计算机科学 2003Vol. 30№. 1

一个适用于遍在计算环境的中间件体系结构框架

张向刚　张云勇　刘锦德

（电子科技大学计算机科学与工程学院微机所　成都 610054）

A Framework Based on Middleware for Pervasive Computing Environment

ZHANG Xiang-Gang　ZHANG Yun-Yong　LIU Jin-De

(College of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054)

Abstract　Pervasive computing has many new features, such as: the wideness related area, mobility of tasks and heterogeneity of devices. How to support user's tasks in the pervasive computing environment is a challenge for engineers. The main idea in the article is that using middleware to mask the heterogeneity of devices and support user's tasks in a uniform platform. The goal of the middleware is to support mobility of task, to make the better use of resources, and to reduce user's intervene. The detail way is as follow: first, decompose application into several components, then represent user's tasks into abstract tasks and encapsulate physical service into virtual service; at last, abstract task is completed by cooperation of several virtual services. A virtual service is mapped into different physical service in different environment. In different environment, the same task could be completed by composing different services. Furthermore, the framework of the middleware has some features, such as: hierarchy of architecture, extensibility of functions.

Keywords　Pervasive computing, Middleware, Mobility, Abstract task, Virtual service, Component, Framework

1. 背景

随着计算技术和通信技术的发展，不仅桌式计算机变得非常普遍，非 PC 设备正以惊人的速度成为市场的主流，如：机顶盒、移动电话、PDA、家用电器、汽车电子、游戏机等等。特别是随着家庭网络和无线网络的发展，网络变得更加的普遍。计算已经不只限于桌面，它渗透到人们生活和工作的各个环节，这就是所谓的无处不在的计算环境[1](pervasive computing，简称遍在计算[1] 环境)。在这样的环境中，人们希望：即使他们在频繁地移动，或者各种资源在动态地加入或离开，他们都能够在任何地点、任何时间进入和控制信息，并且尽量减少自己所做的人为干预；特别是对于一些移动频率较高，所从事的工作具有较强的持续性的用户，希望自己在各种环境中移动的时候，仍能够继续工作和充分利用当地的资源。

遍在计算的基本思想是在我们的工作空间和生活空间中，存在各式各样的设备，其中内置计算机，这些设备互相协作，共同向用户提供对信息的统一且及时的进入，共同支持用户完成他们的工作。传统的计算具有如下的特征：

1)桌面计算。人们主要运用桌面 PC 来完成他们的工作。

2)私有设备和私有软件。软件针对特定设备和特定平台。

3)单块应用。应用程序被编写成内部组织紧密的一整块软件，应用界面主要是与用户的接口。

4)计算任务到应用的人工映射。人们必须跟踪如何使用应用程序，并且配置它们进行相应的任务。(任务是一个比应用程序更高的抽象，任务描述完成什么任务，而不是如何完成。)

5)使用较少的计算设备。一个计算任务，只涉及几个有限的设备。

6)手工配置。配置应用程序是用户或设计人员的工作。

遍在计算环境下的计算特征：

1)计算的高度动态性。由于计算的无所不在，环境的千差万变，用户有可能在移动，设备和服务有可能随时增加或减少，计算需要在各种环境下正确且一致地完成。

2)使用更多的计算设备。一个计算任务可能涉及较多的计算设备。

3)单块应用变成了众多协同服务的联合。由于系统体系结构从一整块软件到众多组件的联合，组件不仅要求和用户进行交互，而且需要组件之间的交互。

4)动态的任务到服务的映射。由于不同的环境有不同的计算设备，在不同的环境下完成相同的任务，可以由不同的计算设备联合完成。

5)多个计算设备能更好地协调工作。随着一些新的工业标准，如：Jini[2]，UpnP 的出现，计算设备能够自动地相互连接，融合为一体。

今天实现这种计算模式的硬件和通信结构正在逐渐成为现实，然而却很少有应用程序运行于这样的环境中，这主要因为在遍在计算的环境下，设计、建立和配置应用程序比较困难。今天的计算体系结构框架并不支持这样的计算模式[4]，如果一个用户希望使用新环境下的计算资源，他必须进行手工的修改工作配置。这样的手工修改在遍在计算的环境下是不可接受的。因为遍在计算意味着巨大的、分布较广的网络，在这个网络中存在数以千万计的设备和服务，而且这些设备和

