中间件技术比较研究

Comparative research on Middleware Techniques

杨德仁 (上海 华东师范大学 计算机应用研究所 200062)
栾 静 (上海 华东师范大学 计算机应用研究所 200062)
(乌鲁木齐 新疆师范大学 数理信息学院 830054)
顾君忠 (上海 华东师范大学 计算机应用研究所 200062)

摘要:本文归纳了中间件的定义,描述了中间件的分类方法和体系结构,比较了中间件的典型实现机制,论述了服务的共享机制和新型中间件,总结了中间件与系统性能的关系,最后展望了中间件的发展趋势及面临的挑战。
关键词:中间件 体系结构 技术 机制

1 中间件概述

中间件的定义不很统一。如:中间件是平台和通信(应用于分布式系统)机制,这种平台性中间件有利于分布式组件通信和协调;中间件是一个软件层,使分布式系统可编程的软件,使分布式软件的交互变得容易;一种系统软件,用于在开发与集成中"桥接"应用软件与底层的硬件、软件基础设施,并简化集成过程。通俗地讲,中间件与底层网络编程相比,相当于数据库管理系统与文件系统相比,即是一种高级构件块。总之,中间件实现了用分布的网络功能取代传统的操作系统功能。

一个比较全面的定义是:中间件是有助于应用软件与其他应用软件、网络、硬件和(或)操作系统交互或通信的软件,支持 QoS 需求。这种软件为高层编程抽象(如通信、事务和数据集合)提供服务,把程序员从分布式软件实施的复杂中解脱出来。它为改进对用户透明的服务质量、安全性、消息传递、目录服务、文件服务等提供了工具。

2 分类方法与体系结构比较研究

2.1 中间件分类方法

中间件分类因标准不同而异,这些分类标准有通信技术、交互原语(如事件、过程、消息和对象/组件)、协议、编程接口、应用域等。如 IDC 把中间件分成六种,也有文献把中间件分成集成和应用两大类。这些分类方法都是基于不同侧面,因此难免片面或重叠现象。科学的分类方法应该基于完全正交的元模型,这样有利于系统地研究中间件技术。

元空间结构:组件应该有检测和调整基础设施的元空间,元空间由不同元模型组成,每个元模型处理元级的独立和正交的方面。这样有效分离不同关系,简化了由元模型提供的接口。用于描述反射中间件的四个元模型分别是封装、组合、环境和资源元模型。

2.2 中间件体系结构研究

模型是系统的抽象观点,它强调系统设计的主要方法而忽视低级细节,用于说明系统的体系结构。目前,主流的中间件体系结构有分布对象计算分层模型和组件框架模型。

2.2.1 分布对象计算(DOC)分层模型

DOC 模型把中间件分成四层即宿主基础设施中间件、分布中间件、公共中间件服务和特定域服务。

宿主基础设施中间件通过封装和增强本地操作系统的通信和并发机制,创建可复用的网络编程组件,从而简化了网络编程。范例有 JAVA 虚拟机(JVM)和.NET 的 CLR。

分布中间件定义了高级分布编程模型,不同程度地实现了位置、硬件、编程语言、操作系统平台、通信协议等透明性。范例有如 CORBA、RMI、DCOM、SOAP 等。

公共中间件服务定义了一些高级通用服务,使得开发商只用关心业务逻辑。如 CORBA 公用对象服务、J2EE 技术、WEB 服务。

特定域服务:应用域相关的服务,如安全中间件、

财务中间件等。

2.2.2 组件框架模型

组件是一种能被其他程序使用、不能修改、而只能扩展的可执行的二进制软件功能单元。其特点是自我描述、自我包含、可部署的,具有通用性、灵活性和重用性。组件可运行在异构的平台上,与平台与语言无关。它利用 Port 通信(基于事件和方法调用),通过接口可以访问到它提供的服务。

框架是特定的应用环境,指一种完全而又是概况性的应用。而组件框架模型是一种开放的中间件平台,具有灵活性、可扩展性和自适应性,是非结构化的、正交的组件(资源与服务)框架。基于结构上和行为上的反射原理,它在运行时检测环境并重新配置组件。

组件框架模型的优点是具有正交的组件集合,有效地分离了关注点,有利于建立研究模型,如四维元空间模型,其中每个元空间模型都有相应的协议。

分布式组件的基础设施即支撑服务有事务处理、名字服务、消息传递服务、安全。核心组件有目录服务、资源管理、信息发现、检索服务、QOS、安全、网络管理、性能与操作、策略等。

2.2.3 两种框架模型的比较

这两种体系结构各有所长,前者有利于问题的分解,后者有利于组件的重用。前者有多个层面,后者只有一个层面。后者也可以是前者的一个层面。

除了这两种主流体系结构外,还有 Gartner 的三层模型即表示中间件,应用中间件(通用编程平台)和数据库中间件。

2.2.4 中间件模型的形式语义描述

体系结构描述语言(ADL)有 Z、Darwin 和 Wright 等。中间件模型也需要形式化方法支持,需要服务及其接口的理论描述和支持。研究者先后用 Z 语言对 CORBA 组件,用 Z 语言和 Wright 对 COM 组件进行过描述。

