计算机科学2002Vol.29№.10

一种支持广域分布应用的中间件的研究与设计

Research and Design of A Middleware to Supporting Wide-Area Distributed Applications

赵 东 姚绍文 周明天

(电子科技大学计算机科学与工程学院 成都610054)

Abstract As a platform for a wide-area distributed application, the middleware architecture should support adaptability, scalability, mobility, reliability and security. However, current middleware solutions cannot meet these requirements. A middleware architecture, called MSWADA, is presented in this paper. It is designed to satisfy requirements of large-scale distributed systems. We focus especially on how MSWADA supports adaptability, scalability and mobility.

Keywords Middleware, Distributed system, Naming service, Adaptability, Scalability, Mobility

1 引言

过去数十年中,大规模的计算机网络日益普及。不幸的是,实际用于广域网中构筑大规模分布式应用的技术远不如构筑该网络本身的技术成熟。例如,当前基于中间件的主流解决方案,包括 DCE[1], TP-Monitor[2], CORBA[3], DCOM[4]和 J2EE[5]等,其设计都是紧耦合的,主要用于小规模网络。随着广域分布式应用,包括虚拟企业,Internet 社区以及各种各样的 B2C 和 B2B 电子商务应用等在全球的迅速增长,分布式应用仍然只用于小规模系统的假设显得越来越不现实,无法支持目前分布系统的实现。这些中间件解决方案仍然基于单服务,多客户的模式,不适用于跨越广域网的大规模分布式应用。

因此,我们提出了一种新的中间件体系结构,可以满足广域分布式应用的需求。本文首先分析广域分布式应用的需求。在此基础上提出 MSWADA 体系结构和相应的解决方案。随后详细说明体系结构中的关键技术。并比较其相关工作。最后是本文的结论和未来将要进行的工作。

2 广域分布式应用对于中间件的需求

我们认为,广域分布系统的基础设施应该满足如下需求:适应性(Adapterbility),可伸缩性(Scalability),机动性(Mobility),可靠性(Reliability)和安全性(Security)。本节分别讨论这些需求。

2.1 适应性

广域分布系统通常由许多企业的自治系统连接而成,这些系统往往使用不同的技术构建,底层的通信协议也各不相同,例如 RPC、IIOP 与 RMI/JRMP 等。因此,一个自治系统中的客户如何通过某种合适的方式透明地访问另一个系统中的服务是必须要解决的问题。对于有业务协作关系的企业,这一问题尤其突出。然而,流行的 browser/server 计算模式无法很好地满足他们之间的服务交互需求。此外,随着业务的发展,提供服务的企业随时可能更新升级其服务,这将导致一些新的问题。比如,服务提供者如何将最新的服务"推"(push)给合作伙伴的系统,又如何使自己适应合作伙伴可能的变化?因此,分布应用体系结构必须具有足够的适应能力以便当面临新的业务挑战时"插入和推出"(plug and push)其新服务。

2.2 可伸缩性

在分布系统开发中采用对象技术已成为目前的主流趋势。而在一个广域分布系统中,对象的数量往往以百万计,甚至更多。因此,如何组织这些对象,以方便高效地查找和定位是大规模分布系统必须要考虑的一个关键问题。另外,由于广域网上并发访问同一服务的客户数量峰值难以预测,使得系统的可伸缩性比起普通的系统来显得尤为重要。不幸的是,目前支持分布计算的传统中间件对可伸缩性的支持都不太好。它们的基于单名字服务器的体系结构很难满足广域分布系统的要求,也不能承受对于同一服务的高并发度访问。

2.3 机动性

在传统中间件中,名字与它们所代表的计算对象的位置紧密绑定在一起。当一个对象将其自身作为服务登记到名字服务器中时,名字服务器将它的信息,例如所在的主机地址和端口号等,编码到对象引用中,随后客户通过该对象的符号名查找到对象引用并调用服务。位置相关的名字很难支持机动性。如果在客户获取某个服务的引用后,该服务迁移到了其它的位置,则客户试图再次访问服务对象时就会遇到问题。而在一个广域分布系统中,服务对象的移动会不时发生,因此其基础设施应提供对于机动性的支持。

2.4 可靠性

作为大规模分布系统的基础设施,中间件平台应提供可靠的服务来保证跨越不可靠的广域网(包括 Internet)的应用的健壮性。然而,主流的分布计算标准都没有给出构筑可靠系统的规范,甚至其中的一些关键设施都不能保证可靠性,从而使得系统可靠性的保证只能依赖于应用系统开发者。因此,面向广域分布系统的中间件的设计也必须支持可靠性。

2.5 安全性

安全性也是广域分布系统必须支持的关键特性之一。例如,现有的 B2C 电子商务系统,往往采用 SSL 协议和 SET 协议等来保证交易的安全性。然而,B2B 系统对于安全性的要求更复杂,涉及身份认证、授权、数据保密性和数据完整性等各个方面,而且,对于不同的合作业务,不同的合作伙伴往往有不同的安全要求。在帮助广域网中的服务和客户相互确定对方是否可信赖等方面,中间件平台必须扮演一个重要角色。这一角色在存在着大量不合法机会的开放环境中是很关键的,这些机会既可能来自于具有欺诈性的服务,也可能来自于行为不端的客户。

