

基于 ASP.NET 的 Web 服务性能优化

邓海生, 李军怀, 刘红英

(西安理工大学 计算机科学与工程学院, 陕西 西安 710048)

摘要: Web 服务作为一种分布式计算技术模式, 在电子商务、电子政务中发挥着重要的作用, 其性能也日益引起大家的关注。针对 Web 服务的额外性能开销问题, 从客户端应用程序和 Web 服务出发, 通过优化服务调度策略、减少网络传输数据量和使用数据高速缓存三种策略提高 Web 服务性能。

关键词: Web 服务; 性能开销; 调度策略; 高速缓存

中图分类号: TP393

文献标识码: A

文章编号: 1673-629X(2007)10-0095-04

Web Services Performance Optimization According to ASP.NET

DENG Hai-sheng, LI Jun-huai, LIU Hong-ying

(School of Computer Science & Engineering, Xi'an University of Technology, Xi'an 710048, China)

Abstract: Web services, as a kind of distributed computing technology pattern, is playing the vital role in e-commerce and e-government and Web services performance has been concerned increasingly by people. Based on the additional performance expenses of Web services, starting with the client application programme and the Web services, this paper enhances the Web services performance through optimizing service dispatch strategy, reducing network transmission data quantity and using data high speed cache.

Key words: Web services; performance expenses; dispatch strategy; high speed cache

0 引言

随着互联网技术与应用的发展, 基于 Web 的分布式技术与应用(如电子商务、电子政务等)已成为十分重要的发展方向, 而 Web 服务^[1]技术是当前基于 Web 的分布式技术与应用的关键技术基础。Web 服务是一种基于 SOAP, WSDL, UDDI 的面向服务的体系结构(Service Oriented Architecture, SOA)^[2], 能够消除在松耦合环境下使用不同组件模型、操作系统和编程语言的系统之间的差异。

Web 服务建立了一种基于 XML, 只需在 HTTP 和 SMTP 这样的互联网标准协议上传输 XML 消息的标准机制。这种标准机制是在多层封装的基础上完成的, 因此比直接的数据访问和过程调用增加了 XML 消息封装和解析的开销, 此外 Web 服务的简单易用性是通过用户对用户屏蔽复杂性和更高层的抽象来达到的, 这需要付出额外的计算。因此 Web 服务技术与其它分布式计算技术(如 DCOM^[3], Java RMI^[4], COR-

BA^[5])相比性能具有一定的差距, 而能否进一步提高 Web 服务的性能已经成为决定 Web 服务能否进一步发展的关键因素之一。

文中介绍了 Web 服务技术和 ASP.NET 技术, 然后针对 ASP.NET 构建的 Web 服务, 从客户端和 Web 服务器出发, 就服务调度策略、网络传输数据量和数据高速缓存三个方面分析了影响 Web 服务性能的因素, 并提出了改进方法。

1 Web 服务与 ASP.NET

1.1 Web 服务概述

简言之, Web 服务所提供的最简单级别的应用是它可以使用 SOAP 协议和 HTTP 协议使 Internet 上的远程消费者程序能够使用业务逻辑。如图 1 所示。

图 1 中, 一个客户端请求是使用 HTTP 协议上的 SOAP 协议, 然后经由 Internet 进行传递的。Web 服务发送响应给客户端应用程序, 该响应是作为 HTTP 上的一个 SOAP 消息被发送的。客户端请求和 Web 服务响应的 SOAP 消息主要是基于 XML 格式的, 并在 XML 有效负载中包含了基本的信息。这种解决方案的优点在于其互操作性, 通过简化边界间的通信, Web 服务能够使不同系统间的通信变得更加容易、简便。

收稿日期: 2006-12-24

基金项目: 西安科技局信息技术专项资助项目(ZX06030)

作者简介: 邓海生(1980-), 男, 山东人, 硕士研究生, 研究方向为数据库安全; 李军怀, 博士, 副教授, 研究方向为网格和高性能计算。

只要客户端应用程序能够为 SOAP 请求生成必要的 XML 语句,并能够解释 SOAP 响应中的 XML 语句,该客户端就可以在某个位置与 Web 服务进行通信。

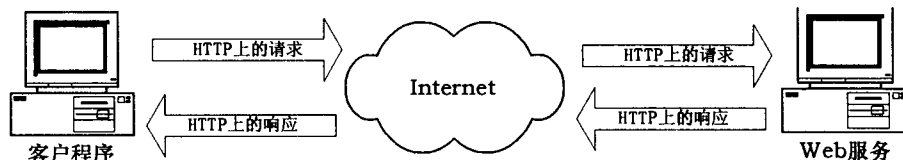


图 1 Web 服务

1.2 ASP.NET

ASP.NET 是位于 .NET Framework 顶层的一组组件和服务。通过使用 ASP.NET 开发 Web 服务,开发人员可以使用 Framework 所包含的众多类库。这为开发人员提供了使用 SOAP 消息和处理 WSDL 文件的所有基本功能。ASP.NET 还提供了主控 Web 服务的功能强大的运行时环境,而主控 Web 服务则减轻了建立和部署 Web 服务应用程序的很多工作。因此,ASP.NET 是 .NET Framework 中创建 Web 服务的一种很好的方式。

