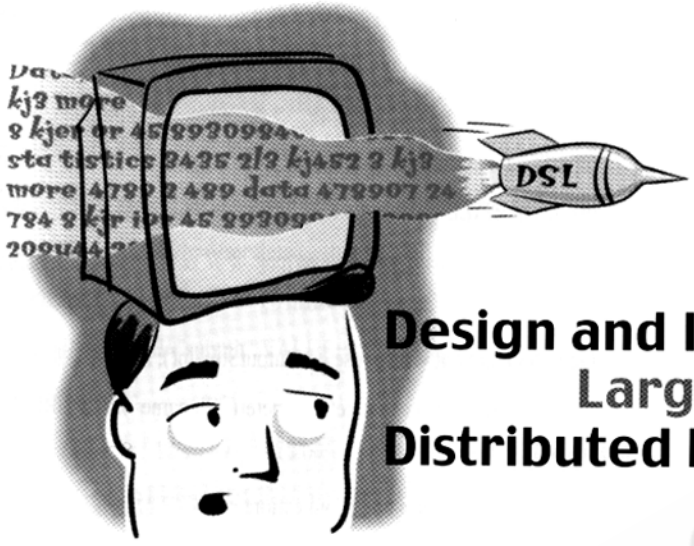
张志远 (新华社技术局 100803)



分布式环境下大对象数据发布的设计及实现

Design and Implementation of Large Object Data Release in Distributed Environments

摘要：大对象数据的发布方式直接决定了系统能否构建成分布式体系结构。通过分析HTTP协议的请求、响应状态及WEB服务器的内容协商机制，本文提出三种实现大对象数据WEB环境发布的方法，分别针对不同数据类型。首先将所有数据存入数据库中，对于图像数据可以使用HTML标签方式发布，对于格式化文档使用HTML表单方式发布，对于静态HTML文件使用Apache重定向模块实现发布，这三种发布方式服务器端均使用Java Servlet程序访问数据库，以流模式实现大对象数据的发布。

关键词：大对象数据　WEB　HTTP　数据库

大对象数据的发布在WEB环境下非常简单，只需将这类数据的链接指针直接写入HTML网页，WEB服务器就可以获取到所需的文件。但问题也随之产生，首先，WEB服务器对这类文件的访问总是通过本地文件系统获取，对于存储在其他主机中的大对象数据就无法访问了；其次，这种工作方式对于存储在数据库中的大对象数据也无能为力。因此，当前的大对象数据发布方式是以牺牲灵活性换取易用性为代价的，必须找出一种不破坏目前的WEB体系结构，具备较强易用性，能够实现分布式环境发布大对象数据的方法。

1 常用大对象数据发布方式

大对象数据在WEB环境下的发布方式，目前经常使用的主要有三种：

(1) 直接将大对象数据的链接指针写入HTML网页，WEB服务器从本地文件系统中获取目标数据，发布到客户端。简单易行，但不具备分布式处理能力，数据和WEB服务器必须物理同机。

(2) 基本工作方式同 (1)，使用NFS(网络文件系统)，使WEB服务器具备透明访问远程主机文件系统的能力，具备了分布特征，但系统效率及安全性都受到质疑。

(3) 通过数据库直接发布大对象数据。这种发布方式有一个约束，即当前要发布的大对象数据类型必须在服务器端已预先设定好，所以一个网页只能发布一个大对象，且不能实现图文混排，具有较强的约束性。

表1比较了这三种实现方式的优缺点

总之，目前经常使用的3种大对象数据的发布方式都不能很好的实现分布式环境下，高效、安全、简易的发布大对象数据。

本文下述的方法均在Apache 1.3.12及Oracle 8.1.6数据库验证通过，其他环境可作适当修改。

2 通过HTML标签（TAG）方式发布大对象数据

2.1 理论分析

HTML标签中针对大对象数据的专用标签目前只有图像格式，一个典型的图像标签如下所示：

<IMG SRC=" /image/1.jpg"></IMG>

使用该标签，浏览器解释该语句时，会产生成一个HTTP连接，向WEB服务器发出一个GET请求，WEB服务器从/image目录返回1.jpg文件给用户，这是正常的工作模式。当前1.jpg文件并未存储在本地文件系统中，而是存放在数据库中，必须让WEB服务器触发一个Servlet程序从数据库中读取图像文件返回给用户，才能完成图像文件的发布。前面的分析已知img标签会产生一个GET请求，而Servlet程序也可以通过GET方式触发，基于这样的考虑，我们可以将SRC属性定向到应用程序，而不直接定位图像文件的URL，这样IMG标签将改写如下：

