

基于 OGS.NET 的网格计算研究

胡明生^{1,2}, 陈学广¹

(1. 华中科技大学 控制科学与工程系, 湖北 武汉 430074;

2. 郑州师范高等专科学校, 河南 郑州 450044)

摘要:文中主要研究在 .NET 框架中的 OGS.NET 网格计算。分析了 OGS.NET 的体系结构和主要组件, 讨论了 OGS.NET 的服务调用过程、安全性和基于属性的网格计算编程。为网格计算的研究提供了一个新的途径。

关键词:OGS.NET; 网格计算; 服务调用

中图分类号:TP316.3; TP393.09

文献标识码:A

文章编号:1005-3751(2006)04-0007-04

Research of Grid Computing Based on OGS.NET

HU Ming-sheng^{1,2}, CHEN Xue-guang¹

(1. Dept. of Control Sci. and Eng., Huazhong Univ. of Sci. and Tech., Wuhan 430074, China;

2. Zhengzhou Teachers College, Zhengzhou 450044, China)

Abstract: This paper is about OGS.NET grid computing based on the .NET framework. It analyses the architecture and major components of OGS.NET, discusses service function invocation and security of OGS.NET and attribute-based programming for grid computing. It gives us a new way for research of grid computing.

Key words: OGS.NET; grid computing; service function invocation

0 引言

网格计算常常被人们认为是互联网之后最重要的技术, 是针对复杂计算的新型计算模式^[1]。网格计算的架构是由 OGSA (Open Grid Services Architecture) 定义的。OGSA 结合或扩充了网格和 Web Services 技术, 定义了什么是网格服务, 统一的结构和网格环境可提供的服务。OGSI (Open Grid Services Infrastructure) 是 OGSA 提出的正式的概念规范, OGSI 制订了一组适用于所有网格服务的术语。目前, 最著名的网格计算研究是美国的 Globus 项目。大多数的网格计算开发是基于 Globus 工具包。这些开发过程集中于 Java 和 J2SE/J2EE^[2]。在 Microsoft .NET 框架中 OGS 参考实现比较少。文中将研究 OGS.NET 的体系结构、在 .NET 平台上的网格服务调用以及编程模型。

1 体系结构

OGS.NET 是基于 OGS 标准的开放式网格计算框架。OGS.NET 的目标是尽可能地利用 Microsoft .NET 核心的技术去实现 OGS。 .NET 框架的概念集中于服务

实例对客户调用之间管理的需求, 以及使用 Internet 信息服务器 (IIS) 将客户请求分派到实际实例的需求^[3]。尽管 .NET 提供了一个丰富的 Web 服务功能集, 但是网格服务的基本需求在某些情况下并不显著。而 OGS.NET 的体系结构提供了用于维护客户调用之间的服务实例, 以及将请求分派给正确服务实例的能力。

在 OGS.NET 的体系结构中, 有一个实体容器容纳了全部的运行在同一台主机上的服务实例。这个容器的进程是由一个应用程序域 (AppDomains) 的集合构成的。 .NET 的这个应用程序域机制保证了进程内部的内存保护, 即使一个 AppDomain 崩溃了, 也完全不会影响到其他的 AppDomain。除了每一个服务实例运行在它们的应用程序域中外, 还有一个 AppDomain 负责容器的逻辑 (比如分派、消息的处理等等功能)。在这个 AppDomain 中的对象可称为分派器 (Dispatcher)。每个服务都会被放在它自己独立的 AppDomain 中。这是因为, 假如被同一个 factory (OGSA 中定义的创建网格服务的接口) 创建的服务都存在于同一个 AppDomain 中, 这样的方式在服务间进行通信显得很有效率。

当一个客户端向 OGS.NET 发送请求的时候, 它实际上是向 IIS 服务器发送信息。OGS.NET 使用了 IS-API 过滤器在请求信息进入 IIS 请求链的前期就把它拦截并重写了请求, 并将请求分派给 OGS.NET 的 ASP.NET HttpHandler。这个 HttpHandler 将消息路由给

收稿日期: 2005-07-14

基金项目: 教育部博士点基金资助项目 (20040487076)

