自适应软件初探*)

王千祥 申峻嵘 梅 宏
(北京大学信息科学技术学院 软件研究所 北京100871)

摘 要 随着因特网向社会各角落的渗透式扩张,普适计算、网格计算等新型应用模式的不断涌现,软件面临的挑战也越来越多:管理和维护成本逐步增加、对健壮性的需求逐步增加、存储态与运行态方面的差异越来越明显,等等。自适应软件针对上述挑战,提供了一系列新的解决机制,试图使软件自身能够在运行过程中,实时收集系统的各种变化信息,并根据预先设定好的策略,在必要时对自身进行自动调整,从而更好地为用户提供服务。本文结合作者的研究工作,对自适应软件的基本概念、研究现状、研究途径等问题进行了初步探讨。

关键词 自适应,软件

An Introduction to Self-Adaptive Software

WANG Qian-Xiang SHEN Jun-Rong MEI Hong
(Software Institute, School of Information Science and Technology, Peking University, Beijing 100871)

Abstract Widespread popularization of Internet and emergence of many application patterns such as pervasive computing and grid computing have taken lots of challenges to software research and industry: the increasing cost on management and maintenance, the more strict robustness requirements, the even larger gap between released software and runtime software, etc. Under this circumstance, self-adaptive software has proposed series of solutions, trying to adjust itself while keeps running, based on the various collected information and pre-defined policies. This paper presents fundamental concepts, research status and implementation techniques of self-adaptive software, and our preliminary study in this field.

Keywords Self-adaptive, Software

1 引言

自适应软件的研究兴起于20世纪90年代前后。近年来,随着因特网向社会各角落的渗透式扩张,普适计算、网格计算等新型应用模式不断涌现。这使得软件的规模和复杂性不断增加,并正在成为软件面临的主要困难。自适应软件就是在这样的一种背景下产生的。

自适应软件的兴起与发展有如下几方面的因素:

首先,是软件在管理和维护方面压力的增加。目前许多大型软件已经达到了千万行代码的量级,伴随这种规模不断增加的,是软件复杂性的持续增加。如果软件仍然保持目前的开发与运行风范的话,由二者带来的软件管理和维护量将持续增加,并直接导致软件维护人员的快速增长。这十分类似于早期电话系统发展过程中面临的问题:当电话系统日趋成熟,需要进行大范围推广时,人工转接技术由于需要耗费巨大的人力而成为一种阻碍因素。后来,由于人们研究出了电话的自动转接技术,才将人们从大量的手工操作过程中解脱出来,并推动了电话的普及应用[1]。类似地,软件应用的发展趋势,要求软件具有一定的自适应性,即要求软件能够在运行过程中,具有对自身进行自动调整的能力,从而降低人工维护费用。

其次,是软件健壮性需求的增加。由于软件日益与人们的日常生产、生活密切相关,软件系统的失效或者错误将带来极大的损失。一方面,软件需求不断变化,使得用户所需要的软件和设计时刻存在着差距,从而很容易导致软件缺乏足够的健壮性;另一方面,大型软件的开发以协作的方式进行,而在开发周期中各协作小组相对独立且缺乏交流,也给程序的健壮性造成不利影响。如果软件能够根据预先设定好的策略,在必要时对自身进行自动调整,则可以减少潜在的灾难性后果。

最后,是软件在存储态与运行态方面的差异。存储态软件通常是指已发布的软件(Released Software),它和运行态软件(Runtime Software)往往不一致:前者可以看成是问题空间向软件解空间的映射,是静态实体(比如程序模块等)的集合;而后者则可以看成是前者在某一特定的运行环境下的一个实例,是动态实体(比如进程、资源占用等)的集合。这种存储态软件和运行态软件之间的不一致,使得传统方式下的存储态软件不能完全描述软件的实际运行状况,从而使运行态软件并不一定按照预先设定的程序执行,并产生大量的"例外"等事件,降低了软件的服务质量。因此,有必要提升传统的软件描述能力,描述当软件在运行时刻遇到一些特殊的事件时,应当如何调整软件的行为,从而弥补软件在存储态与运行态方面的差异。

