Java 的远程数据库访问模型及其实现

陈　楠　（玉门石油管理局信息中心　735200）
韦　捷　（玉门石油管理局通信公司　735200）

摘要：网络计算是分布式应用的新技术。Java 的分布式对象模型能较好地实现网络的远程数据库访问。本文在 JDBC 的基础上探讨了 Java 的远程数据库访问模型，并以 Java 对 Oracle 7 数据库的远程访问为例论述了这一模型的实现。

关键词：Java　JDBC　分布式对象模型　分布式数据库　远程访问　Oracle 7

一、Java 的分布式计算技术

1. Java 的分布式对象模型

分布式系统要求分布于不同地址空间、不同宿主机上的计算之间能够相互通信。虽然基于语言的 Sockets 提供了一个基本的、灵活的通信机制，但 Sockets 要求客户和服务器在对交换的信息进行编码和译码时，须遵循应用级的协议，而这种协议设计起来很复杂且容易出错。

可以代替 Sokets 的机制是 RPC，它把通信接口（界面）抽象到过程调用一级。这样，程序员不用直接使用 Sockets 工作，就象在调用一个本机过程，实际是，该调用的参数被打包，并传送到远程目标。RPC 系统使用外部数据表示（象 XDR），对参数和返回值进行编码。但是，RPC 并不适合于需要在不同地址空间上的程序级对象之间进行通信的分布式对象系统。

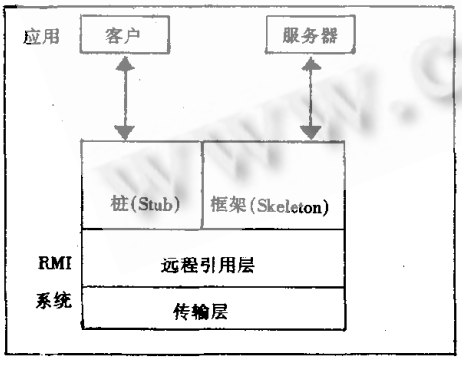

图 1　RMI 系统体系结构

为了与对象调用的语义匹配，分布式对象系统需要远程方法调用（Remote Method Invocation）。该方法用本机桩（Surrogate）对象管理远程对象的调用。RMI 用面向对象的思想继承和发展了 RPC，其体系结构如图 1 所示。系统由三层组成：

·桩/框架（Stub/Skeletons）层：客户端桩（代理）和服务器端框架。

·远程引用（Remote Reference）层：远程引用行为（象单个对象或复制对象的调用）。

·传输（Transport）层：现成连接的建立和管理，以及远程对象的跟踪。

从客户到远程服务对象的远程方法调用，沿 RMI 系统的高层从上到下一直到客户端传输层，然后再从下到上，从服务端传输层到服务器。

从客户到远程服务对象的方法进行调用的客户，实际上是使用远程对象的桩或代理作为访问远程对象的渠道（Channel）。客户所拥有的远程对象的引用是对一个本机桩的引用。这个桩是远程对象的远程接口的实现，它借用远程引用层把调用请求传送给服务对象。

远程引用层负责分析调用的语义，服务对象的引用语义也由远程引用层处理。

为了调用远程对象，传输层把远程调用传送到远程引用层。远程引用层处理任意服务端的行为，这些行为需要在处理对服务端框架的请求之前发生。远程对象的框架把调用传给远程对象实现，后者执行真正的方法调用。调用的返回值以如下过程返回：从服务器端框架层传到远程引用层，再传到传输层，然后返回到客户端的传输层和远程引用层，最后返回到客户端的桩层。Java RMI 的实现涉及对象串行化和动态 Stub 装载两项技术。

2. Java 的分布式计算技术

（1）Java 的分布式编程特性。JavaSoft 为使用 Java

开发分布式应用的用户提供了两种分布式服务：遵循 CORBA 的 IDL 设施和本机上的 Java RMI。

Java IDL 是一种 IDL 编译器，它将 Java 对象映射到 CORBA 对象代理(Object Broker)上，以达到 Java 对象与其他语言中的而且是运行在其他机器上的对象间的互操作。开发者必须在 Java 内遵循 CORBA 对象的命名协议，反之亦然。这使得用户可以方便地实现 Java 系统与遗留代码(非 Java 代码)的集成，并改善其性能。

RMI 只能在不同系统上的各种 Java 程序之间发生交互。RMI 的使用不是无缝的：你必须显示地声明远程类，并执行一些其他的实现所必须的过程，其缺省方式是 RMI 直接与 Java 的 Socket 接口。这种方式可以在任何的 Java 系统上实现用户的系统，并可与用户现有的安全、分布式设施进行通信。同时 RMI 可以充分利用 Java 的安全功能，而且通过 Java 虚拟机可以很容易地把 RMI 移植到任何系统上。

