基于CORBA的分布式数据采集系统软件的设计与实现

Design and Implementation of Software of Distributed Data Collecting System Based on CORBA

摘　要：CORBA规范是分布式计算的重要工业标准，它具有平台独立性和语言无关性，在分布式计算领域应用越来越广泛。本文提出了在CORBA基础上的分布式数据采集系统，讨论了系统的模型及所涉及的关键技术。

关键词：CORBA 分布式对象　数据采集　远程控制

于绍娜　（长春吉林大学计算机科学与技术学院 130012）
李　冶　（长春吉林大学国土资源部现代地球物理仪器开放研究实验室 130026）
蔡学森　（长春师范学院计算机科学系 130032）
sn@163.net

随着网络技术和分布式计算技术的发展，基于计算机网络的分布式数据采集与控制已成为日益重要的远程测控途径。在分布式数据采集系统中，采集终端在地理上是分散的，服务器可以远程对其进行控制，并且能得到设备采集到的数据及设备的状态，从而实现了资源共享。而这些采集终端可能存在于不同的局域网或者不同的操作系统平台，这要求分布式采集系统具有很好的跨平台性。CORBA具有平台独立性和语言独立性，因此成为这种异构环境下分布式计算理想的解决方案。本文在介绍了CORBA技术的基础上，给出了基于CORBA的分布式数据采集模型，然后阐述了系统实现的关键技术。

1 CORBA 概述

CORBA(Common Object Request Broker Architecture，公共对象请求代理体系结构)是由对象管理组织（OMG）提出的一个完整的分布对象处理与集成的体系构架，支持异构分布应用程序间的互操作性及独立于平台和编程语言的对象重用。

CORBA的核心是ORB(Object Request Broker，对象请求代理)，它负责将客户的请求传递给服务器并将结果返给客户。客户并不知道服务器在何处，客户如何与服务器通信，目标对象是如何实现的，ORB提供了对象的位置透明性、对象之间通信和对象实现的透明性。CORBA提出基于不同平台、不同编程语言、不同网络协议的异质系统间互操作的“软件总线”概念。并且支持分布式和面向对象，把客户与服务器抽象为对象，所有功能都封装在对象内部，只向外提供简单的接口。对象间的通信由CORBA代理，通过软件总线机制对象请求代理(ORB)实现，对象不必关心通信对方的实现细节。CORBA还提出了接口定义语言(IDL)这一与平台无关的语言，提供编程语言无关性的支持，实现应用接口与代码的分离。此外，CORBA定义了一组与应用领域无关的基础对象服务，使得对象位置透明，而不用关心其物理位置。图1显示CORBA的原理图。

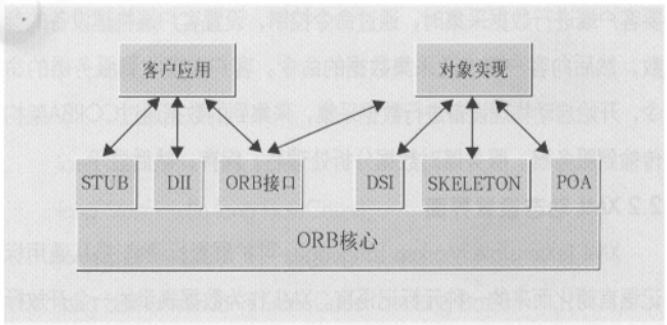

图 1　CORBA 原理图

2 分布式数据采集系统

2.1 基于 CORBA 的分布式数据采集模型

本系统所包括的数据采集设备分布在不同的地理位置上，在客户端可直接对设备进行控制，在此认为物理设备端为客户端。设备的状

态传输到服务器后，服务器根据客户端传来的设备状态，向客户端发控制命令，使服务器在远程可实现对设备的控制。设备采集的数据传送到服务器后，由服务器完成数据的存储、处理和显示。图2给出了本系统的模型。

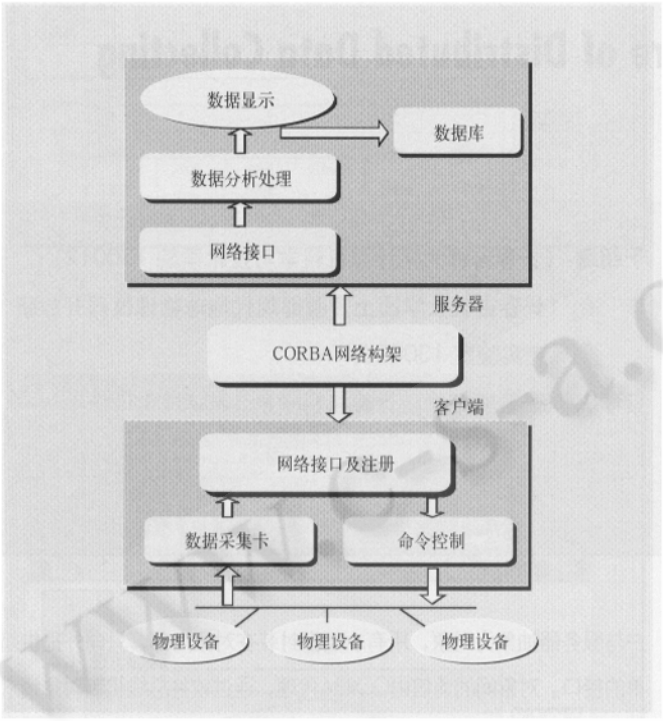

图 2　基于CORBA的分布式数据采集模型

由图1看出，系统是由客户端和服务器通过CORBA构架组成的，系统中存在多个客户端。客户端软件启动时，通过接口在服务器注册，服务器接收客户启动消息，确定那个设备可以工作。当服务器需要客户端进行数据采集时，通过命令控制，设置客户端物理设备的参数，然后向客户端发送采集数据的命令。客户端接收到服务器的命令，开始启动物理设备进行数据采集，采集到的数据通过CORBA架构传输到服务器。服务器对数据分析处理后，保存，然后显示。

