文章编号：1006－4729(2003)01－0013－05

基于 Java RMI 和数据库连接池的电厂信息综合浏览系统

丁　辉，林中达

（东南大学 动力工程系，江苏 南京　210096）

摘　要：电站实时信息的综合浏览对调整电能的生产具有重要作用. 本系统采用 Java 的 RMI 分布式对象及数据库连接池技术来实现厂内信息的综合浏览. Java RMI 应用是 CORBA 技术的纯 Java 应用之一，整个系统基于 B/S 结构，客户端 WEB 浏览器中的 Applet 向 RMI 服务器发出数据请求，RMI 服务器通过数据库连接池访问数据库，然后将结果回应给客户端，客户端由 Applet 进行显示. 整个系统是分布式的，各部分耦合性小，服务器和客户端开发和扩充可以相对独立进行. 实践证明，该系统运行十分稳定和可靠.

关键词：Java；RMI；数据库连接池
中图分类号：TP311.138；G202　　文献标识码：A

引　言

随着信息技术的发展，火电站信息系统的实时性与综合性都有进一步的提高. 为了能够随时随地浏览厂内信息，就需要一套独立的系统，它从厂内监控信息系统(SIS)及管理信息系统(MIS)的数据库中通过数据库连接池获得所需数据，并进行处理后显示. 由于它独立于厂内系统以外，并且通过互联网向外发布，因此，可以不受厂内条件的限制，方便地进行本地或远程网上浏览.

1　系统设计思想

该系统采用 Java 语言开发. Java 语言的安全性、稳定性及丰富方便的网络功能，使其可与网络紧密无缝结合，更重要的是 Java 语言的二进制字节代码运行于(JVM)Java 虚拟机上，而不是直接运行于本地操作平台上，故其与操作平台无关，这使 Java 语言开发的系统具有良好的跨平台的可移植性、分布性和扩展性. 这样，无论是在 unix，linux 操作系统，还是在 windows NT，win2K，windows9x 操作系统下运行，其字节代码都不需要重新编译. 随着很多大型系统都升级到 unix 操作平台上，采用 Java 的优势很快就可体现出来. 客户端采用浏览器访问的方式，Java 客户端的 Applet 小应用程序可以直接内嵌于 Web 浏览器中，用户不需安装专门客户端软件. 所有的生产管理信息统一通过 Web 服务器向外发布，用户只需通过权限认证即可通过浏览器 IE 或 Netscape 进行浏览.

系统在结构上采用 Java 语言的 RMI 分布式对象技术. Java RMI(Remote method Invoke)采用远程方法调用以获得远程数据. 由于所有的数据库访问及复杂算法均在 Java RMI 服务器上进行，客户端只需进行显示，这就减轻了客户机的工作压力，从而实现了“瘦客户”连接. 而且因使用分布式对象，降低了系统的复杂性，使整个系统模块分明，更易维护，这对系统的扩充升级极为有利. 若增加功能，只需修改或添加远程对象即可，而不会影响客户端.

系统客户端采用 Java 的 Applet. Applet 可以内嵌在网页中，而不用专门开发客户端软件. 采用 Applet 的双缓存技术可以使画面动态刷新，无闪烁，图形界面的形式和内容可以进行复杂的组态配置. 界面友好，柔性强.

收稿日期：2002－09－11

整个系统采用 B/S 的 3 层模式, 并运用分布对象技术进行设计, 其层次结构如图 1 所示.

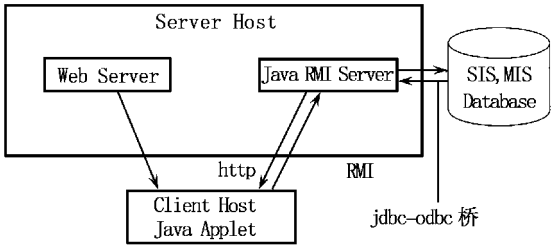

图 1 系统结构

客户端访问 Web 服务器时, Web 服务器将访问转交给信息综合浏览系统的 RMI 服务器, RMI 服务器采用数据库连接池通过 jdbc－odbc 桥从 SIS 与 MIS 系统数据库服务器中取出客户端所要显示的原始数据, 经过 Java RMI 服务器进行统计、计算, 以获得需要显示的数据, 通过远程方法的返回值, 返回客户端浏览中的 Java Applet 中, 再由 Applet 进行画面显示点的定位及显示.

