Laravel Conf 2020 Taiwan

# 大象也能飛翔！

## 聊 Laravel 效能調校

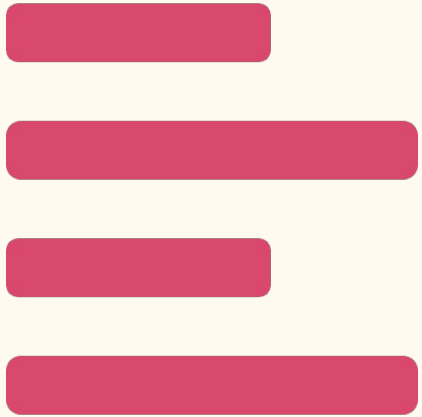

- 2014 年第一次接觸 Laravel
- 2019 年 LaravelConf Taiwan
  - 星移電掣般的 Trace Code！讓你如光速一般的追蹤程式碼

# 目次

- 實際測量效能
- 測量結果
  - 不同版本的效能差異
- 撰寫注意事項
- 擠出最後的效能
  - Laravel 特殊寫法

Laravel Conf 2020 Taiwan

# 實際測量效能

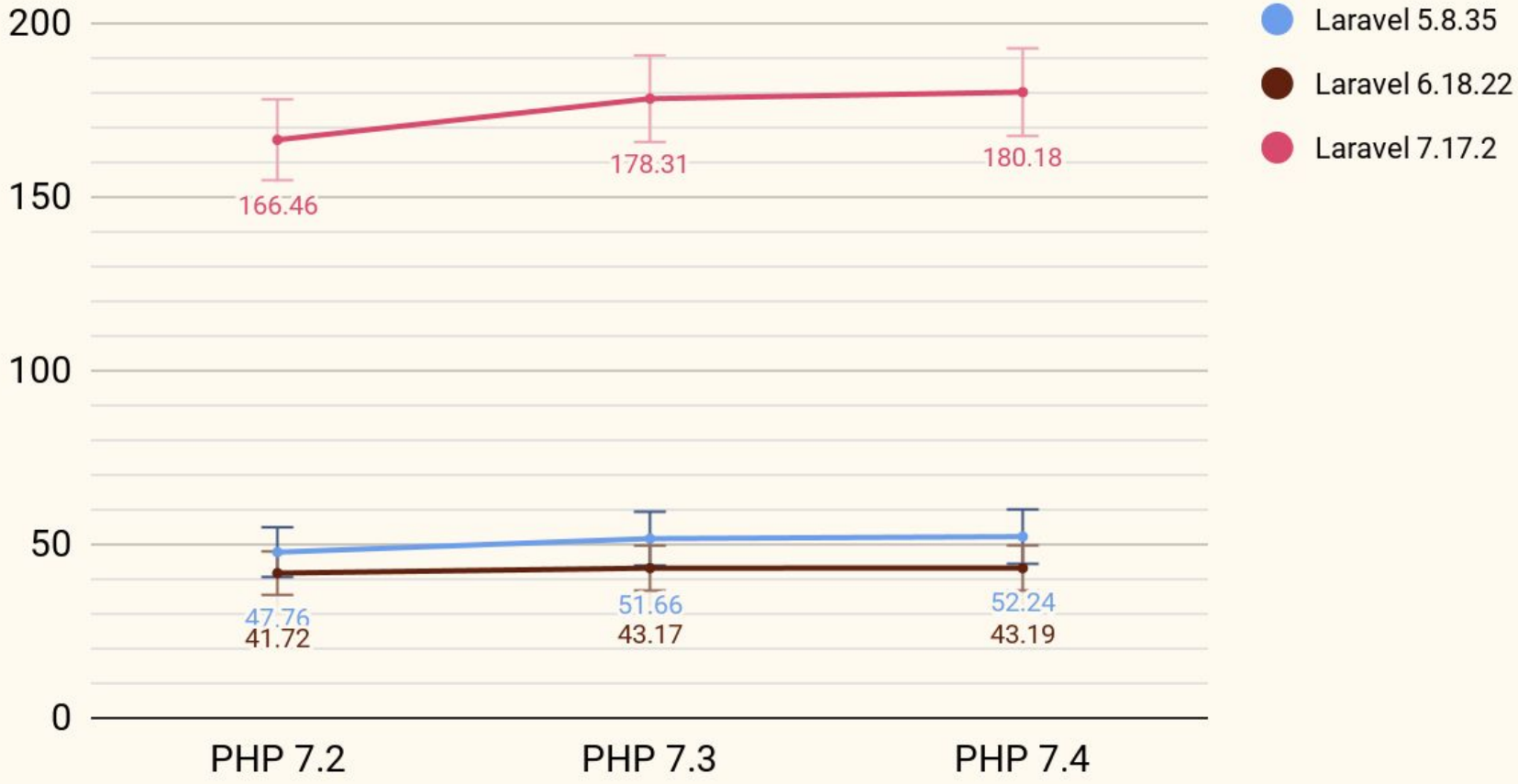


只有經過測量之後 才能精確的討論效能

# 此次測量方法

- AWS EC2 t2.medium
- docker(laradock)
  - nginx
  - Zend OPcache
- Apache Benchmark
