基于CCM的应用服务器及关键技术研究*)

许 峰 陈 勇 黄 皓 谢 立
(南京大学计算机软件新技术国家重点实验室 南京 210093)
(南京大学计算机科学与技术系 南京 210093)

摘 要 CCM是CORBA3.0规范中提出的服务器端构件标准。本文在深入研究CCM的基础上,结合CCM和应用服务器技术的优点,提出了一个易于管理、可重用、可伸缩、健壮的基于CCM的应用服务器中间件集成框架,并对其关键技术进行了深入探讨。

关键词 CORBA构件模型,构件运行环境,模型驱动体系结构,构件组装

Research of CCM-based Application Server and Key Technologies

XU Feng CHEN Yong HUANG Hao XIE Li
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)
(Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract CCM is a server-side component standard defined in CORBA3.0 specification. By researching CCM deeply and combining the advantages of component and application server technology, this thesis puts forward an easy-managemental, reusable, flexible and robust CCM-based middleware integrated framework and discusses the key technologies in details.

Keywords CORBA component model, Component runtime, MDA, Component assembly

1 引言

随着Internet技术的成熟和宽带网络技术的发展,分布异构环境下企业级应用的需求越来越复杂,要求应用系统在结构上提供更好的灵活性、适应性、可移植性、可伸缩性和可维护性。为适应上述要求,分布式计算环境得到迅猛发展,软件构件技术和中间件技术成为分布式系统的关键技术。

构件的存在某种程度上极大地依赖了框架技术或基础设施、计算平台,只有在适当的框架中,软件才有可能被抽象和隔离,最终成为构件。因此,单独讨论构件是十分困难的。框架不是操作系统、数据库或网络协议,也不等同于应用,而是在特定意义上的构件运行容器,层次上界于应用和基础设施之间。框架提供对于相似问题的一种统一的解决方案,框架的最终目标是能够动态地组装构件,实现软件的"即插即用"。

本文比较了当前应用服务器采用的主流构件技术,在深入研究CORBA构件模型规范的基础上提出了基于CORBA构件模型的应用服务器框架,最后对该应用服务器的关键技术进行了详细讨论。

2 主流构件技术比较

应用服务器通过把用户接口、商业逻辑和后端服务分割开来,将一个应用从Web服务器和数据库中分离出来,为处理大量的用户与事务提供了一个更为结构化、更为完美的解决方案。

一个对应用服务器的定义如下[1]:应用服务器,简单而言,就是将业务逻辑从表示逻辑和数据逻辑中独立出来,专门致力于处理大量的业务逻辑与事务对象或构件,并对其进行管理与部署,提供连接会话、均衡负载、线程池、恢复服务以及访问控制等功能的一种软件平台。可以在Web或非Web结构中使用应用服务器,在其上存放服务器端的处理和共享应用逻辑,并可用作处理事务的专门机构。

随着软件技术的发展,当今的应用程序对开放性的要求与日俱增,这些开放性包括了拓扑开放性(分布式系统)、平台开放性(异构的软硬件平台)以及演化开放性(适应不断变化的系统需求)。软件表现构造性和演化性特点促进了软件构件业和集成组装业的形成,软件构件市场初见端倪,CORBA、EJB、COM/DCOM等构件标准的出现为商用构件市场的形成奠定了基础。

2.1 CORBA及CCM

CORBA(Common Object Request Broker Architecture,公共对象请求代理体系结构)[2,3]是由OMG组织制订的一种标准的面向对象应用程序体系规范。CORBA采用标准的对象模型,它通过发布的接口使远程对象调用变得简单可用。CORBA独立于任何语言和平台,并且其ORB核心和服务为对象互操作和系统功能提供了完整而规范的支持。CCM(CORBA Component Model,CORBA构件模型)[4,5]是CORBA3.0规范中的显著特征,它通过定义一些特征和服务,以允许应用程序编程人员实现、管理、配置和使用由在标准环境下的CORBA服务集成的构件来扩展CORBA对象模型,这些CORBA服务包括持久服务、事务服务、事件服务、安全服

