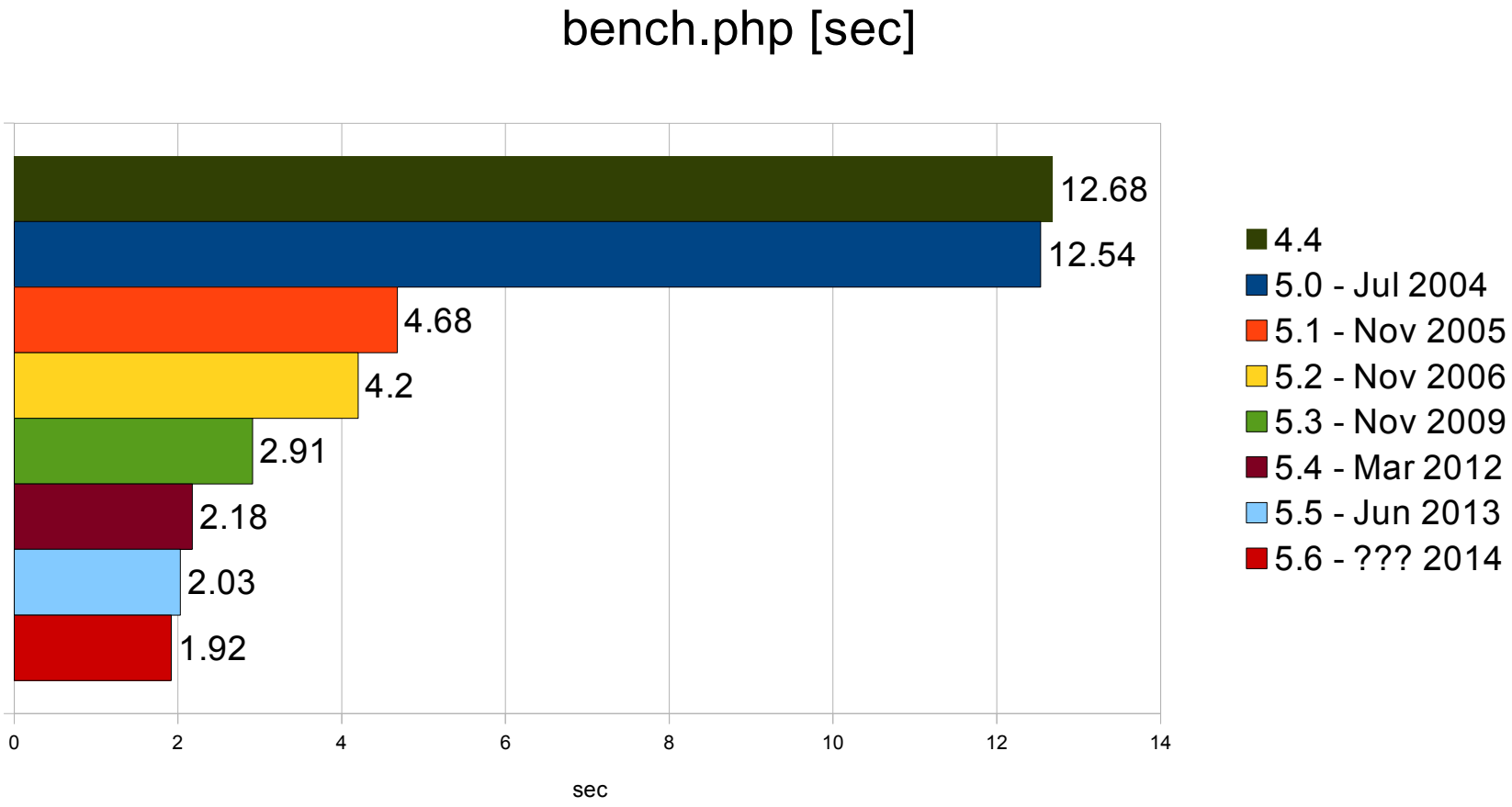
PHPNG – новый движок для старого PHP.

Dmitry Stogov (dmitry@zend.com)

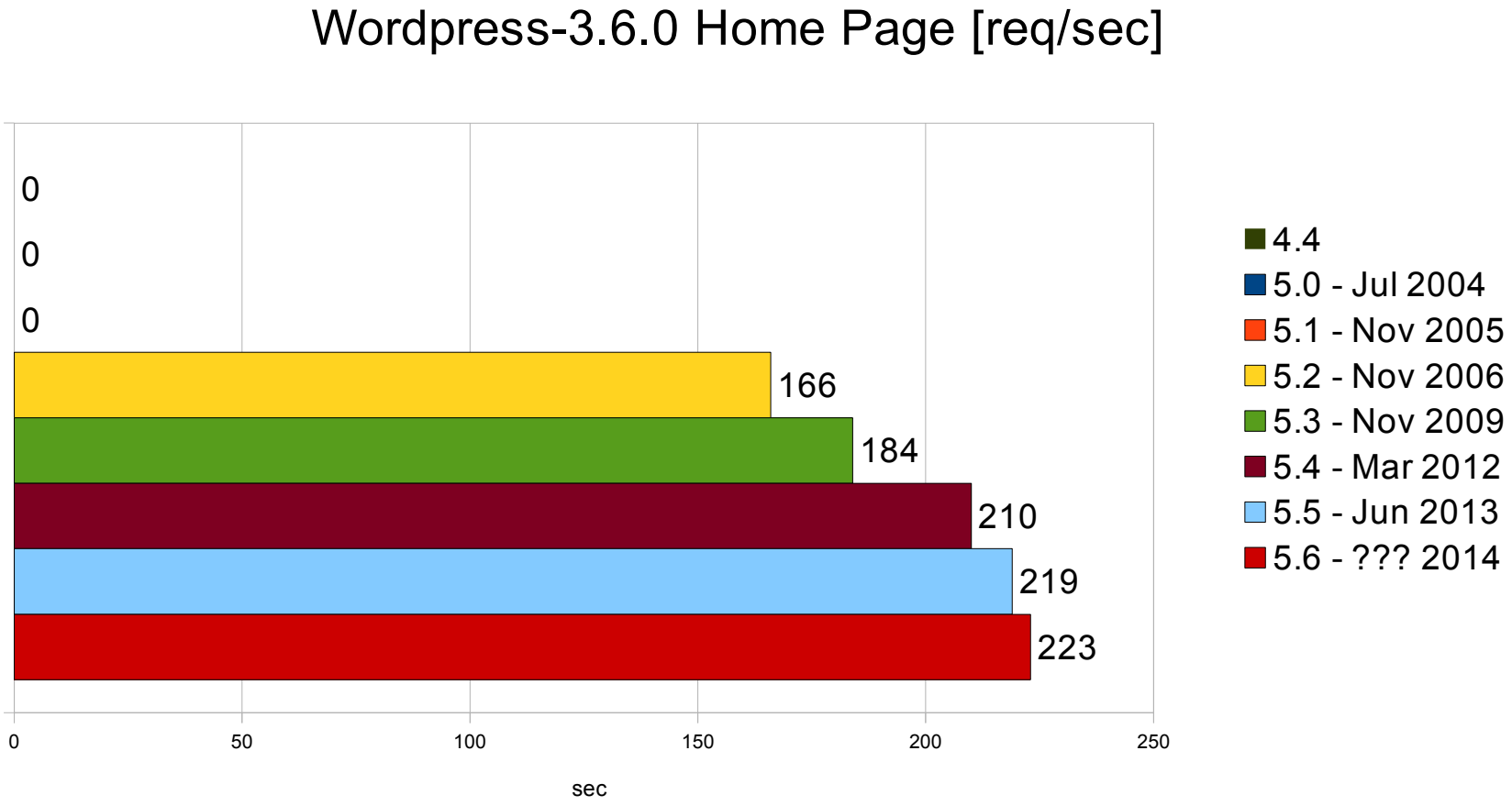
Кто Я?

