# Fine tuning PHP


**24. 11. 2021**

**18:00**

**U Salzmannů, salonek 1. partro**

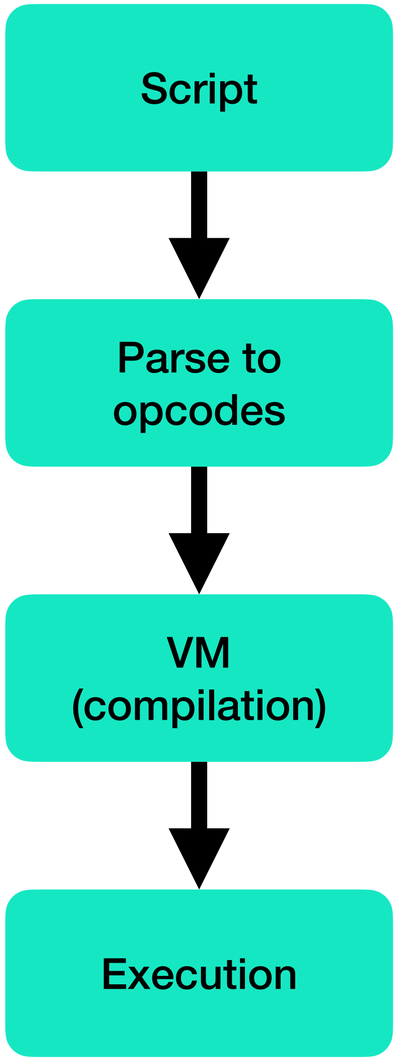
PeoplePath

# PHP Request Lifecycle Evolution

# Before PHP 5.5

## dark ages

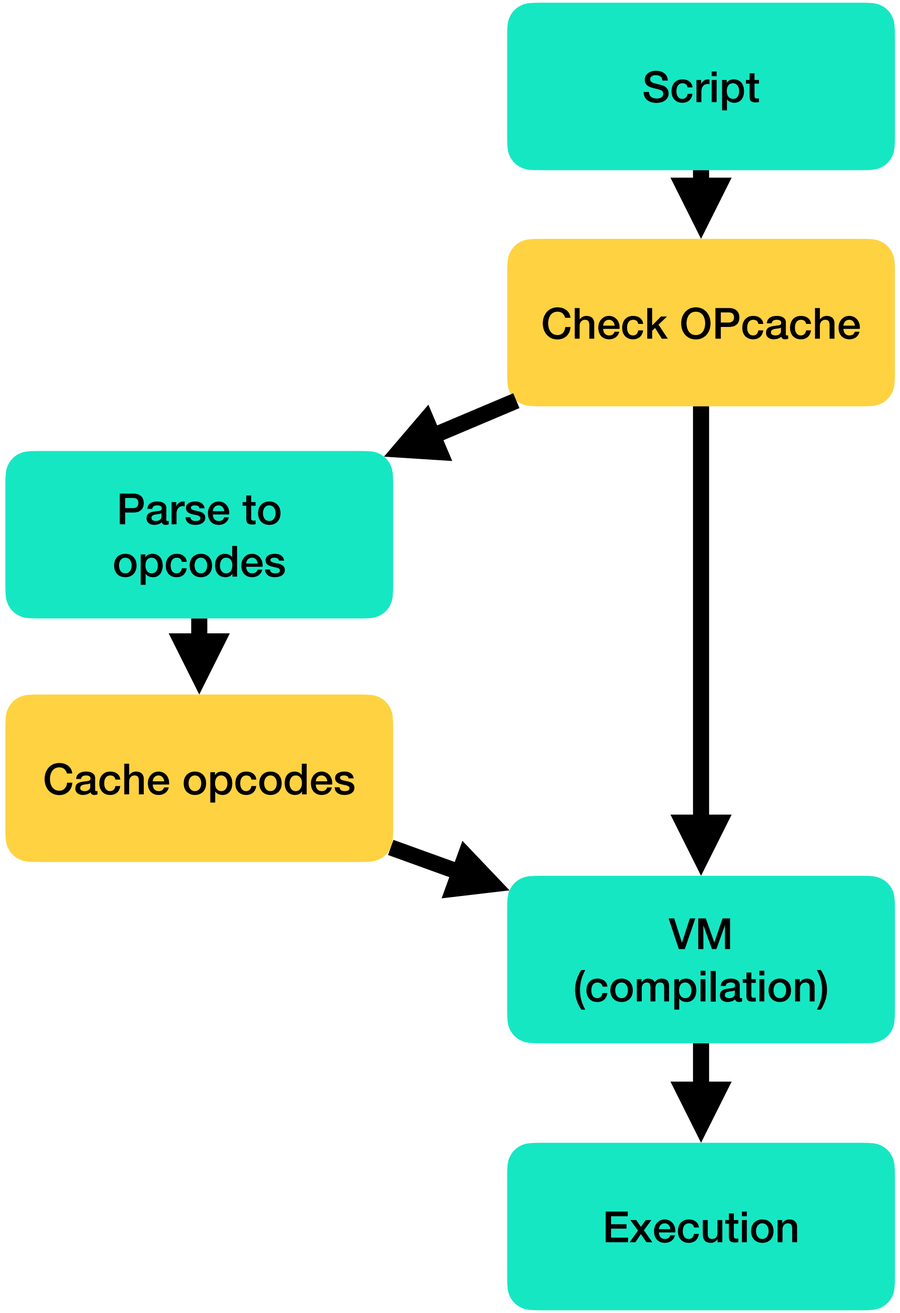
1. read file from disk
2. parse to bytecode (opcodes)
3. compile to machine code
4. execute



