Using the HTTP/2 Server Push Technology to Reduce Web Page Loading Latency

Petrov, Pavel and Petrova, Stefka

University of Economics - Varna

2 December 2016

Using the HTTP/2 Server Push Technology to Reduce Web Page Loading Latency

Pavel Petrov, Stefka Petrova

University of Economics - Varna, Varna, Bulgaria
petrov@ue-varna.bg , s.petrova@ue-varna.bg

Abstract. The paper reviews the factors affecting latency when loading web pages. It concludes that nowadays network bandwidth plays an increasingly small role about the latency. An empirical research was made to find out how much the latency is reduced when using the HTTP/2 server push technology. An average by volume and by content web page is used in the tests, which includes different versions of the protocol HTTP - HTTP/1.1, HTTPS/1.1 and HTTPS/2. Experimental software is created and an external program to simulate network latency is used.

Keywords. Server Push, HTTP, HTTP/1.1, HTTP/2, Hypertext Transfer Protocol, web page, latency, lag, web server, web client.

1. Introduction

As it is well known, the HTTP protocol works at the application level according to the model Open Systems Interconnection (OSI) ISO [1] and the exchange of data between client and server is done in classical request-response manner. The original protocol HTTP, known as HTTP/0.9 [2], is maximally simplified and was suitable for the technological level in the 90 years of the last century. Currently servers that support virtual hosts based on names should not support HTTP/0.9, according to the latest standard for HTTP/1.1 [3] and in fact most web servers today do not support it. Messages exchanged between the two sides look like as those shown in Table 1. The network connection between client and server is interrupted after every response, which requires when sending subsequent requests to open a new network connection.

In the next version of the Protocol - HTTP/1.0 [4], established in 1993 and standardized by IETF in 1996, the request became more complicated by adding new parts - the first line stays, it became so-called start-line, and header and body are added. On the start line, unless the keyword GET, can be used other words indicating the manner in which the server can handle the request. These keywords are known as methods and in HTTP 1.0 there are GET, POST and HEAD methods. Additionally, after the address of the document the protocol and version should be specified.

GET method remains the primary means of requesting documents, but it has no body. In contrast, the POST method has a body and it can convey large amounts of data to the server. HEAD method is similar to the GET, except that the server does not need to return the body of the response, only the header. This is most commonly used for validation of hyperlinks, availability of resources or checking when the document was last modified.

Server-side response in HTTP/1.0 compared to HTTP/0.9 is also complicated and consists not just of the requested document, but also status line and header are added. The status line begins with data for protocol and version, followed by the code status - a three-digit number, and possibly status message. The header contains some system data concerning the

information contained in the message body. The network connection between client and server is interrupted again after each response.

Table 1. Sample client request and server response using different versions of HTTP

Version	Sample Request	Sample Response
HTTP/0.9	GET / [\n]	<html> <head><title>Sample Title</title></head> <body>Sample Web Page Content</body> </html>
HTTP/1.0	GET /img.jpg HTTP/1.0 Referer: /index.html [\n\n]	HTTP/1.0 404 Not Found Content-Type: text/html Content-Length: 17 [\n\n] Object not found.
HTTP/1.1	GET /img.jpg HTTP/1.1 Host: www.example.com User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:47.0) Gecko/20100101 Firefox/47.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: bg,en-US;q=0.7,en;q=0.3 Accept-Encoding: gzip, deflate Connection: keep-alive [\n\n]	HTTP/1.1 404 Not Found Accept-Ranges: bytes Cache-Control: max-age=604800 Content-Encoding: gzip Content-Length: 606 Content-Type: text/html Date: Sun, 23 Oct 2016 08:51:37 GMT Etag: "359670651+gzip" Expires: Sun, 30 Oct 2016 08:51:37 GMT Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT Server: ECS (ewr/1445) Vary: Accept-Encoding X-Cache: HIT x-ec-custom-error: 1 [\n\n] [the binary data which follows in the body of the response are not presented here]

Unlike HTTP/0.9 which lacks header, in HTTP/1.0 it plays a very important role, as there are a lot of useful information: indication of what type of data are sent in the body (field Content-Type), what is the size (field Content-Length), how the data are encoded (field Content-Encoding), when the document was changed (field Last-Modified) and others. Messages exchanged between the two sides look like those shown in Table 1.

