Profiling PHP Applications

ConFoo - Montréal, Canada - Mar 1st, 2012

Derick Rethans - derick@derickrethans.nl - twitter: @derickr

http://joind.in/4695

About Me

Derick Rethans

- Dutchman living in London
- Freelancer doing PHP internals develop
  ie.: writing extensions for a living
- PHP Driver Maintainer for 10gen (the M
  people)
- Anderskor's Camouflage Encoder
- 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

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 (httperf, 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.

Understand your code, application and execution paths

In-code tools:

- Add timing points

External tools:

- Basic overview: inclued
- Deep details: xdebug

Timing Points

Code with analysis in mind:

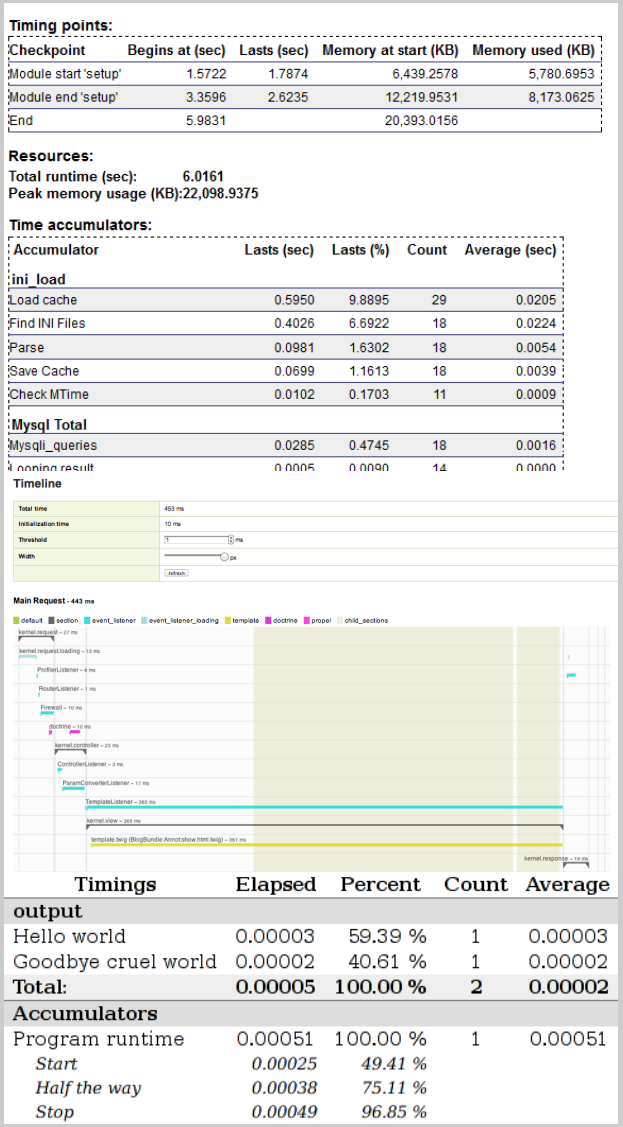
- Add timing points around specific events
- Check changes over time

eZ Publish:

Symfony:

Zeta Components:

```
<?php
// Get the one and only instance of the ezcDebug.
$debug = ezcDebug::getInstance();
// Start the accumulator.
$debug->startTimer( "Program runtime", "Accumulators" );
$debug->switchTimer( "Start", "Program runtime" );
// The name of the timer is: "Hello world" and it will be
// placed in the group: "output".
$debug->startTimer ("Hello world", "output" );
echo "Hello world<br/>";
$debug->stopTimer( "Hello world" );
// Replace the "Start" timer for "Half the way".
$debug->switchTimer( "Half the way", "Start" );
// Measure the time of writing "cruel world".
$debug->startTimer( "Goodbye cruel world", "output" );
echo "Goodbye cruel world<br/>";
$debug->stopTimer( "Goodbye cruel world" );
// Stop the last timer.
$debug->switchTimer( "Stop", "Half the way" );
$debug->stopTimer( "Stop" );
// Get HTML output.
$output = $debug->generateOutput();
?>
```



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				
Mysqli_queries	0.0285	0.4745	18	0.0016
Looping result	0.0005	0.0090	14	0.0000

