计算机科学2000Vol.27№.5

基于CORBA的移动Agent技术研究*)

Study of CORBA-Based Mobile Agent Technology

胡 健 刘锦德

(电子科技大学计算机科学与工程学院 成都610054)

Abstract In the first ,a common conceptual model for mobile agent is introduced. According to this model,the related CORBA object services are discussed and a mobile agent facility interface definition is given.

Keywords CORBA,Mobile agent,Open system,Interoperability

1 引言

移动agent技术为分布式计算提供了一种富有活力的新的计算模式,其优点已广泛地为人们所认识[1,14],成为目前DAI和分布式计算领域的一个研究热点。研究移动agent技术产生的技术背景可以追溯到三种更早的技术:进程迁移,远程求值和移动对象。尽管很多移动agent系统的发展都经历了这样的技术演进过程,但由于移动agent是一门相对较新的技术,使得目前的agent系统(例如Crystaliz的MuBot,Darmouth College的AgentTcl,IBM的Aglets,Open Group的MOA,GMD FOKUS的JMAF/Magna,General Magic的Odyssey)之间在体系结构和具体实现上存在着非常大的差异。移动agent系统间的这种差异,一方面阻碍着各agent系统之间互操作性的实现,另一方面也严重制约着agent技术自身的发展。我们知道,在目前的网络计算环境中,一个系统若不具备开放系统[10]的特征,就很难具有生命力,而一个较为理想的开放系统,必须解决互操作问题,后者可以依靠定制的或现成的可互操作中间件来实现。就互操作中间件而言,当前值得注意的有关技术是:OSF的DCE,ISO和ITU的ODP和OMG的CORBA。由于现有的agent系统许多都是基于Java实现的,而CORBA已很好地完成了与Java和Web的集成,并且一个基于CORBA的移动agent系统的开放系统轮廓正在制定和完善中,这便是MAF(Mobile Agnet Facility)规范。由于它由移动agent系统的一些主要实现者(如Crystaliz, General Magic, GMD FOKUS, IBM, Open Group)共同提出,并受到OMG的高度重视,因而,我们认为该规范必将对移动agent技术的发展产生深远的影响。

2 移动Agent通用概念模型

目前的移动agent系统间存在很大的差异,但是通过研究其在agent交互、agent传送和安全性等方面的共性,可以为一个通用概念模型的提出提供基础,同时也有利于互操作标准化工作的进行和现有的移动agent系统的完善。这里首先给出移动agent技术有关的概念和术语,以便对移动agent技术有一个共同的认识并明确问题域。

2.1 基本概念

这里,对移动agent的有关的概念的描述采用了面向对象的术语,对于非面向对象的移动agent系统,在有关的概念定义中可以用代码(code)代替定义中所用到的类。

·**Agent** 代表个人或组织的可自主行动的计算机程序。

·**非移动Agent** 只能在开始其执行的系统上执行的agent。

·**移动Agent** 可以离开其开始执行的系统,在网络上不同系统间移动的agent。本文中若非特别指明,agent都指可移动的agent。

·**Agent状态** agent移动时,要将它的状态与代码一起移动。这里的状态既可以是它的执行状态也可以是agent的属性值。执行状态是指它的运行时状态,包括它的程序计数器和页框堆栈。agent的属性值是指

*)本课题得到国防预研基金的资助。胡 健 博士生,主要研究方向:分布式对象技术,分布式人工智能和多媒体技术;刘锦德 博士生导师,主要研究方向:开放式系统技术,分布式多媒体技术等。

当它在目的地恢复执行时用于帮助其确定要做什么的有关信息。

·**Agent 职权** 用于确定它所代表的个人或组织。Agent 的职权必须是可以进行认证的。

·**Agent 名字** 用以识别，名字应该是全局唯一和不变的。

·**Agent 位置** 是指其所在场所的地址，场所位于一个 agent 系统之内，因而 agent 的位置应包括 agent 所在的 agent 系统的网络地址（例如 IP 地址和端口号）以及一个场所名。

·**Agent 系统** 是能够创建、解释、执行、传送和终止 agent 的一个平台。与 agent 的职权一样，agent 系统的职权用于确定它所代表的个人或组织。Agent 系统通过它的名字和地址进行识别。一台主机可以包括一个或多个 agent 系统，一个 agent 系统可以包括一个或多个场所，一个场所又可以包括一个或多个 agent。

