

Profiling PHP with XDebug and kCacheGrind

Marcus Bointon at PHP London, September 7th 2006.

You've been here before...

- Notice that your code is a bit slow
- Guess which bits are responsible
- Make some changes, maybe micro-benchmark them:

```
<?php
$z = microtime(true);
f1();
echo "f1 took ".microtime(true) - $z;
?>
```
- Pick the method with the best results
- Hopefully notice an overall speedup, hope you didn't leave any of your micro-benchmarking code behind

Wouldn't it be cool if...

- You could benchmark every single line of code without having to touch it
- You could find out which bits of your code are slowest
- You could see which bits are **not** worth improving (see [Amdahl's Law](#))
- You could easily compare alternative versions
- You could immediately measure what difference your changes make
- You could find performance problems you'd never considered
- You get all this with Profiling

What difference can profiling make?

- Sebastian Bergmann: *“Over the past couple of days, [we] used Xdebug's profiling functionality to locate "hot spots" in PHPUnit 3's report generator. After optimizing most of these hot spots [...], the initial six hours have been reduced to eight minutes. Wow.”*
- In my own projects I've found 400% speedups and hidden performance problems that would otherwise have gone unnoticed.
- Lends weight to Hoare's dictum: *“We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil.”*
http://www.acm.org/ubiquity/views/v7i24_fallacy.html
Generally, a profiler can tell you more interesting things about your code than you can!

Software

- XDebug by Derick Rethans
<http://xdebug.org/>
Debugging and profiling extension for PHP
- kCacheGrind by Josef Weidendorfer
<http://kcachegrind.sourceforge.net/cgi-bin/show.cgi>
Visual presenter of profiling information for multiple languages
- WinCacheGrind by Hendy Irawan
<http://sourceforge.net/projects/wincachegrind/>
Lightweight version of kCacheGrind for Windows
- wxDebug by Harry Fuecks
<http://wxdebug.python-hosting.com/>
A basic Python web app for viewing profiler output.

Installation: XDebug

- Get the CVS version from xdebug.org
It's a moving target, and this is dev software...

```
cvscvs -d :pserver:srmread@cvs.xdebug.org:/repository login
CVS password: srmread
cvscvs -d :pserver:srmread@cvs.xdebug.org:/repository co xdebug
```
- Compile and install:

```
./configure --enable-xdebug \
--with-php-config=/usr/local/bin/php-config
make
make install
```
- To get updates from CVS (Keep an eye on what's new in the Changelog):

```
cvscvs up
make
make install
```
- Don't do this on a live server as it will slow it down!

XDebug php.ini configuration

- `zend_extension="/usr/local/lib/php/extensions/no-debug-non-zts-20050922/xdebug.so"`
`xdebug.default_enable = On`
`xdebug.manual_url = "http://uk.php.net"`
`xdebug.show_local_vars = 1`
`xdebug.profiler_append = 0`
`xdebug.profiler_enable = 1`
`xdebug.profiler_output_dir = "/Users/marcus/development/xdebug_profiler"`
`xdebug.profiler_output_name = "script"`
- 'script' profiler_output_name option (not covered in docs, only Changelog) ensures sensible filename output.

Installation: kCacheGrind

- Linux and friends:

Available in many distros, present in the `kdeskk` package.

Otherwise links from the [xdebug](#) site.

- MacOS X:

Install via [fink](#) (set to use unstable tree first):

```
sudo fink install kcachegrind
```

Prepare to wait for a VERY long time to compile, probably overnight, as it has to compile most of KDE, over 120 dependencies in all!

End result runs in X11 server - add it to your X11 apps menu for convenience.

- Windows:

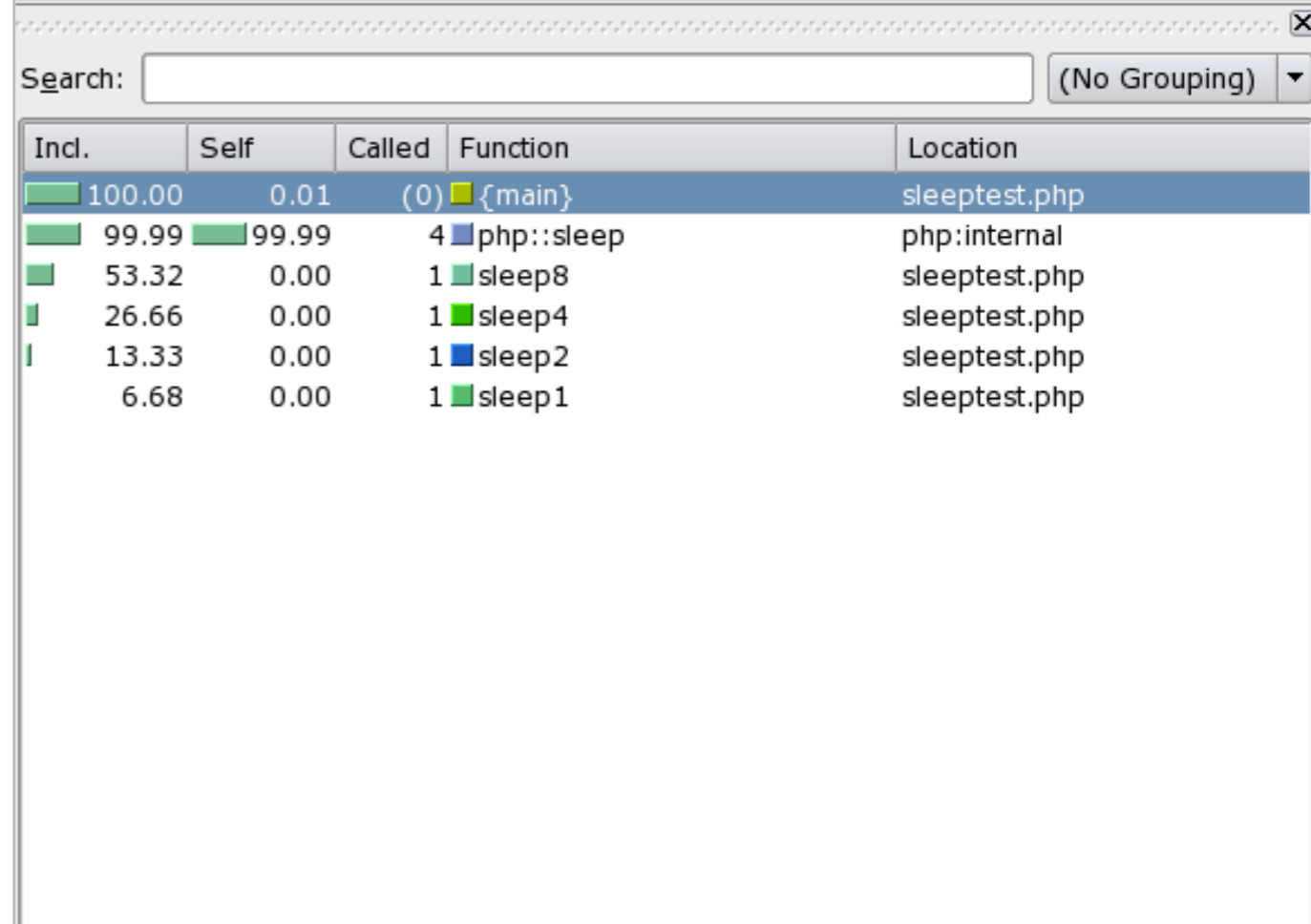
Install WinCacheGrind instead.

Simple example

- `<?php`
function sleep1() {sleep(1);}
function sleep2() {sleep(2);}
function sleep4() {sleep(4);}
function sleep8() {sleep(8);}
sleep1();
sleep2();
sleep4();
sleep8();
`?>`
- It's 'wallclock' time that matters - sleep() doesn't eat CPU

Profiling stats

- Lists all functions called everywhere in the page's execution. (Limit in prefs)
- 'Incl.' is the proportion of time consumed both within the function itself, and by every function that it calls
- 'Self' is the time taken by the internals of the function alone, not including time spent in functions it calls.
- 'Called' is the number of times the function is called.
- Grouping allows you to group functions that share e.g. the same source file.

