输及消息解析。消息封装和消息解析相呼应，由桩(Stub)程序完成对调用请求信息及结果信息的正确映射。消息体系结构是消息封装和解析的依据，由消息体和数据模型定义组成，其中消息体包含了调用方法和参数信息，数据模型定义说明了函数方法、命名方法、基本和复杂数据、请求或应答消息结构等。RPC 在网络体系结构中属于应用层的协议，消息传输就是与底层的传输协议(如 HTTP，TCP，UDP)绑定，并根据传输协议所定义的请求头、响应头的结构规范，提供传输协议的信息。

3. 基于 XML 的 RPC 技术

RPC 实现的多样性和相互间的不兼容，使得用一种 RPC 实现后，后期的维护以及进一步的扩展都要依赖于同一开发者和开发环境。这对于系统的灵活性、可维护性、移植性和互操作性等方面都造成负面的影响。XML 的出现，给 RPC 技术也带来了新的发展。

XML[3,4]是由 W3C 于1998年2月发布的一种标准。它是 SGML 的一个简化子集，以一种开放的自描述方式定义了数据结构，在描述数据内容的同时能突出对结构的描述，从而体现出数据之间的关系。XML 技术具有语言中立、扩展性好、结构化文档、简单易用、工业界支持等特点，基于 XML 的 RPC 的提出使 RPC 技术的通用性、有效性都得到很大的提升。

3.1 基于 XML 的 RPC 技术与传统的 RPC 技术的比较

表1从消息映射和消息处理两方面，对基于 XML 的 RPC、ONC RPC[5]及 DCE RPC[6,7]进行了比较[8]。

表1 RPC 技术比较

		基于 XML 的 RPC	ONC RPC	DEC RPC
消息映射	数据类型映射	XML Schema 或 DTD 进行数据建模	XDR(外部数据表示)公共的数据表示	机器的基本数据表示，以及自定义的 RPC Data Types
	函数方法映射	通过 XML 的语义性和 XML 大纲或 DTD，解析到消息中的函数方法名，主机地址端口等信息实现 RPCBinding	根据程序号，版本号，过程号来调用一个函数	根据 IDL 定义的接口，地址，对象等信息实现 RPC Binding
消息处理	消息封装	封装成 XML 格式的消息，通过 XML Schema 和 DTD 可以灵活地定义消息的细节信息	自定义消息格式。传输的消息格式是固定的，如要有整型的数据表示程序号，版本号过程号等	自定义消息格式。消息格式也是固定的。如 Binding Handle 数据类型定义过程调用的绑定信息
	消息传输	消息封装成 HTTP 格式传输	自定义消息传输协议，底层基于 TCP/UDP 协议	自定义消息传输协议，底层基于 TCP/UDP 协议
	消息解析	XML Paser	特定编译器解析相应的消息内容	特定编译器解析对应的消息内容

·基于 XML 的 RPC

消息映射：使用 XML 作为编码格式，XML Schema 或 DTD(Data Type Definition，文档类型定义)作为数据的基本建模工具，以 XML 数据类型为中介，实现从客户端到服务器端应用程序中数据类型的映射。

应用程序的调用请求转换成遵循 XML 标准的消息请求，根据 XML Schema 或者 DTD 的定义，服务器端可以解析调用请求名、参数、地址等信息，实现函数方法的映射；服务器将结果返回再封装成 XML 形式，由客户端完成返回数据类型的映射。

利用 XML 语言的自描述性和可扩展性，可以有效定义复杂的数据类型，定制函数调用信息。数据交换的跨平台性好、灵活性高。

消息处理：由于 XML 具有语言中立的特点，基于 XML 的远程过程消息便于异种平台的互相理解。

采用 XML 的编码规则定义消息体和数据模型，即消息体系结构，灵活有效地描述消息内容。客户端(服务器端)采用 XML 格式打包调用请求(调用结果)，再封装成 HTTP 的请求或应答进行传输。将消息封装成 HTTP 格式的好处是继承现有技术，屏蔽了很多很难实现的技术细节，实现简单有效。