- Работаю в IT с 1991
- Первое знакомство с PHP в 2002
- Автор Turck MMCache (eAccelerator)
- Работаю в Zend Technologies с 2004
- Автор ext/soap
- Автор pecl/perl
- Один из ведущих разработчиков PHP
- Один из разработчиков phpcloud.com
- Майнтейнер Zend OPcache
- Лидер проекта PHPNG

Производительность PHP



Производительность PHP

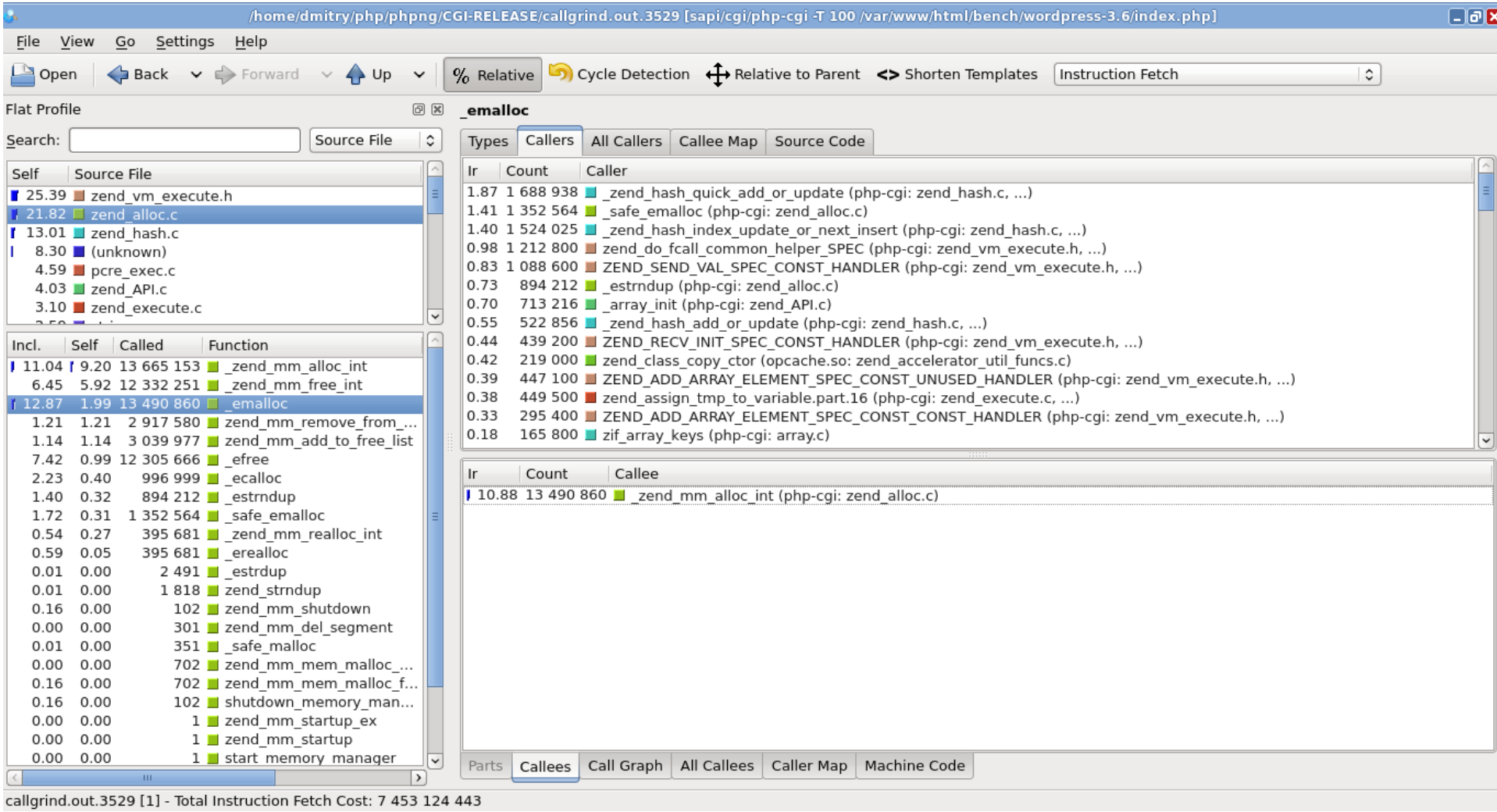


Блуждание в трех соснах

- Почти два года потрачено на прототип JIT для PHP-5.5
- Ускорение для bench.php в 10 раз
- Ускорение для Wordpress 1%

Выводы

- Мы можем генерировать хороший код если можем предсказать типы переменных
- Мы не можем предсказать типы переменных в реальных приложениях
- Даже когда мы догадываемся о типах переменных, мы должны использовать совместимые с PHP структуры данных и это делает генерируемый код не эффективным


