Lynt SERVICES

WordPress + výkon

Vláďa Smitka

vladimir.smitka@lynt.cz

@smitka

Lynt services s.r.o.

Skrytá reklama

- Před pár dny jsme spustili nový blog, kde probíráme možnosti automatizace PPC

http://ppc-scripts.eu

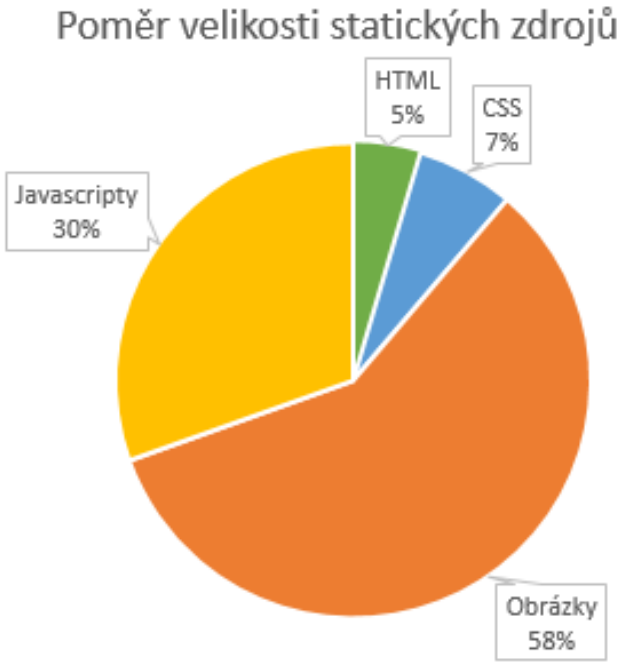
Jak zrychlit WP?

Nechte to na někom jiném!

Děkuji za pozornost!

Přednáška č. 2 – jak na to jít sami

- Funkce zmíněných služeb:
  - Cache zpracované stránky
  - Optimalizace zdrojů – kombinace JS a CSS, optimalizace obrázků



http://lynt.cz/blog/wordpress-v-cz-velky-pruzkum

Optimalizace obrázků

- Správné rozměry
- Zkontrolujte, zda jsou miniatury opravdu miniaturní

- Správné formáty
- „webová grafika“ – 8 bit PNG, SVG
- Fotografie – JPG (kvalita 75 je často OK, v Photoshopu 60)
- Video – MP4, FLV, ~~GIF!~~

Optimalizace obrázků – 24 bit PNG

- Nejčastěji se používá kvůli alfa kanálu
- Často je použit zbytečně – alfa kanál podporují i 8 bit PNG, které jsou mnohem menší, jen Photoshop je donedávna neuměl vytvořit

- https://tinypng.com/

- TruePNG - http://css-ig.net/articles/truepng
- AdvDef -http://advancemame.sourceforge.net/comp-download.html

Optimalizace obrázků - pluginy

- https://wordpress.org/plugins/ewww-image-optimizer/ + https://ewww.io
- https://wordpress.org/plugins/shortpixel-image-optimiser/ - 100 obrázků/měsíc zdarma
- https://wordpress.org/plugins/kraken-image-optimizer/ - je potřeba placený plán

CSS sprites a data URI

- Jaký je problém?
- Hlavičky HTTP komunikace mají velikost průměrně 0,5 – 1KB (u malých obrázků zaberou hlavičky více než užitečná data)
- Samotné připojení k serveru také nějaký čas trvá
- Cílem je tedy snížit počet požadavků:

A) u většího počtu obrázků lze použít jejich kombinaci do jednoho (http://draeton.github.io/stitches/)

B) jednotlivé malé obrázky lze načít pomocí data URI:

```
<img src="…data v base64…">
```

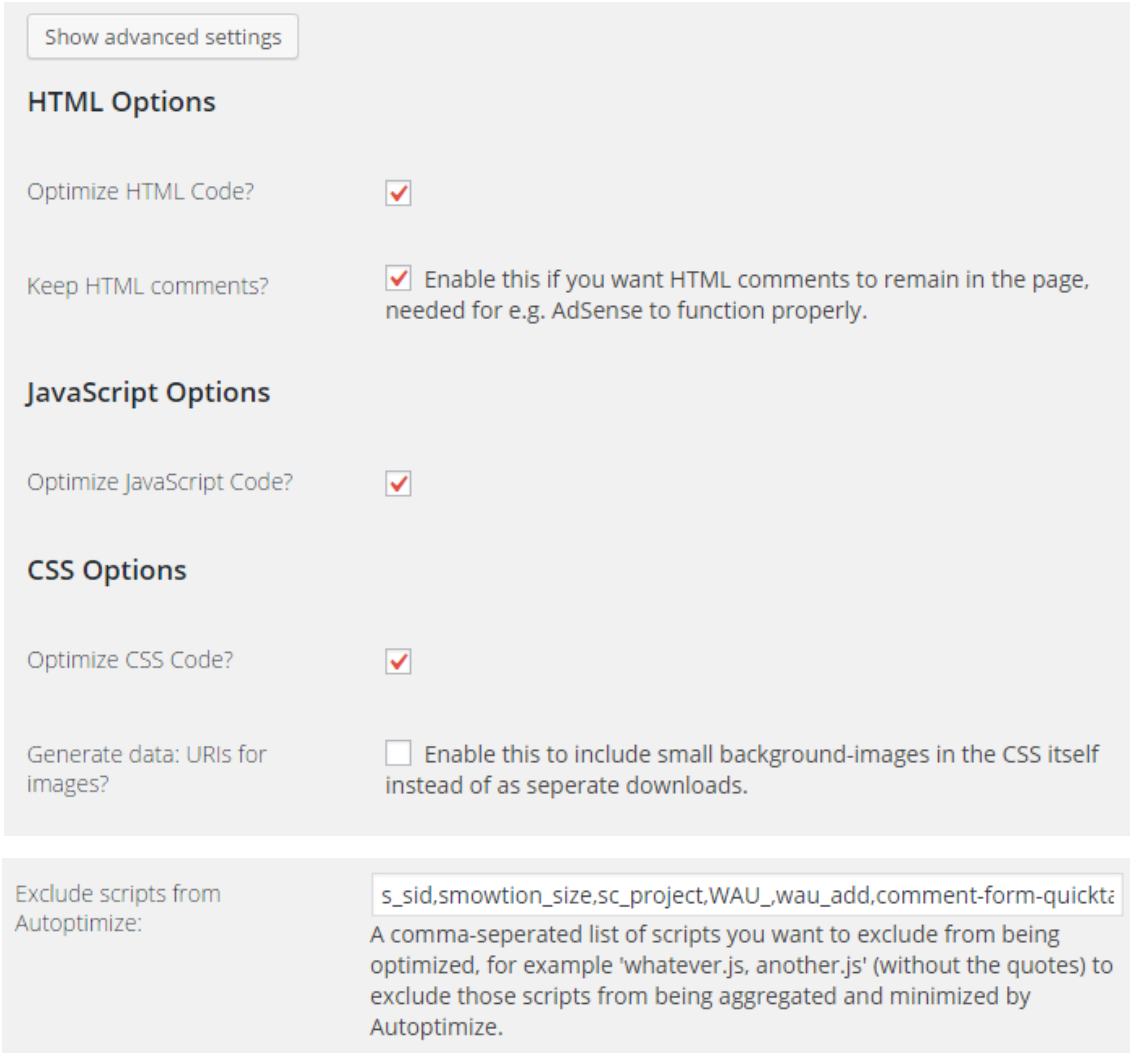
Optimalizace JS a CSS

- Cíl – snížení počtu dotazů na server, snížení velikosti
- Spojení více CSS a JS souborů do jednoho + jejich komprimace

- https://wordpress.org/plugins/autoptimize/
- https://wordpress.org/plugins/bwp-minify/

Lynt SERVICES

Autoptimize

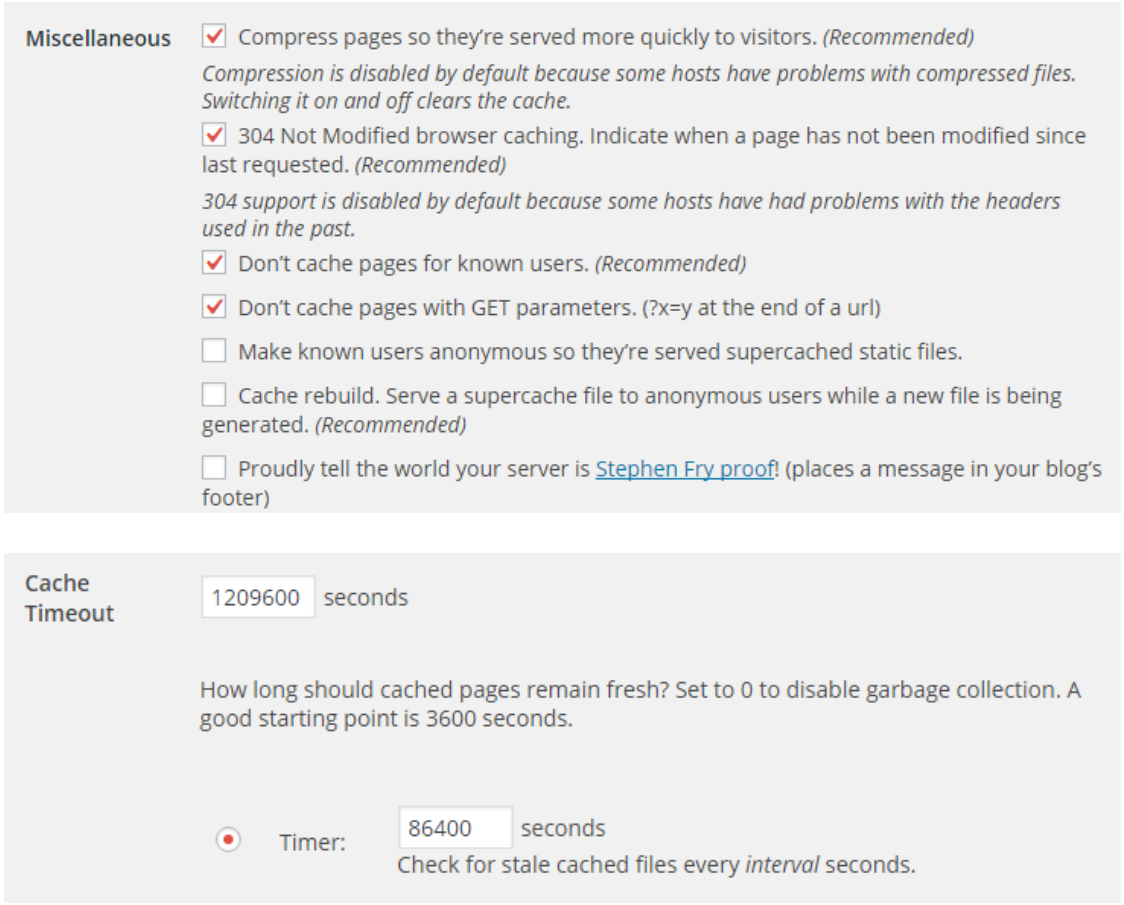


Stránková cache

- Uloží zpracovanou stránku a později ji vrací jako statický soubor
- https://wordpress.org/plugins/wp-super-cache/

- https://wordpress.org/plugins/w3-total-cache/

WP Super Cache



Lynt SERVICES

Děkuji za pozornost!

