基于移动 Agent 技术的构件软件框架研究*

吕建 张鸣 廖宇 陶先平

(南京大学计算机软件新技术国家重点实验室 南京 210093)

(南京大学计算机软件研究所 南京 210093)

E-mail: lj@nju.edu.cn

摘要 随着移动 Agent 技术的快速发展以及构件软件的广泛应用,对一种新型的基于移动 agent 技术的构件软件框架的需求日益增长.通过对传统的构件软件框架的不足的分析,提出了一种新的构件软件框架.与传统的构件软件框架相比,新的构件软件框架具有较强的网络环境的动态适应性.

关键词 构件软件,构件软件框架,智能连线,构件装配.

中图法分类号 TP311

基于结构化程序设计方法学或面向对象程序设计方法学,传统的构件软件技术是一种基于"总线"的标准,其主要机制是过程调用和对象允引.在单机的计算环境中,存在一条"短总线",即本地过程调用和本地对象允引;在网络计算环境中,存在一条"长总线",即远程过程调用和远程对象允引;而在异构环境中,相应地也有一条"混成总线".一般说来,传统的构件软件技术存在以下不足:各个构件是固定和被动的,构件间的连线是"笨拙"(silly and stubborn)的.其"笨拙"性主要表现在:(1) 灵活性差,即它只能表达单一的以数据传输为基础的方法调用;(2) 效率不高,即调用构件和被调用构件之间的多次交互需要通过多次方法调用来完成;(3) 坚定性不强,不能有效地支持调用构件和被调用构件之间的断开式交互;(4) 装配欠灵活,即构件的功能和其结构机制往往融合在一起,难以对灵活的装配手段提供有效的支持.这些不足在某种意义下制约了构件技术在网络环境下的广泛应用.

例如,目前的构件软件技术对客户端应用或者单机交互式应用的开发比较成功,但在服务器端的应用方面,其支持力度相对要弱[1,2].主要问题之一是,在传统方法学支持下的构件组装模型通常具有使用前组装(assembly before use)的特征,因而要求开发者较早地决定构件在分布式系统中的位置,从而使得构件的独立性受到影响.为了避免因面向对象技术中允引机制而要求开发者较早地决定构件在分布式系统中的位置这一问题,CORBA(common object request broker architecture)技术[3]通过基础环境支撑的方法来解决,从而可对跨平台的基于 Client/Server 结构的构件软件提供有效的支持.然而,值得一提的是,由于 CORBA 技术在本质上采用的仍然是面向对象技术,因此,传统的"笨拙"总线所具有的灵活性差、效率不高和坚定性不强等问题依然存在.

移动 agent 技术为解决上述问题提供了可能[4~6].一般说来,移动 agent 是一个独立运行的计算机程序,具有自主性、移动性、协作性、安全性和智能性等特征.(1) 自主性.这是指 agent 可按照自己的意愿完成特定的任务而无需用户的过多干预;(2) 移动性.这是指 agent 可以在任意站点上暂时中断执行,在异构网络(如 Internet)上移动,并在目的站点上停留下来再恢复执行;(3) 协作性.这是指若干个移动 agent 可以在网络中相互通信和

---

* 本文研究得到国家自然科学基金(No.69873021)、国家"九五"重点科技攻关项目基金(No.96-729-1-08)、国家 863 高科技项目基金(No.863-306-ZT02-02-03)、国家杰出青年基金(No.61525204)和江苏省应用基础研究基金(No.BJ99016)资助.作者吕建,1960 年生,博士,教授,博士生导师,主要研究领域为形式化方法,对象技术,分布计算技术,构件技术.张鸣,1973 年生,博士生,主要研究领域为对象技术,构件技术.廖宇,1974 年生,硕士生,主要研究领域为对象技术,构件技术.陶先平,1970 年生,博士生,讲师,主要研究领域为对象技术,移动 agent 技术.

本文通讯联系人:吕建,南京 210093,南京大学计算机软件研究所

本文 2000-01-17 收到原稿,2000-06-12 收到修改稿

合作,共同完成某一任务;(4) 安全性.这是指对agent本身及agent运行环境的安全性保障;(5) 智能性.这是指agent具有一定的自适应能力,可对环境的变化作出适当的反应.移动agent技术的重要特征之一在于它对网络环境的适应能力,如它可以减轻网络负载和支持间断计算.

然而,基于移动agent技术的构件框架的研究是一个全新的研究领域,研究工作才刚刚开始.文献[1]只提出了支持构件灵活组装的必要性和基于移动agent技术的解决思想.CORBA框架已开始意识到移动agent技术在网络环境下的重要性,并在新的版本中考虑对它的支持.但总之,这些方法还都缺乏成熟的技术和已实现的支撑系统.