由于传统的中间件体系结构无法满足广域分布系统的上述需求,因此必须找到另外一种满足广域分布系统需求的综合解决方案。

3 MSWADA 体系结构概述

本节我们提出作为广域系统平台的 MSWADA(Middle-

ware to Supporting Wide-Area Distributed Applications，即支持广域分布应用的中间件)体系结构，可以满足第二节中广域分布系统对于中间件平台的所有需求。本节展示 MSWADA 设计的概念和方法，尤其是对象模型和名字服务两部分。

3.1 对象模型

MSWADA 的体系结构设计采用了面向对象技术。对于广域分布系统而言，有必要支持提供同一接口的同一服务有不同的实现。面向对象的方法能够将服务的状态与服务的操作封装起来，从而使服务接口与实现清晰地分离。需要指出的是，MSWADA 本身的设计虽然是面向对象的，但服务实现不一定要求采用面向对象的技术，更详细的解释在4.1节中给出。从 MSWADA 的观点来看，一个广域分布系统由多个自治系统组成。在每个自治系统中，需要访问其它自治系统的协作应用只是本系统的一部分。为了确保安全性，一个自治系统通常会将与外界连接的部分划分出来，通过防火墙与系统的其他部分隔离。在 MSWADA 中，我们称前者为外部域(External-Domain，通常称为停火区)，后者为内部域(Internal-Domain，例如一个企业的传统业务系统)。

MSWADA 的对象模型如图1所示。在该模型中，基本的计算单元是服务。服务提供了从外部世界访问内部域业务系统的方法，这样的服务在 MSWADA 中称为“输出服务”(exported service)。

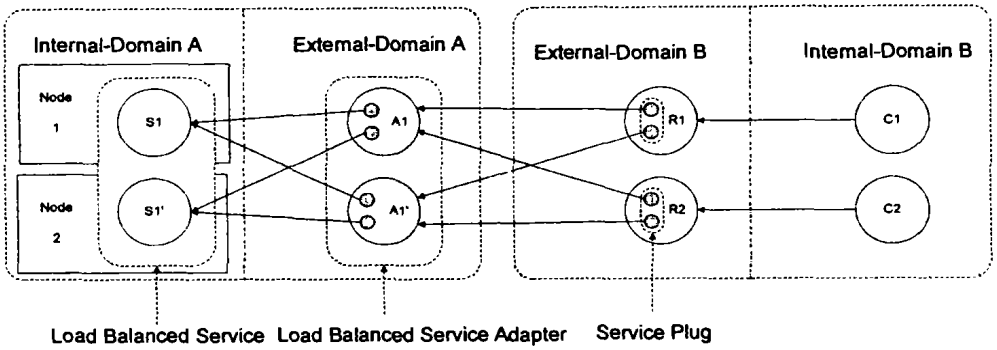

图1 MSWADA 的对象模型

如前所述，出于安全考虑，位于自治系统外部的客户只允许访问外部域。然而，为了完成客户的请求，服务实现通常需要访问内部域，这就产生了矛盾。解决这个问题的方法是在外部域中引入服务代理，在 MSWADA 中称为服务适配器(Service Adapter)。服务适配器本身是面向对象的，尽管它提供的服务不一定是。如图1所示，为了提高性能，若内部域系统的技术支持，MSWADA 可以支持服务的负载均衡。换句话说，如果内部域系统支持同一服务可以有多个运行的实体，当请求相应的服务适配器调用服务方法时，它可以选择一个服务实体执行请求。同样，出于安全考虑，许多自治系统也不允许内部域中的客户直接访问外部系统。因此 MSWADA 在客户方也引入了客户代理，称为接待员(Receptionist)。一个接待员可接受多个客户的并发服务请求，不同的服务代理在接待员中映射为不同的服务插头(Service Plug)。当客户向其接待员请求服务时，接待员通过对应的服务插头访问服务适配器，后者再调用对应的服务方法处理客户的请求。当要访问的服务有多个匹配的服务适配器时，服务插头负责从中选择一个。换句话说，服务插头是一种智能存根，也具有负载均衡能力。

3.2 名字服务

调用服务之前首先必须定位相应的服务。对于 MSWADA 中的客户而言，服务定位即是如何获取对应的接待员，这正是名字服务的功能。为保证方便高效地查找对象，与一般名字服务不同，MSWADA 中设计的名字服务可以提供水平和垂直两个层次上的名字解析。水平层次的解析是为了适应内部域和外部域的划分，以及支持计算对象移动的需要。内部域和外部域各有一个自己的名字服务器，其中内部域中的名字服务器 INS(Internal Name Server)是基于内部现有系统，例如 CosNaming[6]或 JNDI[7]名字服务规范扩展实现的；而外部名字服务器 ENS(External Name Server)是 MSWADA 的一个关键组件，用于在整个分布系统中解析全局的 MSWADA 名字。INS 负责将输出的服务登记到 ENS 中，随后可以在整个分布系统中使用 MSWADA 名通过 ENS 定位这些服务。MSWADA 名字服务中垂直层次的名字解析分为两级，如图2所示。第一级采用树型结构，分为多个层级(hierarchy)，每个层级对应于一个 MSWADA 域，负责定位服务名到 ENS 一级。第二级对应于各个 ENS，负责将服务名解析到服务一级(实际是服务适配器)。

