

eZ publish and performance

LinuxTag 2004

June 24th, 2004. Karlsruhe, Germany

Derick Rethans <dr (at) ez (dot) no>

Questions?

Challenge: Improve eZ publish 3.3 performance

- o Too much memory usage
- o Too much CPU usage

Timing points:

Checkpoint	Elapsed	Rel. Elapsed	Memory	Rel. Memory
Script start	0.0000 sec	0.0517 sec	1,535.9063KB	1,827.2813KB
Module start 'content'	0.0517 sec	0.1870 sec	3,363.1875KB	6,943.0234KB
Module end 'content'	0.2387 sec	0.0402 sec	10,306.2109KB	863.1328KB
End	0.2789 sec		11,169.3438KB	0.8429KB
Total runtime:	0.2791 sec			

- o Accelerators store PHP 'Byte Code'
- o Shared Memory
- o Faster Execution

Timing points:

Checkpoint	Elapsed	Rel. Elapsed	Memory	Rel. Memory
Script start	0.0000 sec	0.0492 sec	182.4609KB	265.5313KB
Module start 'content'	0.0492 sec	0.0409 sec	447.9922KB	741.5781KB
Module end 'content'	0.0901 sec	0.0204 sec	1,189.5703KB	245.1563KB
End	0.1106 sec		1,434.7266KB	0.2394KB
Total runtime:	0.1108 sec			

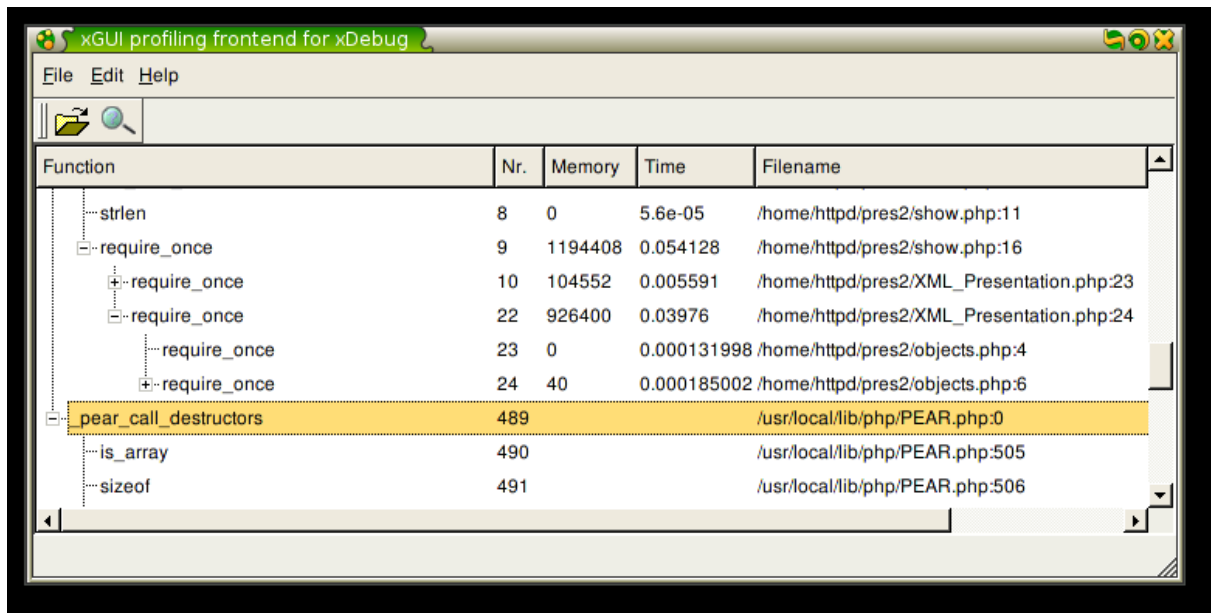
- o Xdebug: a tool to debug PHP
- o Tracing function calls
- o Profiling Applications



