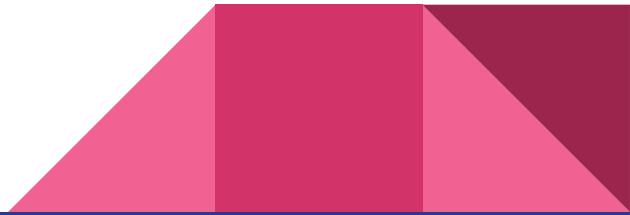
PHP preloading

2020-02-07　えだ

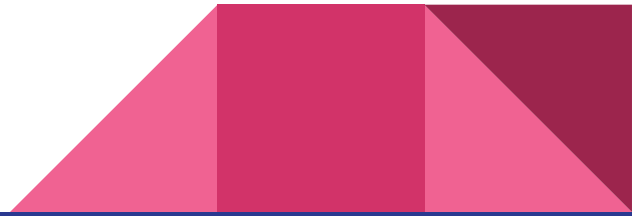
そもそもPHPは

- PHPはスクリプト言語
- インタプリタ方式（実行時コンパイル方式ではない）
- リクエスト毎に逐次解釈しながら実行する
  - ロード（読み込み）
  - パース（構文解析）
  - コンパイル（実行命令への変換）
- 理論上はコンパイル方式のほうが早い



高速化の手法

- OPcache
- preload



OPcache

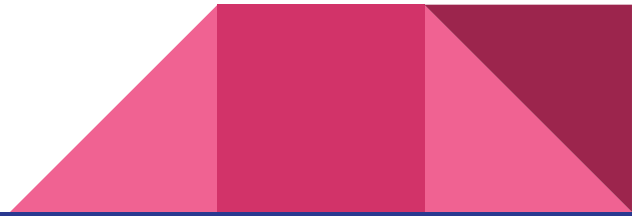
- コンパイル済みのバイトコードを共有メモリに保存
- リクエスト毎のスクリプトリード＆パースを省く
- PHP拡張モジュール
- PHP 5.5.0～バンドル（5.2はPECLで）

OPcacheの設定

opcache.enable	1	キャッシュを有効にする
opcache.enable_cli	0	CLI版のキャッシュを有効にする
opcache.memory_consumption	64	使用される共有メモリのサイズ
opcache.interned_strings_buffer	4	internされた文字列を格納されるために使用されるメモリ量
opcache.max_accelerated_files	2000	スクリプトの最大数最小値は200
opcache.revalidate_freq	2	更新時にタイムスタンプをチェックする頻度。opcache.validate_timestampsがデフォルトで有効なのでチェックされる
opcache.fast_shutdown	0	それぞれに割り当てられたブロックを開放しない高速シャットダウンシーケンスが使用される

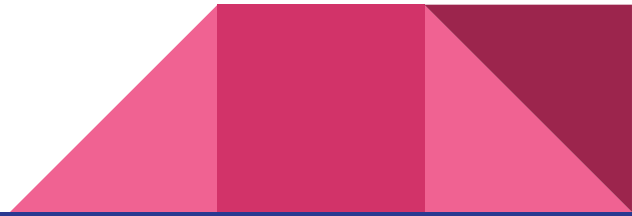
OPcache検証1

- 普通に書いたもの
- Hello World を 1000リクエストしてRequests per secondで比較
- 2982.86 → **4370.92**



OPcache検証2

- Laravel 6.9.x
- Hello World を 1000リクエストしてRequests per secondで比較
- 8.56→ **18.46**



OPcacheのダメなところ

- ファイル単位でしかキャッシュできない
- 別ファイル参照は毎回再リンクしている（らしい）

preload

- サーバ起動時にファイルが読み込まれる
- 変更にはサーバ再起動が必要
- PHP 7.4.0～

preloadの設定

php.ini

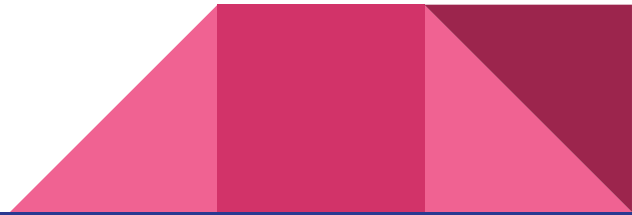
```
1 | opcache.preload="hoge/huga/preload.php"
```

hoge/huga/preload.php

```
1 | opcache_compile_file('apppath/hogehoge.php');
2 | opcache_compile_file('apppath/hugahuga.php');
```

OPcache検証

- Laravel 6.9.x
- Hello World を 1000リクエストしてRequests per secondで比較
- 8.56→ **40.37**



考察

- 当社環境では使うケースは無さそう
- 最終手段として覚えておくレベルか
- Requests per secondの信憑性は大丈夫か不安

終わり