用 DCOM 技术实现 DCS 与其 OPC 服务器的通信

石 韬 薛福珍 方金胜 柏 洁 中国科学技术大学自动化系(230027)

Abstract

This paper introduced how to change DCS to out-of-process COM Server using DCOM technology,and present a simple object model and a general COM interface for DCS,to realize the communication between DCS and its OPC server based on that interface.With good abstractness,the interface isolates DCS from its OPC server so that the OPC server no longer depends on the concrete implementation of DCS.

Keywords:COM,DCOM,interface,DCS,OPC

摘 要

本文介绍了如何使用 DCOM 技术将 DCS 改造成进程外 COM 服务器,为 DCS 提出了一种简单的对象模型并设计了较通用的 COM 接口,利用该接口实现了 DCS 与其 OPC 服务器的通信。该接口具有良好的抽象性,很好的隔离了 DCS 及其 OPC 服务器,使得 OPC 服务器不再依赖于 DCS 的具体实现。

关键词:COM,DCOM,接口,集散控制系统,OPC

OPC 是一个基于 COM/DCOM 技术的工业标准，解决了自动化控制应用、现场设备系统和商业办公室应用系统之间的互操作问题。OPC 数据访问接口 (OPC Data Access Interface)是 OPC 的主要接口,一般简称为 OPC 接口。DCS 作为一个底层数据采集系统，需要对外公开其实时数据,OPC 接口正好是合适的标准接口。为了保证 DCS 的可靠性,并且由于 OPC 接口的独立性和复杂性，最好将 OPC 接口从 DCS 的进程空间分离出来,成为一个独立的 OPC 服务器,这样也使 OPC 服务器可以分布在网络的其他机器上。这就需要解决 DCS 与其 OPC 服务器跨进程甚至跨机器的通信问题。这一问题至今未见文献报道,本文在深入分析和研究工作的基础上,用 DCOM 技术实现了 DCS 与其 OPC 服务器的通信,相比其他通信方式(如 socket 或 DDE 等),具有面向对象、位置透明、高效、安全可靠等优点。

1 DCS 与其 OPC 服务器通信的体系结构

DCS 与 OPC 跨进程通信的基础采用 DCOM(Distributed Component Object Model,分布式组件对象模型),它建立在 DCE RPC(远程过程调用)标准协议之上。DCS 与其 OPC 服务器的通信体系结构如图 1 所示。在 DCS 进程里,数据库存储所有现场设备点的信息,包括组态信息和从现场采集的实时数据;IDCSDb 接口是针对 OPC 服务器的需要而自定义的 COM 接口,是关于 DCS 的一个简单抽象模型,而 DCSDb 对象则实现了这个接口并且要求数据库为其扩充一些必要的接口函数。在 OPC 服务器进程里,点设备对象是一个抽象概念,就是点集合的意思,它利用封装的 IDCSDb 接口指针通过 DCOM 与不同进程甚至是不同机器的 DCSDb 对象互相通信。OPC 客户进程则利用 OPC 接口通过 DCOM 与 OPC 服务器进程里的服务器对象和组对象通信。

