

# Speeding up PHP applications

OSCOM 4

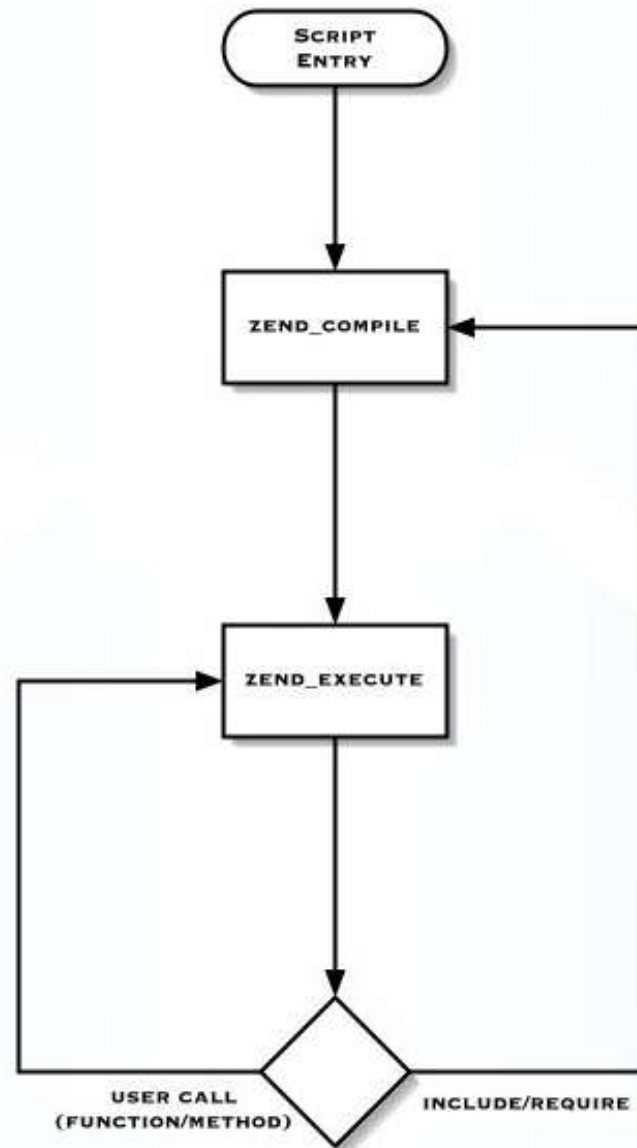
October 1st, 2004. Zürich, Switzerland

Derick Rethans <[derick@php.net](mailto:derick@php.net)>

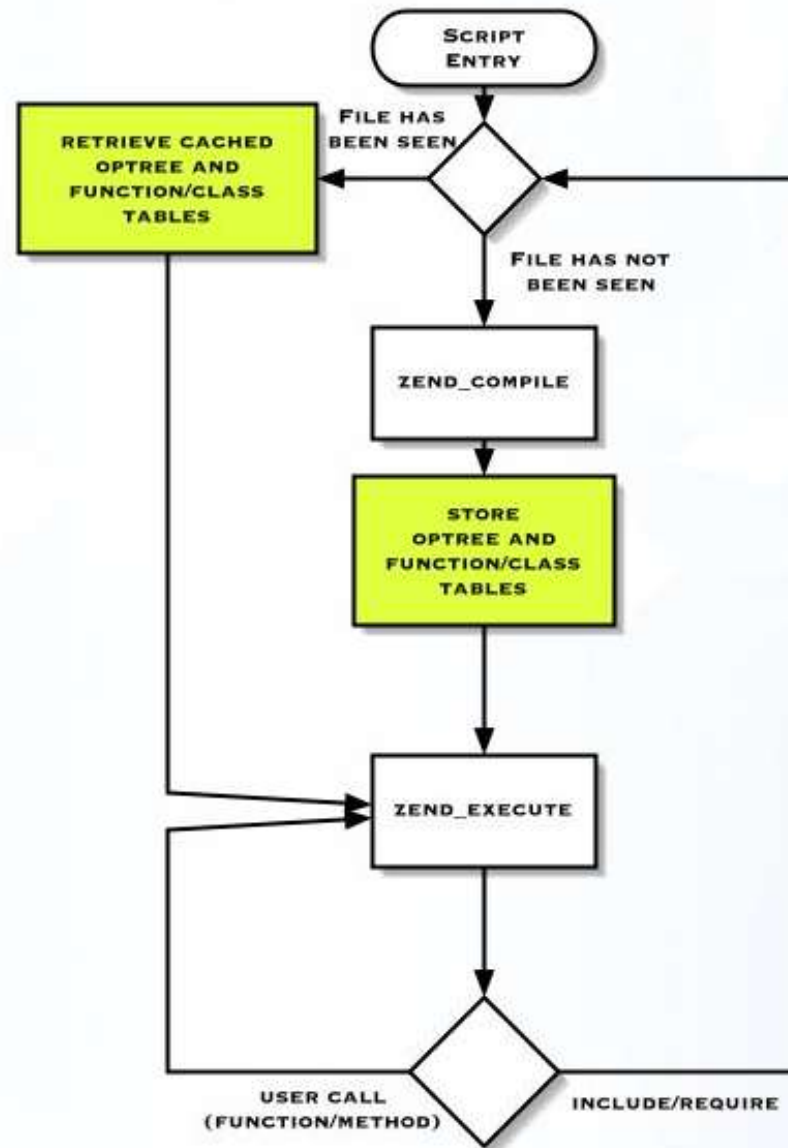
Share your information



- . Tuning the Webserver
