

**Think**  
ThinkinLAMP.com

**PHPCON**  
JUL11-12  
SHANGHAI CHINA

2015 PHPCON SHANGHAI



# PHP 7

- All about New Features and Performance

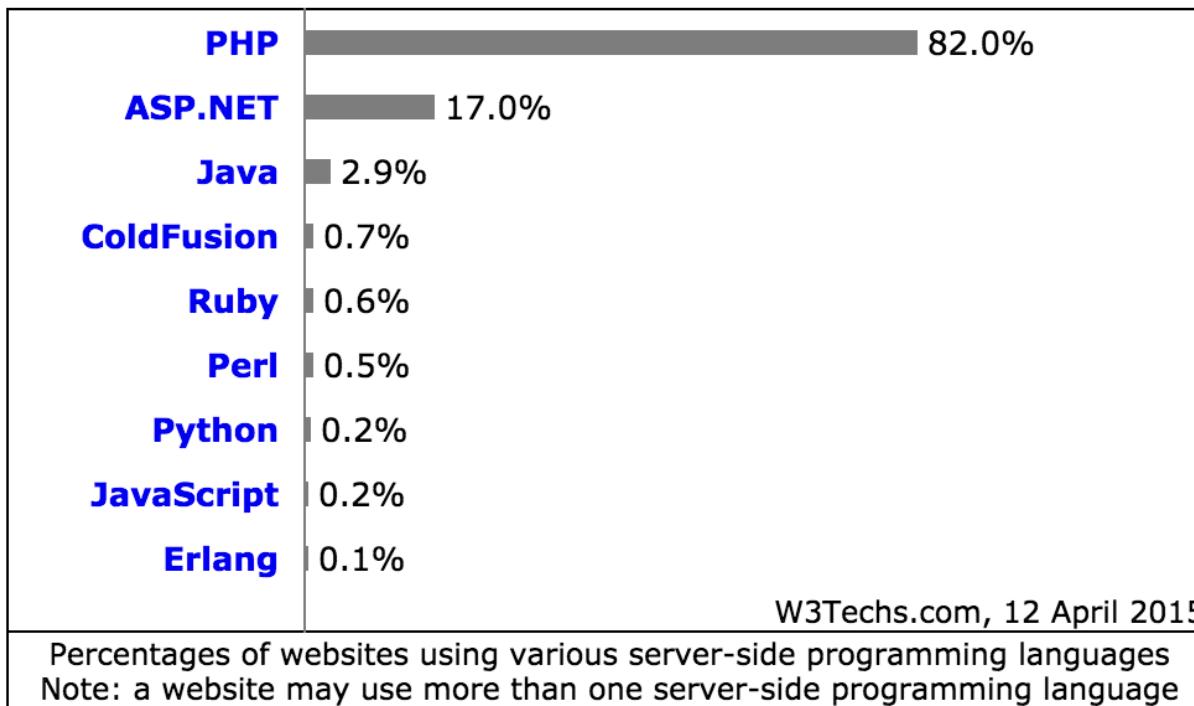
@laruence

# 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

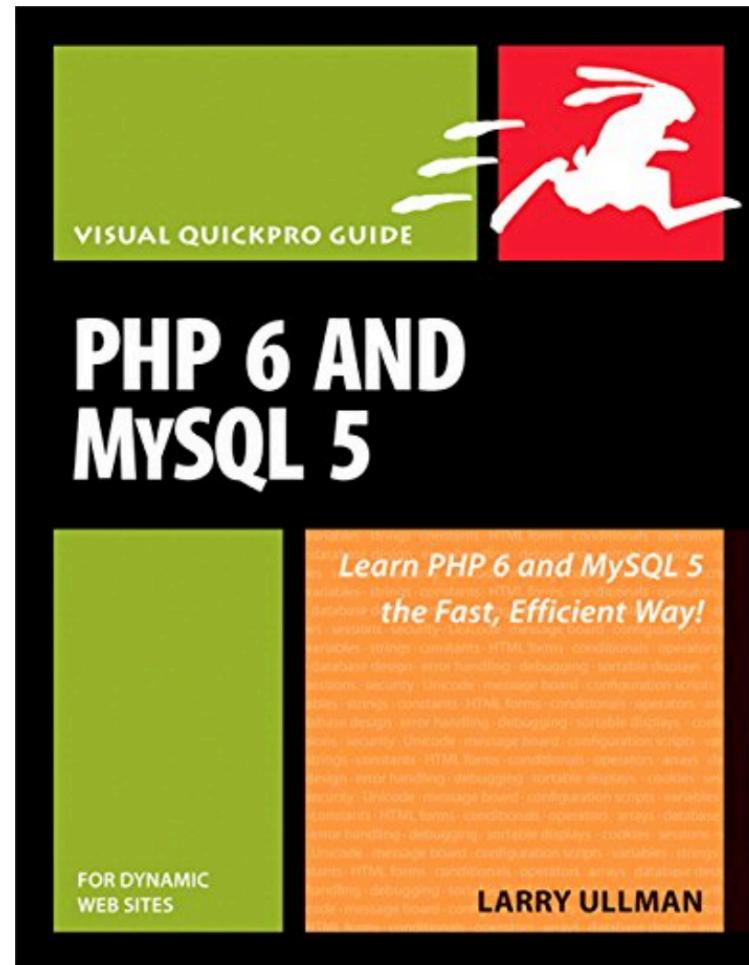
- 20 years history
- Most popular Web service program language
- Over 82% sites are use PHP as server program language



# 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 PHP6?



# PHP 7 New Features

---

# PHP 7?

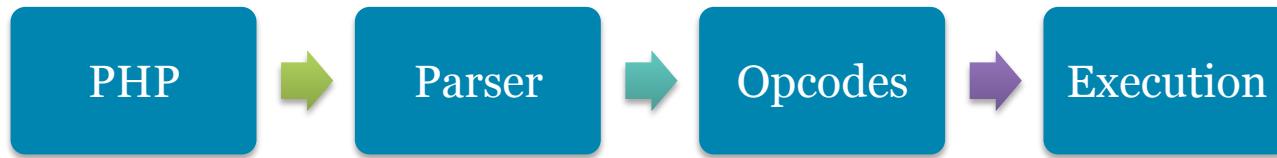
- PHP NG – Engine Refactor(Zend Engine 3) - performance improvements
- Abstract Syntax Tree
- Int64 Improvement
- Uniform variable syntax
- Native TLS
- Consistently foreach behaviors
- New <=>, ?? operators
- Return Type Declarations
- Scalar Type Declarations
- Exceptions in Engine
- And Dozens features...



# Abstract Syntax Tree



- PHP5



- PHP7



# Int64 Improvement



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

Platform	string size	signed integer
	<i>int</i>	<i>long</i>
LP64	32 bit	64 bit
LLP64	32 bit	32 bit
ILP64	64 bit	64 bit

# Uniform Variables Syntax



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

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

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

# New Operators



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

# Unicode Codepoint Escape Syntax



- \u{xxxxx}
  - echo "\u{4f60}\u{597d}, \u{66f4}\u{52ao}\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*

