

Welcome!

Profiling PHP Applications

International PHP Conference - Berlin, Germany - Jun
4th, 2012

Derick Rethans - derick@derickrethans.nl - twitter:
[@derickr](https://twitter.com/derickr)

Derick Rethans

- Dutchman living in London
- PHP MongoDB driver maintainer for 10gen (the company behind MongoDB)
- Author of Xdebug
- Author of the mcrypt, input_filter, dbus, translit and date/time extensions



The Internet is Full of Nonsense

A small selection of PHP optimisation tips I found on the internet

echo is faster than print

Use pre-incrementing where possible as it is 10% faster

```
++$i;  
//is faster than  
$i++;
```

Methods in derived classes run faster than ones defined in the base class.

A function call with one parameter and an empty function body takes about the same time as doing 7-8 `$localvar++` operations. A similar method call is of course about 15 `$localvar++` operations.

Not everything has to be OOP, often it is just overhead, each method and object call consumes a lot of memory.

Avoid overusing function calls. Calling a function in PHP is very expensive, so avoid it whenever you can.

The Internet is Full of Nonsense

I can only say one thing:



Do I need to optimise my code?

Before you can optimise anything:

- Find out if things are running slow
- Find out whether it is the code
- Understand your code, application and execution paths
- Find out which parts of the code are slow
- Find out what you can optimise

How does the application perform?

Benchmark the application: siege (httpperf, jmeter-http)

- check ~/.siegerc and set the logfile setting
- Create a file with urls:
http://derickrethans.nl/
http://derickrethans.nl/spatial-indexes-data-sqlite.html
http://derickrethans.nl/talks.html
http://derickrethans.nl/who.html
- run against your code: `siege -c 4 -r 10 -f /tmp/urls.txt`

```
Transactions:          40 hits
Availability:         100.00 %
Elapsed time:          9.76 secs
Data transferred:     1.34 MB
Response time:         0.18 secs
Transaction rate:      4.10 trans/sec
Throughput:           0.14 MB/sec
Concurrency:          0.73
Successful transactions: 40
Failed transactions:   0
Longest transaction:  0.39
Shortest transaction:  0.08
```

Is it my code that is slow?

Multiple possible reasons:

- The database is slow
- There is lots of IO
- Your code is slow
- The system is busy with other things

```
$ vmstat 1
```

```
procs -----memory----- ---swap-- -----io----- -system-- ----cpu----
r  b   swpd   free   buff   cache   si   so    bi    bo    in   cs  *|dd1111|us sy|* id wa
5  0 121248 367520 767824 3080668   0   0  11272    0 1593 5540 *|dd1111|74 22|*  3  0
5  0 121248 290784 767960 3082980   0   0    268    0 1555 5381 *|dd1111|77 20|*  3  0
5  0 121248 238340 768132 3084336   0   0   1364 21160 2263 6815 *|dd1111|70 21|*  3  7
6  0 121248 170772 768300 3087100   0   0   1652    0 1802 8540 *|dd1111|71 25|*  4  0
```

CPU is (close to) fully in use: 74 + 22 → your code is slow.

```
$ vmstat 1
```

```
procs -----memory----- ---swap-- -----io----- -system-- ----cpu----
r  b   swpd   free   buff   cache   si   so    bi    bo    in   cs  us  sy id *|dd1111|wa|*
0  1 121248  50776 314796 4156000   0   0 245800    0 3018 5394  1 11 71 *|dd1111|17|*
1  0 121248  52112 315108 4236692   0   0 243672    0 3107 5552  1 10 71 *|dd1111|18|*
1  1 121248  50148 315256 4318268   0   0 243096    44 3146 5463  1 11 72 *|dd1111|17|*
1  1 121248  52120 315484 4396356   0   0 243784    0 3006 5419  1 10 72 *|dd1111|18|*
```

wait is > 10 → IO is the bottleneck.

