



Сладкое будущее: Phalcon и Zephir

Dmitry Patsura / [@ovr](#)

В ближайшие 30 минут

- Фреймворки на Си для PHP
- Phalcon 1
- Zephir
- Zephir Runtime
- Phalcon 2

Фреймворки на Си для PHP

Уже не новость, а сложившийся тренд.

Yaf (yet another framework)

Yaf – это первый PHP микро-фреймворк, взявший за основу структуру приложения Zend Framework, но написанный на С и является PHP extension доступным через PECL.

Phalcon

Phalcon - это веб-фреймворк с высокой производительностью и низким потреблением ресурсов, собранный в виде Си-расширения для PHP.

Плюсы

- Производительность (~6 раз быстрее Zend 2)
- Низкое потребление памяти (0.75 МБ)
- Простота использования (в коде)
- Наличие готовых решений
- Большая универсальность

Минусы

- Разработка и исправление
- Отладка (GDB, Valgrind)

А что за железяка?

Intel® Xeon® E3-1270 v3 Quad-Core Haswell 4 core / 8
hyper threads

RAM 32GB

2 x 240 GB SATA RAID 1 SSD

А теперь еще софт

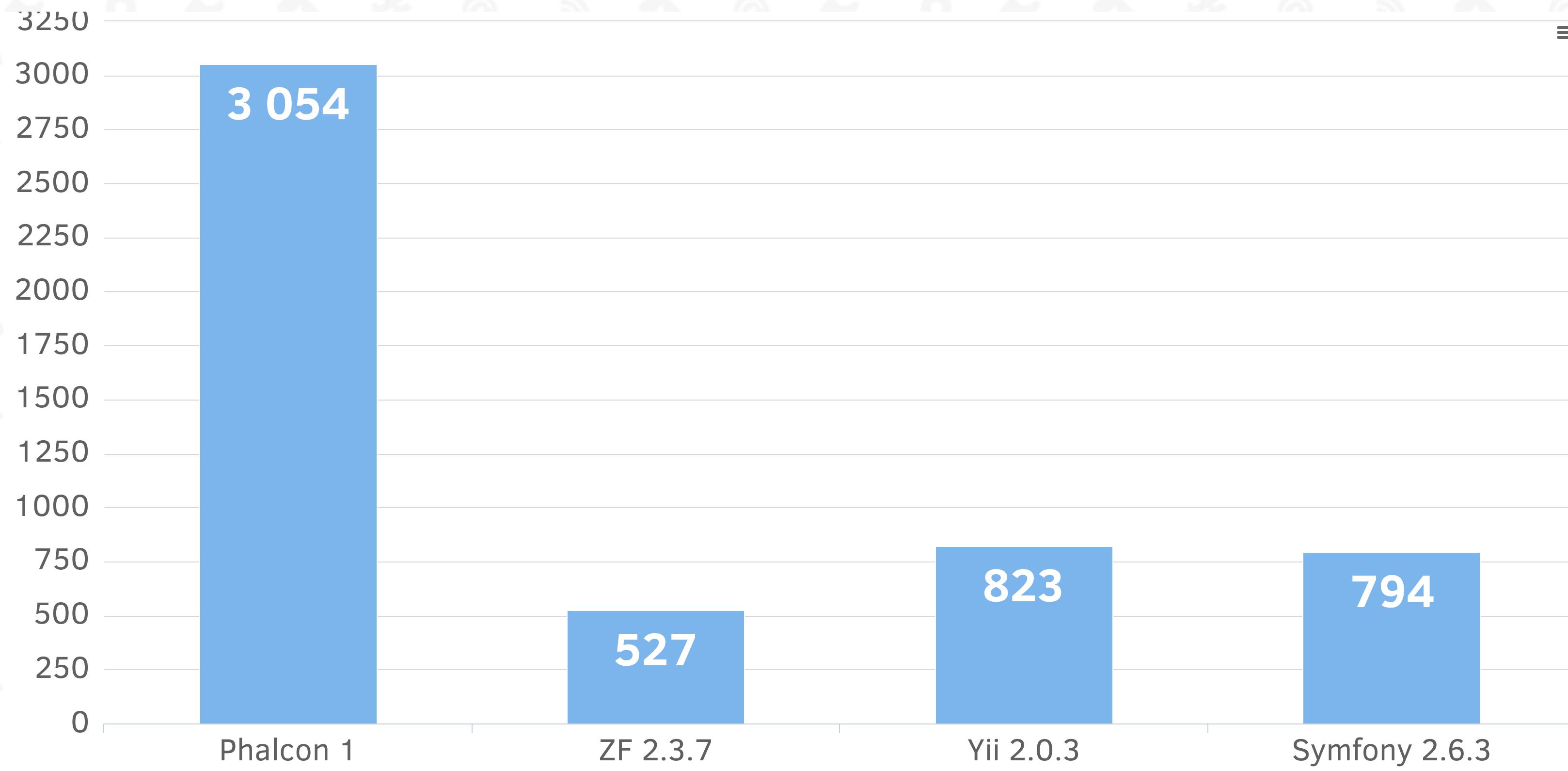
GNU/Linux 3.13 Ubuntu x86_64

PHP 5.6.6 FPM SAPI + OPCache

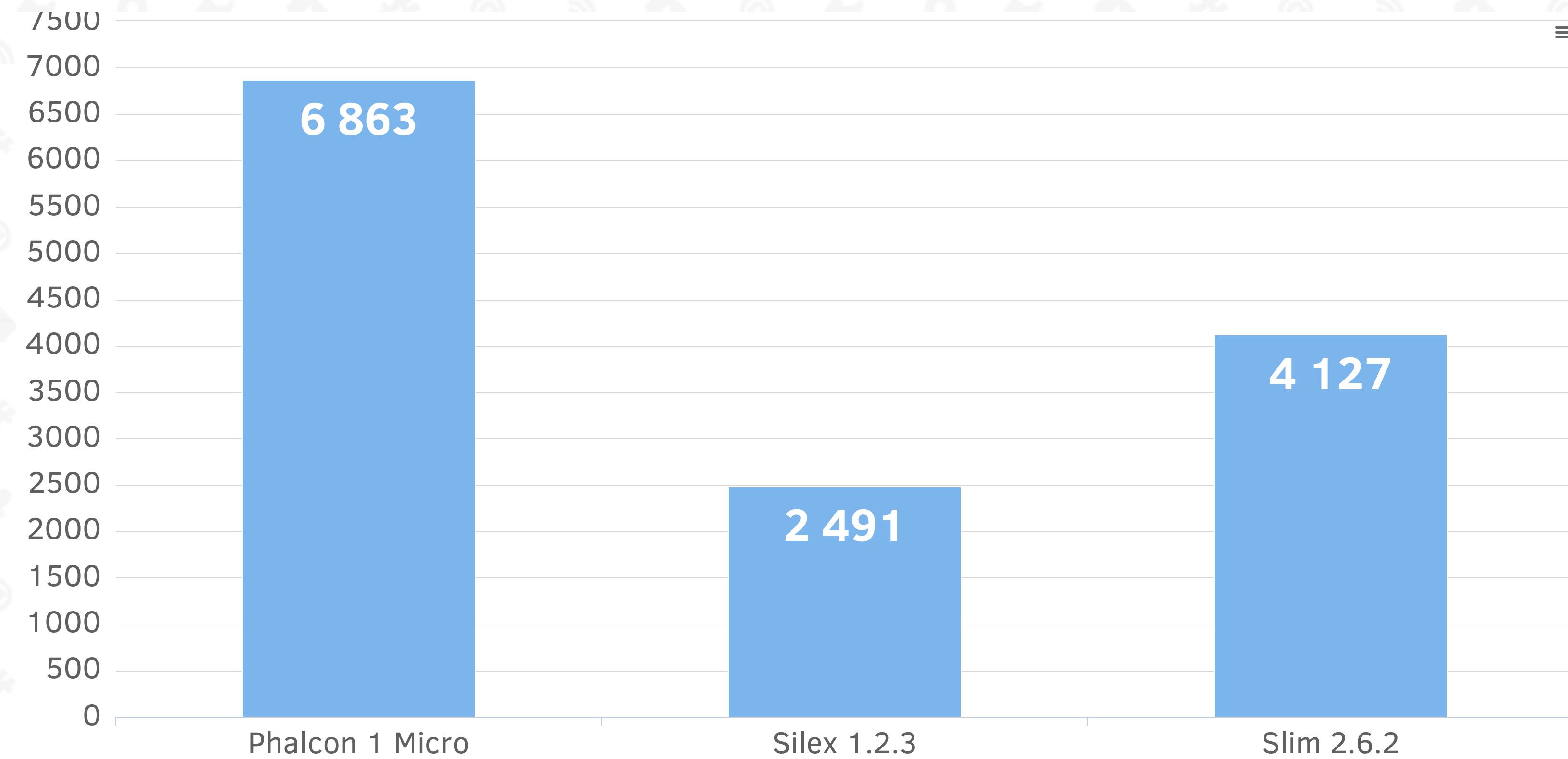
Nginx 1.6.2

Тестируем через Apache benchmark (ab)

Производительность MVC (RPS)



Производительность Micro (RPS)



ЧТО ВНУТРИ?

- Си-код
- Работа с Zend Engine
- Парсер для аннотаций и PQL (Phalcon Query Language)

```
PHP_METHOD(Phalcon_Mvc_Application, registerModules){

    zval *modules, *merge = NULL, *registered_modules, *merged_modules = NULL;

    PHALCON_MM_GROW();

    if (PHALCON_IS_FALSE(merge)) {
        phalcon_update_property_this(this_ptr, SL("_modules"), modules TSRMLS_CC);
