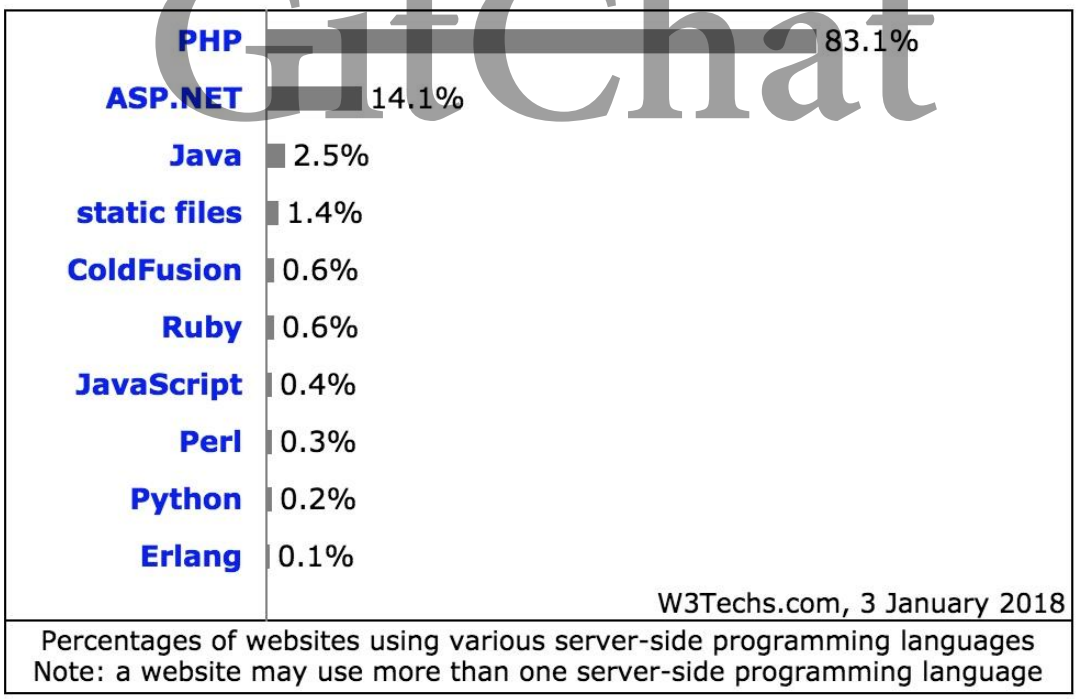
2017 年 PHP 社区总结，2018 PHP 发展展望

2017 PHP 社区总结

回顾 PHP 语言本身的升级和变化

PHP 发展史

（1）PHP 作为 Web 开发领域性价比最高的语言，已经问世20多年了。



（注：一个 Web 站点可以会使用多种语言作为它的开发语言）

（注：本文含有不少从鸟哥 PPT 里的截图，图片版权归鸟哥所有）

（2）PHP 开始于1994年，最初产生于 Rasmus Lerdorf 的一个简单的想法，当时 Rasmus 用 C 语言写了一个应用程序，这个程序就是用来追踪和维护自己的个人主页的。

（3）并且 Rasmus 对其又进行了扩展，使其可以应用于web表单还可以和数据库进行交互。就这样PHP的第一个版本就诞生了。Rasmus称其为“Personal Home Page/Forms Interpreter” 简称PHP/FI。

（4）用Rasmus自己的话说：起初并不想开发一门新的编程语言，但是随着PHP/FI的发展，渐渐的就不再受他的控制了。就这样一个开发团队行程了，并且在1997年的11月发布了第二个版本 PHP/FI 2。

（5）再往后，Zeev Suraski 和 Andi Gutmans 两个人的出现，更是使得PHP的发展走向了一个**新的里程碑**。在1997年，两人重新写了PHP的解释器，形成了PHP的第三个版本PHP3，也就是在此时正式名字由PHP/FI 改为 PHP（Hypertext Preprocessor 超文本预处理器）。时隔一年，两人在1998年又重新写了PHP的核心代码，用了将近一年的时间，Zend引擎在1999年诞生了。接着在2000年5月，带有第一代zend引擎的PHP4正式发布了。随后其发展进入了一个平缓的阶段，带有第一代Zend 引擎的PHP4在2008年8月达到4.4.9以后就再没有进行后续开发，也没有任何的安全更新。我用的最早的一个PHP4的程序应该是DEDECMS了。这时PHP还是面向过程的编程方式。

（6）在2004年6月份的时候，PHP的发展到达了**第二个里程碑**。带有Zend Engine II的PHP5正式发布，在这PHP5中开始支持面向对象，而且性能明显增强。直到2008年很多程序都不再支持PHP4版本了，取而代之的是PHP5.