1　遍在计算，英文为 pervasive computing 或 ubiquitous computing，意思为无处不在的计算，国内文章一般译普及计算，但“普及”一词易与推广基础知识的意识相混淆，所以用“遍在计算”一词来表达计算的无处不在更为贴切。

张向刚　博士生，主要研究领域为：中间件与遍在计算。张云勇　博士生，主要研究领域为：中间件与 agent 技术。刘锦德　教授，博士导师，主要研究领域为：开放系统与中间件技术。

服务是自治的，它们可以完全自主地加入和离开，特别是用户移动的时候，周围的设备和服务将会有较大的变动，如果完全由人工完成配置和转换工作将是不可接受的。

为此本文提出一种适用于遍在计算环境的中间件方案，其核心思想是：在遍在计算环境中引入一个中间件体系结构框架，屏蔽下层设备的异构性，管理低层的事务，提供应用程序在遍在计算环境下移动所需要的基础设施，让用户任务集中于他们所关心的商务逻辑。该中间件框架应该具有如下的基本功能：

1）支持用户任务的移动，支持设备和服务动态的加入和离去；

2）能自动利用当地和远地的资源，进行最佳组合完成用户任务。

3）在环境的改变中尽量减少用户的干预。

2. 一个支持遍在计算的中间件体系结构框架

在遍在计算的环境下，开发应用程序的主要挑战是：不论设备或人员的移动，还是服务的增减，应用程序都应该向用户提供持续、有效的服务。所以遍在计算环境下的中间件体系结构框架应该支持任务很方便地从一个环境到另一个环境的转移，而且允许用户充分利用本地资源（而不是利用远端的资源），同时应该尽量减少用户的干预。不论用户的移动，还是资源的增减，中间件都应该屏蔽这些变化，给用户提供平滑的服务，屏蔽下层的多样性，向应用提供统一一致的界面，让应用程序集中于用户的商务逻辑。与以上要求相适应的体系结构框架如图 1 所示。

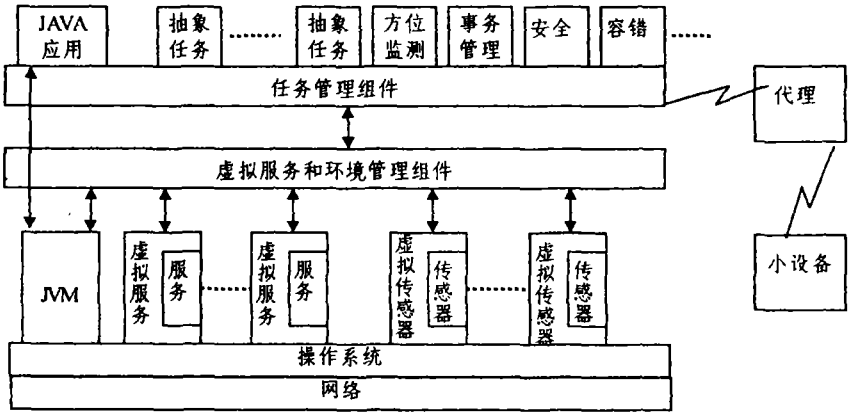

图 1　体系结构框架

3. 主要理念和思想

1）*抽象任务*[3]　表达了用户的计算意图，一个任务可以包含一个行为，也可以包含多个行为，它应该是平台独立和应用独立的。所谓应用独立是指任务描述的是做什么，与用什么软件完成没有什么关系。所谓平台独立性是指与下层的操作平台无关。

