普及计算中多种代码迁移模式的集成模型研究*)

樊长娥　张申生　韩松乔
(上海交通大学计算机系　上海 200030)

摘　要　通过分析普及计算新环境资源受限、动态多变的特征,以及代码迁移技术和策略机制在普及计算中应用的理论基础和系统框架,提出了以策略组件为核心,包括策略库、感知模块、迁移模块的多种代码迁移模式的集成模型。指出,在多种迁移模式集成的方法中,策略机制具有较强的灵活性和可复用性,并提出运用定量和定性的方式制定策略的选择方法。最后,结合智能病房的实例和一些具体的应用任务,验证了多种代码迁移模式集成与选择优化的可行性。

关键词　代码迁移模式,策略,规则,普及计算

Study on Integration of Multiple Code Migration Paradigms Modeling Technology

FAN Chang-E　ZHANG Shen-Sheng　HAN Song-Qiao
(Department of Computer Science, Shanghai Jiaotong University, Shanghai 200030)

Abstract　Through analysis of new pervasive computing environment with characters of limited resources and dynamic levity and analysis of theory base and system frame of code migration technology and policy applications, this paper advances integration of multiple code migration paradigms model with policy module as core and policy library and so on. It points out policy has more agility and reusability in all the integration methods. It also advances using quantitative and qualitative methods to establish select rules. At one time this paper gives a instance about intelligent sickroom and some idiographic application task, which validate the feasibility of integration and optimization.

Keywords　Code migration paradigms, Policy, Rule, Pervasive computing

1　引言

无线技术的发展推动了移动计算的产生,人们对计算的移动性和便捷性的需求不断增强,以资源受限、用户移动、环境异构、计算分布为特征的普及计算环境逐步形成。普及计算环境对软件而言是一个不断发生着变化(如用户的移动、设备的增减、网络的通断等)的运行环境。然而,传统的软件开发技术和框架主要针对稳定的分布式系统,已不能满足普及计算对软件的新要求。因此,需要一种新型软件技术,它能运行在资源受限的设备上,能感知环境的变化,动态改变自身的结构和行为,从而在没有人干预的情况下更优地为用户提供服务。代码迁移技术被普遍认为适合普及计算新环境的要求。然而,目前大多数项目应用的只是移动代理(mobile agent)这种单一的迁移模式,显然已经无法满足普及计算新环境复杂多变、移动不稳定的要求。

针对单一迁移模式的不足,本文提出了以策略组件为核心的多种代码迁移模式的集成模型,采用策略机制根据环境状态选择最佳交互模式,并给出了制定策略的定量和定性的分析方法。

本文第 2 节介绍背景知识;第 3 节讨论代码迁移模式集成模型及实现方法;第 4 节分析模式选择策略的制定规则;第 5 节是实例验证;最后总结全文。

2　背景

2.1　代码迁移技术及模式

代码迁移技术是在周围运行环境发生变化时,迁移系统自身或其中的一些模块,改变系统或模块所处的位置或模块之间的关系,从而动态适应环境的变化。它具有很强的灵活性和自适应性。代码迁移大多应用在大型分布式系统上,系统中程序对位置敏感,位置信息很可能是程序的参数之一,迁移对于程序员是可见的。代码迁移不仅可以平衡负载,还提供了更大范围的用途,如服务客户化、应用程序的动态扩展、自主性、容错性和对断线操作的支持等。

根据迁移过程中迁移的内容和方向,主要有 3 种代码迁移模式。

①代码需求 COD 模式(Code ON Demand):从其他网络节点寻找完成服务所需要的功能代码,并将其迁移到本地,使用本地资源完成服务。

②远程评估 REV 模式(Remote Evaluation):由于本地资源紧缺或其他原因将功能代码迁移到网络上其他具备资源的节点,异地完成服务,并返回服务结果。

