

The Secret of PHP7's Performance

@Laruen



SELF INTRODUCTION

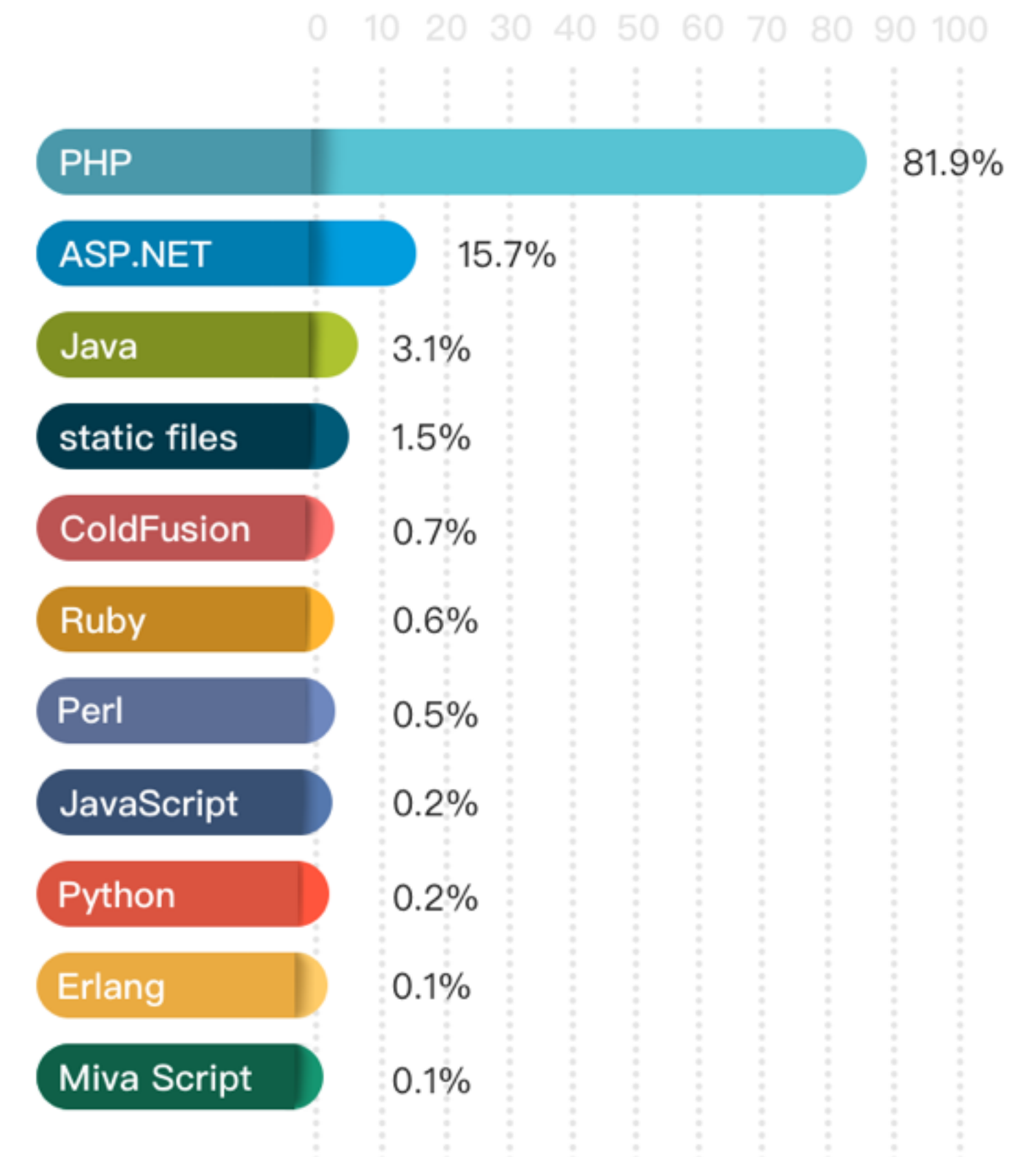
- ▶ Author of Yaf, Yar, Yac, Yacnf, Taint Projects
- ▶ Maintainer of Opcache, Msgpack, PHP-Lua Projects
- ▶ PHP core developer since 2011
- ▶ Zend consultant since 2013
- ▶ PHP7 core developer
- ▶ Chief software architect at lianjia since 2015

Organizations



PHP BRIEF INTRO

- ▶ Created in 1994 by Rasmus Lerdorf
- ▶ 20+ years programming language
- ▶ Most popular web service program language
- ▶ PHP7 is released at 3 Dec 2015
- ▶ Latest version is PHP7.0.8



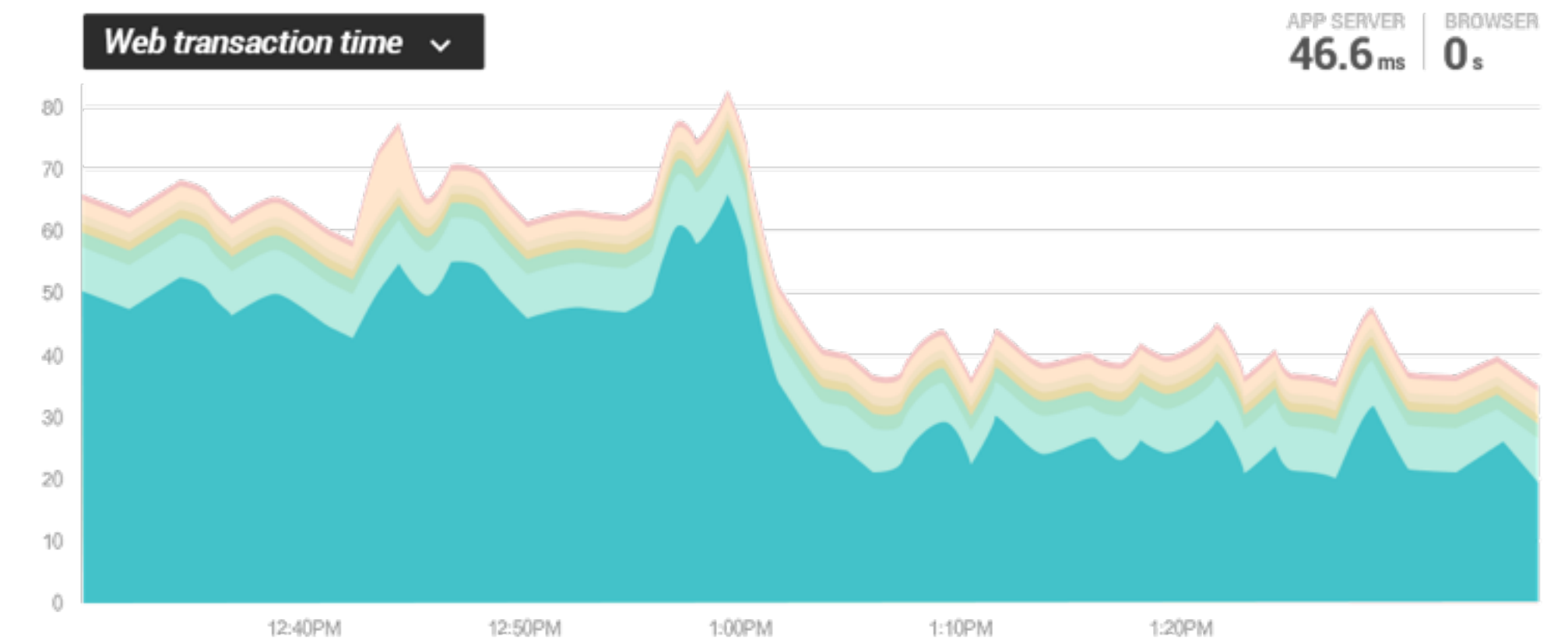
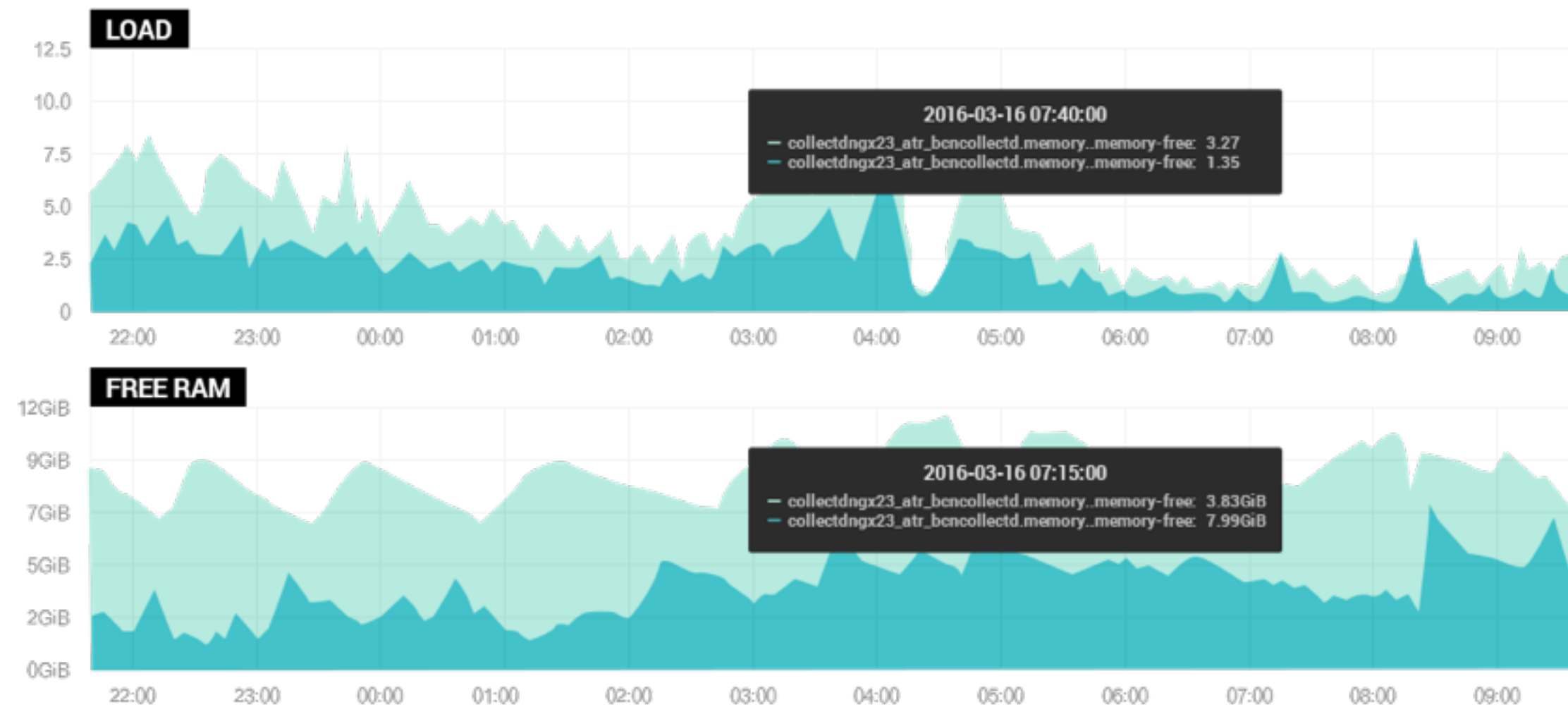
W3Techs.com, 19 March 2016

