PROFILING PHP

A DIVE INTO YOUR APPLICATION

Dennis de Greef / @dennisdegreef

PHPAmersfoort Meetup March 2015

WHAT IS PROFILING?

WIKIPEDIA

profiling is a form of **dynamic program analysis** that **measures**, for example, the space (memory) or time complexity of a program, the usage of particular instructions, or the **frequency and duration of function calls**. Most commonly, profiling information serves to aid program optimization.

SO... DYNAMIC PROGRAM ANALYSIS?

YEAH...

LETS FIRST LOOK AT IT'S COUNTERPART...

STATIC ANALYSIS

WIKIPEDIA

Static program analysis is the analysis of computer software that is performed **without actually executing programs**

The term is usually applied to the analysis performed by an automated tool, with human analysis being called program understanding, program comprehension or code review.

STATIC ANALYSIS TOOLS

There are a set of tools which perform static code analysis. These tools can be integrated within an *automated build*.

- PHP Mess Detector
- PHP Copy/Paste Detector
- PHP CodeSniffer
- PHP Dead Code Detector

There is a nice page containing a predefined set of tools for a build to be found at Jenkins PHP

BUT...

THESE TOOLS ONLY ANALYSE HOW YOUR CODE IS STRUCTURED, NOT HOW IT BEHAVES.

DYNAMIC ANALYSIS

WIKIPEDIA

The analysis of computer software that is performed by **executing programs on a real or virtual processor**.

For dynamic program analysis to be effective, the target program must be executed with **sufficient test inputs** to produce interesting behavior.

Use of software testing measures such as code coverage helps ensure that an adequate slice of the program's set of possible behaviors has been observed.

CALLSTACK

A callstack is the **order** in which statements are **exectuted**. An example commonly known, is an **exception trace**. This trace shows all the statements executed before an exception is thrown.

CALLGRAPH

A callgraph is a **visual representation** of a callstack.
In large applications this graph can give you better **insight** on how an application is wired.

PROFILING DATA

Usually, the data gathered with a profiler can be represented in stacks or graphs.
They can include information regarding **memory- and cpu-usage**.

WHY?

REASONS

- Debugging CPU performance
- Debugging memory performance
- Debugging IO performance
- See which function is called how many times
- Gain insight of the black box that is the application

REASONS

- We live in a digital age where we want everything instantly.
- According to a case study from Radware, 51 percent of online shoppers in the U.S claimed if a site is too slow they will not complete a purchase.
- Nowadays, search engine indexing also accounts for page load.

The psychology of web performance
SEO 101: How important is site speed in 2014?
Case study from Radware

WARNING!

> *Premature optimization is the root of all evil*
> *-- Donald Knuth*

Only perform optimization when there is a need to.

CAUSE OF ISSUES

COMMON ISSUES

- Network slowdown
- Datastore slowdown
- External resources (API, Filesystems, Network sockets, etc)
- Bad code(tm)
- Didn't RTFM(tm)

ACTIVE VS PASSIVE

Profiling PHP Part 1 (Davey Shafik)

ACTIVE PROFILER

- Used during development
- Gather more information than passive profilers (like variables/values)
- Performance impact is bigger
- Should _NOT_ be used in production
- Example: Xdebug

PASSIVE PROFILER

- Minimal impact on performance
- Gathers less but sufficient information to diagnose issues (Only records function calls and cpu + mem)
- Examples: XHProf / UProfiler, New Relic, Blackfire.io

XDEBUG

XDEBUG

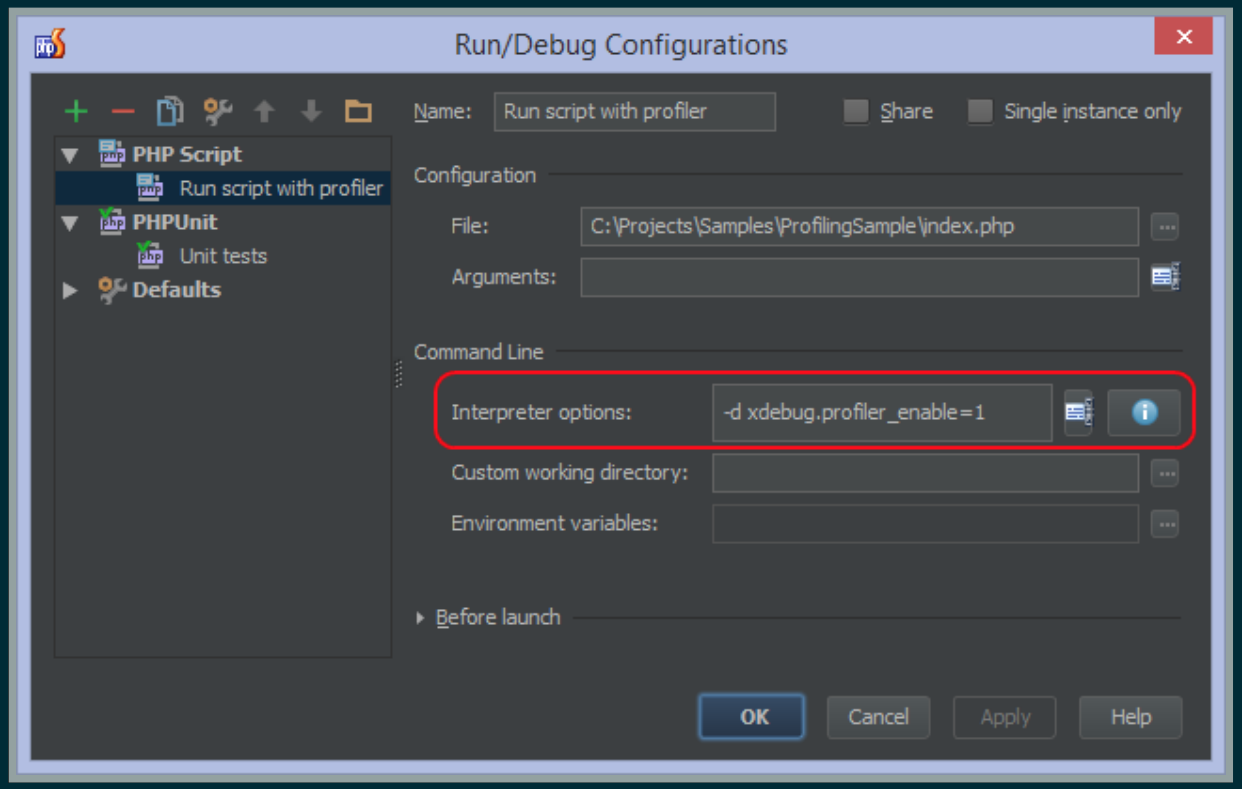
- Generates cachegrind files (like Valgrind for C)
- Can be analysed by KCacheGrind among others
- Cachegrind files are relatively big in size
- Also a developer tool for breakpoints and remote debugging
- Active profiler

ENABLE XDEBUG PROFILING

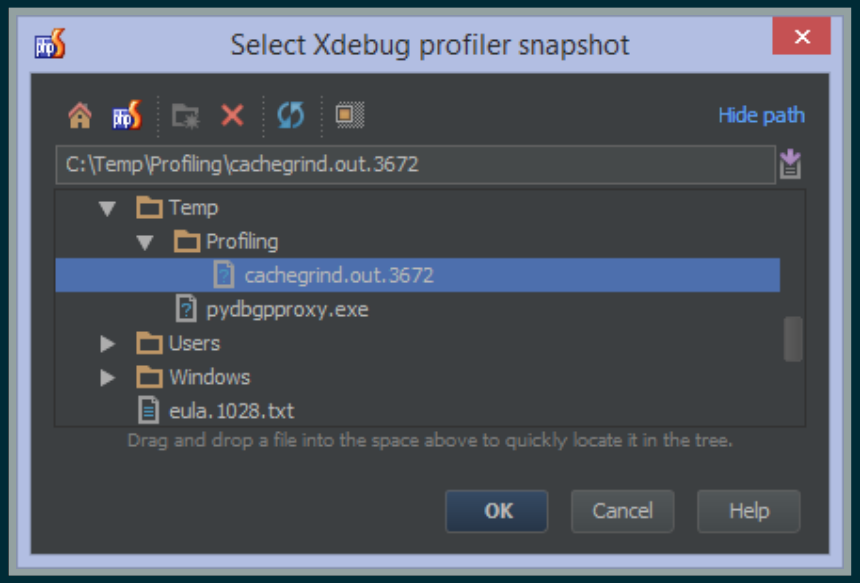
```
# php.ini settings
xdebug.profiler_enable=1
xdebug.profiler_output_dir=/path/to/store/snapshots
xdebug.profiler_enable_trigger=1
```

