第9卷第2期 软 件 学 报 Vol. 9, No. 2
1998年2月 JOURNAL OF SOFTWARE Feb. 1998

分布式集成中间件CSE/MA的设计与实现*

史殿习 王怀民 邹 鹏 高洪奎 吴泉源

(长沙工学院计算机系 长沙 410073)

摘要 为扩展分布式计算环境,支持系统集成,本文开展了基于多agent的分布式集成中间件CSE/MA的研究,试图为应用开发者提供一种开发客户/服务器应用的通用开发框架,支持应用开发者方便灵活地建立各种客户/服务器应用.本文首先提出CSE/MA的框架结构,讨论利用CSE/MA建立客户/服务器应用的组成成分、工作原理和工作过程;其次,讨论CSE/MA的管理核心Register所提供的服务及其实现算法,提出服务agent和请求agent的实现框架及其设计与实现;最后对CSE/MA的技术特点和地位作了概括性的总结.

关键词 agent,客户/服务器,分布计算,中间件,框架.

中图法分类号 TP316

分布式客户/服务器计算机系统是90年代分布式处理的主流系统.[1]该系统通过网络计算机平台和分布式客户/服务器集成中间件,把分布在网络上的客户机与服务器有效地集成在一个系统中,使每个用户可以通过客户/服务器方便地从"幕后"的服务器上获得强大的信息服务、系统管理和运算能力.其中OSF/DCE和Sun/ONC+是代表90年代初分布式计算技术发展水平的主流产品.[2]其技术特点是:(1)主要针对信息共享问题;(2)采用常规的客户/服务器计算模型;(3)应用程序设计界面(API)延用传统的计算概念和设施(如过程调用概念和文件设施等);(4)提供丰富的分布系统管理、服务和应用.

当前,计算技术正进入以网络为中心的计算时期,用户迫切希望在网络上建立更为丰富的分布式客户/服务器应用;不仅实现数据共享,而且支持知识共享和各类计算资源的共享;并能实现各计算实体的协同工作.应用需求的发展使分布式客户/服务器计算机系统的开发面临许多新的技术问题,其中之一是集成中间件问题.许多计算机厂商在推进客户/服务器计算的进程中,把主要注意力集中在解决服务器处理能力的瓶颈问题上,而忽视了解决客户/服务器应用中客户处理部分与服务器处理部分之间缝隙的集成中间件,集成中间件的缺乏已成为制约客户/服务器应用的关键因素.为迎接这一机遇与挑战,满足分布式客户/服务器应用的需求,我们开展了基于多agent[3,4]的客户/服务器计算环境(简称CSE/MA)的研究与开发,试图提供一种支持客户/服务器应用开发的集成中间件,方便用户开发分布式客户/服务器应用.

1 CSE/MA的框架结构

CSE/MA的基本目标是扩展分布式计算环境,支持应用系统的集成,为应用开发者提供开发客户/服务器应用的通用框架(如图1所示),支持应用开发者灵活方便地建立各种客户/服务器应用.

CSE/MA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(简记为SA)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代理(简记为RA)的对象访问SA.RA的存在使得客户应用所需的异地服务如同在本地一样.

从客户/服务器计算角度来看,CSE/MA分为客户环境和服务器环境两个部分.CSE/MA客户环境不仅提供了典型SA的RA,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RA.服务环境

* 本文研究得到国家863高科技项目基金资助.作者史殿习,1966年生,博士,主要研究领域为分布式计算.王怀民,1962年生,博士,副教授,主要研究领域为人工智能,计算机理论,分布式计算环境.邹鹏,1958年生,教授,博士导师,主要研究领域为操作系统,分布式计算.高洪奎,1942年生,研究员,主要研究领域为人工智能,分布式计算环境.吴泉源,1942年生,教授,博士导师,主要研究领域为人工智能,分布式计算.

