ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.8, August 2007, pp.2038−2048
DOI: 10.1360/jos182038

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

普适环境下基于软件代理虚拟化的应用迁移*

周 宇[1,2,3], 马晓星[1,2+], 曹建农[3], 余 萍[1,2], 吕 建[1,2]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210093)

[2](南京大学 计算机科学与技术系,江苏 南京 210093)

[3](香港理工大学 计算学系,香港)

Software Agent-Virtualized Application Mobility in Pervasive Environments

ZHOU Yu[1,2,3], MA Xiao-Xing[1,2,+], CAO Jian-Nong[3], YU Ping[1,2], LÜ Jian[1,2]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210093, China)

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

[3](Department of Computing, Hong Kong Polytechnic University, Hong Kong, China)

+ Corresponding author: Phn: +86-25-83593283, Fax: +86-25-83593283 E-mail: xxm@ics.nju.edu.cn, http://www.nju.edu.cn

Zhou Y, Ma XX, Cao JN, Yu P, Lü J. Software Agent-virtualized application mobility in pervasive environments. *Journal of Software*, 2007,18(8):2038−2048. http://www.jos.org.cn/1000-9825/18/2038.htm

Abstract: In virtualized computing environments, users' mobility in physical space poses a challenge in the research on the applications' mobility in cyber space. The characteristics of network-intensiveness and resource-intensiveness in pervasive environments provide an opportunity for application mobility. To support it, the paper exploits software Agents to realize the resource virtualization in cyber space. It investigates the application mobility from the perspectives of underlying application model, mobility management, and context-awareness, etc. By exploiting software Agents' inherent autonomy and mobility, an Agent-based mechanism for adaptive component binding and migration is proposed. A framework called MDAgent is proposed and the corresponding prototype is implemented. It can provide the support of mobility management, context-awareness, resource matching and logic reasoning for the application mobility. On top of the framework, several applications are implemented and the performances are evaluated.

Key words: software Agent; pervasive computing; application mobility

摘 要: 虚拟计算环境中的用户在物理空间的移动要求其相关应用能够在网络空间进行相应的移动.为支持应用的迁移,利用软件代理技术来实现计算网络空间以及其中资源的虚拟化.具体而言,就是结合情境感知技术,利用软件代理本身所具有的自治性和移动性实现应用组件的动态绑定和迁移,进而提出一种基于代理的支持应用迁移的

* Supported by the National Natural Science Foundation of China under Grant No.60403014 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2006AA01Z159 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2002CB312002 (国家重点基础研究发展计划(973)); the Competitive Earmarked Research Grant of the University Grants Committee of Hong Kong, China, under Grant No.PolyU 5183/04E (大学教育资助委员会角逐研究用途基金(香港)); the Natural Science Foundation of Jiangsu Province of China under Grant No.BK2006712 (江苏省自然科学基金)

Received 2007-03-02; Accepted 2007-05-31

结构模型 MDAgent,并实现了相应的原型系统,可为应用的迁移提供移动管理、情境感知、资源匹配和推理机制等多方面的支持.在此基础上开发出若干应用并给出相关性能分析.

关键词: 软件代理;普适计算;应用迁移

中图法分类号: TP393 **文献标识码:** A

在普适计算环境中,为了达到“随时随地、透明地获得数字化的服务”[1]的目的,人们试图让交互着的应用伴随移动的用户在对应的网络空间中迁移,从而既可以为用户保持会话连续性,又可以给予他们熟悉的操作空间,这对于提高工作效率和用户的满意度有积极作用.这种应用迁移使得用户不必关心应用或者服务分布在哪里、如何分布等技术细节,而是更注重服务本身的质量和内涵.然而,开发这种可无缝迁移的应用远非易事.考察现今的计算平台后不难发现,它们大多着重于提供通信支撑以及对分布资源的透明访问,而缺乏相应的移动管理、状态/对象持久化等方面的直接支持.如果把这些关注都置于应用软件级别上,则要求应用开发者在考虑业务逻辑的同时,还要额外考虑这些繁琐而专门的技术细节,而且也限制了软件复用,增加了维护和演化的难度.

为此,本文将从计算虚拟化的角度来讨论应用迁移所需的中间件支持.计算虚拟化就其本质而言,是为物理存在资源提供较为一致的逻辑表示,用以解耦用户感知到的系统行为和实际的底层实现[2].随着计算技术的发展,虚拟化的外延也从早期的虚拟存储、虚拟机器、虚拟专用网等逐渐扩展到虚拟网络资源、网格计算[3]、虚拟服务[4,5]等.但上述的应用迁移往往对应于用户在物理空间的移动,既有的计算虚拟技术未能明确涵盖这种技术特征.因此我们的问题是,如何以适当的虚拟机制在上层应用的移动之后重新解释底层的资源绑定,比如通过网络引用远程访问或者重新绑定本地的同类资源,从而尽可能地降低由移动带来的底层资源的区域化差异,也就是实现对应用运行“处所”的虚拟化,以支持应用的无缝迁移.

在实现方面,我们使用软件代理(Agent)来实现对处所的虚拟化.这里,软件代理是指运行于开放、动态的网络环境中的封装良好的计算实体.它代表用户自主地在网络上移动,完成指定的任务[6],而且它能够感知环境并根据环境变化做出相应调整[7].应用业务逻辑运行于一致的虚拟环境中,而其实际使用的资源的动态发现、匹配、绑定与重绑定以及相应会话状态的维护由自主移动软件代理依据上下文环境来完成.

