

PHP 7

- New Engine For The Good Old Train

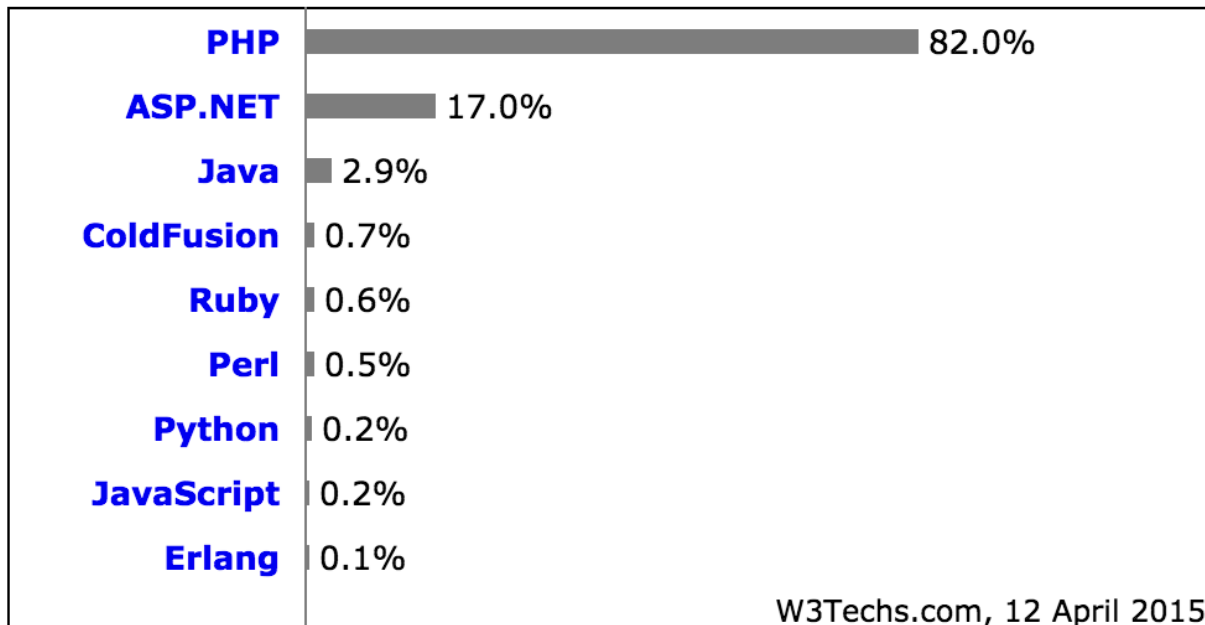
@laruence

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

PHP 7 New Features

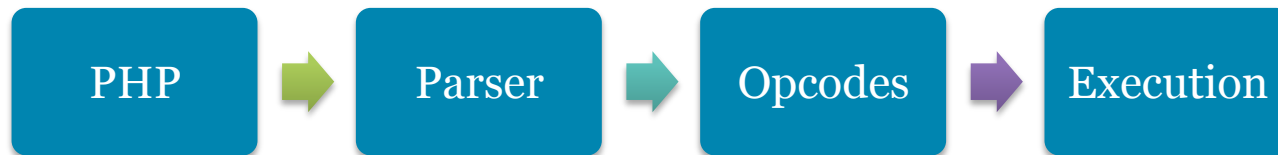
PHP 7?

- PHP NG – Engine Refactor - 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
- >2GB file uploading
- Fully 64bits integers cross platforms

string size		signed integer
Platform	<i>int</i>	<i>long</i>
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']()`

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

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

Exception: Call to undefined function non_exists_func()

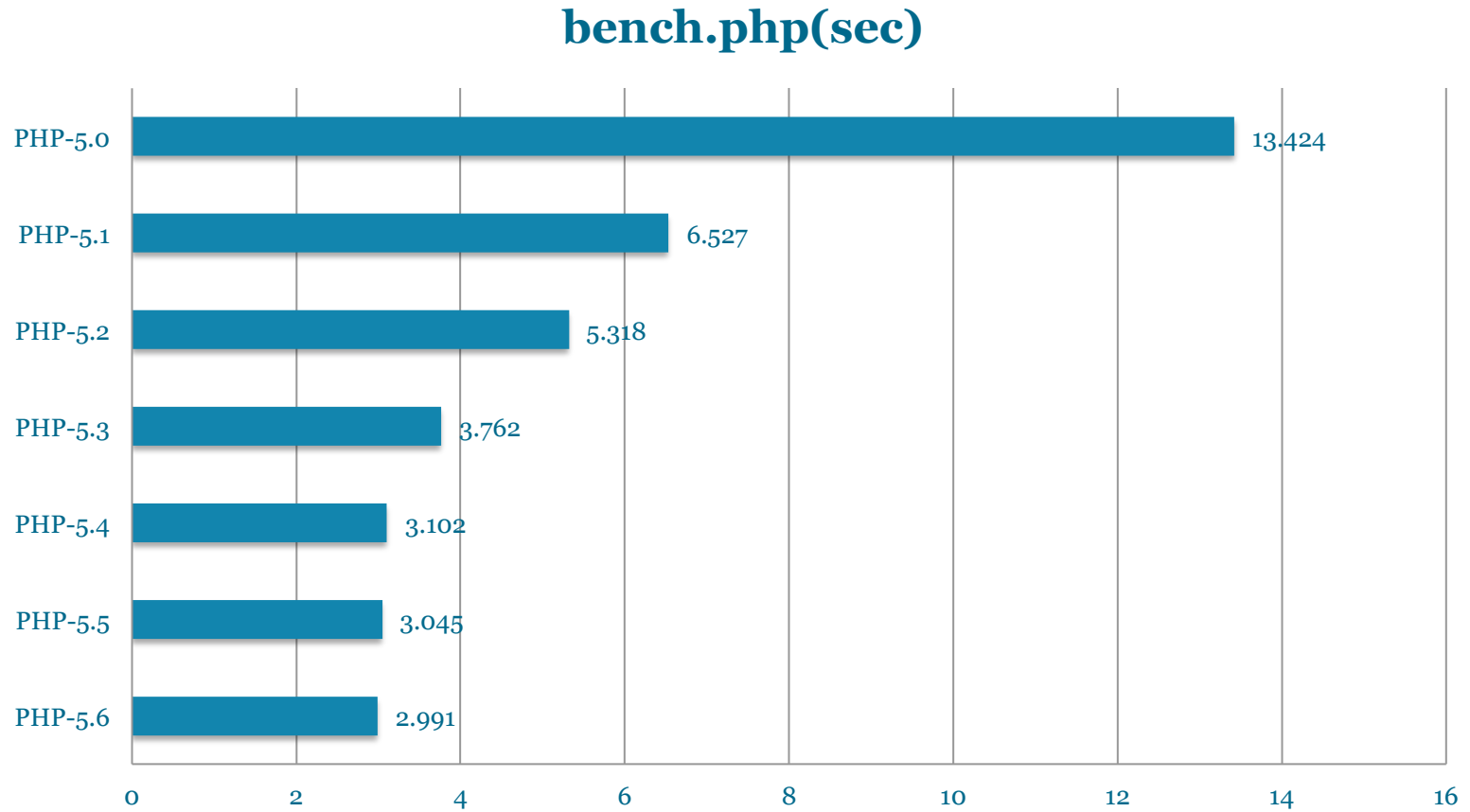
- Uncaught Exception result to FATAL ERROR

```
non_exists_func();
```

PHP Fatal error: Call to undefined function non_exists()

