Tracking Performance of the Web with HTTP Archive
6/13/2018

Paul Calvano
Senior Web Performance Architect
@paulcalvano
pacalvan@akamai.com
About Me

- Web Performance Architect @ Akamai
- HTTP Archive / BigQuery Addict :)
- Working on #WebPerf since 2000
- https://paulcalvano.com
- @paulcalvano on Twitter
The HTTP Archive Tracks How the Web is Built.

httparchive.org
HTTP Archive + BigQuery = Web Performance Answers
By Ilya Grigorik on June 20, 2013

HTTP Archive is a treasure trove of web performance data. Launched in late 2010, the project crawls over 300,000 most popular sites twice a month and records how the web is built: number and types of resources, size of each resource, whether the resources are compressed or marked as cacheable, times to render the page, time to first paint, and so on - you get the point.

The HTTP Archive site itself provides a number of interesting stats and aggregate trends, but the data on the site only scratches the surface of the kinds of questions you can ask! To satisfy your curiosity, all you need to do is download and import ~400GB of raw SQL/CSV data. Easy, right? Yeah, not really. Instead, wouldn't it be nice if we had the full dataset of all the HTTP Archive data to query on demand, and with ad-hoc questions?

Google BigQuery + HTTP Archive
How it Works

- Alexa’s top 500,000 websites
  - Home pages
  - Desktop and emulated mobile
  - Increasing to 1,000,000 soon!

- Powered by WebPageTest
  - Records HAR trace
  - Executes custom metrics
  - Records Lighthouse audits

- httparchive.org
  - Trends and stats
  - Discussion forum

- BigQuery and Cloud Storage
  - Queryable database
  - Raw HARs
HTTP Archive Pipeline

- Internet Archive
- 500K
- Biweekly
- 230 WebPageTest
- BQ
- GCS
Total Kilobytes
The sum of transfer size kilobytes of all resources requested by the page.

See also: Page Weight
HTTPS Requests

The percent of all requests in the crawl whose URLs are prefixed with https.

DESKTOP
63.5%
△3868.8%

MOBILE
64.3%
△2043.3%

Timeseries of HTTPS Requests

Source: httparchive.org

From Nov 15, 2010 To May 1, 2018
Time to Consistently Interactive

The number of seconds from the time the navigation started until the page is fully interactive and the network is idle. This metric comes from Lighthouse as is only available in mobile tests.

**Median Mobile**

14.6 seconds

▲ 20.7%

Timeseries of Time to Consistently Interactive

Source: httparchive.org

From Nov 15, 2010 To May 1, 2018
The tip of the databerg

Curated stats/trends

Raw data
### A Peek Inside the Databerg...

<table>
<thead>
<tr>
<th>DataSet</th>
<th>Description</th>
<th>Size (GB)</th>
<th>Rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>summary_pages</td>
<td>Summary of all Desktop and Mobile Pages</td>
<td>~340MB</td>
<td>Desktop: ~460K</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mobile: ~450K</td>
</tr>
<tr>
<td>summary_requests</td>
<td>Summary of all HTTP Requests for Desktop and Mobile</td>
<td>~45 GB</td>
<td>Desktop: ~48 million</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mobile: ~44 million</td>
</tr>
<tr>
<td>pages</td>
<td>JSON-encoded parent document HAR data</td>
<td>~5 GB</td>
<td>Desktop: ~460K</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mobile: ~450K</td>
</tr>
<tr>
<td>requests</td>
<td>JSON encoded subresource HAR data</td>
<td>~290 GB</td>
<td>Desktop: ~48 million</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mobile: ~44 million</td>
</tr>
<tr>
<td>response_bodies</td>
<td>JSON encoded response bodies for textual subresources</td>
<td>~915 GB</td>
<td>Desktop: ~18 million</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mobile: ~14 million</td>
</tr>
<tr>
<td>lighthouse</td>
<td>JSON encoded Lighthouse Report. Mobile only</td>
<td>~140 GB</td>
<td>Mobile: ~450K</td>
</tr>
</tbody>
</table>

* rows and size stats are based on 5/15/18 run
Who uses the HTTP Archive?