  - ab -c 10 -n 1000 http://127.0.0.1/

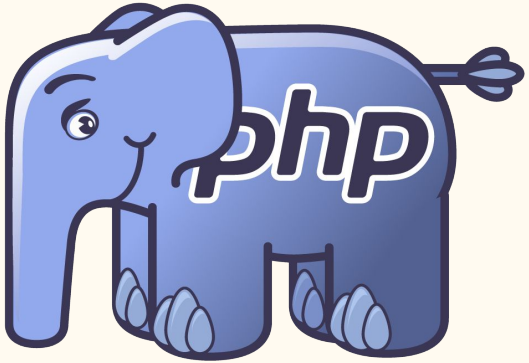
# 不同版本 Request Per Second

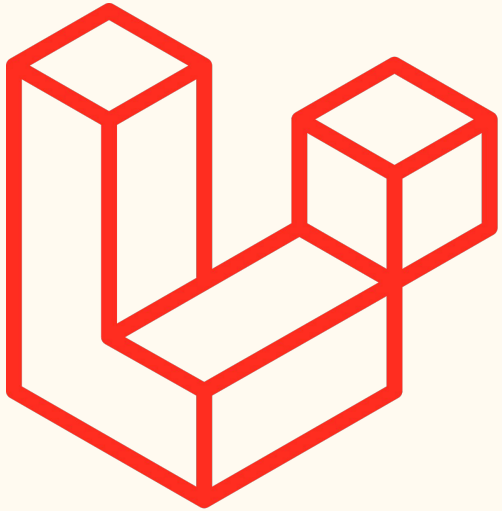


Laravel Conf 2020 Taiwan

# 撰寫注意事項

先優化資料庫和靜態資源



理解後做出假設 再測量結果

# 此次測量方法

- AWS EC2 t2.medium
- PHP 7.4.5
- Zend OPcache

# while

# while

```php
$array = array_fill(0, 1000000, '');
$t = microtime(true);
while(list($key) = each($array)) {
    $array[$key] .= "a";
}
echo(microtime(true) - $t);
```