·传统的 RPC 技术

消息映射：ONC RPC 使用的是 External Data Representation(XDR)[9]外部数据表示(XDR)。XDR 提供了一种与体系结构无关的表示数据，解决了数据字节排序的差异、数据字节大小、数据表示和数据对准的方式。DCE RPC 通过具有调用机器的基本数据表示的特征调用来解决不同的机器的不同数据表示问题。DCE RPC 还有自定义的数据类型，特别是定义了用于描述调用信息的一些数据结构，如 Binding Handle 数据结构，定义了过程调用的绑定信息。

ONC RPC 在描述一个调用信息的时候，需要程序号、版本号、过程号等信息，根据通信双方的约定完成函数方法的绑定[10]。DCE RPC 用接口定义语言（IDL）建立接口定义（interface definition）。IDL 编译器把 IDL 接口定义转换成与客户机和服务器相连的桩（stub）。客户机上的桩可加入到服务器的过程，而服务器上的桩也可加入到客户机过程。位于客户机服务器的 RPCRuntime 与桩（Stub）合作，来提供 RPC 操作。

传统 RPC 技术中，通信双方必须使用同一种机制。如果客户端使用的是 ONC RPC 技术，那么服务器端就要有相应的可以解析 XDR 数据的编译器，对接受到的信息的解析方式要按照 ONC RPC 调用的消息格式。同样对于使用 DCE RPC 的客户端发出的调用请求，服务器需要清楚 DCE RPC 的机制。

消息处理：ONC RPC 和 DCE RPC 的消息格式是定义好的，不具有基于 XML 技术的可定制性。ONC RPC 调用消息要有程序号、版本号和过程号，通信双方约定通过这些信息来决定调用对象和调用方式；DCE RPC 自定义数据类型封装绑定信息，也是双方协商固定的格式，所以可理解性也只能局限在使用同种机制的平台上。RPC 是应用层的协议，传统方法的应用自定义了自己的应用层的传输格式，没有像基于 XML 技术的调用使用的是现有的 HTTP 协议。ONC RPC 和 DCE RPC 在传输层的协议使用的是 TCP 或 UDP。

综上所述，基于 XML 和传统的 RPC 的主要区别在于消息的编码格式上。编码格式的不同，影响着处理方式的不同。同时 XML 以及 XML Schema 带来的 XML 的自描述性，给 XML 文档赋予了丰富的语义；XML 作为数据编码格式的通用性、简单性和可扩展性，使基于 XML 的 RPC 的处理方式更为灵活和简便。

3.2 SOAP RPC 与 XML-RPC

SOAP RPC 与 XML-RPC 是目前两种主要的基于 XML 技术的 RPC 协议。

·SOAP 协议与 SOAP RPC　SOAP RPC 是 SOAP 协议的一部分。SOAP（Simple Object Access Protocol，简单对象访问协议）是在分散、分布式的环境下交换信息的轻量级的协议[11]，是 WebServices 架构[12]中的一个重要的协议（如图2所示）。

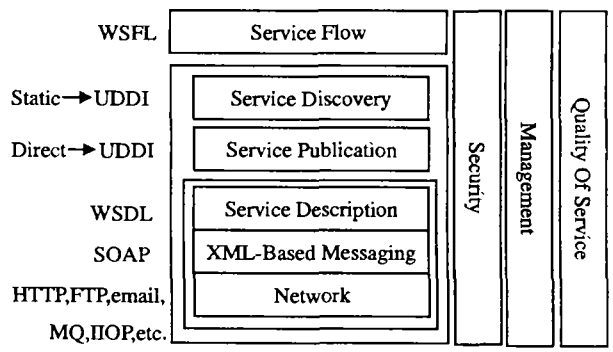

图2　Web Service Conceptual Stack

在该体系中，SOAP 定义了服务间通信的一种简单的机制[11,13]。每个处理 SOAP 的结点表现为一个 Web 服务，这些 Web 服务通过 SOAP 消息完成交互和远程调用，SOAP 为调用 Web 服务提供了一个基本的消息调用机制。