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

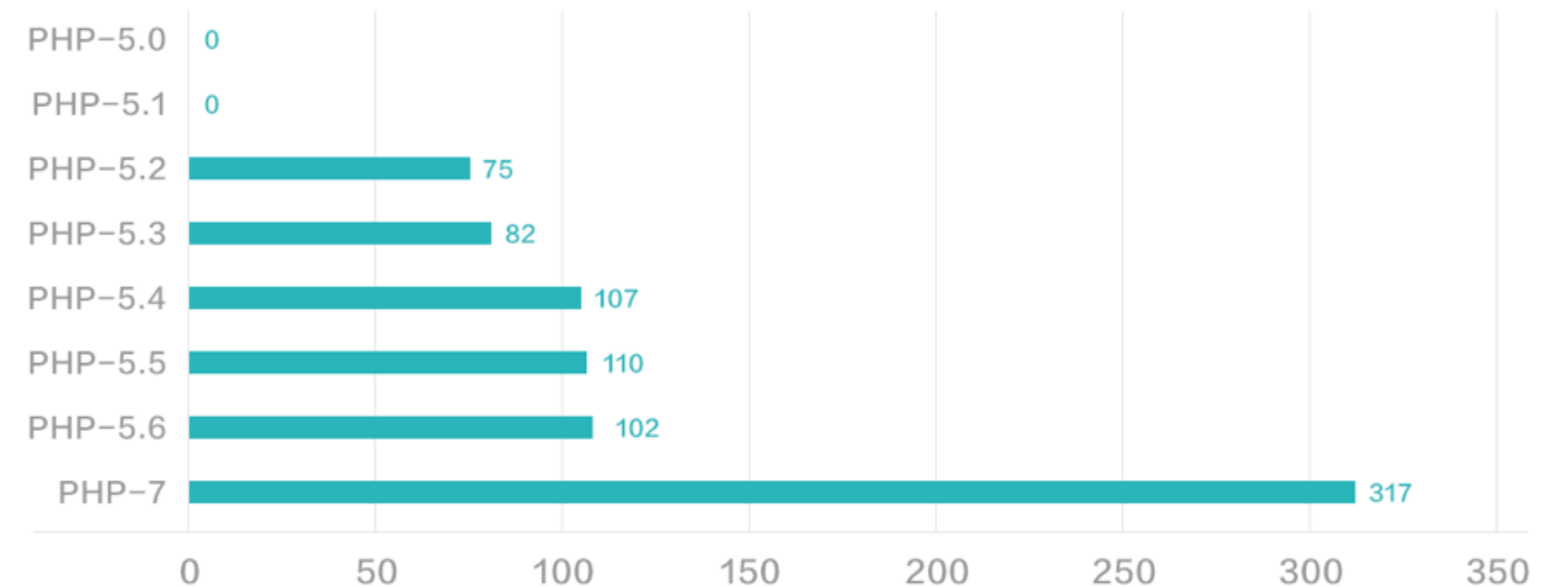
- ▶ **Improved Performance: PHP 7 is up to twice as fast as PHP 5.6**
- ▶ Significantly reduced memory usage
- ▶ Abstract syntax tree
- ▶ Consistent 64-bit support
- ▶ Improved exception hierarchy
- ▶ Many fatal errors converted to exceptions
- ▶ The null coalescing operator (??)
- ▶ Return & Scalar type declarations
- ▶ Anonymous classes
- ▶

PHP7

- ▶ 100 % performance improved in various apps
- ▶ Which optimization is most responsible?
- ▶



wordpress 3.6 home page qps



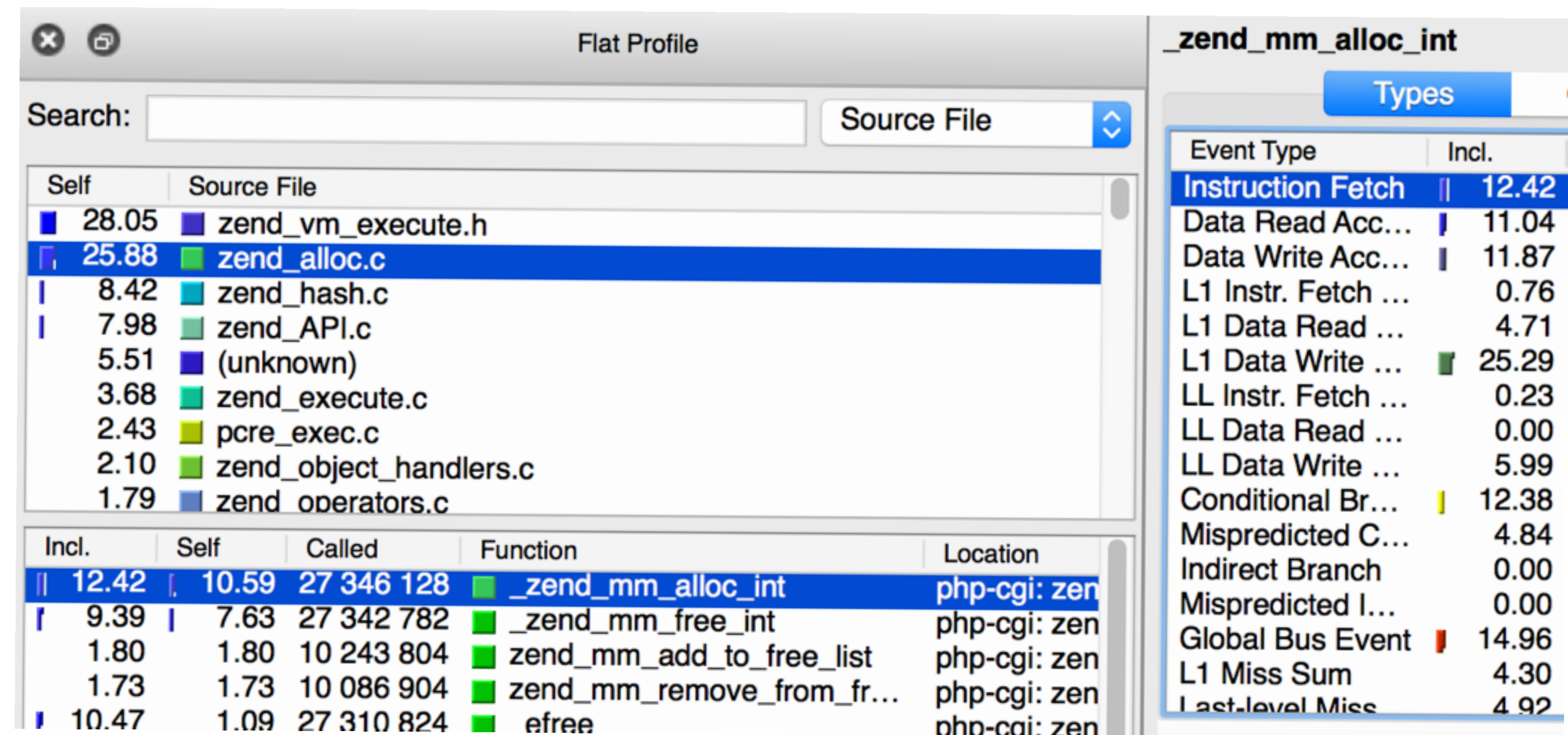
JUST-IN-TIME COMPILER

- ▶ Once upon a time
- ▶ There comes HHVM
- ▶ Performance really matters
- ▶ A secret project in Zend
- ▶ Based on opcache of PHP5.5
- ▶ Invisible performance change in wordpress
 - ▶ Why?

<https://github.com/zendtech/php-src/tree/zend-jit>

WORDPRESS PROFILING (PHP5.5)

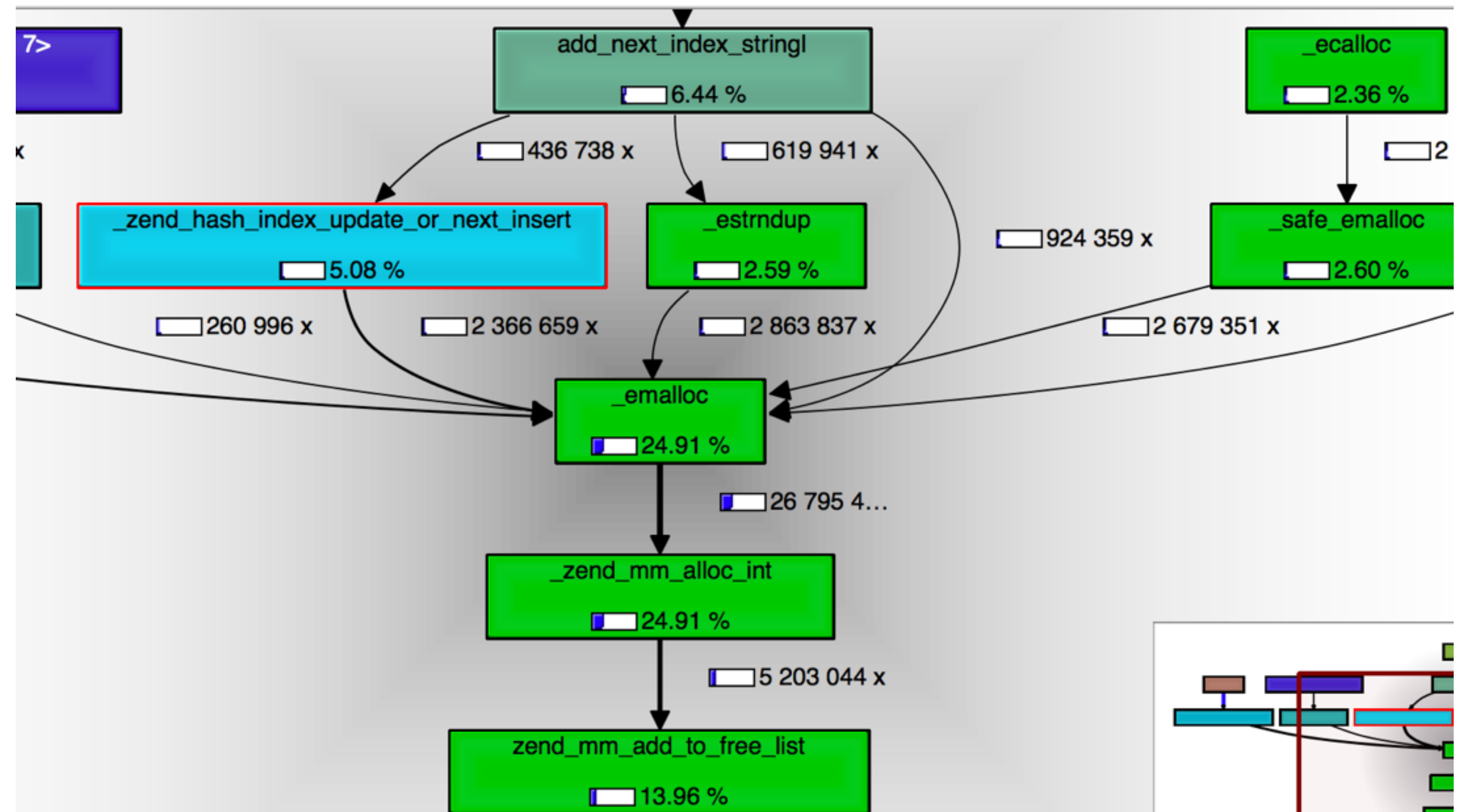
- ▶ Wordpress:
 - ▶ Typical PHP real-life application
- ▶ Callgrind:
 - ▶ 28% CPU time is spent on Zend VM
 - ▶ 25% CPU time is spent on Memory
 - ▶ Top one is `_zend_mm_alloc_int`



