.Net Remoting 构建 Web 服务在远程数据传输上的应用研究①

Remote Data Transmission Based on .Net Remoting Web Service

左 娟 张 毅 (重庆大学 软件学院 重庆 400044)

摘 要: 本文分析了.Net Remoting 和 Web Service 的体系结构，并使用.Net Remoting Web 服务技术搭建了油田钻井数据实时传输平台，克服只能通过 Http 通道来访问 Asp.Net Web 服务、企业内联网解决方案中请求响应速度不快以及服务只能通过 IIS 调用等问题。同时研究了 WinForm 客户端和 Asp.Net 客户端如何使用 Tcp 通道和 Http 通道访问同一进程不同应用程序域的两个远程对象、调用异步远程事件、创建事件接收器等方法，为局域网和互联网之间通信提供了一个比较好的解决方式。

关键词: .Net Remoting Web 服务 远程数据传输

随着计算机和网络技术的不断发展，人们越来越要求程序具有“分布性”，.Net 体系结构中的 Remoting 技术是 DCOM 技术的下一代，支持进程间的通信和更多的通信协议，对于构建一个集效率和跨平台、可扩展性于一体的信息系统是个不错的选择[1]。Web 服务使用基于 XML 的消息处理，作为基本的数据通讯方式，消除了使用不同组件模型、操作系统和编程语言之间存在的差异，使异构系统能作为单个计算机网络协同运行，因此具备很强的生命力。

传统的 Asp.Net Web 服务是在 Internet 上调用服务的一项易用的技术，但在企业内联网解决方案中，这项技术对于某些业务请求来说并不快，同时还需要有 Asp.Net 运行时的支持。使用.Net Remoting 技术后，可以将 Web 服务提供给世界上的任何地方，而且可以在所有应用程序类型中运行 Web 服务。

随着油田企业集团化的重组不断加快，集团企业越来越关心每个生产现场的投入与产出、节能与降耗、增产与增收。本文使用.Net Remoting 构建 Web 服务实现了获取实时数据的需求，并构建了数据实时传输平台，为企业生产决策提供准确、及时、全面的数据分析和管理平台。

1 .Net Remoting框架概述

.Net Remoting 是构建分布式应用程序的一种技术，提供了一种允许一个应用域中的对象与另一个应用域中的对象进行交互的框架，它支持 HTTP，SOAP ，WSDL 以及 XML 此类的开放标准，从而达到了协同工作的目标。.Net Remoting 提供的编程模型和运行时支持功能强大又易于使用,能够实现透明的交互操作[2]。

在图 1 的客户端应用程序域和服务端应用程序域的分界线构成了.Net Remoting 的边界。服务端(即服务端应用程序域)由传输通道、序列化格式程序和服务器端对象组成(运行在服务器上的对象为远程对象)。客户端(即客户应用程序域)如图示由客户端对象、代理、传输通道、序列化格式程序组成。当访问远程对象时客户端并不处理真实对象，而是仅仅调用 Proxy 对象的方法[3]。通过使用格式化程序类，可以串行化这些消息，并将这些消息发送到客户通道中。通道(.NET Remoting 提供了 HTTP 和 TCP 两种通道)是一种远程框架，它隐藏了在客户和服务器应用程序间通信时所使用的底层协议。

① 收稿时间:2008-08-18

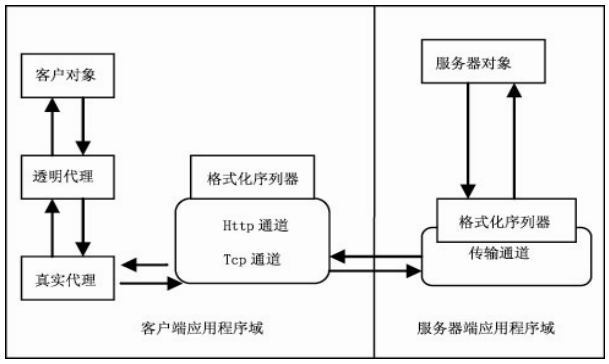

图 1　.Net Remoting 体系结构

2　Web Service体系架构

Microsoft 环境提供四种创建 Web 服务的方法：Asp.Net、.Net Remoting、VB SOAP Toolkit 和 ATL Server。Web 服务是可编程的组件,它提供在可缩放的、松耦合的和特定平台的环境下交换信息的能力,信息交换使用 HTTP、XML 、SOAP 和 WSDL 之类的标准协议。由于 Web 服务基于不特定的平台,因此可以用于多种多样的实现、平台和设备进行通信。

