基于.NET Framework 的 OPC 数据访问服务器的研究与设计①

许南山　郭郁峰 (北京化工大学 信息科学与技术学院 北京 100029)

摘　要： 通过介绍 OPC 技术，结合当前 OPC（OLE for Process Control，用于过程控制的对象链接和嵌入）快速发展的需要，在研究.NET 框架和 COM 的关系后，给出了一个.NET 框架下开发进程外 OPC 数据存取服务器框架的设计方案，并介绍了实现的主要步骤和关键技术，以及具体的实际应用案例，对 OPC 开发人员有一定的借鉴意义。

关键词： OPC 规范; .NET 框架;C#语言;进程外服务器

Research and Design of OPC Data Access Server Based on .NET Framework

XU Nan-Shang, GUO Yu-Feng

(College of Information Science & Technology, Beijing University of Chemical Technology, Beijing 100029, China)

Abstract: This paper introduces the OPC technology in connection with the need for rapid development of OPC, and studies the similarities and differences between the .NET Framework and the COM. It demonstrates a .NET Framework-based framework design for developing out-of-process OPC Data Access server, introduces the realization approach and key technologies and the practical application of case, which can be used as reference for OPC development.

Keywords: OPC specification; .NET framework; C#; out-of-process server

随着计算机技术不断发展，工业自动化中计算机应用已经成为不可或缺的一部分，然而许多厂商的过程控制装置和数据采集设备都采用不同的通信协议，导致软件商必须开发不同的硬件驱动程序来采集现场数据，这大大增加了企业的开发难度。在这种情况下，OPC 技术应运而生，已成为工业控制软件公认的标准。

1　OPC技术概述

OPC 技术基于 OLE/COM，为了解决工业客户机与各种设备驱动程序间互相通讯而产生的一套工业技术规范和标准。OPC 规范由 OPC 基金会(OPC Foundation)制定，内容包括: 数据访问规范-(Data Access)、报警事件规范(Alarm&Event)、历史数据存储规范(Historical Data Access)、批量过程规范(Batch)、安全性规范(Security)等[1]。每一种规范对应一种服务器，其中数据访问服务器功能是向下对现场设备采集数据，向上与 OPC 客户端完成数据传输，如图 1 所示，OPC 服务器封装了设备的驱动程序，客户开发人员只能看到 OPC 服务器提供的统一接口，所以只需要开发符合 OPC 接口规范的应用程序，就能和 OPC 服务器连接，获取现场设备的数据[2]。

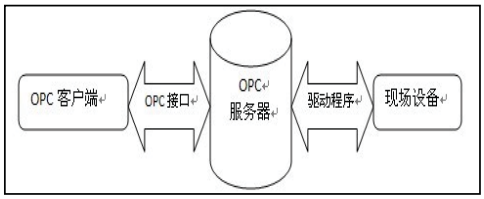

图 1　OPC 客户端/服务器关系图

2　.NET框架与COM的关系

.NET Framework 是一个多语言的开发和执行环

① 收稿时间:2009-11-09;收到修改稿时间:2010-01-01

境，它由 CLR(Common Language Runtime，公共语言运行库)，统一的编程类和活动服务器页面(ASP+)三个主要部分组成。以往开发 COM 组件方法主要有三种：MFC(Microsoft Foundation Classes，微软基础类)、ATL(Active Template Library，活动模板库)和第三方开发工具。在.NET Framework 下开发 COM 组件则和以往不同，它可以让 COM 开发变得比以往容易，它实际上已自动处理了与 COM 息息相关的规则，具体包括以下几点：

(1) COM 对象必须要自己管理生存期，而.NET 对象生存期由 CLR 来管理；

(2) 客户通过调用 QueryInterface 来查询 COM 对象是否支持某一个接口，而.NET 对象则是用 System.Reflection 命名空间的类和方法来获得对象功能的描述；

(3) COM 对象通过指针来引用，而.NET 环境中不使用指针，对象的位置则是可变的。

3 基于.NET Framework的OPC数据访问服务器的设计

.NET Framework 是微软的应用软件开发平台，已得到了广泛的应用。本文通过使用.NET Framework 的 C#语言设计一个进程外的 OPC 服务器框架，来介绍设计过程中需要解决的主要问题。