本文通讯联系人:史殿习,长沙 410073,长沙工学院计算机系

本文1996-07-27收到原稿,1997-03-10收到修改稿

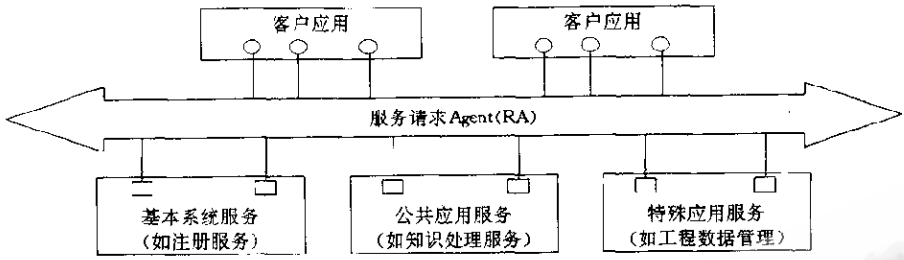

图1 集成中间件CSE/MA的框架结构

中服务可分为基本系统服务、公共应用服务和特殊应用服务.基本系统服务是保证系统正常工作的系统服务.如注册管理服务Register,使得客户方的RA可以通过SA的名字在运行时刻动态地确定SA的物理位置;公共应用服务涉及客户应用所需的常规服务(例如,文件共享服务、数据共享服务和知识处理服务等);特殊应用服务(如产品设计数据管理服务等)为特定客户提供针对性的服务.CSE/MA提供了标准的服务对象模板,应用开发者采用这一模板可方便地建立新的SA,加入到服务环境中,为客户提供服务.

利用CSE/MA的框架结构建立的客户/服务器应用涉及系统服务Register,SA,RA和客户应用.其中:

· 系统服务Register Register是系统实施分布资源管理的核心,相当于分布式系统中的名字服务.[5]其作用是管理系统中的SA和RA的地址信息及其有关管理信息,为RA提供访问SA所需的地址信息,使应用开发者可以方便地实现与SA驻留地址无关的分布式客户/服务器应用.Register的地址为系统中的SA和RA共知.

· SA SA通常驻留在一个服务器节点上.SA在生成时自动将其地址信息及有关的管理信息注册到Register上,然后启动守护进程监听网络上的服务请求.如果有请求消息达到,则进行方法匹配;如果匹配成功,则启动方法分发器,激活被请求的方法,最后将结果返回给请求者.

· RA RA与客户应用驻留在同一个节点上.RA在生成时自动向Register查询它所代理的SA的地址信息.当客户应用欲请求一个SA为其服务,即向SA在本地的代理RA发出请求时,RA利用从Register那里获得的地址信息与SA建立连接,并通过下层的远程消息传递机制RMP(remote message passing)将请求消息发送给它所代理的SA,并接收SA返回的结果,最后将结果传递给客户应用.图2显示了基于CSE/MA建立的客户/服务器应用系统的典型的工作过程.

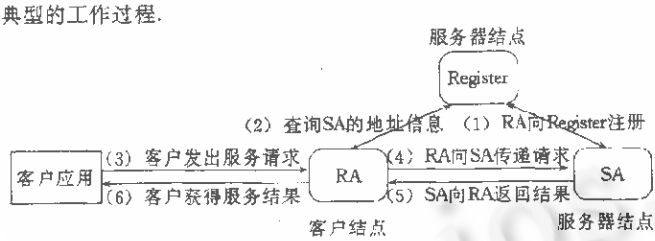

图2 CSE/MA的动态工作过程

(1) SA生成后的第1个动作是向Register注册,然后启动监听进程;

(2) 当客户应用中的RA生成后,即向Register查询它所代理的SA的地址信息;

(3) 当需要SA的服务时,客户应用只需向RA发出服务请求;

(4) RA接收到客户的请求后,即将请求传递给它所代理的SA;

(5) SA处理完RA的请求后,立刻将结果返回给RA;

(6) RA将接收来的结果返回给客户应用.

实现CSE/MA的关键是实现Register以及开发SA和RA的实现框架.以下讨论CSE/MA在UNIX网络[6]环境下的C++实现版本,其中涉及TCP/IP的Socket接口和外部数据表示XDR[7]的接口.