作者简介: 胡明生 (1973-), 男, 河南郑州人, 博士研究生, 研究方向为网格计算与决策支持系统; 陈学广, 教授, 博导, 研究方向为网格技术与决策支持系统。

OGSI.NET 的容器进程。容器进程拥有一个线程池,每一个 IIS 请求都会引起一个线程起执行分派器。分派器会决定将消息路由给 GridServiceWrapper 类,用于将来的处理^[4]。

在 AppDomain 里面,控制被转移到一个叫做网格服务包装器(Grid Service Wrapper, GSW)的对象中。一个 GSW 包括了一个服务的实例,包括它的方法实现和它的服务数据。GSW 维护着服务所支持的 portType 的一个列表(包括 OGSI 指定的由 OGSI.NET 提供的 portType,以及自定义的 portType。比如 GridService portType 和 NotificationSouce portType 等),另外还序列化器和逆序列化器,这两个是为了满足服务支持多种消息协议(比如 SOAP 或 .NET Remoting)的需要而存在的。GSW 执行对每一个特定消息的处理,并逆序列化请求信息,从中得到要调用的方法的名字以及参数,通过请求中的参数还有特定的消息数据(比如 SOAP 标头数据)来调用方法。当调用返回数据时,GSW 再将这些数据序列化成一个二进制 array 并传回给分派器,分派器再将这个 array 传给 IIS 并返回给客户。

图 1 为 OGSI.NET 的体系结构和请求流程。

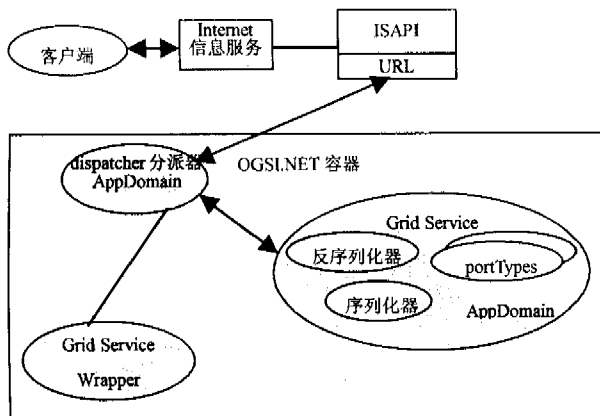


图 1 OGSI.NET 体系结构和请求流程

2 主要组件

OGSI.NET 包含分派器、网格服务包装器、工厂和消息处理器 4 个主要组件。

2.1 分派器(Dispatcher)

分派器是在客户端请求与服务实例之间的一个接口,它的主要功能是将客户端的请求消息路由到适当的服务实例,并不经过处理将结果返回给客户端。分派器保存了从 GSR(Grid Service Reference)到在容器中的 AppDomain 的映射列表。原始的请求被分派到 AppDomain 的 GSW 中^[5],GSW 会执行请求的工作并将一串字节流返回给分派器,分派器再返回给客户端分派器。GSW 能够帮助处理多个并发的请求访问容器,能令请求各自独立,互不干扰。这个体系机构有助于隔离安

全性和其他的运行时问题,提供服务实例 AppDomain 的隔离级别。

2.2 网格服务包装器(GSW, Grid Service Wrapper)

GSW 包装了网格服务实例的各种功能单元。每一个在容器中的 AppDomain(分派器的 AppDomain 除外)都有一个 GSW,每一个 GSW 也包装了一个网格服务实例^[6]。

GSW 提供的功能有:

- 1) 可插入的、服务特定的消息处理程序,用于在向服务实例进行分派之前的处理消息。
- 2) 为分派器提供了一个访问任何一个网格服务实例的接口。
- 3) 为系统定义的 portType 提供可插入的实现。
- 4) 内置的用于支持使用 WSE 来扩展标准的 SOAP 头。

网格服务的作者可以用 .NET 的特性(Attribute)去修饰他们的服务代码,而这些特性也在运行的时候提供给容器有关服务的信息。当 GSW 在一个 AppDomain 中被实例化的时候,它将含有对应的服务的程序集载入。配置文件会告诉 GSW 在容器中注册的哪种序列化/逆序列化的模型是适用于该服务的,服务的类上的特性声明了被服务所支持的 portType。GSW 指明了所有服务的 portType 的实例,包括由服务作者写的以及被 OGSI 规范定义并包含在 OGSI.NET 中的。服务需要创建更多的线程,可以使用 .NET 提供的 System.Threading.Thread 类,在 AppDomain 中创建 .NET 提供的软线程。