2.2 XML动态设置界面

XML (eXtensible Markup Language, 可扩展置标语言)是从通用标记语言简化而来的一种元标记语言。XML作为数据表示的一个开放标准，独立于机器平台、提供商和编程语言。XML具有自描述的功能，可以表达和描述具有复杂结构和丰富语义的信息，并且具有强大的扩展能力。

为了系统的可扩展性，我们对所有设备采用了统一的客户端。系统要对不同采集设备进行控制，而不同的设备具有不同的控制参数，同时系统的设备是动态添加和删除的，这就要求系统能够提供对设备增减的灵活的支持。因此，采用了XML这种可描述的语言，在程序运行时在服务器端动态产生各个设备的控制界面，而不是预先固定产生各个控制界面。当有设备添加或删除时，只需修改这个设备的客户端XML语言。例如，根据如下的描述，在服务器端动态产生RVO的控制界面。

```
- <VL>
-<Proxy
datatype="long" position="Remote" caption="RVO">
- <Param>
- <Frequence
caption="采样频率"
datatype="long"
value="0" controltype="udedit">
 <Relative min="100"
max="32767"
step="100" />
 </Frequence>
- <Range
caption="范围"
datatype="long"
value="5"
controltype="udedit">
 <Relative
min="0"
max="1000"
step="1" />
 </Range>
- <Angle
caption="相位"
datatype="long"
value="0"
controltype="udedit">
 <Relative
min="0"
max="360"
step="1" />
 </Angle>
 </Param>
 </Proxy>
```

</VL>

2.3 分布式采集系统的IDL

OMG IDL是一种与平台无关的语言，它可以映射成不同的语言。OMG IDL接口定义语言不是作为程序设计语言体现在CORBA体系结构中的，而是用来描述产生对象调用请求的客户对象和服务对象之间接口的语言。OMG IDL文件描述数据类型和方法框架，而服务对象则为一个指定的对象实现提供上述数据和方法。OMG IDL文件描述了服务器提供的服务功能，客户机可以根据该接口文件描述的方法向服务器提出业务请求。

本系统服务器与客户端交互信息时包括了各种各样的数据类型，因此，要建立一种灵活的数据类型转换机制是本系统所必须达到的目标之一。CORBA中的DynAny接口可提供一种在运行时动态创建基本数据类型和复合数据类型的途径，但DynAny对系统要求很高，开销也大。对仪器系统而言，多数的数据都是大批量同类型数据。同时，不同的设备数据类型是不同的，为此在本系统中我们实现了一种简化的Any数据类型，并利用一种专门的数据操作类来实现打包和解包，从而达到了较为理想的效果。自定义数据结构的构成如图3所示。

图 3　自定义数据结构

下面给出了IDL语言定义的接口定义文件，它实现了动态创建数据类型、数据传输以及参数、状态操作等。

```
{
.....
typedef enum _DataType{dtLong, dtFloat, dtString, dtOctet, dtBool,
dtTime, dtAny, dtCustom, dtOther}TDataType;
    struct TTVPair
    {
      TDataType Type;
      string  Value;
    };
    typedef sequence<TTVPair>TTVSeq;
    typedef union _DataValue switch(TDataType)
    {
     case dtLong: CORBA::LongSeq  LongValue;
     case dtFloat: CORBA::FloatSeq  FloatValue;
     case dtString: CORBA::StringSequence
          StrValue;
     case dtOctet: CORBA::OctetSequence  OctetValue;
     case dtTime: CORBA::StringSequence   TimeValue;
     case dtAny:  any          AnyValue;
     case dtCustom: TTVSeq      TVValue;
     case dtOther:  CORBA::LongSeq        OtherValue;
    }TDataValue;
    typedef struct _Node
    {
     long      PinID;
       string     Name;
       string     Address;
    }TNode;
    typedef struct _Link
    {
     TNode     CurNode;
     TNode      NextNode;
    }TLink;
    interface IGate
{
 .....
long SetFlow(in TLink Link);
long StartFlow(in long FlowCount);
long EndFlow();
long SetStatus(in long StatusCode, out long Back);
long SetStatus(in long StatusCode, out long Back);
long SetParam(in TDataValue Params, out long Back);
.....
}
}
```

3 结论

本系统利用XML实现了动态界面设置，前端可以采用不同类型的数据采集设备。当增加新采集设备时，只需修改此采集设备的XML语句，而不需要修改服务程序，提高了程序的可扩展性。同时，使用OMG IDL建立了动态数据类型，可在程序执行的过程中动态选择数据类型，不必在编程时限定，增加了系统的灵活性。

基于CORBA技术，更方便、更透明地实现了系统中各分布式应用对象之间的互联互操作，有效地实现了分布式数据采集系统，提高了应用编程的开发效率，方便了系统的升级和维护。

参考文献

1 Michi Henning, Steve Vinoski[美]，徐金梧等译，基于C++CORBA高级编程，清华大学出版社。

2 [美]怀特(White,C.)著，周生炳等译，XML从入门到精通，电子工业出版社,2002.1。

3 周如培等，基于CORBA的电信计费系统的研究与开发，计算机应用研究,2002;8:130-131。

4 张佳雨等，基于CORBA/XML的新一代BOSS系统的设计与实现，电脑开发与应用,2003；16(7)：10－11。

5 尹学举等，基于CORBA的远程控制及实现，计算机工程,2003;29(2): 78-80。

6 Borland/ Inprise 公司著，VisiBroker for Java 开发人员指南，机械工业出版，2000，11。