



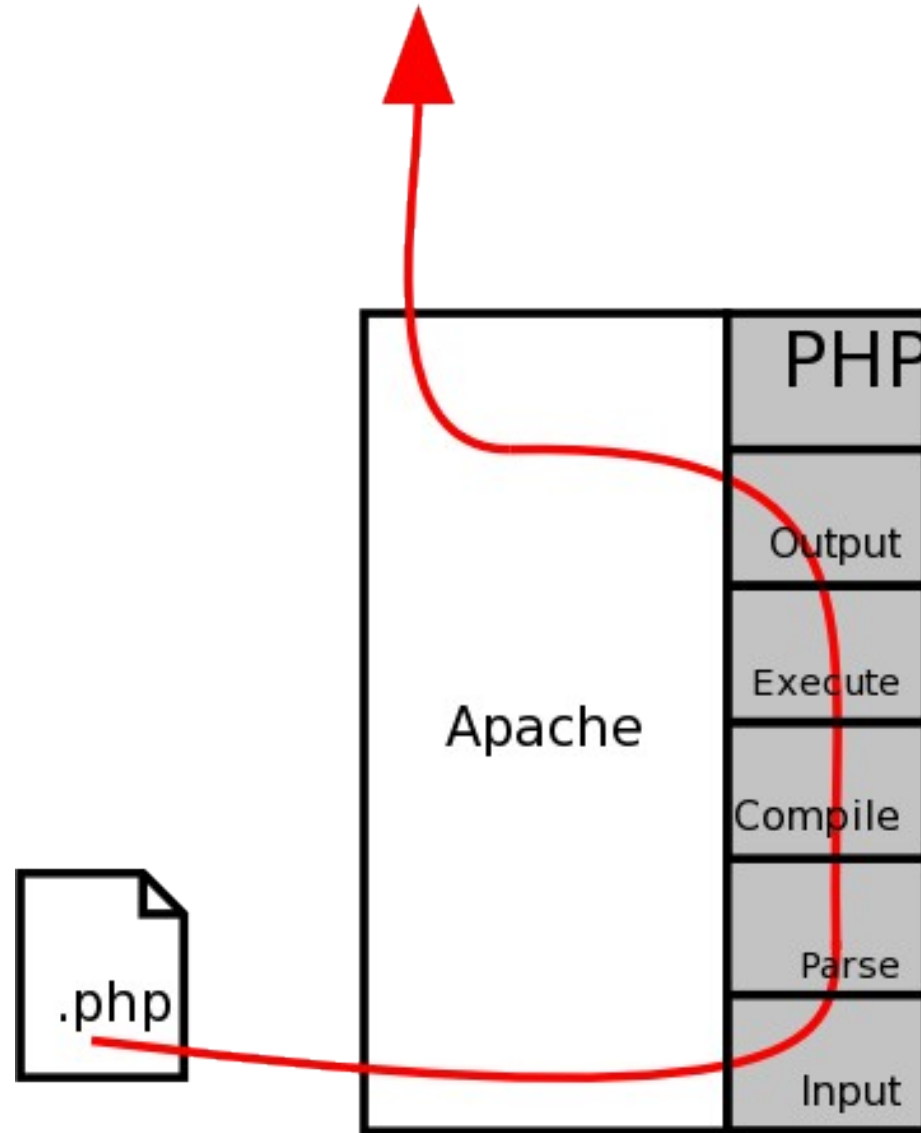
WebCamp 2008 - Prague, Czech Republic

Derick Rethans - dr@ez.no

<http://derickrethans.nl/talks.php>

- Dutchman living in Norway
- eZ Systems A.S.
- eZ Components project lead
- PHP development
- mcrypt, input_filter, date/time support, unicode
- QA

Introduction



- Lexical analyze script source
- Divide into logical blocks of characters
- Give special blocks a meaning
- flex (but only 2.5.4!)

The Parse Error:

```
Parse error: parse error,  
unexpected T_CLASS, expecting ',' or ';' in - on line 2
```

Compiling Diagram

```
<?php
function normalizeColorArray($array)
{
    foreach (array_keys($array) as $key)
    {
        $array[$key] = (float) $array[$key]/255;
    }

    return $array;
}

function rgbToCMYK($rgbArray)
{
    $cya = 1 - min(1, max((float) $rgbArray['r'], 0));
    $mag = 1 - min(1, max((float) $rgbArray['g'], 0));
    $yel = 1 - min(1, max((float) $rgbArray['b'], 0));

    $min = min($cya, $mag, $yel);
    if (1 - $min == 0)
    {
        return array('c' => 1, 'm' => 1,
    }

    return array('c' => ($cya - $min) / (1 - $min),
                'm' => ($mag - $min) / (1 - $min),
                'y' => ($yel - $min) / (1 - $min),
                'k' => $min );
}

function rgbToCMYK2($r, $g, $b)
{
    return e2Nacht::rgbToCMYK(array('r' => $r, 'g' => $g, 'b' => $b));
}
?>
```

Parse

Compile
zend_compile



class symbol table

function symbol table
normalizeColorArray
rgbToCMYK
rgbToCMYK2



fibonacci.php:

```
<?php
    $cache = array();

    function fibonacci($nr) {
        global $cache;

        if (isset($cache[$nr])) {
            return $cache[$nr];
        }
        switch ($nr) {
            case 0:
                die("Invalid Nr\n");
            case 1:
                return 1;
            case 2:
                return 1;
            default:
                $r = fibonacci($nr - 2) + fibonacci($nr - 1);
                $cache[$nr] = $r;
                return $r;
        }
    }

    echo fibonacci($argv[1])."\n";
?>
```