在前期的基于代理应用迁移的工作基础上[8,9],本文对应用的结构框架、移动管理、上下文推理机制、资源绑定等方面做了进一步研究,以更好地支持资源和计算密集性网络环境下的应用迁移.为了提高迁移的效率,使用基于代理的松耦合应用结构,提出了一种组件动态绑定与应用轻量迁移机制.为了提高应用迁移的应用范围,采用基于代理的协同机制,在跟踪迁移(follow-me)模式的基础上,新增了对复制迁移(clone-dispatch)模式的直接支持.此外,还采用了内嵌于自治代理的逻辑推理模块来支持可适应性的迁移需求表达,并用基于本体的描述机制为虚拟环境中的资源提供一个统一的语义级别的描述框架.

本文讨论应用迁移技术在迁移粒度选择、资源描述与服务适应性、移动管理以及情景感知等方面的设计需求和相应的设计框架.在此基础上给出原型系统的实现,开发若干应用并做出相应的性能评估;最后给出相关研究的介绍和比较以及研究工作的总结.

1 应用迁移技术需求

为了使应用开发者从应用迁移所设计的资源绑定、情境感知以及状态管理等细节中解放出来,我们需要开发一种中间件来对物理处所的资源使用进行虚拟化.在本节中,我们首先讨论高效、无缝的应用迁移的若干技术需求.

1.1 粒度选择

应用的迁移可以有不同的粒度.与应用程序的整体迁移相比,组件级别的应用迁移在效率和性能上更具吸引力.组件基本的迁移要求系统能够支持灵活的组件绑定和迁移机制.普适计算环境提供了多样化的网络连接和丰富的计算资源.利用这些基础设施,将应用设计为一些具有松散耦合、较高内聚特性的分布式组件有助于组件迁移的实现.与此同时,这些组件要按照一定的方式组织起来,协调工作需要协同模块的支持.在迁移前后,

应用的状态需要保持一致性和连贯性,因此需要状态管理模块.网络与设备的多样化与情境的动态性也对迁移前后的适应机制提出了要求.

1.2 资源描述与服务适应

如前所述,在应用组件迁移至目的空间之后,由于诸多因素,比如有些资源(例如打印机等设备)的不可移动性,原先的资源绑定可能已经失效,在这种情况下,某些应用会因此而抛出异常.然而在普适计算环境下,资源呈密集型特点,在目的空间可能存在所需的资源,因此需要有一种资源的描述与匹配机制,并以此作为资源重绑定的前提,为上层的应用迁移提供必要的虚拟化支持.由于资源本身的多样化,简单的语法级别的匹配易降低匹配的准确度,在这种情况下,需要一种语义级别的匹配机制.

服务适应的适应性有两重含义.从设备的角度来看,由于不同的设备有不同的硬件特性,比如处理能力、分辨率、存储容量,因此,对于迁移后的应用或其组件要求具有能够根据不同的设备自我调整的能力,以提高用户的满意度,比如一个界面组件迁移到一个 PDA 上之后,可根据硬件信息调整屏幕的大小.从用户的角度来看,特定用户具有其操作习惯和使用偏好,例如,一个习惯于左手操作的用户不会立即适应迁移之后的应用而成为默认的右手操作.这样,对于应用还要求具有能够根据不同用户的使用偏好自我调整的能力.

1.3 移动管理

应用迁移与其他类型的迁移(例如数据迁移等)的不同点在于,应用是一个主动的可执行实体,在到达新的网络空间之后,能够保持会话的连续性.关于应用迁移的特性,可以从 3 个维度考量,这 3 个维度隐含在 3 个问题之内:(1) 哪一个组件应该移动,是逻辑组件、用户接口组件还是其他?(2) 要迁移到哪里,是在同一个虚拟空间,还是在不同的虚拟空间?(3) 需要哪一种迁移模式,是跟踪迁移,还是复制迁移?

图 1 给出了迁移分类的示意.

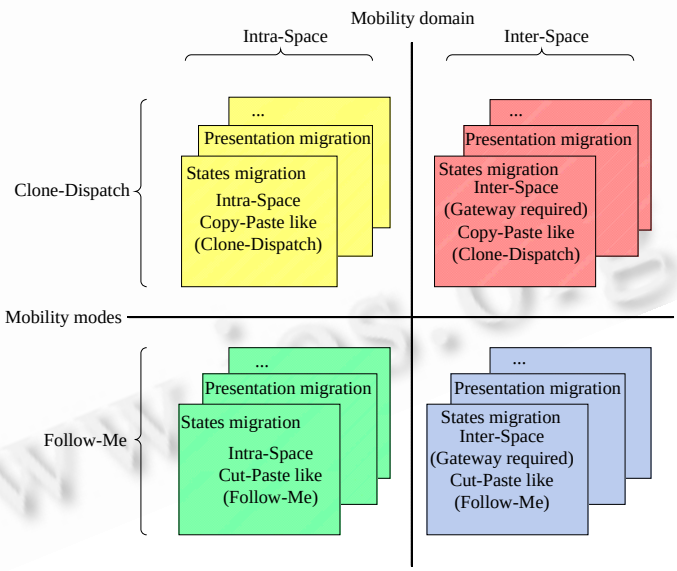


Fig.1 Migration classification

图 1 迁移分类

从迁移模式的维度来看,大致有两种:跟踪迁移和复制迁移.前者跟从单个用户形迹,当用户从一个空间转移到另一个空间时,与其交互的应用也相应地迁移到该用户所到达的空间,这一过程在直觉上类似于文本编辑中常用的剪切-粘贴(cut-paste)操作;后者则是指应用能够根据用户的交互或指令等上下文信息复制自身组件,并迁移到新的空间.在很多情况下,简单的复制-迁移是不够的,还需要在复制组件与原组件之间进行同步控制.从迁移的域维度来看,也有两种:在一个网络空间内部的移动和网络空间之间的移动,后者常见于大尺度范围内

的移动.从迁移的组件维度来看,既可以是应用的用户接口迁移,又可以是状态迁移,也可以是事物逻辑迁移等.