- Scholars
- Community
- Industry
In this article we utilize the httparchive.org [9] publicly available dataset of captured web performance metrics ...

Desktop and mobile web page comparison: characteristics, trends, and implications
IEEE Communications Magazine (Volume: 52, Issue: 9, September 2014)

Recent stats from httparchive.org show that the top 300K URLs in the world need on average 38(!) TCP connections to display the site ...

HTTP2 explained

We make extensive use of the [...] data available at HTTP Archive to expose the characteristics of 3rd Party assets embedded into the top 16,000 Alexa webpages ...

Are 3rd Parties Slowing Down the Mobile Web?
The web community uses HTTP Archive to answer questions about the state of the web.

**Eric Lawrence**
@ericlaw

The HTTP Archive is pretty awesomesauce. I don't even know what I'm doing and I can still use it.

**Domains serving brotli:**

I've been tracking the size and composition of the top million websites (thanks to the HTTP Archive) since 2011. Back in 2012, it was huge news when the average page exceeded 1 MB. At the time, people were incredulous... and even outraged. And they almost immediately began to speculate about when the average page would crack the 2 MB barrier. It looks like the speculation can end now.
Remember when the average page size exceeded Doom?

THE AVERAGE WEBPAGE IS NOW THE SIZE OF THE ORIGINAL DOOM

A compressed copy of the installer for the shareware version of *Doom* takes up about 2.39MB of space. Today’s average webpage, meanwhile, requires users to download about 2.3MB worth of data, according to HTTP Archive, a site that tracks website performance and the technologies they use.
Industrial tools use HTTP Archive data for calibration

Aditya Vohra @adityavohra24 · Jul 13
@TheLarkInn @addyosmani Hello! Are the studies Webpack bases its default performance budgets on public? Curious!

Addy Osmani @addyosmani · Jul 13
Open data: @HTTPArchive averages suggest sites ship ~420KB of JS. Newer numbers for medians are ~215KB. Webpack budget default is still higher

Sean @Seattle
@TheLarkInn

Replinging to @addyosmani @adityavohra24 @HTTPArchive

Thanks Addy! Yes we believe that it's a really sensible default.

10:56 AM · 13 Jul 2017

3 Likes
Is HPACK static table fit for today’s web?

LPardue

Dec ’17

HPACK is a header compression technique that can be used with HTTP/2. RFC 7541 defined a "Static table", used by all implementations, which encodes 61 "common" HTTP headers. This was measured during the design process.

I am wondering if the picture has changed with today’s web. This could be analyzed by listing HTTP header usage by frequency and coming up with a compression ratio during a subsequent analysis step. Other topics take a deep dive on particular headers but I’m interested in a broad look.

Some ideas for queries that might expose the popularity of request and response header fields (individual tables for request and response):

1. Unique key-value pairs are counted carefully (e.g. Accept-Encoding: gzip and Accept-Encoding: br; gzip are counted as different items).
2. Frequency of header field keys are counted.
3. Frequency of header field values are counted.

I’m not familiar with BigQuery and was hoping someone might be kind enough to get me off to a start.

HTTP Archive is used for emerging Internet Standards
Case Study #1

Compression

1. Updating Akamai Gzip Defaults
2. Brotli Compression
Last Mile Acceleration (LMA)

- Akamai feature to gzip compress content at the CDN edge
  - Helps out when origins do not compress certain resources
- Compression is based on HTTP Content Type
- Old defaults were not sufficient and usually required updating...