Compiling Break-down

```

Function fibonacci:
filename:      /home/httpd/html/test/pres/fibonacci.php
function name: fibonacci
number of ops: 38
compiled vars: !0 = $nr, !1 = $cache, !2 = $r
l#      # op                                     return operands
-----
function fibonacci($nr) {
4      0 RECV                                     1
      global $cache;
5      1 FETCH_W                                 $0      'cache'
      2 ASSIGN_REF                               !1, $0
      if (isset($cache[$nr])) {
7      3 ZEND_ISSET_ISEMPY_DIM_OBJ              ~1      !1, !0
      4 JMPZ                                     ~1, ->8
      return $cache[$nr];
8      5 FETCH_DIM_R                             $2      !1, !0
      6 RETURN                                   $2
      }
9      7* JMP                                    ->8
      switch ($nr) {
      case 0:
11     8 CASE                                    ~3      !0, 0
      9 JMPZ                                     ~3, ->12
      die("Invalid Nr\n");
12     10 EXIT                                   'Invalid+Nr%0A'
13     11* JMP                                    ->14
      case 1:
      12 CASE                                    ~3      !0, 1
      13 JMPZ                                     ~3, ->16
      return 1;
14     14 RETURN                                1
15     15* JMP                                    ->18
      case 2:
      16 CASE                                    ~3      !0, 2
      17 JMPZ                                     ~3, ->20
      return 1;
16     18 RETURN                                1
17     19* JMP                                    ->21
      default:
20     JMP                                       ->35
      $r = fibonacci($nr - 2) + fibonacci($nr - 1);
18     21 INIT_FCALL_BY_NAME                     'fibonacci'
      22 SUB                                       ~4      !0, 2
      23 SEND_VAL
      24 DO_FCALL_BY_NAME

```

Disassembler: vld

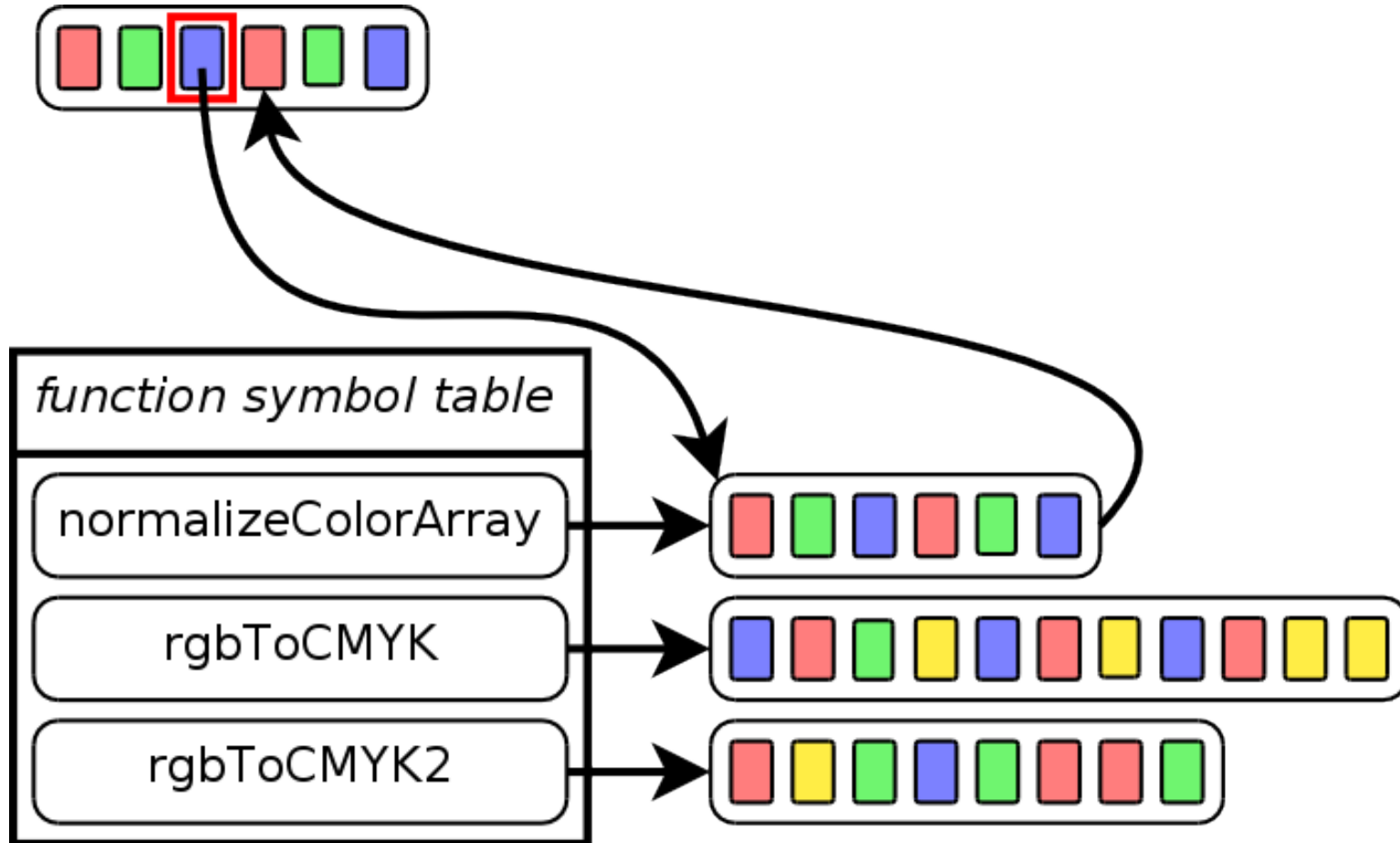
Dumps oparray per element:

- main script
- function
- class method

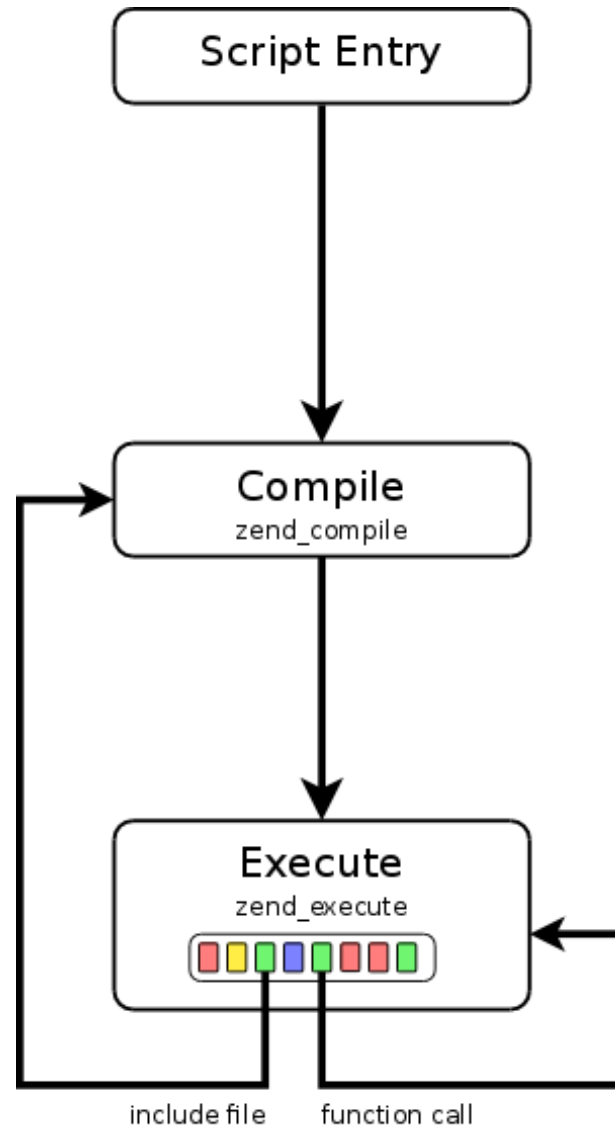
Usage:

```
cvs -d :pserver:cvsread@cvs.xdebug.org:/repository login
# passwd = srmread
cvs -d :pserver:cvsread@cvs.xdebug.org:/repository co -d vld vle
cd vld
phpize && make && make install
php -dextension=vld.so -dvld.active=1 script.php
```