/home/dmitry/php/phpng/CGI-RELEASE/callgrind.out.3529 [sapi/cgi/php-cgi -T 100 /var/www/html/bench/wordpress-3.6/index.php]
File View Go Settings Help
Open Back Forward Up % Relative Cycle Detection Relative to Parent <> Shorten Templates Instruction Fetch
Flat Profile
Search: Source File
Self Source File
25.39 zend_vm_execute.h
21.82 zend_alloc.c
13.01 zend_hash.c
8.30 (unknown)
4.59 pcre_exec.c
4.03 zend_API.c
3.10 zend_execute.c
Incl. Self Called Function
11.04 9.20 13 665 153 _zend_mm_alloc_int
6.45 5.92 12 332 251 _zend_mm_free_int
12.87 1.99 13 490 860 _emalloc
1.21 1.21 2 917 580 zend_mm_remove_from_...
1.14 1.14 3 039 977 zend_mm_add_to_free_list
7.42 0.99 12 305 666 _efree
2.23 0.40 996 999 _ecalloc
1.40 0.32 894 212 _estrndup
1.72 0.31 1 352 564 _safe_emalloc
0.54 0.27 395 681 _zend_mm_realloc_int
0.59 0.05 395 681 _erealloc
0.01 0.00 2 491 _estrdup
0.01 0.00 1 818 zend_strndup
0.16 0.00 102 zend_mm_shutdown
0.00 0.00 301 zend_mm_del_segment
0.01 0.00 351 _safe_malloc
0.00 0.00 702 zend_mm_mem_malloc_...
0.16 0.00 702 zend_mm_mem_malloc_f...
0.16 0.00 102 shutdown_memory_man...
0.00 0.00 1 zend_mm_startup_ex
0.00 0.00 1 zend_mm_startup
0.00 0.00 1 start_memory_manager
_emalloc
Types Callers All Callers Callee Map Source Code
Ir Count Caller
1.87 1 688 938 _zend_hash_quick_add_or_update (php-cgi: zend_hash.c, ...)
1.41 1 352 564 _safe_emalloc (php-cgi: zend_alloc.c)
1.40 1 524 025 _zend_hash_index_update_or_next_insert (php-cgi: zend_hash.c, ...)
0.98 1 212 800 zend_do_fcall_common_helper_SPEC (php-cgi: zend_vm_execute.h, ...)
0.83 1 088 600 ZEND_SEND_VAL_SPEC_CONST_HANDLER (php-cgi: zend_vm_execute.h, ...)
0.73 894 212 _estrndup (php-cgi: zend_alloc.c)
0.70 713 216 _array_init (php-cgi: zend_API.c)
0.55 522 856 _zend_hash_add_or_update (php-cgi: zend_hash.c, ...)
0.44 439 200 ZEND_RECV_INIT_SPEC_CONST_HANDLER (php-cgi: zend_vm_execute.h, ...)
0.42 219 000 zend_class_copy_ctor (opcache.so: zend_accelerator_util_funcs.c)
0.39 447 100 ZEND_ADD_ARRAY_ELEMENT_SPEC_CONST_UNUSED_HANDLER (php-cgi: zend_vm_execute.h, ...)
0.38 449 500 zend_assign_tmp_to_variable.part.16 (php-cgi: zend_execute.c, ...)
0.33 295 400 ZEND_ADD_ARRAY_ELEMENT_SPEC_CONST_CONST_HANDLER (php-cgi: zend_vm_execute.h, ...)
0.18 165 800 zif_array_keys (php-cgi: array.c)
Ir Count Callee
10.88 13 490 860 _zend_mm_alloc_int (php-cgi: zend_alloc.c)
Parts Callees Call Graph All Callees Caller Map Machine Code
callgrind.out.3529 [1] - Total Instruction Fetch Cost: 7 453 124 443

PHPNG (New Generation)

- Refactoring
- Основная задача — поднять производительность и заложить базу для дальнейших улучшений
- Отпочкован от основной ветки PHP в начале 2014
- Все изменения из PHP добавляются в PHPNG
- Ни каких новых фич для пользователя (только внутренности)
- Сохранить поведение оригинального PHP на 100%

- Через 2 недели после начала работы мы смогли его собрать
- Еще через 2 недели мы смогли запустить bench.php
- Еще через полтора месяца мы смогли запустить Wordpress
- Еще через месяц (в начале Мая) мы открыли код проекта

zval

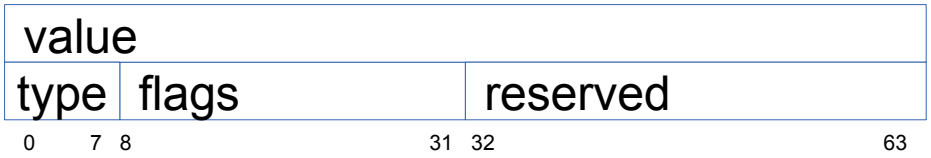
```
Typedef struct _zval_struct {
    union {
        long lval;
        double dval;
        struct {
            char *val;
            int len;
        } str;
        HashTable *ht;
        struct {
            zend_object_handle handle;
            zend_object_handlers *handlers;
        } obj;
    } value;
    zend_uint refcount;
    zend_uchar type;
    zend_uchar is_ref;
} zval;
```

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

```
Typedef 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;
        void *ptr;
    } value;
    union {
        struct {
            zend_uchar type;
            zend_uchar flags;
        };
        zend_uint type_info;
    };
    zend_uint reserved;
} zval;
```

```
sizeof(zval) == 16
```