- o Xdebug's Trace Features
- o Trace function calls
- o 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%
```

o Simple Tracefile Browser



The screenshot shows a window titled "xGUI profiling frontend for xDebug". The window contains a table with the following columns: Function, Nr., Memory, Time, and Filename. The table lists several functions, 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

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

The screenshot displays the KCachegrind interface. On the left, the 'Flat Profile' window shows a list of functions with their self and called times. The function 'ezdatatype::loadandregisteralltypes' is highlighted in yellow, indicating it is the current focus.

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

The right-hand window, titled 'ezdatatype::loadandregisteralltypes', provides a detailed call graph. It shows the callers of this function and the callees it calls. The 'Callers' tab is active, showing a single caller with a cost of 100.000 and a count of 1.

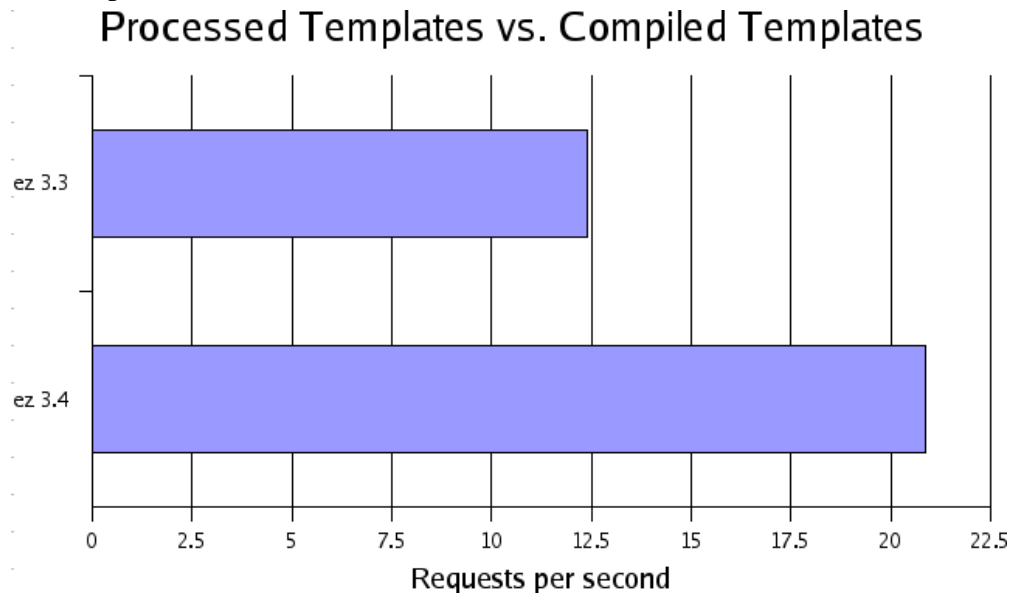
Cost	Count	Caller
100.000	1	include_once::/dat/ez.no-33/kernel/classes/ezdat...

Below this, the 'Callees' section shows the functions called by 'ezdatatype::loadandregisteralltypes':

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

At the bottom of the right window, there are navigation buttons: 'Parts', 'Calls', 'Call Graph', 'All Calls', 'Caller Map', and 'Assembler'.

- o Processed Templates est mort
- o Viva la Compiled Templates

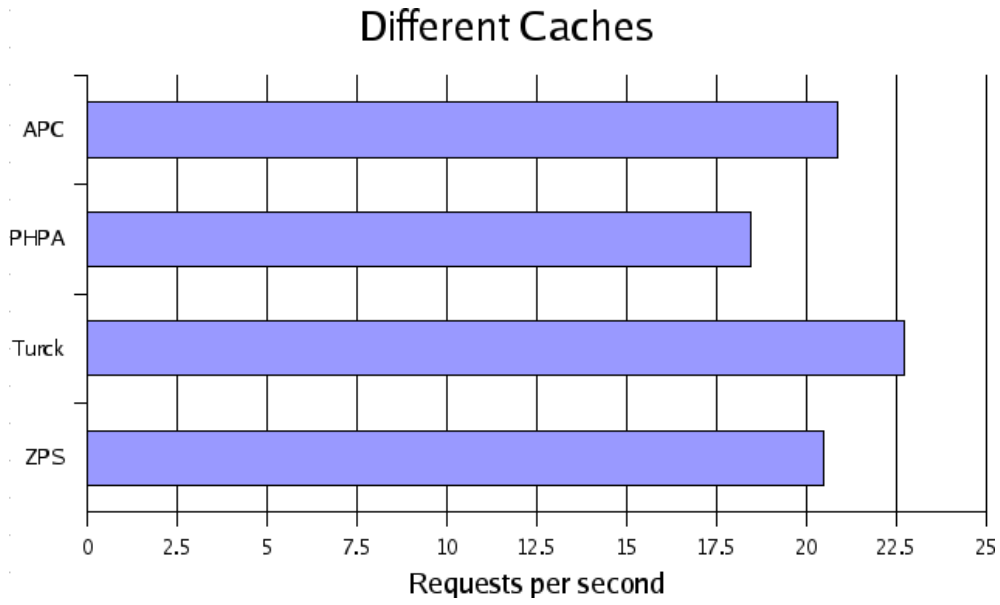


- o Processed vs. Compiled Templates
- o Loading Data Types on demand
- o From 11MB to 7MB