图3是 SOAP 构架，包括信封（Envelope）、编码规则（Encoding Rules）、远程过程调用和响应（SOAP RPC）三部分。信封定义了一个消息框架，描述消息的内容是什么，是谁发送的，谁应当接受并处理它以及如何处理。编码规则定义了一系列的机制用于交换应用程序定义的数据类型的实例。绑定（Binding）定义底层通信协议，进行消息交换。SOAP RPC 利用 XML 的可伸缩性和可扩展性包装和交换 RPC 调用，它的实现依赖 SOAP 信封定义的消息框架、依据编码规则定义的数据类型。SOAP RPC 也是以 SOAP 消息为载体请求和响应调用的。

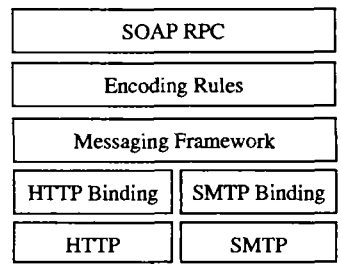

图3　SOAP 构架

·XML-RPC　是在 Internet 上使用的远程过程调用协议[14]，它使用 HTTP 做为传输协议，XML 做为编码规则，规定了在异种操作系统下软件运行的规范和一系列的实现。它的设计力求简单，以能够简单有效的请求和响应信息为根本目的。

表2　XML-RPC 与 SOAP RPC 比较

特性	XML-RPC	SOAP RPC
支持基本数据类型	yes	Yes
结构（structs）	yes	Yes
队列（arrays）	yes	Yes
命名的结构和队列（named structs and arrays）	no	Yes
详细的错误处理（detailed fault handling）	yes	Yes
易于学习（short learning curve）	yes	No
开发者自定义字符集（Developers specified character set）	no	Yes（US-ASCII，UTF-8，UTF-16）
开发者自定义类型（Developer defined data types）	no	Yes
稳定性（stability）	Yes	No
要求客户理解（require client understanding）	no	Yes
多态访问（Polymorphic Accessors）	No	Yes

XML-RPC 的请求/应答消息是 HTTP-POST，消息体是 XML 格式。过程调用传递的参数可以是基本的数据类型 numbers，strings，dates 等，也可以是复杂的记录，结构列表等。

1）请求体的格式：请求体是一个〈MethodCall〉结构，一个〈MethodCall〉必须包含一个〈MethodName〉子项，这个子项说明将要调用的方法的名称。如果远程过程调用中有参数，那么〈MethodCall〉结构中要包含〈params〉子项来描述参数，〈params〉中可以包含多个〈param〉，每个〈param〉描述的参数要有值〈value〉。

2）响应体的格式：响应体是一个 XML 结构的〈MethodResponse〉，这个结构中可以有一个参数〈params〉结构，包含一个〈param〉及其参数值〈value〉，即返回值。

·SOAP RPC 与 XML-RPC 比较　SOAP RPC 优于 XML-RPC 在于实现了用户自定义数据类型传输，增强了消息过程控制等能力，但复杂度也相应的提升。

表2[15]比较了两种技术的主要技术指标。

XML-RPC 主要具有以下特点：

1）数据类型支持：基本数据类型、结构、队列。

2）稳定性：XML-RPC 的已经是比较成熟的协议，发展的时间长，也不是为某一个组织垄断，是稳定的开源的协议。

3）简单性：这是 XML-RPC 的最大特性。容易理解、实现和调试。语法简单，容易避免错误发生。

SOAP RPC 的主要特点表现在：

1）数据类型支持：基本数据类型、结构、队列、命名队列或结构、多态访问、用户自定义数据类型等。命名队列或结构是说队列中包含的元素可以有一个"id"属性，表明队列或结构。SOAP 支持的多态访问是指访问者可以在运行时选择合适的类型的值。

2）可定制：SOAP 的最大特性也是它可以弥补 XML-RPC 缺陷的地方：可以定义消息的每一部分。它允许开发者在消息中加入自己想加入的信息。当然这是以在消息解析端增加更多工作来解析信息为代价的。

3）大公司的支持。IBM 和微软的支持是 SOAP 技术发展的很大动力。

XML-RPC 的主要缺陷有：

1）方法调用：XML-RPC 通过方法名属性来调用远程方法，这些属性规定只能是大小写的 A-Z、0－9的数字、下划线、点、冒号、斜线。这样在参数是一个对象的时候这种调用就很困难了。