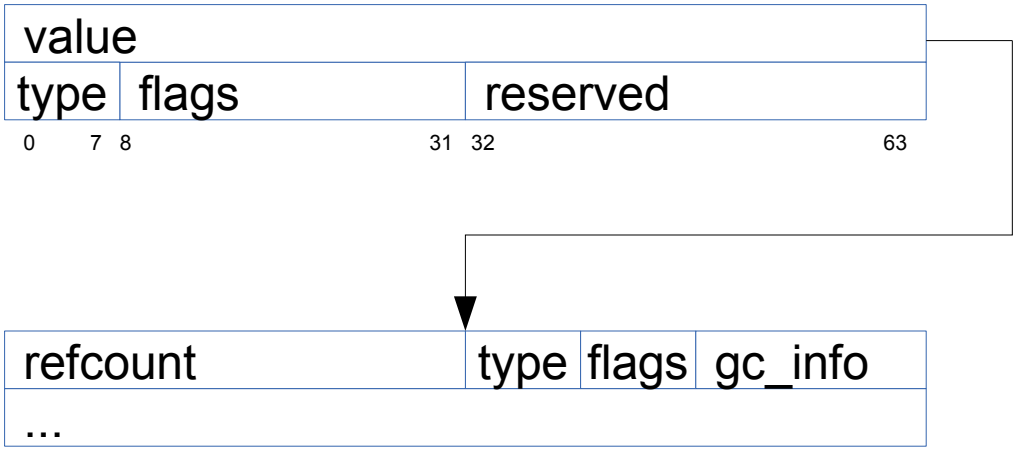
zval



- IS_TYPE_CONSTANT
- IS_TYPE_REFCOUNTED
- IS_TYPE_COLLECTABLE
- IS_TYPE_COPYABLE
- IS_TYPE_IMMUTABLE

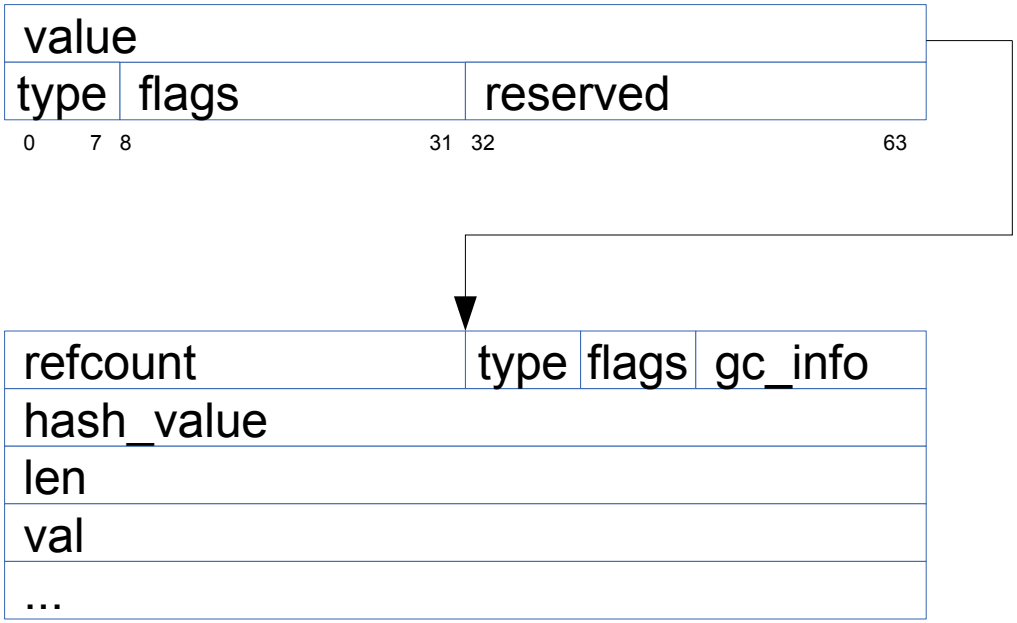
- IS_UNDEF
- IS_NULL
- IS_FALSE
- IS_TRUE
- IS_LONG
- IS_DOUBLE
- IS_STRING
- IS_ARRAY
- IS_OBJECT
- IS_RESOURCE
- IS_REFERENCE
- IS_INDIRECT
- IS_PTR

zval (refcounted)



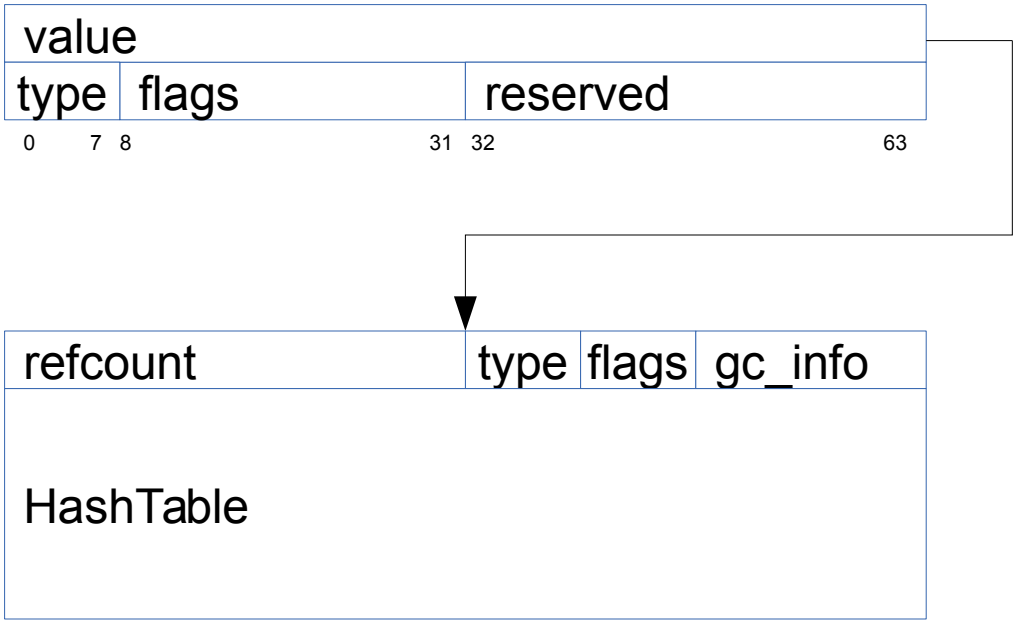
- IS_STRING
- IS_ARRAY
- IS_OBJECT
- IS_RESOURCE
- IS_REFERENCE

zval (string)