2 优化客户端应用程序

优化客户端应用程序包括合理选择 Web 服务调度方式和减少 Internet 数据传输量两部分。为了改善处理时间较长的 Web 服务方法的性能,前者使用异步的 Web 服务方法;后者通过削减 Internet 发送的冗余数据,增加了与 Web 服务通信的有效带宽。

2.1 Web 服务调度

通常的 Web 服务应用是采用同步的 RPC 调用实现,即客户端发送一个请求给 Web 服务端,然后由 Web 服务端执行请求任务,最后将执行的结果返回客户端。在这种情况下,如果 Web 服务立即响应的话,不会产生任何性能问题。但是,如果需要通过 Internet 远程调用 Web 服务,或 Web 服务端需要较长的时间来完成服务处理的话,则客户端程序处理请求的线程会一直被占用,直到 Web 服务调用 Web 方法结束。在这种情况下,采用同步 Web 服务会导致资源利用效率低下,还会引起事务性和伸缩性问题及其它问题。

为了改善服务处理时间较长的 Web 服务方法的性能,可以使用异步 Web 服务方法。异步 Web 服务方法使得客户端程序主线程可以及时返回,继续执行其他的操作,以提高应用程序整体性能和系统的可伸缩性。图 2 阐述了异步化处理的概念。

异步处理的过程描述如下:

①客户端获得一个 Web 服务器对象的接口指针并调用异步化方法,客户端包括用来回收信息的接收器对象的函数指针。

②调用将立刻返回并且调用线程可以自由地运行下一行代码。

③当这个方法完成请求处理时,服务器就通过在接收器对象中回收例程通报客户端。客户端通过轮询检测、软件中断信号或明确等待完成信号来发现调用的完成,并处理响应。

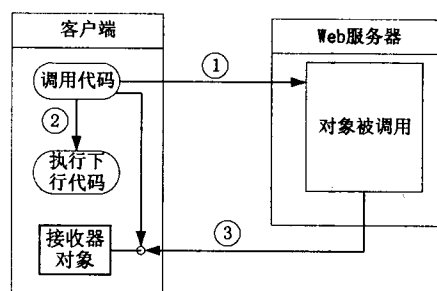


图 2 异步处理

在 .NET 中创建异步 Web 服务遵循 .NET 框架的异步编程模式,客户端到 Web 服务端进行异步调用的基础结构内置于 .NET 框架和用 Web 服务描述语言工具(WSDL.exe)生成的代理类中,由代理类提供与 Web 服务方法进行异步通信的机制。对每一个异步化处理的实现,设计模式中都应该有两个异步化方法:Begin 和 End,Begin 方法从客户端应用程序中获取输入并且取消异步处理操作,End 方法把异步处理的结果返回给客户端。

2.2 Internet 数据传输量

Web 服务经常用来从数据库读取数据和修改数据库内容,对于修改数据库内容涉及两种情况:

* 简单的数据添加:使用 Web 服务添加新的行,不涉及数据库现有的内容。

* 非连接数据修改:把数据加载到一个客户端应用程序,进行修改并更新数据库及记录。

简单的数据添加能以多种方式实现,对于 Web 服务代码只需要执行一个简单的 INSERT 语句就可以完成数据的添加。对于非连接数据修改可以使用 DataSet 对象,这些对象使客户端可以本地访问数据库,然后返回一个修改过的 DataSet 让 Web 服务提交做出的修改。上述两种情况都会导致大量的数据在 Internet 上传输。为了阐述这一点,看一个简单的 Web 服务。该服务实现了数据库更新,主要代码如下:

```
private void Update(DataSet newData)//更新数据库函数
{...
```

```
SqlDataAdapter sqlDaTest = new SqlDataAdapter(sqlCmd);
SqlCommandBuilder SqlCommandBuilderTest = new SqlCommandBuilder(sqlDaTest);
```

```

sqlTest.Update(newData, "Test");
...
}
[WebMethod]
public void UpdateDataSet(DataSet update)
{//保存客户端发送的实际请求
this.Context.Request.SaveAs(@"c:\UpdateRequest.txt", true);
Updata(update); //调用函数 Update,完成数据库更新
}

```

客户端调用该 Web 服务示例代码如下:

```

...
DataSet ds = proxy.GetDataSet();
DataRow newRow = ds.Table["TestTbl"].NewRow();
newRow["ROW"] = "TestRow";
ds.Table["Test"].Rows.Add(newRow);
proxy.UpdataDataSet(ds); //添加一行"TestRow"(模拟数据表只有
一个字段 ROW)
...