2）命名的数据结构：XML-RPC 的结构和队列都是匿名的，当传输多结构或队列的时候就要依赖参数传输的顺序。这个问题不是很严重，但是更多时候为了更灵活地处理数据对不同的结构或队列命名应该更好。

3）简单性：它同时也是 XML-RPC 的一个最大的缺点。对于大多数的远程过程，它是可以实现。但是对于一些特殊的情形，如传输的参数是一个对象，或者需要让接受者知道所传的消息中哪个部分是有用的等更灵活的处理情况时，XML-RPC 的简单的局限性就暴露无遗了。

SOAP RPC 的主要缺陷有：

1）稳定性：SOAP 协议还在研究阶段，它还不是一个官方的标准，还在不断的修正和完善中。

2）文档化：当前的许多 SOAP 规范文档还在不断的纠错和变更中，所以很可能当前实现的某些 SOAP 应用和今后的就会出现不兼容。

4. 实例分析

本应用实例在 Windows 2000 Professional 的平台上，利用 Java 提供的 Web Service 开发软件包 JWSDP(Java Web Service Developer Pack)，通过 Java 的 SOAP RPC 机制，来实现一个"Hello World"的远程过程调用。

本应用实例的结构为：(1) Web 服务接口定义为 HelloIF.java；(2)程序 HelloImpl.java 实现接口中定义的 sayHello (String s)方法。调用服务返回的结果为"hello World"；(3)客户端 HelloClient.java 通过 Java 的 SOAP RPC 机制，远程调用 sayHello(String s)方法。应用程序的运行过程如图4[16]所示。

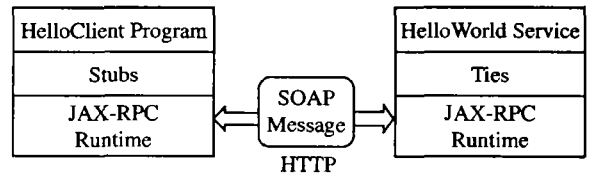

图4　应用实例运行过程

HelloClient 应用程序调用桩(stub，注：远程服务的本地代理)的一个方法。桩触发 JAX-RPC 运行系统的相应例程。运行系统将远程过程调用转换成 SOAP 消息作为一个 HTTP 请求传输到服务器端。

·在服务器端接受到 HTTP 请求时，JAX-RPC 运行系统抽出其中的 SOAP 消息，将其翻译成一个方法调用，并触发服务器端的 Tie 对象的方法。

·Tie 对象调用 HelloWorld 服务实现程序。

·服务器端的运行系统将调用结果转换并封装成 SOAP 消息作为 HTTP 响应会传给客户端。

·在客户端，JAX-RPC 运行系统抽取 SOAP 消息，通过翻译，将方法响应信息返回给 HelloClient 程序。

下面的 XML 描述的是服务发布的信息，是使用 deploytool 发布服务时系统根据打包时用户提供的信息自动生成的。

```
〈?xml version="1.0" enconding="UTF-8"?〉
〈webServices
  xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/dd"
  version="1.0"
  targetNamespaceBase="http://com.test/wsdl"
  typeNamespaceBase="http://com.test/types"
  urlPatternBase="/ws"〉
  〈endpoint
    name="MyHello"
    displayName="Hello World Service"
    description="A simple web service"
    interface="hello.Hello IF"
   implementation="hello.HelloImpl"/〉
  〈endpointMapping
    endpointName="Myhello"
    urlPattern="/hello"/〉
  〈/webServices〉
```