The next version of the protocol HTTP - version HTTP/1.1 was established in 1997 (RFC 2068), renewed in 1999 (RFC 2616) [5] and is undergoing final changes in 2014 (RFC 7230 - RFC 7240). The most interesting pointc are: the request and the response retaining the basic structure of 3 parts; in the request header is added mandatory field "Host:" to allow the usage of virtual hosts; provides an opportunity to maintain open network connection (Field Connection: Keep-Alive) for extended periods of time and use it for other requests- responses; ability to send just part of a file; the number of codes for the status increases significantly; possible fields in request headers and response headers also increased significantly. Messages exchanged between the two sides look like those shown in Table 1.

As was shown, the HTTP protocol to version 1.1 goes through evolutionary development intended to enrich its capabilities while maintaining somewhat compatibility with previous versions, as far as possible. Because the protocol is textual, it can be easily tracked, especially the contents of system parts - request header and response header. The

body, which in most cases constitutes the essence and contents of the data exchanged, is possible to be compressed. Unfortunately this greatest advantage - the textual nature of the protocol, is its biggest drawback when talking about reducing the volume of transmitted data.

2. Capabilities of HTTP/2 to Reduce Latency and Lag While Loading Web Pages

Latency is a general term with a broader meaning in informatics, meaning the time delay in the execution of an operation. In terms of computer network latency incorporates latency that occurs during data transmission, processing and presentation to the end user. Often latency was called lag when it comes to overall subjective feeling of latency in interactive work with the system running in online mode.

For latency contribute many independent from each other factors and it can be represented as the sum of the time for preparation of the message by the client side plus the time it takes a signal to travel through the physical media (fiber optic cables, radio waves, electrical signals, etc.) plus time for signal processing in the intermediate devices plus the time required for processing the message from the server side, generating response, sending it back to the client and again plus time for way back from the server to the client, plus the time required by client to process and eventually visualize the response.

In the computer network in which there is relatively low traffic the latency will depend mainly from the speed of propagation of the signal, which is limited by the laws of physics - the speed of light in a vacuum (about 300000 km/s). In case of LAN where client, server and intermediate devices are relatively closely located, the latency can be as low as 1-2 ms. In an environment of a global network where client and server are at large distances from each other, the latency may exceed 500 ms.

Not like this is the issue where the computer network has a relatively large traffic close to the maximum throughput of the bandwidth. In such cases, intermediate devices start buffering packets or start seeking for other routes and unfortunately part of the data may be lost. So a major share of latency starts to play another factor - the processing time in the intermediate devices. Particularly unfavorable impact of lag, as a subjective feeling of latency, is data packets loss. This loss can occur due to various reasons and cannot directly related to latency, but for whatever reason a data packet is lost the subjective feeling of the user is not good. In such cases, when eliminating the cause of packet loss, the lag is reduced.

We can summarize that the determinants of lag are: bandwidth of the computer network, its current load, the distance between the two sides of the communication link, the packets loss problem, and the ability to quickly process data from the server and from the client. In recent years, bandwidth plays an increasingly small role in reducing lag. Data from studies worldwide show that the average speed is about 6,3 Mbps per connection, which means that many users have access to high-speed Internet connection and bandwidth no longer is the major limiting factor while loading web pages [6]. Accordingly, other factors have a greater impact on reducing latency and lag. One of them is the protocol.

