Think

ThinkInLAMP.com

PHPCon 2015

北京站

PHP 7

- New Engine For The Good Old Train

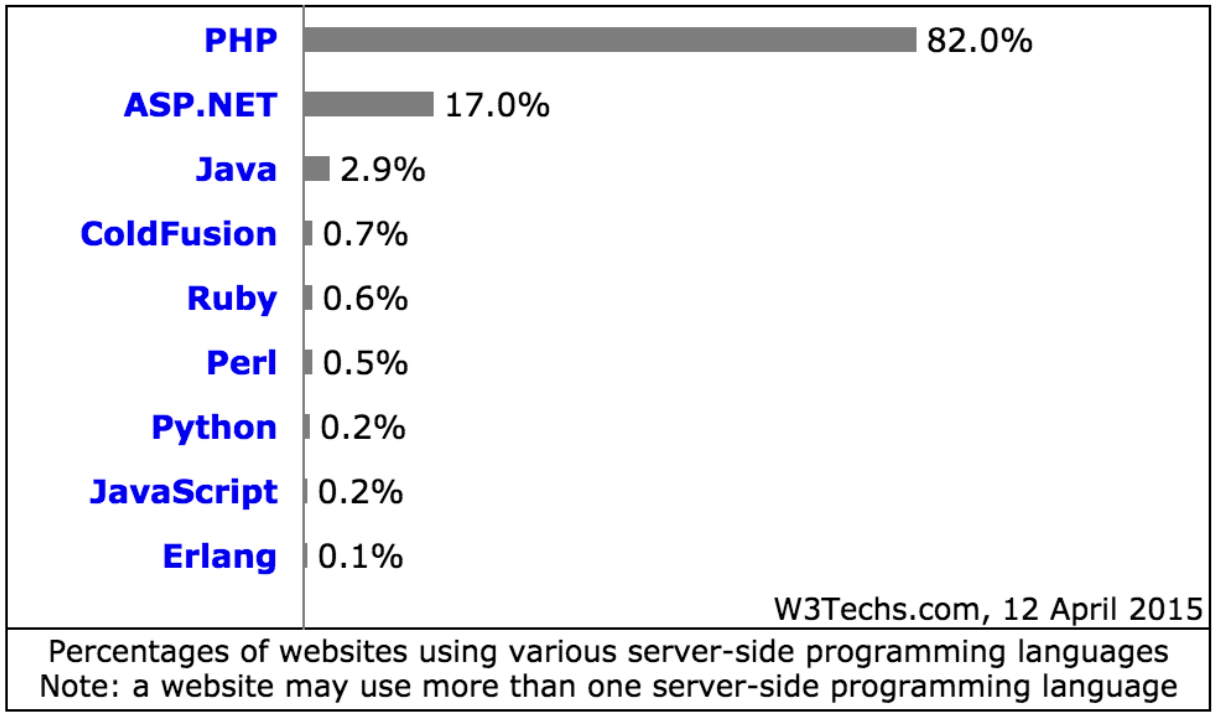
@laruence

About Me

- Author of Yaf, Yar, Yac, Taint, Lua, etc
- Maintainer of APC, Zend Opcache, Msgpack, etc
- Chief software architect At Weibo since 2012
- PHP core developer since 2011
- Zend consultant since 2013
- Core author of PHP7

About PHP

- 20 years history
- Most popular Web service program language
- Over 82% sites are use PHP as server program language

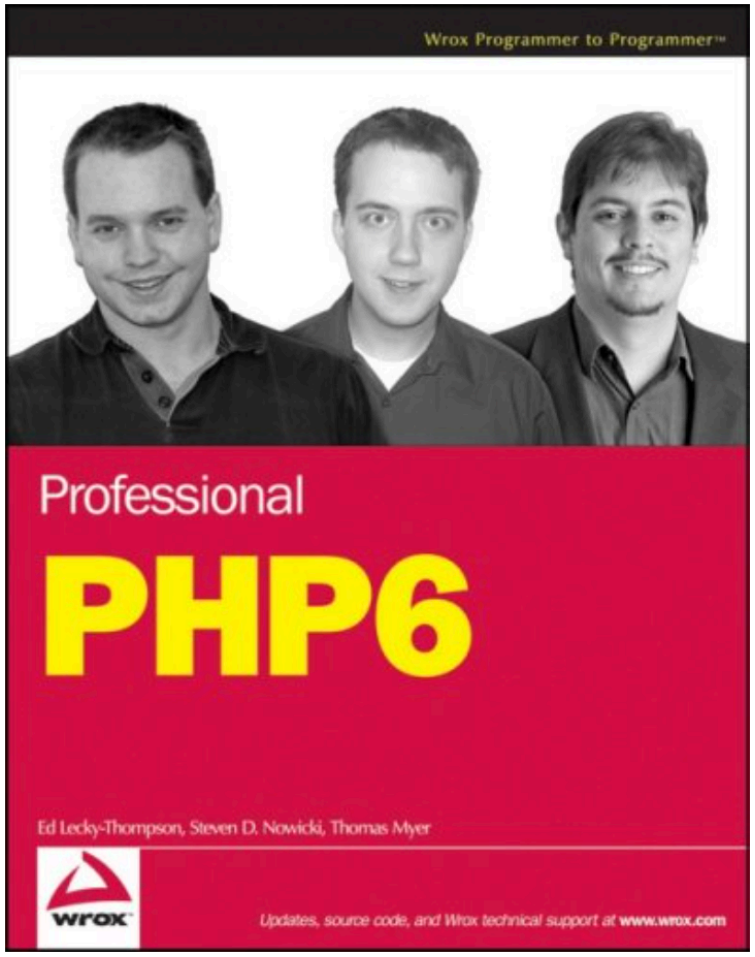


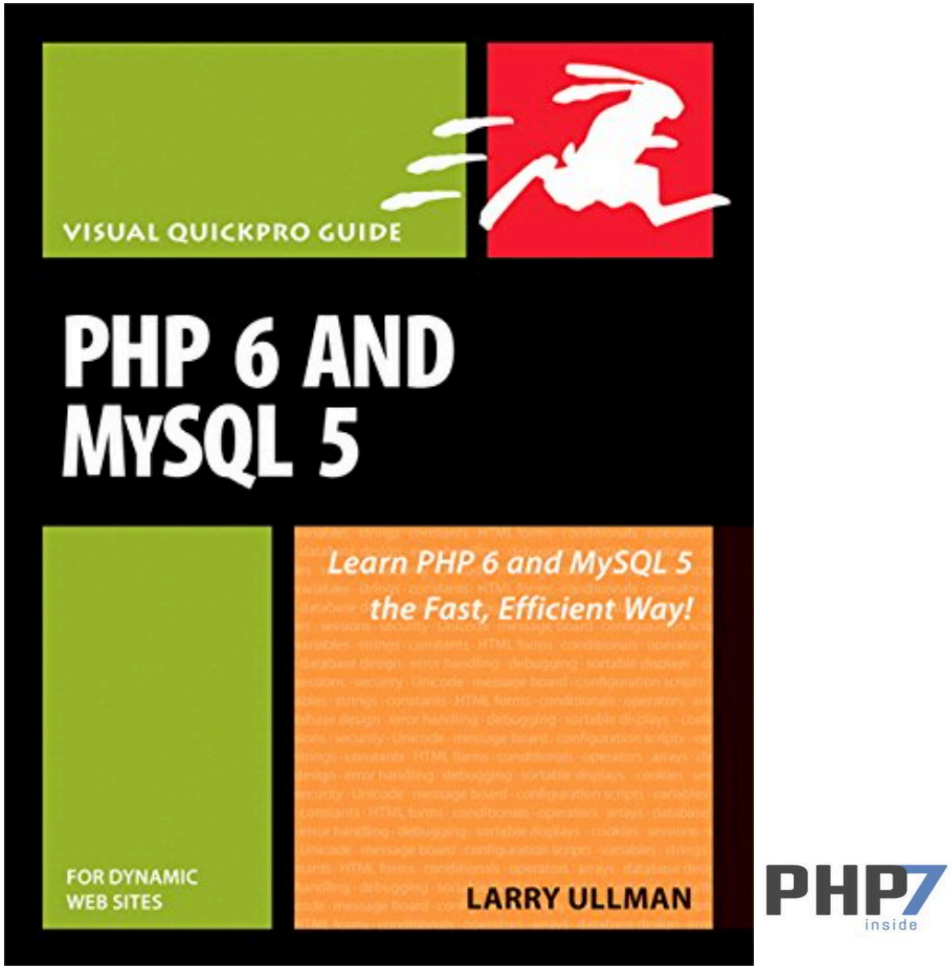
Percentages of websites using various server-side programming languages
Note: a website may use more than one server-side programming language

PHP7 Rodemap

- 2015/06/09 First alpha will be tagged
- 2015/06/11 First alpha will be released
- PHP7 final is planed on 2015/11/12

Where is PHP 6?





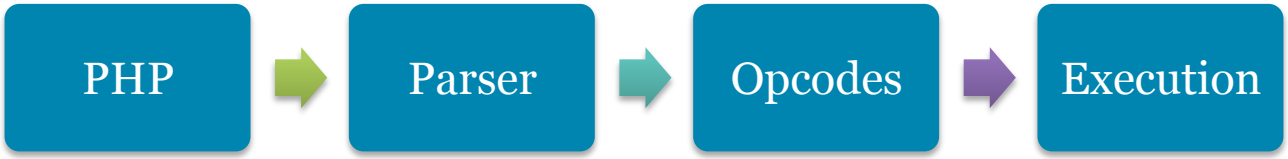
PHP 7 New Features

PHP 7?

- PHP NG – Engine Refactor(Zend Engine 3) - performance improvements
- Abstarct Syntanx Tree
- Int64 Improvement
- Uniform variable syntax
- Native TLS
- Consistently foreach behaviors
- New <=>, ?? operaters
- Return Type Declarations
- Scalar Type Declarations
- Exceptions in Engine
- And Dozens features...

Abstract Syntax Tree

- PHP5



- PHP7

Int64 Improvement

- >2GB string
- System independent 64bits long
- >2GB file uploading
- Fully 64bits intgers cross platforms

	string size	signed integer
Platform	*int*	*long*
LP64	32 bit	64 bit
LLP64	32 bit	32 bit
ILP64	64 bit	64 bit

Uniform Variables Syntanx

- $foo()['bar']()
- $foo['bar']::$baz
- foo()() – (foo())()
- $foo->bar()::baz()
- (function() { ... })()
- $this->{$name}()
- [$obj, 'method']()
- Foo::$bar['baz']()

  PHP5: Foo::{$bar['baz']}()

  PHP7: (Foo::$bar)['baz']()

PHP7 inside

New Operators

- <=>
  - $a <=> $b
  - $a > $b? 1 : (($a < $b)? -1 : 0)
- ??
  - $b = isset($a[“x”])? $a[“x”] : NULL
  - $b = $a[“x”] ?? NULL

PHP7 inside

Unicode Codepoint Escape Syntax

- \u{xxxxx}
  - echo "\u{4f60}\u{597d}, \u{66f4}\u{52a0}\u{5f3a}\u{5927}\u{7684}PHP7";

  你好, 更加强大的*PHP7*

Return Type Declarations

- function foo(): array {
  return [];
  }
- interface A {
  static function make(): A;
  }
- function foo(): DateTime {
  return null;
  }

PHP Fatal error: Return value of foo() must be an instance of DateTime, null returned

PHP7 inside

Scalar Type Declarations

- function foo(int num)
- function bar (string name)
- function foobar() : float {}
- function add(int l, int r) : int {}
- class A {

```
        public function start (bool start) {}
}
```

Exceptions in Engine

- Use of exceptions in Engine

```
try {
    non_exists_func();
} catch (EngineException $e) {
    echo "Exception: {$e->getMessage()}\n";
}
```

Exception: Call to undefined function non_exists_func()

- Uncaught Exception result to FATAL ERROR

```
non_exists_func();
```

PHP Fatal error: Call to undefined function non_exists()

PHP7 inside

Anonymous Class

- Like Anonymous function
  - var_dump(new class {public $a = "anonymous";});

```
object(class@anonymous)#1 (1) {
    ["a"]=>
     string(9) "anonymous"
}
```

