计算机科学 1999Vol. 26№. 7

基于库函数的ORB运行时系统的研究

The Research on the Library-baaed ORB Runtime System

敖青云　尤晋元

（上海交通大学计算机科学与工程系　上海 200030）

Abstract　There are three basic choices for the architecture of an ORB implementation: server-based ORB, system-based ORB and library-based ORB. This paper will mainly describe the scheme of the library-based ORB implementation.

Keywords　ORB, Stub, Skeleton, Java

一、引言

OMG 是对象管理组织(Object Management Group)的缩写，该组织于 1989 年成立，目的是创建具有互操作性、可移植的分布对象计算标准。OMG 对象管理体系结构(OMA)在一个较高的抽象层次上定义了用于分布对象计算的各种机制。OMA 的核心是对象请求中介者(ORB)，它为分布环境中的对象屏蔽了网络、操作系统、和实现语言的异构性，提供对象寻址、激活和通信的透明性，使得分布对象之间通信就如在同一地址空间一样。CORBA(Common Object Request Broker Architecture)规范则是对 ORB 必须提供的服务接口的具体描述。

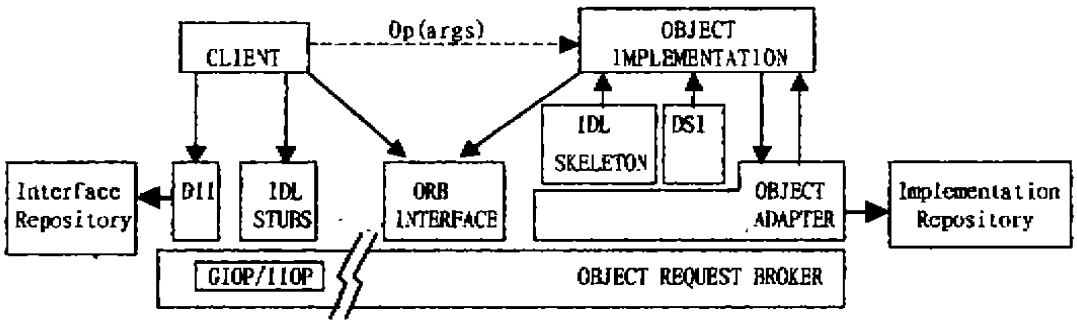

图 1　CORBA ORB 的体系结构

图 1 显示了 ORB 体系结构中的主要组成部分。图中向下的箭头表示常规的对象调用接口，向上的箭头表示上调(upcall)，即系统软件向用户应用程序的调用。ORB 的实现可以分为三种类型：

①基于服务器的 ORB。它把 ORB 作为一个独立运行的程序，它同时充当客户和服务器的服务器。ORB 与客户以及服务器之间的通信可以采用任何进程间通信机制来实现，这种方法的优点在于 ORB 的管理比较集中。

②基于系统的 ORB。ORB 还可以作为操作系统的一部分，嵌入到操作系统的底层。在设计和实现这种 ORB 时，能够更直接地利用操作系统的一些特征，因而具有更高的安全性和更好的性能。

③基于库函数的 ORB。在这种实现中，ORB 不是作为一个独立运行的构件，而是驻留在库函数中。客户程序和服务器程序可以嵌入这些库函数，从而使用 ORB 的功能。

在上述三种方法中，基于库函数的 ORB 实现简单，容易理解。本文主要讨论使用 Java 语言实现基于库函数的 ORB 的一种方案。

二、消息

在分布式环境下，客户程序与服务器程序之间的通信是通过消息的发送与接收来实现的。服务器接收到客户的请求，加以分析，进行相应的处理后向客户送回响应。消息中包含参数等数据，在发送和接收消息时，需要指定数据的类型。除 void 外，Java 语言还提供了 8 种不同的原类型，实现时，可以用一个字符串来区分这些类型：无类型(void)表示为"V"；字节型(byte)表示为"B"；字符型(char)表示为"C"；布尔型(boolean)表示为"Z"；整数型(int)表示为"I"；短整数型(short)表示为"S"；长整数型(long)表示为"J"；浮点型(float)表示为"F"；双精度型(double)表示为"D"。

