PHPNG

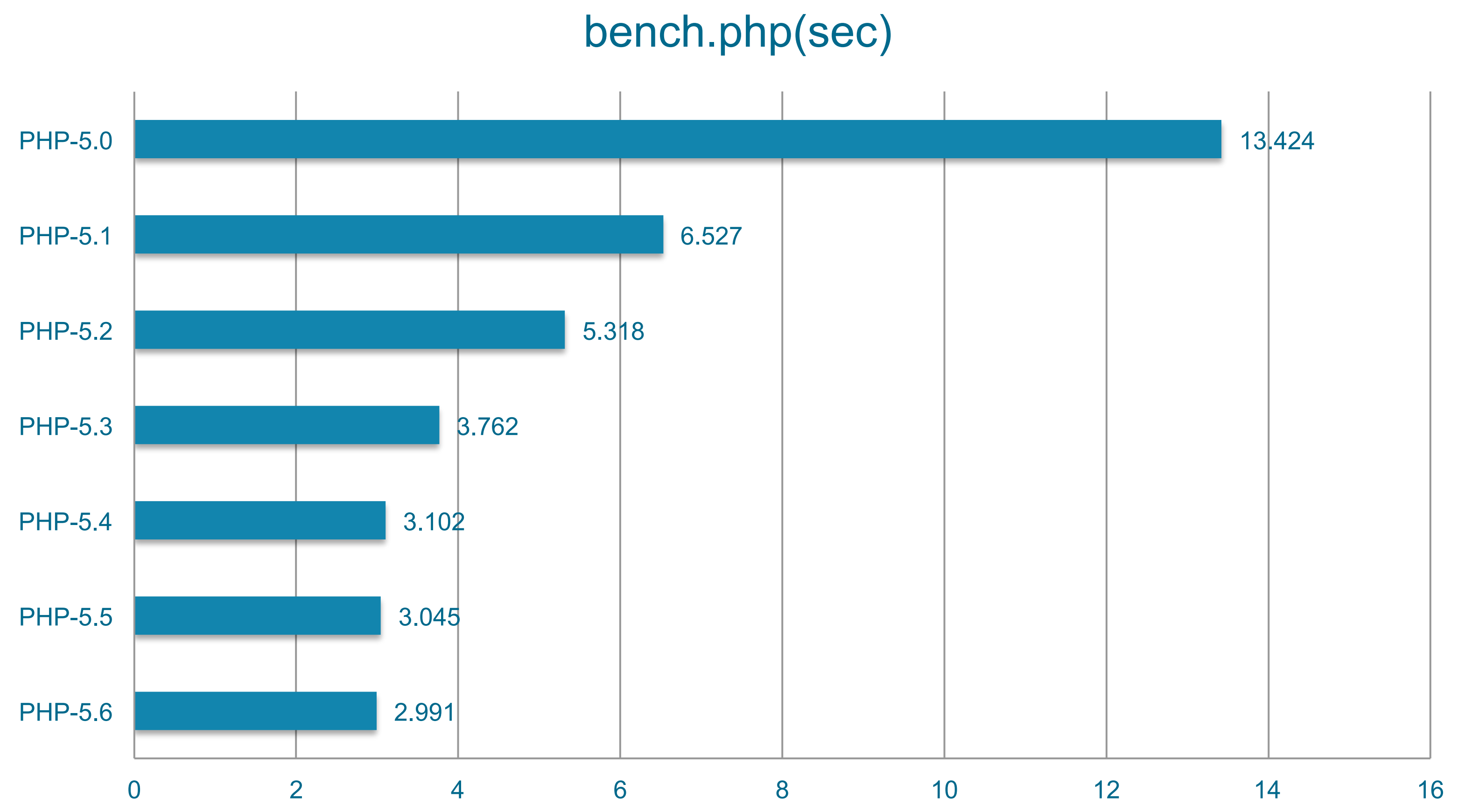
PHP New Engine

Laruence & Dmitry
@laruence
http://www.laruence.com/

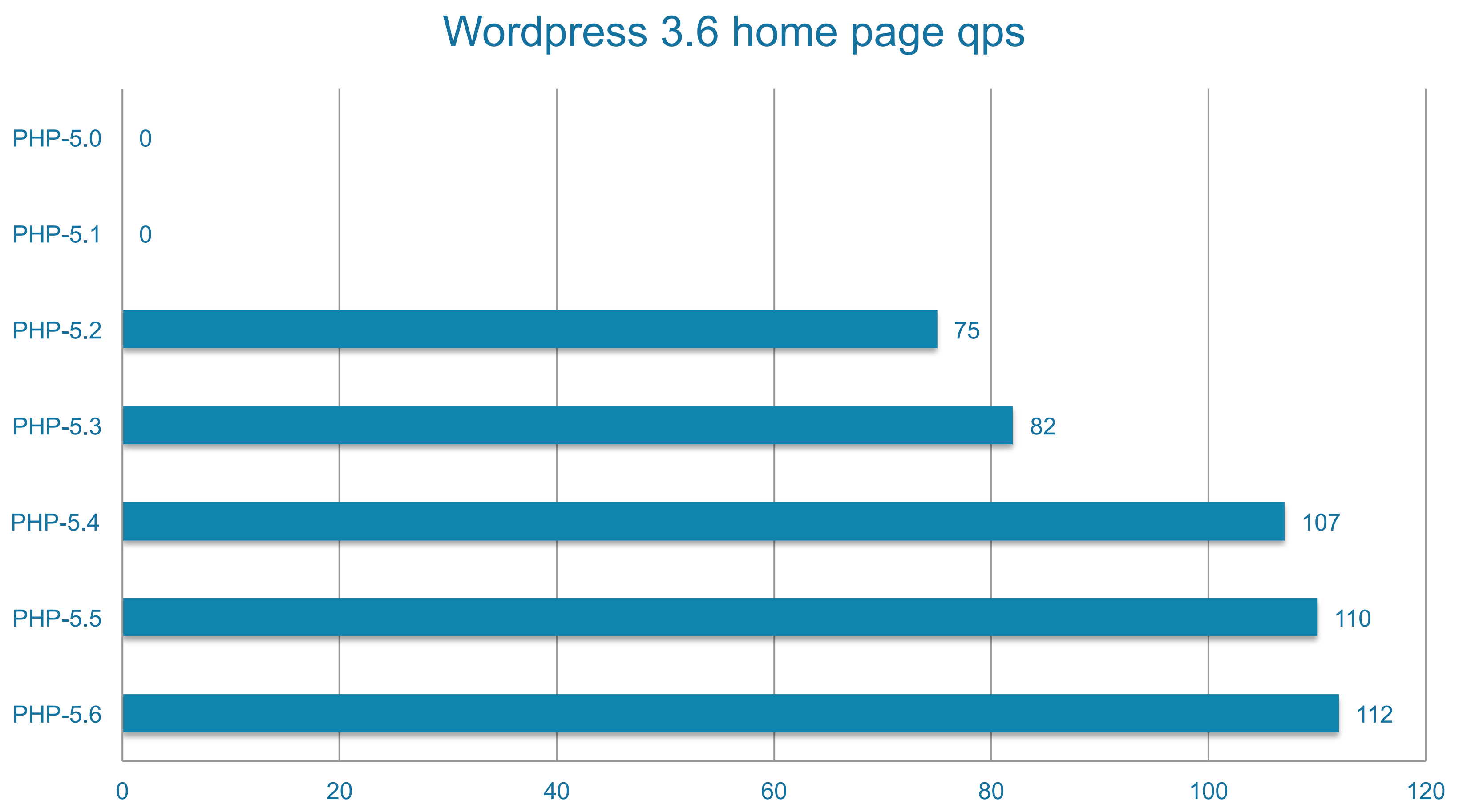
About Me

- 2008 PHP Intern at Yahoo!
- 2010 Author of Yaf
- 2011 PHP core developer
- 2011 Author of taint
- 2012 Author of Yar, Yac
- 2013 Zend consultant
- 2013 PHP llvm-jit (opcache)
- 2014 Works with Dmitry on PHPNG