XDEBUG WITH KCACHEGRIND

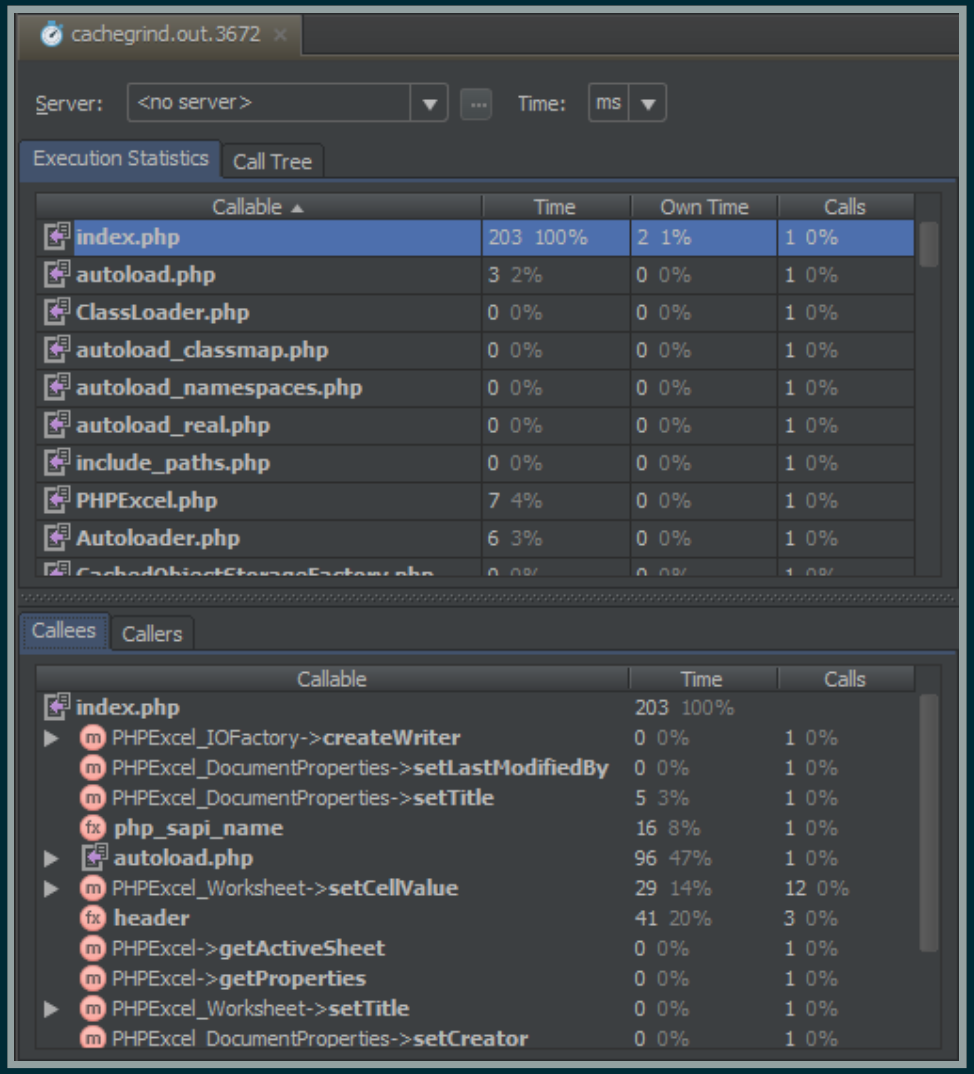
XDEBUG IN PHPSTORM



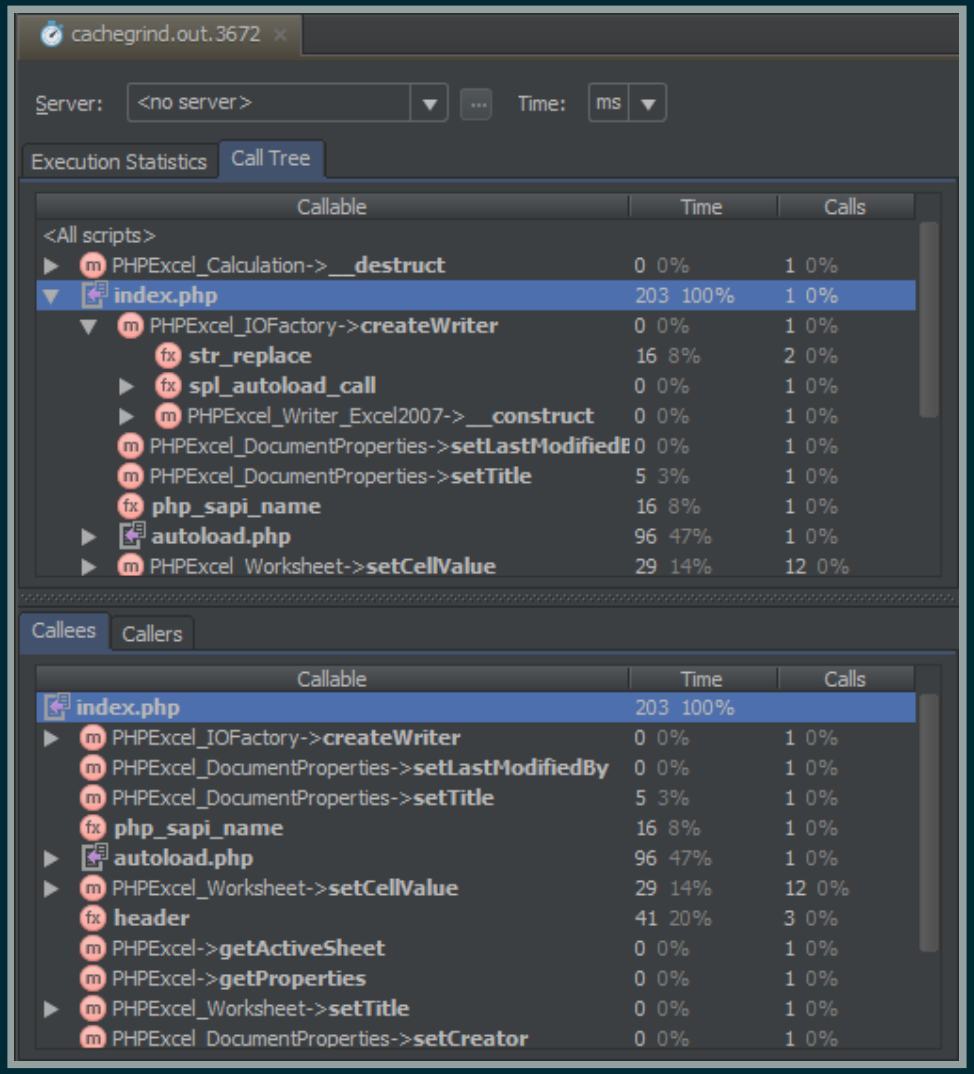
XDEBUG IN PHPSTORM



XDEBUG IN PHPSTORM



XDEBUG IN PHPSTORM



XHPROF

XHPROF

- Developed by Facebook and released as open-source in 2009
- PECL extension
- Lightweight for being a passive profiler
- Includes webgui for reviewing and comparing profiling data

INSTALLATION

```
# Linux (using apt or yum)
apt-get install -y php5-xhprof

# OSX (using homebrew)
brew install php56-xhprof

# For Windows, use PECL or download a .dll somewhere, or compile for your own
```

WORDPRESS EXAMPLE

```php
// index.php

xhprof_enable(XHPROF_FLAGS_CPU + XHPROF_FLAGS_MEMORY);

/** Loads the WordPress Environment and Template */
require( dirname( __FILE__ ) . '/wp-blog-header.php' );

$xhprof_data = xhprof_disable();

include_once 'xhprof_lib/utils/xhprof_lib.php';
include_once 'xhprof_lib/utils/xhprof_runs.php';

$xhprof_runs = new XHProfRuns_Default();

$run_id = $xhprof_runs->save_run($xhprof_data, "xhprof_foo");
```

CALLSTACK

Run Report

Run #54693d532b53a: XHProf Run (Namespace=xhprof_foo)

Tip

Click a function name below to drill down.