# 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( Throwable, Error)

```
try {  
    non_exists_func();  
} catch (Error $e) {  
    echo "Exception: {$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()*

# 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)*



# 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)*



# 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 ("ox10" != "16")
- HTTP\_RAW\_POST\_DATA is removed(use php://input)
- Removed support for static calls to non-static calls from an incompatible \$this context
- Removed support for assigning the result of new by reference  
(\$o = & new classname())
- Removed support for #-style comments in ini files. Use ;-style comments instead



# 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);

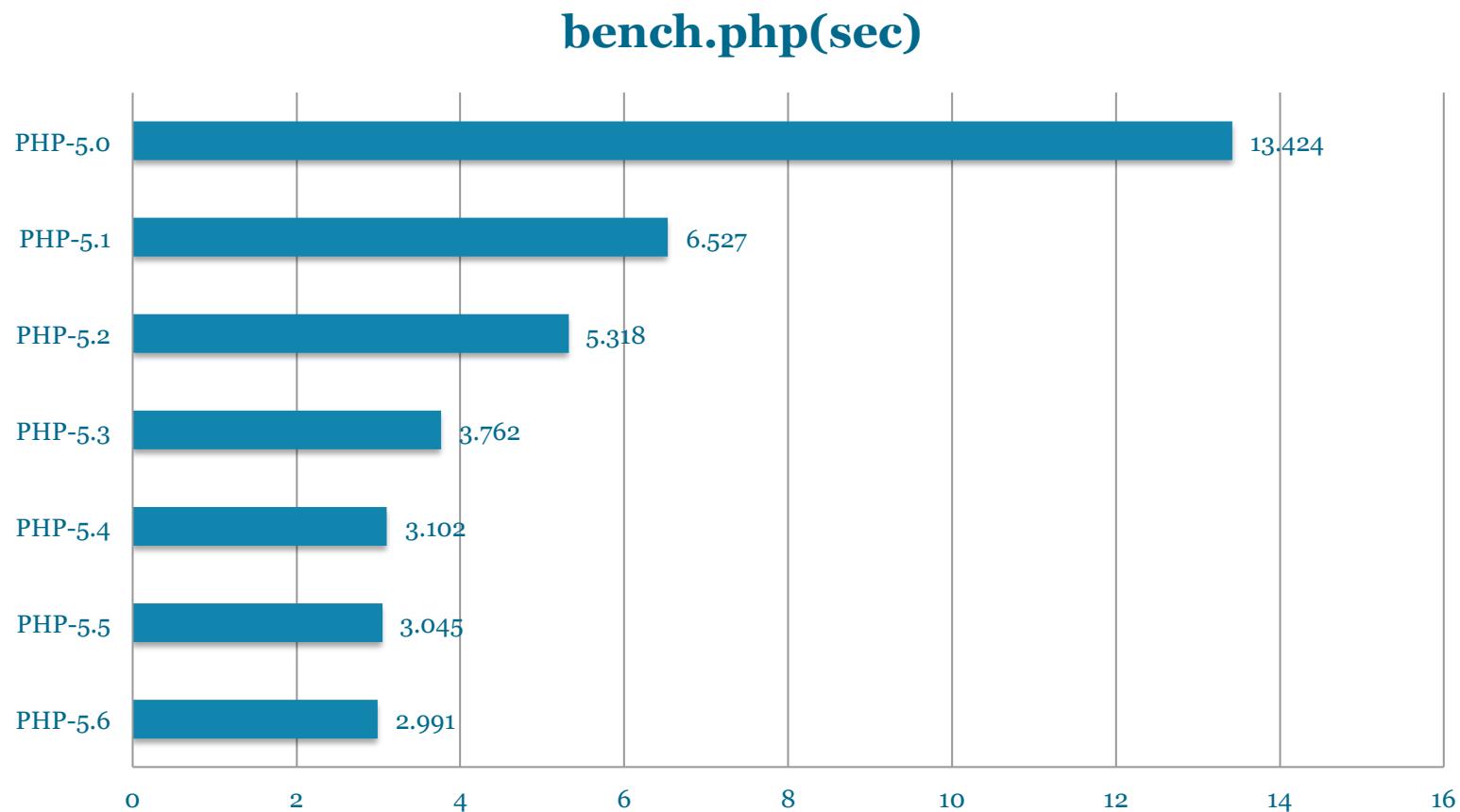
*PHP5 : int(1) VS PHP7: int(2)*



# PHP7 Performance

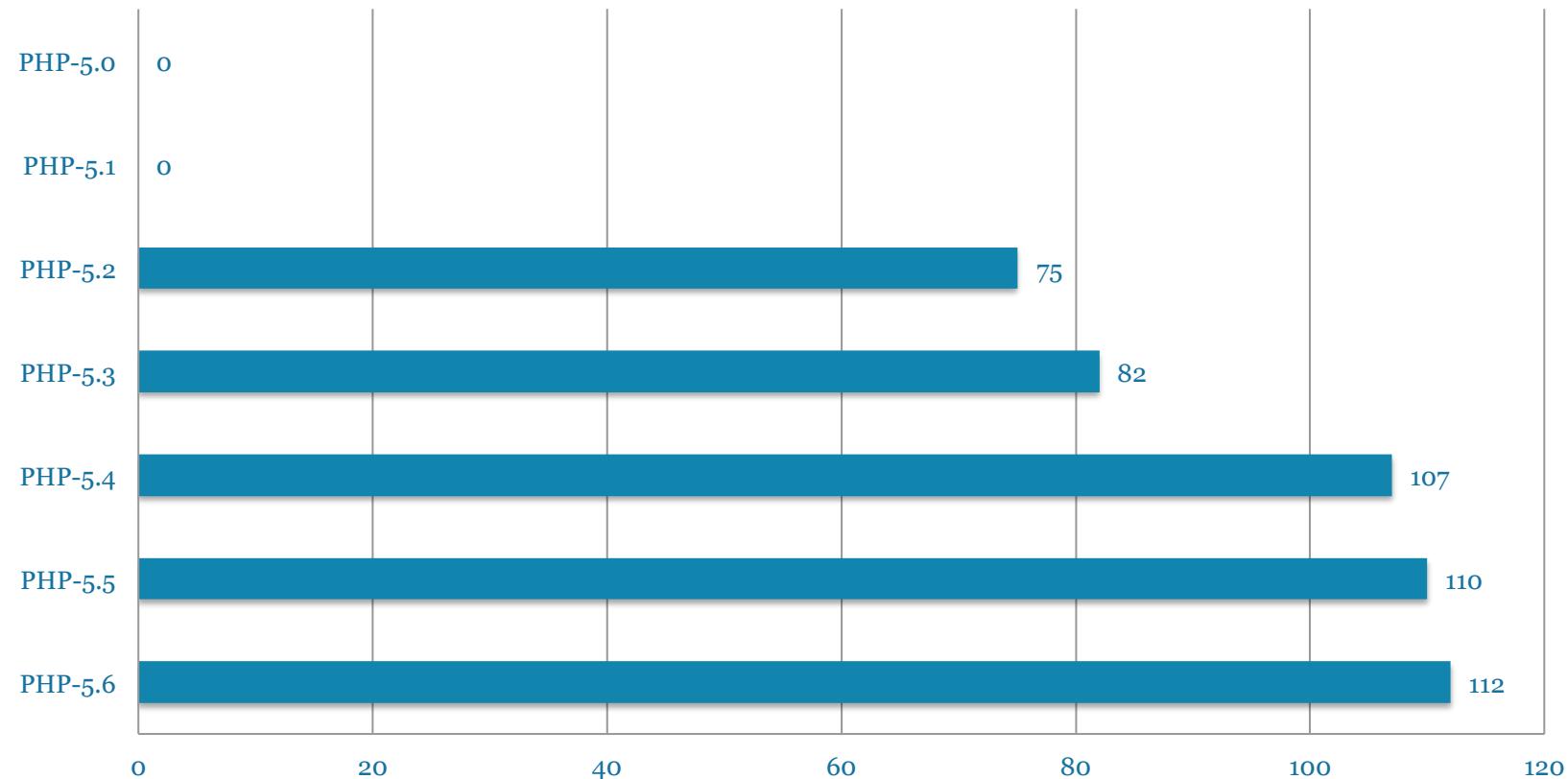
---

# PHP Performance Evaluation



# PHP Performance Evaluation

## Wordpress 3.6 home page qps

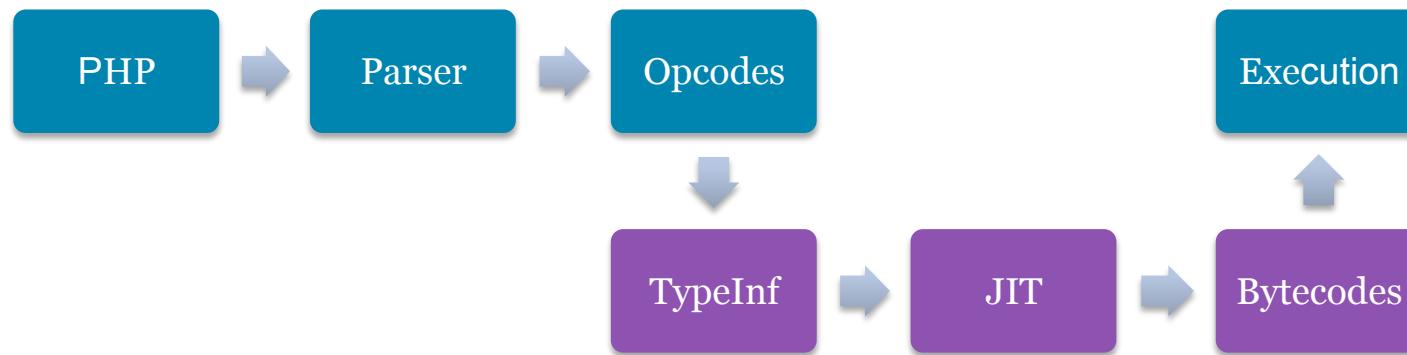


# PHP Performance Evaluation

- ~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

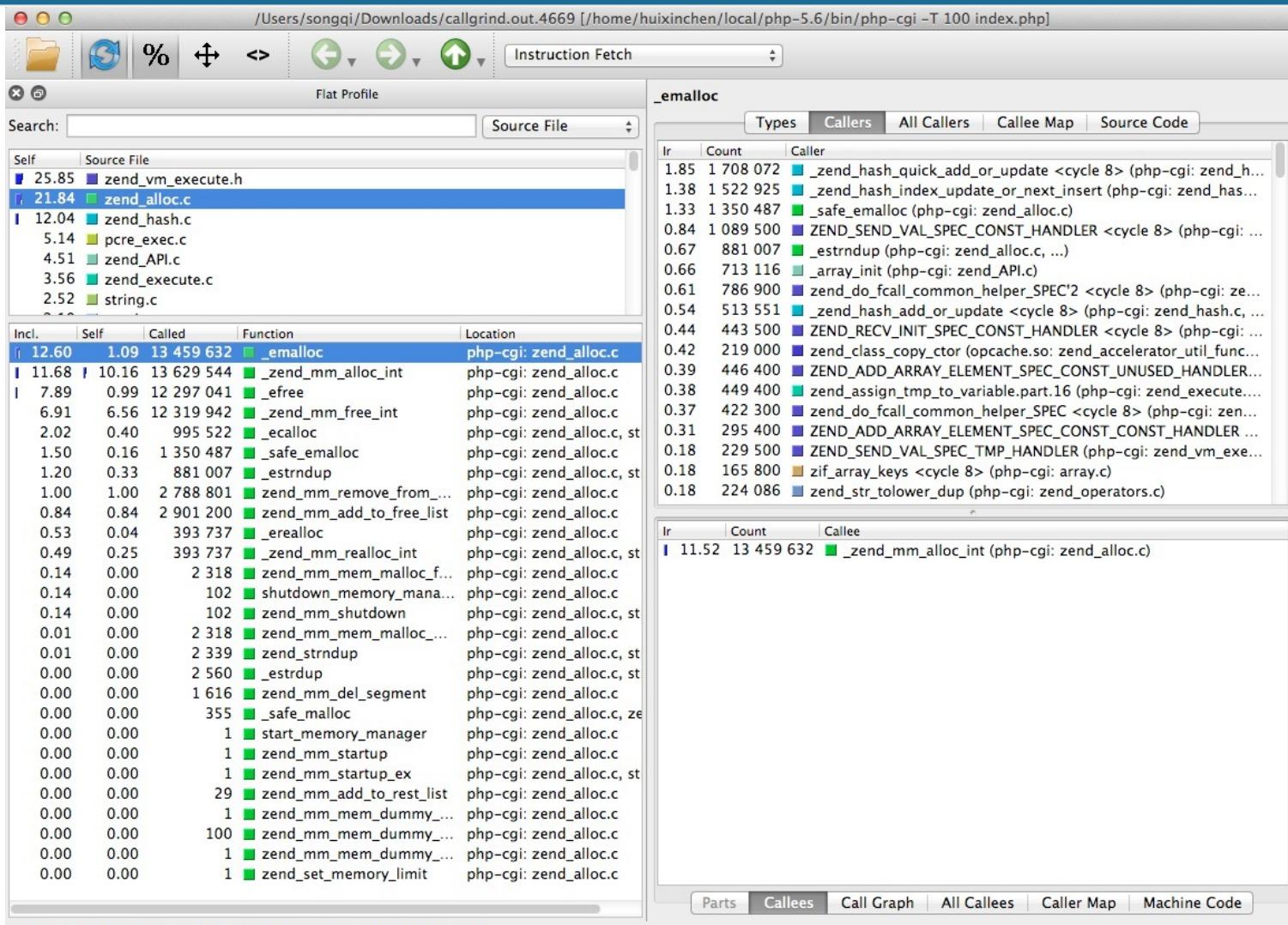


# PHP Just In Time Compiler?

- 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	C	E	F
	PHP 5.5	PHP 5.5 + JIT(24 Aug)		hhvm
bench.php				
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
<b>Total</b>	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

# 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;
};

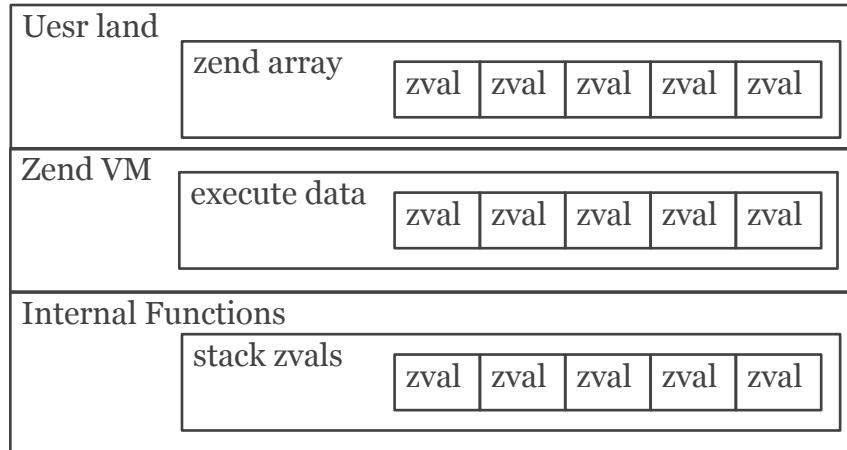
sizeof(zval) == 24
```

```
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;
    } ul;
    union {
        zend_uint var_flags;
        zend_uint next;
        zend_uint str_offset;
        zend_uint cache_slot;
    } u2;
};

sizeof(zval) == 16
```