Callgrind result on wordpress home page

WORDPRESS PROFILING (PHP5.5)

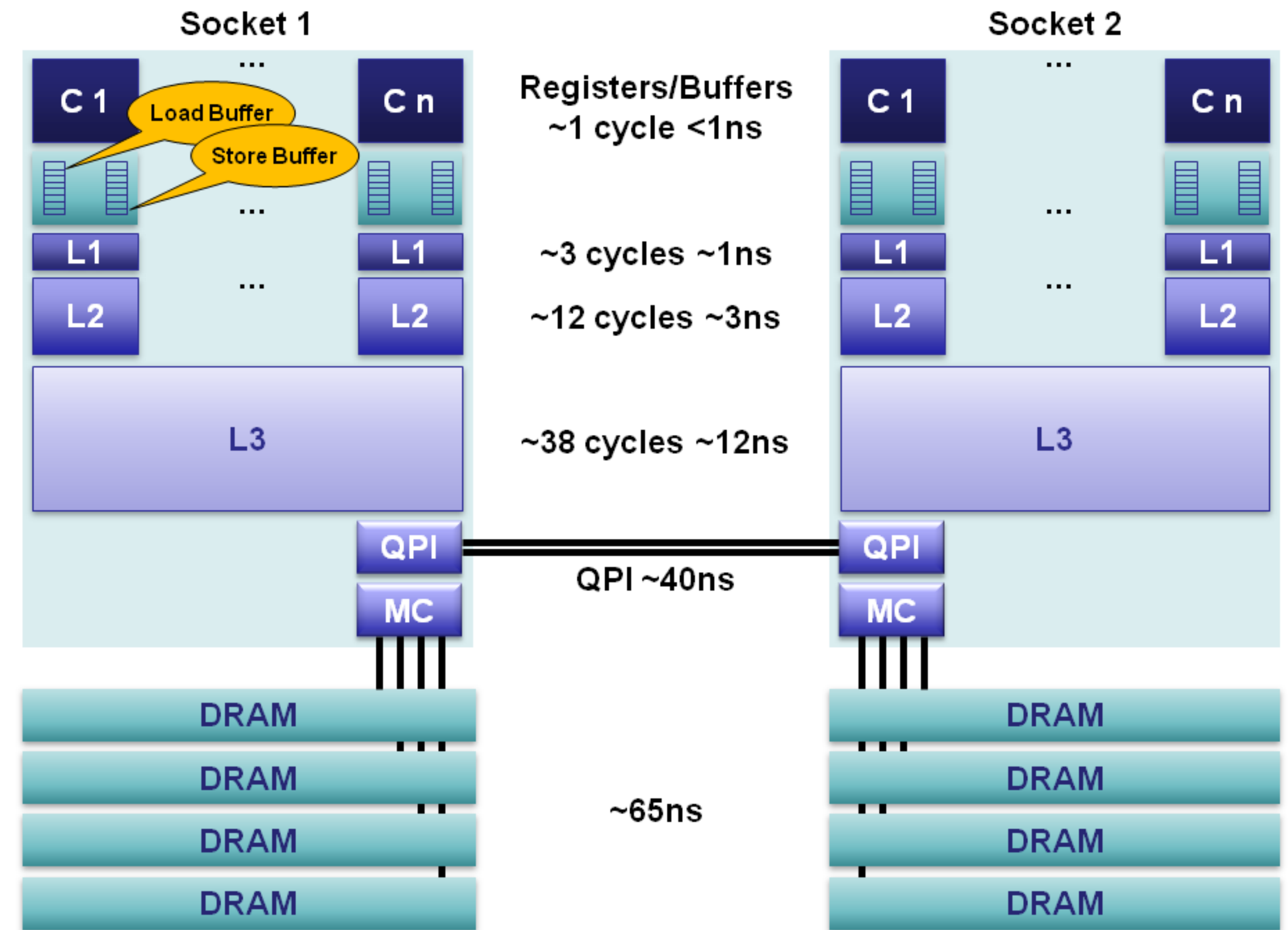
- ▶ We have too many allocations
- ▶ Thoughts:
 - ▶ `_strndup`
 - ▶ HashTable
 - ▶ `MAKE_STD_ZVAL`



__mm_alloc_init callers graph (part)

`MEMORY` IS THE KEY

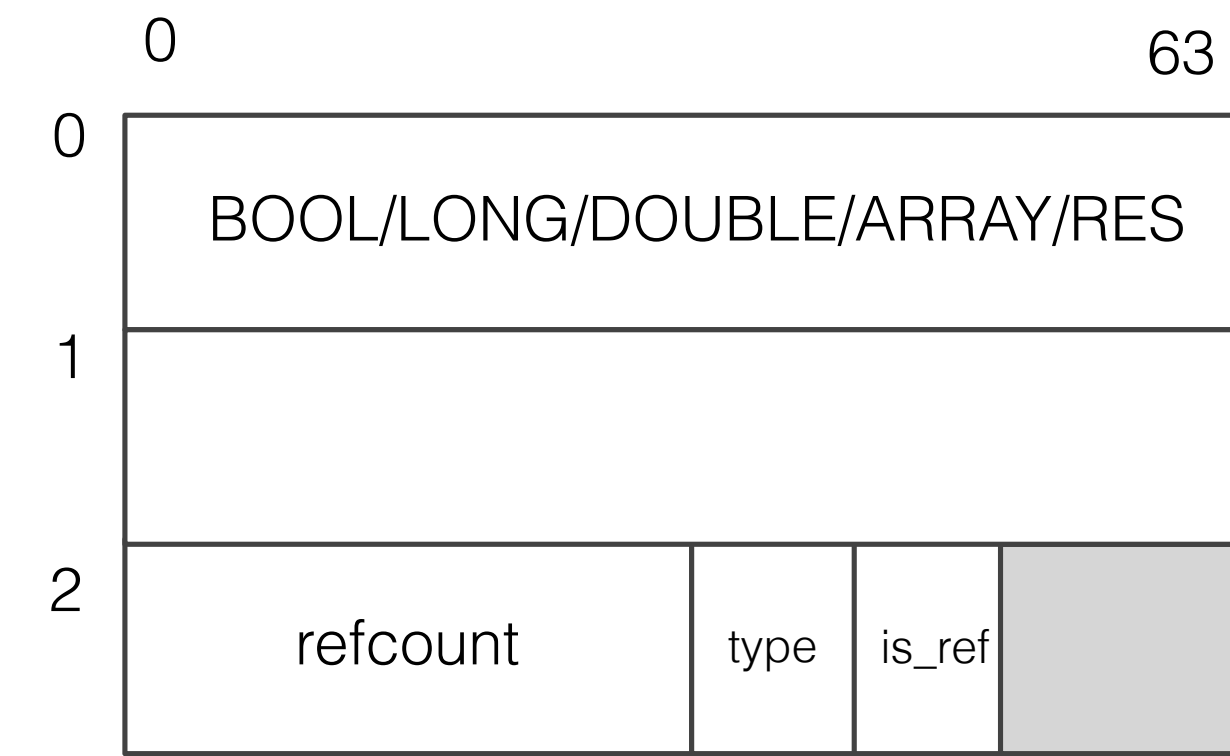
- ▶ `Memory` is the bottle-neck(25%)
 - ▶ High memory usage
 - ▶ High cache misses
 - ▶ High TLB misses
 - ▶ High page faults
 - ▶ Too many allocation
 - ▶ More CPU time
 - ▶ Increase iTLB miss
 - ▶ Increase branch-miss
 - ▶ High level memory indirection
 - ▶ Increase cache misses