View Top Level Run Report

Overall Summary

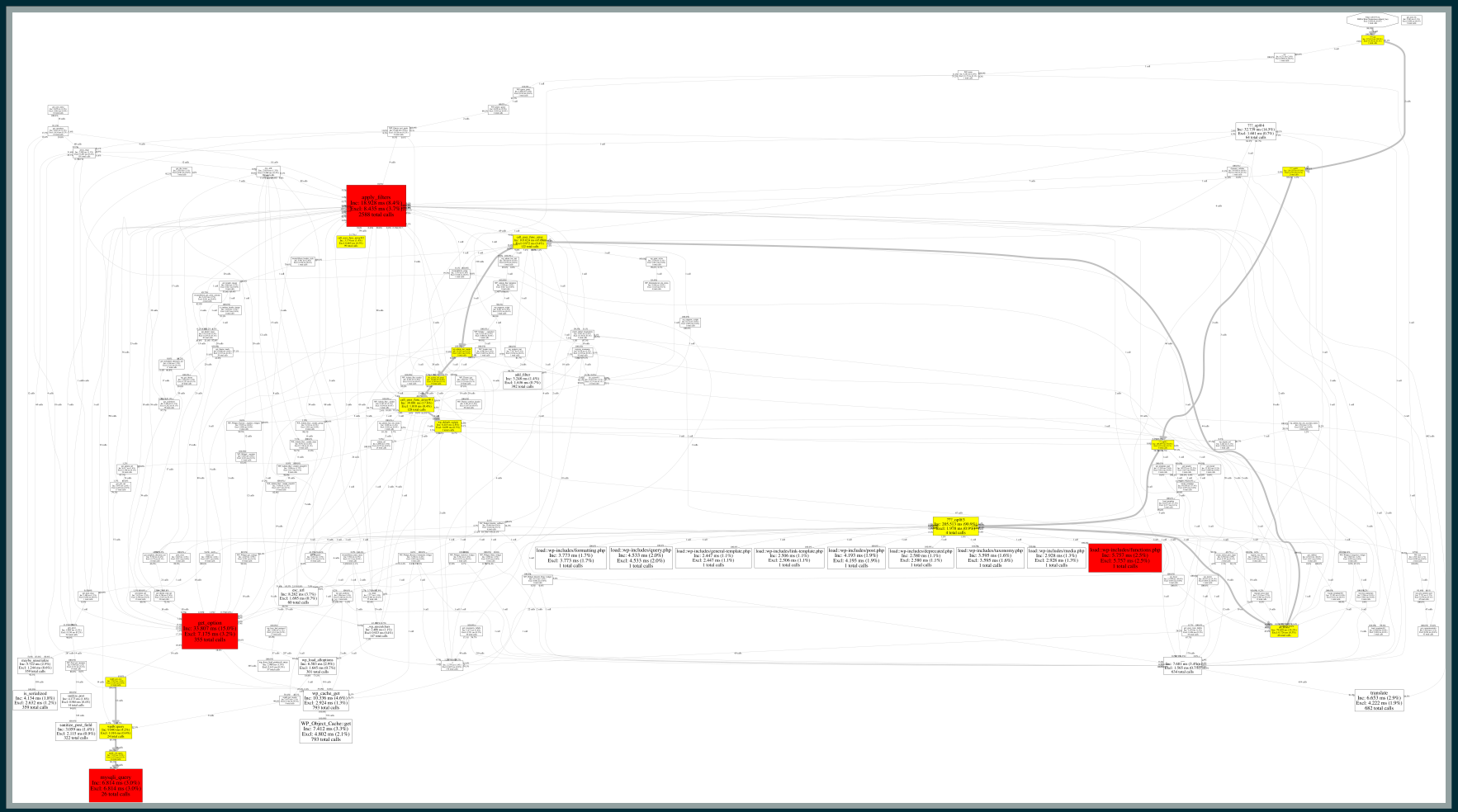
Total Incl. Wall Time (microsec): 226,073 microsecs
Total Incl. CPU (microsecs): 220,064 microsecs
Total Incl. MemUse (bytes): 18,903,968 bytes
Total Incl. PeakMemUse (bytes): 19,068,272 bytes
Number of Function Calls: 34,442

[View Full Callgraph]

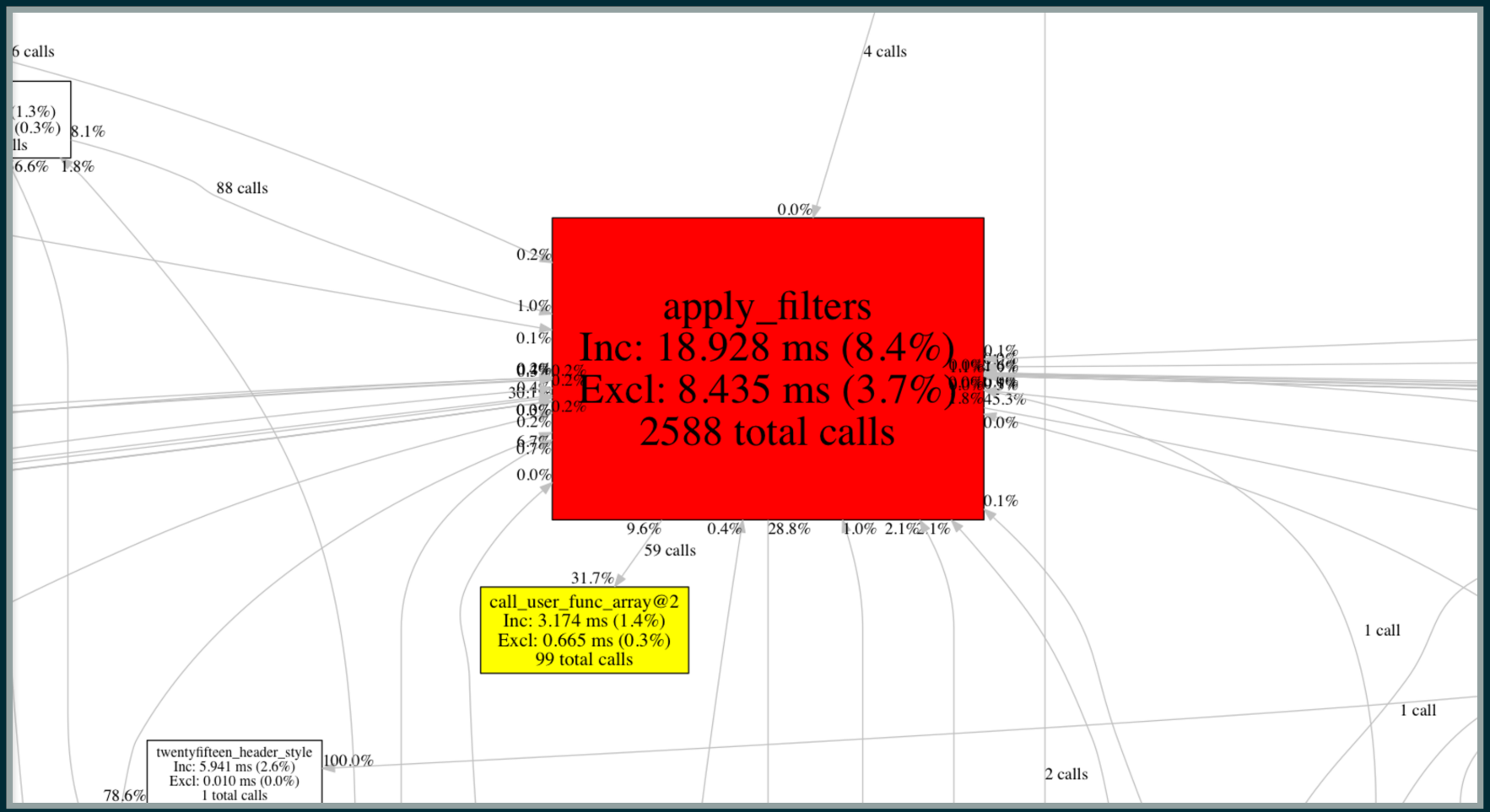
Displaying top 100 functions: Sorted by Incl. Wall Time (microsec) [display all]