下面列出的服务描述是根据上述信息生成的。

```
〈?xml version="1.0" encoding="UTF-8"?〉
〈definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
      xmlns:tns="http://com.test/wsdl/MyHello"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:xsd="http://www.w3.org/2001/XML Schema"
      name="My Hello"
      arget Names pace="http://com.test/wsdl/My Hello"〉
〈ytpes/〉
〈message name="Hello IF_say Hello"〉
 〈part name="String_1" type="xsd:string"/〉〈/message〉
〈message name="Hello IF_say Hello Response"〉
 〈part name="result" type="xsd:string"/〉〈/message〉
〈port Type name="Hello IF"〉
 〈operation name="say Hello" parameter Order="String_1"〉
  〈input message="tns:HelloIF_sayHello"/〉
〈/operation〉〈/portType〉
 〈binding name="HelloIF Binding"type="tns:HelloIF"〉
  〈operation name="say Hello"〉
   〈input〉〈soap:body
encoding Style = http://schemas.xmlsoap.org/soap/encoding/use
 ="encoded"
 namespace="http://com.test/wsdl/MyHello:/〉〈/input〉
    〈output〉〈soap:body encoding Style = " http://
 schemas.xmlsoap.org/soap/
encoding/" use = " encoded" namespace = " http://com.test/wsdl/
   MyHello"/〉〈/output〉〉
```

（下转第174页）

参 考 文 献

1 Brooks F P Jr. The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition, (2nd ed.) Addison-Wesley 1995
2 Fewster M, Graham D. Software Test Automation. Addison - Wesley, 1999
3 PurifyPlus. http://www.rational.com/products/pqc/index.jsp
4 Bush W, Pincus J, Sielaff D. A static analyzer for finding dynamic programming errors. Software - Practice and Experience, 2000, 30 (7): 755～802
5 Engler D, Chelf B, Chou A, Hallem S. Checking system rules using system-specific programmer-written compiler extensions. In: Proc. of the Fourth Symposium on Operating System Design and Implementation, San Diego, Oct. 2000
6 Detlefs D L, Leino K R M, Nelson G, Saxe J B. Extended static checking: [SRC Research Report 159]. Compaq System Research Center, 1998
7 LDRA. http://www.ldra.co.uk
8 Parasoft. http://www.parasoft.com
9 King J C. Symbolic execution and testing. Comm. of the ACM, 1976, 19: 385～394
10 Nelson G, Oppen D. Simplification by cooperating decision procedures. ACM TOPLAS, 1979, 1(2): 245～257
11 Cousot P, Cousot R. Abstract Interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Conf. Record of the Fourth ACM Symposium on Principles of Programming Languages, 1977
12 Splint. http://lclint.cs.virginia.edu
13 JML. http://www.cs.iastate.edu/～leavens/JML.html
14 BANE. http://www.cs.berkeley.edu/Research/Aiken/bane.html
15 Aiken A, Fähndrich M, Foster J S, Su Z. A toolkit for constructing type- and constraint-based program analyses. In: Proc. of the Second Intl. Workshop on Types in Compilation, Kyoto, Japan. March 1998
16 Foster J S, Terauchi T, Aiken A. Flow-sensitive type qualifiers. In: Proc. of the ACM SIGPLAN 2002 Conf. on Programming Languages Design and Implementation, Berlin, Germany. June 2002
17 Das M, Lerner S, Seigle M. Path-sensitive program verification in polynomial time. In: Proc. of the ACM SIGPLAN 2002 Conf. on Programming Languages Design and Implementation, Berlin, Germany, June 2002
18 Flanagan C, Leino K R M. Houdini, an annotation assistant for ESC/Java. FME 2001, Lecture Notes in Computer Science 2021, 2001. 500～517
19 Adams S, Ball T, Das M, et al. Speeding up dataflow analysis using flow-insensitive pointer analysis. In: 9th Static Analysis Symposium, Lecture Notes in Computer Science 2477, Sep. 2002
20 BLAST. http://www-cad.eecs.berkeley.edu/～rupak/blast/
21 Evans D, Guttag J, Horning J, Tan Y M. Lclint: A tool for using specifications to check code. In: Proc. of the ACM SIGSOFT Symposium on the Foundations of Software Engine-ering. Dec. 1994
22 Jim T, Morrisett G, Grossman D, et al. Cyclone: A safe dialect of C. In: USENIX Annual Technical Conf. Monterey, CA, June 2002
23 Vault. http://research.microsoft.com/vault/
24 DeLine R, Fähndrich M. Enforcing high-level protocols in low-level software. In: Proc. of the ACM conference on Programming Language Design and Implementation, June 2001. 59～69