```

检查文件 UpdateRequest.txt 发现, DataSet 中的所有数据已经被重新发回到 Web 服务器端。反映出的唯一变化是:

```

<TestTable diffgr:id="40" msdata:rowOrder="40"
Diffgr:hasChanges="inserted">
<ROW>Test</Row>
</TestTable>

```

也就是说 DataSet 中没有变化的数据也通过 Internet 发回服务器端了,大大浪费了带宽。ASP.NET 中提供了方法 GetChanges() 可以大大缩短这种既包括数据又包括对数据修改的 DiffGram 格式。只要在客户端做如下的修改: proxy.UpdataDataSet(ds, GetChanges()) 就可以实现只返回被修改数据,大幅度地削减通过 Internet 发送的数据量。

3 Web 服务器端

高速缓存是一个缓冲器,它能把应用程序频繁使用的数据保存在一个较快的可以利用的容器内,比如 OS 内存、文件、数据库等等。高速缓存对于创建一个高可用性和可伸缩性的 Web 站点来说,是一项很重要的技术。运用高速缓存,可以明显提高 Web 服务的性能,比如在内存中高速缓存经常使用来创建代价昂贵的数据或数据结构,可以大大改进 Web 服务的性能,而不必为每个请求重建那些来自缓存的数据源。

3.1 输出高速缓存

输出高速缓存是一项在指定的时间内把 Web 服

务输出进行缓冲存储的技术。Web 服务是通过使用一个 WebMethod 声明的属性 CacheDuration 来支持输出高速缓存的。例如:

```

[WebMethod(CacheDuration=60)]
public string GetServerDateTime()
{ return System.DateTime.Now.ToString();
}

```

使用了 WebMethod 的 CacheDuration 属性,设置了输出高速缓存的时间间隔为 60 秒,那么在 60 秒内该服务返回的时间就是第一次服务被调用时的返回时间。也就是说在第一次调用该服务时,输出结果被存储在输出缓存中了,而后的 59 秒中每次服务调用结果均来自输出缓存。

输出高速缓存技术不仅与 CacheDuration 设置的时限有关,还依赖于服务输入的参数变化。例如某 Web 服务有两个输入参数 A 和 B,参数 A 有两种输入选择 A1 或 A2,参数 B 有两种输入选择 B1 或 B2,如此以来可能的输入组合为 {A1, B1}、{A1, B2}、{A2, B1}、{A2, B2}, 那么在 CacheDuration 设定的时间内调用该服务的过程如图 3 所示。

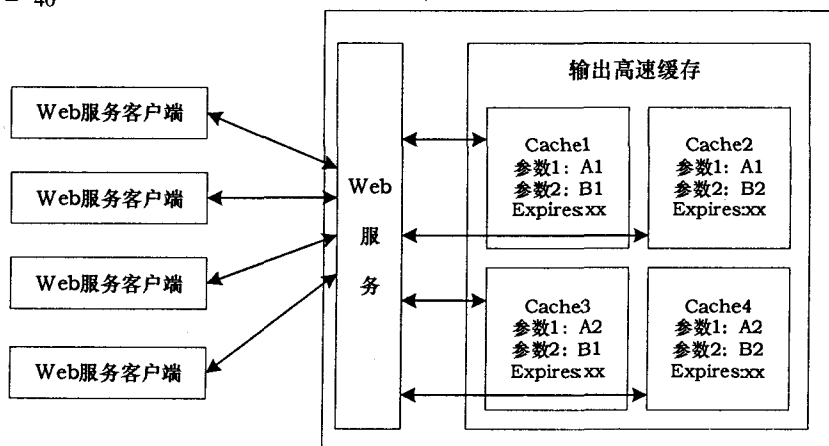


图 3 输出高速缓存

在高速缓存设定的期限内,如果有一个带有同一个 Web 服务参数组合的新请求进来,那么输出将从高速缓存中返回。如果这一特殊的参数组合没有输出高速缓存,那么这个 Web 服务请求将会被处理并且结果会被返回给客户端,同时已产生的 Web 服务输出将会在输出高速缓存中被高速缓存一定的时间(由 CacheDuration 设定的时间决定)。