本文简要介绍自适应软件的基本概念、实现途径以及研究现状。文章第2部分介绍自适应软件的基本概念;第3部分介绍研究现状;第4部分分析自适应软件的实现途径;最后总结全文,并展望未来的工作。

2 基本概念

从一个较为完整的生命周期看,软件处于不断变化的环

*)本文得到国家863项目资助(课题编号:2003AA115430);国家自然科学基金资助(项目编号:60103001)。

境之中。如何适应外部的各种类型变化，始终贯穿着软件技术的发展过程。

从开发角度看，软件经常遇到的一般性变化包括：增加新特性，以吸引客户；重组业务流程，以提高效率；改变数据库表，以适应功能调整；调整界面，以方便人机交互；提高可靠性、提高安全性，等等。近年来备受人们关注的设计模式(DP：Design Pattern)、面向侧面的编程(AOP：Aspect Oriented Programming)、软件体系结构(SA：Software Architecture)等开发技术都与开发角度面临的上述种种变化密切相关。

从运行角度看，软件面临的大量变化包括：资源属性的变化，例如：系统硬件配置增加(存储、节点)、资源发生故障，等等；运行上下文的变化，例如：带宽变窄、受到攻击，等等；用户访问的变化，例如：用户数量的变化、用户偏好的变化、用户访问物理地址的变化、用户访问协议的变化，等等。对于这类变化，长期以来主要由维护人员对软件进行手工调整，以使软件能够提供较好的服务。

自适应软件的出现将可以对运行角度看到的许多变化提供良好的自动性支持。与手工调整相比，自动性支持的优势包括：维护开销低、对变化的响应时间快、不存在误操作等，其不足之处包括：调整能力受限、灵活性不足等。因此，软件自适应通常必须在人员手工操作的支持下方能取得预期效果：其一，自适应策略是由维护人员预先设定的，并可以根据适应效果对这些策略进行调整；其二，软件自身的自适应能力和与手工支持的在线演化能力合作，共同调整，可以以软件适应不同种类的变化。

文[2]认为：自适应软件关注高层感知性、适应性与自动性。它监视自身，并在发生变化或错误时修正或者改进自身，通常利用反馈-控制-系统行为，通过修改或者重组自身的程序或子系统完成。

本文认为：自适应软件是一类特殊的软件，这类软件能够在运行过程中，实时收集系统的各种变化信息，并根据预先设定好的策略，在必要时对自身进行自动调整，以更好地为用户提供服务。

除了自适应软件的基本概念之外，其分类也一直是这个领域的重要研究课题之一。很多研究人员给出了自己的分类方法，比较有影响力的有以下几种：

·按照自适应软件实现所借鉴的领域分类[6]。包括借鉴动态规划系统(应用并不是简单执行特定的算法，而是规划自身的行为，并将计算资源，比如硬件、通信能力和构件作为规划的对象)，借鉴控制系统(将整个运行态软件看作一个工厂，包括输入输出、监视器和调整工厂行为的控制单元)和自意识(Self Aware)系统(采用自建模的方法，评估、修订和重配置都由运行态软件中所包含的模型驱动)等。实现时一个应用可以借鉴多个领域，所以这种分类法对实现方法有指导意义，但是并不能将各种类型明确区分。

·按照自适应软件的应用领域分类[7]。包括嵌入式应用、知觉应用(perceptual Application)、通信协议应用、信息持久性应用等。这种方法通过挖掘自适应软件的不同应用领域，并研究各个领域中所采用的研究方法，以期望发现自适应软件的一些通用性质和开发方法。

·按照自适应软件调节自身行为的方式和效果分类[8]。包括自管理、自配置、自优化、自修复、自保护等。IBM 采用这种分类法，希望将自适应软件的研究统一于其倡导的自治计算(Autonomic Computing)[4]的体系之中。