3.1 OPC 服务器的对象与接口

一个 OPC 服务器即一个 COM 组件，一个组件可以包含多个 COM 对象。如图 2，OPC 服务器有三个层次，Server 对象、Group 对象和 Item 对象，其中前两个是 COM 对象，Server 对象用来提供服务器对象自身的相关信息，并且作为 Group 的容器管理 Group 对象；Group 对象则用来提供关于自身的相关信息，并提供机制来组织和管理 Item 对象。Item 对象是服务器端定义的，不允许客户端直接访问，它代表了服务器到现场设备数据源的一个物理连接。

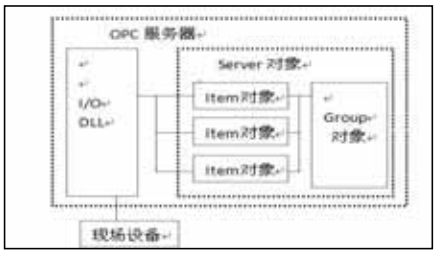

图 2 OPC 服务器整体结构

根据 OPC2.0 规范，开发 OPC 服务器必须实现 Server 对象和 Group 对象以及他们各自的接口，Server 对象包括：

IOPCServer,IOPCCommon,
IConnectionPointContainer,
IOPCItemProperties；

Group 对象包括：

IOPCItemMgt,IOPCGroupStateMgt,
IOPCSyncIO,IOPCAsyncIO2 ；

.NET Framework 不需要实现 IUnKnown 接口，因为它没有指针引用，在建立对象或释放对象时会自动检测并且释放空间。使用 C#语言开发时，由于编译环境等各方面的不同，接口定义也有相应的变化，以 IOPCServer 接口举例如下：

```
[Guid("39C13A4D-011E-11D0-9675-0020AFD8ADB3")]
[InterfaceType(ComInterfaceType.InterfaceIsDual)]
public interface IOPCServer
{void AddGroup
        (string  szName,
         int     bActive,
         int     dwRequestedUpdateRate,
         int     hClientGroup,
 IntPtr   pTimeBias,
         IntPtr  pPercentDeadband,
         int     dwLCID,
         out int  phServerGroup,
         out int  pRevisedUpdateRate,
         ref Guid  riid,
         out object ppUnk) }
```

根据定义实现所有的接口方法，每一个接口对应一个接口类，Server 对象和 Group 对象分别对应着 Server 类和 Group 类，包含了各自的接口类，这种方法让接口相对独立，方便开发者测试。

3.2 数据存取方式

在 OPC 服务器中，Cache 是重要的部分，存储结构将直接影响数据访问的速度，它负责将从设备中读取的数据先存放在数据缓存区，供同步或异步读取[3]。设计出良好的 Cache 对服务器的数据更新速度至关重要。Cache 构造方式有三种，包括连续存储、链式存储和哈希存储。考虑到.NET Framework 有独立的

哈希表类(Hashtable Class)可以直接使用，并且可以大大降低数据的存储和查找消耗的时间。

Hashtable 是 System.Collections 命名空间提供的一个容器，用于处理和表现类似 key/value 的键值对，其中 key 通常可用来快速查找，同时 key 又区分大小写；value 用于存储对应于 key 的值。Hashtable 中 key/value 键值对均为 object 类型，所以 Hashtable 可以支持任何类型的 key/value 键值对。Hashtable 包含了以下基本操作：

在哈希表中添加一个 key/value 键值对：Hashtable Object.Add(key,value);

在哈希表中去除某个 key/value 键值对：Hashtable Object.Remove(key);

从哈希表中移除所有元素：

HashtableObject.Clear();

判断哈希表是否包含特定键 key：

HashtableObject.Contains(key);

使用以上基本操作可以建立一个稳定而快速的 Cache，是 OPC 服务器的开发过程中一个关键步骤。

3.3 Marshal 类方法的调用

开发过程中需要使用.NET Framework 的基础类库，System.Runtime.InteropServices 这个名称空间提供了一系列的类来对 COM 对象进行操作,其中最主要的就是 Marshal 类。Marshal 类支持从托管内存空间复制数据到非托管内存空间或是从非托管内存空间到托管内存空间，它包括了很多实用方法，重要的有以下几个：

