计算机科学 1999Vol. 26№. 7

CORBA 与 RMI 在构造分布式程序中的对比研究*)

A Comparison of CORBA and RMI in the Construction of Distributed System

朱 鹏 王跃华 尤晋元

(上海交通大学计算机科学与工程系 上海 200030)

Abstract This paper first summarizes characteristics and common strongpoint of CORBA and RMI, then compares and analyzes their differences on calling methods, the support to dynamic discovering and dynamic calling, common services, object transmission, easy to develop, support for heterogeneous platforms and heterogeneous languages, and executing speed.

Keywords Distributed system, Object oriented, Java

1 前言

Java 语言[1]对计算机领域研究和应用产生的巨大影响不是偶然的，它近乎完美的跨平台特性满足了当前多平台计算环境的迫切要求，使它成为开发基于 Web 和 Internet 应用的理想语言。与 C++ 相比，Java 要更加容易掌握和使用，因此它可以供更多的开发者使用，也能显著地加速开发的过程。

作为一个在网络计算时代诞生的面向网络计算的语言，对分布计算的支持能力是非常重要的。然而在 Java 的 1.0 版本中对网络计算的全部支持就是一个用于 TCP 和 UDP 编程的网络类库，用这样一个低级的接口来开发分布式的应用程序其复杂性是不言而喻的。为了弥补这个不足，在 Java 的 1.1 版本中引入了 RMI(Remote Method Invocation，远程方法调用)。RMI 本质上是 Java 版的 RPC[2]，通过 RMI 一个 Java 程序可以与调用本地方法相同的方式调用位于网络上其他机器上的 Java 对象的方法，方便了基于 Java 的分布式应用程序的开发。

CORBA[3]是定义分布对象如何交互的一种规范说明，由共同制定开发分布计算标准的 700 余家公司组成的 OMG 控制，它实质上是面向对象技术和 RPC 结合的产物。CORBA 对象可以用 CORBA 软件提供者支持的任何语言 C、C++、Java、Ada、Smalltalk 等编写，这些对象可以存在于 CORBA 软件提供者支持的任何平台上。CORBA 中客户程序可以透明地向远程和本地的对象发送请求和接收应答。

在开发基于 Java 的分布式应用程序时，Java 和 CORBA 都是可行的选择，本文将对这两者的功能、特性和速度进行详细的对比研究。

2 CORBA 与 RMI

CORBA 核心由以下几个部分组成：ORB(Object Request Broker，对象请求代理)使对象可以在分布式环境中透明地接受消息并给出响应；对象服务包括安全、名字、事务等系统级的基本的独立于具体应用领域的服务；公共设施定义了应用级的服务，如复合文档、系统管理等；领域对象是一组对于专用领域有共享价值的对象，例如专门处理银行业务的对象等；应用对象是由供应商开发用于特定问题的解决的产品。

CORBA 中对象的界面是由标准的与语言无关的界面定义语言(IDL)描述的，通过它使对象作到了语言中立。ORB 实际上是一个分布式服务的中介，需要服务的客户程序通过它找到并激活适当的服务对象，并通过它向服务对象发送请求和获得返回结果。对象间的通讯通过 ORB 进行，对象是由对用户程序不透明的对象引用标识的，因此很容易实现对象的位置透明性。在通常情况下，针对不同语言

*)国家"九五"重点科技攻关资助项目(96-737-01-05)。朱鹏 博士，主要研究方向是移动对象、分布对象计算技术；尤晋元 教授，博士导师，主要研究方向是操作系统、软件工程、分布对象计算。

的 IDL 编译器根据对象的界面定义自动生成 Stub 和 Skeleton 程序、它们完成远程调用参数及返回值的打包解包和与 ORB 通讯的细节、向客户程序和对象的实现提供与本地调用相同的界面，因此基于 CORBA 的分布式系统开发类似于集中式系统、十分简便。

OMG 定义了实现不同 ORB 产品互操作的协议 GIOP、GIOP 对应于 TCP/IP 的映射是 IIOP 协议、对应于 DCE 的映射是 DCEESOP。

Java RMI 的设计目的是能够无缝地与 Java 语言的其他方面结合起来，支持分布式程序设计。RMI 中引入了远程对象的概念、远程对象实现了一个远程界面，在这个远程界面中声明了可以被其他对象从远程机器上调用的方法(过程)。远程界面的一部分功能类似于在 CORBA 中用 IDL 声明的界面，Java 开发环境提供的一个编译器能够从远程对象的声明中自动生成它的 Stub 类和 Skeleton 类。