Foreach behavior improvement

- PHP5:
  - *$a = array(1,2,3); foreach($a as $v) { var_dump(current($a)) }*

int(2)

int(2)

int(2)

  - *$a = array(1,2,3); $b=&$a; foreach($a as $v) { var_dump(current($a))}*

int(2)

int(3)

bool(false)

  - *$a = array(1,2,3); $b=$a; foreach($a as $v) { var_dump(current($a))}*

int(1)

int(1)

int(1)

PHP7 inside

Foreach behavior improvement

- PHP7: Doesn't use/modify array internal pointer anymore
  - *$a = array(1,2,3); foreach($a as $v) { var_dump(current($a)) }*

int(1)

int(1)

int(1)

  - *$a = array(1,2,3); $b=&$a; foreach($a as $v) { var_dump(current($a))}*

int(1)

int(1)

int(1)

  - *$a = array(1,2,3); $b=$a; foreach($a as $v) { var_dump(current($a))}*

int(1)

int(1)

int(1)

PHP7 inside

Context Sensitive Lexer

- Keywords can be used in:

  propertie/constant/method names of classes/interfaces/traits

- PHP5:

```
class Collection {
    public function foreach(callable $callback) { }
}
```

PHP Parse error: syntax error, unexpected T_FOREACH

- PHP7:

```
$projects =  Finder::for('project')
        ->where('name')->like('%secret%')
        ->and('priority', '>', 9)
        ->or('code')->in(['4', '5', '7'])
        ->and()->not('created_at')->between([$time1, $time2])
        ->list($limit, $offset)
```

PHP7 Breakages

Removals

- ereg, mysql extensions are moved to PECL
- Isapi, tux etc SAPIs are removed
- ASP(<?) and script (<script language="php") tags are removed
- Hex number string support is removed
- HTTP_RAW_POST_DATA is removed(use php://input)
- Removed support for static calls to non-static calls form an incompatible $this context
- Removed support for assigning the result of new by reference.
- Removed support for #-style comments in ini files. Use ;-style comments instead.

PHP7 inside

Behavior changes

- Same name parameters are not allowed
  - function a($b, $b)

  PHP Fatal error: Redefinition of parameter $b
- PHP4 style constructor is deprecated
  - Class foo { public function foo() {}}
- String, int, float, bool etc can not be used as classname anymore
  - class String { }

  PHP Fatal error: Cannot use 'String' as class name as it is reserved
- Function_get_args return the current state of the arg
  - function a($v) { $v++; var_dump(func_get_args()[0]); }; a(1);

  PHP5 : int(1) VS PHP7: int(2)

PHP7 inside

Behavior changes

- Same name parameters are not allowed
  - function a($b, $b)

  PHP Fatal error: Redefinition of parameter $b
- PHP4 style constructor is deprecated
  - Class foo { public function foo() {}}
- String, int, float, bool etc can not be used as classname anymore
  - class String { }

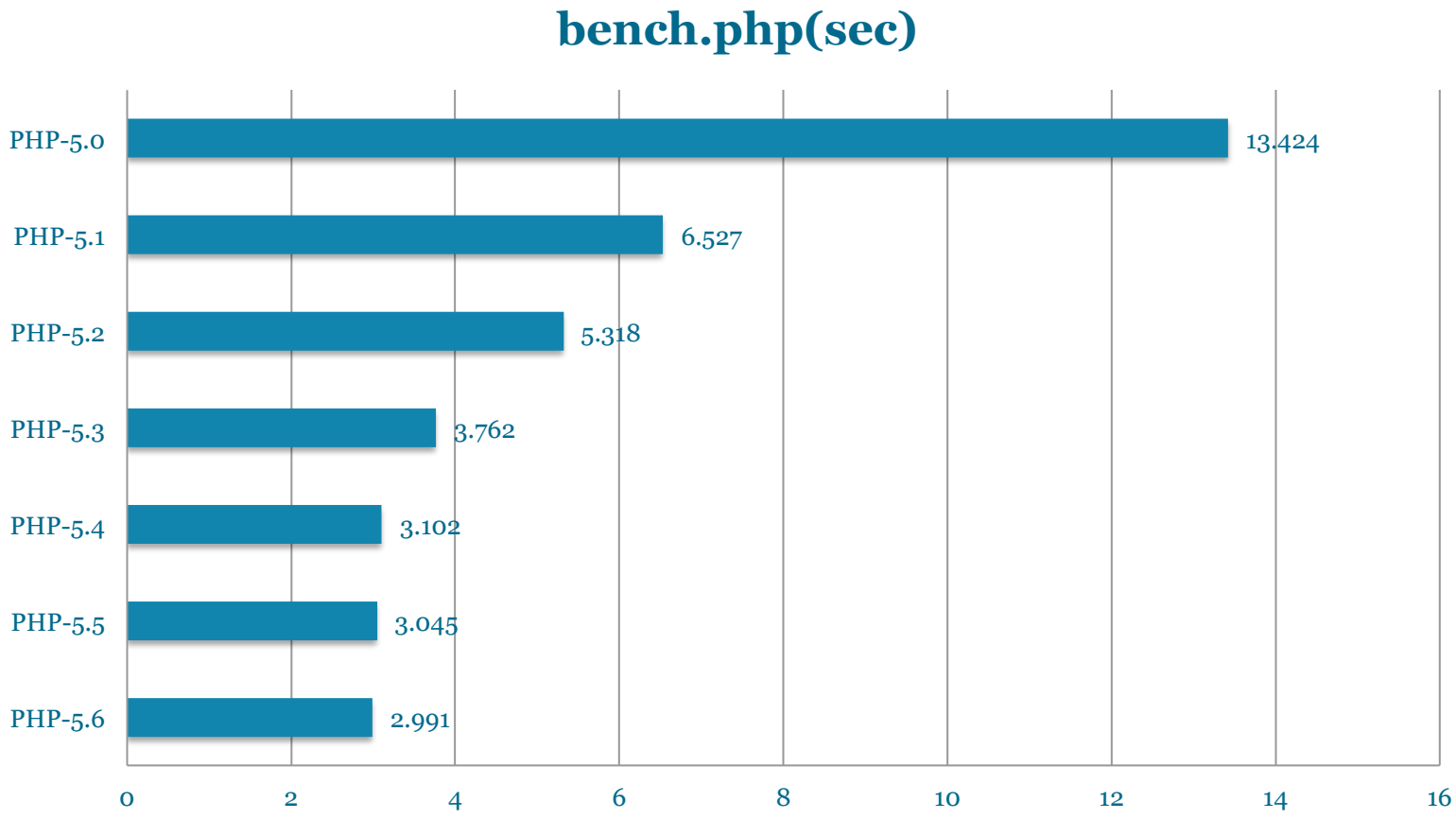
  PHP Fatal error: Cannot use 'String' as class name as it is reserved
- Function_get_args return the current state of the arg
  - function a($v) { $v++; var_dump(func_get_args()[0]); }; a(1);

  PHP5 : int(1) VS PHP7: int(2)

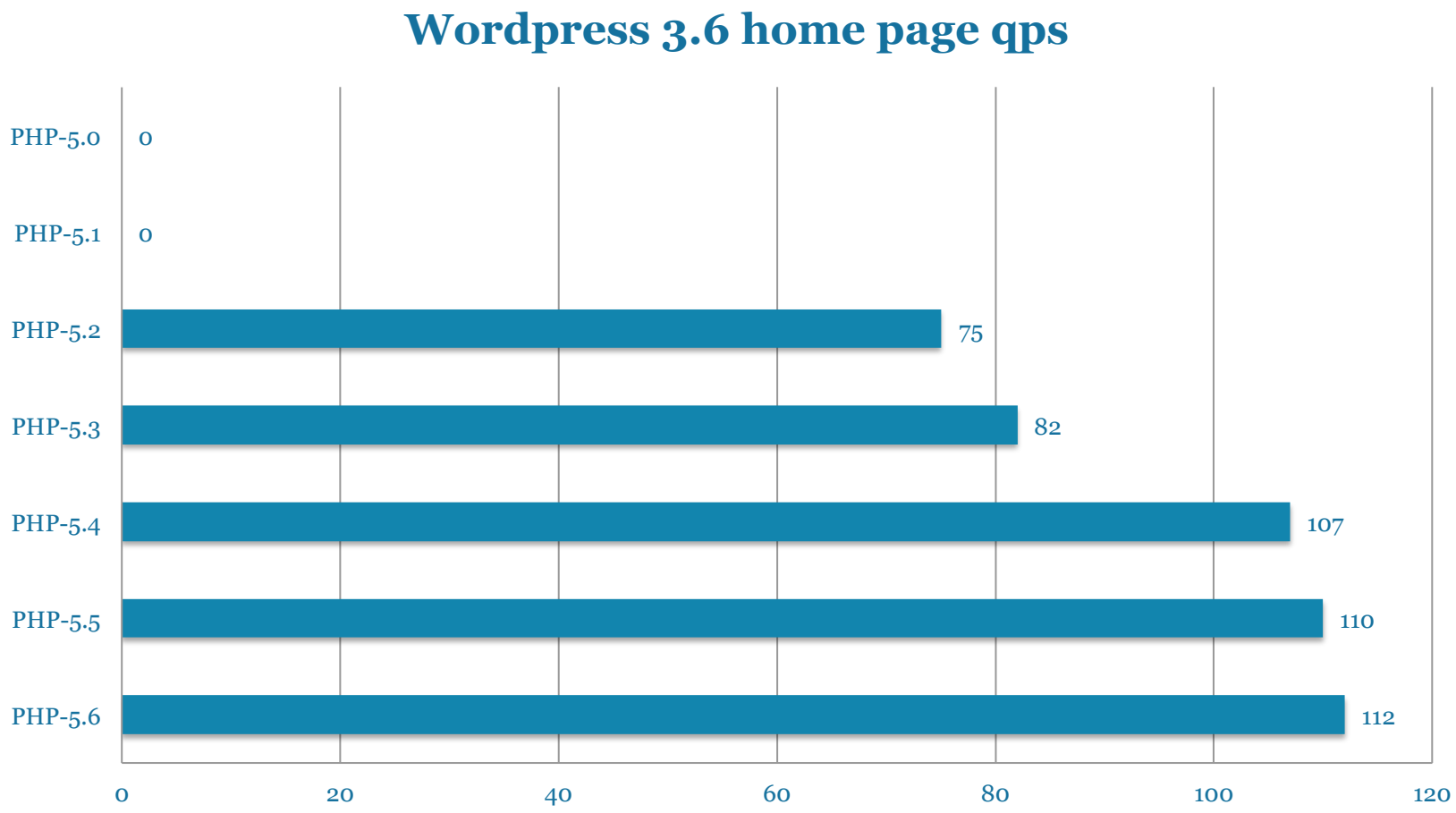
PHP7 inside

PHP7 Performance

PHP Performance Evaluation



PHP Performance Evaluation

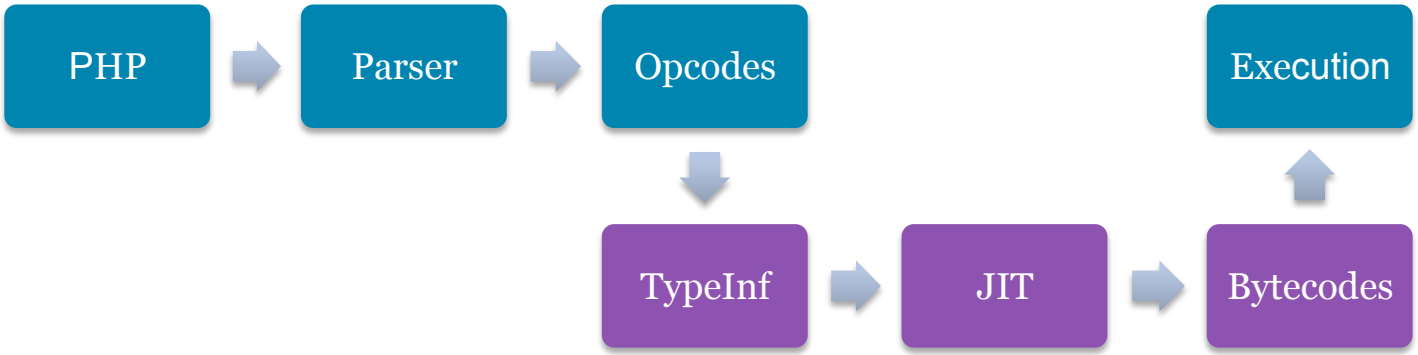


PHP Performance Evaluation

- ~5 times faster from 5.0 to 5.6 in bench
- ~2 times faster from 5.0 to 5.6 in real-life apps
- No big performance improvement after 5.4
- Zend VM is already highly optimized

PHP Just In Time Compiler?

