中间件技术的研究*)

郭 胜 许 平 王 颖 陆桑璐 陈道蓄 谢 立
(南京大学计算机科学与技术系 计算机软件新技术国家重点实验室 南京210093)

摘 要 计算机以及网络通信技术的发展使得中间件技术取得了长足的进步,出现了许多有关中间件的软件开发理论、标准协议以及各种商业软件产品。随着信息系统越来越以网络为中心进行开发和演化,中间件技术将扮演更加重要的角色。本文介绍了中间件领域的发展现状,主要分析了当前中间件领域中主流的DOC技术和正在兴起的Web Services、Grid/OGSA技术,并且我们根据未来分布式系统的多种需求,探讨了中间件领域将面临的挑战。

关键词 中间件,分布式对象计算,Web Services,Grid,OGSA,QoS

Researches on Middleware Technology

GUO Sheng XU Ping WANG Ying LU Sang-Lu CHEN Dao-Xu XIE Li
(Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract With the development of computer and network communication technology, middleware makes significant progress. There have yielded various theories of software development, standards and commercial softwares in middleware area. Middleware will act as a more important role in the development and evolvement of large-scale, network-centric systems. This paper introduces the current state of middleware, summarizes the mainstream middleware technology -- Distribute Object Computing, as well as some booming technologies such as Web Services, Grid and OGSA. Then, according to the requirements of future distributed systems, we discuss the challenges which middleware technology is facing.

Keywords Middleware, DOC, Web services, Grid, OGSA, QoS

1 引言

随着计算机技术的飞速发展,各种软硬件产品越来越多,更新速度也越来越快,软硬件之间呈现出复杂的多样性和异构性;此外,大型信息系统也越来越趋向于以网络为中心的模式进行开发、维护和演化。因此,应用软件通常需要在不同厂家的网络产品、硬件平台、网络协议组成的异构环境下运行,应用的规模也从局域网发展到广域网。在这样一个不断变化的异构的分布式计算环境中,单机系统和简单的Client/Server系统已不再能满足需求,而是迫切需要一种独立于网络、计算机硬件以及操作系统的基于多层体系结构的分布式计算平台,中间件技术因此应运而生。

中间件是一种位于具体应用和底层系统(包括操作系统、网络协议栈、硬件等)之间的软件。中间件在这个软件体系中所扮演的角色是(1)连接应用程序和底层软硬件基础设施,协调应用各部分的连接和互操作;(2)使系统开发者能够实现并简化基于各种不同技术的服务组件之间的集成。在应用系统开发中采用中间件技术有以下优点:

1. 能对软件开发者屏蔽底层的、复杂繁琐的、易出错的平台细节,如socket层的网络编程等,减少应用开发的复杂性。

2. 能提供大批可复用的、构件化的服务。使开发者在应用开发过程中充分利用前人的开发经验,以及一些比较成熟的解决方案,从而不必从头开始构造应用系统,加快应用开发周期,降低开发成本。

3. 能为应用提供一个面向网络的高层抽象的集合,这种一致的高层抽象能简化分布式系统的开发。

4. 协调应用系统各部分之间的互联、互操作,使得从小到组件模块,大到企业应用实体所提供的服务都能够相互集成,并使这种集成得到简化,这样系统的构建就可以基本通过集成而不是编程来完成。

总之,中间件技术为降低分布式应用系统的开发、部署、运行和维护的复杂性提供了有力的工具。但是现在,使用中间件集成复杂系统的要求还不能完全得到满足,相关的技术还不够成熟。因为,从中间件上层的分布式应用需求到下层的基础架构的不断发展都给中间件技术带来了新的问题和潜在的解决方案;而且,以网络为中心的系统开发范式促使多层次中间件的形成,每个层次的中间件都包含了互相交织的技术,这给基于中间件的系统集成带来了新的复杂性。

今天,要构建更复杂的分布式应用系统面临着许多严重的挑战。比如:应用日益要求更苛刻的QoS。系统的可用资源和用户对系统的服务要求都在不断地变化等。因此,一个仅能在特定环境下执行特定功能的系统不再能满足需要,事实上,我们通常都希望系统在不同的条件下有不同的行为。这些变化带来了一系列问题,比如要求提供面向端到端的具有适应性的QoS,要求由商业构件或服务组装系统等。

本文第2部分简单介绍了三类中间件技术——DOC、Web Services、Grid/OGSA。我们主要关注每种技术的特点和它们作为中间件而具有的共性。第3部分中我们将主要从未来分布式应用的多种需求的角度来探讨中间件领域将面临的一些挑

*)本文得到国家八六三技术研究发展计划(编号2001AA113050)资助。郭 胜 硕士研究生,主要研究方向为分布式计算。许 平 硕士研究生。王 颖 硕士研究生。陆桑璐 副教授,主要研究领域为分布式计算。陈道蓄 教授,博士生导师,主要研究领域为分布计算与并行处理。谢 立 教授,博士生导师,主要研究领域为分布与并行计算机系统。