Executing: Diagram



Executing In a diagram



print vs. echo

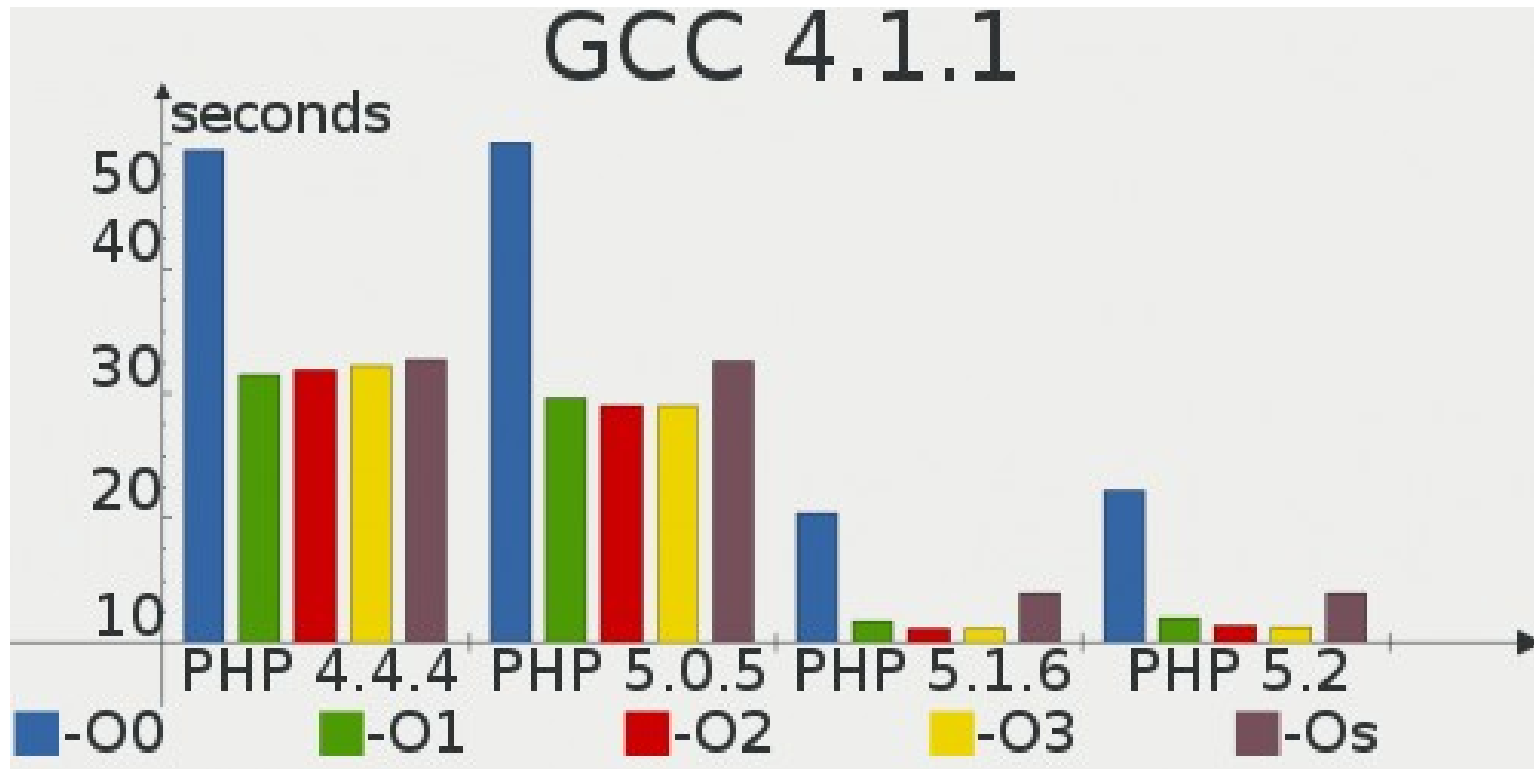
`$i++` vs. `++$i`

single quotes vs. double quotes

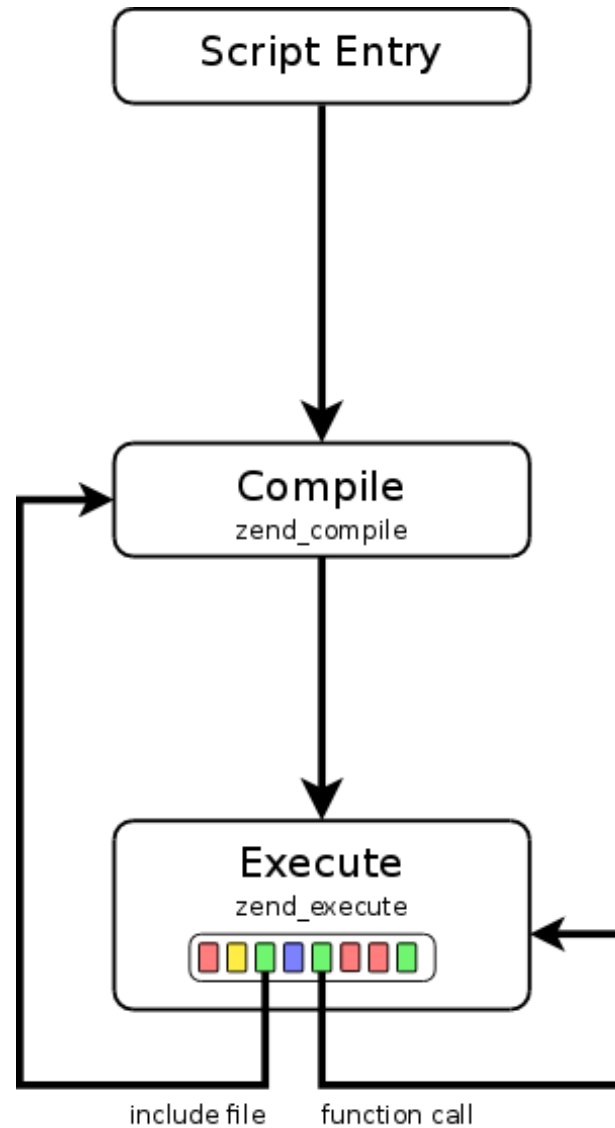
`${o}`



2003

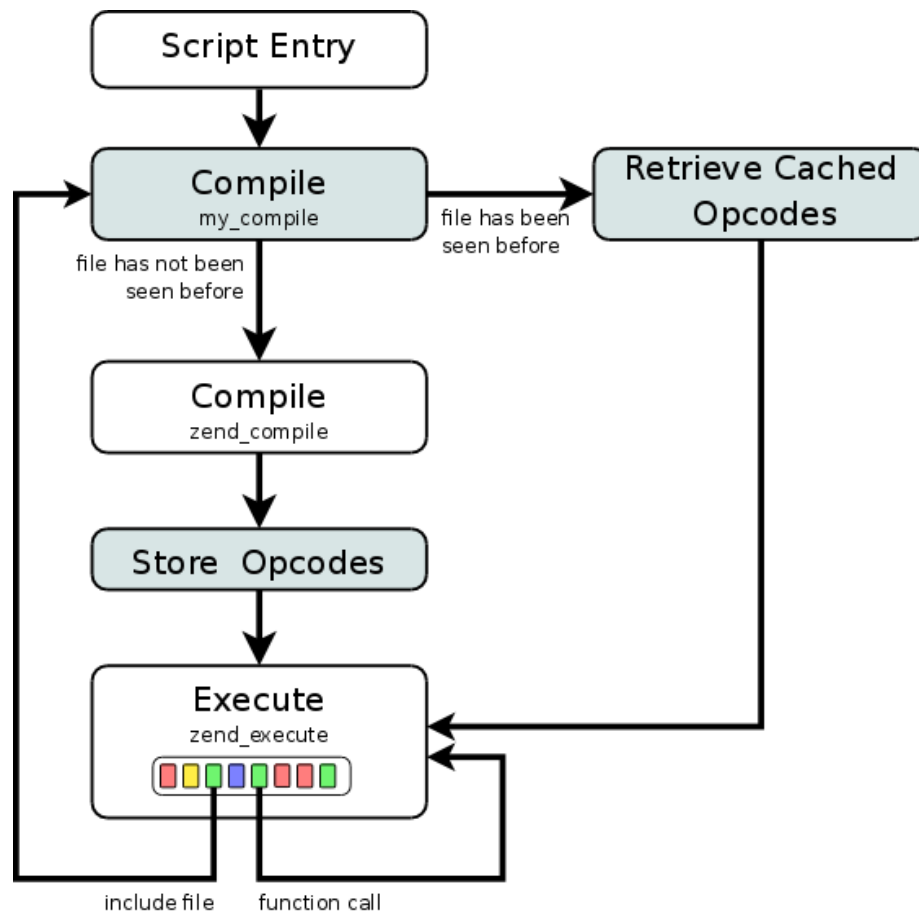


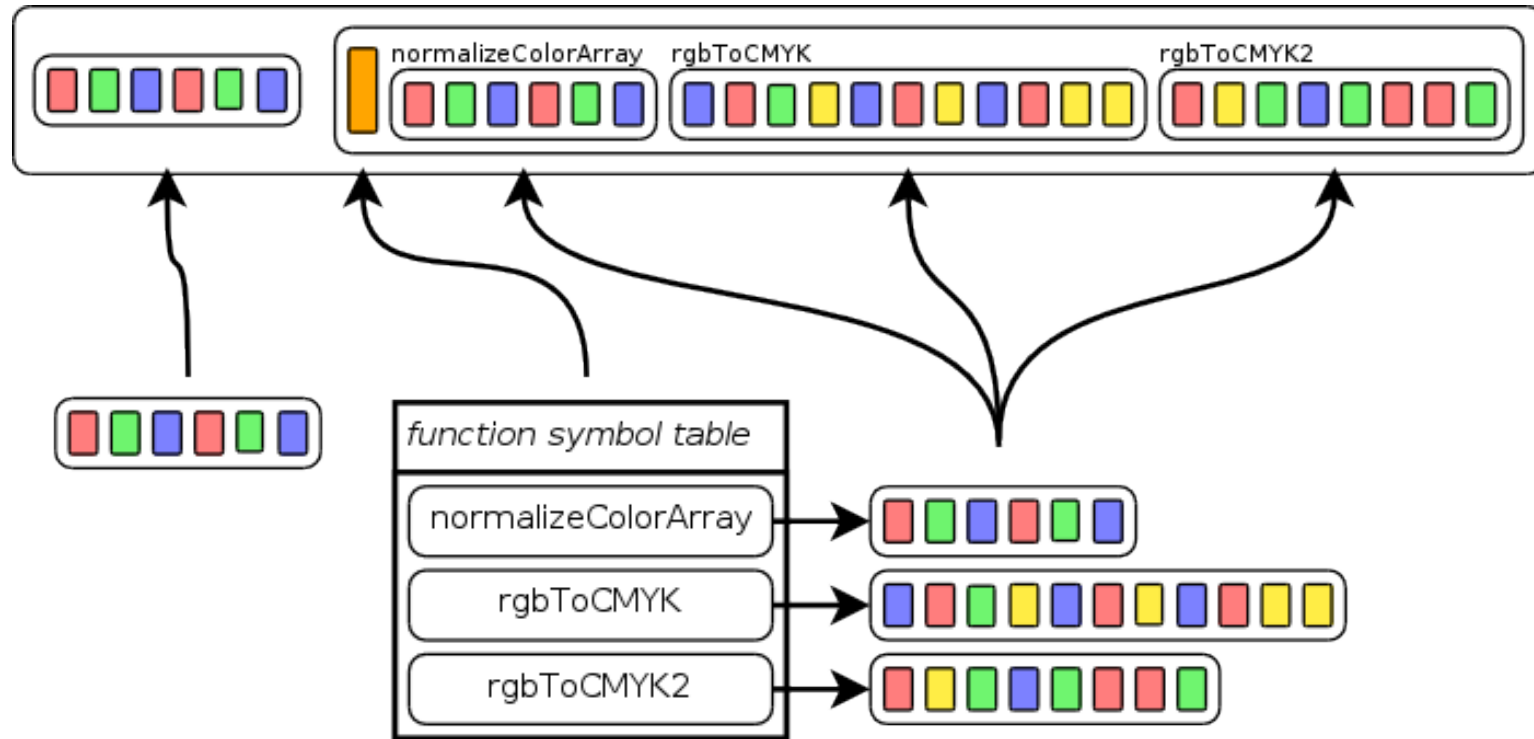
Executing In a diagram



Compiler Caches

How it works





- Serialization into SHM
- Direct execution from SHM (mostly)

- APC: active development, PHP license
- eAccelerator: GPL
- PHP Accelerator: updated, but not improved
- Turck MM Cache: abandoned
- XCache: new
- Zend Platform: commercial