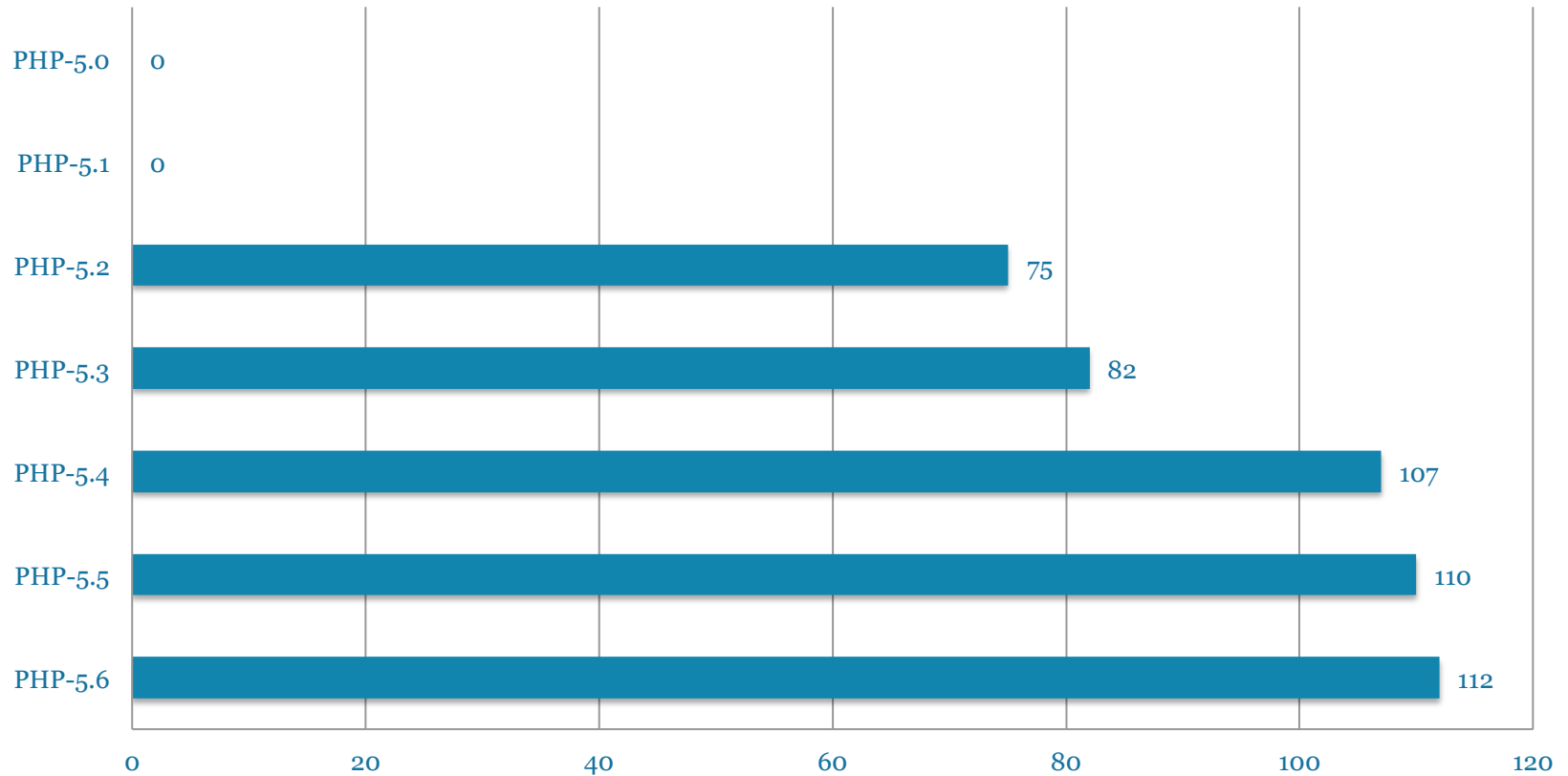
PHP NG (Next Generation)

PHP Performance Evaluation



PHP Performance Evaluation

Wordpress 3.6 home page qps

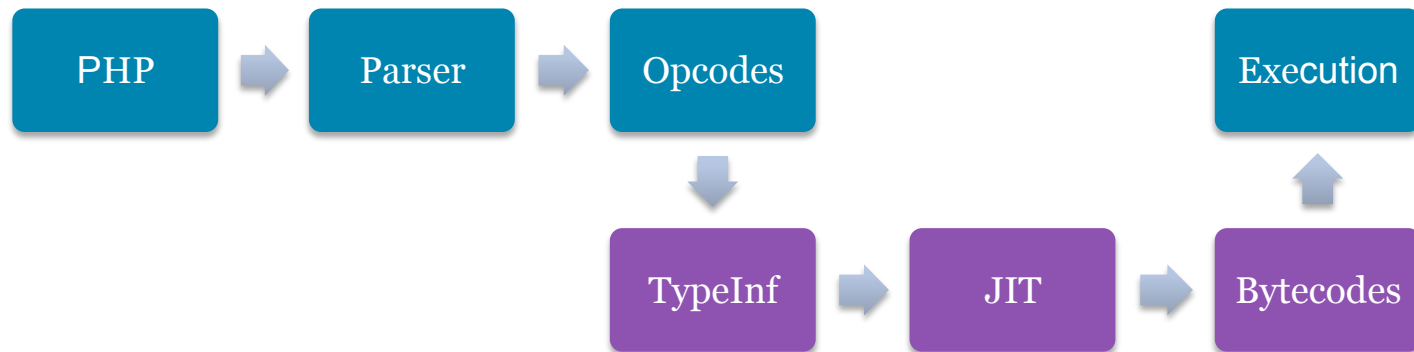


PHP Performance Evaluation

- ~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

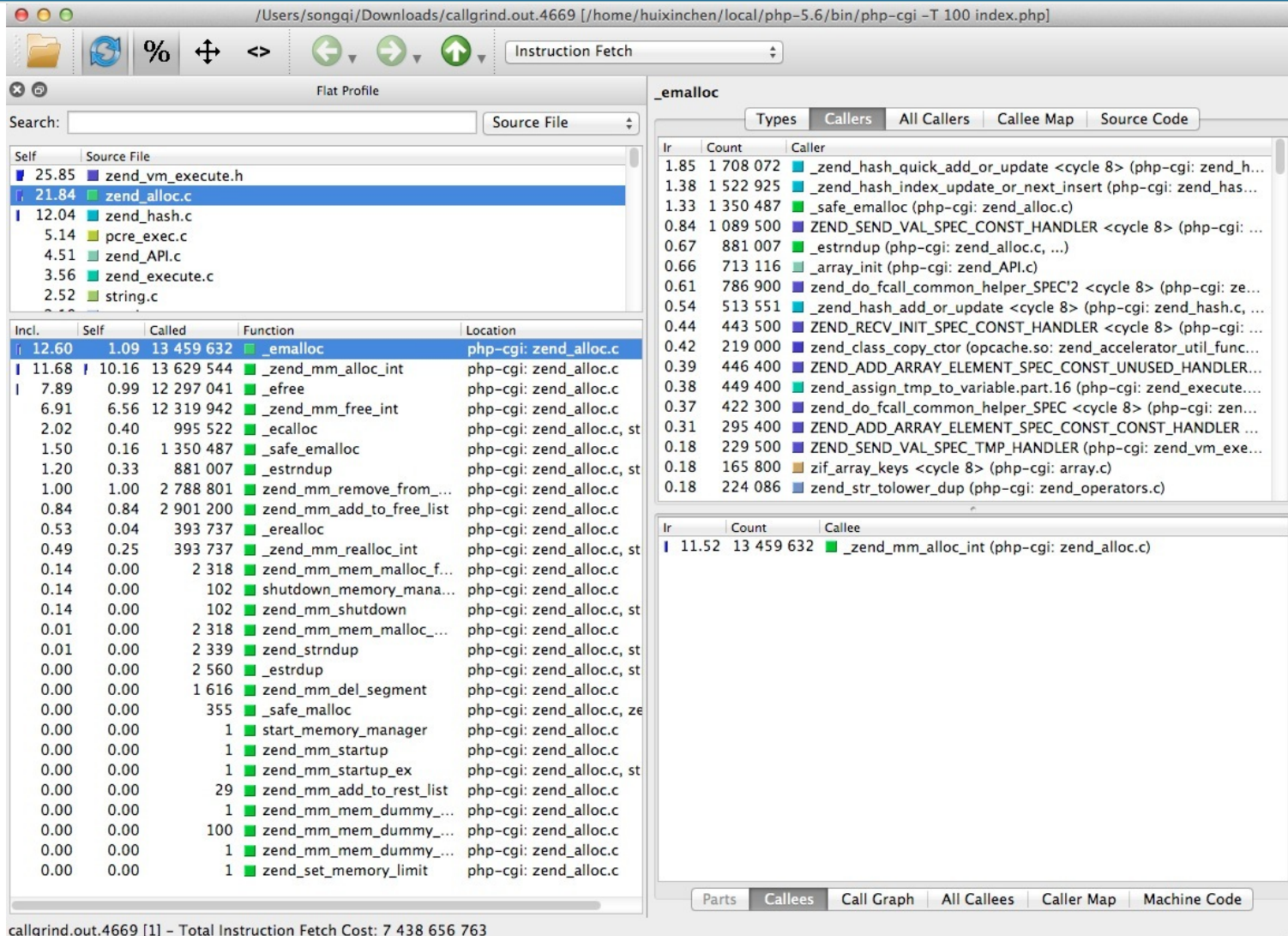


PHP Just In Time Compiler?