<table>
<thead>
<tr>
<th>LAST MILE ACCELERATION</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply to content with Content-type:</td>
<td>text/html*, application/x-javascript*, text/css*</td>
</tr>
<tr>
<td>Disable for Old Browsers:</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- We updated this a few years ago, using HTTP Archive data
SELECT mimeType, count(*) total,
    SUM(IF(resp_content_encoding = "gzip",1,0)) gzip,
    SUM(IF(resp_content_encoding = "deflate",1,0)) deflate,
    SUM(IF(resp_content_encoding IN("gzip","deflate"),0,1)) NoCompression,
    ROUND(
        SUM(
            IF(resp_content_encoding IN("gzip", "deflate"),1,0)
        ) / COUNT(*),2) CompressedPercentage
FROM httparchive.summary_requests.2018_05_15_desktop
GROUP BY mimeType
HAVING total > 1000
ORDER BY gzip DESC
Some New LMA Defaults

- text/javascript
- font/ttf
- application/javascript
- text/xml
- application/json
- application/xml
- ...

Many Content-Types that did not match the original defaults!
Modern Set of Last Mile Acceleration Defaults

**Content Compression**

Criteria: Match All

**If**

- Content Type
- is one of:
  - text/*
  - application/javascript
  - application/x-javascript
  - application/javascript*
  - application/json
  - application/x-json
  - application/*+json
  - application/*+xml
  - application/text
  - application/vnd.microsoft.icon
  - application/vnd.ms-fontobject
  - application/x-font.tff
  - application/x-font-opentype
  - application/x-font-truetype
  - application/xmlfont/eot
  - application/xml
  - font/opentype
  - font/otf
  - font/eot
  - image/svg+xml
  - image/vnd.microsoft.icon

**Behaviors**

Last Mile Acceleration (Gzip Compression)

Compress Response: Always
What About Brotli?

- New compression algorithm developed by Google researchers
- 5% - 25% Reduction over Gzip Compression
- Supported by most browsers

Let’s extend the previous query to include Brotli compression
## Compression Stats - gzip and brotli

```sql
SELECT mimeType, count(*) total,
    SUM(IF(resp_content_encoding = "gzip",1,0)) gzip,
    SUM(IF(resp_content_encoding = "br",1,0)) brotli,
    SUM(IF(resp_content_encoding = "deflate",1,0)) deflate,
    SUM(IF(resp_content_encoding IN("gzip","deflate","br"),0,1)) NoCompression,
    ROUND(
        SUM(
            IF(resp_content_encoding IN("gzip", "deflate", "br"),1,0)
        ) / COUNT(*),2) CompressedPercentage,
    ROUND(
        SUM(
            IF(resp_content_encoding = "br",1,0)
        ) / COUNT(*),2) BrotliCompressedPercentage
FROM httparchive.summary_requests.2018_05_15_desktop
GROUP BY mimeType
HAVING total > 1000
ORDER BY brotli DESC
```