    } else {
        PHALCON_OBS_VAR(registered_modules);
        phalcon_read_property_this(&registered_modules, this_ptr, SL("_modules"), PH_NOISY TSRMLS_CC);

        if (Z_TYPE_P(registered_modules) == IS_ARRAY) {
            PHALCON_INIT_VAR(merged_modules);
            phalcon_fast_array_merge(merged_modules, &registered_modules, &modules TSRMLS_CC);
        } else {
            PHALCON_CPY_WRT(merged_modules, modules);
        }

        phalcon_update_property_this(this_ptr, SL("_modules"), merged_modules TSRMLS_CC);
    }

    RETURN_THIS();
}
```

Си для PHP разработчика - это сложно

Как же исправить ситуацию?

Это DSL

- Java
 - Scala
 - Groovy
 - Clojure
- Javascript
 - CoffeeScript
 - LiveScript
 - TypeScript

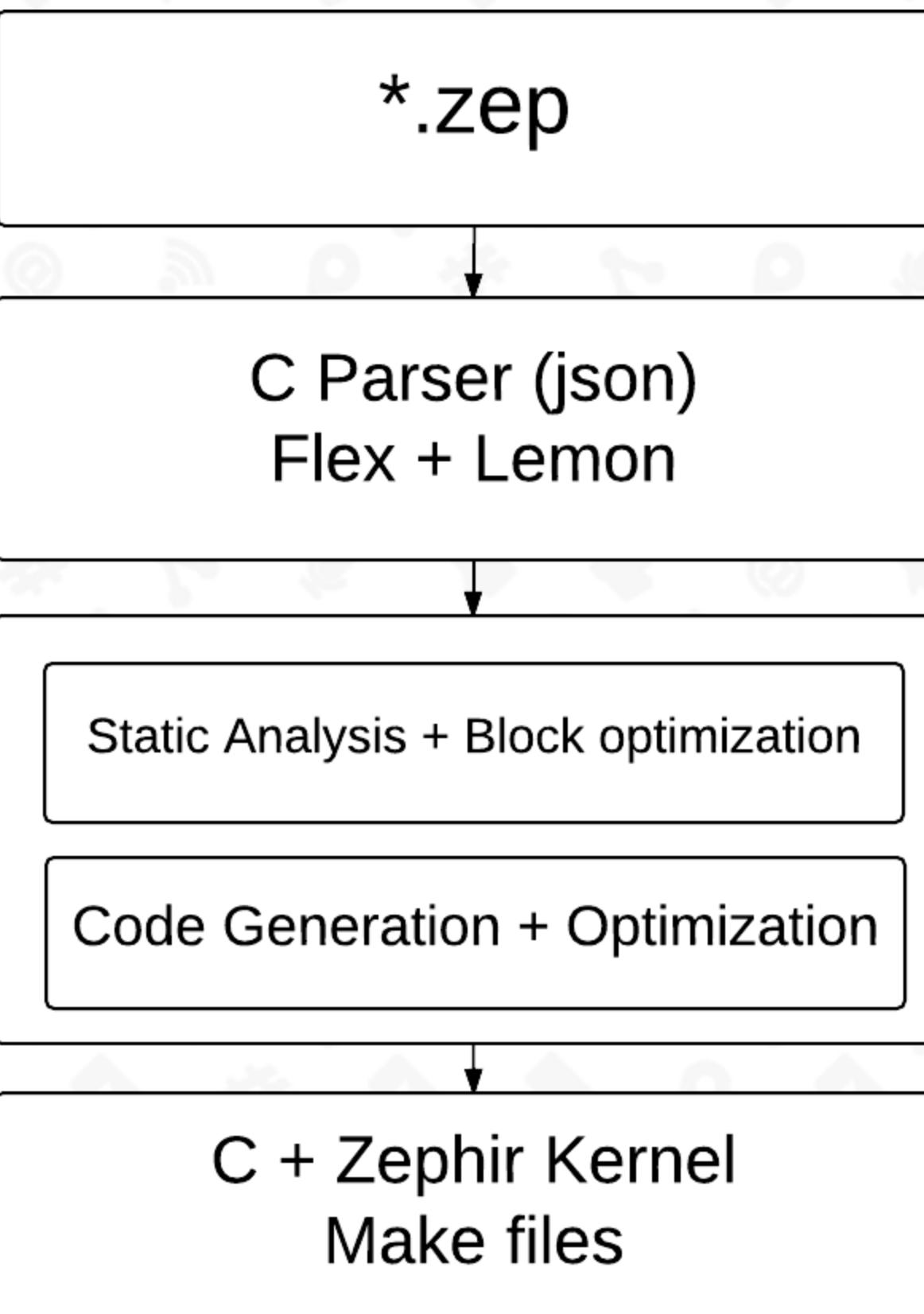
А что же для Zend Engine?