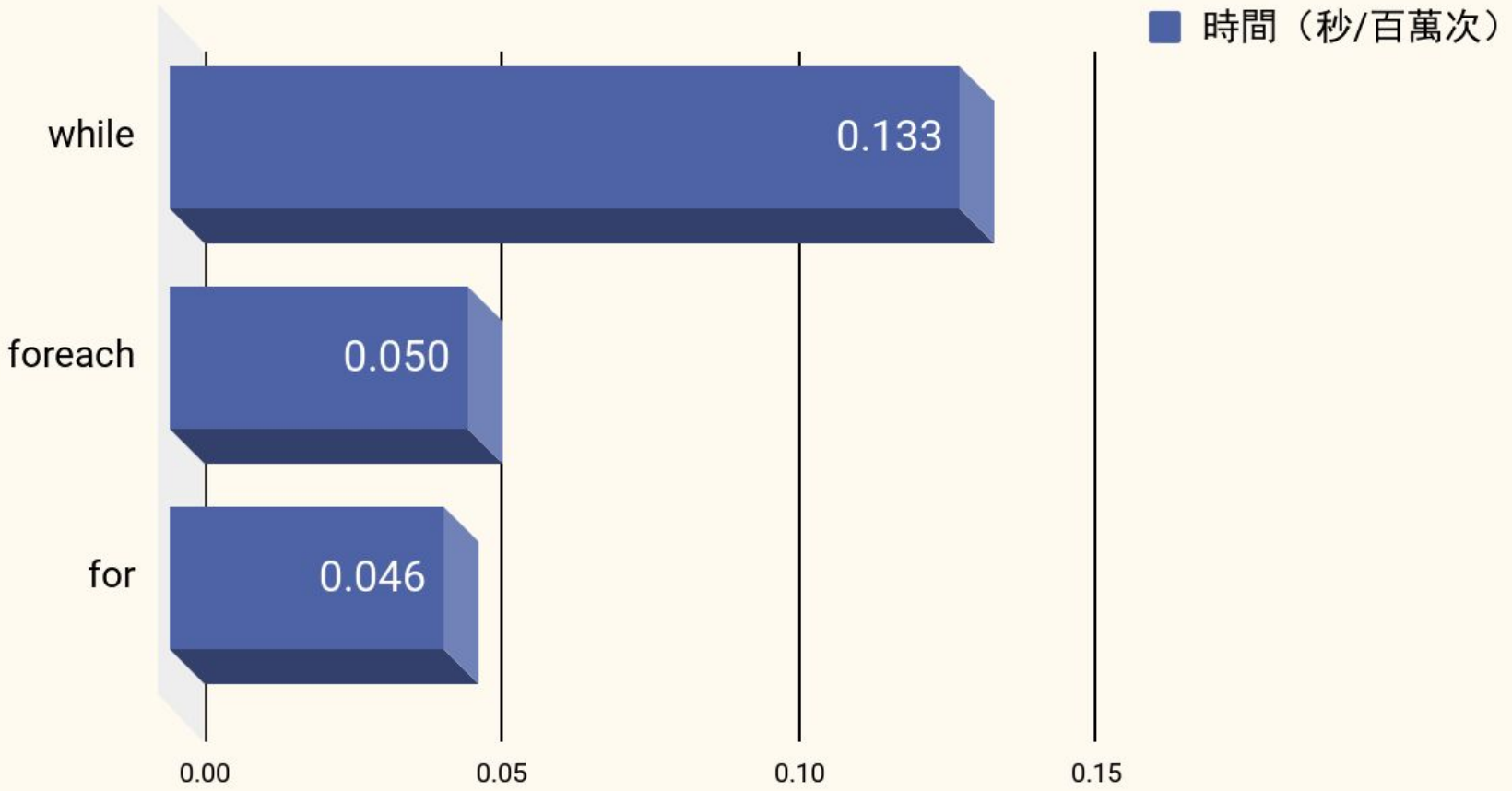
# foreach

```php
$array = array_fill(0, 1000000, '');
$t = microtime(true);
foreach ($array as $key => $val) {
    $array[$key] .= "a";
}
echo(microtime(true) - $t);
```

# for

```php
$array = array_fill(0, 1000000, '');
$t = microtime(true);
$key = array_keys($array);
$size = sizeOf($key);
for ($i = 0; $i < $size; $i++) {
    $array[$key[$i]] .= "a";
}
echo(microtime(true) - $t);
```

