维普资讯 http://www.cqvip.com

轻量级开放式移动 Agent 系统:原理与实现*)

余 萍 马晓星 曹 春 吕 建

(南京大学计算机软件新技术国家重点实验室 计算机科学与技术系 南京 210093)

摘 要 移动 agent 计算模式是一种灵活的基于 Internet 的分布式计算模式,但是"黑盒"式的设计和"单体"结构使得既有移动 agent 系统过于冗余,且不能随着用户需求和计算环境的变化动态修改系统的结构和行为。因此提出利用反射技术将移动 agent 系统"构件化"使其能够支持系统的"可定制"或"可裁剪",并采用开放式的实现为移动 agent 系统的使用者提供扩展系统的能力。反射架构下的移动 agent 系统能够在运行时刻无需关闭系统动态加载或卸载特定模块;通过移动 agent 和底层平台的反射特性能自然地实现 agent 和 agent 平台的自适应。遵循这个途径开发了具有反射特性的 Artemis-Mogent 原型系统,并通过试验示范了新系统应用反射后的效果。

关键词 移动 agent,轻量级,开放式,反射,自适应

Lightness and Openness of Mobile Agent Systems: Principles and Implementation

YU Ping MA Xiao-Xing CAO Chun LU Jian

(State Key Laboratory for Novel Software Technology, Dept. of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract Mobile agent computing paradigm has been considered to be promising in Internet computing. However, its wide acceptance is hindered by the monolithic and "black box" design of mobile agent system. Such kind of design makes mobile agent systems difficult to adapt to the evolving computing environment and user requirements by dynamically adjusting their structure and behavior. In this paper, a lightweight framework of mobile agent system is introduced with the help of reflection technology. By this means, the mobile agent system is structured as an open set of components which can be customized and tailored according to current needs. The open implementation techniques presented in this paper enable the mobile agent system to dynamically load/unload specific modules during runtime without shutting down the system. Eventually, self-adaptation of the agent applications is naturally realized via the reflection of the agents and the underlying system. A prototypical system called "Artemis-Mogent" is implemented as a proof of the approach. An experiment is also carried out to demonstrate the effects of reflection.

Keywords Mobile agent, Lightweight, Open implementation, Reflection, Self-adaptation

1 引言

移动 agent 计算模式是随着 Internet 技术的发展而出现的一种新型分布式计算模式。一般来讲,移动 agent 是一个运行于开放、动态网络环境中的封装良好的计算实体,它按照一定的规程,能够自主地在异构网络上移动,寻找合适的计算资源、信息资源或软件资源,利用与这些资源同处一台主机或一个局部网络的优势,就近处理或使用这些资源,代表用户完成特定的任务[1,2]。

移动 agent 技术在 20 世纪 90 年代末期曾掀起一股研究的热潮,Kotz 等曾预言未来几年内多数 Internet 站点将能接受移动 agent 计算[3],移动 agent 技术可能成为新一代 Internet 环境中最为灵活和高效的计算模式之一。近几年来,移动 agent 技术已被应用到很多领域中,比如电子商务[2]、web 服务[4]、移动计算[5,6]、普及计算[7]等。但是在构建大型的复杂应用时,移动 agent 技术与传统的分布式计算模式相比依然不具有足够的说服力和竞争力,因为传统的应用很难完全放弃原有的底层架构,以移动 agent 系统取而代之。为此,Kotz 等研究者认识到,移动 agent 技术的进一步发展方向不再是一味研制单体(monolithic)的移动 agent 系统,而应考虑如何将移动 agent 系统以组件或可嵌入模块的方式融入到其它成熟的中间件中[8]。这就意味着必须实现"可裁剪"的移动代码或移动 agent 系统,因为要让一个仅使用部分移动代码或者移动 agent 功能的应用负担整个复杂的移动 agent 支撑系统是很难接受的。比如在移动计算或普及计算中,可以采用移动 agent 技术的优点来实现异步的间断计算,但并不需要一个完整的移动 agent 系统支撑平台,通常情况下只需要一个轻量级的支撑系统来完成计算迁移的功能便足矣。因此新型移动 agent 系统架构需要解决以下两个主要问题:

·轻量化

目前已有的较著名的移动 agent 系统,如 Aglets、Concordia 等,大都是单体架构,开发者对非功能属性(non functional properties)比如安全、容错、资源池等的全面关注使得系统过于冗余和复杂,并且通常都无法对特定应用领域进行系统剪裁[8,9],这就导致了现有移动 agent 系统很难得到广泛的认可和应用,仅仅滞留在研究阶段。新型的移动 agent 系统将不再只是一个单体平台,而是可以作为一个支持计算迁移的模块插入到其它中间件中,这就要求移动 agent 系统实现非功能属性的"关注分离",将不同的功能实现为可重用的标准构件,不同的应用开发者可以根据需求的不同加载相应的模块。

