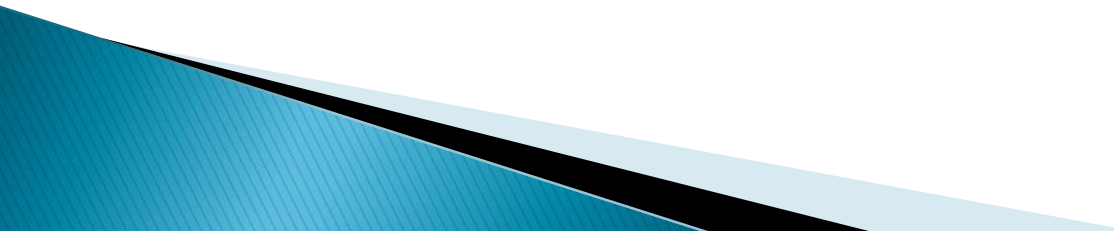


PHP applications profiling and detecting bottlenecks

Vaidas Zlotkus
VilniusPHP 0x11
2014-04-03

Agenda

- ▶ About me
 - ▶ What is Software Profiling?
 - ▶ Why profile your code?
 - ▶ Xdebug
 - ▶ KCacheGrind (QCacheGrind)
 - ▶ Webgrind
 - ▶ Examples (demo)
 - ▶ Summary
- 

About me

- ▶ Currently working at Santa Monica Networks (DC infrastructure architect/specialist). Mainly Enterprise stuff...
- ▶ Hobbies: UNIX-Like OS (Fedora/RHEL family, FreeBSD), OS and app performance, electronics
- ▶ Work experience:
 - PHP programmer. Still working examples: www.music.lt, www.paveikslai.lt, www.knygos.lt (former www.level.lt).
 - Senior system administrator at **Interdata** (later merged with **Hostex** and now **BDC**). Designing, implementing and operating server infrastructure for largest web projects in LT.
 - Team leader at BDC DC division. Responsible for Linux, VMware and Networking teams. 20% of time working with core systems like DNS, email, virtualization clusters.
- ▶ LinkedIn: <https://www.linkedin.com/in/muanton>
- ▶ Blog: <http://www.zlotkus.com/>

What is Software Profiling?

- ▶ In software engineering, profiling ("program profiling", "software profiling") is a form of dynamic program analysis that measures, for example, the space (memory) or time complexity of a program, the usage of particular instructions, or frequency and duration of function calls. The most common use of profiling information is to aid program optimization.
 - Source: Wikipedia.

Types of Profilers

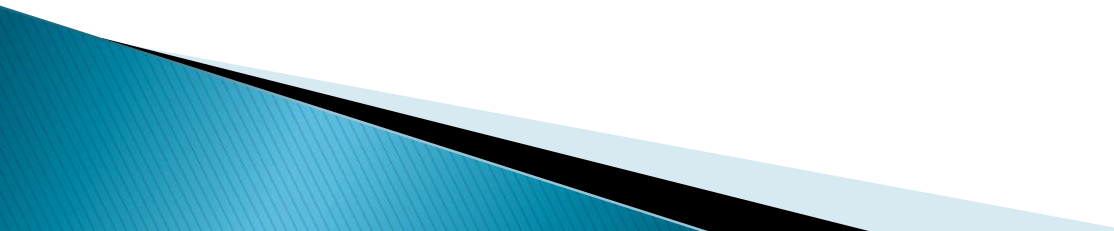
▶ Statistical

- Some profilers operate by sampling. A sampling profiler probes the target program's program counter at regular intervals using operating system interrupts. Sampling profiles are typically less numerically accurate and specific, but allow the target program to run at near full speed.

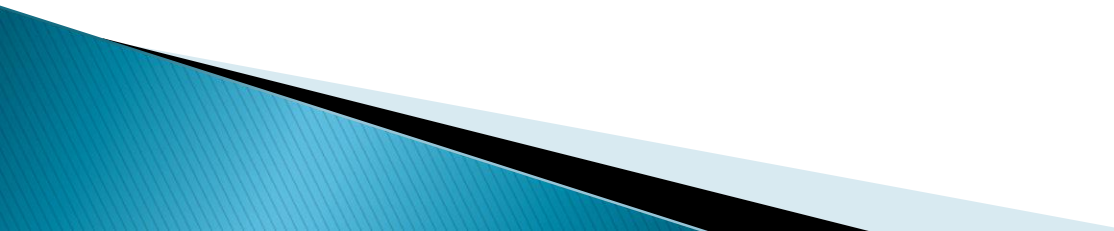
▶ Instrumenting (Xdebug behavior)

- This technique effectively adds instructions to the target program to collect the required information. Note that instrumenting a program can cause performance changes, and may in some cases lead to inaccurate results and/or heisenbugs.