为此,我们对基于移动agent技术的构件框架进行了系统研究,基本思路是,基于移动agent的技术和方法,将构件功能与构件结构两部分加以分离,使构件具有动态的移动性,构件之间的连接总线具有智能性,从而克服了传统总线灵活性差、效率不高和坚定性不强的问题.由于构件功能和构件结构的分离,使得构件的功能部分可独立开发,而构件软件的结构可在使用时装配完成,并允许进行动态调节,即通过静态后组装(late assembly)的方式来解决传统总线装配欠灵活的问题.结果可得到一类新的具有较大结构灵活性和网络环境适应性的构件软件框架[7].

本文主要讨论一种新型的构件软件框架的设计,此框架采用基于轻型agent的智能总线来取代传统构件框架的"笨"总线,从而可以在某种意义下解决传统构件框架的灵活性差、效率不高和坚定性不强的问题,为进一步增加构件技术的网络适应性,扩大构件技术在网络环境下的应用提供有效的支持.

1 新型框架的设计思想

下面,我们首先举例说明构件软件框架的特征.

假设有这样一个应用:客户程序C(在主机host1上)需要请求host2上的3个服务$m1()$,$m2()$和$m3()$(分别由不同的服务对象提供),并且在每次服务执行完成之后需要进行一段很小的处理,分别称为P_1,P_2和P_3.传统的基于RPC(remote procedure call)模型的处理方法的执行流程如图1所示.

在客户C发出的3次请求过程中,host1需要与host2进行3次交互,并且,一般来说,在远程调用的执行过程中必须保持持续的网络连接,从而使得构件软件的坚定性较大地依赖于网络的坚定性.另外,如果这样的请求比较频繁的话,将会使网络开销增大,影响系统的效率.如果我们将3次(或多次)远程请求以及方法的后处理代码P_1和P_2进行组合,通过移动agent发送给目标节点,由移动agent在目标节点进行相应的局部交互,最后再由移动agent将最终结果一次返回给请求的客户(其执行控制流程如图2所示),则这种新的模型在系统坚定性(更好地支持远程服务执行过程中的网络可断)、效率(多次请求避免多次远程交互)等方面较传统的基于RPC的模型就有了明显的改善.如果客户C需要的服务是由在不同节点上的对象提供时,上述新模型也同样适用.

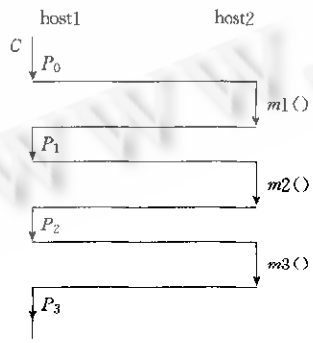


Fig. 1 Execution control based on RPC model
图1 基于RPC模型的执行控制

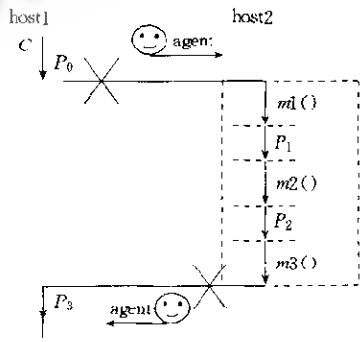


Fig. 2 Execution control of new model (1)
图2 新模型的执行控制(1)

再来看另外一种情况,假设客户C对host2上的几个请求服务($m1()$,$m2()$,$m3()$)由于某种原因需要由其他节点(如host3,host4等)上的其他对象来提供($m4()$,$m5()$),那么开发人员必须重新开发客户端的程序C以

适应这种变化.如果采用新的框架,将构件的结构组装与客户程序体分开,就可以简单地将原来的一组请求替换为一组新的请求,或者对原有的方法调用赋予新的实现,而客户程序可以不受任何影响(如图3所示),从而使构件的组装具有较大的灵活性.显然,这也是传统的基于RPC的构件框架所不具有的.