Function Name	Calls	Calls%	Incl. Wall Time (microsec)	IWall%	Excl. Wall Time (microsec)	EWall%	Incl. CPU (microsecs)	ICpu%	Excl. CPU (microsec)	ECPU%	Incl. MemUse (bytes)	IMemUse%	Excl. MemUse (bytes)	EMemUse%	Incl. PeakMemUse (bytes)	IPeakMemUse%	Ex PeakMemU (byt
main()	1	0.0%	226,073	100.0%	94	0.0%	220,064	100.0%	56	0.0%	18,903,968	100.0%	-1,112	-0.0%	19,068,272	100.0%	
??? op	1	0.0%	225,833	99.9%	130	0.1%	219,826	99.9%	120	0.1%	18,899,152	100.0%	-29,272	-0.2%	19,063,976	100.0%	1,1
??? op@1	2	0.0%	219,214	97.0%	125	0.1%	214,155	97.3%	91	0.0%	18,591,840	98.3%	-11,408	-0.1%	18,775,328	98.5%	3,9
??? op@2	5	0.0%	208,097	92.0%	161	0.1%	203,041	92.3%	130	0.1%	18,005,504	95.2%	-69,144	-0.4%	18,211,600	95.5%	1
??? op@3	4	0.0%	205,513	90.9%	1,978	0.9%	200,458	91.1%	1,731	0.8%	17,959,792	95.0%	-543,376	-2.9%	17,908,000	93.9%	14,9
call user func array	122	0.4%	103,024	45.6%	972	0.4%	99,307	45.1%	936	0.4%	2,939,664	15.6%	35,112	0.2%	3,151,520	16.5%	14,5
locate template	5	0.0%	84,558	37.4%	76	0.0%	81,732	37.1%	65	0.0%	1,440,512	7.6%	-31,320	-0.2%	1,585,656	8.3%	2,0
load template	3	0.0%	84,427	37.3%	177	0.1%	81,603	37.1%	166	0.1%	1,471,064	7.8%	-24,920	-0.1%	1,582,936	8.3%	8
do action	46	0.1%	79,638	35.2%	726	0.3%	78,407	35.6%	584	0.3%	2,349,696	12.4%	14,288	0.1%	2,234,128	11.7%	8,6
get header	1	0.0%	50,972	22.5%	13	0.0%	48,679	22.1%	12	0.0%	855,432	4.5%	888	0.0%	964,568	5.1%	
call user func array@1	128	0.4%	39,881	17.6%	1,010	0.4%	39,013	17.7%	1,001	0.5%	784,792	4.2%	18,568	0.1%	1,032,888	5.4%	13,4
get option	355	1.0%	33,807	15.0%	7,175	3.2%	32,762	14.9%	5,236	2.4%	159,544	0.8%	29,320	0.2%	287,480	1.5%	29,1
??? op@4	64	0.2%	32,739	14.5%	1,681	0.7%	30,499	13.9%	1,401	0.6%	1,710,912	9.1%	66,032	0.3%	1,460,688	7.7%	103,5
wp head	1	0.0%	23,944	10.6%	38	0.0%	23,945	10.9%	37	0.0%	326,304	1.7%	1,056	0.0%	294,624	1.5%	5
get sidebar	1	0.0%	22,521	10.0%	13	0.0%	20,227	9.2%	11	0.0%	407,400	2.2%	1,192	0.0%	418,928	2.2%	
locate template@1	1	0.0%	22,505	10.0%	21	0.0%	20,212	9.2%	21	0.0%	405,312	2.1%	-10,208	-0.1%	418,928	2.2%	
load template@1	1	0.0%	22,468	9.9%	82	0.0%	20,174	9.2%	76	0.0%	414,752	2.2%	-5,816	-0.0%	418,928	2.2%	
get footer	1	0.0%	21,010	9.3%	9	0.0%	20,775	9.4%	7	0.0%	314,512	1.7%	1,192	0.0%	352,472	1.8%	

