SOMETHING ABOUT PHP 7

Hiko Qiu / 2016-08-18

BEST PROGRAMMING LANGUAGE IN THE WORLD !/?

CONTENTS

WHY PHP 7

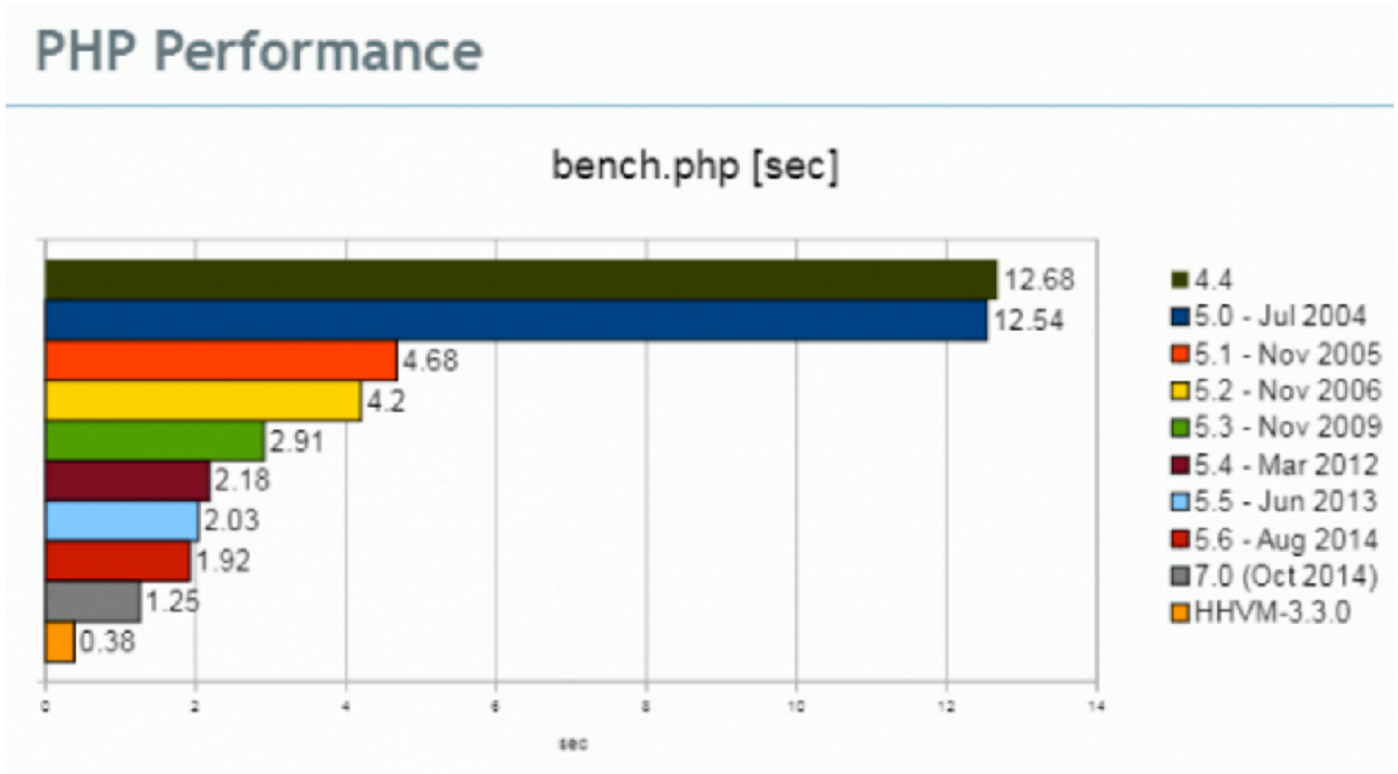
1. WHY PHP 7

PHP 6 呢 ？

什么都别说，

爱过。

WHY PHP 7 ？



为什么纠结于性能？ 专访 PHP 之父

bench.php

新特性

2. 新特性

1. Scalar type declarations

标量类型声明，(string, int, float, bool)

2. Return type declarations

3. Null coalescing operator

Null 合并运算符，??

4. Constant array using define()

使用 define() 定义常量数组

5. Exceptions in Engine

6. Other

Anonymous classes
Spaceship operator (1 <=> 2)
Group use declarations

Unicode codepoint escape syntax
Closure::call()
Filtered unserialize()
Expectations
Session options
Generator Return Expressions
Generator delegation

2. 新特性 - Scalar type declarations

PHP 5+

class name, interface, array, callback

```php
// function dealArray(array $data){}

function sum(int $a, int $b) {
    return $a + $b;
}

var_dump(sum(3, "7"));
```