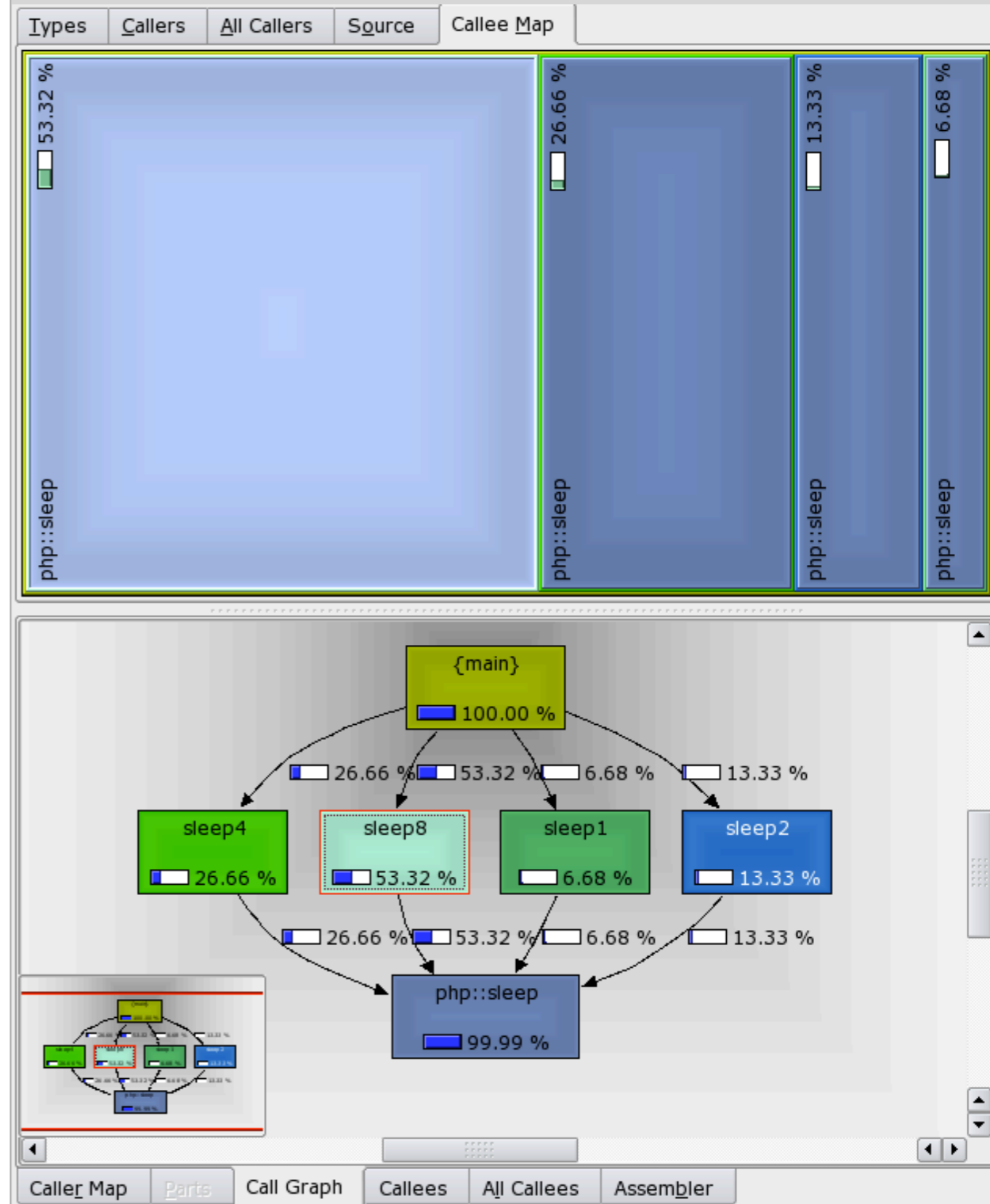


The screenshot shows a window with a search bar and a dropdown menu set to '(No Grouping)'. Below is a table with the following data:

Incl.	Self	Called	Function	Location
100.00	0.01	(0)	{main}	sleeptest.php
99.99	99.99	4	php::sleep	php:internal
53.32	0.00	1	sleep8	sleeptest.php
26.66	0.00	1	sleep4	sleeptest.php
13.33	0.00	1	sleep2	sleeptest.php
6.68	0.00	1	sleep1	sleeptest.php

Profiling graphs

- Many graphs available - these are the most useful
- Top graph is the 'Callee map', showing time taken by each of the functions called.
Time == area
- Lower graph shows which functions called which other functions
- Most of the displays depend on what's selected in the function list or on the graph.
Here sleep8 is selected within a view on {main}



Complex example

- Navigability of graphs really helps to narrow focus on your performance problem. Just double click to zoom in for more detail
- What to look for?
- Functions with large self times
- Functions with excessive call counts
- Functions being called unexpectedly



XDebug Error Reporting

- Much nicer error display
- Stack traces help show where it went wrong
- Optional display of local variables
- Can link directly to PHP docs on a server of your choice
- Also prettifies output from `var_dump()`.

Warning: Division by zero in `/Users/marcus/Sites/phplondon/error.php` on line 5

Warning: Division by zero in `/Users/marcus/Sites/phplondon/error.php` on line 5

Call Stack		
#	Function	Location
1	{main}()	/Users/marcus/Sites/phplondon/error.php:0
2	f1()	/Users/marcus/Sites/phplondon/error.php:9
3	f2()	/Users/marcus/Sites/phplondon/error.php:2
4	f3()	/Users/marcus/Sites/phplondon/error.php:3
5	f4()	/Users/marcus/Sites/phplondon/error.php:4

Variables in local scope (#5)	
Variable	Value
\$a =	array 0 => 1 1 => 2 2 => 3
\$x =	'hello' (length=5)
\$z =	1

Other XDebug features

- Not surprisingly, it's a full PHP debugger with support for remote debugging.
- Code coverage reporting
Checks the effectiveness of your unit tests
- Trace

Addendum

- You'll notice that nearly all kCacheGrind's stats are relative, from which you can't easily tell if your changes gained or lost overall speed. At the bottom of the kCacheGrind Window is a 'total cost' figure (apparently unitless). The trick is to run the script, open the profile in kCacheGrind, note the total cost figure. Look at the stats/graphs to spot where you might make improvements, make some changes, refresh the page in your browser, then switch to kCacheGrind and hit the refresh button in there while keeping an eye on the cost figure - you'll immediately see if your changes actually helped. If you want to keep track of a series of changes, make copies of the profile at each stage so that you can compare them.