AllocCoTaskMem()—从 Com 任务内存分配器分配指定大小的内存块；

FreeCoTaskMem()—释放由 AllocCoTaskMem()方法分配的内存块；

ReleaseComObject()—递减所提供的运行库可调用包装的引用计数；

DestroyStructure()—释放指定的非托管内存块所指向的所有子结构；

StructureToPtr()—将数据从托管对象封送到非托管内存块。

3.4 注册与注销服务器

在一般情况下，.NET Framework 使用 Regasm.exe 来注册进程内(dll 类型)的组件，对于进程外组件(exe 类型)并没有特殊的工具，也必须通过 Regasm.exe，不过我们需要更改一下注册逻辑。

首先找到服务器所在的目录，使用 Regasm.exe 语句将 exe 类型服务器注册，此时注册表内出现“InprocServer32”子键，我们需要更改为“LocalServer32”，就必须在注册后调用一个方法，修改服务器所在的注册表，这里使用.NET 的基础类库 Registry 类和 RegistryKey 类。以上两步骤可以封装在一个 BAT 文件中，在注册服务器时自动完成，操作起来十分简便。注销服务器时只需使用 Regasm /unregister 语句来完成即可。

3.5 OPC 客户端与 OPC 服务器的连接过程

OPC 客户端枚举本地电脑的所有 OPC 服务器并连接，OPC 服务器在.NET 下必须实现 COM 对象的自动注册才能被客户端连接，原因在于进程外服务器需要在启动时主动向 SCM(Service Control Manager)中注册 COM Class Object，这样 SCM 才能创建出对应的 COM 对象返回给客户端。在这里我们选择使用基础类库里的 RegistrationServices 类来完成对象的注册。

RegistrationServices 类有两个方法，RegisterTypeForComClients 和 UnregisterTypeForComClients, 在服务器启动时，使用 RegisterTypeForComClients 方法来注册 Server 对象，以便客户端可以连接 Server 对象进行操作；在客户端操作结束要退出服务器时，服务器使用 UnregisterTypeForComClients 方法自动注销注册的对象。

3.6 OPC 数据访问服务器的应用

在山西某化工厂内包括 TDI、造气、净化和锅炉 4 个车间，通过监测系统自动从车间实时采集生产过程状态参数(温度、压力等)，使厂领导能在线监视生产运行状况。监测系统其中的数据采集子系统使用了 OPC 协议，通过新开发的 OPC 服务器代替之前的 OPC 服务器与山武 DCS 连接获取数据，采集过程如图 3。

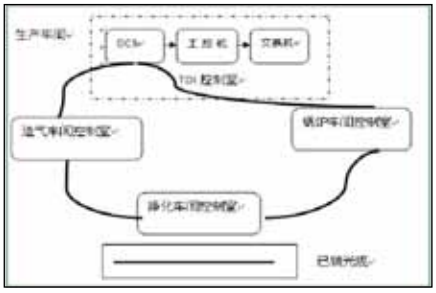

图 3　某化工厂系统结构图

工厂的每个车间都配置有工控机，首先在工控机上安装 Microsoft Visual Studio 2005/2008 平台，之后安装 OPC 服务器和 OPC 客户端，使用本地连接的方法，通过客户端找到服务器并且连接获取底层数据，存储于数据库中。该系统在半年时间的运行测试中表现稳定，说明了该 OPC 服务器的开发过程是值得借鉴的。

OPC 规范在工控领域的广泛使用，使其成为了开发者的焦点，随着.NET 的影响力不断扩大，在这个平台上开发 OPC 服务器将会变得十分常见，.NET 和 OPC 的结合将会越来越紧密，因此，对 OPC 开发人员来说，了解以至熟练使用.NET 平台下的开发手段是十分重要的，本文对该平台下的开发方法进行了探讨，并且给出了实际应用的案例。

参考文献

1 OPC Foundation. OPC Data Access Custom Interface Standard 2.05. 2002-06-28.

2 熊望枝,焦青松.OPC 数据采集服务器的研究与设计.微计算机信息(测控自动化). 2007,23:24－25.

3 马亮,张志鸿.OPC DA 服务器的设计与实现.微计算机信息(测控自动化). 2008,24:228－230.

4 OPC Foundation. OPC .NET API Overview 1.00. 2003-12-01.