计算机科学2000Vol. 27№. 1

Java 分布式计算技术的分析与比较

The Analysis and Comparison of Several Distributed Computing Technologies in Java

王少锋 王克宏
(清华大学计算机科学与技术系 北京100084)

Abstract The Java distributed computing technology represents one of the advanced technology which makes the developing of complicated network applications easy. This paper first gives the rationale of the multi-tier architecture distributed applications, then analyzes and compares with several distributed computing technologies in Java such as Java RMI, Java IDL, EJB and discusses the main characteristics of these technology.

Keywords Distributed computing, Computing technology, Java

一、多层体系结构的应用系统

传统的应用系统基本上是单一结构或 C/S 结构的系统，这种结构的应用系统其构造和维护一般很难，即使是简单的改动也需要对整个系统进行重编译和测试，因此越来越多的分布式应用系统开始采用多层体系结构。三层体系结构的分布式应用系统运行于多台计算机上，一般地，三层结构的应用系统包括用户界面层、事务逻辑层和数据存储层：

用户界面层：用户界面层是应用系统中和用户交互的部分，主要涉及到用户界面的设计和存取性，对于同一个服务器可以有多个用户界面的实现，用户界面层通常调用事务逻辑层中的方法，并作为事务逻辑服务器的客户机。一般地，用户界面层有一个图形用户界面，位于本地机器上，其开发和维护过程较简单。

事务逻辑层：分布式应用系统中包含事务逻辑及执行主要计算功能的部分，事务逻辑层由事务对象组成，这些对象调用数据存储层对象上的方法。一般地，事务逻辑层位于一台共享的机器上以优化资源的使用。

数据存储层：分布式应用中用于存取永久数据及其存储机制的那一部分，数据存储层由封装了数据库例程的对象组成，并直接和 DBMS 交互，例如可能有一个用 SQL 语句实现的方法和数据库交互并取得数据。

传统的 C/S 结构中，客户机端是“胖客户”，各种界面操作、事务逻辑、数据操作逻辑都在客户机端，服务器端一般是一个关系数据库管理系统，在多层结构的体系结构中，大部分的应用系统逻辑从客户机端转移到服务器端，客户机端是“瘦客户”，只包含界面部分，而应用系统逻辑部分独立出来成为分离的部分，位于一个或多个服务器上。

采用多层体系结构的分布式应用系统有很多优点，如提高了应用系统的性能、可伸缩性、灵活性、可靠性、可管理性、可重用性、可移植性等，服务器构件可以根据应用需求迅速地加以修改，且构件在网络中的位置和应用无关，系统管理员可以很容易重新配置系统的负载，因此，多层结构非常适合于大数据量的商业事务系统[1]。

二、Java RMI 技术的特点

分布式系统要求分布于不同主机上的系统各个部分能够通讯，虽然基于 socket 的方法提供了一个基本的通讯机制，但 socket 要求客户机和服务器在交换信息时必须遵守应用层的协议，因此应用开发人员要用 socket 进行通讯还需提供和应用相关的代码，也就是在进行消息传输时提供消息交换协议，而这种协议比较复杂且容易出错。

可以代替 socket 的机制是 RPC(远程过程调用)，RPC 把通讯接口抽象到过程调用一级，开发人员就象在调用一个本地过程一样，调用参数被打包传送到远程目标，但 RPC 的缺点是必须依赖于外部数据表示协

王少锋 博士，主要研究领域为软件重用，智能化软件开发环境，Java 技术等。王克宏 教授，主要研究领域为网络计算环境下的知识处理，分布式计算，Java 技术等。

议 XDR，XDR 是数据描述和编码的标准规范，用于解决不同体系结构的计算机之间数据表示不一致引起的问题，RPC 很难应用于面向对象的分布式应用系统中。

Java 的远程方法调用 RMI(Remote Method Invocation)[2]用面向对象的思想继承和发展了 RPC，RMI 提供了纯 Java 的分布式应用系统的开发方法，也提供了对象之间直接通讯的机制，一个对象可以直接调用其它对象中的方法。用 RMI 开发应用系统一般由服务器和客户机组成，RMI 提供了服务器和客户机之间相互通讯的机制，服务器创建一些远程对象，并等待客户机调用这些远程对象的方法，客户机取得服务器上的一个或多个远程对象上的远程引用，然后调用远程对象上的方法，对于开发人员来说，远程通讯就象是标准的 Java 方法调用。