·开放化

已有移动 agent 系统大多是遵循"black-box"的设计思想,封装了具体的实现细节,用户无法介入到对系统的定制和改造中。而在目前设备多元化(包括 Workstation, Laptop

*)本项目受 973 项目(No. 2002CB312002)、国家自然科学基金(No. 60233010, No. 60403014)、863 项目(No. 2005AA113160, 2005AA113030, 2005AA119010)资助。余 萍 博士研究生,研究领域:移动 agent 技术,软件协同技术,软件体系结构。吕 建 教授,博士生导师,研究领域:分布对象技术,移动 agent 技术,软件协同技术。

维普资讯 http://www.cqvip.com

PC, PDA, Mobile Phone 等计算设备)和网络多样化(存在 Ethernet LAN, Wireless LAN, GSM, GPRS 等网络)的计算环境中,只有赋予移动 agent 系统更多的自适应能力和扩展能力才能更好地发挥移动 agent 技术的优越性。因此新型的移动 agent 系统应该采用开放式的实现方式(Open Implementation),支持用户选择不同的自适应策略甚至加入自己的实现以满足特定的需求[10]。

为此,本文提出使用反射技术实现轻量级可动态裁剪的开放式移动 agent 系统。反射(Reflection)技术使得一个计算系统可以动态调整自身行为,以适应环境和需求的变化。通过元对象协议(MOP)为移动 agent 系统元层(meta level)与基层(base level)建立起因果关联,将移动 agent 系统中的非功能属性(包括日志、安全、监控、认证等模块)从核心功能中抽取出来,利用运行时刻的反射完成这些模块的动态加载,从而实现系统的可裁剪;通过元层提供的接口供用户扩展既有的功能实现系统的开放性。在此架构上,本文分别实现了对 agent 和 agent 平台的定制,可为 agent 动态加入自适应逻辑,提高 agent 执行效率;根据自定义配置,为 agent 平台动态加载性能监控或安全限制等模块,并能进一步利用反射实现 agent 平台资源的管理和优化。

本文第 2 节介绍反射的概念及新型移动 agent 系统的设计原理;第 3 节详细描述轻量级的开放式移动 agent 系统的设计,给出一个使用了反射架构的移动 agent 系统的概念模型,及其原型实现 Artemis-Mogent 系统的应用;第 4 节是相关工作及比较;最后给出结论。

2 设计原理

2.1 反射的概念

反射是指计算系统通过与自身状态和行为具有因果互联的系统自述以描述、推理和操纵自身的能力[11,12]。Smith 于 1982 年首次将反射的概念引入编程语言,他提出"计算的反射性是实体具有的按照描述、操作和处理实体所面临的主要问题领域的相同方式来描述、操作和处理实体自身的一种能力"[11];Maes 指出"反射是指计算系统推理并操作自身的能力"[12]。反射的体系结构模式为实现轻量级的开放式移动 agent 系统提供了一种解决方案:通过反射可以调整系统以满足环境的具体要或者集成特殊用户的需求,从而提高系统的开放性和自适应性。

2.2 选择反射技术的原因

反射可以分为结构反射和行为反射两类[11]。其中结构反射关注的是对象或组件的结构,例如在 Java 1.2 中,JavaBeans 就使用了内省(introspection)来实现对自身结构的描述和感知。行为反射关注的是系统的行为,典型的机制包括提供对调用过程进行具体化的接口,在过程调用时插入 pre-和 post-动作;动态 proxy 可以支持行为反射。由于移动 agent 系统大都采用 Java 语言,而且其发展趋势也是倾向于使用 Java[13],故本文提出的新型移动 agent 架构也是基于 Java 语言的。Java 提供了强大的反射设施(java. lang. reflect 包),通过 java. lang. reflect 包提供的接口,匿名对象的类信息可以在运行期间被完整地表示出来,从而能够对匿名对象执行方法调用[14]。因此利用 Java 语言的反射设施很容易对移动 agent 实现结构反射,这也是选择利用反射实现新型移动 agent 系统架构的一个主要原因。但原有的移动 agent 系统中只支持重新构造 agent 对象并恢复执行,并没有提供对 agent 结构进行调整和修改的接口,也没有提供对 agent 系统行为进行反射的接口,这些是新型移动 agent 系统需要引入的。当执行环境或用户需求发生变化的时候,原有的移动 agent 系统一般需要通过一些外部行为,如终止程序、修改代码,重新部署等来适应这些变化。使用反射技术,移动 agent 系统可以在运行时期动态调整自身的行为,根据实时获取的用户定制信息动态加载不同的模块,实现对 agent 和 agent 平台结构和行为的动态修改。