③移动代理 MA 模式(Mobile Agent):完成服务过程中,将运行状态与功能代码迁移到其他网络节点,继续完成服务,并返回结果。

这 3 种模式各有特点:COD 提供更多的灵活性;REV 能够节约带宽,易于实现分布式处理;而 MA 提供自主和迁移管理能力。

由于客户服务器(C/S)模式的广泛应用,本文从代码迁移的角度,将其作为一种没有迁移代码的特殊迁移模式,即零代码迁移模式,与上述 3 种迁移模式一起讨论。

2.2　策略机制

策略是一种声明性的控制系统行为的规则。在学术和工

*)浙江省重大科技攻关项目计划资助项目(2003C11009)、上海市科委科技攻关计划资助项目(03DZ19320)。樊长娥　硕士,研究方向:普及计算、工作流技术、系统集成;张申生　教授、博导,研究方向:CIMS 关键技术、普及计算、分布异构环境信息集成技术等;韩松乔　博士。

业界，策略作为一种约束系统行为的机制越来越流行，它使得网络和系统管理工具具有更灵活和适应性更好的管理框架。策略将控制机制与功能代码分离开来，不仅功能代码可重复使用，固定的策略都可以在合适的场合拿来重用，因此策略机制有很强的灵活性。

在实际应用中，策略应当能被解释和执行，需要有一种高效的语言用于描述具体环境中的策略。Ponder 是一种面向对象的说明性语言，尤其适合描述分布式环境中的安全策略和管理策略。Ponder 可以描述分布式系统中的一般的管理策略，如由事件触发的“条件-相应”规则。

IETF(Internet Engineering Task Force)策略框架工作组所建议的策略采取下面基于规则的形式：

IF {condition(s)} THEN {action(s)}

意思是如果 condition(s) 满足，action(s)将被执行。

3 迁移模式集成模型

3.1 模式集成模型

C/S，COD，REV，MA 每种模式都有各自的使用情况和优缺点。在普及计算环境中的通信过程中，网络节点可能需要同时用几种模式完成与其他节点之间的交互。在某些情况下，还必须改变交互模式，以适应资源或网络等状况的改变。这就需要将几种模式有机地结合起来，并制定合理的规则，就可以在不同的时机灵活地采用不同的模式进行交互。不仅能够更好地完成服务，提高服务效率，还能节约系统资源。根据上面的分析，本文提出的以策略组件为核心的多种迁移模式集成优化模型如图 1 所示。

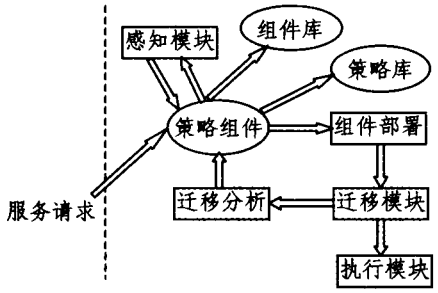

图 1 多种交互模式的集成优化模型

(1)策略组件。是模型的核心，它不仅负责接收服务请求，还要在其他各个模块之间进行组织协调，完成迁移的整个过程，并利用迁移分析模块反馈的信息对策略库的策略进行改进和修正。

(2)感知模块。主要负责感知网络环境及其变化，并将结果返回给策略组件，以做出进一步处理。

(3)组件库。储存系统中所有的功能组件，这些组件可通过选择、部署和组装成为一个应用系统。

(4)策略库。存放的是预先设定的各种迁移模式选择策略。

(5)迁移模块。在实际系统中可以认为是运行环境，如 Aglet 系统的 runtime。

(6)分析模块。迁移分析模块是通过收集迁移过程中的中间值和状态数据，对本次迁移进行定性和定量分析，并将分析结果返回给策略组件。

当用户或应用系统提出服务请求后，策略组件首先接收请求，并调用感知模块，收集网络环境、资源状态等所需数据。然后策略组件根据提出的服务请求从组件库调用相应的功能组件以完成任务。选定功能组件后，策略组件根据感知模块得到的环境信息，选取策略库中的策略，从而决定交互模式。根据选择的交互模式，系统自动进行迁移组件的部署。迁移组件的构成因不同的交互模式而不同：C/S 迁移组件为空；COD 和 REV 迁移组件包括完成服务所需功能代码和初始数据；MA 迁移组件不仅包括功能代码和初始数据，还包括执行状态。迁移组件准备好之后，就可以通过迁移模块实施组件迁移。迁移完成后，分析模块对本次迁移进行必要的分析，并将结果返回给策略组件。策略组件利用这些反馈信息对策略库的策略进行必要的修正。迁移结束后，系统获得了完成任务所需的全部资源，执行模块开始运行，完成应用程序提出的服务请求。