<IMG SRC=" /servlets/GetImage?filename=1.jpg"></IMG>

这种IMG标签产生的GET请求不再是简单的文件获取，而是可以激活服务器端的Servlet程序（这里假定为GetImage），并将文件名以URL参数方式传递给程序，此Servlet程序通过JDBC方式从数据库中获取图像文件发布给用户。

2.2 程序实现

GetImage是一个标准Servlet程序，下面简单介绍其工作机制：

```
import java.sql.*;
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import oracle.sql.*;
import oracle.jdbc.driver.*;
public class GetImage extends HttpServlet{
    public void init(ServletConfig config) throws ServletException
    {
        Class.forName(DB_DRIVER); // 初始化JDBC接口
        super.init(config);
    }
    private Connection openConnection() throws Exception
    {
        return DriverManager.getConnection(DB_NAME, DB_USER,
    DB_PASSWORD);
        // 打开数据库连接
    }
    private void closeConnection(Connection conn) throws Exception
    {
        conn.close();   // 关闭数据库连接
    }
    /**
    * 处理HTTP Get请求
    */
    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
    {
        InputStream in=null; // 设置输入流
        oracle.sql.BFILE bfile=null; // 设置Oracle Bfile变量
        response.setContentType("image/jpeg"); // 设置输出文档类型为图
    片格式
        OutpubStream out = response.getOutputStream(); // 初始化输出流
        String fileName = request.getParameter("filename");// 从URL信息
    中获取文件名
        Connection myconn = openConnection(); // 打开数据库连结
        Statement stmt= myconn.createStatement(); // 实例化连结对象
        String query = "SELECT BFILENAME('" + DB_DIRECTORY_ OB-
    JECT + "', '" + fileName + "') from dual"; // 构造查询串
        ResultSet rset=stmt.executeQuery(query.toString()); // 执行查询
        if (rset.next ()) {
        bfile = ((OracleResultSet)rset).getBFILE (1); // 获取图像文件指针
        bfile.openFile(); // 从数据中打开图像文件
        in = bfile.getBinaryStream(); // 以二进制方式读图像文件
        int length;
        byte[] buf = new byte[4096]; // 以4K字节为一组读取数据
        while ((length = in.read(buf)) != -1) { // 判断是否读到文件尾
        out.write(buf); // 以流模式输出
        buf = new byte[4096]; // 重新初始化字节变量
        }
        in.close();
        bfile.closeFile();
        out.close();
    }
}
```

上面演示的程序片断适用于发布图片数据，对于非图片类型的格式化文档数据就不能使用这种方法了。

表1 常用大对象数据发布方式比较

名称	工作方式	优点	缺点
方式1	WEB服务器直接从本地文件系统获取大对象数据	简单、易行，可以使用全部WEB特征	数据与WEB服务器必须物理同机，扩展性差
方式2	除方式1的特征外，使用NFS访问其他主机的文件	简单、易行，具备分布式能力	安全性及文件访问效率都比较差
方式3	从数据库中获取大对象数据	数据可以集中存储，具备分布式能力	实现比较复杂，发布格式有约束

3 通过HTML表单(FORM)方式发布大对象数据

3.1 理论分析

大对象数据的发布通常使用HTTP的POST方式激活Servlet程序执行发布功能，但这种方式有很大的局限性，通常只适用于发布纯文本或图像数据。下面是一个典型的FORM(表单)方式激活Servlet程序示例：

```
<FORM ACTION="/servlets/GetImage" METHOD="POST" >
<INPUT TYPE="hidden" NAME="fileName" VALUE="1.jpg">
<INPUT TYPE="submit" VALUE=" 显示图片 ">
</FORM>
```

用户在点击"显示图片"按键后，浏览器会向WEB服务器发送一个POST请求，Web服务器接收到该请求后，会激活表单中指定的GetImage程序，完成图片的发布，但是对于非图片数据，比如Word文档，视频流文件这种方法并不适用，原因在于这些格式化文档数据浏览器本身无法解释，必须借助客户端其他程序（比如Microsoft WORD）来完成发布，但浏览器如何以及何时激活客户端程序是必须解决的问题。上一章已经详述了WEB服务器的内容协商机制，即每个HTTP连结都必须设定与内容相一致的头信息，这包括ACCEPT信息及MIME类型，浏览器会自动完成这些头信息的设置(引自参考文献[10])，服务器端的应答程序必须根据客户机的请求设定正确的头信息，即必须与客户机请求相一致。但如何让浏览器设定与非格式化文档相一致的MIME信息呢？必须借助于文件名完成，系统通常是通过判断文件扩展名来决定文件类型，浏览器也不例外，必须在预先发送HTTP请求前设定正确的文件名，以使浏览器在发送本次HTTP请求时已判断出该激活那个应用程序去正确接收服务器端发送的数据，这是成功发布格式化文档的关键。下面给出一个实例讲述Word文档的发布过程：

