基于 Ajax/REST 的 Web 架构的研究

高尚
北京邮电大学计算机科学技术学院，北京（100876）
E-mail：gaoshang1999@163.com

摘　要：REST 是一种被实践证明的适用于 Web 应用的架构风格，但是在传统的开发方法中，要完全实现 REST 架构，却存在着各种困难。近年来，Ajax 技术的兴起，为实现 REST 架构带来了新的机遇。本文分析了 Ajax 技术 REST 架构和 Ajax 技术的特点，并对于使用 Ajax 技术实现 REST 架构的可行性进行的分析，并最终给出了结论。

关键字：REST，AJAX，架构，Web
中图分类号：TP 393

1．引言

尽管 Web 技术已经出现了 18 年，从早期静态的 Web 文档，到当今世界无所不在的动态 Web 应用，目前在开发 Web 应用中所采用的方法却还是不令人十分满意。一方面，Web 应用的性能，虽然随着网络速度的提升而有所提高，但是在现有网络基础设施不变的前提下，如何有效 Web 提升应用性能也没有较好的答案。另一方面，现有 Web 应用可伸缩性较差，所能支持的最大用户数往往有限。一般都通过使用集群系统来提升可支持的用户数。然而，使用集群的副作用是，用户会话必须在各个集群之间保持同步，从而又降低了系统的性能。因此，集群的规模必将受到限制，从而应用所能支持的用户数量也讲得不到提升。这也是为什么很多 Web 应用往往只能支持有限用户并发访问的原因。

因此，迫切需要一种能够解决这些问题的方法。2000年，Roy Thomas Fielding博士提出了基于分布式超媒体应用的架构风格REST（Representational State Transfer）。REST最早被用来指导设计HTTP和URL标准。REST正是一组适用于分布式超媒体系统的架构风格。用于指导基于分布式超媒体系统的应用。把REST架构用于Web应用，是近年来的研究热点。

从 REST 被提出之后，很多人就开始使用 REST 指导 Web 应用的开发。REST 架构风格不同于传统的 Web 应用开发主要在于三个方面[2]：1，统一的 Web 资源调用接口。 网络上的所有事物都被抽象为资源（resource）， 每个资源对应一个唯一的资源标识符（resource identifier），通过通用的连接器接口（generic connector interface）对资源进行操作，对资源的各种操作不会改变资源标识符。2，无状态服务和有状态客户端。在服务端的所有的操作都是无状态的（stateless），由客户端来维护会话的状态。这是 REST 不同与传统 Web 开发的主要特点。这样就最大程度的提高了服务器的可伸缩性。3，支持客户端缓存。添加缓存约束的好处在于，它们有可能部分或全部消除一些交互，从而通过减少一系列交互的平均延迟时间，来提高效率、可伸缩性和系统的性能。

然而现有技术在实现 REST 架构风格的无状态服务和缓存约束时，往往无能为力。2005 年伴随着 Web2.0 技术的发展，Ajax 技术作为一种新兴的 Web 客户端技术受到广大用户的青睐。起初，Web 开发人员只是将它作为一种改善用户体验，增加 Web 应用易用性的工具来使用。Ajax 技术技术的核心是与服务端的异步通信。然而技术上的一个小小的改进，却带来了 Web 应用架构上的革命。使用 Ajax 技术，无需刷新整个页面就能和服务端通信，从而给实现无状态服务带

来了机遇。另一方面，使用 Ajax 技术，不仅是只能得到 HTTP 通信的内容，而且能得到 HTTP 协议的控制信息。从而极大的提升了 Web 客户端的能力，使得大规模的使用 Web 缓存成为可能。

2．REST 的主要内容

总的来说，REST是符合一系列架构属性的基于网络的架构风格，Roy Thomas Fielding博士在对基于网络的应用的常见架构风格调查的基础上，推导的得到了REST风格。

2.1 客户-服务器

首先被添加REST风格中的约束是客户-服务器架构风格[1]。客户-服务器约束背后的原则是分离关注点。通过分离用户接口和数据存储这两个关注点，改善了用户接口跨多个平台的可移植性；同时通过简化服务器组件，改善了系统的可伸缩性。然而，对于 Web来说，最重要的是这种关注点的分离允许组件独立地进化，从而支持多个组织领域的Internet规模的需求。

2.2 无状态

接下来再为客户-服务器交互添加一个约束：通信必须在本质上是无状态的[1]。客户-无状态-服务器风格源自客户-服务器风格，并且添加了额外的约束：在服务器组件之上不允许有会话状态（session state）。从客户端发到服务器的每个请求必须包含理解请求所必需的全部信息，不能利用任何保存在服务器上的上下文（context），会话状态全部保存在客户端。因此从客户到服务器的每个请求都必须包含理解该请求所必需的所有信息，不能利用任何存储在服务器上的上下文，会话状态因此要全部保存在客户端。

