计算机科学1998 Vol. 25№. 1

77-80

从 RPC 到面向对象的 DCE 实现

Applications from RPC to Object-Oriented DCE

贾 彬 王启东 周天爵

(复旦大学 CAD 中心 上海 200433)

TP393

摘 要 在目前的高层网络技术应用中,RPC(远程过程调用)的实现可以说仍然很令人头痛。虽然作出了各种各样的调整、改进甚至妥协,其功能和性能还是不能尽如人意。另外,人们对当前流行的一些技术提出了功能扩展的要求。本文从以上两个角度出发,讨论基于 OSF DCE 的面向对象的应用环境的实现思路和实现模型。

关键词 分布计算环境 面向对象 远程过程调用 客户/服务器 多媒体/超媒体信息编码专家组

DCE 计算机网络

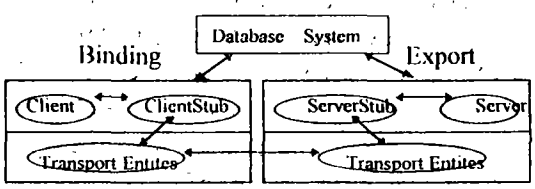
一、DCE 和 RPC 背景及存在的问题

一般说来,RPC 被认为是属于 OSI/RM(OSI 参考模型)的会话层协议,但也有人认为 RPC 的实现应当放在应用层。意见的不同主要是因为 RPC 本身是为快速操作设计的,没有一个多层结构,不能很好地符合 OSI 的参考模型。RPC 的核心主要是实现客户/服务器结构模式,目的是远程资源共享。

RPC 的实现中,客户负责搜集远程进程调用参数传给服务器,然后等待响应。服务器产生一个子进程(从服务器)或一个线程,处理 RPC 请求。从服务器本身再调用具体的本地操作完成 RPC 请求所要求的功能,然后向客户发回执行结果。

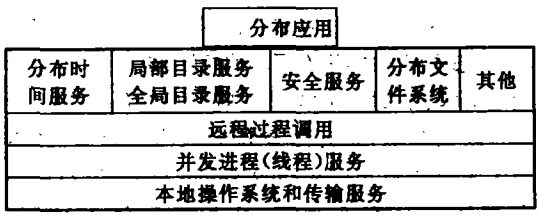
RPC 的客户和服务器之间可显式地建立半双工通信会晤,其处理过程是一个交易过程,只包含一个请求和一个应答,传输报文数目不多。在这种情况下采用多层结构的连接而带来的额外开销对于应用来说往往是很不利的。然而建立在原始数据报功能(尤其是在局网中)上的无连接通信却是一个较好的选择。

RPC 的实现可由下图表示:



RPC 实现中,参数的传递是一个关键的问题,即试图把远程调用在效果上变成本地调用。但是,真正的透明是很难达到的。一方面,变量参数、指针、函数、函数参数难以正确实现;另一方面,客户和服务器面对崩溃时达到精确一次(exactly once)语义是几乎不可能的。许多 RPC 系统解决整个问题的手段是禁止使用引用参数、指针以及各种过程和函数参数。虽然降低透明度,但使得实现简单。更有人认为透明性根本就不应当尝试。

OSF DCE (Open Software Foundation/Distributed Computing Envirment),即分布计算环境是开放式分布计算(ODP)的业界标准。基于客户/服务器模式,分布计算环境提供了多种建立分布应用的服务。它兼容于多数占主导地位的操作系统、平台和众多的运输协议。在标准的网络程序接口和自动 RPC 存根(stub)生成的基础上,DCE 实现多种异类系统的互操作以及应用的灵活性。DCE RPC 支持多种通讯语义,包括基于线程的异步 RPC、幂等操作的至少一次(at least once)语义和基于逻辑管道的成批数据传送等等。DCE 的结构如下:



DCE 结构特征为:

·整个结构的核心是基于 RPC 的客户/服务器结构。

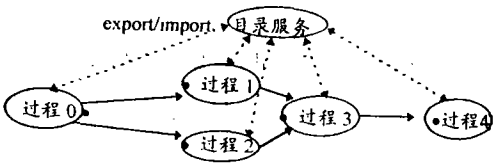
·通过进程(线程)扩展,DCE 提供操作系统和传输层接口,进行并行进程和并发 RPC 的处理。

·由两个互相协作的目录服务配合实现服务器(export server)和客户(import client)的装配(binding)工作。

·安全服务实现通讯双方的数据保密,权限确认等安全方面的工作。

·DCE 还提供时间同步、分布文件管理、无盘工作站的支持等其他服务。

基于 DCE 提供的基础服务(RPC,进程服务,目录服务等),可以建立各种各样的客户/服务器应用。例如:



其中,虚线代表 export/import 操作,实线代表 RPC,黑点代表进程服务发生的位置。过程0中,进程服务实现并行 RPC。过程1和过程2中,DCE 创建线程实现从服务器。过程3中,DCE 创建两个线程实现并发从服务器。在某个特定的过程(或服务器)之中,必须基于 DCE 条件变量进行进程或线程间的同步以协调并行或并发执行顺序。例如,过程3必须在过程1和过程2都结束后进行。另外,除纵向的顺序外,还应在横向上考虑错误和异常的处理。

但是,DCE 中还存在很多问题:

·参数语义。DCE RPC 的参数传递一般采用值参传递的方式。即使在程序一级实现了引用参数的传递,也只不过是将引用参数所指的数据内容全部拷贝到服务器的地址空间,这无论是对于数据的并发操作或是对于大批数据的引用都很不理想。这是 DCE RPC 实现中的最大问题。

·分布的单元。DCE RPC 中的基础分布单元是客户和服务器。更多的数据元素例如文档等不能进行方便的分布处理。

·分布实体。无论是客户、服务器,还是其他数据实体,都是固定的,没有可动的特性。整个 DCE 只是固定实体之间通过 RPC 进行联系的系统。

以上缺点,尤其是参数语义的不完全,导致 DCE 高度的应用相关(application-intensive),体现在程序设计和实现上,就是代码相关(code-intensive)。因此对于广泛的应用而言,就显得不够灵活,应用的扩展和 DCE 功能的扩展都受到限制。

随着通讯和计算机技术的飞速发展,网络应用已不再局限于文件传输和电子邮件,多媒体、超媒体之类的应用要求改进原有协议。虽然对于 OSI/RM 中高层的设计存在着众多的争议,但随着应用的深入,象 TCP/IP 之类的协议中高层结构简单的优点也带来了一些问题。目前流行的许多应用协议如 HTTP(超文本传输协议)虽然深受欢迎,但是功能比较简单,随着应用的愈加复杂和深入,功能扩展和改进的要求不可避免。

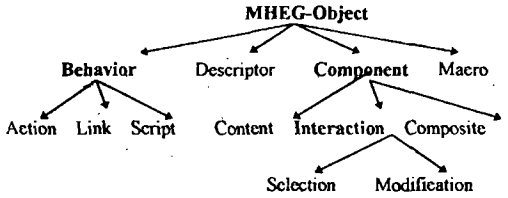
新的应用发展主要集中在几个方面:从单一媒体向多媒体/超媒体发展,从单用户应用向用户组应用发展,从集中向分布发展。考察多种新的应用,如:多点电视电话会议、视频点播 Video on Demand、MBone、MONET 等系统,可以发现:虽然应用的媒体基础和跨越范围大大扩展,客户/服务器结构却依然是应用实现的核心。例如在 MONET 系统中包含一个会议服务器(Conference Server)提供会议功能、一个应用共享服务器(Application Sharing Server)提供共享工作区间、一个目录服务器(Directory Server)实现应用参加者登记和目录功能、一个多媒体服务器(Multimedia Server)实现媒体的同步。事实上,会议服务、共享应用和目录服务正是分布协作计算支持模型(Support Model)的基本要素。当然客户/服务器结构必须有新的实现以满足功能扩展的要求。

二、新应用实现问题分析和 DCE RPC 的改进

1. 问题分析

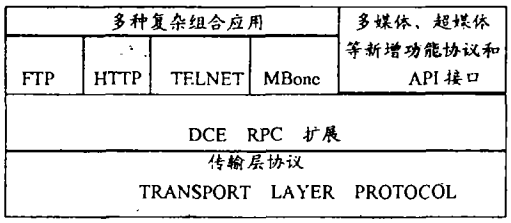
DCE 实现改进的关键是采用面向对象的技术。对于参数语义问题:采用面向对象技术,设计通用的客户/服务器类并根据不同的应用派生特定的客户和服务器对象。参数空间封装在对象中。远程引用采用对象代理来实现。这种实现一方面解决参数的语义问题,一方面可以使各应用单元具有独立和可动的特性。

对于分布计算和多媒体应用,除了客户/服务器结构本身,应考虑数据本身,包括二进制数据、文本、图形、声音、图象也都应实现对象化。这些应用单元的对象化描述可以以 ISO/IEC JTC1/SC-29WG12 的 MHEG(Multimedia and Hypermedia Information Coding Expert Group)标准方案为模板。MHEG 中定义了 Content 类、Behavior 类、User Interaction 类和 Container 类等独立单元,分别描述要表现的信息本身、信息表现动作、用户交互抽象和复合(复杂)信息对象的抽象。MHEG 对象的类层次结构如下图所示,图中黑体代表抽象虚基类。