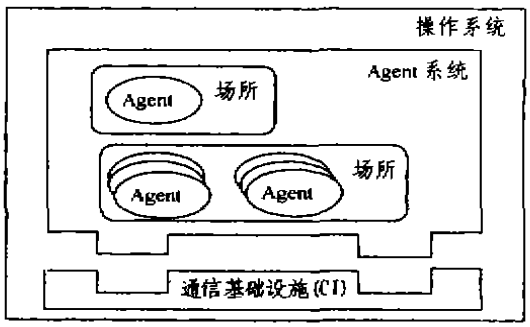

图1 Agent 系统示意图

·**Agent 系统类型** 描述了一个 agent 所拥有的能力。例如，如果 Agent 系统类型为 Aglet，这表明它由 IBM 实现，以 Java 作为 agent 语言，使用 Java ArchiveFormat 作为它的序列化方法。客户方在向 Agent 系统请求服务时必须指明 Agent 系统的类型。

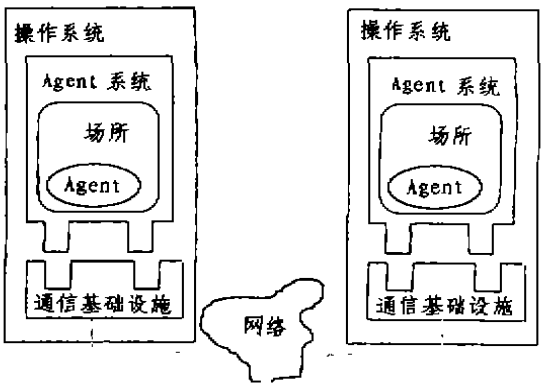

图2 Agent 系统互连示意

·**Agent 系统与 Agent 系统互连** Agent 系统间的所有通讯都是通过下部的 CI（通信基础设施）进行。区域管理器可以提供区域内和区域间通信的不同服务。

·**场所** 是位于一个 agent 系统之内的供 agent 执行的环境上下文（Context），它可以提供诸如访问控制这样的功能。一个 agent 系统可以包含多个场所。

·**区域** 由一组具有相同职权的 agent 系统组成，但这些 agent 系统的类型不一定相同。引入区域的概念使得可用多个 agent 系统来代表一个人或组织。它还可以提供可伸缩性，因为你可以在多个 agent 系统间平衡负载。

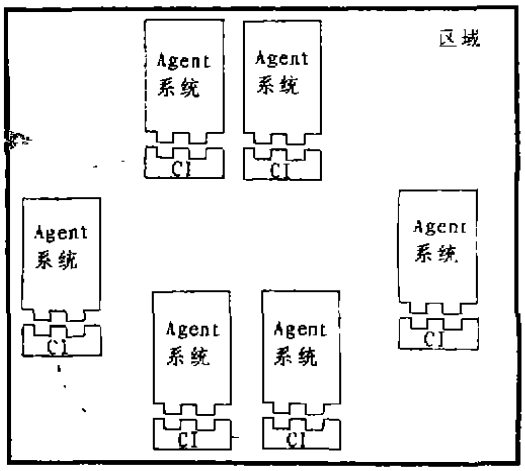

图3 区域结构示意

区域为区域外的与之通信的客户方提供了一个抽象层。一个访问 agent 或 agent 系统的客户可以不必知道 agent 的地址，它只需知道区域的地址（通常为被指定为区域访问控制点的某 agent 系统的地址）以及 agent 的名字。同一区域中的不同 agent 的权限是不同的，与区域具有相同职权的 agent 可以被赋予管理权限。一个区域允许有多个访问控制点。

·**区域与区域互连** 与防火墙的情形类似，区域通过向外提供访问控制点，接受来自区域外的访问，访问者既可以来自其它区域中的 agent 系统，也可以来自非 agent 系统。当然，访问者必须具有合法的权限。

·**序列化与去序列化** 保存和重新获得 agent 的关键是用一种可重构 agent 的序列化的格式来表示 agent 的状态。所采用的序列化的格式一定要能够被 agent 系统识别。序列化是将 agent 保存为序列化的格式的过程；去序列化是从序列化的格式重构 agent 的

过程。

·**通信基础设施(CI)** 负责为 agent 系统提供 RPC,名字和安全服务。

2.2 Agent 交互作用

与互操作性密切相关的 agent 交互作用有以下三类(注意,只有 agent 传送是特别针对移动 agent 的):

