文章编号:1004 - 5694(2001)02 - 0085 - 04

用 CORBA 技术实现分布式对象应用*

高雄英,葛君伟
(重庆邮电学院,重庆 400065)

摘　要:CORBA 技术能够解决异构问题,而且扩展了客户/服务器模式。分析了分布对象计算的特点、CORBA 的原理及如何利用 CORBA 技术实现分布对象应用。

关键词:CORBA;ORB;DCOM;IDL;分布式对象计算
中图分类号:TP393.09　　文献标识码:A

Realizing the Distributed Objects Application with CORBA Technique

GAO Xiong-ying　GE Jun-wei
(*Institute of Computer Science & Technology, Chongqing University of Posts and Telecommunications, Chongqing* 400065, *China*)

Abstract: CORBA techenique can not only solve the heterogeneous problems, but also expand the mode of client/server. This paper analyses the characteristics of distribued objects computing, the principle of CORBA and the realization of distributed objects application with CORBA technique.

Key words: CORBA; ORB; DCOM; IDL; distributed objects computing

随着计算机网络技术和应用的发展,分布对象技术也逐步成为热点。CORBA 作为其中的一种重要技术,也得到了快速发展。目前,国际范围内已经有大量企业、政府和银行等运用了 CORBA 技术,取得了显著效益。

1　分布对象计算

分布式计算可使两个或多个软件互相共享信息,可以拥有稀有资源共享,平衡机器负载、稳定(如:一个机器崩溃,整个系统还可以运转),可扩展等优点。目前,大部分分布式计算是基于客户/服务器模型的。一般说,对象模型是考虑问题及其解决方案的概念性框架,对象模型的基础是对象这一基本概念,对象是有特定的行为和属性的实体。基于对象的分布式计算所面临的挑战是建立一个系统,让软件对象间透明地进行通信,彼此使用对方的服务,而不管这些对象是处于同一编址空间,还是不同的编址空间,或是根本不同的机器上。如果没有一种方法让对象在网络间可相互调用(借助于对象通道),应用就会被限定在某一桌面上。现有的分布式对象技术主要有:对象管理集团(OMG)的 CORBA 和微软的分布式组件对象模型(DCOM)。DCOM 包含在 Windows NT 4.0 中,可作为一个附件下载到 Windows 98。但从服务的角度来看,它只能在 NT 下使用,将 Unix 对象放置到链路上则很困难。相反,CORBA 技术通过在客户端和服务器端加入一个代理,从而屏蔽了不同计算机和不同软件系统的差异,

* 收稿日期:2000 - 10 - 17
作者简介:高雄英(1971-),四川眉山人,研究生,研究方向为网络通信及管理;葛君伟(1961-),男,浙江东阳人,博士,副教授,计算机科学与技术学院副院长,主要研究方向为网络通信及管理、地理信息系统。

实现了异构环境下的客户端和服务器端的通信。

2　CORBA 原理及特点

CORBA 的全称是“公共对象请求体系结构”，ORB 是其核心部分。单个 ORB 的结构如图 1 所示。

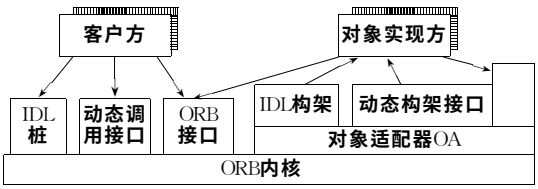

图1　单个ORB体系结构

ORB 核心提供了客户－对象间实现透明通信的方法，可屏蔽对象实现位置、方法、状态、通信机制等细节及不同实现间可能存在的差异。对象适配器位于 ORB 核心和对象实现之间，负责服务对象的注册，对象引用的创建、解释，对象实现的服务进程的激活、去激活，对象实现的激活、去激活及客户请求的分发。IDL 桩为客户提供了静态调用方式；IDL 构架为客户提供了静态实现方式。IDL 编译器编译描述服务对象接口的 IDL 文件，生成对应于具体编程语言的 IDL 桩和 IDL 构架程序。IDL 桩负责把用户的请求进行编码，发送到对象实现端，并对接收到的处理结果进行解释，把结果或异常信息返回给用户。IDL 构架对用户请求进行解码，定位所请求的对象方法，执行该方法，并把执行的结果或异常信息发送回客户。动态调用接口 DII 和动态构架接口 DSI 提供了动态调用方式和动态实现方式。