- . Caching
- . Optimizing SQL Queries
- . Optimizing PHP Code
- . Reducing Bandwidth



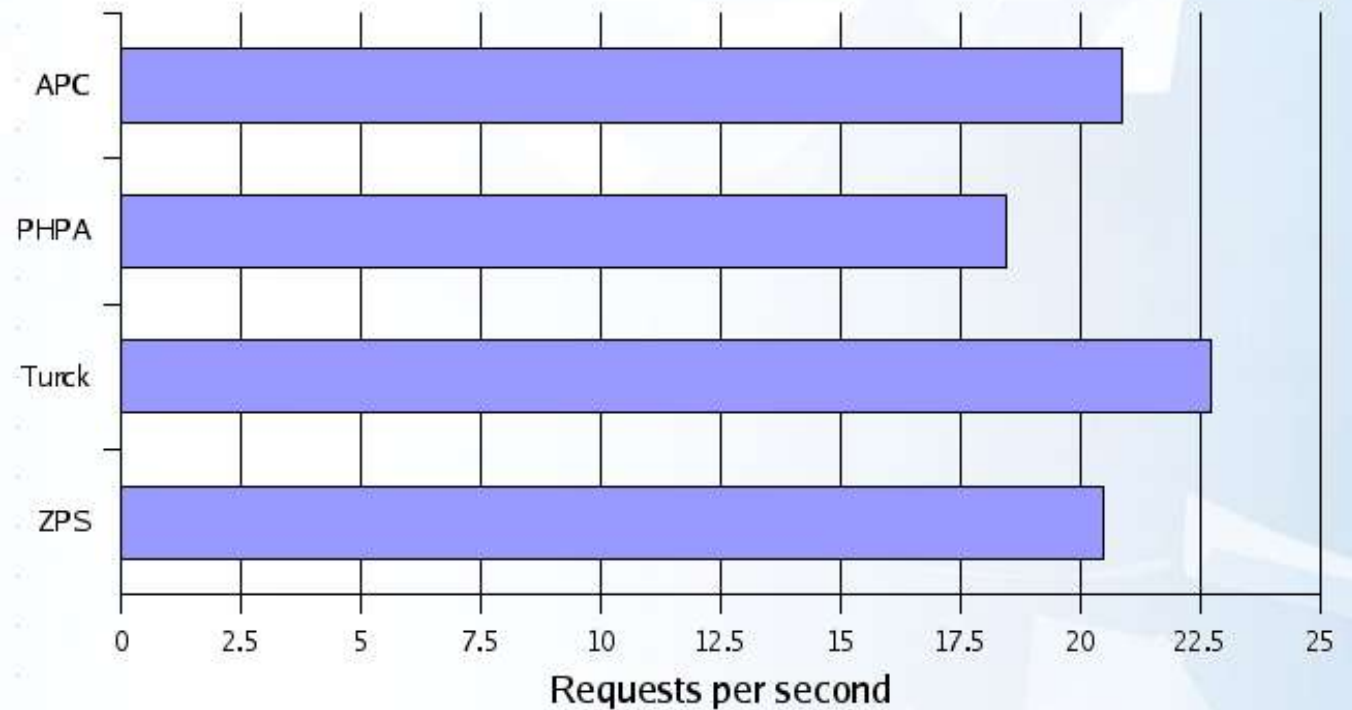
- This cycle happens for every include file, not just for the "main" script
- Compilation can easily consume more time than execution



