



Добро пожаловать в третью статью о xdebug. Уже сейчас вы должны были попробовать xdebug, если нет, сделайте это сегодня ;-).

В первой [статье](#) рассказывалось о том, как установить и настроить xdebug, описывались некоторые простейшие возможности, такие как улучшение вывода функции `var_dump()` или вывод трассировки стека вызовов при получении сообщения об ошибке. Во [второй части](#) мы рассмотрели такую возможность xdebug как трассировку. Трассировка содержит все вызовы функций и методов в программе, время запуска, опционально размер памяти, передаваемые и возвращаемые параметры. Лог трассировки может помочь вам понять пути выполнения сложной программы. Вместо того чтобы вставлять отладочный код внутрь программы, вы включаете или выключаете трассировку в том месте где нужно, а потом используете утилиты подобные `grep` или собственно написанные приложения на PHP для анализа лог файла.

В данной статье мы рассмотрим профайлинг. Профайлинг с первого взгляда похож на трассировку. Профайлинг-лог не предназначен для людей, не предназначен для визуализации потока выполнения программы, однако он обеспечивает нас данными для статистического анализа запущенной программы.

### Создания профайлинг-лога

Ниже короткая выдержка из профайлинг-лога, созданного xdebug:

```
fl=php:internal
fn=php::define
106 3

fl=C:\www\drupal\includes\bootstrap.inc
fn=require_once::C:\www\drupal\includes\bootstrap.inc
1 648
cfn=php::define
calls=1 0 0
13 6
cfn=php::define
calls=1 0 0
18 4
cfn=php::define
calls=1 0 0
23 2
```

Как вы видите, профайлинг-лог не возможно читать напрямую. Мы будем использовать дополнительные инструменты для визуализации и анализа полученных данных. Итак, профайлинг показывает как много раз запущена была та или иная строка и сколько времени занял запуск.

Создание профайлинг-лога сильно ухудшает производительность, подобно созданию лог трассировки, потому что надо описать прохождение каждой строчки. Поэтому также как и в случае трассировки, не запускаете профайлинг на боевых серверах...

Однако существуют случаи, когда профайлинг необходимо запустить на live-системе. В этом случае будьте осторожны в одновременном запуске xdebug с другими расширениями Zend, такими как загрузчики, оптимизаторы или кэши.

Для того, чтобы xdebug начал записывать профайлинг-лог добавьте

```
| xdebug.profiler_enable=On
```

Пожалуйста, отметьте, что вы не можете запустить профайлинг во время запуска путем запуска команды.

Так как профайлинг-лог предназначен для чтения программами-анализаторами, не существует дополнительных настроек, которые позволяют отображать дополнительную информацию, как в случае лог трассировки. Однако есть некоторые настройки, которые позволяют настроить профайлинг, похожие на те, которые мы использовали при настройке трассировки. Во-первых, xdebug по умолчанию пишет профайлинг-лог в каталог `/tmp`. Если вы используете Windows, необходимо исправить `php.ini`, например так:

```
| xdebug.profiler_output_dir=c:\traces»
```

По умолчанию xdebug перезаписывает существующий профайлинг-лог. Вы можете настроить, чтобы он дополнял существующий, для чего добавьте следующую команду

```
| xdebug.profiler_append=On
```

в `php.ini`. Есть такие случаи, когда вы не хотите создавать профайлинг-лог для всех файлов, в тоже время активация профайлинга во время выполнения проблематична. Вместо периодического включения и выключения профайлинга, добавьте команду

```
| xdebug.profiler_enable_trigger=On
```

в php.ini. Теперь вы можете включать и выключать профайлинг, передавая специальный GET или POST параметр XDEBUG\_PROFILE в PHP-скрипт. Это включит профайлинг только для данного PHP-скрипта. Необязательно устанавливать значение этого параметра, не забудьте только добавить этот параметр в адрес test.php?XDEBUG\_PROFILE.