- 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	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
fibonacci(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

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;
};
```

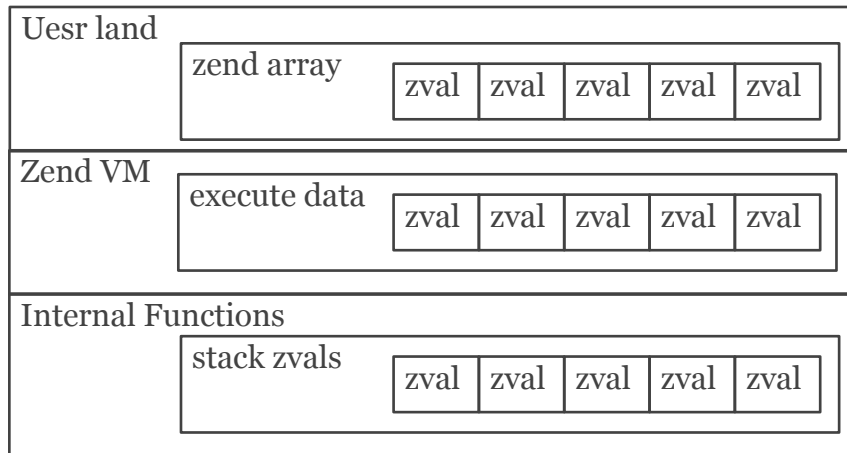
```
sizeof(zval) == 24
```

```
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
```

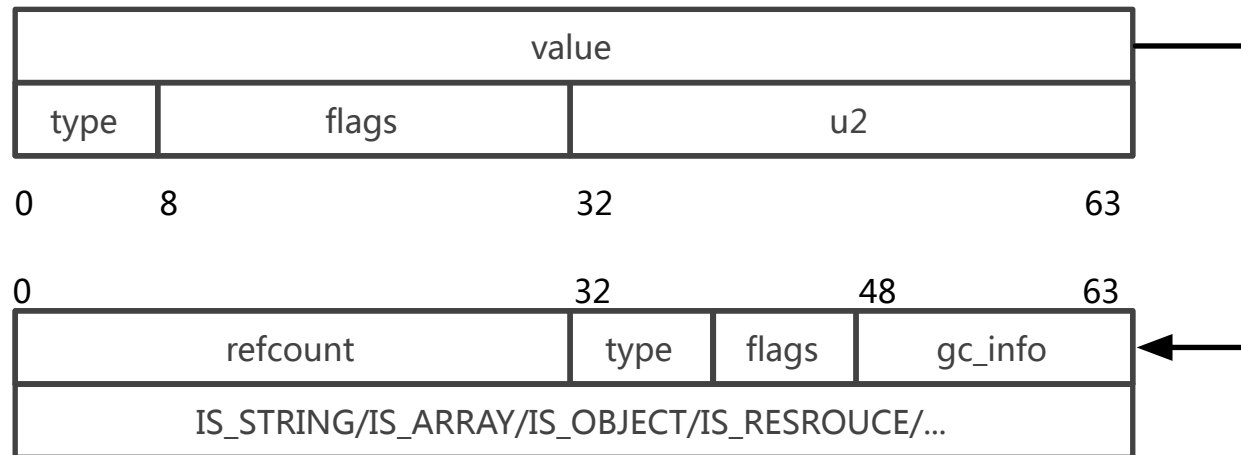
ZVAL

- No refcount for scalar types
- zval are always pre-allocated or allocated in stack
- String using refcount 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



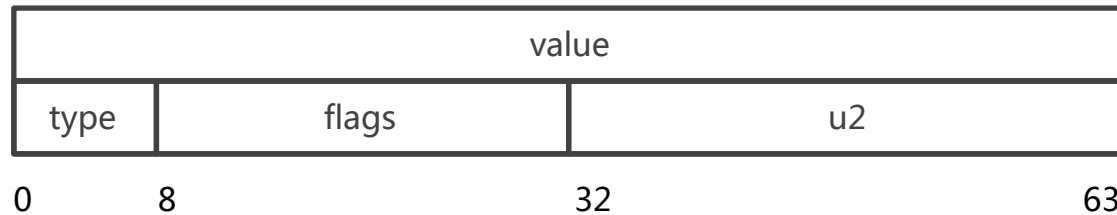
ZVAL

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



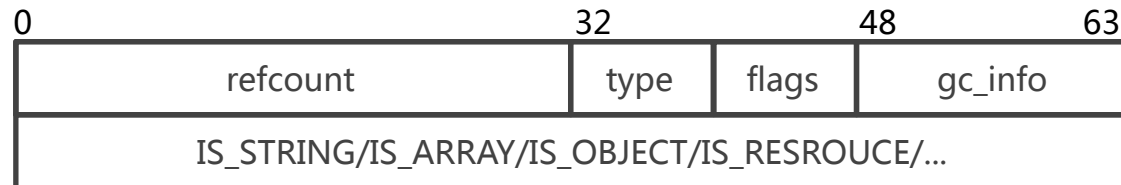
ZVAL NON REFCOUNTED

- IS_NULL
- IS_FALSE
- IS_TRUE
- IS_LONG
- IS_DOUBLE



ZVAL REFCOUNED

- IS_STRING
- IS_ARRAY
- IS_OBJECT
- IS_RESOURCE
- IS_REFERENCE

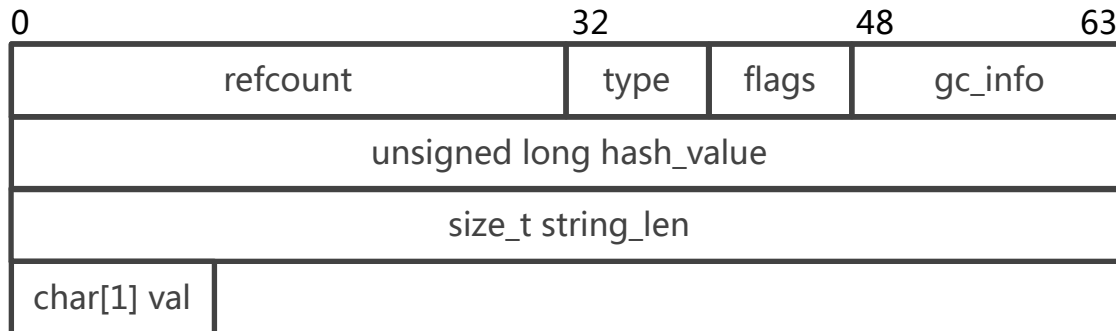


IS_STRING

- 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 from  
    zval               properties_table[1];  
};
```

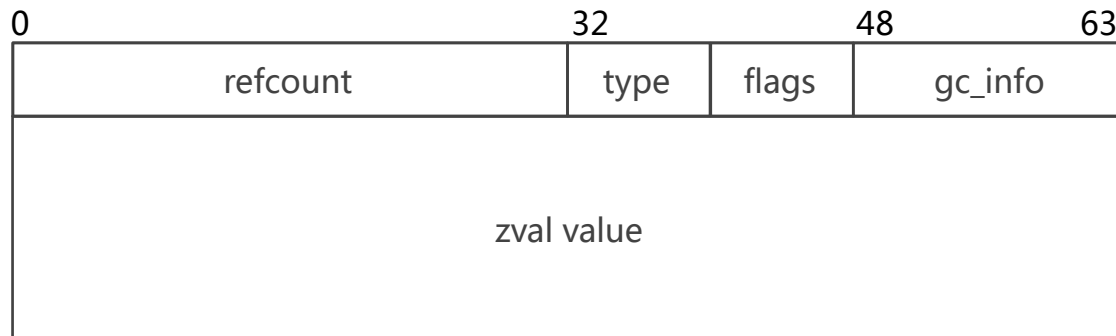
- IS_OBJ_APPLY_COUNT
- IS_OBJ_DESTRUCTOR_CALLED
- IS_OBJ_FREE_CALLED