接着，下一个人物该出场了，他的出现使PHP从5又上升了一格成为了PHP6。他的名字叫Andrei Zmievski。当时PHP5发布以后，PHP收到了各种各样的反馈，反馈的内容就是在PHP中缺少编码转换的支持。所以在2005年的时候，由Andrei领导在PHP中嵌入了ICU库。并且使文本字符串以unicode-16的方式呈现。这一举动，对于PHP本身以及用户的编码方式都产生了大的改变，所以PHP6应运而生了。虽说，这一改变跨越很大，但是由于开发人员不能很好的理解所做的这些改变，并且向unicode-16编码（这一编码方式在web环境中很少被用到）转换会导致性能的下降，种种原因导致这一工程停滞下来。

而且在2009年发布的PHP5.3还有2010年发布的5.4几乎涵盖了所有从PHP6移植来的功能。因此在2010年这项工程停止了，直到2014年也没有被人们所接受。

在2014-2015年期间，PHP7正式发布了。最初对于PHP7的这个版本是存在一些争议的，因为先前的PHP6并没有正式发布，就夭折了，所以直接到7这个版本并不是很合适。但是在一些学术论文还有书籍中已经引用了PHP6这个名称，所以说最终人们将其定位7。对于PHP7其主要的目标就是通过重构Zend引擎，使PHP的性能更加的优化，同时保留语言的兼容性。由于是对其引擎的重构，因此PHP7的引擎目前已是第三代 Zend Engine 3。

今天PHP7已经正式发布，纵观其从诞生到发展壮大，有成功也有失败，而今天的成功又仅仅源于昨天的一个简单的想法。作为一名程序员，如果自己在现在的一个想法，多少年后也能产生如此大的成就，那岂是一个“自豪”所能表达的。类似的情况也发生在另一个人的身上，linux的奠基者linus。不管怎么说，作为一名PHP程序员，看到PHP今天的成绩自然感到高兴，自己也会在PHP的路上一直走下去，希望PHP的发展越来越好。

PHP 7.0的优化

1. 标量类型和返回类型声明（ Scalar Type Declarations & Scalar Type Declarations ）
2. 更多的Error变为可捕获的Exception
3. AST（ Abstract Syntax Tree，抽象语法树 ）
4. Native TLS（ Native Thread local storage，原生线程本地存储 ）
5. Zval的改变
   PHP的各种类型的变量，其实，真正存储的载体就是Zval，它特点是海纳百川，有容乃大。从本质上看，它是C语言实现的一个结构体（ struct ）。对于写PHP的同学，可以将它粗略理解为是一个类似array数组的东西。

PHP5的Zval，内存占据24个字节（截图来自PPT）：

```
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;
};
```

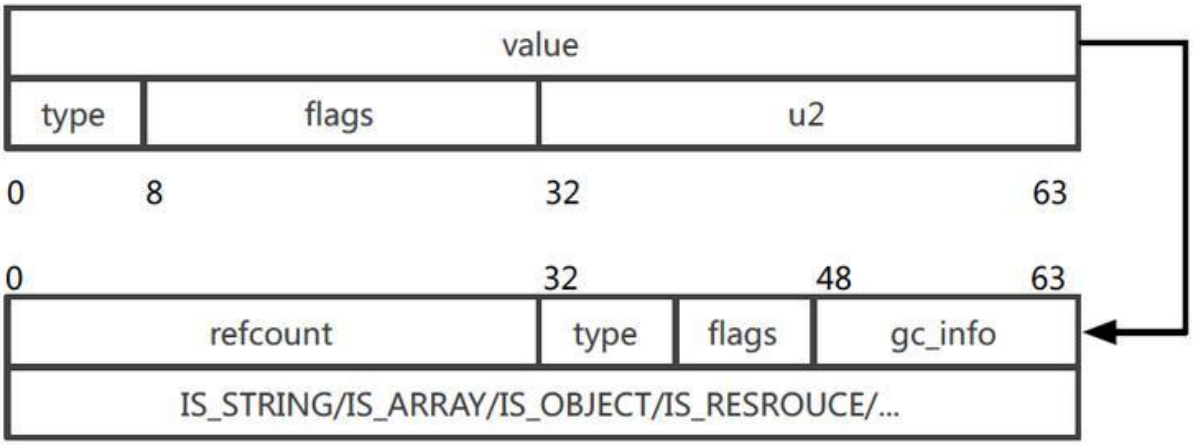

PHP7的Zval，内存占据16个字节（截图来自PPT）：

```
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;
};
```