- In general, each source file is compiled once
- Compilation overhead becomes inconsequential
- Cache introduces its own overhead due to dynamic nature of includes

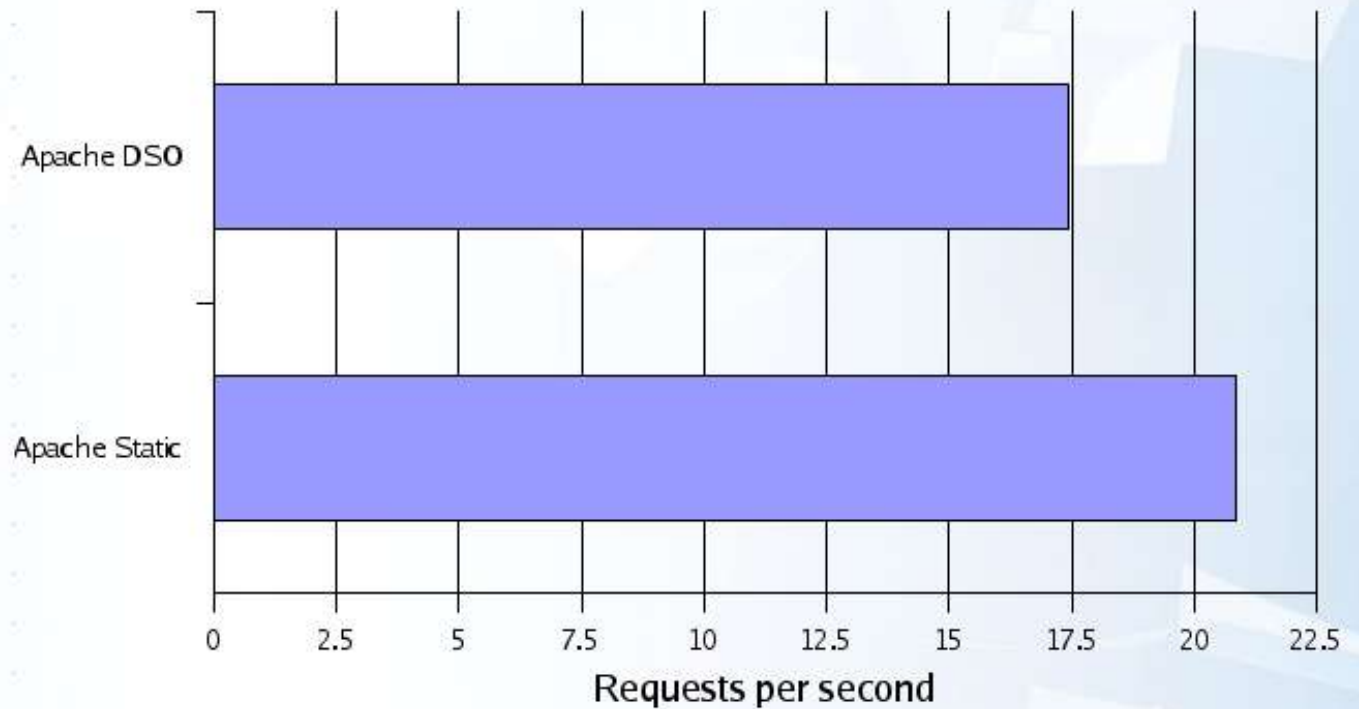
- APC
- Zend Performance Suite
- Turck MM Cache
- PHP Accelerator