- Generate optimized codes based on run-time types inference

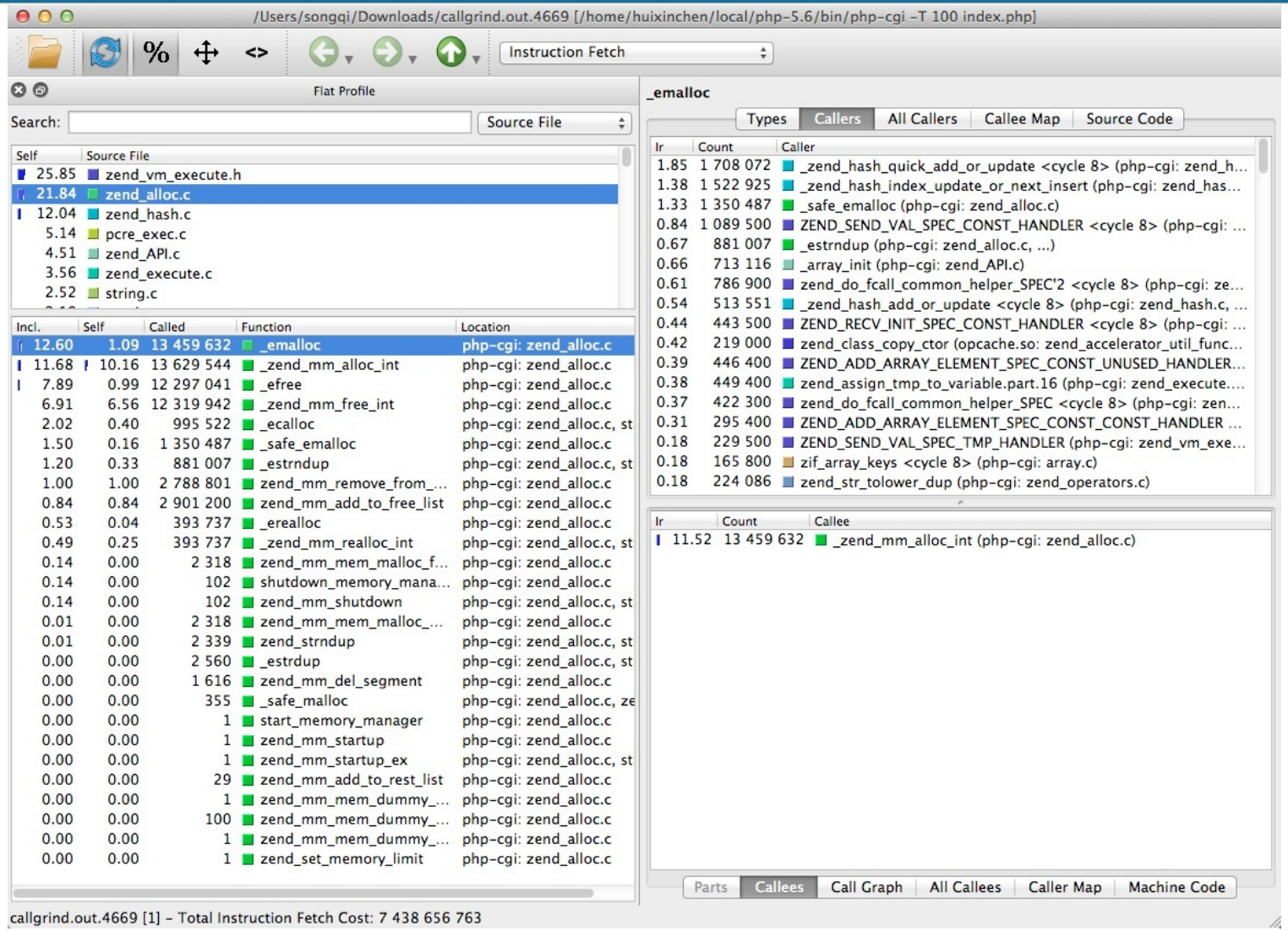


PHP Just In Time Compiler?

- We created a POC of JIT compiler based on LLVM for PHP-5.5 in 2013
- ~8 times speedup on bench.php
- Negligible speedup on real-life apps (1% on Wordpress)
- https://github.com/zendtech/php-src/tree/zend-jit

A	B	E	F
bench.php	**PHP 5.5**	**PHP 5.5 + JIT(24 Aug)**	**hhvm**
simple	0.142	0.005	0.008
simplecall	0.165	0.001	0.003
simpleucall	0.142	0.001	0.010
simpleudcall	0.151	0.001	0.010
mandel	0.389	0.020	0.068
mandel2	0.440	0.044	0.085
ackermann	0.164	0.048	0.013
ary(50000)	0.023	0.013	0.008
ary2(50000)	0.019	0.012	0.009
ar3(2000)	0.203	0.038	0.102
fibo(30)	0.468	0.017	0.026
hash1(50000)	0.041	0.024	0.036
hash2(500)	0.043	0.029	0.023
heapsort(20000)	0.122	0.040	0.045
matrix(20)	0.110	0.033	0.038
nestedloop(12)	0.236	0.008	0.015
sieve(30)	0.121	0.058	0.027
strcat(200000)	0.017	0.012	0.006
Total	2.996	0.404	0.532

Wordpress profile



Wordpress profile

- 21% CPU time in memory manager
- 12% CPU time in hash tables operations
- 30% CPU time in internal functions
- 25% CPU time in VM

New Generation

- It's a refactoring
- Main goal — achieve new performance level and make base for future improvements
- No new features for users (only internals)
- Keep 100% compatibility in PHP behavior
- May 2014 we opened the project

PHP7 inside

ZVAL

- Zval is changed

```
struct _zval_struct {
    union {
         long lval;
         double dval;
         struct {
             char *val;
             int len;
         } str;
         HashTable *ht;
         zend_object_value obj;
         zend_ast *ast;
     } value;
     zend_uint refcount__gc;
     zend_uchar type;
     zend_uchar is_ref__gc;
};


sizeof(zval) == 24
```

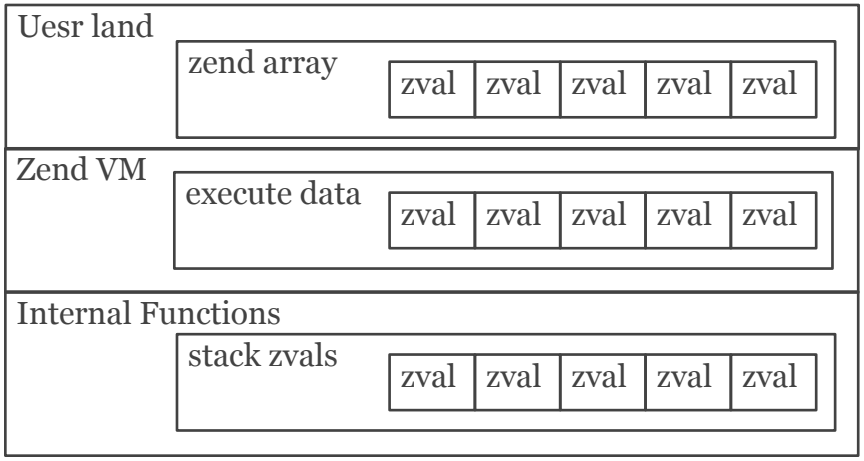
```
struct _zval_struct {
    union {
         long                lval;
         double              dval;
         zend_refcounted  *counted;
         zend_string      *str;
         zend_array       *arr;
         zend_object      *obj;
         zend_resource    *res;
         zend_reference   *ref;
         zend_ast_ref     *ast;
         zval             *zv;
         void             *ptr;
         zend_class_entry *ce;
         zend_function    *func;
    } value;
    union {
        struct {
            ZEND_ENDIAN_LOHI_4(
                zend_uchar    type,
                zend_uchar    type_flags,
                zend_uchar    const_flags,
                zend_uchar    reserved)
        } v;
        zend_uint type_info;
    } u1;
    union {
        zend_uint     var_flags;
        zend_uint     next;
        zend_uint     str_offset;
        zend_uint     cache_slot;
    } u2;
};

sizeof(zval) == 16
```

PHP7 inside

ZVAL

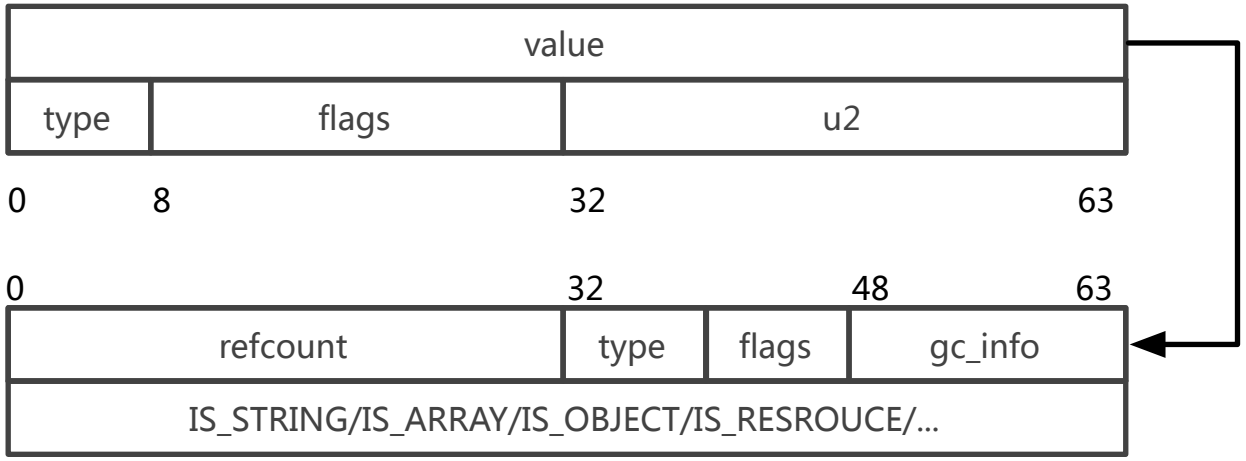
- No refcount for scalar types
- zvals are always pre-allcocated or allocated in stack(nomore MAKE_STD_ZVAL and ALLOC_ZVAL)
- String using refcout instead of copy (zend_string)
- gc_info, temporary_variables, should_free_var, cache_slot all in zval
- New types: IS_TRUE, IS_FALSE, IS_REFERENCE, IS_INDIRECT



PHP7 inside

ZVAL

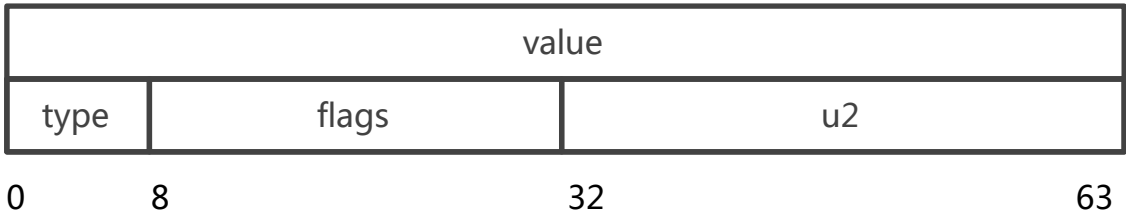
- IS_UNDEF
- IS_NULL
- IS_FALSE
- IS_TRUE
- IS_LONG
- IS_DOUBLE
- IS_STRING
- IS_ARRAY
- IS_OBJECT
- IS_RESOURCE
- IS_REFERENCE



