22-25 分布式计算机 远程过程调用 RPC (5)

计算机科学 1994 Vol. 21 №. 3

TP338.8

分布式计算环境中典型RPC的比较研究*)

唐雪飞 汪文勇 刘锦德

(电子科技大学微机研究所 成都 610054)

摘 要 本文介绍了分布式计算环境中两种典型的远程过程调用(RPC)机制:OSF/DCE中选用的RPC,即NCS 2.0,和Sun/ONC中的RPC(包括Netwise的RPCTool)。文中重点在以下七个方面进行了比较性研究:数据表示、联编、可扩展性与一致性、传输层无关性与透明性、可靠性、安全性和多线程技术。最后,介绍了为实现两种RPC间的可移植性和可互操作性所作的努力。

一、RPC简介

在构成分布式计算环境的各种成份中,远程过程调用(RPC)扮演着极其重要的角色,它是一种面向软件开发者的基本服务。利用RPC,分布式应用的不同部分可以互相通信,彼此访问对方机器上的服务原语——而不管对方机器的体系结构如何。因此,RPC是实现异种机网络上互操作性的重要手段。

RPC是分布式应用程序各部分之间进行通信的一种方式。RPC把一个应用程序的执行需要,在客户和服务器进程之间进行分布。客户是应用的主体,当它需要调用远地过程时,给出远地过程名及调用参数通过网络向服务器发送RPC请求。服务器收到请求后,在本地调用适当过程,然后把结果传回用户。

典型的RPC由实际的过程和一个存根(stub)组成。客户过程中的存根是服务器过程的"替身",负责获取调用参数,把它们组装成适于传输的格式后交RPC库程序传送给服务器;服务器过程中的存根则是客户过程的"替身",过程大致相似,服务器主程序循环等待报文输入,决定调用哪个过程,对参数重新进行格式化(使之适合本地使用),然后调用服务器

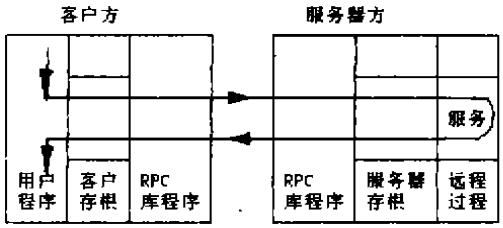

图1 典型RPC的客户/服务器模式

进程。当进程结束时,将结果封装起来并送回客户方。图1给出了以上过程的示意。从概念上讲,RPC模型并不复杂,但在具体实现上要解决的问题很多,如何解决具体问题,全球各大计算机厂商(集团)有着各自的方法。

*)国家教委博士点基金资助项目。

个机壳内能装约100个PE,一个大机壳内则能装500至1000个PE。而且每个PE的价格每三年减半。

根据FGCS计划开发的并行机的实际性能,图5-2显示了第五代机的性能趋势:每个PE的顺序推理性能每3年增至原来的4倍,而每个PE的并行推理性能不如顺序推理性能提高快。图中还显示了用于顺序和并行处理的一块板的性能,以及带有CISC和RISC技术的传统微型处理机的性能。一块板的性能估计约为20M LIPS,比PIM快100倍,这样,大机壳尺寸的并行机能有1G LIPS的速度,不久的将来,并行系统将具有各种知识处理应用所需的处理速度。

6. 最后的评论

FGCS原型系统和R&D成果证明:基于逻辑程序设计的第五代机的基本框架已不仅仅是假设,而且是有效的,可行的。人们坚信在不久的未来将会迎来一个并行处理的新时代。

主要参考文献:Takashi Kurozumi, Overview of the Ten Years of the FGCS Project, Proceedings of the International Conference on Fifth Generation Computer Systems 1992, ICOT.

二、NCS/RPC

美国 OSF 协会(Open Software Foundation)于1989年6月为获得分布式计算环境(DCE)技术而发布了技术征求书(RFT)。到1989年12月便收到了关于DCE的11个方面的74份建议。其中关于RPC的有15份。OSF从中选出了三份决赛选手:来自DEC和HP的网络计算系统(NCS)、来自Netwise的RPC-TOOL以及来自Sun Microsystems的开放网络计算(ONC)的RPC(以下简称ONC/RPC)。经认真筛选,最后于1990年4月14日公布了DCE的RFT结果,其中的RPC选择了NCS(以下简称NCS/RPC),为NCS 2.0。

NCS/RPC向程序员提供了建造客户/服务器应用程序所需要的许多强有力的工具。它们包括两个主要成份:

1. 界面定义语言IDL和IDL编译器。程序员用IDL描述过程之间的界面定义。IDL编译器把这些界面定义转换成客户和服务器方的存根。IDL的句法与大多数程序员熟悉的ANSI C语言相似,使分布式应用程序的开发新手使用起来很容易。IDL还具有其他适应网络环境的语言结构。此外,IDL编译器还负责将过程调用的数据从一种机器格式转变成另一种机器格式。

2. RPC运行时工具。它提供了在网络上将客户的请求转送到服务器,以及发送和接收各种响应的种种机制。

此外,NCS/RPC使用"网络数据表示(NDR)"来表示数据。

三、ONC/RPC

ONC/RPC的早期版本基于Berkeley的套接字(socket),通常在UDP传输层上运行,因此可靠性得不到保证。但随着UNIX System V Release 4(SVR4)的出现,Sun和AT&T提供了一个RPC的增强型版本,称为"传输层无关RPC"(TI RPC)。TI RPC是在传输层界面(TLI)之上实现的,提供了运行时的传输层无关性。TI RPC体系结构如图2所示:

图中的RPC库提供的例程,使应用程序可以请求执行一个远程过程,RPC库还处理所有与传输层的通信;而且,XDR(外部数据表示)也作为库例程提供给用户。在RPC请求报文中,用三个32位字段唯一地标识远程过程。这些字段分别表示程序号、版本号和过程号。为了把RPC的程序号和版本号与服务器上的特定的地址联系起来,Sun提供了portmapper程序。在TI RPC中,有一个新的服务,叫作rpcbind,它是portmapper的与传输层无关的版本,提供一致的寻址机制。

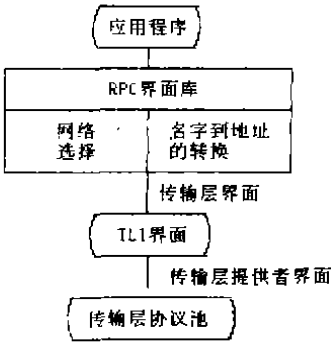

图2 TI RPC体系结构

TI RPC目前提供两个RPC编译程序:旧的rpcgen和新的来自Netwise的RPCTool。用RPCTool生成的代码可以与用rpcgen生成的代码互操作。RPCTool具有很大的灵活性和可扩展性,同时还提供到OSI协议的迁移途径。

四、NCS/RPC和ONC/RPC的比较

NCS/RPC和ONC/RPC的差别主要表现在它们的数据表示方法和联编模型方面。界面语言和下层的网络机制也存在着差别。

Sun早期的RPC语言和"rpcgen"工具比不过NCS的网络界面定义语言(NIDL)。好在与Netwise之间的良好合作关系,Sun能够提供一个更加坚强和通用的界面语言,从而使开发者不仅能够访问Sun工作站,还能更广泛地访问其他厂商的设备资源。而且,在SVR4中使用TLI正在把Sun从互连协议(IP)的限制中解脱出来。

1. 数据表示

目前,两种模式的数据表示存在明显的差异。Sun选择了一种数据表示的标准格式,即XDR。其执行过程如下:数据发送方把数据的本地表示翻译成XDR标准表示,再由数据的接收方把XDR标准表示翻译成其本地的表示。XDR的这种方式非常简单,一旦采用RPC,每种系统只需编写一组转换例程。但是,当数据的发送方和接收方遵守相同的字节顺序时,所进行的字节顺序转换是没必要的。Sun认为:这个缺点不重要,因为花费在转换上的时间无足轻重;为传送而准备一个数据结构所需要的时间大部分不是花费在转换上,而是在全面理解该数据结构成分的过程上;而且在网络应用中,通信的开销更使转换开销微不足道。

NCS 采用 NDR 表示数据。NDR 的策略是只有当输入数据的格式与接收方的本地格式不相同时，才由接收方进行数据转换。

一些专家认为，在拥有许多不同的数据表示的异种机网络中，这种直接的数据转换方法会陷入困境。然而，在一个分布式计算环境中，一个服务器必须处理的不同的数据表示的实际数目相当少。况且计算机工业界正把数据表示标准集中在少数几种选择上。因此，多数情况下，DCE 的这种数据转换方法常常是一种更好的选择。另一个不难理解的事实是：一个工作小组内一般都采用相同类型的计算设备，所以具有相同数据表示的系统之间的 RPC 调用远比异种机网络中的多得多。这就进一步说明了 NCS/RPC 的有效性。