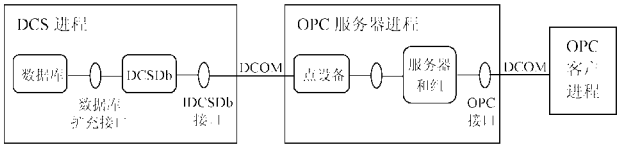

图 1 体系结构图

2 OPC 对象模型简介

OPC 服务器由三种不同层次的对象组成,即:服务器(Server)、组(Group)、点(Item)。三种对象的关系如图 2 所示。

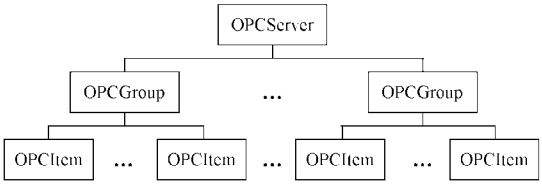

图 2 OPC 对象层次图

服务器对象维护服务器的信息并作为组的容器，一个组维护自身的信息并在逻辑上组织多个点。点提供 OPC 服务器同数据源的连接,是读写数据的最小逻辑单位,即一个点代表了一个设备变量。点由若干属性组成,一个属性包含四个域:ID、数据类型、标准描述、值。OPC 规范将点的属性分成了三类:第一类是 OPC 规定必须提供的属性,ID 范围是 1-99,目前只规定了前六个属性,分别为点的数据类型、值、质量、时间戳、访问权限与服务器扫描频率；第二类是 OPC 推荐使用的属性,ID 范围是 100-4999,包括 EU Units、High EU、Low EU 等一个点常用的一些属性；第三类是服务器提供者自己规定的属性,ID 范围是 5000 及其以上。

3 IDCSDb 接口的设计、实现及使用

IDCSDb 是 DCS 提供给 OPC 服务器的 COM 接口，其功能不但要能满足二者之间的基本通信要求，而且还要很好地将二者隔离开,从而使 DCS 的改变不会影响其 OPC 服务器。设计 IDCSDb 的时候要综合考虑 DCS 和 OPC 二者的特点，按照 OPC 的要求抽象出 DCS 的模型，而且要考虑到设计后接口使用的方便性和高效性。

3.1 设计

IDCSDb 接口的设计基于对 OPC 接口规范和 OPC 对象模型的研究。OPC 服务器要求 DCS 提供的信息主要是 DCS 实时数据库中的数据,还有 DCS 的运行状态、扫描频率以及 DCS 关闭事件等信息。对于 OPC 服务器而言,实时数据库是 DCS 的主要特征。可以把 DCS 实时数据库看作是点的集合,每个点又是属性的集合,每个属性有固定的结构:ID、数据类型、描述和值。

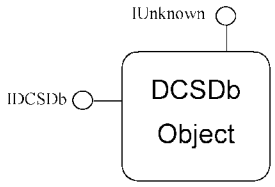

图 3 DCSDb 对象

这样就建立了 DCS 的一个简单的抽象模型,以 DCSDb 对象来表示,见图 3。

IDCSDb 的接口函数设计如下:

(1)HRESULT SetItemRange(
[in] long ItemID__Len, // ItemID array's length
[in] LPWSTR *pItemID, //ItemID array
[out] HRESULT **ppError) //error info occured when set every ItemID

(2)HRESULT GetItemRange([out] long *pItemID__Len, [out] LPWSTR **ppItemID)

(3)HRESULT SetIrPropRange(//Ir means "Item range"
[in] long PropCounts__Len,
[in] long *pPropCounts,
[in] long PropID__Len,
[in] long *pPropID,
[out] HRESULT **ppError)

(4)HRESULT GetIrPropRange(
[out] long * pPropCounts__Len,
[out] long **ppPropCounts,
[out] long * pPropID__Len,
[out] long ** ppPropID)

(5)HRESULT GetIrPrPropDefs(// Ir means "Item range", Pr means "Prop range"
[out] long * pPropValue__Len,
[out] LPWSTR **ppPropDESC,
[out] unsigned short **ppPropDATATYPE,
[out] HRESULT **ppError)

(6)HRESULT GetIrPrPropValues(
[out] long * pPropValue__Len,
[out] VARIANT **ppPropValue,
[out] HRESULT **ppError)

(7)HRESULT SetIrPrPropValues(
[in] long PropValue__Len,
[in] VARIANT *pPropValue,
[out] HRESULT **ppError)

(8)HRESULT GetDCSRunState([out] long *pState)

(9)HRESULT GetDCSUpdateRate([out] long *pRate)

(10)HRESULT SetDCSEvent([in] IUnknown *pIDCSShutDownEvent)