In-code tools:

- Add timing points

External tools:

- Basic overview: included
- Deep details: xdebug

Timing Points

Code with analysis in mind:

- Add timing points around specific events
- Check changes over time

eZ Publish:
Symfony:

Timing points:

Checkpoint	Begins at (sec)	Lasts (sec)	Memory at start (KB)	Memory used (KB)
Module start 'setup'	1.5722	1.7874	6,439.2578	5,780.6953
Module end 'setup'	3.3596	2.6235	12,219.9531	8,173.0625
End	5.9831		20,393.0156	

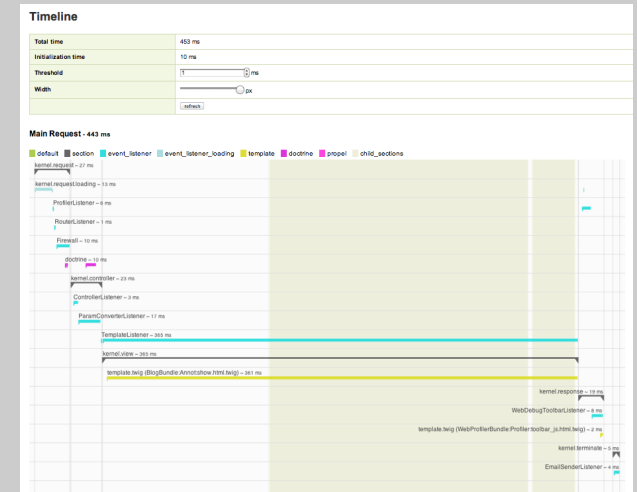
Resources:

Total runtime (sec): 6.0161
Peak memory usage (KB):22,098.9375

Time accumulators:

Accumulator	Lasts (sec)	Lasts (%)	Count	Average (sec)
ini_load				
Load cache	0.5950	9.8895	29	0.0205
Find INI Files	0.4026	6.6922	18	0.0224
Parse	0.0981	1.6302	18	0.0054
Save Cache	0.0699	1.1613	18	0.0039
Check MTime	0.0102	0.1703	11	0.0009
Mysql Total				
Mysql_queries	0.0285	0.4745	18	0.0016
Looping result	0.0005	0.0090	14	0.0000
Template Total	4.4332	73.7	3	1.4777
Template load	1.8871	31.3673	3	0.6290
String conversion in template resource	0.0066	0.1105	75	0.0001
Template parser: create text elements	0.0978	1.6264	42	0.0023
Template parser: remove whitespace	0.0264	0.4382	42	0.0006
Template parser: construct tree	0.3477	5.7793	42	0.0083
Template load and register function	0.0264	0.4394	11	0.0024
Template processing	2.5457	42.3150	3	0.8486
override				
Cache load	1.3062	21.7117	15	0.0871
General				
INI string conversion	0.0065	0.1075	74	0.0001
String conversion	0.0041	0.0683	149	0.0000
String conversion w/ mbstring	0.0004	0.0072	8	0.0001
dbfile	0.0581	0.9665	47	0.0012

Note: percentages do not add up to 100% because some accumulators overlap
Total runtime - incl. debug printing (sec):6.3711



Dumps includes/classes hierarchies

Install:

```
pecl install included
```

Add to php.ini:

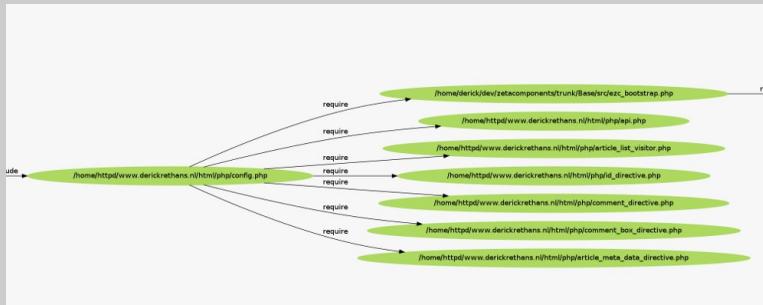
```
extension=included.so  
included.enabled=1  
included.dumpdir=/tmp
```

Generate graphs:

```
php -dincluded.enable=0 gengraph.php -t includes -i /tmp/included.22439.1  
dot -Tpng -o included-includes.png included.out.dot  
php -dincluded.enable=0 gengraph.php -t classes -i /tmp/included.22439.1  
dot -Tpng -o included-classes.png included.out.dot
```

Include overview: included-includes.png:

Class hierarchy: included-classes.png:





- Xdebug: An Open Source debugging tool
- About 10 years old
- Works on "every" operating system
- PHP 5.1, 5.2, 5.3, 5.4 and trunk
- Version 2.2 just out