2. 联编

联编指对远程过程调用的目标建立一个引用的操作。NCS 采用一个面向对象的联编模型。换句话说，它鼓励开发者把远程过程调用看成是在对象上的操作，而不是对特定机器或服务器的进程的调用。NCS 设计者的目标是降低精确性，否则开发者必须为一个调用指明目标所在的位置。

NCS 面向对象模型的一个非常重要的方面是，使用一个 128 位固定长的全局唯一标识符(UUID)，作为网络上所有实体的最低级的标识机构。NCS 的设计师们声称 UUID 比在系统的最底层使用简单的串标识符具有更多的优点，包括较小的尺寸、容易嵌入数据结构中、具有位置透明性等等。NCS 定义的与寻址代理者(Location Broker，LB)对话的界面也具有相应的变元，其取值为对象的 UUID。

LB 是一个分布式应用程序，它帮助客户应用程序查找对象所在位置，即返回服务进程的网络地址。

在 NCS 模式中，作为第三方的 LB 介入了客户-服务器通信。客户使用广播联编来查找 LB，然后由 LB 返回服务进程的网络地址，该服务进程将处理客户所发出的 RPC 请求。

在 ONC 的模型中，客户应用程序首先必须通过一个 RPC 创建过程建立与服务器的联系，该创建过程指定服务进程所在主机的名字、被调用过程的名字和版本号、以及为到达该服务器要经过的数个网络的名字。一个专门的 RPC 服务，rpcbind，负责定位服务器程序。

rpcbind 中一个专门的广播服务可以调用 LAN 上特定类型的所有服务器，而不必知道服务器在哪些主机上运行。Sun 的支持者声称 ONC 具有与 NCS 的 LB 相同的服务定位能力，指的就是这个广播服务。然而，甚至 Sun 也指出这个广播机制过于耗费网络资源，而且受到其他因素的限制。比如：广播 RPC 不能在 TCP 这样的面向连接的传输层上工作，而只能在 UDP 这样的数据报传输层上工作，所以整个 RPC 请求必须适合一个广播数据报，不能超出其最大尺寸。

总起来说，NCS/RPC 的联编模型比 ONC/RPC 的联编模型具有较好的位置透明性。前者使用 UUID 标识网络上的实体，用 LB 获取服务器的网络地址；后者使用主机名、程序名和版本号来标识服务进程；如果不知道服务进程的位置，则需用广播机制来确定之，但广播总是存在很多的限制。

3. 其他区别

除了在数据表示和联编方面的主要区别之外，这两种 RPC 在可扩展性与一致性、传输层无关性与透明性、可靠性、安全性及多线程技术方面，都存在着不同程度的差别。下面分别进行比较论述。

(1)可扩展性与一致性。NCS/RPC 的协议被明确地规定并且不允许用户进行修改。在要求互操作性的异种机环境中，这是一个重要的考虑。相反 ONC/RPC(主要指 Netwise RPCTool)允许(并鼓励)用户改变基本的协议来增强能力，比如用户可增加验证(authentication)机制。在这一点上，OSF 和 ONC 之间存在很大的分歧，在 ONC 看来是灵活的东西，却被 OSF 认为是不一致的；而 OSF 标榜具有一致性的东西却被 ONC 指责为顽固不化。这种争议的是非曲直，很难一概而论。

如果用户的目标是建立一个网络服务，该服务必须支持各种名称的环境而开发者最初可能熟悉也可能不熟悉这些环境，则需要一个高度一致性的机制。如果用户的目标是为一个较受限的环境提供 RPC，则可扩展的体系结构使用户能容易地增加某种与众不同的能力。

RPCTool 利用扩展能力提供了以下基础设施：·异步客户操作；·服务器的停机弱化；·异步和同步回调。

此外，开发者还可以使用这种可扩展能力提供以下设施：·身份验证；·审计跟踪；·数据压缩或别的数据处理；·网络管理系统界面。

(2)传输无关性与透明性。传输无关性是指，RPC 可以在任何广域网或局域网上运行的能力。传输无关性之重要性有两方面的原因。首先，每种传输协议可提供不同的能力。第二，开发者可能不知道将使用

什么样的传输层。NCS/RPC 的 stub/runtime(存根/运行时)界面同时支持各种传输协议并允许引入新的传输协议而不影响应用程序自身的编码。ONC/RPC 的早期版本基于 Berkeley 的套接字,并(通常)运行于 UDP 传输层之上;而其增强型 RPC 版本 TI RPC 则是在传输层界面上实现的,并提供运行时的传输无关性。