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

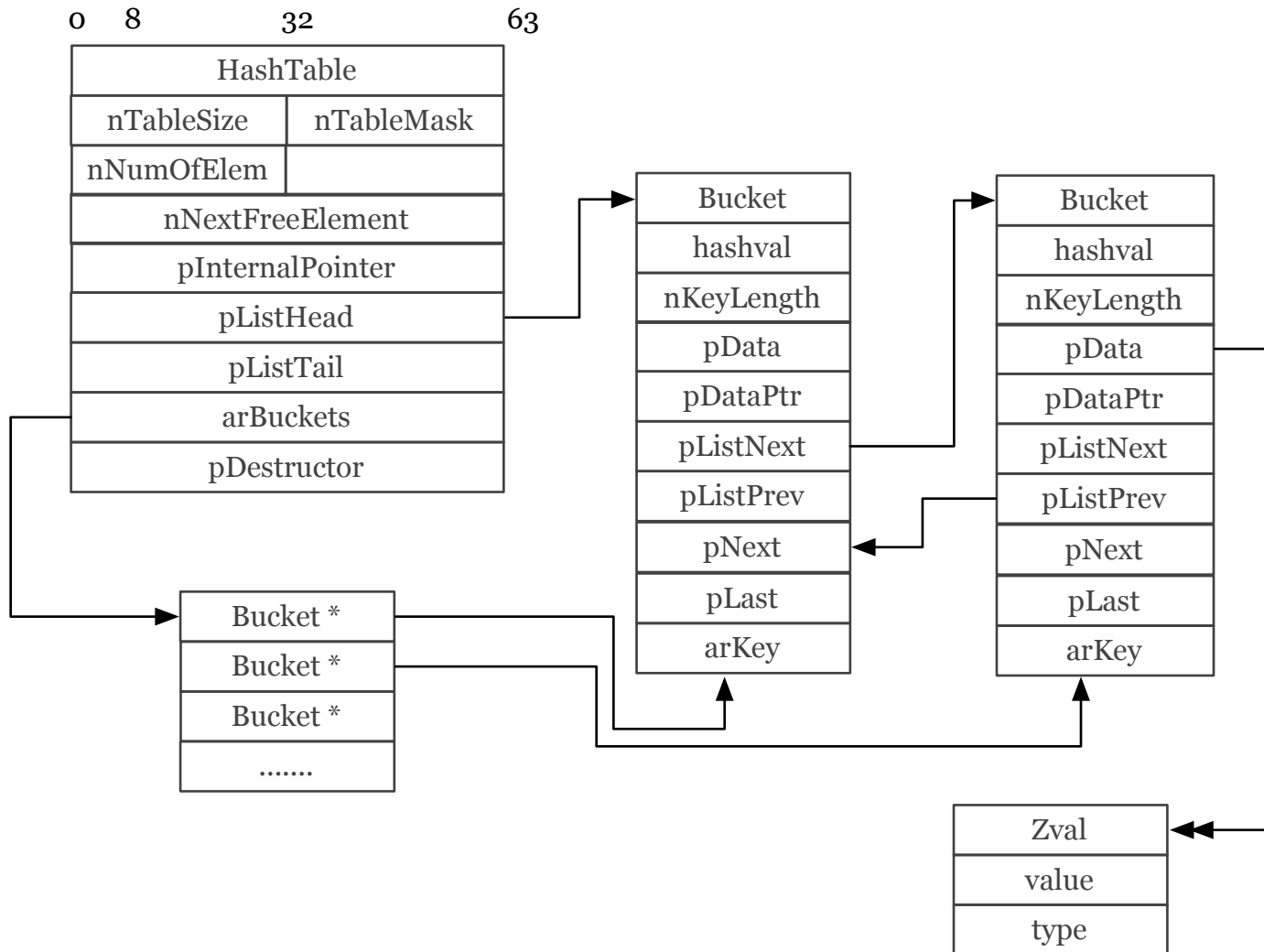
IS_REFERENCE

- New Internal Type: Zend Reference
- Reference is type

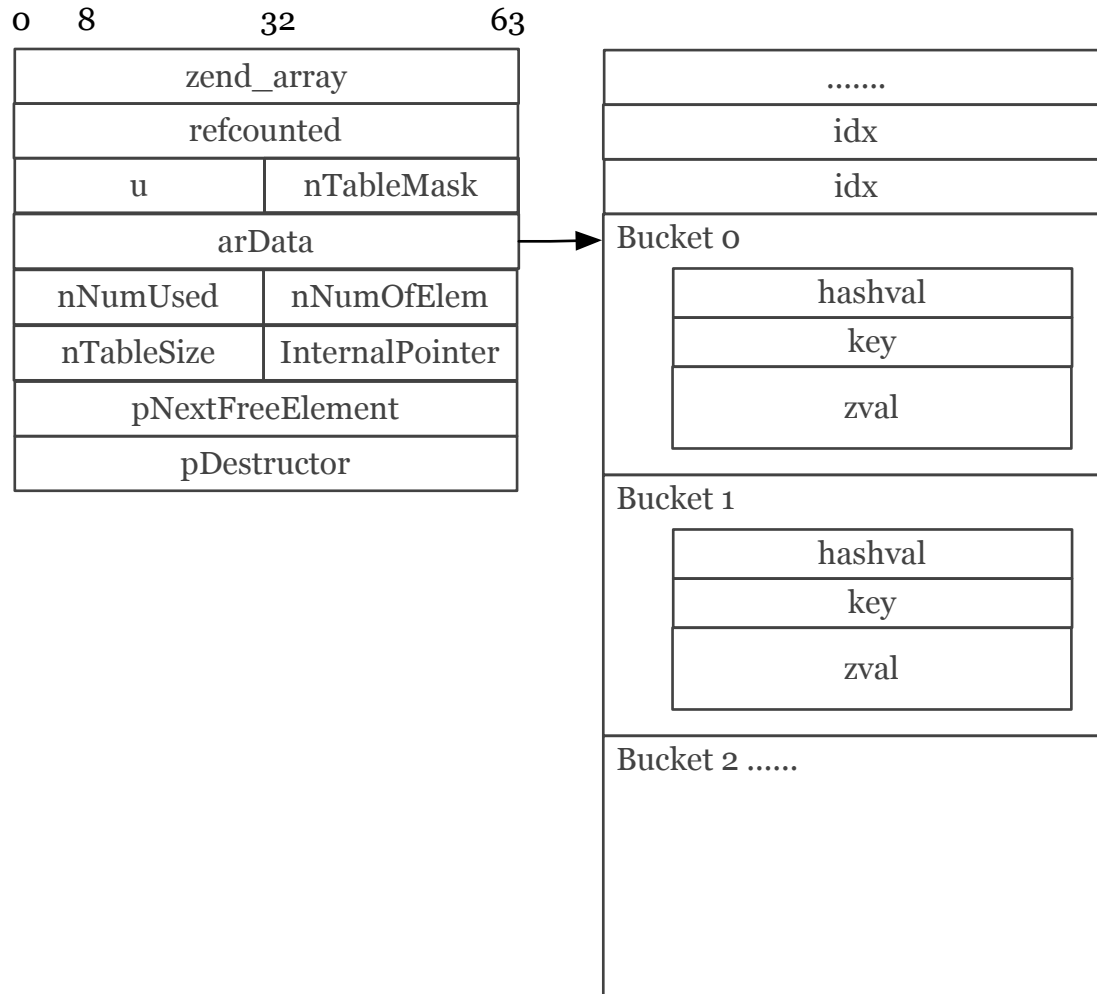
```
struct _zend_reference {  
    zend_refcounted gc;  
    zval val;  
};
```



HashTable – PHP 5



Zend Array – PHP 7



Zend Array(HashTable)