在迁移过程中，如果网络环境发生改变，如网络断开或资源短缺，感知模块捕捉到这些变化，并通知策略组件对交互模式进行调整，不影响通信的有效进行。

3.2 模型的实现

为了实现上述的模型，我们通过比较可能实现的方法，实现了用策略完成的迁移模式集成。将几种模式集成起来，并提供接口，可以供系统按照规则选用。主要有两种方法：

①代码中嵌入实现方法。在设计代码时，根据所给参数和环境的指标，设计选择规则，并将选择规则嵌入应用程序。这不仅使得程序设计变得复杂，而且为了适应模式的改变，执行环境必须做出巨大的调整。整个系统的适应性很差，通用性也很受限制。

②用策略实现。将迁移模式的选择与迁移模式的实现分离开来，用策略来动态地改变采用的模式。在程序运行过程中，根据运行状况灵活地调整，不需要改变基本功能代码。这样，就将基本模式的实现与选择规则分离开来。当需要对规则进行修改的时候，只需要简单地修改策略库中的策略，不需要修改基本迁移模式实现代码。这具有很强的灵活性。

一个典型的 Ponder 描述的策略类型为：

```
Inst oblig policyName{
On          事件描述
Subject     行为执行者表达式
Target      行为对象表达式
Do          权力-行为列表
When        约束条件描述
}
```

通过这种方式，可灵活定义所有的交互模式的选择和修改策略。

4 模式选择规则的定性和定量分析

使用策略控制系统的行为，将设计的重点集中到规则的设计上来。合理而有效的运行规则是系统有效运行的关键。安全权限管理和功能性选择的规则比较容易制定，因为有明确的界限和角色区别。为适应系统资源和环境改变而制定规则，就不像前两者那么容易。我们必须考虑移动代码系统环境的变化，包括网络状况、系统稳定性、移动设备资源有限性以及无线网络带宽限制等。

4.1 定性分析

如果只按照资源的占有情况来选择使用的交互模式，选择变得比较简单，其规则如表 1 所示。

4.2 定量分析

普及计算的网络状况往往是具备多种交互模式的使用环境，要做出选择，就要根据使用者的关注点，如节约能源、效率

最高等进行更高级别的选择。因为无线网络常常考虑流量的问题，而普及计算对信息的及时性、准确性有着较高的要求，所以本文将以网络流量和通信时间为例，对各模式进行一些量化的分析。最优的交互模式是指交互过程的网络流量最小或通信时间最短的模式。

表 1　模式选择的性分析

C/S	网络结点没有完成服务的功能代码也没有完成服务的资源设备
COD	网络结点有完成服务的资源设备但是缺乏功能代码，其他可达网络节点能够提供功能代码。
REV	网络节点有完成服务的功能代码但是缺乏资源设备，其他可达网络节点具备所需资源设备
MA	网络节点有完成服务的功能代码和部分资源，但是要完成服务，必须要根据运行状态使用其他网络节点的资源、信息共同完成。

我们考虑一个场景，网络中的各个节点的资源和连接情况相同，而且每个节点都有移动代码的运行环境。也就是说，网络是标准的简单的互联网络，移动组件可以迁移到任何一个节点上面运行。假设一个网络节点 S_a 需要其它节点（如 S_b）的资源或代码或者配合支持才能完成某项任务。在此过程中，这两个网络节点必须进行交互。设两个节点（S_a，S_b）之间的平均网络带宽为 δ，忽略反应延迟和网络异常，我们分别考虑 4 种模式的网络流量和通信时间。