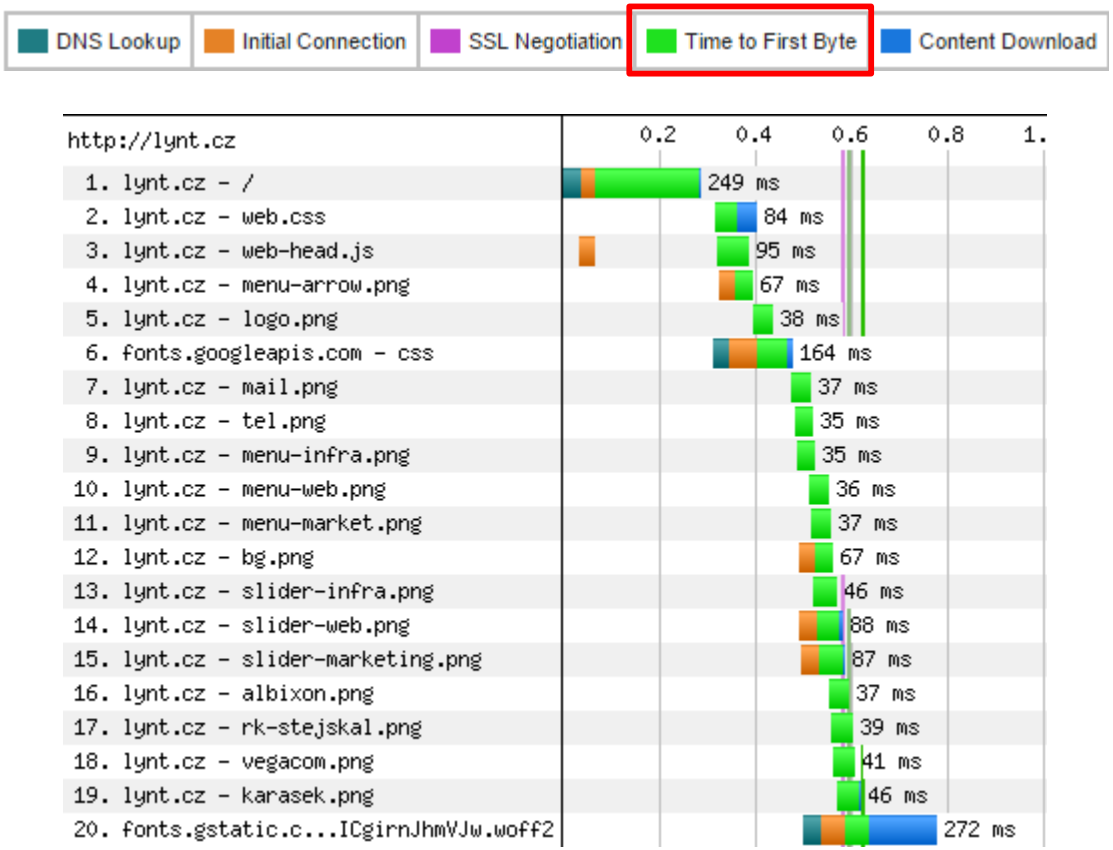
Přednáška č. 3 – ta opravdová

Hledání problému:
https://gtmetrix.com/


APLIKACE ZDROJE

Načítání zdrojů podrobněji

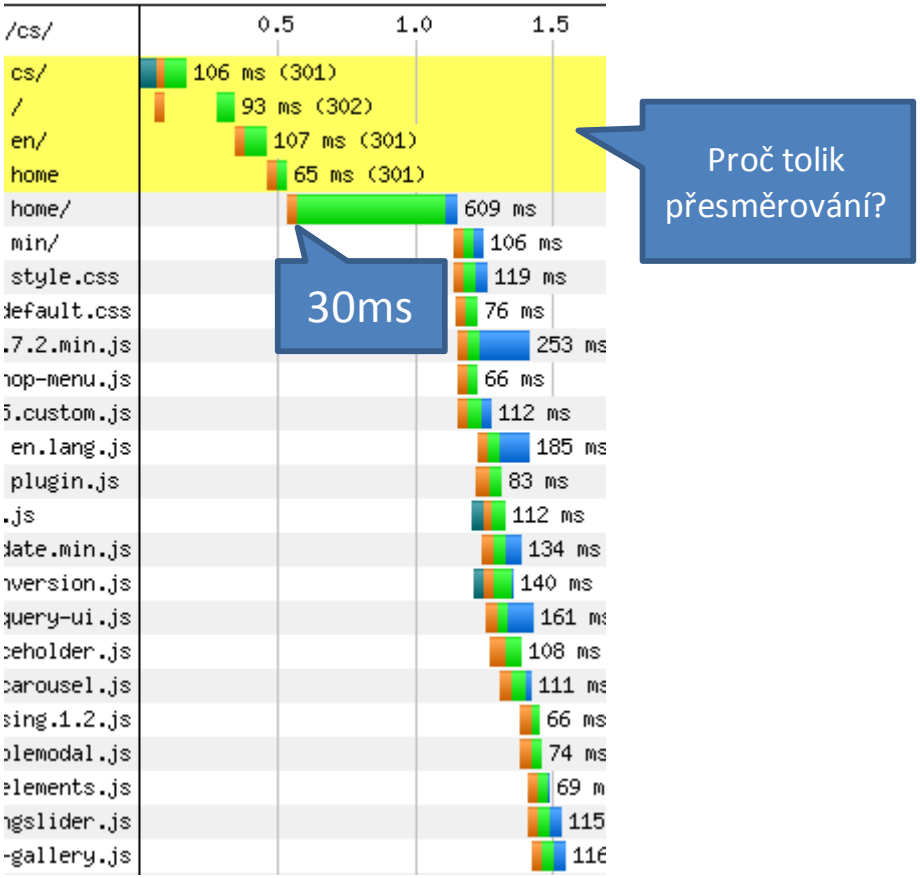
- http://www.webpagetest.org/



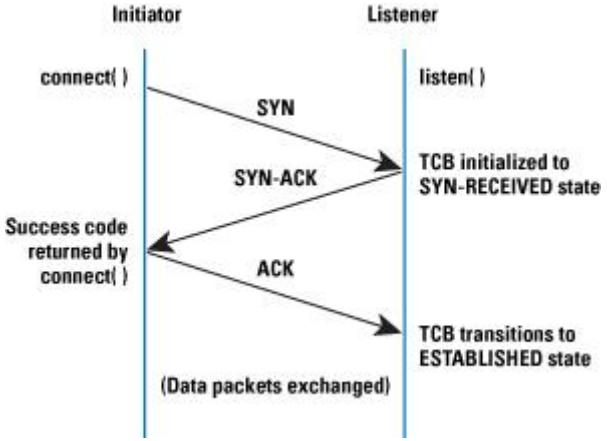
Načítání zdrojů – hlavní problémy

- **Expires hlavičky** (mod_expires) – zaručí, že nebude nutné zdroje stahovat znovu (jinak se využije heuristika prohlížeče)
- **GZIP komprese** (mod_deflate) – běžně 30-70% úspora u textových souborů
- **Keep Alive** – udržuje spojení, není je třeba znovu navazovat, stojí trochu RAM

Keep Alive

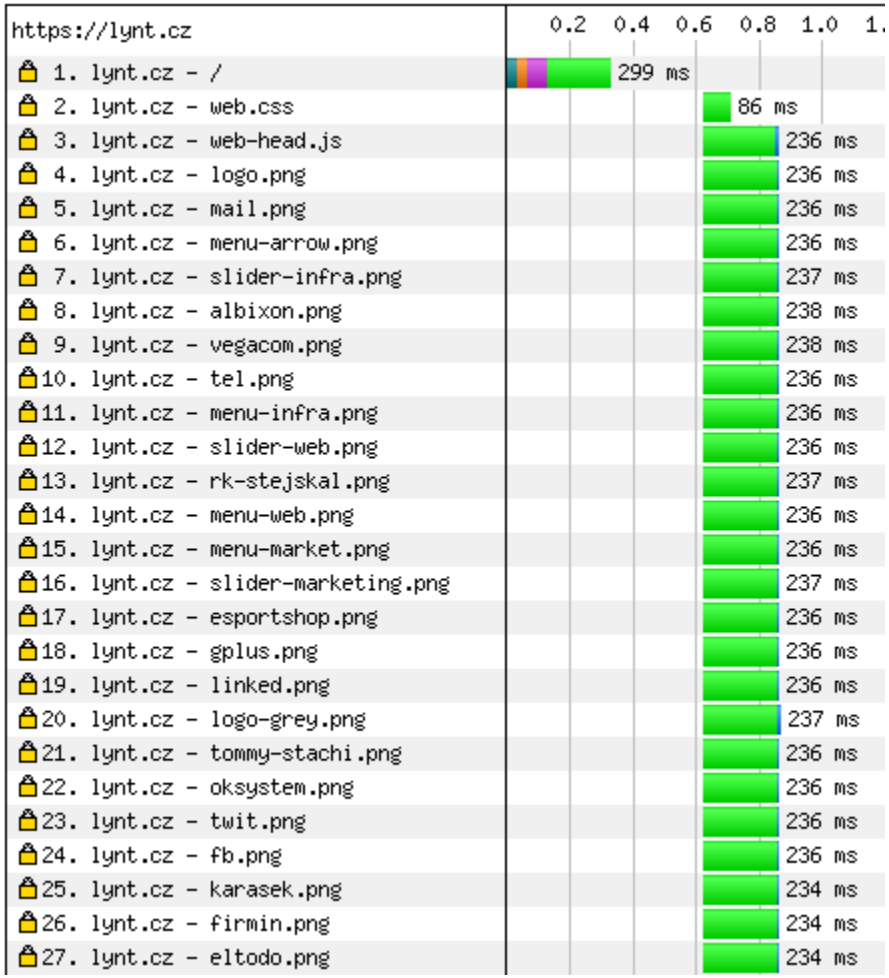


Vypnuté udržování spojení může být u webů s velkým počtem zdrojů kritické na mobilních zařízeních – velká latence



Ping 300ms, 150 zdrojů, paralelismus 5:
2x300x150/5 = **18s čekání na spojení** + ukončování

HTTPS & SPDY & HTTP/2



HTTP/2

- šifrování,
- komprese,
- keep alive,
- binární protokol

Není nutné řešit slučování CSS, JS, ani sprites. Jejich použití zde může i drobně zpomalovat – jsou najednou stahována i data, která nejsou aktuálně potřeba.