# 時間比較



# 補充

each() 這個方法已經被標記為棄用（deprecated）
所以第一種寫法除了效率差
也比較不穩定

# PHP 特性

PHP 原生函式的 overhead 相對大
減少原生函式呼叫，可能會提升效能

# array_push()

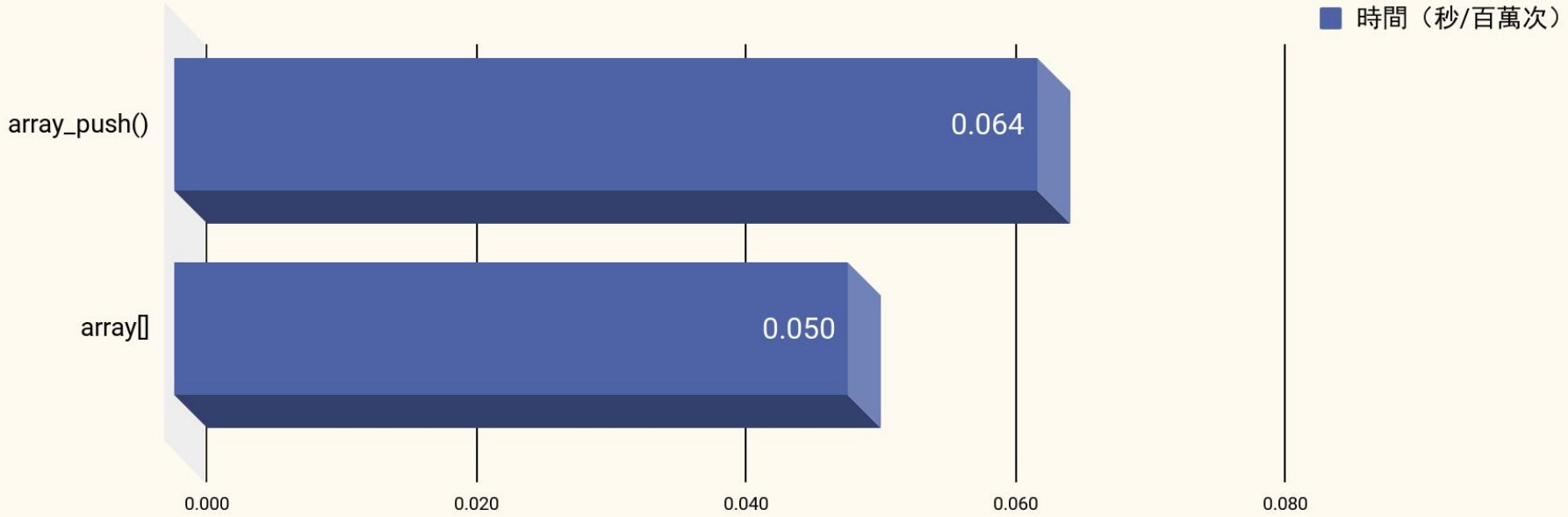
# array_push()

```php
$a = [];
$t = microtime(true);
for ($i = 0; $i < 1000000; $i++) {
    array_push($a, 'a');
}
echo(microtime(true) - $t);
```

# $array[]

```php
$a = [];
$t = microtime(true);
for ($i = 0; $i < 1000000; $i++) {
    $a[] = 'a';
}
echo(microtime(true) - $t);
```

