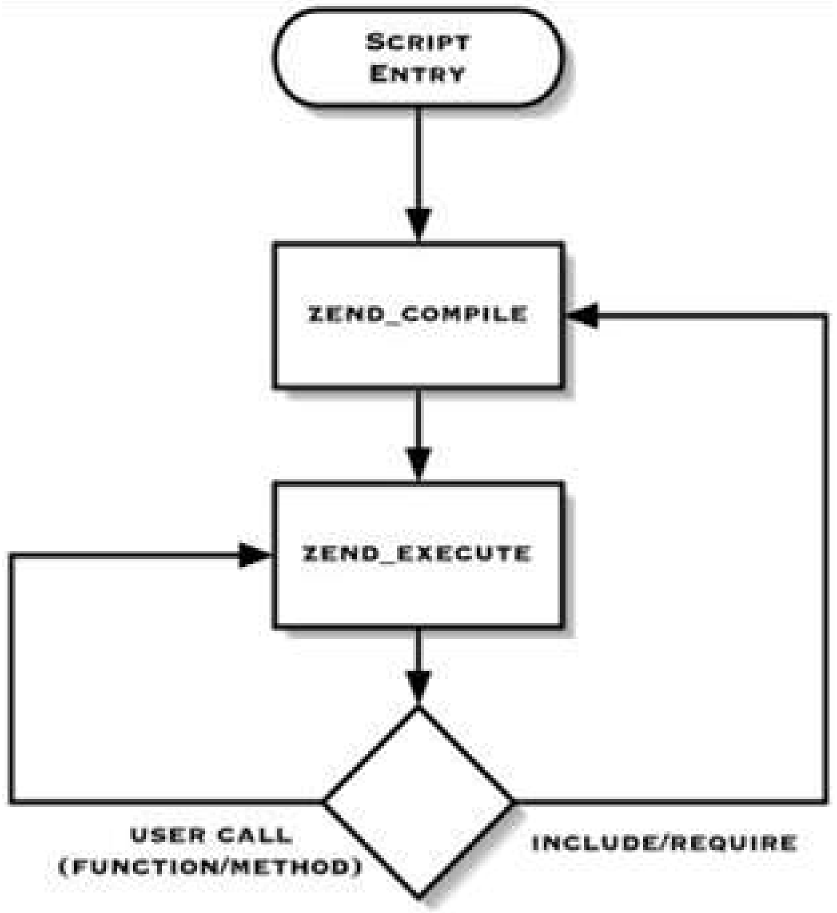
# Accelerating PHP Applications

**Ilia Alshanetsky**
**ilia@ilia.ws**

O'Reilly Open Source Convention
August 3rd, 2005

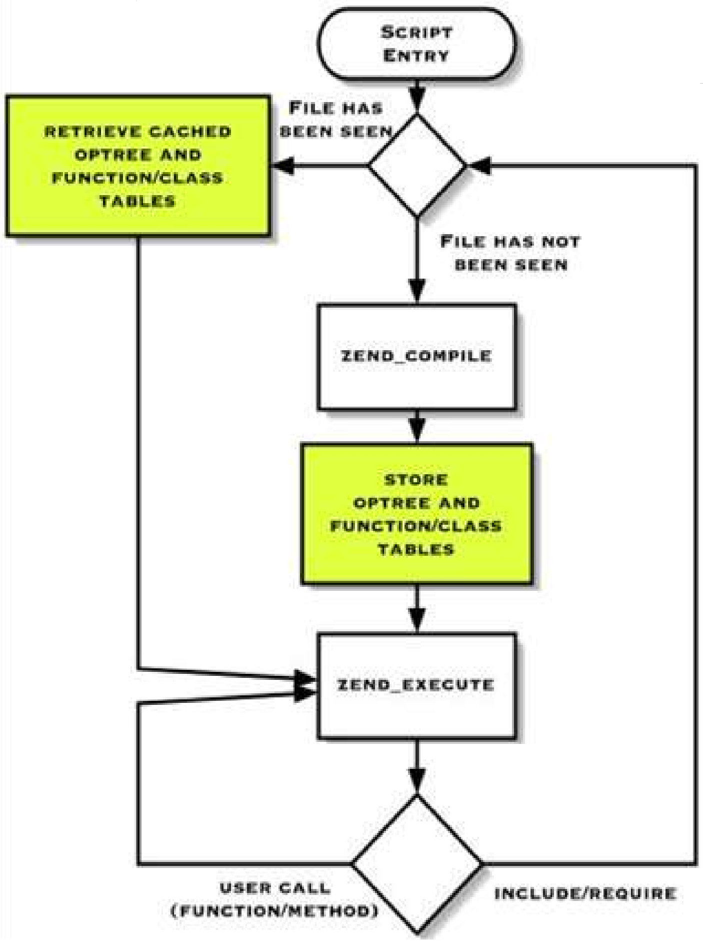
The 7th Annual
O'REILLY
OPEN
SOURCE
CONVENTION

## Bytecode/Opcode Caches



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

# Opcode Caches


Script Entry
File has been seen
Retrieve cached optree and function/class tables
File has not been seen
zend_compile
Store optree and function/class tables
zend_execute
User call (function/method)
include/require

## Benefits:

- Each PHP script is compiled only once for each revision.

- Reduced File IO thanks to opcodes being read from memory rather then being parsed from disk.

- Since compilation is one time event, generated opcodes can optimised for faster execution.