TTFB – co zpomaluje

- MySQL dotazy
- Nepoužívání cache
- HTTP požadavky (např. testy aktualizací)

- Čím více pluginů, tím více dotazů

Vliv pluginů

- Test na VPS od WEDOS (1 jádro, 4G RAM)
  `ab -n 1000 -c 4 http://domena`

- Čistý WP:
  Requests per second: 15.93 [#/sec]
  Time per request: 251.095 [ms]

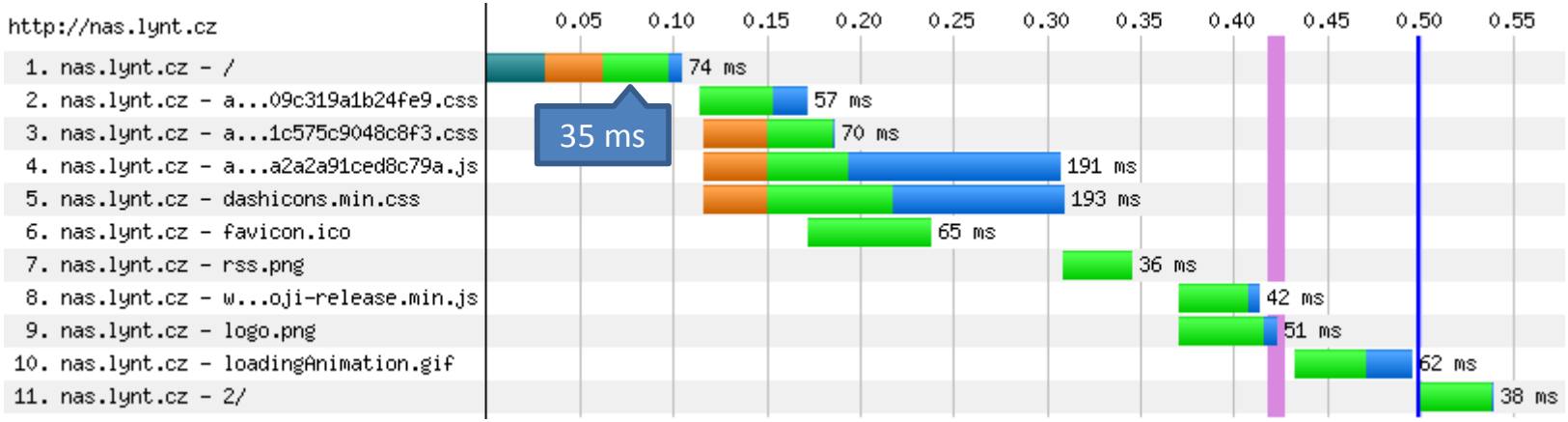
- WP + SliderRevolution + CF7 + Yoast SEO
  Requests per second: 4.61 [#/sec]
  Time per request: 868.450 [ms]

- WP + SliderRevolution + CF7 + Yoast SEO + WPML + Jetpack
  Requests per second: 2.94 [#/sec]
  Time per request: 1360.454 [ms]

Výborně, během několika minut se nám podařilo více než 5x snížit výkon!

Motivace: náš server

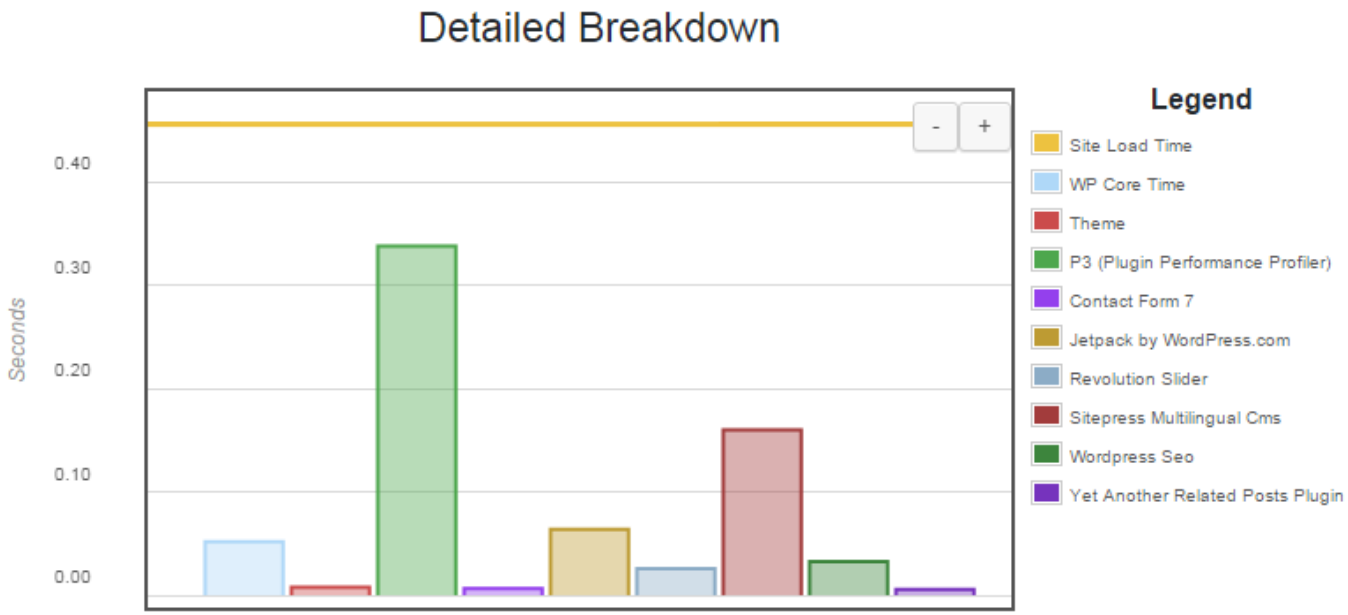
- WP + několik běžných pluginů
  Requests per second: 319.26 [#/sec]
  Time per request: 12.529 [ms]
  Ping 2,5 ms



Samozřejmě jsou použité podlé triky s cache, více o tom později

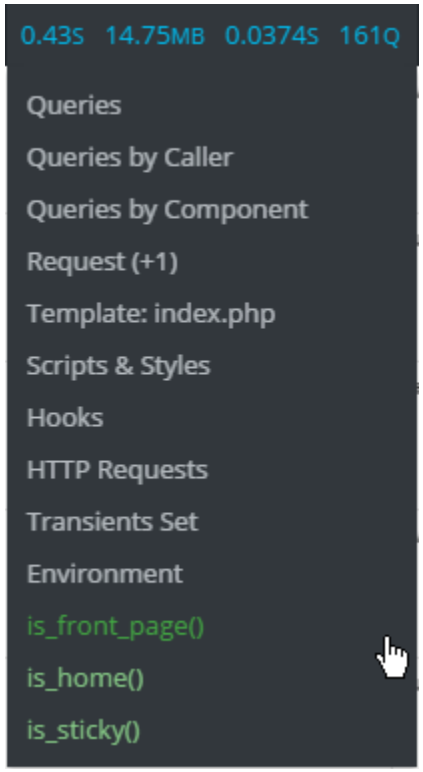
P3 Profiler

- https://wordpress.org/plugins/p3-profiler/
- Nejjednodušší, nejméně informací, nejméně přesné
- Může říci, jaký plugin dělá největší problémy



Query monitor

- https://wordpress.org/plugins/query-monitor/
- Podrobný přehled, co WP dělá
- Skvělý i při vývoji
  (ukazuje např. aktivní podmínky)



Lynt SERVICES

Query Monitor – DB dotazy

Seznam všech dotazů

$wpdb Queries

	Query	Caller	Component	Rows	Time
	All	All	All		
1	SELECT option_name, option_value FROM wp_options WHERE autoload = 'yes'	wp_load_alloptions() +	Core	173	0.0097
	select * from wp_revslider_css	RevSliderDB->fetch() +	Plugin: revslider	109	0.0015
4	SHOW TABLES LIKE 'wp_icl_translation_status'	wpml_site_uses_icl()	Plugin: sitepress-multilingual-cms	1	0.0010
61	SELECT element_id, language_code FROM wp_icl_translations WHERE trid = (SELECT trid FROM wp_icl_translations WHERE element_type = 'post_page' AND element_id = (SELECT option_value	SitePress->pre_option_page() +	Plugin: wordpress-seo	0	0.0008

Dotazy dle funkce

Queries by Caller

Caller	SELECT	SHOW	Time
get_option()	42		0.0129
wp_load_alloptions()	1		0.0097
RevSliderDB->fetch()	39		0.0090
WP_Post::get_instance()	25		0.0032
RevSliderFunctionsWP::isDBTableExists()		6	0.0023

Dotazy dle komponenty

Queries by Component

Component	SELECT	SHOW	Time
Core	35		0.0185
Plugin: revslider	88	6	0.0173
Plugin: jetpack	16		0.0047
Plugin: sitepress-multilingual-cms	8	1	0.0024
Plugin: wordpress-seo	1		0.0008

Query Monitor – další informace

	HTTP Request	Response	Transport	Call Stack	Component	Timeout	Time
		All			All		
1	POST https://jetpack.wordpress.com/xmlrpc.php ?for=jetpack &wpcom_blog_id=99703530	200 OK	curl	WP_Http->request() wp_remote_request() Jetpack_Client::_wp_remote_request() Jetpack_Client::remote_request() Jetpack_IXR_Client->query() Jetpack->get_cloud_site_options() Jetpack::check_identity_crisis() Jetpack->alert_identity_crisis() do_action('admin_notices')	Plugin: jetpack	10	0.3496
							0.3496

HTTP požadavky

HTTP požadavky založené na wp_remote_X

curl, file_get_contents atd. neodhalí [ty lze odchytnout například analýzou provozu (tcpdump, wireshark)]

WordPress	
version	4.3.1
WP_DEBUG	false
WP_DEBUG_DISPLAY	true
WP_DEBUG_LOG	false
SCRIPT_DEBUG	false
WP_CACHE	false
CONCATENATE_SCRIPTS	undefined
COMPRESS_SCRIPTS	undefined
COMPRESS_CSS	undefined
WP_LOCAL_DEV	undefined