PHP7 inside

ZVAL NON REFCOUNED

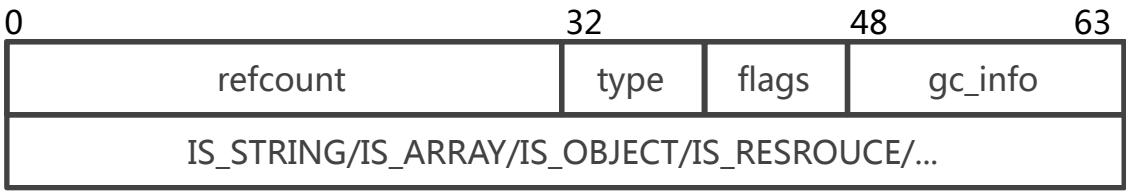
- IS_NULL
- IS_FALSE
- IS_TRUE
- IS_LONG
- IS_DOUBLE



PHP7 inside

ZVAL REFCOUNED

- IS_STRING
- IS_ARRAY
- IS_OBJECT
- IS_RESOURCE
- IS_REFERENCE



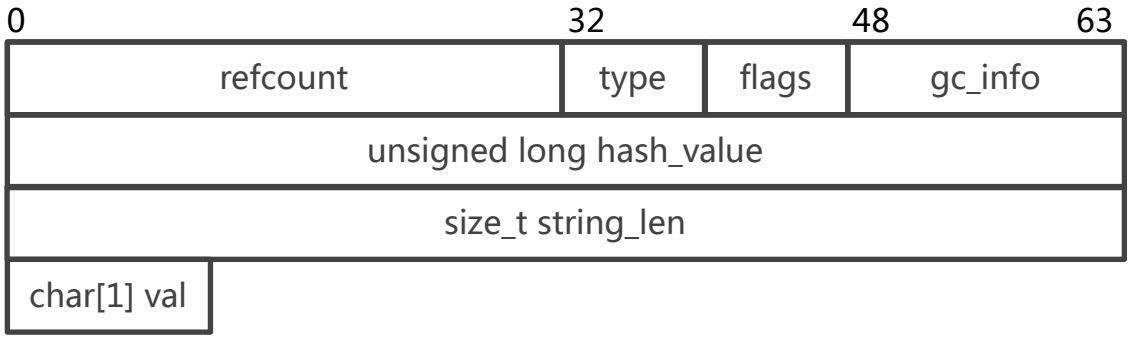
PHP7 inside

IS_STRING

- New Internal Type: Zend String

```
struct _zend_string {
    zend_refcounted   gc;
    zend_ulong        h;
    size_t            len;
    char              val[1]
};
```

- IS_STRING_PERSISTENT
- IS_STR_INTERNED
- IS_STR_PERMANENT
- IS_STR_CONSTANT

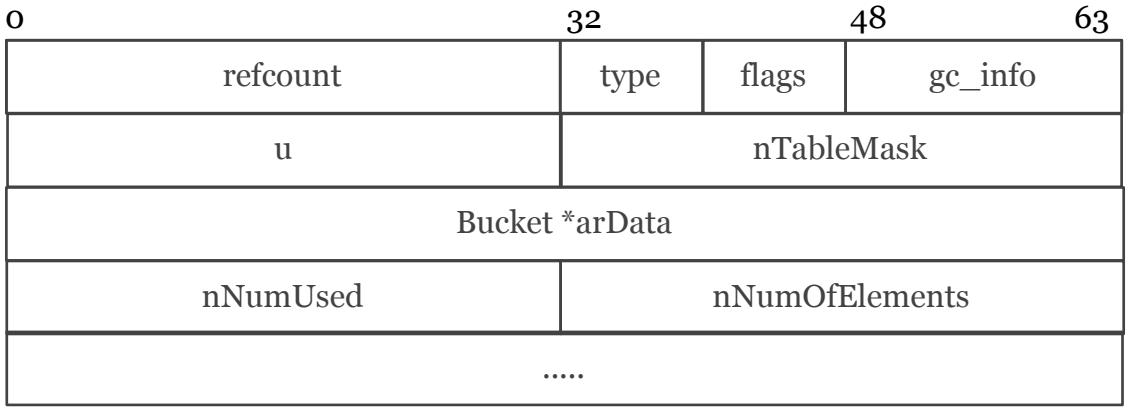


PHP7 inside

IS_ARRAY

- New Internal Type: Zend Array

```
struct _zend_array {
    zend_refcounted   gc;
    union {
        struct {
            ZEND_ENDIAN_LOHI_4(
                zend_uchar    flags,
                zend_uchar    nApplyCount,
                zend_uchar    nIteratorsCount,
                zend_uchar    reserve)
        } v;
        uint32_t flags;
    } u;
    uint32_t          nTableMask;
    Bucket           *arData;
    uint32_t          nNumUsed;
    uint32_t          nNumOfElements;
    uint32_t          nTableSize;
    uint32_t          nInternalPointer;
    zend_long         nNextFreeElement;
    dtor_func_t       pDestructor;
};
```



- IS_ARRAY_IMMUTABLE

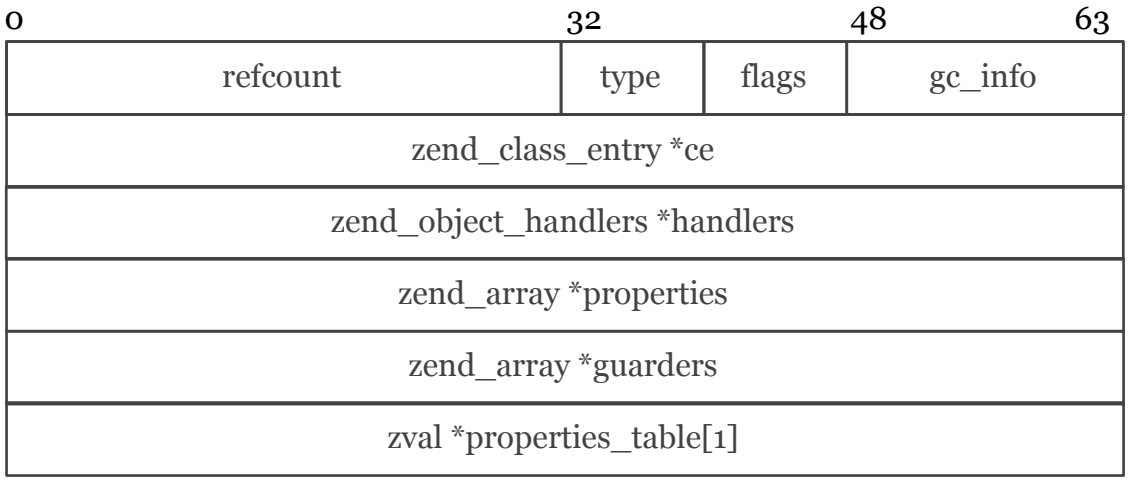
PHP7 inside

IS_OBJECT

- Zend Object

```
struct _zend_object {
    zend_refcounted   gc;
    uint32_t          handle; // TODO: may be
    zend_class_entry *ce;
    const zend_object_handlers *handlers;
    HashTable         *properties;
    HashTable         *guards; /* protects fro
    zval              properties_table[1];
};
```

- IS_OBJ_APPLY_COUNT
- IS_OBJ_DESTRUCTOR_CALLED
- IS_OBJ_FREE_CALLED

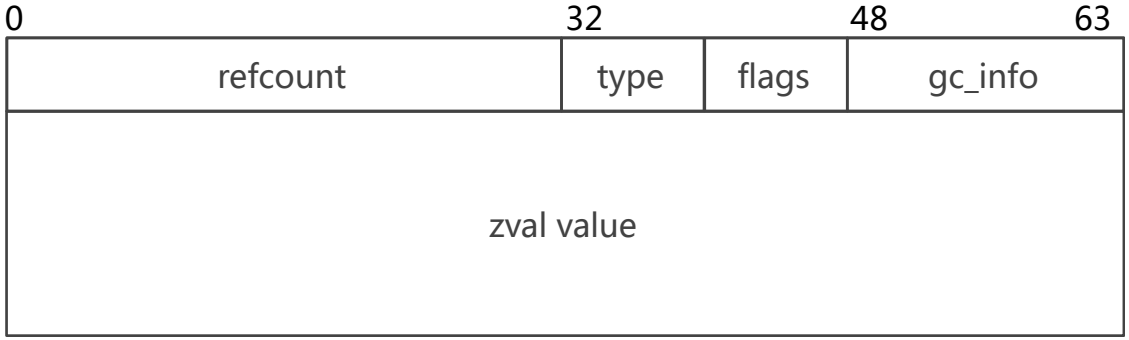


PHP7 inside

IS_REFERENCE

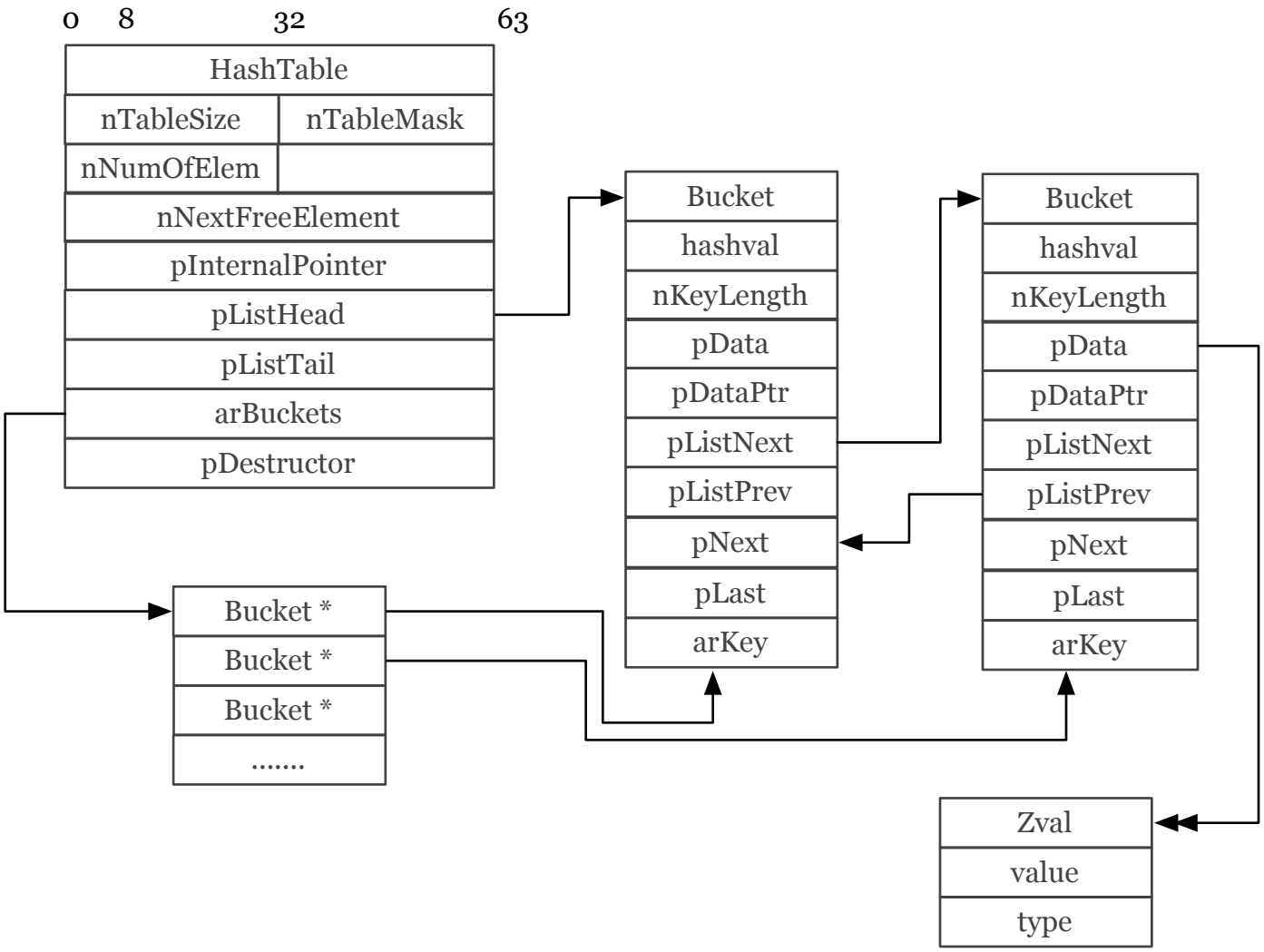
- New Internal Type: Zend Reference
- Reference is type

```
struct _zend_reference {
    zend_refcounted   gc;
    zval              val;
};
```

