NOTE-pipelining-970624

Network Performance Effects of HTTP/1.1, CSS1, and PNG

NOTE 24-June 1997

This version:
http://www.w3.org/TR/NOTE-pipelining-970624
$Id: Pipeline.html,v 1.47 1997/08/09 17:56:02 fillault Exp $
Latest version:
http://www.w3.org/TR/NOTE-pipelining
Authors:
Henrik Frystyk Nielsen, W3C, <frystyk@w3.org>,
Jim Gettys, Visiting Scientist, W3C, Digital Equipment Corporation, <jg@w3.org>,
Anselm Baird-Smith, W3C, <abaird@w3.org>,
Eric Prud'hommeaux, W3C, <eric@w3.org>,
Håkon Wium Lie, W3C, <howcome@w3.org>,
Chris Lilley, W3C, <chris@w3.org>

Status of This Document

This document is a NOTE made available by the W3 Consortium for discussion only. This indicates no endorsement of its content, nor that the Consortium has, is, or will be allocating any resources to the issues addressed by the NOTE. A list of current NOTEs can be found at: http://www.w3.org/TR/

Since NOTEs are subject to frequent change, you are advised to reference the above URL, rather than the URLs for NOTEs themselves. The results here are provided for community interest, though it has not been rigorously validated and should not alone be used to make commercial decisions. In addition, the exact results are obviously a function of the tests performed; your mileage will vary.

To appear in ACM SIGCOMM '97 Procedings.

Abstract

We describe our investigation of the effect of persistent connections, pipelining and link level document compression on our client and server HTTP implementations. A simple test setup is used to verify HTTP/1.1's design and understand HTTP/1.1 implementation strategies. We present TCP and real time performance data between the libwww robot [27] and both the W3C's Jigsaw [28] and Apache [29] HTTP servers using HTTP/1.0, HTTP/1.1 with persistent connections, HTTP/1.1 with pipelined requests, and HTTP/1.1 with pipelined requests and deflate data compression [22]. We also investigate whether the TCP Nagle algorithm has an effect on HTTP/1.1 performance. While somewhat artificial and possibly overstating the benefits of HTTP/1.1, we believe the tests and results approximate some common behavior seen in browsers. The results confirm that HTTP/1.1 is meeting its major design goals. Our experience has been that implementation details are very important to achieve all of the benefits of HTTP/1.1.

For all our tests, a pipelined HTTP/1.1 implementation outperformed HTTP/1.0, even when the HTTP/1.0

at least a factor of two, and sometimes as much as a factor of ten, in terms of packets transmitted. Elapsed time improvement is less dramatic, and strongly depends on your network connection.

Some data is presented showing further savings possible by changes in Web content, specifically by the use of CSS style sheets [10], and the more compact PNG [20] image representation, both recent recommendations of W3C. Time did not allow full end to end data collection on these cases. The results show that HTTP/1.1 and changes in Web content will have dramatic results in Internet and Web performance as HTTP/1.1 and related technologies deploy over the near future. Universal use of style sheets, even without deployment of HTTP/1.1, would cause a very significant reduction in network traffic.

This paper does not investigate further performance and network savings enabled by the improved caching facilities provided by the HTTP/1.1 protocol, or by sophisticated use of range requests.

Introduction

Typical web pages today contain a HyperText Markup Language (HTML) document, and many embedded images. Twenty or more embedded images are quite common. Each of these images is an independent object in the Web, retrieved (or validated for change) separately. The common behavior for a web client, therefore, is to fetch the base HTML document, and then immediately fetch the embedded objects, which are typically located on the same server.

The large number of embedded objects represents a change from the environment in which the Web transfer protocol, the Hypertext Transfer Protocol (HTTP) was designed. As a result, HTTP/1.0 handles multiple requests from the same server inefficiently, creating a separate TCP connection for each object.

The recently released HTTP/1.1 standard was designed to address this problem by encouraging multiple transfers of objects over one connection. Coincidentally, expected changes in Web content are expected to decrease the number of embedded objects, which will improve network performance. The cheapest object is one that is no longer needed.

To test the effects of some of the new features of HTTP/1.1, we simulated two different types of client behavior: visiting a site for the first time, where nothing is in the client cache, and revalidating cached items when a site is revised. Tests were conducted in three different network environments designed to span a range of common web uses: a local Ethernet (LAN), transcontinental Internet (WAN), and a 28.8 Kbps dialup link using the Point-to-Point Protocol (PPP).

In this paper, we present the final results, and some of the thought processes that we went through while testing and optimizing our implementations. Our hope is that our experience may guide others through their own implementation efforts and help them avoid some non-obvious performance pits we fell into. Further information, the data itself (and later data collection runs) can be found on the Web [25].