以上 3 个维度的综合才是对应用迁移的一个比较全面的认识,其支撑软件的设计也要求考虑到以上方面.

1.4 情境感知

为了捕捉用户的移动或意图,应用需要具备感知情境的能力.所谓情境(context),一个较为广泛接受的定义是:能够表征与人机交互相关状态的任何信息[10].与应用迁移密切相关的信息往往包括用户的标识、位置以及资源的绑定和使用情况等.

不同的情境信息往往有着不同的时序特征.例如,用户的位置信息往往动态性较强,然而,用户的身份标识往往在人机交互过程中比较稳定.在建模过程中,需要考虑情境的这一特性,对其加以区分.

通常,底层的传感器件仅能采集一些原始数据,比如距离信息、listener id 等,要想通过这些数据得到空间位置、用户标识等有用信息需要采用相应的情境融合机制,比如映射、过滤、推理、预测等.

2 MDAgent 体系结构框架

针对上述需求和分析,我们设计了一个基于代理的虚拟化应用迁移中间件 MDAgent.图 2 给出了 MDAgent 的结构视图.

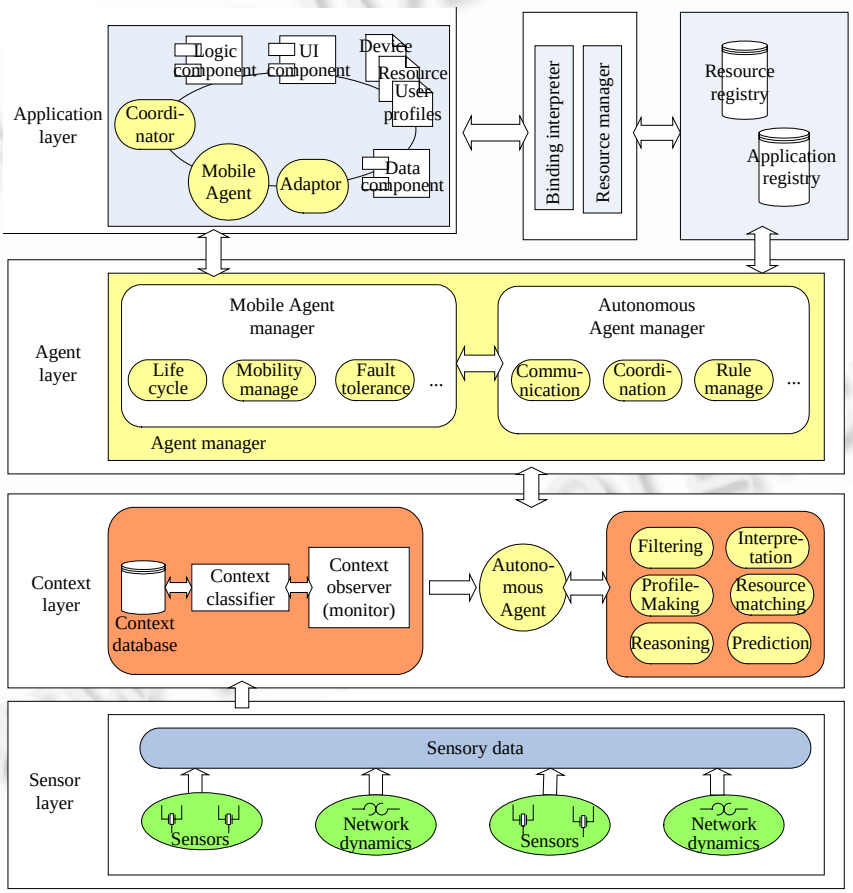


Fig.2 Framework overview

图 2 框架概览

MDAgent 由 4 层组成,分别是传感器层、情境层、代理层和应用层.

传感器层由一系列物理上或逻辑上部署的传感模块组成,其主要部分是用来探测用户的移动、位置、网络

连接、时延等原始数据.由于数据源本身的动态性以及传感模块的延迟性,这些数据往往不能直接被上层所用,暂时汇聚于感知数据队列中.

在情境层,一个情境分类模块根据情境信息的时序特征将变化频率不同的情境数据存储到不同的数据库中.利用观察者设计模式[11],情境监控模块向情境管理模块注册监听事件.当监听事件发生后,自治代理会被自动触发,并接管余下进程.

代理层是连接情境层与应用层的中间层次.它由两种类型的代理管理模块构成:自治代理管理模块和移动代理管理模块.自治代理负责上下文信息的过滤解释以及推理及决策等"脑力劳动";移动代理负责组件的迁移、同步、恢复执行等"体力劳动".两种代理之间通过消息传递进行通信.当自治代理感知到用户的移动或者接收到用户的指令要迁移相应的应用时,它首先会通知移动代理准备迁移.移动代理会准备保存应用的计算状态,并检索目的空间应用与资源注册中心.根据检索结果和应用相关的规则,自治代理进行推理,决定由移动代理绑定哪一个应用组件迁移到目的空间,移动代理接管余下的传输与同步工作.

应用层主要由业务逻辑相关组件以及资源和应用及资源注册管理模块构成.在运行时刻,主要是与代理层的移动代理相交互,在应用移动到相应的空间之后,资源绑定解释模块会根据资源注册管理模块和应用本身提供的信息对资源进行重新绑定,而这个过程对应用来说是透明的.也就是说,移动后由于资源的地域化特性而带来的差异被屏蔽了,取而代之的是一个统一的、虚拟的网络资源绑定接口.关于应用层的相关特征将在下一节加以阐述.

2.1 应用管理

如前所述,为支持组件级的迁移,需要借助于网络的支持和松耦合程序结构.我们借鉴了 MVC 设计范型思想,提出了一种结构模型.