3 新型移动 Agent 系统架构

3.1 传统的移动 Agent 系统架构

传统的移动 agent 系统一般由核心服务及管理服务两大部分组成。核心服务包括对 agent 的解释器、序列化/反序列化器、agent 的迁移、命名、寻址和 agent 之间的通信等几个基本模块组成,它们是一个移动 agent 系统中必不可少的组成。管理服务则由 agent 安全、server 安全、持久化、跟踪、日志、容错等非功能性属性相关模块构成。在已有的移动 agent 系统或原型中,集成了这些管理服务中的某些或全部,且大都和系统核心服务紧密关联。整个移动 agent 系统就像一个"black box"一样,对上层封装了所有的实现细节,只提供给用户编写应用的简单接口。对于移动 agent 的应用开发者而言,他们只能是被动地使用移动 agent 系统,无法选择和定制底层设施或功能。

3.2 新型的移动 Agent 系统架构

针对前面所指出的传统移动 agent 系统的缺陷和不足,我们提出了新型的移动 agent 系统架构,即使用反射技术实现一个开放式的轻量级移动 agent 系统。对原有的移动 agent 系统进行剪裁,将核心的功能性模块集中为"基层",核心功能主要包括移动 agent 系统必须具备的"迁移"、"通信"、"命名"、"寻址"等模块;增加存储元对象的"元层",元对象主要包含一些非功能属性模块(封装为 interceptor)、移动 agent 对象的元数据(称为 meta agent)和将情境信息具体化(reify)后生成的元对象(即 context object)。其中情境信息包括两个方面:一是系统运行的计算环境中的相关实体的状态信息,比如需要访问的数据资源的大小,以及应用所依赖的硬件环境,如 CPU、内存和网络带宽等,甚至包括应用系统本身;二是用户相关的信息,比如用户的个人喜好、对系统的配置要求、当前位置等。反射系统通过元对象协议将"基层"与"元层"建立起因果连接,系统可以在运行时刻进行自我探知和推理,从而动态调整 agent 和 agent 平台的结构和行为。新型移动 agent 系统的架构见图 1。

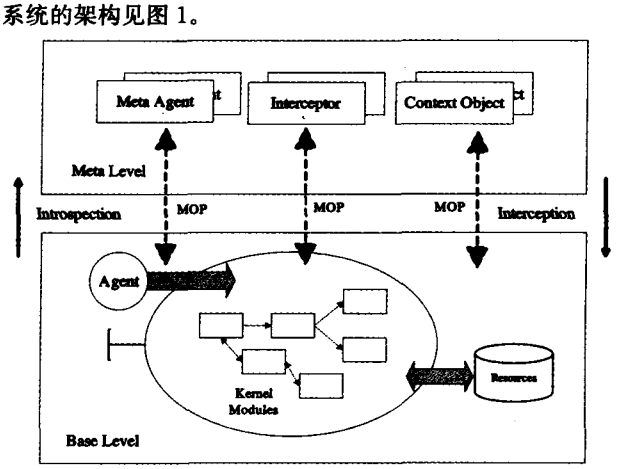

图 1 新型移动 agent 系统架构

从图 1 中可以看出,新型的移动 agent 系统是由一个实

维普资讯 http://www.cqvip.com

现核心服务的轻量内核(Kernel Modules)和构件化的额外服务模块、元数据、情境信息等,再加上一套反射设施所组成。利用反射实现移动 agent 系统的轻量化、开放化。

·"轻量级":在新架构中,移动 agent 系统被简化为一个只包含核心服务的最小内核,该内核可以被集成到其它中间件中(如 J2ME);而其它的非功能属性模块,比如 agent 认证、日志、跟踪等则被构件化为独立的 interceptor,由用户根据计算平台的特点和需求对系统进行定制。interceptor 可以由移动 agent 系统提供,也可以使用成熟的中间件平台中既有的服务,甚至可以由用户自己实现。如何定制 interceptor,由移动 agent 平台管理员制定的 Server Profile 给出。系统运行时刻读取 Profile,根据当前的配置决定是否要为迁移来的 agent 添加 interceptor 及按照什么次序添加 interceptor,见图 2。Profile 使用 XML 描述,其中〈interceptor〉元素包含"name"和"scope"两个属性,"name"属性指出 interceptor 的实现类的名称,"scope"属性表明该 interceptor 的影响范围("pre"表示 interceptor 在 agent 执行前加载,"post"表示 interceptor 在 agent 执行后运行,"global"则表示前后均需要)。