2 Register的设计与实现

从RA和SA的角度来看,Register相当于一个服务器,RA和SA向Register发出请求,Register根据请求类型来提供相应的服务,这些服务包括接收SA的注册、接收RA的地址查询、身份验证、自动通知、接收SA撤销注册以及其他管理功能.其中:

· 接收SA的注册 该服务首先将请求注册的SA的地址信息(如SA的名字、网络地址、端口号及其它有关的管理信息等)插入至Register所管理的注册信息库中,并判断是否是重复登记,根据处理结果给请求注册的SA发送一条相应的消息.

· 接收 RA 的地址查询　该服务是在 Register 所管理的注册信息库中查找 RA 所请求的 SA 的地址信息.如果 SA 在注册信息库中,Register 则将该 SA 的地址信息返回给请求查询的 RA;否,Register 则将请求的 RA 登记在与该 SA 有关的一个队列中,并给 RA 返回一条相应的消息.

· 接收 SA 撤销注册　该服务将请求撤销注册的 SA 从注册信息库中删除,并通知与该 SA 有关的所有 RA,其所代理的 SA 已经不在系统之中.

· 自动通知功能　系统中的 SA 可能会从系统中的一个节点移动到另外一个节点,如果不采取相应的措施,某个 SA 移动之后,它的 RA 再使用该 SA 移动之前的地址就不能找到该 SA,从而出现错误.该服务是当一个 SA 发生移动时,Register 能自动通知与该 SA 有关的 RA 新的地址信息.

· 身份验证　该服务是验证请求 SA 为其提供服务的 RA 是否有权访问所请求的 SA.

· 管理功能　Register 作为 CSE/MA 的核心,必须具有一定的管理功能.为此,CSE/MA 规划出 Register 的管理功能.这些功能包括:浏览系统中所有的 SA,浏览某个 SA 的所有 RA,浏览系统中所有的 SA 和 RA,强制撤消系统中的某一个 SA 以及所有的 RA,撤销系统中所有的 SA 和 RA 并退出系统等.

实现 Register 的算法流程如下:

(1) 系统初始化;

(2) 生成一个 Daemon(守护)进程,监听系统中的请求消息.如果有消息到达,则进行消息格式的转换,对消息进行解码,并判断消息类型;

(3) 如果消息的类型是 SA 请求注册,则调用地址设置过程,进行处理;处理完之后,Daemon 进程继续监听;

(4) 如果消息的类型是 RA 请求查询它所代理的 SA 的地址信息,则调用地址查询过程进行处理;处理完之后,Daemon 进程继续监听;

(5) 如果消息类型是 SA 请求撤销注册,则调用注册撤销过程进行处理,处理完之后,Daemon 进程继续监听;

(6) 如果消息类型是浏览系统中的 SA 或 RA 的信息,则调用相应的过程进行处理;处理完之后,Daemon 进程继续监听;

(7) 如果消息类型是退出系统,则通知系统中所有的 SA 和 RA,并退出系统.

3　SA 实现框架的设计与实现

SA 实现框架的设计思想是支持应用开发者灵活地开发应用 SA.实现框架的作用是完成接收客户应用发来的各种服务请求的通信管理,并提供于特定服务进程的挂接机制,以便应用开发者采用这一框架和挂接机制建立新的服务并加入到服务环境中.

SA 实现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 SA 是如何被定义的,执行框架描述应用 SA 中的服务方法是如何被激活的.构造框架为应用开发者定义应用 SA 提供支持机制,主要包括自动向 Register 注册机制,SA 中服务方法的定义机制,服务方法引用表的定义和生成机制以及将应用 SA 加入到服务环境中挂接机制等.执行框架对应用 SA 中的服务方法的执行提供支持机制,主要包括自动接收来自客户方的服务请求,对服务请求按通信协议进行处理,对处理后的服务请求进行识配,将识配成功的服务方法激活,并将服务结果返回给客户应用等机制.

