Hacklang

Больше, чем **PHP**

Тарас Полищук
TripMyDream

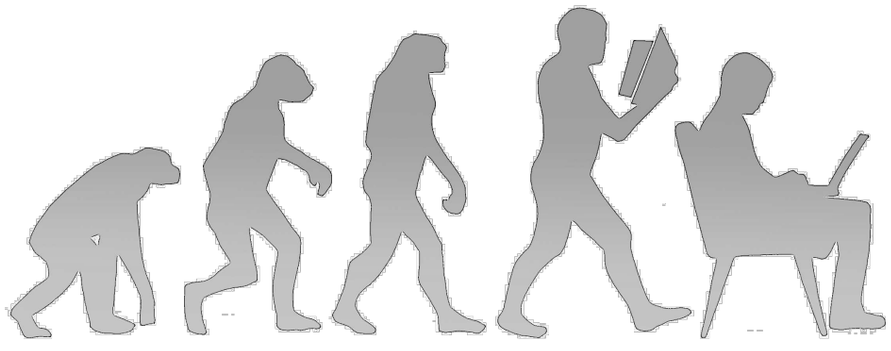
25 мая iForum 2017

О докладчике

- Co-founder & CTO в TRIPMYDREAM
- Работал над проектами **segodnya.ua**, like.ua, viaplay.ua
- Опыт разработки в web с 2007 года
- Увлечения gamedev и музыка

Опыт в программировании

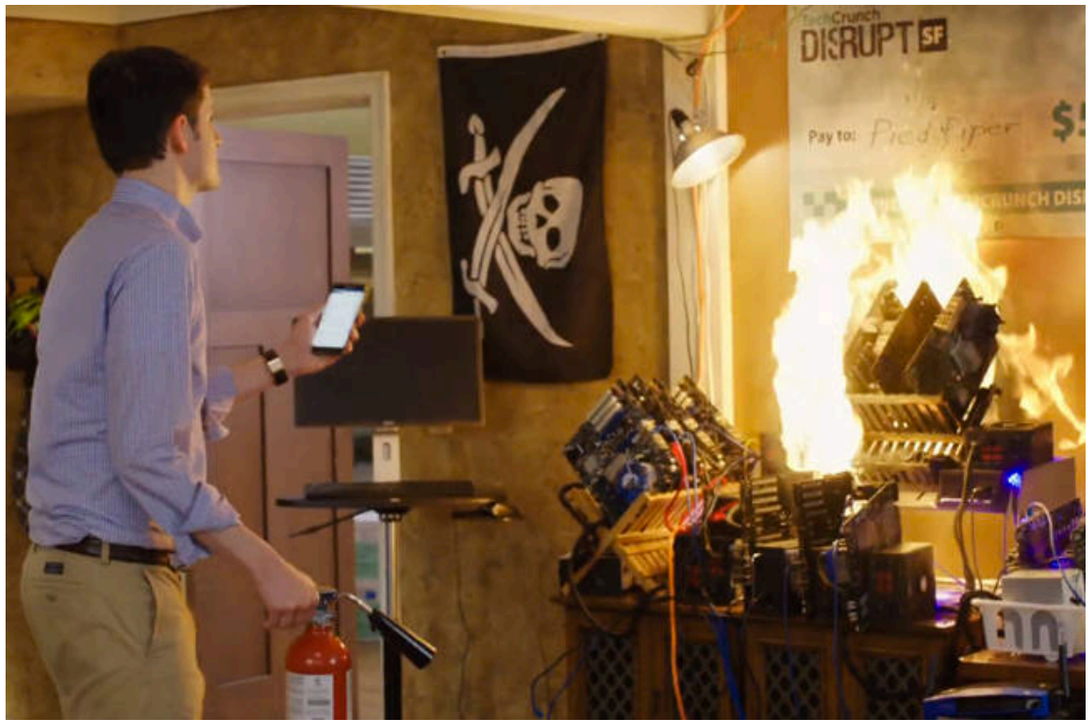
2000	2004	2005	2007	2013	2015 >
Turbo Pascal	**Delphi**	**C++**	**PHP**	**Javascript**	**Hack**
					Javascript



Долгий путь к Hacklang

В 2015 году мы запустили beta-версию сервиса поиска путешествий **TripMyDream**.