虽然 MHEG 标准采用了面向对象技术，但上述各类中的方法都没有作定义，并且不支持方法的继承。在 DCE 功能扩展中进一步实现对象化。

对于功能的扩展，可以从两个方向入手：或是修改原有协议，使之包含新的功能，或是从应用协议的结构出发，增加新层。很明显第一种方法意味着每当新的应用要求出现时，都必须重新制订协议和编制应用程序，代价太大。第二种方法的问题是要考虑增加新层带来的复杂化和效率降低。但是合理的结构设计应能克服这一问题。基于这种考虑，可以采用 DCE 作为增加新层的模板，为应用提供灵活的基础。考虑如下模型：



用户完全感觉不到 DCE 扩展层的增加，实际上，新增功能通过新的实现或原有应用协议经 DCE 扩展后的实现两条途径得到。

2. DCE 改进思路

·考虑到多种新旧应用的集成和保持 RPC 原有的简单、灵活的要求，必须采取复杂性降低的策略，将复杂应用分解为 RPC 易于实现的应用单元。

·考虑到功能方面的需要，应当给出各应用单元交互的接口，确保实现各应用单元的协作，以完成复杂的应用实现，这一点是复杂性降低的前提。

·DCE RPC 应支持面向对象的实现，以解决参数语义和应用扩展的问题。面向对象的实现对复杂性降低和功能协作是至关重要的。除了相应地提供 C++ 语言的集成外，DCE 的编译器也应当特别考虑，以期自动实现面向对象的 RPC 过程和应用单元。

三、分布式多媒体 DCE 的面向对象实现

面向对象的 DCE 基础核心是 RPC 实现，除设计编译器以将抽象的 RPC 调用映射为具体的传输调用之外，RPC 的客户/服务器模型思路基本同原有模型一致。

整个分布式面向对象 DCE 实现的主要特征为：

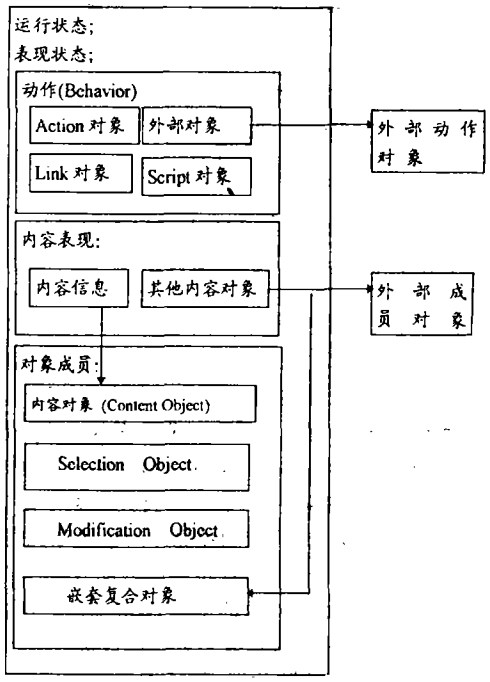
·各实体，包括客户、服务器、数据甚至 MHEG 对象等应用元素的对象化。

·对象具有可动特性，通过分布的对象之间的通讯和对象本身的运动，实现复杂性下降和功能协作。

1. 多媒体应用单元的对象实现

复合多媒体应用单元类可如下描述

Class Multimedia_Application_Element



2. 标准 DCE 环境中服务器的对象实现

服务器类表示如下：

```
class Server:public_system
    {private: rpc_if_handle_t IfHandle;/* 接口连接句柄*/
    public:Server(rpc_if_handle_t Handle);
      unsigned32 Export(unsigned_char_t* EntryName
        = NULL, rpc_binding_vector_t* vector =
        NULL,UUIDVector*ObjectVector=NULL);
        /*执行 export 过程，登记服务器对象*/
    /*初始化例程*/
    void Init(unsigned_char_t *EntryName=NULL,
      UUIDVector*ObjectVector=NULL,
      UUID *MgrType=NULL,
      rpc_mgr_epv_t Manager=NULL,
      unsigned_char_t *ProtSeqs=NULL,
      unsigned32 MaxCurrentCalls=rpc_c_listen_max_
        calls_default,
      unsigned_char_t*Annotation=NULL);
    /*其他初始化例程*/
    void ShutDown();
    /*其他控制例程*/};
```

本文中程序采用 Microsoft 的 RPC 工具--MIDL（Microsoft Interface Definition Language）编写，MIDL 与 OSF DCE 规范中定义的 RPC 标准相一

致，因而使用 MIDL 写的应用程序可调用其他使用 DCE 标准的系统的远程过程。

3. 分布式面向对象 DCE 中各对象的实现

各对象的公共基类如下：

```
Class Object_Reference{
    private:
    uuid object_id;
      /*对象标识*/
    char   *object name;
      /*对象名*/
    Node   *suspected_loc;
      /*对象代理的位置*/
    Node   *creating_node;        /*对象创建位置*/
    pthread_mutex_t mutex;/*同步信号*/
public:
    Object_Reference(char*);
    Object_Reference(RPC_Obj_Ref*);
    Object_Reference();
    void lock();/*同步信号锁定*/
    void unlock();/*同步信号开锁*/
    void update(Location* loc);
      /*更新位置信息*/
    int migrate(Object-Reference*);/*相对迁移*/
    virtual int migrate(Node*)/*绝对迁移*/
    Location* locate();/*确定位置信息*/
    };
```