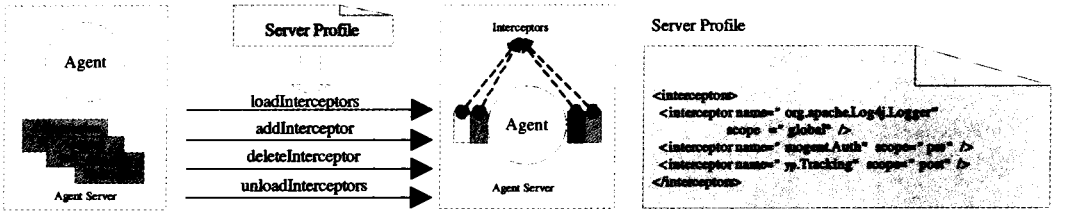

图 2 轻量级移动 agent server 的动态配置

在新型移动 agent 架构的原型实现 Artemis-Mogent 中,为了实现 interceptor 的动态加载,借助了 Java 字节码改写工具"javassist"[15],它允许在加载时刻(load-time)对应用程序的结构进行修改,使得插入 interceptor 成为可能。当携带数据的 agent 对象迁移到远程主机,该主机在加载 agent 时根据 Profile 对它进行动态配置,为 agent 加载本地非功能属性代码。

·"开放式":在新架构中,agent 和 agent 平台可以加载不同的自适应策略。系统预定义了部分策略供选择,同时给出元接口,适当地暴露实现细节,供用户扩展已有功能,提供自己的实现。

(1)移动 agent 的自适应:由于网络、运行平台(例如可持移动设备电池的电量、CPU 使用率、内存)等情境信息的变化,移动 agent 需要更好地利用它的可移动特性及时地调整自己的行为以适应环境的变化,例如改变迁移的路径(迁移到 CPU 较为空闲的主机上执行计算),更改数据访问的方式(若内存不足以完整拷贝数据则改为远程引用)。在 Artemis-Mogent 中,移动 agent 的自适应策略使用 Agent Policy 定义,通过元对象协议(loadPolicy,addPolicy 等方法)将自适应策略及与它相关联的资源具体化为 agent 的 Meta-agent,从而将原 agent 转换为一个新的具有反射特性的移动 agent,并为它提供访问情境信息(被封装为 Context Object)的元接口,见图 3。新的 agent 可向 Context Object 询问特定类型 Context 的值(比如 CPU 使用率),得到返回值后根据策略适时地调整 agent 的行为。

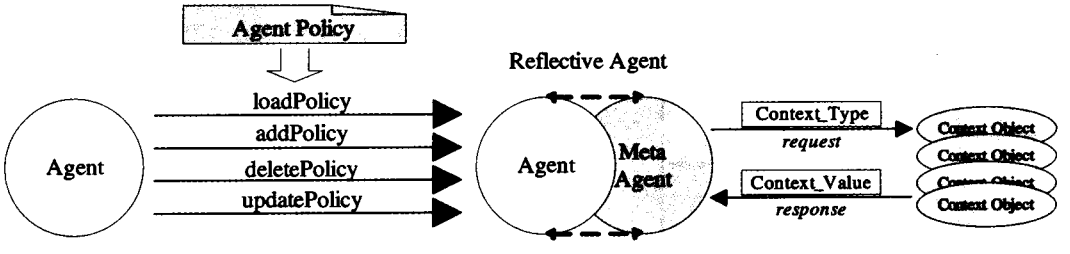

图 3 移动 agent 的自适应

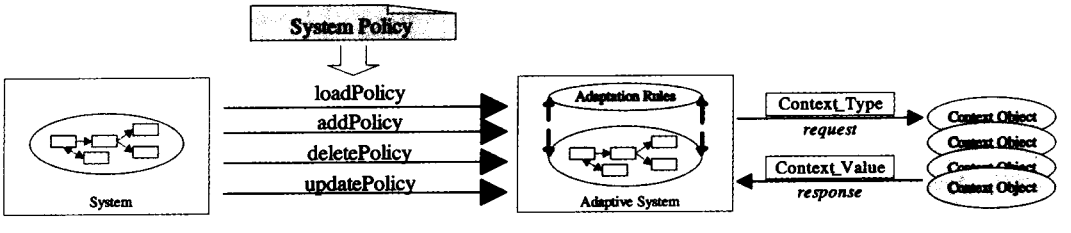

图 4 移动 agent 平台的自适应

(2)移动 agent 平台的自适应:轻量级的移动 agent 系统由于消耗的资源少,可以被更广泛地应用到无线移动设备或其它可持设备中(Pocket PC,PDA,Mobile Phone)。由于这些设备的硬件条件相对较差(网带宽小、网络连接状况不稳定、处理器能力弱、内存小、电量有限等),需要更理、更高效地使用这些有限资源。因此新型的移动 agent 系统允许运行平台定制自适应策略以合理地调度 agent 对资源的使用,比如在 CPU 高负荷运转期间为满足通讯的质量暂时拒绝需要消