2）*虚拟服务*　一个虚拟服务表达了一个抽象的功能，如显示功能和接受输入功能，这个抽象的功能通过虚拟服务的接口来定义。为了激活一个虚拟服务，必须使用一些在实际环境存在的物理服务来完成。所谓一个物理服务代表了某个计算设备和它上面运行的系统。多个虚拟服务可以对应一个物理服务，例如：虚拟服务拼写检查和虚拟服务文本编辑可以对应一个物理服务 Microsoft Word，同样一个虚拟服务可以对应多个物理服务，例如：邮件服务可以对应 PC 上的 Outlook Express 和移动电话上的短消息服务。

在运行时，一个任务被表达成一个服务集合，一个服务集合代表一组协同工作的虚拟服务，同时每个虚拟服务在相应的环境中对应成一个物理服务。这些服务相互合作一起来完成用户的任务。在具体环境中，一个任务通过相应的物理服务相互合作来共同完成，一个任务在不同的环境中可以通过不同的物理服务相互作用来达到同一个目的。一个物理服务可以根据需要加入或离去。

3）*新的支持移动计算的方式*[4]　传统的支持移动计算的方案分为四种：第一种，移动设备上已经包含了所有的硬件、软件和信息；第二种，瘦客户通过网络与远端的服务器相连；第三中，具有标准接口的移动设备在标准的体系结构框架间移动；第四种，在各个环境中都配置一个统一的虚拟层，如 JVM[5]，来支持相应的代码移动。

这些方案都假设环境之间存在一定程度的同质，同时缺乏动态管理资源的能力，不能充分运用环境中资源。本文中的体系结构通过抽象任务，将用户的工作表达成独立于平台和独立于应用的任务，并且通过任务管理组件和虚拟服务/环境管理组件的配合，以及将物理服务包装成虚拟服务，以此来共同支持用户的移动；当用户移动到一个新的环境之后，系统根据用户的计算意图和环境中存在的服务重新组合服务，完成用户的任务。这正是我们既支持动态计算，又充分利用当地资源的思想所在。

4）*数据和功能的分离*[6]　在传统的分布式系统中，我们往往将数据和功能统一为对象。但是，这种数据和功能的统一并不一定适合于遍在计算。

首先，将对象作为一种封装机制，主要基于两个假设：第一，实现和数据比对象的界面变化更频繁；第二，能设计基于不同实现的界面，随着系统的升级，界面保持相对的稳定是可能的。

然而，这些假设在遍在计算环境中不一定能够成立。首先，在遍在计算的环境中由于用户的移动和服务的增减，完成同一任务的服务可能在不断地变化，但工作的内容却是相对稳定的，这时如果将实现和数据分离，将有利于用不同的服务完成同一个任务。第二，从系统安全性考虑，进入被动的数据比进入主动的对象更加容易。进入主动对象将带来严重的安全问题，从资源控制的容易度来看，进入被动的数据更加容易，这可以被当前的进入规则的 HTML 文档和 PDF 文档工作得很好来得到证明。

5)应用程序从单块的软件转变为众多组件的联合　运用组件技术不仅会在应用程序的生产、维护和管理等方面带来革命性的进步，而且将应用分解为众多组件的联合，也是在新的环境中重新组合任务的必要条件。

4. 框架构成及相关功能

中间件框架主要包含两个隔离组件：①任务管理组件，用来提供统一的界面，体现个人环境的概念；②虚拟服务和环境管理组件，用来隔离下层设备的多样性，管理虚拟服务，体现到环境的网关；以及两个必要的抽象任务：①方位检测，用来检测和跟踪移动用户和设备；②事务管理，用来保证某些动作的原子性；最后是虚拟服务提供组件：用来提供虚拟服务。