2.1.1 应用结构模型

我们的应用结构模型如图 3 所示,分为两层.上层主要由应用逻辑有关模块构成,包括逻辑控制、用户界面、资源、数据等.业务逻辑模块处理数据,控制着用户展示模块同时借助于资源描述通过资源解释器绑定特定的资源,比如数据库资源或硬件资源等.此外,这一层还包含一些配置文件,包括应用的接口描述、资源描述等,这主要是为了向服务中心注册时用.由于这一层次直接与用户交互,所以对用户来说是可见的.

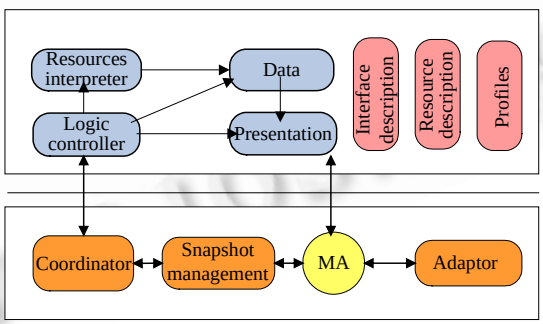


Fig.3 Application model

图 3 应用模型

在基层,主要模块包括协同、快照管理、移动代理和适配器等.协同模块与逻辑控制模块相互交互并负责在不同的展示模块之间建立同步的连接.每个展示模块要向协同模块注册,每当应用状态发生变化时,逻辑控制模块会通知协同模块,这样,所有注册的用户展示模块就可以立即得到更新.我们之所以把这个协同模块单独从逻辑控制模块中划分出来,是考虑到一方面会减轻应用设计者的负担,另一方面,往往需要在不同的逻辑空间中进行迁移,特别是在复制迁移的情况下,需要额外的协同部件对此提供支持.快照管理模块主要提供永存支持,在迁移前后将应用的状态保存下来并提供给移动代理.而移动代理负责应用组件的迁移工作.由于目的空间往往与源空间有差别,因此采用适配器来进行必要的转换工作.这一层由于不是直接与用户交互,所以对用户来说

是透明的.

2.1.2 动态交互

用户首先应将应用的基本数据、接口描述、资源使用等元信息以及其他参数,包括设备需求、使用偏好,通过资源管理部件注册到相应的数据库中.在应用运行过程中,当移动代理接到自治代理传来的迁移信息之后,首先进行选择判断,如果是跟踪迁移,移动代理就与注册中心联系是否目的空间存在相应的组件和相容资源,将结果反馈给自治代理,然后程序执行挂起,记录状态信息,等待自治代理的决策结果.当收到迁移具体组件的消息之后,移动代理将该组件和相应的状态快照信息迁移至指定的空间,利用反射机制从断点处恢复执行.由于目的空间的某些特性与源空间相异,因此可能要这种执行可能要经过一个调整过程.这一调整过程由适配器拦截(intercept)实施.这一过程可由图4进行描述.而对于复制迁移,其区别在于,在移动前后有两个或多个进程在并行执行,由上文介绍的协同模块进行同步.

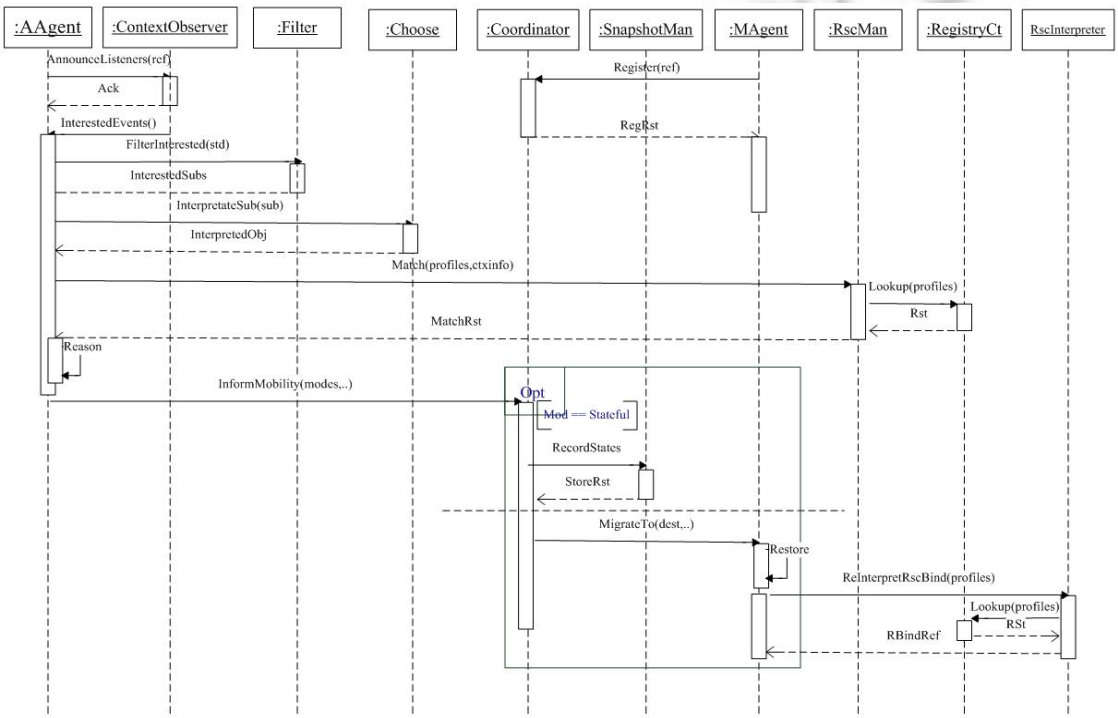


Fig.4 Application management interaction diagram

图 4 应用管理交互图

2.2 代理管理

本节是对图 2 列出的代理管理中的一些关键模块的进一步阐释.在我们的结构模型中,代理起到了类似于连线机制的作用,将应用层和情境层联系起来.特别地,在我们的设计中区分了两种不同特点的软件代理,分别为自治代理和移动代理.