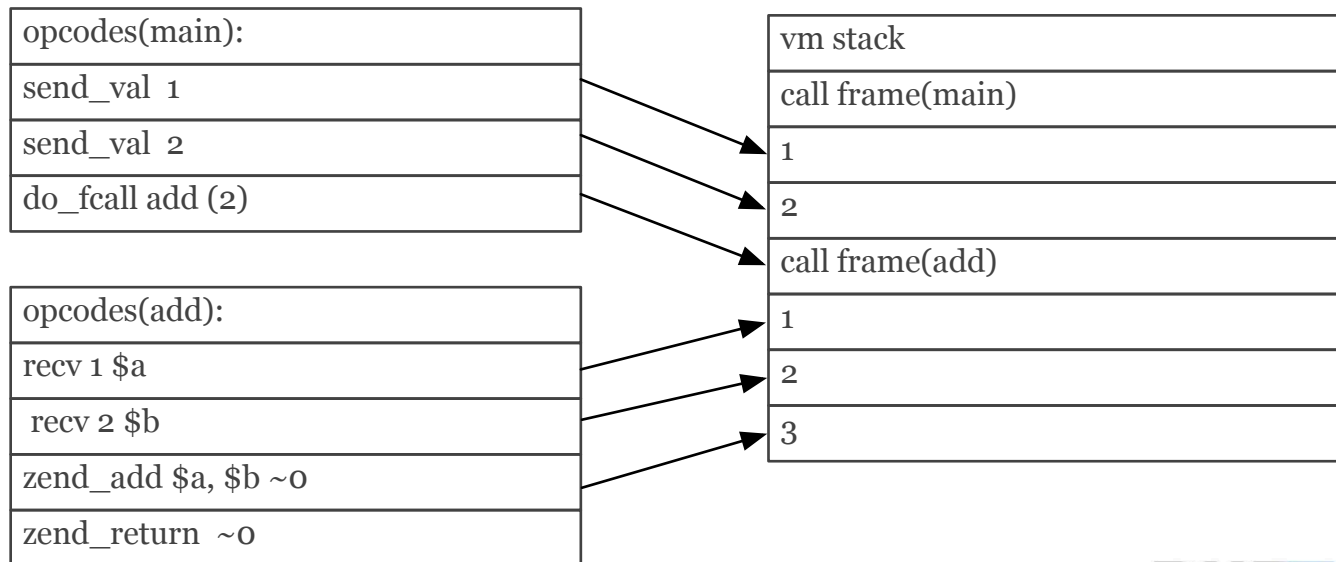
- Values of arrays are zval by default
- HashTable size reduced from 72 to 56 bytes
- Bucket size reduced from 72 to 32 bytes
- Memory for all Buckets is allocated at once
- Bucket.key now is a pointer to zend_string
- Values of array elements are embedded into the Buckets
- Improved data locality =>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

Function calling convention – PHP5



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

add(1, 2);

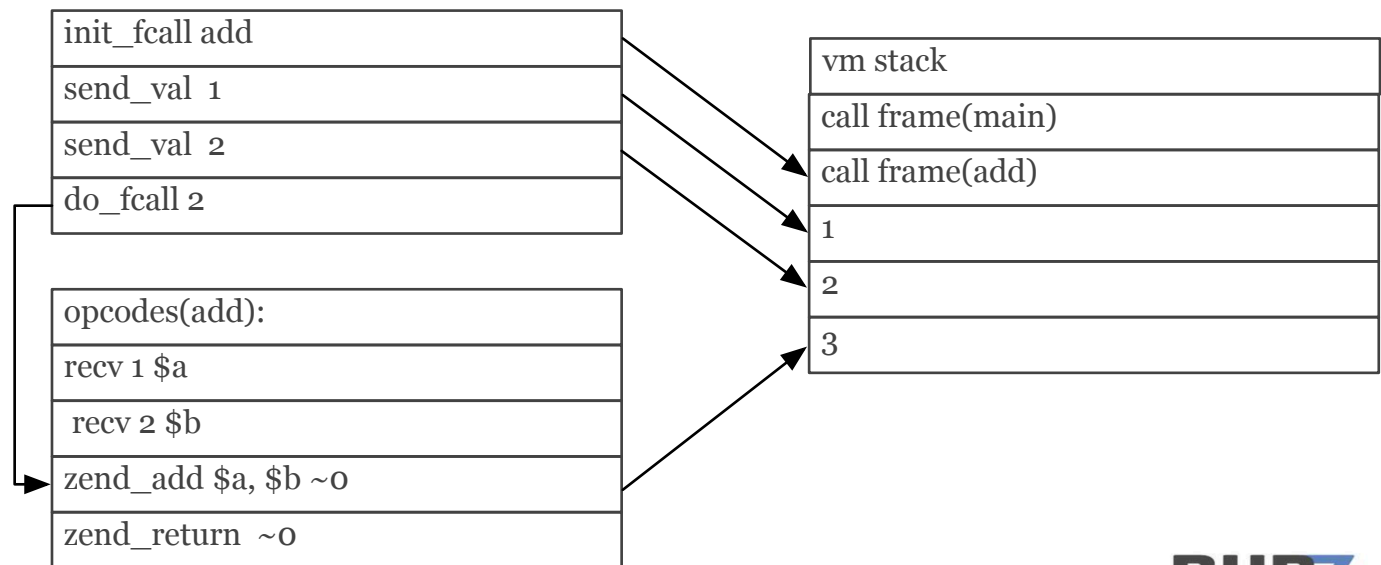


Function calling convention – PHP7



- 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);*

New Memory Manager



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

Dozens of other improvements



- Fast HashTable iteration APIs
- Immutable array
- Array duplication optimization
- PCRE with JIT
- BIND_GLOBAL instead of FETCH and ASSIGN_REF
- More specific 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
- Dozens logic optimizations
-

Wordpress profile (2015-04-14)

/Users/Larurence/Downloads/callgrind.out.16568 [/home/huixincheng/local/php-trunk/bin/php-cgi -T 100 index.php]

Instruction Fetch

Flat Profile

Search: Source File

Self	Source File
40.01	zend_vm_execute.h
12.52	zend_hash.c
3.89	zend_alloc.c
3.52	(unknown)
3.05	block_pass.c
2.23	zend_language_parser.c
2.17	zend_language_scanner.l
2.05	array.c
1.89	php_pcre.c

Incl.	Self	Called	Function	Location
0.00	0.00	102	zend_mm_shutdown	php-cgi: zend_alloc.
0.00	0.00	2 929	_emalloc_8	php-cgi: zend_alloc.
0.00	0.00	195	_safe_erealloc	php-cgi: zend_alloc.
0.00	0.00	4 600	zend_memory_usage	php-cgi: zend_alloc.
0.00	0.00	1 100	_emalloc_64	php-cgi: zend_alloc.
0.00	0.00	900	_emalloc_160	php-cgi: zend_alloc.
0.00	0.00	1	start_memory_manager	php-cgi: zend_alloc.
0.00	0.00	2	zend_mm_chunk_alloc_int...	php-cgi: zend_alloc.
0.00	0.00	1	zend_mm_init	php-cgi: zend_alloc.
0.00	0.00	600	_emalloc_2560	php-cgi: zend_alloc.
0.00	0.00	300	_emalloc_16	php-cgi: zend_alloc.
0.00	0.00	283	_emalloc_256	php-cgi: zend_alloc.
0.00	0.00	8	zend_mm_munmap	php-cgi: zend_alloc.
0.00	0.00	4	zend_mm_mmap	php-cgi: zend_alloc.
0.00	0.00	200	_emalloc_40	php-cgi: zend_alloc.
0.00	0.00	183	_emalloc_192	php-cgi: zend_alloc.
0.00	0.00	83	_emalloc_448	php-cgi: zend_alloc.
0.00	0.00	7	_safe_malloc	php-cgi: zend_alloc.
0.00	0.00	4	zend_strndup	php-cgi: zend_alloc.
0.00	0.00	1	zend_mm_chunk_alloc.isra....	php-cgi: zend_alloc.
0.00	0.00	2	zend_mm_chunk_free.isra.3	php-cgi: zend_alloc.
0.00	0.00	1	zend_set_memory_limit	php-cgi: zend_alloc.