3 研究现状

自适应性问题首先在人工智能领域得到关注，并在遗传算法、神经网络等研究方向上获得了较大的进展。人工智能领域关注自适应的主要目的在于通过对人类智能的模拟，获取产生自适应能力的方法与途径，并努力使这些方法在其他一些领域中得到应用。尽管这些研究成果很少可以直接应用到软件自身上来，但人工智能领域长期积累的一些关于知识表示、策略与规则的实施等方面的研究成果可以为软件自适应提供很好的借鉴。

真正从软件适应性角度对自适应进行研究的时间并不长。美国 DARPA 于1998年将支持自适应的软硬件系统列入研究计划[3]，并资助了一些研究机构从事这方面的研究。IEEE 杂志"智能系统"(Intelligent System)在1999年第3期上出版了一个专集，对早期获得的关于自适应软件/硬件系统的研究成果进行了介绍，当时人们关注的多数是那些嵌入在硬件系统的嵌入式软件，例如嵌入在导弹[10]、机器人[11]等中的软件。这可以视作自适应软件研究的早期阶段。

随着嵌入式软件和因特网的普及与发展，近年来关于软件自适应的研究也逐渐成为软件界的研究热点。由 MIT 的人工智能实验室发起的关于自适应软件的国际研讨会(IWSAS：International Workshop On Self-Adaptive Software)已经分别在2000、2001、2003年召开了三次。第一次会议主要试图解决自适应软件的定义问题，并以自适应性子技术图的形式确定了一些有用的研究方向[9]；第二次会议则主要关注自适应软件的最佳适用领域及其分类，并致力于从已有的案例中发掘自适应软件常用的设计模式；第三次会议则将注意力转移到更加理论和基础的层面，希望能够建立起自适应软件的数学基础和系统理论，并着重讨论了自适应软件的测试、调试、质量、可靠性等方面的问题。从三次会议发表的文章看，现阶段自适应系统的研究受控制论、人工智能领域研究成果的影响较大。

在2002年11月美国 ACM SIGSOFT 召开的年会上，举办了一个关于自修复系统的研讨会 WOSS(Workshop on Self-Healing Systems)'02，在介绍各自研究内容的同时，许多参与 WOSS 的研究人员也指出了关于自适应、自修复等技术需要关注的主要问题，例如人机合作问题、编程语言问题、软件体系结构问题[13]等。

从目前人们对自适应软件技术研究的方式看，多数研究组织都将该研究建立在已有工作的基础上面。例如，从事 Agent、软件体系结构、传统中间件等研究方向的许多组织都在尝试在自己现有研究成果的基础上增加软件自适应技术的内容。

4 实现途径

本节首先介绍自适应软件的理论基础和基本模型，然后讨论需要应用和亟待解决的技术问题，最后介绍三类主要的自适应软件研究方法。

4.1 理论基础和基本模型——控制论

自适应软件在运行时刻将功能、行为与预先设定的目标相比较，并据此对系统进行调整，相比于传统的软件，这种性质和传统的控制系统的"收集-决策-控制"周期更为相似。文[9]总结了这种相似性，并根据控制系统领域的基本理论和方法提出五种基于控制理论的自适应软件模型，分别是开放回

路、闭合回路(反馈)、拥有 QoS 子系统的闭合回路、间接适应和重配置,并将其作为自适应软件的基础。这一观点得到了许多研究者的认同和响应,随之出现较多基于控制理论的自适应软件原型,包括流和阻塞控制、适应个别差异的缓冲、Web 服务、多媒体流、Web 服务器性能、Email 系统控制、网格中分布式资源的分配等。

借鉴传统控制理论,自适应软件系统应包括系统自身(Computer System)和环境(Environment)两个实体:环境是一个动态系统,其行为是它的以前状态、操作和时间的函数;系统则可以通过恰当地选择操作来达到外部定义的目标,而操作由外部传感器的输入、目标和内部状态决定。这样的模型如图1所示。可以看出,自适应模块应当包括输入/输出、监视器和控制单元等几个部分,评估、度量和控制系统处于需要适应的实体(比如应用)上层,并管理整个系统的重配置。在设计自适应软件的时候,需要为这几个部分选择适当的策略,并采用适合的技术。