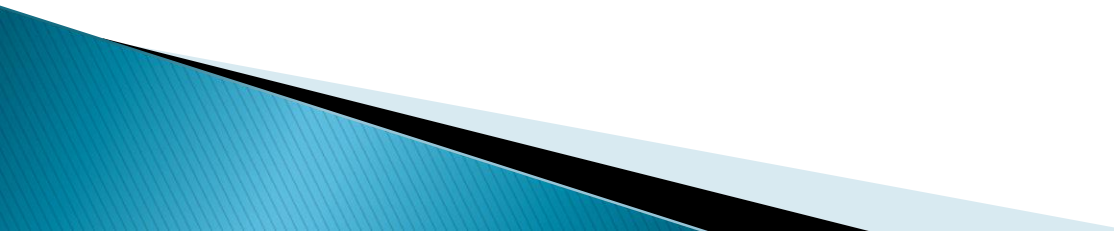
Why profile your code?

- ▶ Better understand your code
 - ▶ Improve your code quality and become better coder
 - ▶ Identify performance bottlenecks and their origin (code, physical resources)
 - ▶ Save money (hardware costs, more you need more you pay)
 - ▶ Improve user experience
- 

Xdebug Advantages

- ▶ Simple to install and configure
 - ▶ Requires no code changes
 - ▶ Lots of details since it is not sampling
 - ▶ Good for non-interactive systems
- 

Cautions!

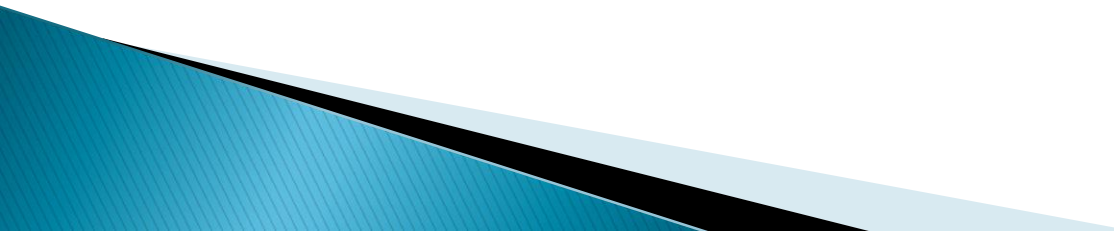
- ▶ Understand your problem: Changing the workload can change the bottleneck.
 - ▶ Over optimization
 - ▶ Profiler slows down your application.
 - ▶ Consider twice before using it in production
 - ▶ Does not work with Zend Encoded files
 - ▶ Does not analyze real workload
- 

Tools

Xdebug

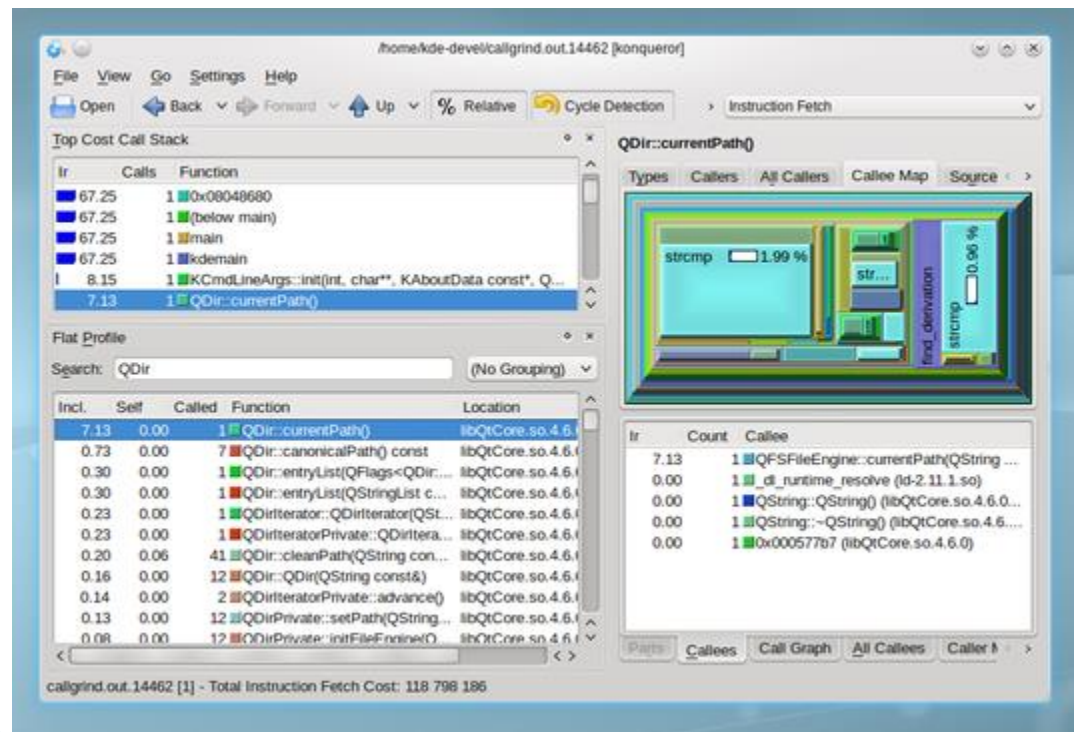
- ▶ Xdebug is a PHP extension which provides debugging and **profiling** capabilities.
- ▶ The debug information that Xdebug can provide includes the following:
 - stack and function traces in error messages
 - memory allocation
 - protection for infinite recursions
 - **profiling information for PHP scripts**
 - code coverage analysis
 - capabilities to debug your scripts interactively with a debugger front-end.