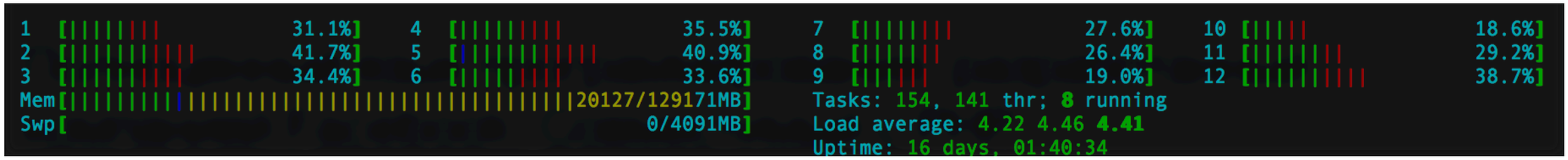
Суть сервиса: поиск выгодных вариантов путешествия отель+перелет под бюджет пользователя.



Неожиданный шквал посетителей ежедневно создавал нагрузку до **500% на 8-ми серверах**.

Задача одного поиска

- Параллельная обработка около 200 внешних API запросов
- Анализ и обработка внешних API ответов, совмещенная с анализом
  ~ 1bn записей собственных данных
- Время ожидания ответа от внешних API до 40 секунд
- Это все работало на PHP



Цель

Максимальное время ожидание первых результатов для пользователя
не более чем время ответа самой быстрой API цепочки

Почему не справился PHP

1. Отсутсвие нативной асинхронности

pthreads, cli, fpm - использования любого из этих методов искусственной асинхронизации PHP съедало тонны серверных ресурсов

2. Сумасшедшее потребление памяти

отсутствие строгой типизации и чрезмерное использование памяти, в совокупности со слабой очисткой памяти, делало невозможным высоконагруженную работу с коллекциями больших размеров

3. Деградация производительности на пиках

с увеличением RPS сильно проседает время ответа. искусственная асинхронность забивает ресурсы сервера и съедает очень много kernel time

Поиск решения

- В стартапе каждый день на счету. У вас нет возможности просто взять и потратить несколько месяцев на смену и изучение нового технологического стэка
- PHP очень популярный язык разработки. На рынке множество толковых специалистов с адекватной заработной платой

Что искали?

1. Строгая типизация и аккуратная работа с памятью
2. Асинхронность
3. Быстрая адаптация специалистов и быстрая миграция

Что такое Hacklang?

Полноценный язык программирования, разработанный компанией Facebook. Работает на виртуальной машине HHVM.

Совместимый с PHP.

Цели Hack:

1. Очень быстрая разработка без багов :)
2. Быстрый runtime
3. Масштабируемость
4. Приятный опыт в написании кода

Типизация

- **"Сильная типизация"** по умолчанию
- Наличие смешанных и нулевых типов
- Акцент на разработке сильно типизированных масштабируемых систем **(type-sensetive)**

- **"Слабая типизация"** по умолчанию
- Отсутствие смешанных и нулевых типов
- Акцент на быстрой слабо типизированной разработке **(type-agnostic)**

Примитивные типы

bool, int, float, string, array, resource, **void**

boolean, integer, float, string, array, resource

Другие типы

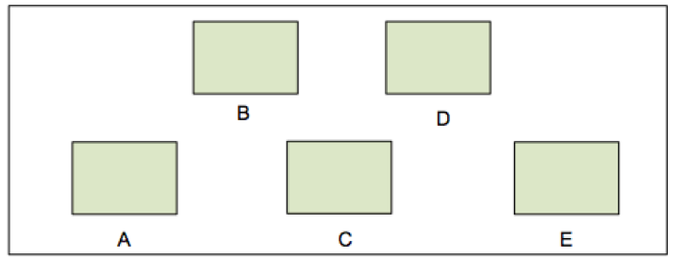
mixed, nullable, arraykey, num, noreturn