对于数组类型，可以用"["和数组元素的类型来表示，例如 int[]的类型为"[I"。Java 中的类(class)的类型用"L"＋类名＋";"表示，因此，"Ljava.lang.String;"代表 java.lang.String 类。

在客户程序向服务器发送的请求消息中，包含有三个部分：消息头、请求头以及数据。消息头中有 GIOP 标志、版本号以及消息长度等。请求头给出了这个请求的信息，如请求的上下文、请求编号、是否需要响应和调用的方法等。数据部分指定调用请求的具体参数，它可以根据 signature 来得到。Signature 的格式为：

(参数 1 的类型参数 2 的类型…参数 n 的类型)结果类型

这里需要说明的是，在 CORBA 规范中定义了三种不同的参数 in，inout 和 out。其中 out 参数只用于获得结果，因此不必包含在客户向服务器发送的请求消息的数据部分中。

服务器向客户程序发送的响应消息也有三个部分：消息头、响应头和数据。消息头同请求消息的消息头一样，由 GIOP 标志、版本号以及消息长度组成。响应头部分包含响应的上下文、所响应的请求编号和响应状态等。而数据由调用返回的结果、inout 参数、out 参数或者执行调用时产生的异常组成。

三、桩程序和轮廓程序

桩程序和轮廓程序均提供对象服务的静态界面，分别相当于客户程序和服务器与 ORB 之间的"胶水"。它们由 IDL 编译器自动生成，映射为目标语言代码。桩程序和轮廓程序的自动生成减少了两者之间不一致的可能性，并增加了编译优化的机会。

桩程序定义了客户程序怎样去调用服务器方相应的服务。从客户的角度来看，桩程序相当于本地调用，是远地对象在本地的代理，它负责将调用的过程和参数通过网络链接传递给远程对象，并接收其返回结果或异常，从而屏蔽了底层传输协议及数据表示等网络细节。

轮廓程序则定义了怎样将客户请求分派给相应的服务。在服务器方轮廓程序根据接收到的客户请求，调用对象实现提供的服务，再将服务的结果返回给相应的客户。同样轮廓程序也向对象实现屏蔽了网络细节。轮廓程序的主要执行过程是：①获取调用参数；②调用对象实现的相应方法；③返回调用结果或异常。

四、实现时的主要对象

1. **Stub 和 Skeleton**　Stub 是服务在客户端的代理，由 IDL 编译器自动生成，其中包含原始接口中列出的所有方法，这些方法与接口中的相应方法有相同的 signature。Skeleton 是服务器端的代理，本质上是多线程的，主要方法是 run()。每个 Skeleton 负责处理某个客户的请求。运行时，不断从输入流中读取请求、检查该请求、调用对象实现的相应方法进行处理，并送回结果(将结果、in/out 参数、out 参数或异常写入到输出流中)。

2. **NetServer**　在 Stub 和 Skeleton 之间建立面向连接的 TCP 链接时，可以使用标准的 Java 套接口库函数。NetServer 类用于管理 Skeleton 方的链接。同时，为了能够并发处理多个客户程序的链接请求，NetServer 类继承 java.lang.Thread 类。NetServer 有两种不同的实例。每个 Skeleton 都有 NetServer 的主实例，运行时，在某端口上监听客户程序的链接请求。一旦建立链接，它创建自己的 NetServer 实例在不同的线程中运行，这个线程负责在一个链接中接收客户程序的请求，在 Skeleton 中分配这些请求。而主 NetServer 继续监听其他的链接请求。

3. **External**　ORB 运行时系统定义了 External 类来处理 Skeleton 和 Stub 之间消息的发送与接收。在 Skeleton 和 Stub 之间建立面向连接的 TCP 链接时，它们各自实例化一个 External 对象。External 实例中包含有该套接口的输入与输出流，并定义了四个方法：SendRequest，ReceiveRequest，SendReply，以及 ReceiveReply。Skeleton 调用 ReceiveRequest 和 SendReply 分别从输入流和输出流中接收请求以及发送回答，而 Stub 则调用 SendRequest 方法向输