Skeleton 对象运行在远程对象所在的 Java 虚拟机上、用于隐藏网络通讯和与 RMI 运行环境交互的细节、把 RMI 运行系统从网络上接收到的远程请求转换成对远程对象对应方法的调用，完成参数的解包和返回值的打包；Stub 对象运行在需要访问远程对象的程序所在的 Java 虚拟机上，它向客户程序提供远程界面定义的接口，把客户程序的请求通过网络送到远程对象所在机器、完成参数的打包和返回值的解包。

Java 对象与基本数据类型的变量或常量一样，可以直接作为远程对象中供远程调用的方法的参数和返回值。对于普通对象，传递的语义是传值；对于远程对象，传递的语义是传引用。RMI 中远程对象的 Stub 和作为远程方法参数或返回值传递的对象的代码和可以动态地从远程下载，不必事先安装在接收端，因此用 RMI 开发分布式应用程序几乎与编写非分布式 Java 程序一样方便。RMI 还提供了一个简单的基于 URL 的名字服务。RMI 的主要优点在于其可移植性、对象的自动序列化、自动类装入和分布式垃圾收集。

3 CORBA 与 RMI 功能的比较

3.1 共同的优点

CORBA 和 RMI 都继承和发展了 RPC 的思想，即用通常本地过程调用的语义对分布式应用程序间的交互操作进行封装，根据以远程过程方式表示的分布式程序间的交互界面自动生成 Stub 和 Skeleton，由它们负责完成远程请求的打包、发送、和接收、为使用远程服务的程序和提供远程服务的程序提供与本地过程调用相同的界面。对程序设计者来说，编写分布式应用程序与编写本地程序类似：调用远程程序提供的服务与调用本地程序提供的服务相同，编写供远程访问的过程与编写本地过程类似，简化了分布式系统的开发、降低了程序员的工作量。程序员在编写分布式应用程序时不必去学习和使用复杂繁琐的网络 API，原来编写非分布式程序的方法就足够了，因此能够把精力集中在任务分配、界面定义、算法设计等方面。

面向对象的方法与传统的结构化方法相比在软件的分析、设计、编码、维护等方面都有着巨大的优势，这已经成为共识。但在基于 RPC 或网络 API 的分布式系统设计中，分布的系统间的界面仍然是结构化的，其设计过程和设计出的软件具有结构化方法的一切弊端。CORBA 和 RMI 都基于面向对象的思想，分布式程序间的远程访问界面的基本组成元素是对象、远程调用都是对这些对象的方法的调用。它们都支持面向对象的分析设计中的继承、封装、多态等思想，从而为使用面向对象的方法进行全系统设计提供了有力的支持。

CORBA 和 RMI 都支持远程回调，远程回调是指客户方可以把自己的一个过程(函数、方法)向服务端注册，当服务端得知某个事件发生时调用这个过程通知客户端。例如客户端是一个实时股票行情察看程序，需要尽快从服务器上得知行情的变动以刷新显示。对于客户端需要等待服务端某个事件发生的情况，采用远程回调方式的效率要比让客户端定期向服务端查询效率高、反应快。远程回调允许客户程序异步地接收其他对象的事件。

3.2 调用方式

CORBA 支持三种远程调用方式：同步方式、单向方式和异步方式。在同步方式中，客户程序发出远程调用后即进入等待状态，直到远程调用完成收到结果后才继续运行，这种方式与通常的过程调用相同；在单向方式中，客户程序启动一个远程调用后继续向下执行，不等待远程调用的结束，远程过程不向调用者返回结果；在异步方式中，客户程序发出一个远程请求后不等待结果继续往下执行，过一段时间后通过 CORBA 提供的功能调用取得结果，异步调用只能通过 CORBA 的 DII(动态调用界面)进行。CORBA 提供的调用方式多样、灵活，能满足不同类型应用程序的需要，而且开发较简单，但 DII 的使用