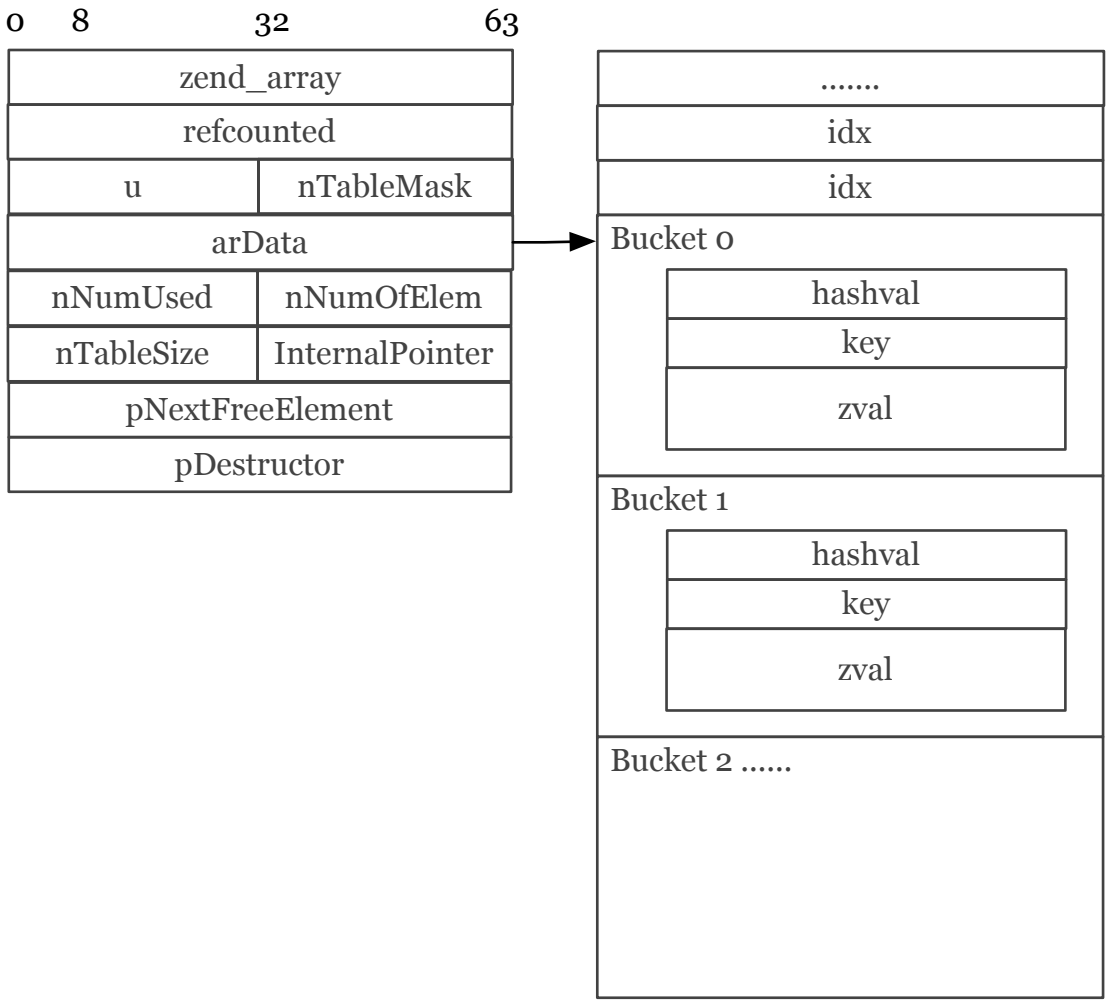


PHP7 inside

HashTable – PHP 5



Zend Array – PHP 7

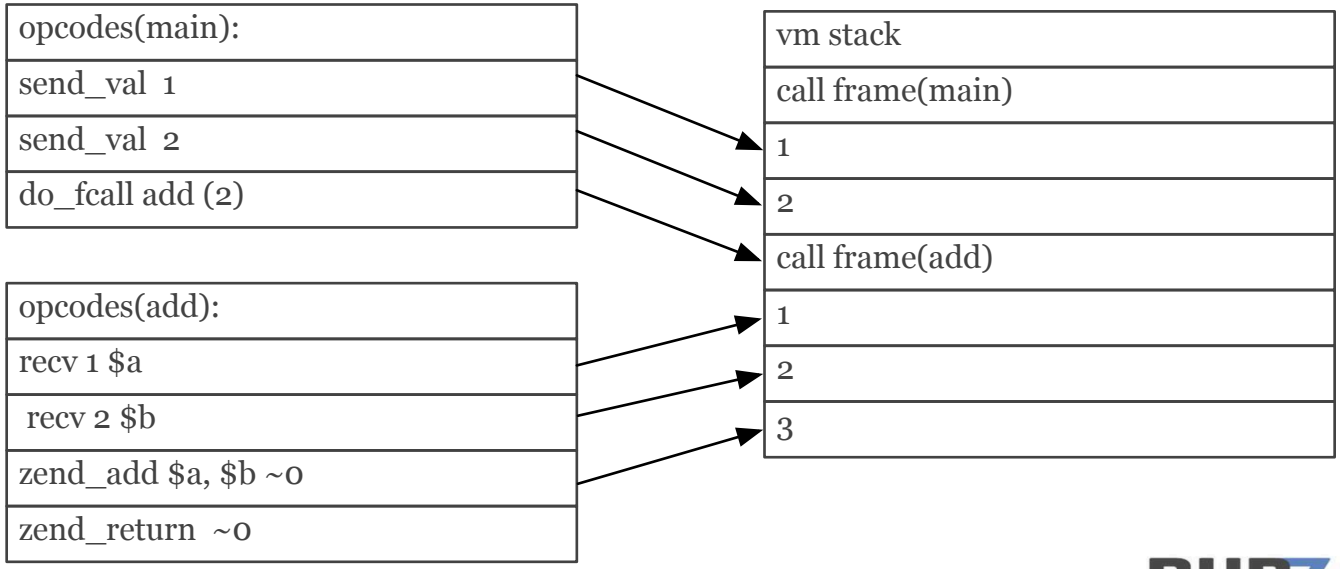


Zend Array(HashTable)

- Values of arrays are zval by default
- HashTable size reduced from 72 to 56 bytes
- Bucket size reduced from 72 to 32 bytes
- Memory for all Buckets is allocated at once
- Bucket.key now is a pointer to zend_string
- Values of array elements are embedded into the Buckets
- Improved data locality => less CPU cache misses

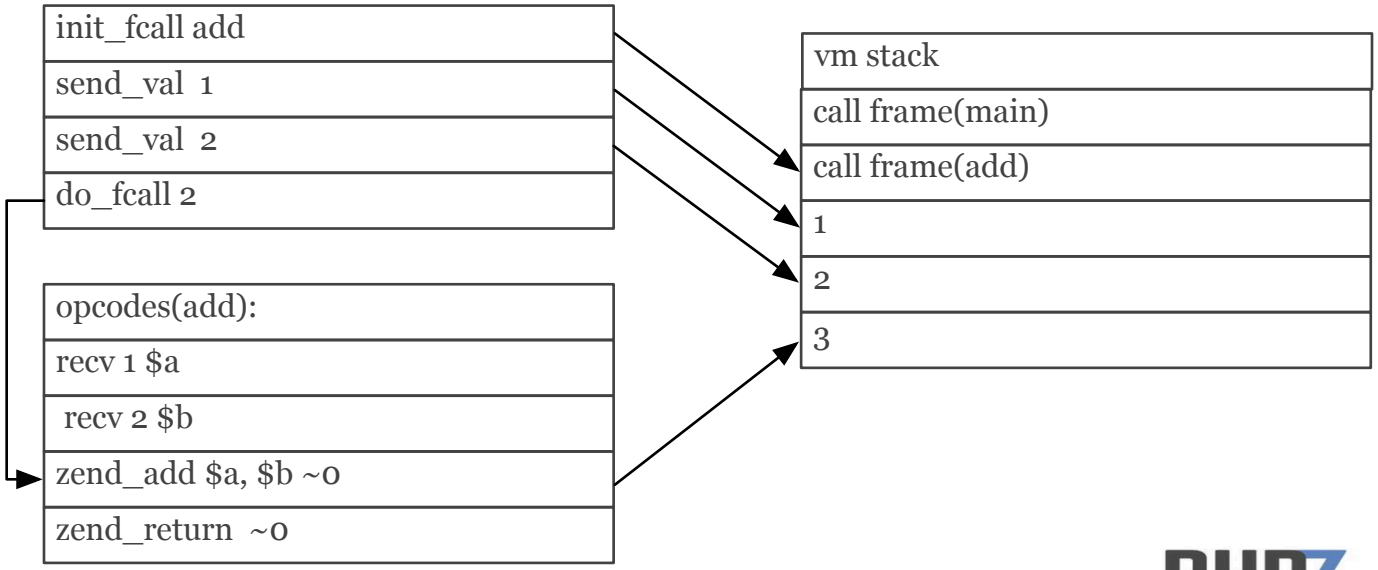
Function calling convention – PHP5

- function add ($a, $b) {
  return $a + $b;
  }
  add(1, 2);



Function calling convention – PHP7

- function add ($a, $b) {
  return $a + $b;
  }
  add(1, 2);



Fast Parameters Parsing APIs

- ~5% of the CPU time is spent in zend_parse_parameters()
- For some simple functions the overhead of zend_parse_parameters() is over 90%

```
if (zend_parse_parameters(ZEND_NUM_ARGS()
    TSRMLS_CC, "za|b",
    &value, &array, &strict) == FAILURE) {
    return;
}
```

```
ZEND_PARSE_PARAMETERS_START()
    Z_PARAM_ZVAL(value)
    Z_PARAM_ARRAY(array)
    Z_PARAM_OPTIONAL
    Z_PARAM_BOOL(strict)
ZEND_PARSE_PARAMETERS_END();
```

PHP7 inside

Inline Frequently used simple functions

- call_user_function(_array) => ZEND_INIT_USER_CALL
- is_int/string/array/* etc => ZEND_TYPE_CHECK
- strlen => ZEND_STRLEN
- defined => ZEND+DEFINED
- ...

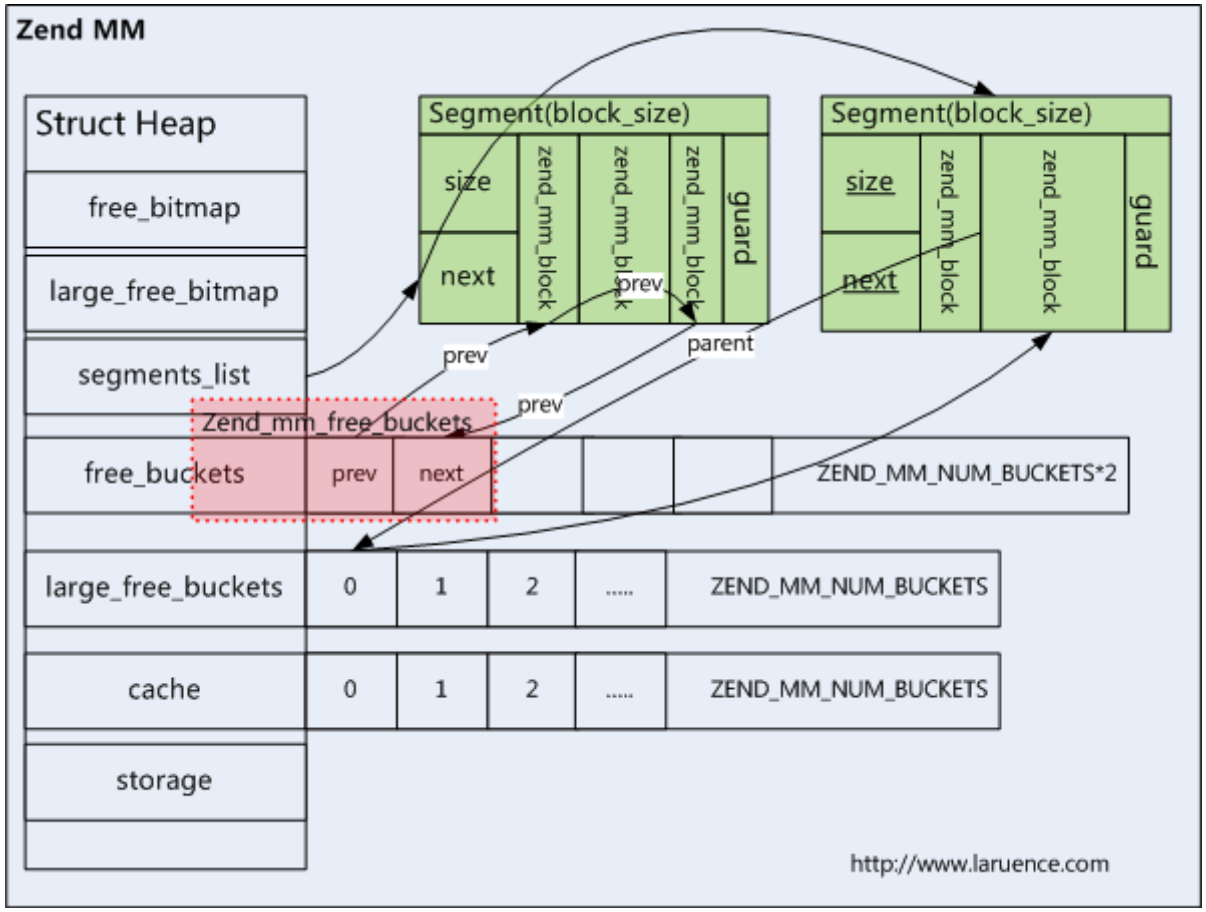
PHP7 inside

Faster zend_qsort

- Refactor zend_qsort for better performance
- Hybrid Sorting Algo(Quick Sort and Selection Sort)
- <16 elements do stable sorting
- $array = array(0 => 0, 1=>0); asort($array);
  - *PHP5: array(1=>0, 0=>0);*
  - *PHP7: array(0=>1, 1=>0);*