### Название профайлинг-лога

Имя, которое по умолчанию xdebug присваивает профайлинг-логу «cachegrind.out.» плюс идентификатор процесса. Также как и в случае лога трассировки можно изменить названия лога, добавляя в php.ini соответствующие настройки. Название параметра xdebug.profiler\_output\_name. Аргументом является строка, которая может содержать различные модификаторы. Самые важные ниже:

- %p – идентификатор процесса
- %r – случайное число
- %u – время
- %H – значение \$\_SERVER['HTTP\_HOST']
- %R – значение \$\_SERVER['REQUEST\_URI']
- %s – имя, включающее полный путь, слэши конвертируются в знаки подчеркивания

Пожалуйста, отметьте, что модификатор %s используется только для xdebug.profiler\_output\_name. Если вы хотите узнать название профайлинг-лога, вы можете вызвать функцию xdebug\_get\_profiler\_filename().

### Анализ профайлинг-лога

Как уже говорилось выше, для анализа профайлинг-лога необходимы дополнительные программы для визуализации данных. Все профайлинг-логи, которые создает xdebug, находятся в формате похожем на Cachegrind-формат. Cachegrind это профайлер, который является частью более мощной программы под названием [Valgrind](#), программы для отладки и профайлинга программного обеспечения для Linux. Cachegrind был предназначен для анализа статистик кэшей, использования памяти и команд программы. Другой инструмент программы Valgrind, Callgrind рисует графы вызовов. В отношении PHP, мы можем использовать это приложение для визуализации и анализа профайлинг-лога.

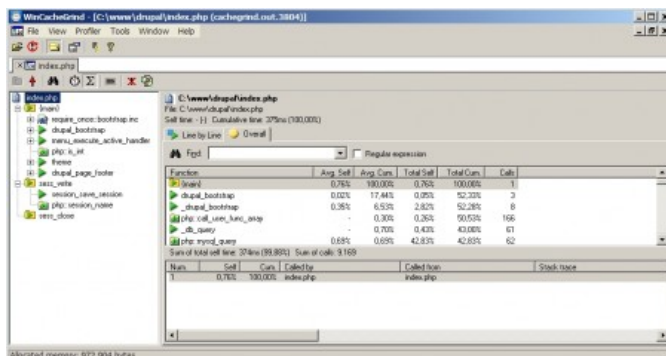
Инструмент, который обычно используется для анализа профайлинг-лога, созданного xdebug, называется [KCacheGrind](#). KCacheGrind – это свободное программное обеспечение доступное по лицензии GPL (работает только на Unix-системах). Однако, есть простенькая программка и для Windows – [WinCacheGrind](#), которая также бесплатна. Давайте рассмотрим сначала Windows-версию.

#### WinCacheGrind: анализ профайлинг-логов в Windows

Текущая версия (на момент написания автором данной статьи) WinCacheGrind – 1.0.0.12. Эта версия датирована далеким 2005, что говорит о том, что WinCacheGrind давно не разрабатывается. Если посмотреть на примечания к релизу release notes, авторы пишут, что в программе есть ошибки, которые иногда делают ее поведение странным.

Поэтому я рекомендую использовать KCacheGrind, запущенной на основе виртуальной машины на последнем дистрибутиве Линукса, например Ubuntu (примечание переводчика, вообще говоря странная рекомендация, я бы рекомендовал в этом случае просто поставить линукс, а не городить огород виртуальных машин). Существует огромное количество виртуальных машин, доступных под Windows. Если не возможно использовать Unix или виртуальную машину по каким-либо причинам, вы можете продолжать использовать WinCacheGrind для простого анализа профайлинг-лога. WinCacheGrind не рисует графы вызовов в отличие от KCacheGrind.

Установка Wincachegrind чрезвычайно проста. Запустите установщик, нажмите на кнопку согласия с лицензией и установка завершена. Теперь вы можете запустить программу, открыть в ней один из cachegrind профайлинг-логов, созданных xdebug.



Кликав на часы или иконку сигмы, вы можете переключаться между выводом информации в абсолютных значениях и процентах. Процентное отображение показывает, сколько времени в процентах от общего времени занимает вызов функции в данном блоке.

Две полезные настройки Profiler -> Hide Fast Functions и Profiler -> Hide Library Functions. Первый переключатель скрывает функции, временной вклад которых в общее время выполнения программы незначителен.