PHP 7

string, int, float, bool

Type	Description	Minimum PHP version
Class/interface name	The parameter must be an *instanceof* the given class or interface name.	PHP 5.0.0
self	The parameter must be an *instanceof* the same class as the one the method is defined on. This can only be used on class and instance methods.	PHP 5.0.0
array	The parameter must be an array.	PHP 5.1.0
callable	The parameter must be a valid callable.	PHP 5.4.0
bool	The parameter must be a boolean value.	PHP 7.0.0
float	The parameter must be a floating point number.	PHP 7.0.0
int	The parameter must be an integer.	PHP 7.0.0
string	The parameter must be a string.	PHP 7.0.0

2. 新特性 - Return type declarations

支持类型：与 Scalar Type Declarations 相同

```
3 // Strict Mode
4 declare(strict_types = 1);
5
6 // function sum($a, $b): int {
7 function sum(int $a, int $b): int {
8     return $a + $b;
9 }
0
1 var_dump(sum(3, "7"));
```

2. 新特性 - 类型声明两种模式

1. Coercive Mode （强制模式， 默认）

2. **Strict Mode**（严格模式，通过在文件头 **declare(strict_types = 1),** 当前文件有效）
   i. 形参、返回值
   ii. 与强制模式区别
   a. 强制类型转换
   b. 强制模式下，如果转型失败，报 TypeError
   c. 严格模式下，直接报 TypeError

2. 新特性 - Null coalescing operator

?? = isset + ?

```php
$idMapName = [2 => 'Name_2'];

// isset($idMapName[1]) ? $name = $idMapName[1] : $name = 'No name';
$name = $idMapName[1] ?? 'No name';

$target = $idMapName[2] ?? 'No name';

var_dump($name, $target);
```

```php
// other example
$me = true ?? false;
var_dump($me);

$location = '' ?? 'No living room';
var_dump($location);
```

2. 新特性 - Constant array using define()

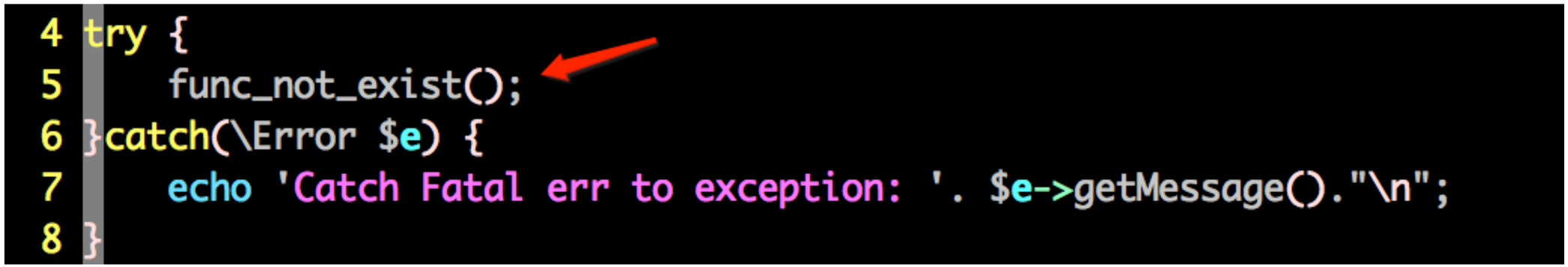
define(‘MONEY_TYPE’ ,[‘¥’ ,’ $’])

```
4 define('COMPANY', '360');
5 define('LOCATION', ['Beijing', 'Jiuxianqiao']);
6
7 var_dump(COMPANY, LOCATION);
```

2. 新特性 - Exceptions in Engine

Exceptions in Engine (核心错误捕获)

几乎所有的 Fatal 和 Catchable-Fatal (可以通过 set_error_handler() 处理) 都被替换成了 Engine exceptions ,所有未被catch的异常仍旧会导致一个 Fatal error.



```
4 try {
5     func_not_exist();
6 }catch(\Error $e) {
7     echo 'Catch Fatal err to exception: '. $e->getMessage()."\n";
8 }
```

2. 新特性 - Exceptions in Engine

其他 Engine Exception

所有新的 Engine Exception 并没有继承之前的 \Exception 类，而是继承了一个新的叫做 \Error 的基类，\Error 异常代表了PHP 7 中标准的 Fatal 和 Catchable-Fatal 错误

```
class Error implements Throwable {
    /* Inherited methods */
    abstract public string Throwable::getMessage ( void )
    abstract public int Throwable::getCode ( void )
    ...
}
```

基于\Error exception，派生了5个新的 Engine Exception：ArithmeticError / AssertionError / DivisionByZeroError / ParseError / TypeError

优化 / 提升

简单了解一下

3. 优化 / 提升

1. 变量优化

ZVAL 大小从24字节减少到16字节
Zend Array / HashTable
HashTable 大小从72字节减少到56字节
HashTable bucket 大小从72字节减少到32字节