2 RMI 远程方法调用

2.1 RMI 分布式对象原理

RMI(Remote method Invoke)分布式对象技术是指对位于远程机器上的对象方法进行调用, 而不必将这些对象移回调用方法所在的客户机上. 它是 CORBA 技术的一种纯 Java 应用. 当客户机代码想对远程对象调用远程方法时, 它就像调用了普通的 Java 方法. 该方法被封装在一个称为存根(stub)的代理对象(surrogate object)中. 存根位于客户机上. 存根首先取得远程方法中使用的参数, 并将其打包为字节块, 每个参数的打包使用了独立于设备的编码机制, 接着使用对象序列化机制来实现参数序列化, 并向服务器发送该信息. 在服务端, 有骨架(skeleton)对象, 它将收到的信息反序列化, 以获得客户机传来的参数, 并将该参数传递给执行远程方法的实际对象, 以调用期望的方法. 然后, 骨架对象将返回值序列化后发送给客户机的存根对象, 存根在反序列化从服务器上返回的值, 就得到了远程方法调用的返回值[1]. 见图 2.

该过程很复杂, 但它是完全自动进行的. 对上层程序来说, 它是透明的, 就像调用本地方法一样. 上层应用程序完全感受不到服务对象是在远程机上. 这也是 RMI 通讯比 Socket 通讯方便的地方. Socket 通讯客户机与服务器传递的只是字节流, 其每部分字节流所代表的含义必须通过双方自定义的协议才能互相知道. 发送方与接收方都必须经过接受时解码与发送时编码的过程, 这对发送方与接受方来说都不透明, 它们都必须自己维护该过程. 此外, 不透明的另一不足之处是 Socket 的客户端与服务器端都必须维护连接状态, 任何异常与超时, 网络都必须自己给出异常处理代码, 而在 RMI 通讯中, 上述一切都由 RMI 协议维护, 用户不必操心, 只需直接调用所需方法即可.

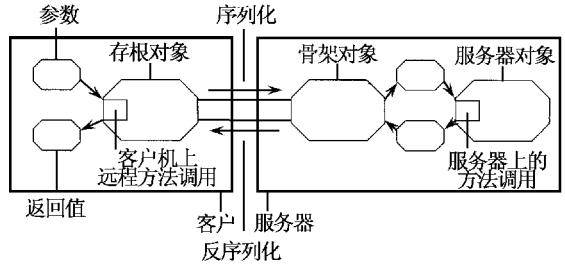

图 2 RMI 原理

远程对象中有远程方法, 即可被客户端直接调用方法, 也可以有本地方法. 本地方法不能被客户端远程直接调用, 但它可以被远程方法调用, 即客户端可通过远程方法调用远程对象中的本地方法.

客户程序需要对服务器对象进行控制处理, 但实际上却没有该对象的拷贝, 对象本身位于远程服务器上, 客户只需要知道能对该对象执行哪些操作, 其能力可用接口函数表示. 将服务器中能执行的操作放入一个接口中, 然后, 由客户与服务器共享该接口即可. 这样, 客户机就可以知道服务器中有哪些远程方法可以调用[2].

远程对象的返回值既可以是数值也可以是对象, 当返回值为对象时, 若该对象为远程对象, 则返回值为该远程对象的存根引用, 源对象内存空间在远处机上; 若返回值为非远程对象, 则返回为该对象的拷贝. 客户机及远程机上各有一份与该种类型内容相同的对象.

2.2 功能实现

RMI 功能实现过程主要有以下几个步骤.

1 建立 DBServerInterface 接口 该接口扩展

自远程类(Remote)接口，并声明 GetResult()为远程方法. 该方法抛出远程异常(RemoteException). 因为远程方法实现于远程服务器上，在与客户端通讯时有很多网络异常需要处理，例如网络忙、连接超时等，客户代码必须知道这些情况[3,4]. 该接口只声明(GetResult()方法，而并没有实现该方法.

2　远程服务器类 DBServer　它扩展自接口 DBServerInterface，并重载接口中的远程方法 GetResult(). 该服务器类共有一个远程方法 GetResult()、两个本地方法 QueryDB()和 HandleData(). 出于安全性及降低复杂性考虑，客户机无需连接数据库，也无需获得数据库访问权限. 因此，采用远程对象的本地方法 QueryDB()负责访问 SqlServer 数据库，执行数据库操作，获得显示的原始数据. 该方法由远程方法调用. 本地方法 Handle Data()负责对数据库访问的数据进行计算及将其统计后转换成最终要显示的数据，而远程方法 GetResult()则负责将以上两个本地方法获得的结果返回给客户端.

