How to PageSpeed with Apache

Abhishek Tiwari

Citation: *A. Tiwari*, "How to PageSpeed with Apache", Abhishek Tiwari, 2014. doi:10.59350/2a5yq-p7n39

Published on: June 17, 2014

This is a quick rundown on how to install, configure and tune the PageSpeed on Ubuntu with Apache as web server.

Installing

First get the appropriate `mod_pagespeed` (64-bit or 32-bit version) `.DEB` package for your Ubuntu server.

```
wget https://dl-ssl.google.com/dl/linux/direct/mod-pagespeed-
    stable_current_amd64.deb
wget https://dl-ssl.google.com/dl/linux/direct/mod-pagespeed-
    stable_current_i386.deb
```

To install run following commands in same directory where you have downloaded `.DEB` binary. Post installation we restarted Apache and removed the downloaded package to keep it clean.

```
sudo dpkg -i mod-pagespeed-*.deb
apt-get -f install
service apache2 restart
rm mod-pagespeed-*.deb
```

To restart Apache instead of `service apache2 restart` you can also use `/etc/init.d/apache2 restart`.

Validate

Next you need to validate that if your PageSpeed installation is working or not by looking at response headers of page returned by Apache. Response header will normally include `X-Mod-Pagespeed` header which is indication that installation is working properly.

Configuring

On Ubuntu, PageSpeed configuration file for Apache (`pagespeed.conf`) is located at `/etc/apache2/mods-available/`. Every time we make change in the `pagespeed.conf`an Apache restart is required to activate new configuration change.

Filters

By default `CoreFilters` are enabled which includes following filter,

```
    add_head
    combine_css
    combine_javascript
    convert_meta_tags
    extend_cache
    fallback_rewrite_css_urls
    flatten_css_imports
    inline_css
    inline_import_to_link
    inline_javascript
    rewrite_css
    rewrite_images
    rewrite_javascript
    rewrite_style_attributes_with_url
```

Tuning

Enable PageSpeed Server-Side Cache

In your `pagespeed.conf` file, PageSpeed server-side cache location is set by `ModPagespeedFileCachePath`. By default it will be enabled and look like following.

```
ModPagespeedFileCachePath               "/var/cache/mod_pagespeed/"
```

Please note that PageSpeed server-side cache will not work unless the cache location is writable by Apache user/group (`www-data`).

```
ps aux | grep -v root | grep apache | cut -d\  -f1 | sort | uniq
chown -R www-data:www-data /var/cache/mod_pagespeed
```

Flushing PageSpeed Server-Side Cache:

Normally PageSpeed server-side cache files live for 5 minutes. After that it automatically refresh using a conditional check (if file changed get fresh version and cache it, flush old version; else keep current version in cache just refresh the expiry header). That said you can manually flush PageSpeed server-side cache before the cache lifetime expires simply by touching the file “cache.flush” in cache directory. If you are loading static files from disk (next) this is not required.

```
touch /var/cache/mod_pagespeed/cache.flush
```

Loading static files from disk

Loading static files from disk enables PageSpeed to load sub-resource faster. By default PageSpeed loads sub-resources via an HTTP fetch.To read directly from the filesystem you can add `ModPagespeedLoadFromFile` to your Apache `pagespeed.conf` file. This also mean you don't need to flush the server-side cache maintained by PageSpeed. In following example I have added 4 different folder location to load static files from disk directly.

```
ModPagespeedLoadFromFile "http://example.com/static/" "/var/www/static/"
ModPagespeedLoadFromFile "http://example.com/_themes/mytheme/css/" "/var/
    www/_themes/mytheme/css/"
ModPagespeedLoadFromFile "http://abhishek-tiwari.com/_themes/mytheme/js/"
    "/var/www/_themes/mytheme/js/"
ModPagespeedLoadFromFile "http://abhishek-tiwari.com/_themes/mytheme/img/"
     "/var/www/_themes/mytheme/img/"
ModPagespeedLoadFromFile "http://abhishek-tiwari.com/_themes/mytheme/fonts
    /" "/var/www/_themes/mytheme/fonts/"
```

Enable Additional Filters

For majority of cases enabling filters for prioritising critical CSS, lazy loading images (below the fold) and inline preview images will improve performance. ~~~ ModPagespeedEnableFilters prioritize_critical_css
ModPagespeedEnableFilters lazyload_images,inline_preview_images,insert_image_dimensions
~~~

Inline preview images filter generates low-quality image and inline them. These low-quality images are swapped with by high quality versions after an `onload` event is triggered (LQIP, Low Quality Image Placeholders Approach). Here `insert_image_dimensions` was used to avoid reflow as the images load.

In addition by inserting image dimensions (height and width) web page will load faster because when browser knows the image dimension it can layout faster and avoid multiple layout exercise.

> If you do not tell the browser the size of your images then it must "build" the page not once, but twice (or more times depending on how many images you have on the page). It will build it once to display all the text, and then it will wait until an image is downloaded. When one image is downloaded the browser can now determine the size of the image and will rebuild the page to wrap the text around that image. This process will happen for every image on your page.

A similarly filter, resize mobile images is available for mobile only browsers.

```
ModPagespeedEnableFilters resize_mobile_images,insert_image_dimensions
```
~~~

If possible then experiment with defer JavaScript filter and see how it impacts your page rendering. If page does not break by deferring JavaScript you should enable it.

```
ModPagespeedEnableFilters defer_javascript
```

Testing

Using following you can test if your PageSpeed configuration changes are improving the performance or making worse is,

1. PageSpeed Insights: Higher PageSpeed score means improvement
2. Web Page Test: Lower page load is better

Epilogue

That's it. For more details have a look at list of PageSpeed filters and PageSpeed example directory.