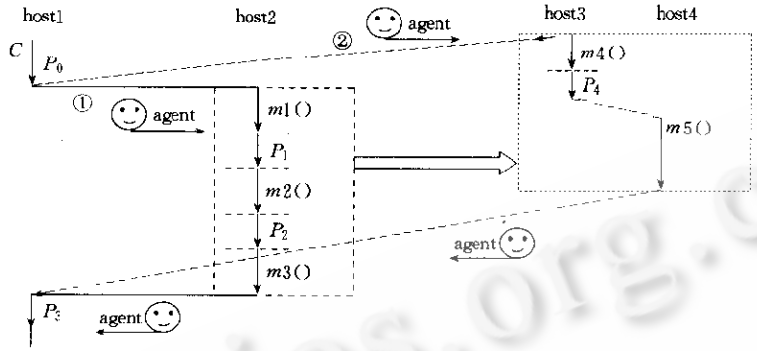


Fig. 3　Execution control of new model (2)
图3　新模型的执行控制(2)

在新型框架下,我们首先将构件的功能和构件的结构机制加以分离;然后用基于移动agent的"智能连线"来取代传统的基于RPC的连线.构件的功能部分可以与CORBA相类似,采用各种语言书写;而基于移动agent的智能连线则以Java语言为基础.为了实现基于Java的智能总线与异构的构件功能体之间的连接,从而支持跨平台的构件间的互操作,我们采用一种类似于CORBA IDL的中性语言来描述构件的接口(包括请求接口和服务接口).为了给构件的组装提供较大的灵活性,我们将智能连线分为两部分,一部分是组调用表(group table),它基于其他构件中所提供的接口,可采用各种形式对功能体中的各种方法在组装时刻灵活地加以解释;另一部分是定位表(location table),它提供了组调用表中所涉及的各个构件在网络中的位置信息.由于智能连线部分和功能体部分是分离的,从而为构件的组装提供了较大的灵活性.实际上,组调用表和定位表这两部分内容组合在一起即可以形成一个特定的移动agent,这个agent通过流动将该构件的一系列相关的调用请求传给相应的服务构件执行,执行完成后返回结果.在某种意义上说,构件软件框架就是用移动agent替代了CORBA中的ORB(object request broker),如图4所示.

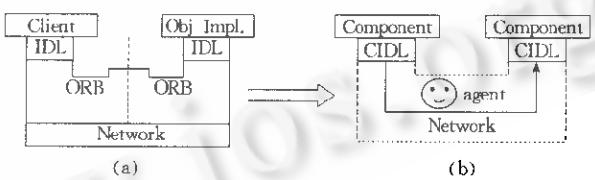


Fig. 4　CORBA model and new componentware framework model
图4　CORBA模型与新型构件软件框架模型

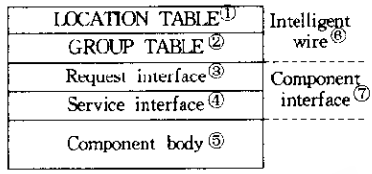
与CORBA模型相比,虽然CORBA中的动态激发机制使得调用方可在运行时刻动态地确定服务方,以适应服务方的灵活变化,但是其不足之处在于,它需要将这种灵活性以程序代码的形式体现,一方面增加了调用方的复杂性和运行开销;另一方面,这种灵活性受限于相应的程序代码.更进一步地,动态激发机制还是基于传统的RPC方式,仍然没有从根本上解决传统连线方式只能传输数据、坚定性不强和效率不高等问题.而构件软件框架只是在装配时用基于移动agent的机制来建立调用者和被调用者的关系,不仅执行效率高、坚定性强,而且装配的灵活性更大,更能适应网络环境的变化需求.更进一步地,由于智能连线中包括程序代码,我们不仅可以提供装配时的灵活性,而且可以支持类似于动态激发机制的灵活性.

2　新型框架的基本结构

在新型框架下,构件的一般形式如图5所示.

一个完整的构件可描述如下:

```
component com_name{
    location{}          //构件位置信息
    group{}             //构件组调用
    request{            //请求接口
        rm1;
        ...}
    service {           //服务接口
        sm1;
        ...}
    body{               //构件功能体
    ...}
}
```



①定位表,②组调用表,③请求接口,④服务接口,⑤构件体,⑥智能连线,⑦构件接口.

Fig. 5 The component structure
图5 构件结构

下面,我们详细讨论新型框架下构件的具体结构:

(1) 构件体. 构件的具体实现代码,主要是指对构件所提供的服务接口的具体实现. 各个构件体可以用不同的语言加以编写.

(2) 构件接口. 用构件接口描述语言 CIDL(component interface definition language)描述,包括构件请求接口(request interface)和服务接口(service interface)两部分. 请求接口是指该构件需要向其他构件请求的服务. 服务接口是指构件本身实现的接口,它包括了本构件提供给其他构件的服务. 构件接口是进行构件装配的根本依据,也是建立构件之间请求和服务关系的唯一途径,因此,构件之间必须采用一种统一的接口规范. 基于 CORBA IDL 的思想,我们也设计了构件的接口定义语言 CIDL.

