

# Profiling PHP Applications

4 Developers - Warsaw, Poland - Apr 4, 2011

Derick Rethans - [derick@derickrethans.nl](mailto:derick@derickrethans.nl) - twitter:  
@derickr

<http://derickrethans.nl/talks.html>

## Derick Rethans



- Dutchman living in London
- PHP development
- Author of the `mcrypt`, `input_filter`, `dbus`, `translit` and `date/time` extensions
- Author of `Xdebug`
- Contributor to the Apache Zeta Components Incubator project (formerly eZ Components)
- Freelancer doing PHP (internals) development

# The Internet is Full of Nonsense

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

echo is faster than print

Use sprintf instead of variables contained in double quotes, it's about 10x faster.

Use `<?php ... ?>` tags when declaring PHP as all other styles are depreciated, including short tags.

# 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

- 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/using-  
openstreetmap-with-flickr.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
```

# Is it my code that is slow?

There could be multiple reasons why the application is slow:

- 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:  $74 + 22 \rightarrow$  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  $> 10 \rightarrow$  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

```
<?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();
?>
```

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

## Dumps includes/classes hierarchies

### Install:

```
pecl install included
```

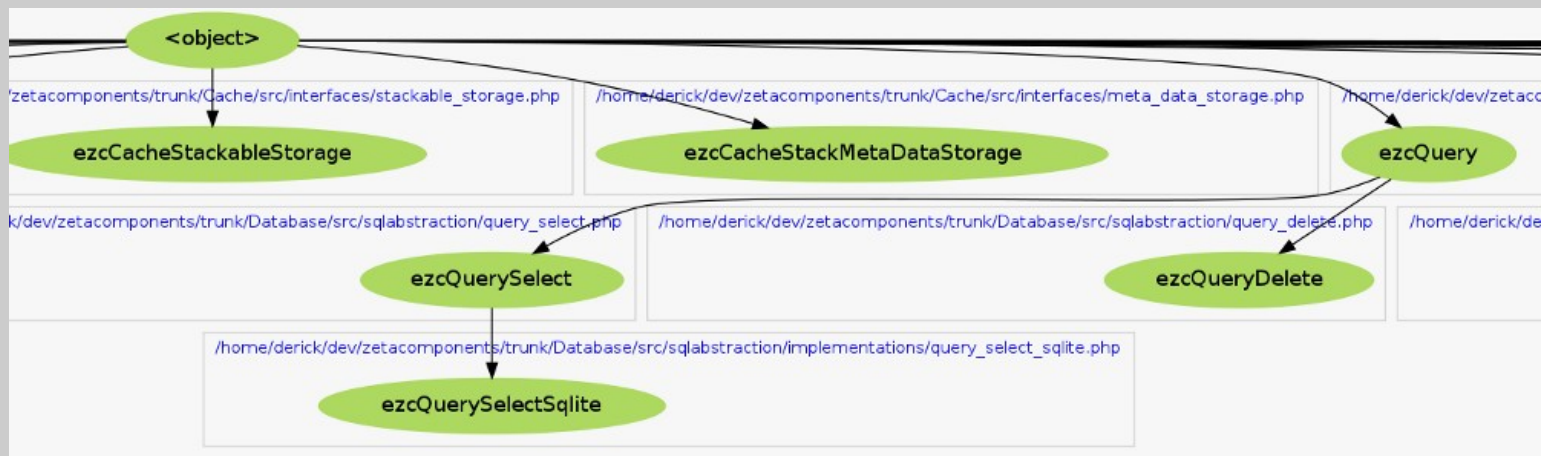
### Add to php.ini:

```
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
```

## Class hierarchy: included-classes.png:





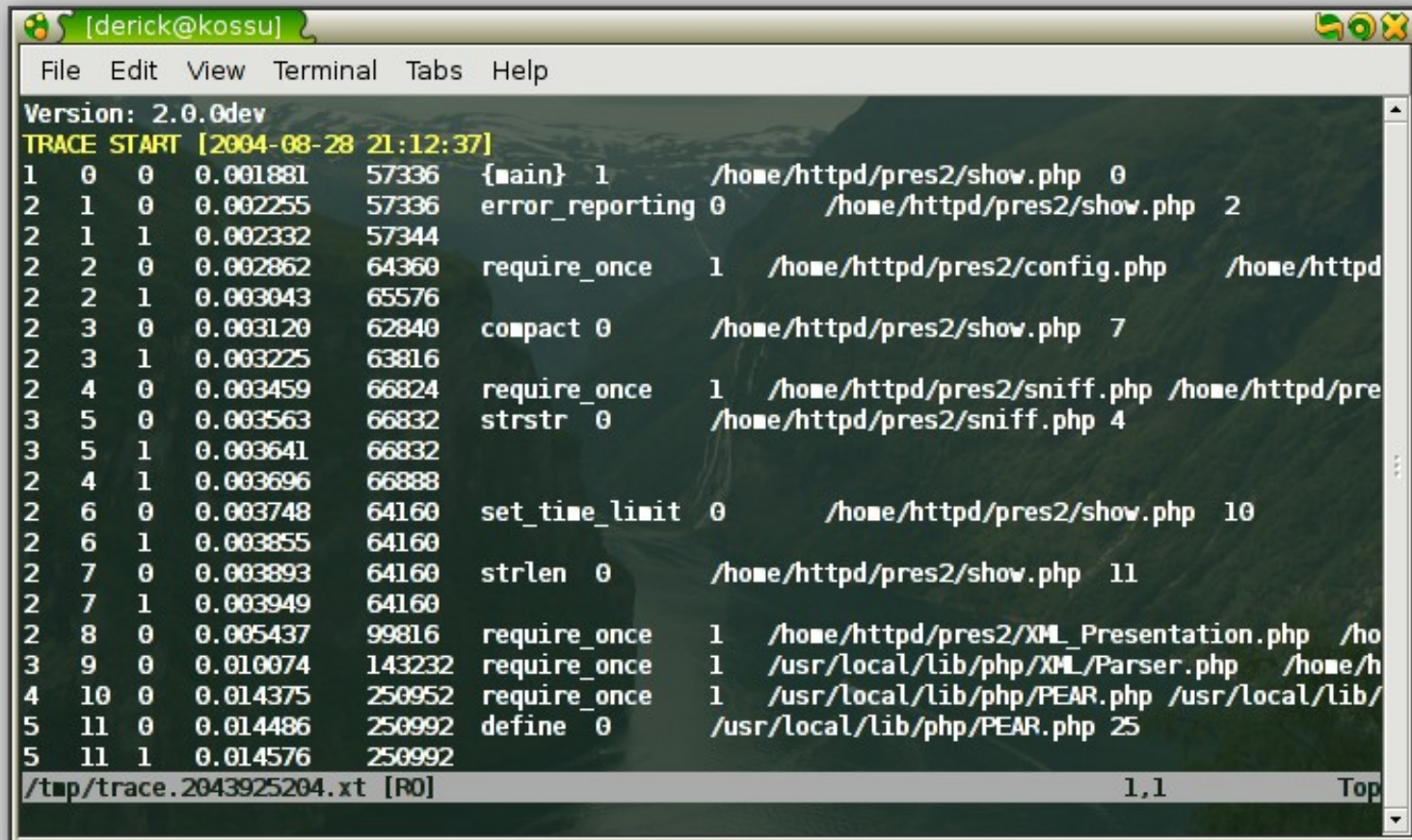
- Xdebug: An Open Source debugging tool
- About 8 years old
- Works on "every" operating system
- Version 2.1 released earlier this year
- PHP 5.1, 5.2, 5.3 and trunk

# 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