Examining Compression By Content Type - With Brotli

<table>
<thead>
<tr>
<th>Row</th>
<th>mimeType</th>
<th>total</th>
<th>gzip</th>
<th>brotli</th>
<th>deflate</th>
<th>NoCompression</th>
<th>CompressedPercentage</th>
<th>BrotiCompressedPercentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>text/css</td>
<td>3426047</td>
<td>2298021</td>
<td>550668</td>
<td>2637</td>
<td>577721</td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>text/html</td>
<td>5215665</td>
<td>1772140</td>
<td>514270</td>
<td>1999</td>
<td>2927376</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>application/x-javascript</td>
<td>2906516</td>
<td>1515229</td>
<td>510954</td>
<td>2141</td>
<td>268082</td>
<td>0.88</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>application/javascript</td>
<td>4754612</td>
<td>3614446</td>
<td>255098</td>
<td>871</td>
<td>884197</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>text/javascript</td>
<td>3472708</td>
<td>2657038</td>
<td>132750</td>
<td>821</td>
<td>382099</td>
<td>0.89</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>image/gif</td>
<td>5401125</td>
<td>167431</td>
<td>91083</td>
<td>1094</td>
<td>5141537</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>image/svg+xml</td>
<td>468138</td>
<td>281031</td>
<td>18312</td>
<td>126</td>
<td>166597</td>
<td>0.64</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>application/json</td>
<td>760947</td>
<td>427649</td>
<td>14352</td>
<td>504</td>
<td>318442</td>
<td>0.58</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>image/jpeg</td>
<td>10415661</td>
<td>402845</td>
<td>7129</td>
<td>556</td>
<td>10005121</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>image/x-icon</td>
<td>219103</td>
<td>68164</td>
<td>5275</td>
<td>12</td>
<td>145712</td>
<td>0.34</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>text/xml</td>
<td>93819</td>
<td>64411</td>
<td>5024</td>
<td>28</td>
<td>24355</td>
<td>0.74</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>application/font-woff</td>
<td>120747</td>
<td>33993</td>
<td>4016</td>
<td>7</td>
<td>81831</td>
<td>0.32</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>image/png</td>
<td>6615475</td>
<td>261853</td>
<td>4627</td>
<td>421</td>
<td>6348674</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>application/x-www-form-urlencoded</td>
<td>3621</td>
<td>2</td>
<td>3417</td>
<td>0</td>
<td>202</td>
<td>0.94</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>text/plain</td>
<td>476285</td>
<td>91057</td>
<td>2607</td>
<td>187</td>
<td>382214</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>font/tf</td>
<td>16525</td>
<td>12309</td>
<td>528</td>
<td>0</td>
<td>5688</td>
<td>0.69</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>application/font-sfnt</td>
<td>9174</td>
<td>2575</td>
<td>379</td>
<td>0</td>
<td>6220</td>
<td>0.32</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>application/font-woff</td>
<td>149142</td>
<td>29919</td>
<td>315</td>
<td>0</td>
<td>119907</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>image/webp</td>
<td>668597</td>
<td>12615</td>
<td>281</td>
<td>0</td>
<td>685701</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>application/xml</td>
<td>66662</td>
<td>22580</td>
<td>203</td>
<td>16</td>
<td>43863</td>
<td>0.34</td>
<td></td>
</tr>
</tbody>
</table>

Brotli use has been growing, but where is it most prevalent?
When we exclude Google and Facebook content, the bulk of Brotil encoded content is JS and CSS.
Most byte savings are obtained by using the highest compression level.

Data on compression speeds from [quixdb.github.io/squash-benchmark/#results](quixdb.github.io/squash-benchmark/#results)
Resource Optimizer: Automated Brotli Compression at the Edge

- Automatically compresses CSS and JS with Brotli
- Resources are compressed offline and then cached
- Brotli compression level 11, without the overhead!
Case Study #2
Server Technologies

1. Akamai Varnish Connector
2. Security Vulnerability Research
Akamai Varnish Connector

- **Purge Synchronization**: Invalidate or purge objects in Akamai cache.
- **Define Cache Rules**: Change Akamai cache and no-store rules.
- **Ensure Best Practices**: Assure your site is taking full advantage of Akamai features and functions.
- **Defer ESI Processing**: Ensure ESI content is processed at the Edge.
- **Parse EDC Header**: Populate inbound device characteristics into a VCL object.
- **Control Refresh Traffic**: Convert stale-while-revalidate header to its Akamai equivalent.

[developer.akamai.com/connector-for-varnish/](developer.akamai.com/connector-for-varnish/) (free!)
How Many Akamai Customers Use Varnish at the Origin?

- HTTP Archive data helped determine which Akamai’s customers were using Varnish at the origin.
- Akamai Product Management was able to discuss desirable functionality with existing customers.
Investigating Security Threats - 0 Day Vulnerability

- Example: 0 Day Vulnerability on an Ecommerce App Server

- No CVE or Historical Attack Patterns
  - Target attack observed and mitigated (Kona Managed Security)
  - Akamai WAF rules prepared
  - Vendor notified

- HTTP Archive
  - Investigate other sites that contain similar characteristics
    - Server, Via, URL Regex Patterns, Other Headers
  - Export a list of sites that appear vulnerable
  - Cross Reference with Akamai Account Data
    - Notify 24x7 Security Contacts,
    - Help customers proactively protect themselves
Drupal announced that a “highly critical” security release would be happening within a week...

Expectation is that it would give sites time to prepare for an emergency security patch before 0 Day exploits begin...