Коллекции

У Hack есть нативные коллекции: **Vector**, **Map**, **Set** и **Pair**. Для каждого коллекции можно задавать типы ключей и значений

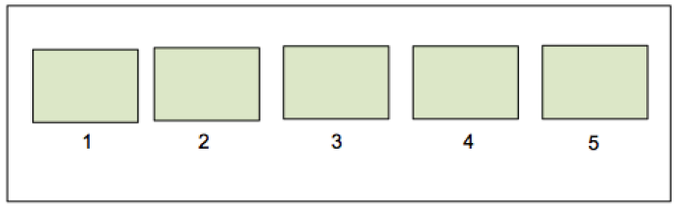
Map **ключ-значение**

Map<string, mixed>



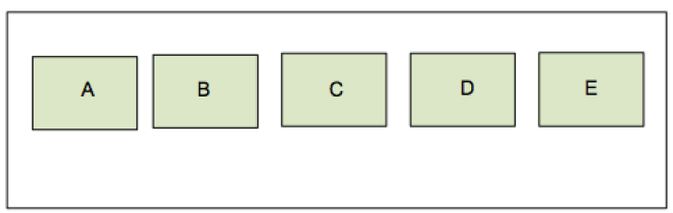
Set **уникальные значения**

Set<string>



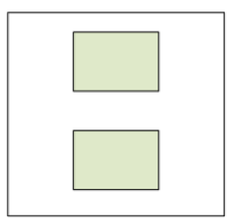
Vector **стэк**

Vector<int>



Pair **два значения**

Pair<int,float>



Другие возможности

Generics: Классы, интерфейсы и трэйты, а также их методы могут быть параметризированы.

```
<?hh

class Box<T> {
  public T $value;
  public function __construct(T $v) {
    $this->value = $v;
  }
}
```

Shape: Специальный тип, объединяющий 0 и более полей как единое целое, благодаря чему отслеживаются с помощью тайпчекера (в отличии от массивов)

```
shape('x' => int, 'y' => int)
```

Enum: группировка констант, с поддержкой int и string значений

Код без багов

Нативный статический тайпчекер позволяет полностью избежать опечаток и багов до runtime. Тайпчекер работает только с кодом на языке Hack.

Partial Mode. [<?hh] Разрешено совмещать Hack код и PHP код.

- Разрешено писать top-level код.
- Доступность суперглобальных переменных

Strict Mode. [<?hh //strict] Режим строгой типизации.

- Весь код должен быть полностью аннотирован типами.
- Запрещено использовать top-level код (кроме use, namespace, require).
- Запрещены указатели.
- Запрещены динамические классы.
- Недоступны суперглобальные переменные

Decl mode. [<?hh //decl] Декларативный режим, не проверяется тайпчекером.

Зачастую используется при конвертации с PHP

Just-in-time compilation

В виртуальную машину HHVM встроен JIT компилятор, транслирующий Hack или PHP код в инструкции машинного кода.

```
$j = 0;
for ($i=0;$i<100000000;$i++) {
  $j++;
}
```

PHP 5.6	**PHP 7.3**	**HHVM 3.19** (no JIT)	**HHVM 3.19** (JIT)
2.60 sec	2.02 sec	6.40 sec	0.22 sec

Runtime custom cache

Memoize позволяет кешировать ответ функции в runtime кэш. Аргументами функции должны быть скалярные типы или коллекции со скалярными типами. Возвращаемый результат может быть любого типа.

```
class A {

  <<__Memoize>>
  public function bar(): int {
    return $this->takes_a_long_time();
  }

  public function takes_a_long_time(): int {
    sleep(3); // искусственный слип
    return 5;
  }
}
```

Время ответа функции **bar** (первый вызов):
3.008~ сек

Время ответа функции **bar** (второй вызов):
0.002~ сек

Свойства класса

Объявление свойств класса можно реализовать с помощью аргументов в конструкторе, таким образом избегая избыточного кода.