1)任务管理器　体现个人环境的概念，努力减少用户的干预。当用户移动到一个新的环境，任务管理器协调与用户任务相关的信息迁移，并且与新环境管理组件磋商以提供任务支持；任务管理器监视支持用户任务的组件的服务质量，一旦相应组件消失，或无法满足用户任务的需要，任务管理器询问虚拟服务和环境管理组件去寻找一个替代服务；任务管理器监视用户的指示，一旦得知用户将结束当前任务，切换到新任务，任务管理器保存当前状态，且相应地激活新的任务；用户任务的执行对上下文有一定的要求，一旦因为什么原因，要求不能得到满足，任务管理器协调暂停当前任务或上下文的相关部分。

2)虚拟服务组件　提供虚拟服务的概念，不同的虚拟服务组件提供不同的功能；一个得到虚拟服务组件的方法是包装现有的任务，如对 vi，Microsoft word 或 Notepad 进行包装，都能成为提供文本编辑功能的虚拟服务组件。包装的这层软件主要提供抽象服务描述到特定应用的映射。

3)方位检测组件　提供与物理上下文相关的信息报告，监测和跟踪移动用户和设备，方位监测组件可以有不同的复杂程度，越复杂，在关于用户意图方面，任务管理器越少依赖于用户的明确指示。

4)虚拟服务和环境管理组件　它体现了到环境的接口，它意识到在当前的环境中存在什么服务，它们能被配置在哪里，当一个虚拟服务安装到一个环境中，它首先向当地的环境管理器注册，当需要在一个新的环境中激活一个任务时，运用相应的发现机制去查询注册，寻求响应的服务，发现机制可利用现有的技术。如：Jini，名字服务等[2]，如果发现多个可利用服务，环境管理组件将进行评估，以选择一个最能满足用户要求的服务。它也为分布文件进入提供封装机制。

5)连接器　所有的组件类型都有标准的接口，接口之间的互连是通过连接器来完成。各个连接器根据它们连接的组件运用不同的交互协议。每一种连接器，对于不同的低层通信机制和所连接组件的分布情况有不同的实现。例如，若连接双方的组件在同一地址空间，则连接器的实现只是一个方法调用，若连接的组件分布在地理上不同的位置，则连接器的实现包含一定的远程通信机制，如：RPC。

6)事务管理　在遍在计算的环境中，一些动作必须在一个环境中全部完成，事务管理提供这种保障。

7)小设备的接入　某些小设备由于自身有限的计算能力，不能具有上述的体系结构框架，同时又移动到了一个不具有上述环境的空间，如何通过小设备继续向用户提供连续的服务？有两种小设备接入的方式：第一，将小设备作为整个应用的众多组合组件的一个，通过远程通信协议，加入到组件联合中；第二，将任务描述传递给具有计算能力的环境，之后相应的环境向小设备返回信息。

8)对移动代码的支持　主要通过任务管理组件感觉到用户任务是通过移动代码完成，所以并不在新的环境中重新组合任务，而是通过环境管理组件直接将移动代码置于相应的 JVM 之上。

9)附加组件　是根据任务的性质来选择组件，如抽象安全任务，可以运用新环境中的虚拟服务向用户任务提供安全服务。抽象容错任务可以将一个虚拟服务对应为多个物理服务，从而向用户任务提供容错保证。

5. 框架效用描述

一个会议讲演者正在家里进行即将召开的会议的演讲准备，但由于时间的紧迫，他不能全部完成准备工作，需要在汽车中完成最后的编辑工作。在家里的时候，他可能是运用键盘输入和屏幕显示；在汽车上，可能最方便的输入方式变为语音输入。通过以下的工作过程，我们可以看到在具有前述中间件框架的遍在计算环境下，他是如何连续进行工作。其工作过程如图 2 所示。