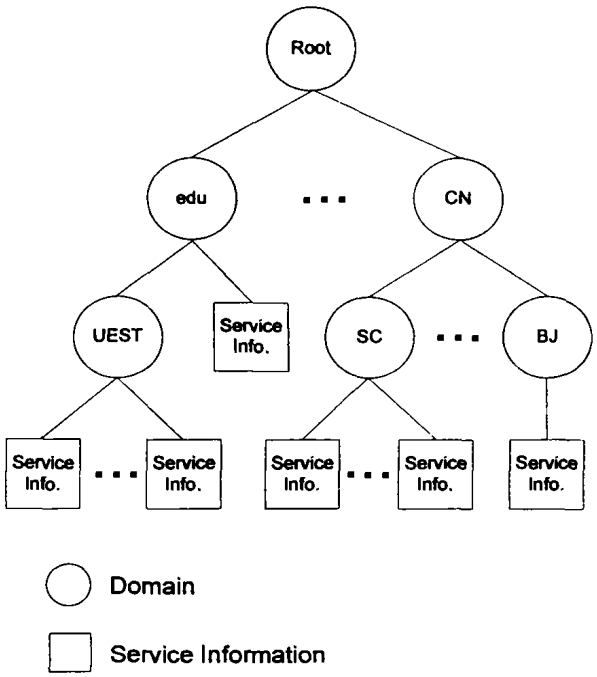

图2 MSWADA 的名字服务

由图可见，名字服务层级的划分可基于不同的策略，例如按照地域，类别，拓扑等进行。每个服务器可通过为维护指向不同层级的父/子指针参加多种层级。当然，需要有一个“基本”的层级以保证名字解析请求可以到达树的所有部分。MSWADA 名字层次中构造父子关系的机制可能包括各种利用外部信息的方法，例如 DNS 和拓扑数据等。

内部域需要访问的输入服务和提供给外部访问的输出服务信息存放在一张服务映射(Service Mapping)表中，INS 启动时读入该表内容，通知对应的 ENS 将内部域中的哪些内部服务输出并告知相应的 MSWADA 名字。由于此时这些服务尚未启动，因此在 ENS 中其状态均为“冻结”。ENS 与传统的名字服务器的另一区别在于，它综合采用了名字服务与交易服务[9]技术。ENS 不仅可按名字查找服务，还可指定服务的查找属性。如4.1节中将详细说明，这是为了支持多版本的服务实现和多种多样的客户。

4 设计面临的挑战与解决方案

作为广域分布系统的平台，MSWADA 必须满足适应性，可伸缩性，机动性，可靠性和安全性需求，本节讨论 MSWADA 如何满足适应性，可伸缩性和机动性需求，另外概述了 MSWADA 对可靠性和安全性的支持。

4.1 挑战1：使系统具有良好的适应性

4.1.1 与基于传统中间件的遗留系统集成 通常认为，要使分布式应用适应广域系统，就应该采用全新的体系结构构造。然而我们认为这一方法不适合许多已有大量遗留系统、与新技术不兼容的环境。

许多企业已经构造了大量基于传统中间件技术的关键业务系统，例如某些面向过程的联机事务处理系统。由于企业业务发展或者变革的需要，现在他们要求和其它企业进行协作。显然，仅仅为了使不同的企业系统能够互联以支持这些协作，就采用全新的面向对象中间件技术重构整个系统是不现实的。换句话说，企业面临的共同问题是，既要能引入新的服务和扩展老的服务到广域分布应用中，又要保证不会中断原有系统的稳定运行。为解决这一问题，MSWADA 的设计采用了代理机制，它引入服务插头和服务适配器以连接在不同内部域中的客户与服务。插头和适配器分别作为服务在客户方和服务方的代表共同屏蔽了采用不同技术的内部域系统中的客户和服务的差异，并使之对外部域透明。同样，INS 和 ENS 的划分使得用户可以灵活地选择多种技术实现其全局名字服务，而不受内部域遗留系统所采用的名字服务技术的影响。

采用 MSWADA 构筑分布系统所要做的全部工作如下：在外部域中设置 ENS 和接待员，再将内部域中的名字服务器替换为兼容的 INS，配置好所有需要访问的外部服务(输入服务)和提供给外部访问的服务(输出服务)。企业现有系统的所有客户/服务器应用无须进行任何修改就可以运行。名字服务器、接待员和服务适配器一起透明地屏蔽了本地服务和远程服务的差异。