PHP7 inside

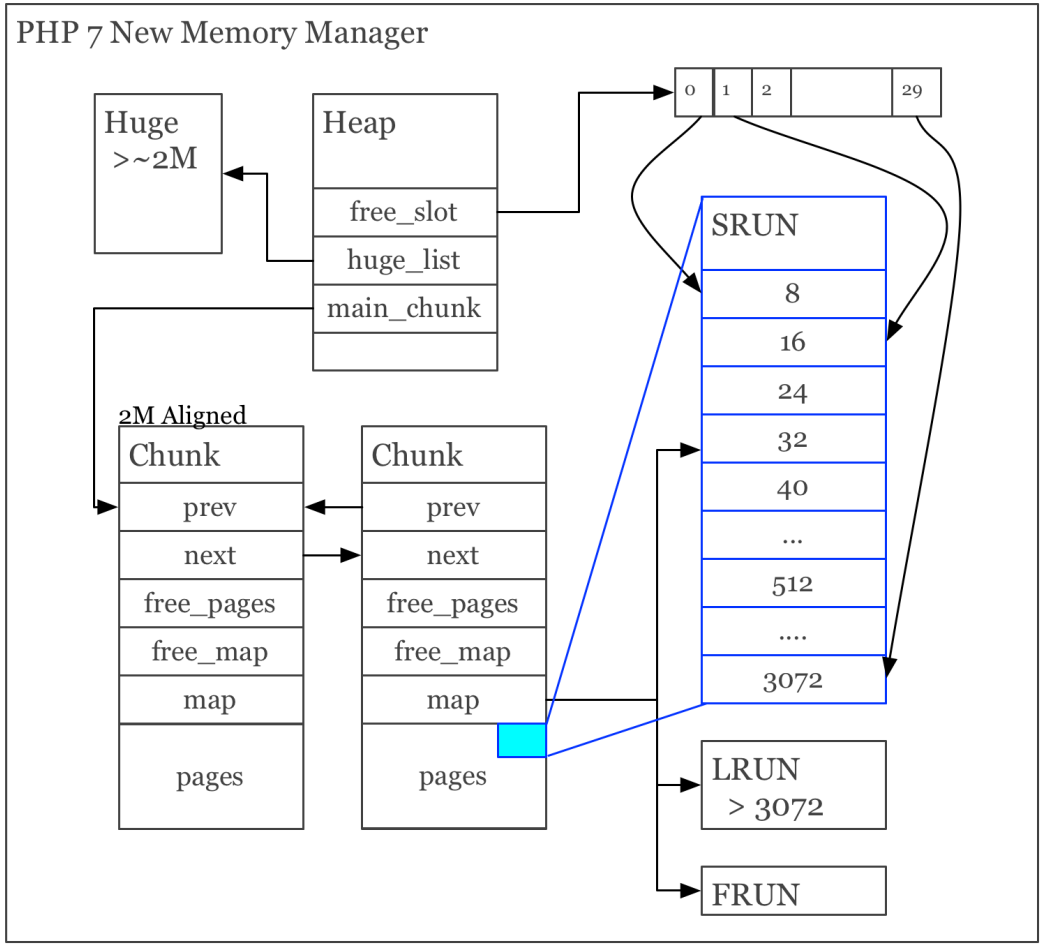
Memory Manager PHP5



COMPATIBLE

PHP7 inside

Memory Manager PHP7



New Memory Manager

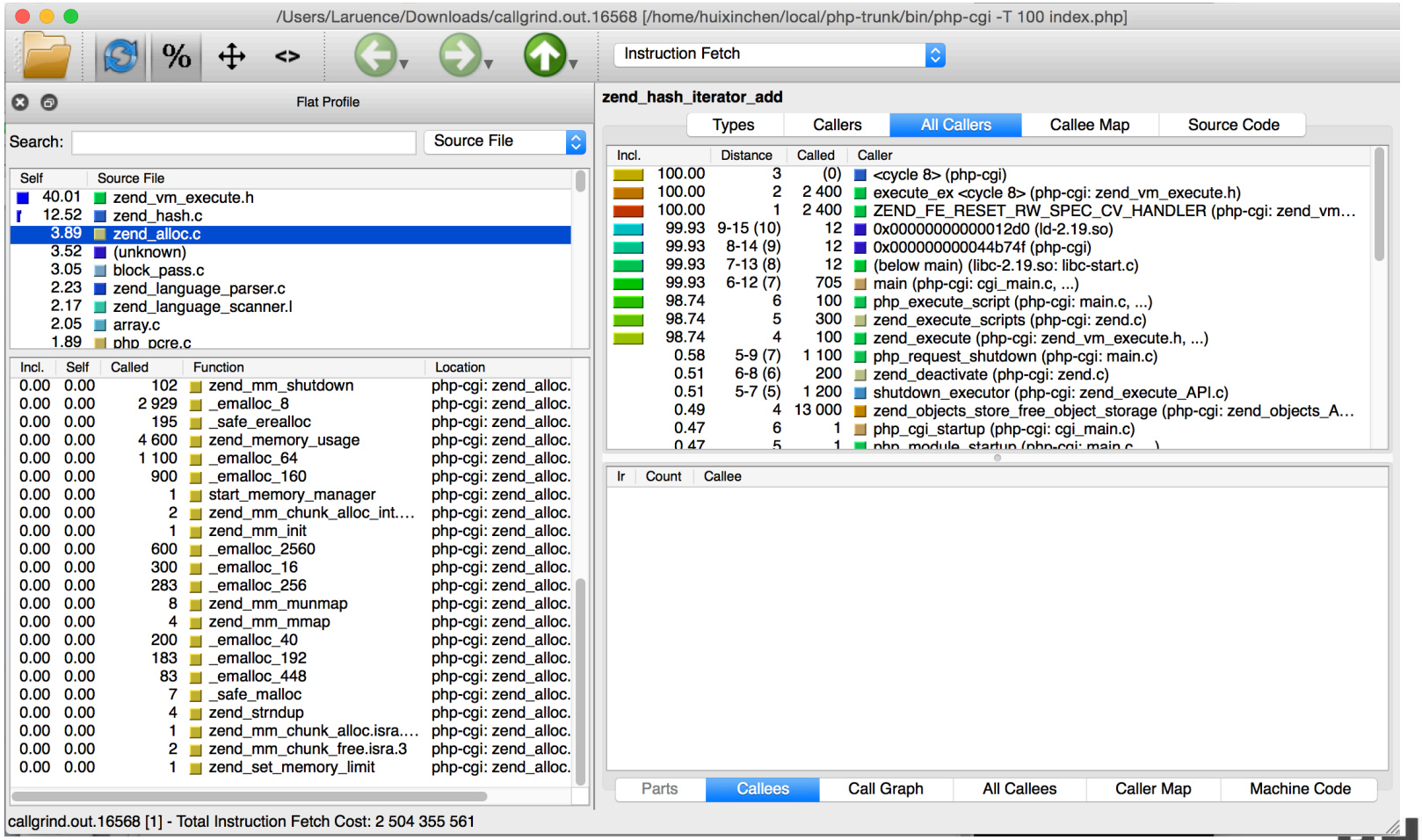
- Less memory wasting
- Friendly to moder CPU cache
- less CPU cache misses
- Faster builtin types allocating
- ~5% CPU time reduce in wordpress homepage

PHP7 inside

Dozens of other improvements

- zend_op size reduced from 48 to 32 (x86_64)
- Fast HashTable iteration APIs
- Immutable array
- Array duplication optimization
- PCRE with JIT
- BIND_GLOBAL instead of FETCH and ASSIGN_REF
- More specifical DO_UCALL and DO_ICALL
- Global registers for execute_data and opline(GCC-4.8+)
- ZEND_ROPE_* for faster string concating
- ZEND_CALL_TRAMPOLINE for faster__call/__callstatic
- ....

Wordpress profile (2015-04-14)

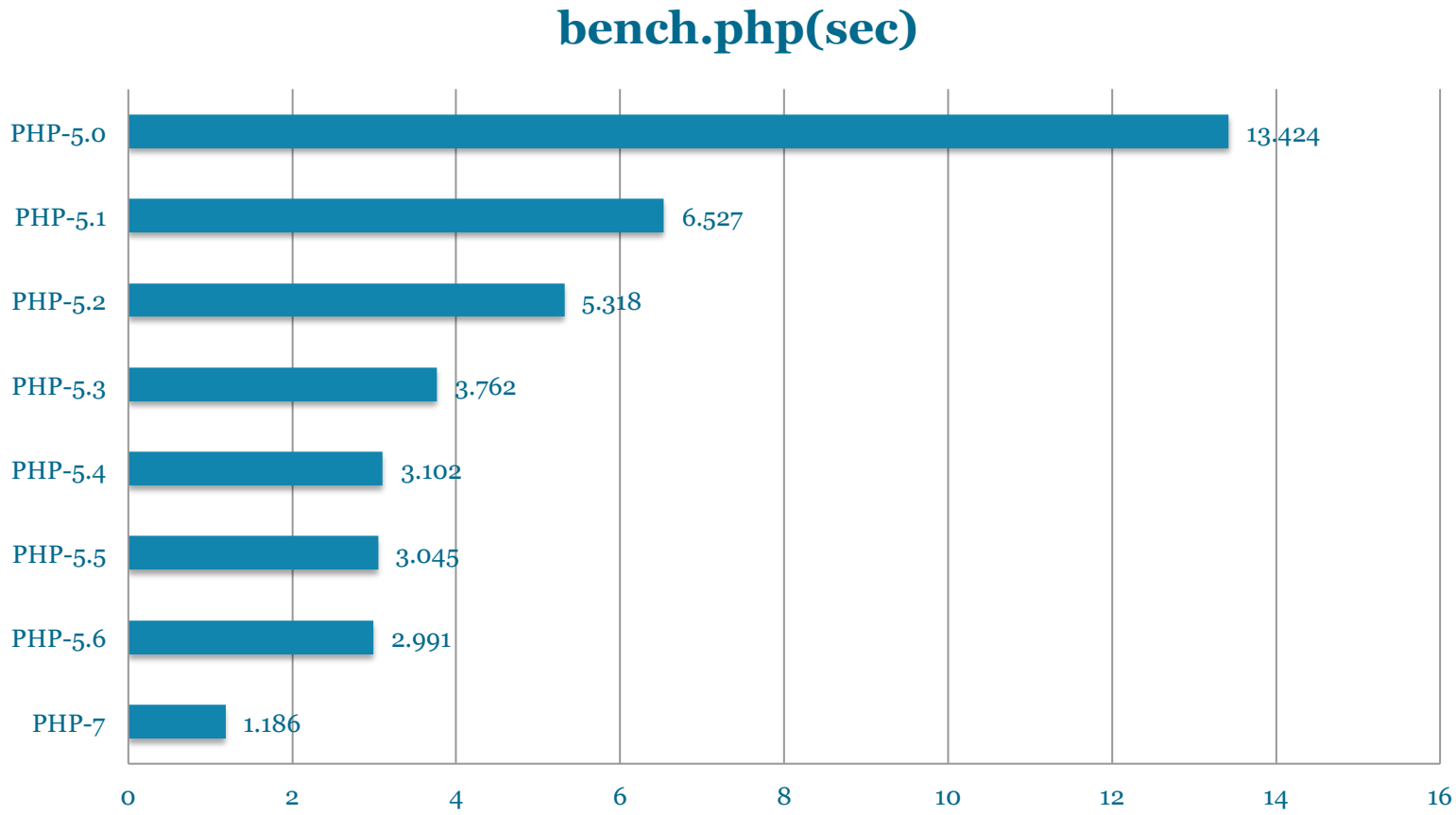


Wordpress profiled (2015-04-14)