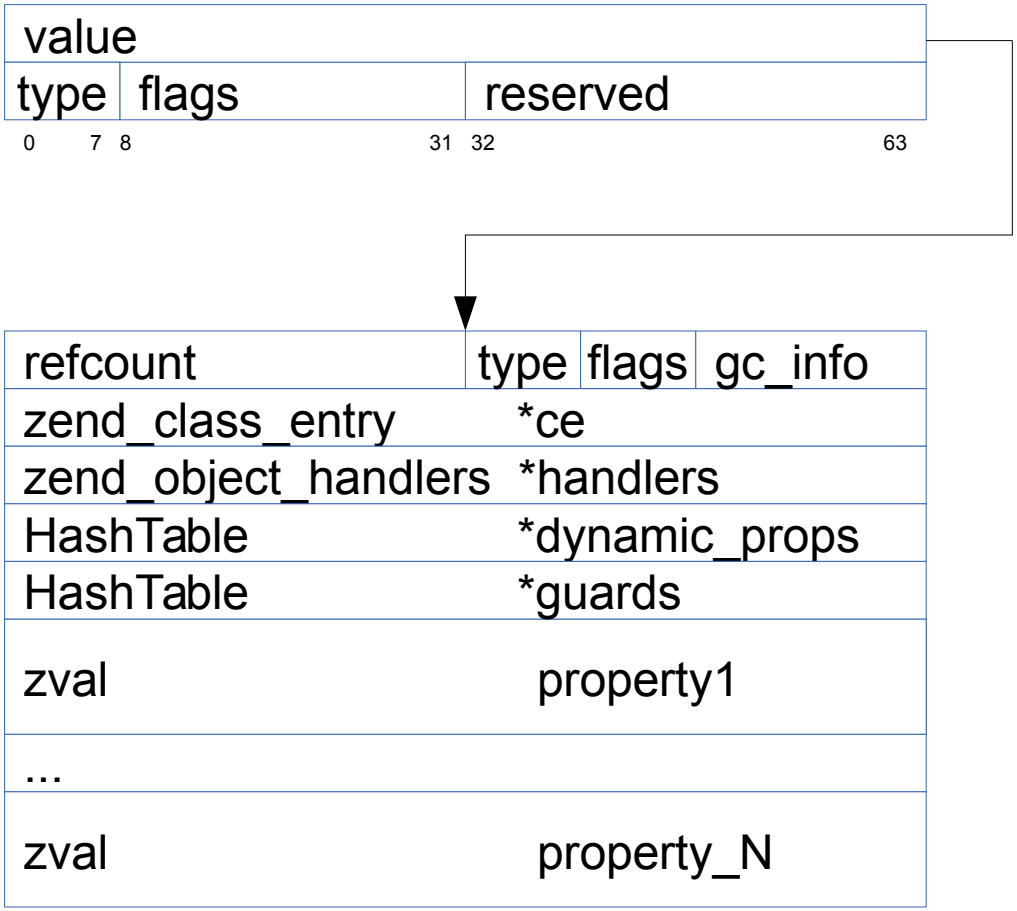


- IS_STR_PERSISTENT
- IS_STR_INTERNED
- IS_STR_PERMANENT
- IS_STR_CONSTANT

zval (array)

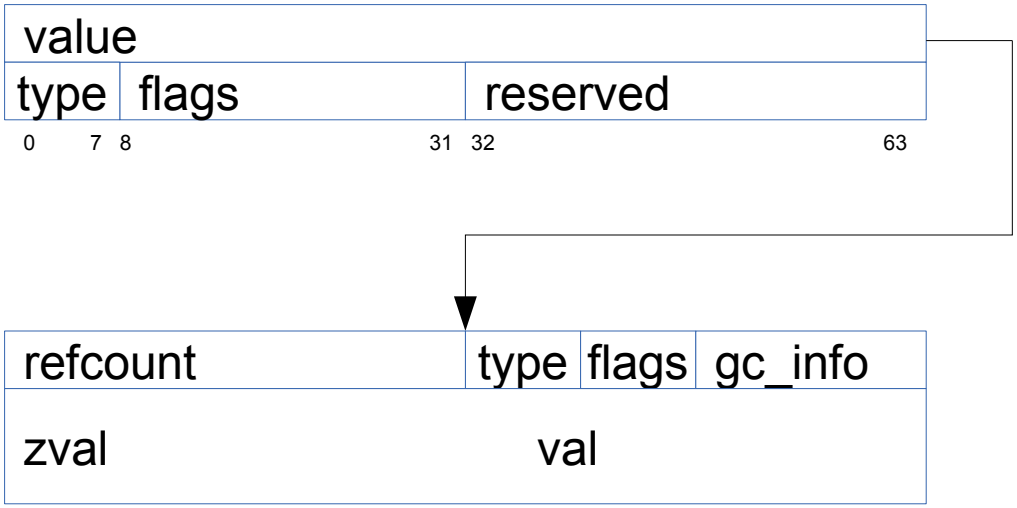

value
type
flags
reserved
0
7
8
31
32
63
refcount
type
flags
gc_info
HashTable

zval (object)



- IS_OBJ_DESTROYED
- IS_OBJ_FREED

zval (reference)