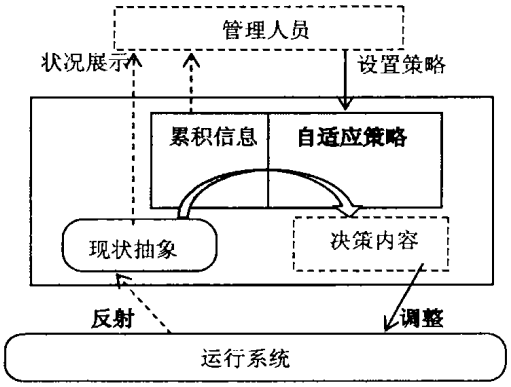

图1 基于控制理论的自适应软件模型

但是,尽管由"收集-决策-控制"等构成一个完整的反馈与控制回路是自适应的基本框架,但由于人们对自适应系统的研究时间还不长,目前的自适应系统离形成稳定的框架体系尚有较长的距离。

4.2 相关技术问题

在上述自适应软件模型的指导下,设计与开发自适应软件需要明确以下几个问题:

·自适应的目标是什么?对运行态软件的调整是修正错误、增减功能还是提高性能?

·自适应软件所处的环境如何?软件是在一般的单机下运行,还是位于嵌入式系统中或者因特网环境下?

·自适应所面临的约束怎样?软件是安全关键的,还是要求很高的服务质量,或者对性能和响应时间有很高的要求?所要求的软件扩展性如何?

·所允许的人工干预的程度?是完全的软件控制还是允许人工参与?

·采用何种方式描述软件自身的知识?是采用软件体系结构还是其它方式的模型?

·如何评估软件是否满足期望?期望如何描述?是采用集中式的整体评估,即建立系统的整理模型,还是对每个部分单独评估,让系统的每个构件对自身的状态和行为负责?

·监视器应当收集哪些信息以获得对运行态软件的认识?收集信息的频率如何?以何种方式收集?

·如何描述并执行期望系统做出的修改?是采用形式化的语言描述,还是通过选择预先设定的程序片断?是通过中间件的反射机制控制,还是采用构件自身的方法调整?如何提供对修改运行态软件这一事务的支持?

·如何判断和评估所做的修改是有效的?如果无效,如何恢复到修改之前的状态?

对以上问题的回答构成了软件自适应策略的核心,自适应策略包括用户的期望、系统的设计决策和自适应调整的规则,是连接结构信息提取与应用调整的纽带,是衡量自适应能力的标准。

在确定了自适应策略之后,还需要考虑自适应软件需要采用的技术。文[8]认为,在人工智能问题得到完全解决之前,在现有技术和理论的基础上发展软件自适应是可能的。对于自适应软件而言,一方面可以采用现有的具备一定适应性的技术,另一方面可以通过开发出新的框架、模式、理论和方法来实现。文[9]指出了自适应软件可以采用的技术,它将自适应视作一个复杂的特征,并定义了一系列的特征。本文参考它提出了如图2所示的自适应软件技术图。

图2的下半部分是一些简单的自适应技术,也是高级程序设计语言可以直接支持的。比如待选函数(过程、方法)库,简单的运行时刻显式选择(如根据函数基调)。在此基础上增加一些自监控、自测试和用户监控的能力,则可以使选择待选方案有了一定的参考依据。待选方案,动态分派和监控构成了自适应的第一层次,实际的自适应的操作都需要由第一层次的机制来执行。在此基础上增加语义和语法的自描述机制,同时辅以适当的反射式编程技术,就可以使得运行态软件能够获得设计信息,从而有助于修改软件系统的结构和行为特征。这是自适应的第二层次,和第一层次一起构成了图中其它自适应子技术的基础,可以看成是整个自适应框架的基础设施。