2.3 工厂(Factories)

工厂是一种服务实例,可以在不同的 AppDomain 中创建其他的服务实例。图 2 说明了一个网格服务实例的基本服务创建模型^[6]。一个工厂服务使用公开的实例名存储着一个对这个新的 AppDomain 中的 GSW 引用(GSR),这个映射也会被传送到分派器。对象的引用可以看成是一个包装,并可跨应用程序域传送,新的服务实例的引用就是这样从工厂的域传送到分派器的域。在容器启动的时候,容器的配置文件会指定由工厂创建的实例的类型。它们利用 createService 操作实现 OGSI 工厂的 portType。这些服务实例包装器是类型 MarshallByRefObject 的远程可引用对象,存储在分派器的 AppDomain 中。分派器用表存储这些对象,将服务实例句柄(GSI)作为主键。

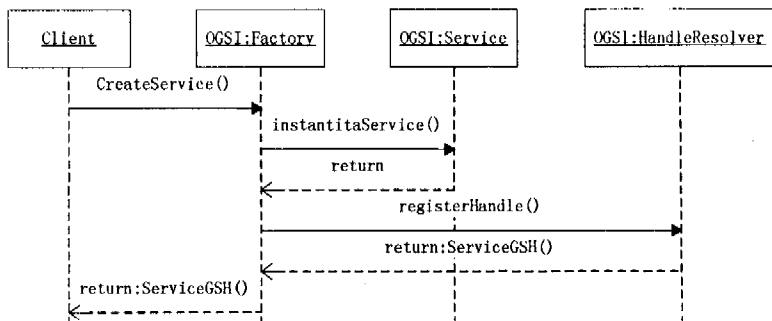


图 2 网络服务的工厂使用模型

2.4 消息处理器

OGS.NET 提供了两个消息处理器,一个用来处理 SOAP 消息;另一个用来处理 .NET 的远程(Remoting)消息格式^[6]。当一个服务实例被创建的时候,它的 GSW 也会为服务所支持的每种消息格式创建消息处理器。当请求信息以字节数组的形式到达分派器的时候,消息处理器会将它反序列化,这个过程包括创建每一个所需要的参数对象及处理消息头。当请求完成后,或者一个异常被抛出的时候,处理器会序列化这些结果,形成一个字节流。然后交给分派器,并返回给客户端。通过这种方式,分派器处理传输协议,而消息处理器处理消息的协议。

3 服务调用过程

每一个网格服务的实例都提供了一些可以让客户端调用的函数。在一个调用的服务处理中,IIS 发送一个请求到分派器,分派器发送一个请求到正确 GSW。GSW 负责执行正确的服务实例的调用,并返回结果。下面将讨论处理客户请求的过程。

当一个请求到达 IIS WEB 服务器的时候,会发生以下的步骤:

- 1) ISAPI 过滤器:一个 ISAPI 过滤器“重写”请求的目的 URI (统一资源标志符,Uniform Resource Identifier),这样请求就可以被 OGS.NET 的托管代码中的 HttpHandler 所处理。IIS 根据所请求的页面的扩展名调度请求,ISAPI 过滤器通过“重写”请求 URI 将请求调度给 ASP.NET 基础架构。

- 2) HttpHandler 路由:HttpHandler 将消息调度出 IIS/ASP.NET 系统,送往 OGS.NET 容器,请求消息被送往容器中的分派器。

- 3) 分派器寻找网格服务:分派器通过寻找分派器的服务列表中的 URI 寻找合适的 GSW。然后,分派器得到 GSW 的句柄,并称它为 ProcessRequest 方法,这个方法以原始的消息作为参数。

- 4) 处理消息头:GSW 的消息处理器处理原始的消息的消息头,假如是个 SOAP 消息,那么它将会通过 WSE (Web Services Enhancements)的流水线进行处理。容器会使用消息头的信息来检查调用是否符合服务的安全策略。