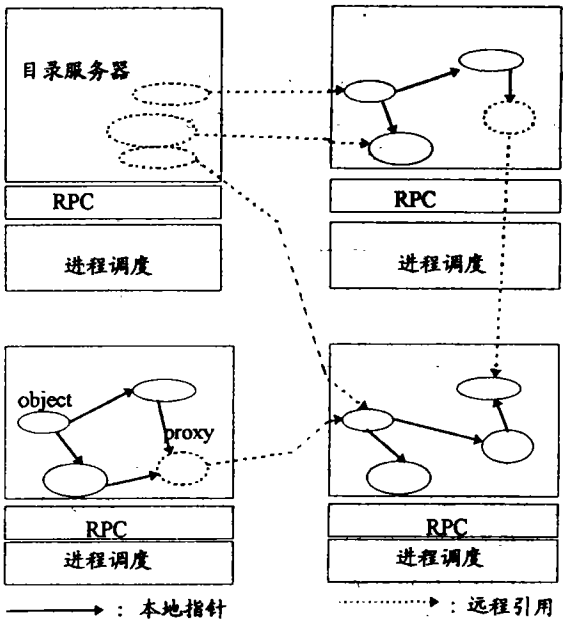
上述实现中，对象包含一个 DCE UUID（全局标识）作为对象标识，另外还有一个可选的对象名，对象引用请求和迁移的同步信号也包含在 Object_Reference 中。对象创建位置和对象代理的位置分别存储于 creating_loc 和 suspected_loc。第一个构造函数用来建立新对象。当对象引用传递到某个给定的结点时，调用第二个构造函数建立对象代理。数据结构 RPC_Obj_Ref 包含对象位置的内部 RPC 地址信息。Locate 方法确定对象的位置，Unpdate 方法在 RPC 请求返回时根据返回结果更新代理链。相当于迁移方法先确定对象的位置，然后调用绝对迁移方法，绝对迁移方法与具体应用相关，所以是虚函数。

4. 对象的分布、可动性及应用功能的分散和协作

在上述的实现中，应用由分布在不同结点的对象组成。每个结点都维护一个散列表，以实现全局对象标识到物理内存的映射。应用实体，无论是客户、服务器、纯数据还是多媒体应用单元都直接以对象来表示。对象间的联系建立在 RPC 的基础上。为处理远程引用参数的传递，提出了代理（proxy）的概念。代理对象含有位置信息，并且能够透明地发出基于 DCE RPC 的请求。DCE 中包含一个或多个目录服务器，目录服务器中存储已登记对象的代理，通过这样的方法，同级对象只要将它的名字传给目录服务器，就可以获得远程对象的代理。

对象是可以运动的，就是说，对象的数据在运行过程中可以在结点之间往来传递。对象的运行有两种形式：迁移和作为参数的传递。后者是将对象作为请求参数，这样会在被调结点上创建代理。这种方式主要针对多媒体应用单元等数据对象，可以解决引用的参数语义问题。迁移主要针对多个功能单元对象协作实现功能要求。复杂功能根据某种原则分解成功能单元，分布于不同的结点上，通过功能单元对象的迁移来实现功能协作，完成复杂的功能。迁移以相关对象在互斥的基础上发出请求来实现同步。迁移时，源结点上建立代理对象，并且随着对象的继续前进产生代理链。另外，多媒体或超媒体应用单元包含丰富的表现方式和巨大的信息量，面向对象使这些特点与灵活、方便的处理很好地结合起来。对象迁移的方法使处理多媒体应用对象避免了大量的、不必要的数据传送，节省了带宽和处理时间。

面向对象的分布 DCE 整体结构如下图：



总结 以上讨论了对传统 RPC 的改进，提出了一种面向对象的分布 DCE 的实现。由讨论可以看出，无论是从结构和实现简单的角度，还是从应用功能的扩展的角度，采用面向对象的模型来加以实现都是很自然的，它既解决了传统 DCE RPC 中参数语义实现的困难，又为多媒体、超媒体等新的应用提供了基础，同时这种实现也是高效的。

参考文献

[1] Andrew. S. Tanenbaum,《Computer Networks》3rd Edition

[2] Ralf. Steinmetz,《Multimedia: Computer, Communication and Applications》

[3] W. Stallings ,《Data and Computer Communication》

[4] C. A. Sunshine,《Computer Network Architechtures and Protocols》