网页中图片离线加速下载方案的设计研究

宗 平[1],高 斐[2]
(1. 南京邮电大学 海外教育学院,江苏 南京 210023;
2. 南京邮电大学 计算机学院,江苏 南京 210023)

摘 要:通过对目前互联网流量的抽样分析,图片占全球所有互联网流量的比重超过 50%,几乎每个网页中都会产生非常可观的图片流量。并且随着日益增长的 Web 流量以及迅猛发展的移动互联网,人们需要一种比传统图片传输方式更加优化的方式来解决日益严重的性能问题。基于这种背景,文中给出一种将服务器图片缓存在 HTTP 页面中的离线加速下载方案,可以有效地提高图片的传输效率,加快页面的加载速度,在不改变现有 Web 框架基础的情况下使用户获得更好的用户体验。

关键词:图片传输;DATA-URI;HTTP
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2014)10-0071-04
doi:10.3969/j.issn.1673-629X.2014.10.017

Research on Design of Image Offline Accelerating Download Program in Web Page

ZONG Ping[1], GAO Fei[2]
(1. College of Overseas Education, Nanjing University of Posts and Telecommunications, Nanjing 210023, China;
2. College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)

Abstract: Through the sampling analysis of present Internet traffic, image flow accounts for over 50% of all the global Internet traffic. Almost every web page will produce a considerable flow of images. Because of the increasing Web traffic and the rapid development of mobile Internet, people need a better optimization approach than the traditional image transmission to solve the increasingly serious performance issues. Based on this background, propose a design scheme for the offline download acceleration that server images are cached in the HTTP page in this paper. Without changing the current Web framework, this scheme can effectively improve the efficiency of the transmission of the picture, speeding up page loading speed, and let users get a better user experience.

Key words: image transmission; DATA-URI; HTTP

0 引 言

从 1989 年诞生至今,Web 技术已经经历了 20 余年的发展[1],其用途也从最初的纯学术交流,延伸至如今的门户网站、电子商务网站、社交网站等,涉及人们生活及工作、学习中的方方面面。互联网世界有数以千亿计的网页,负责信息的承载和展示。从 GMAIL 开始,随着 AJAX[2] 的成熟和 Web2.0 的兴起,图片和视频等富媒体在网络中占据了越来越重要的地位。互联网传统的图片加载方式,每一张图片都需要一个独立的 HTTP 请求,导致网页加载过慢[3]。由于网页加载过慢,使得网络零售商每年在该问题上造成大量的资金损失,同时也使消费者变得沮丧。因此,需要把提升网站速度作为首要的任务进行考虑。文中通过对传统 Web 的传输方式进行解析,给出一种将服务器图片缓存在 HTTP 页面中的离线加速加载方式,以提升网页打开的速度性能。

1 Web 中图片的传输方式

1.1 传统 Web 中图片的载入方式

在传统的 HTML 页面中,图片通常使用 img 标签

收稿日期:2013-12-12 **修回日期**:2014-03-18 **网络出版时间**:2014-07-28
基金项目:江苏省科技支撑计划(BE2009157)
作者简介:宗 平(1956-),男,教授,硕士生导师,CCF 会员,研究方向为计算机网络技术等;高 斐(1989-),女,江苏无锡人,硕士研究生,研究方向为分布式计算机技术与应用。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20140728.1230.052.html

载入,图片的地址通过 img 标签的 src 属性指定一个 HTTP 的 URL,如<img class="featurette-image pull-left" width="512" src="http://wosaimg.com/img/iwosai/index/detail-02.jpg">。

当浏览器渲染 HTML,执行到 img 标签时,会发起另一个 HTTP 请求去请求图片,当图片返回 HTTP 200 成功时,浏览器的渲染引擎会将图片渲染到 window 的 body 中[4]。

