.NET Remoting 中多线程实时控制中间件研究

The research of multi－threads real－time controlling middleware in the .Net Remoting

周相兵　杨兴江　（阿坝师范高等专科学校教务处 四川汶川 62300）
兰青青　（四川师范大学计算机科学学院 四川成都 610066）

摘要：就现阶段实时控制系统在各领域的发展，通过在.NET Remoting 体系结构中采用多线程机制来构建分布式实时控制软件系统，并对网络环境中的对象流实时控制处理事务和交互处理数据的中间件进行研究，以及多线程在实时控制软件系统中应用进行分析。最后在.NET Remoting 机制下给出分布式实时控制软件系统的应用方法，并进行相关的性能分析。

关键词：.NET Remoting　中间件　实时控制　多线程　软件系统

随着计算机实时控制软件系统在网络应用领域迅速发展。因此通过在.NET Remoting 体系结构下运用多线程机制进行分布式软件系统中的多任务实时控制研究，并建立实时控制中间件来提高计算机网络环境下实时系统的运算和处理事务的效率，如并行运算、实时通信、实时请求/响应、实时交互、嵌入式实时容错处理等。在.NET Remoting 未出现前，DCOM (Distributed Component Object Model)机制是实时控制系统的主要技术之一，它使用 Winsock 技术完成客户端与服务器应用程序之间通信。而在.NET Remoting 体系结构中用(代理客户)对象与远程对象通过有效的信道进行处理对象流，同时让多任务的对象流在中间件中的调度机制、中断机制控制下进行高效的分配和利用资源。

1　.NET Remoting 原理概述

现今许多软件系统都不是独立运行在一个操作系统上的应用程序，但都可以使用.NET Remoting 网络通信技术调用远程服务器上的应用程序域对象。在.NET Remoting 下这些对象可以直接封装数据处理协议、IP 地址、连接端口以及软件接口等，通过调用方法完成数据实时控制传输。

在.NET Remoting 下应用程序域是公共语言运行库的隔离单元，它们是在进程内创建并运行的。.NET Remoting 提供了一种允许对象通过应用程序域与另一对象进行交互的框架。这种框架提供了多种服务，包括激活和生存期支持，以及负责与远程应用程序进行消息传输的通讯通道。同时也是应用程序域之间管理跨应用程序域的同步和异步远程过程调用(RPC)会话首选技术。在同一应用程序域中的对象可以直接进行交互，但是在访问不同应用程序域中的对象时，必须使用代理对象。其将完成所有格式化数据封送和拆收、安全控制、传输通道配置和任何其他附加工作。此时在客户端可使用激活器创建远程对象或获取一个被服务器激活的对象的代理对象。

格式化程序用于在对象流通过通道传输接收之前，而通道在应用程序之间跨越远程处理边界传输对象流，无论这种边界是应用程序域之间、进程之间还是计算机之间的边界，对对象流进行编码和解码，同时.NET Remoting 体系结构适用于在.NET 框架的 CRL 中建立各种网络结构的实时控制系统，这是因为在.NET 框架下允许客户端与远程应用程序域和进程或远程计算机所承载的对象进行通信。

在.NETRemoting 下提供了基于 HTTP 信道的 SOAP 协议的 XML 编码的 WSDL 和基于 TCP 信道的二进制编码方式格式化程序，而且这两种信道可相互使用 SOAP 协议的 XML 编码和二进制编码进行网络通信，传输到目标 URI(Universal Resource Iden-

tifier),使得了.NET Remoting是具有很大灵活性。这样就可以让控制台应用程序、Windows应用程序、ASP.NET辅助进程、Windows服务或COM+组件、P2P等通信中使用.NET Remoting。

2 .NET Remoting下多线程实时控制系统原理

2.1 系统构建原理

Windows NT以后的版本是一种以消息机制的非抢占式多任务嵌入式操作系统,而在.NET框架下的.NET Remoting为管理远程对象的生存期提供了功能强大的机制,因此在.NET Remoting体系结构下构建多线程实时控制的分布式软件系统时,建立一个实时控制中间件来解决实时控制的多任务运行是具有价值的。在中间件中以多线程机制确定不同的控制器和定时器来解决远程对象的实时控制以及使用多线程同步和异步方式来控制管理远程的对象实时控制操作。在多线程机制下的控制器和定时器为多任务在.NET Remoting下顺利完成实时控制提供了重要解决方法;同时多线程同步和异步是提高多任务实时控制软件系统效率的关键,也是网络环境下多用户协作、多用户操作、实时控制完成事务的重要解决方案。在.NET框架下同步对象包括CSyncObject、CSemaphore、CMutex、CCriticalSection和Cevent;同步访问对象包括CMultiLock和CsingleLock。在异步控制中,线程池是解决线程异步的主要方法,线程池是用来在后台执行多个任务的线程集合,使得主线程可以自由地异步执行其他任务,通常一个线程池包括:①线程池空间大小,②线程池内的工作线程,③等待处理的消息队列,④实现调度任务的接口,⑤通信软件接口等部分组成。