战。最后是总结，认为要以系统的、动态的观点来研究与发展未来以网络为中心的中间件系统。

2 中间件技术的现状

当前中间件技术领域中的主流是分布式对象计算技术(DOC)。DOC 中间件经过了十多年的发展，已经比较成熟，产生了许多标准协议、框架模型、体系结构、软件工程方法以及商业构件、中间件服务等。这些技术为整个中间件领域的发展奠定了坚实的基础。现在，DOC 中间件正不断完善并扩展到大规模实时嵌入式系统中。

电子商务的发展要求企业 IT 应用能突破企业界限，在 Internet 范围内实现服务的集成。但是由于商业利益和技术竞争，DOC 中间件在解决系统异构性的同时又人为地引入了新的异构性，因此要在 Internet 这个动态的不断变化的环境中实现应用的集成，需要新的解决方案。基于 XML 标准的 Web Services 技术为解决这个问题提供了新的思路。Web Services 定义了一套协议，希望通过这些协议的标准化实现企业应用的集成。但这套协议还未成熟，需要进一步发展。

除了 DOC 和 Web Services 之外，网格技术(Grid)也已进入中间件领域。网格技术一开始是用于 e-science 计算的，它试图利用互联网把分散在不同地理位置的计算节点组织成一个“虚拟的超级计算机”。网格技术的目标与企业计算有很多相似之处，但它缺少 Web Services 的互操作性和灵活性。现在 Web Services 技术和网格技术大有相互融合的趋势，最新出现的 OGSA 技术即是这种融合的高级中间件技术。

下面，我们将分别对这三种中间件技术进行介绍。

2.1 分布式对象计算(DOC)

分布式对象计算(DOC)是随着网络和面向对象技术的发展而不断地完善起来的。90年代初 CORBA 1.0标准的颁布，揭开了分布式对象计算的序幕。

DOC 是一种高级的、成熟的中间件连接规范，它支持灵活的、适应的应用行为。基于 DOC 的中间件体系结构，由一批相对自主的软件对象组成，这些软件对象能在大范围的网络中部署，客户端能调用目标对象上的操作、执行交互、完成应用所需的功能。目前，DOC 已经成为建立服务应用框架和软件构件的核心技术，在开发大型分布式应用系统中表现出强大的生命力，并逐渐形成了3种具有代表性的主流技术，即 Microsoft 的 COM/DCOM 技术、Sun 公司的 Java/J2EE 技术和 OMG 的 COBRA 技术。其中 CORBA 和 J2EE 都成为中间件领域中比较成熟的标准。

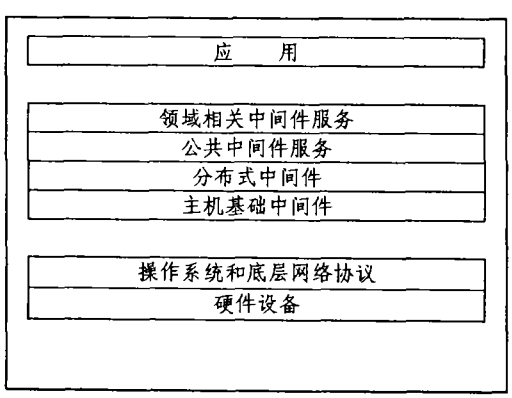


图1　DOC 的分层结构及其上下文环境

DOC 中间件的发展产生了一系列软件协议和服务层，这些协议和服务层有助于解决系统集成中的异构性和互操作性等问题。DOC 中间件能分为多个层次[1]，如图1所示。

主机基础中间件：其功能是封装并提高本地操作系统的通讯和并发机制，创建可复用的网络编程组件。这些组件对本地操作系统的特性进行抽象，通过使用编程 API(如 sockets，POSIX pthreads)减少开发维护网络应用所涉及的复杂性、易错性和不可移植性。代表性的中间件有：JVM，.NET，ACE(适应性通讯环境)[2]。

分布式中间件：定义了更高层的分布式编程模型，分布式中间件通过提供可复用 API 及组件，自动化且扩展了主机基础中间件所提供的网络编程能力，使用户建立分布式应用就像建立本地应用一样，可直接调用目标对象的操作，在编码上提供了对对象的位置、语言、操作系统平台、通讯协议、硬件等的透明性。分布式中间件的核心是具有 QoS 能力的“请求代理”(broker)，这一层上的代表性的中间件有：CORBA，RMI，DCOM。

公共中间件服务：进一步扩展了分布式中间件的功能，定义了更高层的独立于应用领域的服务(如事务，安全，数据库连接等)。分布式中间件主要集中于管理端系统的资源；而公共中间件使用组件编程模型，在整个分布式系统上定位、调度、协同各种端到端的资源。开发者可重用这些服务，管理全局资源，执行公共的分布式任务。代表中间件有：CORBAServices；J2EE 等。