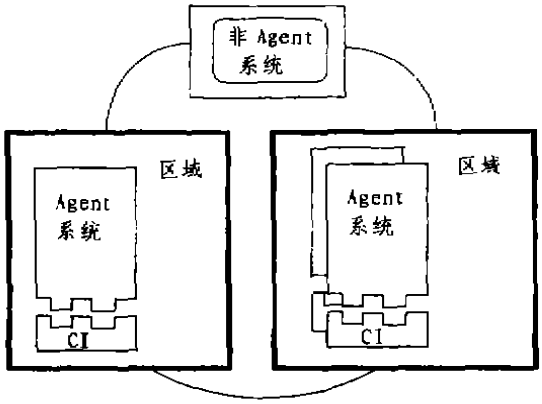

图4 区域互连示意图

1)*远程 agent 创建* 远程 agent 创建时,客户方程序请求 agent 系统创建特定类的 agent。客户方程序可以是与远程 agent 系统不同类型系统中的 agent。此外,agent 系统也可以根据自己的意愿创建 agent。当然,这两种情况下所创建的 agent,其职权是不同的。

2)*agent 传送* 当一个 agent 想传送到别的 agent 系统中时,它所在的 agent 系统将为它创建传送请求。作为传送请求的一部分,该 agent 必须提供用于识别目的场所的名字或地址信息。此外,还可以在传送请求中指明所要求的通信服务质量。

如果源 agent 系统将传送请求送到了目的 agent 系统,目的 agent 系统必须满足传送请求或向 agent 返回请求失败的指示。如果目的 agent 系统不可达,源 agent 系统必须向 agent 返回失败指示。

当目的 agent 系统同意进行传送时,agent 的状态、职权、安全凭证以及 agent 的代码都可以传到目的 agent 系统中。目的 agent 系统负责重新激活并继续 agent 的执行。

3)*agent 方法调用* agent 可以调用其它 agent 或别的对象的方法,agent 系统的 CI 负责方法调用的实现。这一点,一般的分布式对象系统都提供支持。

2.3 Agent 系统的功能

作为一个功能完备的 agent 系统应具备以下的功能:

·**传送 Agent** 包括发起 agent 传送、接收 agent 传送和类(class)传送三个方面的内容。

·**创建 Agent** 为创建一个 agent,agent 系统应该完成以下工作:为 agent 启动一个线程;实例化 agent 类(class);为 agent 产生和指定一个可认证的全局唯一的名字;在线程内开始 agent 的执行。

·**提供全局唯一的名字和位置** agent 系统必须能够为它自己和它所创建的 agent 与场所提供全局唯一的名字。

·**支持区域的概念** 职权相同的 agent 系统,也就是说属于同一区域的 agent 系统之间应相互协作。区域访问控制点的思想也应得到支持。

·**寻找移动 Agent** 当一个 agent 想与另一个 agent 通信时,它必须首先能够找到目标 agent 以便建立通信联系。

·**保证一个进行 Agent 操作的安全环境** 移动 agent 可以在不同的 agent 系统之间移动,可能会带来各种安全隐患,agent 系统必须对 agent 系统自身和其它资源提供保护。

2.4 Agent 系统互操作性

一个开放的系统应该具备可互操作性,为资源共享提供最佳途径。不同的系统之间能否互操作取决于彼此之间是否具有相应的协作机制,也就是必须通过标准化的接口才能达到。不同的 agent 系统之间的互操作性也是 agent 技术所追求的一个重要目标,通过互操作可以大大地提高分布式计算的能力。为保证移动 agent 系统之间的互操作性,应该对移动 agent 技术的以下几个方面进行标准化:agent 管理;agent 传送;agent 和 agent 系统的命名;agent 系统类型和位置语法;语言透明性。

agent 之间的通信是进行互操作的基础,可以利用 CORBA 的对象通信机制来实现,因而不属于 MAF 规范的内容。由于语言透明性(例如 Java 与 Tcl 之间)相当复杂,需要在不同类型的 agent 系统之间标准化的互操作桥来实现,所以有待于进一步研究。表1对 MAF 规范所涉及的互操作性功能类型、各 agent 系统对它们提供支持的复杂程度进行了归纳:

表1 MAF 互操作性概要

互操作性功能类型	是否涉及	复杂程度
agent 管理	是	比较简单
agent 系统类型和位置语法	是	比较简单
agent 和 agent 系统命名	是	比较简单
agent 传送	是	复杂
语言透明性	否	非常复杂

3 与移动 Agent 技术有关的 CORBA 服务

在 OMA 参考模型中,CORBA 对象服务和 COR-

BA 公共设施都是其组成部分之一(如图5所示)。与移动 agent 技术有关的 CORBA 对象服务有以下几种:

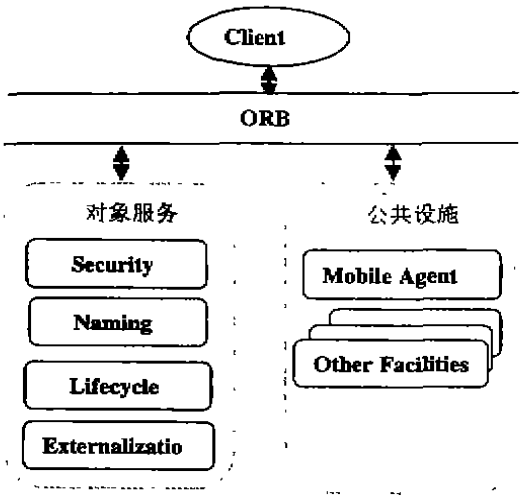

图5 CORBA 对象服务和公共设施

·名字服务 CORBA 名字服务将名字(表示为字符串)与 CORBA 对象绑定。应用程序可以利用该服务来公布命名对象,或通过对象名来查找对象引用。MAF 定义了两种 CORBA 对象接口:MAFAgentSystem 和 MAFFinder,其对象可以通过名字服务来公布,这样可以带来一些编程上的方便。例如,当 agent 进入某区域后可以利用名字服务查找 MAFFinder 对象的引用,以便进一步使用 MAFFinder 对象提供的服务。

·生命周期服务 用于创建、删除、拷贝静态的和被动的 CORBA 对象,但不能用于象 agent 这样的可移动的和主动的对象。但是 agent 系统则是一个可由 CORBA 生命周期服务进行创建或删除的静态对象的例子。区域(region)管理器可以利用这一特点,增加区域中 agent 系统的数量以处理增加的 agent 负载。

·序列化服务 提供了一种将对象状态记录为数据流或从数据流恢复为对象状态的标准化的机制。agent 系统可以选择该服务来对 agent 的状态进行序列化和去序列化。当然,agent 系统也可以采用非 CORBA 的序列化机制(比如 Java 的对象序列化方法)。

·安全服务 移动 agent 是可以在 agent 系统之间移动的计算机程序,这对 agent 系统的安全性提出了很高的要求。利用 CORBA 的安全服务,可以向 agent 系统提供以下的安全支持:远程 agent 创建时的客户方认证;agent 系统间的相互认证;agent 认证和授权(delegation);信息的完整性、保密性及重放检测。

4 MAF 接口定义

上面,我们对基于 CORBA 的移动 agent 技术的概念模型和它与 CORBA 的关系进行了分析,以上内容构成了 MAF 接口定义的基础。

MAF 由一组定义和接口组成,通过它们可保证移动 agent 系统间的互操作性。考虑到移动 agent 技术将来的发展,MAF 规范尽量以简单和通用为其原则。

MAF 模块包括两个接口:MAFAgentSystem 接口和 MAFFinder 接口。前者定义了对 agent 的有关操作,包括 agent 的接收、创建、暂停和终止等操作,MAFFinder 接口定义了 agent 登录、去登录以及 agent、场所和 agent 系统定位的有关操作,它的作用相当于是 agent、场所和 agent 系统的名字和位置的动态数据库。MAF 模块与 ORB 的关系如图6所示。

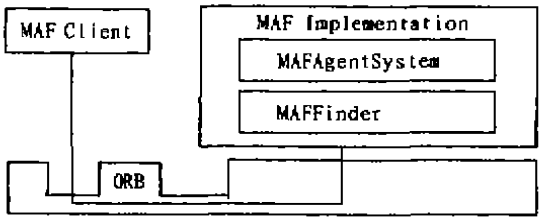

图6 MAF 与 ORB 的关系

出于互操作性的考虑,MAF 是在 agent 系统层上定义接口的。agent 系统和 agent 都可以是 CORBA 对象,但非必须是。由于 agent 居于 agent 系统之内,所以它的具体实现要依赖于创建它的 agent 系统的具体实现。

下面简单给出两个接口的 IDL 定义,为节省篇幅,省去了有关的参数说明。

```
Interface MAFAgentSystem{
    creat_agent();
    receive_agent();
    resume_agent();
    suspend_agent();
    terminate_agent();
    terminate_agent_system();
    fetch_class();
    find_nearby_agent_system_of_profile();
    get_agent_status();
    get_agent_system_info();
    get_authinfo();
    get_MAFFinder();
    list_all_agents();
    list_all_agents_of_authority();
    list_all_place();
    };
interface MAFFinder{
    register_agent();
    register_agent_system();
    register_place();
    lookup_agent();
    lookup_agent_system();
    lookup_place();
```

```
unregister_agent();
unregister_agent_system();
unregister_place();
};
```