PHP Performance Evaluation



PHP Performance Evaluation

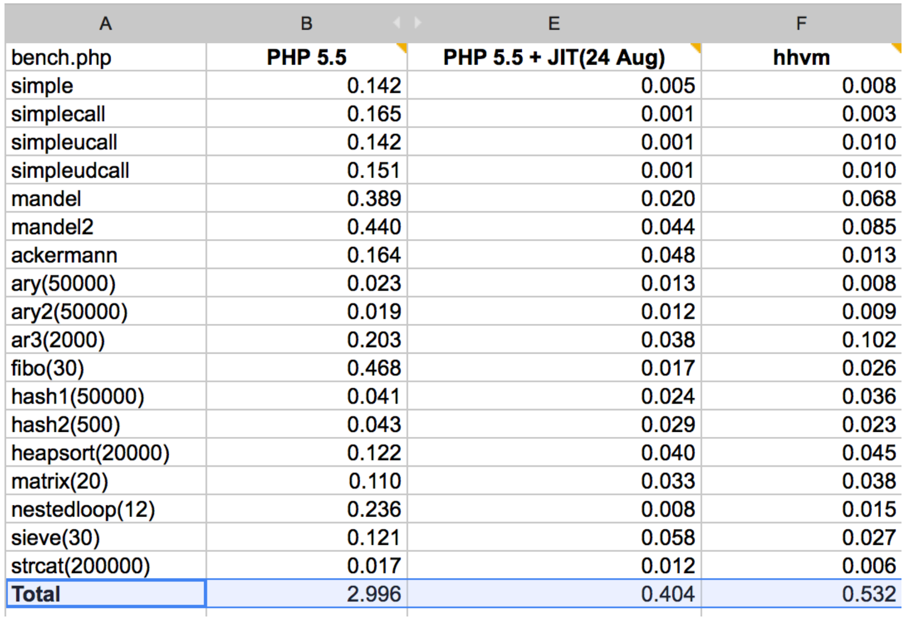


PHP Performance Evaluation

- ~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

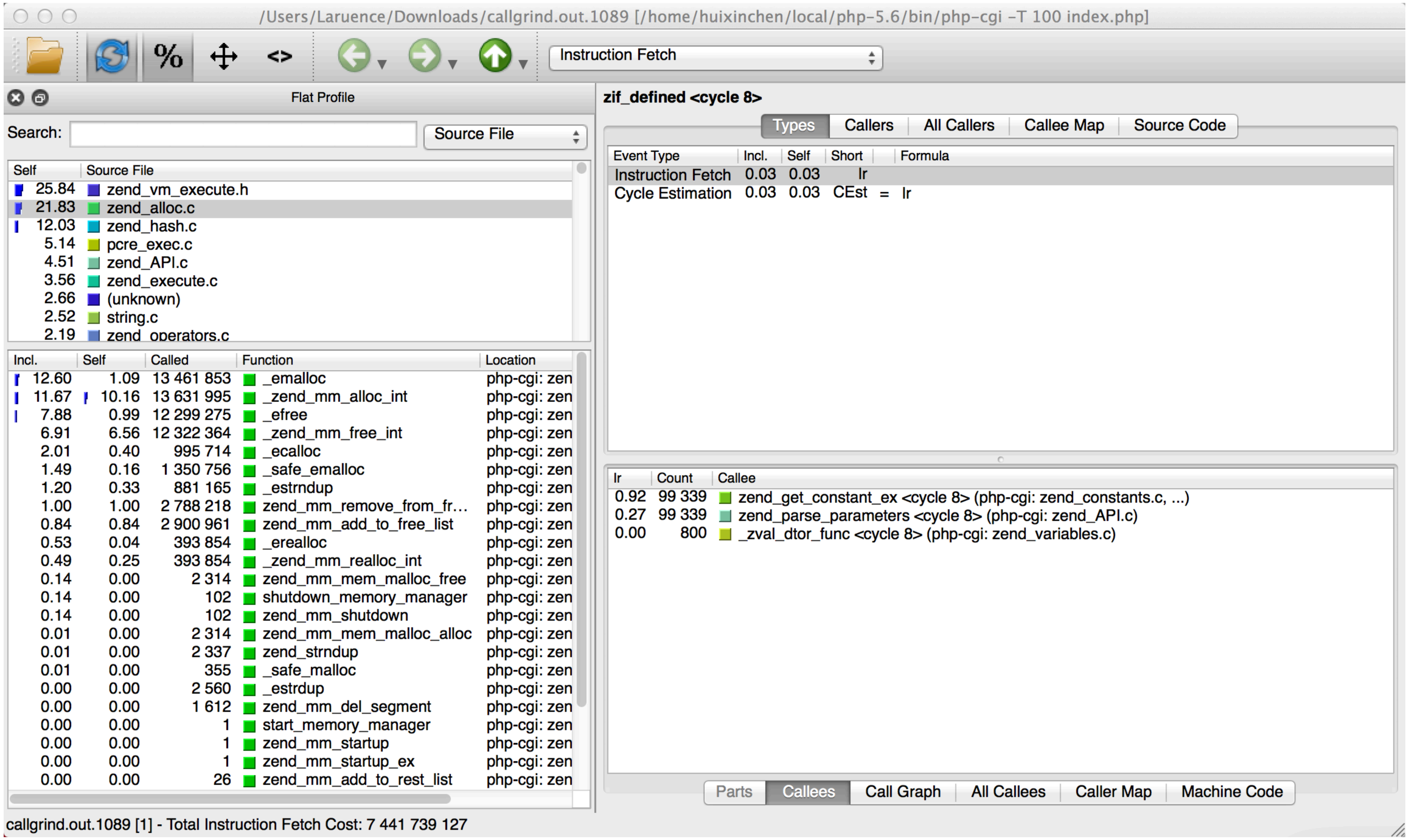
Worked at darkness

- About 2 years were “wasted” on PoC of JIT for PHP-5.5
- We created a POC of JIT compiler based on LLVM
- ~8 times speedup on bench.php
- Negligible speedup on real-life apps (1% on Wordpress)



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 profiled



/Users/Laruence/Downloads/callgrind.out.1089 [/home/huixinchen/local/php-5.6/bin/php-cgi -T 100 index.php]