领域相关中间件服务：这是根据特殊应用领域(如电信，电子商务，医疗保险，航空等)的要求定制的服务，这层中间件是其它三种中间件中最不成熟的一种，其原因是由于历史上缺乏分布中间件和公共中间件的标准，而这层中间件的创建需要这些标准提供一个稳定可靠的基础，由于领域中间件包含了领域知识，因此最有潜力增强系统 QoS，降低开发周期和成本。

DOC 中间件经过十多年的发展，在分布式计算领域取得了不少的成功，定义、推动、扩展了人们对中间件概念的理解；增加了中间件的层次并划分了各层的功能；形成了以网络为中心的开发范式；产生了不少较为成熟的标准协议、框架模型、体系结构、软件工程方法以及商业构件。当然，DOC 领域还有不少挑战：如适应网络环境和需求变化的全局资源管理，满足端到端的 QoS 性质，动态的配置集成，为开发大规模以网络为中心的应用提供软件工程方法和原则，为实时嵌入式系统领域提供中间件支持等。

2.2 Web Services 技术

Internet 的本质包括非集中式、普遍的互联性、开放性，以及动态变化性。但是传统 IT 技术的成功发展从另一种角度来看并没有很好地体现 Internet 的本质要求，如各种平台的异构、编程语言的差异、防火墙的阻隔、没有在互操作性上一致的协议等。这些都给在 Internet 上部署和使用软件、实现大范围应用系统的对接、装配和快速定制带来了迫切需要解决的问题。正在兴起的 Web Services 技术提出了软件是服务的概念，被认为有望能解决上述的某些基本问题。

Web 服务是一种基于 Internet 的应用，这种应用完成一个或一组特定任务，并能够与其它 Web 服务结合，共同完成一个工作流或商业事务[3]。一个完整的 Web 服务的体系结构需要有一系列的协议规范来支撑。Web Services 协议栈如图2所示。

未来可能的服务	未来可能的协议	管理	服务质量QoS	安全
路由、可靠性、事务等	正在制定			
服务工作流描述	WSFL			
服务发现/集成	UDDI			
服务描述	WSDL			
服务调用	SOAP			
网络	HTTP, FTP, MQ, IIOP, SMTP, etc.			

图2 Web Services 协议栈

在 Web Services 协议栈[4]中，最下面网络层是已经定义好的并且广泛使用的标准网络协议；而中间四层是目前开发的 Web Services 的相关标准协议，包括服务调用协议 SOAP、服务描述协议 WSDL 和服务发现/集成协议 UDDI 以及服务工作流描述语言 WSFL；最上面两层描述的是更高层的待开发的关于路由、可靠性以及事务等方面的协议；右面部分是各个协议层的公用机制，这些机制一般由外部的正交机制来完成。

Web Services 协议栈是模块化的，开发人员可以使用自己所需的部分 Web Services 协议实现 Web 服务，而且随着 Web Services 技术的发展，能方便地集成更多的使用新协议的模块[5]。比如在目前状况下，可能只要使用 WSDL 和 SOAP 就能架构一个符合规范的 Web 服务了。

Web Services 的技术理论与传统的 DOC 中间件技术没有本质区别。DOC 技术的特点是由它架构的系统的组件是一组软件对象，而 Web Services 技术把对象组件的概念扩展到服务，所有组件都是服务，这些服务通过自描述的标准协议进行互操作。Web Services 的松耦合性使得它更适合高层抽象和应用，它的强大能力在于它可以集成和扩展而不是取代现存的中间件解决方案[6]。Web Services 与现存的中间件技术是相辅相成的，其基于 XML 标准的特点，不仅在应用、操作系统和硬件平台方面，同时也在其它中间件系统方面，都可以解决多样性和异构性问题。所以有一种观点认为 Web Services 是中间件的中间件。

Web Services 要实现 Internet 上的商业应用的集成有两方面的挑战。

(1)安全性问题。Web 服务是一种分布在 Internet 上的松散耦合的服务组件，因此对商务应用来说，为这些服务提供足够的安全机制十分重要[7]。Web Services 需要对诸如授权认证、数据完整性(比如签名机制)、消息源认证以及事务的不可否认性等运用规范的方法来描述、传输和交换。

(2)服务质量(QoS)的保证。在 Internet 范围内实现跨服务的事务[8,9]、满足 QoS[10]是 Web Services 需要解决的主要问题。要实现复杂商业应用的集成，必须提供服务端的 QoS 性质描述。标准描述语言描述的将不仅仅是服务界面，它将被延伸到 Web 服务的聚合、跨 Web 服务的事务、工作流等，而这些又都需要服务质量(QoS)的保障，现在研究人员正在开发服务的 QoS 接口描述和交互模型[11]。另一方面，商业应用是长时间运行的并涉及组成应用的各 Web 服务之间的交互，因此 Web Services 必须支持业务流程的表示和有状态的服务(因为有状态的服务依赖客户端的一系列方法调用控制其状态)，提供这方面的标准协议。