参 考 文 献

1 Lange D B, Oshima M. Seven good reasons for mobile agents. CACM, 1999, 42(3): 88～89

2 Wong D, et al Java-based Mobile Agents. Some to[1], 92～102

3 Shoham Y. Agent-oriented programming Artificial Intelligence, 1993, 60: 51～91

4 Nardi B A, Miller J R Collaborative, Programmable Intelligent Agents. CACM, 1998, 41(3): 96～104

5 Shehory O, Kraus S. Methods for task allocation via agent coaliton formation. Artificial Intelligence, 1998, 101: 165～200

6 OMG. CORBAservices. Common Object Services Specification. Nov. 1997

7 OMG. The Common Object Request Broker: Architecture and Specification for CORBA V2. 2. Feb. 1998

8 Vinoski S. New features for CORBA 3. 0. CACM, 1998, 41(10): 44～52

9 Crystaliz, Inc. General Magic, Inc IBM Corporation, et al. Joint Submission. Mobile Agent Facility Specification. June 1997

10 刘锦德. 对于开放系统内涵的澄清. 计算机应用, 1997, 17(6): 1～3

11 苏 森, 唐雪飞 开放系统中的互操作性. 计算机应用, 1997, 17(6): 4～7

12 苏 森. 面向对象的互操作技术. [电子科技大学博士论文]. 1998

13 陈 羽中, 麦中凡. 可移动的软件 Agent 研究. 计算机科学, 1998, 25(5): 62～66

14 陶先平, 吕建, 等. 流动 agent: 一种未来的分布计算模式. 计算机科学, 1999, 26(2): 1～4

(上接第29页)

为。用 IDL 为每个组件定义对象包装接口，并在接口处提供一组操作去映射组件提供的方法名。在接口处提供的操作通常是简单的数据访问，当获知组件对象中的某些数据发生变化时，调度其他组件对象进行工作。

·规则定制。这是最灵活的一种方法。组件中的每个方法都代表了组件的功能行为。我们可以引入主动数据库中的 ECA 规则：在一个组件中发生的事件被其他组件所关注，当一个事件发生时，该组件发送(Post)另一个事件来通知其他组件。被发送的事件触发一组规则，当规则的条件满足时执行相应的一组动作。事件与事件触发规则可以被用于定制代码，从而有效地定制组件的行为。此时，事件是钩链(hook)，而规则是必要的代码。基本上，事件和事件触发规则提供了一种能够添加和替换代码的机制，在某个方法的实现中加入或替换部分代码而不必访问源代码。因此，规则定制是最合适的一种方法。

结束语 综上所述，虚拟企业环境下分布式对象的集成问题的核心是如何调度各组件系统协同工作以满足不同企业的企业逻辑。本文在回顾了集成系统发展过程的基础上，总结了分布式组件的几种动态调度结构，对其进行了全面的比较，并提出了在非确定型调度结构中的组件定制方法。

参 考 文 献

1 Ozkarahan E. Database management concepts, design and practice, Chapter 10. Prentice Hall, 1990

2 Yang X, et al. Study on a CORBA-based Plug and Play Mechanism for Distributed Information Sharing Environments: [SCOPE Technical Report]. Dept. CS, Northeastern University, Sept. 1998

3 Noll J, Scacchi W. Supporting Software Development in Virtual Enterprises. Journal of Digital Information, Dec. 1998, 1, issue 4

4 Berger M. CoNus- A CSCW System Supporting Synchronous, Asynchronous and Auronomous Collaboration. In: Proc. Intl. Conf. on Distributed Platforms/Industrial Stream, Dresden, Feb 27-March 1, 1996

5 Lan H, Su S Y W. Component Interoperability in a Virtual Enterprise Using Events/Triggers/Rules.

6 Morrie M C, et al. CIM through Database Coordination. In: Proc. of the Int. Conf. on Data and Knowledge Systems for Manufacturing and Engineering, Hongkong, May 1994

7 Yang X, et al. VIASCOPE: an CORBA/IIOP-based Information Integration System for Virtual Enterprises. In: Proc. of the Int. Conf. on ICYCS' 99. Aug. 1999