The main issue before the HTTP/2 is that the precedents are synchronous. They require the client to wait the server to return the entire response and after then may submit a new request. In this mode of operation if the server slowly generates any of the resources, it effectively blocks all subsequent communication. To overcome this problem most browsers simultaneously open multiple network connections to the server in order to receive multiple resources simultaneously (for example Google Chrome opens simultaneously 6 network connections). HTTP version 2 is trying to solve the problems associated with latency by optimizing the way the resources are requested and send. It differs significantly from protocol

versions 0.9/1.0/1.1 - it is not synchronous, but asynchronous; not textual, but binary; it using only one TCP/IP connection to the domain through which performs multiplexing, i.e. transmission of multiple data streams simultaneously. The latter is achieved as each pair of request-response is associated with its own stream and therefore the data must be sent divided into individual frames [7]. Frames associated with a certain flow and thereby through a network connection may be transmitted to a plurality of data streams. The flows are relatively independent of each other, so blocking occurred of any response or application does not interfere with the other streams. Thus multiple requests-responses can be executed simultaneously and streams can be prioritized.

The protocol HTTP/2 supports the ability to compress headers. This is not done by classical algorithms for data compression, but through organizational technique, the use of which provides no resending for header fields already have been sent. For this purpose the client and the server support tables of sent and received fields in the headers and their values. Another important feature of HTTP/2 is the ability the server to send resources that were not explicitly requested by the client (server push technology). These resources are cached on the client side for future use. The server can start sending these resources as soon as a connection has been established, without even waiting for the client to send a request. In this operating mode it is possible to increase unnecessary network traffic, but as overall the web pages will load faster.

3. Reducing the Lag Using the HTTP/2 Server Push Technology

According to worldwide data from httparchive.org for the first 500 thousand most visited sites for a period of one year to 15/10/2016 (according to Alexa rank), to fully load and display an average web page it takes about 102 requests for resources, and the average volume a fully loaded web page is about 2,3MB [8].

In order to empirical research the lag reducing when loading web pages using HTTP/2 server push technology, we created a test unit through which to identify lags at different levels of network latency. An important feature when using HTTP/2 is that modern browsers support version 2 only if the connection is encrypted, although this is not specifically required under the standard.

During the testing we used components with the following characteristics: operating system - Windows 7 32 bits; processor - i5-2430M; RAM - 4GB; web server - Apache 2.4.18; web browsers - Google Chrome 54.0.2840 and Mozilla Firefox 47.0.1 (the latest versions at October 2016).

Since the client and the server are physically located together on one machine to simulate network latency we are using Google Chrome built capabilities.

When setting up a web server for handling HTTP/2 the instructions from the documentation of Apache are followed [9], namely - in the configuration file httpd.conf a directive of charging mod_http2 is activated, add a directive to switch from ver. 1.1 to ver. 2 is used:

LoadModule http2_module modules/mod_http2.so

Protocols h2 h2c http/1.1

Directives SSLCipherSuite and SSLProtocol were changed to support SSL v2 and v3 as follows:

SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-

AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!3DES:!MD5:!PSK

SSLProtocol All -SSLv2 -SSLv3

In conducting tests we following the next approach: the web server is stopped; the appropriate settings are carried out and then start again; the resource is loaded three times in order to "warm up" the system and then into the new "incognito" windows the resource is freshly loaded; we repeated the tests in both browsers three times, then times are averaged for each of them.

The program module that generates a web page which in turn pushes additional resources is performed by the interpreter to PHP 7 and is as follows:

```
<?php
$PUSH = 1; //0 - Push Off; 1 - Push On
$files = 102;
$size = 23; //KB
$latency = 0; // milliseconds
if(!isset($_GET['t'])) {
        if($PUSH)
                for($i=0; $i<$files; $i++)
                        header("link: <$_SERVER[PHP_SELF]?t=$i>; rel=preload; as=script", false);
        print<<<EOT
<html>
<head>
<script>
var begin=0, end=0, delta = 0;
function print(s) {
        document.getElementById("txt").innerHTML += (Date.now()) + ":" + s + "<br />";
}
document.addEventListener("DOMContentLoaded", function(event) {
        begin = Date.now();
        print("DOMContentLoaded");
});
window.onload = function () {
        end = Date.now();
        print("load");
        print("delta="+(end-begin));
}
</script>
</head>
<body>
<div id="txt"></div>
EOT;
        usleep($latency*1000);
        echo "<script>print('START: $files files, $size KB');</script>\n";
        for($i=0; $i<$files; $i++) echo "<script src='?t=$i' async></script>\n";
        print<<<EOT
```

```
</body>
</html>
EOT;
} else {
        usleep($latency*1000);
        echo "print('$_GET[t]'); tmp = '" . str_repeat("1234567890", $size*100) . "';";
}
?>
```