出流中发送请求，调用 ReceiveReply 方法从输入流中读取调用结果。

4. Requester　Requester 类也继承 java.lang.Thread 类。每个已建立了连接的 Stub 都有一个 Requester 实例的线程。Requester 线程在运行时，不断从输入流中读取数据，根据请求编号查找上述表。若找到，设置相应位，唤醒因等待该数据而被挂起的 Stub，以便它取得结果。

五、调用的执行过程

如图 2，客户程序的执行过程为：

· Stub 与服务器建立 Socket 连接。

· Stub 根据 Socket 连接创建 External 实例以处理它与服务器的消息传送。

· Stub 创建 Requester 线程以接收调用结果。

· Stub 通过 External 实例向服务器发送调用请求并阻塞，等待调用结果。

· Requester 线程从 External 实例中获得调用结果。

· Requester 线程拷贝调用结果，并设置"准备好"标志。

· Requester 线程唤醒等待的 Stub。

· Stub 获得调用结果。

服务器的执行过程为：

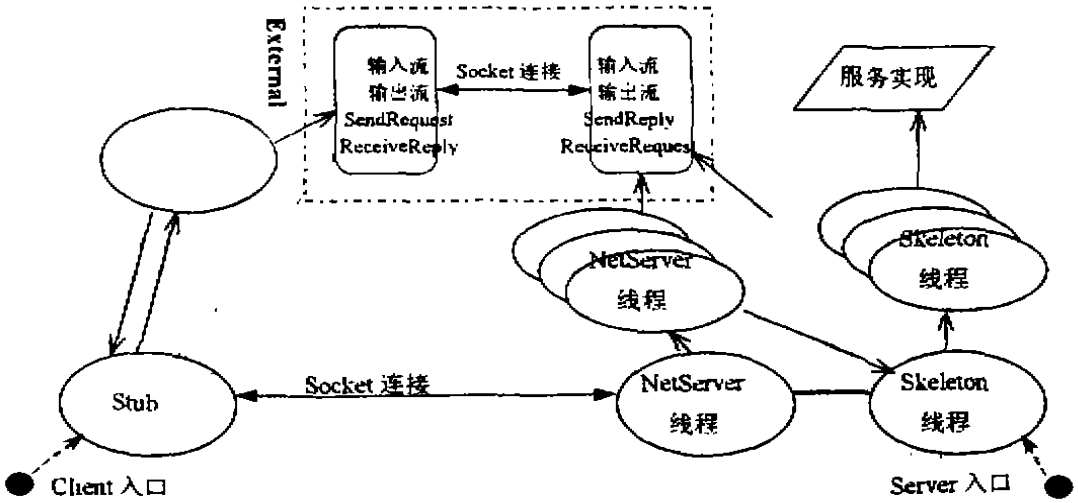

图 2　调用执行过程

· 在服务器启动后，有一个主 Skeleton 线程和一个主 NetServer 同时运行，NetServer 等待并接收来自客户端的连接请求。

· 在接收到客户端的连接请求后，主 NetServer 创建另一个 NetServer 线程处理客户端的调用请求。

· NetServer 线程根据 Socket 连接创建 External 实例。

· NetServer 线程通过 External 实例接收客户的调用请求。

· NetServer 将调用请求交给主 Skeleton 线程分配。

· 主 Skeleton 线程创建另一个 Skeleton 线程处理该调用。

· Skeleton 线程调用服务实现的方法，并得到结果或异常。

· Skeleton 线程将调用结果或异常通过 External 实例发送给客户。

结论　本文我们仅仅讨论了基于库函数的 ORB 运行时系统的一种简单设计方案，在实际中，ORB 运行时系统提供的功能远不止处理静态调用，它还可以提供名字服务，处理动态调用等等，这时 ORB 运行时系统要复杂得多。

参 考 文 献

1　OMG. RFP5 Submission-Trading Object Service. OMG Document orbos/96-05-06, Object Management Group, Framingham, MA, 1996

2　OMG. OMG CORBA Services: Common Object Services Specification. OMG T Document Number 95-3-31, Object Management Group, Framingham, MA, 1995