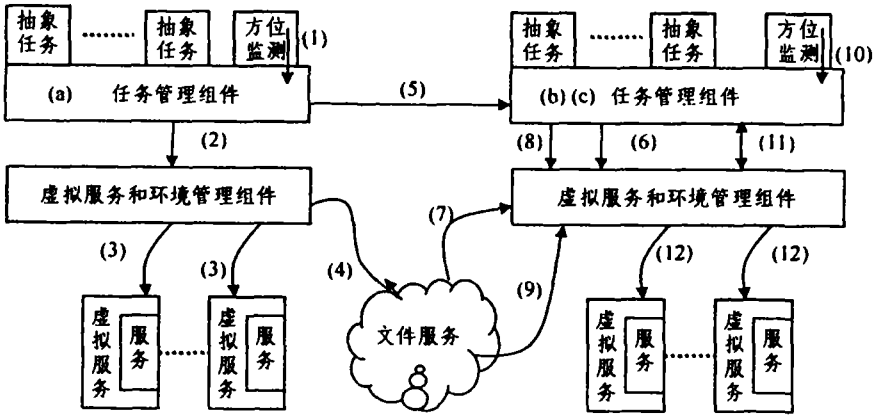

图 2　工作过程示意图

起初会议讲演者在家里进行准备工作。当他准备离开房间，前往办公室的时候，首先是方位监测组件意识到演讲者将要离开房间(这可以通过相应的传感器得到相应信息，或者直接接受用户的命令)，于是方位监测组件让任务管理组件知道演讲者将要离去，如图 2 的(1)所示。这个消息将引起任务管理组件进行状态转换(a)，并且挂起正在家里进行的作业。然

后，任务管理组件要求环境管理组件检查每一个组成任务的服务的状态，保持相应的状态和文件，接着在(3)中环境管理组件回收这些服务，并且保存相应的状态和文件到一个文件服务器(4)。然后，任务管理组件检查演讲者的时间安排，了解到他的行踪动向，然后传送相应的信息(包括抽象任务描述)到汽车的任务管理组件(5)；这个消息将触发汽车的任务管理组件发生相应的状态转移(b)，同时将引起它要求汽车环境管理组件(6)去得到更新的演讲者的工作描述(7)。

通过以上的操作，汽车任务管理组件将了解到那些文件是继续演讲者工作必需的文件，然后任务管理组件将要求汽车的环境管理组件提取这些文件(8)，汽车环境管理组件检查当地的文件是否已经更新，如果没有，则提取这些更新的文件(9)。一旦汽车的方位监测组件了解到演讲者已经进入汽车环境，它就会立即通知任务管理组件(10)，这将引起任务管理组件经历相应的状态转变(c)。然后将向环境管理组件提出要求完成任务的服务支持(11)，接着环境管理组件在相应的分配的虚拟服务组件上恢复执行的状态，然后，工作可以在汽车环境中继续进行了。

小结 遍在计算涉及到一个范围较广的环境，而且环境中的设备，不仅数量众多，而且结构各异。如何在这样的环境中，为用户提供连续、平滑的服务是软件工作者所面临的挑战。本文中所提出的基于中间件的体系结构框架主要包括两个隔离层：虚拟服务和环境管理组件屏蔽下层设备的异构性，任务管理组件向用户提供个人空间的概念。同时将用户任务视为多个组件的联合，在不同的环境中，可以用不同的组件组合完成同一个任务，同时提供与移动代码的兼容和支持小设备的接入。

参考文献

1 Weiser M. The computer for the twenty-first century. Scientific American, 1991, 265(3): 94～104

2 Arnold K, O'Sullivan B, Scheifler R, Waldo J, Wollrath A. The Jini Specification. Addison-Wesley, 1999

3 Wang Z, Garlan D. Task-Driven Computing: [Technical Report, CMU-CS-00-154]. School of Computer Science, Carnegie Mellon University, May 2000

4 Sousa J P, Garlan D. Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. In: Proc. of the 3rd Working IEEE/IFIP Conf. on Software Architecture 2002, Montreal, Aug. 2002

5 Gosling J, Joy B, Steele G. The Java Language Specification. Addison-Wesley, 1996

6 Grimm R, et al. Systems Directions for Pervasive Computing. In: Proc. of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), Elmau, Germany, May 2001. 128～132