维普资讯 http://www.cqvip.com

耗大量资源的移动 agent 的计算请求。平台自适应策略 System Policy)的加载过程、平台自适应调整过程与移动 agent 的自适应类似,见图 4。

(3)自适应策略:在 Artemis-Mogent 系统中,移动 agent 和移动 agent 平台的自适应策略使用 XML 描述,具有良好的可读性和易扩展性。它们被看作是系统的元数据,独立于系统的核心功能实现(即是否定义由使用者决定)。图 5.1 和图 5.2 分别给出了各自的实例。

```
<AgentPolicy>
  <class name=" Migration " >
    <context type=" CPU"  value=" 80"
             opeartor=" largerThan" />
    <policy>JUMP</policy>
  </class>
  <class name=" DataTransfer " >
    <context type=" MEM"  value=" 20K"
             opeartor=" lessThan" />
    <policy>REF</policy>
  </class>
</AgentPolicy>
```

该Agent Policy包含两个自适应方案:
1. 迁移自适应策略:当获知欲迁移的主机CPU使用率超过80%时,agent将跳过该节点,到下一个节点执行计算;
2. 数据传输自适应策略:当可用内存容量小于20K时,agent将使用远程引用的方式传输数据。

图 5.1 移动 agent 的自适应策略

```
<SystemPolicy>
  <class name=" Scheduler" >
    <context type=" CPU"  value=" 90"
             opeartor=" largerThan" />
    <policy>DENY</policy>
  </class>
  <class name=" MemResv" >
    <context type=" MEM"  value=" 10K"
             opeartor=" lessThan" />
    <policy>SUSPEND</policy>
  </class>
</SystemPolicy>
```

该System Policy包含两个自适应方案:
1. CPU调度策略:当系统CPU使用率超过90%时,将拒绝agent前来计算的请求,直到CPU使用率低于90%;
2. 内存管理策略:当可用内存容量小于10K时,将挂起当前agent的计算,执行内存垃圾回收操作,直到可用内存大于10K。

图 5.2 移动 agent 平台的自适应策略

(4)开放式实现:为了实现开放性,Artemis-Mogent 系统使用了观察者模式和策略模式[16],移动 agent 系统"观察"感兴趣的资源使用情况。当满足自适应规则的时候,根据 agent 携带的策略及 agent 平台制定的策略,动态调整 agent 的行为及平台赋予 agent 的执行权限。用户可以通过实现迁移策略(MigrationPolicy)和调度策略接口(Scheduler)提供自己的自适应方法(图 6 仅给出了简化了的部分 UML 类图)。