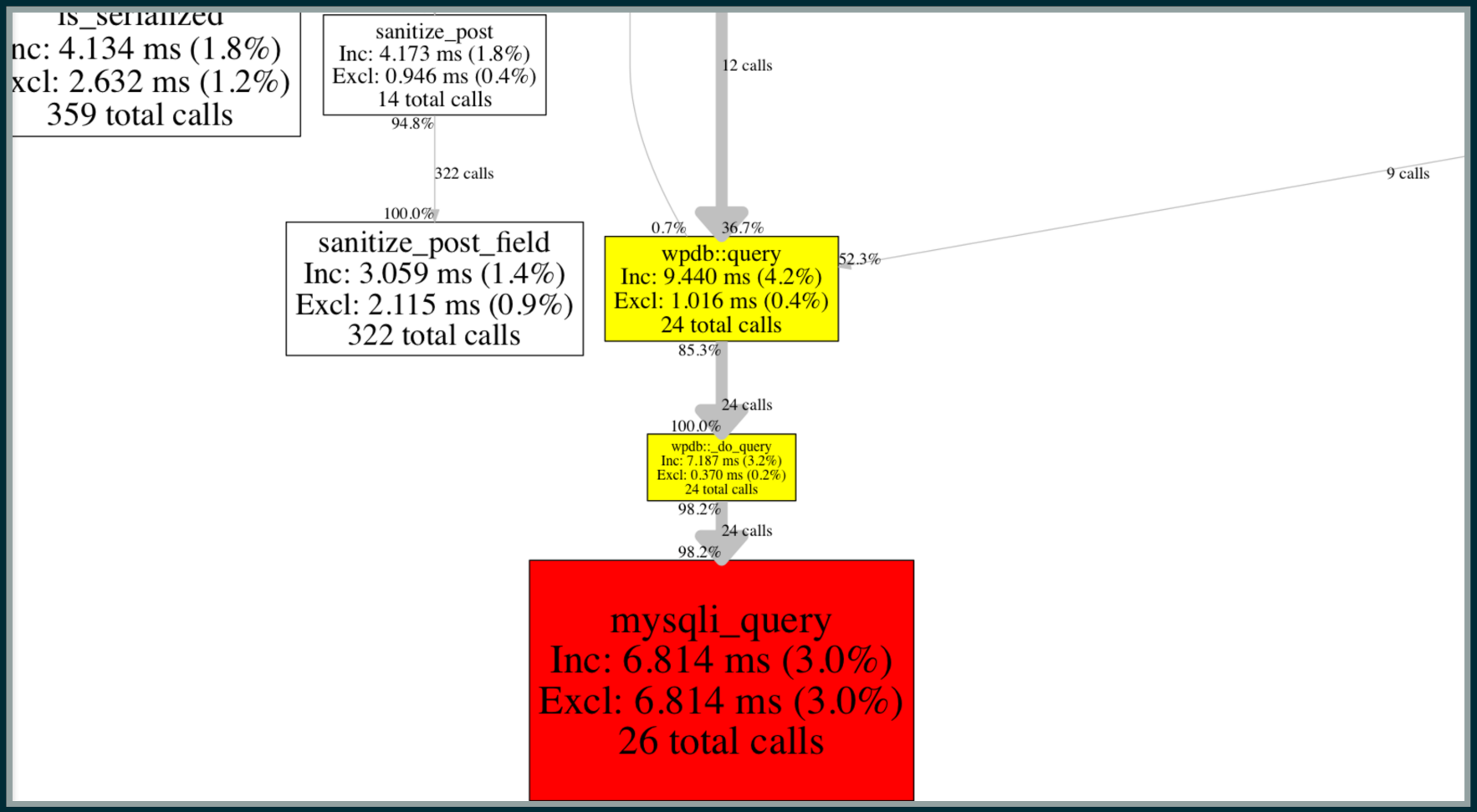
CALLGRAPH


apply_filters
Inc: 18.928 ms (8.4%)
Excl: 8.435 ms (3.7%)
2588 total calls
add_filter
Inc: 3.244 ms (1.4%)
Excl: 1.636 ms (0.7%)
382 total calls
load::wp-includes/formatting.php
Inc: 3.773 ms (1.7%)
Excl: 3.773 ms (1.7%)
1 total calls
load::wp-includes/query.php
Inc: 4.533 ms (2.0%)
Excl: 4.533 ms (2.0%)
1 total calls
load::wp-includes/general-template.php
Inc: 2.447 ms (1.1%)
Excl: 2.447 ms (1.1%)
1 total calls
load::wp-includes/link-template.php
Inc: 2.506 ms (1.1%)
Excl: 2.506 ms (1.1%)
1 total calls
load::wp-includes/post.php
Inc: 4.193 ms (1.9%)
Excl: 4.193 ms (1.9%)
1 total calls
load::wp-includes/shortcodes.php
Inc: 2.588 ms (1.1%)
Excl: 2.588 ms (1.1%)
1 total calls
load::wp-includes/taxonomy.php
Inc: 3.949 ms (1.8%)
Excl: 3.395 ms (1.6%)
1 total calls
load::wp-includes/theme.php
Inc: 2.928 ms (1.3%)
Excl: 2.928 ms (1.3%)
1 total calls
load::wp-includes/functions.php
Inc: 5.737 ms (2.5%)
Excl: 5.737 ms (2.5%)
1 total calls
get_option
Inc: 35.905 ms (15.9%)
Excl: 7.175 ms (3.2%)
355 total calls
wp_load_alloptions
Inc: 8.585 ms (3.8%)
Excl: 1.665 ms (0.7%)
263 total calls
wp_cache_get
Inc: 10.336 ms (4.6%)
Excl: 2.924 ms (1.3%)
793 total calls
WP_Object_Cache::get
Inc: 7.412 ms (3.3%)
Excl: 4.802 ms (2.1%)
793 total calls
mysqli_query
Inc: 6.814 ms (3.0%)
Excl: 6.814 ms (3.0%)
26 total calls
translate
Inc: 6.653 ms (2.9%)
Excl: 4.222 ms (1.9%)
682 total calls

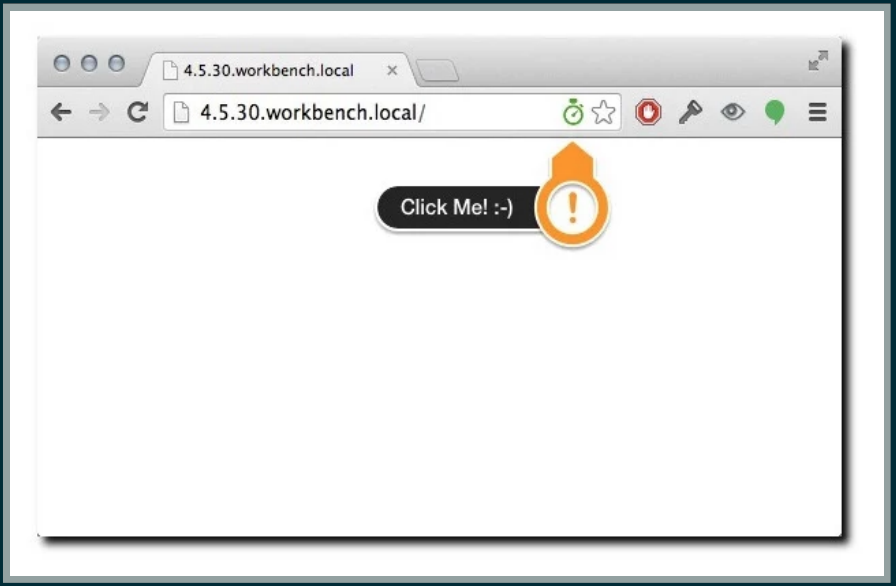
CALLGRAPH


6 calls
4 calls
88 calls
apply_filters
Inc: 18.928 ms (8.4%)
Excl: 8.435 ms (3.7%)
2588 total calls
59 calls
call_user_func_array@2
Inc: 3.174 ms (1.4%)
Excl: 0.665 ms (0.3%)
99 total calls
1 call
1 call
2 calls
twentyfifteen_header_style
Inc: 5.941 ms (2.6%)
Excl: 0.010 ms (0.0%)
1 total calls

CALLGRAPH


is_serialized
nc: 4.134 ms (1.8%)
xcl: 2.632 ms (1.2%)
359 total calls
sanitize_post
Inc: 4.173 ms (1.8%)
Excl: 0.946 ms (0.4%)
14 total calls
94.8%
322 calls
100.0%
sanitize_post_field
Inc: 3.059 ms (1.4%)
Excl: 2.115 ms (0.9%)
322 total calls
12 calls
0.7%
36.7%
wpdb::query
Inc: 9.440 ms (4.2%)
Excl: 1.016 ms (0.4%)
24 total calls
52.3%
9 calls
85.3%
24 calls
100.0%
wpdb::_do_query
Inc: 7.187 ms (3.2%)
Excl: 0.370 ms (0.2%)
24 total calls
98.2%
24 calls
98.2%
mysqli_query
Inc: 6.814 ms (3.0%)
Excl: 6.814 ms (3.0%)
26 total calls