3.2 程序实现

首先需要设定正确的表单格式：

```
<FORM ACTION="/servlets/CallWord/Test.doc" METHOD="POST" >
<INPUT TYPE="hidden" NAME="fileName" VALUE="1.doc">
<INPUT TYPE="submit" VALUE="Word 编辑 ">
</FORM>
```

当用户点击"Word编辑"键，浏览器构造本次HTTP头信息时判断出目标文件指向Test.doc，通过扩展名可知需要调用Word程序才能正确显示服务器应答的信息，服务器端接收到用户的HTTP请求后，可以解析出是Post方式请求，通过分析URI信息，即"/servlets/CallWord/Test.doc"，WEB服务器会激活名为CallWord的Servlet程序，该程序通过读取参数信息，从数据库中获取相应的Word文档，通过设定正确的ContentType来应答客户机请求。下面简述CallWord的实现过程：

```
import java.sql.*;
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import oracle.sql.*;
import oracle.jdbc.driver.*;
public class CallWord extends HttpServlet{
public void init(ServletConfig config) throws ServletException
{
Class.forName(DB_DRIVER); // 初始化JDBC接口
super.init(config);
}
private Connection openConnection() throws Exception
{
    return DriverManager.getConnection(DB_NAME, DB_USER,
DB_PASSWORD);
    // 打开数据库连接
}
private void closeConnection(Connection conn) throws Exception
{
  conn.close();   // 关闭数据库连接
}
/**
* 处理HTTP Post 请求
*/
public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
{
InputStream in=null; // 设置输入流
oracle.sql.BFILE bfile=null; // 设置Oracle Bfile 变量
response.setContentType("application/msword"); // 设置输出文档类型
为MS WORD格式
OutpubStream out = response.getOutputStream(); // 初始化输出流
String fileName = request.getParameter("filename");// 从URL信息中获
取文件名
Connection myconn = openConnection(); // 打开数据库连结
Statement stmt= myconn.createStatement(); // 实例化连结对象
String query = "SELECT BFILENAME('" + DB_DIRECTORY_OBJECT
+ "', '" + fileName + "') from dual"; // 构造查询串
```

```
ResultSet rset=stmt.executeQuery(query.toString()); // 执行查询
if (rset.next ()) {
bfile = ((OracleResultSet)rset).getBFILE (1); // 获取图像文件指针
bfile.openFile(); // 从数据中打开图像文件
in = bfile.getBinaryStream(); // 以二进制方式读图像文件
int length;
byte[] buf = new byte[4096]; // 以 4K 字节为一组读取数据
while ((length = in.read(buf)) != -1) { // 判断是否读到文件尾
out.write(buf); // 以流模式输出
buf = new byte[4096]; // 重新初始化字节变量
}
in.close();
bfile.closeFile();
out.close();
}
}
```

上面的程序片断演示了通过表单方式发布大对象数据的方法，不难看出，这种方法适用于多种格式化文档的发布，比较灵活。

4 通过WEB服务器URL重定向方式发布大对象数据

4.1 理论分析

对于已经预先编写完成的静态HTML文档，前两种发布方式并不非常适用，原因在于这些静态文档包含了大量大对象数据的链接，如果将这些链接全部重新改写非常复杂，而且可能出错，比如保存了这样一个HTML文档在数据库中：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
</head>
<BODY ><center>
<img src="/images/1.gif"/>
<img src="/images/2.gif"/>
<img src="/images/3.gif"/>
<br>这是一个 HTML 文件
<BR>
</center></BODY>
</html>
```

一个简单的HTML文件链接了3张图片，若使用前两种方法必须重新改写上面的HTML文件，对于大量的静态HTML文档这种方式显然不适用，必须找出一种方法，让系统自动将这些大对象数据的超链指针重定向到一个Servlet程序，通过该程序实现对大对象数据的发布。

通过认真分析Apache的Rewrite模块，我们发现它具备服务器端URL重定向功能。再来分析静态HTML文件，所有图像文件都放置在images路径下，若能将images路径信息作为激活Apache Rewrite功能的虚拟URL，即可实现在服务器端触发Servlet程序。

Apache的Rewrite模块功能非常强，它通过预先设定的RewriteRule（重写规则）实现服务器端URL重定向，当开启重写功能后，Apache服务器要检查每个HTTP请求所定位的URL信息(引自参考文献[19])，如果与某个重写规则相同则执行该重定向逻辑，现举例说明如何实现前面的images URL重写逻辑：