Xdebug

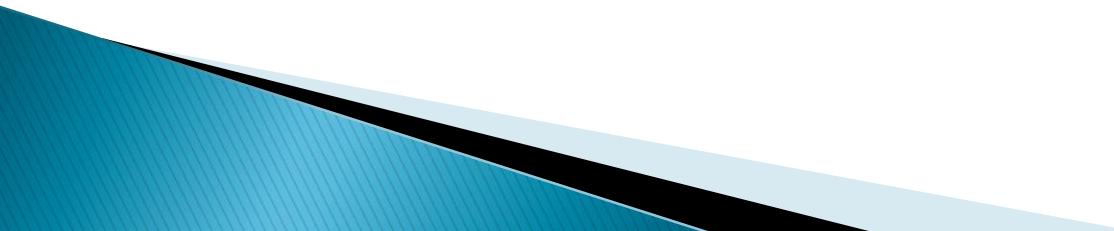
- ▶ Xdebug's Profiler is a powerful tool that gives you the ability to analyze your PHP code and determine bottlenecks or generally see which parts of your code are slow and could use a speed boost.
 - ▶ The profiler in Xdebug 2 outputs profiling information in the form of a cachegrind compatible file.
 - ▶ This allows you to use the excellent KCacheGrind tool (Linux/Windows, KDE) to analyse your profiling data.
- 

KCacheGrind

- ▶ KCachegrind is a profile data visualization tool, used to determine the most time consuming execution parts of program.



KCacheGrind visualizes Callgrind files

- ▶ Callgrind is a profiling tool that records the call history among functions in a program's run as a call-graph. By default, the collected data consists of the number of instructions executed, their relationship to source lines, the caller/callee relationship between functions, and the numbers of such calls. Optionally, cache simulation and/or branch prediction (similar to Cachegrind) can produce further information about the runtime behavior of an application.
- 

Webgrind

- ▶ Webgrind is an Xdebug profiling web frontend in PHP5. It implements a subset of the features of kcachegrind and installs in seconds and works on all platforms.
- ▶ Features:
 - Super simple, cross platform installation – obviously :)
 - Track time spent in functions by self cost or inclusive cost. Inclusive cost is time inside function + calls to other functions.
 - See if time is spent in internal or user functions.
 - See where any function was called from and which functions it calls.

Live Demo Session

» Xdebug, QCacheGrind, Webgrind

Demo Environment

- ▶ Server 1 (VM):
 - Apache 2.4.9
 - Fedora 20
 - PHP 5.5.10
 - Xdebug 2.2.4
 - MariaDB 5.5.36
- ▶ Server 2:
 - Apache 2.2.15
 - CentOS 6.5
 - PHP 5.3.3
 - Xdebug 2.1.4
 - MySQL 5.1.73

Summary (links)

- ▶ Xdebug
 - <http://xdebug.org/>
- ▶ KCachegrind
 - <http://kcachegrind.sourceforge.net/html/Home.html>
- ▶ Qcachegrind
 - <http://sourceforge.net/projects/qcachegrindwin/>
- ▶ Webgrind
 - <https://github.com/jokkedk/webgrind>

» Experiment

»» Thank You!