ISSN 1009-3044
Computer Knowledge and Technology 电脑知识与技术
Vol.8, No.16, June 2012.
E-mail: info@cccc.net.cn
http://www.dnzs.net.cn
Tel:+86-551-5690963 5690964

基于WebSocket的实时Web应用解决方案

温照松[1],易仁伟[2],姚寒冰[2]

(1.武汉市建设工程交易中心,湖北 武汉 430023;2.武汉理工大学 计算机科学与技术学院,湖北 武汉 430063)

摘要:由于用户对Web信息实时性要求的提高,实时Web应用开始被广泛关注,基于各种服务器推送技术的实时Web应用解决方案被提出并广泛应用。然而这些解决方案都存在系统资源消耗大等方面的问题,有待进一步的改进和完善。该文介绍了当前较为广泛应用的两种基于HTTP协议的实时Web应用方案,即基于Ajax的长轮询方式和基于Iframe的流方式,并分析了这两种方案存在的不足。在对HTML5标准中的WebSocket协议进行深入分析的基础上,该文提出了一种基于WebSocket协议的实时Web解决方案,目的是大幅提升实时服务性能,并更高效地利用网络负载和服务器端的处理能力。
关键词:服务器推送技术;实时Web应用;HTML5;WebSocket协议;HTTP协议

中图分类号:TP393 文献标识码:A 文章编号:1009-3044(2012)16-3826-03

WebSocket Based Real Time Web Application Solution

WENG Zhao-song[1],YI Ren-wei[2],YAO Han-bin[2]

(1.Wuhan Construction Project Trading Center,Wuhan 430023,China;2.Wuhan Uinversity of Technology,Wuhan 430063,China)

Abstract: Due to the improving demands on instantaneity of web information, real-time web applications is beginning to cause widespread interest, and a lot of real-time web applications based on server-push technology has been proposed and used widely. Though these solution are effective, all those solutions suffer from problems such as great resource consumption; there is much space for improvement. This paper introduces two popular HTTP protocol based real-time web application solutions, that is, long polling based on Ajax and streaming based on Iframe, and analysis their weaknesses. With an in-depth study on WebSocket protocol in HTML5 standard, this paper proposes a WebSocket protocol based real-time web application solution, aiming at improving the service instantaneity on a large scale, and more efficiently using the network capacity and processing power of the server.
Key words: server-push;real-time application; HTML5;WebSocket protocol;HTTP protocol

基于HTTP协议的Web应用已得到了十足的发展和应用,成为人们获取互联网上海量信息的主要途径。然而随着信息时代的发展,人们开始对Web应用提出了实时性要求,希望这些信息更新后能够立即获取到。对这项需求,多种实时Web应用解决方案被提出,其中较为成功并广泛应用的包括:基于Ajax的长轮询以及基于Iframe的流方式。基于这些解决方案,互联网上涌现出了一系列实时应用,如在线游戏、在线证券、设备监控、RSS阅读推送、邮件提示等等。

然而这些技术在广泛运用的过程中也暴露出了大量的问题,如实现复杂、资源消耗大、实时性不高等,这些问题制约了实时Web应用的性能。

随着HTML5标准的发展,该标准中提出的一项名为WebSocket的通信协议得到了广泛的关注。WebSocket可以在浏览器和服务器之间提供一条基于TCP连接的双向通道,服务器和浏览器通过这个通道可以实现双向通信。这种双向通信能力使得利用WebSocket来构建实时web应用成为可能。

该文将在总结现有的两种实时Web应用方案的基础上,提出一种基于WebSocket的解决方案,新的方案将在很大程度上解决传统方案暴露出的问题。

1 传统实时Web应用解决方案

在实时Web应用出现之初,最简单的实现方案是轮询。所谓轮询就是客户端以一定的时间间隔向服务器端发出请求,以频繁请求的方式来不断刷新客户端呈现的信息。这种方案缺乏灵活性,无论服务器端是否有信息更新,请求都会不断被发送,频繁的连接请求会给服务器端带来巨大的处理压力。所以这种方案逐渐被舍弃,进而发展出了基于Ajax的长轮询方式和基于Iframe的流方式。这两种方式都在原有轮询的基础做了改进,一定程度上克服了简单轮询的不足。