3　创建 DBServerInterface 的骨架和存根　骨架和存根是服务器级的客户级的类，它们都由 RMI 机使用，以获得序列化参数和序列化结果. 使用 Java 的 rmic 工具产生存根类 DBServerInterface_Stub. class 与骨架类 DBServerInterface_Skel. class. 它们负责客户机与服务器之间的通讯.

4　使用引导程序(bootstrap registry service)登记远程服务对象　其目的是使客户机可以检索到这些对象的存根并使用它们的服务. 将远程服务器对象的引用传给引导登记服务，并给该对象指定一个名字，就登记了该远程服务器对象.

5　启动服务程序　客户端程序在获得远程对象的存根后即可直接调用服务器程序上的远程方法.

6　客户端 Applet　其在获得远程方法返回显示数据后，进行点的定位与显示.

2.3　运行过程

1　启动服务器程序，服务器登记服务对象.

```
DBServer dbserver=new DBServer();
Naming.bind("DBServer", dbserver);
```

2　客户机访问 wed Server 上的 HTML 页面，该页面中内嵌 Java Applet 程序. 这样，Applet 二进制字节代码就被下载到客户机上，由 Web 浏览器内嵌的 Java 虚拟机(JVM)运行，在 Applet 程序的初始化代码中，通过 URL 格式指定服务器名和对象名，获得对象的存根.

```
DBServerInterface db=(DBServerInterface);
Naming. lookup ("RMI://www. server. com/DBServer");
……
Vector vtDBResult=db.GetResult(szCondition);
```

接着，客户端执行 db.GetResult()远程方法. 参数 szCondition 为查询条件. 该方法在服务器上运行，它首先调用服务器的本地方法 QueryDB()访问数据库，取得显示所需的原始数据. 然后，调用另一个本地方法 HandleData()来处理原始数据，将结果写入 Vector 对象中，并将其作为返回值. 该返回值通过 GetResult()的返回值由 RMI 机制返回到客户端. 由于该返回值为 Vector 对象，不是远程对象，因此，返回的不是它的存根，而是拷贝. 以上过程尽管复杂，但在客户端却完全感受不到这一切，好像 GetResult 方法就在本地客户机上一样. 最后，客户端程序将结果在客户机上以适当的方式显示出来.

3　数据库连接池技术

在数据库访问中，若使用一个数据库连接，则不能满足多个客户同时访问的要求；而使用多个数据库连接时，由于每个客户机都产生一个数据库连接，访问结束后，又要将数据库连接关闭. 这样频繁连接然后断开数据库，会给数据库服务器带来沉重的负担，使其性能下降，数据库响应速度变慢. 这对于电站这种实时性要求很强的数据库系统是极其不利的. 鉴于以上分析，可考虑同时打开一定数目的数据库连接，并一直保持打开状态直到系统关闭. 这样，使用时可获得一个连接句柄，使用结束就释放该连接句柄，从而避免数据库的频繁打开、关闭，并可以保证多个客户的并发连接请求. 本系统的数据库连接采用的就是这种数据库连接池技术. 见图 3. 其具体实现过程如下：

首先，在系统初始化时，打开 n 个数据库连接(Connection)，将这些连接的引用句柄放入一个队列中，该队列称为连接池，并将这些连接的状态全置为"闲"(free).

其次，当系统中 DBServer 中的本地方法

QueryDB( )作为一个客户访问数据库时, 连接池就会检查连接标志位, 看是否有连接空闲. 若有连接空闲, 将该连接的引用句柄传给客户程序使用, 并将其状态位置为"忙"(busy); 若没有连接空闲, 则新打开一个数据库连接, 将其引用句柄加入连接池, 传给客户端使用, 并将其标志位置为"忙"(busy). 此时, 连接池的大小变为 $n+1$, 直到连接池的连接上限为止. 若某个客户端使用完数据库连接, 并不需要将数据库关闭, 此时, 只需将其引用句柄放回连接池, 并将其标志位设为"闲"(free)即可.