SetItemRange 和 SetIrPropRange 函数的设计是为了更高效地访问数据,利用它们可以将需要读写的点和属性的 ID 范围先设置好,然后用 GetIrPrPropDefs 读取这个范围内的属性数据类型和描述,用 GetIrPrPropValues 和 SetIrPrPropValues 读写这个范围内的属性值,连续的多次读写都基于这个范围,避免了每次都提供读写范围。特别的,SetItemRange(-1,NULL)设置范围为 DCS 的全部点和全部属性。利用 GetItemRange 和 GetIrPropRange 可以获取设置好的范围内的点和属性 ID。

GetDCSRunState 得到 DCS 的运行状态是否正常。

GetDCSUpdateRate 得到 DCS 实时数据的刷新频率。

SetDCSEvent 给 DCS 设置一个事件接口指针,当 DCS 关闭的时候利用这个接口指针给相连的 OPC 服务器发送一个关闭事件。事件接口设计如下:

```
class IDCSShutDownEvent : public IUnknown
{
public:
virtual HRESULT STDMETHODCALLTYPE OnDCSShutDown(LPWSTR Reason) = 0;
};
```

3.2 实现

在实现 IDCSDb 接口的时候,需要解决 DCS 与 OPC 数据结构之间异构的问题。一般而言,DCS 利用组态平台配置数据点及其属性,点可分为固定的几类(如模拟量输入、模拟量输出等),每一类点的属性结构固定,并且映射到编程语言中的类或者结构类型。DCS 数据库在内部实现中一般是结构数组,利用点 ID 来存取点,但对点属性的存取是语言级内置的,并没有像 OPC 那样的属性 ID 标识。由此可见,OPC 的数据结构更加通用,实现 DCS 数据结构向 OPC 数据结构转化要求 DCS 提供更加通用的点属性机制。一个解决方法是不改变现有的 DCS 数据结构,采用静态的或者组态的方式完成二者的转换;另一个方法是改变 DCS 的数据结构,使得二者一致。相比而言,前一个方法对于改造现有的 DCS 更加合适。

OPC 服务器对象的 IOPCServer 接口的 GetStatus 函数要求返回 OPCSERVERSTATUS 结构,其中的一个域是 OPCSERVERSTATE 类型的枚举变量,反映了 OPC 服务器的运行状态。OPC 服务器的运行状态需要参考 DCS 的运行状态,主要是了解 DCS 数据库是否正常运作。

如果 DCS 运行失败,那么,OPC 服务器的状态应该是 OPC_STATUS_FAILED。

由于 DCS 可能通过 IDCSDb 接口连接多个 OPC 服务器,当 DCS 关闭的时候要通知所有已经连接的 OPC 服务器,因此,可以设置一个全局的 IDCSShutDownEvent 事件接口指针的容器,SetDCSEvent 向这个容器中插入事件接口指针,在 DCS 关闭过程中利用这个容器向所有连接的 OPC 服务器发送关闭事件。在 OPC 服务器端实现 IDCSShutDownEvent 接口,需要注意的是,因为 DCOM 采用的是同步调用方式,所以在 OPC 服务器的事件响应中不应该弹出模式窗口导致线程阻塞而影响 DCS 的正常关闭。

3.3 使用

OPC 服务器使用点设备对象来封装和使用 IDCSDb 接口指针,以达到同 DCS 通信的目的,主要是对 DCS 数据库的读写操作。在 OPC 服务器的初始化阶段,用 GetDCSRunState 得到 DCS 的运行状态,用 GetDCSUpdateRate 得到 DCS 实时数据的刷新频率,用 SetDCSEvent 给 DCS 设置 OPC 服务器端实现的 IDCSShutDownEvent 事件接口指针,然后是同 DCS 数据库的通信。下面是 IDCSDb 数据库部分的基本使用过程:

1)首先用 SetItemRange(-1,NULL)设置范围为 DCS 的全部点和全部属性,然后利用 GetItemRange 和 GetIrProPRange 得到 DCS 全部点及其属性 ID。