Identifying Sites Running Drupal with HTTP Archive

First Try:
- WHERE url LIKE "%drupal.js%"
- Found 97 sites using Akamai and Drupal

Second Try:
- Expires header = 'Sun, 19 Nov 1978 05:00:00 GMT'
  - https://www.ostraining.com/blog/drupal/5-ways-drupal/
- Found more sites using Akamai and Drupal
  - ~26K total requests with this expires header!

What Did We do With this Data?
- Customer Outreach (Are you aware and prepared to patch?)
- Prepared WAF rules for those not able to apply patches immediately.
Do any of my customers have cryptocurrency miners?
  ○ Are they aware?
  ○ Do they know how it got there?

Now Easier with Wappalyzer!

- Wappalyzer is a Cross Platform utility that uncovers technologies used on websites.
- Integrated into HTTP Archive since April 2018

https://discuss.httparchive.org/t/using-wappalyzer-to-analyze-cpu-times-across-js-frameworks/1336/
Investigating Security Threats - What Domains are Serving Malware?

- Akamai’s ETP Service
  - Millions of malicious domains and IP addresses.
  - Are any of my customers serving 3rd party content from known malware hosts?

- HTTP Archive parsed against the ETP DB
  - Notified accounts if they served content to the HTTP Archive from known malware hosts
Case Study #3
Third Party Research

1. How 3rd Parties Influence Render Time?
2. Researching a specific 3rd party
Do Third Parties Impact Load Time?

- CrUX = Chrome User Experience Report
- JOINed’ w/ HTTP Archive data for Alexa Ranks
- Load times are faster for sites with less third party content.

SELECT mimeType, COUNT(*) num_requests, SUM(IF(req.startedDateTime < (pages.startedDateTime + (renderStart/1000) ),1,0)) BeforeRenderStart, SUM(IF(req.startedDateTime < (pages.startedDateTime + (renderStart/1000) ),0,1)) AfterRenderStart FROM httparchive.summary_requests.2017_09_01_desktop req JOIN (SELECT rank, NET.HOST(url) hostname, url, pageid, startedDateTime, renderStart FROM httparchive.summary_pages.2017_09_01_desktop) pages ON pages.pageid = req.pageid WHERE NET.HOST(req.url) != pages.hostname AND rank > 0 AND rank < 100000 GROUP BY mimeType HAVING num_requests > 1000 ORDER BY BeforeRenderStart DESC
### What 3rd Party Content Loads Before RenderStart?

<table>
<thead>
<tr>
<th>Content-Type</th>
<th>All Sites</th>
<th>Top 10K</th>
<th>Top 1K</th>
<th>Top 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>image/jpg</td>
<td>121%</td>
<td>66%</td>
<td>65%</td>
<td>56%</td>
</tr>
<tr>
<td>image/gif</td>
<td>10%</td>
<td>9%</td>
<td>8%</td>
<td>8%</td>
</tr>
<tr>
<td>text/html</td>
<td>15%</td>
<td>14%</td>
<td>11%</td>
<td>11%</td>
</tr>
<tr>
<td>image/png</td>
<td>50%</td>
<td>55%</td>
<td>53%</td>
<td>51%</td>
</tr>
<tr>
<td>text/javascript</td>
<td>25%</td>
<td>25%</td>
<td>21%</td>
<td>19%</td>
</tr>
<tr>
<td>application/javascript</td>
<td>9%</td>
<td>7%</td>
<td>6%</td>
<td></td>
</tr>
<tr>
<td>application/x-javascript</td>
<td>53%</td>
<td>49%</td>
<td>42%</td>
<td>40%</td>
</tr>
<tr>
<td>text/css</td>
<td>72%</td>
<td>71%</td>
<td>69%</td>
<td>71%</td>
</tr>
<tr>
<td>font/woff2</td>
<td>50%</td>
<td>62%</td>
<td>56%</td>
<td></td>
</tr>
<tr>
<td>application/json</td>
<td>6%</td>
<td>8%</td>
<td>8%</td>
<td></td>
</tr>
<tr>
<td>image/webp</td>
<td>65%</td>
<td>64%</td>
<td>59%</td>
<td>47%</td>
</tr>
<tr>
<td>text/plain</td>
<td>10%</td>
<td>10%</td>
<td>8%</td>
<td></td>
</tr>
<tr>
<td>text/xml</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td></td>
</tr>
<tr>
<td>image/svg+xml</td>
<td>52%</td>
<td>60%</td>
<td>61%</td>
<td>67%</td>
</tr>
<tr>
<td>application/octet-stream</td>
<td>36%</td>
<td>39%</td>
<td>40%</td>
<td></td>
</tr>
<tr>
<td>application/font-woff2</td>
<td>40%</td>
<td>59%</td>
<td>58%</td>
<td></td>
</tr>
<tr>
<td>video/mp4</td>
<td>3%</td>
<td>4%</td>
<td>5%</td>
<td></td>
</tr>
<tr>
<td>image/x-icon</td>
<td>5%</td>
<td>6%</td>
<td>12%</td>
<td></td>
</tr>
<tr>
<td>application/xml</td>
<td>4%</td>
<td>3%</td>
<td>1%</td>
<td></td>
</tr>
</tbody>
</table>