# 時間減少 21.8%



# array_key_exists()

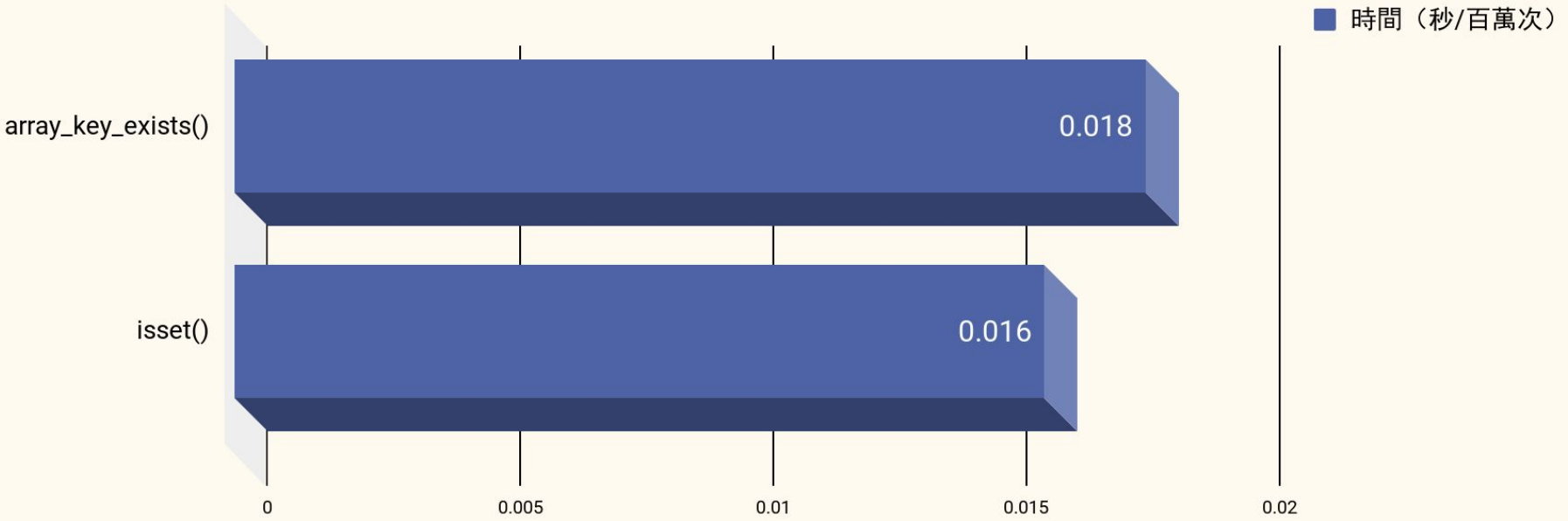
# array_key_exists()

```php
$a = [];
$t = microtime(true);
for ($i = 0; $i < 1000000; $i++) {
    array_key_exists('a', $a);
}
echo(microtime(true) - $t);
```

# isset()

```php
$a = [];
$t = microtime(true);
for ($i = 0; $i < 1000000; $i++) {
    isset($a['a']);
}
echo(microtime(true) - $t);
```

# 時間減少 11.1%



# 還有很多地方

- array()
  - 改成 []
- call_user_func($func)
  - 改成 $func()
- isset($string[10])
  - 改成 strlen()

.....

可以實測看看效率差異

用 20% 的心力提高 80% 的效能

# 中場總結

# 中場總結

- 實際測量，才能討論利弊
- 透過理解做出假設，再測量結果
- 針對版本優化
  - 選 PHP 7.4 和 Laravel 7
- 針對 PHP 語法優化
  - 用 foreach 或 for
  - 用 `$a[]`
  - 用 `isset()`


如果你需要剩下的 20%

Laravel Conf 2020 Taiwan

# 擠出最後的效能

## Laravel 專屬調整

# 此次測量方法

- AWS EC2 t2.medium
- PHP 7.4
- Laravel 7.17.2
- docker(laradock)
  - nginx
  - Zend OPcache
- Apache Benchmark
  - ab -c 10 -n 1000 http://127.0.0.1/

# Laravel cache

```
$ php artisan optimize

// php artisan config:cache
// php artisan route:cache

$ php artisan view:cache
```

# Laravel cache clear

```
$ php artisan optimize:clear

// php artisan view:clear
// php artisan cache:clear
// php artisan route:clear
// php artisan config:clear
// php artisan clear-compiled
```