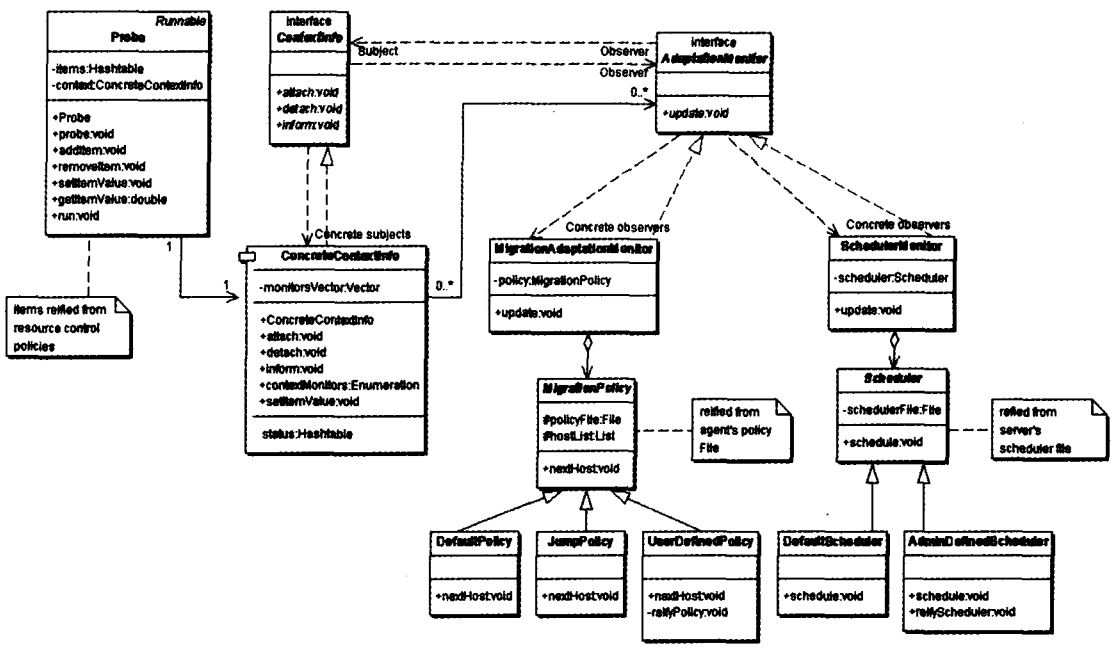

图 6 Artemis-Mogent 系统中的设计模式

3.3 新型移动 Agent 系统的应用

·应用领域

本文利用反射计算技术,实现了轻量级、开放式的移动 agent 系统。其中"轻量级"的特点使得新型移动 agent 系统可以在运行时"按需"(on-demand)加载模块,减少了资源占用,适合应用于资源有限的"轻量级"设备中(如 PDA,Mobile Phone);interceptor 机制使得系统具备灵活的可配置能力,系统的功能得以动态扩展;非功能属性的分离则进一步提高了移动 agent 系统的构件化程度,"构件化"的移动 agent 系统可以作为一个独立的模块嵌入到成熟的中间件中,目前已尝试将 Artemis-Mogent 无缝集成到 JBoss 应用服务器和 J2ME 中,以支持计算的迁移。新型移动 agent 系统的"开放性"体现在可自由选择 agent 和 agent 平台的自适应策略;用户可自定义策略并提供自己的实现等方面。具有情境感知和自适应能力的移动 agent 系统在复杂和多变的计算环境中,可以更好地选择合适的网络资源,提高计算效率。同样地,移动 a-

维普资讯 http://www.cqvip.com

gent 的计算平台亦可以通过自适应更合理地分配和管理资源，增强自我保护能力。此外，新型移动 agent 系统也为多元化的计算设备提供了根据设备自身的硬件条件“个性化”移动 agent 系统的能力。

· 实验评估

为了检验 Artemis-Mogent 系统利用反射实现自适应的效果，我们开发了一个简单的基于移动 agent 的股票信息查询应用。在该应用场景中，若干移动 agent 被分别派发到两台不同的提供股票信息的主机上，查询用户感兴趣的股票价格及走势。移动 agent 迁移自适应策略和平台自适应策略如图 5.1 和图 5.2 所示。在实验的过程中我们模拟了网络中断、阻塞及主机高负荷运转等多种情况。agent 通过探测到的这些情境信息实时地调整了迁移路径，使得整个系统的执行效率得到了很大的提高，同时 agent 平台也可以拒绝为消耗大量资源的 agent 提供服务，安全性得到了保障。在这个试验中，90% 的 agent 被派往 Host1，10% 的 agent 被派往 Host2。从图 7.1 中可以看到，由于 90% 的 agent 被派往 Host1，所以 Host1 长期处于满负荷状态；而 Host2 由于只有 10% 的 agent 在此工作，所以大部分时间处于空闲状态。从图 7.2 中可以看到，加载了自适应策略后，agent 在迁移的过程中能够动态选择当前最合适的主机完成计算，这样 Host1 的负荷部分被转移到了 Host2 上，达到了一定程度上的负载平衡，agent 也就在相对空闲的服务器上更快地完成了自己的计算任务。