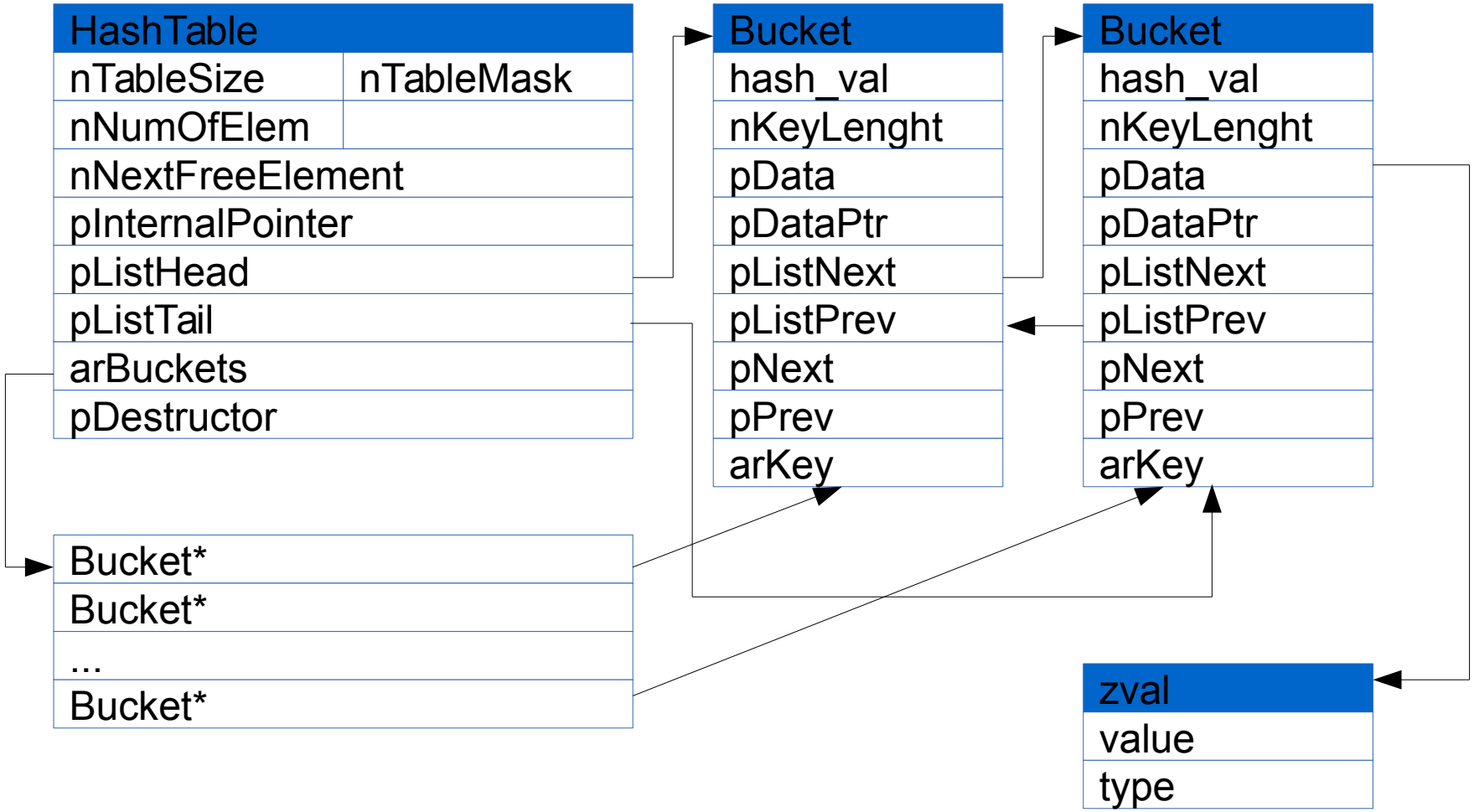


zval (IS_BOOL -> IS_FALSE + IS_TRUE)

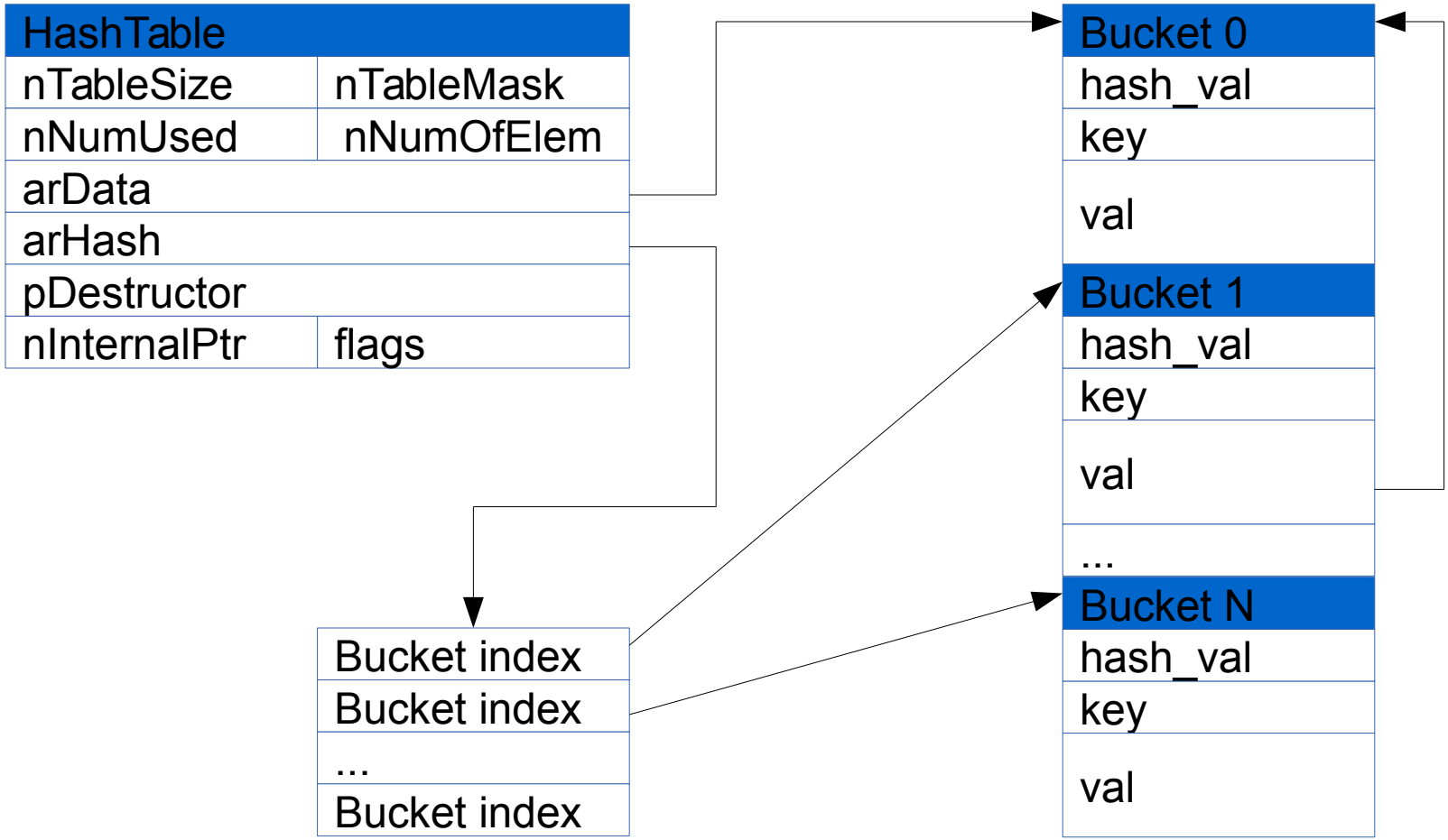
```
ZEND_VM_HANDLER(43, ZEND_JMPZ,
  CONST|TMP|VAR|CV, ANY)
{
  long ret;
  zval *val =
    GET_OP1_ZVAL_PTR(BP_VAR_R);

  if (OP1_TYPE == IS_TMP_VAR &&
      Z_TYPE_P(val) == IS_BOOL)) {
    ret = Z_LVAL_P(val);
  } else {
    ret = i_zend_is_true(val TSRMLS_CC);
  }

  if (!ret) {
    ZEND_VM_SET_OPCODE(
      opline->op2.jmp_addr);
    ZEND_VM_CONTINUE();
  }

  ZEND_VM_NEXT_OPCODE();
}
```

```
ZEND_VM_HANDLER(43, ZEND_JMPZ,
  CONST|TMP|VAR|CV, ANY)
{
  zval *val =
    GET_OP1_ZVAL_PTR_DEREF(BP_VAR_R);

  if (OP1_TYPE == IS_TMP_VAR) {
    if (Z_TYPE_P(val) == IS_TRUE) {
      ZEND_VM_SET_OPCODE(opline + 1);
      ZEND_VM_CONTINUE();
    } else if (Z_TYPE_P(val) <= IS_TRUE) {
      ZEND_VM_SET_OPCODE(
        opline->op2.jmp_addr);
      ZEND_VM_CONTINUE();
    }
  }
  if (i_zend_is_true(val TSRMLS_CC)) {
    opline++;
  } else {
    opline = opline->op2.jmp_addr;
  }
  ZEND_VM_JMP(opline);
}
```