(2) Java IDL 网络计算技术。Java IDL 是 Sun 的 100％纯 Java 对象请求代理(ORB)系统，它为在 Internet 上传输企业客户/服务器应用提供了必须的软件基础。使用 Java IDL 的应用程序能与非 Java 程序和其它厂商提供的程序进行无缝集成。Java IDL 系统是基于最新 CORBA 和 IIOP 工业标准的，它提供了与 CORBA 标准的连接性和互操作性。Java IDL 包括：

·纯 Java 对象请求代理结构。

·Internet 上 ORB 之间协议(IIOP)的完整实现。

·CORBA 2.0 标准 IDL 到 Java 的映射。

·CORBA 2.0 标准的命名服务。

①IDL CORBA 的核心思想是：凡是在网络上任意地方可供使用对象(服务、构件等)，都有清晰的、严格的接口定义，这些接口可以被认为是客户与服务器之间的一种约定，一个用 IDL 说明的约定。

IDL 只描述接口，而不描述实现。也就是说，它不是一种程序设计语言，它只是一种以独立于程序语言的方式来描述对象的接口，起语法类似于 Java 和 C++。IDL 可以被映射到每一种程序设计语言，为这种语言提供对对象接口的自然访问。在使用 Java IDL 时，这些 IDL 定义可以使用 idltojava 桩产生工具编译，以产生 Java 接口定义和 Java 客户桩、服务器桩。Java IDL 允许用户透明地调用驻留在远程服务器上的 CORBA 对象。同时，它也允许 Java 服务器定义可被远程 CORBA 客户透明地调用的对象。

②IIOP 如果使用同一网络协议进行通信，那么 CORBA 实现之间可以透明地互操作。在 CORBA 2.0 中，CORBA 实现遵循了 IIOP 协议，从而达到了它们之间的完全的互操作。Java IDL 使用的是 IIOP 的 1.0 版本。

值得注意的是，Java IDL 是基于可移植的 Java ORB 内核的，ORB 内核的构造使得插入新的 ORB 协议很容易。idltojava 桩产生器产生独立于 ORB 的桩，它调用 ORB 的特定协议模块，以进行各种数据的编组(Marshling)或其他的 ORB 操作。

Java IDL 包括以下几个构件成分：

·类属(Generic) ORB 内核：这是 ORB 运行系统，它允许 Java IDL 应用程序作为独立的 Java 应用程序，或者作为支持 Java 浏览器中的小应用程序运行。

·idltojava：一个开发工具，它自动地为具体的远程接口产生桩代码。

·nameserv：CORBA 对象服务(COS)中名字服务的一个实现。

(3)用分布式对象技术开发 Java 应用程序的思路。Java 语言与分布式对象技术的结合，可以建立新的客户/服务器开发环境。用 Java 开发基于 CORBA 的 Java Applet 为在 Intranet 中访问和管理分布式数据库提供了基础。利用分布式对象技术开发 Java Applet 的方法如下：

①用 IDL 源码描述出基于 CORBA 的分布式应用的接口规范；

②把 IDL 源码转换成 Java 源码，再与用 Java 语言编写的应用程序结合，产生基于 CORBA 的 Java 拼接应用程序；

③利用 Java 编译程序，把 Java 源代码变换成中间码(字节码)，再转换成 Java Applet。

二、用 Java applet 访问 Oracle 7 数据库

1. JDBC 的两种应用方案

JDBC 有两种常用的应用方式：Java applet 和 Java application。Applet 是作为 Web HTML 文档的一部分在网络上传输的小应用程序，用于数据库访问的 Applet 正是使用 JDBC 的三层数据访问模型来与数据库进行交互的。

Java application 是一种独立的应用程序，与用其他语言编写的应用程序在功能上和使用范围上没有什么区别。Application 最常用的方式是应用在企业 Intranet 中，这种独立的应用通常可以位于客户机上，具有访问数据库灵活、扩展性好以及适合分布数据访问等优点。

基于对网络安全方面的考虑，Applet 的操作权限有

许多的限制，如不能访问本地的文件系统，或者通过 Applet 去访问网络上其他节点的文件系统等等。Applet 的执行需要装入到 AppletViewer 或能运行 Applet 的浏览器上，通常运行在远程机上；Application 则需要用 Java 解释器来解释执行，通常运行在本地机上。

从程序形式上看，Applet 和 Application 有所不同，Applet 的典型形式为：

```
public classAppletClassName
extent Applet implementsRunnable {
public void init() {... ...}
//其他的方法和参数定义
public void start() {... ...}
public void run() {... ...}
public void stop() {... ...}
public void exit() {... ...}
}
```