zend_hash_iterator_add

Types Callers All Callers Callee Map Source Code

Incl.	Distance	Called	Caller
100.00	3	(0)	<cycle 8> (php-cgi)
100.00	2	2 400	execute_ex <cycle 8> (php-cgi: zend_vm_execute.h)
100.00	1	2 400	ZEND_FE_RESET_RW_SPEC_CV_HANDLER (php-cgi: zend_vm...
99.93	9-15 (10)	12	0x00000000000012d0 (ld-2.19.so)
99.93	8-14 (9)	12	0x0000000000044b74f (php-cgi)
99.93	7-13 (8)	12	(below main) (libc-2.19.so: libc-start.c)
99.93	6-12 (7)	705	main (php-cgi: cgi_main.c, ...)
98.74	6	100	php_execute_script (php-cgi: main.c, ...)
98.74	5	300	zend_execute_scripts (php-cgi: zend.c)
98.74	4	100	zend_execute (php-cgi: zend_vm_execute.h, ...)
0.58	5-9 (7)	1 100	php_request_shutdown (php-cgi: main.c)
0.51	6-8 (6)	200	zend_deactivate (php-cgi: zend.c)
0.51	5-7 (5)	1 200	shutdown_executor (php-cgi: zend_execute_API.c)
0.49	4	13 000	zend_objects_store_free_object_storage (php-cgi: zend_objects_A...
0.47	6	1	php_cgi_startup (php-cgi: cgi_main.c)
0.47	5	1	php_module_startup (php-cgi: main.c, ...)

lr	Count	Callee
----	-------	--------

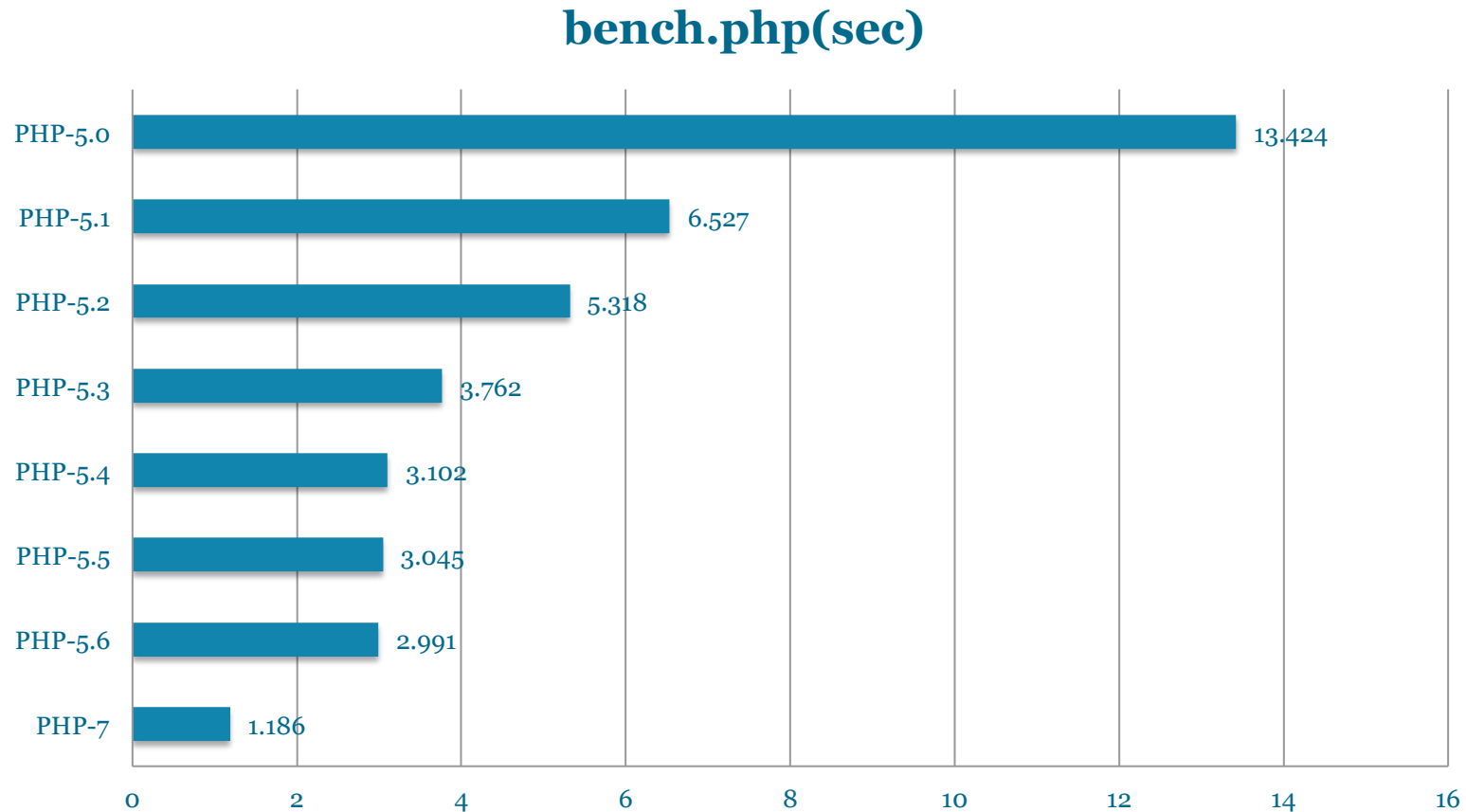
Parts Calleees Call Graph All Calleees Caller Map Machine Code

callgrind.out.16568 [1] - Total Instruction Fetch Cost: 2 504 355 561

Wordpress profiled (2015-04-14)

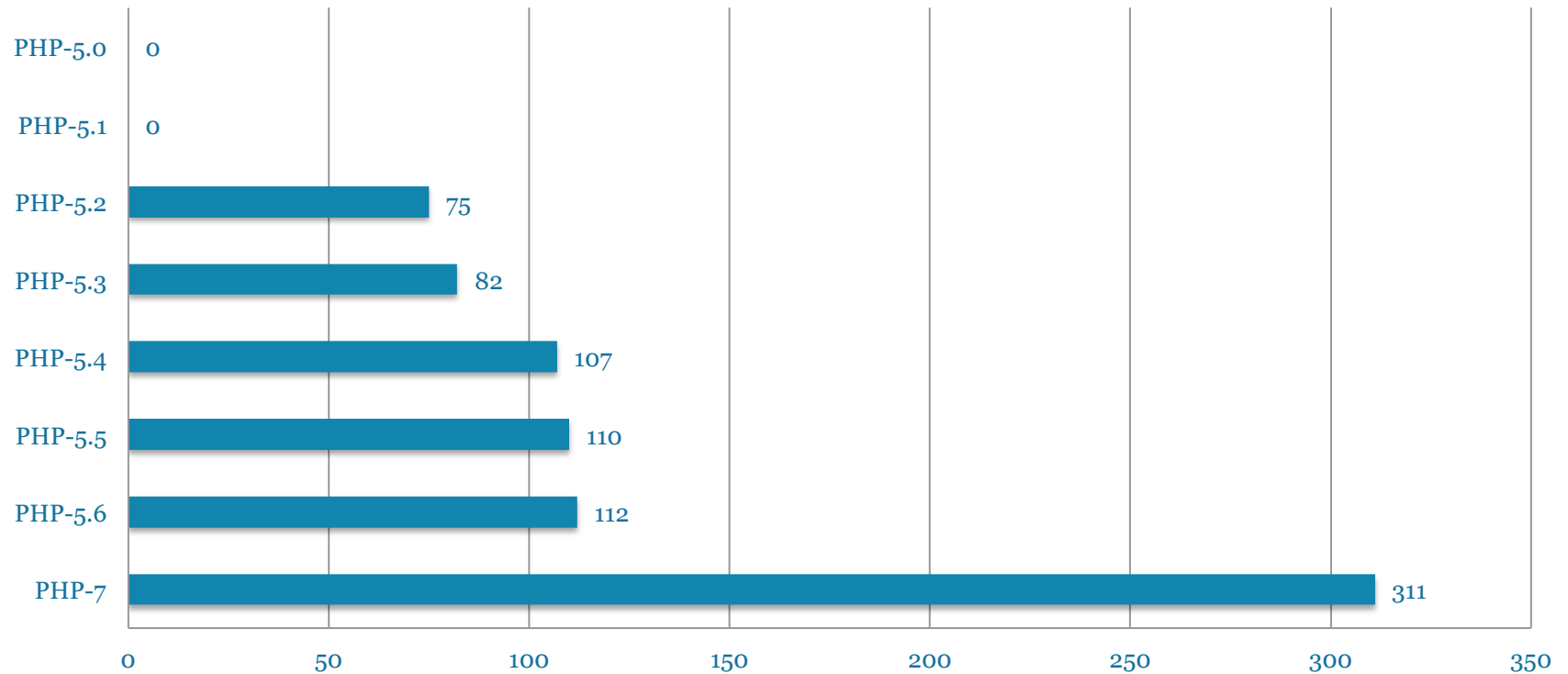
- >50% CPU IRs reduced
- 5% CPU time in meory manager
- 12% CPU time in hash tables operations

