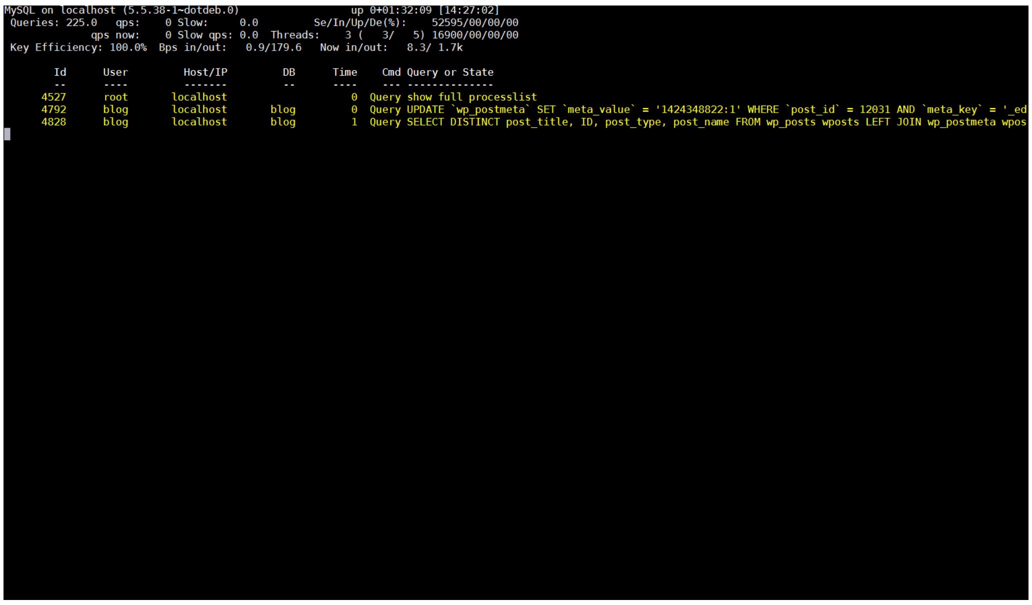
# Wordpress Plugins to monitor and debug WP enabled plugins - Find Errors / Warnings and Remove WP problematic plugins slowing down your Website (blog) database

**Author :** admin

Recent days, I'm **spending a lot of time again trying to optimize my wordpress blog. Optimizing WP for better efficiency is becoming harder and harder task day by day as the website file content data is growing along with SQL databases. Moreover situation gets even worse because the number of plugins enabled on my blog is incrementally growing with time because, there is more and more goodies I'd like to add.**
Optimizing Wordpress to run for Speed on a server is a whole a lot of art and its a small universe in itself, **because as of time of writting this post the count (number) of Wordpress available PLUGINS is 36,197 !**

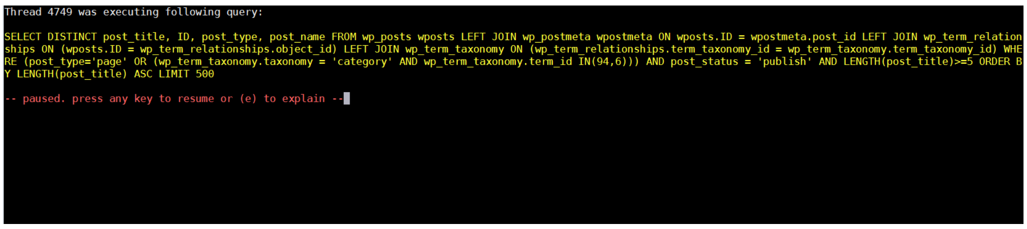
1. Manually Tracking Wordpress Plugins causing Slow SQL Queries (MySQL bottleneck) issues directly using console / SSH