Стандартный подход:

```
<?hh

class User {
  protected int $id;
  protected string $name;
  public function __construct(int $id, string $name) {
    $this->id = $id;
    $this->name = $name;
  }
}
```

Используя Constructor parameter promotion

```
<?hh

class User {
  public function __construct(protected int $id, protected string $name) {}
}
```

Операторы

Lambda `==>`

Оператор для лямбда функций, позволяющий использовать текущий scope без использования use.

PHP:

```php
<?php
$user = 'Joel';
$greeting = function() use ($user) {
  return 'Hello '.$user;
};
echo $greeting();
```

Hack:

```hack
<?hh
$user = 'Joel';
$greeting = () ==> 'Hello '.$user;
echo $greeting();
```

Операторы

Null-safe operator `?->`

Позволяет избежать дополнительных проверок на null-ность объекта, предоставляя возможность безопасного доступа к методам.

```
<?hh

class Bar {
public function baz(int $x): int {
  return $x + 5;
 }
}

function get_Bar(): ?Bar {
 if (rand(0, 10) < 5) {
  return null;
 }
 return new Bar();
}

function foo(): ?int {
 $b = get_Bar();
 // foobar does not exist; this could raise a fatal if $b is not null
 // This will raise a typechecker error regardless.
 return $b?->foobar(6);
}
```

Операторы

Pipe `|>`

Синтаксический сахар, позволяющий избежать избыточно вложения вызовов функций.

Стандартный подход:

```
<?hh

function pipe_operator_example1($arr): int {
  return count(array_filter(
    array_map($x ==> $x->getNumber(), $arr),
      $x ==> $x % 2 == 0));
}
```

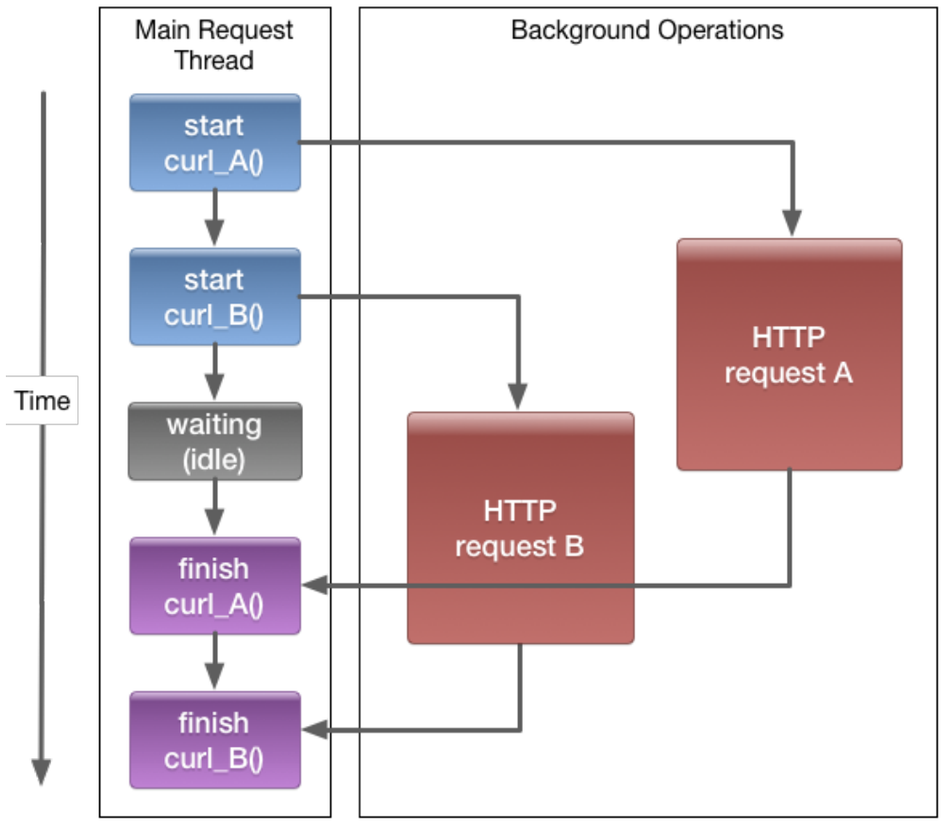
С использованием Pipe:

```
<?hh

function pipe_operator_example1_piped($arr): int {
  return $arr
    |> array_map($x ==> $x->getNumber(), $$)
    |> array_filter($$, $x ==> $x % 2 == 0)
    |> count($$)
}
```