Instruction Fetch

Flat Profile

Search: Source File

Self	Source File
25.84	zend_vm_execute.h
21.83	zend_alloc.c
12.03	zend_hash.c
5.14	pcre_exec.c
4.51	zend_API.c
3.56	zend_execute.c
2.66	(unknown)
2.52	string.c
2.19	zend_operators.c

Incl.	Self	Called	Function	Location
12.60	1.09	13 461 853	_emalloc	php-cgi: zen
11.67	10.16	13 631 995	_zend_mm_alloc_int	php-cgi: zen
7.88	0.99	12 299 275	_efree	php-cgi: zen
6.91	6.56	12 322 364	_zend_mm_free_int	php-cgi: zen
2.01	0.40	995 714	_ecalloc	php-cgi: zen
1.49	0.16	1 350 756	_safe_emalloc	php-cgi: zen
1.20	0.33	881 165	_estrndup	php-cgi: zen
1.00	1.00	2 788 218	zend_mm_remove_from_fr…	php-cgi: zen
0.84	0.84	2 900 961	zend_mm_add_to_free_list	php-cgi: zen
0.53	0.04	393 854	_erealloc	php-cgi: zen
0.49	0.25	393 854	_zend_mm_realloc_int	php-cgi: zen
0.14	0.00	2 314	zend_mm_mem_malloc_free	php-cgi: zen
0.14	0.00	102	shutdown_memory_manager	php-cgi: zen
0.14	0.00	102	zend_mm_shutdown	php-cgi: zen
0.01	0.00	2 314	zend_mm_mem_malloc_alloc	php-cgi: zen
0.01	0.00	2 337	zend_strndup	php-cgi: zen
0.01	0.00	355	_safe_malloc	php-cgi: zen
0.00	0.00	2 560	_estrdup	php-cgi: zen
0.00	0.00	1 612	zend_mm_del_segment	php-cgi: zen
0.00	0.00	1	start_memory_manager	php-cgi: zen
0.00	0.00	1	zend_mm_startup	php-cgi: zen
0.00	0.00	1	zend_mm_startup_ex	php-cgi: zen
0.00	0.00	26	zend_mm_add_to_rest_list	php-cgi: zen

zif_defined <cycle 8>

Event Type	Incl.	Self	Short	Formula
Instruction Fetch	0.03	0.03	Ir	
Cycle Estimation	0.03	0.03	CEst	= Ir

Ir	Count	Callee
0.92	99 339	zend_get_constant_ex <cycle 8> (php-cgi: zend_constants.c, ...)
0.27	99 339	zend_parse_parameters <cycle 8> (php-cgi: zend_API.c)
0.00	800	_zval_dtor_func <cycle 8> (php-cgi: zend_variables.c)

callgrind.out.1089 [1] - Total Instruction Fetch Cost: 7 441 739 127

Wordpress profiled

- ~20% of the CPU time in memory manager
- ~10% doing hash tables
- ~30% in internal functions
- ~30% in VM

PHP 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

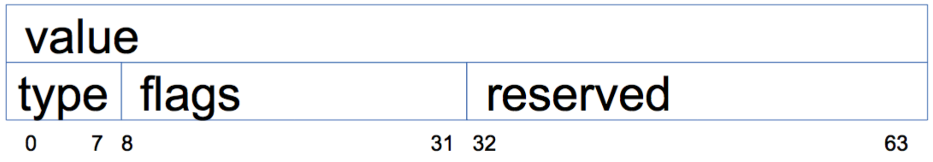
ZVAL

```
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
```

U2 -> Reserved

ZVAL



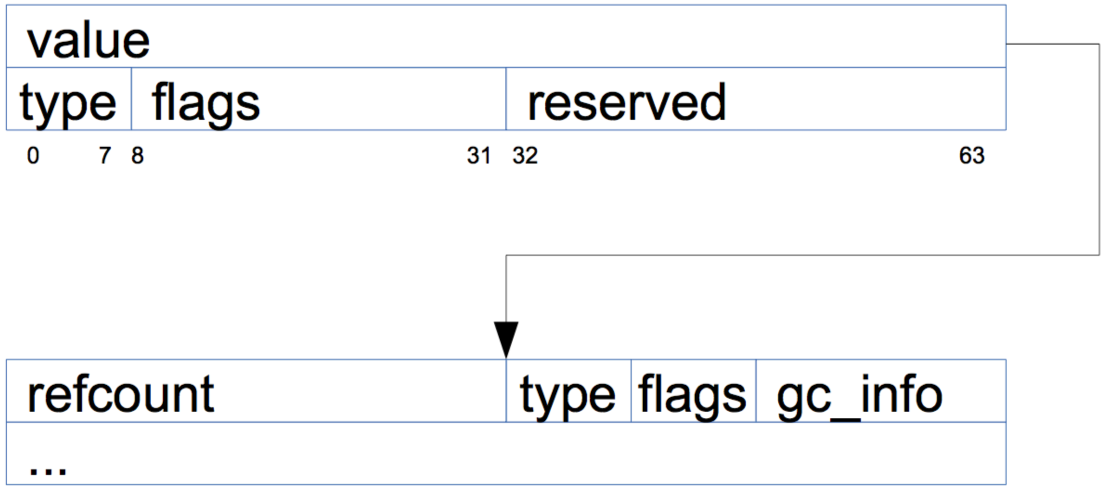
- IS_TYPE_CONSTANT
- IS_TYPE_REFCOUNTED
- IS_TYPE_COLLECTABLE
- IS_TYPE_COPYABLE
- IS_TYPE_IMMUTABLE

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

ZVAL

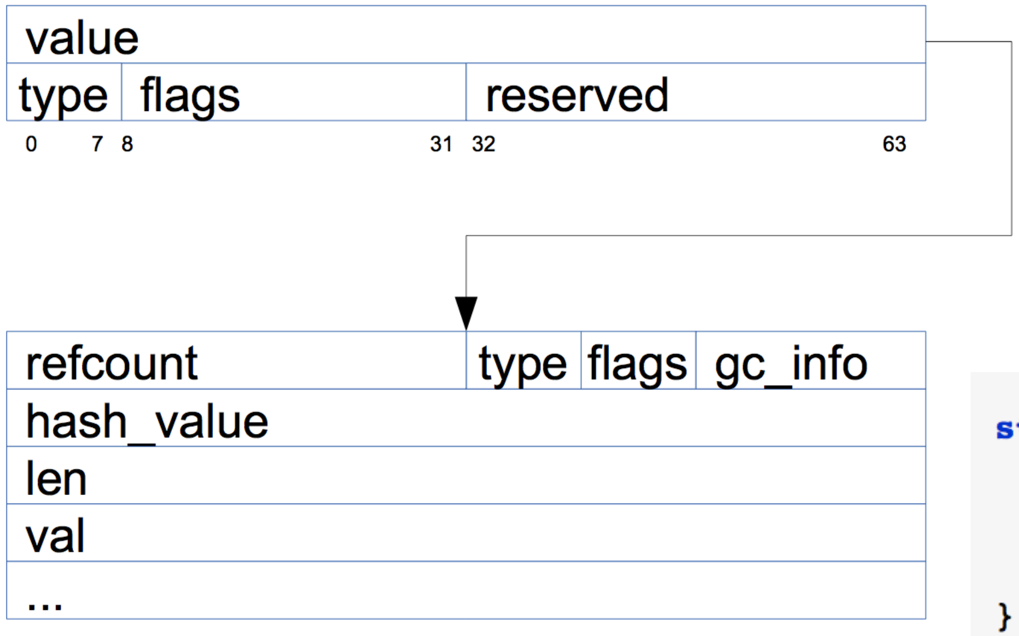
- No more:
    - MAKE_STD_ZVAL
    - zval **
    - temp_varaible
    - zend_literal
    - pListNext, pListLast