Zapisuje do souboru: IO operace

Bude používat neminifikované knihovny: více dat

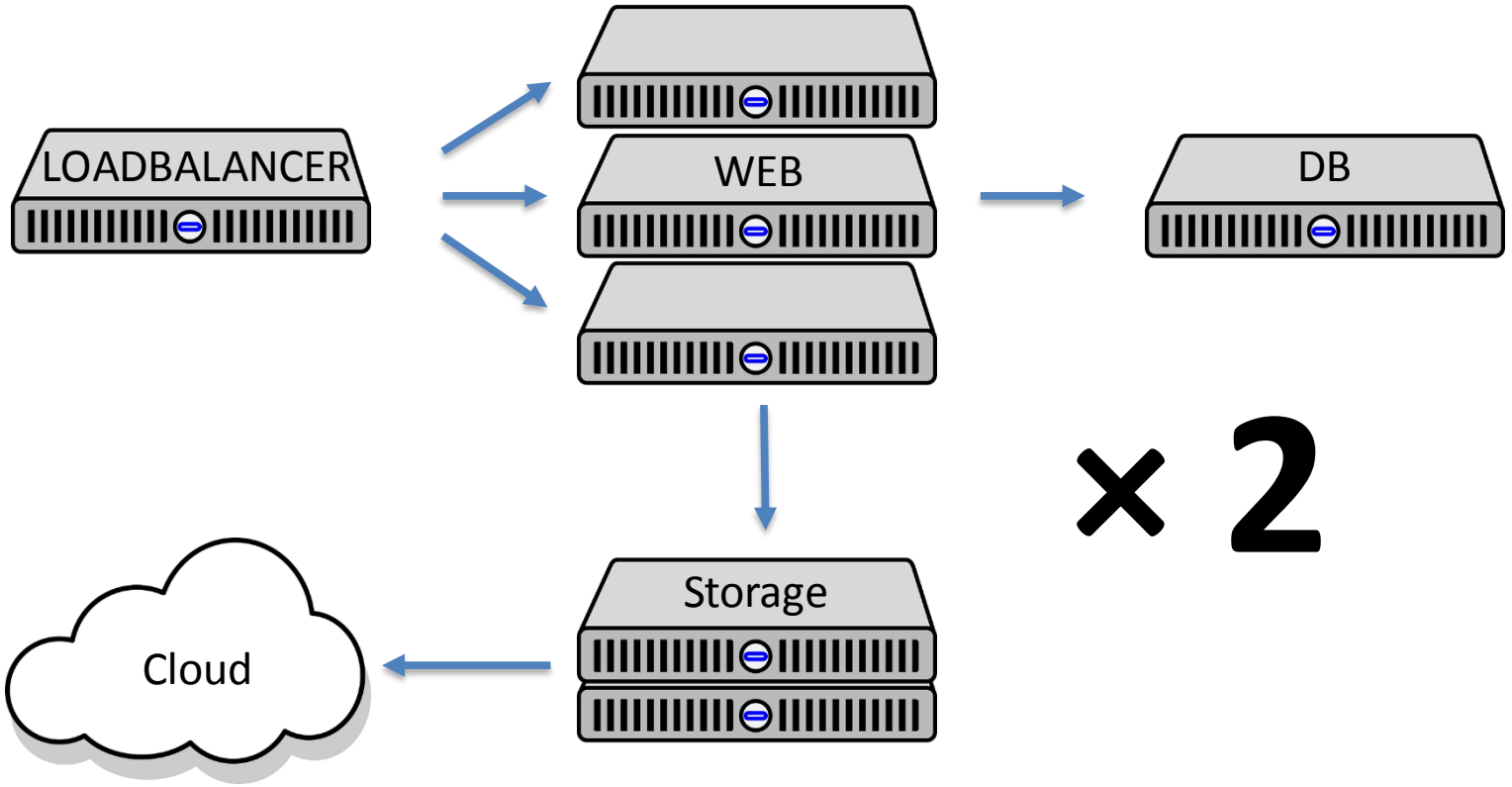
Chybí ale třeba SAVEQUERIES – další IO

Pohled z druhé strany

- Zatím jsme zkoumali web z pohledu aplikace
- Přišel čas podívat se na to, kde aplikace běží

Infrastruktura

3 komponenty – HTTP server, DB server a Zdroje dat
Každá má úplně jiné nároky…



Monitoring – hledáme úzké hrdlo

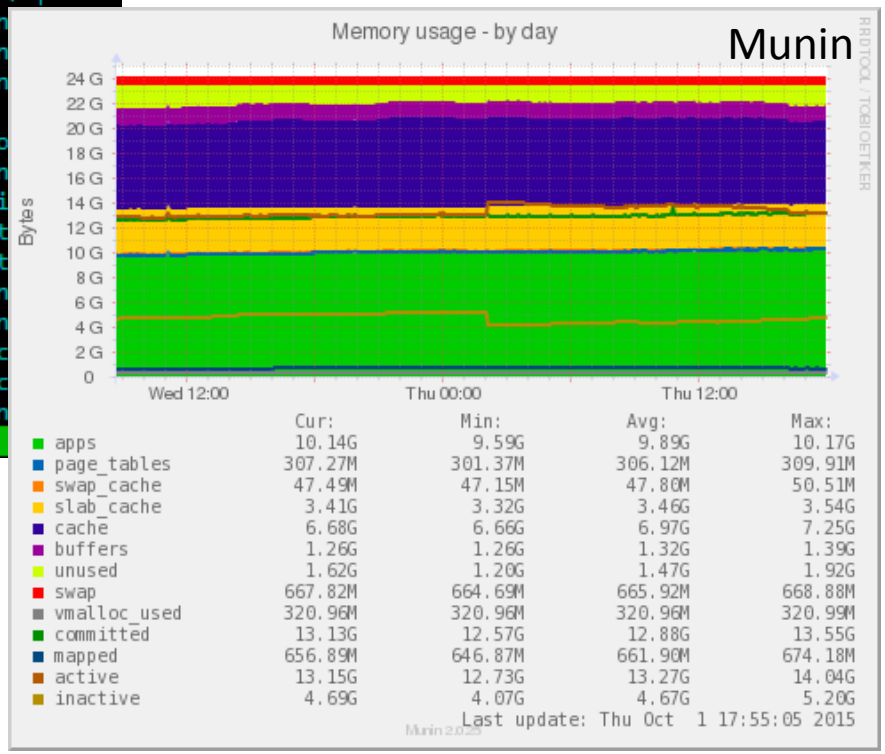
htop

```
  CPU[||||||||||||||||||||||||100.0%]     Tasks: 146, 66 thr; 5 running
  Mem[||||||||||||||||||1776/3965MB]      Load average: 2.71 0.83 0.32
  Swp[                          0/0MB]    Uptime: 14 days, 01:40:23

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
27984 www-data   20   0  436M 68116 42448 R 24.4  1.7  0:06.48 /usr/sbin/apache2
27977            20   0  436M 68120 42448 R 23.9  1.7  0:06.48 /usr/sbin/apache2
27978            20   0  436M 67852 42448 R 23.4  1.7  0:06.40 /usr/sbin/apache2
28011            20   0  436M 67052 41664 R 23.4  1.7  0:01.98 /usr/sbin
27988            20   0  431M 64640 44656 R  5.0  1.6  0:04.58 /usr/sbin
31334            20   0  600M 95564 14920 S  1.0  2.4 22:01.68 /usr/sbin
28024 root       20   0 26104  5304  2904 R  0.5  0.1  0:00.05 htop
26902 root       20   0 82668  5728  4880 S  0.5  0.1  0:00.11 sshd: roo
28389            20   0  600M 95564 14920 S  0.5  2.4  0:01.32 /usr/sbin
    1 root       20   0 28656  4776  2964 S  0.0  0.1  1:26.29 /sbin/ini
  129 root       20   0 32960  4976  4664 S  0.0  0.1  2:27.39 /lib/syst
  139 root       20   0 40892  3068  2504 S  0.0  0.1  0:00.05 /lib/syst
  262 root       20   0 10032   116     0 S  0.0  0.0  0:00.00 /sbin/rdn
  264            20   0 18444  1536  1372 S  0.0  0.0  0:00.00 /sbin/rdn
  553 root       20   0 37068  2664  2228 S  0.0  0.1  0:17.67 /sbin/rpc
  562            20   0 37268  2768  2184 S  0.0  0.1  0:00.00 /sbin/rpc
  576 root       20   0 23348   208     4 S  0.0  0.0  0:00.00 /usr/sbin
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill
```

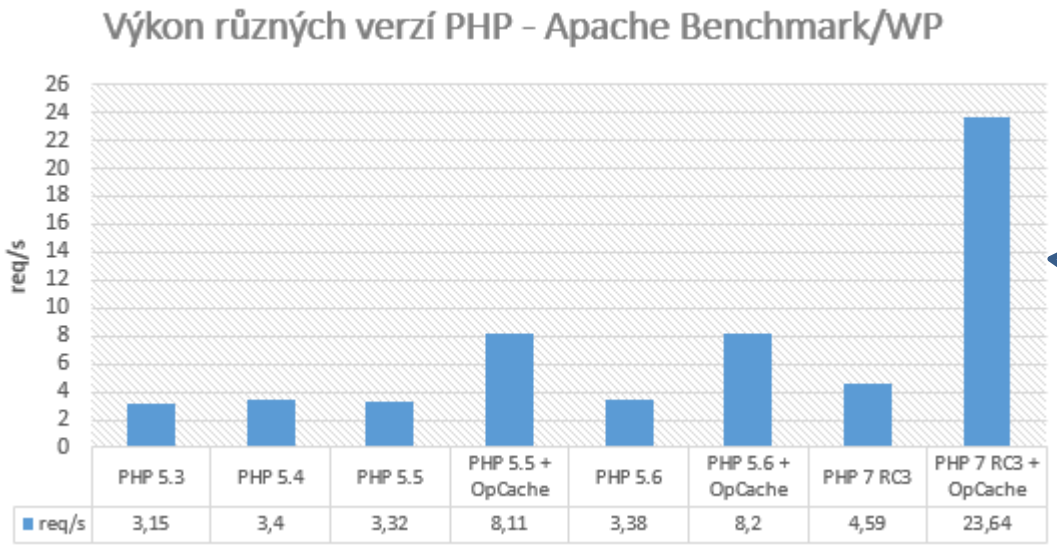
- htop
- iotop
- iftop
- nmon

Munin



http://munin-monitoring.org/
http://www.zabbix.com/
http://www.librenms.org/

Verze PHP



Novější verze PHP zrychlují a hlavně umí lépe využít OpCache (což také stojí více RAM)