*)国家自然科学基金(60473091)和国家"八六三"高技术研究发展计划项目基金(2003AA142010)资助。许 峰 博士研究生,主要研究方向为软件构件和分布式计算;陈 勇 硕士,主要研究方向为软件构件;黄 皓 教授,博士生导师,主要研究领域为软件自动化;谢 立 教授,博士生导师,主要研究领域为分布式计算。

务等[6]。CORBA构件模型标准不只使得服务方更多的软件可重用，而且为动态配置CORBA应用程序提供了更大的灵活性。

2.2 EJB

Sun公司发布的EJB规范[7~9]说明中对EJB的定义是：EJB是用于开发和部署多层结构的、分布式的、面向对象的Java应用系统的跨平台的构件体系结构。采用EJB可以使得开发商业应用系统变得容易，应用系统可以在一个支持EJB的环境中开发，开发完之后部署在其它的环境中。随着需求的改变，应用系统可以不加修改地迁移到其它功能更强、更复杂的服务器上。

2.3 COM/DCOM

COM(Component Object Model，组件对象模型)[10~12]是由Microsoft提出的构件标准。它定义了一些为保证能互操作、客户(一个术语，指需要某种构件的程序)构件必须遵循的标准。COM规范就是一套为构件架构设置标准的文档形式的规范。COM的发布形式是：以win32动态链接库(DLL)或者可执行文件(EXE)的形式发布的可执行代码组成。COM构件是动态连接的，而且是完全与语言无关的。同时，COM构件可以以二进制的形式发布，还可以在不妨碍老客户的情况下被升级成新的版本。

2.4 几种构件技术的比较

表1 上述几种构件技术的比较

	CORBA (CCM)	EJB	COM/ DCOM
跨语言性能	好	差	好
跨平台性能	好	好	差
网络通讯	好	好	一般
公共服务构件	好	好	一般
事务处理	好	一般	一般
消息服务	一般	一般	一般
安全服务	好	好	一般
目录服务	好	一般	一般
容错性能	一般	一般	一般
软件开发商支持度	一般	好	好
产品成熟度	一般	一般	好
可扩展性	好	好	一般

应该说，这三者之中，COBRA标准是做得最完备的。CORBA的特点是内容全面、互操作性和开放性非常好，缺点是庞大而复杂，并且技术和标准的更新相对较慢。

相比之下，Java标准的制订就快得多。Java是Sun公司自己定的，演变得很快。Java的优势是纯语言的，跨平台性非常好。Java分布对象技术通常指远程方法调用(RMI)和企业级JavaBean(EJB)。RMI提供了一个Java对象远程调用另一Java对象的方法的能力，与传统RPC类似，只能支持初级的分布对象互操作。Sun公司于是基于RMI提出了EJB。基于Java服务器端构件模型，EJB提供了像远程访问、安全、交易、持久和生命期管理等多种支持分布对象计算的服务。另外，Sun公司的J2EE(Java2企业版)体系结构提供中间层集成框架，用来满足没有太多费用而又需要高可用性、高可靠性以及可扩展性的应用的需求。目前，Java技术和CORBA技术有融合的趋势。

COM/DCOM技术是Microsoft独家做的，是在Windows中最初为支持复合文档而使用OLE技术上发展而来，经历了OLE2/COM、ActiveX、DCOM和COM+等几个阶段。目前COM+把消息通讯模块MSMQ和解决关键业务的交易模块MTS都加进去了，是分布对象计算的一个比较完整的平台。Microsoft的COM平台效率比较高，同时有一系列相应的开发工具支持，应用开发相对简单。但它有一个致命的弱点就是COM的跨平台性较差。

3 基于CCM的应用服务器

结合CCM和应用服务器技术的优点，本文提出一个基于CCM的应用服务器框架。应用服务器负责给CCM构件运行提供运行环境，构件被应用服务器调用，实现企业的业务逻辑和业务数据，应用服务器负责处理启动构件、实例化构件、调用构件实例、撤销构件实例等流程，以及线程/进程安排、消息分派、事务处理、安全许可等底层工作，而构件则关心它的某个方法被调用时应该处理的业务。