这个约束导致了可见性、可靠性和可伸缩性三个架构属性。改善了可见性是因为监视系统不必为了确定一个请求的全部性质而去查看该请求之外的多个请求。改善了可靠性是因为它减轻了从局部故障中恢复的任务量。改善了可伸缩性是因为不必在多个请求之间保存状态，从而允许服务器组件迅速释放资源，并进一步简化其实现，因为服务器不必跨多个请求管理资源的使用。

与大多数架构上抉择一样，无状态这一约束反映出设计上的权衡。其缺点是：由于不能将状态数据保存在服务器上的共享上下文中，因此增加了在一系列请求中发送的重复数据（每次交互的开销），可能会降低网络性能。

2.3 缓存

为了改善网络的效率，添加了缓存约束，从而形成客户-缓存-无状态-服务器风格[1]。缓存约束要求一个请求的响应中的数据被隐式地或显式地标记为可缓存的或不可缓存的。如果响应是可缓存的，那么客户端缓存就可以为以后的相同请求重用这个响应的数据。

添加缓存约束的好处在于，它们有可能部分或全部消除一些交互，从而通过减少一系列交互的平均延迟时间，来提高效率、可伸缩性和用户可觉察的性能。然而，付出的代价是，如果缓存中陈旧的数据与将请求直接发送到服务器得到的数据差别很大，那么缓存会降低可靠性。

早期的Web架构，，是通过客户-缓存-无状态-服务器的约束集合来定义的。也就是说，1994年之前的Web架构的设计基本原理聚焦于在Internet上交换静态文档的无状态的客户-服务器交

互。而之后越来越多的Web应用开始背离这一约束。

2.4 统一接口

使REST架构风格区别于其他基于网络的架构风格的核心特征是，它强调组件之间要有一个统一的接口[1]。通过在组件接口上应用通用性的软件工程原则，整体的系统架构得到了简化，交互的可见性也得到了改善。实现与它们所提供的服务是解耦的，这促进了独立的可进化性。然而，付出的代价是，统一接口降低了效率，因为信息都使用标准化的形式来转移，而不能使用特定于应用的需求的形式。REST接口被设计为可以高效地转移大粒度的超媒体数据，并针对Web的常见情况做了优化，但是这也导致了该接口对于其他形式的架构交互并不是最优的。

2.5 分层系统

为了进一步改善与Internet规模的需求相关的行为，添加了分层的系统约束。分层系统风格通过限制组件的行为（即，每个组件只能“看到”与其交互的紧邻层），将架构分解为若干等级的层[1]。通过将组件对系统的知识限制在单一层内，为整个系统的复杂性设置了边界，并且提高了底层独立性。能够使用层来封装遗留的服务，使新的服务免受遗留客户端的影响，通过将不常用的功能转移到一个共享的中间组件中，从而简化组件的实现。中间组件还能够通过支持跨多个网络和处理器的负载均衡，来改善系统的可伸缩性。

分层系统的主要缺点是：增加了数据处理的开销和延迟，因此降低了用户可觉察的性能。对于一个支持缓存约束的基于网络的系统来说，可以通过在中间层使用共享缓存所获得的好处来弥补这一缺点。在组织领域的边界设置共享缓存能够获得显著的性能提升。这些中间层还允许对跨组织边界的数据强制执行安全策略，例如防火墙所要求的那些安全策略。

2.6 按需代码

为REST添加的最后的约束来是按需代码风格[1]。通过下载并执行applet形式或脚本形式的代码，REST允许对客户端的功能进行扩展。这样，通过减少必须被预先实现的功能的数目，简化了客户端的开发。允许在部署之后下载功能代码也改善了系统的可扩展性。然而，这也降低了可见性，因此它只是REST的一个可选的约束。

3. Ajax 技术的特点

Ajax 不是单一的技术，而是四种技术的集合。表 1 简要介绍了这些技术以及他们在 Ajax 中所扮演的角色[4]。

表 1 Ajax 的关键元素

JavaScript	JavaScript 是通用的脚本语言，用来嵌入在某种应用之中。Web 浏览器中嵌入的 JavaScript 解释器是允许通过程序与浏览器的很多内建的功能进行交互。Ajax 应用程序是用 JavaScript 编写的
CSS（层叠样式表）	CSS 是为 Web 页面元素提供一种可重用的可视化样式的方法。它提供了简单而又强大的方法，以一致的方式定义和使用可视化样式。在 Ajax 应用中，用户界面样式可以通过 CSS 独立修改。