- >50% CPU IRs reduced
- 5% CPU time in meory manager
- 12% CPU time in hash tables operations

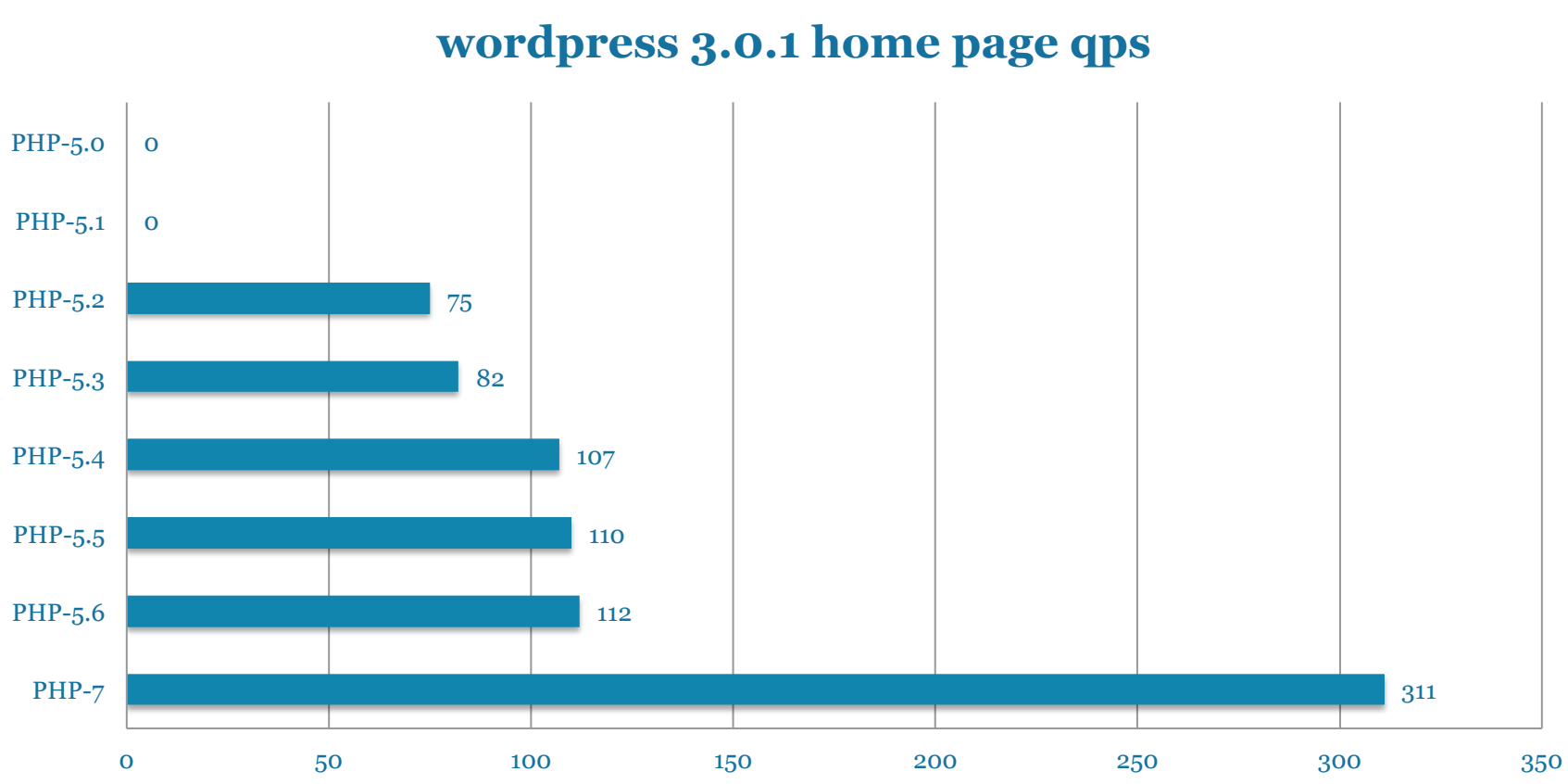
PHP7 inside

PHP7 Performance – Benchmark (2014-04-14)



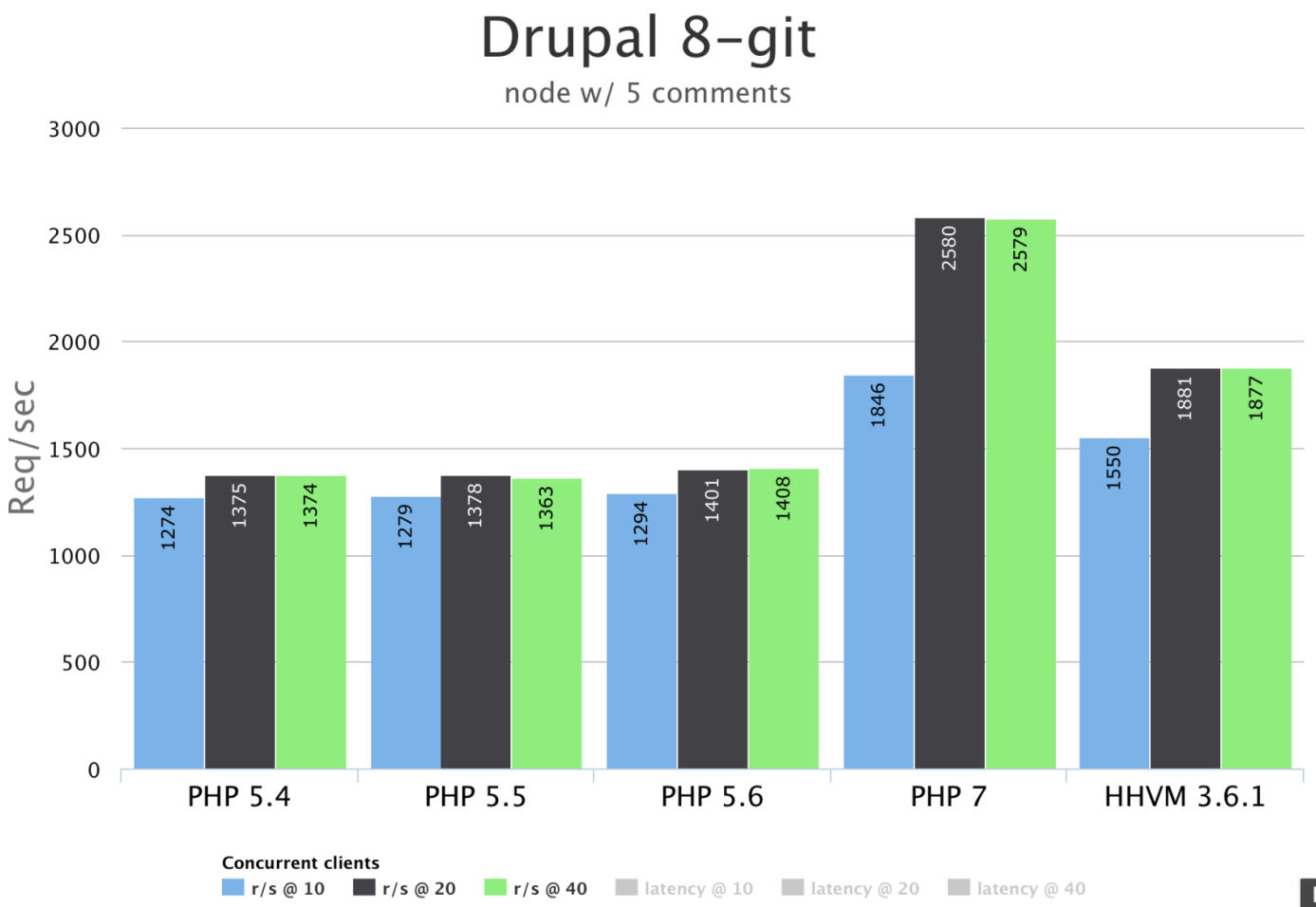
PHP7 inside

PHP7 Performance – Reallife App (2015-04-14)



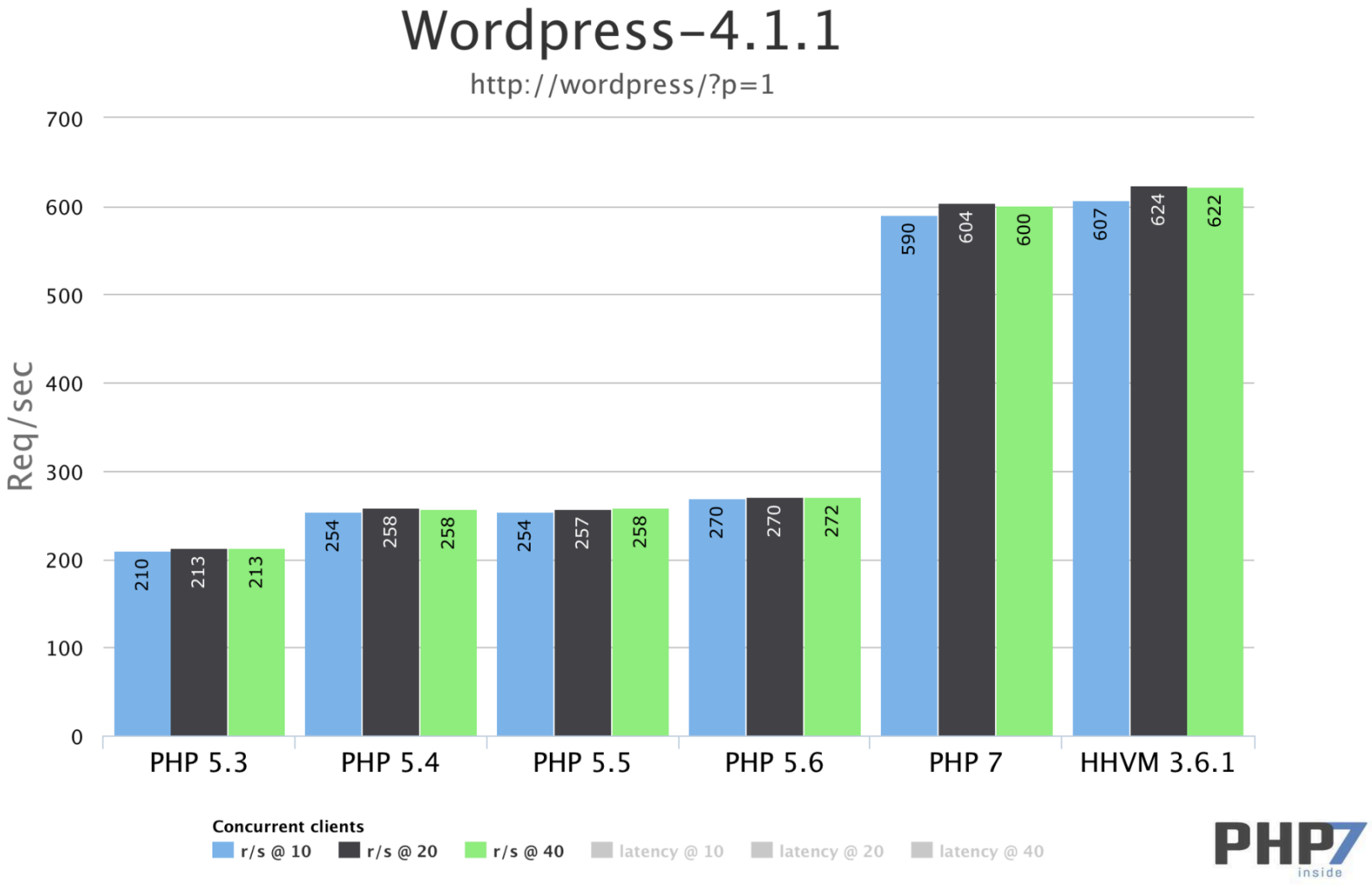
PHP7 inside

PHP7 Performance (By Rasmus 2015-04-21)