O'REILLY®

## Opcode Caches: Implementations

- APC (Alternative PHP Cache)
  - Open Source
  - Works with PHP 5.0+
  - Easy to install (pecl install apc)
  - Being actively maintained
- eAccelerator (Turck MMCache)
  - Open Source
  - Kinda/Sorta/Maybe works with PHP 5.0
  - VERY FAST (fastest cache for 4.X)
- Zend Performance Suit
  - On par performance with APC
  - Includes other acceleration tools (content caching)

## Compiler Optimisations

For absolute maximum performance it may be a good idea to ensure that all software is compiled to the take maximum advantage of the available hardware.

- Enable all compiler optimizations with -O3
- Make the compiler tune the code to your CPU via -march -mcpu
- Try to make the compiler use CPU specific features -msse -mmmx -mfpmath=sse

```
export CFLAGS="-O3 -msse -mmmx -march=pentium3 \
-mcpu=pentium3 -funroll-loops -mfpmath=sse"
```

## Apache/PHP Integration

For maximum performance compile PHP statically into Apache (up to 30% speed increase). Or use PHP 4.3.11+ where **--prefer-non-pic** is default.

**How to compile PHP statically into Apache**

```
# PHP
./configure --with-apache=/path/to/apache_source

# Apache
./configure --activate-module=src/modules/php4/libphp4.a
```

## Web Server: File IO

- Keep **DirectoryIndex** file list as short as possible.
- Whenever possible disable **.htaccess** via **AllowOverride none**.
- Use Options **FollowSymLinks** to simplify file access process in Apache.

- If logs are unnecessary disable them.
- If logging is a must, log everything to 1 file and break it up during analysis stage.

## Bandwidth Optimizations

**Less output is good because…**

- Saves server bandwidth (saves $$ too).
- Reduces server resource usage (CPU/Memory/Disk)
- Pages load faster for clients.
- Reduces network IO high traffic sites, where it is the primary bottleneck in most cases.

## Content Compression

- Most browser support retrieval of compressed pages decompressing them before rendering.

- Compressed pages are on average are 7-10 times smaller, however compression can take 3%-5% of CPU.

**Implementations:**

- Apache 1 (mod_gzip)
- Apache 2 (mod_deflate)
- PHP
  - php.ini ( zlib.output_compression=1 )
  - script (ob_start("ob_gzhandler") )

O'REILLY

## Tuning PHP Configuration

- register_globals = Off **
- magic_quotes_gpc = Off
- expose_php = Off
- register_argc_argv = Off
- always_populate_raw_post_data = Off **
- session.use_trans_sid = Off **
- session.auto_start = Off **
- session.gc_divisor = 1000 or 10000
- output_buffering = 4096

**** Off by default**

## Tuning PHP File Access

Whenever opening files or including scripts into the main script try to specify a full path or at least an easily resolvable partial path.

```
Bad Approach:
<?php
include "file.php";
?>

Performance Friendly Approach:
<?php
include "/path/to/file.php";
// or
include "./file.php";
?>
```

O'REILLY

## Regular Expressions

While very useful tool for string manipulation, regular expression leave much to be desired when it comes to performance.

```
<?php
// Slow
if (preg_match("!^foo_!i", "FoO_")) { }
// Much faster
if (!strncasecmp("foo_", "FoO_", 4)) { }

// slow
if (preg_match("![a8f9]!", "sometext")) { }
// Faster
if (strpbrk("a8f9", "sometext")) { }
?>
```

## Reference Tricks

References can be a valuable tool to simplify and accelerate access to complex data types as well as a memory saving tool.

```
<?php
$a['b']['c'] = array();
// slow 2 extra hash lookups
per access
for($i = 0; $i < 5; $i++)
    $a['b']['c'][$i] = $i;
// much faster reference
based approach
$ref =& $a['b']['c'];
for($i = 0; $i < 5; $i++)
    $ref[$i] = $i;
?>
```

```
<?php
$a = "abc";
// memory intensive
solution
function a($str) {
    return $str . "d";
}
$a = a($a);
// more effecient approach
function b(&$str) {
    $str .= "d";
}
b($a);
?>
```

O'REILLY®

## What Is Caching?

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

## Pros and Cons of Caching

- Pros:
  - Significant Speed Increases
  - Reduction in consumption of some resources
- Cons:
  - Increase in Architectural Complexity
  - Potential for Stale or Inconsistent Data

## On-Demand Caching

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

```
RewriteEngine on
RewriteRule /.*\.[^h][^t][^m][^l]$ /$1.html
ErrorDocument 404 /index.php
DirectoryIndex 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));
}
/* content generation */
if (!FORCE_DYNAMIC) {
 echo $contents = ob_get_clean();
 file_put_contents($lang."/".$current_page.".html",
                                                   $contents);
}?>
```

O'REILLY

## SQL & Performance

**Most large applications will end up using databases for information storage. Improper use of this resource can lead to significant and continually increasing performance loss.**

## Check Your Queries

Most databases offer mechanisms to analyze query execution and determine if it’s running in an optimal manner.

**SLOW**

```
EXPLAIN select * from mm_users where login LIKE '%ilia%';
+----------+------+---------------+------+---------+------+-------+------------+
| table    | type | possible_keys | key  | key_len | ref  | rows  | Extra      |
+----------+------+---------------+------+---------+------+-------+------------+
| mm_users | ALL  | NULL          | NULL |    NULL | NULL | 27506 | where used |
+----------+------+---------------+------+---------+------+-------+------------+
```

**FAST**

```
EXPLAIN select * from mm_users where login LIKE 'ilia%';
+----------+-------+---------------+-------+---------+------+------+------------+
| table    | type  | possible_keys | key   | key_len | ref  | rows | Extra      |
+----------+-------+---------------+-------+---------+------+------+------------+
| mm_users | range | login         | login |      50 | NULL |    2 | where used |
+----------+-------+---------------+-------+---------+------+------+------------+
```

## Questions