ZVAL REFCOUNTED



- IS_STRING
- IS_ARRAY
- IS_OBJECT
- IS_RESOURCE
- IS_REFERENCE

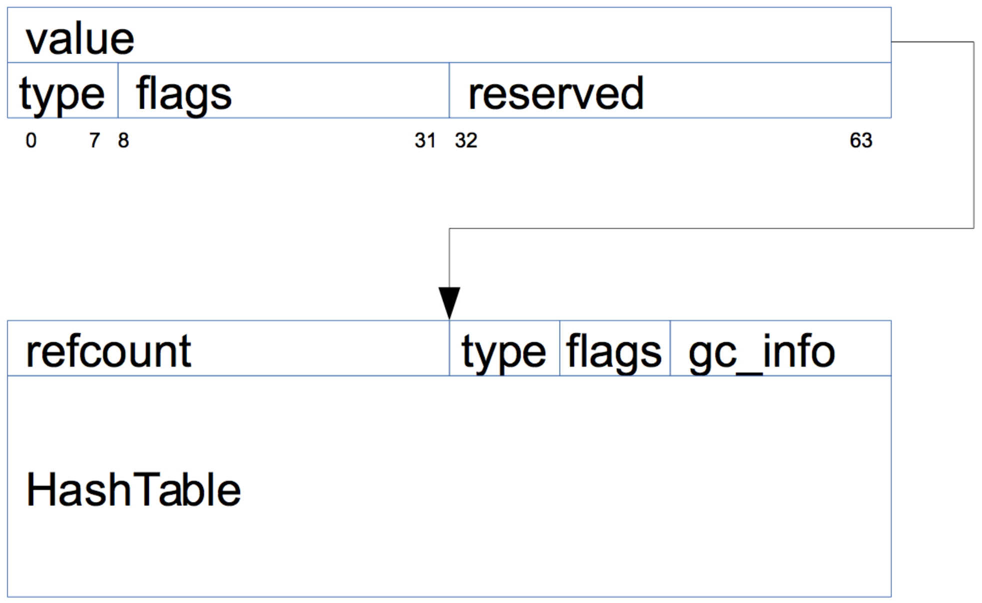
ZVAL IS_STRING



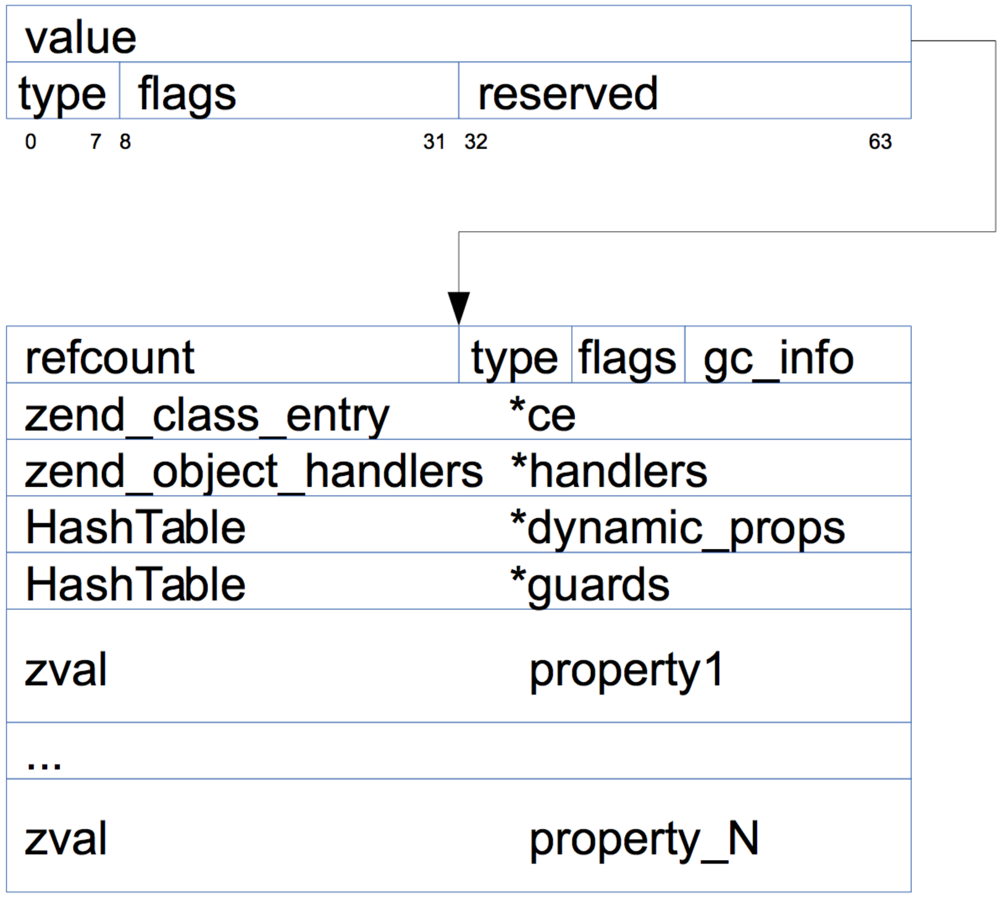
- IS_STR_PERSISTENT
- IS_STR_INTERNED
- IS_STR_PERMANENT
- IS_STR_CONSTANT