1.1 基于Ajax的长轮询方式和基于Iframe的流方式

1.1.1 基于Ajax的长轮询方式

所谓Ajax,是异步JavaScript和XML技术的简称。采用Ajax方法,客户端利用JavaScript的XMLHttpRequest对象以异步的方式

收稿日期:2012-03-23
基金项目:武汉理工大学自主创新研究基金资助(2011-IV-081)
作者简介:易仁伟(1987-),男,湖北宜昌人,武汉理工大学计算机科学与技术学院,硕士在读,研究方向为现代数据库理论与应用。

向服务器发送HTTP请求,在不重载页面的情况下完成与服务器的数据交换。利用Ajax的这一特性,可以避免对于客户端页面的频繁刷新。

基于Ajax的长轮询是目前实时Web应用中使用频率较高的方式,这种方式采用Ajax方法向服务器端发送请求,在服务器端没有信息更新时,服务器会一直维持这个请求连接,直到有新的信息需要返回给客户端或者连接超时才会关闭这个长连接。

基于Ajax的长轮询的具体工作过程如下:客户端发出请求,进而与服务器建立连接;然后服务器端会将这个连接保持一段时间,通常是数秒钟,也可能是一分钟甚至更长;当服务器端检测到客户端请求的数据有更新,即有必要将新的数据发送给客户端时,会立即通过维持的连接将数据发送给客户端,然后关闭连接;如果服务端在维持这个连接过程期间没有新的数据发送到客户端,则会在维持时间超时后关闭连接。无论是否有数据更新,在上一次连接关闭后,客户端会立即重新发出一个请求,再次与服务器端建立连接,如此循环,确保新的数据能及时的发送到客户端。

1.1.2 基于Iframe的流方式

Ifram是HTML提供的一种标签,利用Iframe可以在HTML页面中创建一个内联的文档框架,该标签的SRC属性用来设置该内联框架需要显示的文档的URL。当包含Iframe元素的页面加载显示时,Iframe会通过SRC属性设置的URL获取文档内容并显示。

基于Iframe的流方式也是应用较为广泛的方案。基于Iframe的流方式在页面中内置一个隐藏的Iframe元素,将Iframe的SRC属性设置为一个长连接请求,服务器端会不断更新连接状态,使这个长连接在执行过程中一直处于连接状态,服务器端在数据更新后会立即通过这个长连接将数据传送到客户端隐藏的Iframe中,客户端通过Iframe获取这些数据完成页面内容的更新。

基于Iframe的流方式的具体的工作过程如下:客户端页面在浏览器加载时,其内嵌的Iframe发起长连接请求,服务器端响应请求建立长连接;服务器端不断更新该连接的状态,使其始终保持连接状态;当服务器端检测到客户端请求的数据有更新时,立即通过该连接将数据发送给客户端;该连接在出现错误或异常关闭后,客户端会立即发出连接请求进而再次建立连接。

1.1.3 两种方案的比较

可以看出,这两种方案相对于轮询都有了较大的改进,而且两种方案的改进方向是一致的,主要包含两点:客户端页面内容更新时不会刷新整个页面,减少了服务器端返回的数据量,并能带来较好的用户体验;服务器端通过维持长连接,减少了客户端请求的频率,避免频繁请求连接的建立和关闭带来的资源浪费。

但是,两种方案在实现方式上存在一些不同:

1)服务器端传送给客户端的数据形式不同,页面内容的更新方式不同。采用基于Iframe的流方式,服务器端返回的是包含新数据的JavaScript代码,客户端浏览器执行这些JavaScript代码即可实现页面内容的更新;采用基于Ajax的长轮询方式,服务器端返回的是XML格式或者JSON格式的数据,客户端浏览器的JavaScript引擎需要首先解析这些数据,然后再操作页面元素实现页面内容的更新;