Compiling

Conditional functions

```
filename:      /home/httpd/html/test/conditional_func.php
function name: (null)
number of ops: 10
compiled vars: none
line   # op                                     operands
-----
if (true) {
  2     0 EXT_STMT
      1 JMPZ                                     true, ->5
      function foo() {
  3     2 EXT_STMT
      3 ZEND_DECLARE_FUNCTION                 '%00foo%2Ftmp%2Fconditional_func.php0xfe455f', 'foo'
  6     4 JMP                                     ->7
      function foo() {
  7     5 EXT_STMT
      6 ZEND_DECLARE_FUNCTION                 '%00foo%2Ftmp%2Fconditional_func.php0xfe458b', 'foo'
  }
  12    7 EXT_STMT
      8 RETURN                                  1
      9* ZEND_HANDLE_EXCEPTION
}
```

Function foo:

```
filename:      /home/httpd/html/test/conditional_func.php
function name: foo
number of ops: 6
compiled vars: none
line   # op                                     operands
-----
function foo() {
  3     0 EXT_NOP
      echo "1\n";
  4     1 EXT_STMT
      2 ECHO                                     '1%0A'
  }
  5     3 EXT_STMT
      4 RETURN                                  null
      5* ZEND_HANDLE_EXCEPTION
}
```

End of function foo.

Function foo:

```
filename:      /home/httpd/html/test/conditional_func.php
function name: foo
number of ops: 6
compiled vars: none
line   # op                                     operands
-----
function foo() {
  7     0 EXT_NOP
      echo "2\n";
  8     1 EXT_STMT
      2 ECHO                                     '2%0A'
  }
  9     3 EXT_STMT
      4 RETURN                                  null
      5* ZEND_HANDLE_EXCEPTION
}
```

End of function foo.

demo

Dumps includes/classes hierarchies

Download from <http://t3.dotgnu.info/blog/tags/included/>

Install:

```
tar -xvzf included-0.3.tgz
cd included
phpize && make && make install
```

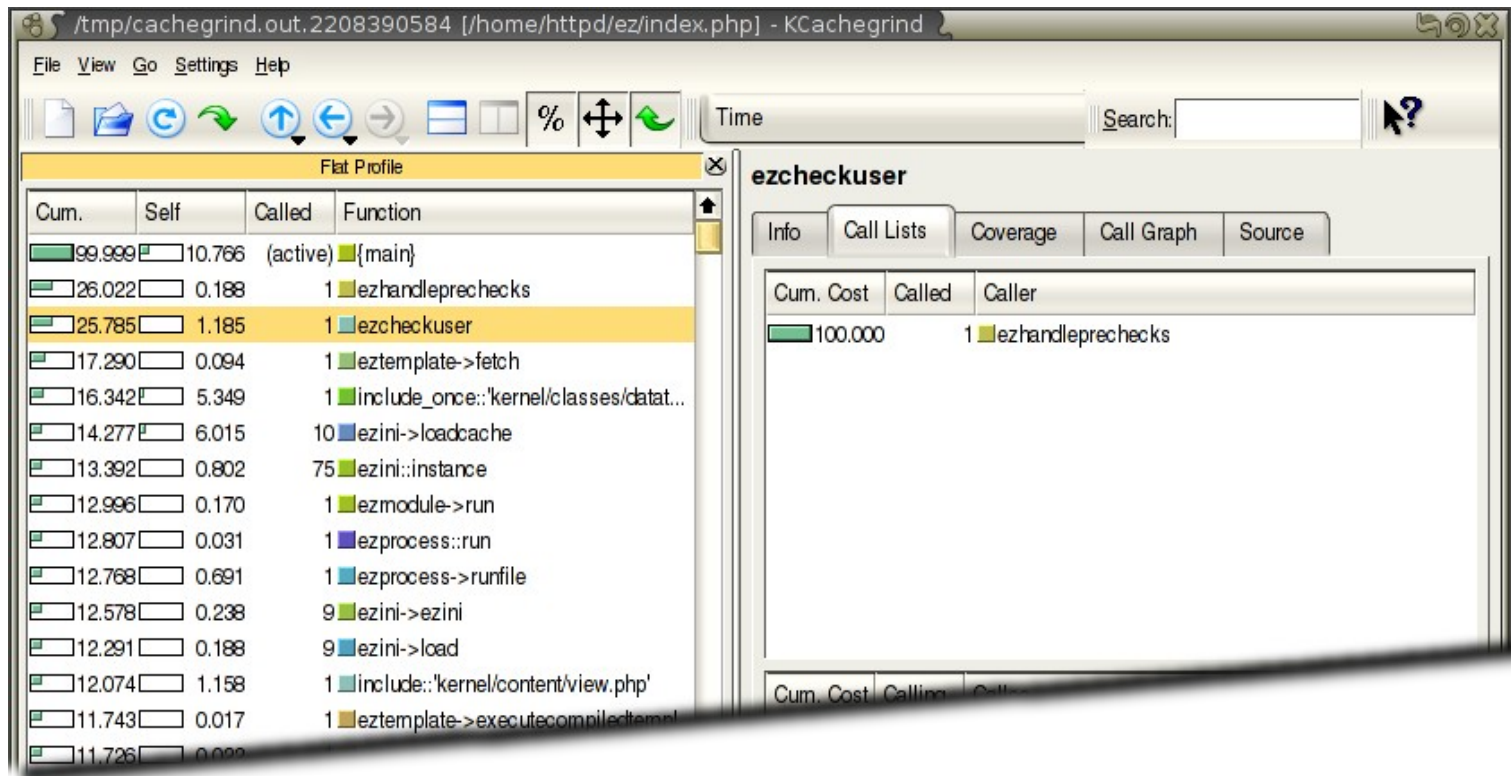
Add to php.ini:

```
extension=included.so
included.enabled=1
included.dumpdir=/tmp
```

demo

Profiling

KCacheGrind's Flat Profile and Call List



The screenshot shows a web browser window displaying the KCacheGrind profiling results. The browser address bar shows the URL: `/tmp/cachegrind.out.2208390584 [/home/httpd/ez/index.php] - KCacheGrind`. The browser interface includes a menu bar (File, View, Go, Settings, Help) and a toolbar with various navigation icons. The main content area is divided into two panels.