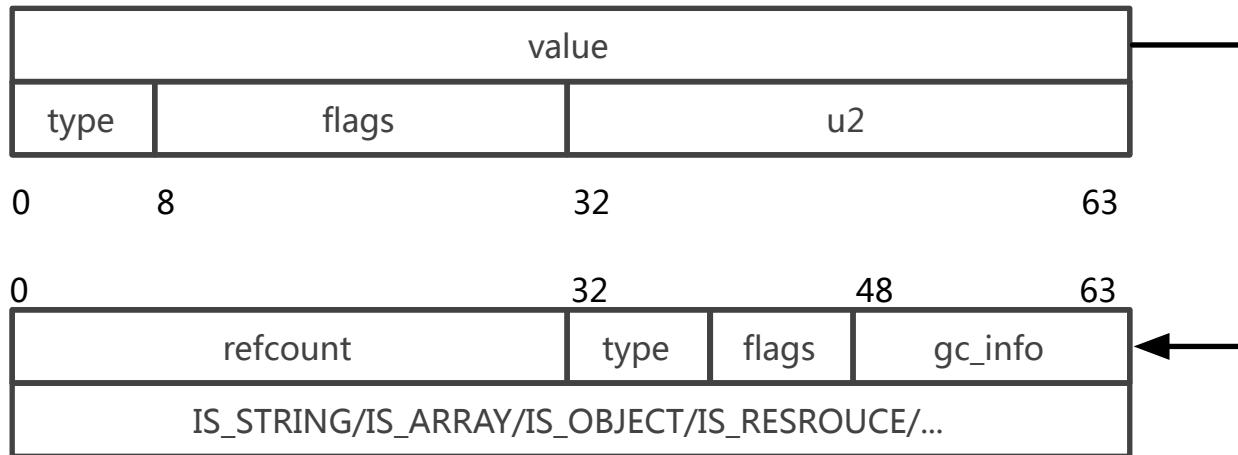
# ZVAL

- No refcount for scalar types
- zvals are always pre-allocated 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



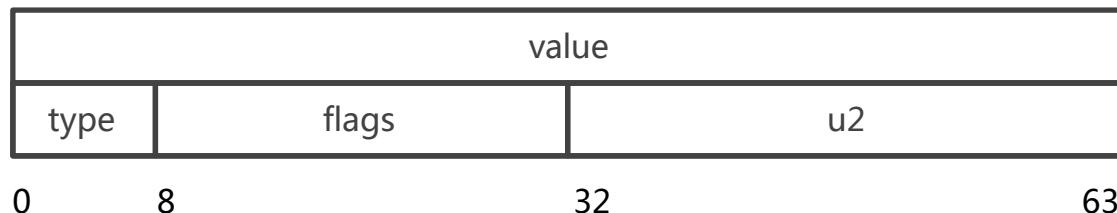
# ZVAL

- IS\_UNDEF
- IS\_NULL
- IS\_FALSE
- IS\_TRUE
- IS\_LONG
- IS\_DOUBLE
- IS\_STRING
- IS\_ARRAY
- IS\_OBJECT
- IS\_RESOURCE
- IS\_REFERENCE



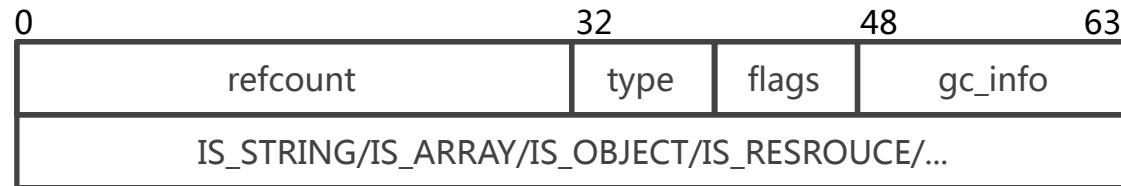
# ZVAL NON REFCLUDED

- IS\_NULL
- IS\_FALSE
- IS\_TRUE
- IS\_LONG
- IS\_DOUBLE



# ZVAL REFCOUNDED

- IS\_STRING
- IS\_ARRAY
- IS\_OBJECT
- IS\_RESOURCE
- IS\_REFERENCE

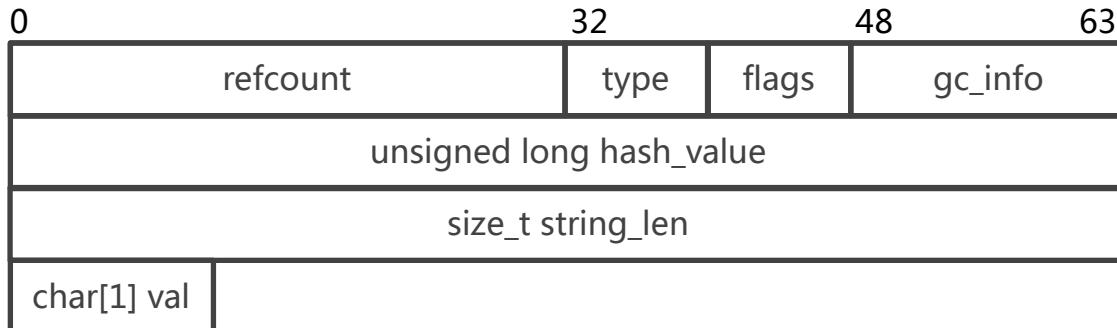


# IS\_STRING

- 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



# 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;
    };
};
```

0	32	48	63
refcount	type	flags	gc_info
u	nTableMask		
Bucket *arData			
nNumUsed	nNumOfElements		
.....			

- IS\_ARRAY\_IMMUTABLE

# 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];
};
```

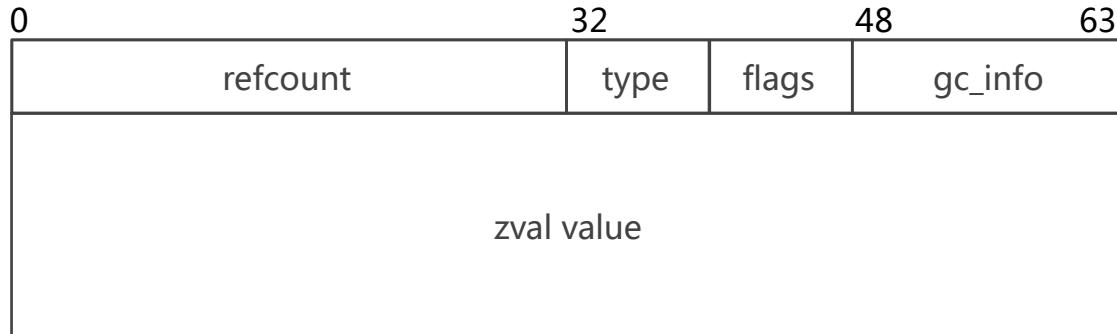
- IS\_OBJ\_APPLY\_COUNT
- IS\_OBJ\_DESTRUCTOR\_CALLED
- IS\_OBJ\_FREE\_CALLED

0	32	48	63
refcount	type	flags	gc_info
zend_class_entry *ce			
zend_object_handlers *handlers			
zend_array *properties			
zend_array *guarders			
zval *properties_table[1]			