4.1.2 适应技术和需求的变化 一般来说，一个大规模分布系统的生命周期应该很长。在此期间，系统必须具有足够的灵活性以适应新技术和新需求。例如，某个企业业务系统向外输出的服务升级(比如增加了一些新方法)了，从而要求客户方必须使用新的服务插头访问升级的服务，同时又要保证使用老版本服务插头的客户仍然能继续访问该服务。另一方面，使用服务的客户本身也是各种各样的，比如，某个客户可能要传输大量的数据交给服务处理，因此要求服务能够支持数据的压缩传输，而其他的客户可能就没有这个要求。

接待员，服务插头与服务适配器一起为 MSWADA 体系结构提供了一种关键特征，即动态服务适配，它可以适应技术和需求的变化。动态服务适配的思想是，同一服务可以有多个接待员，各自为不同的用户提供服务。他们通过不同的服务插头访问同一服务而不是直接访问服务，就可以支持服务的升级和满足不同客户的要求。适配器和服务插头都是由服务提供者生成的。因此他们之间的通信协议也对客户透明，不同的适配器与对应的插头可使用不同的协议进行通信。例如，如果服务方所在域的外防火墙只开放 HTTP 端口，则服务实现者可通过提供支持 HTTP 协议的服务插头和服务适配器解决这一问题，服务插头和服务适配器之间可直接使用 HTTP 通信，还可以通过 HTTP Tunnel 支持使用其它协议的通信。同理，只要服务插头的接口对接待员保持不变，其实现对接待员是透明的。在 MSWADA 中，服务插头可以被动态更新，即 ENS 在认为需要时可以自动从服务提供者所在的域中下载最新版本的服务插头实现。

4.1.3 MSWADA 中的服务调用过程 上面描述的技术可以从一次完整的服务调用过程中得到体现，如图3所示，内部域的服务器启动并向 INS 注册其服务时，INS 检查服务映射表，识别出该服务需要输出，并调用 ENS 的“bind”方法，将该服务输出为指定的全局名字。随后，ENS 为该服务生成一个对应的服务适配器对象并将服务和服务适配器的状态都标为“激活”。

图3中①-⑧示出了一次完整的名字解析过程：

①域 A 中的客户使用符合内部系统规范的符号名通过 INS 查找服务

②INS 读 Service Mapping(服务映射表)，识别出该名字对应的是外部服务，将该名字转化为一个 MSWADA 全局名字，并调用 ENS 的“resolve”方法解析该名字

③ENS 确定该名字属于域 B，并调用域 B 中 ENS 的 resolve 方法解析

④D_B ENS 判断出域 B 可以提供满足查找要求的服务，且该服务目前为激活状态。随后它检查是否有激活的服务适配器(可能已被冻结)，若无对应的激活服务适配器，则创建一个

⑤随后，ENS 按照③中的查询要求返回确认信息，包括服务适配器引用和服务插头信息

⑥D_A ENS 确定本地的服务插头实现版本是否满足要求。若否，则需要从域 B 中下载更新的插头实现版本到域 A 中。最后通知接待员用返回的服务适配器引用实例化一个服务插头对象。

⑦ENS 向 INS 返回确认信息证实名字解析已完成

⑧INS 向客户返回符合内部系统规范的服务代表(可能是接待员进程，也可能是接待员进程中的远程对象引用，取决于内部系统采用的技术)

对于许多 TP-Monitor，例如 BEA Tuxedo，服务实现不使用动态的名字服务，而是静态地绑定到一个固定的名字。这种情况下的服务调用过程稍有不同。现在接待员充当 INS 的角色，负责名字解析过程。接待员进程在启动时访问 ENS，获取输入服务的信息，并通过名字解析为这些服务生成相应的服务插头。