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

ZVAL IS_ARRAY

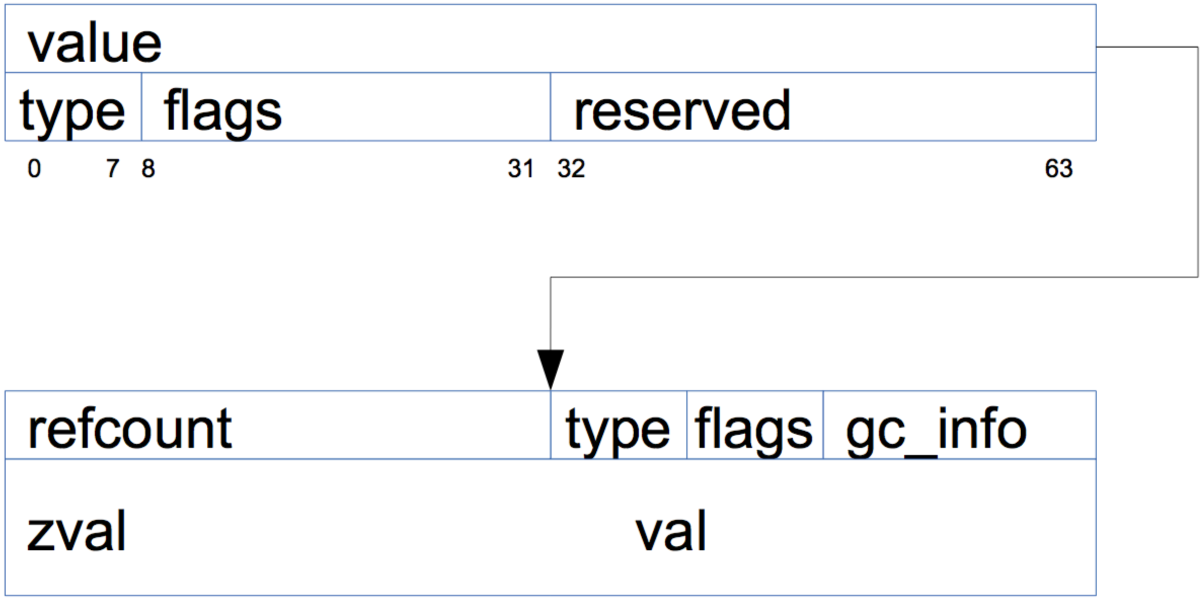


ZVAL IS_OBJECT

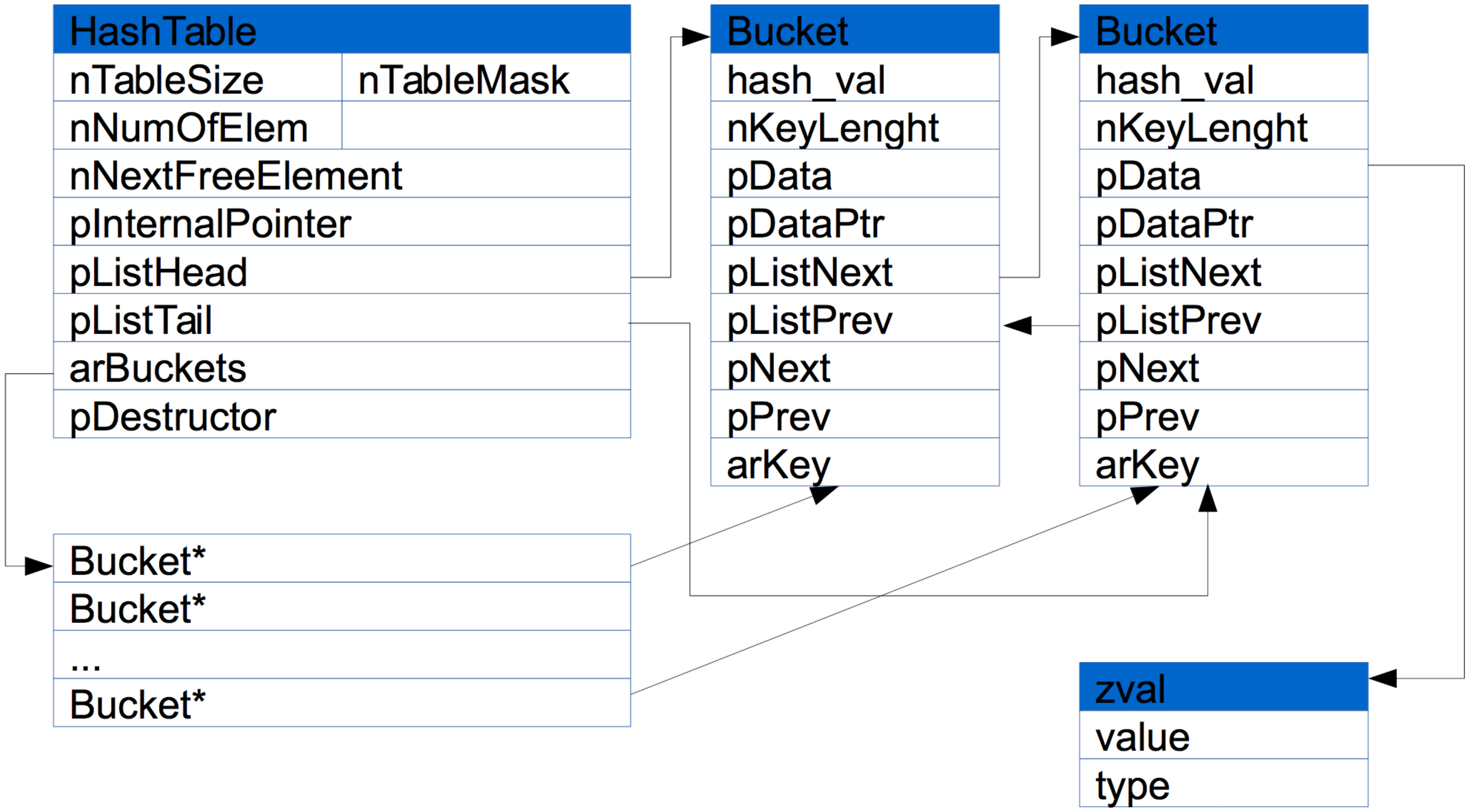


- IS_OBJ_DESTROYED
- IS_OBJ_FREED

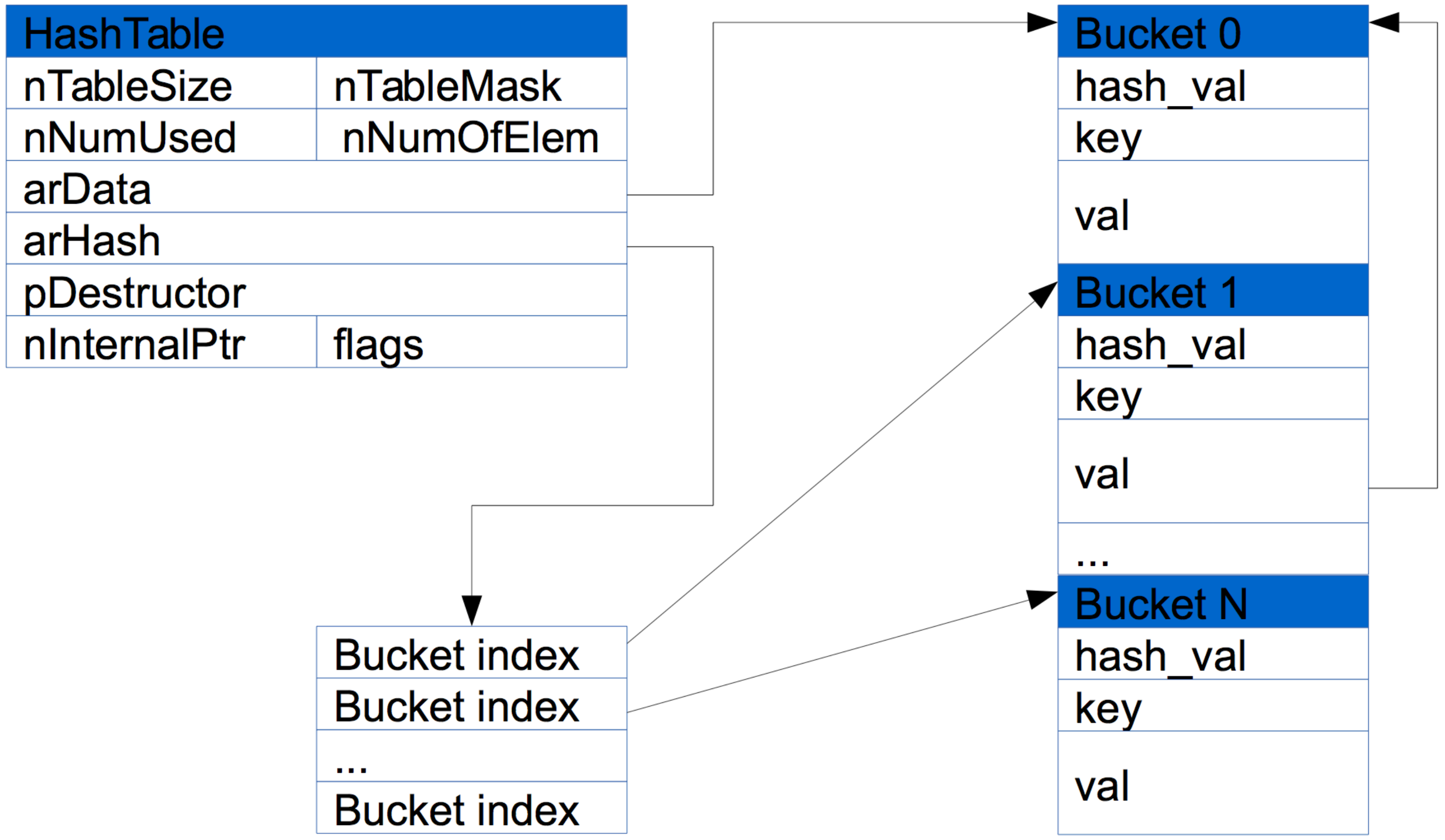
ZVAL IS_REFERENCE



HashTable – PHP5



HashTable – PHPNG



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

Immutable array

```
$a = array();
for ($i = 0; $i < 1000000; $i++) $a[$i] = array("hello");
echo memory_get_usage(true);
```

	PHP	PHPNG
Memory Usage	428 MB	33 MB
Time	0.49 sec	0.06 sec

```
if (in array($color, array(“red”, “yellow”, “green”)) {
    ...
}
```