SA 实现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,设计一个有效的方法分发器来激活 SA 中被客户应用所请求的方法,为客户应用提供服务.有两种方案可供选择:其一是在 SA 的构造函数中由用户自己来设计,用户根据接收来的方法的名字来调用 SA 中用户自己设计的、准备为客户应用提供服务的方法,并将结果返回给请求服务的客户方.这种方案的缺点是增加用户的负担,对用户来说缺乏透明性;另一种方案是为用户提供一个基本的联编机制,由用户在构造应用 SA 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,该方法引用在用户的 SA 刚刚生成时初始化,并由 SA 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,执行框架对服务请求进行处理,根据处理结果使用方法引用表的信息激活被请求的服务方法,为客户方的应用提供服务,并将服务结果返回给客户应用.比较这两种方案可以看到,第 2 种方案比第 1 种方案的透明性要好,给用户增加的负担较小,体现了框架的思想.其另外一个好处是,在构造方法引用表时可以将一些控制信息填入到表中(比如访问权限等),以控制客户对服务方法的访问,增加 CSE/MA 的安全性和可靠性.为此,在 CSE/MA 中选择了第 2 种方案来设计和实现方法分发器.

在具体实现 SA 的实现框架时,CSE/MA 将构造框架和执行框架中的基本机制定义在标准类 Service_Agent_Base 中,并为应用开发者提供生成应用 SA 的定义模板,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SA挂接的服务环境中;
启动执行框架;
};
……//服务方法的实现;
```

应用开发者按照该定义模板可以方便地生成应用SA.

4 RA实现框架的设计与实现

RA是SA在客户方的代理,其作用是在客户应用与SA之间建立一座通信的桥梁,负责客户应用与SA之间的通信管理.RA实现框架的设计思想是为应用开发者生成SA的本地代理RA提供支持,实现代理机制的思想.

RA的实现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SA在本地的代理RA,远程消息传递机制描述RA与SA之间的消息如何进行传递.构造框架为应用开发者定义RA提供支持机制,主要包括自动向Register查询它所代理的SA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RA的定义模板等.远程消息传递机制提供各种通信手段,将客户应用的服务请求和服务方法的参数传递给SA,并接收SA的返回结果,最后,将结果返回给客户应用.

RA实现框架的核心是远程消息传递机制.远程消息传递机制的核心问题是如何解决服务方法的参数传递问题.对基本的数据类型(如*int*,*char*等),远程消息传递机制可以利用字符串如"*int*","*char*"等作为标记,通过判断这些标记来获得参数的类型.例如服务方*SA*中有一个方法*fun*(*int x*,*char y*),它的两个参数类型分别为整型和字符型;在客户方,客户应用在调用*fun*(*x*,*y*)为其服务时,可以将参数以"*int* 5 *char A*"这样的形式传递给RA,RA经过处理得知"5"是整型,"*A*"是字符型,然后将其传递给SA;SA经过处理,得知"5"是整数,"*A*"是字符,并激活*fun*(5,*A*).但对于复杂的数据类型如struct类型,由于远程消息传递机制事先并不知道用户会定义什么样的结构,结构中包含什么样的内容等,因而无法解决这一类复杂的数据类型的参数传递问题.因此必须为用户提供一种系统、可靠和方便的参数传递机制.在对XDR详细研究的基础上,远程消息传递机制利用XDR机制很好地解决了参数的传递问题.利用XDR中提供的基本的编码、解码函数来构筑服务方法的参数的编码、解码函数,在SA中采用相同的解码、编码函数,从而可以很好地解决参数传递问题.

同时,为了提高系统的性能和灵活性,远程消息传递机制可以使用过去(past)、现在(now)和将来(future)等3种交互协议来实现消息的传递,它们的含义分别如下:

• 过去类型 该类型表示当一个客户应用发送一个past类型的请求时,不需等待从服务方返回的任何结果,消息发送之后,它可以继续自己的工作,而且任何时候都不会再去取请求结果.

• 现在类型 该类型表示当一个客户应用发送一个now类型的请求时,该应用必须等待来自服务方的回答消息,直到服务方返回了应答消息,该应用才能继续执行.实际上,now类型的消息提供了客户应用与SA之间的一种同步机制.

• 将来类型 该类型表示当一个客户应用发送一个future类型的请求时,客户应用希望从所请求的SA那里获得一个返回结果,但并不需要立刻获得该结果.因此,客户应用可以继续执行而不必等待来自SA的回答结果;而当客户应用需要结果时,便去取从SA返回的结果,如果SA还没有返回结果,则立刻返回;如果需要,可以继续查看结果是否返回,或者继续执行,或者等待,直到结果返回.实际上,future类型的消息提供了客户应用与SA之间的一种异步机制.

根据实际需要,在RA的远程消息传递机制中,CSE/MA提供了同步和异步这两种消息传递机制.

在具体实现RA的实现框架时,CSE/MA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RA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SA服务方法的实现;
```