图1说明了 RMI 如何使用注册器来获得远程对象的引用，服务器调用注册器把一个名字和远程对象绑定在一起，客户机通过名字从服务器注册器查找远程对象，图1也说明了 RMI 系统如何利用 Web 服务器下载对象的字节码到客户机以及从客户机上载对象的字节码到服务器。

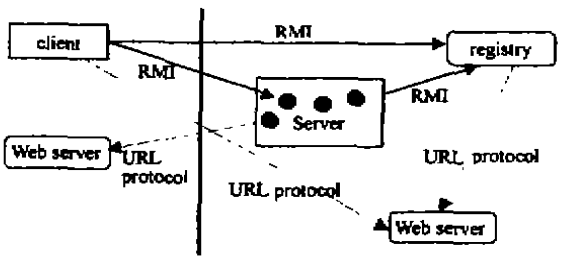

图1 RMI 系统运行环境

和其它类型的应用系统一样，用 Java RMI 开发的分布式应用系统也由接口和类组成，接口声明了方法，类实现了接口中定义的方法，在一个分布式系统中，应用系统的某些实现部分可以位于不同的机器上，RMI 的一个主要特色是其动态装载类的字节码的能力，一个对象的类型和行为可以被传递到远程的虚拟机上，RMI 传递对象时不会改变一个对象的行为，系统可以在运行时确定所执行任务的具体内容。

然而，RMI 也有一些局限性，例如它与 Java 的紧密集成，使得它不能为采用其它语言编写的对象和应用所利用，如果我们要实现 C++或其它非 Java 语言写的服务器，只用 RMI 则无法实现，因为 RMI 提供的只是 Java 虚拟机到虚拟机的通讯机制。

三、用 Java IDL 开发分布式应用系统

Java IDL(Interface Definition Language)是用于分布式计算的技术[3]，可实现网络上不同平台上的对象相互之间的交互，该技术基于通用对象请求代理体系结构 CORBA 规范说明[4]，CORBA 和 socket，RPC，RMI 一样，是用于开发分布式应用系统的一种技术，和其它的分布式对象技术相比，用 CORBA 开发分布式应用具有很多优点：对传统遗产系统的支持，可伸缩性，多平台支持，安全性等，CORBA 使得开发人员在使用新的技术的同时又可以重用传统遗留系统，CORBA 的一个主要特色是 IDL，IDL 是不依赖于语言的接口定义语言，所有支持 CORBA 的语言都有 IDL 到该语言的映射，就象其名字所表示的那样，Java IDL 支持到 Java 语言的映射。

IDL 语法和 C++非常类似，利用 Java IDL，可以在 Java 中定义、实现、存取 CORBA 对象，对于每个 IDL，idltojava 编译器生成一个 Java 接口和其它一些必要的.java 文件，包括一个客户端桩(stub)和服务器端骨架(skeleton)，和 Java 中的接口一样，IDL 接口不包含方法的具体实现，Java 开发人员需在 Java 类中提供对这些方法的具体实现。

图2说明了从客户机发送一个消息请求到服务器的过程，其中客户机也可以是一个 CORBA 对象，客户机不需要了解 CORBA 对象的位置、实现细节，也不需要了解哪个 ORB 用于存取对象。

在客户机端，应用系统包括远程对象的引用，对象引用使用桩方法作为远程方法的代理，这个方法事实上是编写到 ORB 中的，所以调用桩方法会调用 ORB 的连接功能，ORB 会把对桩方法的调用传递到服务器端。