Асинхронность

Hack поддерживает **Async/Await**, реализуя однопоточную многозадачность. Работая в едином потоке Hack позволяет запускать ряд задач одновременно используя планировщик, и переключаться между задачами на ходу



Встроенные асинхронные компоненты (неблокирующие):

- AsyncMysql
- MCRouter (Memcached)
- cURL
- Streams

Асинхронность

Простая задача для Javascript. Сложная для PHP.

Необходимо каждую секунду отправлять параллельно ровно 3 API запроса. При получении ответа необходимо обрабатывать результат и сохранять. Время ответа API от 0.5 до 5 секунд. Все должно работать в едином потоке одного процесса.

Решение на Hack

На минуту вперед создается массив async функций, для каждого запроса, и запускается одновременно:

```
\HH\Asio\join ($asyncTasks);
```

Каждая async функция запускает асинхронный cURL со своим заданным callback:

```
await \HH\Asio\curl_exec($ch);
```

Когда планировщик попадает в async функцию, время которой еще не пришло, ставим её выполнение в конец очереди:

```
await RescheduleWaitHandle::create(RescheduleWaitHandle::QUEUE_DEFAULT, 0);
await \HH\Asio\usleep(100);
```

Repo Authoritative

Стандартно HHVM ведет себя также, как и PHP движок - загружает и компилирует код на лету. Режим Repo Authoritavive позволяет скомпилировать весь код в бинарный репозиторий и существенно ускорить скорость работы на пиковых нагрузках.

1. Отсутствие повторного чтения исходных файлов с диска
2. Отсутствия чтения require файлов
3. Оптимизация работы кода
4. Быстрое переключение между режимами работы для обновления кода

Типы серверов

Есть 2 типа серверов, которые можно использовать в HHVM

FastCGI

Стандарное использование, схожее с использование nginx + php-fastcgi. HHVM запускается независимо от web серверов (apache, nginx, и др.).

Proxygen

Виртуальная машина HHVM имеет встроенный полноценный web server, позволяющий принимать запросы напрямую, исключая промежуточный веб-сервер, например nginx.

Встроенный шаблонизатор

В Hack разработан полноценный шаблонизатор XHP, который предоставляет нативную XML репрезентацию для HTML кода:

Традиционный подход:

```
<?hh
$user_name = 'Fred';
echo "<tt>Hello <strong>$user_name</strong></tt>";
```

С помощью XHP:

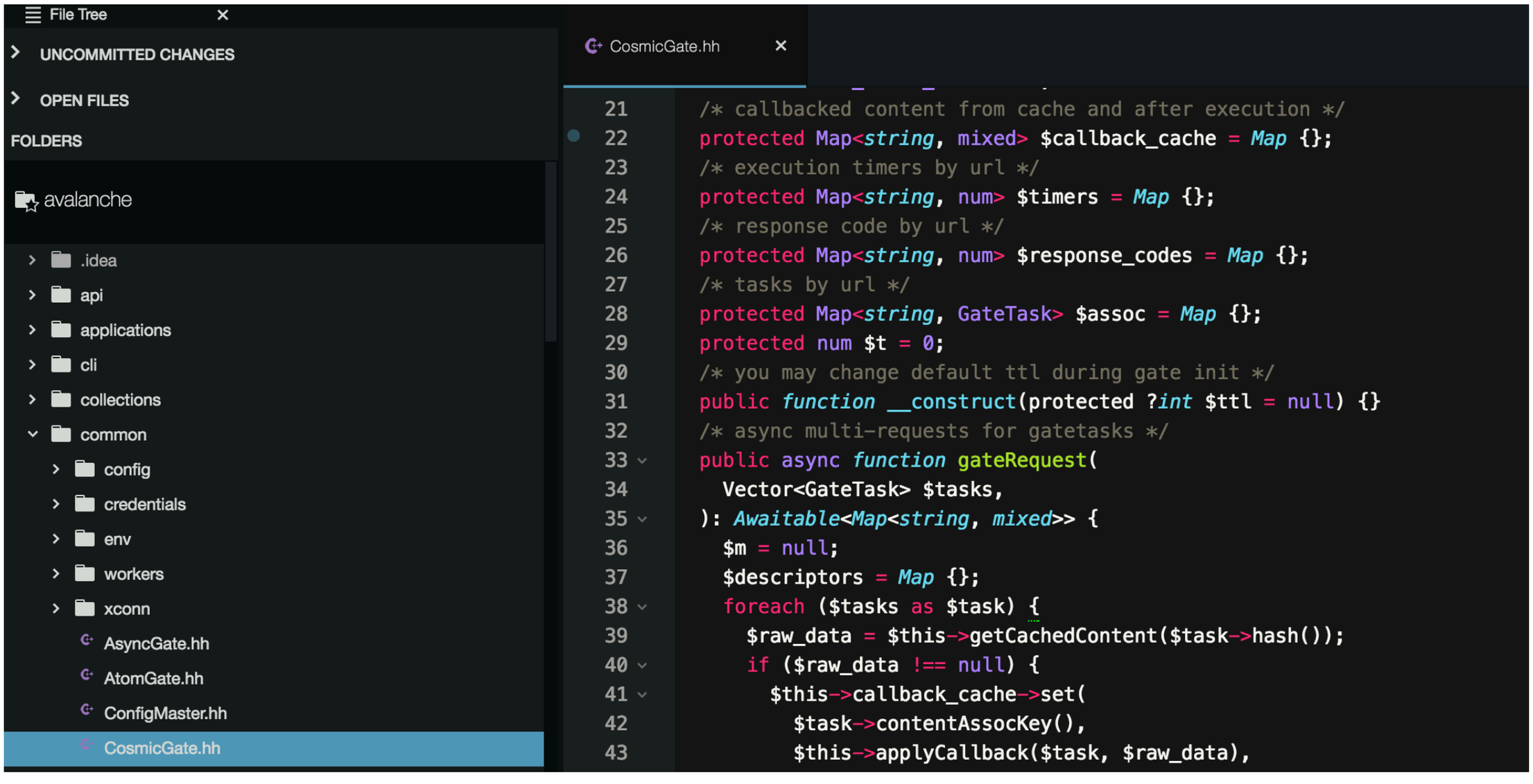
```
<?hh
$user_name = 'Fred';
echo <tt>Hello <strong>{$user_name}</strong></tt>;
```

Положительным момент: html код проверяется тайпчекером, и html с ошибками не позволит скомпилировать код.

Отрицательным момент: html код проверяется тайпчекером, и html с ошибками не позволит скомпилировать код. :)

IDE для разработки

Atom + Nuclide совмещает в все необходимые компоненты для разработки на Hack, включая инлайновый typecheker, autocomplete и go-to definition



Слабые стороны

1. Отсутсвие PECL

Hack поддерживает множество PHP расширений, однако добавление новых не доступно. При этом вы можете скомпилировать hhvm с добавлением собственных компонентов на C++

2. Отсутствие полноценного Framework

Для кого-то слабая сторона, для кого-то не имеющая значения. Однако нужно принять факт того, что на "чистом Hack" прийдется писать с нуля.