其中请求服务方法的实现可由SA服务方法的定义(即服务方法名、参数和类型)唯一确定.例如,请求服务方法 *int method1(int arg)*的实现为:

```
int RequestAgent::method1(int arg){
int retval;
Method_Call (method1,(xdrproc_t)xdr_int,(char *)&arg,(xdrproc_t)xdr_int,
            (char *)&retval);
return retval;
}
```

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.

5 结束语

进入90年代,异构环境下的应用互操作性和系统集成成为分布式计算技术面临的首要问题,分布对象技术成为分布式计算环境发展的主流方向.其技术特点是:(1)主要针对异构环境下的应用互操作问题;(2)系统核心的对象管理将(对等的)客户/服务器模型与面向对象技术结合在一起;(3)提供面向对象的API;(4)已经成为建立集成框架和软件部件标准的核心技术.其中代表性的技术有Microsoft的OLE/COM和OMG(Object Managment Group)的CORBA(common object request broker architecture).[8]目前已有包括主要计算机公司在内的500余家软硬件厂商宣布支持CORBA,并有一批与CORBA兼容的产品如IBM的DSOM(distributed system object model)和DEC的ObjectBroker面市.CSE/MA在技术上遵循与CORBA兼容的原则,其主要技术特点是采用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.其中,基于agent的服务请求代理机制和自动服务机制有所创新,超越了传统的客户/服务器计算的概念,有助于适应新的协同工作的应用需求.我们与航天工业总公司二院合作,利用CSE/MA建立航天产品数据管理的Framework已取得成效.CSE/MA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.该成果已通过国家863高科技成果验收.

参考文献

1 Lewie T G. Where is client/server software headed? Computer, 1995,28(4):49~55
2 Khanna R. Distributed computing: implementation and management strategies. Prentice-Hall, 1994
3 Genesereth M R, Ketch S P. Software agents. Communications of the ACM, 1994,37(7):48~53
4 Wang Huai-min, Chen Huo-wang, Gao Hong-kui. Agent-oriented programming. Chinese Journal of Computers, 1994,17(5):367~375
5 Shi Dian-xi, Wang Huai-min. The naming service in the distributed systems. Computer Engineering and Science, 1995,3:56~65
6 Stevens W R. UNIX network programming. Prentice Hall, 1990
7 Corbin J R. The art of distributed applications: programming techniques for remote procedure calls. New York: Spring-Verlag, 1991
8 Mowbray T J, Brando T. Interoperability and CORBA-based open systems. Objec Magazine, September~October, 1993. 50~54

The Design and Implementation of the Integrated Middleware CSE/MA

SHI Dian-xi　WANG Huai-min　ZOU Peng　GUO Hong-kui　WU Quan-yuan

(Department of Computer Science　Changsha Institute of Technology　Changsha　410073)

Abstract　In order to extend distributed computing environment and support system integration, the authors are doing some research on multi-agent distributed integrated middleware CSE/MA to provide a general developing

architecture of C/S applications for application developers and support developers to bulid various C/S applications easily. Firstly, this paper provides the architecture of CSE/MA, and discusses the components, principal and process of the applications developing with CSE/MA. Secondly, it introduces the services and implementing algrithoms about the management kernal of CSE/MA——register, and provides implemcntation framework of service agent and request agent and their design and implementation. Finally, it concludes the technical features and importance of CSE/MA.

Key words Agent, client/server, distributed computing, middleware, framework.

Class number TP316