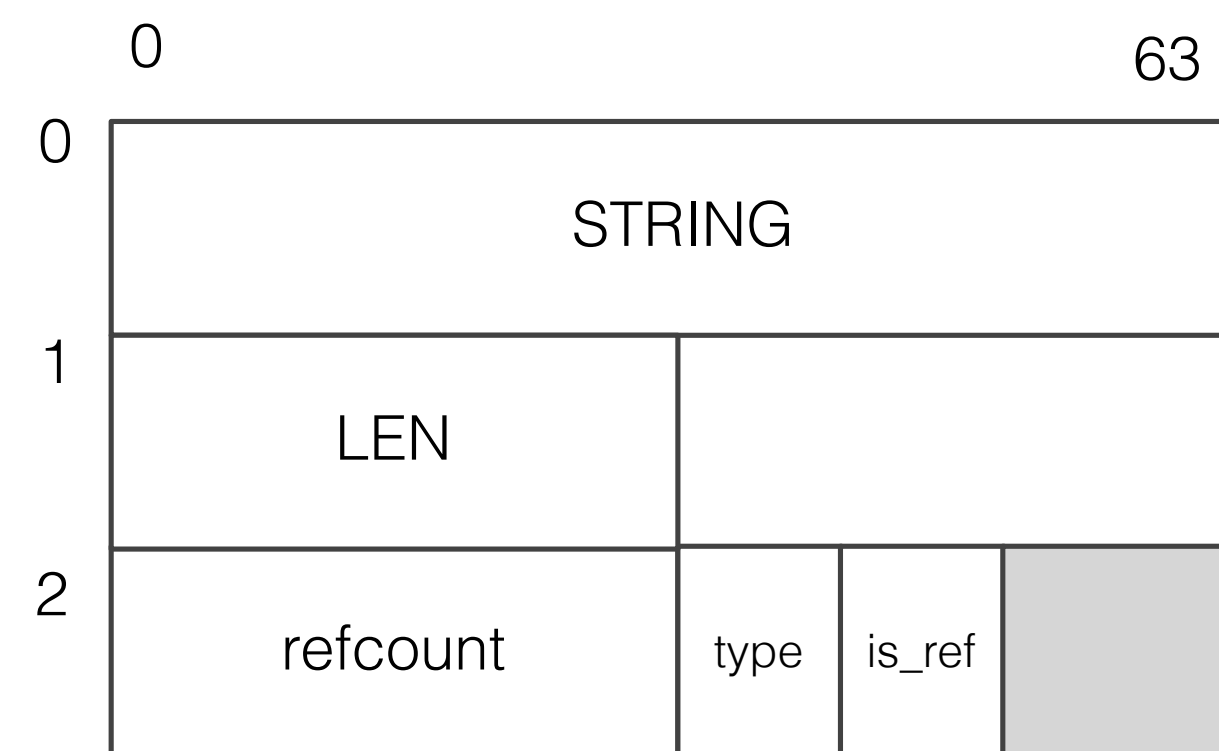
Cache hierarchy latency

INSPECT ZVAL

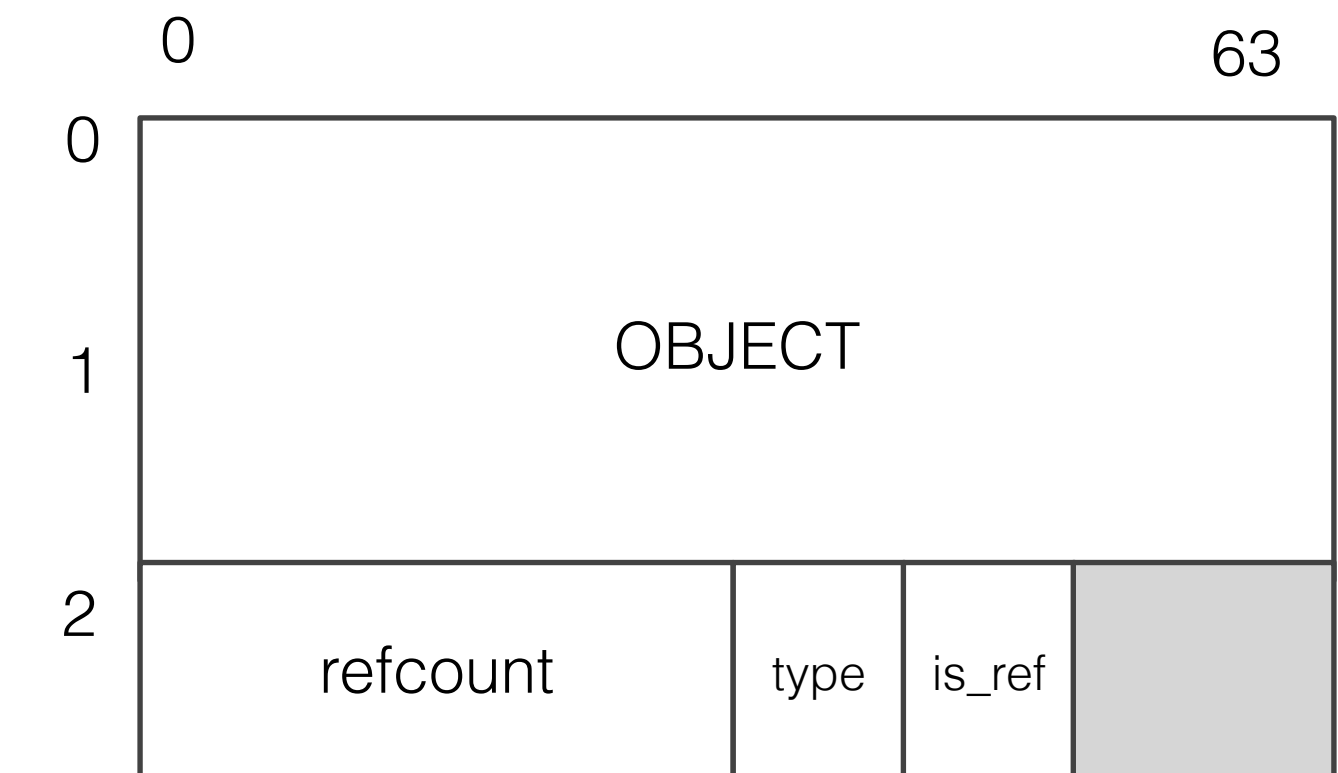
- ▶ Total 24 bytes
- ▶ Value uses 16 bytes
- ▶ Thoughts:
 - ▶ Most types use 8 bytes
 - ▶ String uses 12 bytes
 - ▶ Only Object uses 16 bytes
 - ▶ Only a little types are ref



Scalar types



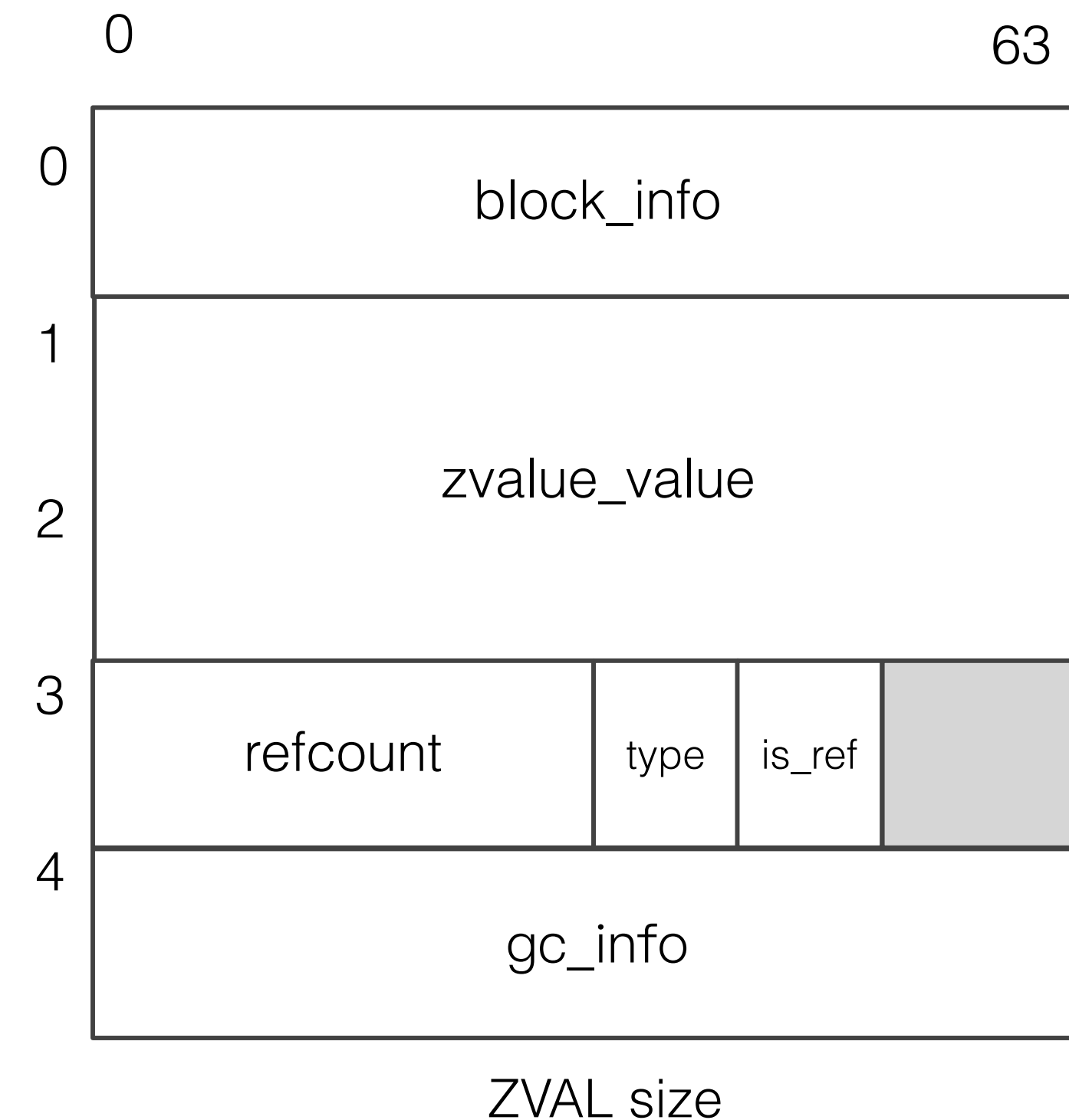
String



Object

INSPECT ZVAL

- ▶ Not only 24 bytes
 - ▶ GC info(for GC): Added 16 bytes
 - ▶ Block info(for MM): Added 8 bytes
- ▶ Total 48 bytes
- ▶ Thoughts:
 - ▶ Only array and object need gc info
 - ▶ Block info?
 - ▶ Stack allocating?
 - ▶ New MM?



PROFILING WP

- ▶ String is the most used type
- ▶ Object is only used in 2%
- ▶ 40% types only used 8 bytes in zval.value
- ▶ Only 15% types are GC cared
- ▶ ~10% is reference type
- ▶ Thoughts:
 - ▶ String needs to be optimized
 - ▶ We don't need unified `zval`
 - ▶ Reducing zval's size should be possible

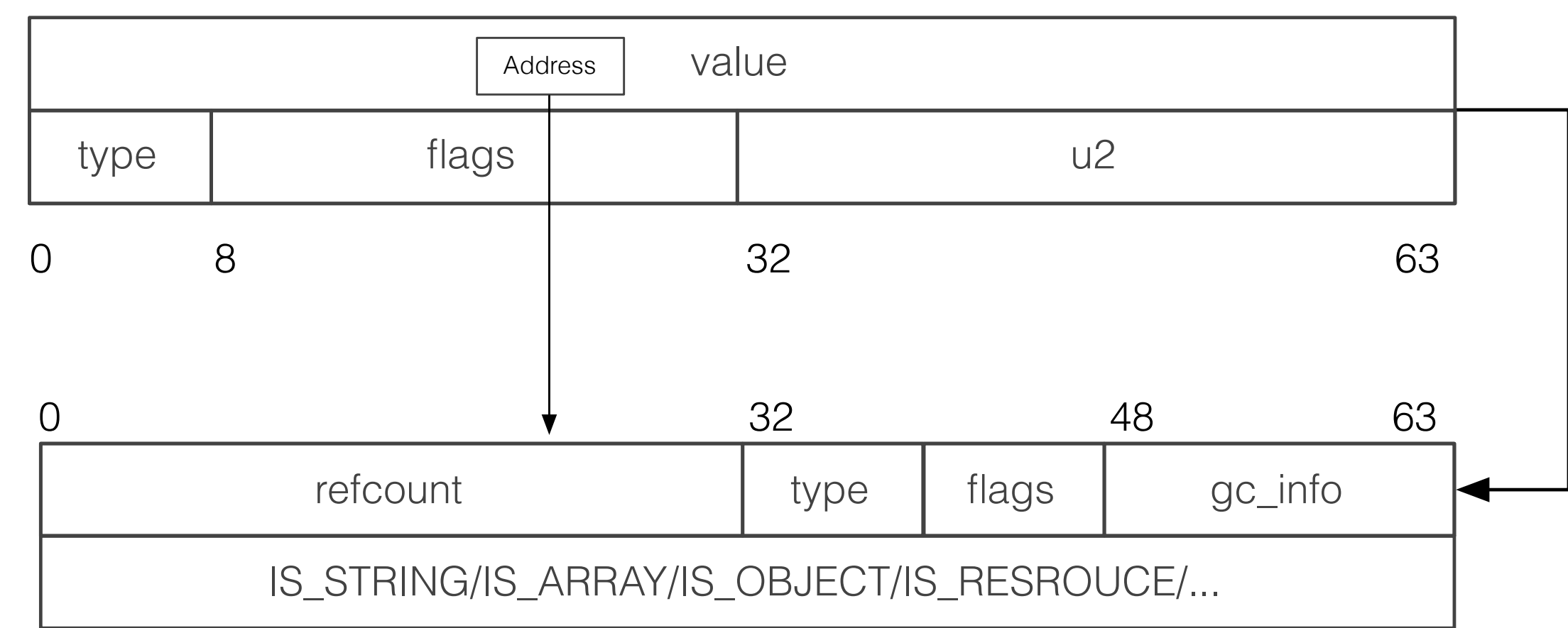
NULL	2798	4%
Bool	11894	17%
Double	6	..
Long	4134	6%
Resource	25	..
Array	8709	13%
Object	1582	2%
String	37564	56%