- OpCode cache
- Object cache

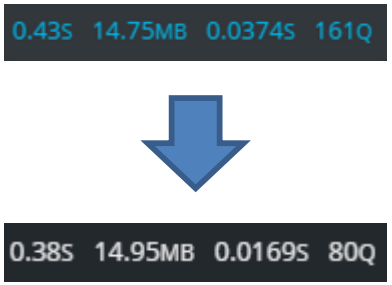
OpCode cache

- Při každém volání je skript přeložen z jazyka PHP do Bytecode, aby mohl být vykonán
- OpCache výsledek uloží a není třeba opakované kompilace
- Nutno doinstalovat (PHP < 5.5)

- APC
- Xcache
- Zend OpCache (přibaleno od PHP 5.5)

Object Cache

- Často součást modulu s OpCache, někdy samostatná
- Uživatel si může sám ukládat data pro pozdější použití
- Je potřeba povolit a alokovat velikost
- Do WP je třeba nahrát drop-in object backend
  - APC
  - Xcache
  - APCu (pro Zend OpCache)
  - Redis
  - Memcached



71 z nich patří Slider Revolution

- WP do ní bude ukládat mnoho dat, která jinak ukládá jako tranzientní proměnné do databáze – rapidně klesne počet DB dotazů

Zjištění z Query Monitoru

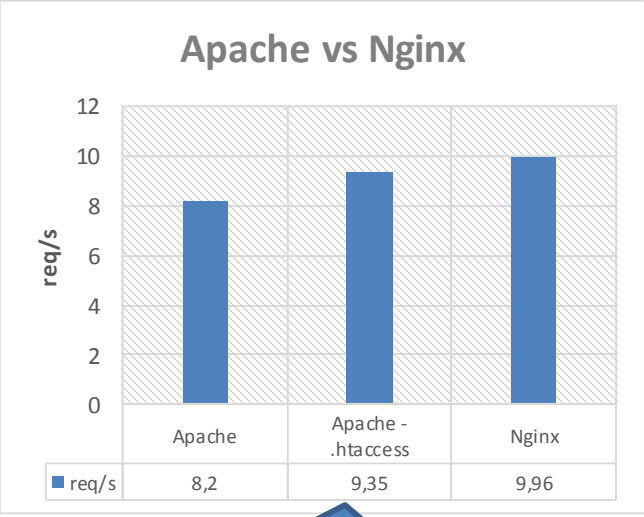
- Pravděpodobně zjistíte, že se necachuje menu a některé widgety

- http://afterburner.voceplatforms.com/back-end.html#voce-widget-cache
- http://afterburner.voceplatforms.com/back-end.html#voce-cached-nav

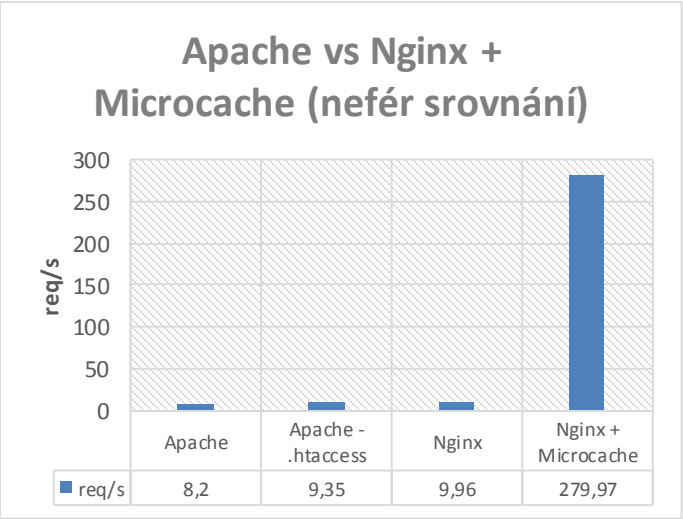
- Doporučuji projít i další nástroje a tipy z tohoto webu

HTTP server

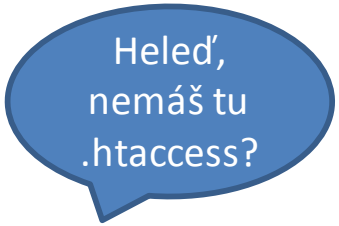
- Je několik možností, jak PHP provozovat
- Apache (prefork) + mod_php
- Apache (mpm event/worker) + PHP-FPM, FastCGI
- Nginx + PHP-FPM



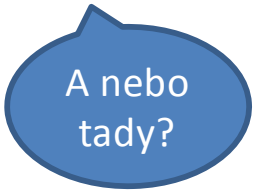
AllowOverride None



http://lynt.cz



/wp-content/uploads/revslider/classicslider/bike.jpg



Apache se příliš nehodí pro statické soubory

Zpět k našemu serveru

- Dříve zmíněný skvělý výsledek byl právě díky microcache – Nginx si zapamatuje výsledek dotazu na několik vteřin/minut a ten vrátí
- Tato technika se hodí v případech, že očekáváme zvýšenou špičkovou návštěvnost – např. po publikaci nového obsahu

- Microcache: Requests per second: 319.26 [#/sec]
- WP Supercache: Requests per second: 270.84 [#/sec]
- Bez cachování *: Requests per second: 13.52 [#/sec]

* je třeba vzít v úvahu, že byl test prováděn v plném provozu se stovkami jiných požadavků

Poslední test

- Test na WEDOS VPS s velkou konkurenčností

```
ab -n 1000 -c 40  http://domena
```

- Requests per second: 1191.49 [#/sec] (mean)
- Time per request: 33.572 [ms] (mean)
- Time per request: 0.839 [ms] (mean, across all concurrent requests)

Poznámka na konec:

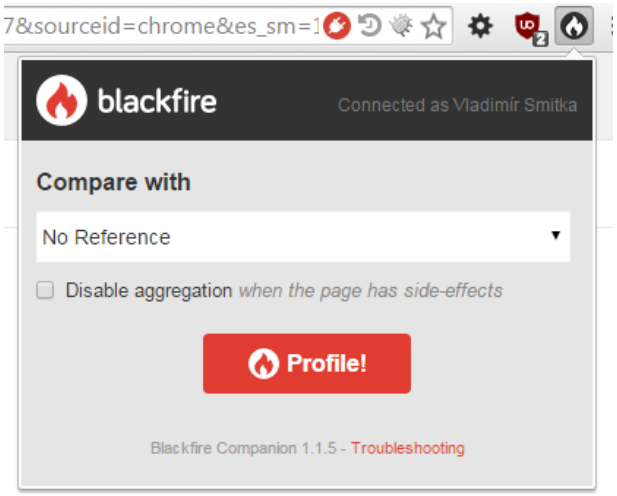
„Použití cache s sebou vždy nese problém její invalidace!“

Opravdové nástroje

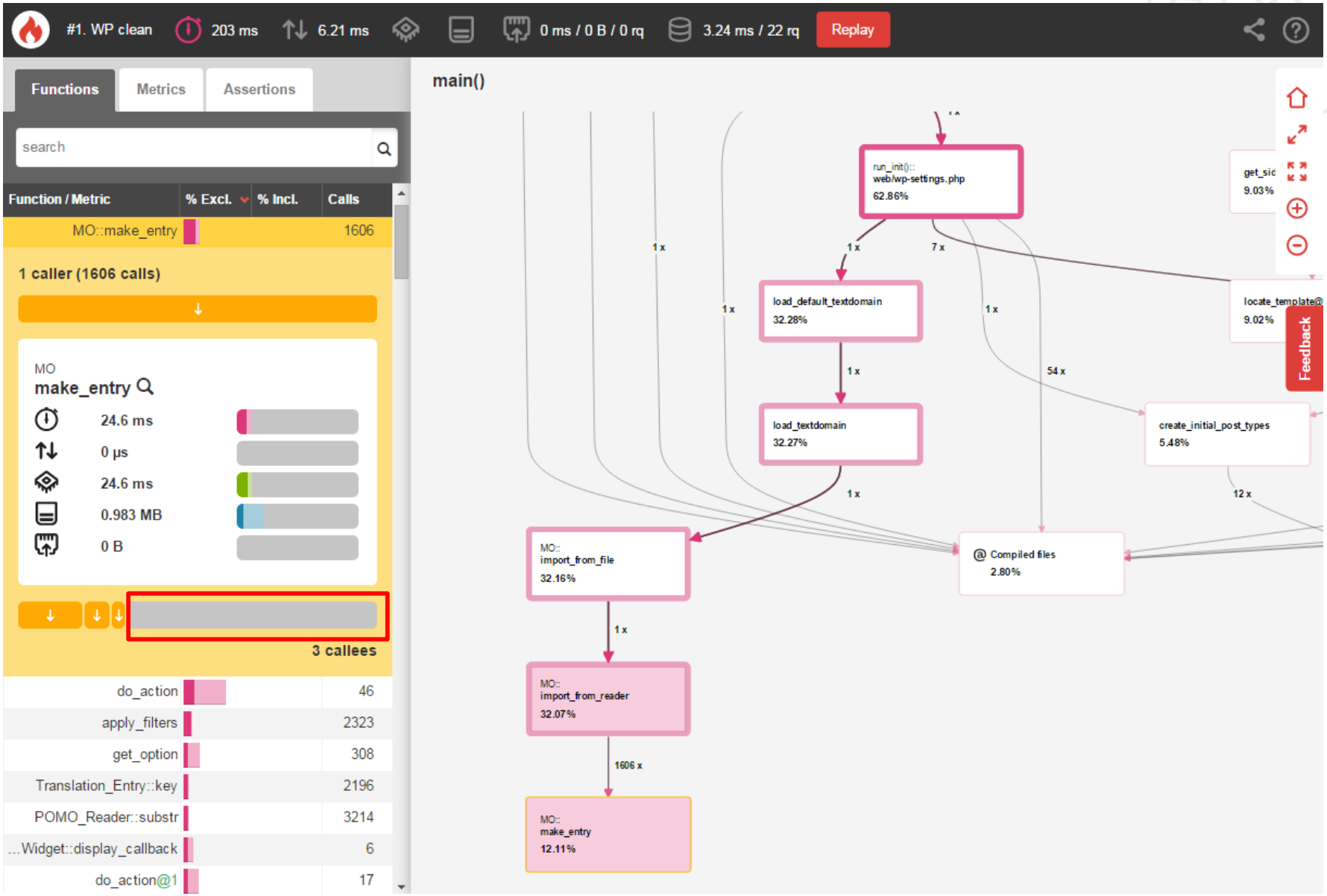
Blackfire.io

blackfire

- Profilace výkonu a skvělá vizualizace
- Je třeba na server nainstalovat agenta a rozšíření do PHP - návod
- Profilaci lze spouštět z Chrome pomocí Blackfire Companion