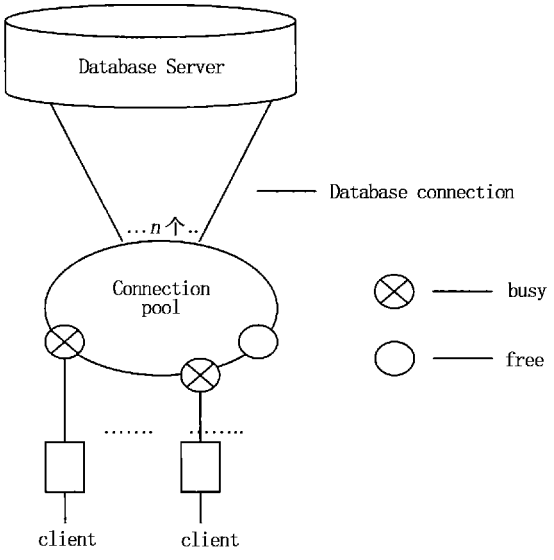

图 3 数据库连接池

最后, 当系统退出时, 遍历连接池中连接, 将其中标志位为"闲"(free)的连接全部关闭. 若池中还有标志位为"忙"的连接, 则延迟 300ms 后再将其关闭, 以确保该次数据库操作的完成.

使用数据库连接池, 最大限度地利用了已打开的数据库连接, 从而使数据库系统性能显著增强.

4 结束语

上述方案所设计的综合查询系统已在实际工程中获得应用. 运行表明, 该系统稳定可靠, 已完全达到各项要求. 该系统在开发过程中遇到了一些问题, 取得了一些经验. 系统开发初期, 由于没有采用数据库连接池, 结果在多个用户同时浏览系统时, 系统的响应极慢. 针对上述问题, 采用了数据库连接池技术, 使问题得到圆满解决.

本系统采用 Java RMI 的 B/S 3 层结构, 具有以下明显的特点:

1 若采用两层结构, 使用 Java Applet 直接访问数据库, 由于 Java Applet 的安全特性, Applet 小应用程序只能访问源主机上的资源, 即 Applet 所存在的那台服务器上的资源, 这势必要将数据库服务器也移植到 Web 服务器上, 就会对数据库服务器的性能及安全性造成极大隐患. 同时, 在客户机上既要访问数据库又要处理数据, 势必对客户端的性能提出更高要求, 这就需要配置性能较高的客户机, 从而造成资源的不合理配置.

2 若采用 Java Socket 链接槽连接, 则传送的全是字节流, 因此, 必须考虑适当的数据编码方法和传送的适当协议, 客户与服务器端都要进行相应的编码与解码. Java RMI 将网络通讯细节封装起来, 对上层应用程序是透明的, 并擅长传输任何类型的对象, 而不必局限于生硬的数据结构. 这样, 使系统更加模块化, 扩充和维护变得十分容易.

该系统所采用的 Java RMI 应用是 CORBA 技术一种纯 Java 应用, 属于分布式对象技术的应用之一, 具有极强的分布性与可扩展性. 如系统需要增加功能, 只需扩充远程接口的功能或增加远程对象即可, 对客户端与服务器端的升级改造也可分别进行. 但该系统也有一些局限性, 就是它在进行扩充时不支持其他语言的远程对象, 而只支持 Java 语言开发的对象.

参考文献:

[1] Cornell G, Horstmann C S 编著. Java 核心[M]. 杨秀军, 丁兴农, 何 鄴等译. 北京: 科学出版社, 1997.

[2] Bruce E. Java 编程思想[M]. 京京工作室译. 北京: 机械出版社, 1999.

[3] Elliatle R H. Java 揭密[M]. 陈遗风译. 广东: 世界图书出版社, 1998.

[4] Patrick C, Rosanna L. Java 类库手册[M]. 玄伟剑, 龚志波, 陈永智等译. 北京: 北京大学出版社, 1997.

[5] 闪四清. SQL Server 系统[M]. 北京: 清华大学出版社, 2000.

The Syntheses Browse of the Power Plant Information Based on Java RMI and Database Connection Pool

DING Hui　LIN Zhong-da
（*Power energy Department*, *Southeast University*, *Nanjing* 210096, *China*）

Abstract: The syntheses browse of the power plant real-time information is very important to modulation of the production of the power energy. In this system, the technology of the distributed objects based on Java RMI and database connection pool is used to realize the syntheses browse of the real-time information. The Java RMI application is one of the pure Java applications of the CORBA technology. The whole system is based on the B/S structure, the Applet in Web browser of the client submitting data request to RMI server and RMI server looking through the DBMS by the database connection pool, then results are returned to the client and displayed by the Applet program. The whole system is distributive, so the chances of coupling of every parts of it are very little. It is proven by practice that the development and expansion of serve and client may be carried out relatively independently. The operation of the system is steady and reliable.

Key words: Java; RMI; database connection pool