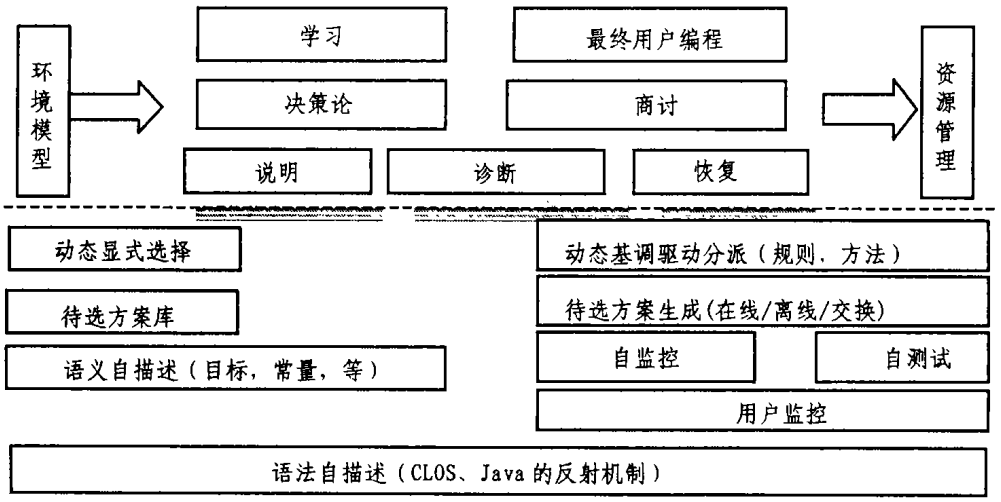

图2 自适应软件技术图

接下来讨论图中上半部分列出的较为复杂的自适应技术。

"环境模型"提供软件所处环境的结构和行为的描述，包括嵌入式系统所处的物理模型，行为统计模型，外部状态的关系模型和软件的输入、输出等。交互式调试工具对应着特定程序设计语言的实现，可以用于诊断和说明行为，也可以在人工参与下恢复软件。

"说明"指有效地刻画软件的行为，并协调自监控的行为与高层的功能、目标描述。"诊断"指发现正确和错误行为之间的关键依赖性。简单地说，"说明"是描述"是什么"的问题，而"诊断"则要指出发生错误的原因。"说明"和"诊断"为底层的可选方案选择提供了向导。"恢复"包括记录功能或性能的矛盾，诊断问题并用正确的构件替换发生错误的构件，还包括判断、修正副作用，必要时回滚，以及找到正确的修复点。

"商讨"是指通过投票机制、演绎推理、归纳推理、基于实例的推理等形式，从监控所获得的数据和现象向特定的活动推演的方法。"决策论"(比如 Bayesian 决策论)是"商讨"所采用的最为普及的形式之一，它通过通用的度量方式，并采用概率来衡量数据和现象，采用功效来估算所选择的参数。

"学习"和"最终用户编程"属于高级自适应能力。"学习"指软件记录下错误和尝试的修复，并学习更好的方法。学习包括从经验归纳出通用原则，人工辅助学习等。"最终用户编程"是基于自适应软件拥有丰富的关于自身的组织、结构和待选方案的前提，给用户提供一些有指导意义的对话框，使用户能够根据自身的经验和偏好，通过配置以获得更加个性化的服务。

"资源管理"是指自适应软件对它所拥有的计算和通信资源的管理，包括时间、空间的权衡等。资源管理也是许多实际系统所面临的问题，并对自适应软件的环境模型有所影响。

除去上述技术之外，在自适应软件设计的过程中，还需要考虑校验、元数据(模型)、稳定性、安全性、约束、需求改变等问题。

现阶段的自适应软件技术主要是在各自原有工作的基础上，比如 Agent、软件体系结构、中间件等，通过将上述技术进行合理的组合，从而形成新的自适应理论和范型。运行时刻软件体系结构(RSA：Runtime Software Architecture)[5]是我们在"说明"性方面所采用的主要途径。