Types in one WP lifecycle

BRAND NEW ZVAL

- ▶ Total 16 bytes
- ▶ Copy instead of refcount for basic types
- ▶ Refcount is not against zval anymore
- ▶ External struct is used for complex types
 - ▶ values can not be stored in size_t mem
 - ▶ refcount
 - ▶ gc_info
 - ▶ value flags

IS_LONG

 Can be kept in 64bits?

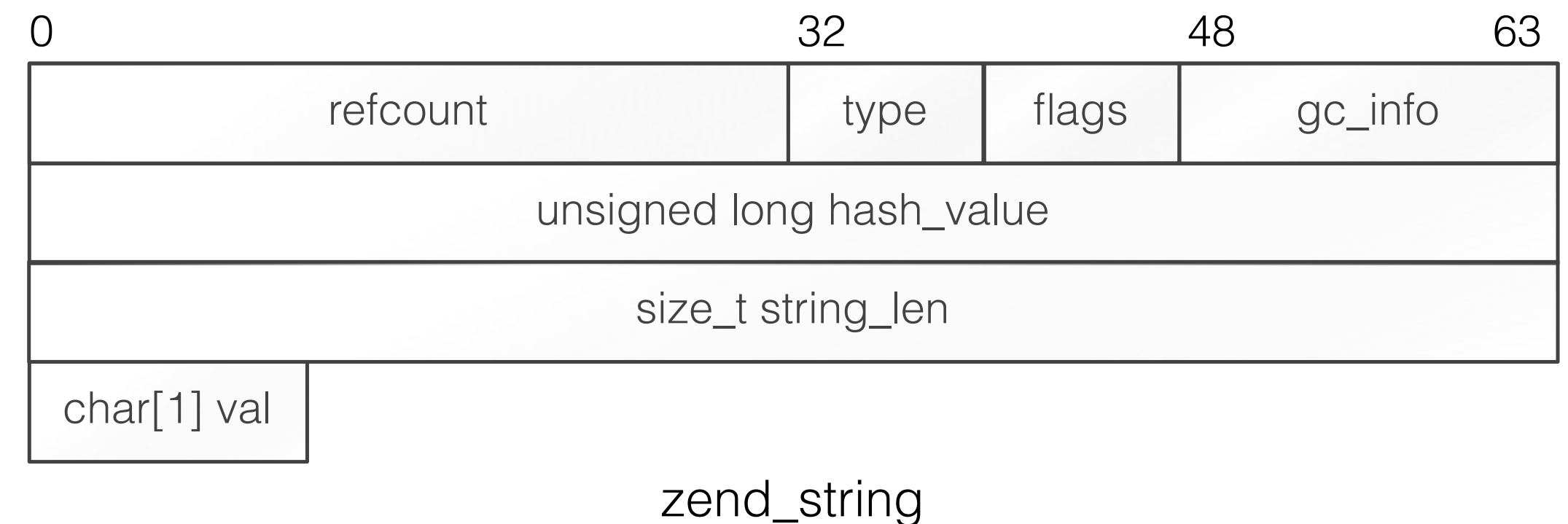
ZVAL in PHP7

IS_STRING

 Can be kept in 64bits?

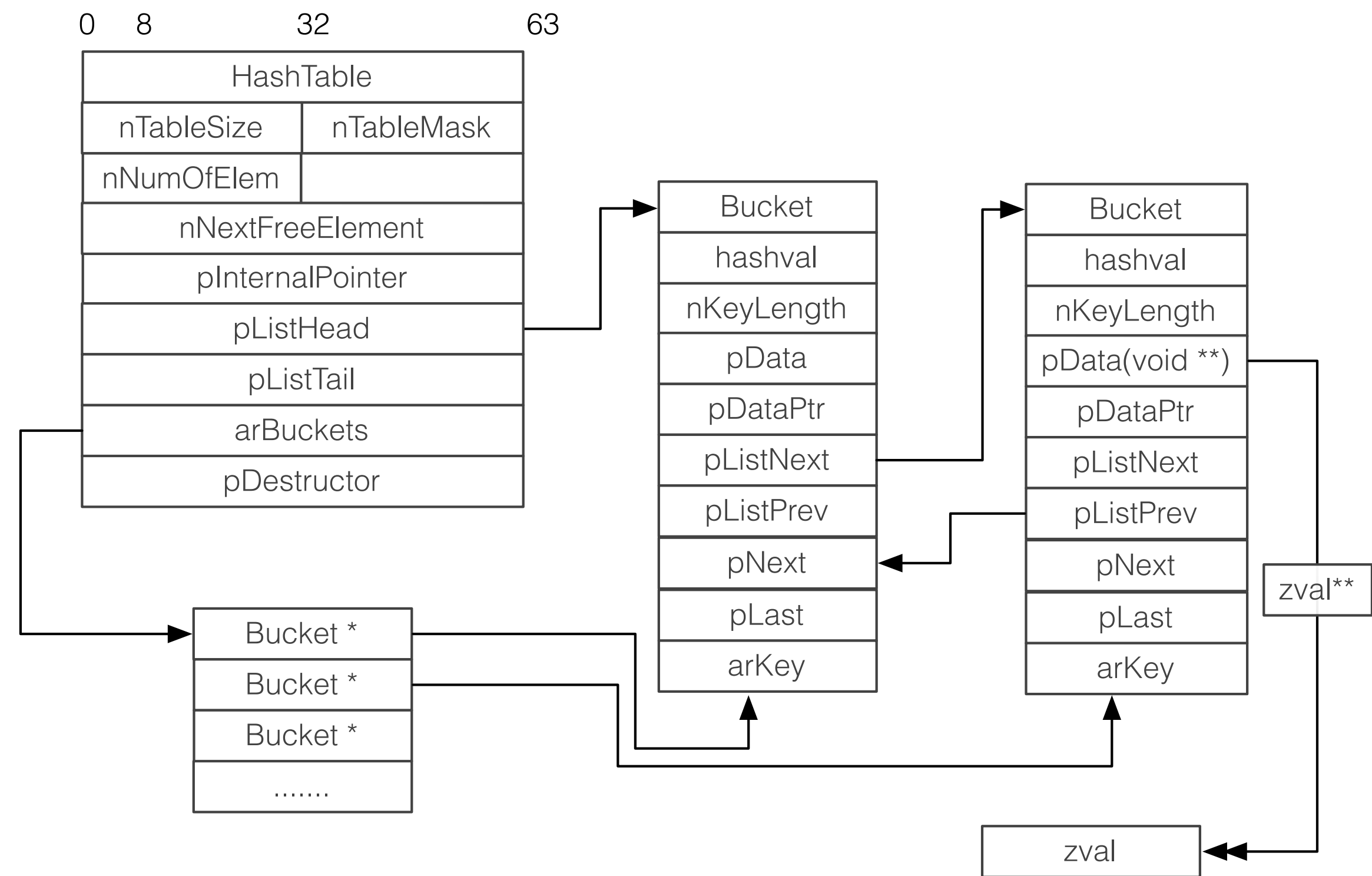
ZEND STRING

- ▶ Most used type in real world
- ▶ PHP5
 - ▶ C string
 - ▶ int length
 - ▶ Hash value needs to be calculated every time
 - ▶ Interned string is distinguished by address
- ▶ PHP7
 - ▶ Brand new type: zend_string
 - ▶ Size length
 - ▶ Hash value is kept after being calculated
 - ▶ Interned string is distinguished by flags
 - ▶ COW instead of copying



INSPECT HASHTABLE

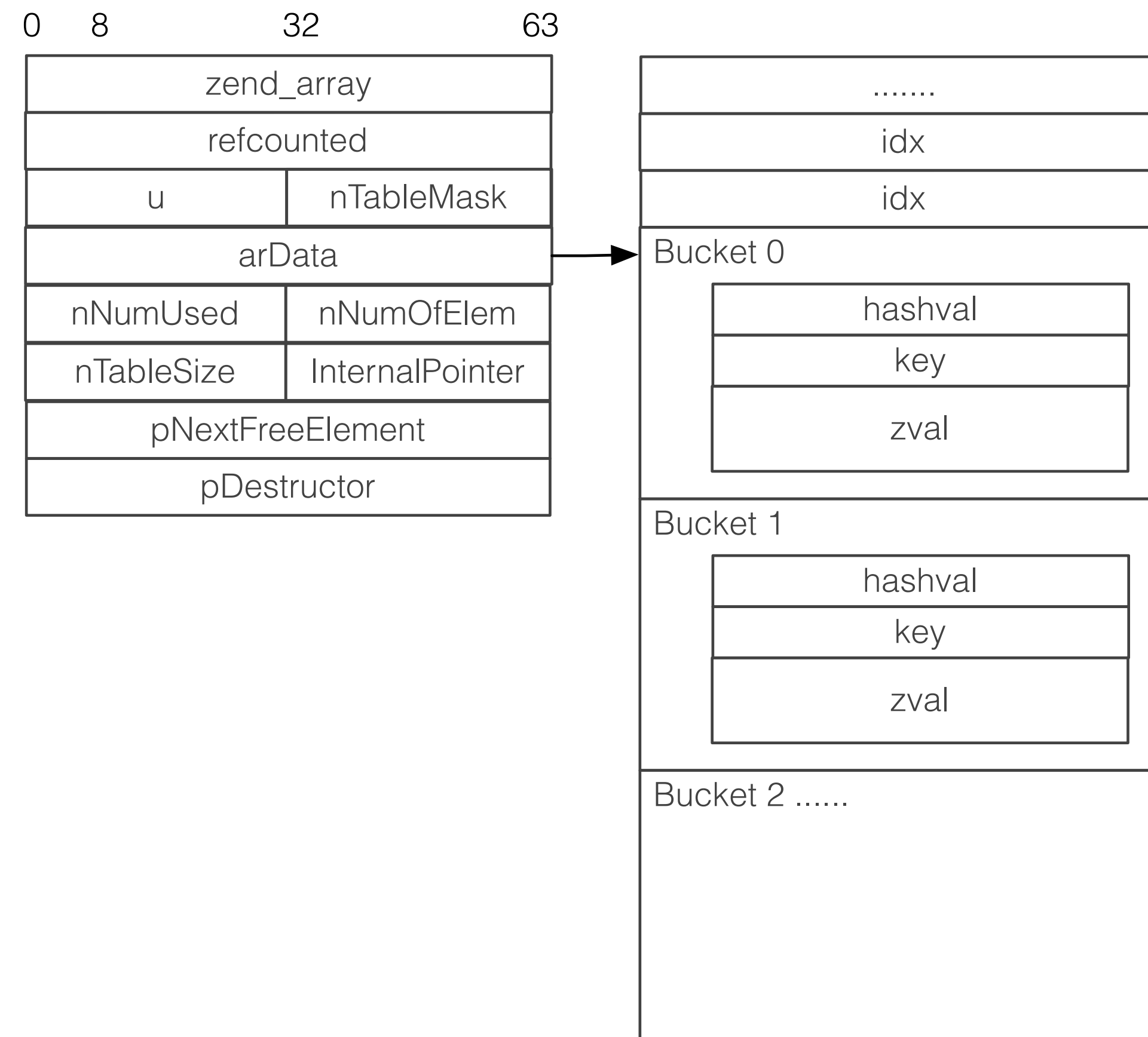
- ▶ Total 72 bytes
- ▶ `typeof bucket->pData` is `void **`
- ▶ Thoughts:
 - ▶ In most cases, zval are stored
 - ▶ Reduce memory usage
 - ▶ Reduce memory indirection
 - ▶ `pListNext`
 - ▶ `HashTable -> Bucket`
 - ▶ `Bucket -> ZVAL ** (void **)`