Вторая настройка, Profiler -> Hide Library Functions скрывает из общего анализа встроенные в PHP функции. Когда включены обе эти настройки, вы видите меньше данных, вследствие чего можно сфокусироваться на тех участках кода, которые требуют оптимизации.

Главное окно содержит две вкладки: Line by line и Overall. Обе вкладки показывают одинаковую информацию, однако вкладка

Overall агрегирует информацию для лучшего представления. Self time отображает время запуска кода в текущем блоке, в то время как Cumulative time (Cum.) показывает общее время запуска функций в данном блоке.

### KCacheGrind: анализ профайлинг-логов в Unix

Unix версия KCacheGrind предоставляет больше функциональности, чем WinCacheGrind. KCacheGrind визуализирует данные и строит граф вызовов.

Для начала использования необходимо установить KCacheGrind. Текущая версия [0.4.6](#). Более новая версия (0.10.1) доступна, однако она является частью пакета Valgrind.

Если возможно используйте менеджер пакетов для установки пакета KCacheGrind. KCacheGrind использует GraphViz для рисования графов вызова, поэтому необходимо также установить пакет GraphViz, если ваш менеджер пакетов автоматически не устанавливает зависимые пакеты.

Если вы не нашли бинарный пакет KCacheGrind, необходимо скомпилировать KCacheGrind самостоятельно. После загрузки исходников, запустите

```
./configure --prefix=/opt/kde3
make
make install
```

Как вы можете отметить, необходимо прописать путь к текущей установке KDE-библиотеки. Если вы не знаете, где в вашей системе находятся библиотеки KDE, используйте

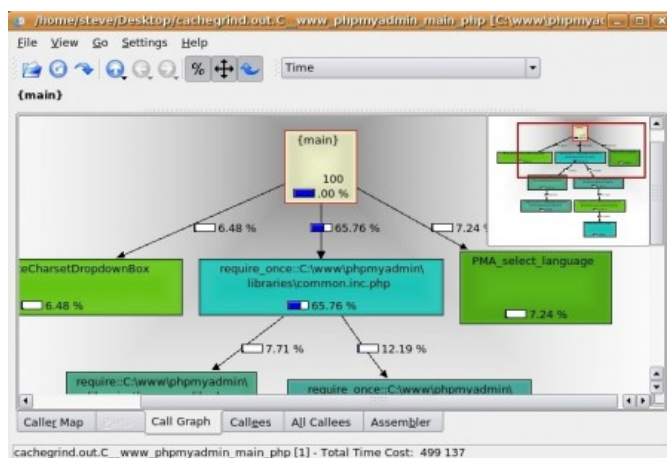
```
kde-config --prefix
```

для отображения пути к библиотекам KDE.

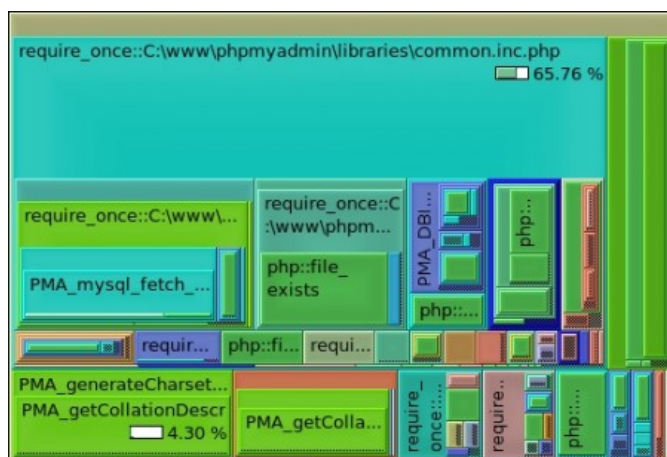
После установки, вы можете запустить KCacheGrind из командной строки

```
opt/kde3/bin/kcachegrind &
```

Табличное отображение данных в KCacheGrind очень похоже на WinCacheGrind. Вы также можете переключаться между абсолютными и процентными значениями. Некоторые возможности KCacheGrind не предназначены для PHP. На картинке внизу показан граф вызовов программы phpMyAdmin:



Как вы можете видеть, большая часть времени запуска прошла внутри common.inc.php. Следующий скриншот показывает визуализацию вызовов функций внутри common.inc.php:



В этом блоке кода запускаются два require\_once, которые составляют половину времени запуска common.inc.php. Кликнув дважды на любой прямоугольник, вы опуститесь глубже в анализ данных.