在服务器端，ORB 利用骨架代码把远程调用转换成本地对象的方法调用，骨架需要对调用和参数的格式进行转换，同时，当方法返回时，骨架对结果进行变换，然后通过 ORB 把结果返回给客户机。

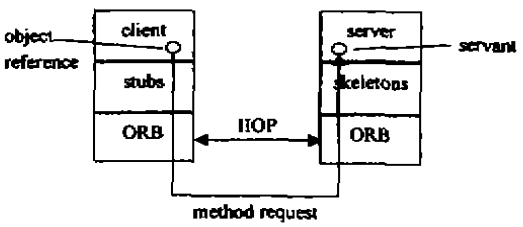

图2 CORBA/IIOP 中客户机和服务器的通讯过程

Java 2平台提供了两种不同的方法来构造分布式

应用系统：即 Java RMI 和 Java IDL，它们具有相似的特征和功能，Java RMI 支持用 Java 语言写的分布式对象，Java IDL 可以和支持 CORBA 的任何程序设计语言，如 C，C++，COBOL 等写的分布式对象交互，这两种方法各自具有不同的特点：

1）100%纯 Java 和对遗产应用系统的支持。Java RMI 是对分布式应用系统的100%纯 Java 解决方法，具有 Java 的"write once，run anywhere（一写永行）"的优点，用 Java RMI 开发的应用系统可以部署在任何支持 Java 运行环境的平台上。

相反，Java IDL 是基于 CORBA 规范标准的技术，可以远程调用非 Java 语言写的对象，因此 Java IDL 提供了对那些用非 Java 语言开发的遗产应用系统的支持。

2）所使用的通讯协议。Java RMI 和 Java IDL 目前使用不同的通讯协议，Java IDL 使用 CORBA/IIOP 协议，IIOP（Internet Inter-ORB Protocol）协议是建立在 TCP/IP 之上的协议，可以使位于不同平台上、用不同语言写的对象以标准的方式进行通讯；Java RMI 目前使用 Java 远程消息交换协议 JRMP（Java Remote Messaging Protocol）进行通讯，JRMP 是专为 Java 的远程对象制定的协议，不过 Sun 和 IBM 已经宣布将来会支持在 RMI 中使用 IIOP 协议以便和遵从 CORBA 规范的远程对象通讯。

3）是通过引用调用对象和/还是通过值调用对象。在 Java IDL 中，客户端通过引用和远程对象交互，即客户机使用桩对远程服务器上的对象进行操作，客户机在运行时并不拷贝服务器上的对象。

相反，RMI 使得客户机既可以通过引用和远程对象交互，也可以把远程对象下载到客户机运行环境进行操作，由于在 RMI 中使用的对象都是 Java 对象，RMI 使用 Java 中的对象串行化功能在服务器和客户机之间传输对象，不过 CORBA 规范的以后版本将包括按值调用对象的功能。

Java RMI 和 Java IDL 各有自己的优缺点，从某种意义上说，RMI 可以看作是 RPC 的面向对象版本，RMI 的最大优势是可以用它来提供100%纯 Java 的解决方案，这意味着构造 RMI 应用系统将比较简单，但这也正是基于 RMI 的应用系统的一个缺点，即只能在 Java 环境中运行，不能充分利用遗产应用系统的资源。

总之，Java RMI 和 Java IDL 均可满足一定范围的用户需求，都适用于一定范围的应用，两者之间存在着重叠，有些应用可使用两者中的任何一种技术进行开发，但对于某些应用来说，采用其中的某一种比采用另一种更为恰当。