3.1 应用服务器的体系结构

基于CCM的应用服务器，其体系结构如图1所示。应用服务器架构在ORB之上，ORB、安全服务、负载平衡、管理控制台是整个应用服务器的基础设施，为应用程序提供运行平台、安全保护，使系统资源得到合理有效的运用；在管理控制台之上是各种服务(名字服务、事务服务、通知服务等)和业务构件，这些服务在启动时注册到管理控制台，可以被构件使用；容器是CORBA业务构件的运行环境，它同时负责构件生命周期的管理，客户通过ORB或HTTP找到容器为构件提供的接口，利用接口激发构件。客户激发构件时，容器捕获全部消息并对其进行分派。CORBA业务构件与数据库之间通过数据库中间件进行连接，数据库中间件负责企业异构数据的集成，为CORBA业务构件的开发提供统一的数据。

3.2 关键技术

上述应用服务器体系结构中涉及到很多关键技术，其中最主要的有：构件的实现框架、构件容器、应用服务器安全模型、企业数据集成、资源管理及系统集成等。本文接下来针对这些关键技术进行分析。

(1)CORBA构件实现框架(CIF)

在CORBA构件实现的全部过程中，构件的开发人员定义构件支持的IDL接口，并使用CCM供应商所提供的工具实现构件；如此这般，构件实现接下来可以被打包到一个DLL中；最后，利用CCM供应商所提供的装配机制将构件装配到一个构件容器中，这个构件容器是一个通过加载相关DLL来宿主构件实现的独立的进程。这样，构件就在构件服务器中运行并可以处理客户请求了。整个过程可以分为以下几步：设计和定义CORBA构件；编程实现CORBA构件(CIDL)；编程实现CORBA构件业务功能；打包CORBA构件(如有必要进行构件组装)；部署CORBA构件到应用服务器。

(2)构件容器

CORBA构件容器构造在POA(Portable Object Adaptor，可移植对象适配器)的基础上，以便于实现对象实例的激活与撤消，以及资源使用的优化。容器提供与CORBA服务(安全、事务、持续性及事件)的简化接口，以得到ORB的支持。容器使用回调来管理构件实例，并提供空容器供用户自定义应用服务器的框架特性采用不同的管理策略。引用则通过构件HomeFinder、Naming或Trade服务发出。