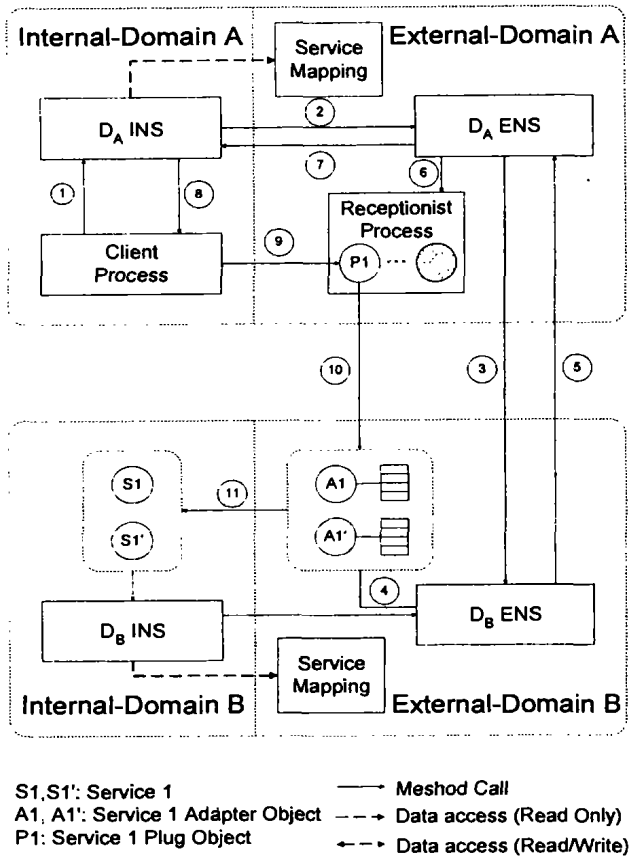

图3 MSWADA 中一次完整的服务调用过程

具体的服务调用过程则如图3中⑨-⑪所示。

⑨域 A 中的客户通过符合内部系统规范的调用方式访问接待员

⑩接待员将该调用转换为服务插头识别的格式并通过服务插头将请求发送给域 B 中的服务适配器

⑪服务适配器收到请求后，调用域 B 内部系统中对应的服务完成请求，并将结果返回给服务插头，插头再将结果送给接待员。最后接待员再将结果转换为域 A 现有系统的格式返回

4.2 挑战2：实现可伸缩的系统

MSWADA 的设计考虑了对可伸缩性的支持，基于 MSWADA 的广域分布系统既能满足用户的要求，又不至于消耗过多的系统资源。对可伸缩性的支持集中体现在 MSWADA 的名字服务和服务访问方面。

4.2.1 *可伸缩的名字服务* MSWADA 中的名字服务是可伸缩的。虽然采用集中式名字服务器构造名字服务体系是最直接的方式，然而一般来说，集中式名字服务器缺乏可伸缩性。对于广域分布系统来说，它很容易使 CPU、网络过载工作，并占用大量的内存，因为整个广域分布系统中的对象数量通常是惊人的。因此，MSWADA 没有采用集中式名字服务器，而是将名字服务按层级组织成许多个域。由于每个 ENS 只管理相应域的名字空间，从而避免了 ENS 占用大量的计算机与网络资源。MSWADA 的设计支持将已有的名字空间，例如 DNS[8]和 X.500等合并到一个单一全局名字空间中，而不是创建一个全新的名字空间，因为创建新的名字空间的非技术因素很复杂。事实上，由于 MSWADA 对适应性的支持，它能够与现有的主要名字系统共存。

MSWADA 的名字服务还可支持负载均衡，同一服务的多个服务适配器可以绑定到 ENS 中的同一名字，当解析该名字时，返回的实际上是对多个服务适配器的引用。接待员可根据这些引用创建服务插头。当调用该服务时，服务插头会根据负载均衡策略从中选择一个引用发出对相应服务适配器的调用。

为防止高层级中的名字服务器被大量的查找请求压垮，ENS 解析名字时只允许使用重复查询(recursively lookup)方式，不允许使用递归查找(iteratively lookup)方式。MSWADA 还使用缓存技术来提高名字服务的整体性能。每次 ENS 成功解析一个服务名时，都会将对应的服务名-服务适配器引用对保存到自己的缓存中，当本地域中的其它客户再次要求 ENS 解析该服务时，响应速度就大大提高了。缓存中每个映射对的生命周期由时间戳控制。

4.2.2 *可伸缩的服务* MSWADA 中的服务访问也是可伸缩的。

服务插头第一次向服务适配器请求服务时，需要建立连接。建立连接的开销是比较大的，为加快随后的服务调用过程，该连接可以保活(Keep Alive)，这样再次调用服务时就省去了建立连接的开销。

服务适配器的设计可以保证在有大量客户访问同一服务的高峰时刻，系统的性能不会明显下降。系统为每个服务适配器对象分配自己的请求队列。当大量请求到来时，服务适配器只向对应的服务提交一定数目的请求，剩下的请求则缓存在请求队列中，仅当服务处理完先前的请求并返回应答到应答队列后，队列中缓存的请求才会继续发送。因此，即使在重载时服务也保持健壮性，它们可以平滑地处理负载超过服务处理容量的大量突发请求。

在广域分布系统中，某一高峰时刻可能有多个客户并发地访问同一服务。如图1所示，为了提高性能，MSWADA 中的服务支持负载均衡。支持多线程的服务适配器可以并发将请求调度到这些服务实例中。相应地，同一服务可以有多个服务适配器对象，与服务一起配合支持负载均衡，如3.1节所述。服务输出时可通过策略定义为有状态或无状态服务。无状态服务与服务适配器之间是一种多对多的关系；如果为有状态服务，则需内部域现有系统本身支持服务的复制才能进行负载均衡，否则服务适配器与服务只能是一一对应关系。

服务适配器对象本身也有生命周期，超时即被系统回收，以防浪费系统资源。相应地会产生一个问题：如果这时客户发来请求，则会引发例外。解决这一问题的方法的原理与下节中计算对象移动时的原理相同。

4.3 挑战3：支持计算对象移动

与支持移动计算(Mobile Computing)的系统不同，MSWADA 着重于对计算对象移动(Mobile Computations)[10]的支持。计算对象移动是指用户基本不移动，相反对象由于负载均衡、动态复制等的考虑时常移动。然而，传统中间件对于计算对象移动的支持往往仅限于局域网中的分布系统，缺乏对跨越广域网的大规模分布系统的支持。MSWADA 使用基于宿主的方法来支持计算对象移动，这意味着总是有一个宿主位置负责跟踪服务的当前位置。但 MSWADA 避免了这一方法在大多数系统中的缺点：缺乏可伸缩性，因此可支持广域分布系统中的计算对象移动。