每一个图片链接都需要一个独立的 HTTP 请求,而每次 HTTP 请求都会带来较大的网络开销,HTTP 请求需要建立 TCP 的连接[5],并且每次请求会带上 HTTP 的头部,加上网络传输中的时间消耗,不同浏览器的并行下载数量也会有一定的限制,所以大量的图片下载会带来严重的性能下降。

现在浏览器都会有并发图片下载数量限制,表 1 为主流浏览器在同一域名下图片并发下载的数量限制。

表 1 浏览器并发下载图片的限制

Browser	HTTP/1.1	HTTP/1.0
IE 6,7	2	4
IE 8	6	6
Firefox 2	2	8
Firefox 3	6	6
Safari 3,4	4	4
Chrome 1,2	6	?
Chrome 3	4	4
Chrome 4	6	?
iPhone 2	4	?
iPhone 3	6	?
iPhone 4	4	?
Opera 9.63,10.00alpha	4	4
Opera 10.51+	8	?

1.2 使用 DATA-URI 载入图片

通过使用 DATA-URI 模式可以在 Web 页面中包含图片但无需任何额外的 HTTP 请求。尽管低版本的 Internet Explorer 浏览器目前还不支持这种方式,但它能给其他浏览器带来的节省值得关注。

人们都很熟悉包含 http://模式的 URI 形式[6]。其他类似的模式包括 ftp:、file:和 mailto:。但除此之外还有很多模式,如 smtp:、pop:、dns:、whois:等。这其中有一些是官方注册的,还有一些由于广泛使用而被接受。

DATA-URI 模式在 1995 年被首次提议[7]。规范(http://tools.ietf.org/html/rfc2397)对它的描述为:“允许将小块数据内联为‘立即数’”。数据就在其 URL 自身之中,其格式如下:

data:[<midiatype>][;base64],<data>

一个红色五角星形状的内联图片可以定义为:

<img alt="red star" src=" H1KBLQRFIJAIKywRgmhwAIlEEADs=">

大多数情况下 DATA-URI 用户内联图片,但它可以用在任何需要指定 URL 的地方,包括 script 标签和 a 标签。

DATA-URI 模式的主要缺陷是不受 IE 浏览器的支持,另一个缺陷是可能存在数据大小上的限制。另外 base64[8]编码在不启用 GZIP[9]压缩时会明显增加图片的大小,因此整体下载量会增加。

由于 DATA-URI 是内联在页面中的,在跨越不同的页面时不会被缓存。文中通过优化 Web 传输中图片传输的策略,使得图片可以不经过 HTTP 请求而直接从服务器“离线”地下载到页面中。

1.3 传统方式与 DATA-URI 载入图片的优缺点分析

直接通过 http://demo.png 这种传统方式载入图片在 Web 的逻辑架构上有天然的简单性,并且可以通过 HTTP 的缓存策略方便地将图片内容缓存[10],浏览器或者其他移动设备下载完二进制内容后渲染图片消耗的 CPU 资源相比于需要 data64 编解码的 DATA-URI 方式更少,每一个图片都会发起一个 HTTP 请求,在网络链路质量不佳情况下会影响用户体验;而通过 DATA-URI 方式载入图片可以有效减少 HTTP 请求,在移动互联网的应用情境下可以大幅提升用户体验,但是 base64 编解码需要消耗更多的 CPU 资源,在移动处理器还不够大的情况下也会影响用户体验,且 DATA-URI 如果需要缓存则需要 CSS 和代码生成器去实现缓存逻辑,相比而言在软件架构上比传统方式更为复杂。

2 服务器离线下载图片策略

文中设计的将服务器图片缓存在 HTTP 页面中的处理流程如图 1 所示。用户在通过 HTTP 协议访问 Web 服务器获取资源时,所有请求会通过图片离线下载模块,该模块会分析 HTML 中所有的 img 标签并获取其 src 属性,并将其遍历。图片离线下载模块会在图片信息索引表中查询图片的 src 是否已经存在索引[11]。