CORBA 规范标准导致了分布式应用系统开发的新时代，和 CORBA 紧密关联的是 Java 技术，对企业 IT 部门来说，作为一种创建 CORBA 分布式对象的语言，Java 会越来越重要。Java 的长处是功能强大、简洁和可移植性，而 CORBA 的长处在于其所提供的软件总线和相关的服务，Java 和 CORBA 的结合对于开发分布式系统来说是一个非常有效的方法，这是由于 Java 和 CORBA 解决了分布式计算中的许多技术问题并简化了相关开发和部署过程，CORBA 是异质分布式系统的主要标准，Java 对于开发客户端的功能来说具有很强的可移植性和强大的功能，由于这两种技术的结合，分布式应用系统的开发已变得相对容易。

四、Java 的服务器端构件模型 EJB[5]

98年在美国旧金山召开的 Java One98 开发者大会上，Sun 公司正式发布了业界期待已久的 EJB（Enterprise JavaBeans）1.0版规范说明[6]，在众多的大公司和开发人员中引起了巨大的反应，这标志着用 Java 开发企业级分布式应用系统将变得非常简单，目前，EJB 已经成为 Java 技术研究中的重点。

Sun 公司发布的文档中对 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。

在分布式应用系统的开发中，采用多层体系结构的方法有很多优点，如增加了应用系统的可伸缩性、可靠性、灵活性等。由于服务器端构件可以根据应用需求迅速地加以修改，且构件在网络中的位置和应用无关，因此系统管理员可以很容易重新配置系统的负载、EJB 把 Java 的"write once，run anywhere"思想提到一个新的高度。服务器端构件在构件执行系统内执行，规范说明定义了构件执行系统所需要的服务，遵从 EJB 规范说明开发的构件可以在任何一个支持 EJB 的系统中执行。

一个开发商可以开发一个新的支持 EJB 的执行系统，但通常的做法是供应商对已有的系统进行改进以支持 EJB，可以进行改进以支持 EJB 的系统包括：

数据库管理系统，如 Oracle，Sybase，DB2等；

Web 应用服务器，如 Java Web Server，Netscape Enterprise Server，Oracle Application Server 等；

CORBA 平台，如 Iona Orbix/OTM，Borland VisiBroker/ITS 等；

事务处理监控器，如 IBM TXSeries（CICS and

Encina),BEA 公司的 Tuxedo 等;

构件事务服务器,如 Sybase Jaguar CTS 或 Microsoft Transaction Server 等。

4.1 软构件模型

软构件模型的思想是创建可重用的构件并将其组合到容器中以得到新的应用系统,软构件模型思想已经在软件开发界迅速流行,因为它可以达到以下这些目的:重用、高层开发、通过工具进行自动化开发,简化开发过程等[9],EJB 是软构件模型的一个例子。

EJB 软构件模型给开发者提供了以下的一些支持:构件包含应用系统逻辑;可重用的构件;可伸缩性;资源管理;事务支持;并发性管理。

分布式应用系统中的中间件以其复杂性著称,它不仅涉及到应用逻辑、并发性和伸缩性等问题,也涉及到如何把不兼容的系统组合在一起的问题。EJB 软构件模型解决了中间件开发的复杂性问题[7],它使得中间件开发人员集中于应用系统的逻辑部分,而不用处理同步、可伸缩性、事务集成、网络、分布式对象框架等一些和分布式应用系统有关的复杂的细节问题。

采用 EJB 开发应用系统有很多优点,如:1)标准的 Java 技术使得应用系统可以在许多不同的服务器平台上运行;2)修改应用系统变得容易,对单个构件进行增加、修改、删除等操作,对应用系统体系结构的影响很小;3)应用系统经过划分之后,使得构件之间既相互独立,又可以相互协作,提供给用户的是该用户所需要的构件;4)应用系统的开发变得容易,基本上是即插即用的方式;5)应用系统从本质上说是可伸缩的,可以运行在多线程、多处理机的环境中;6)EJB 可以在新的应用系统中得到重用,减少了开发时间。