Changes to HTTP

HTTP/1.1 [4] is an upward compatible protocol to HTTP/1.0 [3]. Both HTTP/1.0 and HTTP/1.1 use the TCP protocol [12] for data transport. However, the two versions of HTTP use TCP differently.

HTTP/1.0 opens and closes a new TCP connection for each operation. Since most Web objects are small, this practice means a high fraction of packets are simply TCP control packets used to open and close a connection. Furthermore, when a TCP connection is first opened, TCP employs an algorithm known as slow start [11]. Slow start uses the first several data packets to probe the network to determine the optimal transmission rate. Again, because Web objects are small, most objects are transferred before their TCP connection completes the slow start algorithm. In other words, most HTTP/1.0 operations use TCP at its least efficient. The results have been

HTTP/1.1 leaves the TCP connection open between consecutive operations. This technique is called "persistent connections," which both avoids the costs of multiple opens and closes and reduces the impact of slow start. Persistent connections are more efficient than the current practice of running multiple short TCP connections in parallel.

By leaving the TCP connection open between requests, many packets can be avoided, while avoiding multiple RTTs due to TCP slow start. The first few packet exchanges of a new TCP connection are either too fast, or too slow for that path. If these exchanges are too fast for the route (common in today's Internet), they contribute to Internet congestion.

Conversely, since most connections are in slow start at any given time in HTTP/1.0 not using persistent connections, keeping a dialup PPP link busy has required running multiple TCP connections simultaneously (typical implementations have used 4 TCP connections). This can exacerbate the congestion problem further.

The "Keep-Alive" extension to HTTP/1.0 is a form of persistent connections. HTTP/1.1's design differs in minor details from Keep-Alive to overcome a problem discovered when Keep-Alive is used with more than one proxy between a client and a server.

Persistent connections allow multiple requests to be sent without waiting for a response; multiple requests and responses can be contained in a single TCP segment. This can be used to avoid many round trip delays, improving performance, and reducing the number of packets further. This technique is called "pipelining" in HTTP.

HTTP/1.1 also enables transport compression of data types so those clients can retrieve HTML (or other) uncompressed documents using data compression; HTTP/1.0 does not have sufficient facilities for transport compression. Further work is continuing in this area [26].

The major HTTP/1.1 design goals therefore include:

- lower HTTP's load on the Internet for the same amount of "real work", while solving the congestion caused by HTTP
- HTTP/1.0's caching is primitive and error prone; HTTP/1.1 enable applications to work reliably with caching
- end user performance must improve, or it is unlikely that HTTP/1.1 will be deployed

HTTP/1.1 provides significant improvements to HTTP/1.0 to allow applications to work reliably in the face of caching, and to allow applications to mark more content cacheable. Today, caching is often deliberately defeated in order to achieve reliability. This paper does not explore these effects.

HTTP/1.1 does not attempt to solve some commonly seen problems, such as transient network overloads at popular web sites with topical news (e.g. the Schumacher-Levy comet impact on Jupiter), but should at least help these problems.

This paper presents measured results of the consequences of HTTP/1.1 transport protocol additions. Many of these additions have been available as extensions to HTTP/1.0, but this paper shows the possible synergy when the extensions to the HTTP protocol are used in concert, and in with changes in content.

Range Requests and Validation

To improve the perceived response time, a browser needs to learn basic size information of each object in a page (required for page layout) as soon as possible. The first bytes typically contain the image size. To achieve better concurrency and retrieve the first few bytes of embedded links while still receiving the bytes for the master document, HTTP/1.0 browsers usually use multiple TCP connections. We believe by using range requests

HTTP /1.1 defines as part of the standard (and most current HTTP/1.0 servers already implement) byte range facilities that allow a client to perform partial retrieval of objects. The initial intent of range requests was to allow caching proxy to finish interrupted transfers by requesting only the bytes of the document they currently do not hold in their cache.

To solve the problem that browsers need the size of embedded objects, we believe that the natural revalidation request for HTTP/1.1 will combine both cache validation headers and an *If-Range* request header, to prevent large objects from monopolizing the connection to the server over its connection. The range requested should be large enough to usually return any embedded metadata for the object for the common data types. This capability of HTTP/1.1 is implicit in its caching and range request design.

When a browser revisits a page, it has a very good idea what the type of any embedded object is likely to be, and can therefore both make a validation request and also simultaneously request the metadata of the embedded object if there has been any change. The metadata is much more valuable than the embedded image data. Subsequently, the browser might generate requests for the rest of the object, or for enough of each object to allow for progressive display of image data types (e.g. progressive PNG, GIF or JPEG images), or to multiplex between multiple large images on the page. We call this style of use of HTTP/1.1 "poor man's multiplexing."