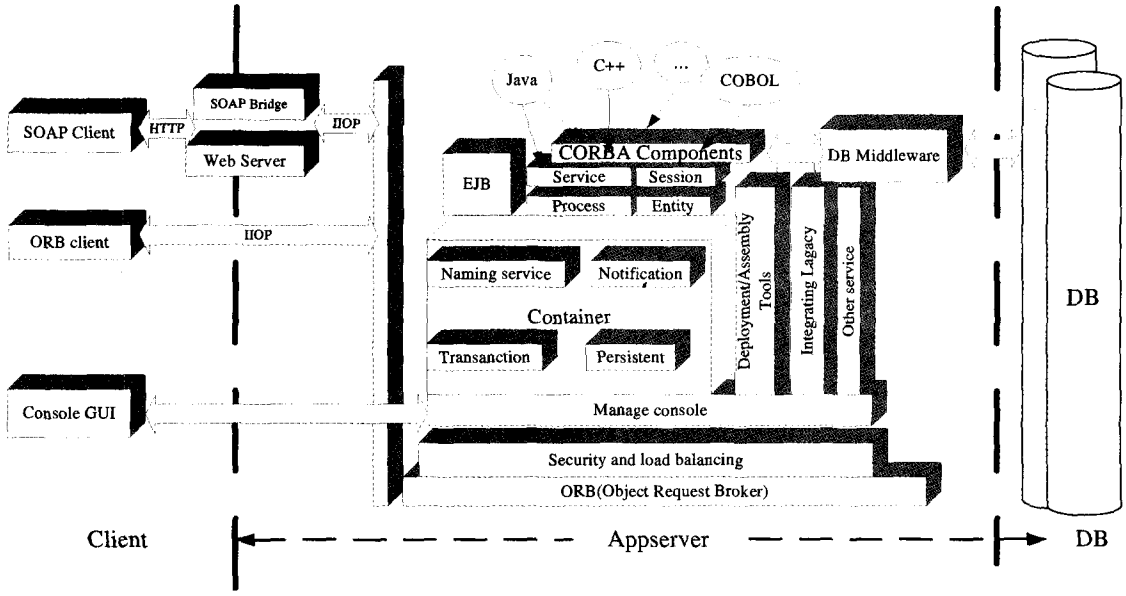

图 1　基于 CCM 的应用服务器的体系结构

(3)应用服务器安全模型

应用服务器安全模型遵循 CORBA 安全服务规范[13]，它将 CORBA 安全对象模型的框架分成 3 个层次：应用层、实现层和管理层。每个层次涉及到不同的 CORBA 安全服务系统对象，每个对象为该层模型完成特定的功能。将每一层的对象按完成的功能分类，分别有认证和安全关联、授权和访问控制、责任 3 大类。

应用层：完成主体身份认证及证书的建立、应用层的访问控制、应用层的审计和防否认活动。在该层，应用可以调用 CORBA 安全服务应用层 API，实现应用层的安全性；

实现层：完成安全对象调用(包括安全关联和安全上下文的建立、调用代理策略、调用审计策略、安全调用策略的实施)和访问控制(包括调用访问策略的实施)；

管理层：完成对域实施的安全调用策略、代理策略、访问策略、调用审计策略和应用审计策略的管理，完成对象在域中的迁移，制定对象实施的安全策略。

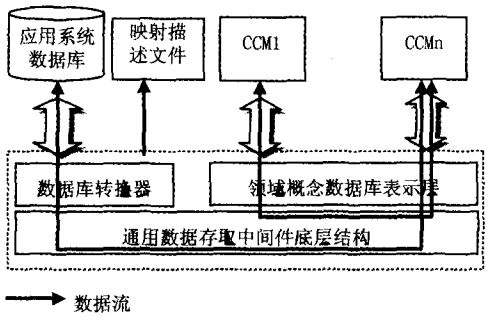

图 2　通用数据存取中间件概念模型

(4)企业数据集成

本文提出并构造针对领域数据的通用数据存取中间件，通过在业务构件和数据源之间建立一个中间层，利用构件技术来屏蔽数据源的差异。数据库中间件向构件提供一致的数据源视图(概念数据库)，完成从实际数据源到数据源视图的转换，并在构件之间充当数据总线。

概念数据库的表示层：主要是将数据视图及提供的服务以一定形式提供给构件。

通用数据存取中间件底层：负责数据库转换器与领域概念数据库之间的通信、内部功能调用。

映射描述文件：概念数据库中定义的数据视图与应用系统数据库中的数据结构不同，因此它们之间需要有一个映射关系，概念数据库才能够知道它所描述的特征在具体应用数据库中存放的形式和位置。

数据库转换器：应用数据库系统中的数据，由数据库转换器读取相应的映射描述文件，根据一定的规则转换后经由领域概念数据库传送给业务构件；业务构件对数据的任何修改都通过领域概念数据库，经由数据库转换器反映到相应的应用系统数据库中。数据转换器的另一个功能是数据库连接池的实现。数据库连接池为应用构件提供所使用的数据源，数据转换器负责维护一个数据库连接缓冲池，缓冲池中是一定数量的数据库连接，每一连接可以连通后台的数据库服务器。连接方式可以是 ODBC、JDBC，或者通过专用的数据库访问 API。数据转换器提供创建/释放数据库连接、打开/关闭数据库连接、控制连接活动的周期以及连接缓冲池中连接的数量等功能。