Different Caches



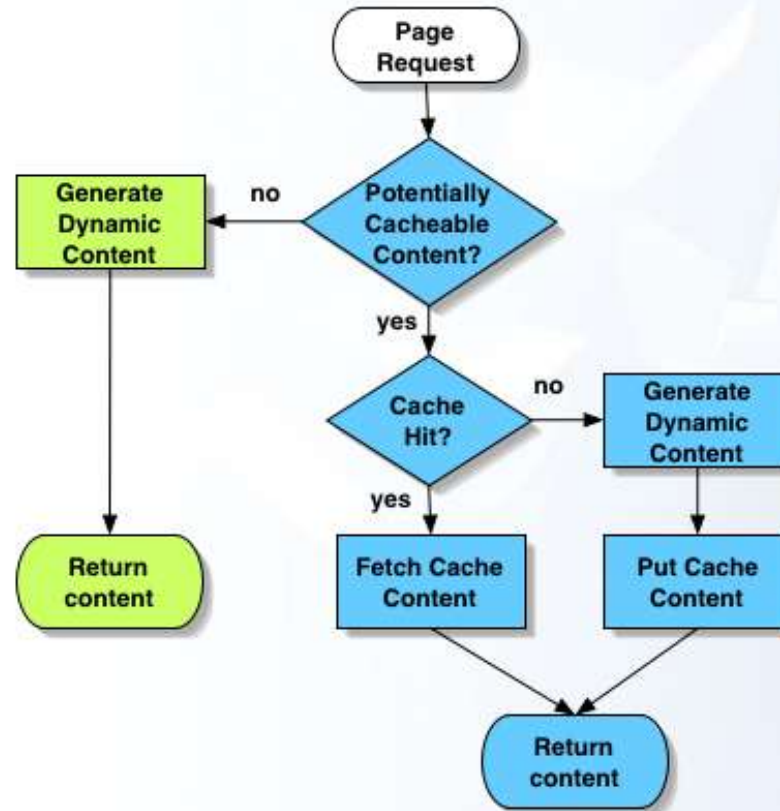
- When you're using PHP as DSO module
- So use static compiled modules
- or compile with `--prefer-non-pic`

### DSO vs. Static



- .Optimized ./configure
- .Memory Limit
- .Shorter Path
- .Use *include()* in a smarter way
- .Ramdisk

**Caching is the recognition and exploitation of the fact that most "dynamic" data does not change every time you request it.**



## Caching Example

*PEAR::Cache\_Lite*

```
<?php
/* Include the class */
require_once('Cache/Lite.php');

/* Set a key for this cache item */
$id = 'newsitem1';

/* Set a few options */
$options = array(
    'cacheDir' => '/var/cache/news/',
    'lifeTime' => 3600
);

/* Create a Cache_Lite object */
$Cache_Lite = new Cache_Lite($options);

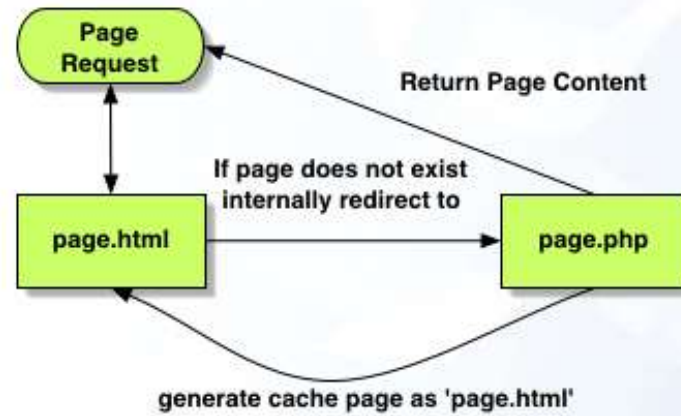
/* Test if there is a valid cache-entry for this key */
if ($data = $Cache_Lite->get($id)) {
    /* Cache hit! */
} else {
    /* Cache miss! */
    ob_start();
    /* Generate content */
    $data = ob_get_contents();
    $Cache_Lite->save($data);
}
?>
```

