Empirical Results from Page Speed and mod_pagespeed

Joshua Marantz & Matthew Steele
Google
November 8, 2011

@jmarantz
www.modpagespeed.com
© 2011 Google, Inc. All rights reserved.
What is mod_pagespeed?

Rewrites web sites so they load & display faster, use less bandwidth

Open-source Apache Module and C++ API

Launched November 2010, now on 70k sites
What does mod_pagespeed do?

- Automatic Image Compression & Resizing
- Minify CSS, JavaScript and HTML
- Inline small images, CSS, and JavaScript
- Cache Extension
- CSS/Javascript Combining
- Domain Mapping
- Domain Sharding
What do we get out of these optimizations?
What can we learn about WPO from this tool?
Case Study: androidacademy.com
IE8 Page-load Speedup Breakdown (% improvement)

- Best of all Rewriters Combined
- Disable inlining
- Minify CSS & Extend Cache
- Shard Domains to 2
- Minify CSS, JS, HTML
- Optimize Images (lossy)
- Optimize Images (lossless)
- Combining CSS, JS
- Inlining Images, CSS, JS
- Convert Meta-tags to Request Headers

Source: Google internal data
Web Page Performance Test for www.androidacademy.com

From: IE8 - Corp - custom
Fri Nov 4 19:21:19 2011
Scripted test

Test runs: 20
Connectivity: 500/150 Kbps, 150ms Latency

Performance Results (Median Run)

<table>
<thead>
<tr>
<th>Load Time</th>
<th>First Byte</th>
<th>Start Render</th>
<th>DOM Elements</th>
<th>Time</th>
<th>Requests</th>
<th>Bytes In</th>
</tr>
</thead>
<tbody>
<tr>
<td>First View (Run 18)</td>
<td>6.705s</td>
<td>0.544s</td>
<td>2.252s</td>
<td>826</td>
<td>6.705s</td>
<td>52</td>
</tr>
<tr>
<td>Repeat View (Run 5)</td>
<td>3.389s</td>
<td>0.580s</td>
<td>1.506s</td>
<td>826</td>
<td>3.389s</td>
<td>5</td>
</tr>
</tbody>
</table>
Look at your scatter plots
<table>
<thead>
<tr>
<th>mod_pagespeed setting</th>
<th>PLT Percent Improvement</th>
<th>Page S Score</th>
<th>Load Time</th>
<th>First Byte</th>
<th>Start Render</th>
<th>Elements</th>
<th>DOM Elements</th>
<th>Document Complete</th>
<th>Fully Loaded</th>
<th>Source: Google internal data</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache_extend, minify_css</td>
<td>-1.13%</td>
<td>64, First</td>
<td>7.57s</td>
<td>0.532s</td>
<td>2.178s</td>
<td>838</td>
<td>7.571s</td>
<td>79</td>
<td>519 KB</td>
<td>12.985s, 82, 537 KB</td>
</tr>
<tr>
<td>convert_meta_tags</td>
<td>-0.61%</td>
<td>Rep</td>
<td>3.735s</td>
<td>0.530s</td>
<td>1.776s</td>
<td>838</td>
<td>3.735s</td>
<td>6</td>
<td>20 KB</td>
<td>3.736s, 5, 20 KB</td>
</tr>
<tr>
<td>minify_css</td>
<td>21.08%</td>
<td>64, First</td>
<td>9.797s</td>
<td>0.539s</td>
<td>2.928s</td>
<td>857</td>
<td>9.797s</td>
<td>81</td>
<td>528 KB</td>
<td>14.631s, 85, 546 KB</td>
</tr>
<tr>
<td>trim urls only</td>
<td>0.27%</td>
<td>Rep</td>
<td>4.813s</td>
<td>0.539s</td>
<td>2.031s</td>
<td>857</td>
<td>4.813s</td>
<td>6</td>
<td>20 KB</td>
<td>6.115s, 7, 21 KB</td>
</tr>
<tr>
<td>convert_meta_tags</td>
<td>14.35%</td>
<td>64, First</td>
<td>8.298s</td>
<td>0.523s</td>
<td>2.181s</td>
<td>857</td>
<td>8.298s</td>
<td>81</td>
<td>530 KB</td>
<td>12.607s, 85, 548 KB</td>
</tr>
<tr>
<td>combine js &amp; css</td>
<td>5.83%</td>
<td>Rep</td>
<td>4.356s</td>
<td>0.524s</td>
<td>1.745s</td>
<td>657</td>
<td>4.356s</td>
<td>6</td>
<td>20 KB</td>
<td>4.000s, 7, 21 KB</td>
</tr>
<tr>
<td>image compress 85</td>
<td>5.39%</td>
<td>65, First</td>
<td>9.123s</td>
<td>0.537s</td>
<td>2.348s</td>
<td>851</td>
<td>9.123s</td>
<td>72</td>
<td>525 KB</td>
<td>13.974s, 75, 543 KB</td>
</tr>
<tr>
<td>image compress lossless</td>
<td>4.56%</td>
<td>Rep</td>
<td>4.365s</td>
<td>0.560s</td>
<td>1.791s</td>
<td>851</td>
<td>4.562s</td>
<td>8</td>
<td>20 KB</td>
<td>4.893s, 7, 21 KB</td>
</tr>
<tr>
<td>inlining &amp; rewrite_css</td>
<td>-0.19%</td>
<td>75, First</td>
<td>9.246s</td>
<td>0.582s</td>
<td>2.986s</td>
<td>857</td>
<td>9.246s</td>
<td>61</td>
<td>499 KB</td>
<td>14.075s, 85, 517 KB</td>
</tr>
<tr>
<td>shard 2 &amp; rewrite_css</td>
<td>0.33%</td>
<td>Rep</td>
<td>4.793s</td>
<td>0.561s</td>
<td>2.530s</td>
<td>857</td>
<td>4.793s</td>
<td>6</td>
<td>21 KB</td>
<td>5.031s, 7, 21 KB</td>
</tr>
<tr>
<td>extend_cache</td>
<td>0.92%</td>
<td>71, First</td>
<td>9.599s</td>
<td>0.529s</td>
<td>2.941s</td>
<td>857</td>
<td>9.599s</td>
<td>81</td>
<td>522 KB</td>
<td>13.372s, 85, 540 KB</td>
</tr>
<tr>
<td>rewrite_css</td>
<td>-1.29%</td>
<td>Rep</td>
<td>4.768s</td>
<td>0.532s</td>
<td>1.991s</td>
<td>857</td>
<td>4.768s</td>
<td>6</td>
<td>21 KB</td>
<td>5.017s, 7, 21 KB</td>
</tr>
<tr>
<td>shard 2 &amp; rewrite_css</td>
<td>-0.44%</td>
<td>64, First</td>
<td>9.731s</td>
<td>0.531s</td>
<td>2.929s</td>
<td>857</td>
<td>9.731s</td>
<td>81</td>
<td>528 KB</td>
<td>14.075s, 85, 546 KB</td>
</tr>
<tr>
<td>extend_cache</td>
<td>-0.25%</td>
<td>Rep</td>
<td>4.796s</td>
<td>0.526s</td>
<td>1.998s</td>
<td>857</td>
<td>4.796s</td>
<td>6</td>
<td>20 KB</td>
<td>5.046s, 7, 21 KB</td>
</tr>
<tr>
<td>rewrite_css</td>
<td>14.55%</td>
<td>81, First</td>
<td>9.081s</td>
<td>0.531s</td>
<td>2.013s</td>
<td>836</td>
<td>9.081s</td>
<td>79</td>
<td>519 KB</td>
<td>13.930s, 82, 537 KB</td>
</tr>
</tbody>
</table>
Lessons & Caveats