DOM（文档对象模型）	DOM 以一组可以使用 JavaScript 操作的可编程对象展现出 Web 页面的结构，通过使用脚本修改 DOM，Ajax 应用程序可以在运行时改变用户界面，或者高效地重绘页面中的某个部分。
XMLHttpRequest 对象	XMLHttpRequest 对象允许 Web 程序员从 Web 服务器以后台活动的方式获取数据。数据的格式通常是 XML，但是也可以很好的支持任何基于文本的数据格式。

在 Ajax 的应用程序中，JavaScript 则是作为关联其他几种技术的方法。在具体的实现过程中，通过使用 JavaScript 操作 DOM 来改变和刷新用户的界面，不断地重绘和重新组织显示给用户的数据，并且处理用户基于鼠标和键盘的交互。CSS 则为应用提供了一致的外观，并且为以编程方式操作 DOM 提供了强大的捷径。XMLHttpRequest 对象则用来与服务器进行异步通信。

Ajax 技术的实质是应用程序的调用绑定和动态数据与显示逻辑的绑定都推迟到了客户端进行。从而可以减轻服务器的负担，提升用户使用 Web 应用的体验。目前，越来越多的 Web 应用开始使用 Ajax 技术。

一般来说，Ajax 技术主要具有以下几个特点：（1）浏览器中展示的是应用而不是内容，（2）服务器交付的是数据而不是内容，（3）流畅而连续的用户交互。

4. 基于 Ajax 技术的 REST 架构

REST 架构的两个主要原则，有状态客户和缓存在传统的 Web 应用开发中，不可能实现。然而 Ajax 技术的出现却为实现 REST 架构带来了新的机遇[3]。

4．1 不使用 Ajax 的客户端处理

随着访问 Web 应用程序的用户越来越多，系统需要的资源也会逐渐增加。可以让服务器来处理这一切，但将需要容量更大的服务器或集群服务器（服务器端状态在集群环境中并不太适用）。但如果将处理分布到客户机上，那么每增加一名新用户，您就相当于有了一台支持部分新负载的新电脑。如果将会话状态分布到客户机上，那么就有了一个无状态的服务器 —— 这是可伸缩 Web 应用程序中令人满意的一项特性。这看上去应该是种非常明智的做法，那么为什么不按这种方法设计所有动态 Web 应用程序呢？在 Ajax 出现之前，答案非常简单：每次用户访问一个新的 Web 页面时，应用程序状态时就会被销毁。

4．2 利用有状态 Web 客户机的优点

传统的服务器端 Web 应用程序将数据的标识和服务器上的动态数据元素合并在了一起，并将所构成的完整 HTML 文档返回给浏览器。Ajax 应用程序在其主要 UI 和浏览器中的主要逻辑方面有所不同；基于浏览器的应用程序代码可以在必要时获取新的服务器数据，并将这些数据织入当前页面呈现。数据绑定的位置看起来可能是一个实现细节，但是这种区别会导致完全不同的架构风格。

通常将 Ajax 应用程序描述成无需在每次点击时彻底地刷新整页的 Web 页面。尽管这个描述非常确切，但是根本的动机在于彻底刷新整页会令用户不耐烦，从而无法获得愉快、交互

式的用户体验。从架构的角度来看，整个页面全部刷新的设计甚至非常危险，这种设计使您无法选择在客户机存储应用程序状态，这可能会导致妨碍应用程序充分利用 Web 最强大的架构风格 REST。

Ajax 让不需要进行完全刷新就可以与服务器进行交互，这一事实使有状态客户机再次成为可用选择。这一点对于动态 Web 应用程序架构的可能性有深远的影响：由于应用程序资源和数据资源的绑定转换到了客户端，因此这些应用程序都可以享受到一些新的特性 ——动态的 Web 应用程序、个性化的用户体验，以及遵守 REST 准则的应用程序中简单、可伸缩的架构。

4．3 缓存 Ajax 引擎

Ajax 引擎一个有趣的特征就是：尽管它包含了很多应用程序逻辑和表示框架元素，但是如果经过恰当的设计，它可以不包含任何业务数据或个性化内容[3]。在典型的 Web 环境中，应用程序资源可能很少发生变动。这意味着负责隔离应用程序资源和数据资源的 Ajax 引擎是高度可缓存的。引擎包含大量应用程序逻辑（以 JavaScript 代码实现），另外还有此后将使用从服务器上异步获取的数据填充的 UI 框架（见图 1）：