在.NET Remoting中的CLR Object Remoting可以处理对象激活、分布式验证、生存期、调用多线程控制的对象流等,因此是.NET Remoting一个重要方面。同时,一个进程可以有多个应用程序域,而一个应用程序域又可以有不同的环境,环境用于把具有相似的执行要求的对象组合在一起。环境由一组属性组合而成,它用于线程同异步、实时控制、事务处理和安全管理等。在环境之间通信时,客户端使用代理对象代替真实对象,代理对象创建的消息传输到信道中来完成多线程实时控制工作,同样这种机制也可以用于不同的软件系统中不同程序域的不同信道中。在.NET Remoting下也提供了异步远程调用,这为建立实时控制中间去执行远程对象调用提供返回到给客户端提供了依据。

在.NET Remoting下的多线程中建立一个中间件来封装实时控制所需的部件,其直接提供一组可调用的接口给外部程序,可以减少多种烦琐的操作,如很方便的实行系统数据交换等。建立一个实时控制中间件,其可以由一个元组组成:

主要部件分别是:远程操作系统识别、服务接口、服务和客户代理、多线程机制、控制器、定时器、数据源封装、实时控制上下文、.NET Remoting通道和编码机制、实时控制代理程序、系统配置等部分组成。

(1) 远程操作系统识别。确定用.NET Remoting体系结构来构建分布式实时控制软件系统,识别相关客户端的操作系统是否都是基于.NET框架。

(2) 服务接口(软件通信接口)。使服务器与客户端建立正确的通信连接方式,通过网络将方法调用及其参数发送到服务器,然后将响应发回客户端的所有传送工作。

(3) 服务和客户代理。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。

(4) 多线程机制。是实现程序的序列单元,是如何确定实时控制系统运行的重要解决方式,如选择合理的多线程同步对象和建立相应的线程池以及相关的软件通信接口等,同时设置多线程的优先级以及与线程关联的稳定标识,为线程调度机制和中断机制提供操作的句柄参数。

```
Thread thread = new Thread (new ThreadStart ());
```

(5) 控制器。一般控制器由硬件来完成,本文控制器是使服务器与客户端之间的对象流稳定的完成事务、事件处理以及页面相关的页面控制。同时控制相应的中断标识符,当某一客户在规定的定时器范围内没有响应,应该采用中断机制暂时中断此客户端。

(6) 定时器。是在服务器与客户端之间的对象流实时完成事务、事件的处理,同时也是提高多线程处理事务、事件的吞吐率和平均响应时间以及时延控制重要方法。在.NET下提供了Timer组件来建立定时

器，Timer 是基于服务器的。

(7) 数据源封装。提供对信息格式、数据库连接、数据库连接池、客户端连接和服务器操作以及相关异构数据连接接口的方法，有时在其内部可以为分布式逻辑层和代理层来实现分布式交互计算等方法，如可使用 DCOM、.NET Remoting 等组件来实现。其中数据库连接池是用来管理和优化异构数据库连接的机制，这样就可以方便地控制程序中数据库连接数目和使用情况。在.NET 框架为 ODBC、Oracle、Ole、SQLServer 等提供了数据连接程序。