Zephir

Ze(nd Engine) Ph(p) I(nt)r(mediate)

Zephir — это высокоуровневый язык программирования с открытым исходным кодом для быстрого и простого создания PHP расширений.

Как изнутри?



Самые видимые различия после PHP

- Переменная должна быть объявлена перед использованием
- Статическая\Динамическая типизация
- Пишем все в пространстве имен
- Глобальный код запрещен
- Компиляция Ahead-of-time (АОТ)
- Ура больше нет <?php и \$
- Безопасная работа с памятью (Memory safety)

Статическая\Динамическая типизация

```
// Динамическая Примитив\Zval\HashTable
var a = 1;

// Массив
var b = [1, 2, 3];
array b = [1, 2, 3];

// Статические
int a = -1;
uint a = 1;
float pi = 3.14;
bool a = true;
string a = "Test string";
char a = 'A';
```

Установка значений

осуществляется через оператор let

```
let expr = expr или scalar value;
```

```
let a = [1, 2, 3, 4, 5];
```

```
let b = true;
```

```
let c = new Object();
```

Установка возвращаемого типа

```
namespace Test;

class Test {
    public function testReturnInt() -> int
    {
        return 1;
    }

    public function testReturnArray() -> array
    {
        return [1, 2, 3, 4, 5];
    }

    /**
     * Но устанавливать возвращаемый тип не обязательно
     */
    public function testWithoutReturn()
    {

    }
}
```

Нативные getter/setter

```
class Test {  
    protected $request;  
  
    public function setRequest($request)  
    {  
        $this->request = $request;  
    }  
  
    public function getRequest()  
    {  
        return $this->request;  
    }  
}
```

Zephir

```
class Test {  
    protected request {get, set};  
}
```

Диапазоны значений (Range)

Было в PHP

```
<?php  
  
foreach (range(0, 100) as $number) {  
    echo $number;  
}
```

Стало в Zephir

```
for n in [1..100] {  
}
```

Использование диапазонов в for

```
namespace Test;

function iterateVarRange()
{
    var a;

    for a in [1..100] {
        echo a; // Компилятор будет решать, что использовать для типа
    }
}

function iterateIntRange()
{
    int i;

    for i in [1..100] {
        echo i;
    }
}
```

Встроенные методы для всех типов

(Built-in methods)

- Array
- Char
- Double
- Int
- String

Built-in for int

Built-in for array

```
[4, 8, 12, 32, 5]->map(  
    function(int x) {  
        return x * x;  
    }  
)
```

