国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第 19 卷第 4 期　1997 年 6 月　Vol. 19 No. 4

基于 agent 的分布式集成环境*

史殿习　王怀民　邹　鹏　吴泉源

(国防科技大学计算机系　长沙　410073)

摘　要　本文提出了一种基于多 agent 的分布式集成环境 DIEMA 的通用框架，描述了 DIEMA 的工作原理；提出了服务 agent 和请求 agent 的实现框架及其设计与实现。

关键词　客户/服务器，分布式计算，agent，框架

分类号　TP39

The Distributed Integrated Environment Based on Multi_ agents

Shi Dianxi　Wang Huaimin　Zhou Peng　Wu Quanyuan

(Department of Computer Science, NUDT, Changsha, 410073)

Abstract　First, this paper promotes the framework architecture of DIEMA. Then, we discuss the mechanism and executive of the DIEMA, and promote the implementation framework of Service agent and Request agent and their design and implementation.

Key words　client/server, distributed computing, agent, framework.

进入 90 年代后，分布与协同计算的应用问题变得越来越复杂，用户对分布式计算环境的要求也越来越高，用户迫切希望在网络上建立更为丰富的分布式应用，不仅实现数据共享，而且能实现各计算实体的协同工作。现在的计算环境在很多方面不能满足应用的需要，这迫使我们采用新的技术和方法来研究和开发满足用户需要的、支持协同计算的分布式计算环境。为此，我们开展了基于多 agent 的分布式集成环境(简称 DIEMA)的研究与开发，试图提供一种支持分布式应用的集成环境，方便用户开发客户/服务器应用。

1　agent 的概念

agent 是指在协同计算环境中持续自主发挥的计算实体。从计算的角度来看，agent

* 图家 863 高技术基金资助
1996 年 8 月 28 日收稿

通常由一组相互关联的并发进程来实现。进程是实现并发计算的基本计算实体,而 agent 则是作为实现协同计算的基本计算实体。并发计算概念与协同计算概念联系密切,可以说,相互关联的并发进程之间的关系就是一种协同计算的关系。由于进程的概念和结构是在并发计算的需求下提出的,进程机制难以很好地反映在网络计算环境下协同计算对计算实体的一般要求。这些要求涉及:

(1) 支持 agent 的自主性和相对独立性,使其能够根据内部状态和外部消息事件控制自身的行为,无需外界的频繁指导或干涉;

(2) 支持数据、过程和通信设施的封装,使 agent 有明确的边界、标识和引用的概念;

(3) 支持消息的通信,直接给出与通信意图有关的信息;

(4) 支持新的过程加入,以及数据与通信信息修改的重组能力。

而 agent 通常具有独立性、自主性、交互性和分布性等特性,能较好地满足在网络计算环境下协同计算对计算实体的一般要求,从而能够有效地促进协同计算。

2 DIEMA 的通用框架

针对 agent 的上述特性,我们开展了基于 agent 的分布式集成环境 DIEMA 的研究和开发。DIEMA 的基本目标是为应用开发者提供开发分布式应用的通用框架(见图 1),支持应用开发者灵活方便地建立各种客户/服务器应用。DIEMA 的框架可理解为"软件总线",其核心是基于 agent 的服务请求代理机制,各类服务 agent 可作为"软部件"插接到该框架上。客户应用则通过本地称为服务请求 agent 的对象,访问服务 agent. 请求 agent 的存在使得客户应用所需的异地服务如同在本地一样。