Zval从24个字节下降到16个字节，为什么会下降呢，这里需要补一点点的C语言基础，辅助不熟悉C的同学理解。struct和union（联合体）有点不同，Struct的每一个成员变量要各自占据一块独立的内存空间，而union里的成员变量是共用一块内存空间（也就是说修改其中一个成员变量，公有空间就被修改了，其他成员变量的记录也就没有了）。因此，虽然成员变量看起来多了不少，但是实际占据的内存空间却下降了。

除此之外，还有被明显改变的特性，部分简单类型不再使用引用。

Zval结构图（来源于PPT中）：



图中Zval的由2个64bits（1字节=8bit，bit是“位”）组成，如果变量类型是long、bealoon这些长度不超过64bit的，则直接存储到value中，就没有下面的引用了。当变量类型是array、objec、string等超过64bit的，value存储的就是一个指针，指向真实的存储结构地址。

对于简单的变量类型来说，Zval的存储变得非常简单和高效。

不需要引用的类型：NULL、Boolean、Long、Double

需要引用的类型：String、Array、Object、Resource、Reference

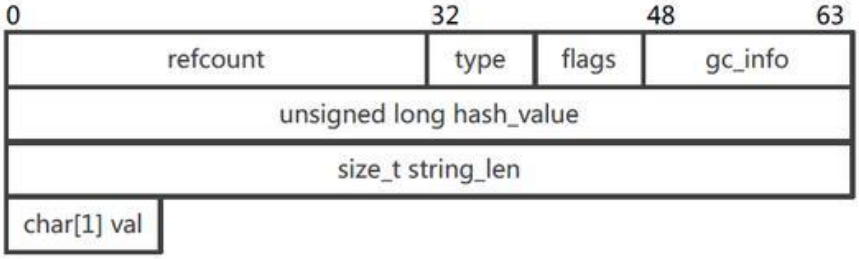
（1）内部类型zend_string

Zend_string是实际存储字符串的结构体，实际的内容会存储在val（char，字符型）中，而val是一个char数组，长度为1（方便成员变量占位）。

- 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



结构体最后一个成员变量采用char数组，而不是使用char*，这里有一个小优化技巧，可以降低CPU的cache miss。

如果使用char数组，当malloc申请上述结构体内存，是申请在同一片区域的，通常是长度是sizeof(_zend_string) + 实际char存储空间。但是，如果使用char*，那个这个位置存储