- 5) 确定方法的名称和参数:消息处理器逆序列化消息体,取得客户端所要调用的方法的名称,以及为所调用的参数进行编码。

- 6) 为方法名找到方法句柄:GSW 从它 portType 数组中找到实现了所指定的方法的 portType,然后反向找到所需要的方法的句柄。

- 7) 调用方法:GSW 调用所请求的方法并取得结果。

- 8) 序列化结果:GSW 使用消息处理器序列化返回结果或抛出的异常,形成一个字节数组。

- 9) 将结果返回给客户端:将结果的数组送到分派器,分派器再将其送到 HttpHandler,再送到 IIS,IIS 会将结果

再传回给客户端。

4 安全性

创建一个基于 .NET 的容器会涉及到很多安全性的问题。一部分是与 .NET 相关,一部分是与宿主环境相关。

首先,OGS.NET 通过在 GSW 的消息处理器运行的 WSE 流水线,提供了对服务实例的消息层的安全机制。即消息层的安全性是使用 WS-Security 和 WS-Security 的 WSE 扩展进行处理的。

另外一些安全问题涉及到分派器和容器。主要是系统组件怎样避免错误的实现或者恶意的服务。应用程序域(AppDomains)提供了进程内的内存保护机制,而不需要为每一个服务创建重量级的进程。它允许每一个服务实例在其自己的 AppDomain 中存活,并为在容器内的其它服务提供保护。因为 GSW 实际上只调用在其 AppDomain 内的服务的方法,所以即使出现了问题令服务崩溃或者中止,也不会影响到分派器和容器中其它的服务。但是,假如允许服务调用非托管的代码的话,它们会绕过 AppDomain 的保护机制。

同样,Factory 创建网格服务实例的时候,也会创建一个新的 AppDomain,然后在 AppDomain 中创建一个 GSW,GSW 再将网格服务所需要的程序集装载进来。在一个新的 AppDomain 中装载服务的程序集,并初始化网格服务,可以令容器中的其余部分受到保护,不会因为创建服务的代码的潜在 BUG 而受到破坏。

有时候,需要某些特定的用户拥有一些特权,让该服务能够以某种用户的身份访问某些本地资源。在 OGS.NET 中提供了两个方案^[7],第一是拥有一个具有分级机制的容器系统,在这个系统里面,一个单独的主控制容器调度其它具有各自不同的 Windows 用户身份的容器。每一个这样的用户容器只可以创建基于特定的用户 ID 的服务,因此,容器中的每一个服务就可看成是容器的所有者。这种方案与使用 GT3 Managed Job Service 进行计算很相似。另一个方案是让每一个服务中活跃的线程以服务作者的 Windows 身份来运行。通过这种方式,多个拥有不同 ID 的线程可以在同一个容器中运行。

5 基于特性的编程模型

在 OGS.NET 中,编写网格服务与编写一个 Web 服务一样简单。它具有与 Web 服务的 .NET 编程模型之间的合作性。NET 中最吸引人和最强大的功能之一就是基于特性的编程。这个功能被用于使用元数据来注释类、方法以及代码段。这个语言功能被广泛应用到 .NET 中,用以支持 Web 服务。OGS.NET 利用这个特点为网格服务定义某些公共定义的特性,从而帮助开发者形成实现的内部细节。

图 3 的代码清单解释了一个天气预报的网格服务配

置过程^[7]。在这个过程中,每个服务工厂拥有一个惟一的名称,并使用 OGSi 定义的 factory-service.wsdl。真实的实现工厂类使用 createClass 进行定义,接受类名称和程序集名称。服务工厂的处理程序列表在 messageHandler 属性中指定。服务实例的处理程序列表在 createMessageHandlers 属性中指定。并且,服务必须分别使用 createSchemaPath 和 createClass 参数提供它的 wsdl 和类名称。

```
<service name = samples/weather/WeatherForecastFactoryService
>
<parameter name = "schemaPath" value = "schema/core/factory/
factory-service.wsdl">
<parameter name = "class" value = "UVa.Grid.OGSISamples.Fac-
tory.BasicFactoryService,OGSISampleServices.dll">
<parameter name = "messageHandlers"
value = "UVa.Grid.SoopMessageHandlerLib.SoopMessageHandler,
SoopMessageHandlerLib.dll">
<parameter name = "createSchemaPath"
value = "schema/weather/WeatherForecastService.wsdl">
<parameter name = "createClass"
value = "UVa.Grid.SampleServices.WeatherForecastService, Sam-
pleServices.dll">
<parameter name = "createMessageHandlers"
value = "UVa.Grid.SoopMessageHandlerLib.SoopMessageHandler,
SoopMessageHandlerLib.dll">
</service>
```