2)服务器端对于长连接的处理不同。采用基于Iframe的流方式,服务器端会不断更新连接状态,一直维持该长连接;而基于Ajax的长轮询则会在每次返回数据给客户端后关闭连接,再由客户端重新发起请求建立连接。

1.2 传统解决方案的困境

传统的两种解决方案虽然被广泛运用,但是都还存在很多问题。

基于Iframe的流方式中,由于Iframe中始终维持一个连接,用户的浏览器会始终显示当前页面处于加载过程中,始终不能显示加载完成,这将影响用户体验。而基于Ajax的长轮询方案中,当服务器端数据更新速度较快时,长轮询将退化为普通的轮询,这样将大大降低其性能,并会对服务器端带来较大的处理压力。除了上述的问题之外,这两种方案还存在一些共性的不足:

1) 由于这两种方案仍然采用HTTP作为通信协议,而HTTP连接的建立和关闭过程都有一定的资源和时间消耗。这不仅会导致服务器端资源的浪费,而且在连接建立和关闭过程产生的新数据无法及时发送到客户端,可能导致客户端数据丢失;

2)每一次的 HTTP 请求和应答都带有完整的 HTTP 头信息,这就增加了每次实时信息更新时的传输数据量,造成了网络带宽的浪费;

3)上述两种方案没有区分Web服务中实时信息和非实时信息,均采用相同的请求方式,服务器端的响应方式也相同,由于实时信息请求较为频繁,这就会对服务器端带来较大的处理压力,进而可能影响非实时信息的呈现。

2 基于WebSocket的实时Web应用解决方案

WebSocket是HTML5标准中提出的一种新的协议,它支持浏览器与服务器的双向通信[2]。不同于传统的HTTP方式,在使用WebSocket时,服务端和客户端处于同等的地位,可以随时相互发送消息。

WebSocket 协议本质上是一个基于 TCP 的协议,WebSocket连接与TCP连接的建立过程类似。浏览器通过 JavaScript 向服务器发出建立 WebSocket 连接的请求,服务器解析该请求并会向客户端返回应答信息,然后建立连接;连接建立以后,客户端和服务器端就可以通过 TCP 连接直接交换数据。

除了具备双向通讯能力,WebSocket协议的传输数据格式相对HTTP更加简洁。WebSocket在传输信息时,相对于HTTP大大降低了数据帧中头信息占用的字节数,从而降低了传输的数据量,节省了带宽资源,降低网络负载。

以往的实时web系统中,由于采用基于HTTP协议的通信方式,因此实时请求和非实时请求均以相同的方式来处理,这正是传统方案的症结所在。在一个实时Web应用中,并不是所有呈现的内容都是需要实时更新的,因此实时内容和非实时内容在服务端的处理应该加以区分,避免相互影响,这样也可以一定程度上减轻服务器端的处理压力。

基于WebSocket的实时Web应用解决方案将应用中的实时部分和非实时部分进行了分离,客户端呈现的非实时内容仍然采用HTTP来获取,而实时内容则使用WebSocket来获取。由于HTTP和WebSocket是两种不同的协议,服务器端将采用不同的处理方

式。如图1所示,服务器端将通过两个不同的模块,分别处理非实时的HTTP请求和实时的WebSocket连接,确保HTTP和WebSocket各自发挥自身的优势,而又相互不影响,合理分配服务端的处理资源。

新的解决方案相对于传统方案有以下几点优势:

1)新的应用模型可以使服务器端结构更加明确。两个模块的独立性可以降低系统的耦合,相互之间不会产生影响,最大程度发挥不同模块的效能,并方便针对模块自身的处理特点添加优化措施;

2)由于采用了可以在服务端和浏览器间建立稳定TCP连接的WebSocket协议来实现实时服务,可以保证数据传输的稳定性和及时性,并降低网络负载,可以较大幅度地提高实时服务的性能;