The left panel, titled "Flat Profile", displays a table with the following columns: Cum., Self, Called, and Function. The data is as follows:

Cum.	Self	Called	Function
99.999	10.766	(active)	{main}
26.022	0.188	1	ezhandleprechecks
25.785	1.185	1	ezcheckuser
17.290	0.094	1	eztemplate->fetch
16.342	5.349	1	include_once: 'kernel/classes/datat...
14.277	6.015	10	ezini->loadcache
13.392	0.802	75	ezini::instance
12.996	0.170	1	ezmodule->run
12.807	0.031	1	ezprocess::run
12.768	0.691	1	ezprocess->runfile
12.578	0.238	9	ezini->ezini
12.291	0.188	9	ezini->load
12.074	1.158	1	include: 'kernel/content/view.php'
11.743	0.017	1	eztemplate->executecompiledtempl
11.726	0.022		

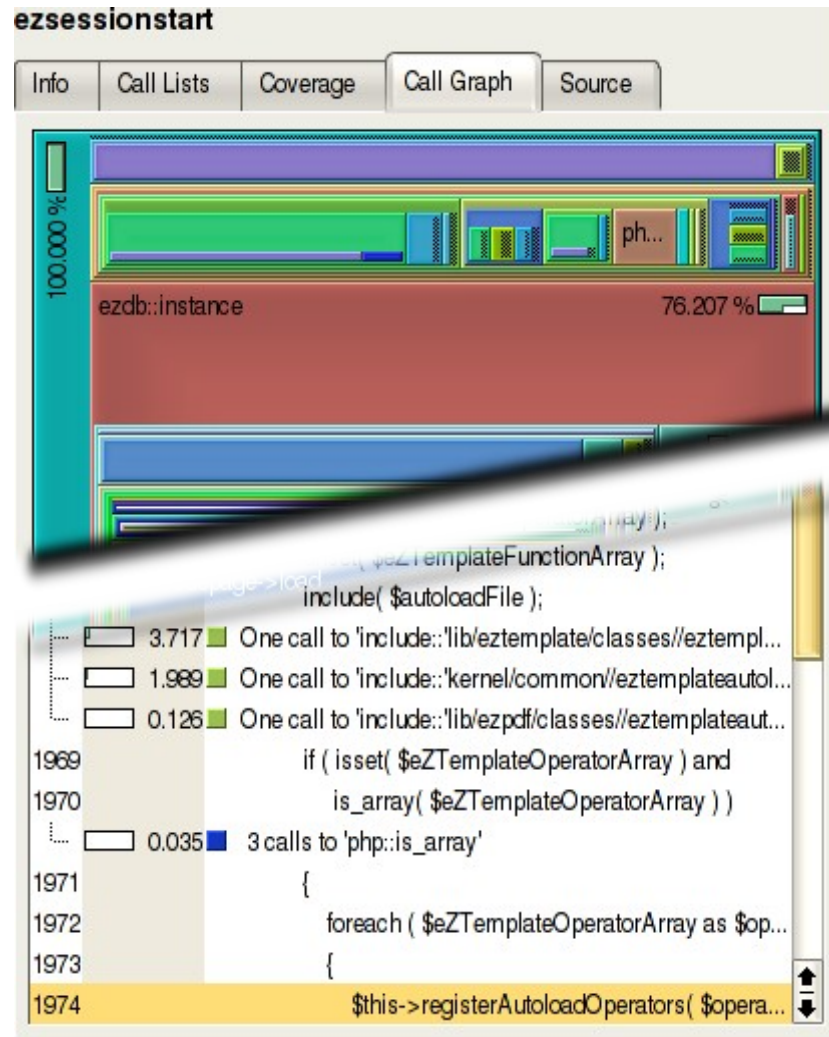
The right panel, titled "ezcheckuser", displays a call list with the following columns: Cum. Cost, Called, and Caller. The data is as follows:

Cum. Cost	Called	Caller
100.000	1	ezhandleprechecks

```
xdebug.profiler_enable=1           ; enable profiler
xdebug.profile_output_dir=/tmp     ; output directory
xdebug.profile_output_name=crc32  ; file extension
```