Timeline

Main Request - 443 ms

Timings	Elapsed	Percent	Count	Average
output				
Hello world	0.00003	59.39 %	1	0.00003
Goodbye cruel world	0.00002	40.61 %	1	0.00002
Total:	**0.00005**	**100.00 %**	**2**	**0.00002**
Accumulators				
Program runtime	0.00051	100.00 %	1	0.00051
Start	*0.00025*	*49.41 %*		
Half the way	*0.00038*	*75.11 %*		
Stop	*0.00049*	*96.85 %*		

Inclued

Dumps includes/classes hiearchies

Install:

```
pecl install inclued
```

Add to php.ini:

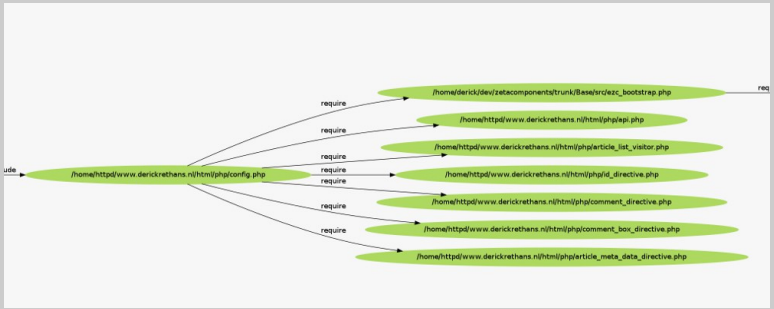
```
extension=inclued.so
inclued.enabled=1
inclued.dumpdir=/tmp
```

Generate graphs:

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

Include overview: inclued-includes.png:

Class hiearchy: inclued-classes.png:



I Do Need a Debugger

- 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 about to follow with PHP 5.4

Function trace

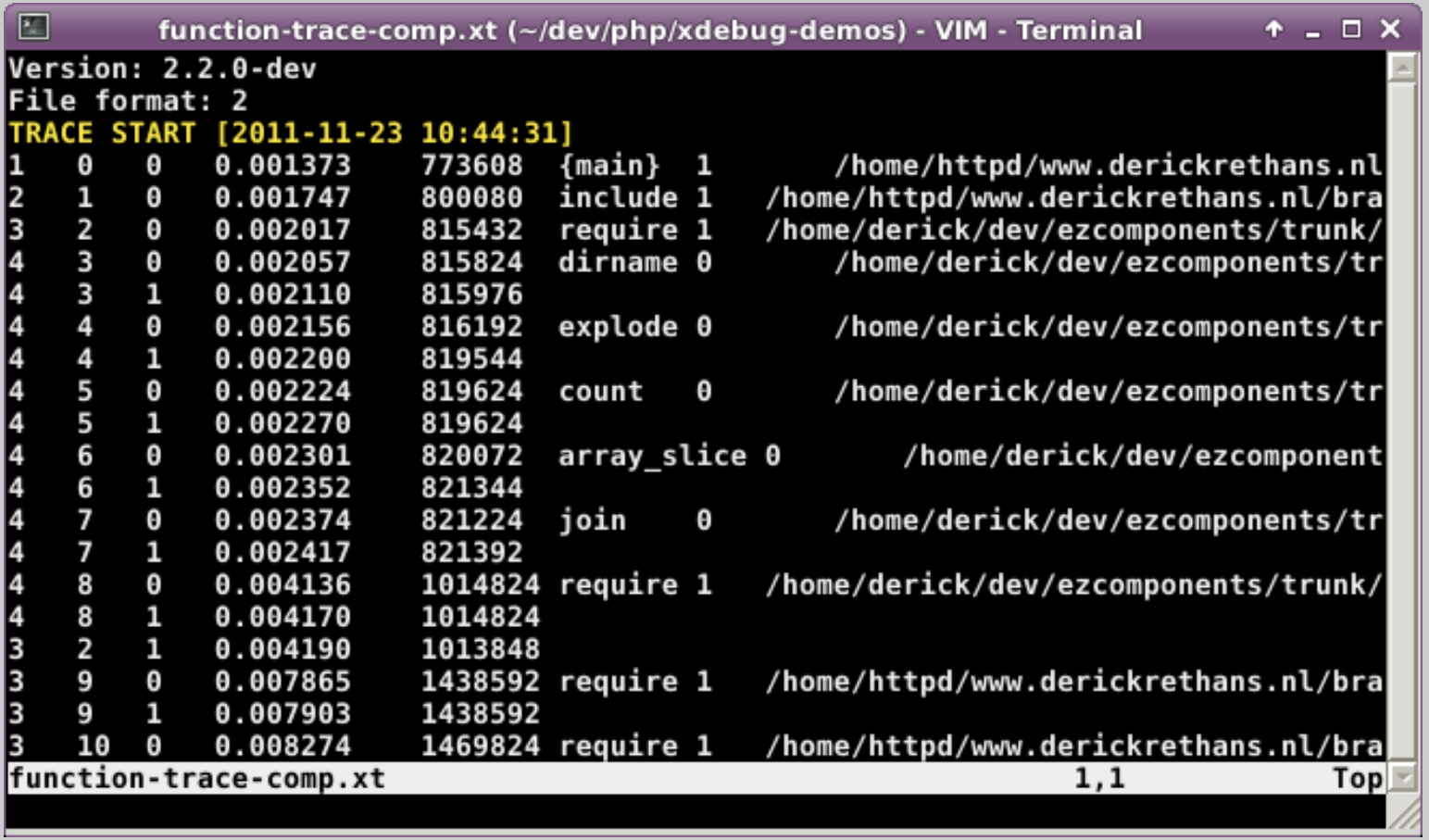
```
TRACE START [2010-03-24 11:03:12]
    0.0022     787776   -> {main}() /home/httpd/pres2/show2.php:0
    0.0028     803160     -> require(/home/derick/dev/ezcomponents/trunk/Base/src/
ezc_bootstrap.php)/home/httpd/pres2/show2.php:2
    0.0029     803552       -> dirname('/home/derick/dev/ezcomponents/trunk/Base/src/
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
```