(1)如果索引存在,向图片内存缓存服务发起查询请求,图片内存缓存服务将返回该 src 所对应的图片的 base64 资源。图片离线下载服务器将 img 标签中的 src 属性替换为 base64 资源;

(2)如果索引不存在,则图片离线下载服务器不

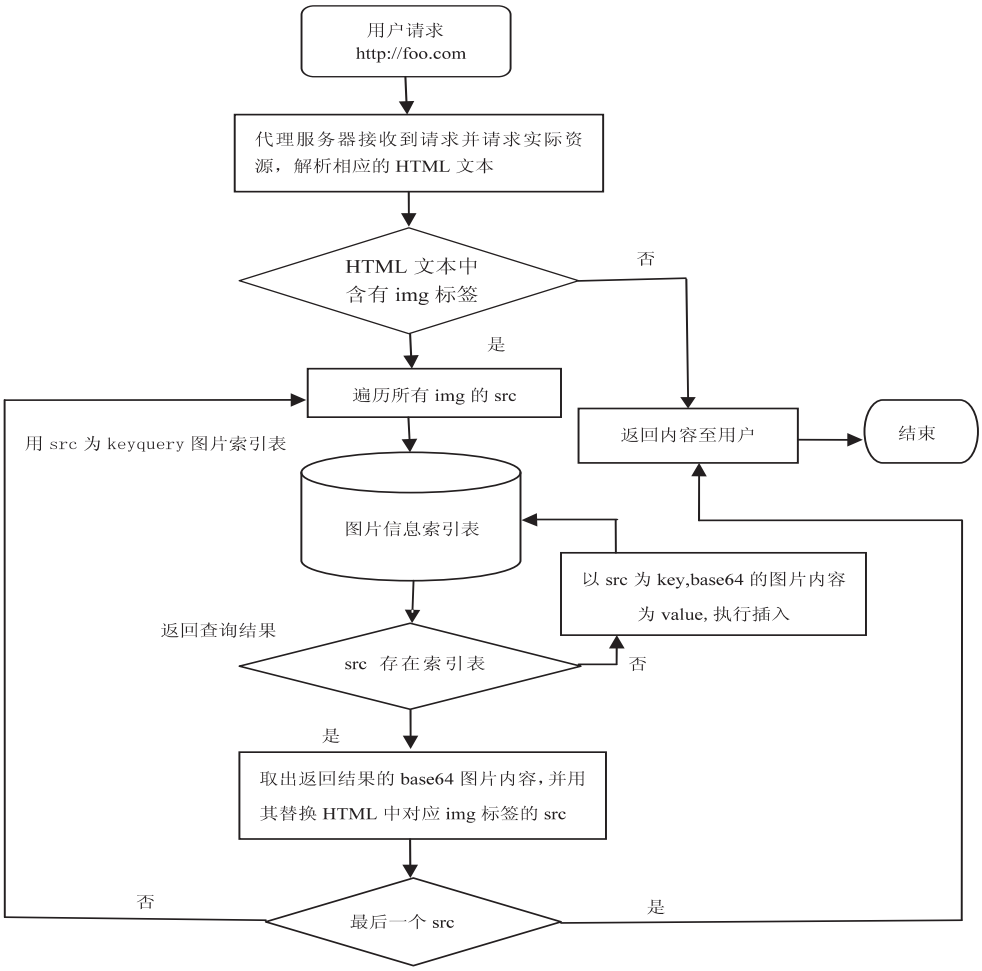

图 1　图片离线下载策略的流程

改变该 img 标签的 src,而会发起一个 HTTP 请求请求该图片资源,将其信息插入图片信息索引表,并将内容插入图片内存缓存服务。

最后图片离线下载模块将替换后的 HTML 页面(将 HTML 页面 GZIP 压缩后会大大优化性能)返回给用户。其中图片内存缓存服务所使用的内存大小可能有限,如有需要则需使用一个落地存储来承载更大量的图片存储,由内存和硬盘提供一个图片内存 cache 服务。