HashTable struct

ZEND ARRAY

- ▶ Total 56 bytes
- ▶ Key is zend_string
- ▶ Less memory indirection
 - ▶ Bucket.val
 - ▶ Bucket.val.zval
 - ▶ Buckets are allocated together



zend_array struct

ILLUSTRATION PHP5

\$a = 5
\$b = \$a
\$m = "PHP5"
\$n = \$m
\$l = &\$n

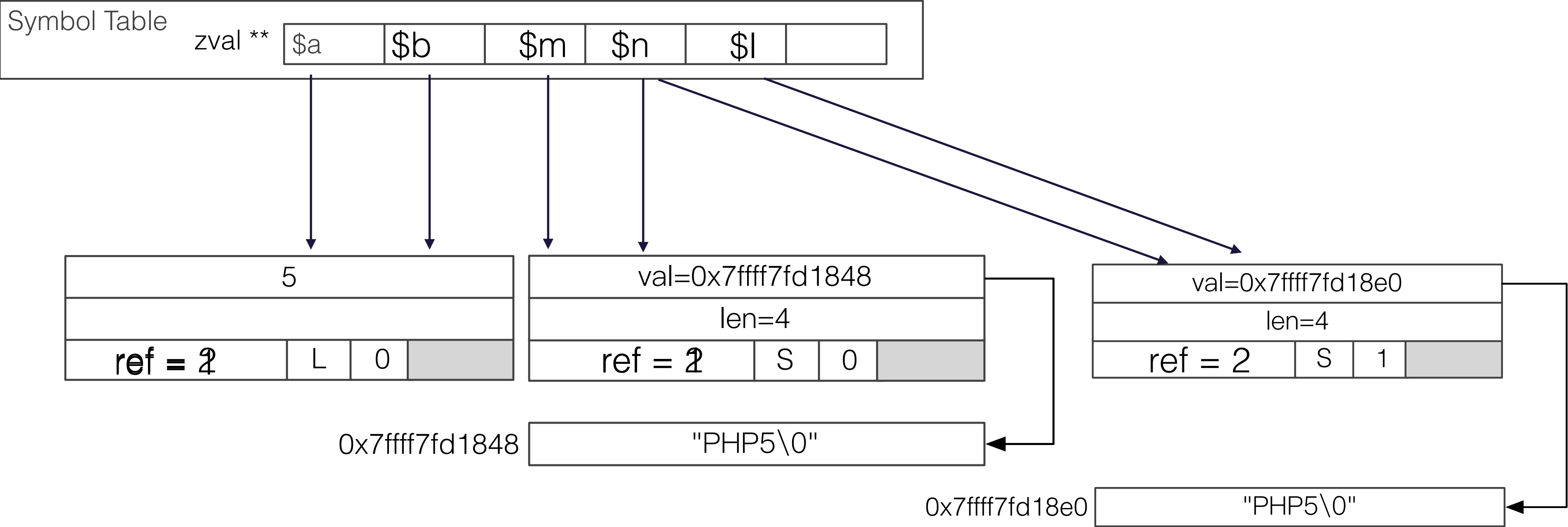


ILLUSTRATION PHP7

```
$a = 7
$b = $a
$m = "PHP7"
$n = $m
$l = &$n
```

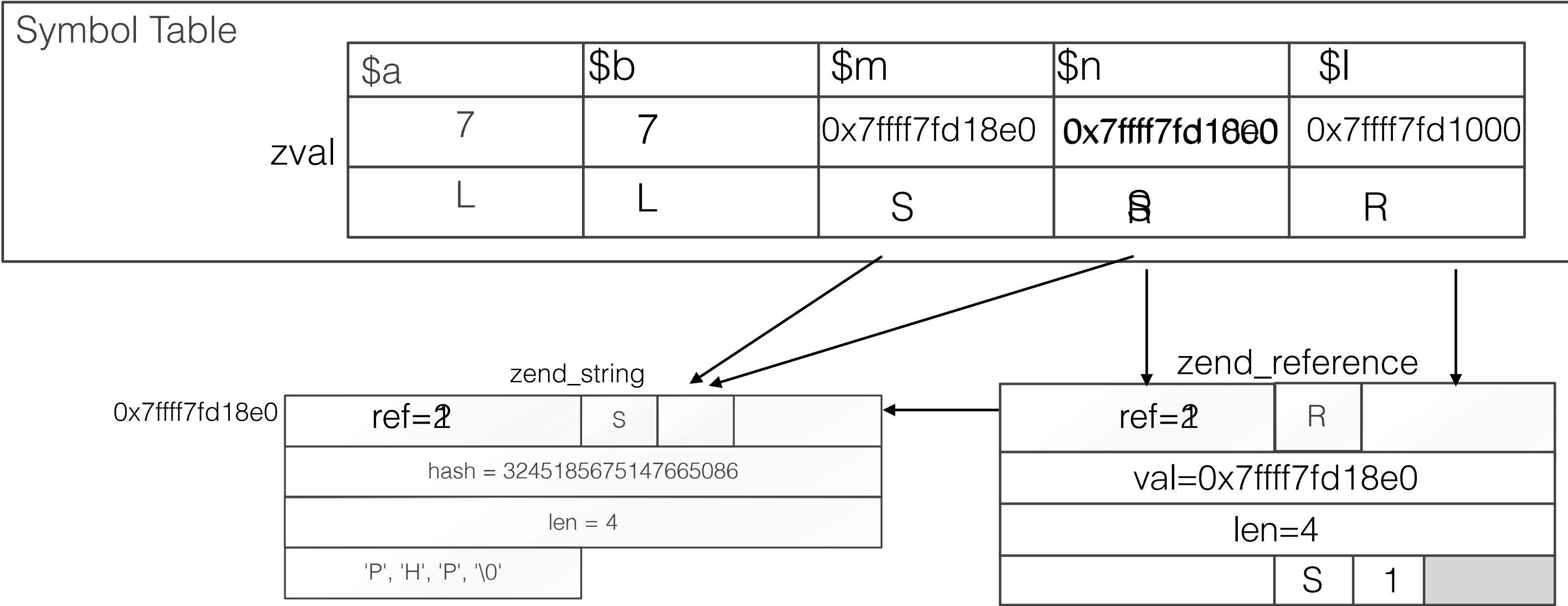


ILLUSTRATION PHP5

```
$arr = range(0, 5)
foreach($arr as $val) {
}
```

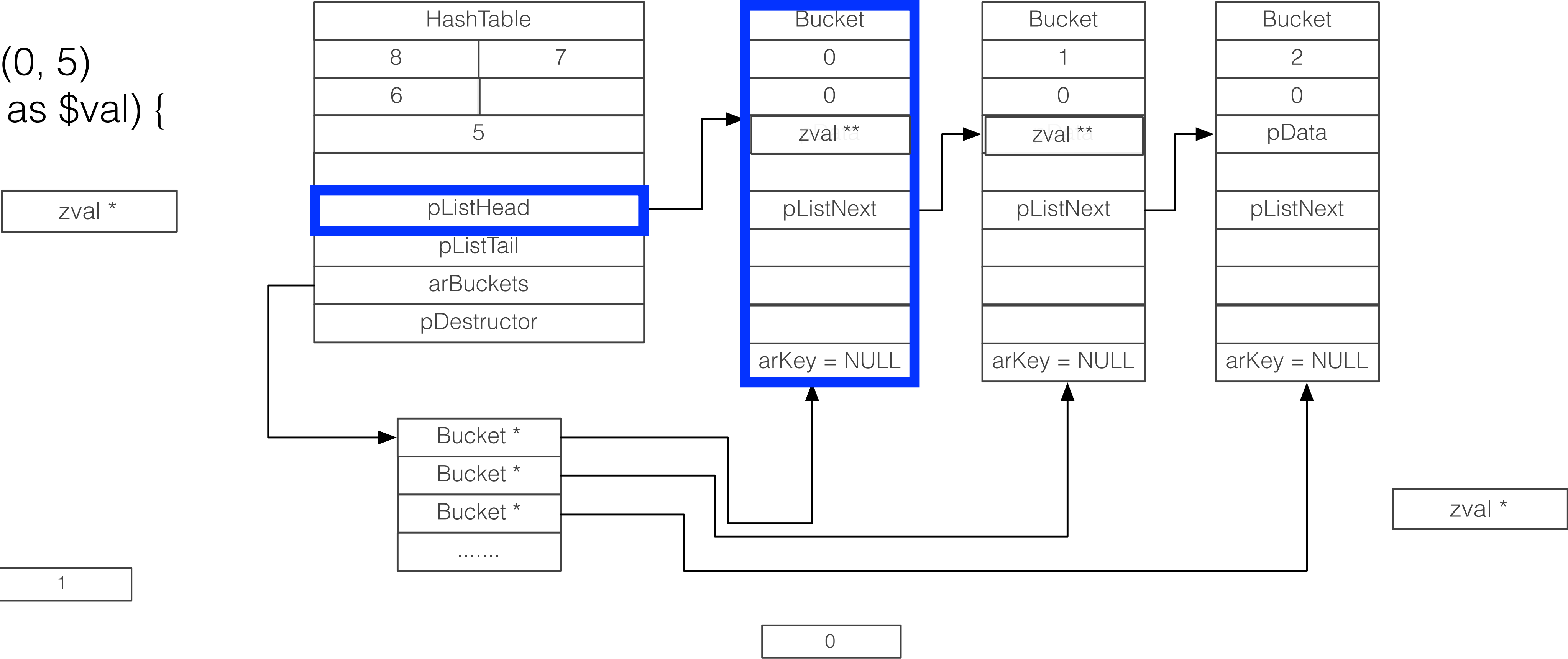
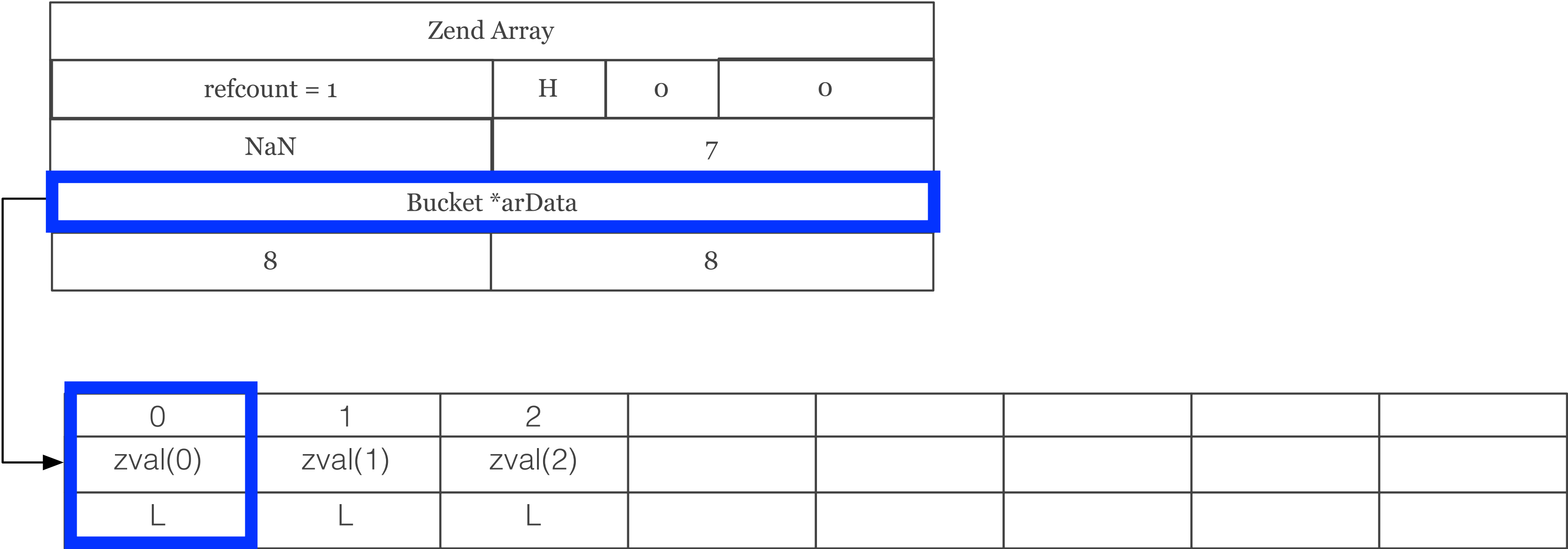


ILLUSTRATION PHP7

```
$arr = range(0, 7)
foreach($arr as $val) {
}
```