We believe cache validation combined with range requests will likely become a very common idiom of HTTP/1.1.

Changes to Web Content

Roughly simultaneously to the deployment of the HTTP/1.1 protocol, (but not dependent upon it), the Web will see the deployment of Cascading Style Sheets (CSS) [30] and new image and animation formats such as Portable Network Graphics (PNG) [20] and Multiple-image Network Graphics (MNG) [31].

In the scientific environment where the Web was born, people were generally more concerned with the content of their documents than the presentation. In a research report, the choice of fonts matters less than the results being reported, so early versions of HyperText Markup Language (HTML) sufficed for most scientists. However, when non-scientific communities discovered the Web, the perceived limitations of HTML became a source of frustration. Web page designers with a background in paper-based desktop publishing wanted more control over the presentation of their documents than HTML was meant to provide. Cascading Style Sheets (CSS) offer many of the capabilities requested by page designers but is only now seeing widespread implementation.

In the absence of style sheets, authors have had to meet design challenges by twisting HTML out of shape, for instance, by studding their pages with small images that do little more than display text. In this section of the study, we estimate how Web performance will be affected by the introduction of CSS. We will not discuss other benefits to be reaped with style sheets, such as greater accessibility, improved printing, and easier site management.

On the web, most images are in GIF format. A new image format, PNG, has several advantages over GIF. PNG images render more quickly on the screen and - besides producing higher quality, cross-platform images - PNG images are usually smaller than GIF images.

MNG is an animation format in the PNG family, which - along with other advantages - is more compact than animated GIF.

Prior Work

Padmanabhan and Mogul [1] show results from a prototype implementation which extended HTTP to support

connection and pipelining design. HTTP/1.1 primarily relies on pipelining rather than introducing new HTTP methods to achieve the performance benefits documented below. As this paper makes clear, both pipelining and persistent connections are needed to achieve high performance over a single HTTP connection.

Pipelining, or batching, have been successfully used in a number of other systems, notably graphics protocols such as the X Window System [15] or Trestle [16], in its original RPC based implementation.

Touch, Heidemann, and Obraczka [5] explore a number of possible changes that might help HTTP behavior, including the sharing of TCP control blocks [19] and Transaction TCP (T/TCP) [17], [18]. The extended length of deployment of changes to TCP argued against any dependency of HTTP/1.1 on either of these; however, we believe that both mechanisms may improve performance, independently to the improvements made by HTTP/1.1. T/TCP might help reduce latency when revisiting a Web server after the server has closed its connection. Sharing of TCP control blocks would primarily help HTTP/1.0, however, since the HTTP/1.1 limits the number of connections between a client/server pair.

In independent work, Heidemann [7] describes the interactions of persistent connections with Nagle's algorithm. His experience is confirmed by our experience described in this paper, and by the experience of one of the authors with the X Window System, which caused the original introduction of the ability to disable Nagle's algorithm into BSD derived TCP implementations.

Simon Spero analyzed HTTP/1.0 performance [6] and prepared a proposal for a replacement for HTTP. HTTP/1.1, however, was constrained to maintain upward compatibility with HTTP/1.0. Many of his suggestions are worthwhile and should be explored further.

Style sheets have a long history in the Web [30]. We believe that the character of our results will likely be similar for other style sheet systems. However, we are not aware of any prior work investigating the network performance consequences of style sheets.

Test Setup

Test Web Site

We synthesized a test web site serving data by combining data (HTML and GIF image data) from two very heavily used home pages (Netscape and Microsoft) into one; hereafter called "*Microscape*". The initial layout of the Microscape web site was a single page containing typical HTML totaling 42KB with 42 inlined GIF images totaling 125KB. The embedded images range in size from 70B to 40KB; most are small, with 19 images less than 1KB, 7 images between 1KB and 2KB, and 6 images between 2KB and 3KB. While the resulting HTML page is larger, and contains more images than might be typical, such pages can be found on the Web.

First Time Retrieval Test

The first time retrieval test is equivalent to a browser visiting a site for the first time, e.g. its cache is empty and it has to retrieve the top page and all the embedded objects. In HTTP, this is equivalent to 43 *GET* requests.

Revalidate Test

This test is equivalent to revisiting a home page where the contents are already available in a local cache. The initial page and all embedded objects are validated, resulting in no actual transfer of the HTML or the embedded objects. In HTTP, this is equivalent to 43 *Conditional GET* requests. HTTP/1.1 supports two mechanisms for cache validation: *entity tags*, which are a guaranteed unique tag for a particular version of an object, and date stamps. HTTP/1.0 only supports the latter.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.