```
function &create( $dataTypeString )
{
    $types =& $GLOBALS["ezDataTypes"];
    $def = null;
+   if ( !isset( $types[$dataTypeString] ) )
+   {
+       ezDataType::loadAndRegisterType($dataTypeString);
+   }
+
    if ( isset( $types[$dataTypeString] ) )
    {
        $className = $types[$dataTypeString];
@@ -709,6 +716,4 @@
        var $Name;
    }

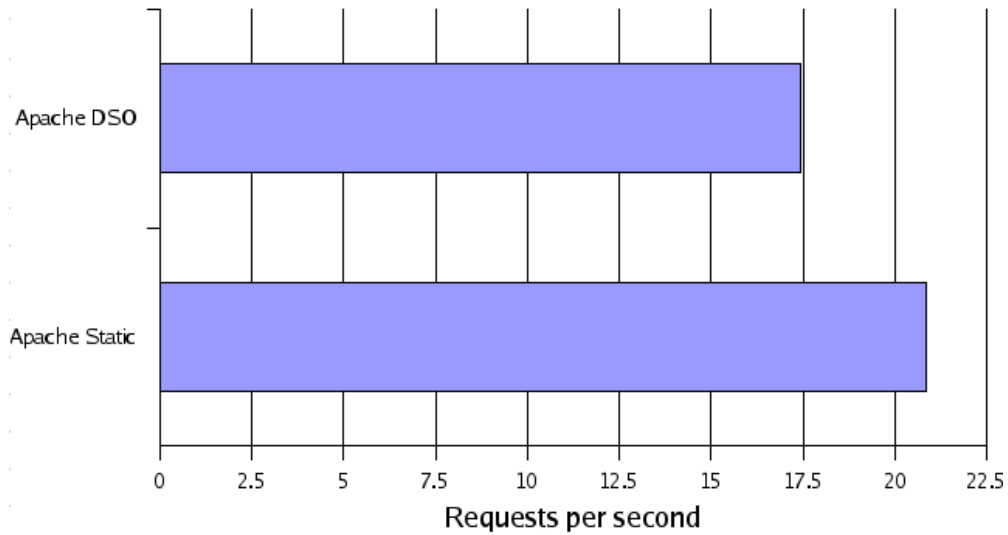
-ezDataType::loadAndRegisterAllTypes();
```

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



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

DSO vs. Static



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



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

- o Generated on Publish
- o Static Generated
- o First 'hit' is fast
 - [ContentSettings]
 - PreViewCache=enabled

- o Search Engine
- o Indexing takes Time
- o Delay Indexing to Cron Job
- o Publishing is Faster

```
[SearchSettings]  
DelayedIndexing=enabled
```


- o Static .html generation part of eZ publish
- o Further Optimizations (Template Compiler, Memory Usage)
- o Clustered Object Caching
- o SRM

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

Xdebug site: <http://xdebug.org>

Non-pic: <http://news.php.net/php.internals/9828>

eZ publish: <http://ez.no>

Questions?: dr (at) ez (dot) no

Index

Questions	2
Challenge: Improve eZ publish 3.3 performance	3
Accelerating: APC	4
The Hunt for Speed	5
Tracing the Problems	6
Xgui	7
Finding the Bottleneck	8
Speeding up	9
Reducing Memory Usage	10
Accelerating More	11
Apache Is Slow!	12
Say Squeeze	13
Static Squids	14
Preview Caching	15
No Indexes for Now!	16
For the Future	17
Resources	18