4.3 研究方法案例

本小节简要介绍三种自适应软件的研究方法：

·基于 Agent 的自适应软件研究　在从事 Agent 研究的组织中，以 IBM 研发部门的投入最为显著。2001年3月，IBM 的研发副总裁 Paul Horn 做了一个关于自治计算的演讲[1]，鼓励 IBM 的研发人员从事可以进行自管理软件的研究。这一号召得到了 IBM 研发人员的广泛响应，分别从不同的领域、不同的角度开展研究，所取得的主要成果发表在了2003年的"IBM Systems Journal"专刊上。其中，所开发的 ABLE (Agent Building and Learning Environment)是一个基于 Java 的 Agent 开发与部署环境，提供了一个开发库，这个库由智能推理、学习构件包与构造智能 agent 的轻权 Java agent 框架构成。Diao 等人利用 ABLE 对 Apache Web Server 进行自动调节，明显地改善了 Apache 的性能[11]。实际上，ABLE 本身是一种特殊的中间件，只是由于人们太关注于 ABLE 在智能性方面的特征，而忽略了它的中间件特征。在取得这些研究成果的同时，IBM 研究人员也进一步认识到了自适应系统面临的一些工程与科学挑战[4]。其中工程挑战包括：自治系统的生命周期问题、它们之间的相互关系、目标规约问题等；科学挑战包括：行为的抽象与模型化、健壮性理论、学习与优化理论、自动统计理论等等。从 Agent 角度研究自适应的好处在于有大量现有的成果可以借鉴。

·基于软件体系结构的自适应软件研究　从事软件体系结构的许多研究机构也十分重视自适应技术的研究。软件体系结构描述软件所包含的构件、构件之间的关系、构件的约束以及关系的约束等内容。这些内容通常也是自适应软件关注的内容。尤其是当体系结构被用来描述运行时刻的软件时，软件体系结构信息为设置软件的监控点提供了非常好的参考依据。卡内基-梅隆大学 David Garlan 教授领导的项目 Rainbow 主要研究复杂系统中基于体系结构的适应技术，目前正在增强关于自适应的研究，并尝试将所取得的一些初步进展应用到普适计算中[13]。加州大学(Irvine)的 Richard Tailor 教授很早就开展了基于软件体系结构的软件自适应技术研究[12]，并在基于消息传递体系结构风格的软件自适应技术取得了良好的进展。

·基于反射式中间件的自适应软件研究　从事传统中间件技术研究的一些组织也十分关注软件的适应性问题。对反射式的支持是中间件近期关注的重要内容，并为自适应提供了良好的基础。所谓反射式中间件，就是一种能够监测并为调整系统状态和行为提供支持的中间件系统。OpenCorba 是一种基于反射式语言 NeoClasstalk 的反射式 CORBA，通过元类将 ORB 内部特征分离并单独实现，从而允许系统在运行过程中监测并调整这些内部功能单元，如，监视远程对象调用的行为，在服务器进行类型检查等[14]。dynamicTAO 是实时 ORB 软件 TAO 扩展，其反射能力体现在对系统服务的配置与重配置，即，通过配置管理器将特定的策略(如安全策略、调度策略等)及其实现机制关联起来[15]。从自适应角度看，关于反射式中间件的研究为自适应的实现提供了非常好的运行平台。

结束语　因特网环境下软件复杂度的增长和用户对软件健壮性和自主性的要求使得自适应软件的研究成为学术界和产业界关注的热点。现今已经有了不少支持自适应软件的原型和系统出现，关于自适应软件的开发范型、基本理论、设计模式的研究也逐步浮出水面。但是，和传统的软件相比，自适应软件的发展历史还相对较短，许多原型产品的自适应性还处于比较低级的层面，还有许多理论和技术问题需要解决。本文主要讨论了自适应软件兴起的动因、自适应软件的基本概念和分类、研究现状和自适应软件实现的基本技术。由于自适应软件的研究刚刚起步，本文仅作初步介绍，希望能对从事自适应软件的研究起到促进和推广作用。

参 考 文 献

1 IBM, Autonomic Computing: IBM's Perspective on the State of Information Technology, URL. http://www-1.ibm.com/industries/government/doc/content/bin/auto.pdf

2 New Visions for Software Design & Productivity: Research & Applications. http://www.itrd.gov/pubs/sdp_wrkshp_final.pdf.