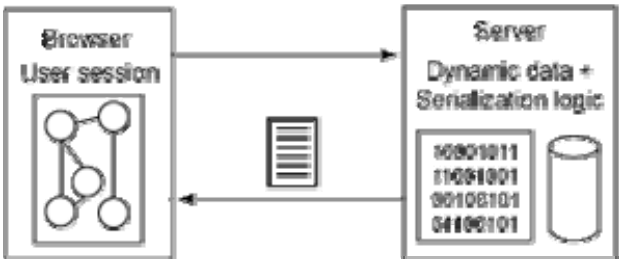

图 1. Ajax 应用程序

Dojo Toolkit 就是一个很好的例子。Dojo 提供了构建时工具来创建一个包含所有应用程序逻辑、表示和风格的压缩 JavaScript 文件。由于它终究只是一个文件，因此 Web 浏览器可以对其进行缓存，这意味着第二次访问启用 Dojo 的 Web 应用程序时，很可能就会从浏览器缓存中加载 Ajax，而不是从服务器上加载它。可以将这种情况与高度动态的服务器端 Web 应用程序进行一下对比，后者每次请求都会导致大量的服务器处理，因为浏览器和网络中介不能对缓存不断变化的资源。

由于 Ajax 应用程序引擎只是一个文件，因此它也是可以使用代理缓存的。在大型的企业内部网中，只要有一名员工曾经下载过某个特定版本的应用程序的 Ajax 引擎，其他任何人都可以从内部网网关上上获取一个缓存过的拷贝。

因此对于应用程序资源来说，经过良好定义的 Ajax 应用程序引擎符合 REST 准则，与服务器端 Web 应用程序相比，它具有显著的可伸缩性优势。

4．4 缓存 Ajax 数据

毫无疑问，和从服务端获取整个 HTML 页面（动态数据混合在其中）不同。Ajax 使得在客户端可以独立处理从服务端获取到的动态数据。因此，就可以在客户端灵活的使用 JavaScript 建立缓存池，缓存动态数据[2]。

另一方面，HTTP 协议在通信是，协议主要有：协议头和协议体两部分组成[5]。使用 Ajax 技术时，XMLHttpRequest 对象不仅允许获取从服务端得到的数据（HTTP 协议体），并且用能力获取到 HTTP 通信的控制信息（HTTP 协议头）。这样极大的提升了客户端的能力。使得使用 HTTP 缓存机制成为可能。

4．5 Ajax 和健壮性

Ajax 架构风格的另外一个优点是它可以轻松处理服务器的故障[3]。正如前面介绍的一样，传统的动态服务器端 Web 应用程序通常会在服务器上保存大量的用户会话状态。如果服务器发生了故障，会话状态就丢失了，那么用户就会体验到非常奇怪的浏览器行为（“为什么我又回到主页上来了？我的购物车中的东西都到哪里去了？”）。在采用有状态客户机和无状态服务的 Ajax 应用程序中，服务器崩溃/重新启动对于用户来说都是完全透明的，因为服务器崩溃不会影响会话状态，这些都保存在用户的浏览器中；无状态服务的行为是幂等的，可以由用户请求的内容来单独确定。

5．结论

本文分析了 REST 架构的主要特点，以及使用 REST 架构构建 Web 应用的优势。近年来，随着 Ajax 技术的兴起，为实现 REST 架构提供了的新方法。通过对 Ajax 技术特点的研究，可以证明，使用 Ajax 技术实现 REST 架构是完全可行的，尤其是使用实现 REST 架构中的有状态客户端和缓存约束。

可以预见，在不久的将来，越来越多的 Web 应用将会使用基于 Ajax 技术的 REST 架构来实现。

参考文献

1． Roy Thomas Fielding， Architectural Styles and the Design of Network-based Software Architectures, 2000
2． Christian Gross 著，Ajax 模式与最佳实践，李锟等译，电子工业出版社，2007 年 3 月
3． Bill Higgins, Ajax and REST, Advantages of the Ajax/REST architectural style for immersive Web applications, 2006, http://www.ibm.com/developerworks/cn/web/wa-ajaxarch/
4． Jesse James Garrett，Ajax: A New Approach to Web Applications，2005，http://www.adaptivepath.com/ideas/essays/archives/000385.php
5．David Gourley, Brian Totty, Marjorie Sayer, Sailu Reddy, Anshu Aggarwal，HTTP: The Definitive Guide, O'Reilly Press, 2003

The Research on Web Architecture based on Ajax/REST

Abstract

REST is an architecture which has been proven that it can be used for Web application. But in traditional Web application, technology does not allow to implement REST thoroughly. In recent years, Ajax technology is emerging, which give us a new chance to follow REST in our Web application. This paper analyzed the characteristic of Ajax and REST respectively, and analyzed the possibility of using Ajax to implement REST architecture in Web application. In the end, the conclusion is drawn.

Keywords：REST，AJAX，architecture，Web

作者简介：高尚，北京邮电大学计算机科学技术学院硕士研究生，研究方向：网络信息处理。