Web 服务体系基本体系结构基于三种角色:服务提供者、服务注册中心和服务请求者之间的信息交换。Web Service 体系结构如图 2[4]。一般情况下,服务通过网络访问软件模块,服务提供者定义并发布到服务请求者或者服务注册中心。服务请求者使用查找操作从本地或服务注册中心搜索服务描述,然后使用服务描述与服务提供者进行绑定并调用相应的 Web 服务实现同它信息交换。

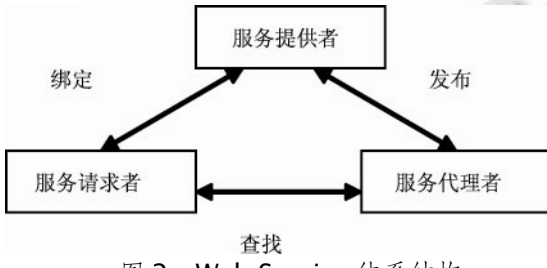

图 2　Web Service 体系结构

3　油田钻井数据实时传输平台的设计及实现

3.1 系统简介及主要功能

把井场端数据实时地传输到油田基地，实现基地对现场分散目标的实时监控是现代化生产管理的重要手段。目前大量现成的数据传输软件大多适用于文件的传输，并且需要过多操作，不适用于数据的实时传输。井场端的计算机负责实时数据的获取、以及数据文件和图像文件的发送，并接收来自基地的指令。基地端的计算机负责数据的接收、显示数据和下达指令。该系统整体结构如图 3。

油田钻井远程数据实时传输平台主要实现现场采集数据，并通过井场端实时传输到技术分析中心，中心经过分析处理，再把分析数据发回到现场，指导现场生产操作。系统主要由数据采集井场端，信息处理平台和技术分析监测站三部分组成。系统各部分的功能如下：

(1)数据采集井场端：通过现场采集的数据汇总传输到基地技术分析中心，指导现场生产操作。

(2)信息处理平台：对现场数据进行运算处理，根据用户的需要对现场原始生产数据进行统计计算，对数据传输功能外，还具有数据、曲线和图形的显示的实现、参数的计算和实时入库完成数据处理及实现异常报警等功能。

(3)技术分析监测站：负责数据的接收、显示数据和下达指令，同时每个技术分析监测站还可以和中心技术分析监测站进行实时通信。

当有很多客户端同时向中心监测站传输文件时，一是由于宽带有限,可能会出现网络堵塞,二是数据库读写速度慢而影响文件上传速度。针对以上两种情况，采用排队的方式，即一次允许一定数量的客户端同时传输文件和可以先让文件写入缓存中,然后再写入数据库的方式解决。

3.2 基于.Net Remoting 构建 Web 服务在本系统的设计

本方案采用服务端/客户端模式, Asp.Net 客户端发送数据，完成现场采集数据的汇总、数据包的封装、发送等工作。 服务器通过订阅 Asp.Net 客户端提供的数据服务获取数据， 完成数据包的接收、解析，以及大量的数据处理等工作。同时服务器端也完成向 WinForm 客户端发送经过分析处理过后的数据等功能。而 WinForm 客户端也负责接收处理和显示数据并可以进行实时通信等工作。其中封装和解析分别完成从数据缓存区获得实时数据，根据约定的数据协议封装成数据包和根据约定的数据协议对数据包进行解