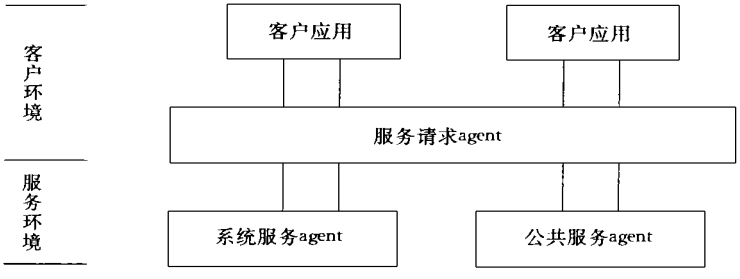

图 1 DIEMA 的通用框架结构

从客户/服务器计算角度来看,DIEMA 分为两个部分:一部分是客户环境,一部分是服务环境。其中,

(1) 客户/环境:客户环境由客户应用和请求 agent 组成。该环境不仅提供了典型服务 agent 的请求 agent,而且提供了一组请求构造框架及其相应的功能简洁的 API,支持应用开发者建立所需的请求 agent.

(2) 服务环境:服务环境由一组系统服务 agent、公共服务 agent 以及一组服务构造框架组成。系统服务 agent(如 Register)负责提供系统服务,保证 DIEMA 计算环境的正常工作;公共服务 agent 是某类服务的提供者,负责为客户应用提供服务;服务构造框架用于完成接收客户应用发来的各种服务请求的通信管理,并提供与特定服务进程的挂接机

制。应用开发者采用这一框架和挂接机制可以建立新的服务,并加入到服务环境中。

利用 DIEMA 的框架结构建立的客户/服务器应用涉及系统服务 Register,服务 agent,请求 agent 和客户应用。其工作过程为:首先启动系统服务 agent Register,其作用是管理整个 DIEMA 环境中服务 agent 的地址信息及相关的管理信息(如使用权限等),实现 agent 的名字与物理地址之间的映射,服务 agent 生成后首先向 Register 注册,然后启动监听进程;当客户应用启动后,其中的请求 agent 自动向 Register 查询并获得服务 agent 的访问信息;当需要服务 agent 的服务时,客户应用只需向请求 agent 发出服务请求;请求 agent 接收到客户的请求后,通过下层的网络计算设施将请求传递给服务 agent,接收服务 agent 接收到客户的请求后,通过下层的网络计算设施将请求传递给服务 agent,接收服务 agent 的服务结果,并返回给客户应用。

实现 DIEMA 的关键是开发服务 agent 和请求 agent 的实现框架。以下讨论请求 agent 和服务 agent 的实现框架在 UNIX 环境下的 C++ 实现版本。

3 请求 **agent** 的框架结构

请求 agent 的框架包括构造框架和远程消息传递机制两个部分。构造框架描述如何生成服务 agent 在本地的代理请示 agent,远程消息传递机制描述请求 agent 与服务 agent 之间的消息如何进行传递。构造框架为应用开发者定义请求 agent 提供支持机制,主要包括自动向 Register 查询它所代理的服务 agent 的地址信息机制,自动接收来自 Register 的通知消息机制以及生成请求 agent 的定义模板等。远程消息传递机制提供各种通信手段(如同步、异步消息传递机制等),将客户应用的服务请求和服务方法的参数传递给服务 agent,并接收服务 agent 的返回结果,最后,将结果返回给客户应用。

在具体实现请求 agent 的框架时,DIEMA 将构造框架中基本机制和远程消息传递机制定义在标准类 Request __ agent __ Base 中,并为应用开发者提供生成请求 agent 的定义模板,其形式如下:

```
class __Requestagent: public Request __agent __Base{
____//有关变量定义
Public:
__Requestagent(相关参数),
服务方法请求的定义;
};
—Requestagent::__Requestagent(相关参数):
Request __agent __Base(相关参数){};
____//请求服务 agent 服务方法的实现;
```

其中,请求服务方法的实现可由服务 agent 服务方法的定义(服务方法名、参数和类型)唯一确定。例如,请求服务方法 int methodl(int arg) 的实现为:

```
int __Requestagent::methodl(int arg){
int retval;
Method __Call(methodl,(xdrproc __t)xdr __int,(char*)&arg,(xdrproc __t)xdr
```