2.3 网格技术(Grid)与开放网格服务体系结构(OGSA)

网格是一种分布式计算基础设施(Infrastructure)，用来实现在动态的、跨组织域的虚拟组织(VO)内实现协同的资源共享和问题求解，所谓虚拟组织就是一些个人、组织或资源的动态组合[12]。单从逻辑功能的角度来看，网格是一种中间件系统[13](虽然全面地看网格不仅仅是中间件，它还需要物理基础设施的支持并希望构筑下一代 Internet 平台[14])。网格面临的环境是一个分布异构、松散耦合、动态变化的环境。网格体系结构中最重要最有代表性的是五层沙漏结构[12]，如图3所示。

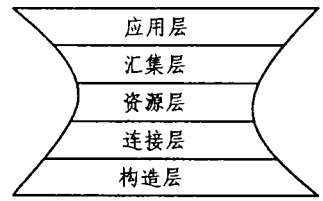


图3 沙漏形状的五层网格结构

五层沙漏结构是一种以“协议”为中心的体系结构，强调协议在网格的资源共享和互操作中的地位。为此根据与底层具体物理资源的距离，从下到上将网格划分为五层，分别是

构造层(fabric)：向上提供网格中可供共享的资源，它们是物理或逻辑实体。

连接层(connectivity)：网格中网络事务处理通信与授权控制的核心协议。

资源层(resource)：对单个资源实施控制，与可用资源进行安全握手、对资源做初始化、监测资源运行状况、统计与付费有关的资源使用数据。

汇聚层(collective)：将资源层提交的受控资源汇集在一起，供虚拟组织的应用程序共享、调用。

应用层(application)：网格上用户的应用程序。

我们可以看到 Grid 的分层结构与 DOC 中间件各层有着类似的对应关系。这种结构更有利于分布式系统的扩充和进化。以 Globus 为代表的 Grid 技术提供了安全、资源管理、资源发现、信息管理、数据管理等原则和机制。这些技术支持在动态虚拟组织中共享和一致地使用不同的资源——即利用地理上和组织上分布的组件构建虚拟计算系统，并将这些虚拟计算系统充分地整合以获得期望的 QoS[15]。

与此类似的是 e-business 的不断发展也同样需要跨越分布式的、异构的、动态的“虚拟组织”整合商业服务，这种虚拟组织由单个企业内部的不同资源和/或外部共享资源以及服务提供商组成。而在商业虚拟组织中获得所需的 QoS 同样成为关键问题[15]。开放网格服务体系结构(OGSA)为实现商业应用的集成开辟了另一条道路。

OGSA 以两项技术为基础：一是 Grid 技术(其代表性解决方案是 Globus 工具包)；二是 Web Services 技术，它已成为一个访问网络应用的基于标准的通用框架。

虽然 Grid 技术是顺应 e-science 的要求发展起来的，但正如上面所述，它的特点也十分适合企业计算的要求。Grid 概念对于企业计算的贡献，主要不是因为它是一个提高性能的方法，而是在于它为构建可靠、可扩展、安全的分布式系统提供了一个良好的解决方案。但是要把 Grid 技术扩展到分布式企业计算领域，还需要支持良好的通用性和灵活性，因此需要融合 Web Services 技术。OGSA 希望获得 Web Services 的属性，例如，服务描述和发现；从服务描述自动生成客户端和服务器端代码；将服务描述与可互操作的网络协议绑定；与正在

出现的更高级的开放标准、服务和工具兼容;以及广泛的商业支持等。OGSA 吸取了 Grid 和 Web Services 的经验和优势,以虚拟化的网格服务为中心是其一大特色。网格服务是一个支持可靠的安全调用、生命周期管理、通知、策略管理、信任状管理和虚拟化的有状态的(潜在短暂的)服务实例。而服务的虚拟化能够使所有的网格服务都基于一组相对统一的核心接口来实现,因此可以很容易地构造出具有层次结构的更高级别的服务,这些服务可以跨越不同的抽象层次,以一种统一的方式集成;其次,虚拟化使得多个逻辑资源实例可以映射到相同的物理资源上,在对服务进行组合时,不必考虑具体的实现,可以以底层资源组成为基础,在虚拟组织中进行资源管理,通过网格服务的虚拟化,可以将通用的服务语义和行为,无缝地映射到本地平台的基础设施之上。其结果是,OGSA 将会带来一个基于标准的公共分布式服务系统,该系统支持创建现代企业和跨组织计算环境所要求的复杂的分布式应用。