```
<IfModule mod_rewrite.c>
RewriteEngine on  // 打开 Apache 重写引擎
RewriteRule ^/images(.*) /servlets/GetImage$1 [P]
</IfModule>
```

首先打开Apache重写引擎，接着定义重写规则，当发往WEB服务器的URL信息中含有images字符串时，Apache服务器将把images后面的信息赋值给变量$1，并将$1追加到GetImage程序后面，其实我们很清楚$1保存的就是图像文件名，使用[P]功能实现强制代理，则Apache服务器在完成重写逻辑后，将继续执行GetImage程序，GetImage程序可从URI信息中获取到用户所需的图像文件名。

一个完成的重写过程在服务器端会产生如下动作：

客户提交的URL信息：/images/1.jpg

经过Apache服务器重写后的URL信息：/servlets/GetImage/1.jpg

4.2 程序实现

接下来的数据发布过程由GetImage程序完成，程序片断实例如下：

```
import java.sql.*;
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import oracle.sql.*;
import oracle.jdbc.driver.*;
public class CallWord extends HttpServlet{
public void init(ServletConfig config) throws ServletException
  {
   Class.forName(DB_DRIVER); // 初始化 JDBC 接口
   super.init(config);
  }
```

```
private Connection openConnection() throws Exception
{
    return DriverManager.getConnection(DB_NAME, DB_USER,
DB_PASSWORD);
    // 打开数据库连接
}
private void closeConnection(Connection conn) throws Exception
{
  conn.close();    // 关闭数据库连接
}
/**
* 处理 HTTP Post 请求
*/
public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
{
String  URI_DIRECTORY_PREFIX = "/servlets/GetImage";
InputStream in=null; // 设置输入流
oracle.sql.BFILE bfile=null; // 设置 Oracle Bfile 变量
String requestURI = new String(request.getRequestURI().getBytes("iso-
8859-1"), charSet); // 获取完整 URI 信息
String fileName = requestURI.substring(URI_DIRECTORY_PREFIX.
length());
// 从 URI 信息中获取文件名
Connection myconn = openConnection(); // 打开数据库连接
Statement stmt= myconn.createStatement(); // 实例化连接对象
String query = "SELECT BFILENAME('" + DB_DIRECTORY_OBJECT
+ "', '" + fileName + "') from dual"; // 构造查询串
ResultSet rset=stmt.executeQuery(query.toString()); // 执行查询
if (rset.next ()) {
bfile = ((OracleResultSet)rset).getBFILE (1); // 获取图像文件指针
String BfileName = bfile.getName(); // 得到文件名
response.setContentType(getServletContext().getMimeType(BfileName));
// 设定所需数据的文件类型属性
OutpubStream out = response.getOutputStream(); // 初始化输出流
bfile.openFile(); // 从数据库中打开图像文件
in = bfile.getBinaryStream(); // 以二进制方式读图像文件
int length;
byte[] buf = new byte[4096]; // 以 4K 字节为一组读取数据
while ((length = in.read(buf)) != -1) { // 判断是否读到文件尾
out.write(buf); // 以流模式输出
buf = new byte[4096]; // 重新初始化字节变量
}
in.close();
bfile.closeFile();
out.close();
}
}
```

通过Apache重定向功能实现的大对象数据发布方式，非常适用于存储在数据库中的HTML文档，这些文档中存储的大对象链接指针可以通过不同的重写规则实现对大对象数据的发布，数据类型基本不受限制，所有的重定向过程全部在服务器端完成，对用户端是透明的，很好的隐藏了技术细节，保证了一定的系统安全性。

5 三种发布方式的应用比较

前面三种大对象数据发布方式，分别针对不同的数据类型及应用场合，现将其特征总结如表2：

大对象数据的发布方式直接决定了系统能否构建成分布式体系结构，通过分析HTTP协议的请求、响应状态及WEB服务器的内容协商机制，本文提出三种实现大对象数据WEB环境发布的方法，分别针对不同数据类型，首先将所有数据存入数据库中，对于图像数据可以使用HTML标签方式发布，对于格式化文档使用HTML表单方式发布，对于静态HTML文件使用Apache重定向模块实现发布，这三种发布方式服务器端均使用Java Servlet程序访问数据库，以流模式实现大对象数据的发布，大对象数据的灵活发布方式较好的解决了数据集中存储，分布发布的问题，对于构建三层体系结构的信息服务系统提供了基本技术保证，将所有数据存储到数据库不仅实现了较高的安全级别，而且较好的保证了数据的一致性。

表2 三种发布方式的应用比较

发布方式	适用数据类型	其他特点
HTML 标签方式	图像数据	简单、易行，服务器端不需做设置，适用于直接链接发布的大对象数据
HTML 表单方式	任何格式化文档数据	简单、易行，服务器端不需做设置，适用于二次链接发布的大对象数据
Apache 重定向方式	适用于静态 HTML 文件	服务器端需设置URL 重写规则，略复杂，适用于各种链接数据，重写过程在服务器端完成，对用户端透明，具有较高安全性