Because of its open source development and its nice modular design wordpress has turned into a standard for building small, middle sized and large websites (some **Wordpress based blogs and sites have from 50 000 to 100 000 unique pages!).** My blog is still a small Wordpress site with only 1676 posts, so I still haven't reached the high volume traffic optimization requirements but still even though **I have a relatively good server hardware  8GB RAM / (2x2.70 Ghz Intel CPU) / 500 GB (7400 RPM HDD)** at times I see **Apache Webservers is unable to properly serve coming requests because of MySQL database (LEFT JOIN) requests being slow to serve (taking up to few seconds to complete) and creating a** ***MySQL table lock*****,** ***putting all the rest SQL queries to stay in a long unserved queues line*****,** I've realized about this **performance issue** by using a a mysql cli (command) client and few commands and console command (tool) called mytop **(also known as mtop). MyTop** refreshes every 3 seconds, so the slow query will immediately stay on screen to view moer info about it press **"f"** and type the  in **query ID.**



```
MySQL on localhost (5.5.38-1~dotdeb.0)                  up 0+01:32:09 [14:27:02]
 Queries: 225.0   qps:    0 Slow:     0.0         Se/In/Up/De(%):    52595/00/00/00
             qps now:    0 Slow qps: 0.0  Threads:    3 (   3/   5) 16900/00/00/00
 Key Efficiency: 100.0%  Bps in/out:   0.9/179.6   Now in/out:   8.3/ 1.7k

       Id      User         Host/IP         DB      Time    Cmd Query or State
       --      ----         -------         --      ----    --- --------------
     4527      root       localhost                    0  Query show full processlist
     4792      blog       localhost       blog         0  Query UPDATE `wp_postmeta` SET `meta_value` = '1424348822:1' WHERE `post_id` = 12031 AND `meta_key` = '_ed
     4828      blog       localhost       blog         1  Query SELECT DISTINCT post_title, ID, post_type, post_name FROM wp_posts wposts LEFT JOIN wp_postmeta wpos
```



```
Thread 4749 was executing following query:

SELECT DISTINCT post_title, ID, post_type, post_name FROM wp_posts wposts LEFT JOIN wp_postmeta wpostmeta ON wposts.ID = wpostmeta.post_id LEFT JOIN wp_term_relation
ships ON (wposts.ID = wp_term_relationships.object_id) LEFT JOIN wp_term_taxonomy ON (wp_term_relationships.term_taxonomy_id = wp_term_taxonomy.term_taxonomy_id) WHE
RE (post_type='page' OR (wp_term_taxonomy.taxonomy = 'category' AND wp_term_taxonomy.term_id IN(94,6))) AND post_status = 'publish' AND LENGTH(post_title)>=5 ORDER B
Y LENGTH(post_title) ASC LIMIT 500

-- paused. press any key to resume or (e) to explain --
```

**Finally it is very useful to run  for a while MySQL server logging to** **/var/log/mysql/slow-query.log:**

**Slow query is enabled (on my Debian 7 Wheezy host) by adding to /etc/mysql/my.cnf**

after conf section

```
  vim /etc/mysql/my.cnf
#
# * Logging and Replication
#
# Both location gets rotated by the cronjob.
# Be aware that this log type is a performance killer.
# As of 5.1 you can enable the log at runtime!
#general_log_file        = /var/log/mysql/mysql.log
#general_log             = 1
#
# Error logging goes to syslog due to /etc/mysql/conf.d/mysqld_safe_syslog.cnf.
#
# Here you can see queries with especially long duration
```

Paste:

```
  slow_query_log = 1
slow_query_log_file = /var/log/mysql/slow-query.log
long_query_time = 2
log-queries-not-using-indexes
```

And then to make new mysql configuration load restarted mysql server:

```
  debian-server:~# /etc/init.d/mysql restart
Stopping MySQL database server: mysqld.
Starting MySQL database server: mysqld ..
Checking for tables which need an upgrade, are corrupt or were
not closed cleanly..
```

**Leaving mysql-slow.log to be enabled for 30 minutes to an 1 hrs is a good time to track most problematic slow queries** and based on this queries, I took parts of SQL **UPDATE / SELECT /**

**INSERT etc. Db** queries which was problematic and grepped throughout /var/www/blog/wp-content/plugin files in order to **determine which Wordpress Plugin is triggering the slow query, causing blog to hang when too many clients try to see it in browser**.

My **main problematic SQL query having long execution time (about 2 to 3 seconds!!!) most commonly occuring in *slow-query.log*** was:

```
SELECT DISTINCT post_title, ID, post_type, post_name FROM wp_posts wposts LEFT JOIN wp_postmeta wpostmeta ON wposts.ID = wpostmeta.post_id LEFT JOIN wp_term_relationships ON (wposts.ID = wp_term_relationships.object_id) LEFT JOIN wp_term_taxonomy ON (wp_term_relationships.term_taxonomy_id = wp_term_taxonomy.term_taxonomy_id) WHERE (post_type='page' OR (wp_term_taxonomy.taxonomy = 'category' AND wp_term_taxonomy.term_id IN(11))) AND post_status = 'publish' AND LENGTH(post_title)>=5 ORDER BY LENGTH(post_title) ASC LIMIT 500
```

**Because above query uses SQL Column names and Tables which are not hard coded in PHP code, to find out which plugins is most probably to launch this complex LEFT JOIN query, I used a quick bash one-liner:**

```
# cd /var/www/blog/wp-content/plugins
# for i in $(grep -rli 'SELECT DISTINCT' *); do grep -rli 'LEFT JOIN' $i; done
./seo-automatic-links/seo-links.php
./wp-postviews/wp-postviews.php
./yet-another-related-posts-plugin/classes/YARPP_Cache_Tables.php
```

I wanted to put less load on CPU during grep so looked for string only in .PHP extensioned files with:

```
# for i in $(find . -iname '*.php' -exec grep -rli 'SELECT DISTINCT' '{}' \;); do grep -rli

'LEFT JOIN' $i; done
./seo-automatic-links/seo-links.php
./wp-postviews/wp-postviews.php
./yet-another-related-posts-plugin/classes/YARPP_Cache_Tables.php
```