4.2.1　C/S 模式

S_a 向 S_b 发送请求指令和所需参数。S_b 完成功能后将结果返回 S_a。设命令头的平均长度为 h，参数长度为 B_{pa}，返回结果的长度为 B_r，需要的指令数为 Q，那么网络流量 $B_{c/s}$ 和通信时间 $T_{c/s}$ 为：

$$B_{c/s}=(2h+B_{pa}+B_r)Q \tag{1}$$

$$T_{c/s}=(2h+B_{pa}+B_r)Q/\delta \tag{2}$$

4.2.2　COD 模式

S_a 缺乏完成服务所需要的功能代码，发请求指令给 S_b，请求发送功能代码。S_b 将所需代码发送到 S_a。S_a 继续完成服务。设功能代码长度为 B_{code}，不考虑 S_a 在网络中搜寻所需功能代码的时间，那么网络流量 B_{cod} 和通信时间 T_{cod} 为：

$$B_{cod}=(2h+B_{pa}+B_{code}) \tag{3}$$

$$T_{cod}=(2h+B_{pa}+B_{code})/\delta \tag{4}$$

4.2.3　REV 模式

REV 模式与 COD 模式的迁移方向不同，数量上具有类似的分析。

4.2.4　MA 模式

移动代理开始在 S_a 运行，中途由于功能需要，迁移到 S_b 中继续执行。

MA 的迁移组件的大小可表示为代码、数据状态和执行状态的总字节数：

$$B_{mac}=B_{code}+B_{data}+B_{execution}$$

为便于分析，将 MA 的移动过程简化为：MA 序列化、MA 传输和 MA 解序列化。其中 B_{mac} 表示代码请求包的字节数。设 MA 序列化和解序列化需要的时间为 T_s。考虑 MA 将执行结果回传给源主机的情况，则网络流量 B_{ma} 和通信时间 T_{ma} 为：

$$B_{ma}=(2h+B_{mac}+B_rQ) \tag{5}$$

$$T_{ma}=(2h+B_{ma}+B_rQ)/\delta+T_s \tag{6}$$

由上可见，COD，REV，MA 方式都需要传送功能代码甚至状态信息。尽管还需要考虑指令头和参数的大小，但相对于功能代码而言指令头和参数都是较小的。

4.2.5　模式间比较

①两个网络节点情况下的 C/S 和 REV

C/S 虽然不需要传送额外的功能代码和其他信息，但是为了完成一项服务，必须多次传送指令和传回结果。

$$B_{c/s}-B_{rev}=(2h+B_{pa}+B_r)Q-(2h+B_{code}+B_rQ)$$
$$=2hQ+B_{pa}Q-2h-B_{code}$$

因此当 $(2h+B_{pa})*Q>2h+B_{code}$ 时，C/S 模式具有更高的传输流量。也就是说，REV 方式是较优的交互模式。式子左边是完成服务中所有指令头和指令参数的总大小，右边主要是功能代码的长度。可见，只要 REV 中功能代码的长度小于 C/S 指令的总大小，REV 便是比较好的选择。C/S 模式中，多次传递指令还很可能引起更长的网络延迟和交互响应时间，因此只要可以优化地将指令编写成功能代码，REV 的通信时间也将会比较短。

②多个网络节点情况下的 REV 和 MA

如果只需要在两个网络节点之间交互来完成服务，REV 方式显然比 MA 方式更加合适。因为 MA 方式除了功能代码 B_{code} 外，还需传送状态数据 B_{data} 和执行状态数据 $B_{execution}$，两者的交互次数和形式是一样的。当需要大于 2 个（设为 n 个）网路节点交互完成服务时，情况就不同了，因为需要根据运行中出现的中间结果决定下一步的行动。对于 REV 模式：

$$B_{revn}=(2h+B_{code}+B_rQ)*n \tag{7}$$

$$T_{revn}=(2h+B_{code}+B_rQ)*n/\delta \tag{8}$$

对于 MA 模式：

$$B_{man}=(h+B_{mac})*n+B_rQ \tag{9}$$

$$T_{man}=((h+B_{mac}*n+B_rQ)/\delta+T_s*n \tag{10}$$

REV 方式需要多次传送中间结果，MA 则可以直接将中间结果打包为状态数据，迁移到下一个网络节点。MA 为了收集状态数据并且打包，占用了一些额外的时间，因此只要 MA 模式打包的效率足够高，那么在多网络节点合作完成服务的过程中，MA 方式便具有更多的优越性。