然而,仅仅具备传输无关性还不足以保障应用软件在任何传输层上都能一致地运行,RPC 还必须提供传输透明性,以确保其特性在所有传输层上都保持不变。NCS/RPC 不管下层使用的是什么协议都提供一致的行为。因此它提供了传输透明性。与传输透明性有关的一个问题是"最多一次"语义。NCS/RPC 通过"最多一次语义"确保了正确的运转,并通过等幂语义确保了高性能。NCS 允许开发者规定各种语义而 RPC 确保这些语义在任何传输系统中都能正确地工作。ONC/RPC 选择不同的传输层时则影响 RPC 的语义,只能凭借可靠的传输协议来确保远程过程最多执行一次。例如,如果使用用户数据报协议 UDP,则对变元的尺寸和数量都有限制,并影响调用的可靠性。

(3)可靠性。在传统的单机系统中,可以把应用程序设计成能够"干干净净"地从系统的失败中恢复,在分布式应用程序中也需要优美的恢复。不过在网络环境中,出错的可能性更大。

NCS/RPC 被设计成面对各种网络故障(包括报文丢失、重复报文、延迟报文、报文失序、服务器崩溃等)仍能可靠地运行,它的可靠性独立于传输层的可靠性。

ONC/RPC 则不保证传输的可靠性,因此应用程序必须知道传输层协议的类型。如果运行于像 TCP 这样的可靠的传输层之上,则 RPC 服务不必再作可靠性工作;如果 RPC 正运行于像 UDP 这样的不可靠的传输层之上,则它必须提供自己的可靠性机制。目前,ONC/RPC 并未提供这方面的可靠性服务。

(4)安全性。在分布式计算环境中,用户身份的鉴别、资源访问的控制等安全性工作属于独立的安全性服务。在 DCE 中,安全性服务通过向安全服务器(受托方)发出一个 RPC 请求,建立一种在两个委托人(客户和服务器)之间使用的许可证。该许可证含有两个委托人交换的加密鉴别信息。通过这种方法,两个受托人便彼此证明了自己享受受托服务的身份。然后,受托服务器给出一个共用的独特密钥,把委托人之间的所有通信加密,以确保数据的保密性和完整性。委托人经过鉴别之后,便可用授权机制来控制对资源的访问。

ONC 没有独立的安全性服务。ONC/RPC 提供三种级别的身份鉴别:无鉴别、UNIX 形式的鉴别和安全的鉴别。UNIX 形式的鉴别使用用户和组标识符凭证,提供与 UNIX 系统相同的用户的身份。安全的 RPC 则使用 DES 密码检验程序,鉴别参数对 RPC 消息本身是不透明的,并且厂商和开发者可以对其进行扩展(这也是 ONC 所推崇的可扩展性的一个方面)。

(5)多线程。在一个程序中具有多线程意味着:在任何一个时刻,该程序都拥有多个执行点,每个执行点对应一个线程。各线程的执行是异步的,即一个线程不必等待另一个线程执行完毕。在分布式计算环境中,可共享的网络服务器,可以为来自多个客户的请求提供服务。多线程的使用允许服务器并行地处理客户们的请求,而不是顺序地处理它们。

DCE 提供一个与 NCS/RPC 集成的、强大的、高度可移植的线程包,其线程服务基于 POSIX 规范草案 1003.4a,从而使一个本来不支持线程的环境也能提供线程服务。

Sun 也打算提供多线程,但不采用 OSF 的方法。

五、结束语

通过以上的讨论,可以看出,NCS/RPC 与 ONC/RPC 之间确实存在许多差别。在较多的方面前者比后者在技术先进性上略胜一筹;但 ONC/RPC 毕竟是最早的商业化实现,其技术成熟和市场占有方面仍有优势,再加上与 Netwise 的合作,ONC/RPC 与 NCS/RPC 的差距正不断缩短。因此,在竞争中,一方要想淘汰另一方是不现实的。所以,可以预料的结局是:"求同存异、相互沟通",并且在这方面已初露曙光*)。

*)注:作为 OSF 一方的主要代表之一的 HP 已开始与 Sun 合作:

- 双方共同开发一个既可用于 ONC 也可用于 NCS 的 RPC 界面。该界面称为类定义语言(CDL)。
- 通过 CDL,开发者可获得 RPC 之间和网络传输层之间的可移植性,使 ONC 和 NCS 应用都与传输层无关。
- 为支持互操作性,双方还同时联合开发了一个公共的数据表示标准。
- HP 和 Sun 正与 ISO 委员会合作,将定义和实现一个正式的 RPC 标准。