3 msrc.wvu.edu/nsf_epscor/cluster_research/ arpa_baa98_1.html

4 Kepbart J O, Chess D M. The Vision of Autonomic Computing. IEEE Computer, Jan. 2003. 41～50

5 Wang Qianxiang, Huang Gang, Shen Junrong, Mei Hong, Yang Fuqing. Runtime Software Architecture Based Software Evolution

(下转第178页)

的发生，这就要求调控 Agent 之间具有良好的交互以协调各自的工作。如图3所示。

5 实验

为了测试上述方法的可行性，我们采用图3的网格计算模型，并设由 PC3提交一个计算任务，采用 RIPS(Runtime incremental parallel schedule)任务调度策略进行作业调度。在执行调度任务的过程中，可能会产生新的任务。当系统进行进一步调度时，上一轮调度没有执行完的任务与新产生的任务一同调度，两个阶段交替进行，直到全部的任务执行结束，为了简化问题，假定提交的任务有 n 个相互独立并有一定执行序列的子任务 $A_i(i=1,\cdots,n)$。

建立一个 Task Agent 来完成子任务执行的同步，如果操作 A_i 依赖于操作 A_j，那么执行操作 A_i 的 Agent 在执行操作 A_i 前通知 Task Agent 等待操作 A_j 的完成，此时 Agent 即通知处于等待状态的 Agent：操作 A_j 已完成，可以继续执行。

为此，我们定义两个消息：Wait(A_i)——它用于等待操作 A_i 的完成；Complete(A_i)——用于表示操作 A_i 已完成。

假定执行操作 A_i 和 A_j 的 Agent 分别为 $Agent_i$ 和 $Agent_j$，协调者为 $Agent_k$。则给出的 $Agent_i$ 和 $Agent_j$ 的执行计划如下：

```
Agenti:
Char * name;//站点名称
LOAD loadnum; //站点当前负载
LOAD maxloadnum;//站点最大负载
Int stanum: //站点编号
……
Send Agentk Wait(Aj)
Wait Cimplete(Aj)
Execute Ai
…
…
Agentj:
Char * name; //站点名称
LOAD loadnum; //站点当前负载
LOAD maxloadnum;//站点最大负载
Int stanum: //站点编号
……
Execute Aj
Send Agentk Wait(Ai)
…
…
```

其中，$Agent_k$ 需要提供的处理过程如下：

```
On Complete(Ai)Do
Send Agenti Complete(Ai)
At Priorty(xxx)
```

当所有的子任务调度、执行完毕之后，集成子任务的执行结果作为一组既得的目标知识，它即为整个任务的结果。

通过作为协调者的 Agent 向所有的 Task Agent 搜集它们执行操作所获得的知识，便可以得到任务的执行结果。

结束语 本文介绍了网格计算的体系结构和运用网格计算环境进行作业计算、资源管理、负载平衡等方法存在的问题，详细论述了在网格计算环境下引入移动代理技术对于解决网格计算问题的作用和优点。然后分别对网格计算中的移动 Agent 结构、资源管理模式、作业分配与调度方法、负载平衡等方面进行了研究。

