Profiling with XHProf

Ilia Alshanetsky
@iliaa
Me, Myself and I

PHP Core Developer

Author of
Guide to PHP Security

CIO at Centah Inc.
Why are we here?
Speed Matters!

Speed

Greater User Engagement
Speed is all about perception!
The Web Story

- 80ms for PHP
- 100ms for Browser Rendering Stuff - 800 milliseconds
- 1,000 milliseconds for:
  - 20ms Request Processing, Compression
  - DNS, SSL Negotiation, Downloading Content
Making Things Faster

=

Eliminating Bottlenecks
You find bottlenecks by Profiling!
Benchmarking Sucks!
Profiling Can Suck Too!

I RARELY PROFILE MY CODE

BUT WHEN I DO, I DO IT LOCALLY!
Profiling in Production

* Needs to be fast
* Simple to Implement
* Not introduce “breakage”
* Must be aggregatable
This is where **XHProf** Excels!
Installation

• Two Part Process
  • First install the PHP Extension
  • Then install the storage engine & results interface.
PHP Extension

- Via PECL
  
  `pecl install xhprof`

- Via Package Manager
  
  `apt-get install php5-xhprof`

- Enable inside php.ini
  
  `extension=xhprof.so`
XHProf Gui

• For MySQL backend
  • [https://github.com/preinheimer/xhprof](https://github.com/preinheimer/xhprof)

• For MongoDB backend
  • [https://github.com/perftools/xhgui](https://github.com/perftools/xhgui)
class MagicProfiler {
    static public $profiling = 0;

    public function __construct() {
        self::$profiling = !(mt_rand() % 9);
        if (self::$profiling) {
            xhprof_enable(XHPROF_FLAGS_CPU | XHPROF_FLAGS_MEMORY);
        }
    }

    public function __destruct() {
        if (self::$profiling) {
            $data = xhprof_disable();
            include '/path/to/xhprof/xhprof_lib/config.php';
            $GLOBALS['_xhprof'] = $_xhprof;
            include '/path/to/xhprof/xhprof_lib/utils/xhprof_lib.php';
            include '/path/to/xhprof/xhprof_lib/utils/xhprof_runs.php';
            $x = new XHPProfRuns_Default();
            $x->save_run($data, 'App Name', null, $_xhprof);
        }
    }
}

$me = new MagicProfiler();
$_xhprof['dbtype'] = 'mysql'; // Only relevant for PDO
$_xhprof['dbhost'] = 'localhost';
$_xhprof['dbuser'] = 'root';
$_xhprof['dbpass'] = 'root';
$_xhprof['dbname'] = 'profile';
$_xhprof['dbadapter'] = 'Pdo'; // Or mysql or mysqli
$_xhprof['namespace'] = 'myapp'; // could be different per-app

// I recommend using igbinary
$_xhprof['serializer'] = 'php';

// Ignore URLs containing the following keywords
$ignoreURLs = array('/images/');

// Ignore requests for the domains with following keywords
$ignoreDomains = array('static.');

// Don't log POST input for URLs containing the following
$exceptionPostURLs = array('login','payment');
$controlIPs = false; // Disabled by default
// this means everyone's got access

// Otherwise list IPs that can manually
// trigger profiling via GET _profile=1
$controlIPs = array();
// both IPv4 and IPv6 values are allowed
$controlIPs[] = '127.0.0.1';
Interpreting the Results

![Cartoon by xkcd](https://xkcd.com/)

- Intelligence increases as human proximity to a cat decreases.
- The inanity of statements increases as human proximity to a cat increases.
- The cartoon humorously suggests that as one gets closer to a cat, one is more likely to be called a 'kitty'.

xkcd
# General Summaries

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Cpu</th>
<th>Wall Time</th>
<th>Peak Memory Usage</th>
<th>URL</th>
<th>Simplified URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 17 11:41:25</td>
<td>6381030</td>
<td>8627592</td>
<td>62294608</td>
<td>//? %5Bpage%5D=scheduler&amp; %5Bstate_id%5D=5 53.</td>
<td></td>
</tr>
<tr>
<td>Sep 22 16:27:54</td>
<td>4883258</td>
<td>5233343</td>
<td>74901608</td>
<td>//? %5Bpage%5D=scheduler&amp; %5Bstate_id%5D=0 21.</td>
<td></td>
</tr>
<tr>
<td>Sep 17 13:58:13</td>
<td>4112375</td>
<td>4338331</td>
<td>71964272</td>
<td>//? %5Bpage%5D=scheduler&amp; %5Bstate_id%5D=4 59.</td>
<td></td>
</tr>
<tr>
<td>Sep 22 16:29:17</td>
<td>3720435</td>
<td>3914633</td>
<td>42557944</td>
<td>//? %5Bpage%5D=scheduler&amp; %5Bstate_id%5D=6 13.</td>
<td></td>
</tr>
<tr>
<td>Sep 17 15:35:52</td>
<td>3572456</td>
<td>5574733</td>
<td>182382752</td>
<td>//? %5Bpage%5D=scheduler&amp; %5Bstate_id%5D=4 59.</td>
<td></td>
</tr>
</tbody>
</table>