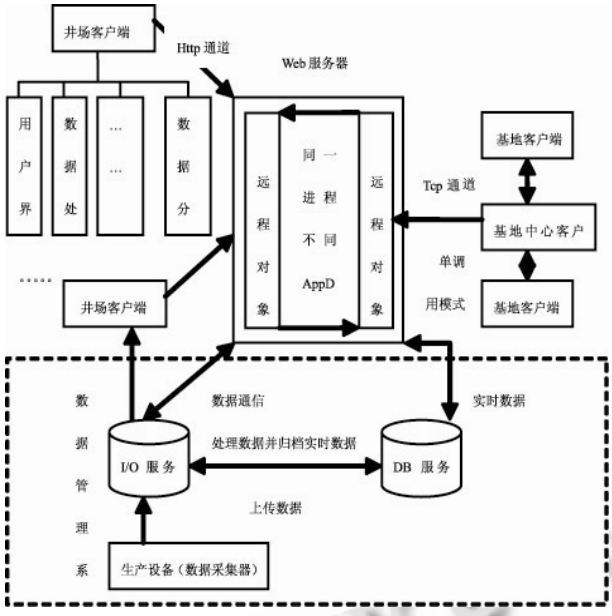

图 3　油田钻井远程数据实时传输平台结构图

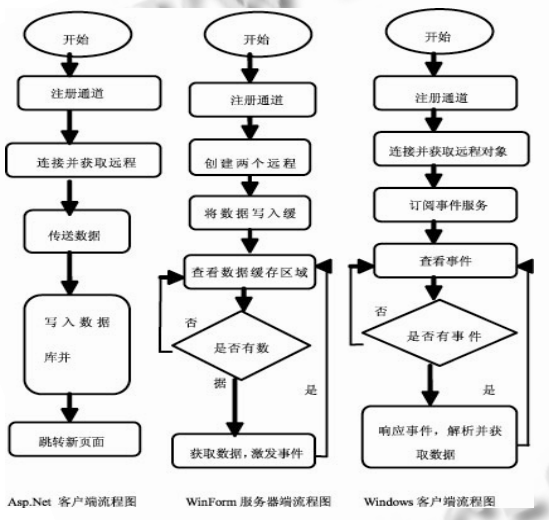

图 4　远程数据实时传输各部分流程图

析，获得实时数据，数据实时自动化传输各部分流程如图 4。

3.3 远程对象

远程接口是 web 服务提供给外界的方法,具体实现 web 服务方法是在远程对象中进行的。在本系统中包括两个 web 服务：InforOderWebService 和 InforDeliveryWebService。这两个 Web 服务是运行在同一进程但不同应用程序域之中。主要实现如下：

```
public interface InforInterface;
public class InforOderWebService:MarshalByRefObject，InforInterface
    {
    ......//访问组件间的调用，处理等代码
    }
Public class InforDeliveryWebService:MarshalByRefObject，InforInterface
    {
    ......
    }
```

在.Net Remoting 的体系结构中只允许在一个应用程序域使用一个远程应用程序[5]，因此创建一个新的应用程序域 NewAppDomain，目的是使 WinForm 客户端和 Asp.Net 客户端的对象能使用不同的通道。但在公共语言运行库中禁止在不同域中对象之间的直接调用，所以通过 ObjectHandle 代理类在应用程序域间传递对象。主要代码如下：

```
// 创建一个新的应用程序域 DataDeliveryAppDomain
AppDomain DataDeliveryAppDomain=AppDomain.CreateDomain("NewAppDomain");
//类 ObjectHandle 用于在多个应用程序域之间传递对象
ObjectHandle oh=DataDeliveryAppDomain.CreateInstance("InfoWindowsService","InfoWindowsService. DataDeliveryAppDomain ");
//在应用程序域中解包返回的 ObjectHandle 对象
DataDeliveryAppDomain datadelivery = (DataDeliveryAppDomain)oh.Unwrap();
```

3.4 设计用于转载数据的数据包装类

数据包装类用来装载各种实时数据，是数据传递的载体。数据包装类主要包含井测深度数据队列、钻杆数据队列等。由于此类实例要通过网络传输，因此必须可以序列化，方法是给此类加序列化属性。设计如下：

```
[Serializable]  //可序列化属性
public class OilDataWrap  //数据包装类
{……
Private ArrayList remoteDeapthList = new
```

```
ArrayList();
……
Public void AddRemoteMeasure(UpdateRemote
MeasureList measuredata)
//添加方法
public UpdateremoteDeapthList [] GetremoteDeapthList()
……}
```

最终将其编译成 OilDataWrap.dll 程序集，以便在程序中引用。

3.5 异步远程事件处理

通过网络调用远程对象可能会花费一段时间。当异地调用本地对象时，一个线程在响应用户请求进行其他操作的时候，可以通过委托的方式，自动创建另一个进行远程调用的线程。通过使用异步调用远程方法的委托回调，即当异步远程方法结束后，调用 BeginInvoke()方法传递 AsynCallback 委托获得回调[6]。主要实现如下：

```
//WebClient
private delegate int HoriDrillingDepth(int MonitorErp, int MonitorAdf,int ms);
public static voidLongTimeAddCallback(IAsync Result ar)
{.....
    //hddp 是 HoriDrillingDepth 的静态实例
      int result=hddp.EndInvoke(ar);
.....}
```