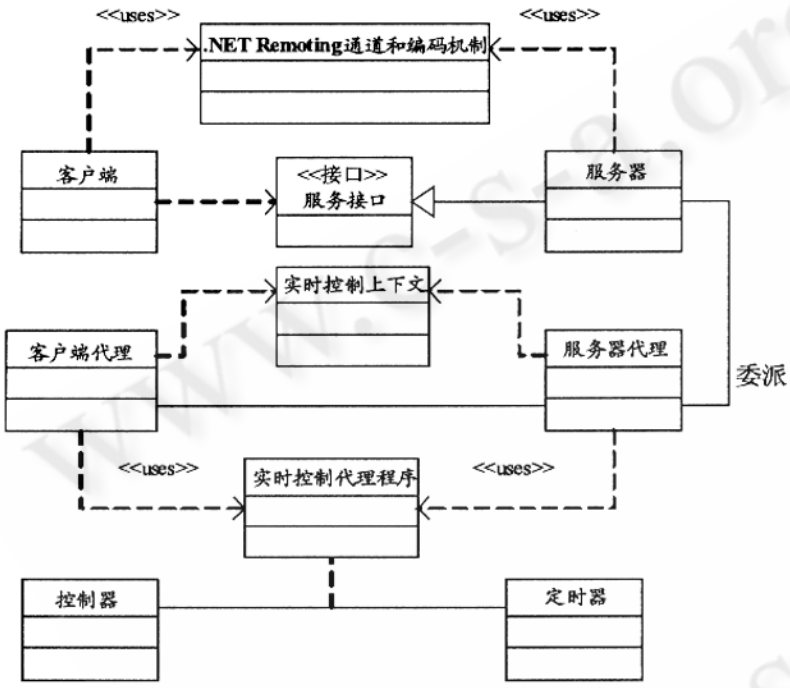

图 1　实时控制系统中间件的 UML 图

(8) 实时控制上下文。它是包含共享公共运行库属性的对象的边界，可以绑定到上下文的类称为上下文绑定类，是从 System.ContextBoundObject 派生的。以及.NET 远程处理技术用元数据动态创建代理对象，使对象流间保持一致。

(9) .NET Remoting 通道和编码机制。建立服务器与客户端的进程启动时的应用程序域、激活方式、通道、端口、状态管理等。因此是通过 System.MarshalByRefObject 继承的.NET 服务器对象 (DLL)，通过 Remoting.Channels 选择远程(本地)计算机对象流实时控制的传输信道和适当选择用 System.Runtime.Serialization.Formatters 创建格式化编码编码方式，通过 RemotingConfiguration.Configure 加载服务器对象的服务器加载器程序。以及通过 Activator.GetObject 访问服务器对象的客户端程序，等等。来完成在.NET Remoting 体系结构下的实时控制系统之间的通信。如下是实现二进序列化的代码方法：

```
BinaryFormatterObject bfo = new BinaryFormatterObject(); //需要序列化对象
……
BinaryFormatter bf = new BinaryFormatter();
    MemoryStream ms = new MemoryStream();
    bf.Serialize(ms,bfo);
    ……
```

(10) 实时控制代理程序。实时控制系统中的一组实现多任务、多事务的程序集。

(11) 系统配置。是完成服务器与客户端进行正确部署以及通信进行设置的一组 XML 文件。

其组成实时控制系统的 UML 图如图 1 所示。

2.2　系统构建结构

系统构建结构如图 2 所示

在图 2 中，是在.NET Remoting 体系结构中构成分布式实时控制软件系统基本原理图，其激活方式有基于服务器的 Singleton/SingCall 和客户端的 new 或 Activator.CreateInstance() 激活方式，使用 RemotingServices 注册。

当某客户端时延超过定时器的范围时，就调用中断机制来暂时中断该客户端，以确保系统的资源得到合理的分配、释放资源和完成实时控制的要求，中断根据不同请求方式可以由操作系统来完成，也可以由实时控制中间件来完成。

在进行多线程实时控制完成任务处理时，在服务器与客户端之间要发生不同的加载时延，加载时延直接与多线程调度算法相关(如线程优先级)，与选择何种信道与编码方式进行交互相关。在 Windows 操作系统下的多线程机制中最常用的方法是采用线程优先级由操作系统自动完成调度，线程都是由操作系统分配处理器时间片的，而时间片的长度取决于操作系统

和处理器。在.NET 框架下可以使用 Thread.Priority 属性获取或设置任何线程的优先级，优先级高 CPU 就先执行。当基于中间件的应用程序的用户界面在前台和后台之间移动时，操作系统还可以动态调整线程优先级。同时其他操作系统可以选择使用不同的调度算法。在实际分布式软件系统开发过程中可以建立相