3 未来中间件研究的挑战

中间件的核心目标是集成。随着网络和信息技术的发展,这种集成的内容、范围、复杂性都在不断扩大:不仅要集成应用的功能属性,还要集成 QoS 等非功能属性;不仅要实现静态的集成,还要实现动态的集成;不仅要在紧耦合的嵌入式分布系统中集成,还要在 Internet 范围内的企业应用系统中集成。而未来的中间件所面临动态的、异构的、分布互联的以网络为中心的环境给中间件的集成目标带来了巨大的挑战。这种动态性、异构性、分布互联性将渗透到系统的各个方面。

动态性:资源环境和用户需求是动态变化的,异构组件将要动态的加入/退出、生成/消亡,组件之间的交互模式,系统调度管理模式等都将动态变化;

异构性:硬件设备的品种、所使用的技术将日趋多样,平台、语言的异构,软件体系结构、开发范式、运行模式的异构等;

分布互联性:应用本身是分布的,它将要求把各部分互联集成起来,实现应用目标。计算机之间,计算机与其它物理设备之间连接方式越来越多,网络的规模和复杂性将日益扩大。

在这种复杂的条件下,如何提供异构组件、平台之间无缝集成;如何屏蔽局部错误,使系统性能不受影响;如何在系统运行过程中动态配置;如何在资源紧张或变化时保证端到端的系统范围的 QoS—这些都是未来中间件所要面对的问题。下面我们将从几个角度来探讨未来分布式系统的需求对中间件技术提出的挑战。

3.1 满足用户多种端到端的 QoS 要求

下一代以网络为中心的应用系统必需满足用户多种端到端的 QoS 要求。一方面,不同用户运行同一应用时,根据各自当前的配置会有不同的 QoS 需求;另一方面,即使是同一个用户,在不同时间运行同一个应用也会有不同的 QoS 需求,比如根据当前的操作模式或外部因素。用户不断改变的 QoS 需求是下一代分布式应用系统的特点,因此需要开发一种中间件机制,满足这种要求。多种合同(contract)为解决这方面的需求提供了一种策略[16]。合同是用户与系统基础之间的一个契约,它说明了某种服务级别的确保程度(如定义对不同的用户需求提供何种级别的 QoS 保证,以及相应的适应性行为)。

为了适应用户需求的动态变化,下一代中间件中还应包含一种广泛的与用户协商的能力,提供控制协商行为的机制来动态定制合同,保证系统的 QoS。实现这一点最有效的方式是这些机制应该是用户友好的,比如用户或管理员能通过简单地提供一张需求列表来与系统进行协商。而协商机制很可能是领域相关的甚至是用户相关的。

3.2 满足系统的总体性能需求

全局资源管理是复杂的分布式应用的必然要求。如 DOC 中间件体系中的公共服务层,Web Services 中的 WSFL,Grid 中的汇聚层都是处理系统范围内的提供端到端性能的中间件。现在已经广泛认识到未来的系统是通过集成组件实现的。尽管把这些组件连接起来十分容易,但最后系统的可用性主要依靠各部分组装后的总体性能[17],所以即使各部分组件运行良好且组件之间的接口都是标准的,组合起来实现总体的端到端的功能和 QoS 依然是一大挑战。而且未来系统的上层应用需求和下层运行环境条件都会动态地端到端地变化,应用必然还将要求系统能:1、同时满足应用的多类 QoS 性质,每类 QoS 性质有多个级别;2、能组合、权衡、协调、管理各种 QoS 性质,实现系统总体的 QoS。因此,实现端到端的总体需求是下一代中间件发展的重点。总体需求要求中间件能:1、收集系统范围内资源的必要信息;2、提供全局资源管理的机制和政策。中间件本身不能管理系统底层的资源——除非底层资源管理和实施机制能提供管理的接口——但中间件能提供协调机制和政策,驱动底层的独立资源管理器协同管理。协同政策的作用是指导资源管理的决策,协同机制的作用是执行所作出的决策。而且根据政策所作出的决策应该能允许多种 QoS 级别。此外,未来的基于中间件的系统将是复杂的多层次系统,随着这些系统的发展演化,甚至会出现系统的系统,要在系统的演化中保证集成的复杂性不会成倍上升,未来中间件需要在更高的抽象级别上实现全局资源管理,组件提供一个统一的虚拟的接口将是一种解决方案,当前 OGSA 技术中的虚拟化服务已经开始了这方面的实践。实现系统的总体性能需求的研究方向包括:

1、接入控制机制:允许或拒绝某些用户对资源的访问;

2、资源预留、数据复制:保证某种服务级别;

3、协同策略和政策:协同分配各种分布的资源,优化各种 QoS 性质;