自治代理受代理层的相应管理模块的管理,逻辑上存在于该层,但由于与情境层交互较多,物理上可以认为存在于情境层.当情境监听器发现情境的变化时,会以事件形式通知自治 Agent.由于情境信息的多样化,自治Agent要对这些信息进行过滤和融合,将其转换成有用的信息.举例来说,当情境监听器发现用户的位置由位置1变为位置 2 时,立刻发出事件通知,自治 Agent 捕捉之后,根据具体应用场景进行解释,例如用户离开卧室去了客厅,并将这一消息通知到协同模块进行状态记录等.自治代理的管理主要包括提供通信、协同以及规则管理等内容.

移动代理受代理层的相应管理模块的管理,逻辑上存在于该层,但由于与应用层交互较多,物理上可以认为存在于应用层.当其接到移动指令后,要连同状态信息和组件迁移到指定的目的空间.移动代理与应用组件的绑定不是静态确定的,但是要求组件满足可序列化基本要求.移动代理的管理主要包括移动管理、容错处理、生命周期管理、通信协同这些内容.

2.3 资源描述和推理机制

资源的描述匹配以及自治代理的推理决策机制是 MDAgent 设计中的重要方面.在普适计算环境下存在着密集的资源,这些资源有着不同的属性,有些可以移动,有些可以被替换.比如,打印机不可随用户移动,但却可以被目的空间的其他打印机所替换;PDA 是可以随身移动的,但是由于 PDA 携有用户数据,不易被替换;而一个数据库管理系统则既不易移动也不能被替换.为了合理利用这些资源,需要有一种机制能够较为全面地描述资源的属性,并对匹配推理提供一定的帮助.基于以上的分析,我们采用本体对资源进行建模描述.

本体原为哲学上的一个概念,是指关于存在的本质.在知识工程领域,本体是指某一共享论述域(a shared domain of discourse)的代表性词汇的描述规约[12];应用到软件工程领域,则可以指有关类、类之间关系以及对象等相关的形式概念定义的集合.

为了支持本体描述,需要采用一定的工具支持.网络本体描述语言(Web ontology language,简称OWL)[13]是由W3C组织拟订的用于发布和共享本体的语义标记语言,由当时的资源描述框架(resource description framework,简称RDF)标准发展而来.OWL遵循XML的语法标准,有着与平台无关的优势.与此同时,它也是目前在语义网络领域使用得最为广泛的本体描述语言.在MDAgent中,我们采用OWL作为本体描述工具进行资源的描述.例如,我们可以如下描述一台彩色打印机,表示它是属于打印机类型、可被替换而不能迁移、位于会议室之中,而且‘位于’这个性质是可传递的.该示例描述如图 5 所示.

```
〈owl:Class rdf:ID=“HPLaserPrinter”〉
    〈rdfs:comment〉Hp Color Laser Printer〈/rfds:comment〉
    〈rdfs:subClassOf rdf:resource=“#Printer;Substitutable;UnTransferable”/〉
    〈owl:ObjectProperty rdf:ID=“locatedIn”〉
        〈rdfs:range rdf:resource=“#ConferenceVenue”/〉
        〈rdfs:type rdf:resource=“TransitiveProperty”/〉
    〈/owl:ObjectProperty〉
    …
〈/owl:Class〉
```

Fig.5 OWL resource description illustration

图 5 OWL 资源描述示例

通过抽象和规约资源的一些关键特性,我们就可以用工具自动地进行语义级别的相容性检查.软件代理通过 OWL 查询语言向注册中心查询资源信息,根据反馈的查询结果的其他必要情境信息以及应用相关的逻辑规则进行相应的推理.逻辑规则可用 RDF 规格进行描述.图 6 给出了这种表示的一组规则的示例.

```
[Rule 1: (?p imcl:locatedIn ?q),(?q imcl:locatedIn ?t)→(?p imcl:locatedIn ?t)]
[Rule 2: (?ptr imcl:printerObj ‘printer’),(?srcRsc rdf:type ?ptr),(?destRsc imcl:printerObj ?ptr)→
    (?srcRsc imcl:compatible ?destRsc)]
[Rule 3: (?addr1 imcl:address ?value1),(?addr2 imcl:address ?value2),(?srcRsc imcl:compatible ?destRsc),
    (?n imcl:responseTime ?t),lessThan(?t, ‘1000’^^xsd:double)→(?action imcl:actName “move”),
    (?action imcl:srcAddress ?add1),(?action imcl:destAddress ?add2)]
…
```

Fig.6 RDF rule illustration

图 6 RDF 规则示例

以上示例脚本定义了一系列规则:“locatedIn”是一个具有传递属性的谓词;当源空间和目的空间都有 printer 类型资源时,其资源相容,而不必关心它们具体是哪一种打印机;当源和目的空间的资源相容且网络情况良好(延迟小于 1 秒)时,可以发出迁移消息.

3 实现与性能分析

为了证明MDAgent框架模型的可行性,我们实现了一个原型系统.原型系统用J2sdk1.4+Eclipse3.2 来实现,采用JADE3.4[14]作为软件代理容器.为采集用户的位置、身份标识等情境信息,我们部署了 20+左右的Cricket传感器[15],传感层的部件用C语言编写.该原型系统包含了一个情境管理内核、自治代理管理器和移动代理管理器、一个资源解释及管理器以及若干抽象应用接口供上层应用开发之用.自治代理和移动代理都是由JADE平台提供的Agent类继承而来.为了辅助自治代理的规则推理,我们采用Jena[16]作为推理引擎嵌入到自治代理之中,用以处理应用相关的规则.后台采用Juddi和MySql提供应用的注册和查询服务.