__int,(char*)&retval);
return relval;}

其中Method__Call是在父类中定义的传递服务请求并同步接收服务结果的方法。

4 服务agent的框架结构

服务agent框架的设计思想是支持应用开发者灵活地开发应用服务agent. 服务agent框架包括构造框架和执行框架两个部分。构造框架描述应用服务agent是如何被定义的,执行框架描述应用服务agent中的服务方法是如何被激活的。

服务agent框架的核心是如何有机地将构造框架和执行框架结合起来。DIEMA的做法是为用户提供一个基本的联编机制,由用户在构造应用服务agent时,通过这一联编机制将服务方法与服务方法的引用信息联编得到一个方法引用,并由服务agent的构造框架来管理和维护;当有来自客户方的请求服务的消息时,执行框架根据对服务请求的处理结果使用方法引用表的信息,激活被请求的服务方法,为客户应用提供服务。这种方法的优点是透明性好,同时可以在构造方法引用表时,将一些控制信息填入到表中(如访问权限等),以控制客户对服务方法的访问,增加DIEMA的安全性和可靠性。

在具体实现服务agent的框架时,DIEMA将构造框架和执行框架中的基本机制定义在标准类Server__agent__Base中,并为应用开发者提供生成应用服务agent的定义模板,应用开发者按照该定义模板可以方便地生成应用服务agent,其形式如下:

```
class __Serviceagent: public Service __agent __Base {
____//有关变量定义
public:
- Serviceagent(相关参数);
服务方法的定义;
};
__Serviceagent::__Serviceagent(相关参数);
Service __agent __Base(相关参数){
各个服务方法的地址与引用特征联编;
将应用服务agent挂接的服务环境中;
启动执行框架;
}
____//服务方法的实现;
```

5 DIEMA与OSF DCE的比较

OSF DCE是一个由OSF组织开发的分布式计算环境。该环境对开发分布式客户/服务器应用提供了相应的支持机制。与OSF DCE相比,DIEMA具有很多优点,主要表现在如下几个方面:

(1)DIEMA是建立在对象技术之上,是一个面向对象的分布式计算环境;OSF DCE是建立在RPC机制之上,是一个面向过程的分布式计算环境;

(2) 从面向对象的角度来看, 在 DIEMA 中, 服务 agent 管理自己的状态信息, 客户只与服务 agent 的本地代理发生关系, 由代理和实际的服务 agent 相互作用完成客户请求; OSF DCE 是基于远程过程调用的, 客户只能以过程调用的方式调用服务器的过程;

(3) 从透明性的角度来看, DIEMA 将很多复杂的工作交给请求 agent 隐式完成, 增加了透明性, 用户不需知道服务 agent 的物理位置, 而由请求 agent 自动定位服务 agent 的物理位置; OSF DCE 中的远程过程调用操作复杂, 通常情况下客户需要事先知道服务器的位置。

6 结论

DIEMA 采用 90 年代先进的分布对象技术, 提供了灵活开发客户/服务器应用的通用框架(软件总线)以及基本的分布管理。利用 DIEMA 的通用集成框架, 我们与北京某院合作, 在短短的两周时间内将 CAD 开发工具 ProE 及有关的工具集成为一个支持 CAD 应用开发的分布式计算环境, 大大缩短了应用开发周期, 同时提高了 CAD 应用产品的开发效率。DIEMA 的开发与应用已经体现出基于 agent 的分布式集成中间件的优势。

参考文献

1 史殿习等, 分布式系统中的名字服务 . 计算机科学与工程 . 1995, 3

2 Khanna R. (editor) Distributed Computing: implementation and managcment stratcgies. Prenticc Hall, 1994

3 Mowbray T J, Brando T. Interopcrability and CORBA-based open systems. Object magazine, 1993, Scptcmber-Octobcr: 50～54

4 王怀民等 . 基于 agent 的分布式计算环境 . 计算机学报 . 1996. 3

5 Corbin J R. The Art of Distributcd Applications: Programming Techniques For Remote Proccdure Calls. Spring-Verlag, 1991

(责任编辑　张　静)