比较复杂。

RMI 只支持同步一种调用方式，但是通过在编程时采取专门的措施，可以用同步调用达到与 CORBA 的单向和异步调用近似的效果。对于单步调用方式 RMI 有两种模拟方法，第一种是客户程序在调用远程方法前启动一个线程，由这个线程去调用远程过程并阻塞等待返回，主程序继续向下运行；第二种是修改服务方方法的代码，让它在启动了一个执行原来工作的线程后直接返回。这两种方式虽然都能达到与 CORBA 中单向调用类似的效果，但它们都需要额外的开销（启动线程和等待远程调用的返回），后一种方式还需要修改服务方代码，不及 CORBA 灵活和高效。对于异步调用方式，RMI 的客户端程序可以启动一个线程，由这个线程去调用远程方法并取得结果，主程序在启动线程后继续运行，并可以在随后的执行过程中查询远程调用是否结束及取得结果，这种用线程模拟异步调用的方法因为对每个远程调用都要启动一个线程，因此开销比较大。在编程方面，CORBA 异步调用需要使用复杂的 DII 界面，难以理解和掌握，Java 需要编写较多的代码，但由于它始终使用的都是同步的调用，容易理解和掌握，与 Java 语言结合得也更加紧密。

3.3 动态发现和动态调用

在 CORBA 中，客户程序运行时能够通过接口库（Interface Repository）动态地得到一个 CORBA 对象的接口定义，并且能够通过动态调用接口（DII）动态地调用接口中的操作，这是 CORBA 提供的除了使用 Stub 之外的另一种调用远程过程的方式，使程序能够使用在编写时还不知道的对象提供的服务，因此非常灵活。

一些资料[4]认为 RMI 没有动态发现和调用远程对象方法的能力，这是不正确的。Java 在其 1.1 版本中与 RMI 一起引入了反射（Reflection）API，通过这个 API 程序能够（在运行时动态地）取得一个对象的类（包括远程对象）、数据成员和方法的信息，并且可以通过这个 API 访问对象的数据成员和调用它的方法。在 RMI 中，远程对象在客户程序中是由它的一个 Stub 类的对象代表的，这个 Stub 对象与它代表的远程对象都实现了相同的远程界面。因此在 RMI 中，客户程序可以通过反射 API 动态地得到远程对象的方法并调用这些方法，因此通过 RMI Java 程序可以调用在编写时还不知道的远程对象的方法。RMI 的动态调用方式与 CORBA 相比更容易掌握、与 Java 语言结合得也更加紧密，不需要程序员掌握复杂的接口库和 DII。

3.4 通用服务

在 CORBA 的文档中定义了丰富的对象服务[5]，包括生命周期、事件、名字等。这些服务为应用软件的开发提供了莫大的方便，因为这都是一些大部分分布式应用系统必需的基本服务，如果系统没有提供这些服务，那么在开发分布式应用软件时只能由应用软件的设计者自行开发这些功能，不仅工作量大，容易造成重复开发，也会影响系统的通用性和互操作性。

RMI 提供的服务极为有限，目前只有一个基于 URL 的简单的名字服务，其他的服务正在开发之中，其完备和成熟程度都远不如 CORBA 中的服务。

3.5 对象的传递

在 CORBA 中，由于它支持的应用平台和语言的异构性，以及 CORBA 自己作为填平语言和平台不同而造成的沟壑的中间件，造成在 CORBA 体系中缺乏一种标准的对象存储格式和跨平台的代码格式。在 CORBA 中对象不能以传值的方式进行存储和传输，因此无法把对象的值作为远程调用的参数或返回值。在 CORBA 中通常所说的对象实际上都是对远程对象的引用，通过远程过程的参数和返回值传递的对象也是这些引用，这些远程对象本身是不能被传递的。CORBA 的生命周期服务虽然定义了几个基本的用于对象移动和复制的操作，但它需要对象的设计者自行编写大量的代码，并且移动的范围也极为有限。有些对象适合固定在某个位置上供远程调用，但也有很多对象只有传送到客户端才有意义。例如一个歌曲对象，只有传到客户机播放才有意义。因此支持对象以传值的方式移动是分布式系统一个很重要的能力，而 CORBA 不具备这个能力。

在这个问题上 Java 有着标准的跨平台的代码格式，有着突出的优势，JDK1.1 的对象序列化 API 可以把 Java 对象保存在永久介质上或通过网络传输而不需要程序员编写任何额外的代码。Java 的对象可以直接作为 RMI 远程调用的参数或返回值，Java 运行环境会自动保存和传递的状态，对于通过 RMI 调用传递的对象，如果其代码接收端没有，Java 运行环境还会自动从发送端把代码下载下来。由于 RMI 与 Java 语言的结合十分紧密，在把对象作为参数或返回值传递方面，调用远程对象的方法与调用本地方法一样方便。RMI 使分布在不同机器上的 Java 持续能够无缝地结合到一起。