### Pros:

- Potentially Immense Speed-Up
- Decreased Resource Consumption

### Cons:

- Increase in Architectural Complexity
- Potential for Stale or Inconsistent Data



### Set up a 404 error handler in .htaccess:

```
RewriteEngine on
RewriteRule /**\.[^h][^t][^m][^l]$ /$1.html
ErrorDocument 404 /index.php
DirectoryIndex index.php
```

### index.php:

```
<?php
if (!empty($_SERVER['REDIRECT_URL'])) {
    // This is the requested page that caused the error
    $current_page = substr($_SERVER['REDIRECT_URL'], strlen(WEBBASE));
}

...

if (!FORCE_DYNAMIC) {
    $contents = ob_get_contents();
    ob_clean();
    $f = fopen($lang . "/" . $current_page . ".html", 'w');
    fwrite($f, $contents);
    fclose($f);
    echo $contents;
}
?>
```

- .Reverse Proxy
- .Tied into Apache
- .Static Generated
- .Huge Performance Boost
- .No 'Dynamic' Content



## Some effort to use with PHP:

```
<?php
$timestamp = /* timestamp for last updated part */;
$tsstring = gmdate("D, d M Y H:i:s ", $timestamp)."GMT";

if ($_SERVER["HTTP_IF_MODIFIED_SINCE"] == $tsstring) {
    /* The UA has the exact same page we have. */
    header("HTTP/1.1 304 Not Modified");
    exit();
} else {
    header("Last-Modified: ".$tsstring);
}
?>
```

## Set Cache-Control headers so external caches know how to respect your content:

```
<?php
header("Last-Modified: Mon, 13 Sep 2004 09:44:13 GMT");
header("Expires: Mon, 13 Sep 2004 10:44:13 GMT");
header("Cache-Control: must-revalidate, max-age=15, s-maxage=0, private");
?>
```

- Avoid Using frequent array accesses, remember, each array access is a hash lookup.

```
<?php
$ar = array();
$ar['name'] = 'Optimization';

for ($i = 0; $i < count($entries); $i++) {
    print $ar['name'];
}
?>
```

- Shorten "required" executions, such as loops.
- Avoid rebuilding large data structures.
- Look for alternatives to regular expressions