4.2 EJB 和其它技术的关系

4.2.1 *EJB 和 JavaBeans 的关系* 很多人往往把 JavaBeans 和 EJB 混淆起来,JavaBeans 提供了基于构件的开发机制[8],一般地,JavaBeans 是可视化的构件,也有一些 JavaBeans 是非可视化的,JavaBeans 可以在多个应用系统中重用,一个标准的 JavaBean 是一个客户端构件,客户端的 JavaBeans 容器可以根据 JavaBeans 的属性、方法、事件的定义在设计时或运行时对 JavaBeans 进行操作,JavaBeans 不一定要用于 Client/Server 系统。

EJB 没有用户界面,并完全位于服务器端,EJB 可以和远程的客户机通讯,并提供一定的功能,根据规范说明,EJB 是 Client/Server 系统的一部分,如果不和客户机交互,EJB 一般不执行具体的功能,EJB 和 JavaBeans 的一个重要区别就是 EJB 提供了网络功能。

和一般的 JavaBeans 一样,EJB 是高度可定制的,对 EJB 进行定制不需要存取源代码,EJB 规范说明定义了编写可重用构件的过程,开发人员将为应用系统开发可重用构件,然后用开发工具把它们连接在一起。例如,在一个库存管理系统中,一个开发人员可能会写一个 EJB 构件来查询库存数据库,另一个 EJB 在某一产品的库存低于某一具体值时会向用户发出警告,还有一个 EJB 可能是对库存的产品进行记数,而这些构件除了用在库存管理系统中外,也可以用在其它的应用系统中,如销售报表系统等。

4.2.2 *EJB 和 CORBA 的关系* 为了保证多个开发商之间的基于 CORBA 的 EJB 产品之间的互操作性,EJB 规范说明定义了 EJB 到 CORBA 的映射,分为四个部分:

1)分布映射——定义了 EJB 和 CORBA 对象之间的关系,以及 EJB 规范说明中定义的 Java RMI 远程接口到 OMG IDL 的映射。

2)命名映射——说明了如何利用 COS 命名服务来确定 EJBHome 对象。

3)事务映射——定义了 EJB 的事务支持到 OMG Object Transaction Service(OTS)v1.1的映射。

4)安全性映射——定义了 EJB 中的安全性特征到 CORBA 安全性的映射。

映射确保了不管哪一种类型的客户机,通过生成相同的字节流,可以和基于 CORBA 的 EJB 服务器进行互操作。

从以上的讨论中可以知道,对于 EJB 服务器来说,有两种类型的客户机可以使用 EJB:

1)EJB/CORBA 客户机——一个使用 EJB APIs 的 Java 客户机,客户机利用 JNDI 确定对象,用 Java RMI 来调用远程方法,其中 CORBA IDL 的使用是隐含的,也就是说,开发人员只使用 Java 代码,开发客户机时可以不用了解 CORBA 及其 IDL 知识。

2)纯 CORBA 客户机——用 CORBA IDL 支持的任何语言写的客户机。客户机用 COS 命名服务来确定对象,用 CORBA IDL 来调用远程方法,用对象事务服务 OTS 来执行事务,其中系统开发人员要创建一个 IDL 文件,即 CORBA IDL 的使用是显式的。

4.2.3 *EJB 和网络计算的关系* 由 Beans 构造的应用系统可以根据用户的需求分解成不同的构件,根据用户当前所需要的功能提供相关的构件,并随着用户新的需求随时下载新的构件,而用户没有用到其功能的构件可以驻留在服务器上,这就是网络计算所倡导的概念。很多人并没有完全理解 Java 的概念,他们认为为了在一个客户端上运行 Java 程序,需要把一个庞大的、可能达几兆字节的 Java 应用程序一次性通过网络传输到客户端,事实上,这也是一些开发人员计

(下转第35页)

行有力的支持:通过构造程序切片,或者对不同程序版本的切片的比较,可以对原程序进行精简,除去界面管理,输入输出处理等我们所不关心的部分,将原程序中分散的关键部分集中进行分析。