Function trace

```
TRACE START [2010-03-24 11:03:12]
0.0022    787776    -> {main}() /home/httpd/pres2/show2.php:0
0.0028    803160     -> require(/home/derick/dev/ezcomponents/trunk/Base/src/
ezc_bootstrap.php)/home/httpd/pres2/show2.php:2
0.0029    803552     -> dirname('/home/derick/dev/ezcomponents/trunk/Base/src/
ezc_bootstrap.php')/home/derick/dev/ezcomponents/trunk/Base/src/ezc_bootstrap.php:12
0.0030    803968     -> explode('/', '/home/derick/dev/ezcomponents/trunk/Base/src')/home/
derick/dev/ezcomponents/trunk/Base/src/ezc_bootstrap.php:13
0.0031    807352     -> count(array (0 => '', 1 => 'home', 2 => 'derick', 3 => 'dev', 4 =>
'ezcomponents', 5 => 'trunk', 6 => 'Base', 7 => 'src'))/home/derick/dev/ezcomponents/trunk/Base/src/
ezc_bootstrap.php:15
0.0032    807800     -> array_slice(array (0 => '', 1 => 'home', 2 => 'derick', 3 => 'dev', 4
=> 'ezcomponents', 5 => 'trunk', 6 => 'Base', 7 => 'src'), 0, -2)/home/derick/dev/ezcomponents/
trunk/Base/src/ezc_bootstrap.php:17
0.0033    808952     -> join('/', array (0 => '', 1 => 'home', 2 => 'derick', 3 => 'dev', 4
=> 'ezcomponents', 5 => 'trunk'))/home/derick/dev/ezcomponents/trunk/Base/src/ezc_bootstrap.php:17
0.0065    1002544    -> require(/home/derick/dev/ezcomponents/trunk/Base/src/base.php)/home/
derick/dev/ezcomponents/trunk/Base/src/ezc_bootstrap.php:18
0.0066    990560     -> __autoload('ezcTemplateCustomFunction')/home/derick/dev/ezcomponents/
trunk/Base/src/ezc_bootstrap.php:0
0.0066    990560     -> ezcBase::autoload('ezcTemplateCustomFunction')/home/derick/dev/
ezcomponents/trunk/Base/src/ezc_bootstrap.php:38
0.0067    990560     -> ezcBase::setPackageDir()/home/derick/dev/ezcomponents/trunk/Base/
src/base.php:125
0.0068    990832     -> dirname('/home/derick/dev/ezcomponents/trunk/Base/src/base.php')/
home/derick/dev/ezcomponents/trunk/Base/src/base.php:244
0.0069    990728     -> array_key_exists('ezcTemplateCustomFunction', array ())/home/
derick/dev/ezcomponents/trunk/Base/src/base.php:128
0.0069    990728     -> array_key_exists('ezcTemplateCustomFunction', array ())/home/
derick/dev/ezcomponents/trunk/Base/src/base.php:142
0.0070    991536     -> preg_match('/^([a-z0-9]*)([A-Z][a-z0-9]*)?([A-Z][a-z0-9]*)?/',
'ezcTemplateCustomFunction', NULL)/home/derick/dev/ezcomponents/trunk/Base/src/base.php:157
0.0072    993304     -> sizeof(array (0 => 'ezcTemplateCustom', 1 => 'ezc', 2 =>
'Template', 3 => 'Custom'))/home/derick/dev/ezcomponents/trunk/Base/src/base.php:161
0.0073    993664     -> strtolower('Template_Custom_autoload.php')/home/derick/dev/
ezcomponents/trunk/Base/src/base.php:165
0.0073    993496     -> ezcBase::requireFile('template_custom_autoload.php',
'ezcTemplateCustomFunction', 'ezc')/home/derick/dev/ezcomponents/trunk/Base/src/base.php:167
0.0074    994184     -> file_exists('/home/derick/dev/ezcomponents/trunk/Base/src/../../../../
autoload/template_custom_autoload.php')/home/derick/dev/ezcomponents/trunk/Base/src/base.php:278
0.0075    993848     -> strtolower('Template_autoload.php')/home/derick/dev/ezcomponents/
trunk/Base/src/base.php:175
0.0076    993888     -> ezcBase::requireFile('template_autoload.php',
'ezcTemplateCustomFunction', 'ezc')/home/derick/dev/ezcomponents/trunk/Base/src/base.php:177
0.0077    994568     -> file_exists('/home/derick/dev/ezcomponents/trunk/Base/src/../../../../
autoload/template_autoload.php')/home/derick/dev/ezcomponents/trunk/Base/src/base.php:278
0.0100    1105728    -> require(/home/derick/dev/ezcomponents/trunk/autoload/
template_autoload.php)/home/derick/dev/ezcomponents/trunk/Base/src/base.php:280
```