- Hack verze zdarma
- Premium verze 82,5€/měsíc
  (navíc analyzuje DB dotazy,
  HTTP požadavky, lze pracovat v týmu,
  delší dobu si pamatuje data)

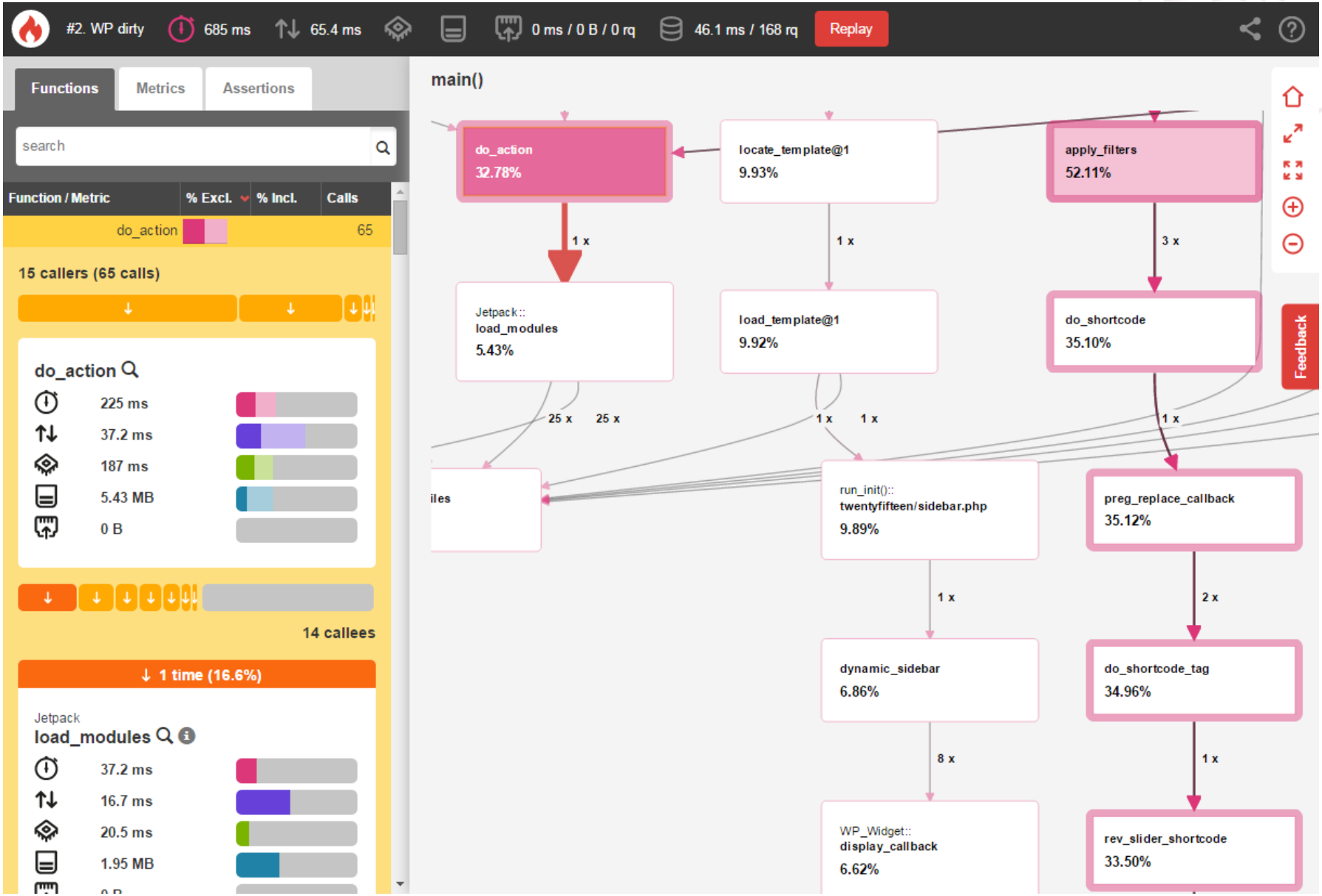

#1. WP clean
203 ms
6.21 ms
0 ms / 0 B / 0 rq
3.24 ms / 22 rq
Replay
Functions
Metrics
Assertions
search
Function / Metric
% Excl.
% Incl.
Calls
MO::make_entry
1606
1 caller (1606 calls)
MO
make_entry
24.6 ms
0 µs
24.6 ms
0.983 MB
0 B
3 callees
do_action
46
apply_filters
2323
get_option
308
Translation_Entry::key
2196
POMO_Reader::substr
3214
...Widget::display_callback
6
do_action@1
17
main()
run_init()::
web/wp-settings.php
62.86%
load_default_textdomain
32.28%
load_textdomain
32.27%
MO::
import_from_file
32.16%
MO::
import_from_reader
32.07%
MO::
make_entry
12.11%
@ Compiled files
2.80%
create_initial_post_types
5.48%
locate_template@
9.02%
get_sid
9.03%
1 x
7 x
54 x
12 x
1606 x
Feedback

Přes 30% na načítání lokalizace?

- WP využívá Gettext PHP implementaci, která není příliš rychlá

- Řešení:
- https://github.com/LyntServices/WP-lang-cache (staré, vyžaduje zásah do jádra, modifikace pro ObjectCache)
- https://wordpress.org/plugins/mo-cache/

Lynt SERVICES



Proč je do_action pomalé?

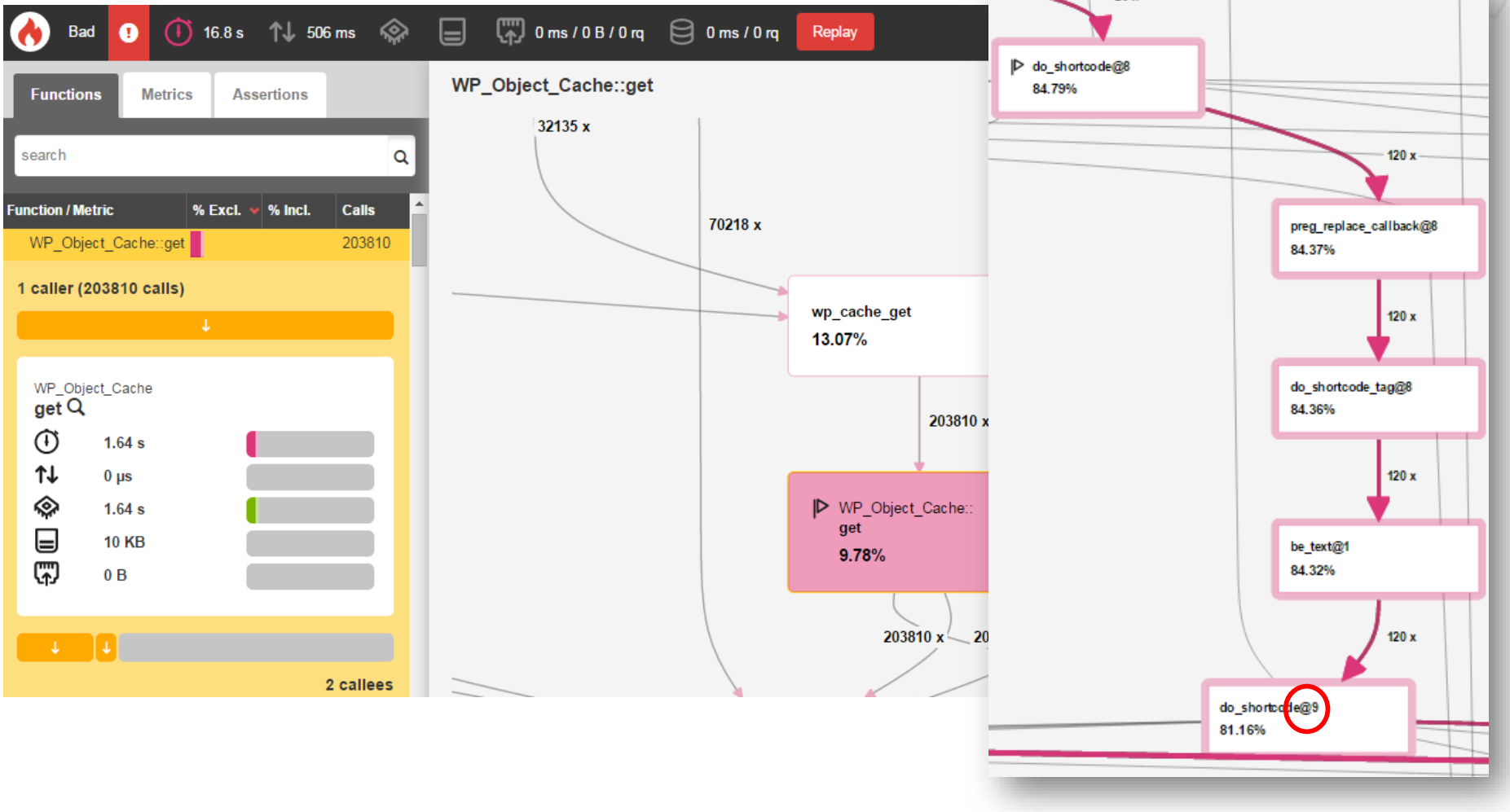
```
499
500 do {
501         foreach ( (array) current($wp_filter[$tag]) as $the_ )
502                 if ( !is_null($the_['function']) )
503                         call_user_func_array($the_['function'], array_slice($args, 0, (int) $the_['accepted_args']));
504
505 } while ( next($wp_filter[$tag]) !== false );
506
507 array_pop($wp_current_filter);
```

Od PHP5.6 lze použít optimalizovanou funkcionalitu **Argument Unpacking:** …$the_['acceped_args']