3.6 开发的简易程度

CORBA 作为一个跨平台的系统，平台和语言中立是其重要的设计目标，这意味着它提供的功能和操作必须是各种语言和平台都支持的。这样做固然在一方面保证了对多个语言和平台的支持，然而付出的代价却是 Java 等功能较强的新型语言无法在 CORBA 界面上使用本语言许多强有力的特性，只能使用传统语言都支持的比较基本的功能。CORBA 中远程对象的界面是由与平台无关的 IDL 语言描述的，然后由针对不同语言的编译器把它编译成用某种语言写成的 Stub 和 Skeleton。这种方式虽然保证了对象界面的语言中立性，但也确实给开发带来了许多不便。每次修改远程调用界面时都要修改 IDL 定义，然后重新进行编译，十分繁琐。程序员在开发时不仅要熟悉他使用的语言环境，还要熟悉 IDL 语言、ORB、BOA、接口库等内容，而 IDL 有些成分，如 Array 与 Sequence 的区别、Union、struct 和 ORB 界面中的一些内容对 Java 程序员来说显得多余和过于繁琐。

RMI 则不同，Java 程序员使用 RMI 不需要学习 Java 以外的其他系统，开发的全过程都是在纯 Java 环境中进行的，开发自然、效率高。

3.7 对异构平台和异种语言集成的支持

CORBA 对异构平台和异种语言有着良好的支持，CORBA 中 ORB、BOA 的界面都是以与语言无关的形式来描述的，对象的界面是由标准的 IDL 语言描述的，目前几乎所有常见的平台和语言都有 CORBA 产品支持。用 CORBA 可以很容易地把位于不同平台上用不同语言开发的软件集成起来，通过简单的封装把各种遗留软件集成起来。CORBA 还详细定义了不同 ORB 产品互操作的协议。

RMI 是一种纯 Java 的解决方案，它对异构环境有着很强的支持，这是由 Java 良好的跨平台特性所支持的，但它对集成的支持却很弱，由于 RMI 的两端都必须是 Java 程序，如果我们要用它集成非 Java 的系统，就必须在非 Java 系统所在地机器上用 Java 对它进行封装，由这个封装程序把它的请求传给其他机器或把其他机器的请求传给这个程序，对于 Java 与其他语言程序接口的问题，在 Java 中定义有 JNI，用于与其他语言接口，但 JNI 接口是针对 C 语言的，难以用于其他语言，而且相当复杂，难以使用。Java 包装程序与其他语言程序接口的另一种方式是通过某种通讯机制，如管道、Socket 等，但这样作也很复杂，又回到了 RPC 以前的世界。因此，RMI 虽然可以通过一些措施实现与其他语言的互操作，但其功能和开发的简便程度都远不如 CORBA。

3.8 速度

虽然 CORBA 和 RMI 的功能有许多差别，但有一点可以肯定，大量基于 Java 的分布式应用程序既可以用 CORBA 开发也可以用 RMI 开发。如何选用合适的工具，除了功能上的考虑之外，运行速度也是一个重要的因素。为此，我们对两者的速度做了一个测试。

CORBA 和 RMI 在功能上有很多差异，我们比较的范围限于两者都支持的操作，即同步方式远程调用的速度和参数及返回值打包传送的速度，测试的数据类型限于基本数据类型和引用类型，复合数据类型传送的性能可以看作是基本数据类型性能的加权和。我们的测试环境是若干台用 10M 以太网连接的 SUN Sparc 20 工作站，Solaris 2.5.1，Java 开发和运行环境是 JDK1.1.5，CORBA 产品是 Visigenic 的 VisiBorker 3.2[6]和 IONA 的 OrbixWeb 3.0[7]。结果如表 1 所示。我们在测试中所使用的都具有不同类型的送入参数、没有返回值的同步方式调用，表中所列的耗费时间是连续进行 10000 次调用后的平均值。

表 1 RMI 与 CORBA 的远程调用耗费时间(ms)

	Java RMI	VisiBroker 3.2	OrbixWeb 3.0
无参数简单调用	3.03	2.53	4.37
有 10 个 8 位整数参数的调用	3.16	2.59	5.38
有 10 个 16 位整数参数的调用	3.25	2.67	5.49
有 10 个 32 位整数参数的调用	3.45	2.73	5.51
有 10 个 64 位整数参数的调用	3.76	2.87	5.65
有 10 个 16 位字符参数的调用	3.26	2.69	5.47
有 10 个布尔型参数的调用	3.18	2.63	5.39
有 10 个 32 位浮点数参数的调用	3.48	2.77	5.55
有 10 个 64 位浮点数参数的调用	3.76	2.91	5.75
含 40 个 16 位字符串参数的调用	4.00	3.23	5.15
有 10 个对象引用参数的调用	9.63	11.07	18.89