参考文献

1 Chess D. Security Issues in Mobile Code Systems, in: Giovanni Vigna (ED.), Mobile Agent Security, LNCS 1419, Springer, 1998. 1～14
2 Karnik N M, Tripathi A R. Design Issues in Mobile Agent Programming Systems. IEEE Concurrency, 1998,6(3):52～61
3 Wilhelm U G, Staamann S M, Buttyan L. A Pessimistic Approach to Trust in Mobile Agent Platforms. IEEE COMPUTING, SEPTEMBER·Oct. 2000. 40～48
4 Eckel B. Thinking in Java, 2nd Edition, Release 11. Prentice-Hall, 2000
5 Caro G D, Dorigo M. Two Ant Colony Algorithms for Best-effort Routing in Datagram Networks. In: Proc. of the 10th IASTED Intl. Conf. on Parallel
6 Schoonderwoerd R, Holland O, Bruten J. Ant-like Agents for Load Balancing in Telecommunications Networks. In: Proc. of the First Intl. Conf. on Autonomous Agents, ACM Press, 1997. 209～216
7 Schaaf M, Maure F. Integrating Java and CORBA: A Programmer's Perspective. IEEE Internet Computing, Jan.-Feb. 2001. 72～78
8 Farmer W M, Gutman J D, Swarup V. Security for Mobile Agents: Authentication and State Appraisal. In: The 4th European Symp On Research in Computer Security. Rome, Italy, 1996. 18～130
9 Eckel B. Thinking in Java, 2nd Edition, Release 11. New Jersey: Prentice-Hall, 2000
10 Davies W, Price W L. Security for Computer Networks. Chichesfer: John Wiley & Sons, 1989
11 Pfleeger C P. Security in Computing, Second Edition. Upper Saddle River, New Jersey: Prentice Hall, 1997
12 Wooldridge M. Intelligent Agents: Theory and Practice, Knowledge Engineering Review, 1995, 10(2)
13 Muller J P. The Design of Intelligent Agents, A Layered Approach, 1996
14 DunhamMH, HelalA. Mobile computing and databases: anything new? ACMSIGMOD Record, 1995, 24 (4)
15 Villinger K, Burger C. Generic mobile agents for electronic markets. In: The 4th Conf. of the Internet Society for Decision Support Systems, Switzerland, 1997
16 OMG, The common object request broker, architecture and specification. July 1995
17 Kraus S, Sycara K, Evenchik A. Reaching agreement through argumentation: A logical model and implementation. Artificial Intelligence, 1998, 104(1-2): 1～69
18 何炎祥，陈莘萌. Agent 和多 Agent 系统的设计与应用. 武汉：武汉大学出版社，2001
19 张云勇. 移动 Agent 及其应用. 北京：清华大学出版社，2002

(上接第171页)

And Adaptation, COMPSAC 2003, Dallas, Nov. 2003
6 Laddaga R. Creating robust software through self-adaptation. IEEE Intelligent Systems, 14, May/June 1999. 26～29
7 Laddaga R, et al. Introduction to Self-adaptive Software: Applications. Lecture Notes in Computer Science, 2001, 1936: 1～5
8 Kephart J O, Chess D M. The Vision of Autonomic Computing. IEEE Computer, 2003, 36(1): 41～50
9 Laddaga R, et al. Results of The First International Workshop on Self Adaptive Software, Lecture Notes in Computer Science, 2001, 1936: 242～247
10 Kokar MM, Baclawski K, Eracar Y A. Control theory based foundations of self controlling software. IEEE Intelligent Systems, 1999, 14(3): 37～45
11 Diao Y, Hellerstein J L, Parekh S, Bigus J P. Managing Web Server Performance with AutoTune Agents. IBM Systems Journal, 2003, 42(1). URL. www. research. ibm. com/journal/sj/421/diao. pdf
12 Oreizy P, et al. An architecture-based approach to self-adaptive software. IEEE INTELLIGENTSYSTEMS, MAY/JUNE 1999
13 Garlan D, Schmerl B. Model-based Adaptation for Self-Healing Systems. WOSS'02, 2002
14 Ledoux T. OpenCorba: A Reflective Open Broker. In: Cointe P, ed. the 2nd Intl. Conf. on Reflection, LNCS 1616. Heidelberg: Springer-Verlag, 1999. 197～214
15 Klefstad R, Schmidt D C, O'Ryan C. Towards Highly Configurable Real-time Object Request Brokers. In: Fifth IEEE Intl. Symposium on Object-Oriented Real-time Distributed Computing. Washington, D. C., USA., 2002. 347～447