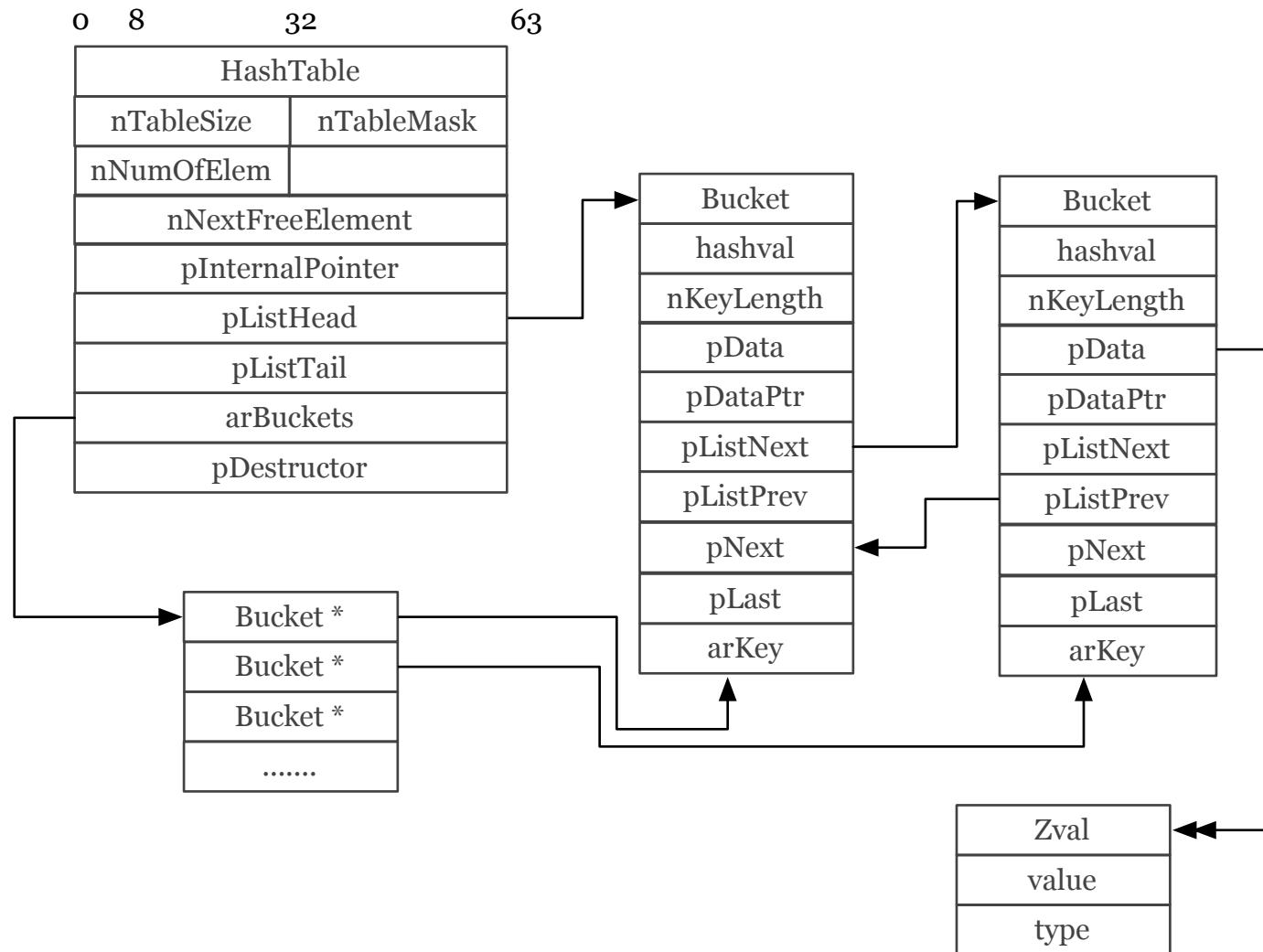
# IS\_REFERENCE

- New Internal Type: Zend Reference
- Reference is type

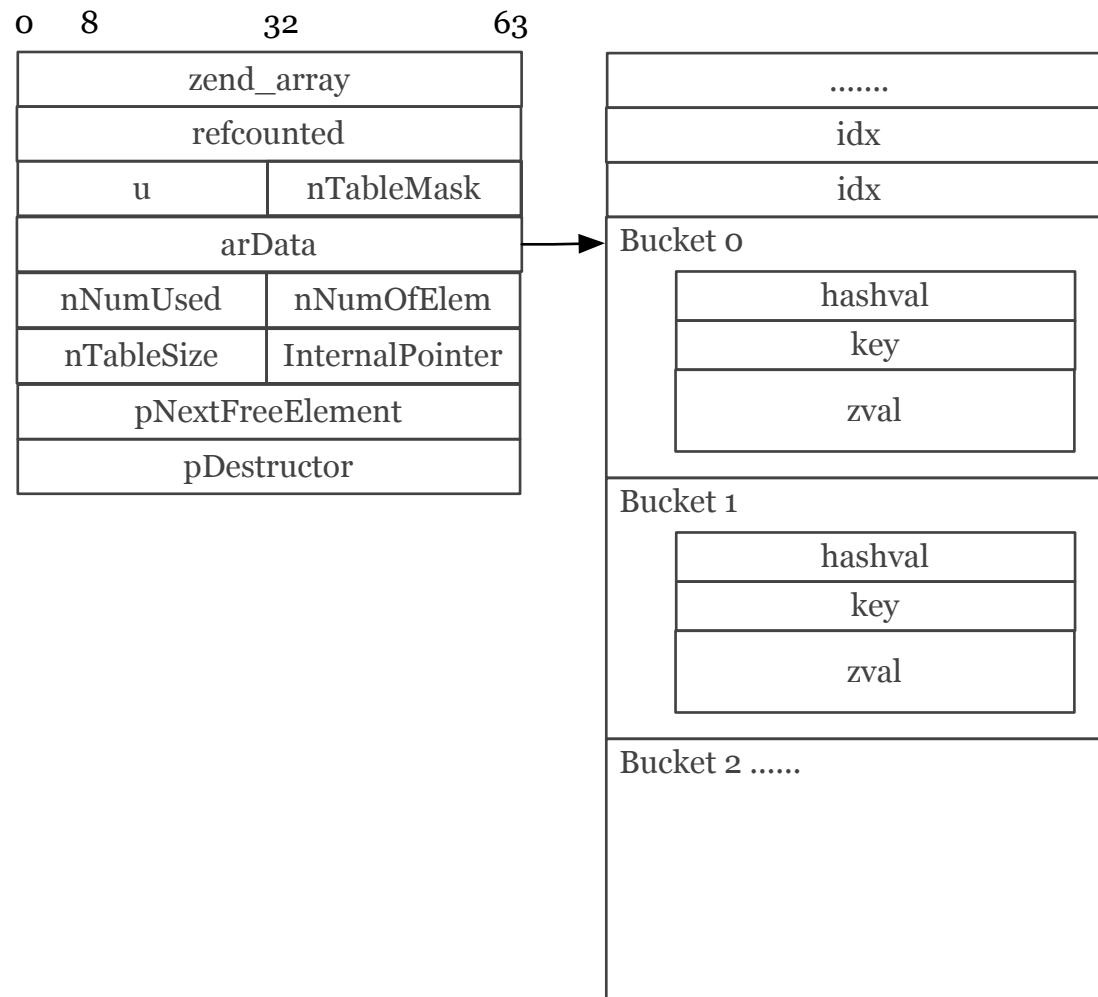
```
struct _zend_reference {
    zend_refcounted  gc;
    zval              val;
};
```



# HashTable – PHP 5



# Zend Array – PHP 7



# Zend Array(HashTable)



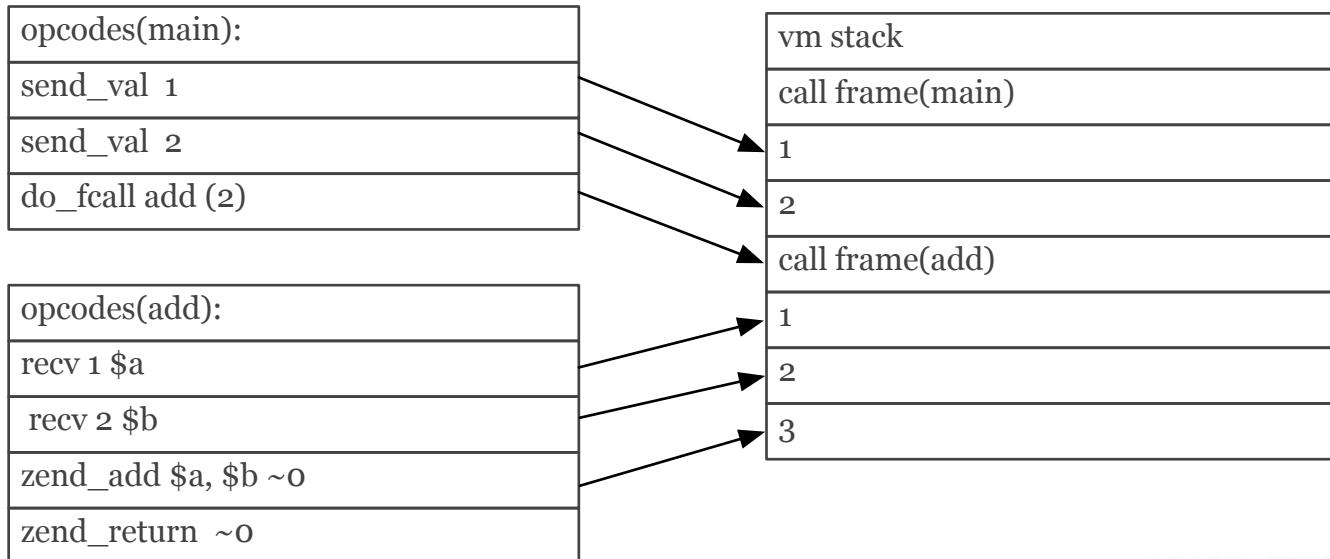
- Values of arrays are zval by default
- HashTable size reduced from 72 to 56 bytes
- Bucket size reduced from 72 to 32 bytes
- Memory for all Buckets is allocated at once
- Bucket.key now is a pointer to zend\_string
- Values of array elements are embedded into the Buckets
- Improved data locality =>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