5　实例分析

下面以智能病房的场景为例，对模型进行分析。

5.1　场景和任务描述

如图 2 所示的智能病房中，为了检测病人和环境的状况，设置了能够及时反馈信息的医疗传感器。这些传感器设置在病房不同的地方，测量并记录各种需要的即时数据值。病房中还有医疗设备，它们收到医疗指示后，可以对病人进行必要的紧急治疗和处理。另外，为了对医疗传感器获取的数据进行处理，并以此为依据激发医疗设备的医疗行为，病房中还要有服务器对总体进行管理和控制。

实例中一个典型的应用任务是：为了获取医疗传感器的数据，服务器可以通过 CS 模式访问医疗传感器的总体服务程序；或者采用 REV 模式设计特制的功能代码，并将其迁移到医疗传感器，获取所需信息；或者服务器采用 MA 模式，将功能代码和运行状态按照一定顺序迁移到各个传感器，收集数据。因此，服务器可能需要通过 C/S、REV 或 MA 方式的一种或几种与医疗传感器进行交互，完成任务（图中箭头 2）。

（下转第 67 页）

义。

参 考 文 献

1 Foster I, kesselman C. The Grid: Blueprint for a New Computing Infratructure[M]. Morgan Kaufmann, 2004
2 Douglis F, Foster I. The Grid Grows Up [J]. IEEE Internet Computing, 2003, 7(4): 24～26
3 Leff A, Rayfield J T, Dias D M. Service-Level Agreements and Commercial Grids [J]. IEEE Internet Computing, 2003, 7(4): 44～50
4 Menascé M A, Casalicchio E. QoS in Grid Computing. Journals of IEEE Internet Computing [J], July • August 2004
5 Chen Hanhua, Jin Hai, Mao Feng, et al. Q-GSM: A QoS Oriented Grid Service Management Framework
6 Raman R, Livny M, Solomon M. Matchmaking: Distributed resource management for high throughput computing [A]. Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing (1998), Chicago, Illinois, July 28-31, 1998
7 Khan A Y. Experiments with Scheduling Using Simulated Annealing in a Grid Environment [A]. In: Proceedings of GRID 2002. Baltimore, MD, USA, 2002. 232～242
8 Abraham A, Buyya R, Nath B. Nature's Heuristics for Scheduling Jobs on Computational Grids [A]. In: Proceedings of the 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), December 14-16, 2000, Cochin, India, 2000. 45～52
9 Martino V, Mililotti M. Scheduling in a Grid Computing Environment Using Genetic Algorithms [A]. In: Proceedings of International Parallel and Distributed Processing Symposium, 2002

(上接第 60 页)