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();
```

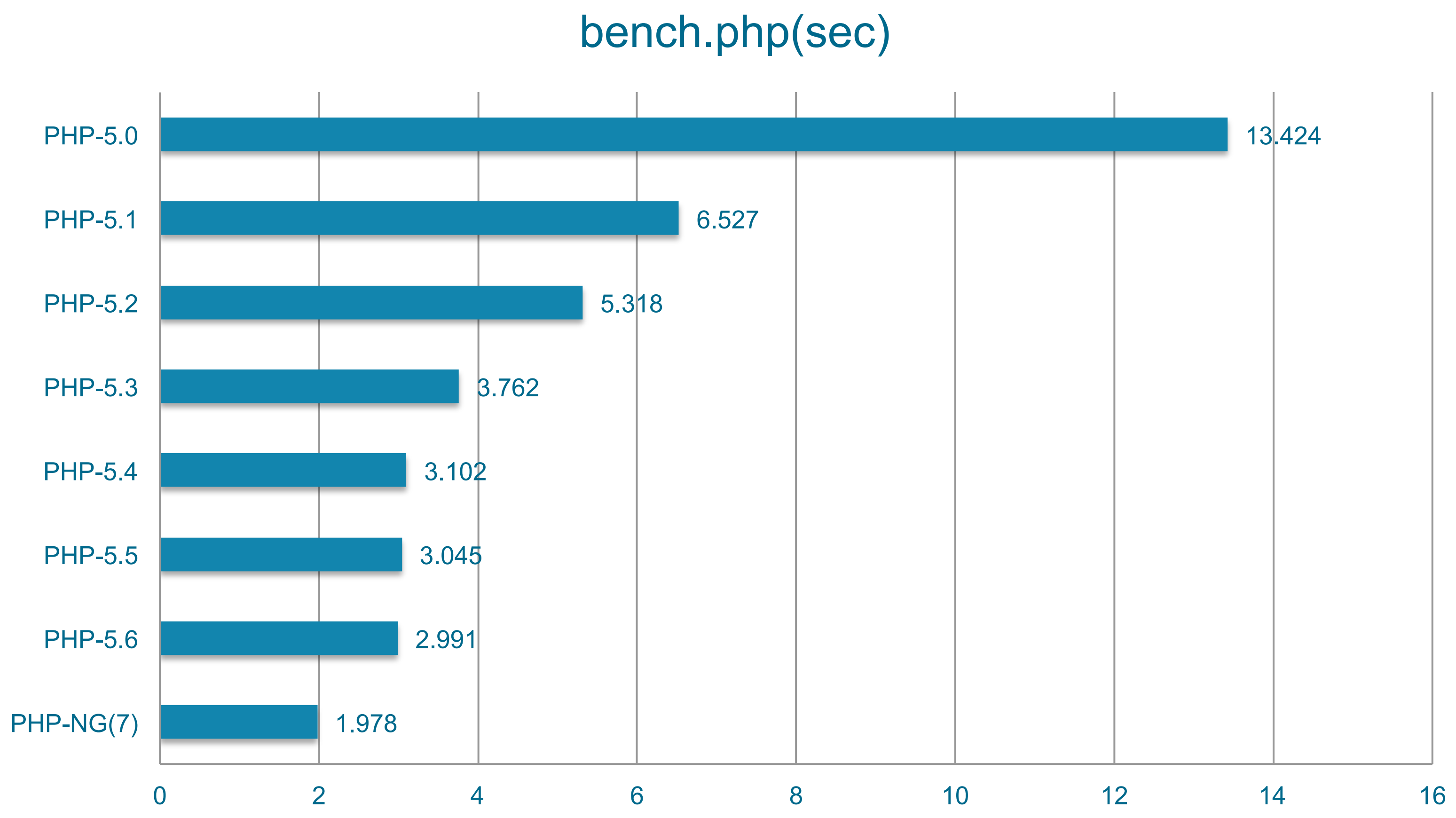
Inline Frequently used simple functions

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

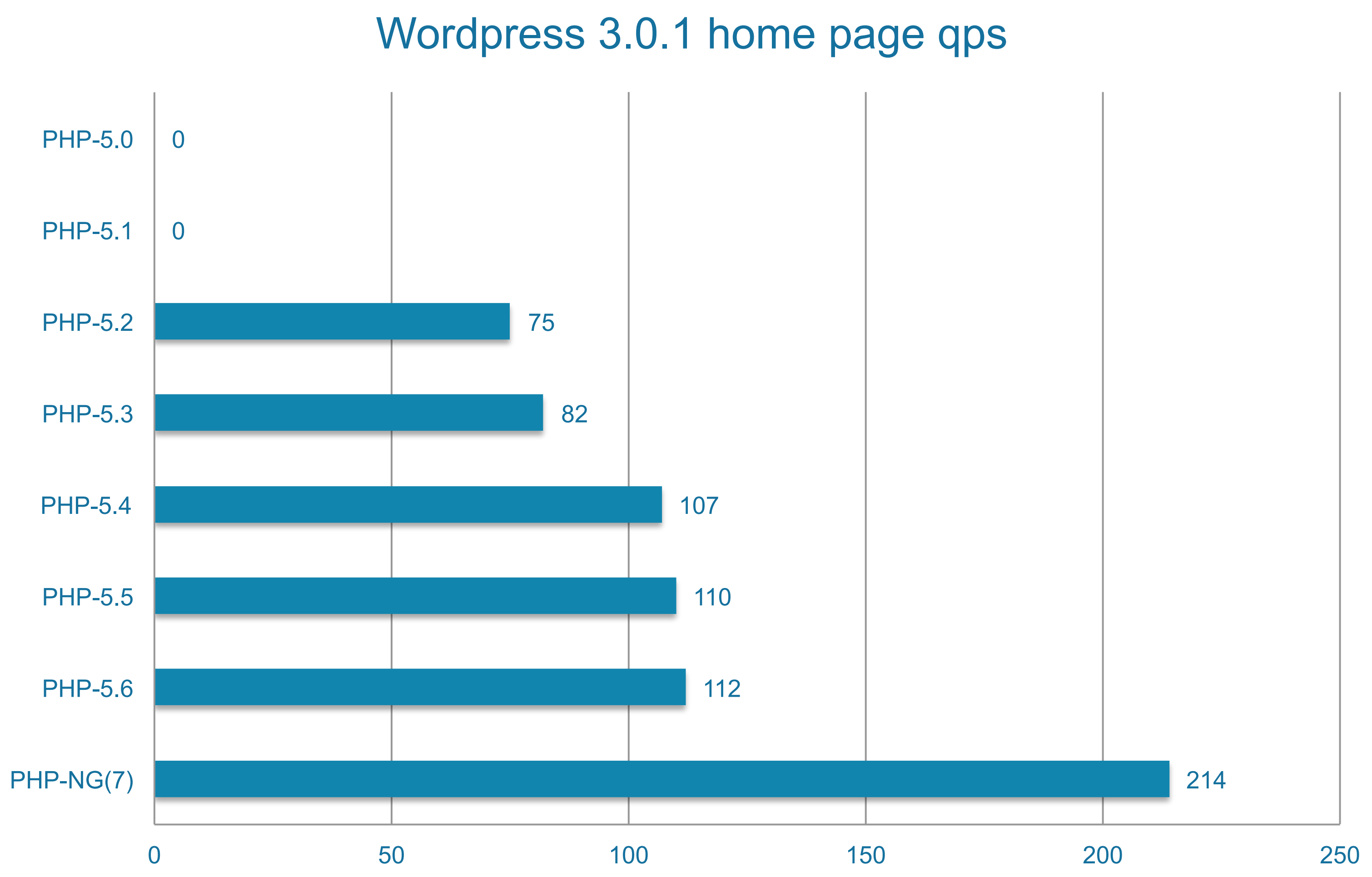
Small Optimaztions

- New HashTable iteration AP
- Array duplication optimization
- Reference-counting instead of copying
- PCRE with JIT
- ……..

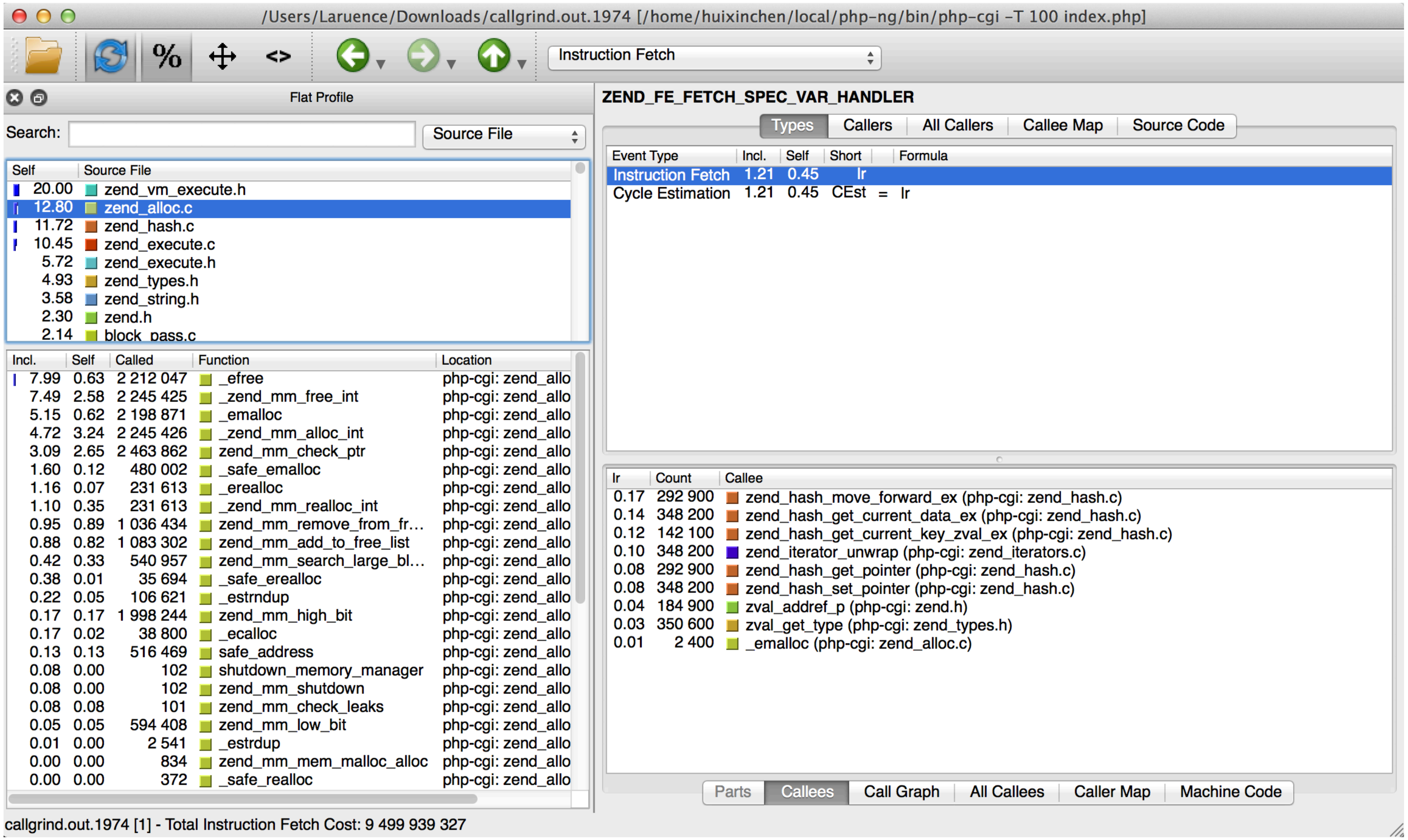
PHPNG Performance



PHPNG Performance



Wordpress profiled



Self	Source File
20.00	zend_vm_execute.h
12.80	zend_alloc.c
11.72	zend_hash.c
10.45	zend_execute.c
5.72	zend_execute.h
4.93	zend_types.h
3.58	zend_string.h
2.30	zend.h
2.14	block_pass.c

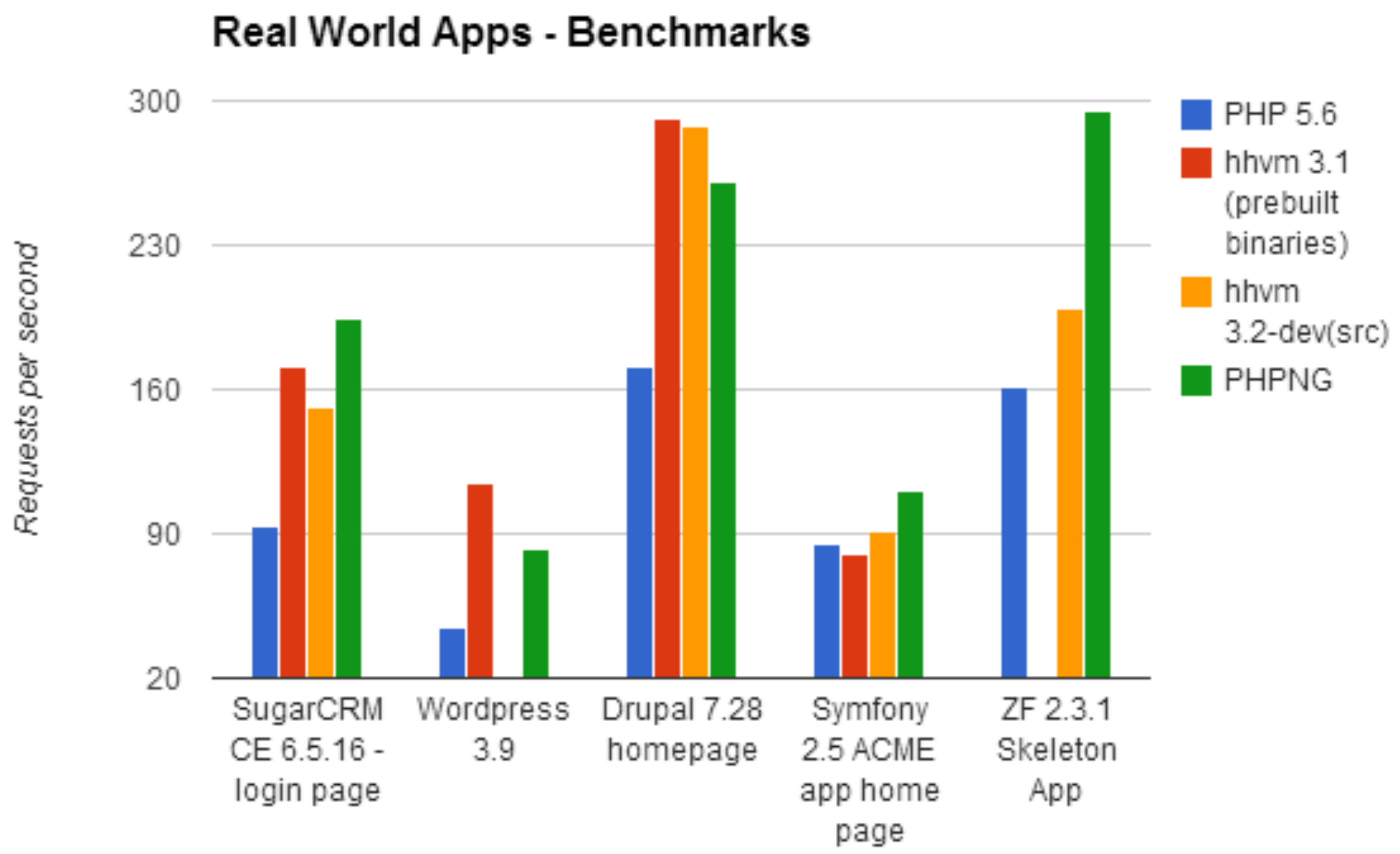
Incl.	Self	Called	Function	Location
7.99	0.63	2 212 047	_efree	php-cgi: zend_allo
7.49	2.58	2 245 425	_zend_mm_free_int	php-cgi: zend_allo
5.15	0.62	2 198 871	_emalloc	php-cgi: zend_allo
4.72	3.24	2 245 426	_zend_mm_alloc_int	php-cgi: zend_allo
3.09	2.65	2 463 862	zend_mm_check_ptr	php-cgi: zend_allo