PHP7 Performance – Benchmark (2014-04-14)



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

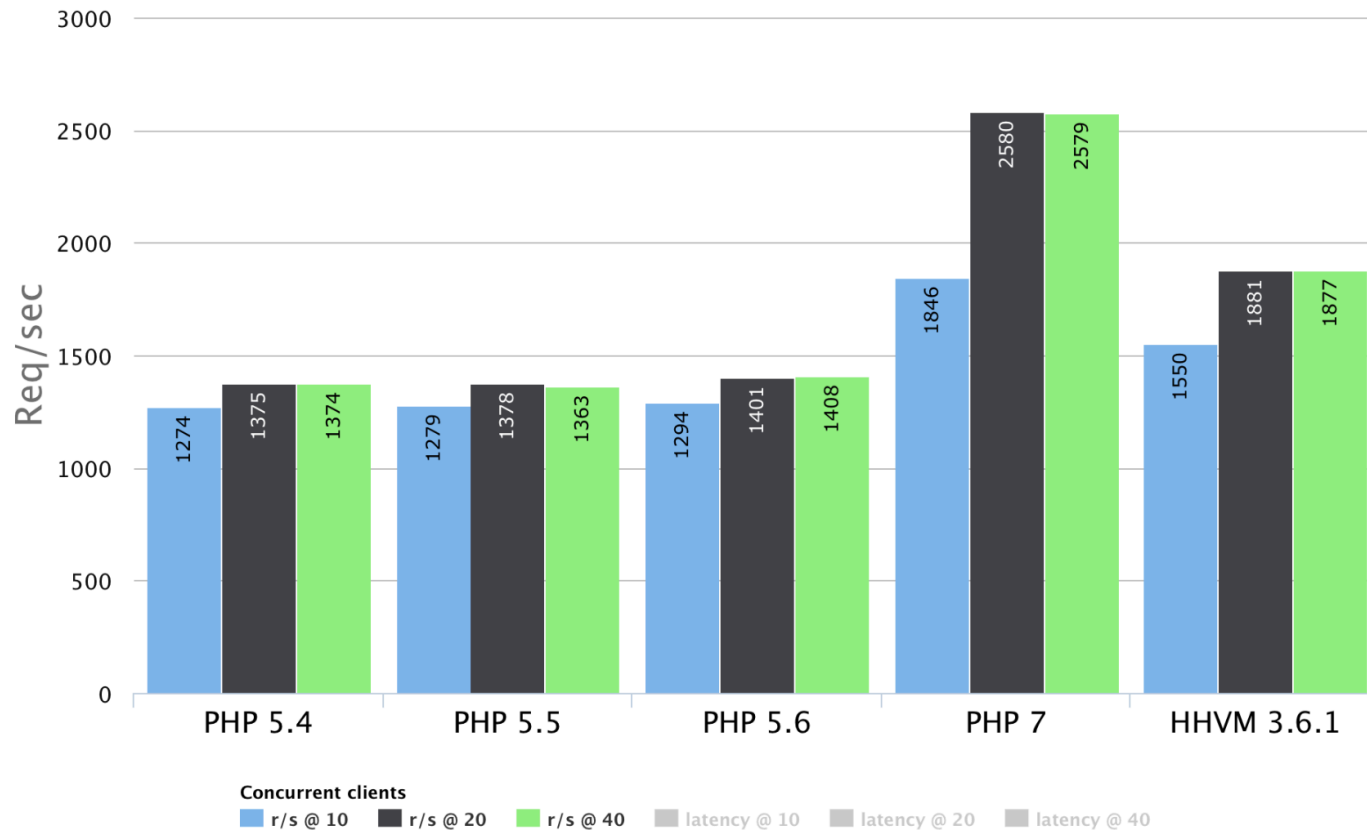
wordpress 3.0.1 home page qps



PHP7 Performance (By Rasmus 2015-04-21)

Drupal 8-git

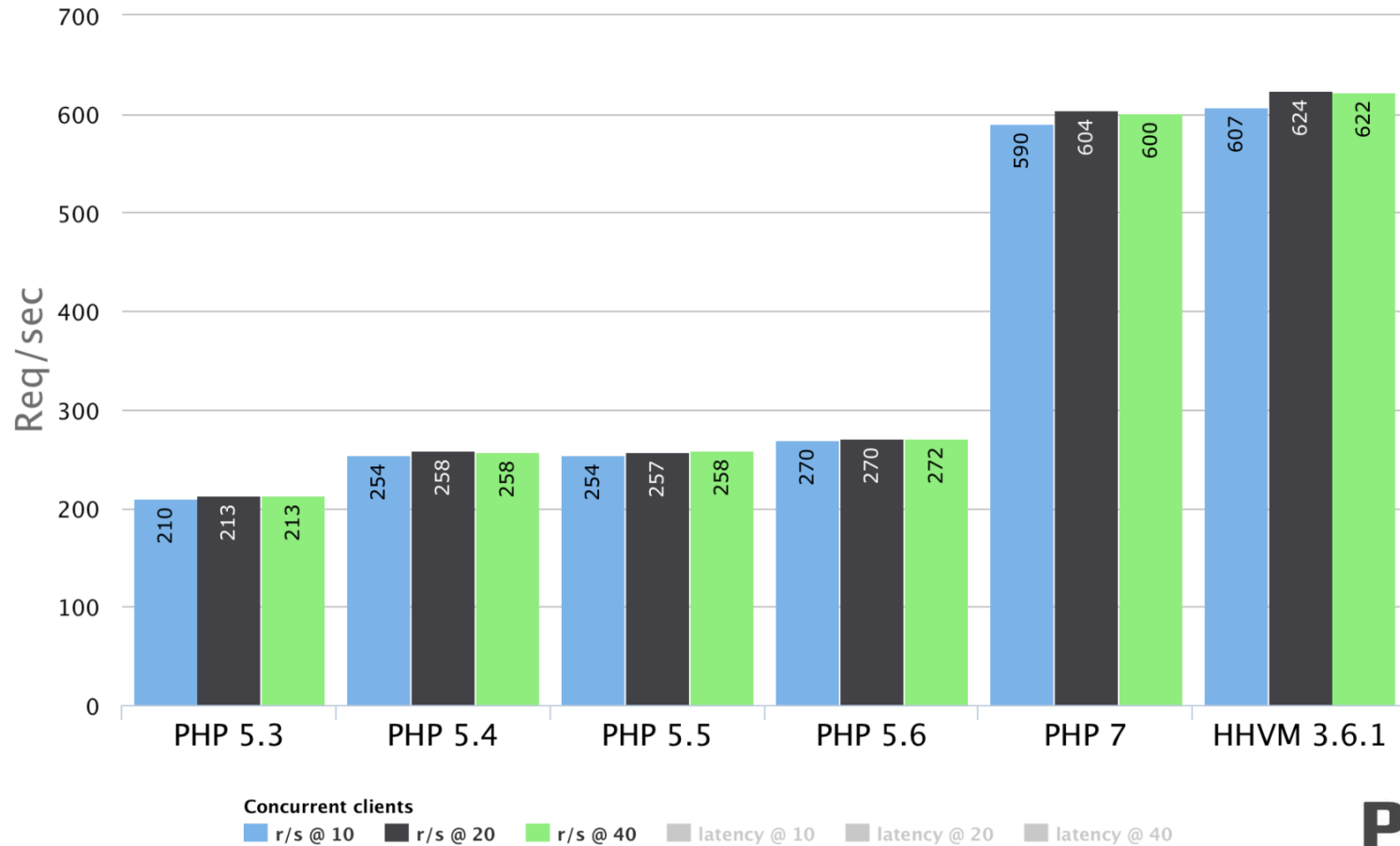
node w/ 5 comments



PHP7 Performance (2015-04-21)