2.1　图片资源在服务器端的离线存储的设计

文中所说的离线存储是指首次用户访问 foo. com/ 的时候,在域名 foo. com/index. html 中访问了域名 bar. com/bar. png 图片时,将 bar. com/bar. png 图片下载到 foo. com 所在的服务器上,待有用户再次访问 foo. com 时,图片无需再次从 bar. com/bar. png 中拉取,而是已经以某种方式存储在 foo. com/index. html 中,无需再次向 bar. com 发起获取 bar. png 的 HTTP 请求。

2.2　图片索引的建立

网页中图片离线加速下载的实现首先需要对经过 web server 的图片建立索引。索引的格式表示成 JSON 如下所示:

```
{
  path:'http://wosaimg.com/img/foo.png',
  size:102333,//bytes
  md5:23f35a332f23f35a332f23f35a332fdd,
  last_modify:1382284501080
}
```

当服务器每次收到 HTTP 请求的资源是图片类型时,服务器首先检测图片是否存在于索引表中,图片的唯一性由 path 决定,如果两次请求的 path 一致,则认为请求的资源是一致的。如果图片索引已经存在于索引表中,则不再添加索引表,否则新建该路径的索引表信息,并将信息插入全局的索引表。

由于索引条数过多会带来一定的性能下降,如果一个网站的访问图片是 10 万数量级以内的范围并且不会大量产生 UGC 图片内容的情况下,则可以使用一个单独的索引表;如果网站的图片是海量的,则可以考虑根据图片路径的 md5[12] 来进行分库分表的存储,从而提高性能。

2.3　使用内存 cache 加速热点图片的存储

由于传统的机械硬盘在磁盘 I/O 时将耗费比内存多数万数量级的时间[13],所以高性能的离线图片服务

器必须有一个内存缓存服务。将每条存储请求过来时,首先将其存储在硬盘。当每次请求来到时,首先访问内存缓存服务,如果缓存中不存在则查询磁盘的同时将图片信息放入内存,为保持缓存的及时更新,每一条缓存记录都必须设定一个超时时间。内存和磁盘的大小应该根据图片服务的数量级而定。

2.4 实现与对比测试

文中采用对比实验来验证上述提出的图片离线下载功能服务器的实际效用。实验 1 通过传统的方式来提供 Web 服务,通过代理服务器调用一个 Web 页面生成服务来转发实际的 HTML 页面,该服务会从一个存有 100 张图片的样本中随机取出 10 张,生成一个 Web 页面。实验 2 通过使用 Nodejs 实现的图片离线服务器作为代理,使用 redis 作为图片索引的 NOSQL 存储,访问与实验 1 中相同的后端 Web 页面生成服务。图片 url 解析模块的核心代码如下:

```
var jsdom=require('jsdom').jsdom;
var fs=require("fs");
var nodegrass=require('nodegrass');
var iconv=require('iconv');
var jquery=fs.readFileSync("../../../lib/jquery.js").toS-
tring();
var mongoskin=require('mongoskin');
nodegrass.get(url,function(data,status,header){
jsdom.env({
html:data,
src:[jquery],
done:function(errors, window){
var $ = window.$;

$('body img').each(function(index){
var element=$(this);
var src=element.src;
var imgInCache=isImgInCache(src);
if(! imgInCache){
insertImgToCache(src);
} else {
var imgInBase64=imgInCache.data;
element.attr('src',imgInBase64);
}
});
window.close();
}
});
},'binary').on('error', function(e){
console.log("Got error: " + e.message);
}
);
```