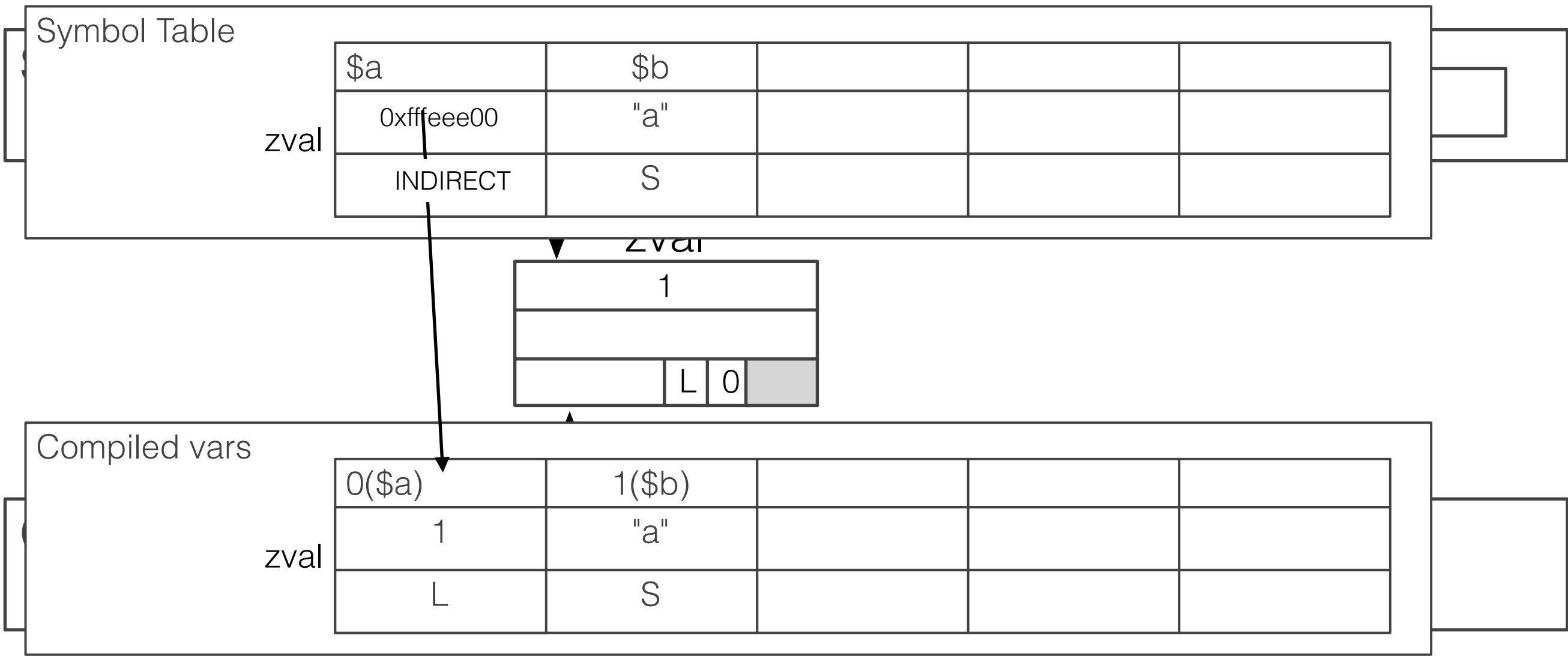
THERE COMES TROUBLES

```
function func() {  
    $a = 1;  
    $b = "a";  
    $$b = 2; //build symbol table  
    var_dump($a);  
}
```

This is not a problem in PHP5

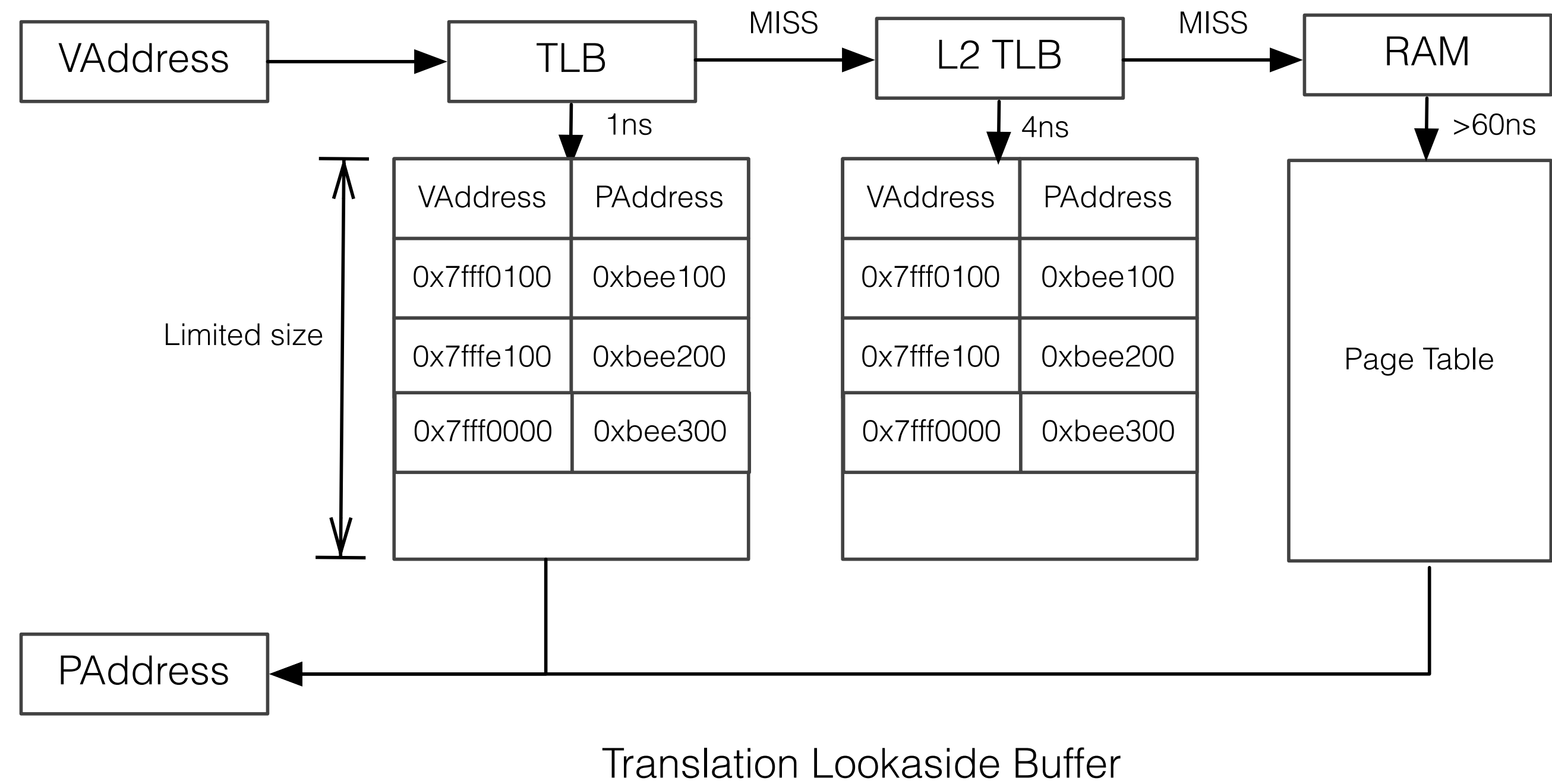
But this is a problem now

This is why IS_INDIRECT was born



Huge Pages

- ▶ 2M(4M) Page Size
- ▶ Not swappable
- ▶ Reduce TLB misses
 - ▶ $64 * 4k = 256K$
 - ▶ $8 * 2M = 16M$
 - ▶ size php binary(o2) text size $\approx 10M$
 - ▶ `opcache.huge_code_pages(iTLB)`
 - ▶ shared memory(dTLB)
 - ▶ regular memory(dTLB)
- ▶ Hugepage is not always good:
 - ▶ SIGBUS on OOM after fork
 - ▶ Hugepage on NUMA