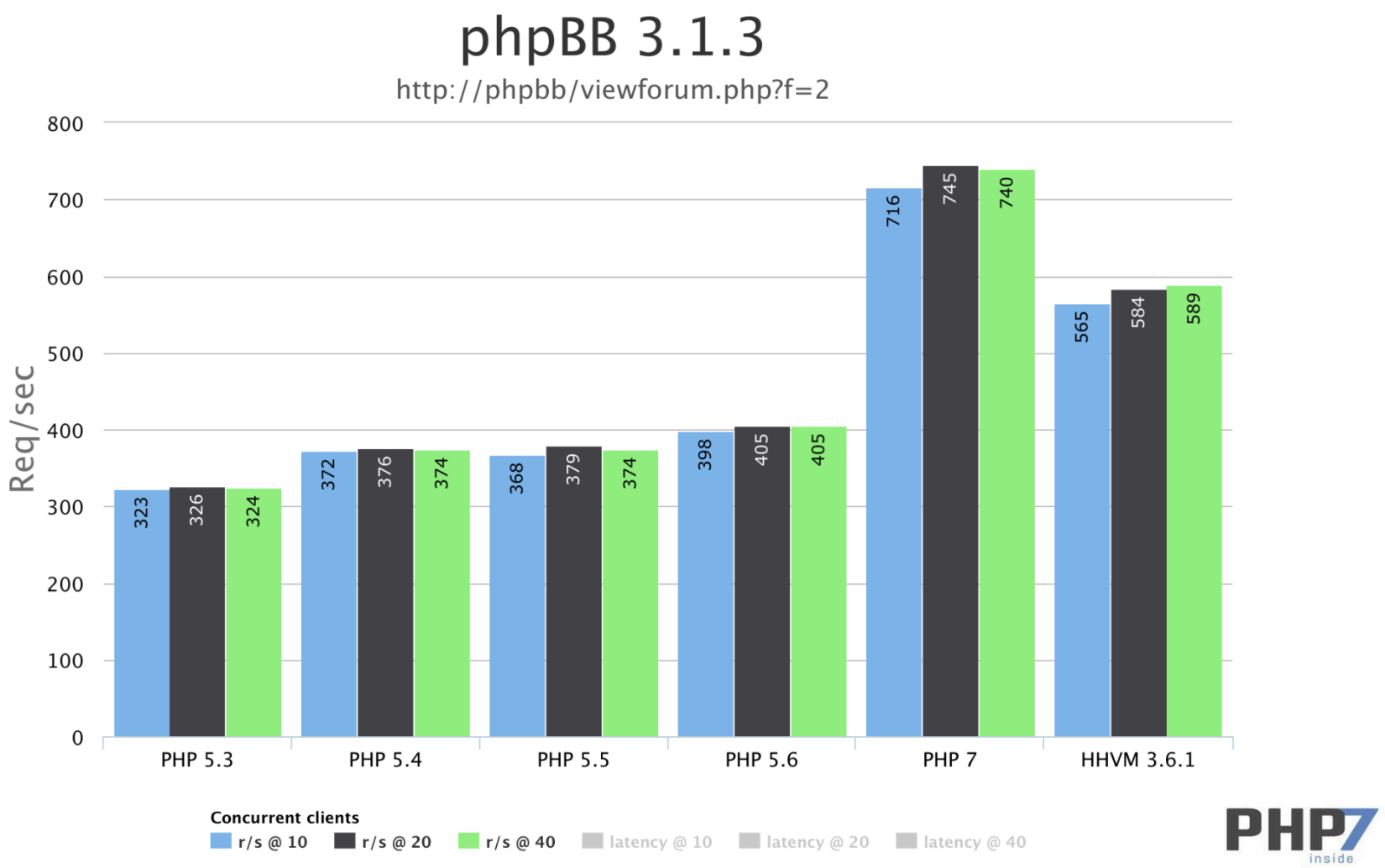


PHP7 inside

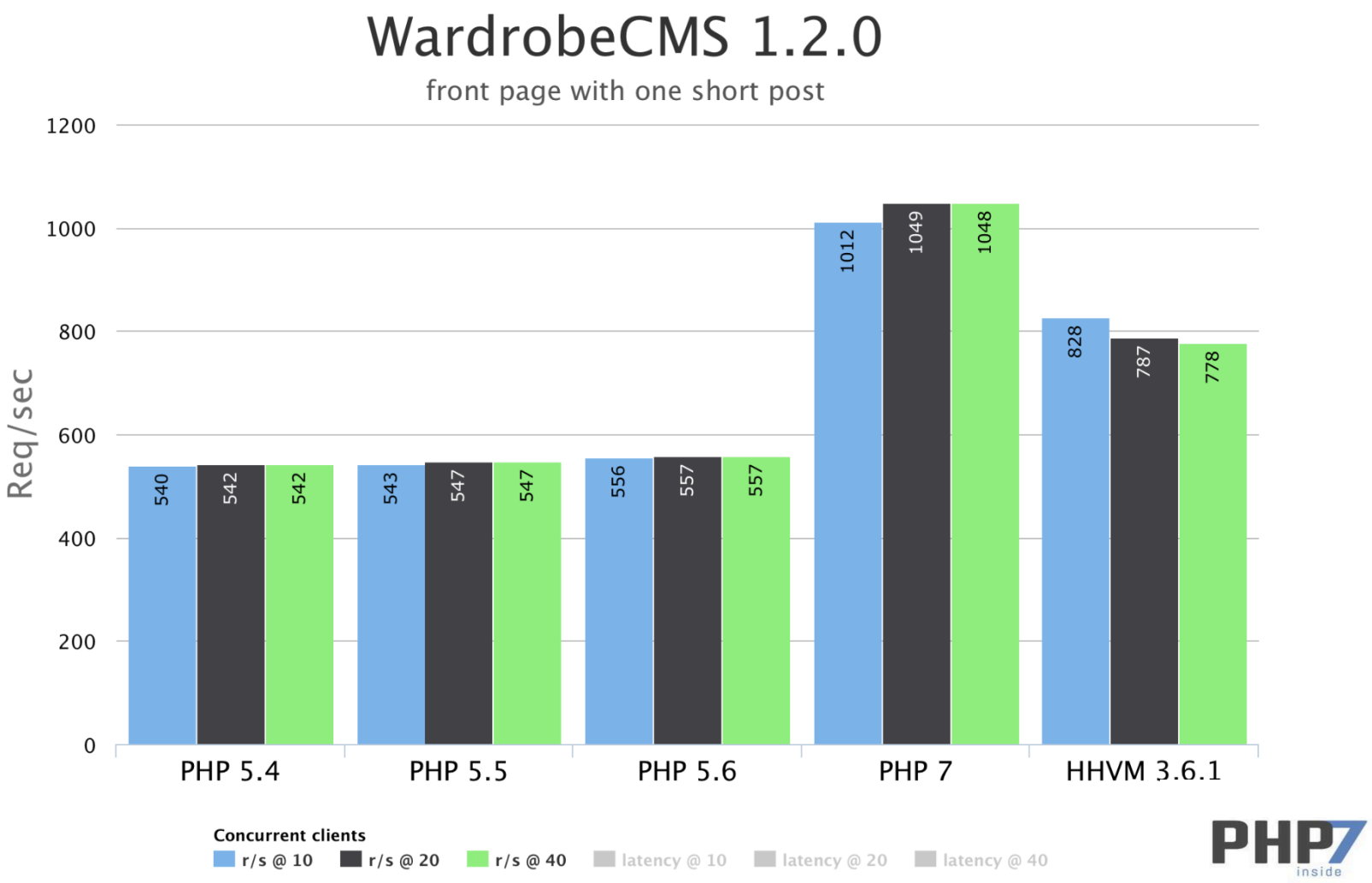
PHP7 Performance (2015-04-21)



PHP7 Performance (2015-04-21)



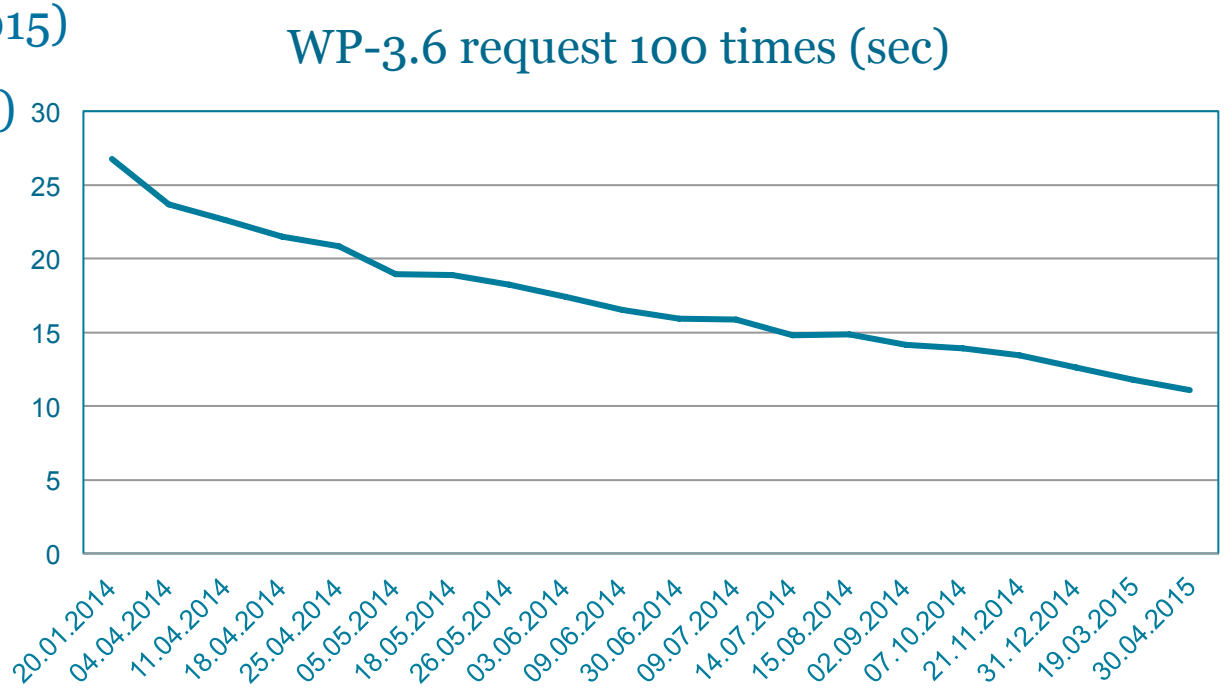
PHP7 Performance (2015-03-15)



Always Do Your Own Benchmark

PHP 7 Next

- Keep 99.99% compatible with PHP5
- Keep Improving performance
- Port common used PECL extensions (memcached, redis etc)
- Release PHP 7 (oct 2015)
- Restart JIT (Sep 2014)



Links

- phpng:_Refactored_PHP_Engine_with_Big_Performance_Improveme nt: http://news.php.net/php.internals/73888
- PHPNG RFC: https://wiki.php.net/phpng
- PHPNG Implementation details: https://wiki.php.net/phpng-int
- Upgrading PHP extensions from PHP5 to PHPNG: https://wiki.php.net/phpng-upgrading
- Zeev <Benchmarking PHPNG>: http://zsuraski.blogspot.co.il/2014/07/benchmarking-phpng.html
- Rasmus <SPEEDING UP THE WEB WITH PHP 7>: http://talks.php.net/fluent15#/

PHP7 inside

Questions?

Thanks