As you can see the complex query is being called from PHP file belonging to one of 3 plugins

- **SEO Automatic Links** - this is SEO Smart Links WP plugin (Does internal bliog interlinking in order to boast SEA)
- **WP PostViews** - Wordpress Post Views plugin (Which allows me to show how many times an article was read in WP Widget menu)
- **Yet Another Related Posts** - Which is WP plugin I installed / enabled to show Related posts down on each blog post

###### 2. Basic way to optimize MySQL slow queries (EXPLAIN / SHOW CREATE TABLE)

Now as I have a basic clue on plugins locking my Database, **I disabled them one by one while keeping enabled mysql slow query log and viewing queries in mytop and I figure out that actually all of the**

**plugins were causing a short time overheat (lock) on server Database because of *LEFT JOINs.*** Though I really like what this plugins are doing, as they **boast SEO and attract prefer to disable them for now and have my blog all the time responsible light fast instead of having a little bit better Search Engine Optimization (Ranking) and loosing many of my visitors because they're annoyed to wait until my articles open** ...

Before disabling I tried to optimize the queries using MySQL **EXPLAIN** command + **SHOW CREATE TABLE (2 commands often used to debug slow SQL queries and find out whether a Column needs to have added INDEX-ing to boast MySQL query)**.

Just in case if you decide to give them a try here is example on how they're used to **debug problematic SQL query**:

```
  mysql> explain SELECT DISTINCT post_title, ID, post_type, post_name
      -> FROM wp_posts wposts LEFT JOIN wp_postmeta wpostmeta
      -> ON wposts.ID = wpostmeta.post_id LEFT JOIN wp_term_relationships
      -> ON (wposts.ID = wp_term_relationships.object_id) LEFT JOIN wp_term_taxonomy
      -> ON (wp_term_relationships.term_taxonomy_id = wp_term_taxonomy.term_taxonomy_id)
```

```
    -> WHERE (post_type='page'
    -> OR (wp_term_taxonomy.taxonomy = 'category'
    -> AND wp_term_taxonomy.term_id IN(11,15,17)))
    -> AND post_status = 'publish'
    -> AND LENGTH(post_title)>=5
    -> ORDER BY LENGTH(post_title) ASC
    -> LIMIT 500;
 +----+-------------+-----------------------+--------+----------
--------+---------+---------+-----------------------------------
---------+------+---------------------------------------+
14.   | id | select_type | table                 | type   | possible_keys    | key     | key_len | ref                  | rows | Extra                                        |
15.   +----+-------------+-----------------------+--------+------------------+---------+---------+----------------------+------+----------------------------------------------+
16.   |  1 | SIMPLE      | wposts                | ALL    | type_status_date | NULL    | NULL    | NULL                 | 1715 | Using where; Using temporary; Using filesort |
17.   |  1 | SIMPLE      | wpostmeta             | ref    | post_id          | post_id | 8       | blog.wposts.ID       |   11 | Using index; Distinct                        |
18.   |  1 | SIMPLE      | wp_term_relationships | ref    | PRIMARY          | PRIMARY | 8       | blog.wposts.ID       |   19 | Using index; Distinct                        |
|  1 | SIMPLE      | wp_term_taxonomy      | eq_ref | PRIMARY           | PRIMARY | 8       | blog.wp_term_relationships.term_taxonomy_id |    1 | Using where; Distinct                        |
+----+-------------+-----------------------+--------+-------------------+---------+---------+---------------------------------------------+------+----------------------------------------------+
4 rows in set (0.02 sec)

mysql>

mysql> show create table wp_posts;
+----------+-------------------------+
| Table    | Create Table
                                                      |
+----------+-------------------------+
| wp_posts | CREATE TABLE `wp_posts` (
  `ID` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `post_author` bigint(20) unsigned NOT NULL DEFAULT '0',
  `post_date` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `post_date_gmt` datetime NOT NULL DEFAULT '0000-00-00
00:00:00',
  `post_content` longtext NOT NULL,
  `post_title` text NOT NULL,
  `post_excerpt` text NOT NULL,
  `post_status` varchar(20) NOT NULL DEFAULT 'publish',
  `comment_status` varchar(20) NOT NULL DEFAULT 'open',
  `ping_status` varchar(20) NOT NULL DEFAULT 'open',
  `post_password` varchar(20) NOT NULL DEFAULT '',
  `post_name` varchar(200) NOT NULL DEFAULT '',
  `to_ping` text NOT NULL,
  `pinged` text NOT NULL,
  `post_modified` datetime NOT NULL DEFAULT '0000-00-00
00:00:00',
  `post_modified_gmt` datetime NOT NULL DEFAULT '0000-00-00
00:00:00',
  `post_content_filtered` longtext NOT NULL,
  `post_parent` bigint(20) unsigned NOT NULL DEFAULT '0',
  `guid` varchar(255) NOT NULL DEFAULT '',
  `menu_order` int(11) NOT NULL DEFAULT '0',
  `post_type` varchar(20) NOT NULL DEFAULT 'post',
  `post_mime_type` varchar(100) NOT NULL DEFAULT '',
  `comment_count` bigint(20) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  KEY `post_name` (`post_name`),
      KEY `type_status_date` (`post_type`,`post_status`,`post_date`,`ID`),
      KEY `post_parent` (`post_parent`),
      KEY `post_author` (`post_author`),
      FULLTEXT KEY `post_related` (`post_title`,`post_content`)
    ) ENGINE=MyISAM AUTO_INCREMENT=12033 DEFAULT CHARSET=utf8 |
    +----------+---------------------+
    1 row in set (0.00 sec)