CIDL 是用于描述构件的请求接口和服务接口的语言. 一个 CIDL 规约内包含一个或多个模块定义、接口定义、类型定义、常量定义以及异常定义. 一个模块定义内可以包含一个或多个模块定义、接口定义、类型定义、常量定义以及异常定义.

CIDL 主要描述的是构件接口的语法. 用户或系统在对构件请求接口和服务接口进行匹配时,主要也是基于语法级别的匹配. 接口语义的描述是一个很困难的问题,目前尚未有较为理想的方法. 我们在进一步的研究中将会作一定的考虑,希望能在 CIDL 中增加一些有关语义描述的方法.

(3) GROUP 表. GROUP 表将一组相关的调用及调用后的处理组合在一起,形成一个逻辑单位. 这一组调用请求通过移动 agent,根据 GROUP 表所提供的执行逻辑次序,流动到提供该服务的 component 所在的节点,完成请求的执行. GROUP 表中所有的调用必须是该 component 所不能提供的服务,属于请求接口. 关于 GROUP 表中请求调用所作用的 component 对象的初值以及各 component 的位置信息由 LOCATION 表提供.

GROUP 表提供了构件请求的执行控制流程. 考虑到 GROUP 表对生成 agent 的影响,从简单、灵活以及表达能力等多方面出发,我们设计 GROUP 表的结构如下:

Group name (p_1,...)				
Group private data				
Structure type①	(condition②)	Component reference③	Method④	After processing⑤

①结构类型,②条件,③构件指引,④方法,⑤方法后处理.

上面的结构包括 GROUP 表的名 Group_name 和可能有的参数表、私有数据以及控制结构. 其中,根据实际应用可能会有的控制逻辑,我们定义了4种类型的控制结构,它们包括顺序结构、并行结构、顺序循环结构和并行循环结构. 现举例说明如下:

(a) 并行结构

PAR	R_1	$I_1.M_1$	P_1
	R_2	$I_2.M_2$	P_2
	...	...	...

其含义是,并行执行表中的所有方法 $I_i.M_i$(接口 I_i 的方法 M_i)及其方法后处理 P_i,直到所有的方法及其(可能有的)方法后处理全部执行完成后结束.其中 $R_1,R_2,\ldots$ 表示 $M_1,M_2,\ldots$ 操作作用的具体构件对象.

(b) 顺序循环结构

SLOOP	Condition	R_1	$I_1.M_1$	P_1
		R_2	$I_2.M_2$	P_2
		...	...	...

其含义是,当条件 Condition 为真时,循环按顺序执行表中的方法及其方法后处理($I_1.M_1,P_1,\ldots$),直到条件 Condition 为假时结束循环的执行.其中 $R_1,R_2,\ldots$ 表示 $M_1,M_2,\ldots$ 操作作用的具体构件对象.

由于选择结构可以将选择条件提取出来,在构件功能体中加以实现,因此,这里的 GROUP 表的结构类型没有包含选择结构.4种结构之间不允许嵌套,并且方法后处理 P 的代码段应该尽可能小(基于 agent 应该是轻量级的考虑).

(4) LOCATION 表.LOCATION 表用于描述构件软件系统所需的构件(包括构件对象的初值)所在的位置信息.移动 agent 正是根据 LOCATION 表的信息来决定流动的目标节点.LOCATION 表的结构可以简单地描述如下:

Location_Table_Name			
R_1	host1	Component11	*Inst*1
R_2	host1	Component12	*Inst*2
R_3	host2	Component21	*Inst*3
...	...	...	...
...	...	...	...

其含义是,GROUP 表中 R_1 的初值为节点 host1 上的构件 Component11 的实例 *Inst*1,R_2 的初值为节点 host1 上的构件 Component12 的实例 *Inst*2 等等,依此类推.若 R_i 初值为空,则表示 R_i 在 GROUP 表中将由方法后处理来给它赋以初值.

3 智能连线与后组装机制

传统的 RPC 机制是基于控制移动的机制,就是说程序的执行控制通过网络从请求节点转到服务节点继续执行,执行完成后返回请求节点.在整个过程中没有执行代码的移动,仅是执行控制的转移.实际上,RPC 模型实现的是数据移动:数据通过网络以参数或调用返回结果的形式进行互换转移.

新框架构件的 GROUP 表和 LOCATION 表组成了智能连线,它实际上是程序代码和数据的结合.因此,这种代码加数据的移动比传统 RPC 的单纯的数据移动具有功能强、灵活性大等优点.