- Hardest Hit
- Slowest
- Biggest memory hogs
- Longest Running

**Today or Week to Date**
## Trending Baseline

<table>
<thead>
<tr>
<th>Stat</th>
<th>Exact URL</th>
<th>Similar URLs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>5596</td>
<td>5596</td>
</tr>
<tr>
<td>Min Wall Time</td>
<td>20.1520 ms</td>
<td>20.1520 ms</td>
</tr>
<tr>
<td>Max Wall Time</td>
<td>2.4764 minutes!</td>
<td>2.4764 minutes!</td>
</tr>
<tr>
<td>Avg Wall Time</td>
<td>686.5729 ms</td>
<td>686.5729 ms</td>
</tr>
<tr>
<td>95% Wall Time</td>
<td>2.0456 s</td>
<td>2.0456 s</td>
</tr>
<tr>
<td><strong>Display run Incl. Wall Time (microsec)</strong></td>
<td>1,006,332 microsecs</td>
<td>1,006,332 microsecs</td>
</tr>
<tr>
<td>Min CPU Ticks</td>
<td>16.9970 ms</td>
<td>16.9970 ms</td>
</tr>
<tr>
<td>Max CPU Ticks</td>
<td>2.4066 s</td>
<td>2.4066 s</td>
</tr>
<tr>
<td>Avg CPU Ticks</td>
<td>385.5804 ms</td>
<td>385.5804 ms</td>
</tr>
<tr>
<td>95% CPU Ticks</td>
<td>1.0678 s</td>
<td>1.0678 s</td>
</tr>
<tr>
<td><strong>Display run Incl. CPU (microsecs)</strong></td>
<td>923,859 microsecs</td>
<td>923,859 microsecs</td>
</tr>
<tr>
<td>Min Peak Memory Usage</td>
<td>6,006,952 bytes</td>
<td>6,006,952 bytes</td>
</tr>
<tr>
<td>Max Peak Memory Usage</td>
<td>309,707,216 bytes</td>
<td>309,707,216 bytes</td>
</tr>
<tr>
<td>Avg Peak Memory Usage</td>
<td>72,154,616 bytes</td>
<td>72,154,616 bytes</td>
</tr>
<tr>
<td>95% Peak Memory Usage</td>
<td>208,371,968 bytes</td>
<td>208,371,968 bytes</td>
</tr>
<tr>
<td><strong>Display run Incl. PeakMemUse (bytes)</strong></td>
<td>168,900,960 bytes</td>
<td>168,900,960 bytes</td>
</tr>
<tr>
<td><strong>Number of Function Calls:</strong></td>
<td></td>
<td>10,123</td>
</tr>
</tbody>
</table>
Trending

Runs with Simplified URL: /?_[page]=reports
## Specific Run Comparison

http://xhprof/?run1=RUN_ID&run2=RUN_ID

<table>
<thead>
<tr>
<th></th>
<th>Run One ID: 5419abae5c36b</th>
<th>Run Two ID: 5420864fef1ba</th>
<th>Diff</th>
<th>Diff%</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of Function Calls</strong></td>
<td>1,334,557</td>
<td>170,517</td>
<td>-1,164,040</td>
<td>-87.2%</td>
</tr>
<tr>
<td><strong>Incl. Wall Time (microsec)</strong></td>
<td>8,627,592</td>
<td>5,233,343</td>
<td>-3,394,249</td>
<td>-39.3%</td>
</tr>
<tr>
<td><strong>Incl. CPU (microsecs)</strong></td>
<td>6,381,030</td>
<td>4,883,258</td>
<td>-1,497,772</td>
<td>-23.5%</td>
</tr>
<tr>
<td><strong>Incl. MemUse (bytes)</strong></td>
<td>53,870,696</td>
<td>67,092,616</td>
<td>13,221,920</td>
<td>24.5%</td>
</tr>
<tr>
<td><strong>Incl. PeakMemUse (bytes)</strong></td>
<td>62,294,608</td>
<td>74,901,608</td>
<td>12,607,000</td>
<td>20.2%</td>
</tr>
</tbody>
</table>
In-Depth Specifics