图 3 网格服务配置代码

(上接第 6 页)

的候选集生成-测试方法来处理一个比较小的数据库(例如只有 10k 的序列),需要相当多的时间来生成和测试大量的候选序列模式。

2) 基于投影的分治是数据缩减(reduction)的有效方法。序列模式 α 的投影数据库包含且仅包含用来挖掘那些由 α 扩展得到的模式的必需信息,投影数据库的大小随着挖掘过程向更长的序列模式进行而迅速缩减。

3) PrefixSpan 需要的内存空间相对稳定。原因在于它采用分治的方法,不生成候选集。而 GSP 和 SPADE,当支持度阈值(support threshold)降低时,由于需要容纳大量候选序列,需要相当数量的内存。

基于模式扩展的方法,可以应用到多层次、多维度的序列模式中,也可以挖掘其他结构化的模式。

4 结论与展望

序列模式挖掘是当前数据挖掘领域中一个较新而且非常活跃的研究分支,有着广泛的应用价值。文章在介绍了序列模式挖掘的相关概念后,对两类序列模式挖掘的几个经典的算法进行了描述和分析,不难发现,基于模式扩

6 结束语

文中涉及的在 .NET 框架下基于 OGSi.NET 的网格计算技术,结合实例能灵活、高效地应用于网格计算环境中。为网格计算的研究提供了一个新的途径。同时仍然还有很多问题需要进一步说明,包括持久性、可伸缩性和 GT3 与 OGSi.NET 之间的互操作性等。

参考文献:

- [1] 都志辉,陈渝,刘鹏. 网格计算[M]. 北京:清华大学出版社,2002.
- [2] Globus Toolkit 3.0.2. The Globus Alliance. Available[EB/OL]. <http://www.globus.org/toolkit/gt3-factsheet.html>, 2003.
- [3] Microsoft Corporation. .NET Framework[EB/OL]. <http://www.microsoft.com/net/>, 2003.
- [4] OGSi.NET Programmer's Reference[EB/OL]. <http://www.cs.virginia.edu/humphrey/GCG/ogsi.net.html>, 2004.
- [5] Tuecke S, Czajkowski C, Foster I, et al. Grid Service Specification-Draft 11/4/02. OGSi Working Group, Global Grid Forum[EB/OL]. <http://www.ggf.org/ogsi-wg>, 2002.
- [6] Joseph J, Fellenstein C. 网格计算[M]. 战晓苏,张少华,译. 北京:清华大学出版社,2005.
- [7] OGSi.NET: OGSi-compliance on the .NET Framework[EB/OL]. <http://www.cs.virginia.edu/humphrey/GCG/ogsi.net.html>, 2004-12.

展的方法是个前途很好的发展方向。模式扩展方法还有很多工作要做,如闭合集挖掘、在特定领域的针对性研究等等。

参考文献:

- [1] 毛国君,段立娟,王实,等. 数据挖掘原理与算法[M]. 北京:清华大学出版社,2005.
- [2] Han Jiawei, Pei Jian, Yan xifeng. From Sequential Pattern Mining to Structured Pattern Mining: A Pattern-Growth Approach[J]. Journal of Computer Science and Technology, 2004,19:257-279.
- [3] Agrawal R, Srikant R. Mining sequential pattern[A]. Proc 1995 Int Conf Data Engineering (ICDE's95)[C]. Chinese Taipei:[s.n.], 1995.3-14.
- [4] Srikant R, Agrawal R. Mining sequential pattern: Generalizations and performance improvements[A]. Proc 5th Int Conf Extending Database Technology(EDBT's96)[C]. Avignon, France:[s.n.], 1996.3-17.
- [5] Pei Jian, Han Jia-Wei, Mortazavi-Asl B, et al. Mining sequential pattern by Pattern-Growth: The PrefixSpan Approach[J]. IEEE TKDE, 2004,16(10):9-13.