图4说明 MSWADA 如何支持计算对象移动。为简单起见，图中略去了对负载均衡的处理，因为它和用于实现服务的

内部域系统采用的技术相关。图中①-⑤示出了一个完整的服务迁移过程：

①D_B(域 B)中的服务 S1释放它在 D_B 中占用的资源，包括 A1，随后它迁移到 D_C(域 C)中，称之为 S1'

②S1' 向 D_C INS(即域 C 中的 INS)注册

③D_C INS 识别出该服务是迁移的服务，通知 D_C ENS 有一个新的活动迁移服务

④D_C ENS 为该服务创建对应的服务适配器对象 A1'，并登记其状态为"Active"(随后若 D_C 中有客户要访问该服务，则无须再访问域 B)

⑤域 C 中的 ENS 通知服务的宿主 ENS(域 B 的 ENS)其当前位置，后者将该服务状态标为 Relocated，并登记其服务适配器为 A1'

域 A 中对已迁移服务的调用过程如图4中⑥-⑪步：

⑥客户向接待员发出服务请求

⑦接待员使用对应的服务插头 P1访问服务适配器 A1，由于 A1已不存在，该调用引发"服务不存在"例外

⑧接待员捕获该例外，并调用 D_A ENS 的 relocate 方法，要求重定位 P1

⑨D_A ENS 检查 P1对应的服务目前未迁移到 D_A 域，然后使用 P1对应的服务名访问服务所在的宿主域 ENS，即 D_B ENS，D_B ENS 返回该服务目前对应的服务适配器 A1' 的引用给 D_A ENS，后者再将该引用返回给接待员

⑩接待员根据返回的 A1' 引用将服务插头更新为 P1'

⑪接待员使用 P1' 重新发出服务调用，A1' 接收到该调用请求，并调用 S1' 完成请求

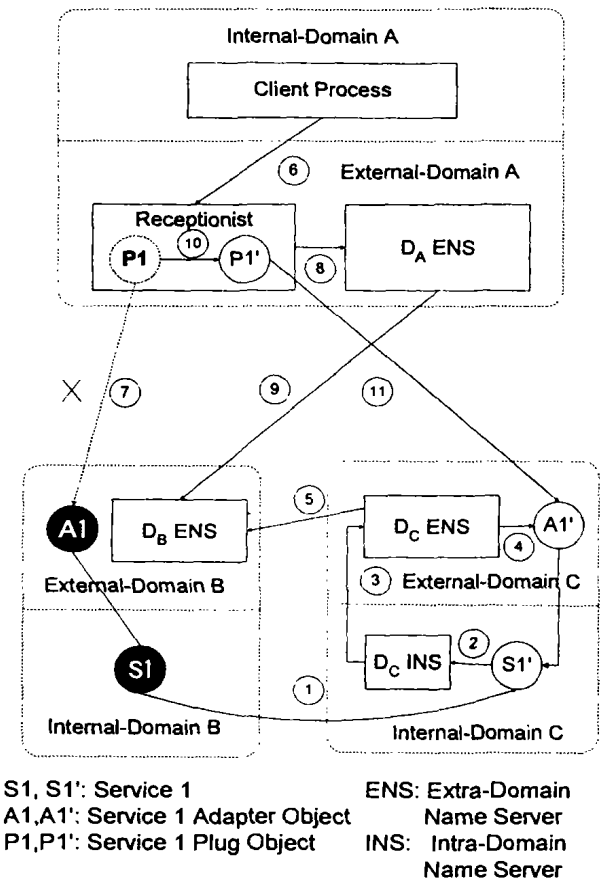

图4 MSWADA 中的计算对象移动

一次重定位完成后，只要服务没有再迁移，则随后同一域内的所有客户调用该服务不再需要重定位，而是仅经过图4中第6步和第11步就完成正常调用。因此，MSWADA 中对计算对象移动的支持不会对性能有明显的损害。

为简单起见，图中略去了对负载均衡的处理。实际上，如3.1节所述，同一服务可能同时有多个对应的实例在运行(参见图1)。在这种情况下，D_B ENS 中登记的服务实际上表现为一个名字对所有适配器对象引用的映射，当其中的一些实例迁移到其它域时，原所在域中对应的服务适配器与它的绑定关系中断(适配器如何与服务实例"绑定"与内部域现有系统采用的技术有关)，取而代之的是在服务实例迁移到的新域中创建新的服务适配器并将它与迁移后的服务实例绑定。相应地，D_B ENS 对应映射的引用部分中增加对新建的服务适配器的引用，而其它引用保持不变。当服务插头选择一个失效的引用访问服务适配器时(此时该服务适配器绑定的所有服务实例都已迁移，否则它必定存在，因而不会导致异常)，导致异常，发生和图4中类似的重定位过程，导致该失效的引用被更新。由此可见，MSWADA 中的负载均衡实际上可以支持同一服务的不同实例分布在不同域中运行。