## Оптимизация кода на основании данных профайлинга

Всегда профилируйте ваши приложения до начала оптимизации. Вы можете сами начать оптимизацию, в том месте, где вам кажется, что эта оптимизация принесет эффект, однако это не всегда верно. Оптимизация, главным образом, имеет эффект лишь в тех частях, которые занимают, больше всего времени в процессе выполнения.

Если запускается множество копий программы одновременно, все равно может возникнуть необходимость оптимизации той части вашей программы, которая занимает большую часть времени выполнения. В этом случае оптимизация не сделает обслуживание одного индивидуального запроса быстрее, но позволит вашему серверу выдерживать высокую нагрузку, потребляя меньше ресурсов для обслуживания подобных запросов.

Когда вы смотрите на продолжительность запуска по данным профайлера, имейте в виду, что абсолютные значения менее важны, чем относительные. Измеренные на разных системах, абсолютные значения могут быть различными. Однако до того как приступить к оптимизации кода, рассмотрите следующие вещи.

Важное правило в оптимизации – сокращение количества операций ввода/вывода. Некоторые операции ввода/вывода требуют очень много времени по сравнению с вычислениями. Уменьшение таких операций может быть очень эффективным путем ускорения вашей программы. Удаление одного вызова I/O может дать более эффективное улучшение, чем куча часов оптимизации кода. Поэтому вы должны сфокусироваться вначале на операциях I/O до того как вы приступите к коду.

Также перед оптимизацией вы можете увеличить количество ваших серверов. Вы можете купить огромный, что позволит вам ненамного увеличить производительность. Время разработки более дорогое, чем цена нового сервера. И если вы увеличите количество железа, вы можете быть уверены, что вы получите увеличение сразу же без какого-либо воздействия на PHP код. Когда разработчик проводит один или два дня над оптимизацией кода, вы никогда не скажете, на сколько увеличится производительность. И в конце концов, вы уже не можете быть уверены в том, что оптимизация не принесет никаких ошибок.

Преобразование некоторых страниц в статические – это один из путей достичь большей производительности. Допустим, есть сайт с большим трафиком, где PHP скрипт на каждый запрос создает первую страницу, выбирая информацию из базы данных или XML-файла. Если данные на странице изменяются достаточно часто, то вы можете пересоздавать ее статическую копию. Если преобразование в статический вид для страницы не возможно (на странице выводится какая-то персональная информация), вы можете преобразовывать в статику некоторые блоки.


Другой уровень оптимизации не требует изменения кода PHP. Как мы знаем PHP это интерпретируемый язык. Это значит, что его команды транслируются во время выполнения в промежуточный код. Трансляция повторяется каждый раз, когда запускается скрипт. Это делает PHP медленнее по сравнению с такими языками как C или Java, которые не требуют разбора кода каждый раз при запуске. Для PHP, вы можете использовать кэши промежуточного представления (смотри мой перевод .... ) для сохранения и повторного использования промежуточного кода, это делает запуск и выполнение быстрее.

Все это не говорит о том, что не время и не место оптимизировать PHP-код. Некоторые оптимизации кода могут очень сильно увеличить производительность. Однако всегда помните, что изменение кода всегда несет риск внесения дополнительных ошибок и проблем безопасности. Также не забывайте, что оптимизация кода делает его менее читаемым.

## Заключение

Создание и визуализация профайлинг-лога это одна из важных условий для оптимизации PHP-кода. Вы должны знать, какие места в программе требуют больше всего времени, и именно там начать оптимизацию.

В следующей статье мы рассмотрим отладку, используя xdebug. xdebug может предоставить вам возможность для удаленной отладки. Используя клиент, в котором реализована такая возможность, например Eclipse PDT, вы можете производить отладку вашего кода, не изменяя его, устанавливая breakpoints, перескакивать через участки кода, смотреть, как и где переменные изменяют значения.