```

```
mysql>
```

By the way above output is a paste from the the new PasteBin Open Source (Stikked powered) service I started on pc-freak.net - paste.pc-freak.net (p.pc-freak.net) :)

Before I took final decision to **disable slow WP plugins**, I've experimented a bit trying to add INDEX to Table Column (wposts) in hope that this would speed up SQL queries with:

```
mysql> ALTER TABLE TABLE_NAME ADD INDEX (wposts);
...
```

But this didn't improve query speed even on the contrary it make execution time worse.

###### 3. Tracking Wordpress Plugin PHP Code Execution time and Plugins causing Slow SQL Queries (MySQL bottleneck) issues through WP itself

Well fine, I'm running my own hosted Blog and Wordpress sites, but for people who have wordpress sites on shared hosting, there is usually no SSH (Terminal) Access to server, those people will be happy to hear there are **2 Free** easy installable ***Wordpress plugins which can be used to Debug Slow Wordpress Plugins SQL Queries*** **as well as** ***plugin to Track which plugin takes most time to execute***, *this are:*

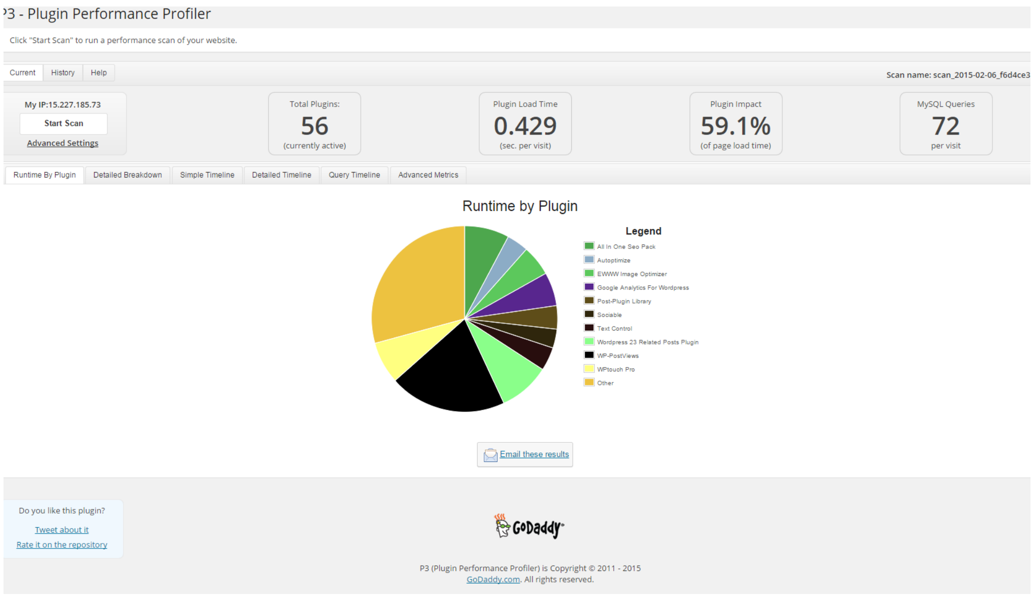
- Wordpress Query Monitor
- (P3 Plugin Performance Profiler)