以上讨论的情况仅针对某些服务临时迁移到其他域中时 MSWADA 如何处理。如果服务永久地迁移到另一个域，则服务原来所在域中的 ENS 会将该服务状态标为永久迁移(Emigrated)。客户通过先前获得的无效服务插头(引用)第一次调用服务时，重定位过程同样会发生，类似于图4中①-⑨的处理过程。区别在于，第⑨步中宿主域的 ENS 不仅会返回新的服务适配器引用，还会通知客户域中的 ENS，该服务已永久迁移并返回新的 MSWADA 名。随后在第⑩步中，接待员会通知客户域的 INS 将服务映射表中对应服务的名字永久更改为新名字。服务原宿主域的 ENS 会将对应信息保留一段时间以便所有相关的其它域都更新此信息。

4.4 增强可靠性与确保安全性概述

当前，主流的中间件平台基本上都采用"多客户-单服务器"模式，使服务器成为系统的单故障点。MSWADA 解决这一问题的方法是采用复制技术支持多个服务器，包括 ENS 和服务适配器。由于 MSWADA 对适应性的支持，必须保证客户访问接待员和 INS 透明，即接待员和 INS 同内部域现有系统提供的服务一样。因此接待员和 INS 是否支持复制技术取决于内部域系统采用的技术。

通过代理模式的设计，MSWADA 可以方便地支持在内部域与外部域之间设置内防火墙以保证用户内部关键业务系统的安全。MSWADA 客户只能访问输出到外部域的服务。而对于内部域向外输出的服务，MSWADA 则可通过身份认证、数据保密性、数据完整性和授权等多种安全机制保证其安全。如果服务要求保证安全，则服务插头在调用服务前必须首先与服务适配器建立一个安全信道进行通信。MSWADA 通过 ACL(Access Control List，即访问控制表)支持授权机制。要访问具有 ACL 的服务，经认证的用户名字必须显式地出现在 ACL 中。

在 MSWADA 中，可靠性和安全性的支持对于现有系统中的客户和服务实现是透明的，因此不会损坏系统对于适应性的支持。

5 相关工作的比较

传统的中间件，例如 DCE，TP-Monitor，CORBA，DCOM 和 J2EE 等，可以在小规模网络中支持分布透明性。然而，这

些中间件解决方案中的每一种都有自己的体系结构和通信协议，从而使得基于它们的分布系统只能选择一种体系结构。换句话说，系统缺乏适应性。而且，这些系统中的客户代理有共同的特性：转发请求和处理应答，因而很难支持机动性。

近来很多研究是关于利用 Internet 的计算资源进行分布计算的，包括 Legion[11]、Globus[12]和 Globe[13]。Legion 基于一个分布式的反省对象模型构造，着重于从头开发一个新的对象模型。构建 Legion 应用必须使用它自己的库。Globus 项目提供了一个"计算网格"用以将异种的分布资源集成到一个单一的广域系统中。它是为诸如大规模仿真这样的计算密集型应用量身定做的。Globe 则是一个构造为 Internet 上面的中间件层的广域分布系统。

尽管上面提到的系统中有些是可以定制的，但它们都没有将适应性作为一个基本需求。另外的工作，例如 Web Service[14]，可以在一定程度上支持适应性，然而缺乏对计算对象移动的支持。我们的工作基于与已有的成果和正在进行的研究类似的技术，但我们在分布对象技术，计算对象移动和软件体系结构方面引入了不同的特性。这些研究中应用的很多技术可以很容易地添加到 MSWADA 的体系结构中以提高其性能。

结论与未来的工作 MSWADA 在提供适合大规模分布系统的基础设施方面迈出了重要的一步。我们的工作的主要贡献在于它可以在同一个体系结构中支持适应性、可伸缩性、机动性、可靠性和安全性。我们介绍了 MSWADA 体系结构的对象模型与名字服务。MSWADA 中有5个关键组件：INS，ENS、接待员，服务插头与服务适配器。文中讨论了这些组件如何协作，以便在有多个遗留系统的环境中支持广域分布式应用。MSWADA 支持在一个高度动态，且具有许多遗留系统的非集中式网络中透明、高效地定位和访问服务。其代理模式的使用将内部域遗留系统中的客户和服务器实现与外部域中的 MSWADA 组件隔离开来。因此，基于 MSWADA 的分布系统可以很容易适应遗留系统并提供对机动性的支持。可伸缩性则是通过 MSWADA 体系设计中的多层次名字服务和服务适配器一起配合实现的。我们按照 MSWADA 体系结构实现了一个原型系统，以证明其可行性。ENS，接待员、服务适配器和服务插头选择用 Java 实现。Java 的特性对于构造一个满足适应性、可伸缩性、机动性、可靠性和安全性需求的原型实现是大有裨益的。该原型中 ENS 层级划分基于现有的 DNS 名字系统定位到域一级，而 ENS 本身负责解析本域中的服务名。服务插头与服务适配器之间使用 RMI/IIOP 协议进行通信。服务插头的自动更新/升级则是通过 Java JNLP 协议实现的。目前的原型实现支持两种类型的遗留系统。一种基于 DCOM 实现，而另一种基于 BEA Tuxedo。