的只是一个指针，真实的存储又在另外一片独立的内存区域内。

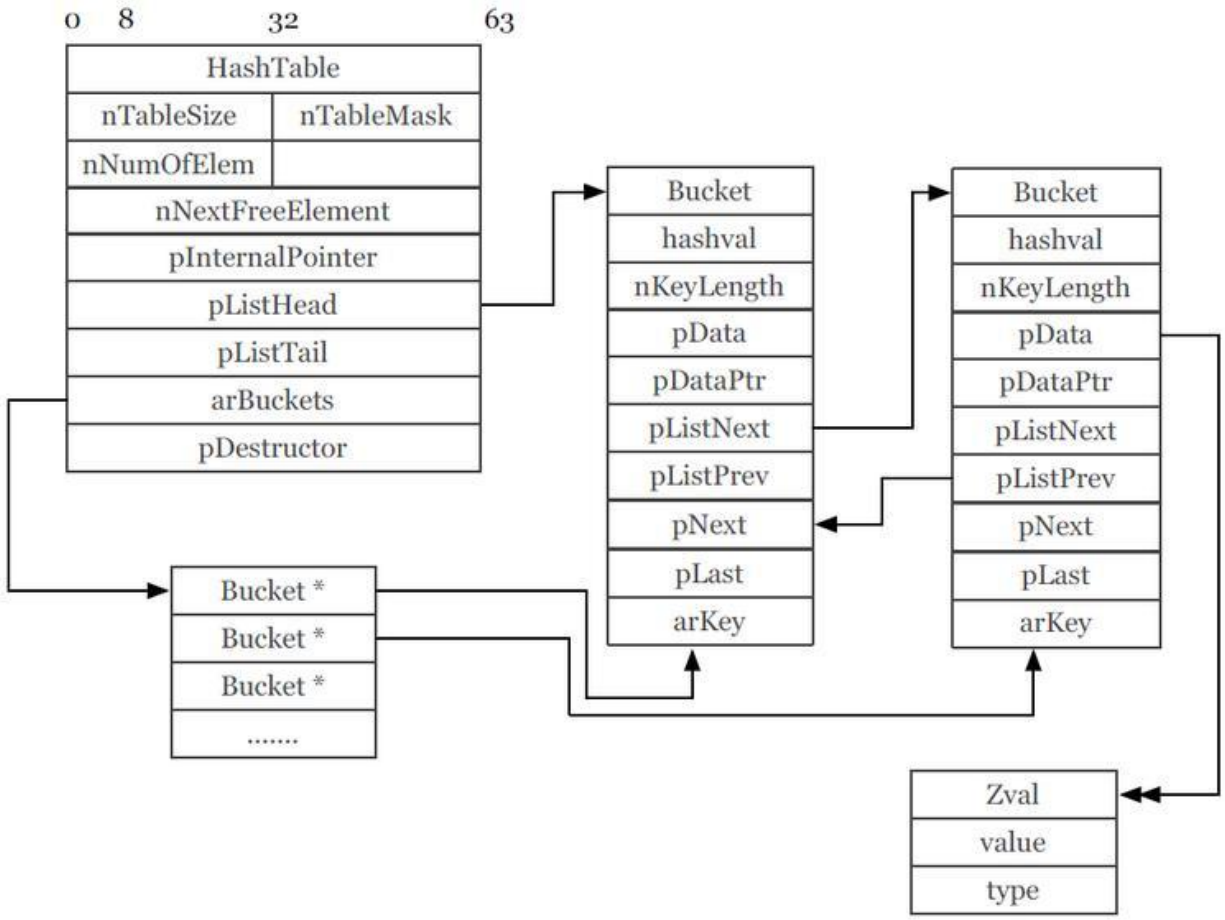
使用char[1]和char*的内存分配对比：

从逻辑实现的角度来看，两者其实也没有多大区别，效果很类似。而实际上，当这些内存块被载入到CPU的中，就显得非常不一样。前者因为是连续分配在一起的同一块内存，在CPU读取时，通常都可以一同获得（因为会在同一级缓存中）。而后者，因为是两块内存的数据，CPU读取第一块内存的时候，很可能第二块内存数据不在同一级缓存中，使CPU不得不往L2（二级缓存）以下寻找，甚至到内存区域查到想要的第二块内存数据。这里就会引起CPU Cache Miss，而两者的耗时最高可以相差100倍。