- Xdebug: a tool to debug PHP
- Tracing function calls
- Profiling Applications



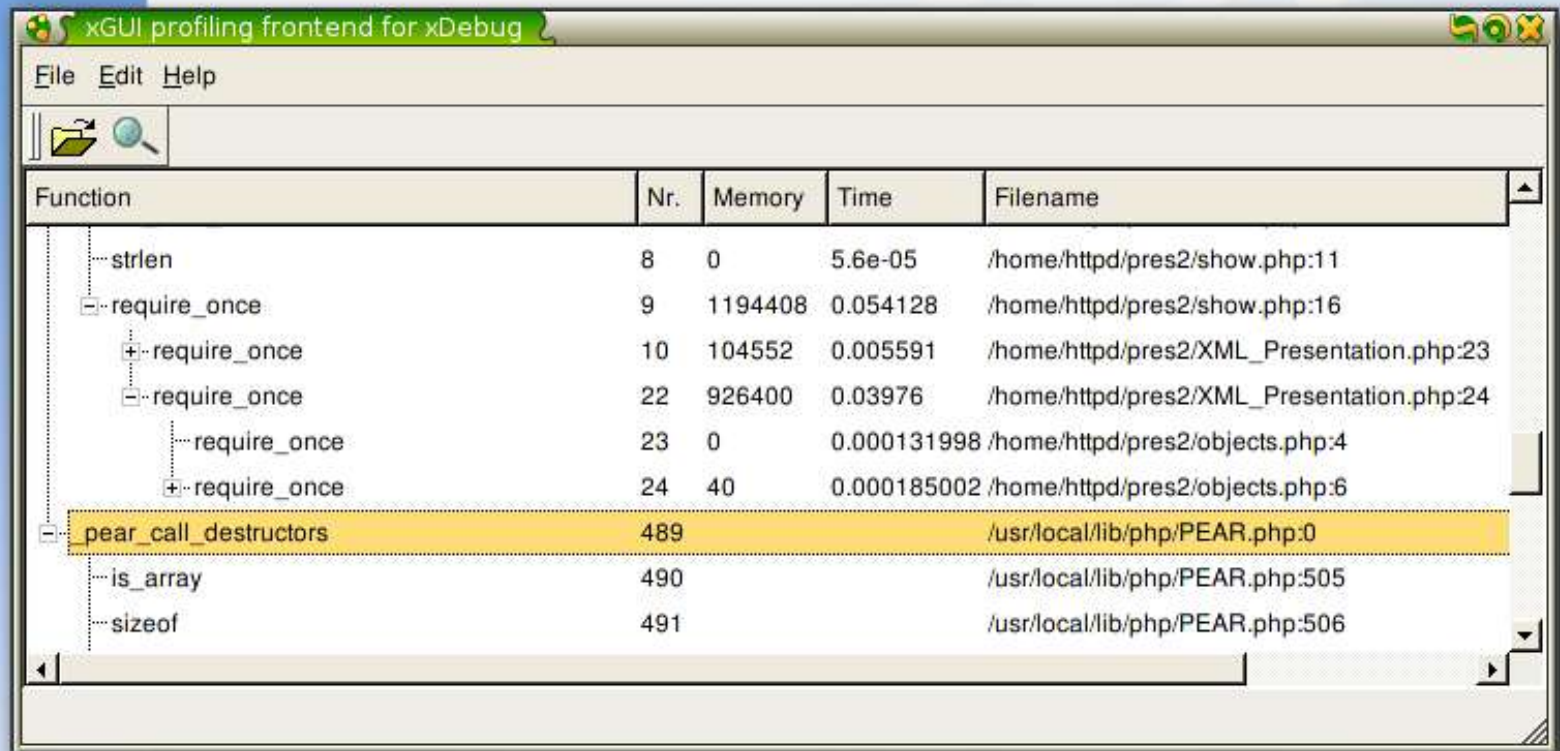
- Xdebug's Trace Features
- Trace function calls
- Provides details on memory usage

```

[derick@kossu] - mc - /dat/ez.no-33/var/ezno
TRACE START [2004-06-03 13:52:21]
0.0032 235760 -> {main}() /dat/ez.no-33/index.php:0
0.0036 235840 -> microtime() /dat/ez.no-33/index.php:34
0.0037 235856 -> ob_start() /dat/ez.no-33/index.php:35
0.0037 278880 -> error_reporting(2047) /dat/ez.no-33/index.php:74
0.0084 611512 -> include_once(/dat/ez.no-33/lib/ezutils/classes/ezdebug.php) /dat/ez.no-33/index.php:77
0.0106 766520 -> include_once(/dat/ez.no-33/lib/ezutils/classes/ezsys.php) /dat/ez.no-33/lib/ezutils/cl
0.0106 766568 -> define('EZ_SYS_DEBUG_INTERNALS', FALSE) /dat/ez.no-33/lib/ezutils/classes/ezsys.php:
0.0107 765856 -> define('EZ_LEVEL_NOTICE', 1) /dat/ez.no-33/lib/ezutils/classes/ezdebug.php:85
0.0107 765872 -> define('EZ_LEVEL_WARNING', 2) /dat/ez.no-33/lib/ezutils/classes/ezdebug.php:86
/tmp/trace.2043925204.xt [R0] 2,1 Top
0.0456 1698456 -> eztextcodec::internalcharset() /dat/ez.no-33/lib/ezil8n/classes/eztextcodec.php:581
0.0456 1698456 -> ezcharsetinfo::realcharsetcode('iso-8859-15') /dat/ez.no-33/lib/ezil8n/classes/eztex
0.0457 1698424 -> ezcharsetinfo::aliastable() /dat/ez.no-33/lib/ezil8n/classes/ezcharsetinfo.php:212
0.0457 1702584 -> is_array(array ('ascii' => 'us-ascii', 'latin1' => 'iso-8859-1', 'latin2' => 'is
0.0461 1702664 -> strtolower('iso-8859-15') /dat/ez.no-33/lib/ezil8n/classes/ezcharsetinfo.php:213
0.0500 1966816 -> include_once(/dat/ez.no-33/lib/ezlocale/classes/ezlocale.php) /dat/ez.no-33/index.php:24
0.0501 1966904 -> define('EZ_LOCALE_DEBUG_INTERNALS', FALSE) /dat/ez.no-33/lib/ezlocale/classes/ezlocale
0.0501 1965888 -> ezini::instance() /dat/ez.no-33/index.php:246
0.0502 1966224 -> get_class(class ezini { var $Charset = 'iso-8859-1'; var $BlockValues = array ('Databa
0.0510 1966840 -> ezini->variable('RegionalSettings', 'Debug') /dat/ez.no-33/index.php:247
/tmp/trace.2043925204.xt [R0] 349,5 1%