而 Application 的典型程序结构为（基于 AWT 编程）：

```
public class ApplicationClassName extends Frame
{
ApplicationClassName() {... ...}
//这是一个对象的构造函数
public static void main(String args[])
{
FrameApplicationClassObject = new ApplicationClassName();
resize(200,400)
show();
}
//其他的方法和参数定义
}
```

从程序的具体内容来看，无论是 Applet 还是 Application，都是一个公共类。而且，为了能使 AppletViewer 装载这个类，它必须是类 Applet 的子类（由类 Applet 继承而来）；为了使 java 能解释执行这个 Application，它就必须含有如下说明的方法：

```
public static void main (String args[]) {... ...}
```

因此，Application 与 Applet 的最大区别是：Application 有一个 main(String args[])的方法，它用于声明该 Applicayion 对象，并在该对象生成之后，调用方法 show()来显示 Application 的结果；而 Applet 则由 Applet 继承而来，它用 init()方法负责对象的初始化，在初始化后按 Start()→run()→stop()→exit()的顺序执行 Applet 程序。此外，Java Application 定义数据库最常用的方式是由用户或应用程序声明一个数据库名，即由系统来寻找指定的机器、DBMS、JDBC 驱动程序和数据库。而 Applet 是在程序中直接指定这些内容。

2.用 Java applet 访问 Oracle 7 数据库

利用 Oracle 的 JDBC 支持也可以建立 Java Application（应用程序）和 Java Applet（小应用程序）。前者通常位于客户端，通过 WRB（Web Request Broker，Web 请求代理）与 Oracle 建立联系，访问数据库。后者则不需要任何客户端 Oracle 部件，只需要支持 Java 的浏览器在客户端解释执行，对数据库的访问可以在任何支持浏览器的平台上执行。

假定要访问的数据源名为 mydb，JDBC Applet 服务器运行于 TCP/IP 端口号为 2222 的端口上，JDBC 服务器所在的服务器为 cherry.ncic.ac.cn，操作系统为 AIX 4.1（UNIX 技术的操作系统）。访问 Oracle 数据库的具体方法如下：

①引入适当的 Java 类：

若是 Application，需引入

java.net.URL,jva.sql.*,oracle.sql.*;

若是 Applet，需引入 tempjava.sql.*.oracle.netsql.*。

②装入适当的 JDBC 驱动程序：

若是 Application，需装入 oracle.sql.OracleDriver；相应的语句为：Class.forName("oracle.sql.OracleDriver");

若是 Applet，需装入 oracle.netsql.OracleDriver；相应的语句为：

Class.forName("oracle.netsql.OracleDriver");

③与数据库建立连接：通过 URL 与数据库建立连接，在 URL 中需提供数据库位置、子协议名（Subprotocol）。对于 Oracle，子协议名为 oracle。

若是 Application，其 URL 为：jdbc:oracle:mydb。所建立的连接对象格式如下：

Connection con = DriverManager.getConnection("jdbc:oracle:mydb");

若是 Applet，其 URL

为：jdbc:oracle://cherry,ncic.ac.cn:2222/mydb。所建立的连接对象格式如下：

Connection con = DriverManager.getConnection

(jdbc:oracle://chery.ncic.ac.cn:2222/mydb,userid,password);

需要特别注意的是，对于 Applet，访问端口号必须大于 1024。

④向数据库传递 SQL 语句，执行查询并返回结果，例如，可以创建一个 SQL 语句对象 stmt：

```
Statement stmt = con.createStatement
```

然后执行查询，并返回结果：

```
ResultSet rs = stmt.executeQuery("SQL 语句");
```

⑤关闭连接：将所建立的连接对象关闭：

```
con.close();
```

从上面的过程可以看出，Java applet 与 Java Applicatin 的编程基本类似，主要的不同点表现在 JDBC URL 上。由于 Application 与数据库的通信是交由 WRB 来完成的，所以在其 JDBC URL 中，不再包含主机名和端口号这一部分，而这些信息对于 Applet 则是必需的。此外，JDBC 驱动程序的实现也不相同，对于 Applet，JDBC 驱动程序为 oracle.netsql.OracleDriver，而对于 Application，其 JDBC 驱动程序为

oracle.sql.OracleDriver。

参考文献

[1] Sun Microsystems. The JDBC? Database Access API, (http://splash.javasoft.com/jdbc/)

[2] JavaSoft. JDBC?: A Java SQL API. Jan. 1997, (ftp://splash.javasoft.com./pub/jdbc.ps)

[3] Doug Kramer, 《Java API Overview》, 1996

[4] Sun Microsystems Inc., 《JDBC Guide》, Sun Microsystems Inc., 1996

[5] (美)Micheal Girdley, Kathryn A. Jones 等，曹康等译，《怎样用 Java 开发 Web 应用》，北京：人民邮电出版社，西蒙与舒斯特国际 出版公司，1997.10

（来稿时间：1998 年 4 月）