智能连线中起主控作用的是 GROUP 表,LOCATION 表仅存放构件的位置信息,对于构件执行流程不起任何控制作用.

当在构件体中有某个 GROUP 调用 gcall 时,首先根据 gcall 在 GROUP 表中定义的内容,找到需要执行第1个方法 M_1 的构件对象 R_1,查找 LOCATION 表,得到构件对象 R_1 所在的节点位置 host1,由 GROUP 表和 LOCATION 表生成的 agent 则移动到节点 host1 上,在节点 host1 上解释 agent 并执行方法 $R_1 \rightarrow I_1.M_1$(本地调用)以及方法 M_1 的方法后处理部分代码;然后再根据下面需要执行方法 M_2 的构件对象 R_2 所在节点 host2 位置,agent 再继续流向下一节点;直到 GROUP 表的执行结束返回初始节点.

由于功能体和智能连线机制是分离的,我们可提供分别编译的方式对构件的组装提供灵活的支持.例如:

(1) 分布系统的构件位置可在组装时确定.即可根据网络的实际情况在组装时再确定构件在网络中的位置.

(2) 组装结构的灵活调整.如果在使用中发现一个或多个更好的构件可以替代现有的构件,所要修改的只是位置表,而其余的各个部分可以保持不变.例如,客户可以通过仅修改 LOCATION 部分来重新选择 server 端构件提供的服务.

(3) 功能体中组方法调用的灵活解释.对于构件功能体中的一个组方法调用,对其配以不同的组调用表;它便可以具有不同的含义,从而使得人们可在不改变构件体的情况下,只需通过改变组调用表而使构件体具有网络适应性.例如,客户可以通过仅修改智能连线而使得原来由一个服务器提供的服务现在可以由多个服务器并行地或顺序地完成.

概括说来,新框架具有功效较高、坚定性好、灵活性强等特点,对网络环境具有良好的适应性.

4 结束语

目前,我们已经基本完成对新型构件框架的详细设计,自行设计实现了其基础支撑环境——移动 agent 系统 Mogent 1.0[6,9],同时,正在对构件加入移动特性后形成的可移动构件软件框架进行关键技术的分析和研究.这类新的构件框架结构具有较大的结构灵活性和网络环境的适应性.

进一步的工作包括对可移动构件软件框架的设计以及如何将该构件框架和可移动构件框架结合起来,形成一种灵活的构件框架.当然,这是一项难度较大的工作.

参考文献

1 Szyperski C. Component Software — Beyond Object-Oriented Programming. Cambridge, MA: Addison-Wesley Publishing Company, 1997

2 Microsoft Corporation. Microsoft COM homepage. 2000. http://www.microsoft.com/com

3 Object Management Group. CORBA, OMG Website. 2000. http://www.omg.org

4 Vigna G. Mobile Agents and Security. Lecture Notes in Computer Science 1419, Berlin, Heidelberg: Springer-Verlag, 1998

5 Lange D B. Mitsuru Oshima: Programming and Deploying Mobile Agents with Aglets. Reading, MA: Addison-Wesley Publishing Company, 1998

6 White J. Mobile agents. In: Bradshaw J ed. Software Agents. Cambridge, MA: MIT Press, 1996. 437~472

7 Lü Jian. Some research on componentware frameworks based on mobile agent technology. Technical Report, State Key Laboratory for Novel Software Technology, Nanjing University, 1999. 1~35

8 Lü Jian, Tao Xian-ping, Dong Huan *et al*. The design of a mobile agent system Mogent 1.0. Technical Report, State Key Laboratory for Novel Software Technology, Nanjing University, 1999. 57~71

9 Lü Jian, Tao Xian-ping, Dong Huan *et al*. A new structured navigation mechanism in the mobile agent system Mogent 1.0. Technical Report, State Key Laboratory for Novel Software Technology, Nanjing University, 1999. 102~113

Research on Componentware Framework Based on Mobile Agent Technology

LÜ Jian ZHANG Ming LIAO Yu TAO Xian-ping

(*State Key Laboratory for Novel Software Technology Nanjing University Nanjing* 210093)
(*Institute of Computer Software Nanjing University Nanjing* 210093)

Abstract With the development of mobile agent technology and the widespread of componentware, the requirements for a new componentware framework based on mobile agent technology are increasing. After analyzing the limitations of traditional componentware framework, the authors propose a new componentware framework in this paper, which has better dynamic adaptability to network environments than the traditional componentware framework.

Key words Component software, componentware framework, intelligent wire, component assembly.