November 26, 2016

Mechanism For Measuring XHR Client Side Processing

Olga Shershevsky  
*Hewlett Packard Enterprise*

Avi Kabizon  
*Hewlett Packard Enterprise*

Nir Pluderman  
*Hewlett Packard Enterprise*

Follow this and additional works at: [http://www.tdcommons.org/dpubs_series](http://www.tdcommons.org/dpubs_series)

Recommended Citation
Shershevsky, Olga; Kabizon, Avi; and Pluderman, Nir, "Mechanism For Measuring XHR Client Side Processing", Technical Disclosure Commons, (November 26, 2016)  
Mechanism For Measuring XHR Client Side Processing

Abstract
XHR is a mechanism used to send HTTP/S requests to a web server, and use the response to alter the current document without loading a new page. XHR performance is an important metric for Web site performance monitoring. A significant metric is the client side processing of the response, because of its potential for isolating a pure client side performance problem. The disclosed technique provides a unique way of measuring this metric.

Description
This disclosure relates to the field of performance measurement.

A technique is disclosed that provides a unique way of measuring XHR performance of web sites.

A Web application is not what it used to be some years ago. Today it’s usually a rich Single Page Application, and most of the application’s logic is done on the Client side, rather than on the Server side. One of the major building blocks of a modern Web site/application is the Ajax set of technologies, and it’s based on the XHR mechanism.

XHR is a mechanism used to send HTTP/S requests to a web server, and use the response to alter the current document without loading a new page. XHR performance is a very important metric for Web site performance monitoring. XHR mechanism consists of three major parts: (1) the request as it’s sent through the network + the response as it’s received through the network. (2) server side processing of the request; and (3) client side processing of the response.

When monitoring the Web site’s client side, a significant metric is the client side processing of the response, because of its potential for isolating a pure client side performance problem, such as heavy JavaScript processing, major DOM manipulations, prolonged animations, etc. Therefore while monitoring a Web site, it’s very important to measure its XHRs. Network and Server parts of the XHR are relatively easy to measure using existing monitoring tools. But because most of the logic is done on the Client side, and in many cases the application’s owner and developers are looking for performance analysis on the Client side, measuring the XHR on the Client side is becoming more important.

According to the present disclosure, a technique is provided which includes a mechanism to measure XHR processing on the Client side which does not presently exist in monitoring tools. This mechanism may be included as part of a full solution for monitoring the user experience for web sites/applications.

First, XHR is initiated, sent, and the response from the server is received and processed, in the following manner:
- Xhr.open is called - This initializes the request.
- Xhr.send is called - This sends the request to the server.
- The response is received by using the XMLHttpRequest.onreadystatechange property, which contains the event handler (the callback) to be called when the readystatechange event is fired. This happens every time the readyState property of the XMLHttpRequest changes.

Since JavaScript is a single-threaded language, the callback execution time affects the User experience, and prevents the Browser from performing other UI tasks during this time. Therefore the total time it took to execute the callback is a valuable piece of information about how this XHR affects the End User.

In order to measure the entire response processing time, the readystatechange event lifecycle is monitored. The XMLHttpRequest.onreadystatechange behavior is unique, since it’s not an ordinary event listener which is called only once, when the event occurs.

According to the XHR spec, the readystatechange event can be fired multiple times, because it’s triggered on every XMLHttpRequest.readyState property change. It can have the following values:

0 UNSENT open() has not been called yet.
1 OPENED send() has been called.
2 HEADERS_RECEIVED send() has been called, and headers and status are available.
3 LOADING Downloading; responseText holds partial data.
4 DONE The operation is complete.

The disclosed technique measures the duration of the callback in each of the readystatechange occurrences, which happens for each readystate value change. It sums up all callback durations, which gives the XHR client processing time.

The mechanism begins by overriding the XHR API, in the following way:

```javascript
var originalOpen = xhr.constructor.prototype.open;
xhr.constructor.prototype.send = function (data) {
    ...
    wrapReadyStateFunction(this);
    ......
    originalSend.apply(this, arguments);
    ......
}
```

The wrapReadyStateFunction function has handling for each of the readystatechange statuses.

- For each of the statuses (2,3 and 4), the start time is saved. For status 3 the callback can be called multiple times. This can occur, for example, if the server is
streaming the data in chunks. Therefore a list is established that stores the status 3 starting times.

- For each of the statuses, the original monitored application’s callback is called.
- After the callback completion, the end time for each of the statuses is stored. For status 3, a list of end times is stored.
- When status=4 is identified, it means that the XHR response is completed. At this stage the duration of each time the callback was called is calculated, for each of the states, as follows:

\[
\text{totalXhrProcessing} = (\text{status2End} - \text{status2Start}) + \text{sum}(\text{status3End} - \text{status3Start}) + (\text{status4End} - \text{status4Start})
\]

The totalXhrProcessing value is the performance metric we are looking for – the XHR Client processing time.

This disclosed technique advantageously opens up a whole new dimension of XHR client side processing performance for developers to dive into for investigating performance issues which their customers encounter every day.

Disclosed by Olga Shershevsky, Avi Kabizon, and Nir Pluderman, Hewlett Packard Enterprise