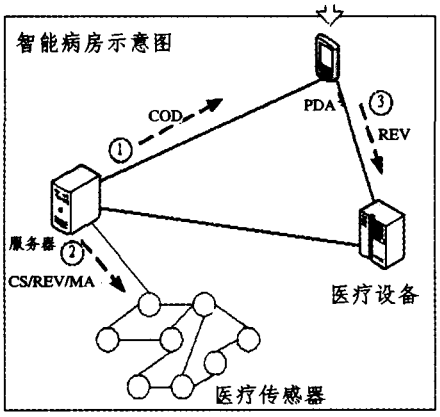

图 2 智能病房示例

5.2 模式选择策略

下面给出了用 Ponder 语言对模式进行选择与优化的部分描述，其中策略被直接实例化：

```
/*需要设定检测参数，完成一定功能的检查，如按照病人的呼吸、心跳、脉搏等确诊是否有呼吸道疾病，则采用 REV 模式，具体检查方法和需要数据在功能代码(know-how code)中描述*/
inst oblig CustomizedInfo{
  on SpecialCheck(MainDoctor)
  subject /HospitalDevices /RoomDevice
                           /Server
  target HospitalDevices /RoomDevice
                         /Sensor
  action REVParadigm(know-how code)
}
/*服务器在运行过程中，由于供电或带宽资源不足或系统故障出现不稳定状态，就采用 REV 或 MA 模式，使查询在传感器总体服务程序处本地完成，然后将结果传回服务器*/
inst oblig InstabilityServer{
  on InstabilityServer(RoomServer)
subject HospitalDevices /RoomDevice
                        /Server
  target /HospitalDevices /RoomDevice
                          /Sensor
  action MAInteraction(know-how code,evn state,execution state)
}
```

5.3 策略制定规则分析

在这个场景中，服务器与医疗传感器之间的通信可以采用 C/S、REV、MA 3 种方式。在网络和设备资源都充足的情况下，选择哪种方式更优化，可以做一个简单的分析比较。为了获取医疗传感器的值，服务器要与各个传感器进行交互。也就是说，这个场景属于多个网络节点的情况。根据上面的分析，假设有 n 个传感器需要通信，则

C/S 模式中：

$$B_{c/s}=(2h+B_{pa}+B_r)Qn \tag{11}$$

$$T_{c/s}=(2h+B_{pa}+B_r)Qn/\delta \tag{12}$$

REV 模式和 MA 模式按照公式(7)～(10)。

假设移动组件的大小远远大于指令头的大小。设 $h=20$ Byte，$B_{pa}=20$Byte，$B_r=150$ Byte，$Q=n=15$，$B_{code}=400$Byte，$B_{mac}=600$ Byte，$T_s=0.1$s，带宽 $\delta=10000$ Byte/s。可以将 3 种交互模式的网络流量和通信时间的值做一个柱状图做比较，如图 3 所示。

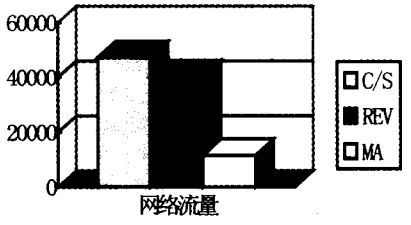

图 3(1) 3 种迁移模式的网络流量

由此可见，在网络中需要多个节点相互通信时，移动代理方式的通信流量和通信时间都是最佳的。远程评估方式次之，C/S 方式最差。在实际制定策略的时候，就可以类似上面的例子，按照所关注的指标量进行分析计算，配合以资源所有情况决定的定性分析，选择最佳交互模式。

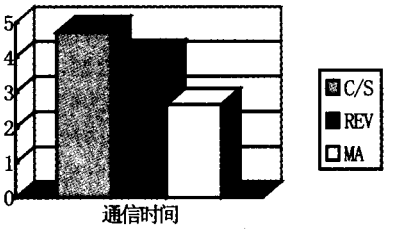

图 3(2) 三种迁移模式的通信时间

结论 本文分析了迁移模式集成的必要性和实现方法，提出了用策略实现的多种迁移模式集成模型。并对如何制定迁移规则进行了定性和定量的分析。集成模型不仅能够有效地满足普及计算动态多变环境的要求，还使得整个系统具有很强的灵活性和通用性。

参 考 文 献

1 Fuggetta A, Picco G P, Vigna G. Understanding code mobility. Software Engineering, IEEE Transactions on, 24(5)
2 Montanari R, Lupu E, Stefanelli C. Policy Based Dynamic Reconfiguration of Mobile Code Applications. PDF Computer, 2004, 37(7): 73～80
3 Montanari R, Tonti G, Stefanelli C. A policy-based mobile agent infrastructure. Proceedings, Applications and the Internet, 2003