Profiling

KCacheGrind's Call Graph and Source Annotations

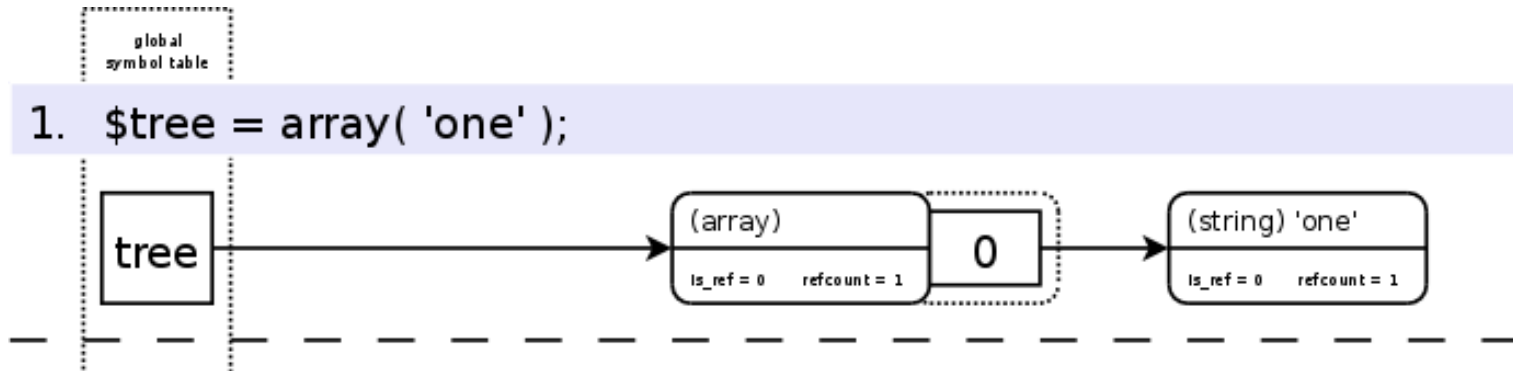


demo

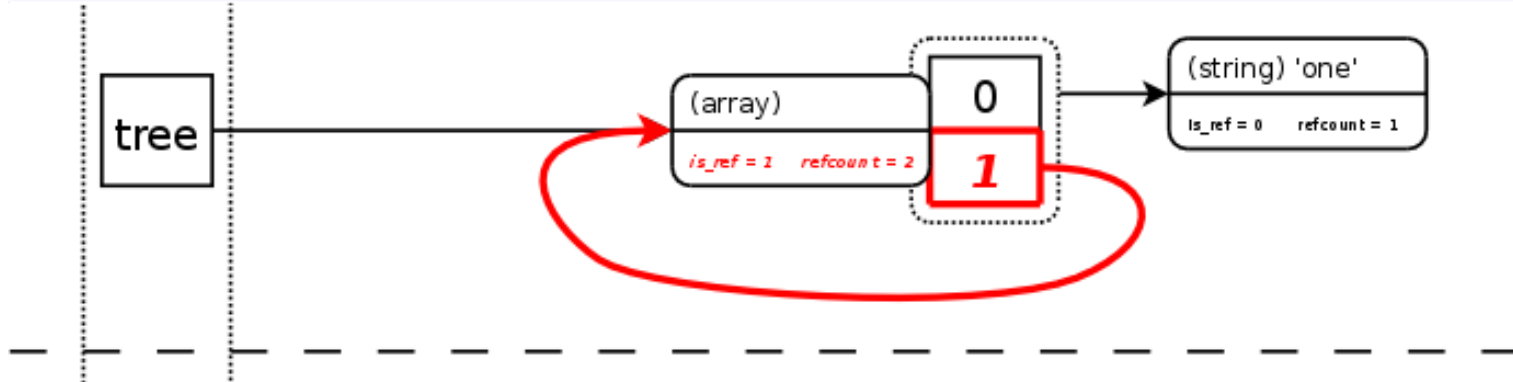
Variables

Circular References

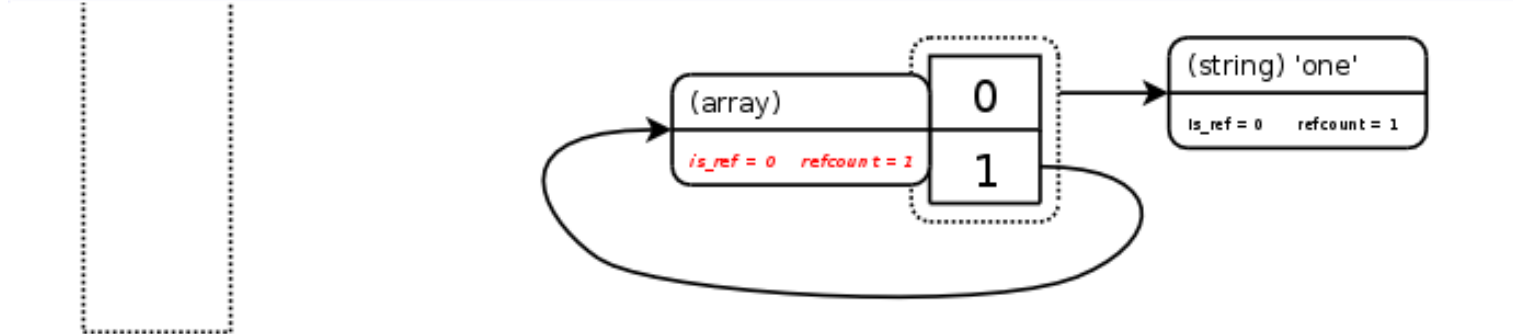
1. `$tree = array('one');`



2. `$tree[] = &$tree;`



3. `unset($tree);`





These Slides: <http://derickrethans.nl/talks.php>

VLD: <http://www.derickrethans.nl/vld.php>

Xdebug: <http://xdebug.org>

Included: <http://t3.dotgnu.info/blog/tags/included/>

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