- Wordpress Memory Viewer plugin

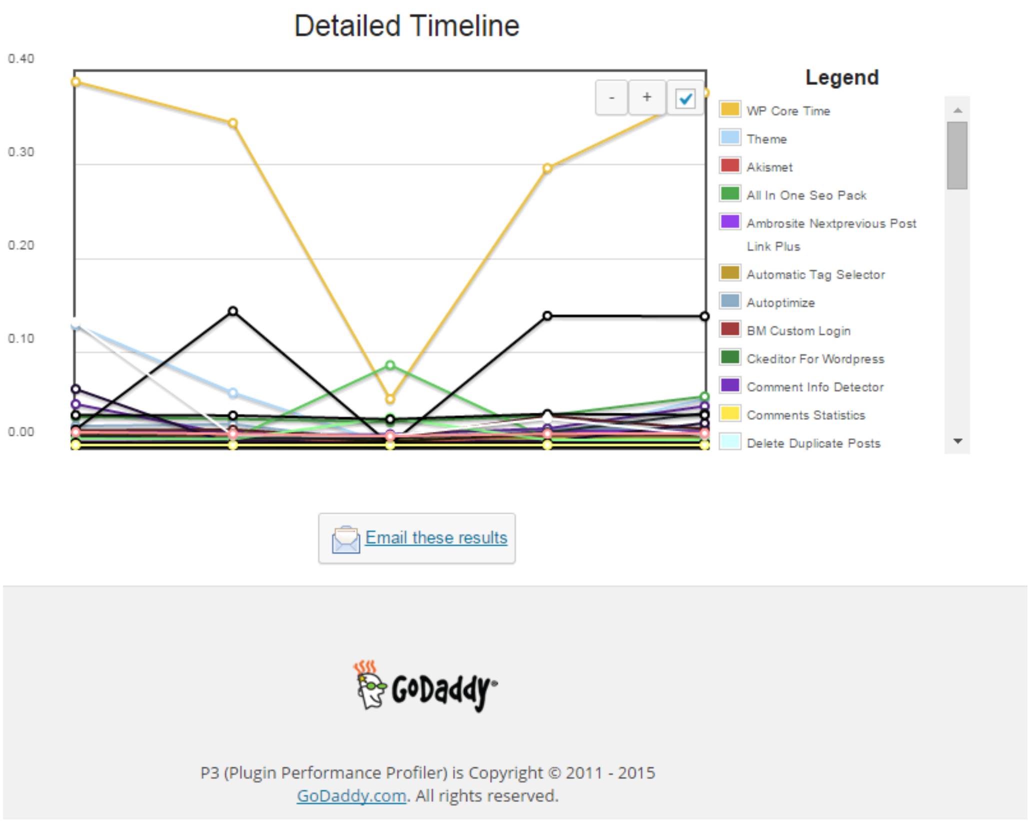
- Wordpress DebugBar Memory reporter plugin

a) ***P3 Plugin Performance Profiler***

runs a scan over your site to determine what resources your plugins are using, and when, during a standard page request. **P3 PPP** Can even create reports in a beatiful Excel like Pie chart sheet.



Another useful thing to see with **P3 PPP is Detailed Timeline** it shows when the plugins are being loaded during new page request so you can see if there is a certain sequence in time when a plugin slows down the website.



P3 (Plugin Performance Profiler) is Copyright © 2011 - 2015 GoDaddy.com. All rights reserved.

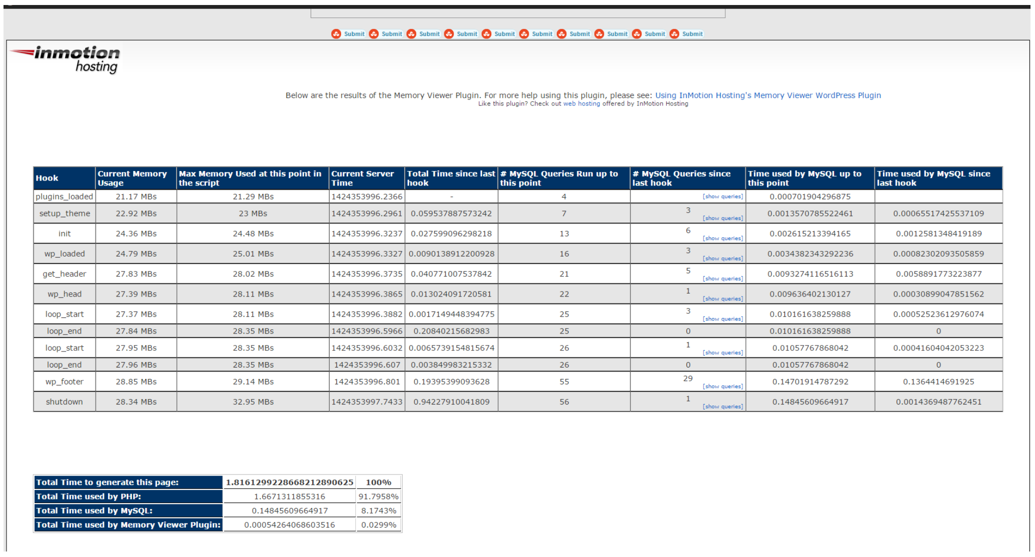
The pictures says it all as P3 PPP is Godaddy's work, congrats to GoDaddy, they've done great job.