类似地,在对软件系统进行重用方面,也可以用程序切片工具进行处理,使用程序切片工具提取原系统中的可用部分,可有效地避免对同一过程的重复提取,避免提取出冗余代码。

结束语 由于篇幅所限,我们无法对程序切片技术的每项应用给出完整的论述和证明,但是毋庸置疑的是,各种切片标准和切片工具已经实际应用到软件系统的开发中,程序切片的思想给予其他软件技术的研究以极大的启发与帮助,同时,程序切片技术也是一门不断发展的技术,不断有新的切片标准,新的算法被提出,以使其更加完善,更加贴近实际应用。相信在以后的研究,将更加证实程序切片技术的价值与前景。

参考文献

1 Binkley D W, Gallagher K B. Program Slicing. In: Advances in Computers, 1996

2 Livadas P E, Croll S. Computer and Information Sciences Department University of Florida. Program Slicing: [Technical Report SERC-TR-61-F]. Software Engineering Research Centre, October 1992

3 Harman M, Simpson D. Department of Computing, University of Brighton. The next 700 slicing criteria Formal Aspects of Computer Science, 1995

4 M Sloane A, Holdsworth J. Beyond Trational Program Slicing. Department of Computer Science, James Cook University

5 Zhao J J, et al. Static Slicing of Concurrent Object-Oriented Programs. In: Proc. of the 20th IEEE Annual Intl. Computer Software and Applications Conference (COMPSAC'96). Seoul, Korea, 1996. 312～320

(上接第27页)

划用 Java 改写旧的应用系统时易犯的错误。

在网络计算环境中利用 Java 的最好途径是由 EJB 提供服务器端的构件,而由 JavaBeans 提供客户端的构件,两者结合在一起,将向"网络就是计算机"之路迈出一大步。

总之,EJB 技术将使得 Java 在分布式计算中的地位得到加强,为基于 Java 的应用系统提供了一个框架,和目前的许多系统和模型相比,EJB 具有许多优越性,种种迹象表明,EJB 有可能成为分布式应用系统的服务器端构件模型的首要选择。

结论 将来的软件开发必定包含三个技术:面向对象、网络及数据库,Java 的分布式计算技术代表的是一种前沿技术,使得复杂的网络应用系统开发变得容易,Java 的设计目标之一是支持开发代码少而又易于理解的复杂系统,对于这个目标来说,Java 所取得的成功领域之一是在分布式计算方面,传统上,分布式应用系统的开发以其复杂性著称,一个简单的分布式计算任务可能需要数页的代码,对于 Java 来说,这些任务中的许多任务已被减少到一条或几条语句,显然由于 Java 技术对分布式计算的强有力的支持,分布式应用系统的开发将变得容易。

参考文献

1 Gupta A, et al. Implementing Java Computing: Sun on architecture and applications deployment. IEEE Internet Computing, 1998, March-April: 60～64

2 Sun Microsystems. Java RMI. Available at: http://java.sun.com/products/jdk/rmi/index.html

3 Sun Microsystems. Java IDL. Available at: http://java.sun.com/products/jdk/1.2/docs/guide/idl/index.html

4 Object Management Group. CORBA/IIOP Version 2.2 Specification. Available at: http://www.omg.org/corba/corbaiiop.html

5 Anne Thomas. Enterprise JavaBeans—— Server Component Model for Java. Available at: http://java.sun.com/products/ejb/white_paper.pdf

6 Sun Microsystems Inc. Enterprise JavaBeans Specification. Available at: http://java.sun.com/products/ejb

7 Rawn Shah. Enterprise JavaBeans: Industrial-strength Java. Available at: http://www.nc worldmag.com/ncworld/ncw-01-1998/ncw-01-ejbeans.html

8 Sun Microsystems Inc. JavaBeans. Available at: http://java.sun.com/beans.

9 Kozaczynski W, Booch G. Component-based software engineering, IEEE software, 1998, 15(5): 34～36