3.2 数据高速缓存

Application 对象和 Cache 对象都可以高速缓存一些 Web 服务经常使用的数据。但是 Cache 对象在操作高速缓存的数据方面更为灵活,笔者就以 Cache 对象为例,讨论数据高速缓存。 .NET Framework 在命名空间 System.Web.Caching 中提供了 Cache 类,而 Cache

类实例在应用程序域中创建,并且在 Web 服务中可以使用 Context 属性访问 Cache 对象。Cache 对象为高速缓存重要数据提供了简便的方法,即通过使用键对重要数据高速缓存,然后可以使用键标找回数据。示例代码见下面的 Web 服务,该服务读取著名的 Northwind SQL Server 数据库并通过一个 ADO.NET DataSet 对象把所有产品种类返回到客户端。

```
...
using System.Web.Caching;
...
[WebMethod]
public DataSet GetProductCategories()
{ //如果 cache 变量为空,将产品种类数据集添加到 cache 变量中
if(Context.Cache["CategoriesDataSet"] == null)
{ SetCache(); }
//返回产品种类数据集到客户端
return (DataSet)Context.Cache["CategoriesDataSet"];
}
private void SetCache()
{...
string str = "select CategoryID,CategoryName from Categories";
SqlDataAdapter SqlAdp = new SqlDataAdapter(str, conn);
DataSet ds = new DataSet();
SqlAdp.Fill(ds, "Categories");
//将产品种类数据集保存到 Cache 对象中
Context.Cache.Insert("CategoriesDataSet", ds);
}
```

如此一来,所有产品种类信息以 DataSet 的形式被 Cache 高速缓存,以后要检索产品种类信息只要从 Cache 中取数据即可,而不必再访问数据库,从而提高了 Web 服务的性能。如果数据表经常变化,需要每隔

一段时间刷新一次数据的话,只要对 Context.Cache.Insert 稍加改动即可,在语法 Context.Cache.Insert() 中为数据提供了基于时间的高速缓存功能。

4 结束语

针对 Web 服务的额外性能开销问题,从客户端应用程序和 Web 服务出发,提出了优化服务调度策略、减少网络传输数据量和使用数据高速缓存三种优化 Web 服务策略,提高了 Web 服务性能。

服务性能优化不是一个单一的问题,从协议设计到协议实现、从服务传输到运行环境管理、从某种具体优化技术到整体优化方案,都需要考虑如何优化改善服务性能。笔者将深入研究这些问题,探索整体优化方案,以期能更好地改善 Web 服务的性能。

参考文献:

- [1] Pham H. Software Reliability and Testing[M]. Los Alamitos, CA, USA: IEEE Computer Society Press, 1996: 105 - 132.
- [2] Mc Gregor J D. Integrating Object - Oriented Testing and Development Processes [M]. New York, NY, USA: ACM Press, 1994: 59 - 77.
- [3] Microsoft Corp. Distributed Component Object Model Protocol - DCOM/1.0[S]. US: MSDN Library, Microsoft Corporation, 1998.
- [4] Sun Microsystems. Java Remoted Method Invocation(RMI) Specification[S/OL]. 2000 - i2. <http://java.sun.com/products/jdk/rmi/>.
- [5] Object Management Group. The Common Object Request Broker: Architecture and specification, rev 2.0[S/OL]. 2002 - 02. <http://www.omg.org/>.

(上接第 94 页)

4 结 论

B/S 模式下客户端之间的信息交互是比较棘手的问题,文中运用 Ajax 和 Jsp/Servlet 技术完美地实现了 B/S 模式下客户端之间的信息交互。通过两个有实际应用的例子证明了其方法是可行的。B/S 模式下客户端之间需要交互的应用比较广泛,所以文中提出的方法对于对实时要求不是很严格的领域具有一定的通用性。

参考文献:

- [1] 陈文博,尹 维. 基于矢量图的动态 Web 方法与客户端的

交互技术[J]. 计算机工程与应用, 2001(5): 118 - 121.

- [2] 牛 娜,田 李,彭晓明. 一种改进的即时消息传递系统认证方案[J]. 计算机工程与设计, 2006, 27(3): 490 - 492.
- [3] 李 峰,罗 谦. 企业信息化中的多层次可靠消息传递体系研究[J]. 计算机工程与应用, 2006(3): 227 - 229.
- [4] 吕林淘,万经华,周红芳. 基于 AJAX 的 web 无刷新页面快速更新数据方法[J]. 计算机应用研究, 2006, 23(11): 199 - 200.
- [5] 许南山,刘长华. 基于 B/S 结构的实时检测系统在 .net 平台下的设计与实现[J]. 计算机系统应用, 2004(9): 53 - 56.
- [6] 徐 驰. Ajax 模式在异步交互 Web 环境中的应用[J]. 计算机技术与发展, 2006, 16(11): 228 - 230.