CORBA 作为一种技术得到快速发展是因具有如下特点：

· CORBA 定义了一种面向对象的软件构件构造方法，使不同的应用可以共享由此构造出来的软件构件；

· 每个对象都将其内部操作细节封装起来，同时又向外界提供了精确定义的接口，从而降低了应用系统的复杂性，也降低了软件开发费用；

· CORBA 的平台无关性实现了对象的跨平台引用，开发人员可以在更大的范围内选择最实用的对象加入到自己的应用系统之中；

· CORBA 的语言无关性使开发人员可以在更大的范围内相互利用别人的编程技能和成果，是实现软件复用的实用化工具。

3　CORBA，WWW，Java 的结合

3.1　Web 对象计算体系

在目前逐渐兴起的 Web 体系结构的动态应用阶段中，分布对象技术成为了 Web 应用开发的基石。浏览器和服务器不仅将超媒体信息提供给用户，更重要的是能够使用户通过 Web 浏览器这一通用终端，自由地访问分布在网络上有对象组成的各种应用，从而真正实现网络计算。CORBA 和 Java 的结合又使应用程序具有跨平台的功能。

如图 2 所示，Web 浏览器作为客户层，提供图形用户界面，负责与用户进行交互。它通过 HTTP 协议从 Web 服务器下载超文本页面，并把交互信息送回 Web 服务器，通过 Web 服务器中的客户方程序调用位于不同服务器上的对象，在数据库服务器的协助下完成客户端的请求。应用服务器上的对象也可以调用位于其它应用服务器上的对象。

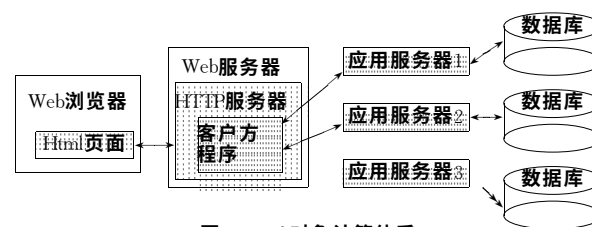

图2　Web对象计算体系

在上述体系结构中可把 Web 服务器和应用服务器合在一起，也可分开，同样数据库服务器也可和应用服务器合在一起，因此大大增加了系统的灵活性，也可实现在不同的地方开发，开发完之后，只需在客户方用 IDL 语言进行封装即可。要实现同样的功能只需在一个地方编写一次，即可在任何地方进行调用，而且可实现对象与对象之间的互访。

上述结构以分布对象技术为基础构架，把 Web 服务器与应用服务器分离，应用服务器与数据库分离，降低了 Web 服务器的负载，避免了 Web 服务器的性能缺陷对整体性能的影响。并具有连接缓冲、负载均衡、安全管理等功能，从而大大提高了 Web 应用整体的灵活性、可伸缩性和可扩展性。

3.2　实现 WWW，Java，CORBA 的结合

目前，遵从 CORBA 规范的产品主要有 IONA

公司的 Orbix，Inprise 分公司的 VisiBroker 等。这些产品提供一个工具，可以把用 IDL 语言描述的接口编译后转化成 Java 语言，从而实现 CORBA 与 Java 的结合，也可以用 Applet 编写客户端代码，把 Applet 小程序嵌到一个 HTML 文件中，从而实现 WWW，Java，CORBA 三者的结合。

本文的实例采用了 VisiBroker 产品，采用 IDL－Java 工具来实现 IDL 与 Java 的结合。在客户端采用 Jsp 技术，因 Jsp 与 Asp 比较类似，只需 ASP 使用的构件是 ActiveX，而 Jsp 使用的是 JavaBean 构件，用户从 Web 服务器下载 Jsp 文件，然后输入信息，执行操作；浏览器把用户输入的信息送给 Web 服务器中对应的 JavaBean，然后通过 JavaBean 调用其他程序，并向应用服务器发出请求，应用服务器接到请求后，进行处理，处理完毕后，把信息返回给客户端（Web 服务器），把更新后的页面（Jsp 文件）返回给用户。从而实现了 WWW，Java，CORBA 技术三者的结合。