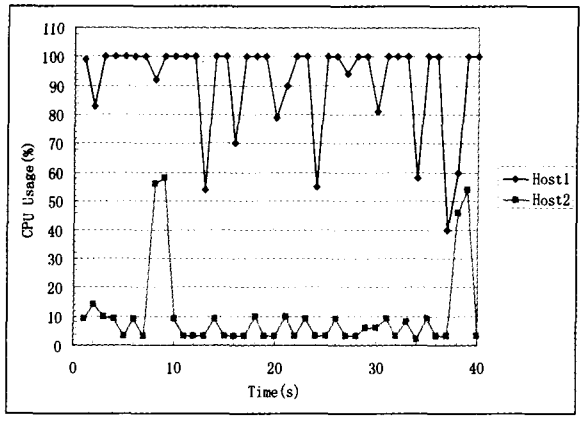

图 7.1 没有自适应的 CPU 使用率

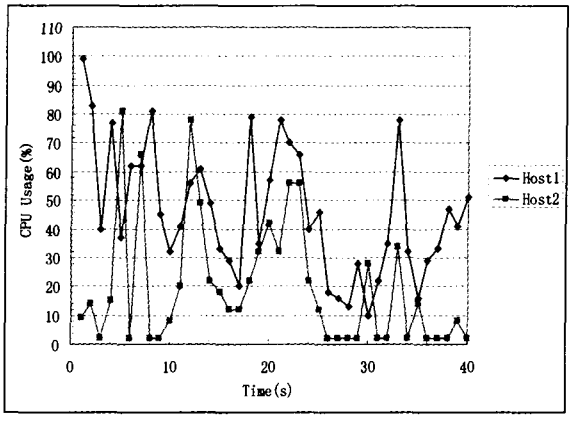

图 7.2 支持迁移自适应的 CPU 使用率

4 相关工作及比较

4.1 相关工作

最早提出轻量级移动代码的 Picco 开发了模块化的移动代码工具集 μCode[17]，其目标在于设计一个灵活的可扩展的轻量级移动代码支撑系统。μCode 关注的是移动代码的迁移粒度，它能支持不同形式的迁移。目前工业界已经进入一个基于构件的软件开发时代，Picco 认为移动代码或移动 agent 系统也应该实现构件化。Artemis-Mogent 系统将非功能属性封装为独立的 interceptor，允许用户按照各自要求对系统进行配置。另外，目前已有一些关注反射技术在移动 agent 系统中应用的前期工作。Ledoux 在文[18]中指出对资源及 agent 与资源之间绑定关系的“具体化”是实现自适应的关键。Watanabe 等在文[19]中提出可以利用元层 agent 定制基层 agent 的容错策略，即对每个元层的 agent 对象引入用户定义的 pre-和 post-出错处理方法，提高基层 agent 的容错性。Artemis-Mogent 系统则不仅将反射应用于执行环境中实现情境感知，而且将 agent 和 agent 平台作为反射的对象，通过反射实现 agent 和 agent 平台运行时刻的自适应修改。

4.2 移动 Agent 系统比较

表 1 将 Artemis-Mogent 与 Aglets[20]，Concordia[21] 和 μCode[17] 分别从迁移方式、是否支持自适应和可定制，以及系统是否为轻量级这几个方面进行了比较。通过比较看出，在使用了反射机制后，Artemis-Mogent 系统的开放性和灵活性都得到了提高，可以支持运行时刻 agent 和 agent 平台的自适应，并能通过元对象协议动态加载模块，使系统更加轻量化与个性化。

表 1 几个移动 agent 系统的比较

	Aglets	Concordia	μCode	Artemis-Mogent
迁移方式	弱迁移	弱迁移	弱迁移，强迁移	弱迁移，远程计算
自适应	没有	没有	没有	支持 agent 和 agent 平台的自适应
可定制性	不可定制	不可定制	可选择不同的迁移方式	可由用户定制自适应策略和非功能属性模块
轻量级/重量级	重量级	重量级	轻量级	轻量级

总结 为了适应 Internet 计算环境尤其是移动环境中设备的多样性、网络的多变性，本文将反射技术应用到移动 agent 系统中，实现了一个轻量级的开放式移动 agent 系统，提高了移动 agent 系统的构件化程度，使移动 agent 技术能够融入主流的中间件。该系统在原有的移动 agent 系统架构中引入了元层，利用元对象协议动态修改 agent 和 agent 平台的行为，提高系统的可裁剪性和扩展性，支持系统的“按需”配置。在原型系统 Artemis-Mogent 中实现了移动 agent 和移动 agent 平台的自适应，通过情境感知不仅可以进行资源监控而且还为移动 agent 提高 QoS、平台避免 DoS 攻击提供了可能。

参 考 文 献

1 Fuggetta A, Picco G P, Vigua G. Understanding code mobility. IEEE Transactions on Software Engineering, 1998, 24(5): 342～361

2 陶先平. 基于 Internet 的移动 agent 技术和应用研究:[博士学位论文]. 南京：南京大学，2001

维普资讯 http://www.cqvip.com

3 Kotz D, Gray R S. Mobile agents and the future of the Internet. ACM SIGOPS Operating Systems Review, 1999, 33(3): 7～13

4 Palathingal P, Chandra S. Agent approach for service discovery and utilization. In: Proc. of the 37th Annual Hawaii International Conference on System Sciences, Jan. 2004

5 Bellavista P, Corradi A, Stefanelli C. Mobile agent middleware for mobile computing. IEEE Computer, 2001, 32(3): 73～81

6 Cao Jiannong, Tse D C K, Chan A T S. PDAgent: A platform for developing and deploying mobile agent-enabled applications for wireless devices. In: ICPP 2004. Montreal, Canada, Aug. 2004. 510～517

7 Liu R, Chen F, et al. Agent-based web services evolution for pervasive computing. In: Proc. of the 11th Asia-Pacific Software Engineering Conference, Busan, Korea, 2004. 726～731

8 Kotz D, Gray D, Rus D. Future directions for mobile agent research. IEEE Distributed System online, 2002

9 Aleksy M, Kouthaus A, Schader M. CARLA - a CORBA-based architecture for lightweight agents. In: Proc. of the IEEE/WIC International Conference on Intelligent Agent Technology, Halifx, Canada, Nov 2003. 111～118