HashTable (php-5.*)



HashTable (phpng)



HashTable

- Размер HashTable уменьшился с 72 до 56 байт
- Размер Bucket уменьшился с 72 до 32 байт
- Память под все Bucket-ы выделяется одновременно
- Bucket.key теперь указатель на zend_string и значит может не копироваться (достаточно увеличить reference counter)
- Память под элементы массива выделяется вместе с Bucket-ами
- Улучшена data locality => меньше промахов в CPU кэше

Immutable Arrays

```
$a = array();
for ($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 Parameter Parsing API

- 5% времени тратится на разбор параметров внутренних функций
- Для некоторых простых функций время на разбор параметров составляет более 90%

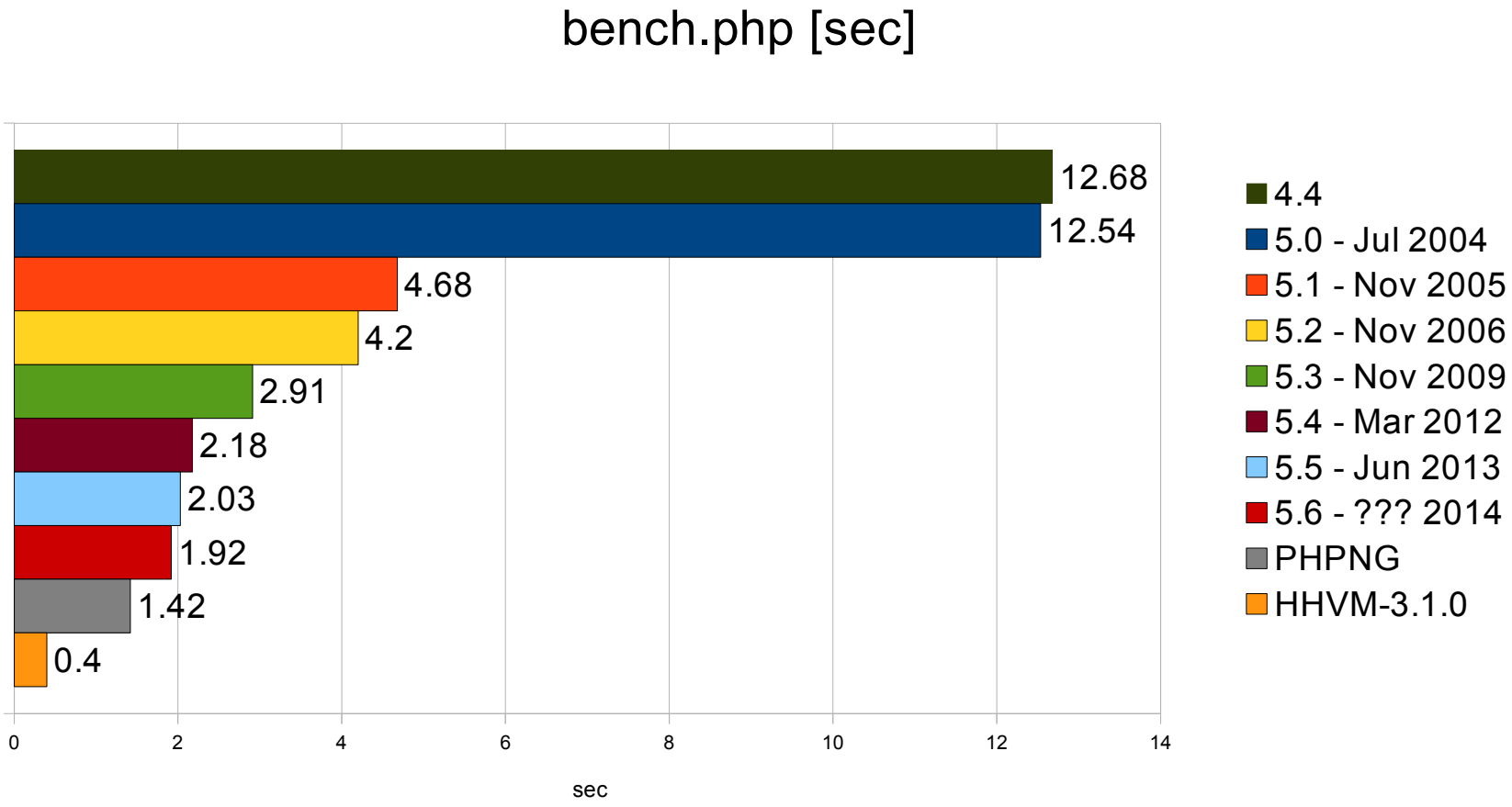
```
if (zend_parse_parameters(ZEND_NUM_ARGS()
    TSRMLS_CC, "za|b",
    &value, &array, &strict) == FAILURE) {
    return;
}
```

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

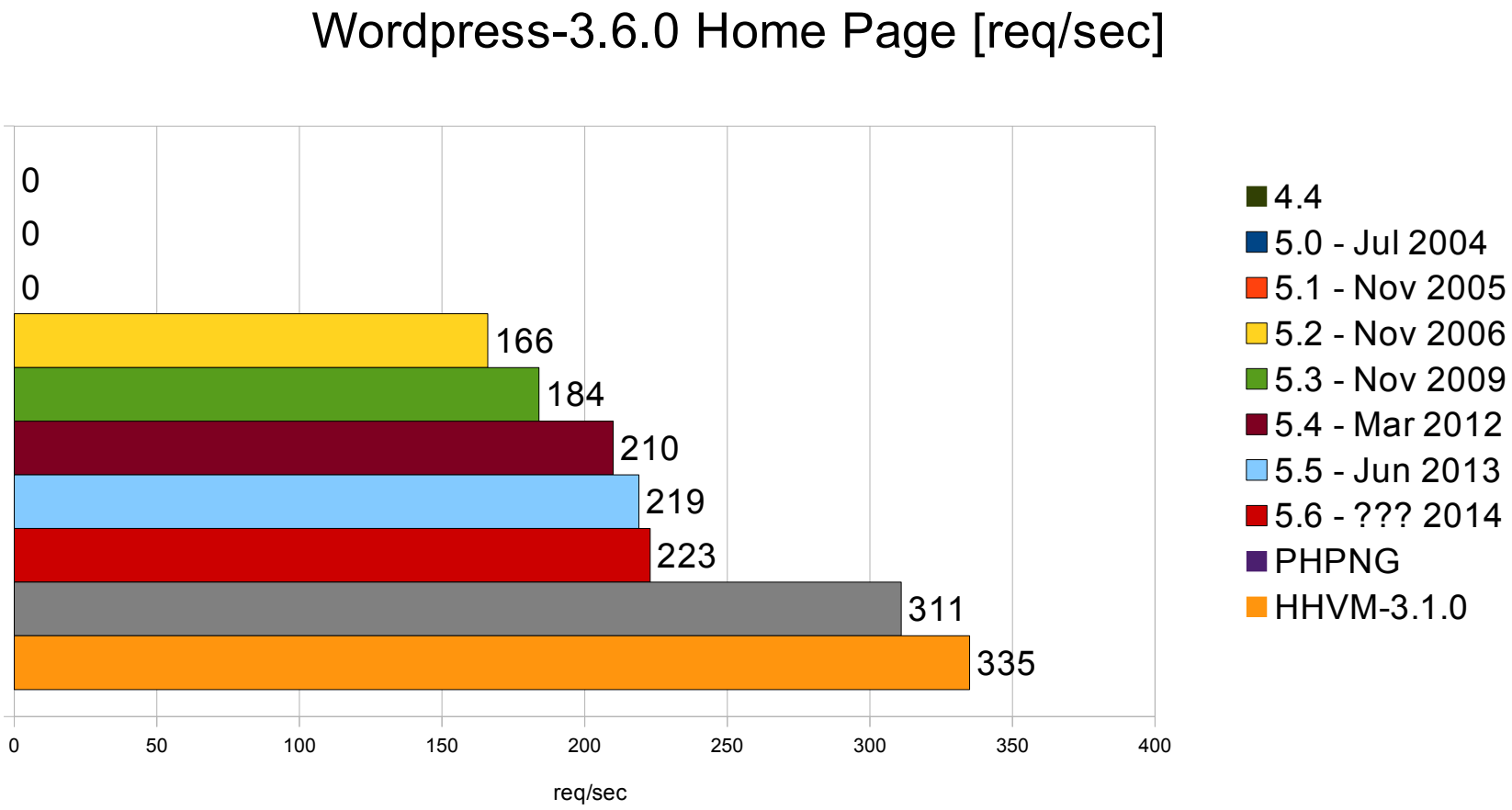
“Мелкие” Улучшения

- Новое API для обхода HashTable
- Оптимизация копирования массивов
- Ref-counting вместо копирования
- PCRE with JIT
- strtr() fix

Производительность PHP/PHPNG



Производительность PHP/PHPNG



Self	Source File
1 892 365 865	zend_vm_execute.h
1 626 067 770	zend_alloc.c
969 735 168	zend_hash.c
618 466 270	(unknown)
342 440 700	pcre_exec.c
300 176 725	zend_API.c
231 229 200	zend_execute.c

Incl.	Self	Called	Function
822 938 147	685 578 250	13 665 153	_zend_mm_alloc_int
480 394 300	441 522 620	12 332 251	_zend_mm_free_int
959 348 054	148 399 460	13 490 860	_emalloc
90 547 610	90 547 610	2 917 580	zend_mm_remove_from_...
85 300 555	85 300 555	3 039 977	zend_mm_add_to_free_list
552 968 058	73 833 996	12 305 666	_efree
166 110 333	29 909 970	996 999	_ecalloc
104 052 341	24 143 724	894 212	_estrndup
127 993 324	22 993 588	1 352 564	_safe_emalloc
40 502 495	20 093 175	395 681	_zend_mm_realloc_int
44 063 624	3 561 129	395 681	_erealloc
405 377	72 239	2 491	_estrdup
444 021	54 432	1 818	zend_strndup
11 565 824	40 809	102	zend_mm_shutdown
51 272	6 020	301	zend_mm_del_segment
419 014	4 563	351	_safe_malloc