# Function calling convention – PHP5



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

add(1, 2);

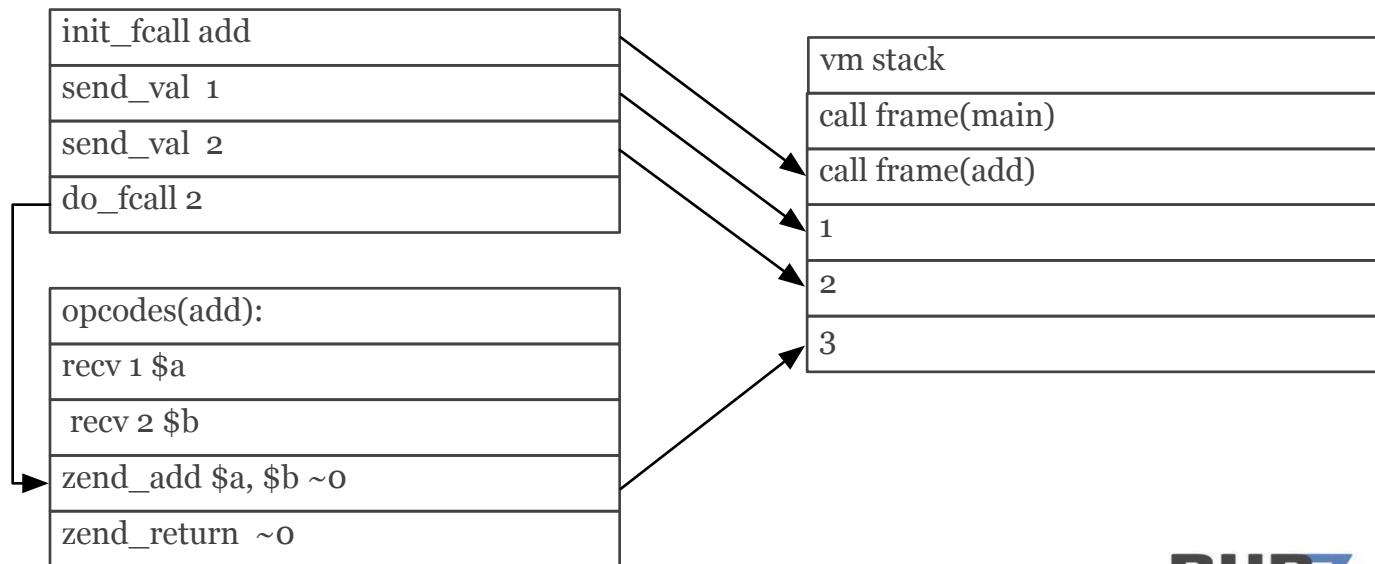


# Function calling convention – PHP7



- 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();
```

# Inline Frequently used simple functions



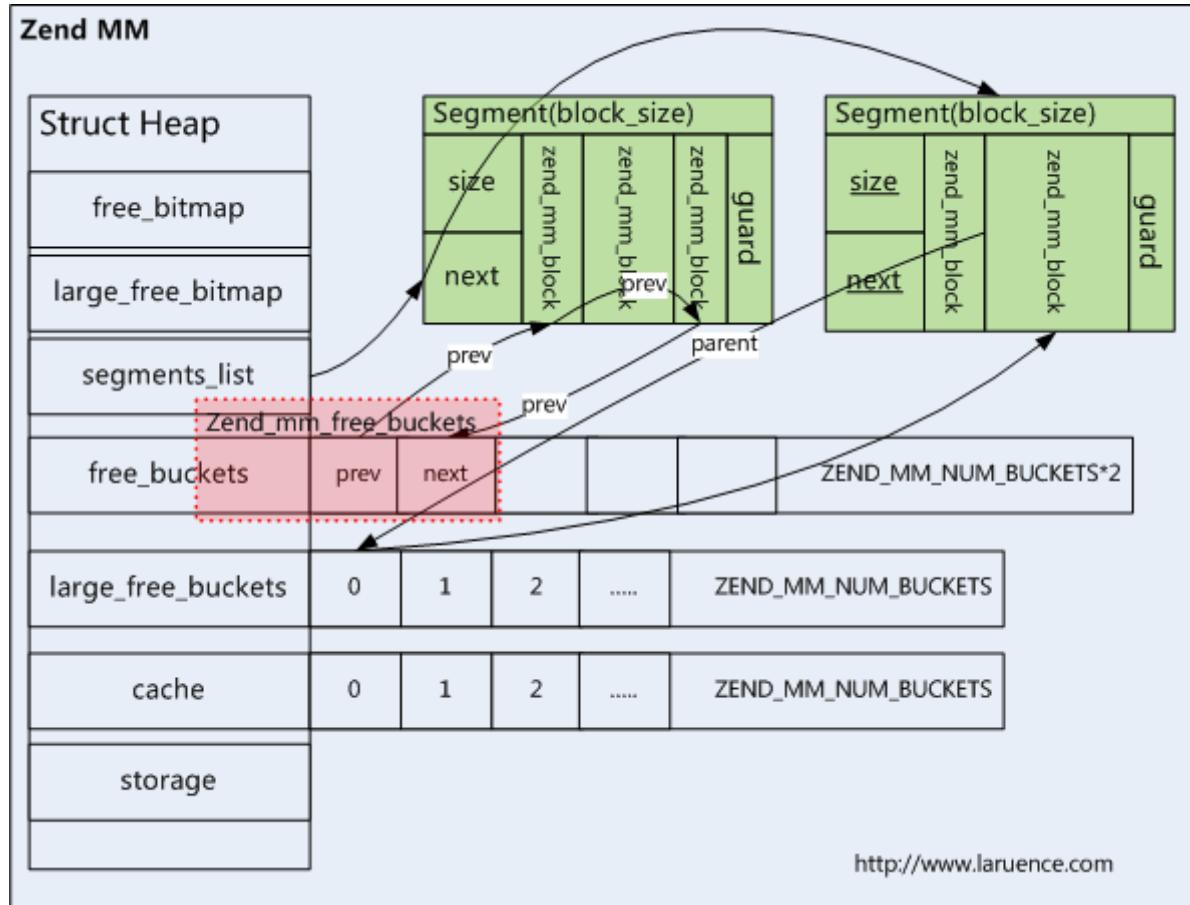
- `call_user_function(_array)` => `ZEND_INIT_USER_CALL`
- `is_int/string/array/* etc` => `ZEND_TYPE_CHECK`
- `strlen` => `ZEND_STRLEN`
- `defined` => `ZEND+DEFINED`
- ...

# Faster zend\_qsort

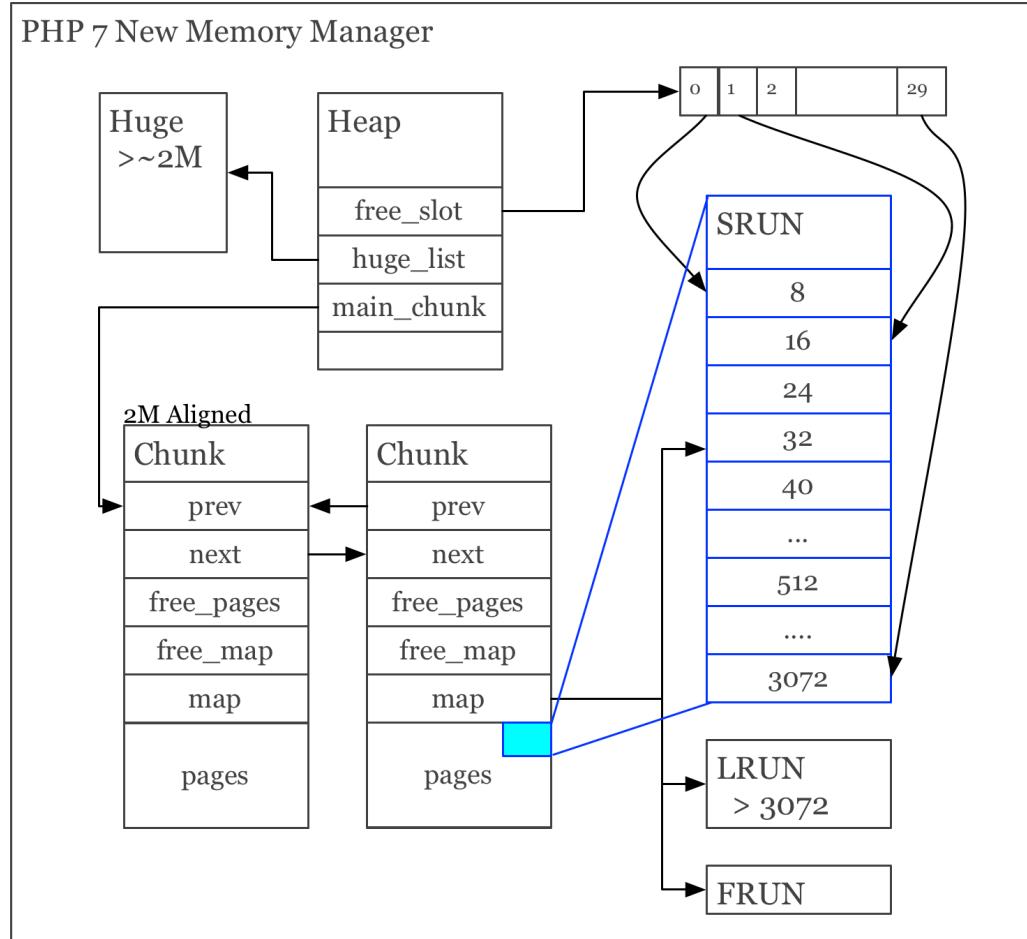