4、实施机制:以适当的粒度,规模,性能实施。

3.3 能够控制系统运行时的行为

从系统的生命周期来看,管理 QoS 的决策,可以在三个阶段作出:设计阶段、配置部署阶段以及运行时阶段。其中,运行时的决策是最具有挑战性的,因为要求在尽可能短的时候内作出。而总的来说,这方面的研发经验也最少,但这个领域是最接近高级中间件概念的。这个领域的研究重点是需要提供度量(监控)、报告、控制、反馈、稳定、(系统行为的)转变机制[18],使得应用和系统具有适应性的运行时行为:如动态的重新配置,协调的性能递减。无论从单个应用的角度还是从整体系统的角度来看,这六个基本机制在提供端到端的 QoS 中都扮演着十分重要的角色。架构这样一个具有 QoS 适应性的运行时环境,一个关键的组成部分是要有一种测量以及反映资源状态的公共中间件服务(如 Grid 技术中的 GRIS,QuO 项目[16]中的系统条件对象),并且这种中间件应面向多种粒度。上述六种机制不仅能使应用和系统在资源不足时拥有性能平滑递减的能力,还有望支持其它多种可能的行为,而不必改变应用的基本实现结构。

3.4 实现集成需求

中间件的基本功能是集成，这种集成不仅包括静态地集成，还包括运行时动态地集成，为满足集成需求，未来的中间件系统将需要解决下列问题：

1、标准的组件接口。要实现异构组件之间的互操作性，就必须建立通用标准协议，开发构建应用系统的关键组件（如操作系统、网络管理、安全、数据管理、资源管理等）的通用标准接口。由于这些组件都有各自的 QoS 解决方案，现在所要做的是把这些解决方案集成到一个通用接口（如 OGSA 和 Web Services 中的用 WSDL 描述服务接口）中去，使用户或应用开发者能把他们集成起来，鉴别出何种条件下哪个组件的属性是占主要地位的，对各独立属性权衡管理，从而得到合适的综合属性。现在有一种趋势，就是组件将提供两种接口，即功能性接口和非功能性接口（如 QoS 接口），中间件也将分别管理应用的功能属性和非功能属性[16]。这两种属性的分离，使得 QoS 性质能独立改变，更有利于系统的灵活配置。当前的中间件大多仅提供了端到端的功能集成和组件级别的 QoS，系统范围的无缝 QoS 集成还没有一个很好的解决方案，因此这些中间件仅对资源充足的环境有效。

2、高级集成框架和工具。组件的集成离不开系统的集成框架，通用接口的目的是为了支持各部分属性的权衡，综合。这就要求研究高级的集成框架、工具、机制[18]。它们包括如下特点：

(1)能允许不同的功能和非功能属性，通过权衡机制实现集成。

(2)能理解多种 QoS 请求，并且这些请求能在每个相关的通用接口中翻译转换传递，这种高级的集成框架还应该能保证正确综合那些潜在的功能。

3、通用开放的体系结构。集成框架应该是通用开放的体系结构的实现（此处的框架和体系结构都是软件工程中的概念），例如 globus 框架已开始融入到 OGSA 体系结构中。一个通用开放的体系结构意味着组件的差别（例如在可见性和可访问性上的差别）仅来自于与所有者、隐私和安全有关的策略控制，而不来自于交互机制。这使得在整合多层的分布式系统过程中，无论是对高层系统还是低层子系统，都能以统一的方式架构，从而减少集成的复杂性，增加系统的伸缩性。通用开放的体系结构还要求能兼容现有与未来的技术，这也是使其能推广开来的必要条件（正如 Web Services 技术）。此外，为了满足动态集成的要求，未来系统还要求中间件体系结构提供更高效的组件、资源、服务的动态发现和绑定机制。

3.5 满足系统的适应性需求

现在许多系统往往在资源供应符合设计要求时工作得很好，但只要有一些异常，系统就会彻底崩溃。这些系统缺乏足够的灵活性，许多适应性工作都要由端用户或管理员来完成。为了使系统具有足够的适应性和灵活性，我们希望在系统资源不足的情况下：

1. 能通过自动重新配置获得所需的资源；

2. 若无法获得，能自动平滑地递减 QoS 质量，而不是一下子崩溃。

要实现重新配置，需要有"控制互操作"和管理的机制。现在对互操作的研究大多集中在组件之间的数据互操作和调用互操作上，而对端到端的整体系统行为的控制机制研究得很少。因此首先需要在个体资源中实现控制互操作能力；然后，通过基于中间件的资源管理服务实现全局的互操作机制，控制整体系统行为。为了实现控制互操作能力，未来的中间件有可能使用具有适应性（adaptive）和反射性（reflective）的解决方案。所谓适应性的中间件[16]，它的功能特性和与 QoS 相关的非功能特性将允许随资源环境的变化静态或动态地改变。而反射性的中间件[19]则更先进，反射性中间件使用抽象元接口（meta-interface）提供一种称为因果相连的自表示（causally connected self-representation）的能力。所谓因果相连的自表示指的是元接口（指对组件内部结构的抽象表示）的改变将导致组件内部实现的变化，反之亦然。因此，反射性中间件技术允许组件根据外界环境的变化自动检查自身所实现的功能，并自动地调整和优化这些功能。反射技术使组件内部组织和构造机制对于其他组件和应用程序都是可见的并且是可操纵的。因此，反射性的中间件技术支持更高级的适应性行为，同样也允许对它采用更加动态的控制策略。