[discuss.httparchive.org/t/which-3rd-party-content-loads-before-render-start/1084](discuss.httparchive.org/t/which-3rd-party-content-loads-before-render-start/1084)
SELECT NET.HOST(req.url) thirdparty,
    mimeType,
    COUNT(*) num_requests,
    SUM(
        IF(req.startedDateTime < (pages.startedDateTime + (renderStart/1000)),1,0)
    ) BeforeRenderStart,
    SUM(
        IF(req.startedDateTime < (pages.startedDateTime + (renderStart/1000)),0,1)
    ) AfterRenderStart
FROM httparchive.summary_requests.2017_09_01_desktop req
JOIN (SELECT rank, NET.HOST(url) hostname, url, pageid, startedDateTime, renderStart
FROM httparchive.summary_pages.2017_09_01_desktop)
    pages ON pages.pageid = req.pageid
WHERE NET.HOST(req.url) != pages.hostname AND rank > 0 AND rank < 100000
GROUP BY thirdparty, mimeType
HAVING num_requests > 100
ORDER BY BeforeRenderStart DESC
What 3rd Party Domains Loads Before RenderStart?

<table>
<thead>
<tr>
<th>Row</th>
<th>thirdparty</th>
<th>ContentType</th>
<th>num_requests</th>
<th>Be</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>fonts.gstatic.com</td>
<td>font/wofl2</td>
<td>130256</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>fonts.googleapis.com</td>
<td>text/css</td>
<td>63069</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><a href="http://www.google-analytics.com">www.google-analytics.com</a></td>
<td>text/javascript</td>
<td>78156</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><a href="http://www.google-analytics.com">www.google-analytics.com</a></td>
<td>text/javascript</td>
<td>76587</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>pagead2.googlesyndication</td>
<td>text/javascript</td>
<td>83512</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>connect.facebook.net</td>
<td>application/javascript</td>
<td>64028</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>ajax.googleapis.com</td>
<td>text/javascript</td>
<td>25772</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td><a href="http://www.googletagmanager.com">www.googletagmanager.com</a></td>
<td>application/javascript</td>
<td>20641</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>i0.wp.com</td>
<td>image/webp</td>
<td>10109</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>cdnjs.cloudflare.com</td>
<td>application/javascript</td>
<td>13522</td>
<td></td>
</tr>
</tbody>
</table>
SELECT rank, site,
    SUM(
        IF(req.startedDateTime < (pages.startedDateTime + (renderStart/1000)), 1, 0)
    ) BeforeRenderStart,
    SUM(
        IF(req.startedDateTime < (pages.startedDateTime + (renderStart/1000)), 0, 1)
    ) AfterRenderStart
FROM httparchive.summary_requests.2017_09_01_desktop req
INNER JOIN (SELECT rank, NET.HOST(url) site, pageid, startedDateTime, renderStart
    FROM httparchive.summary_pages.2017_09_01_desktop
) pages ON pages.pageid = req.pageid
WHERE NET.HOST(req.url) LIKE "%ensighten.com%" AND rank > 0
GROUP BY rank, site
HAVING AfterRenderStart > 0
ORDER BY rank ASC

Which Websites Load <3rd Party> Before vs After Render Time?  
bit.ly/2LuD1Jq44
Which Sites Load a Specific 3rd Party Before RenderStart?