一些文献对通用组件作了形式语义描述,提出了实施原型。两级执行者机器(TLAM)是用于说明和论证中间件服务的语义框架,基于执行者计算模型:基级执行者为应用功能建模,元级执行者为中间件建模;这利于核心服务分离的实施。

2.3 中间件模型及其分布式应用范例比较

随着分布式应用系统模型的不断演变,计算资源纵横双向得到共享。表 1 比较了各种分布式应用范例的中间件模型、解耦性和分布性。

3 中间件典型机制比较

3.1 标准编程接口

分布式系统复杂,难于编程。中间件提供了通用的应用编程接口(接口是一些操作的命名集合,用于标记实体如类和组件的行为),实现了位置、并发、复制、错误、移动等透明化,这种编程抽象是一种简单的编程环境,有利于编程。

表 1 分布式应用范例的中间件模型比较

分布式 Paradigms	中间件模型	解耦性	分布性
网络文件系统	FILE	低	差
C/S	RPC	低	较差
B/S	CGI	较低	一般
N – tiers	ROI/RMI	一般	较高
P2P	MESSAGE	高	高
Web 服务	MESSAGE	高	高
移动计算	AGENT	较高	较高
网格计算	OGSI/WSRF	高	高

标准组件通过接口实现连接:CORBA 的接口和连接都用接口定义语言(IDL);DCOM 的接口也用 IDL 而通过接口指针连接;JavaBeans 的接口是 Java 语言,通过事件和侦听连接。

3.2 编程模型比较

几种主要编程模型的调用方向、交互方式等的比较如表 2 所示。

表 2 编程模型比较(*根据具体实施而异)

	请求/响应		共享内存	AGENT
调用方向	Pull	push *	读写内存	自移动
交互方式	同步/异步 *		同步	自主移动
解耦性	低	高 *	低	高
灵活性	较高		低	高
可扩展性	较高		低	高

3.3 通信机制比较

通信协议是定义通信实体之间交换消息的一组规则,几种典型的通信机制如表 3 所示。

表 3　通信机制比较

	RPC	ROI/RMI	MOM	WEB 服务
实体粒度	过程	对象/接口	程序	服务(程序)
协调形式	同步/异步	异步	异步	异步
API 抽象级别	低	高	低	高
解耦性	低	较低	高	高

4　服务、资源与新型中间件

4.1　中间件服务

服务是一组原语(操作),是一套定义良好的行为,是自包含的、无状态的,不依赖于其他服务的状态;从服务提供者和服务请求者的观点看,这些行为形成了一个有机整体。而中间件提供的服务是可见的,这种服务是通过交换信息来提供的网络实体,且能被远程调用。

中间件的基本服务有通信、命名与目录服务、生命周期管理和调度。一般而言,用于系统集成的服务有网络连接、数据流、格式、交互、接口、转换等,这些是低级服务;而用于分布式应用的服务,如命名、发现、数据库连接、事务机制、安全机制等则是高级服务。

中间件服务用于通信、协调、控制、事务支持、安全、名字、表示管理、信息管理、计算和系统管理,也用于保证系统的可靠性、扩展性、综合性能和提供经过包装的高级原语即高级服务。

4.2　Web 服务与 OGSI

Web 服务是一个用 URI 识别的软件系统,其公共接口用 XML 定义和描述了一些可以通过标准化 XML 消息进行网络访问的操作。Web 服务与其他系统按 XML 消息机制交互。Web 服务是基于标准的框架,用于访问网络应用。Web 服务是程序之间通信的模型,用 SOAP 与 Web 服务通信。

Web 服务的特点是基于 Internet 标准的、自我描述的、自我包含的、能被注册和发现的功能单元。用单一数据格式 XML 隐藏异构性:供应商、编程语言、编程模型、系统软件、系统平台中立。其优势是接口定义与具体实施分离,是一种标准机制;为程序之间交互提供了标准,属于集成性中间件。

在 Internet 时代,商业趋势要求应用访问和共享分布的资源和服务,要求中间件服务通过各层计算和通信基础设施向上和向下两个方向扩展。网格是分层中间件服务的集合,为应用提供分布式资源组件和相应的集成机制。实际上网格也是一种应用框架,建立在面向服务的体系结构(SOA)之上。SOA 虚拟化了资源,并且统一了资源、服务和信息。

网格的实施标准是开放网格服务体系结构(OGSA),OGSA 的基础是开放网格服务基础设施(OGSI),OGSI 是一种中间件,其核心是网格服务(在网格环境中为客户端提供的服务),通过 WSDL 接口呈现。网格服务遵循一套规范以实现生命周期管理、检测和通知服务状态变化的 Web 服务。OGSI 网格服务有计算资源、存储资源、网络资源、程序、数据库等,提供了高级接口,隐蔽了底层细节,从而实现了服务与资源的彻底共享,完全实现了设计和实施中间件的初衷。