另外，在字符串复制的时候，采用引用赋值，zend_string可以避免的内存拷贝。

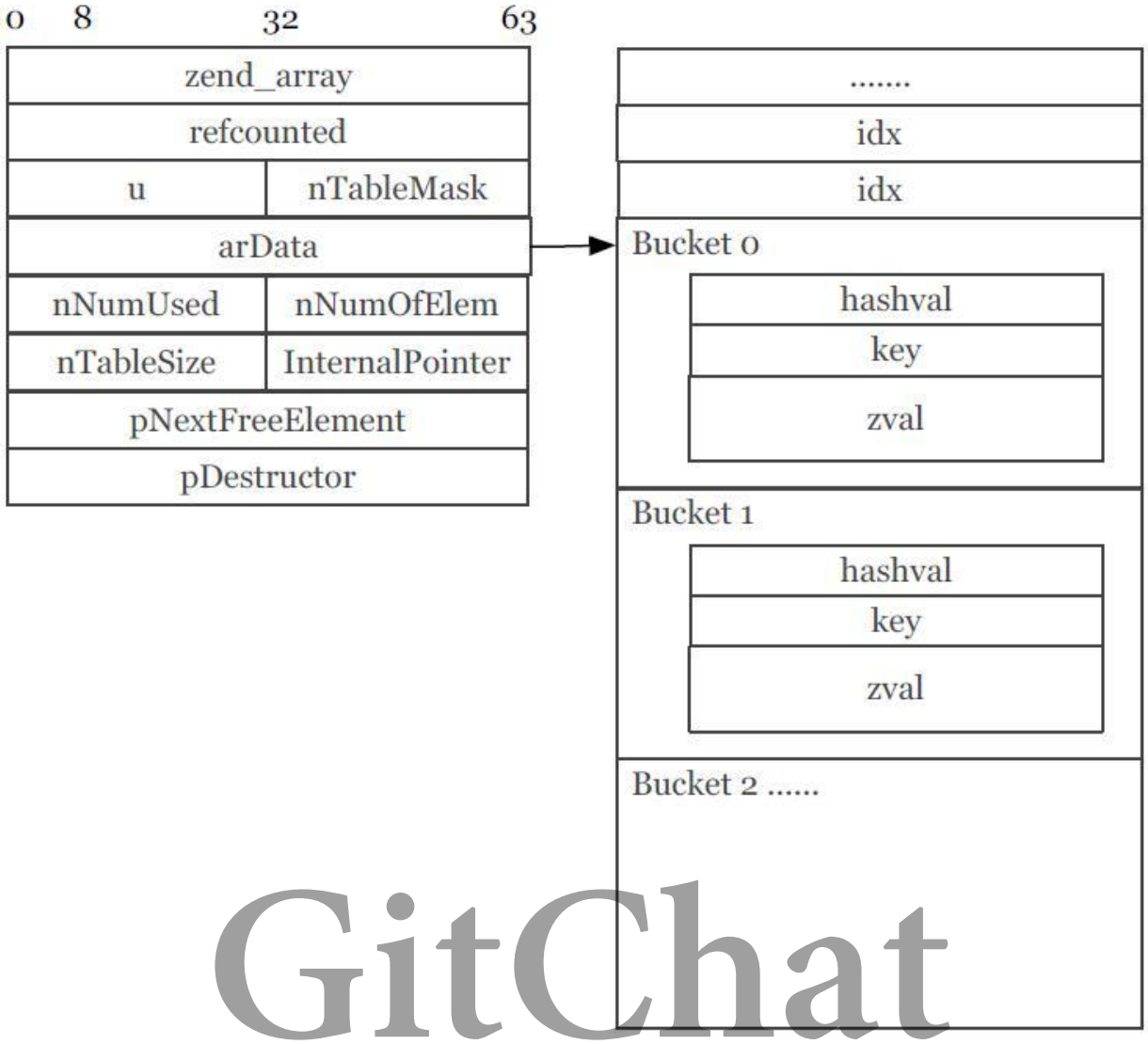
（2）PHP数组的变化（HashTable和Zend Array）

在编写PHP程序过程中，使用最频繁的类型莫过于数组，PHP5的数组采用HashTable实现。如果用比较粗略的概括方式来说，它算是一个支持双向链表的HashTable，不仅支持通过数组的key来做hash映射访问元素，也能通过foreach以访问双向链表的方式遍历数组元素。



这个图看起来很复杂，各种指针跳来跳去，当我们通过key值访问一个元素内容的时候，有时需要3次的指针跳跃才能找对需要的内容。而最重要的一点，就在于这些数组元素存储，都是分散在各个不同的内存区域的。同理可得，在CPU读取的时候，因为它们就很可能不在同一级缓存中，会导致CPU不得不到下级缓存甚至内存区域查找，也就是引起CPU缓存命中下降，进而增加更多的耗时。

PHP7的Zend Array（截图来源于PPT）：



新版本的数组结构，非常简洁，让人眼前一亮。最大的特点是，整块的数组元素和hash映射表全部连接在一起，被分配在同一块内存内。如果是遍历一个整型的简单类型数组，效率会非常快，因为，数组元素（Bucket）本身是连续分配在同一块内存里，并且，数组元素的zval会把整型元素存储在内部，也不再有指针外链，全部数据都存储在当前内存区域内。当然，最重要的是，它能够避免CPU Cache Miss（CPU缓存命中率下降）。