USEFUL TOOLS

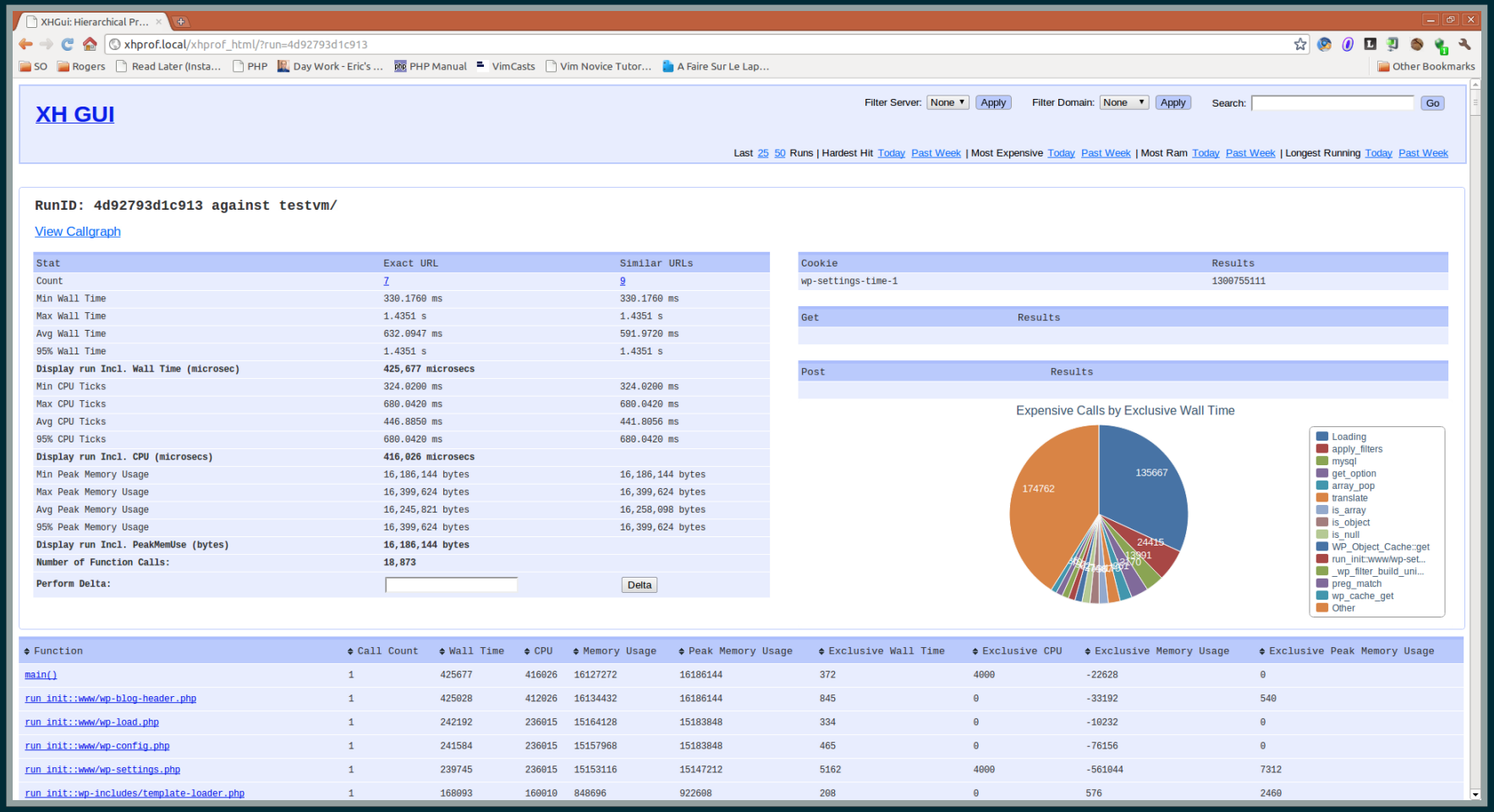


- XHProf Helper for Chrome
- XHProf Helper for Firefox
- Sets $_COOKIE['_profile'] to 1

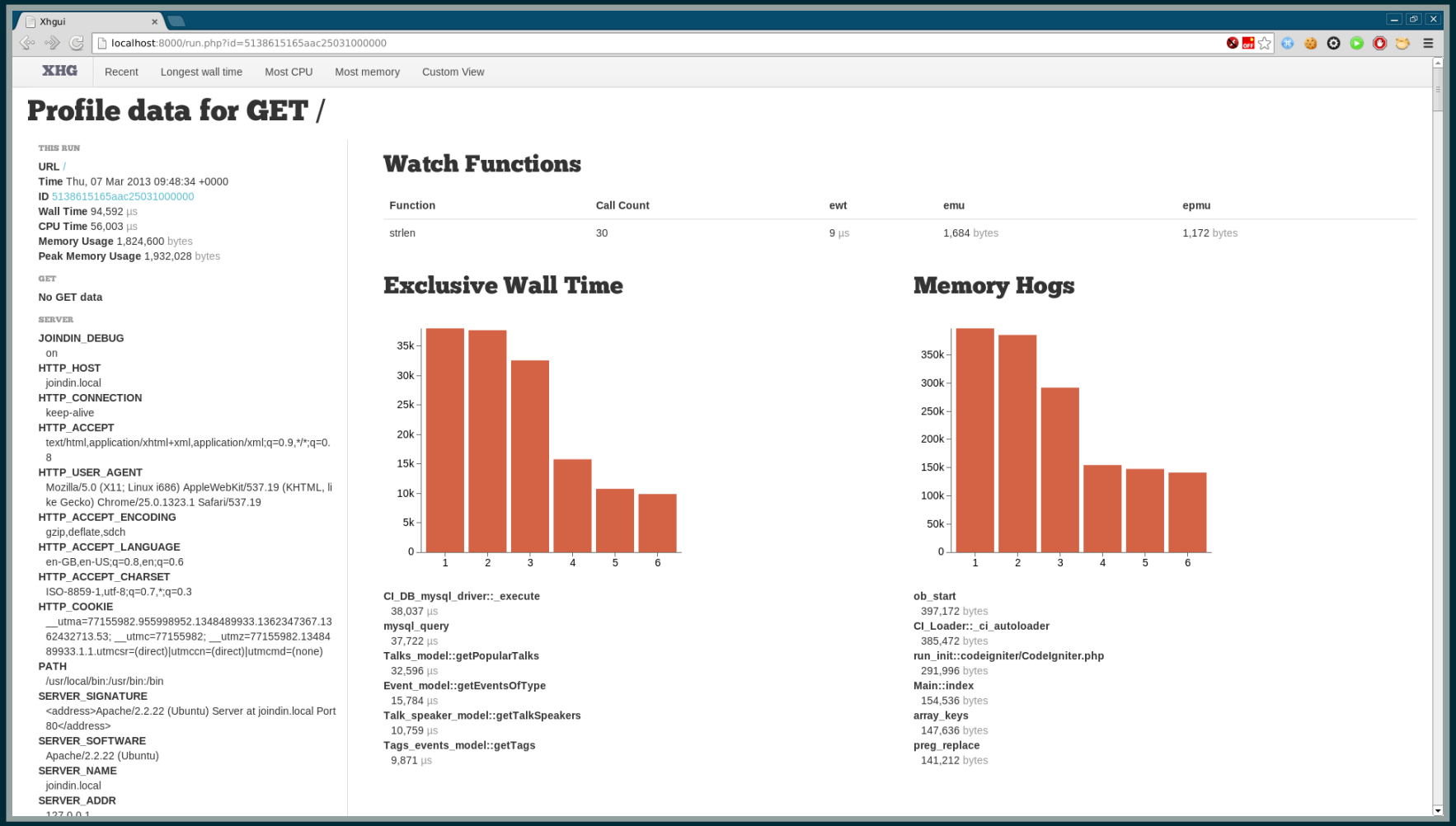
XHGUI

- Web frontend for profile data
- Requires MongoDB
- Shows callstacks
- Shows callgraphs
- Can compare different runs