```
[derick@kossu]
File Edit View Terminal Tabs Help
Version: 2.0.0dev
TRACE START [2004-08-28 21:12:37]
1 0 0 0.001881 57336 {main} 1 /home/httpd/pres2/show.php 0
2 1 0 0.002255 57336 error_reporting 0 /home/httpd/pres2/show.php 2
2 1 1 0.002332 57344
2 2 0 0.002862 64360 require_once 1 /home/httpd/pres2/config.php /home/httpd
2 2 1 0.003043 65576
2 3 0 0.003120 62840 compact 0 /home/httpd/pres2/show.php 7
2 3 1 0.003225 63816
2 4 0 0.003459 66824 require_once 1 /home/httpd/pres2/sniff.php /home/httpd/pre
3 5 0 0.003563 66832 strstr 0 /home/httpd/pres2/sniff.php 4
3 5 1 0.003641 66832
2 4 1 0.003696 66888
2 6 0 0.003748 64160 set_time_limit 0 /home/httpd/pres2/show.php 10
2 6 1 0.003855 64160
2 7 0 0.003893 64160 strlen 0 /home/httpd/pres2/show.php 11
2 7 1 0.003949 64160
2 8 0 0.005437 99816 require_once 1 /home/httpd/pres2/XML_Presentation.php /ho
3 9 0 0.010074 143232 require_once 1 /usr/local/lib/php/XML/Parser.php /home/h
4 10 0 0.014375 250952 require_once 1 /usr/local/lib/php/PEAR.php /usr/local/lib/
5 11 0 0.014486 250992 define 0 /usr/local/lib/php/PEAR.php 25
5 11 1 0.014576 250992
/tmp/trace.2043925204.xt [R0] 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)
```

- 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`.

## One bundled with Xdebug:

```
php ~/dev/php/xdebug/trunk/contrib/tracefile-analyser.php
...
Showing the 25 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

## 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



## Install:

```
pecl download xhprof-beta
tar -xvzf xhprof-0.9.2.tgz
cd xhprof-0.9.2/extension
phpize && ./configure && make install
```

RunID: 4d95c5905ef51 against derickrethans/?

[View Callgraph](#)

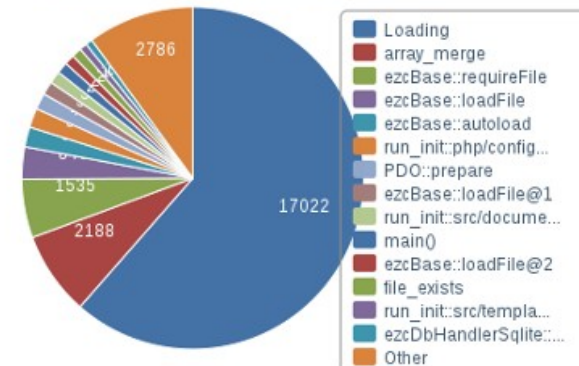
Stat	Exact URL	Similar URLs
Count	<u>2</u>	<u>2</u>
Min Wall Time	27.6930 ms	27.6930 ms
Max Wall Time	36.3700 ms	36.3700 ms
Avg Wall Time	32.0315 ms	32.0315 ms
95% Wall Time	36.3700 ms	36.3700 ms
<b>Display run Incl. Wall Time (microsec)</b>	<b>27,693 microsecs</b>	
Min CPU Ticks	24.0010 ms	24.0010 ms
Max CPU Ticks	36.0020 ms	36.0020 ms
Avg CPU Ticks	30.0015 ms	30.0015 ms
95% CPU Ticks	36.0020 ms	36.0020 ms
<b>Display run Incl. CPU (microsecs)</b>	<b>24,001 microsecs</b>	
Min Peak Memory Usage	4,276,816 bytes	4,276,816 bytes
Max Peak Memory Usage	4,276,816 bytes	4,276,816 bytes
Avg Peak Memory Usage	4,276,816 bytes	4,276,816 bytes
95% Peak Memory Usage	4,276,816 bytes	4,276,816 bytes
<b>Display run Incl. PeakMemUse (bytes)</b>	<b>4,276,816 bytes</b>	