(上接第 29 页)

方接收的情况下所得到的结果。由图中可以看到，在小于约 2000 字节大小数据包的情况下，MyVIA 的带宽性能较 BVIA 为低，而在 2000 字节以上，MyVIA 的带宽性能要比 BVIA 高。在 MTU 范围内，BVIA 的带宽最大可达到 47.46Mbytes/sec，而 MyVIA 可达到 76.2Mbytes/sec。

分析在发送 2000 字节下的小数据包时，MyVIA 的带宽性能比 BVIA 低的原因，是由于 MyVIA 实行新的基于状态机的软流水机制，对网卡上各资源进行开锁、解锁的竞争使用方式后，使 MCP 程序的控制逻辑复杂了许多，而网卡上的主频为 33MHz 的 CPU 的处理能力较弱，并且在此时的满负荷发送情况下，由于小数据包的发送较快，网卡上主机-网卡间 DMA 控制器与发送 DMA 控制器没有能够进行并行处理或并行程度不够大，从而使 MyVIA 比 BVIA 在软件控制上的执行延迟大的弱点充分暴露出来。在 2000 字节以上时，网卡上主机-网卡间 DMA 控制器与发送 DMA 控制器的并行处理所带来的好处抵消了 MyVIA 在复杂的软件控制上的引起的延迟，从而使带宽比起 BVIA 得到了进一步的提升。

6. 当前工作总结及今后的任务

Myrinet 网络提供了一个灵活的硬件平台，其网卡的可编程特性使我们可以设计实现自己的各种数据传输方案，并进行实验验证。

我们对 BVIA 的分析及 MyVIA 的设计，是对 VIA 研究的进一步深入。MyVIA 的实现取得了一定的成绩但同时也暴露出不少的问题，需要我们在今后的工作中加以解决。

我们下一步的工作，将在现有的基础上对 MCP 程序进行调整，尤其是对小数据包的处理。并且，我们准备将整个设计方案移植到 Linux2.4 内核下，使用 Myricom 公司的 LANai9.x 网卡，这是其最新的产品，网卡上 CPU 的主频为 200MHz，卡上可用内存 2M，可以进行 64 位 PCI 总线数据的传输处理。

总结 本文介绍了 VIA 的原理，分析了 Berkeley 大学在 Myrinet 网络上对 VIA 的实现，并介绍了我们自己的 MyVIA 实现的主要方面，最后通过 BIA 与 MyVIA 的性能比较分析，指出了我们实现中的不足之处，对我们今后的工作做出指导。

参考文献

1 Virtual Interface Architecture Specification. Version 1.0. Compaq, Intel and Microsoft Corporations, Dec 16, 1997. Available at http://www.viarch.org

2 Buonadonna P, Geweke A. An Implementation and Analysis of the Virtual Interface Architecture. University of California at Berkeley, Computer Science Department, Berkeley, California, May 1998. Available at: http://www.millennium.berkeley.edu/proj/Vineyard/

3 M-VIA: A High Performance Modular VIA for Linux. http://www.nersc.gov/researc h/FTG/via/

4 Banikazemiy M, et al. Effient Virtual Interface Architecture (VIA) Support for the IBM SP Switch-Connected NT Clusters. Available at: http://nowlab.cis.ohio-state.edu/projects/via/

5 Du Zhihui, et al. Hardware Based TH-VIA User Level Communication System Supporting Linux Cluster Connected by Gigabit THNet. DCABES 2001 Proceeding

6 The UC Berkeley Millennium Project. http://www.millennium.berkeley.edu

7 Bhoedjang R A F, Rühl T, Bal H E. User-Level Network Interface Protocols. 0018-9162/98/ IEEE November 1998

8 董春雷，郑纬民. 基于 Myrinet 的用户空间精简协议. 软件学报，1999.3