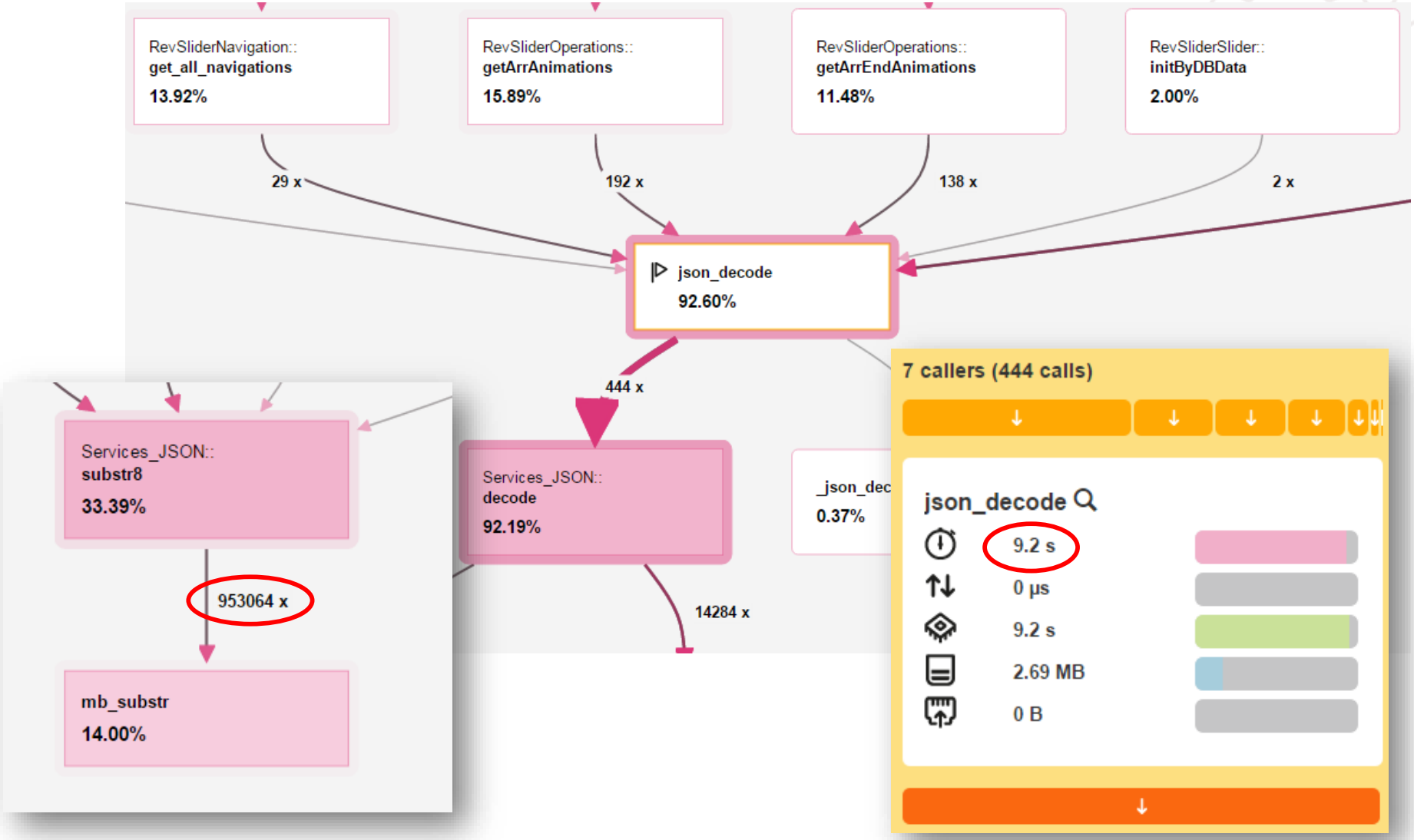
Samotná funkce call_user_func_array nepředstavuje extrémní problém, problém je, že se volá velmi často, v některých případech se jedná i o tisíce volání.

Lynt SERVICES

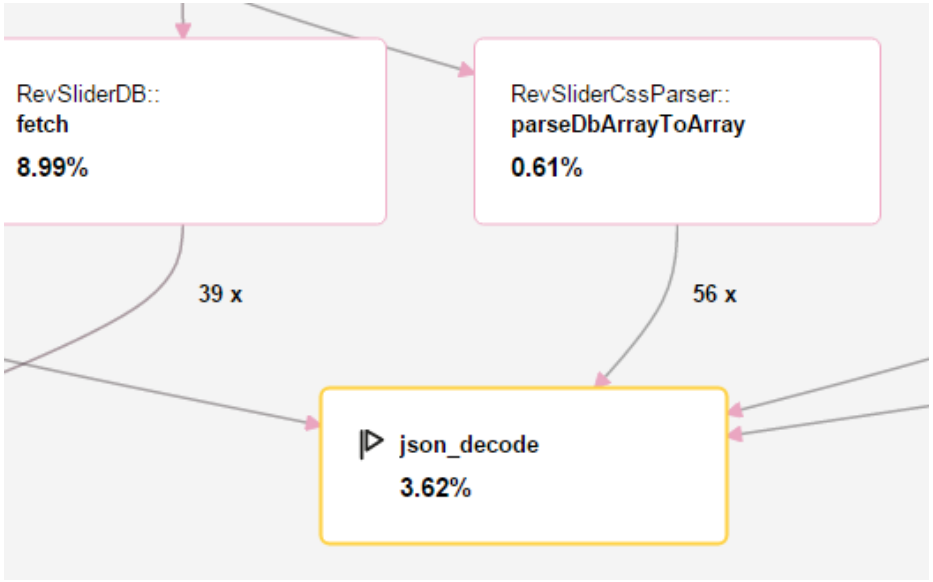
Ukázka velkého problému

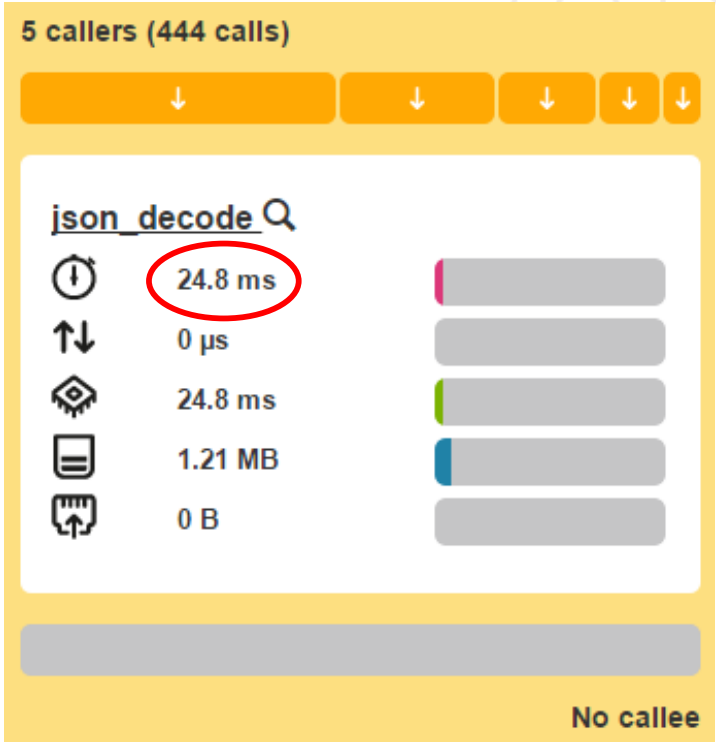


Další ukázka – i s opravou!



Další ukázka – i s opravou!





Oprava:
extension=json.so

chyběl nativní modul pro práci s JSON
9.2s => 0.0248s = 370x zvýšení výkonu ☺

Konec profilování

- Při profilování WP v Blackfire většinou skončíte na 2 případech:

1) **do_action** – obecná WP akce (podobně jako apply_filters), v Blackfire nelze přesně vysledovat příčiny a důsledky konkrétního volání
2) **šedý proužek** – výkon spotřebovaný samotnou funkcí, není vidět kód

- V tento moment přichází **debugování**

Xdebug

- Rozšíření do PHP: http://xdebug.org/
- Pomocník, který poradí, co stáhnout a jak nakonfigurovat podle výstupu z phpinfo(): http://xdebug.org/wizard.php
- Vysvětlení konfiguračních voleb: https://gist.github.com/IngmarBoddington/5311858

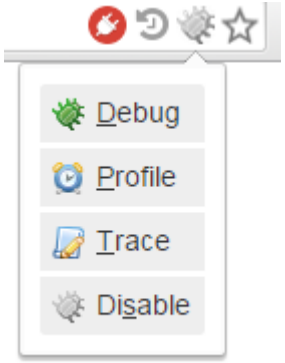
```
xdebug.remote_enable = 1
xdebug.profiler_enable_trigger = 1
xdebug.trace_format = 1
xdebug.trace_enable_trigger = 1
```

Xdebug – co to umí?

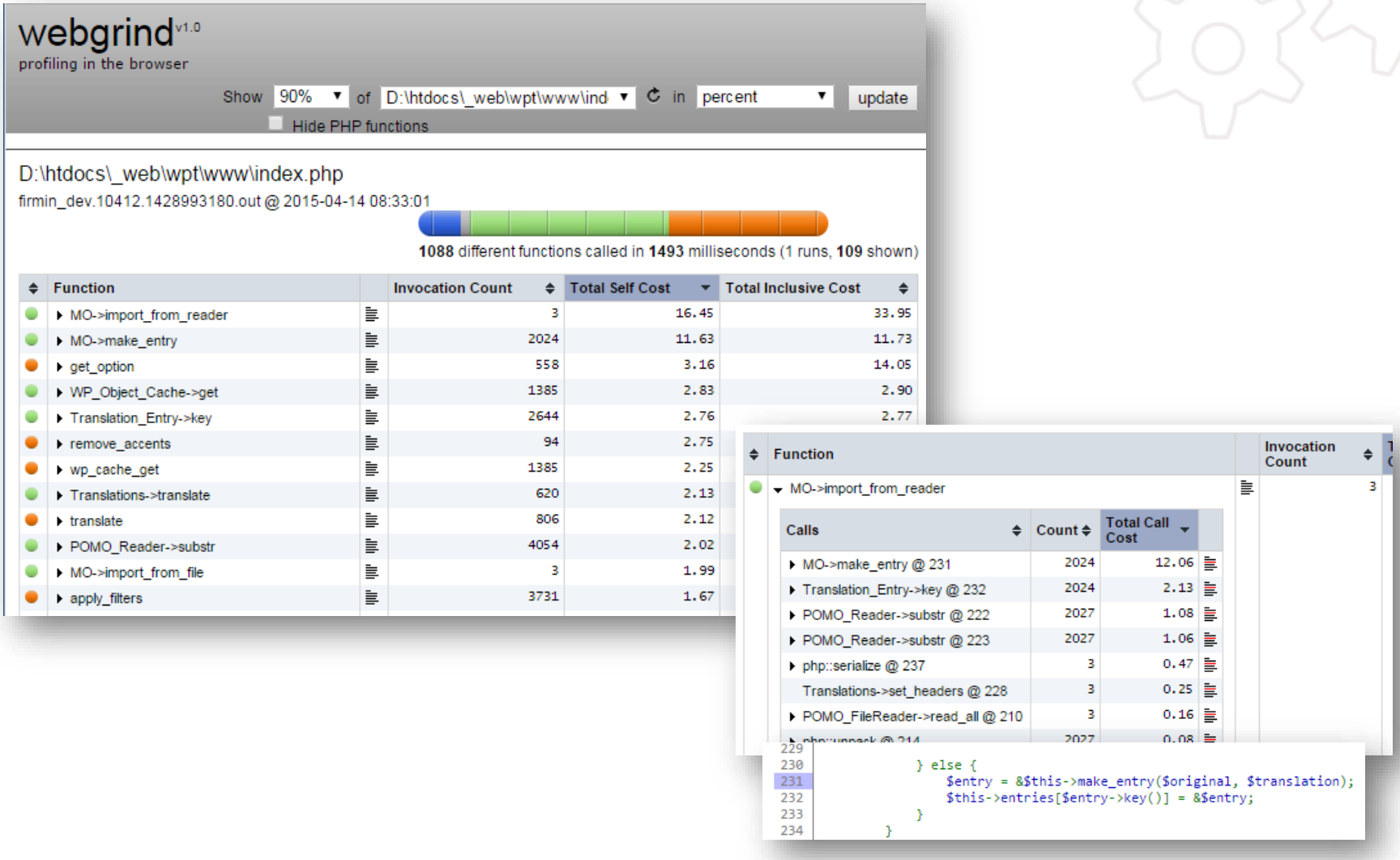
- Umožňuje **krokovat kód** z vývojového prostředí
- Generuje Trace-log
- Analyzuje pokrytí kódu
- Umí také profilovat (data jsou více zkreslená vlastní režií)
  xdebug.profiler_enable_trigger = 1
  ?XDEBUG_PROFILE