Function trace to file

Automatic readable format

```
function-trace-comp.xt (~/dev/php/xdebug-demos) - VIM - Terminal
Version: 2.2.0-dev
File format: 2
TRACE START [2011-11-23 10:44:31]
1 0 0 0.001373 773608 {main} 1 /home/httpd/www.derickrethans.nl
2 1 0 0.001747 800080 include 1 /home/httpd/www.derickrethans.nl/bra
3 2 0 0.002017 815432 require 1 /home/derick/dev/ezcomponents/trunk/
4 3 0 0.002057 815824 dirname 0 /home/derick/dev/ezcomponents/tr
4 3 1 0.002110 815976
4 4 0 0.002156 816192 explode 0 /home/derick/dev/ezcomponents/tr
4 4 1 0.002200 819544
4 5 0 0.002224 819624 count 0 /home/derick/dev/ezcomponents/tr
4 5 1 0.002270 819624
4 6 0 0.002301 820072 array_slice 0 /home/derick/dev/ezcomponent
4 6 1 0.002352 821344
4 7 0 0.002374 821224 join 0 /home/derick/dev/ezcomponents/tr
4 7 1 0.002417 821392
4 8 0 0.004136 1014824 require 1 /home/derick/dev/ezcomponents/trunk/
4 8 1 0.004170 1014824
3 2 1 0.004190 1013848
3 9 0 0.007865 1438592 require 1 /home/httpd/www.derickrethans.nl/bra
3 9 1 0.007903 1438592
3 10 0 0.008274 1469824 require 1 /home/httpd/www.derickrethans.nl/bra
function-trace-comp.xt 1,1 Top
```

```
xdebug.auto_trace=1 ; enable tracing
xdebug.trace_format=1 ; selects computerized format
xdebug.trace_options=0 ; sets extra option (1 = append)
```

One bundled with Xdebug:

```
php ~/dev/php/xdebug/trunk/contrib/tracefile-analyser.php time-own 5
```

```
...
```

```
Showing the 5 most costly calls sorted by 'time-own'.
```

function	#calls	Inclusive time	memory	Own time	memory
array_pop	715	1.0252	-139880	1.0252	-139880
preg_match	2986	0.3718	1016336	0.3718	1016336
{main}	1	6.4562	7335704	0.3476	-15198832
next	432	0.2386	0	0.2386	0
count	3302	0.2132	0	0.2123	0

ValaXdebugTools (<http://tinyurl.com/vxdebugtools>):

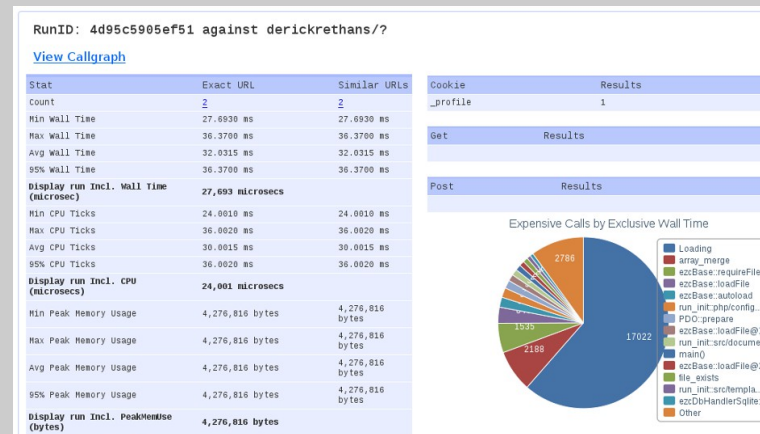
- `trace4func`: Give a function name, get a stack trace for each time it is called.
- `evaled`: Prints an ordered list of evals, together with what file is responsible for this eval.
- `whocalls`: Find out what files contain calls to the given function.
- `whatisincluded`: Find out what files are included and where they are included from.

demo

Profiling



- Was originally developed by Facebook
- XHProf and XHGui
- Meant to be run in a production environment



- Written for C-applications
- Xdebug uses the same format
- KCacheGrind is an awesome viewer
- Profile your app with both CacheGrind and Xdebug
- Alternative viewers: WinGrind, Macgrind, Webgrind, PHP Storm

demo

- Slides: <http://derickrethans.nl/talks/profiling-ipc2012s>
- Contact me: Derick Rethans: @derickr, derick@derickrethans.nl