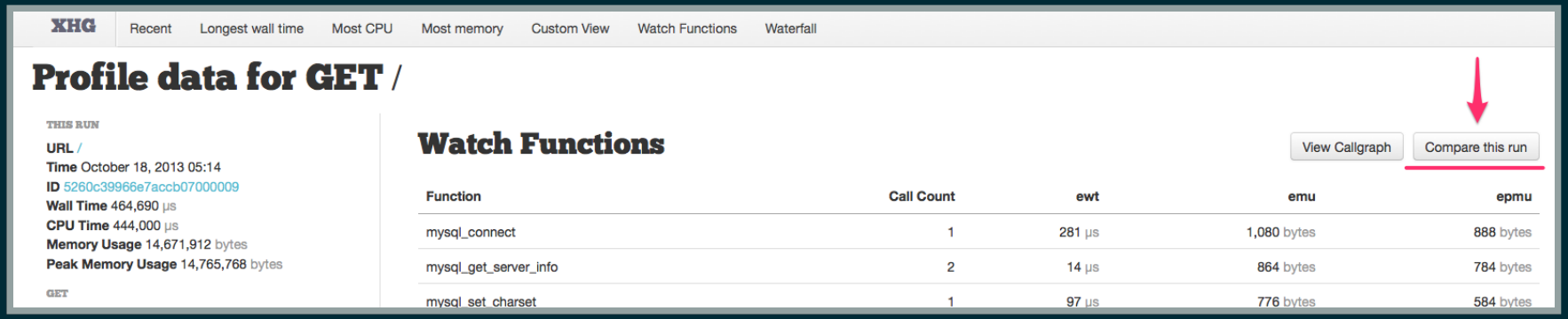
XHGUI


XHGui: Hierarchical Pr...
xhprof.local/xhprof_html/?run=4d92793d1c913
XH GUI
Filter Server: None Apply
Filter Domain: None Apply
Search: Go
Last 25 50 Runs | Hardest Hit Today Past Week | Most Expensive Today Past Week | Most Ram Today Past Week | Longest Running Today Past Week
RunID: 4d92793d1c913 against testvm/
View Callgraph
Stat | Exact URL | Similar URLs
Count | 7 | 9
Min Wall Time | 330.1760 ms | 330.1760 ms
Max Wall Time | 1.4351 s | 1.4351 s
Avg Wall Time | 632.0947 ms | 591.9720 ms
95% Wall Time | 1.4351 s | 1.4351 s
Display run Incl. Wall Time (microsec) | 425,677 microsecs
Min CPU Ticks | 324.0200 ms | 324.0200 ms
Max CPU Ticks | 680.0420 ms | 680.0420 ms
Avg CPU Ticks | 446.8850 ms | 441.8056 ms
95% CPU Ticks | 680.0420 ms | 680.0420 ms
Display run Incl. CPU (microsecs) | 416,026 microsecs
Min Peak Memory Usage | 16,186,144 bytes | 16,186,144 bytes
Max Peak Memory Usage | 16,399,624 bytes | 16,399,624 bytes
Avg Peak Memory Usage | 16,245,821 bytes | 16,258,098 bytes
95% Peak Memory Usage | 16,399,624 bytes | 16,399,624 bytes
Display run Incl. PeakMemUse (bytes) | 16,186,144 bytes
Number of Function Calls: | 18,873
Perform Delta: Delta
Cookie | Results
wp-settings-time-1 | 1300755111
Get | Results
Post | Results
Expensive Calls by Exclusive Wall Time
135667
174762
24415
13991
Loading
apply_filters
mysql
get_option
array_pop
translate
is_array
is_object
is_null
WP_Object_Cache::get
run_init:www/wp-set...
_wp_filter_build_uni...
preg_match
wp_cache_get
Other
Function | Call Count | Wall Time | CPU | Memory Usage | Peak Memory Usage | Exclusive Wall Time | Exclusive CPU | Exclusive Memory Usage | Exclusive Peak Memory Usage
main() | 1 | 425677 | 416026 | 16127272 | 16186144 | 372 | 4000 | -22628 | 0
run_init::www/wp-blog-header.php | 1 | 425028 | 412026 | 16134432 | 16186144 | 845 | 0 | -33192 | 540
run_init::www/wp-load.php | 1 | 242192 | 236015 | 15164128 | 15183848 | 334 | 0 | -10232 | 0
run_init::www/wp-config.php | 1 | 241584 | 236015 | 15157968 | 15183848 | 465 | 0 | -76156 | 0
run_init::www/wp-settings.php | 1 | 239745 | 236015 | 15153116 | 15147212 | 5162 | 4000 | -561044 | 7312
run_init::wp-includes/template-loader.php | 1 | 168093 | 160010 | 848696 | 922608 | 208 | 0 | 576 | 2460

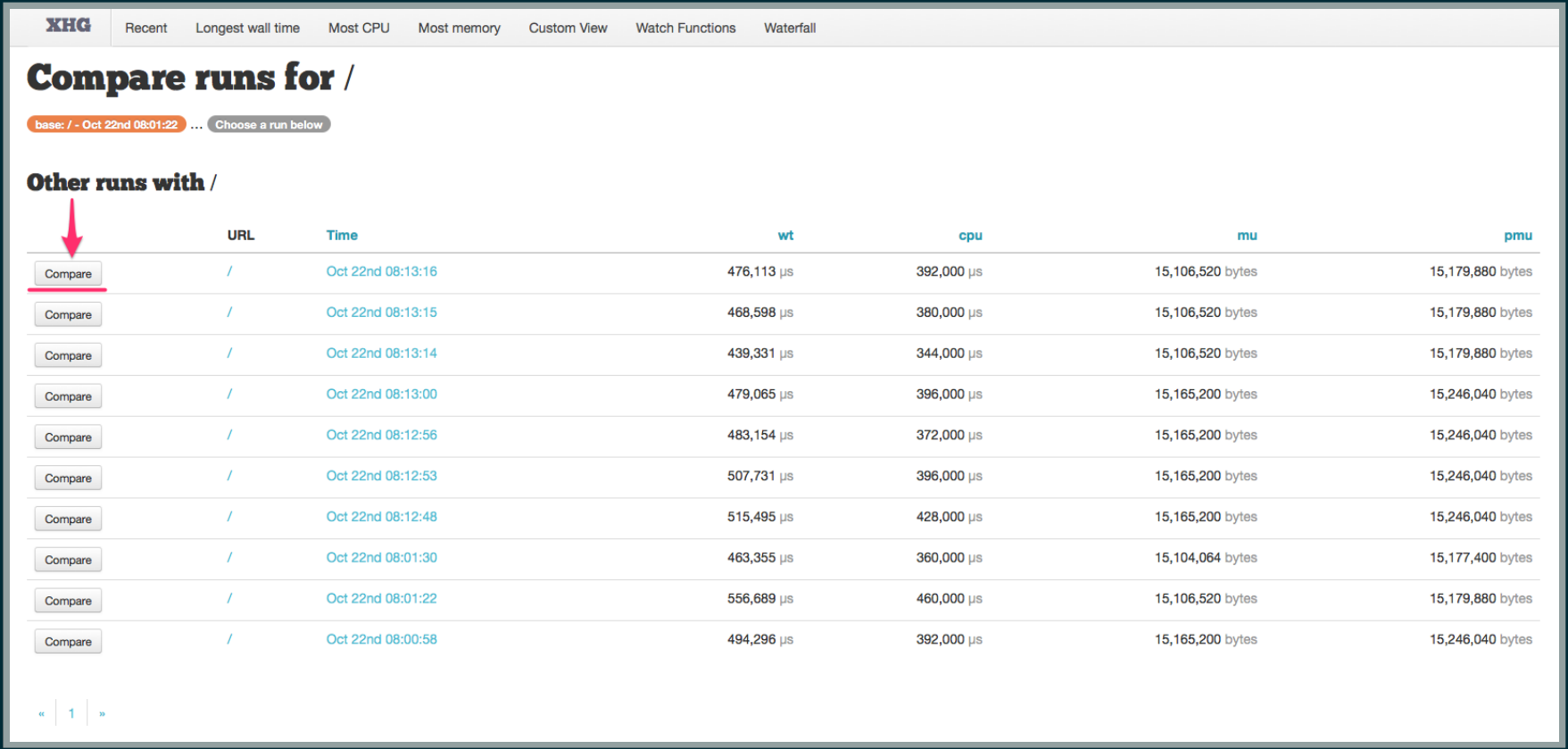
XHGUI


Xhgui
localhost:8000/run.php?id=5138615165aac25031000000
XHG
Recent
Longest wall time
Most CPU
Most memory
Custom View
Profile data for GET /
THIS RUN
URL /
Time Thu, 07 Mar 2013 09:48:34 +0000
ID 5138615165aac25031000000
Wall Time 94,592 µs
CPU Time 56,003 µs
Memory Usage 1,824,600 bytes
Peak Memory Usage 1,932,028 bytes
GET
No GET data
SERVER
JOINDIN_DEBUG
on
HTTP_HOST
joindin.local
HTTP_CONNECTION
keep-alive
HTTP_ACCEPT
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
HTTP_USER_AGENT
Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.19 (KHTML, like Gecko) Chrome/25.0.1323.1 Safari/537.19
HTTP_ACCEPT_ENCODING
gzip,deflate,sdch
HTTP_ACCEPT_LANGUAGE
en-GB,en-US;q=0.8,en;q=0.6
HTTP_ACCEPT_CHARSET
ISO-8859-1,utf-8;q=0.7,*;q=0.3
HTTP_COOKIE
__utma=77155982.955998952.1348489933.1362347367.1362432713.53; __utmc=77155982; __utmz=77155982.1348489933.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none)
PATH
/usr/local/bin:/usr/bin:/bin
SERVER_SIGNATURE
<address>Apache/2.2.22 (Ubuntu) Server at joindin.local Port 80</address>
SERVER_SOFTWARE
Apache/2.2.22 (Ubuntu)
SERVER_NAME
joindin.local
SERVER_ADDR
127.0.0.1
Watch Functions
Function
Call Count
ewt
emu
epmu
strlen
30
9 µs
1,684 bytes
1,172 bytes
Exclusive Wall Time
35k
30k
25k
20k
15k
10k
5k
0
1
2
3
4
5
6
CI_DB_mysql_driver::_execute
38,037 µs
mysql_query
37,722 µs
Talks_model::getPopularTalks
32,596 µs
Event_model::getEventsOfType
15,784 µs
Talk_speaker_model::getTalkSpeakers
10,759 µs
Tags_events_model::getTags
9,871 µs
Memory Hogs
350k
300k
250k
200k
150k
100k
50k
0
1
2
3
4
5
6
ob_start
397,172 bytes
CI_Loader::_ci_autoloader
385,472 bytes
run_init::codeigniter/CodeIgniter.php
291,996 bytes
Main::index
154,536 bytes
array_keys
147,636 bytes
preg_replace
141,212 bytes