(上接第170页)

```
  〈soap:operation soapAction=""/〉〈/operation〉
 〈soap: binding transport =" http://schemas.xmlsoap.org/soap/
 http"style="rpc"/〉
〈/binding〉
 〈service name="MyHello"〉
  〈port name="HelloIF Port" binding="tns:Hello IF Binding"〉
〈soap:address xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
location="https://localhost:8443/security-jaxrpc/security"/〉
〈/port〉〈/service〉〈/definitions〉
```

如上服务描述语言描述了服务调用的名称、接口、地址以及调用的参数、数据模型等细节信息。客户端应用程序根据这一服务描述，生成客户端的 Stubs 对象程序，和服务器端的 Tie 对象交互，完成远程过程调用。客户和服务器之间消息传递的 XML 文档被系统屏蔽了，这一部分用户不需要自己写交互的 SOAP 消息。系统中可见的是系统自动生成的 SOAP 消息生成和解析的类。如下：

HelloIF_SayHello_RequestStruct__MyHello__SOAPBuilder;

HelloIF_SayHello_RequestStruct__MyHello__SOAPSerializer;

HelloIF_SayHello_ResponseStruct__MyHello__SOAPBuilder;

HelloIF_SayHello_ResponseStruct__MyHello__SOAPSerializer。

结论 RPC 是分布式系统的重要机制。基于 XML 的 RPC 将请求信息封装成 XML 的格式，利用 XML 的灵活性和可扩展性，简单有效地实现了异构平台上不同应用程序的互操作。基于 XML 的 RPC 同时简化了高层程序设计，最大限度地屏蔽了异构平台间的通讯机制的差异。

基于 XML 的 RPC 的通用性会造成其调度效率不能达到最好，原因之一就是消息传递时在 XML 解析、类型转换等方面时间的浪费。但是对于分布式系统，RPC 的主要问题是网络拥塞。并且，随着硬件性能的提高，解析、类型转换等花费的时间会越来越少，在调用过程中所占的资源耗费的比率会越来越低。

总的来说，基于 XML 的 RPC 是很有效的一种远程调用机制。

参 考 文 献

1 Birrell A D, Nelson B J. Implementing Remote Procedure Calls. ACM Transactions on Computer Systems, 1984, 2(1): 39～59
2 http://www.sei.cmu.edu/str/descriptions/rpc_body.html.
3 http://www.w3.org/XML/, W3C Extensible Markup Language (XML) page.
4 http://www.xml.com/pub/a/98/10/guide0.html, A Technical Introduction to XML, by Norman Walsh, October 03, 1998
5 http://www.ietf.org/rfc/rfc1831.txt. "RPC: Remote Procedure Call Protocol SpecificationVersion 2", R. Srinivasan, Sun Microsystems. Aug. 1995
6 http://www.opengroup.org/onlinepubs/009629399/, CDE1.1 Remote Procedure Call. Copyright@1997 The Open Group
7 http://www.opengroup.org/dce/info/papers/tog-dce-pd-1296.htm#intro, DCE Overview
8 http://www.itl.nist.gov/div897/staff/barkley/5277/titlerpc.html, Comparing Remote Procedure Calls. John Barkley, Oct. 1993
9 http://www.ietf.org/rfc/rfc1832.txt. "XDR: External Data Representation Standard", R. Srinivasan, Sun Microsystems. Aug. 1995
10 http://www.ietf.org/rftc/rfc1833.txt. "Binding Protocols for ONC RPC Version2" R. Srinivasan, SunMicrosystems. Aug. 1995
11 http://www.w3.org/TR/SOAP/, Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000
12 "Web Services Conceptual Architecture (WSCA) 1.0" by Heather Kreger, IBM Software Group, May 2001
13 http://www.soapware.org/
14 http://xmlrpc.com/spec. XML-RPC Specification. By Dave Winer. Tue, Jun15, 1999
15 http://weblog.masukomi.org/writings/xml-rpc_vs_soap.htm, Kate Rhodes, XML-RPC vs. SOAP".
16 http://java.sun.com/webservices/, Java Web Services tutorials 1.4