基于 MDAgent 平台,我们开发了若干示例应用,包括智能音乐播放器、随身编辑器、手持编辑器、普及幻灯演示应用等.限于篇幅,本文以具有代表性的智能随身音乐播放器为例来展示迁移效果.它在感知到用户离开的情况下自动终止音乐播放并保存当前的状态.这些状态主要有播放列表、音量大小、界面主体风格、所播放曲目当前位置.当感知到用户到达新的空间之后,会由代理根据自动绑定相关用户接口组件以及状态信息迁移到目的空间,并从断点处恢复播放.结合这个应用我们做了性能分析,测量了相应的时间消耗;同时,为了表明代理动态绑定组件的高效性,我们还给出了相对于前期工作静态应用绑定的性能比较.

3.1 度量参数

性能实验主要考虑时间开销,从以下 3 个方面展开:

Suspend:代理接到移动消息,到移动前的状态保存等工作所需要的时间;

Migrate:代理迁移至目的空间所需要的时间;

Resume:代理到达之后应用从断点处恢复执行所需要的时间.

由于 Suspend 的时间和 Resume 的时间是在同一个空间发生的,所以度量起来比较容易.由于 Migrate 往往会牵涉到两个空间,这会引入时间的同步误差.但是观察到石英晶振频率的恒定性,即 $T@S1-T@S2=C$,($T@S1$, $T@S2$ 分别表示时刻 T 在空间 $S1$ 和空间 $S2$ 处的时间值,C 是常数),在实验时通过计算来回的迁移时间,叠加之后取其平均,可以抵消由于时钟异步问题带来的误差.用公式来表达就是

$$(T2@S2-T1@S1+T4@S1-T3@S2)/2=(T2@S2-T1@S2+T4@S1-T3@S1)/2.$$

3.2 性能比较

在实验中,我们采用大小不同的音乐文件对 MDAgent 的性能进行了度量.不失一般性,音乐文件由 URI 进行定位,这样,应用组件在其移动的过程中不必关注资源文件的具体位置,而其定位由相应的资源解释和管理模块进行解释绑定,这样,在迁移时该资源文件无须随之移动.音乐播放组件逻辑部分含有线程类,不能序列化,这在应用配置文件中标明;用户界面主题可被序列化,在目的空间的注册服务器上列有相应的逻辑组件,这样,在自治代理的推理决策过程中只会选择移动状态信息.

源空间包含有一台 Pentium D 2.8GHz CPU,1G 内存的台式计算机,目的空间有一台 Pentium Mobile 1.8GHz,512M 内存的笔记本电脑,分别由 100Mbps 以太网和 11Mbps 802.11B 连接至同一局域网.我们选取了大小不同的 8 个文件进行测试.图 7 显示,在音乐文件大小从 2M 到 8.6M 的增长过程中,相应的时间消耗只有小幅度的上扬.在实验过程中可以发现,Resume 的时间要高于 Suspend 的时间,这是因为当 Suspend 时仅仅需要序列化当前的曲目状态,包括播放位置、音量大小、是否循环等信息;而在到达目的空间进行 Resume 的过程中,则首先要重新解释绑定音乐文件,定其位置,然后要启动音乐播放器组件,最后还要恢复成原来的状态,因此,这一过程要花费比 Suspend 更多的时间.

为了给出与静态应用绑定对比的性能分析,我们在同一环境下测量了应用级别绑定的时间消耗.该绑定将资源文件、图形界面以及应用逻辑由移动代理一起迁移到目的空间.结果如图 8 所示.可以看出,静态应用绑定所耗费的时间比动态绑定要多一个数量级;而且随着文件体积变大,相应的时间消耗增长迅速,性能迅速下滑.图 9 给出了两者时间消耗的总和的对照.从静态的应用绑定过程中可以看出,在资源文件较小的情况下,其 Resume 时间相对较少,这是因为此时所有的文件都已经保存在本地空间,于是就避免了网络播放的延迟.然而,

随着文件体积的增大,其 Resume 的时间逐渐超过动态资源绑定时相应的 Resume 的时间,这是因为随着体积的增大,传递过来的数据恢复成原文件格式的开销超出了由在本地播放带来的时延降低的幅度,故而总时延增大.