# 移除多餘 ServiceProvider

config/app.php

```
// App\Providers\AuthServiceProvider::class,
// App\Providers\EventServiceProvider::class,
```

# 移除多餘 Middleware

app/Http/kernel.php

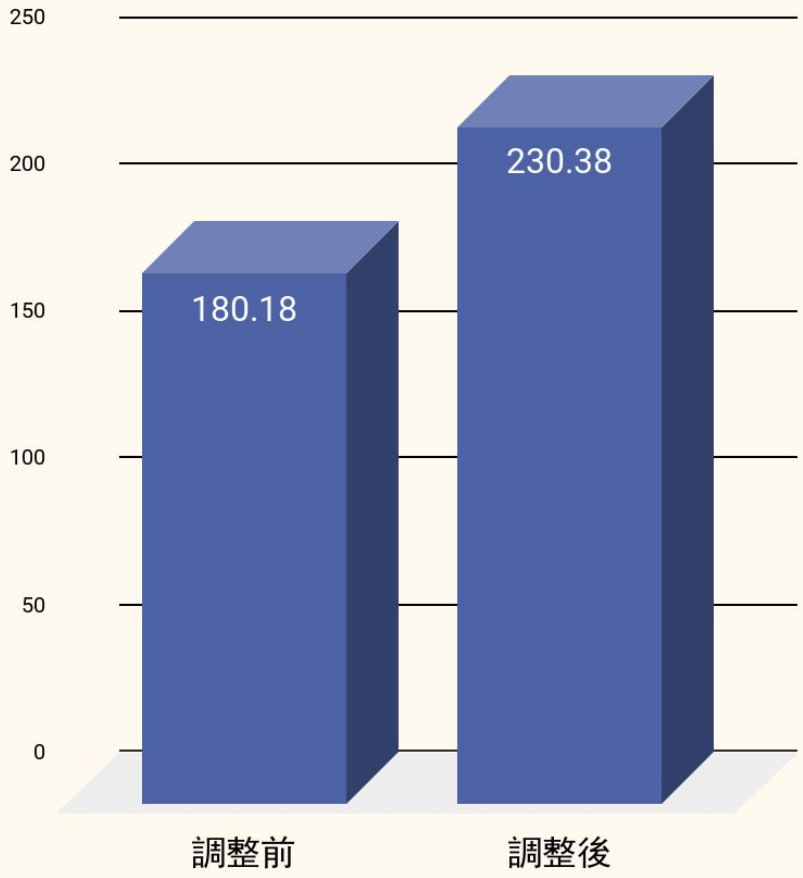
```
// \App\Http\Middleware\EncryptCookies::class,
// \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
// \Illuminate\Session\Middleware\StartSession::class,
// \Illuminate\View\Middleware\ShareErrorsFromSession::class,
// \App\Http\Middleware\VerifyCsrfToken::class,
```

# 移除多餘 Middleware

app/Http/kernel.php

```
// 'auth' => \App\Http\Middleware\Authenticate::class,
// 'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
// 'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
// 'can' => \Illuminate\Auth\Middleware\Authorize::class,
// 'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
// 'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
// 'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,
// 'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
// 'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
```

# Req/Sec 提升 27.8%



# 還有很多地方

- 善用 with() 改善 sql query（Eager Loading）
- LazyCollection

……

實測看看效率差異

# 總結

# 總結

- 實際測量，才能討論利弊
- 透過理解做出假設，再測量結果
- 針對版本優化
  - 選 PHP 7.4 和 Laravel 7
- 針對 PHP 語法優化
  - 用 `array_keys()` `sizeOf()` 避免迴圈每次呼叫
  - 用 `$a[]`
  - 用 `isset()`
- 針對 Laravel 優化
  - php artisan optimize
  - 移除不必要 Service Provider
  - 移除不必要 Middleware

# 提問時間