1.60	0.12	480 002	_safe_emalloc	php-cgi: zend_allo
1.16	0.07	231 613	_erealloc	php-cgi: zend_allo
1.10	0.35	231 613	_zend_mm_realloc_int	php-cgi: zend_allo
0.95	0.89	1 036 434	zend_mm_remove_from_fr…	php-cgi: zend_allo
0.88	0.82	1 083 302	zend_mm_add_to_free_list	php-cgi: zend_allo
0.42	0.33	540 957	zend_mm_search_large_bl…	php-cgi: zend_allo
0.38	0.01	35 694	_safe_erealloc	php-cgi: zend_allo
0.22	0.05	106 621	_estrndup	php-cgi: zend_allo
0.17	0.17	1 998 244	zend_mm_high_bit	php-cgi: zend_allo
0.17	0.02	38 800	_ecalloc	php-cgi: zend_allo
0.13	0.13	516 469	safe_address	php-cgi: zend_allo
0.08	0.00	102	shutdown_memory_manager	php-cgi: zend_allo
0.08	0.00	102	zend_mm_shutdown	php-cgi: zend_allo
0.08	0.08	101	zend_mm_check_leaks	php-cgi: zend_allo
0.05	0.05	594 408	zend_mm_low_bit	php-cgi: zend_allo
0.01	0.00	2 541	_estrdup	php-cgi: zend_allo
0.00	0.00	834	zend_mm_mem_malloc_alloc	php-cgi: zend_allo
0.00	0.00	372	_safe_realloc	php-cgi: zend_allo