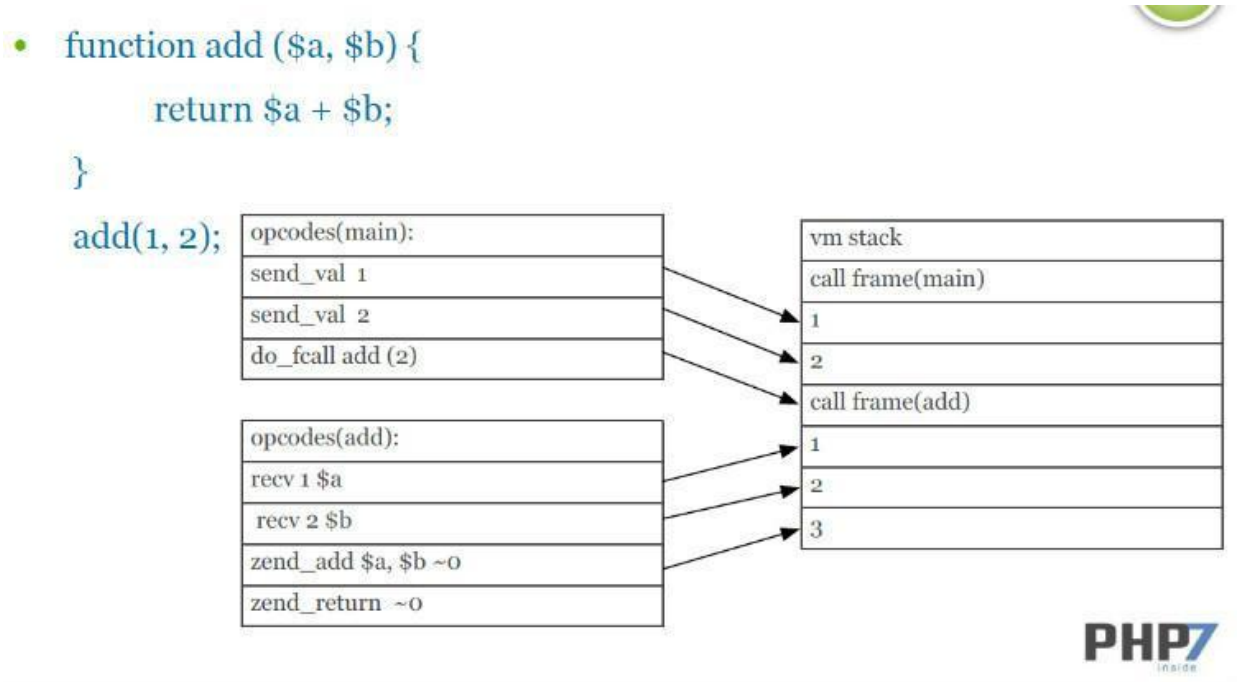
Zend Array的变化：

- 数组的value默认为zval。
- HashTable的大小从72下降到56字节，减少22%。
- Buckets的大小从72下降到32字节，减少50%。
- 数组元素的Buckets的内存空间是一同分配的。
- 数组元素的key（Bucket.key）指向zend_string。
- 数组元素的value被嵌入到Bucket中。
- 降低CPU Cache Miss。

（3）函数调用机制（Function Calling Convention）

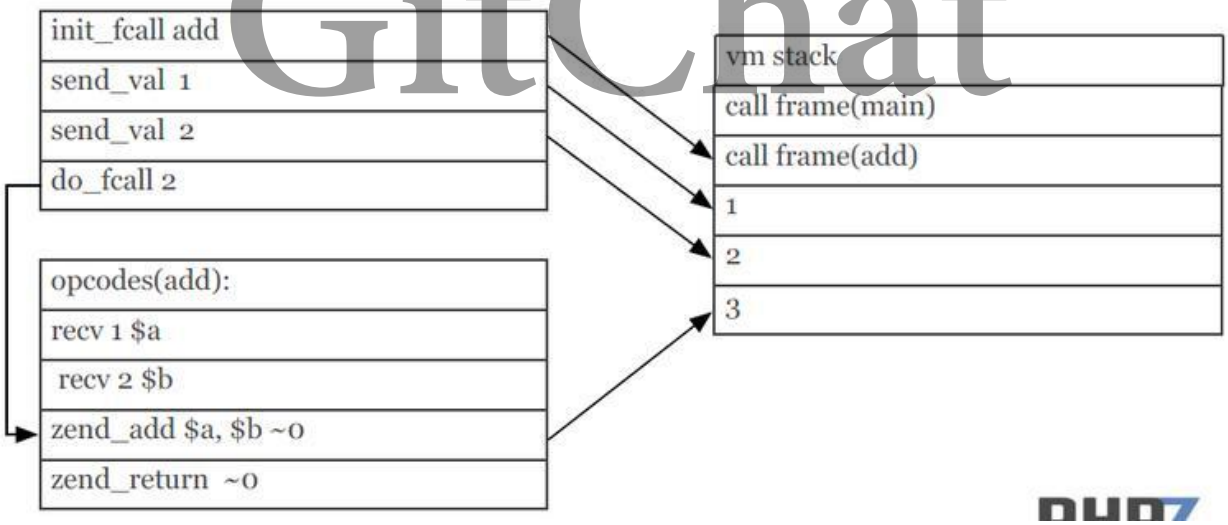
PHP7改进了函数的调用机制，通过优化参数传递的环节，减少了一些指令，提高执行效率。