- WebPageTest is *amazing* but is by nature a lab test
- Sites are different – not all rewrites will benefit all sites
- Rewriters combine non-linearly
- Run many iterations and look at scatter-plots
- Measuring finds opportunities for improvement
Case Study #2: international sports site

- Not running mod_pagespeed at origin
- Site mirrored & tested with mod_pagespeed on & off
- Image rich
- Benefits from compression & inlining
IE8 Image Compression & Inlining Breakdown (% improvement)

- Inlining 4k, Jpeg Compression 85
- Inlining 1k
- No Inlining
- Lossless Image Compression
- No Image Compression

Source: Google internal data
Take-Away

• Site-specific optimization is time-consuming

• mod_pagespeed provides zero-effort Pretty Good Speedup

• mod_pagespeed & webpagetest can guide manual effort
mod_pagespeed roadmap

• Make more of the web faster: 70,000 sites is a start…
  – High-volume partners (Hosting Providers, CDNs)
  – Leveraged integrations with servers, proxies, caches, etc
  – Extreme Support

• Make the web more fast
  – Tune default inlining thresholds & image compression ratios based on these results
  – Dynamic Threshold Tuning based on User-Agent
  – Deferring JavaScript execution

• Measure Better
  – Make it easier to reproduce these experiments for any site
  – Run experiments across a broader swath of the internet
  – Collect data from real browsers & web sites in production
mod_pagespeed: Automatically Make Your Web Sites Faster

<table>
<thead>
<tr>
<th>Action</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read About it</td>
<td><a href="http://code.google.com/speed/page-speed/docs/module.html">http://code.google.com/speed/page-speed/docs/module.html</a></td>
</tr>
<tr>
<td>Try it</td>
<td><a href="http://www.modpagespeed.com">http://www.modpagespeed.com</a></td>
</tr>
<tr>
<td>Download it</td>
<td><a href="http://code.google.com/speed/page-speed/download.html">http://code.google.com/speed/page-speed/download.html</a></td>
</tr>
<tr>
<td>Discuss it</td>
<td><a href="http://groups.google.com/group/mod-pagespeed-discuss">http://groups.google.com/group/mod-pagespeed-discuss</a></td>
</tr>
<tr>
<td>Build From source</td>
<td><a href="http://code.google.com/p/modpagespeed/wiki/HowToBuild">http://code.google.com/p/modpagespeed/wiki/HowToBuild</a></td>
</tr>
<tr>
<td>Google I/O 2011 videos</td>
<td><a href="http://www.youtube.com/watch?v=30_AIEhar-l">http://www.youtube.com/watch?v=30_AIEhar-l</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://www.youtube.com/watch?v=hLQbsqSx6Y4">http://www.youtube.com/watch?v=hLQbsqSx6Y4</a></td>
</tr>
</tbody>
</table>