实验 1 和实验 2 通过由 phantomjs 实现的测试工具来完成对比测试。phantomjs 是一个无界面的 webkit[14] 浏览器引擎,还有配套的 javascript API,它支持各种 WEB 标准技术,如 DOM 处理、CSS 选择器、JSON、canvas 以及 SVG。测试用例的核心代码 loadspeed.js 如下所示:

```
var page = require('webpage').create(),
system = require('system'),
t, address;

if (system.args.length === 1) {
console.log('Usage: loadspeed.js <some URL>');
phantom.exit();
}

t = Date.now();
address = system.args[1];
page.open(address, function(status) {
if (status !== 'success') {
console.log('FAIL to load the address');
} else {
t = Date.now() - t;
console.log('Loading time ' + t + ' msec');
}
phantom.exit();
});
```

针对实验 1 和实验 2,通过脚本每隔 1 s 运行 loadspeed.js 得到如下的实验数据,如表 2 所示。

表 2 传统 Web 图片加载和离线图片下载服务的时间对比

实验	实验次数	首次加载时间/ms	最长加载时间/ms	最短加载时间/ms	平均加载时间/ms
1	100	3 452	5 612	2 989	3 224
2	100	4 219	4 219	984	1 382

通过表 1 可以看出,在不开启离线图片服务的情况下,从第一次到第一百次总体时间都稳定在一定的区间内,波动较小,平均延时较开启离线图片下载服务的情况下高出 133.2%;而在开启图片离线下载服务的情况下,首次加载时间较传统 Web 图片加载略高,但是随着缓存的建立,图片离线下载服务的加载延时不断下载,当缓存完全建立后延时则稳定在一个较小的区间。在平均延时方面,开启图片离线下载服务的时延仅为传统 Web 图片加载时延的 42.86%,优化效果是明显的。

3 结束语

随着移动互联网时代的到来,图片服务的性能瓶

(下转第 79 页)

执行系统复位,phy_init_done 位有效后再执行正常读写操作。

若在系统复位后不配置自测试模式寄存器,则当 phy_init_done 位有效后,控制器不进入自测试模式,可以直接开始正常读写操作。

5　结束语

H.264 高清视频编码的大数据量高速存储问题,已成为制约高清实时编码的重要因素。文中所阐述的基于 DDR2 控制器的高清视频编码存储器接口的设计与实现,能高速处理多路数据的读写操作;多个仲裁器的设计能有效地调配读写资源;带有自校准和自测试逻辑的控制器能智能地优化读写数据的质量。

通过芯片的测试结果分析可知,该实现方案能够有效地解决大数据量的交互给编码系统造成的存储压力问题,能够流畅完成 1 920×1 080 分辨率下的实时高清编码。

参考文献:

[1] Marpe D, Wiegand T, Sullivan G J. The H.264/MPEG4 advanced video coding standard and its applications[J]. IEEE Communications Magazine, 2006, 44(8): 134-143.

[2] 孟　超,张曦煌. 基于嵌入式系统的图像采集与传输设计[J]. 计算机工程与设计,2008,29(17):4414-4416.

[3] 毕厚杰,王　建. 新一代视频压缩编码标准-H.264/AVC[M]. 北京:人民邮电出版社,2009.

[4] 马力妮,郑志辉,潘　峰. H.264/AVC 视频编码技术研究[J]. 计算机技术与发展,2008,18(7):163-166.

[5] 吴晓军,白世军,卢文涛. 基于 H.264 视频编码的运动估计算法优化[J]. 电子学报,2009,37(11):2541-2545.

[6] 李秋山,马　妍. H.264 帧内预测模式的快速选择算法[J]. 计算机工程与设计,2009,30(22):5136-5139.

[7] Wenger S. H.264/AVC over IP[J]. IEEE Transactions on Circuits Systems for Video Technology, 2003, 13(7): 645-656.

[8] Chen L, Cheng L J, Yu S Y. Accurate rate control method in transcoding[J]. Electronics Letters, 2004, 40(1): 16-18.

[9] Micron Technology. 256MbDDR2[S]. [s.l.]: Micron Technology, Inc., 2009.

[10] JEDEC. Association DDR2 SDRAM specification[S]. [s.l.]: JEDEC Solid State Technology Association, 2006.