332 140	2 106	702	zend_mm_mem_malloc_...
11 555 478	2 106	702	zend_mm_mem_malloc_f...
11 566 844	1 020	102	shutdown_memory_man...
1 108	334	1	zend_mm_startup_ex
2 506	40	1	zend_mm_startup
3 007	13	1	start_memory_manager

callgrind.out.3529 [1] - Total Instruction Fetch Cost: 7 453 124 443

Self	Source File
1 494 294 896	zend_vm_execute.h
506 330 851	zend_alloc.c
503 976 554	zend_hash.c
465 430 317	(unknown)
95 177 627	zend_operators.c
88 476 400	zend_object_handlers.c
79 532 058	block_pass.c

Incl.	Self	Called	Function
274 871 110	202 920 510	3 168 228	_zend_mm_alloc_int
135 839 620	100 933 520	2 696 923	_zend_mm_free_int
53 738 218	53 738 218	1 706 988	zend_mm_remove_from_...
52 829 845	52 829 845	1 737 061	zend_mm_add_to_free_list
300 056 600	34 010 108	3 091 828	_emalloc
36 108 808	16 564 809	252 040	_zend_mm_realloc_int
149 130 024	15 955 206	2 659 201	_efree
115 167 112	14 795 525	870 325	_safe_emalloc
48 583 452	8 436 660	281 222	_ecalloc
10 978 143	2 927 502	108 426	_estrndup
38 377 168	2 268 360	252 040	_erealloc
13 661 597	822 888	37 404	_safe_erealloc
392 224	65 888	2 272	_estrdup
201 729	41 833	102	zend_mm_shutdown
223 383	4 650	310	_safe_realloc
88 578	4 537	349	_safe_malloc
34 834	4 342	217	zend_mm_del_segment
253 803	2 160	720	zend_mm_mem_malloc_...
163 476	2 160	720	zend_mm_mem_malloc_f...
202 749	1 020	102	shutdown_memory_man...
651	651	21	zend_mm_add_to_rest_list
1 108	334	1	zend_mm_startup_ex

callgrind.out.22864 [1] - Total Instruction Fetch Cost: 4 143 517 540

Чего мы достигли?

- 25% ускорение на синтетических тестах
- 10-40% ускорение на реальных приложениях (40% на домашней странице wordpress)
- Существенное уменьшение используемой памяти
- 99% совместимость с поведением PHP
- Поддержка наиболее используемых SAPI
- Поддержка почти всех расширений включенных в PHP (адаптировано 62, осталось 13)

Что Дальше?

- Реализовать оставшиеся идеи
- Решить имеющиеся проблеммы
- Адоптировать оставшиеся расширения
- Слить в main-stream ветку PHP
- Не дать испортить то что уже сделано
- Релиз PHP-Next (mid 2015)
- JIT

Вопросы?

Страница проекта: http://wiki.php.net/phpng