PHP5的函数调用机制（截图来自于PPT）：



图中，在vm栈中的指令send_val和recv参数的指令是相同，PHP7通过减少这两条重复，来达到对函数调用机制的底层优化。

PHP7的函数调用机制（截图来自于PPT）：



通过宏定义和内联函数（inline），让编译器提前完成部分工作
C语言的宏定义会被在预处理阶段（编译阶段）执行，提前将部分工作完成，无需在程序运行时分配内存，能够实现类似函数的功能，却没有函数调用的压栈、弹栈开销，效率会比较高。内联函数也类似，在预处理阶段，将程序中的函数替换为函数体，真实运行的程序执行到这里，就不会产生函数调用的开销。

PHP7在这方面做了不少的优化，将不少需要在运行阶段要执行的工作，放到了编译阶段。例如参数类型的判断（Parameters Parsing），因为这里涉及的都是固定的字符常量，因此，可以放到到编译阶段来完成，进而提升后续的执行效率。

例如下图中处理传递参数类型的方式，从左边的写法，优化为右边宏的写法。

```
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();
```

PHP7.1的优化

1. **可空类型** 可空类型主要用于参数类型声明和函数返回值声明。
2. **list 的方括号简写** 我们知道在 PHP5.4 之前只能通过 array() 来定义数组，5.4之后添加了 [] 的简化写法
3. **允许在 list 中指定 key**
4. **void 返回类型**PHP7.0 添加了指定函数返回类型的特性，但是返回类型却不能指定为 void，7.1 的这个特性算是一个补充
5. **类常量属性设定**
6. **多条件 catch**
7. 详见RFC 地址

php7.2的变化

语法、函数方面更严格了,性能也有提升
详见
PHP 7 ChangeLog

PHP的性能演进(从PHP5.0到PHP7.1的性能全评测)

PHP周边优秀开源项目

框架篇

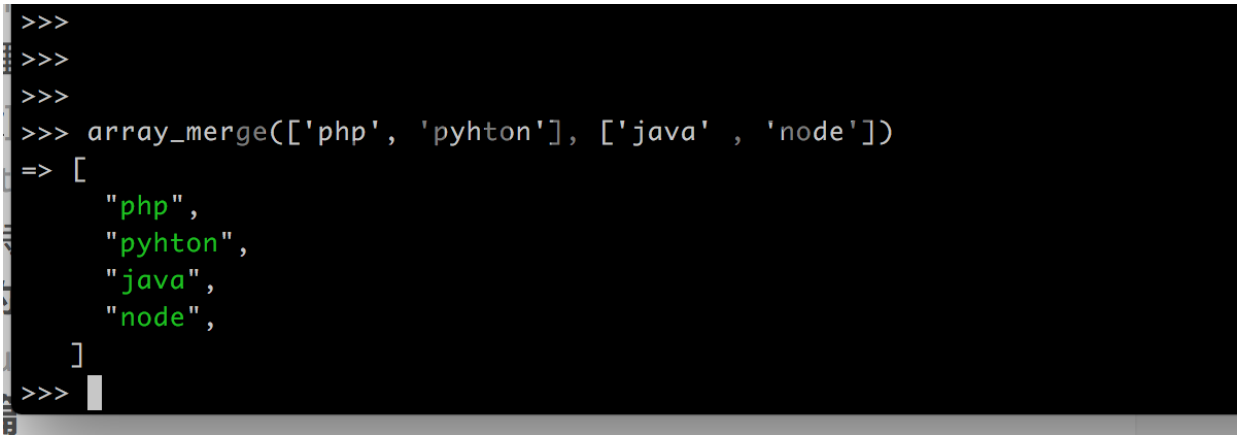
1. 首先就是目前最火热的Laravel了, 类似ROR的语法 丰富的组件 活跃的社区 更多的语法糖 (糖虽好,不要多吃哦)
2. 然后就是由我们的华人开发的Yii框架 Gii生成代码 简直是爽的不要的不要的
3. 还有国产的ThinkPHP 快速上手
4. 还有codeigniter简洁方便 还有Phalcon是用C语言开发的框架 以PHP拓展的形式 性能强悍 功能强大 等等优秀的框架
5. 我就抛砖引玉吧 更多优秀的框架等待大家发现 使用