***b) Wordpress memory Viewer WP* plugins**

**Is** useful to check how much memory each of Wordpress plugin is taking on user (visitor) request.
***Memory Viewer*** is allows you to view WordPress' memory utilization at several hooks during WordPress' execution. It also shows a summary of MySQL Queries that have ran as well as CPU time.
To use it download it to plugins/ folder as usual enable it from:

**Installed Plugins -> (Inactive) -> Memory Viewer (Enable)**

To see **statistics from Memory Viewer** *open any post from your blog website and scroll down to the bottom you will notice the statistics, showing up there, like on below screenshot.*


Submit Submit Submit Submit Submit Submit Submit Submit Submit Submit

inmotion hosting

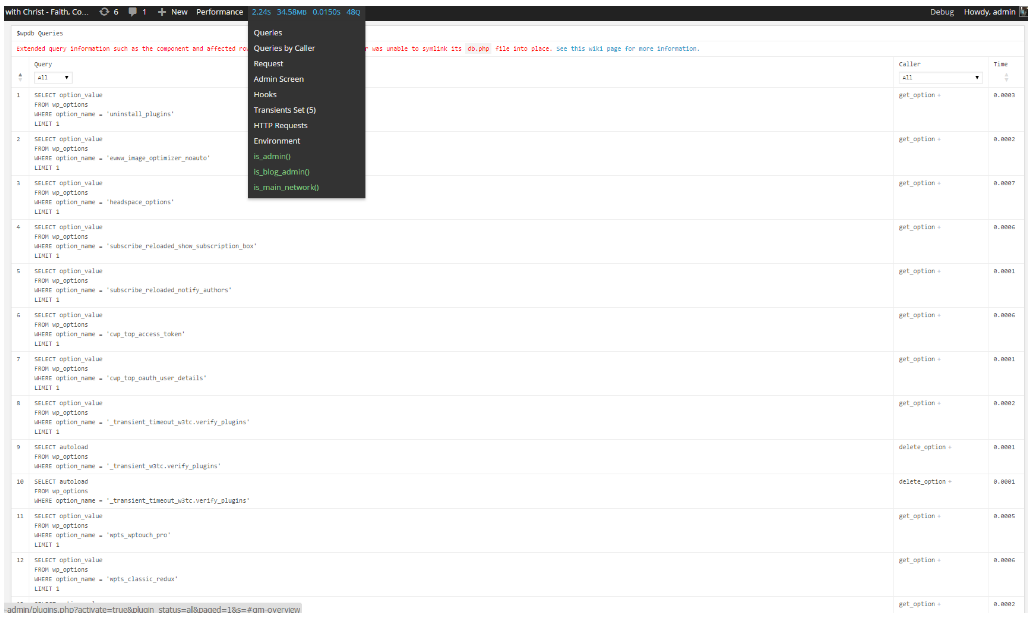
Below are the results of the Memory Viewer Plugin. For more help using this plugin, please see: Using InMotion Hosting's Memory Viewer WordPress Plugin
Like this plugin? Check out web hosting offered by InMotion Hosting

| Hook | Current Memory Usage | Max Memory Used at this point in the script | Current Server Time | Total Time since last hook | # MySQL Queries Run up to this point | # MySQL Queries since last hook | Time used by MySQL up to this point | Time used by MySQL since last hook |
|---|---|---|---|---|---|---|---|---|
| plugins_loaded | 21.17 MBs | 21.29 MBs | 1424353996.2366 | - | 4 | [show queries] | 0.000701904296875 | |
| setup_theme | 22.92 MBs | 23 MBs | 1424353996.2961 | 0.059537887573242 | 7 | 3 [show queries] | 0.0013570785522461 | 0.00065517425537109 |
| init | 24.36 MBs | 24.48 MBs | 1424353996.3237 | 0.027599096298218 | 13 | 6 [show queries] | 0.002615213394165 | 0.0012581348419189 |
| wp_loaded | 24.79 MBs | 25.01 MBs | 1424353996.3327 | 0.0090138912200928 | 16 | 3 [show queries] | 0.0034382343292236 | 0.00082302093505859 |
| get_header | 27.83 MBs | 28.02 MBs | 1424353996.3735 | 0.040771007537842 | 21 | 5 [show queries] | 0.0093274116516113 | 0.0058891773223877 |
| wp_head | 27.39 MBs | 28.11 MBs | 1424353996.3865 | 0.013024091720581 | 22 | 1 [show queries] | 0.009636402130127 | 0.00030899047851562 |
| loop_start | 27.37 MBs | 28.11 MBs | 1424353996.3882 | 0.0017149448394775 | 25 | 3 [show queries] | 0.010161638259888 | 0.00052523612976074 |
| loop_end | 27.84 MBs | 28.35 MBs | 1424353996.5966 | 0.20840215682983 | 25 | 0 | 0.010161638259888 | 0 |
| loop_start | 27.95 MBs | 28.35 MBs | 1424353996.6032 | 0.0065739154815674 | 26 | 1 [show queries] | 0.01057767868042 | 0.00041604042053223 |
| loop_end | 27.96 MBs | 28.35 MBs | 1424353996.607 | 0.003849983215332 | 26 | 0 | 0.01057767868042 | 0 |
| wp_footer | 28.85 MBs | 29.14 MBs | 1424353996.801 | 0.19395399093628 | 55 | 29 [show queries] | 0.14701914787292 | 0.1364414691925 |
| shutdown | 28.34 MBs | 32.95 MBs | 1424353997.7433 | 0.94227910041809 | 56 | 1 [show queries] | 0.14845609664917 | 0.0014369487762451 |