Cookie	Results
_profile	1

Get	Results
-----	---------

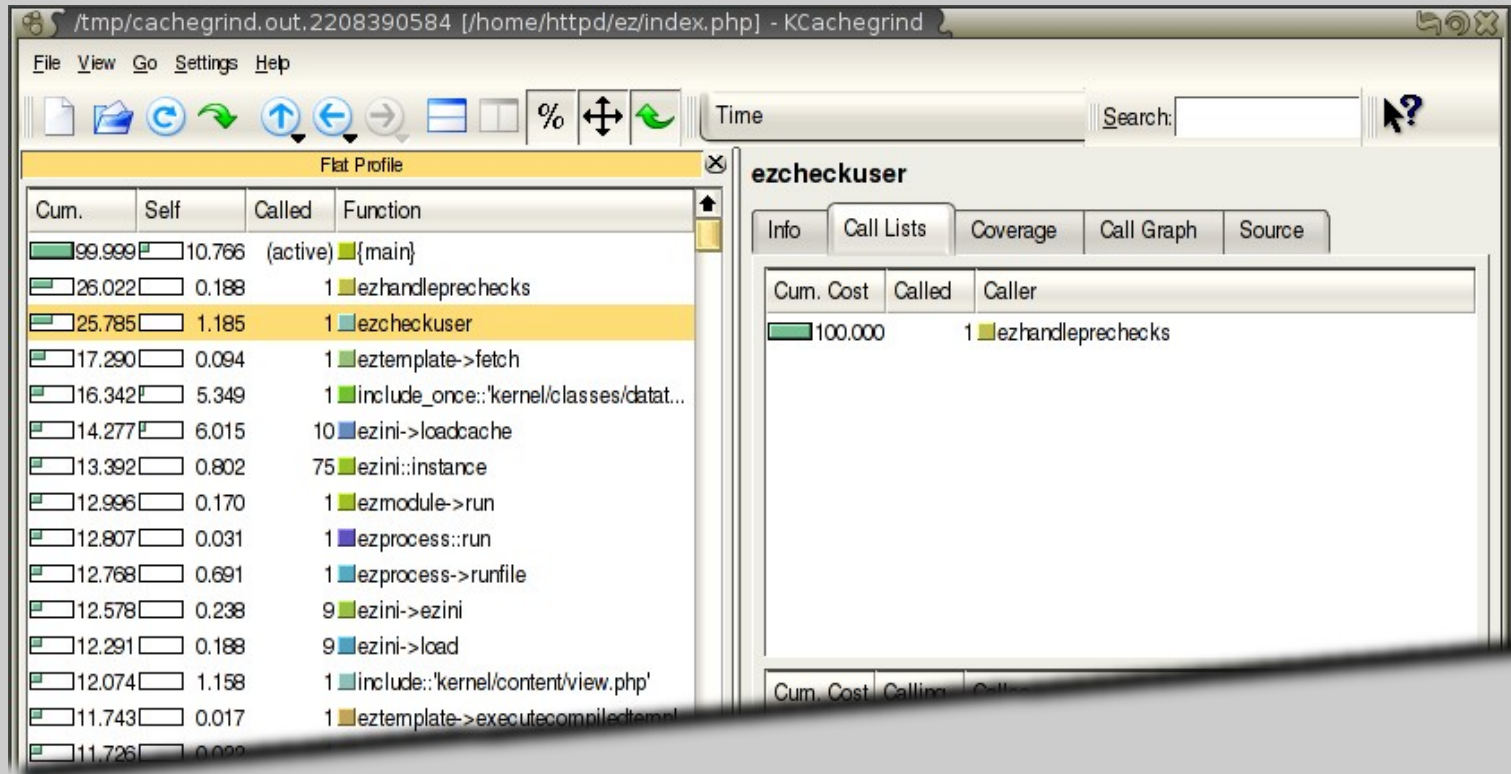
Post	Results
------	---------

Expensive Calls by Exclusive Wall Time



# Profiling

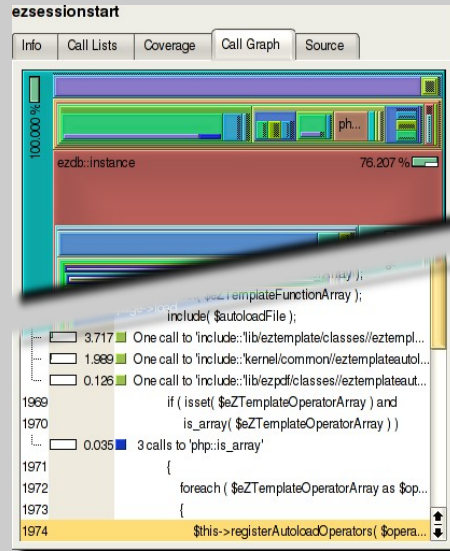
## KCacheGrind's Flat Profile and Call List



```
xdebug.profiler_enable=1           ; enable profiler
xdebug.profiler_output_dir=/tmp     ; output directory
xdebug.profiler_output_name=cachegrind.out.%p
```

# Profiling

## KCacheGrind's Call Graph and Source Annotations

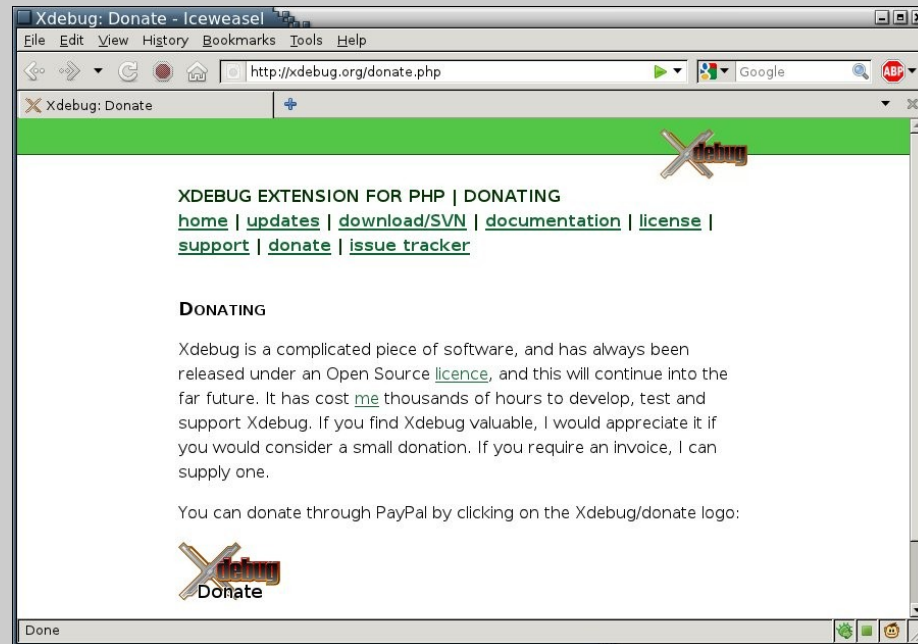


- Call graph
- Area shows time spend
- Stacked to show callees
- Source annotations
- Number of calls
- Total time per function

demo

# Xdebug

- It's Open Source and free (as in "free beer")
- Working on Xdebug takes up a lot of spare time
- I don't have a lot of spare time



# Resources

- Siege: <http://www.joedog.org/index/siege-home>
- Apache Zeta Components:  
<http://incubator.apache.org/zetacomponents/>
- Included: <http://uk2.php.net/included>
- ValaXdebugTools: <http://tinyurl.com/vxdebugtools>
- XHProf:  
<http://mirror.facebook.net/facebook/xhprof/>
- XHGui: <https://github.com/preinheimer/xhprof>
- Xdebug: <http://xdebug.org>
- If you like Xdebug: <http://xdebug.org/donate.php>
- These slides: <http://derickrethans.nl/talks.html>