[11] 褚晶辉,俞斯乐,鲁照华. 视频转换编码及其实现技术的研究[J]. 电子学报,2004,32(10):1678-1683.

[12] 干宗良,齐丽娜,朱秀昌. H.264 中基于先验预测的帧间编码模式选择算法研究[J]. 电子与信息学报,2006,28(10):1883-1887.

[13] Xilinx. Memory Interface Generator (MIG) user guide UG086 (v2.3)[S]. [s.l.]: Xilinx Inc, 2008.

(上接第 74 页)

颈在移动链路上会比在 PC 时代更多的放大,文中给出的图片优化方法能大大提高图片传输的效率和稳定性,在后 PC 时代与移动互联网时代都有广阔的应用前景。通过使用文中的改进方法并且在服务器端开启 GZIP 的情况下,可以明显减少网站在流量带宽上的开销,降低网站的成本;同时可以带来更快的页面加载速度,以达到更好的用户体验。

参考文献:

[1] 王伟军,孙　晶. Web2.0 的研究与应用综述[J]. 情报科学,2007,25(12):1907-1913.

[2] 熊　文,熊淑华,孙　旭,等. Ajax 技术在 Web2.0 网站设计中的应用研究[J]. 计算机技术与发展,2012,22(3):145-148.

[3] 覃秋密,韦永军,蒋家斌. CSS Sprites 提升网页加载速度的应用研究[J]. 电脑知识与技术,2011,7(27):6668-6670.

[4] 牛　津,杨　涛,王　林. 网页浏览器内核的比较研究[J]. 微计算机应用,2009,30(3):30-35.

[5] Fei Ge, Tian Liansheng, Sun Jinsheng, et al. Latency of FAST TCP for HTTP transactions[J]. IEEE Communications Letters, 2011, 15(11): 1259-1261.

[6] 祝　瑞,车　敏. 基于 HTTP 协议的服务器程序分析[J]. 现代电子技术,2012(4):117-119.

[7] Lee Jae-Yong, Hong Jong-Joon, Lee Kyoon-Ha. URI directory service agent model for WWW intelligent service[C]// Proc of TENCON 99. [s.l.]: [s.n.], 1999: 1403-1406.

[8] Xu Congfu, Chen Yafang, Chiew K. An approach to image spam filtering based on Base64 encoding and N-Gram feature extraction[C]//Proceedings of 22nd IEEE international conference on tools with artificial intelligence. Arras: IEEE, 2010: 171-177.

[9] Rauschert P, Klimets Y, Velten J, et al. Very fast GZIP compression by means of content addressable memories[C]//Proc of TENCON 2004. [s.l.]: [s.n.], 2004: 391-394.

[10] 赵玉伟,赵小雨,乔　木. 缓存技术在 B/S 架构信息系统中的应用[J]. 计算机工程,2008,34(1):233-235.

[11] 陈锡明,杨国纬. 一种用 B—树的最佳阶数组织内存索引文件的方法[J]. 小型微型计算机系统,1998,19(2):60-64.

[12] 张裔智,赵　毅,汤小斌. MD5 算法研究[J]. 计算机科学,2008,35(7):295-297.

[13] 陈　琼,张江陵,冯　丹. 一种提高磁盘阵列 I/O 性能的策略[J]. 小型微型计算机系统,2000,21(1):13-15.

[14] Hodovan R, Kiss A. Security evolution of the Webkit browser engine[C]//Proc of 14th IEEE international symposium on web systems evolution. [s.l.]: [s.n.], 2012: 17-19.

网页中图片离线加速下载方案的设计研究

作者： 宗平， 高斐， ZONG Ping， GAO Fei
作者单位： 宗平,ZONG Ping(南京邮电大学 海外教育学院,江苏 南京,210023)， 高斐,GAO Fei(南京邮电大学 计算机学院,江苏 南京,210023)
刊名： 计算机技术与发展 ISTIC
英文刊名： Computer Technology and Development
年，卷(期)： 2014(10)