适应性有两种基本类型[16]：1、在应用层以下改变，保证所需的服务级别；2、在应用层改变，或者重新恢复服务级别，或者提供新的服务级别。因此，这两个基本适应需求决定了未来的中间件应该提供如下功能：

信息搜集方面：提供持续收集并快速分析资源信息的机制；

行为实施方面：实现资源重新分配，重新配置，满足多级别的 QoS，控制适应性行为的机制；

决策方面：用多种政策和策略配置上述机制，使之能适应变化的环境。

总结 网络和计算机技术的迅速发展给中间件领域的研究和应用带来了繁荣，出现了许多相互竞争又相互补充的中间件技术，如分布式对象计算技术（DOC），Web Services 技术、网格技术（Grid）以及它的未来 OGSA 等。DOC 中间件经过了十多年的发展，已经成为主流，其他新生技术也在不断发展。中间件技术的问题主要是集成，而随着应用系统日益以网络为中心架构，这种集成的复杂性进一步增加，给中间件的研究和发展带来了许多挑战，我们要解决系统的 QoS 问题，要更加关注系统的整体性能，而不仅仅是各部分的性能；我们要解决组件或服务的动态发现、资源和用户需求的动态变化的问题，而不是向传统中间件那样仅满足静态环境；我们要解决大范围网络中所强调的安全性，可靠性，授权认证等问题；我们还有研究实现这种集成的软件工程方法论等等。未来的基于中间件系统将是复杂的多层次系统，随着这些系统的发展演化，甚至会出现系统的系统，这种系统会呈现出与自然界中复杂事物类似的自相似性和自适应性，这要求我们从动态的、系统论的视角来看问题，不仅要关注局部，更应着眼于系统的全局。中间件技术方兴未艾，不论是学术界还是企业界，都将有许多工作需要去要做。

参 考 文 献

1 Schmidt D, et al. Towards Adaptive and Reflective Middleware for Network-Centric Combat Systems, Crosstalk. Nov. 2001

2 Schmidt D C. The ADAPTIVE Communication Environment: An Object-Oriented Network Programming Toolkit for Developing Communication Software. In: 12th Annual Sun Users Group Conf. San Francisco, CA, 1994. 214～225

3 Aoyama M, et al. Web services engineering: promises and challenges Software Engineering, 2002. ICSE 2002. In: Proc. of the 24rd Intl. Conf. on, 2002. 647～648

4 Web services news, articles, and software information. www. webservices. org

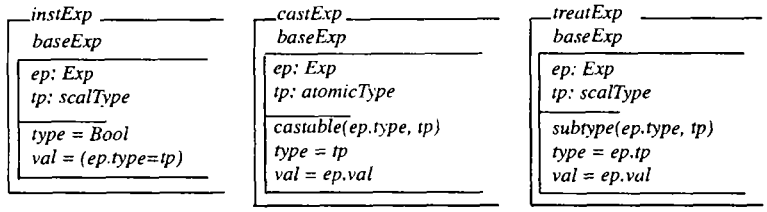
（下转第180页）

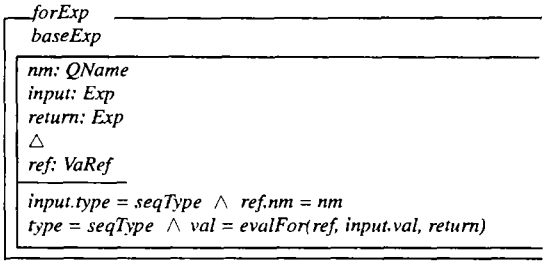
ref。

for 表达式的计算：对输入序列 *input* 中每项，计算返回表达式 *return*；对每次计算所得的结果序列进行连接，形成的单个序列作为表达式的结果值。

函数 *evalFor* 用于计算单变量 *for* 表达式的值。

$$evalFor : VaRef \times seqType \times Exp \rightarrow seqType$$





结论 本文利用形式化描述语言 Object-Z 的符号系统建模 XPath 语言构造。这种面向对象的语义表示不仅具有简洁性，而且便于扩展和复用。采用 Object-Z 的一个好处是：它提供了一个 XML 环境[12]，该环境可自动扩展 Object-Z 类的继承，并生成 UML 例图。利用该工具将有助于语义模型的可视化和易理解性。

我们已经看到 XML 家族语言之间的相互依赖性，将来的一个研究方向是：建模 XML 家族语言的通用语义构造，并复用这些语义构造描述 XML 家族语言的语义。这种方法将有助于说明各个规范之间的相互一致性和协调性。