10 Montanari R, Tonti G, Stefanelli C. Policy-based separation of concerns for dynamic code mobility management. In: Proc. of 27th Annual International Computer Software and Applications Conference. Dallas, Texas, Nov 2003. 82～90

11 Smith B C. Reflection semantics in a procedural programming language: [dissertation]. MIT, 1982

12 Maes P. Concepts and experiments in computational reflection. ACM SIGPLAN Notice, 1987, 22(12): 147～155

13 Lange D B. Mobile objects and mobile agents: The future of distributed computing? In: Proc. of the ECOOP'98. Brussels, Belgium, July 1998. 1～12

14 SUN Microsystems. Java 2 Platform, Standard Edition, Version 1.4.2, 2003

15 Chiba S. Load-time Structural Reflection in Java. In: Proc. of the ECOOP'2000. Malaga, Spain, Jun. 2000. 313～336

16 Gamma E, Helm R, Johnson R, et al. Design patterns: Elements of reusable object-oriented software. Boston: Addison-Wesley Professional, 1995

17 Picco G P. μCode: A lightweight and flexible mobile code toolkit, In: Proc. of 2nd International Workshop on Mobile Agents. Berlin, Germany, 1998. 160～171

18 Ledoux T. Adaptability in mobile agent systems using reflection. RM 2000, IFIP/ACM Workshop on Reflective Middleware, Middleware'2000. New York, 2000

19 Watanabe T, Noriki A, Shinbori K. A reflective framework for reliable mobile agent. ECOOP 2000, Workshop on Reflection and Metalevel Architectures. Cannes, 2000

20 Lange D B, Oshima M, Mitsuru O. Programming and deploying mobile agents with aglets. Boston: Addison-Wesley Professional, 1998

21 Wong D, Paciorek N, Walsh T, et al. Concordia: An infrastructure for collaboration mobile agents. In: Proc. 1st International Workshop on Mobile Agents 97. Berlin, Germany, Apr 1997. 98～110

(上接第 4 页)

参 考 文 献

1 Li Zhi, Mohapatra P. Impact of Topology On Overlay Routing Service. http://www.ieee-infocom.org/2004/Papers/09_1.PDF

2 Chu Yang-hua, Rao S, Zhang Hui. A Case for End System Multicast. In: Proceedings of ACM Sigmetrics, June 2000. 1～12

3 Jannotti J, Gifford D K, Johnson K L, et al. Overcast: Reliable Multicasting with an Overlay Network. Proceedings of Operating Systems Design and Implementation(OSDI), October 2000

4 Francis P. Yoid: Your own Internet distribution: [Tech Rep]. www.aciri.org/yoid, UC Berkeley ACIRI Tech Report, Apr. 2000

5 Chawathe Y. Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service: [PhD thesis]. Department of EECS, UC Berkeley, Dec. 2000

6 Pendarakis D, Shi S, Verma D, et al. ALMI: An application level multicast infrastructure. 3rd Usenix Symposium on Internet Technologies and Systems(USITS), Mar. 2001

7 Li Zhi, Mohapatra P. QRON: QoS-Aware Routing in Overlay Networks. QRON IEEE Journal on Selected Areas in Communications, 2004, 22(1): 29～40

8 Andersen D G, Balakrishnan H, Kaashoek M F, et al. Resilient Overlay Networks. In: Proc. 18th ACM SOSP, Banff, Canada, October 2001

9 Duan Zhenhai, Zhang Zhili, Hou Yiwei T. Service overlay networks: SLAs, QoS, and bandwidth provisioning. IEEE/ACM Trans Netw, 2003, 11(6): 870～883

10 Subramanian L, Stoica I, Balakrishnan H, et al. OverQoS: An Overlay Based Architecture for Enghancing Internet QoS. NSDI, 2004. 78～84

11 Yun Pan. Research on Active Overlay Network and its key technology: [Ph D Thesis]. the China of Mining and Technology University, December 2003

12 Charikar M, Guha S. Improved combinatorial algorithms for the facility location and k-Median problems. IEEE Symposium on Foundations of Computer Science

13 Shmoys D B, Tardos E, Aardal K. Approximation algorithms for facility location problems. In: 29th ACM Symposium on Theory of Computing, 1997. 265～274

14 Medina A, Lakhina A, Matta I, et al. BRITE: An Approach to Universal Topology Generation. In: Proc. of MASCOTS '01, August 2001

15 Waxman B. Routing of Multipoint Connections. IEEE Journal on Selected Areas in Communications, 1988, 6(9): 1617～1622

16 Yu Z W, Pan Y, Wang L C. On the model of QoS multicast routing problems in active networks. International Conference on Computer Networks and Mobile Computing, Proceedings, 2003. 419～422

17 Choi S, Turner J, Wolf T. Configuring Sessions in Programmable Networks. Computer Networks, 2003, 41(2): 269～284