组件篇

1. 国产的Swoole 重新定义PHP 异步 高性能
2. 国产的SeasLog C拓展高性能的Log组件
3. 在此强烈推荐大家看看Symfony 这个框架解耦做到极致,每个模块都是一个组件,都可以单独拿出来使用
4. 大家应该会和微信开发打交道 强烈推荐EasyWeChat 封装适度 简洁的API
5. PHP-ml 是 PHP 的机器学习库。同时包含算法，交叉验证，神经网络，预处理，特征提取等
6. 等等等优秀组件

工具篇

1. 首当其冲推荐Composer,优秀的PHP包管理工具,有了它,你就可以”借刀杀人”=>把社区优秀的开源组件为自己所用
2. 还有一个我经常使用的的Psysh,调试验证小段代码信手拈来!



1. Xdebug 断点调试利器,让BUG无处可逃!
2. Xhprof 专注于性能分析,找出你代码慢在哪里,持续优化 持续提高性能
3. Deployer项目部署工具 简单 快捷
4. Piplint 新鲜出炉的 项目部署系统 国产 中文文档支持
   更多有趣的PHP组件 可以关注这个项目 awesome-php

项目篇

1. Fecshop 基于 PHP Yii2 框架开发的一款优秀的开源电商系统 可持续发展的电商系统
2. zentaopms 国内市场市场占有率最高的项目管理 BUG管理软件 用过的都知道
3. OpenCart 国外电商适用的开源电商系统

PHP在Web生态中的变化

一句话总结PHP在Web生态中的变化 **“更快更高更强”**
更快=> 从php5.6到php7.0 从7.0到7.1 从7.1到7.2 持续的让内核性能提高 压榨每一个字节 优化每一行代码
更高=> 从DedeCMS 帝国CMS Discuz 老牌的Web系统 到现在的组件化 抽象程度更高了
更强=> 从经典的Web开发 到现在PHP-ML机器学习 UI 桌面软件开发 PHP更强了

国内的PHP社区

1. 首当其冲推荐Laravel-Chain 不仅仅是Laravel
2. 还有SF 有技术问题可以去提问哦
3. 一个小而美的社区V2EX
4. 更多有趣的社区,大家留言评论哦

2018PHP发展展望

PHP周边生态的发力

编程语言最重要的生态，社区活跃则语言繁荣，社区衰落则语言式微，就好比水与鱼的关系。

正在开发中的libpkd => 拓展语言原生能力，构建运行时标准库 ==> 有搞头哦

目前已经基于Swoole生态涌出很多优秀的框架

1. swoft-cloud崭露头角的PHP微服务框架 可能是PHP社区的Spring-Cloud哦 很看好
2. SwooleDistributedSwooleDistributed 是一款 PHP 结合 swoole 扩展开发的开源高性能的服务器框架。 Swoole打造底层基石,SD构建上层应用 完美结合
3. 等等

PHP程序猿的努力

随着开源的流行 越来越多的PHPer在贡献自己的一份力量 GitHub上PHP的开源项目越来越多
[packagist] (http://packagist.org/)上的组件越来越丰富
PHP语言本身的性能不断提升 让PHP语言换发出新的生命力

最后

最期待的还是PHP8 据说会加上JIT 让PHP上一个新台阶
最后祝大家在新的一年里 BUG少少的 幸福多多的

笔者才疏学浅,文中有不恰当之处,烦请大家斧正

参考资料

1. PHP
2. 技术行者
3. 风雪之隅