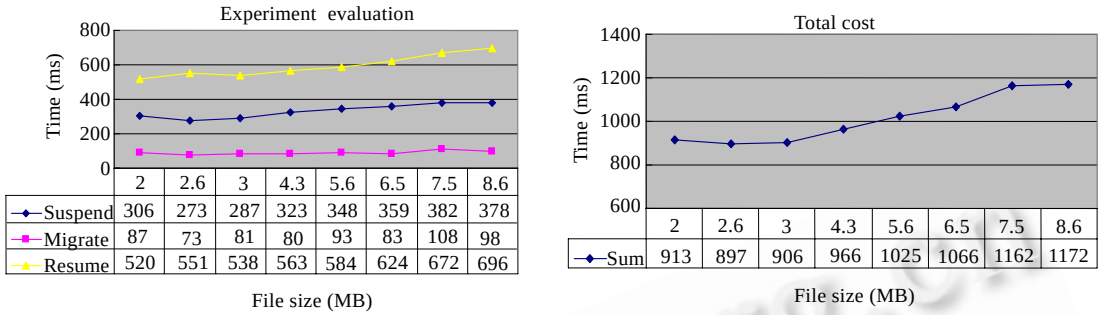


Fig.7 Experimental results

图 7 实验结果

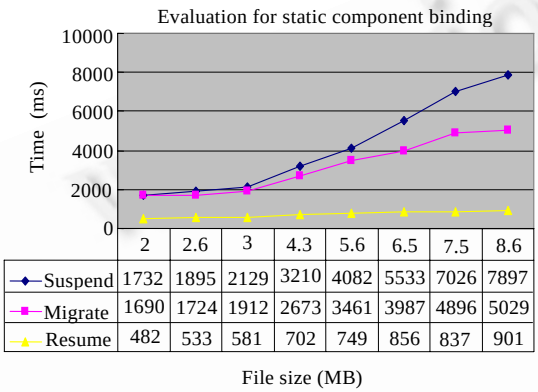


Fig.8 Performance with static application bindings

图 8 静态应用绑定性能

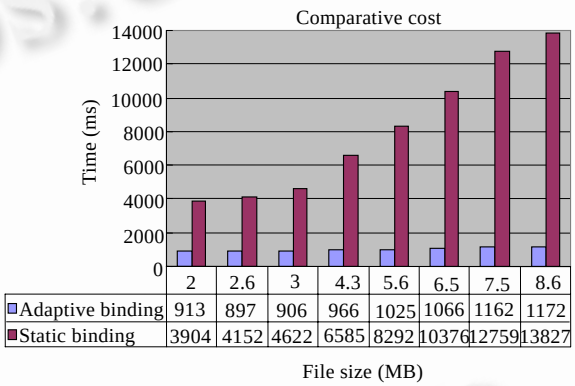


Fig.9 Comparative time cost

图 9 时间消耗比照

4 主要相关工作

应用迁移的研究工作源于网络环境下用户对个性化以及具备移动性和自适应能力的应用的需求,因此,对此的研究在移动计算以及在提倡以人为本的普适计算领域受到越来越多的重视.

Gaia[17]是UIUC开发的一个在普适环境下支持应用迁移的中间件系统.应用程序结构分为基层和元层两个层次,采用反射机制将两者联系起来.基层主要由应用逻辑构成,包括模型、控制、视图等组件以及必要的传感数据采集模块;元层次关注应用底层的元数据,例如资源配置、系统状态等,主要由一个协同模块构成.在应用迁移过程中,由元层的协同模块负责组件的注册、生命周期以及移动的管理.元层与基层的分离在一定程度上缓解了应用开发者的负担,然而,将移动、协同以及生命周期管理等诸多要素集中于静态协同模块,不可避免地会增加其设计复杂度,而且在动态网络环境中也会引入单一失效点的弊病.

NetChaser[18]是一种基于代理的支持用户在移动的情况下可以透明地访问网络服务的框架结构.通过在服务器方增加由多个互相合作的代理组成的中间层,用户在移动到另一个工作环境后,由移动代理将会话相关数据和属性随之进行迁移,以保持网络会话的连续性.然而,用户在移动到新的环境后,需登陆NetChaser系统以通知原相关联的代理该用户已到达新的环境.首先,从情境计算角度来说,这种感知方式对用户而言并非透明;其次,由于该系统对网络虚拟资源缺少统一的表达方式,只是对用户的会话有一个基本的描述,对于网络的动态性、平台的多样化等情景信息缺少调整机制,故而仅提供有限的虚拟化支持.

SpatialAgent框架[19]为支持用户或设备的物理移动(physical mobility)提出了基于代理的逻辑移动(logic

mobility)的概念.该框架由两大模块组成,分别是位置信息服务和移动代理.由位置信息服务捕捉情景的变化,支持移动代理的两种迁移方式:一种是跟随用户的代理,当用户从一个网络空间移动到另一个网络空间时,相应的移动代理会随之迁移到相应的服务节点上;另一种是固定在一个网络空间的代理,当用户到达该网络空间时,该代理会迁移到用户的计算设备上.SpatialAgent框架缺少一个统一的松散耦合应用模型,在具体的上层开发过程中,往往整个应用实现为一个移动代理.这样固然简化了移动的管理,但是相应的网络开销代价较大.同时,由于缺少对虚拟资源统一的描述机制,因而在迁移后,对资源重绑定和适应性调整支持力度有限.

相对于前期关于应用迁移的研究工作[9],我们进一步对应用结构模型、移动管理展开研究并根据软件代理的不同角色对其加以区分,赋之以不同的任务.前期工作采用了应用与软件代理的静态绑定,这不适应于动态、多变的网络环境,同时也会带来不必要的网络传输开销.因此,我们借助于一种松耦合体系结构模型,利用代理动态绑定应用组件,根据上下文感知信息自主迁移.前期的工作侧重于对跟踪迁移(follow-me)的支持,这是一种移动的方式,而在实践过程中我们发现,也存在相当一部分的复制迁移(clone-dispatch)需求.在前期研究工作中,上下文推理部分与移动管理部分组织在一个软件代理之中,这不利于系统的管理,同时也增加了移动的开销.

在虚拟计算环境方面,卢锡城、王怀民等人提出了以网络资源的按需聚合和自主协同为核心建立虚拟计算环境的思路.其目的是在开放的网络基础设施之上,为终端用户或应用系统提供和谐、可信、透明的一体化服务[5].怀进鹏、胡春明等人开发的CROWN系统可为开放网络上计算密集和数据密集的应用提供良好的支持[3].我们相信,本文的工作可为在这些工作的基础上进一步支持移动用户的应用迁移提供一种可能的技术途径.

5 结束语

本文讨论了软件代理对普适计算环境下的虚拟化应用迁移的支持,提出一个中间件结构模型 MDAgent,实现了一个原型系统,并在此基础上开发了若干应用.通过相关的性能测试,初步证明该模型是可行而有效的.