3)WebSocket协议简单易用,开发成本低,降低了系统开发的复杂性和代价;

4) 相对于传统方案,降低了对服务器端处理资源的浪费,减轻了服务器端处理的压力。

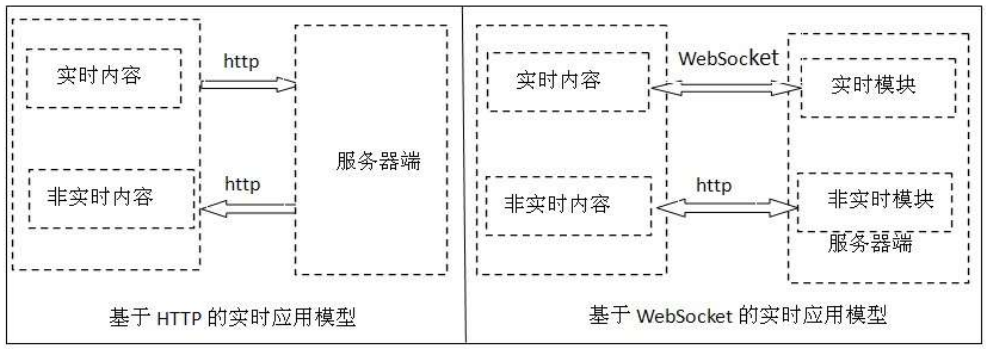

图1 实时Web应用模型

新的方案相对于原有方案可以大幅提升实时服务性能,并能更高效地利用网络负载和服务器端的处理能力。然而新方案也有一定的局限性,主要原因是:

1)WebSocket协议还处于草案阶段,还有变动的可能。目前,FireFox、Chrome以及Safari等浏览器均支持WebSocket,但市场占有率最高的IE浏览器还没有提供对WebSocket的支持。

2)WebSocket协议提供了统一的客户端API,然而服务器端的标准还没有明确。目前,各个服务器厂商对服务器端的支持存在差异,因此增加了服务器端程序的开发难度,也妨碍了WebSocket协议的普及和应用。

3 结束语

基于Ajax的长轮询和基于Iframe的流方式是经过长时间的应用实践而发展出来的,这两种方案成为了实时Web应用主要的实现方式。针对这两种方案存在的问题,技术人员进行了多种方式的优化,但HTTP协议本身特性的制约却无法逾越。

WebSocket协议的出现,提供了一种新的服务器端和浏览器的通信方式,利用该协议可以更加方便地构建出简单高效的实时Web应用。该文针对传统解决方案的不足,提出了基于WebSocket的实时Web解决方案,从理论层面分析了方案的可行性和相对于传统方案的优势。相信随着WebSocket协议的进一步发展,基于WebSocket的实时web解决方案将逐渐被广泛接受。

参考文献:

[1] Clinton Wong.HTTP Pocket Reference-Hypertext Transfer Protocol[M].Beijing:O'Reilly Media,2006.

[2] IETF.The WebSocket Protocol[EB/OL]. http://grenache.tools.ietf.org/html/rfc6455.

[3] IETF.The WebSocket protocol draft-ietf-hybi-thewebsocketprotocol-10[EB/OL].http://tools.ietf.org/html/draft-ietf-hybi-thewebsocket-protocol-10.

[4] W3C.HTML5 differences from HTML4[EB/OL]. http://www.w3.org/TR/html5-diff.

[5] 周婷.Comet:基于HTTP长连接的服务器推技术[EB/OL].http://www.ibm.com/developerworks/cn/web/wa-lo-comet/#resources/.

[6] Victoria Pimentel,Bradford G.Nickerson.WebSocket for Communication and Display of Real-Time data[J].IEEE Internet Computing,2012:1-12.

[7] 黄晓安,何亮.使用HTML5 WebSocket构建实时Web应用[EB/OL].http://www.ibm.com/developerworks/cn/web/1112_huangxa_web-socket/.

[8] Peter Lubbers.HTML5 Programing—Using the HTML5 WebSocket API[M].Apress,2010.