OGSI 机制:OGSI 利用扩展的 WSDL 和 XML 模式定义,定义了编程模型,引进了状态化的 Web 服务即网格服务,定义了创建、命名和管理服务实例的生命周期的方法、通用的元数据概念,检测和异步通知状态的变化情况等,引进了标准工厂和注册接口以创建和发现网格服务。OGSI 只定义了一套服务原语,并用于构建和组合高级服务。

表 4 比较了 Web 服务和网格服务的典型通信机制。

表 4　Web 服务和网格服务通信机制比较

	Web Services	Grid Services
交互方式	Messages exchange	Messages exchange
接口	Documents/XML	PortType
指针	/	GSH
面向	service	Object

4.3　服务、资源与 WS – Resource Framework

在一切皆服务的时代,OGSI 中间件提供的服务包括了资源,但其接口定义并不规范,违背了服务没有状态的规律。随着 Web 服务标准的不断成熟,WS – Resource Framework(简称 WSRF)应势而生,它分离了服务和资源。WSRF 是 OGSI 的重构和演变,旨在利用新的 Web 服务标准特别是 WS – Addressing、以往实施和应用经验来规范化 OGSI。WSRF 基本上保留了 OGSI 的所有功能,但修正了语法格式,在描述时也采用了不同

的术语。WSRF 主要关心具有状态的资源的创建、寻址、检测和生命周期管理。

相同点：OGSI 和 WSRF 都关心如何通过 Web 服务接口管理状态化的资源，虽然两种方法的状态化资源的模型不同即分别为网格服务和 WS－Resource，但它们提供了基本相同的功能，利用语义上与 WSDL 接口定义相似的机制。创建、寻址、检测、销毁两者的方法也基本相同。

不同点：WSRF 有两个优势。一是它利用了 XML 和 Web services 新标准，标准规范，容易实施。二是返朴归真，自然地分离了信息处理者即服务和状态化的资源。

5 中间件与系统性能

随着应用系统向实时系统、嵌入式、移动计算、普及计算演变，对运行环境和设备状态如位置、资源可用性、带宽、电源等比较敏感，因此要求中间件具有自适应性，即能提供端对端的 QoS。自适应中间件基于元数据模型，在技术上，可将反射技术和组件技术相结合。这种中间件能适应环境变化。

J2EE 和. NET 都是开发自适应中间件的良好平台，两者的比较见表 5。

表 5　J2EE 和. NET 的性能比较

	. NET	J2EE
混合语言编程能力	C# 、VB 等	JAVA
自适应性	有	有
组件	能	能

5.1 基本性能

透明性，为了隐藏系统的复杂性、异构性和简化编程，中间件应该提供透明性，包括位置、语言、设计、实施、软件、硬件等元素的透明性。但这不适合移动计算和普及计算即与环境有关的分布应用。

扩展性，分布式应用系统的扩展性一般通过透明性如访问透明性、位置透明性、迁移透明性、复制透明性等实现。

互操作性，在异构环境中，分布式应用的组件应该有通过接口相互交互的能力，这就要求组件有一套通信和交互的协议。

5.2 高级性能

传统中间件因其黑盒封装策略而满足不了普及计算、移动计算、多媒体等新型应用的需求，缺乏弹性和灵活性。基于反射技术的反射中间件能满足这种需求，根据运行环境自动调整性能，实现了系统在运行时的动态配置，具有一定的容错能力。

与自适应中间件相比，智能代理道高一筹。它具有自主性、响应性、主动性、推理、学习和自适应能力、可移动性、协作能力，被广泛应用于移动计算和普及计算领域。

智能路由应该具有负载平衡。利用迁移透明性和复制透明性实现系统的负载平衡。可采用主动网络技术，根据网络资源和后台服务器资源的可用性，实现低级资源管理、适应负载增长和优化系统性能。

6 中间件发展趋势与挑战

在市场应用的推动下，计算软件开发的演变历程：从数据结构＋算法经历面向模块、面向对象、组件到中间件。网络应用也从网页、动态数据、多层构架到网格服务。分布式应用越来越复杂，对中间件的要求也空前高，如何设计和实施语义完整、性能良好和具有 QoS（包括带宽、调度和安全性等）的中间件及其编程模型，并制定出相应的标准将是分布式计算的长期研究热点。

参考文献

1 K. Geihs. Middleware Challenges Ahead[J]. Computer 34(6), June, 2001.

2 Nikos Parlavantzas, Geoff Coulson, Mike Clarke, and etc. Towards a Reflective Component Based Middleware Architecture[A]. ECOOP´2000 Workshop on Reflection and Meta level Architectures[C], June 2000.

3 Philip T Cox, Baoming Song. A Formal Model for Component－Based Software[A], IEEE 2001 Symposia on Human Centric Computing Languages and Environments (HCC'01)[C], September, 2001.

4 [美] Chris Britton. IT 体系结构与中间件[M]，人民邮电出版社，2003：70－78。

5 胡志远、顾君忠，中间件的体系结构研究[J]，小型微型计算机系统，2003，.24(8)。