致谢 感谢联合国大学软件技术研究所对该研究的资助。

参 考 文 献

1 XQuery 1.0 and XPath2.0 Data Model. 2002. http://www.w3.org/TR/querydatamodel/.

2 Boag S, et al. Anders Berglund. XML Path Language (XPath) 2.0, 2002. http://www.w3.org/TR/xpath20/.

3 Clark J. XSL Transformations (XSLT) Version 2.0. 2002. http://www.w3.org/TR/xslt20/.

4 Dong J S. Formal Object Modelling Techniques and Denotational Semantics Studies: [PhD thesis]. University of Queensland, 1995

5 Dong J S, Duke R, Rose G. An object-roiented approach to the semantics of programming languages. Australian Computer Science Communications, 1994, 16

6 Duke R, Rose G. Formal Object Oriented Specification Using Object-Z. Macmillan, 2000

7 Draper D, et al. XQuery 1.0 and XPath 2.0 Formal Semantics, 2002. http://www.w3.org/TR/query-semantics/.

8 Hinchey M. The Object-Z Specification Language. Kluwer Academic Publishers, 1999

9 Cowan R T J. XML Information Set, 2001. http://www.w3.org/TR/xmlinfoset/.

10 Biron A M P V. XML Schema Part 2: Datatypes, 2001. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

11 Fernandez M F, et al. Scott Boag, Don Chamberlin. XQuery 1.0: An XML Query Language, 15 Nov. 2002. http://www.w3.org/TR/2002/WD-wquery-20021115/.

12 Sun J, Dong J S, Liu J, Wang H. Obiect-Z Web Environment and Porjections to UML. In: WWW-10: 10th Intl. World Wide Web Conf. refereed papers track, ACM Press, May 2001. 725～734

13 Thompson H S, Beech D, Maloney M, Mendelsohn N. XML schema Part 1: Structures, 2001. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/.

14 Sperberg-McQueen C M, Bray T, Paoli J. Extensible Markup Language (XML) 1.0 (Second Editon), 2001. http://www.w3.org/TR/REC-xml/.

15 Wadler P. A formal semantics of patterns in XSLT, MIT Press, JUNE 2001

(上接第159页)

5 Curbera F, et al. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI IEEE Internet Computing, 2002, 6(2): 86～93

6 Vinoski S. Where is is middleware. IEEE Internet Computing, 2002, 6(2): 83～85

7 Clark D. Next-generation web services. IEEE Internet Computing, 2002, 6(2): 12～14

8 Sahai A, Machiraju V, Ouyang J, Wurster K. Message tracking in SOAP-based Web services. Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP, 2002. 33～47

9 Preuner G, Schrefl M. Integration of Web services into workflows through a multi-level schema architecture. Advanced Issues of E-Commerce and Web-Based Information Systems, 2002. (WECWIS 2002). In: Proc. Fourth IEEE Intl. Workshop on, 2002. 51～60

10 Curbera F, et al. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing, 2002, 6(2): 86～93

11 Foster I, Kesselman C, Nick J M, Tuecke S. The Physiology of the Grid. Computer, 2002, 35(6)

12 Foster I, Kesselman C, Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications

13 Johnston W, et al. 关于计算网格的论述. http://www.itg.lbl.gov/～johnston/Grids/DOE-Science-Grid-PImeeting-Jan, 2002. 1.ppt. http://www.itg.lbl.gov/～johnston/Grids/DOEScienceGrid.and.ComputationalScience.ppt

14 Foster I. Internet Computing and the Emerging Grid. NatureWeb Matters, Dec. 2000. http://www.nature.com/nature/webmatters/grid/grid.html

15 Foster I, Kesselman C, Nick J M, Tuecke S. The Physiology of the Grid. Computer, 2002, 35(6)

16 Loyall JL, Gossett JM, Gill CD, et al. Comparing and Contrasting Adaptive Middleware Support in Wide-Area and Embedded Distributed Object Applications. In: Proc. of the 21st IEEE Intl. Conf. on Distributed Computing Systems (ICDCS-21), Phoenix, Arizona, 2001

17 Narain S, Vaidyanathan R, Moyer S, et al. Middle-ware For Building Adaptive Systems via Configuration. ACM Optimization of Middleware and Distributed Systems (OM 2001) Workshop, Snowbird, Utah, June, 2001

18 Schantz R E, Schmidt D C. Research Advances in Middleware for Distributed Systems: State of the Art, 2002. www.ifip.tu-graz.ac.at/TC6/events/ WCC/WCC2002/papers/Schmidt.pdf

19 Blair G S, Costa F, Coulson G, Duran H, et al. The Design of a Resource-Aware Reflective Middleware Architecture. In: Proc. of the 2nd Intl. Conf. on Meta-Level Architectures and Reflection, St.-Malo, France, Springer-Verlag, LNCS, 1999. 1616