This query outputs a summary containing the following information:

- Which sites use the 3rd party?
- How many resources are served by it?
- How many of them are loaded before/after the page renders?

Results can help sites learn from each other’s best practices
- Even across industries!!!
Getting Started

1. Google Cloud project
2. BigQuery config
Up and running - Create a Google Cloud Project

- console.cloud.google.com/projectcreate

Google Cloud Platform

New Project

You have 11 projects remaining in your quota. Learn more.

Project name

httparchive-bigquery

Your project ID will be httparchive-bigquery-182602

Create  Cancel
Up and running - Add the HTTP Archive Tables

- bigquery.cloud.google.com


---

**a) click the project dropdown menu**

**b) click “Display Project”**

**c) Manually enter “httparchive” and click OK**
Up and running - Start Exploring!

- Detailed Setup Instructions:

- BigQuery Standard SQL Documentation

- HTTP Archive Discussion Forum
  - [discuss.httparchive.org](http://discuss.httparchive.org)
<table>
<thead>
<tr>
<th>Topic</th>
<th>Category</th>
<th>Users</th>
<th>Replies</th>
<th>Views</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>M dot or RWD. Which is faster?</td>
<td>Analysis</td>
<td></td>
<td>0</td>
<td>10.2k</td>
<td>Jun '14</td>
</tr>
<tr>
<td>Cache Control Immutable - A Year Later</td>
<td>Analysis</td>
<td></td>
<td>1</td>
<td>9.3k</td>
<td>Jan 7</td>
</tr>
<tr>
<td>What is the least common colour used on web pages?</td>
<td>Analysis</td>
<td></td>
<td>5</td>
<td>9.0k</td>
<td>Sep '17</td>
</tr>
<tr>
<td>Are Popular Websites Faster?</td>
<td>Analysis</td>
<td></td>
<td>2</td>
<td>8.3k</td>
<td>Dec '17</td>
</tr>
<tr>
<td>What is the distribution of 1st party vs 3rd party resources?</td>
<td>Analysis</td>
<td></td>
<td>12</td>
<td>7.9k</td>
<td>Nov '17</td>
</tr>
<tr>
<td>1MB+ of HTTP overhead due to TowerData cookies</td>
<td>Analysis</td>
<td></td>
<td>1</td>
<td>7.7k</td>
<td>Dec '16</td>
</tr>
<tr>
<td>Sites that deliver Images using gzip/deflate encoding</td>
<td>Analysis</td>
<td></td>
<td>8</td>
<td>7.6k</td>
<td>Apr '14</td>
</tr>
<tr>
<td>JavaScript Library Detection</td>
<td>Analysis</td>
<td></td>
<td>11</td>
<td>6.2k</td>
<td>7d</td>
</tr>
<tr>
<td>Analyzing HTML, CSS, and JavaScript response bodies</td>
<td>Analysis</td>
<td></td>
<td>6</td>
<td>6.0k</td>
<td>Sep '16</td>
</tr>
<tr>
<td>The Performance Impact of Cryptocurrency Mining on the Web</td>
<td>Analysis</td>
<td></td>
<td>10</td>
<td>5.7k</td>
<td>14d</td>
</tr>
<tr>
<td>Tracking JavaScript library versions in HTTP archive</td>
<td>Analysis</td>
<td></td>
<td>24</td>
<td>5.6k</td>
<td>Mar '17</td>
</tr>
</tbody>
</table>

[discuss.httparchive.org](https://discuss.httparchive.org)
Contact team@httparchive.org if interested
Thanks!

Contribute
github.com/HTTPArchive

Collaborate
discuss.httparchive.org

Chat
bit.ly/ha-slack

@paulcalvano
pacalvan@akamai.com