- Pro vizualizaci lze použít například WebGrind: https://github.com/jokkedk/webgrind

- Xdebug Helper do Chrome: https://chrome.google.com/webstore/detail/xdebug-helper/eadndfjplgieldjbigjakmdgkmoaaaoc



WebGrind


webgrind v1.0
profiling in the browser
Show 90% of D:\htdocs_web\wpt\www\ind in percent update
Hide PHP functions
D:\htdocs_web\wpt\www\index.php
firmin_dev.10412.1428993180.out @ 2015-04-14 08:33:01
1088 different functions called in 1493 milliseconds (1 runs, 109 shown)
Function | Invocation Count | Total Self Cost | Total Inclusive Cost
MO->import_from_reader | 3 | 16.45 | 33.95
MO->make_entry | 2024 | 11.63 | 11.73
get_option | 558 | 3.16 | 14.05
WP_Object_Cache->get | 1385 | 2.83 | 2.90
Translation_Entry->key | 2644 | 2.76 | 2.77
remove_accents | 94 | 2.75
wp_cache_get | 1385 | 2.25
Translations->translate | 620 | 2.13
translate | 806 | 2.12
POMO_Reader->substr | 4054 | 2.02
MO->import_from_file | 3 | 1.99
apply_filters | 3731 | 1.67
Function | Invocation Count
MO->import_from_reader | 3
Calls | Count | Total Call Cost
MO->make_entry @ 231 | 2024 | 12.06
Translation_Entry->key @ 232 | 2024 | 2.13
POMO_Reader->substr @ 222 | 2027 | 1.08
POMO_Reader->substr @ 223 | 2027 | 1.06
php::serialize @ 237 | 3 | 0.47
Translations->set_headers @ 228 | 3 | 0.25
POMO_FileReader->read_all @ 210 | 3 | 0.16
229
230 } else {
231 $entry = &$this->make_entry($original, $translation);
232 $this->entries[$entry->key()] = &$entry;
233 }
234 }

Trace

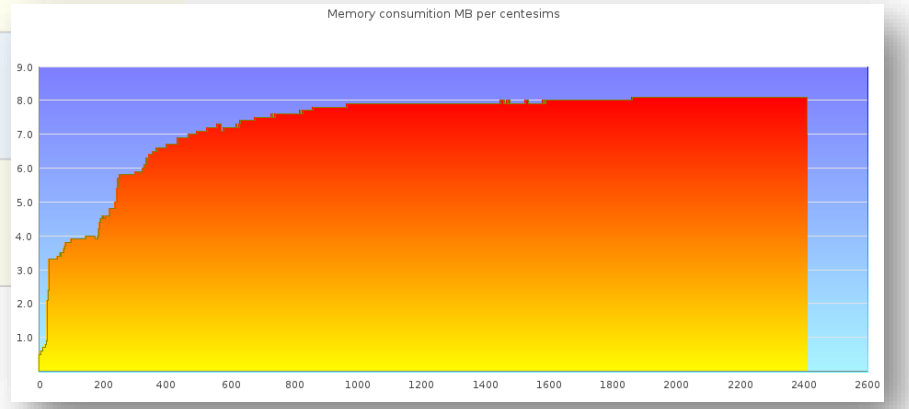
- Tracelog – ohromné množství dat
- Hodí se na řešení problémů s RAM
- https://github.com/corretge/xdebug-trace-gui

define D:\htdocs\wp4\index.php	14	138 µs	400	
dirname D:\htdocs\wp4\index.php	17	187 µs	-24	
require D:\htdocs\wp4\index.php	17	214 µs	656	
◦ dirname D:\htdocs\wp4\wp-blog-header.php	12	102 µs	216	

řádek

strávený čas

změna využití paměti



Stav využití paměti v průběhu vykonávání skriptu

Trace části kódu

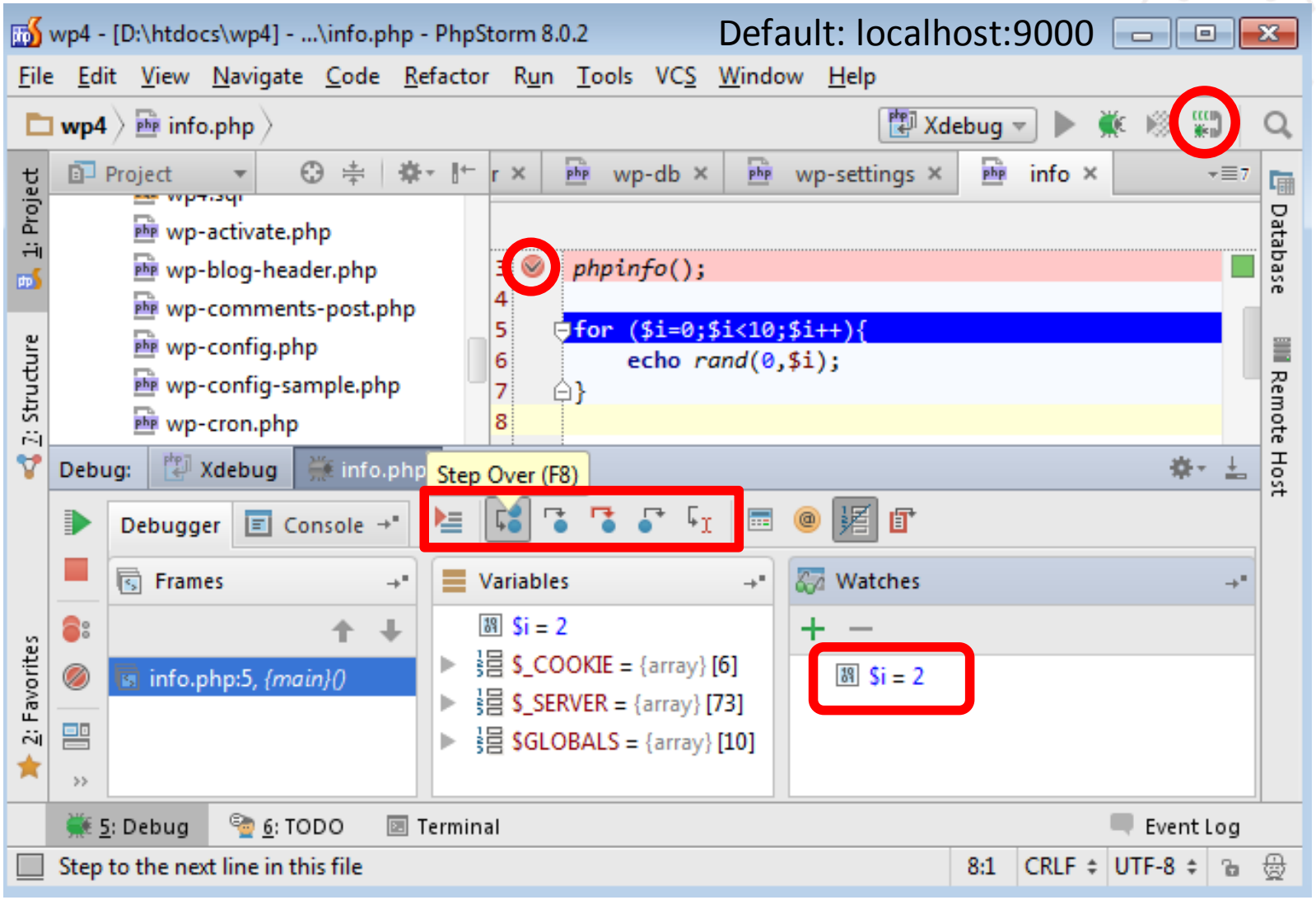
```php
xdebug_start_trace('for.xt');

for ($i=0;$i<3;$i++){
   echo rand(0,$i);
}

xdebug_stop_trace();
```

```
TRACE START [2015-10-01 11:49:49]
2  2  1  0.005554  223088
2  3  0  0.005621  223128  rand  0  D:\htdocs\wp4\info.php  7
2  3  1  0.005740  223128
2  4  0  0.005770  223128  rand  0  D:\htdocs\wp4\info.php  7
2  4  1  0.005879  223128
2  5  0  0.005908  223128  rand  0  D:\htdocs\wp4\info.php  7
2  5  1  0.006016  223128
2  6  0  0.006044  223096  xdebug_stop_trace  0  D:\htdocs\wp4\info.php  9
          0.006130  223120
TRACE END   [2015-10-01 11:49:49]
```

Krokování programu



Myšlenka otce Fura

„Končí zlatá éra webových vývojářů,
začíná éra systémových administrátorů.“

Souhrn - analýza výkonu

- Jak se web jeví navenek?
  - https://gtmetrix.com/
  - http://www.webpagetest.org/
  - nástroje v prohlížeči
- Co zdržuje WordPress?
  - P3 profiler – orientační detekce pomalých pluginů
  - Query monitor – konkrétní DB dotazy a další
- Profilace a ladění
  - Blackfire.io
  - Xdebug (vizualizace webgrind)

A to je vše, přátelé.

Malý dárek: Blackfire.io Premium na měsíc zdarma:

DODWEDOS102015

platí do 19.10.2015

kód se zadává při nákupu měsíční verze

Můžete mne samozřejmě sledovat na twitteru @smitka a navštívit náš blog http://lynt.cz/blog