与其他支持应用的迁移系统相比,MDAgent 的特点在于以下几个方面:采用软件代理结合传感器技术支持应用的跟踪、复制迁移以及灵活的组件绑定和跨空间迁移等多种迁移模式;基于角色的不同将自治代理与移动代理分开,各司其职,在一定程度上简化了对这两种代理的关注管理,同时减轻了网络传输负担;基于 Ontology 的资源描述机制,从语义级别对虚拟计算环境的资源进行描述,既避免了肤浅的基于语法资源的匹配方式,支持更丰富的信息表达和处理,同时也对规则推理提供了良好的支持.

更进一步地,MDAgent 的上层应用在跟踪迁移方面目前仅支持单个用户,当有多个用户同时存在时,如何区分不同的用户所属的应用并进行相应的虚拟化迁移管理,是我们下一步的研究工作.

致谢 在此,我们向对本文的工作给予支持和建议的同行,尤其是南京大学软件研究所的陶先平教授,香港理工大学计算机学系的邹洋、刘阳、Vaskar Raychoudhury、Joanna Siebert 等老师和同学以及审稿专家表示感谢.

References:

[1] Xu GY, Shi YC, Xie WK. Pervasive computing. Chinese Journal of Computers, 2003,26(9):1042−1050 (in Chinese with English abstract).

[2] Adabala S, Chadha V, Chawla P, Figueriedo R, Fortes J. From virtualized resources to virtual computing grids: The In-VIGO system. Future Generation Computer Systems, 2005,21(6):896–909.

[3] Huai JP, Hu CM, Li JX, Sun HL, Wo TY. CROWN: Service oriented grid middleware and trust management. Science in China (Series E), 2006,36(10):1127−1155 (in Chinese with English abstract).

[4] Foster I. Service-Oriented science. Science, 2005,308(5723):814–817.

[5] Lu XC, Wang HM, Wang J. Virtualized computing environment iVCE: Concept and architecture. Science in China (Series E), 2006,36(10):1081−1099 (in Chinese with English abstract).

[6] Tao XP. Research on Internet based mobile Agent technology and application [Ph.D. Thesis]. Nanjing: Nanjing University, 2001 (in Chinese with English abstract).

[7] Franklin S, Graesser, A. Is it an Agent, or just a program?: A taxonomy for autonomous Agents. In: Muller JP, Wooldridge MJ,

Jennings NR, eds. Proc. of the 3rd Int'l Workshop on Agent Theories, Architectures, and Languages. London: Springer-Verlag, 1996. 21−35.

[8] Cao JN, Tse CK, Chan TS. PDAgent: A platform for developing and deploying mobile Agent enabled applications for wireless devices. In: Eigenmann R, ed. Proc. of the 2004 Int'l Conf. on Parallel Processing. Montreal: IEEE, 2004. 510−517.

[9] Yu P, Cao JN, Wen WD, Lu J. Mobile Agent enabled application mobility for pervasive computing. In: Ma JH, Jin H, Yang LT, Tsai JP, eds. Proc. of the 3rd Int'l Conf. on Ubiquitous Intelligence and Computing. Berlin: Springer-Verlag, 2006. 648−657.

[10] Abowd GD, Dey AK. Towards a Better Understanding of Context and Context-Awareness. London: Springer-Verlag, 1999. 304−307.

[11] Gamma E, Helm R, Johnson R, Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley Longman Publishing Co., 1995.

[12] Gruber TR. A translation approach to portable ontology specifications. Knowledge Acquisition, 1993,5(2):199−220.

[13] Antoniou G, Harmelen F. Web ontology language: OWL. In: Staab S, Studer R, eds. Int'l Handbooks on Information Systems. Heidelberg: Springer-Verlag, 2003. 76−92.

[14] Bellifemine F, Caire G, Trucco T, Rimassa G. JADE Programmer's Guide. 2006. http://jade.tilab.com/doc/programmersguide.pdf

[15] Privantha NB, Chakraborty A, Balakrishnan H. The cricket location-support system. In: Pickholtz R, Das SK, Caceres R, Garcia-Luna-Aceves JJ, eds. Proc. of the 6th Annual Int'l Conf. on Mobile Computing and Networking. Boston: ACM, 2000. 32−43.

[16] Reynolds D. Jena 2 Inference Support. 2003. http://jena.sourceforge.net/inference

[17] Ranganathan A, Chetan S, Campbell R. Mobile polymorphic applications in ubiquitous computing environments. In: Proc. of the 1st Annual Int'l Conf. on Mobile and Ubiquitous Systems: Networking and Services. Boston: IEEE, 2004. 402−411.

[18] Stefano AD, Santoro C. NetChaser: Agent support for personal mobility. IEEE Internet Computing, 2000,4(2):74−79.

[19] Satoh I. Physical mobility and logical mobility in ubiquitous computing environments. In: Suri N, ed. Proc. of the Mobile Agents: 6th Int'l Conf. London: Springer-Verlag, 2002. 186−201.

附中文参考文献:

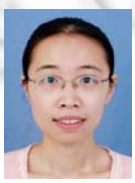
[1] 徐光祐,史元春,谢伟凯.普适计算.计算机学报,2003,26(9):1042−1050.

[3] 怀进鹏,胡春明,李建欣,孙海龙,沃天宇.CROWN:面向服务的网格中间件系统与信任管理.中国科学(E 辑),2006,36(10):1127−1155.

[5] 卢锡城,王怀民,王戟.虚拟计算环境 iVCE:概念与体系结构.中国科学(E 辑),2006,36(10):1081−1099.

[6] 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.

周宇(1981－),男,江苏徐州人,博士生,主要研究领域为软件代理,软件体系结构,普适计算.



余萍(1979－),女,博士生,主要研究领域为软件代理,软件体系结.

马晓星(1975－),男,博士,副教授,主要研究领域为面向对象技术,软件体系结构,分布式计算.

吕建(1960－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为分布式计算,面向对象技术,Internet 软件技术.

曹建农(1960－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为分布式并行计算,网络技术,移动及普适计算.