数据存取服务构件：企业应用中不同数据库对同一特征的描述可能是不同的，设计数据存取服务构件专门用于为用户提供一致的数据视图，其由统一的 IDL 语言描述。将关系数据库中数据以接口属性的形式表示，对数据的各种操作表现为接口的方法，以此对数据进行封装。数据存取服务构件将客户方针对概念数据库进行的操作，通过解释器进行字段匹配后转化为针对相应数据库的 SQL 语句，交给相应的数据库执行，再将结果返回的数据通过解释器进行取值，翻译后将结果传回客户方。

(5)资源管理

资源管理是一个重要话题。如果准备建造一个健壮的大规模系统，进程必须仔细地管理好它们对资源的使用，在 CORBA 环境中更是如此。在这个环境下，应用程序逻辑分布在客户机和服务器之间。困难在于服务器经常代表远程客户机来管理资源，但是一个客户机究竟需要使用某个资源多长时间并不总是清楚的。这里从服务器的角度讨论看来很重要的 3 个话题：内存管理、连接管理和线程管理。内存管理侧

重于伺服对象的服务器端控制。连接管理则解决构件间网络连接的使用问题以及相关的可伸缩性问题。线程管理讨论在CORBA应用程序中使用多个线程，侧重于在CORBA服务器中使用多线程的方法。

(6)系统集成

与EJB集成：CCM规范几乎是在EJB的基础上建模的。与EJB不同，CCM使用CORBA对象模型作为底层的对象互操作框架，从而不局限于特定的编程语言。鉴于这两种技术非常相似，CCM专门定义了这两个标准之间的标准映射。因此，通过使用适当的桥接技术，CCM构件可以表现为对于EJB来说的EJB Beans，EJB Beans也表现为CCM构件。EJB也支持CORBA IIOP作为其通讯框架。CCM和EJB是互补的。有关CCM和EJB互操作的详细信息见文[14]。

与遗产系统集成：应用服务器使用包装器来完成企业遗留系统的集成。我们采用与构件相同的规则来描述包装器，通过在软件包描述器定义一个特殊的实现标记，提供一个链结到可执行应用。这种机制使得遗产系统能够被描述使用，包装器确保了应用能够通过IIOP被访问。包装器可以是一个分布式的管理体系结构的管理对象，可以灵活使用管理工具来完成遗产系统的集成。

3.3 应用服务器的特点

应用服务器框架集成了CORBA构件模型诸多优点，具有以下主要特点：(1)位置透明。构件对象可以相互作用，无论宿主位置如何；(2)多语言支持。构件对象可以使用任何语言实现，映射规则由CORBA规范定义；(3)实现独立。构件对象的通信由标准的接口实现，这样一个对象实现的改变对其它部分不会产生影响；(4)结构和操作系统独立；(5)完整的构件开发、部署机制；(6)集成了CORBA的基本服务，支持持久服务、事务服务、通知服务、安全服务等；(7)支持容错和负载平衡；(8)支持企业数据集成。

结束语 本文在深入研究CCM后提出了一个基于CORBA构件模型的应用服务器框架，并就应用服务器构件的定义与类型、构件的实现框架、构件容器、应用服务器安全模型、资源管理及系统集成等方面的内容进行了详细讨论。由于CORBA构件模型是一个开放的标准，本身仍在不断地完善和发展，而且至今还没有一个完整的商业实现，所以需要进一步研究的工作还有许多。以后的工作主要集中在应用服务器的安全、构件模型的优化、与WebServices集成等方面，并需要进一步实现和完善应用服务器的核心技术和功能。

参 考 文 献

1 Enterprise Application Server. http://dev2dev.bea.com
2 OMG CORBA Specification. http://www.omg.org/corba/whatiscorba.html
3 Ben-Natan R. CORBA--A Guide to the Common Object Request Bröker Architecture. New York:McGraw-Hill,1995
4 OMG CORBA Components. http://cgi.omg.org/cgi-bin/doc?orbos/02-06-33
5 OMG CORBA Component Scripting. http://cgi.omg.org/cgi-bin/doc? orbos/01-12-05
6 Overview of the CORBA Component Model. http://www.ditec.um.es/~dsevilla/ccm
7 SUN. http://develoner.iava.sun.com/developer
8 Sun Microsystem, JavaBeans API Specification Vision 1.01, July 24,1997
9 EJB specification. http://WWW.sun.com
10 Microsoft. http://www.microsoft.com/isapi
11 Microsoft. Distributed Component Object Model Protocol DCOM/1.0 Specification. May,1996
12 Brocjschmidt K. Inside OLE2. Microsoft Press,1994
13 CORBA Secutity Specification. http://www.omg.org/corba/
14 OMG CORBA Components, Integrating with Enterprise JavaBeans. http://www.omg.org/corba