Expensive Calls by Exclusive Wall Time

- Loading: 462023ms
- Other
- CoreCache::cache_fetch
- Other
## Really In-Depth

<table>
<thead>
<tr>
<th>Function</th>
<th>Call Count</th>
<th>Wall Time</th>
<th>CPU</th>
<th>Memory Usage</th>
<th>Peak Memory Usage</th>
<th>Exclusive Wall Time</th>
<th>Exclusive CPU</th>
<th>Exclusive Memory Usage</th>
<th>Exclusiv Peak Memory Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>main()</code></td>
<td>1</td>
<td>8627592</td>
<td>6381030</td>
<td>53870696</td>
<td>62294608</td>
<td>668</td>
<td>1000</td>
<td>-98416</td>
<td>552</td>
</tr>
<tr>
<td>CoreBaseController::run</td>
<td>1</td>
<td>8528193</td>
<td>6282045</td>
<td>38455144</td>
<td>46846056</td>
<td>85</td>
<td>0</td>
<td>3040</td>
<td>0</td>
</tr>
<tr>
<td>CoreComponentController::process</td>
<td>1</td>
<td>8522065</td>
<td>6277046</td>
<td>36747664</td>
<td>44988216</td>
<td>158</td>
<td>0</td>
<td>6840</td>
<td>0</td>
</tr>
<tr>
<td>CoreComponent::dispatchEvent</td>
<td>3</td>
<td>8202059</td>
<td>6096073</td>
<td>17628888</td>
<td>26802216</td>
<td>54</td>
<td>0</td>
<td>2848</td>
<td>0</td>
</tr>
<tr>
<td>CoreEventDispatcher::sendEvent</td>
<td>2</td>
<td>8201893</td>
<td>6096073</td>
<td>17621056</td>
<td>26802216</td>
<td>27</td>
<td>0</td>
<td>-2008</td>
<td>0</td>
</tr>
<tr>
<td>CoreEventDispatcher::sendEvent</td>
<td>1</td>
<td>8201852</td>
<td>6096073</td>
<td>17621336</td>
<td>26802216</td>
<td>22</td>
<td>0</td>
<td>1592</td>
<td>0</td>
</tr>
<tr>
<td>CoreComponentFactory::dispatch</td>
<td>1</td>
<td>8201740</td>
<td>6096073</td>
<td>17613584</td>
<td>26802216</td>
<td>27</td>
<td>0</td>
<td>1728</td>
<td>0</td>
</tr>
<tr>
<td>SchedulerPageComponent::onSubmit</td>
<td>1</td>
<td>8201644</td>
<td>6096073</td>
<td>17607464</td>
<td>26802216</td>
<td>23298</td>
<td>15995</td>
<td>-842344</td>
<td>0</td>
</tr>
<tr>
<td>CoreModel::insert</td>
<td>4</td>
<td>8102264</td>
<td>6019087</td>
<td>14610480</td>
<td>22661424</td>
<td>28299</td>
<td>14999</td>
<td>-15023888</td>
<td>192</td>
</tr>
<tr>
<td>CoreModel::createPlan</td>
<td>3137</td>
<td>4859440</td>
<td>3539457</td>
<td>22594160</td>
<td>22167056</td>
<td>1493159</td>
<td>863857</td>
<td>-22767328</td>
<td>7288</td>
</tr>
<tr>
<td>CoreModel::executePlan</td>
<td>3137</td>
<td>3169555</td>
<td>2422637</td>
<td>3377328</td>
<td>0</td>
<td>104107</td>
<td>27996</td>
<td>-13932208</td>
<td>0</td>
</tr>
</tbody>
</table>
Graphical Trace

Requires Graphviz package

apt-get install graphviz
Profiling Tricks
Only Examine User Functions

```php
xhprof_enable(
    XHPROF_FLAGS_NO_BUILTINS | XHPROF_FLAGS_CPU | XHPROF_FLAGS_MEMORY
);
```
Ignore Functions

```php
xhprof_enable(
    // usual flags,
    array('ignored_functions' => array(
        'exec',
        'ClassName::__destruct'
    ))
);
```
Lightweight Profiling

// use
xhprof_sample_enable();

// instead of xhprof_enable();
// termination of profiling is done via
xhprof_sample_disable();

** Does not support flags or function ignore lists**
User Tracing

```
session_start();

if (empty($_SESSION['user_id'])) {
    $_SESSION['user_id'] = init_user();

    if (!$($_SESSION['user_id'] % 10)) {
        $_SESSION['profile'] = 1;
    }
}
```

** Avoid using Cookies or GET/POST params for triggering profiling**
Database Maintenance

Profiling DBs grow quickly, so clean-up data frequently if you don’t have a lot of storage place…

42k profiled pages, take 500+ MB in MySQL
Alternative UI to XHGUI

A nicer UI, simpler code, but appears to be abandoned for >1 year now :-(

Commercial Solution also available via Tideways profiler**
Honorary Mentions ;-) 

BlackFire from Sensio Labs

AppDynamics

New Relic
THANK YOU FOR LISTENING

Slides: http://ilia.ws

@iliaa