```

## ◦ Simple Tracefile Browser



The screenshot shows a window titled "xGUI profiling frontend for xDebug" with a menu bar (File, Edit, Help) and a toolbar with a folder and search icon. Below is a table with columns: Function, Nr., Memory, Time, and Filename. The table lists several function calls, with "pear\_call\_destructors" highlighted in yellow.

| Function                  | Nr. | Memory  | Time        | Filename                                  |
|---------------------------|-----|---------|-------------|---|
| strlen                    | 8   | 0       | 5.6e-05     | /home/httpd/pres2/show.php:11             |
| [-] require_once          | 9   | 1194408 | 0.054128    | /home/httpd/pres2/show.php:16             |
| + require_once            | 10  | 104552  | 0.005591    | /home/httpd/pres2/XML_Presentation.php:23 |
| [-] require_once          | 22  | 926400  | 0.03976     | /home/httpd/pres2/XML_Presentation.php:24 |
| require_once              | 23  | 0       | 0.000131998 | /home/httpd/pres2/objects.php:4           |
| + require_once            | 24  | 40      | 0.000185002 | /home/httpd/pres2/objects.php:6           |
| [-] pear_call_destructors | 489 |         |             | /usr/local/lib/php/PEAR.php:0             |
| is_array                  | 490 |         |             | /usr/local/lib/php/PEAR.php:505           |
| sizeof                    | 491 |         |             | /usr/local/lib/php/PEAR.php:506           |

- Xdebug's Profiling Features
- GUI analysing with KCachegrind
- Provides details on CPU usage

Flat Profile

| Incl.  | Self  | Called | Function                              |
|--------|-------|--------|---------------------------------------|
| 99.998 | 7.109 | (0)    | {main}                                |
| 32.480 | 2.347 | 1      | include_once:::/dat/ez.no-33/kernel/c |
| 29.447 | 1.832 | 1      | include_once:::/dat/ez.no-33/kernel/c |
| 27.226 | 0.850 | 1      | include_once:::/dat/ez.no-33/kernel/c |
| 26.349 | 0.668 | 1      | include_once:::/dat/ez.no-33/kernel/c |
| 25.660 | 0.068 | 1      | include_once:::/dat/ez.no-33/kernel/c |
| 25.584 | 0.405 | 1      | ezdatatype::loadandregisteralltypes   |
| 23.783 | 9.236 | 30     | ezdatatype::loadandregistertype       |
| 18.024 | 0.084 | 1      | eztemplate->fetch                     |
| 16.286 | 3.832 | 13     | ezini->loadcache                      |
| 15.156 | 1.281 | 143    | ezini::instance                       |
| 13.841 | 0.268 | 11     | ezini->ezini                          |
| 13.524 | 0.204 | 11     | ezini->load                           |
| 12.491 | 0.015 | 1      | eztemplate->executecompiledtempl      |
| 12.476 | 0.021 | 1      | eztemplatedesignresource->execute     |
| 12.450 | 0.051 | 1      | eztemplatecompiler::executecompila    |
| 12.206 | 1.490 | 1      | eztemplatecompiler::executecompila    |
| 10.716 | 1.660 | 1      | include:::/dat/ez.no-33/var/ezno/cach |