Some settings:

```
xdebug.auto_trace=1
xdebug.trace_output_dir=/tmp
xdebug.collect_params=1
xdebug.collect_return=1
xdebug.collect_includes=1
xdebug.collect_assignments=1
```

Function trace to file

Automatic readable format




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

Function trace

Other functionality

- HTML traces
- Tracing only parts of an application with xdebug_start_trace() and xdebug_stop_trace().
- Fetching the trace file name that is being used with xdebug_get_tracefile_name().
- Changing how much data is shown with xdebug.var_display_max_children, xdebug.var_display_max_data and xdebug.var_display_max_depth.

Trace file analysers

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'.

                                                                  Inclusive         Own
function                                                  #calls  time     memory  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):

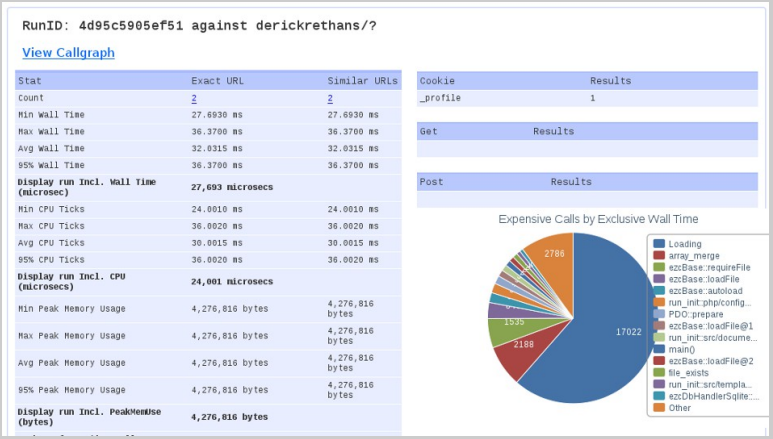
- trace4func: Give a function name, get a stack trace for each time it is called.
- evaled: Prints an ordered list of evals, together with what file is responsible for this evil.
- whocalls: Find out what files contain calls to the given function.
- whatisincluded: Find out what files are included and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

demo

Profiling

XHProf

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



Profiling

Valgrind / Cachegrind

- 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

Resources

- Slides: http://derickrethans.nl/talks/:-:talk_id:-:
- Contact me: Derick Rethans: @derickr, derick@derickrethans.nl
- Feedback: http://joind.in/4695