In the variables $files and $size accordingly can be set how many additional resources need to be requested to full visualization of the website and what is their size. Additionally, the variable $latency can be set, which can simulate the performance of more processing on the server side while returning the resource. In conducting the tests we set the values of these variables as close as possible to reproduce the average web page, according to the aforementioned statistical survey from httparchive.org - a web page that loads additional 102 resources totaling 2,3MB.

In Table 2 shows the time interval in seconds from the time of loading of an average statistical web page until the load of all the resources necessary to visualizing (lag) using push and not using push technology.

Table 2. Lag depending on the protocol and push technology

Technology	**Browser**	
	Google Chrome 54.0.2840	**Mozilla Firefox 47.0.1**
HTTP/1.0	0,5 s	0,5 s
HTTP/1.1	0,4 s	0,6 s
HTTPS/1.1	0,5 s	0,8 s
HTTPS/2 no PUSH	0,3 s	0,2 s
HTTPS/2 with PUSH	0,2 s	0,1 s

We think that in the data provided essential is not the absolute values, but rather ratios between them. In some of the tests better performs one of the browsers and this is not essential. The more interesting is that in general HTTP/2 has speed advantage compared to his predecessors. Using push technology also give a speed gain compared to scenario not using push.

4. Conclusion

HTTP protocol goes a long way in its development. The new version 2 has many advantages and should be adopted in practice fast. Our experiments shown that reducing the lag is achievable goal by using HTTP/2 server push technology.

References

1. ISO/IEC 7498-1:1994, Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model, <http://www.ecma-international.org/activities/Communications/TG11/s020269e.pdf>

2. T. Berners-Lee, The Original HTTP as defined in 1991, <https://www.w3.org/Protocols/HTTP/AsImplemented.html>
3. R. Fielding et al., RFC 7230, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", 2014, <https://tools.ietf.org/rfc/rfc7230.txt>
4. T. Berners-Lee et al., RFC 1945, "Hypertext Transfer Protocol - HTTP/1.0", 1996, <http://www.rfc-editor.org/rfc/rfc1945.txt>
5. R. Fielding, et al., RFC 2616, Hypertext Transfer Protocol - HTTP/1.1, 1999, <https://tools.ietf.org/rfc/rfc2616.txt>
6. P. Petrov et al., Opportunities for Using the Protocol HTTP/2 to Reduce Lag When Loading Web Applications, Izvestia, Journal of the Union of Scientists - Varna, Economic Sciences Series, Issue: 2, 2016, p.160. <http://www.su-varna.org/izdanij/2016/ikonom-2-016/p%20155-165.pdf>
7. M. Belshe, R. Peon, M. Thomson, RFC 7540, Hypertext Transfer Protocol Version 2 (HTTP/2), <https://tools.ietf.org/rfc/rfc7540.txt>
8. Trends, <http://httparchive.org/trends.php?s=All&minlabel=Oct+15+2015&maxlabel=Oct+15+2016>
9. Apache HTTP Server Version 2.4, HTTP/2 guide, <https://httpd.apache.org/docs/2.4/howto/http2.html>
10. Apache HTTP Server Version 2.4, Environment Variables in Apache, Special Purpose Environment Variables, <http://httpd.apache.org/docs/2.4/env.html#special>