2. 函数调用优化

通过优化参数传递的环节，减少了一些指令，加快参数解析

3. 新内存管理方式

3. 优化 / 提升 - zval

PHP 5+

```
1322 typedef union _zvalue_value {
1323     long lval;                  /* long value */
1324     double dval;                /* double value */
1325     struct {
1326         char *val;
1327         int len;
1328     } str;
1329     HashTable *ht;              /* hash table value */
1330     zend_object_value obj;
1331     zend_ast *ast;
1332 } zvalue_value;
1333
1334 struct _zval_struct {
1335     /* Variable information */
1336     zvalue_value value;     /* value */
1337     zend_uint refcount__gc;
1338     zend_uchar type;    /* active type */
1339     zend_uchar is_ref__gc;
1340 };
```

sizeof(zval) = 24 bytes

PHP 7

[github: _zval_struct]

[github: zend_val]

sizeof(zval) = 16 bytes

Link: [struct & union 大小计算]

3. 优化 / 提升 - zval

zval优化点概述

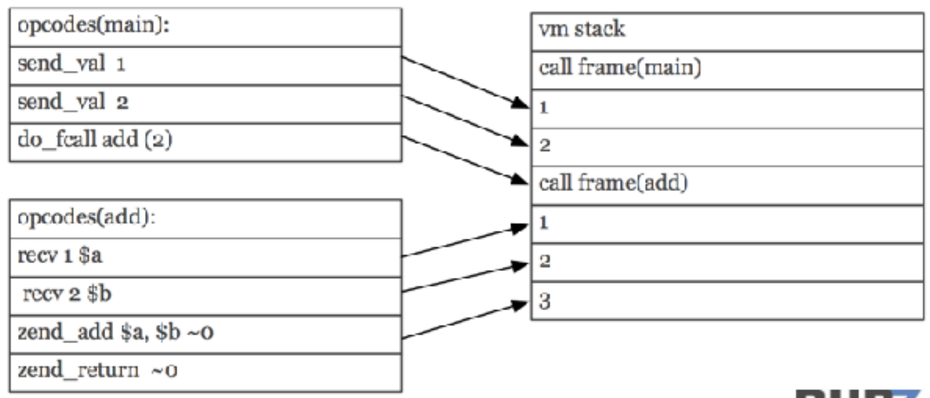
- No refcount for scalar types
- zval are always pre-allcocated or allocated in stack
- String using refcout instead of copy (zend_string)
- gc_info, temporary_variables, should_free_var, cache_slot all in zval
- New types: IS_TRUE, IS_FALSE, IS_REFERENCE, IS_INDIRECT

来自鸟哥PPT

3. 优化 / 提升 - 优化函数调用

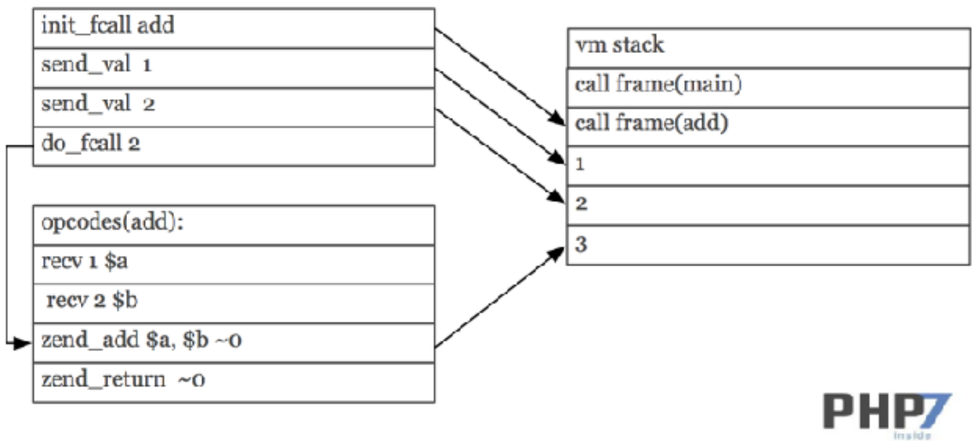
PHP 5+

- function add ($a, $b) {
  return $a + $b;
  }
  add(1, 2);



PHP 7

- function add ($a, $b) {
  return $a + $b;
  }
  add(1, 2);



信息：函数调用时，首先参数压栈，参数的入栈顺序是从右向左，最后压入函数地址。

3. 优化 / 提升 -新内存管理方式

新内存管理概述

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

来自鸟哥PPT

3. 优化 / 提升 -新内存管理方式

从CPU到	大约需要的 CPU 周期	大约需要的时间
主存		约60-80纳秒
QPI 总线传输 (between sockets, not drawn)		约20ns
L3 cache	约40-45 cycles,	约15ns
L2 cache	约10 cycles,	约3ns
L1 cache	约3-4 cycles,	约1ns
寄存器	1 cycle	

来自鸟哥PPT

[关于 CPU Cache Line]

参考

1. PHP 7.0 New Features

2. PHP 7 革新与性能优化

3. https://wiki.php.net/rfc/php6

4. PHP 7 来了，PHP 6 去哪了？

5. PHP 7 和 HHVM 性能之争

6. 专访 PHP 之父

7. https://wiki.php.net/phpng

THANKS

Q & A