 [Profiling, Xdebug](#)

 +23 

 9310  229  [Stefan Priebisch](#)  [great\\_boba](#) 34,1


## комментарии (16) ☐ отслеживать новые: ☐ в почте ☐ в трекере

 [Snart](#), 17 марта 2008 в 17:32 #  0  


Супер! В избранное.

+99  [evilbloodydemon](#), 17 марта 2008 в 17:42 #  +1  

для визуализации есть еще [CachegrindVisualizer](#)

 [Devgru](#), 17 марта 2008 в 17:54 #    0  

классно, спасибо

 [red\\_pilot](#), 17 марта 2008 в 18:04 #    0  

WinCacheGrind - не адекватная программа. Нулевое узабилити и очень глючная.  
CachegrindVisualizer - выглядит неплохо, сейчас буду тестировать.

 [betttrrr](#), 17 марта 2008 в 18:37 #    0  

Да ужжжж! Установка CachegrindVisualizer достаточно геморройная, т.к. надо ставить AIR, CachegrindVisualizer, Graphviz, ZGRViewer. И для получения результата надо поочерёдно запускать CachegrindVisualizer и ZGRViewer (java, запуск через run.bat), но в результате получается график, который мне кажется более наглядным, чем WinCacheGrind.



**Develar**, 17 марта 2008 в 21:02 # ☆ h ↑

0 ↑ ↓

WinCacheGrind использовать нельзя, или сами делите/опирайтесь на относительность — <http://www.xdebug.org/archives/xdebug-de...>  
А CachegrindVisualizer — философия unix. Времени на интеграцию, реализацию Roamer и DataGrid пока нет — и мне особо не нужно, но народ да, требует <http://phpclub.ru/talk/showthread.php?po...>.



**Devgru**, 17 марта 2008 в 18:42 # ☆ h ↑

0 ↑ ↓

Я не знаю, не могу сказать это про неё. Юзабилити выше среднего, всё в принципе понятно, ИМХО. Но CV поставлю тоже.



**i4ck**, 17 марта 2008 в 18:29 # ☆

0 ↑ ↓

xdebug можно включать и через .htaccess,  
не модифицируя глобальный php.ini.

```
php_value xdebug.profiler_output_dir /tmp
php_value xdebug.profiler_enable 1
```



**Angerslave**, 17 марта 2008 в 19:46 # ☆

0 ↑ ↓

Хорошая оптимизация не ухудшает читаемость кода, а улучшает, особенно если бутылочные горлышки расположены в местах с витиеватыми алгоритмами.

Сам юзаю WinCacheGrind для повседневной оптимизации и CacheGrindVisualizer для редких вылазок глобального масштаба.



**mx2000**, 17 марта 2008 в 23:43 # ☆ h ↑

0 ↑ ↓

Как раз таки оптимизация не улучшает, а ухудшает читаемость кода. Улучшает, как правило, рефакторинг.



**evqwest**, 17 марта 2008 в 23:16 # ☆

0 ↑ ↓

спасибо) освежил тему)))



**AndryX**, 18 марта 2008 в 01:44 # ☆

0 ↑ ↓

круто, автору спасибо. Ждём следующих статей.



**aaaa**, 18 марта 2008 в 16:36 # ☆

0 ↑ ↓

кг/ам \*CRAZY\*



**temaonline**, 18 октября 2008 в 11:40 # ☆

0 ↑ ↓

Реально, большое человеч. спасибо!



**vrazbros**, 22 января 2009 в 22:45 # ☆

0 ↑ ↓

сохранил на <http://www.xdebug.ru>



**serivPS**, 20 ноября 2009 в 02:24 # ☆

0 ↑ ↓

Охх... Расцаловал — бы)))

Отличная статья, отличный инструмент!

[Список скептика](#)

Управленческие инструменты:  
4-фазный алгоритм решения  
проблем с людьми или «А чего  
ты хочешь, если ты такой  
хреновый менеджер?»

250 строк кода, распознающих  
дату на русском языке

**Brainstorage**  
Все мозги в одном месте

**ТОСТЕР**  
Q&A-сервис для разработчиков

**ФРИЛАНСИМ**  
Заказы для фрилансеров

**ХАНТИМ**  
Вакансии для айтишников

**АВТОКАДАБРА**  
Уютная и дружелюбная