- Refactor zend\_qsort for better performance
- Hybrid Sorting Algo(Quick Sort and Selection Sort)
- <16 elements do stable sorting
- `$array = array(0 => 0, 1=>0); asort($array);`
  - *PHP5: array(1=>0, 0=>0);*
  - *PHP7: array(0=>1, 1=>0);*

# Memory Manager PHP5



# Memory Manager PHP7



# New Memory Manager



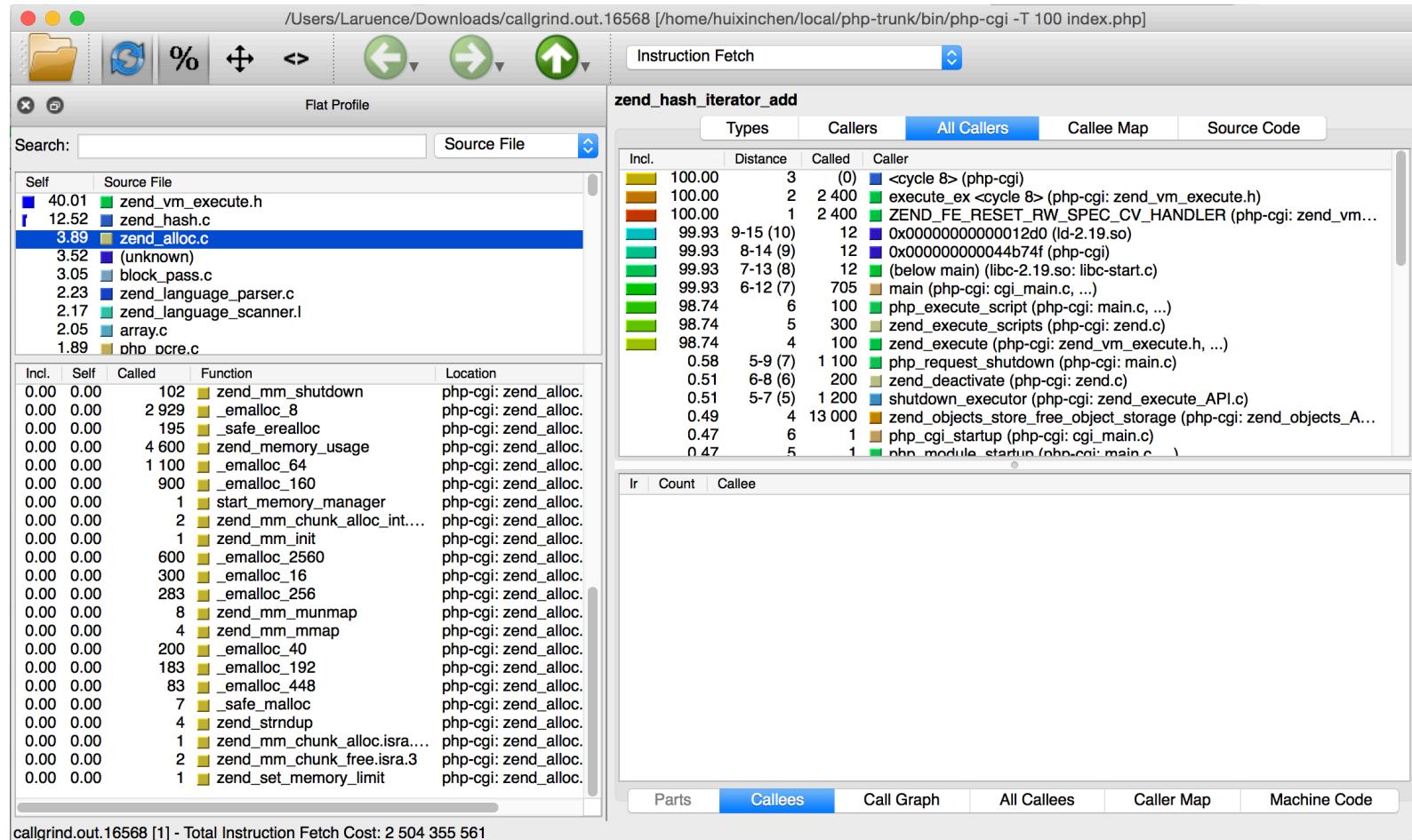
- Less memory wasting
- Friendly to modern CPU cache
- less CPU cache misses
- Faster builtin types allocating
- ~5% CPU time reduce in wordpress homepage