XHGUI COMPARE



XHGUI COMPARE



XHGUI COMPARE DIFFERENCE

XHG | Recent | Longest wall time | Most CPU | Most memory | Custom View | Watch Functions | Waterfall

Compare runs for /

base: / - Oct 25th 18:32:47 ... head: / - Oct 25th 18:32:55 change reverse

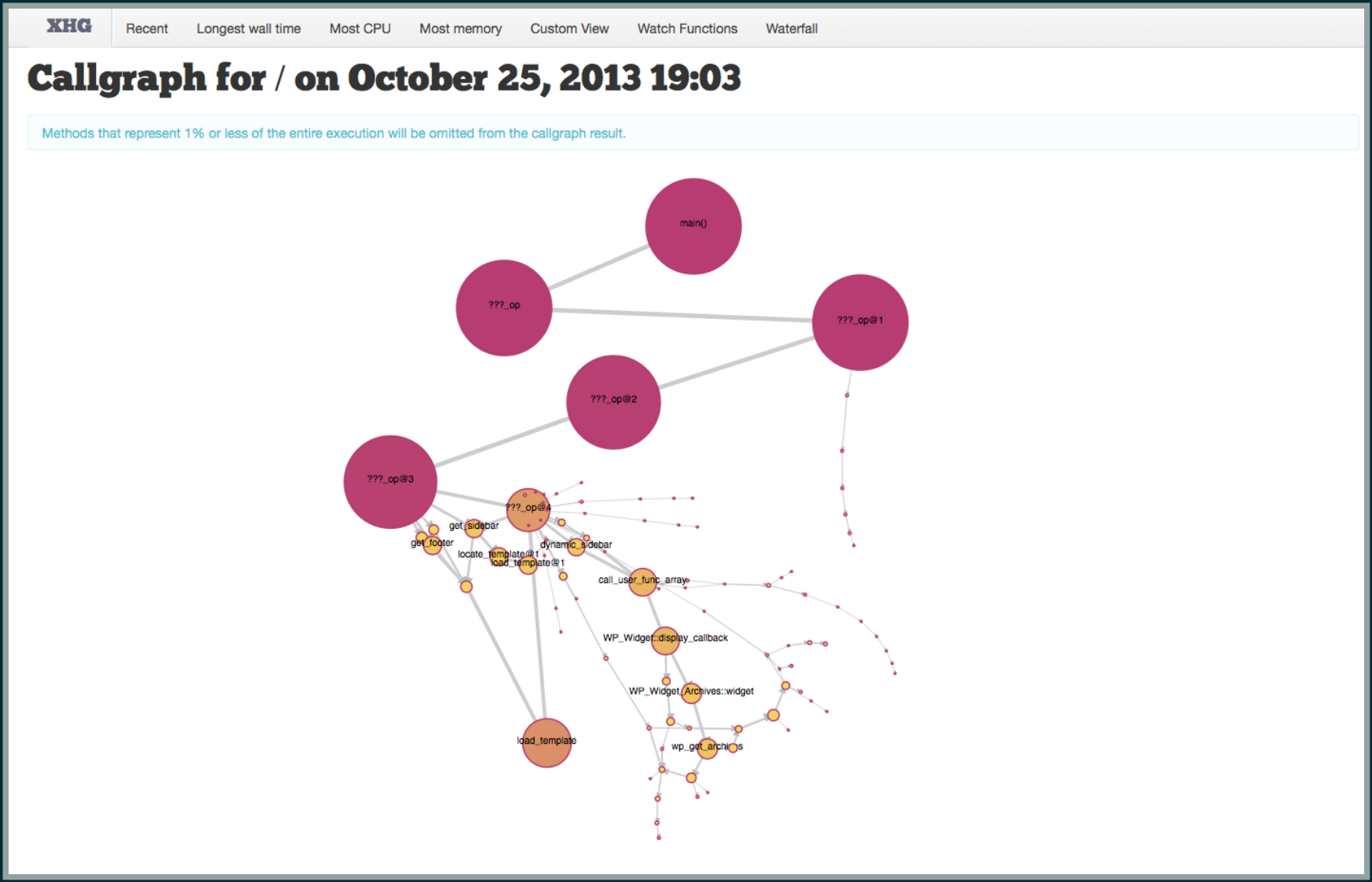
Summary

	Number of function calls	Inclusive Wall time	Inclusive CPU	Inclusive Memory	Inclusive Peak Memory
base 526ab94f66e7acae14000000	52084	657,508 µs	460,000 µs	14,712,072 bytes	14,799,352 bytes
head 526ab95766e7acae14000002	51230	519,137 µs	416,000 µs	14,643,288 bytes	14,683,096 bytes
Difference	-854	-138,371 µs	-44,000 µs	-68,784 bytes	-116,256 bytes
Difference %	98 %	79 %	90 %	100 %	99 %

Details

Function	Call Count	Exclusive Wall Time	Exclusive CPU	Exclusive Memory Usage	Exclusive Peak Memory Usage	Inclusive Wall Time	Inclusive CPU	Inclusive Memory Usage	Inclusive Peak Memory Usage
register_shutdown_function	0	-35	0	0	0	-35	0	0	0
load::wordpress/index.php	0	-813	0	0	0	-813	0	0	0
define	0	-6	0	32	0	-6	0	32	0
load::./wp-blog-header.php	0	241	0	-8	0	241	0	-8	0
dirname	0	-4	0	16	0	-4	0	16	0
load::wordpress/wp-load.php	0	-489	0	16	-128	-489	0	16	-128
error_reporting	0	-5	0	-8	-8	-5	0	-8	-8
file_exists	0	-2,200	0	0	0	-2,200	0	0	0
load::wordpress/wp-config.php	0	-493	0	24	-40	-493	0	24	-40
defined	-43	266	4,000	-3,048	-1,768	266	4,000	-3,048	-1,768

XHGUI CALLSTACK



LINKO

(LINK ZERO)

LINK0/PROFILER

- Focused on XHProf and Uprofiler
- Released v1.0.0 last week!
- Has multiple persistence layers for storing profiles
  - Memory
  - Flysystem
  - Zend\Db\Adapter
  - MongoDB (XHGui compatible)
- Available on composer/packagist
- Symfony2 bundle available to hook into kernel events
- Fully object-orientated
- 100% code coverage (as far as that's relevant)
- http://github.com/link0/profiler

GETTING STARTED

Bootstrapping the profiler

```
$profiler = new \Link0\Profiler\Profiler();
$profiler->start();
print_r($profiler->stop());
```

Adding a PersistenceHandler

```
$persistenceHandler = new \Link0\Profiler\PersistenceHandler\MemoryHandler();
$profiler = new \Link0\Profiler\Profiler($persistenceHandler);
```

Flysystem example

```
$filesystemAdapter = new \League\Flysystem\Adapter\Local('/tmp/profiler');
$filesystem = new \Link0\Profiler\Filesystem($filesystemAdapter);
$persistenceHandler = new \Link0\Profiler\PersistenceHandler\FilesystemHandler
$profiler = new \Link0\Profiler\Profiler($persistenceHandler);
```

DEMO TIME!

OH NOES! IN A TALK?

FUTURE?

EXCITING SOUNDS

SOME IDEAS

- Enable on production with sampling (1 in 1000 reqs)
- Aggregate all profiles to centralized machine/cluster
- Integrate into continuous deployment
  - Run profiling on acceptance environment
  - Alert when compared differences surpass threshold
- Codeception integration
  - Find business use-cases that are slow
  - Make a case for refactoring to the business
  - Focus on the customers emulated experience

QUESTIONS? I <3 FEEDBACK

- Joind.in: https://joind.in/talk/view/13685
- GitHub: http://github.com/dennisdegreef
- Twitter: @dennisdegreef
- IRC: link0 on Freenode

SLIDES ARE ALSO ON JOIND.IN

USEFUL LINKS

- Profiling PHP with PhpStorm and Xdebug
- Profiling PHP with PhpStorm and Zend Debugger
- XDebug Profiler documentation
- XHProf PHP documentation
- Profiling with XHProf and XHGui
- http://github.com/link0/profiler