Экосистема

Активный **Github.** Разработчики дают обратную связь
http://github.com/facebook/hhvm

281 вопрос на **Stackoverflow**.

Хорошая документация + пересечение с **PHP**
https://docs.hhvm.com/hack/

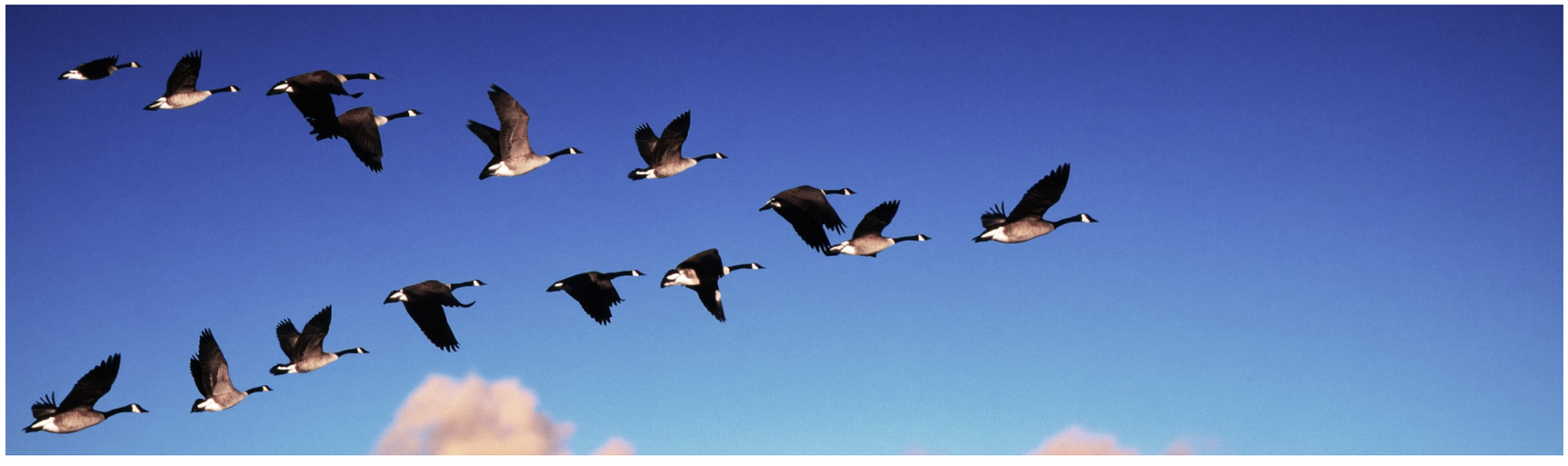
Стабильное развитие

- Полноценный релиз каждые 8 недель
- Ночные сборки
- Разработчики принимают внешние коммиты

Наш опыт миграции

Миграция кода

- Существуют конвертаторы из PHP кода в Hack
- Наш опыт показывает, что лучше писать с нуля



Разработчики

- Через первый месяц Middle PHP Developer свободно пишет на Hack
- Через 2 месяца он проектирует и пишет код полноценно с измененной парадигмой

Установка

Ubuntu, Debian

- Установка из готовых пакетов
- Возможность скомпилировать исходники с GIT

OSX

- Установка через brew
- Возможность скомпилировать исходники с GIT

Другие Linux системы:

- Возможность скомпилировать исходники с GIT

Docker:

- Образы в Docker Hub, которые можно использовать для контейнеризации hhvm

Windows (пока не доступно):

- В разработке портирование с помощью MSVC

Благодарю!

Напишите мне:

taras@tripmydream.com

facebook.com/**r4mpge**

linkedin.com/in/**taraspolishchuk**