在今后的工作中我们将努力使 MSWADA 及其原型实现更加成熟。下一阶段的工作包括：增加对持久性机制的支持，扩展文中描述的名字服务，完善 MSWADA 的原型实现以提供 GUI 工具对名字服务和安全配置进行管理，支持使用 SOAP[15]协议作为服务插头与服务适配器之间的通信协议。

参 考 文 献

1 Rosenberry W, Kenney D, Fisher G. Understanding DCE. O'Reilly & Associates, Sebastopol, CA. , 1992

2 BEA Systems. http://e-docs. bea. com/tux71/

3 Object Management Group. The Common Object Request Broker: Architecture and Specification, Revision 2. 5. OMG Document 01-09-01, Object Management Group, Sep. 2001

4 Eddon G, Eddon H. Inside Distributed COM, Microsoft Press, Redmond, WA, 1998

5 Sun Microsystems, Java™2 Platform Enterprise Edition Specification, V1. 3. http://java. sun. com/j2ee/. Jul. 2001

6 Object Management Group. Naming Service Specification. OMG Document 01-02-65, Object Management Group, Feb. 2001

7 Sun Microsystems, Java™ Naming and Directory Interface Application Programming Interface, V1. 2. http://java. sun. com/ Jul. 1999

8 Albitz P, Liu C. DNS and BIND. O'reilly & Associates, Sebastopol, CA. , 3rd edition, 1998

9 Object Management Group. Trading Object Service Specification V1. 0. OMG Document 00-06-27, Object Management Group, Jun. 2000

10 Caughey S, Shrivastava S. Architectural Support for Mobile Objects in Large-Scale Distributed Systems. In: L. F. Cabrera and M. Theimer, eds. Proc. Fourth Intl. Workshop on Object Orientation in Operating System, Lund, Sweden, IEEE, Aug. 1995. 38～47

11 Grimshaw A, Wulf W, et al. The Legion Vision of a World-wide Virtual Computer. Communications of the ACM, 1997, 40(1)

12 Foster I, Kesselman C. Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Applications, 1997, 11(2): 115～128

13 van Steen M, Homburg P, Tanenbaum A. The Architectural Design of Globe: A Wide-Area Distributed System. Scheduled for publication in IEEE Concurrency, 1999

14 IBM Software Group. Web Services Conceptual Architecture (WSCA 1. 0). http://WWW-106. ibm. com/developerworks/library/w-ovr/. May 2001

15 W3C, SOAP Version 1. 2, W3C Working Draft 9 July 2001. http://WWW. w3. org/TR/2001/WD-soap12-20010709/. Jul. 2001

16 Othman O, O'Ryan C, Schmidt D C. An Efficient Adaptive Load Balancing Service for CORBA. IEEE Distributed Systems Online, 2001, 2

17 Schmidt D C, et al. Pattern-Oriented Software Architecture: Patterns for Concurrency and Distributed Objects, Volume 2. New York, NY: wiley & Sons, 2000

18 Sun Microsystems, Jini technology specifications. White paper. http://WWW. sun. com/jini/specs

19 Matsuoka S, et al. OpenJIT: a reflective Java JIT compiler. In: Proc. of OOPSLA'98, Workshop on Reflective Programming in C++ and Java, 1998. http://openjit. is. titech. ac. jp/

20 Grimshaw A, et al. Campus-Wide Computing: Results Using Legion at the University of Virginia: [Technical Report CS-95-19]. University of Virginia, March 1995

21 van Steen M, et al. Algorithmic Design of the Globe Wide-Area Location Service. The Computer Journal, 1998, 41(5): 297～310

22 The Ninja Team. The Ninja Project. http://ninja. cs. berkeley. edu

23 Czerwinski S, et al. An architecture for a secure service discovery service. In: Proc. of Mobicom'99, ACM, Seattle, WA, Aug. 1999

24 Fugetta A, Picco G P, Vigna G. Understanding Code Mobility. IEEE Trans. Softw. Eng. , 1998, 24(5): 342～361

25 Kon F, et al. Secure Dynamic Reconfiguration of Scalable CORBA Systems with Mobile Agents. In: Proc. of the IEEE Joint Symposium on Agent Systems and Applications/Mobile Agents (ASA/MA'2000), Zurich, Sep. 2000

26 Gamma E, et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addision-Wesley, Reading, Mass. , 1994

27 Zenel B, Duchmap D. A general purpose proxy filtering mechanism applied to the mobile environment. In: Proc. of the third Annual ACM/IEEE Conference on Mobile Computing and Networking (Mobicom'97), ACM, New York, USA, 1997

28 Qian T. Dynamic Authorization Support in Large Distributed Systems: [PhD thesis]. Department of Computer Science, University of Illinois at urbana-Champaign, Nov. 1999

29 Sandhu R S, et al. Role-based Access Control Models. IEEE Computer, 1996, 29(2): 38～47