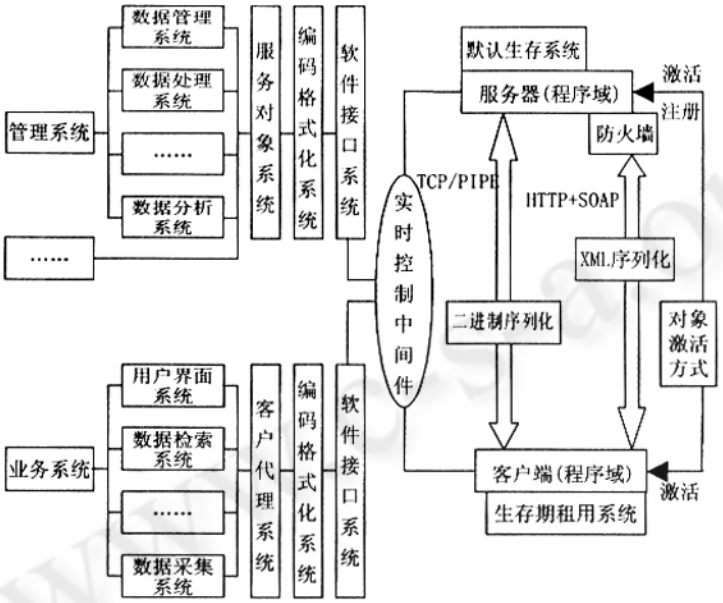

图 2　.NET Remoting 下多线程实时控制中间个在软件系统中应用原理

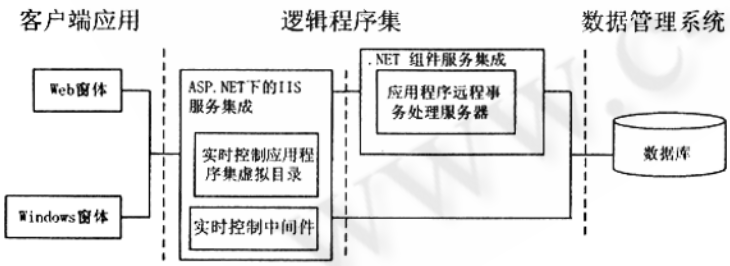

图 3　.NET Remoting 分布式实时软件系统应用部署图

应的消息队列等结构来完成多客户端远程调用(RPC. NET Remoting 应用程序域集成在 Aspnet_wp.)对象流完成实时控制事务。相应的每个线程都维护异常处理程序、调度优先级和一组系统用于在调度该线程前保存线程上下文的结构。因此多线程特别适用于需要不同资源的任务处理。

3　应用方法与性能分析

3.1　应用方法

一般.NET Remoting 采用集成方式来运行系统：①.NET Remoting 对象集成在任何常规的 .NET EXE 或托管服务中，②.NET Remoting 对象可以在 ASP.NET 下的 IIS 集成，③.NET Remoting 对象可以集成在.NET 的 COM+ 组件服务基础结构中，从而利用各种 COM+ 服务，例如：事务、JIT、对象池等。在本文中主要分析后两种集成方式，应用结构如图 3 所示。

exe 辅助进程中进行实时控制系统工作。使用 HTTP 信道的 SOAP 格式化程序时.NET Remoting 支持 WSDL，使用 Soapsuds.exe 可获得元数据。.NET Remoting 没有自己的安全模式，身份验证和授权是由通道和主机进程执行的，在这种情况下完全执行 IIS 的安全功能。当通过.NET 组件服务 COM+ 时，为了使远程组件参与到 COM+ 环境中，并在 COM+ 的上下文中运行，需要从 ServicedComponent 中继承。ServicedComponent 和 System.EnterpriseServices 命名空间中提供的其他功能都允许 CLR 组件指定多个 COM+ 属性，如表示事务要求和服务器进程执行的属性等。

3.2　性能分析

.NET Remoting 的性能是相对的，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。一般地：采用二进制编码 TCP 信道传输性能高，但安全性不是很好，采用 SOAP 的 XML 编码 HTTP 信道传输性能降低，但安全性增强。由于采用不同数据访问方式、系统体系结构和客户请求数量的不同，就导致执行的效率也不同。在性能分析主要从 WS_TCP_ Binary、WS_TCP_SOAP、IIS_HTTP_ Binary、IIS_HTTP_SOAP、WS_HTTP_ Binary、WS_HTTP_SOAP 集成执行效率来确定.NET Remoting 的性能，其中 WS 表示.NET Remoting 集成在.NET 组件服务中，IIS 表示做成在 ASP.NET 下的 IIS 中。

(下转第 44 页)

(上接第 40 页)

4 总结

本文通过.NET Remoting 中使用多线程机制的实时控制系统进行了深入、仔细的分析,同时详细研究了构架实时控制系统的构建结构,并建立中间件来解决实时控制交换数据,以及怎样在.NET Remoting 体系结构下建立多线程机制的实时控制软件系统应用框架图,并做了相应分析。其可以运用到各类实时控制系统、企业内业绩、工资、销售相关的实时控制系统中等。在实际开发中多线程用来处理不同任务,但不是线程越多越好,因为大量使用线程会使资源难控制和资源共享等问题,应尽量控制线程使用数量。

参考文献

1 Jim Beveridge&Robert Wiener 著. 侯捷译. Win32 多线程程序设计. 武汉:华中科技大学出版社.2002.1.

2 芦 帅、张建明、王树青,基于网络的实时控制系统理论研究进展,信息与控制,2004.10:(573－578).

3 牛秦洲、张烈平、叶恒舟,实时网络的分布式调度. 通信学报,2005.7(44－48).

4 李玉凯、朱有产、秦金磊, .NET Remoting 及其在 SCADA 主站系统中的应用,计算机工程,2006.10:(245－247).