(上接第256页)

第二个验证程序"Driver.c"来自文[12]，BLAST在文[13]中也使用其作为实验对象，它是从Windows的PCI设备驱动程序中摘出来的一段C语言代码，用于在锁的保护下处理中断请求包(IRP)，我们要验证的性质是该程序中锁的使用符合规范(即加、解锁要交替进行)。我们验证工具仅调用了Simplify定理证明工具46次。与我们相比，BLAST调用了定理证明工具260次，并花费了0.06s的时间[13]。第3个C程序"Partition.c"来自于文[14]，它的功能是遍历一个链表，根据链表中每个节点的value域的值是否大于一个给定的值，将节点链到两个不同的链表中，待验证的性质是在程序的循环中的某处指向前趋节点的指针和指向当前节点的指针不相等。我们的方法调用了Simplify定理证明工具68次，而SLAM则调用了Simplify 263次，并花费了9s的时间[14]。

结束语 我们提出了一种C程序断言的全自动静态验证方法，该方法综合使用了程序切片、符号执行、谓词抽象、基于反例的抽象精化等多种思想和技术，从而取得了较好的验证效果。

下一阶段我们的主要任务是提高切片的精度并支持基于变量别名的切片，然后通过对实际程序的断言验证评估我们的方法的效果。

参 考 文 献

1 King J C. Symbolic execution and program testing. Communications of the ACM,1976,19(7):385～394
2 Weiser M. Program slicing. IEEE Transactions on Software Engineering(TSE)1982,SE-10(4):352～357
3 Clarke E M, Grumberg O, Jha S, et al. Counterexample-guided abstraction refinement. In:Proceedings of CAV 1855,2000. 154～169
4 Henzinger T A, Jhala R, Majumdar R, et al. Software Verification with BLAST. 10th Int SPIN Workshop (SPIN'2003)2648 of Lecture Notes in Computer Science,2003. 235～239
5 Chaki S, Clarke E, Groce A. Modular Verification of Software Components in C. In: ACM-SIGSOFT Distinguished Paper in the 25th International Conference on Software Engineering (ICSE), 2003. 385～395
6 Gries D. The Science of Programming. Springer-Verlag,1981
7 Ball T, Rajamani S K. Generating abstract explanations of spurious counterexamples in C programs:[Technical Report]. MSR-TR-2002-09, Microsoft Research, Microsoft Corporation, 2002
8 Zhang Xiangyu, Gupta R, Zhang Youtao. Precise Dynamic Slicing Algorithms. IEEE/ACM International Conference on Software Engineering (ICSE), 2003. 319～329
9 Detlefs D, Nelson G, Saxe J. Simplify: A theorem prover for program checking. http://research compaq com/src/esc/simplify html. 2003
10 Graf S, Saidi H. Construction of abstract state graphs with PVS. In:CAV 97:Computer-aided Verification, LNCS 1254, Springer-Verlag,1997. 72～83
11 Weibenbacher G. An abstraction/refinement scheme for model checking C programs:[PhD Dissertation]. Technischen University of Graz (TUG),2003
12 Ball T, Rajarnani S K. Automatically Validating Temporal Safety Properties of Interfaces. PLDI2001,2001
13 Henzinger T A, Jhala R, Majumdar R, et al. Lazy Abstraction. In:Proceedings of the 29th Annual Symposium on Principles of Programming Languages (POPL),2002. 58～70 ACM
14 Ball T, Majumdar R, Millstein T, et al. Automatic predicate abstraction of C programs. PLDI2001: Programming Language Design and Implementation, 2001