同时还需要创建 AsyncCallback 委托的新实例，并为将被异步调用的委托传递一个引用。

```
InforOderWebServiceinforOder= new InforOderWebService();
hddp.HoriDrillingDepth(inforOder.LongTimeAdd);
AsyncCallback qwt= new AsyncCallback(WebClient.LongTimeAddCallback);
//调用 BeginInvoke()方法传递委托
IAsync Resultary=qwt.BeginInvoke(21,52,300, qwt,null);
```

3.6 创建一个事件接收器类

为了能在数据处理后立即显示，创建了一个接收器，名称为 InforDataArrivedSink，且派生于 MarshalByRefObject，因为只有这样网络才能访问它。在客户端程序创建一个实例，而该实例在服务器端被调用。[OneWay]属性声明的方法是一个只能被调用能没有返回结果的方法，该方法将被异步调用[7]。创建接收器主要实现代码如下：

```
Public class InforDataArrivedSink:MarshalByRefObject
{  [OneWay]
    //InforStatus 类带有[Serializable]属性，能在网络上编组
    Public void InforDataArrivedSink(object sender, InforStatus ins)
    {……
      LableNewDataArrived.text=ins.DataText;
……
    }
}
```

3.7 客户端与服务器端的实现

客户端和服务器端主要完成所有数据封送和拆收、安全控制、传输通道配置和任何其他附加工作等。

本系统的客户端有 WinForm 客户和 Asp.Net 客户端两部分，都是用配置文件实现。其中，WinForm 客户端注册的是 TCP 通道，<channel ref="tcp" port="8003" >，而 Asp.Net 客户端注册的是 HTTP 通道，<channel ref="http client">。并且需要将序列化级别设为高，代码如下：

```
<formatter ref="soap" typeFilterLevel= "Full" />
<formatter ref="binary"typeFilterLevel= "Full" />
```

Asp.Net 客户应用程序配置文件为 Web.Config，不用显示地调用RemotingConfiguration.Configure()，它是由 Asp.Net 运行时自动读取的。

为了使该服务可以使用已配置的帐户来使用并且能在系统启动时自动启动，所以把主控服务器做成了 Windows 服务，通过添加安装服务程序，可以安装和配置服务应用程序，可以通过使用命令行实用程序 Installutil 来安装服务。使用 OnStart()方法的配置文

件来配置通道,主要代码如下:

```
protected override void OnStart(string[] args)
{……
    RemotingConfiguration.Configure("InforOd
erWebService.config");
.....}
protected override void OnStop()
{
    IChannel[]channels=ChannelServices.Regist
eredChannels;
    foreach (IChannel channel in channels)
    {ChannelServices.UnregisterChannel(chann
el);}
}
```

使用 OnStop()方法停止正在运行的通道，通过 RegisteredChannels()返回所有在运行中注册的通道。

3.8 性能分析

.Net Remoting 性能是相对的，与硬软环境以及相应集成环境相关，与客户连接处理数据方式相关等。基地客户端通信用二进制编码，传输性能高，对安全性要求不是很高。井场客户端用 Http 信道传输性能有所降低，但安全性增强，同时为了保证数据传送的可靠性，本文对传送的报文数据要进行数据校验处理。在发送端对报文数据加入校验码，在接收端对数据及校验码进行检验，对误码进行纠错处理有效地解决线路干扰严重、误码率较高的问题。

4 结论

本文利用.Net Remoting 技术构建 Web 服务，克服了 Asp.Net Web 服务需在 Asp.Net 运行时环境下支持的特点，使得在在局域网内传输速度可以得到优化。同时利用该技术构建的本系统具有很强的灵活性和扩展性，为处理局域网和互联网范围内的通信提供了一个好的实现方法。

参考文献

1 Walther S.ASP.NET 技术内幕[2].马朝晖.等译.北京:机械工业出版社,2002.

2 张永峰,李也白,宋磊. . NET Remoting 构建分布式数据库查询.计算机与信息技术, 2008-02-02.

3 陈琨,陈福民.基于.NET Remoting 利用软件方法实现网络教学的探索.计算机应用,2003,23(8):128－134.

4 Feusel D, Bussler C. The web service Modeling Framework WSMF. Electronic commerce Research and Applications, 2002, (1):113－116.

5 Srinivasan P. An Introduction to Microsoft . Net Remoting Framework. Microsoft Corporation, 2001.

6 Pham TQ , Garg PK. Multithreaded Programming with Windows NT. Prentice Hall , 1996.

7 Matthew MacDonald. NET 分布式应用程序:集成 XML Web 服务与.NET 远程处理.戴中东译.北京:清华大学出版社, 2005.