Event Type	Incl.	Self	Short	Formula
Instruction Fetch	1.21	0.45	Ir	
Cycle Estimation	1.21	0.45	CEst	= Ir

Ir	Count	Callee
0.17	292 900	zend_hash_move_forward_ex (php-cgi: zend_hash.c)
0.14	348 200	zend_hash_get_current_data_ex (php-cgi: zend_hash.c)
0.12	142 100	zend_hash_get_current_key_zval_ex (php-cgi: zend_hash.c)
0.10	348 200	zend_iterator_unwrap (php-cgi: zend_iterators.c)
0.08	292 900	zend_hash_get_pointer (php-cgi: zend_hash.c)
0.08	348 200	zend_hash_set_pointer (php-cgi: zend_hash.c)
0.04	184 900	zval_addref_p (php-cgi: zend.h)
0.03	350 600	zval_get_type (php-cgi: zend_types.h)
0.01	2 400	_emalloc (php-cgi: zend_alloc.c)

callgrind.out.1974 [1] - Total Instruction Fetch Cost: 9 499 939 327

PHPNG Performance



PHPNG Next

- Merge into main-stream PHP branch
- Solve few incompatibility problems
- Port more extensions
- Don't make new language features to break the performance
- Release PHP-Next (mid 2015)
- Restart JIT ?

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

Q&A