# 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
- ....

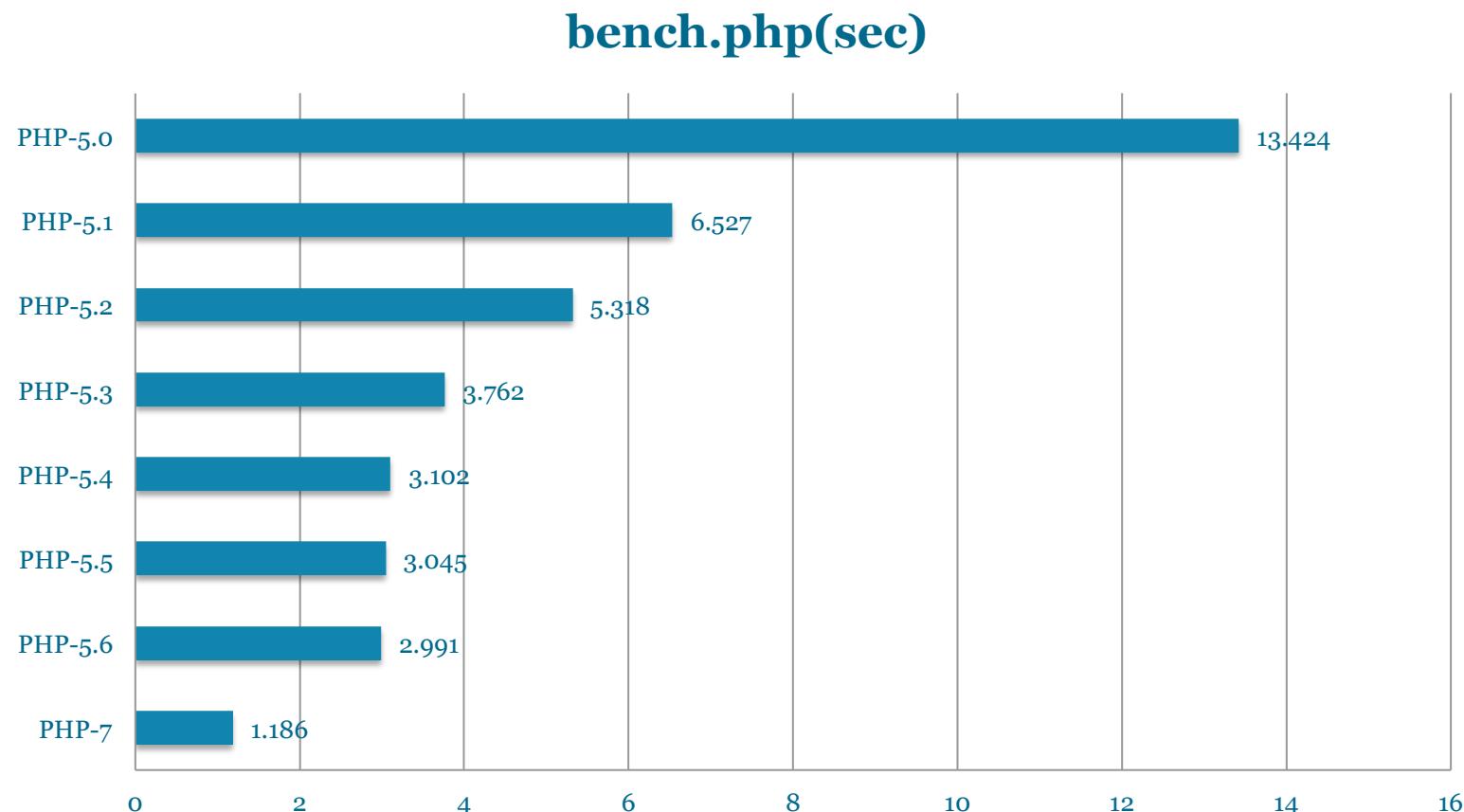
# Wordpress profile (2015-04-14)



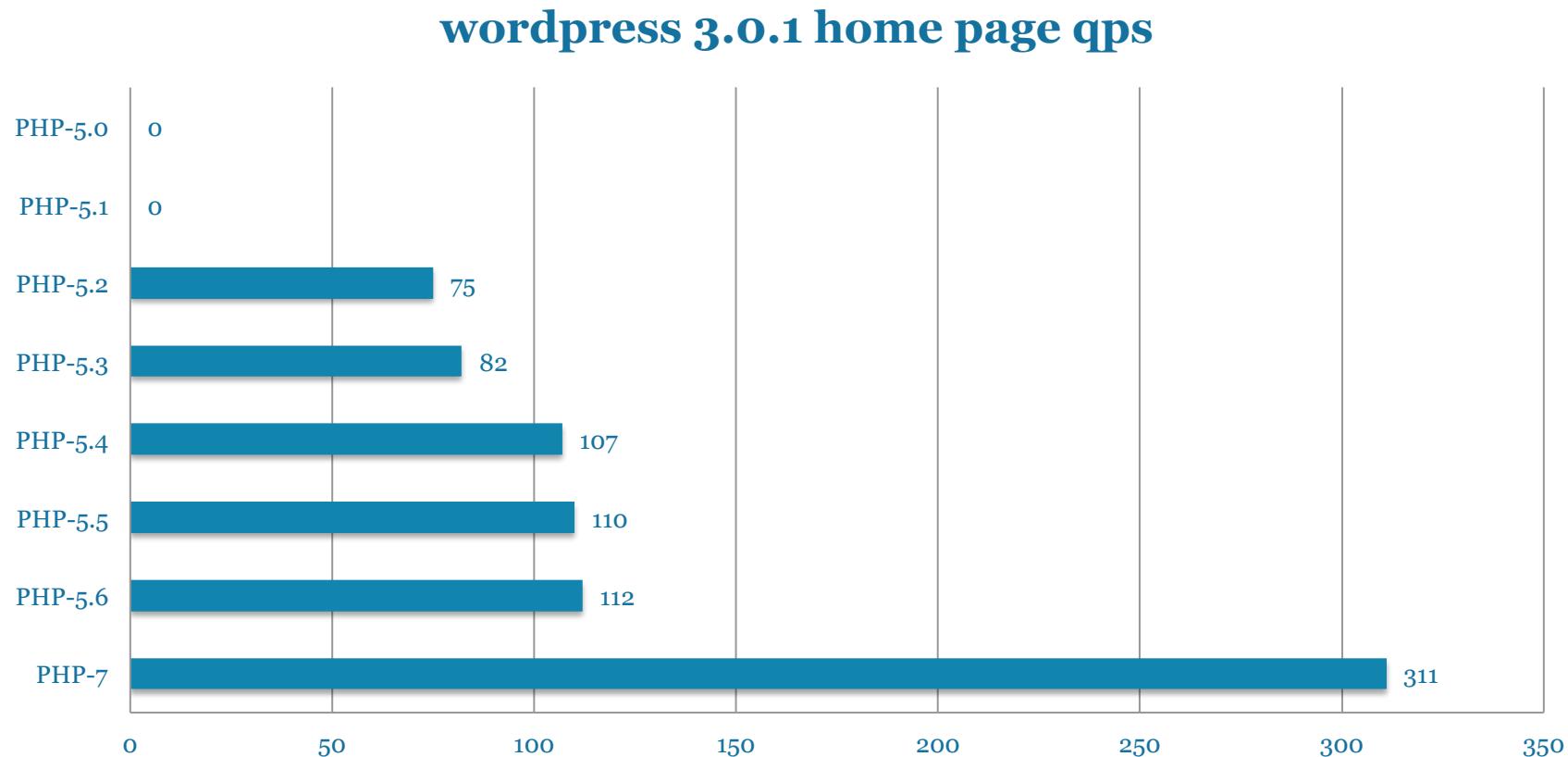
# Wordpress profiled (2015-04-14)

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

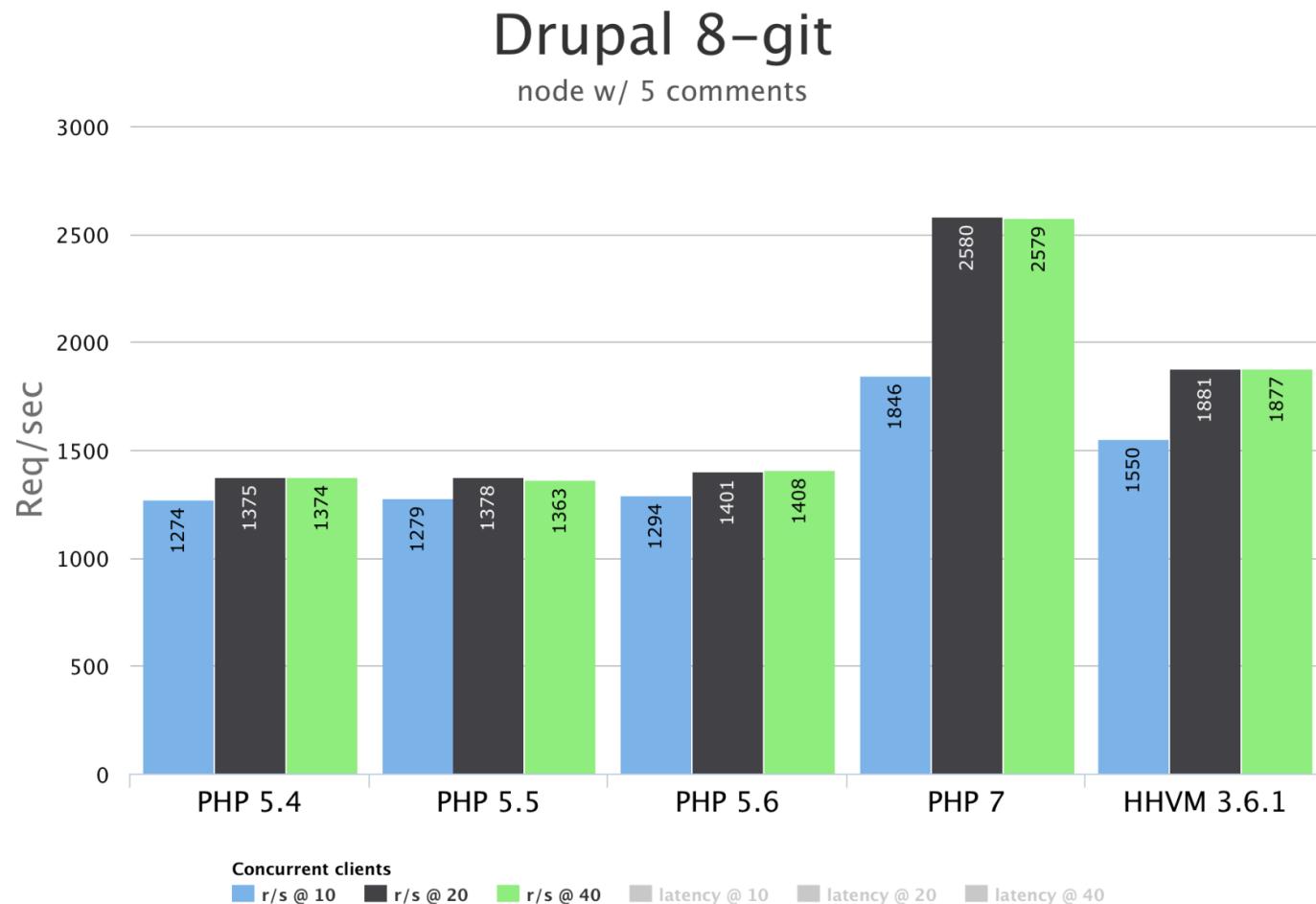
# PHP7 Performance – Benchmark (2014-04-14)



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



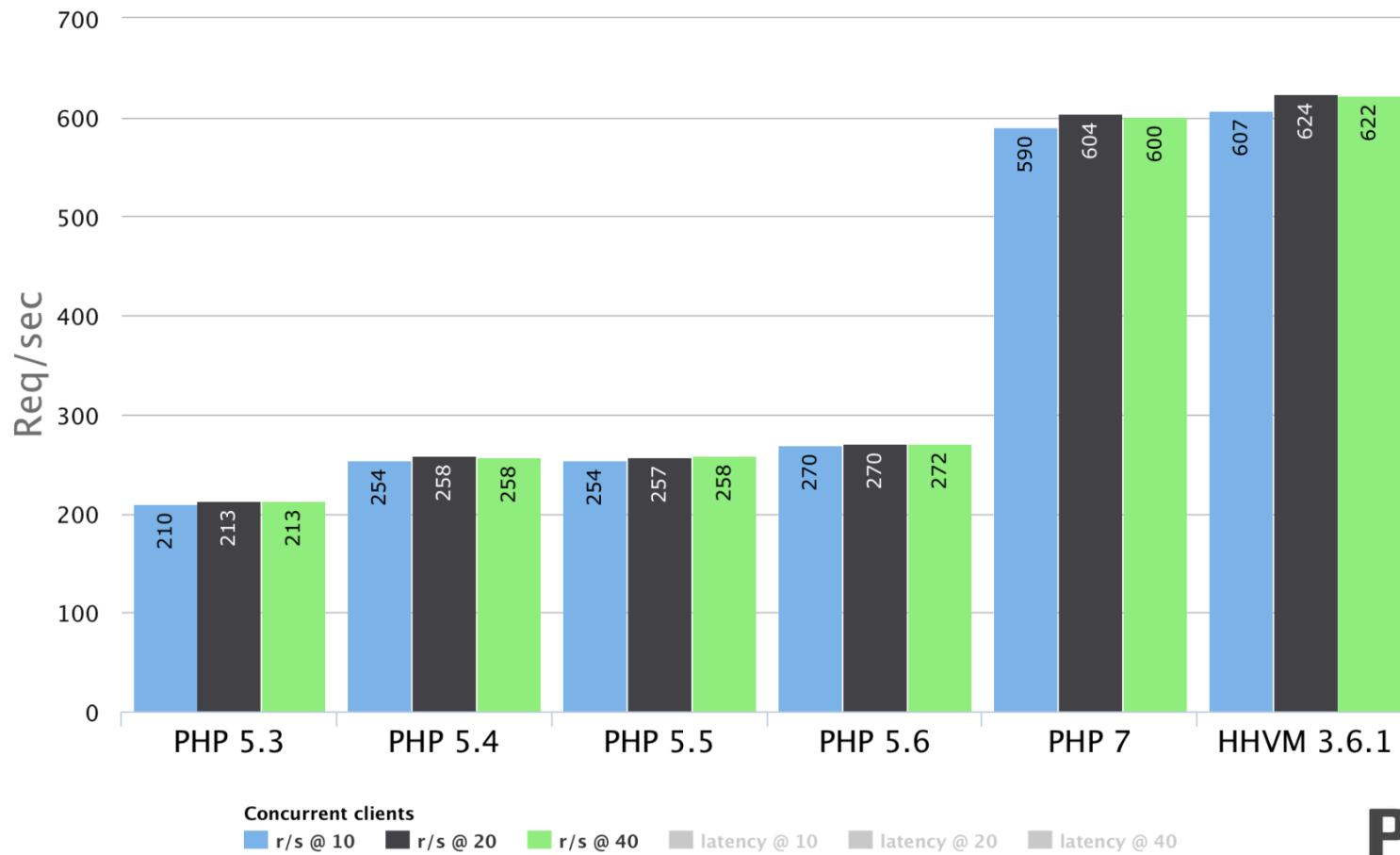
# PHP7 Performance (By Rasmus 2015-04-21)



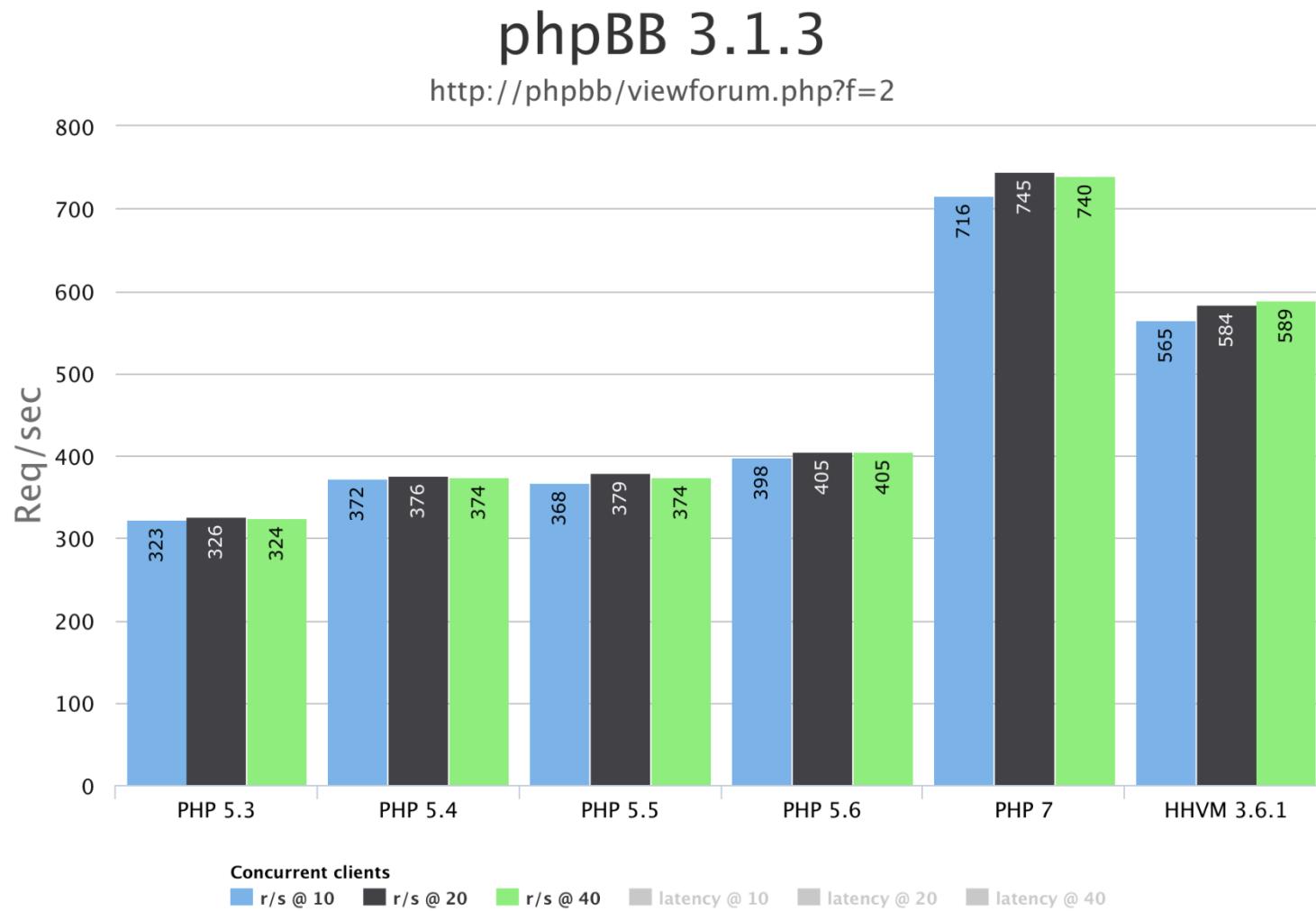