Wordpress-4.1.1

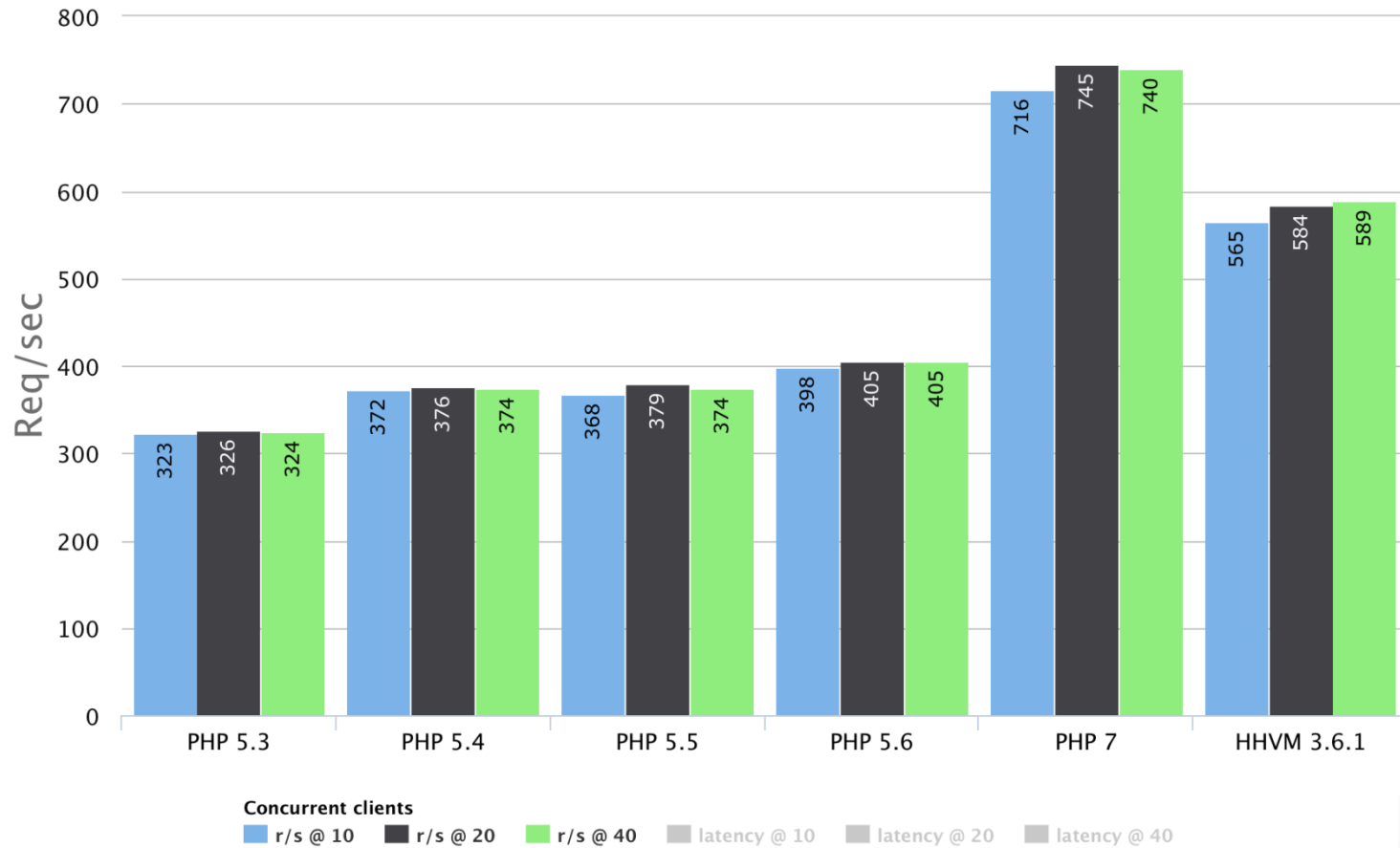
<http://wordpress/?p=1>



PHP7 Performance (2015-04-21)

phpBB 3.1.3

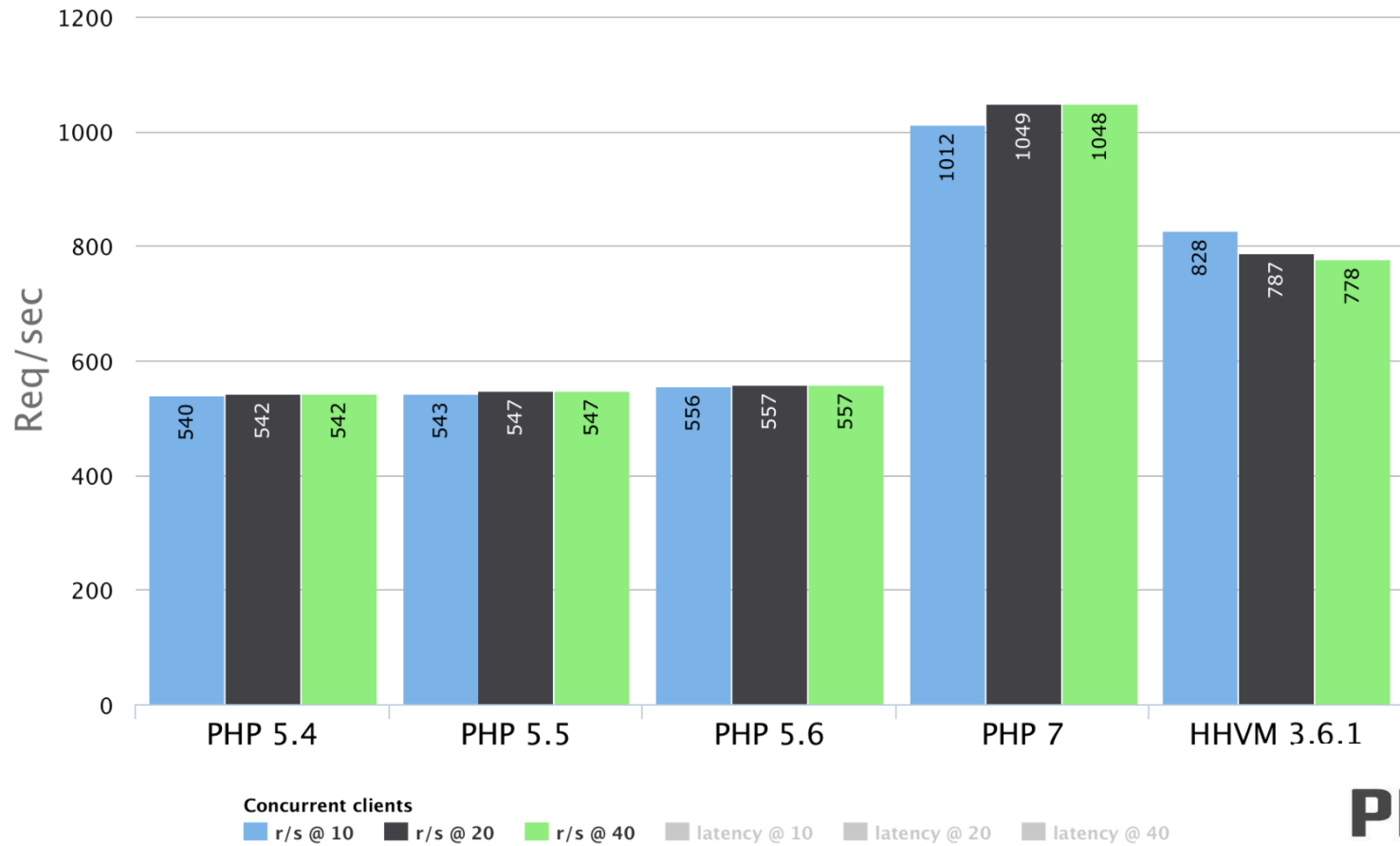
<http://phpbb/viewforum.php?f=2>



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 (oct 2015)
- Restart JIT (Sep 2014)



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