5,795,728 iTLB-loads
1,639,782 共享的 iTLB-load-misses

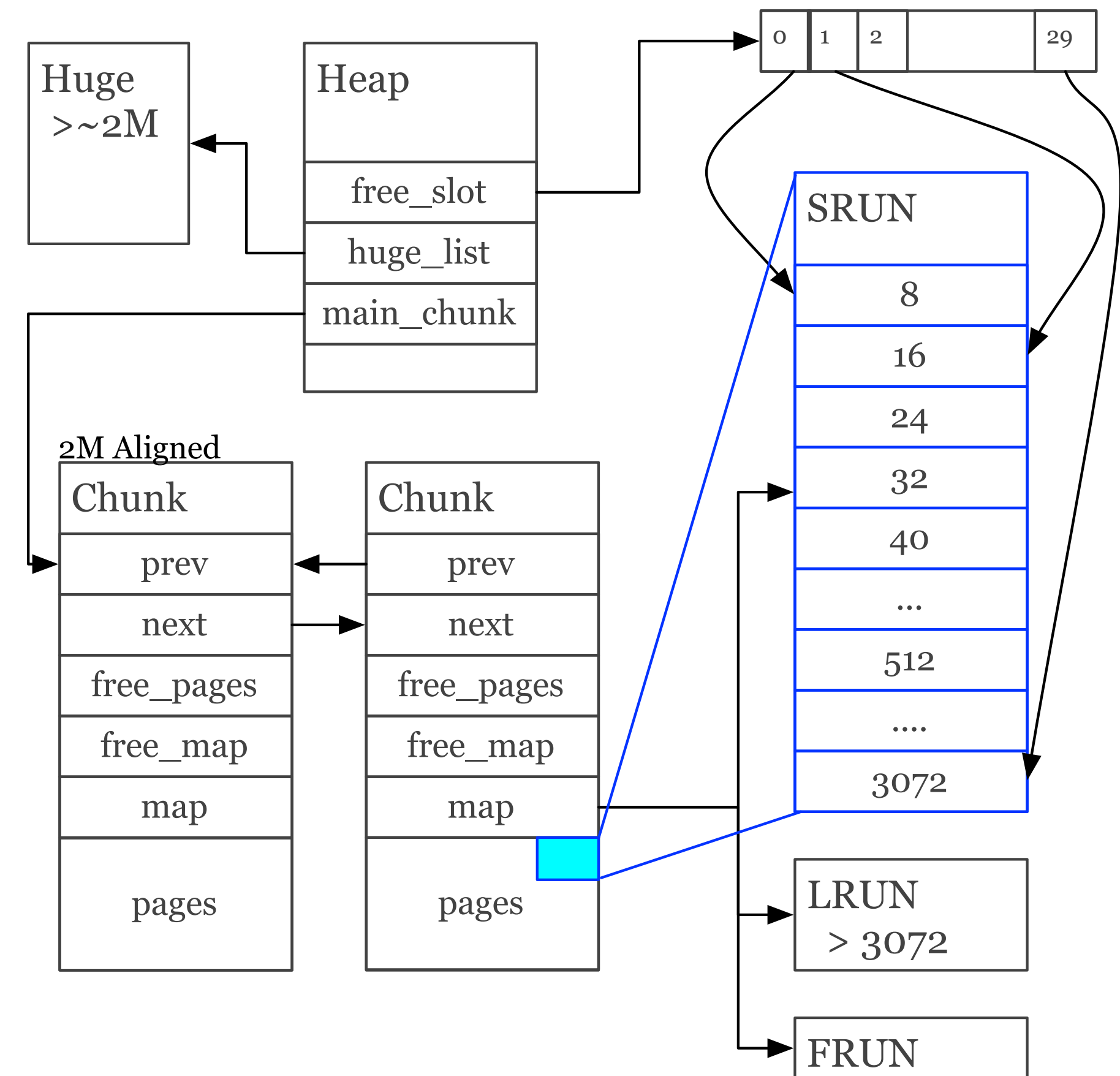
Wordpresss homepage 100 runs PHP5.5 iTLB stat

386,034 iTLB-loads
274,566 共享的 iTLB-load-misses

PHP7 (with huge_code_page) iTLB stat

PHP7 MM

- ▶ New memory manager
 - ▶ Memory is allocated in pages
 - ▶ Pages are fixed sizes in one chunk
 - ▶ Chunk is 2M aligned
- ▶ Block info is unnecessary anymore:
 - ▶ $\text{Chunk} = \text{Address} \& \sim(2M - 1)$
 - ▶ $\text{Page} = \text{Address} \& (2M - 1)$
 - ▶ `efree_size`
- ▶ Similar size mem are probably allocated nearly



PHP7 memory manager

`MEMORY` IS THE KEY

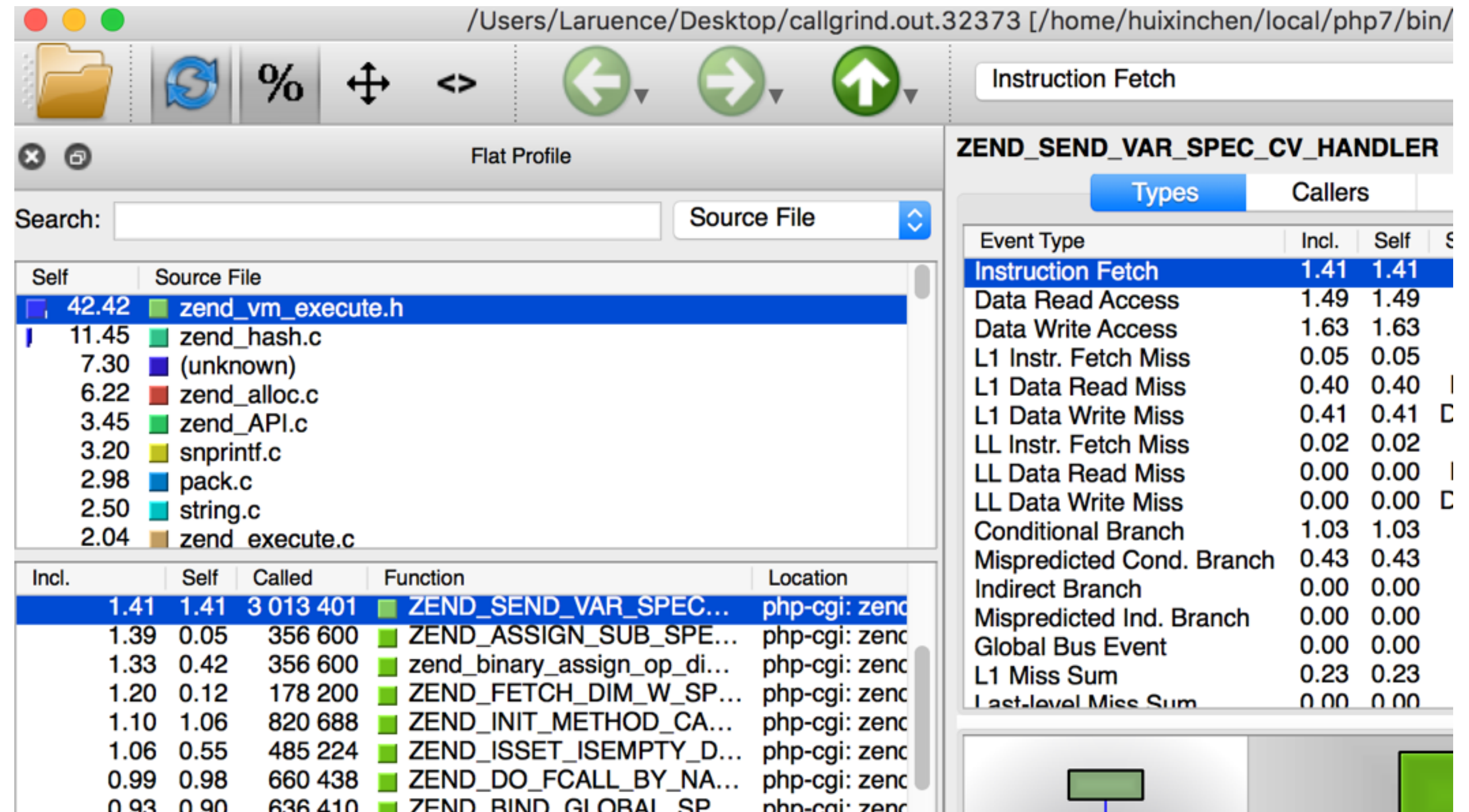
- ▶ Basically
 - ▶ Memory is reduced almost by half
 - ▶ Cache misses is significant reduced
 - ▶ TLB misses is significant reduced
 - ▶ Memory indirection is significant reduced

NOT ONLY, BUT ALSO (TL;DR)

- ▶ Zend VM refactor
- ▶ Supper global registers
- ▶ Huge Pages
- ▶ File based opcache
- ▶ No refcount for scalar types
- ▶ Function calling convention improved
- ▶ zvals are always pre-allococated or allocated in stack(no more MAKE_STD_ZVAL and ALLOC_ZVAL)
- ▶ Faster string comparing also
- ▶ New HashTable iteration AP
- ▶ Array duplication optimization
- ▶ PGO supported
- ▶ Reference-counting instead of copying
- ▶ `call_user_function(_array)` => `ZEND_INIT_USER_CALL`
- ▶ `Is_int/string/array/*` etc => `ZEND_TYPE_CHECK`
- ▶ `strlen` => `ZEND_STRLEN`
- ▶ `defined` => `ZEND+DEFINED`
- ▶ Faster sorting algo
- ▶ Immutable array
- ▶ Fast arguments parsing API
- ▶ Optimized strings concatenation.
- ▶
- ▶

PHP7 PROFILING

- ▶ 100% performance increased
- ▶ 60% IR reduced
- ▶ 40% memory usage reduced
- ▶ 20% branches reduced
- ▶ 15% iTLB misse reduced
- ▶ What a great life :)



PHP7 PERFORMANCE NEXT

- ▶ PHP 7.1
 - ▶ DFA optimization
 - ▶ Type inference
 - ▶ Type specific opcode handlers
 - ▶ Faster static vars binding
 - ▶ Dozens small improvements
 - ▶ 30% performance improvement in bench.php already
 - ▶ Significant performance improvement in reallife application
 - ▶ Alpha has been released in July 2016

Q&A