(下转第 49 页)

Web 页的修改模式。监测 Agent 监测用户感兴趣的 Web 页的修改。

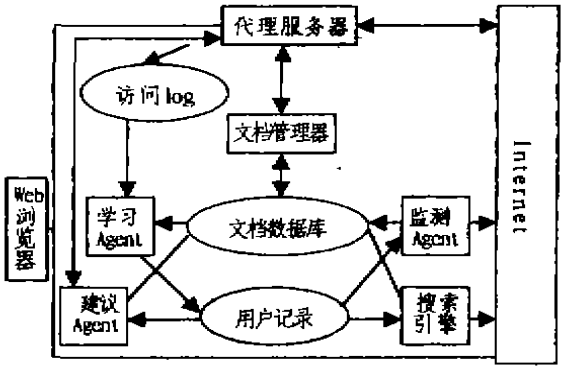

图 2

结束语 当前数据采掘和知识发现(DMKD)研究正方兴未艾，预计在 21 世纪将形成更大的高潮，由于 Internet 用户迅速增加，为了快速高效地找到网上的知识，研究在网络下的数据采掘，特别是在 Internet 上建立 DMKD 服务器与数据库服务器配合，实现数据采掘，加强对非结构化数据如文本数据，图形图像数据，多媒体数据的采掘，将是近期数据采掘的重要课题。

参 考 文 献

1 Chen M-S, et al. Data Mining: An Overview from a Database Perspective IEEE Transactions on Knowledge and Data Engineering, 1996, 8(6): 866～883

2 Agrawal R, Srikant R Fast Algorithm for Mining Association Rules. In: Proc. of the 20th VLDB Conf Santiago, Chile, 1994. 487～499

3 Charpiot B. Dynamic Clustering Using Associative Data Mining. ftp. research. microsoft. com/pub

4 Chen M-S, et al. Data Mining for Path Traversal Patterns in a Web Environment. In: Proc. of the 16th Intl. Conf. of Distributed Computing System. 1996. 385～392

5 伯晓晨. Internet 与 Intranet 中的人工智能技术 计算机世界报, 1998-5-18

6 Cheung D W, et al Discovering User Access Patterns on the World-wide Web; Knowledge Based Systems. Journal Elsevier Science, 1998, 10(7)

7 陈宁，周龙骧. 数据采掘技术. 计算机世界报, 1998-5-24, 1998-6-1

(上接第 36 页)

从表中可以看出：OrbixWeb 远程调用所需时间比 RMI 平均约多 40%，RMI 又比 Visibroker 约多 20%。RMI 居然比某些 CORBA 产品速度还要慢，这多少让人感到意外。从这个测试结果看不出这两种技术在速度上有什么必然的差异，具体的差异取决于不同系统的实现。另一方面，也说明了 RMI 还有很多潜力可以挖取。

结论 本文对开发基于 Java 的分布式应用程序之两种可用的分布对象技术——Java RMI 和 CORBA 的功能和速度进行了对比和分析。CORBA 具有良好的语言中立性，定义有功能较强的各种服务，适合于规模较大，需要与其他语言集成的系统，其缺点是开发过程较为繁琐，Java 的一些先进特性无法在远程调用的界面上使用。RMI 与 Java 语言结合得十分紧密，开发方便，支持对象传值，但它不太适合需要异种语言集成的场合。RMI 的性能高于一些 CORBA 实现，又低于另外一些 CORBA 实现。

参 考 文 献

1 JDK 1.1.6 Document, SUN Microsystems. Available at: http: // www. javasoft. com/products/jbk/1.1/docs/index. html

2 DCE1.1: Remote Procedure Call Specification: [Technical Report]. 1995, Available at: http: // www. rdg. opengroup. org/public/pubs/catalog/c706. htm

3 OMG, X/Open. The Common Object Request Broker: Architecture and Specification, Revision 2.2. Feb. 1998, Available at: http: // www. omg. org

4 Morgan B. Java 1.2 extends Java's distributed object capabilities Java World, April 1998. Available at: http: // www. javaworld. com/javaworld/jw-04-1998/jw-04-distributed. html

5 OMG. X/Open. CORBA Services: common Object Services Specification, Revised Edition, Updated at Nov 1997, Available at: http: // www. omg. org

6 Visgemic VisiBroker for Java. Available at: http: // www. visigenic. com

7 IONA Technologies Orbix Web. Available at: http: // www. IONA. com