| | | |
|---|---|---|
| **Total Time to generate this page:** | **1.8161299228668212890625** | **100%** |
| **Total Time used by PHP:** | 1.6671311855316 | 91.7958% |
| **Total Time used by MySQL:** | 0.14845609664917 | 8.1743% |
| **Total Time used by Memory Viewer Plugin:** | 0.00054264068603516 | 0.0299% |

**Though WP Memory Viewer is said to work only up to WP version 3.2.1, I've tested it and it works fine on my latest stable Wordpress 4.1 based blog.**

## c) Wordpress Query Monitor



```
$wpdb Queries
Extended query information such as the component and affected ro[...] was unable to symlink its db.php file into place. See this wiki page for more information.

    Query                                                        Caller            Time
    All ▼                                                        All ▼

1   SELECT option_value                                          get_option +      0.0003
    FROM wp_options
    WHERE option_name = 'uninstall_plugins'
    LIMIT 1
2   SELECT option_value                                          get_option +      0.0002
    FROM wp_options
    WHERE option_name = 'ewww_image_optimizer_noauto'
    LIMIT 1
3   SELECT option_value                                          get_option +      0.0007
    FROM wp_options
    WHERE option_name = 'headspace_options'
    LIMIT 1
4   SELECT option_value                                          get_option +      0.0006
    FROM wp_options
    WHERE option_name = 'subscribe_reloaded_show_subscription_box'
    LIMIT 1
5   SELECT option_value                                          get_option +      0.0001
    FROM wp_options
    WHERE option_name = 'subscribe_reloaded_notify_authors'
    LIMIT 1
6   SELECT option_value                                          get_option +      0.0006
    FROM wp_options
    WHERE option_name = 'cwp_top_access_token'
    LIMIT 1
7   SELECT option_value                                          get_option +      0.0001
    FROM wp_options
    WHERE option_name = 'cwp_top_oauth_user_details'
    LIMIT 1
8   SELECT option_value                                          get_option +      0.0002
    FROM wp_options
    WHERE option_name = '_transient_timeout_w3tc.verify_plugins'
    LIMIT 1
9   SELECT autoload                                              delete_option +   0.0001
    FROM wp_options
    WHERE option_name = '_transient_w3tc.verify_plugins'
10  SELECT autoload                                              delete_option +   0.0001
    FROM wp_options
    WHERE option_name = '_transient_timeout_w3tc.verify_plugins'
11  SELECT option_value                                          get_option +      0.0005
    FROM wp_options
    WHERE option_name = 'wpts_wptouch_pro'
    LIMIT 1
12  SELECT option_value                                          get_option +      0.0006
    FROM wp_options
    WHERE option_name = 'wpts_classic_redux'
    LIMIT 1
                                                                 get_option +      0.0002
-admin/plugins.php?activate=true&plugin_status=all&paged=1&s=#qm-overview
```

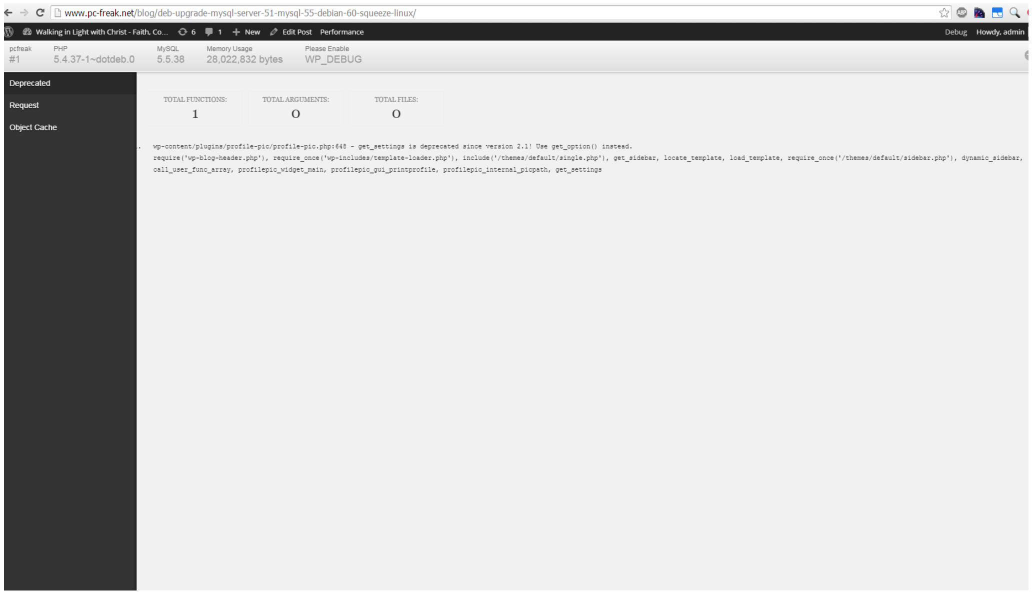
**Query Monitor** is a **debugging plugin for anyone developing with WordPress but also very helpful for anyone who want to track issues with plugins who use the database unefficient.**
It has some a**dvanced features not available in other debugging plugins, including automatic AJAX debugging and the ability to narrow down things by plugin or theme**.
You can view **plenty of precious statistics on how enabled plugins query the database server**, here is a short overview on its *Database Queries capabilities*:

- **Shows all database queries performed on the current page**
- **Shows affected rows and time for all queries**
- **Show notifications for slow queries and queries with errors**
- **Filter queries by query type (SELECT, UPDATE, DELETE, etc)**
- **Filter queries by component (WordPress core, Plugin X, Plugin Y, theme)**
- **Filter queries by calling function**
- **View aggregate query information grouped by component, calling function, and type**
- **Super advanced: Supports multiple instances of wpdb on one page**
- Once enabled from Plugins you will see it appear as a new menu on bottom Admin raw.

An important note to make here is latest ***Query Monitor extension fails when loaded on current latest Wordpress 4.1, to use it you will have to*** download and useolder Query Monitor plugin version 2.6.8 you can download it from here

*d) Debug Bar*

If you want you want a Memory Viewer like plugin for more complex used components memory debugging, reporting if (WP_DEBUG is set in **wp-config.php**) also check out Debug Bar .
For me **Debug Bar was very useful because it show me depreciated functions some plugins used, so I substituted the obsoleted function with new one**.


www.pc-freak.net/blog/deb-upgrade-mysql-server-51-mysql-55-debian-60-squeeze-linux/
Walking in Light with Christ - Faith, Co...
6
1
New
Edit Post
Performance
Debug
Howdy, admin
Deprecated
Request
Object Cache

pcfreak #1 | PHP 5.4.37-1~dotdeb.0 | MySQL 5.5.38 | Memory Usage 28,022,832 bytes | Please Enable WP_DEBUG

| TOTAL FUNCTIONS: | TOTAL ARGUMENTS: | TOTAL FILES: |
|---|---|---|
| 1 | 0 | 0 |

```
wp-content/plugins/profile-pic/profile-pic.php:648 - get_settings is deprecated since version 2.1! Use get_option() instead.
require('wp-blog-header.php'), require_once('wp-includes/template-loader.php'), include('/themes/default/single.php'), get_sidebar, locate_template, load_template, require_once('/themes/default/sidebar.php'), dynamic_sidebar,
call_user_func_array, profilepic_widget_main, profilepic_gui_printprofile, profilepic_internal_picpath, get_settings
```

4. Server Hardware hungry (slow) Wordpress plugins that you better not use

While spending time to Google for some fixes to **WP** slow query plugins - I've stumbled upon this post

giving a good list with Wordpress Plugins better off not to use because they will slow down your site
This is a publicly **well known list of WP plugins every Wordpress based site adminstrator should avoid, but until today I didn't know so my assumption is you don't know either ..**

Below **plugins are extremely database intensive mentioned in article that we should better (in all cases!) avoid**:

- **Dynamic Related Posts**
- **SEO Auto Links & Related Posts**
- **Yet Another Related Posts Plugin**
- **Similar Posts**
- **Contextual Related Posts**
- **Broken Link Checker** — Overwhelms even our robust caching layer with an inordinate amount of HTTP requests.
- **MyReviewPlugin** — Slams the database with a fairly significant amount of writes.
- **LinkMan** — Much like the MyReviewPlugin above, LinkMan utilizes an unscalable amount of database writes.
- **Fuzzy SEO Booster** — Causes MySQL issues as a site becomes more popular.
- **WP PostViews** — Inefficiently writes to the database on every page load. To track traffic in a more scalable manner, both the stats module in Automattic's **Jetpack plugin** and Google Analytics work wonderfully.
- **Tweet Blender** — Does not play nicely with our caching layer and can cause increased server load.

A good Complete list of known Wordpress slow plugins that will hammer down your wordpress performance is here

There are few alternatives to this plugins and when I have some free time I will download and test their alternatives but for now I plan the **plugins to stay disabled**.

**For the absolute WP Performance Optimization Freaks, its good to check out the native way to Debug a wordpress installation through using few embedded**

**variables**

```
  define('WP_DEBUG', true);
define('WP_DEBUG', false);
define('WP_DEBUG_LOG', true);
define('WP_DEBUG_DISPLAY', false);
define('SAVEQUERIES', true);
```

An article describing how you can use native WP debug variables is here

**Happy Optimizing ! :)**