4 VisiBroker 的一个分布式应用

4.1 采用标准的接口定义语言（IDL）定义接口

如定义一个银行模块，里面有 2 个接口，一个是查询余额，一个是当用户有疑难问题可请求帮助。Account 接口和 AnswerQuestion 分别在不同的地方实现，并且和不同的数据库相连。

```
module Bank{
interface Account{
int balance(in string name, in string password);
};
interface AnswerQuestion{
void query(in string problem, out string answer);
};
};
```

4.2 写好的 IDL 文件编译生成 Java 代码

生成的文件有 Account.java，AccountHelper.java，AccountHolder.java，AccountPOA.java，_AccountStub.java；AnswerQuestion 接口编译后生成的文件与 Account 接口类似，它在后面的实现中与 Account 接口的实现类似，故略。

如果需要支持 IDL 语言中的 out inout in 等类型参数传递模式，则必需使用支持类，对于用户自定义的类型，其支持类的名字是在映射得到的类型名字后，加上 Holder 而构成的，如 AccountHoder.java。所有用户自定义的 IDL 类型都有一个“助手”类，其名字是生成的类型名字后加 Helper 后缀，如 AccountHelper.java。它提供了几种静态方法，对该类型的对象进行操作，包括该类型的插入、获得库标识操作、获得类型编码操作以及把该类型写入流或从流中读出的操作。

AccountStub.java 是对应于 Account 接口的 IDL 桩，负责把用户的请求进行编码，发送到对象实现端。AccountPoa 对应于 IDL 构架，负责对用户的请求进行解码，定位所请求的对象的方法，并把执行结果或异常信息编码后发送回客户。在把客户端与服务器端分离时，必须把后缀为 Poa 的文件随服务器端的实现方法一起拷贝到服务器。Account.java 是一个抽象接口，并在其它文件中实现。

4.3 在服务器方实现 Account 接口及其主程序

```
public int balance(String name, String password) {
    String url="jdbc:odbc:database1";
    try{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection con=DriverManager.getConnection(url);
    Statement stmt=con.createStatement();
     ResultSet rs = stmt.execQuery("SELECT * FROM table1");
    while(rs.next()){
      String s1=rs.getString("姓名");
    String s2=rs.getString("口令");
      if(s1.compareTo(name)==0){
        if(s2.compareTo(password)==0)
        {
          int s3= rs.getInt("余额");
          return s3;
        }
      }
    }
    }catch(Exception e){
    System.out.println("EXCEPTION:"+e.getMessage());
}
    return -1;
}
```

上面是实现接口 Account 的代码，其中，database1 是存放 table1 的数据库，人名和口令放在

table1 中。如果，姓名和口令正确则查询并返回余额，否则，返回－1。该代码存在于文件 AccountImpl. java 中，另外，服务器方主程序是：BankServerApp. java.

4.4　客户方的程序

客户方程序有 FormBank. jsp，JspBeanAccount. java AccountClientImpl. java 及 JspBeanAnswerQuestion. java，AnswerQuestionClientImpl. java 文件。在 AnswerQuestionClientImpl. java 文件中，用下面的代码可以实现与远程主机的绑定。我们假定远程主机的 IP 地址为 202. 202. 42. 100.

```
answerQuestion = Bank. AnswerQuestionHelper. bind(orb, "/"
+ name + "_ poa", name. getBytes ( )," 202. 202. 42. 100", _
options);
```

其它的代码限于篇幅有限，故略。

调试成功后，启动 OSAGENT，再运行在 2 台不同主机上的服务器端程序，然后，运行客户端程序，就可以下载 FormBank. jsp 文件，并进行查询余额，或进行疑难解答。

我们介绍了 CORBA 的原理，分析了 CORBA 的特点，并提出了一种构建分布式系统的体系结构，并用 visibroker 实现了它。这种体系可以对分布在不同地方的对象进行访问，而且分布在不同地方的对象相互之间也可实现互访。

参 考 文 献

[1] 韦乐平. CORBA 系统结构、原理与规范[M]. 北京：电子工业出版社，2000.

[2] 汪芸. CORBA 技术及其原理[M]. 南京：东南大学出版社，1999.

[3] 李师贤. CORBA 教程[M]. 北京：清华大学出版社，1999.

[4] 陆丽娜. 分布式操作系统[M]. 北京：电子工业出版社，1999.

[5] Common Object Request Broker Architecture Specification[S]. 1998.

[6] INPRISE. Visibroker for Java 3. 3[S]. 1999.