Closures short syntax

```
[4, 8, 12, 32, 5]->map(x => x * x);
```

Анонимные переменные

```
namespace Test;

function anonymousKeyFor(array data)
{
    for _ in data {
        echo "hello";
    }
}

function anonymousValueWalk(array data)
{
    data->walk(
        function(_, int key) { echo key; }
    );
}
```

Статический анализ

```
namespace Test;

function testUnusedVal()
{
    var v;
    array data = [1, 2, 3, 4, 5, 6, 7];

    for v in data {
        echo "hello";
    }
}
```

Но получим Warning

```
Warning: Variable "v" assigned but not used in Test::testUnusedVal in
/Users/ovr/projects/zephir/functional.zep on 5 [unused-variable]
```

```
for v in data {  
-----^
```

Забыл про анонимные переменные?

```
namespace Test;  
  
function testUnusedVal()  
{  
    for _ in data {  
        echo "hello";  
    }  
}
```

Оптимизации

- FunctionCall - Optimizers
- Runtime function's call cache
- SkipVariantInit
- Constant folding

Вызов функции в PHP - не торт

Медленно

- Много кода
- Работа со стеком
- Парсинг параметров
- Нет нормальной декларации функции в Си коде

```
namespace Test\Math;

function calculateMath() {
    int i;

    // 200 000
    for i in range(1, 200000) {
        abs(i);
        acos(i);
        asin(i);
        atan(i);
        floor(i);
        exp(i);
    }
}
```

Zephir time: 0.08000

PHP: 10.0

SkipVariantInit (if-else-statement)

```
class Test {
    public fn test() {
        if {expr} {
            var a = 1;
        } else {
            var a = 1;
        }
    }
}
```

SkipVariantGet (еще в работе)

```
class Test {  
    property prop1 = 1;  
  
    public fn test() {  
        var a, b, c, d;  
  
        let a = this->prop1,  
            b = this->prop1,  
            c = this->prop1,  
            d = this->prop1;  
  
        //....  
    }  
}
```

Constant folding

Раскрытие статических мест

```
class Test {
    const TEST1 = 1;

    public fn getTEST1()
    {
        return self::TEST1; //Константу можно подставить во время компиляции
    }

    public fn getClass()
    {
        return __CLASS__; //Знаем
    }

    public fn getPHPVersion()
    {
        return PHP_VERSION; //Возьмем runtime
    }
}
```

Дополнительные возможности

- Генерация автocomплита для IDE (IDE stubs)
- Генерация документации (аналогично PHPDocumentor)

и другие возможности ;)

Минусы

- Еще не реализованы Trait
- Не хватает оптимизаций
- Версия 0.6.1 Alpha (Уже практически Beta)

Zephir Runtime

Zephir Runtime - это JIT компилятор встроенный в расширение.

func.zep

```
fn myFunction(var a) {  
    return 1;  
}
```

index.php

```
include_once 'func.zep';  
  
$result = myFunction(1);  
var_dump($result);
```

Будущее Zephir

- Поддержка PHP 7 (Zend Engine 3)
- Больше оптимизаций
- Переработка архитектуры на раздельные компоненты
- Больше бекендов (PHP, Hack, PHP-CPP)
- Развитие Zephir Runtime
- Хотим реализовать ZephirVM*

Phalcon 2

- Переписан на Zephir
- 99.99% Совместимый фреймворк
- Улучшенная кодовая база
- Улучшены тесты

Как установить на локале?

```
sudo apt-get install php5-dev php5-mysql gcc make re2c libpcre3-dev  
git clone -b 2.0.0 https://github.com/phalcon/cphalcon.git  
cd ext  
sudo ./install
```

Ну а вообще

Ansible, Chef или Puppet

Как разрабатывать?

Берем виртуалку на Vagrant +

- VirtualBox *
- LXC (Linux Kernel Containers) ****
- Parallels под OSX ***
- VMWare (Плагин платный) ***

Сообщество Phalcon и Zephir

- Сайт
- Форумы
- Группы для Phalcon и Zephir в ВКонтакте
- Чатик на Гиттер
- Блоги
- Твиттер

Спасибо за внимание :3



<http://dmtry.me/codefest2015>