# PHP7 Performance (2015-04-21)

## Wordpress-4.1.1

http://wordpress/?p=1



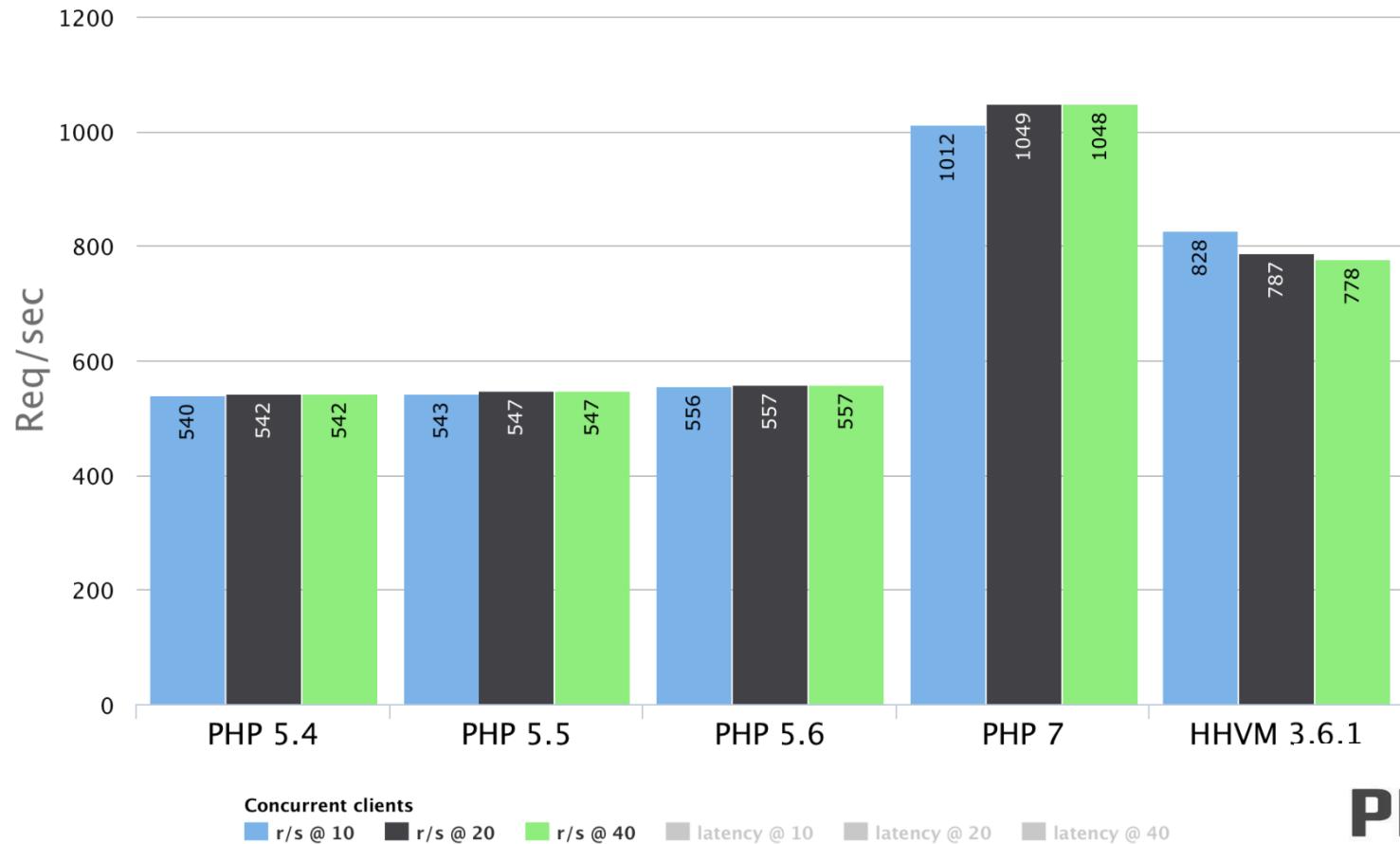
# PHP7 Performance (2015-04-21)



# PHP7 Performance (2015-03-15)

## WardrobeCMS 1.2.0

front page with one short post



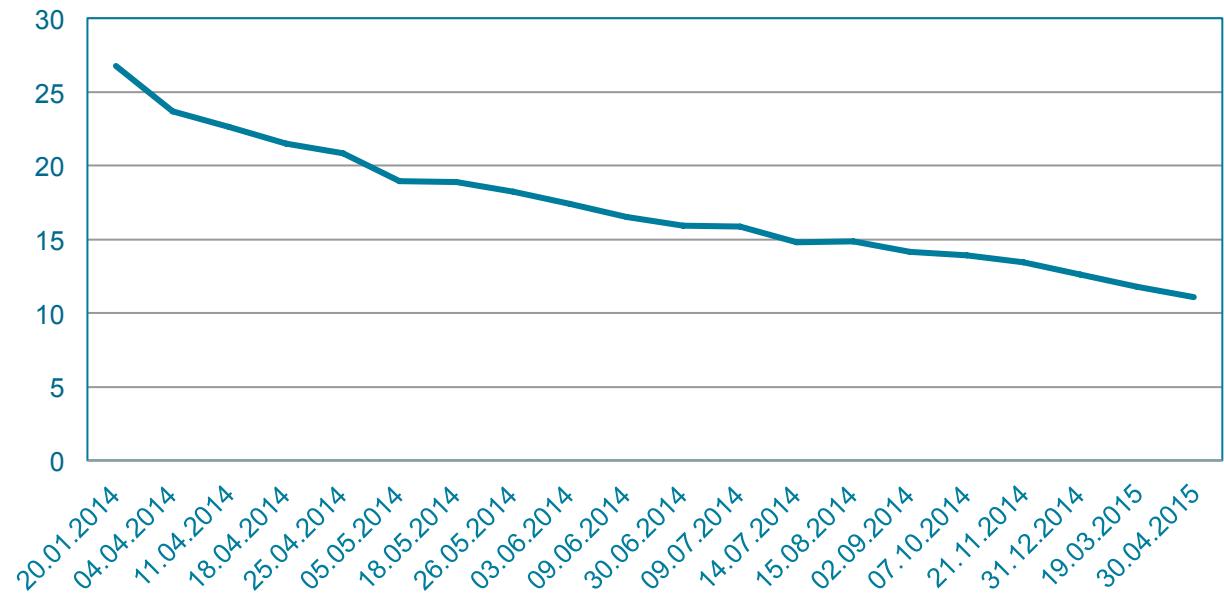
# Always Do Your Own Benchmark

---

# PHP 7 Next

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

WP-3.6 request 100 times (sec)



# Links

- phpng:\_Refactored\_PHP\_Engine\_with\_Big\_Performance\_Improvement: <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#/>

# Questions?

---

Thanks