2)用 GetIrPrPropDefs 得到各个点的属性数据类型和描述。

3)根据已经获得的信息选择感兴趣的点及其属性 ID,并利用 SetItemRange 和 SetIrPropRange 设置新的范围。

4)用 GetIrPrPropValues 和 SetIrPrPropValues 读写这个范围内的属性值。在 OPC 属性中,少数是实时属性,多数是静态属性(在组态期间确定)。对实时属性的读取要比静态属性的读取频繁得多。在 OPC 规定的 6 个属性中,点值(属性 ID 为 2,描述为"Item Value")是最重要的实时属性,经常需要读写。

5)按需要重新设置点及其属性的范围,并读写新范围内的属性值。

4 DCS 数据库扩充接口的设计

为了解决 DCS 与 OPC 点属性的数据结构之间异构的问题，需要 DCS 数据库扩充一些接口函数来完成 DCS 点属性向 OPC 点属性的转换。IDCSDb 接口的实现要依赖于这些函数。下面是扩充接口的设计：

(1)VARTYPE GetItemPropDataType(char* ItemID,long PropID)

(2)char* GetItemPropDescription(char* ItemID,long PropID)

(3)HRESULT GetItemPropValue (char* ItemID,long PropID, VARIANT* PropValue)

(4)HRESULT SetItemPropValue (char* ItemID,long PropID, VARIANT* PropValue)

通过上面这些函数可以按照点 ID 和属性 ID 来得到某个点的某个属性的数据类型和描述，并进行属性值的读写。

5 将 DCS 进程改造成进程外 COM 服务器

OPC 服务器是通过 DCOM 协议同 DCS 通信的，DCSDb 是 COM 对象，DCS 通过 DCOM 向 OPC 服务器进程输出 DCSDb 对象的 IDCSDb 接口指针。一般的 DCS 进程并不支持 DCOM，因此，需要将 DCS 进程改造成进程外 COM 服务器。

微软的 Visual Studio 6 提供了 ATL(活动模板库)和 MIDL(IDL 接口定义语言的编译器）等一整套工具来支持 COM/DCOM 技术，其中 ATL 是开发 COM 组件的首选类库，Visual C++ 6.0 和 C++ Builder 6.0 都支持 ATL。利用 MIDL 可以将接口定义文件(.IDL)转化成 C/C++文件、类型库文件(.TLB)和代理/存根(Proxy/Stub)文件(PS.DLL)，这些都是开发和运行进程外 COM 服务器不可缺少的文件。

将 DCS 进程改造成进程外 COM 服务器的大致过程为：

1)编写 IDCSDb 接口的 IDL 文件。

2)用 MIDL 编译上面的 IDL 文件，生成接口定义的 C/C++文件、TLB 文件和 PS.DLL。

3）在 Visual C++ 6.0 或 C++ Builder 6.0 中用上面生成的 C/C++文件或 TLB 文件为 DCS 工程引入接口的 C/C++类型定义。

4)在 DCS 工程中利用 ATL 类库为 DCSDb 类及其类工厂的实现和注册提供必要的 COM 支持。

5)编译运行 DCS 以注册 DCSDb 组件。

6）用 windows 自带的 regsvr32.exe 工具注册第二步生成的 PS.DLL 以提供使用自定义接口的进程外 COM 服务器所必需的代理/存根文件。

6 结束语

本方案为 DCS 提出了一个较为实用的接口，不但很好地隔离了 DCS 及其 OPC 服务器，而且由于其简单有效，还可用于 DCS 与其他客户程序的相连，以扩展 DCS 的功能。

参考文献

1 陆会明，等.过程控制软件标准接口剖析(2)-OPC 技术的核心:COM[J].现代电力，2002，19(2)：50~55

2 Thuan L. Thai 著，陈逸译.DCOM 入门[M].北京：中国电力出版社，2001

[收稿日期：2004.1.6]