ezdatatype::loadandregisteralltypes

Types Callers All Callers Call Map Source

| Cost    | Count | Caller   |
|---------|-------|--|
| 100.000 | 1     | include_once:::/dat/ez.no-33/kernel/classes/ezdat... |

| Cost   | Count | Callee                          |
|--------|-------|---------------------------------|
| 92.959 | 30    | ezdatatype::loadandregistertype |
| 5.456  | 1     | ezdatatype::allowedtypes        |

Parts Calls Call Graph All Calls Caller Map Assembler

Share your information

## File I/O Problems

- Disk access is slower than memory access
- Simultaneous write/write operations cause data corruption. Therefore locking must occur—which is slow

### Tips:

- Use the disk in concentrated bits, meaning do all your writing together.
- Try and put write operations where they will cause the least loss in user experience (remember, web pages are chunked!)
- Close the session handler ASAP with `session_write_close()`

Its often beneficial to replace regular expressions with simpler functions that have the same effect.

With Regular Expression:

```
<?php
if (preg_match ('/(.*)\@(.*)/', $data, $matches)) {
    print $matches[2];
}
?>
```

Without Regular Expression:

```
<?php
$pos = strrpos ('@', $data);
if ($pos !== false) {
    print substr ($data, $pos);
}
?>
```

In other cases it may help to use regular expressions instead of custom parse routines in your PHP code.

Less output is good because...

- Saves server bandwidth
- Decreases server CPU and memory usage
- Has the same effect on the client side
- On large sites, bandwidth is most often the most costly bottleneck

*How?*

- Use preprocessors to eliminate whitespace
- If you save 20 bytes per file, and have 1,000,000 hits a day, you've just saved 20,000,000 extra bytes
- Send content "when ready", and use progressive images

- Remove All Comments!
- Use CSS
- Fully qualify links (ie, personal/images/ not personal/images)
- **ext/tidy**

```
<?php
    $opts = array("clean" => true,
                 "drop-proprietary-attributes" => true,
                 "drop-font-tags" => true,
                 "drop-empty-paras" => true,
                 "hide-comments" => true,
                 "join-classes" => true,
                 "join-styles" => true);

    $tidy = tidy_parse_file("php.html", $opts);
    tidy_clean_repair($tidy);

    echo $tidy;
?>
```

- .Some browsers can accept certain content compressed.
- .We can trade some CPU cycles for reduced bandwidth usage.
- .Compression is usually a minor overhead.

- .A very nice apache module for compressing content
- .Can compress any content
- .Can switch on MIME Types and Browser Types
- .Uses Temporary Files

- .A built-in output buffer handler
- .Set in `php.ini`
- .Cannot be used with a custom `output_handler`
- .Does generally not work with `trans-sid`

These Slides: <http://pres.derickrethans.nl>

HTTP Cache Explanation: [http://www.mnot.net/cache\\_docs/](http://www.mnot.net/cache_docs/)

HTML Tidy: <http://www.w3.org/People/Raggett/tidy/>

mod\_gzip: [http://www.schroepl.net/projekte/mod\\_gzip/index.htm](http://www.schroepl.net/projekte/mod_gzip/index.htm)

PHP: <http://www.php.net>

squid: <http://www.squid-cache.org>

Webreference HTML Optimization:

<http://www.webreference.com/authoring/languages/html/optimize/>