# PHP 5.5

## raise of the OPcache

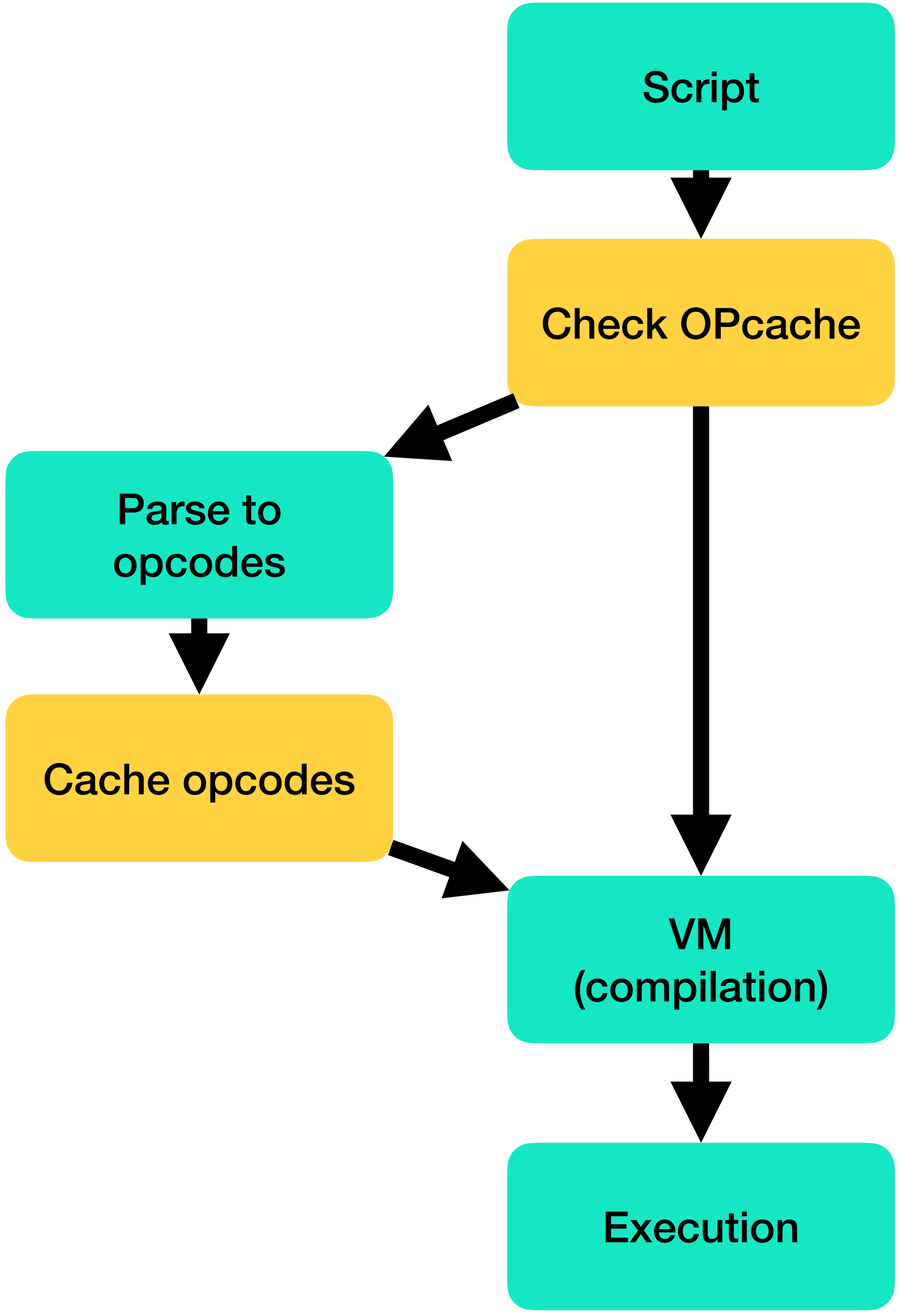
- cache parsed code in the first request
- fetch cached from previous requests
- huge performance gains in big PHP applications



# PHP 7.0

## invisible revolution

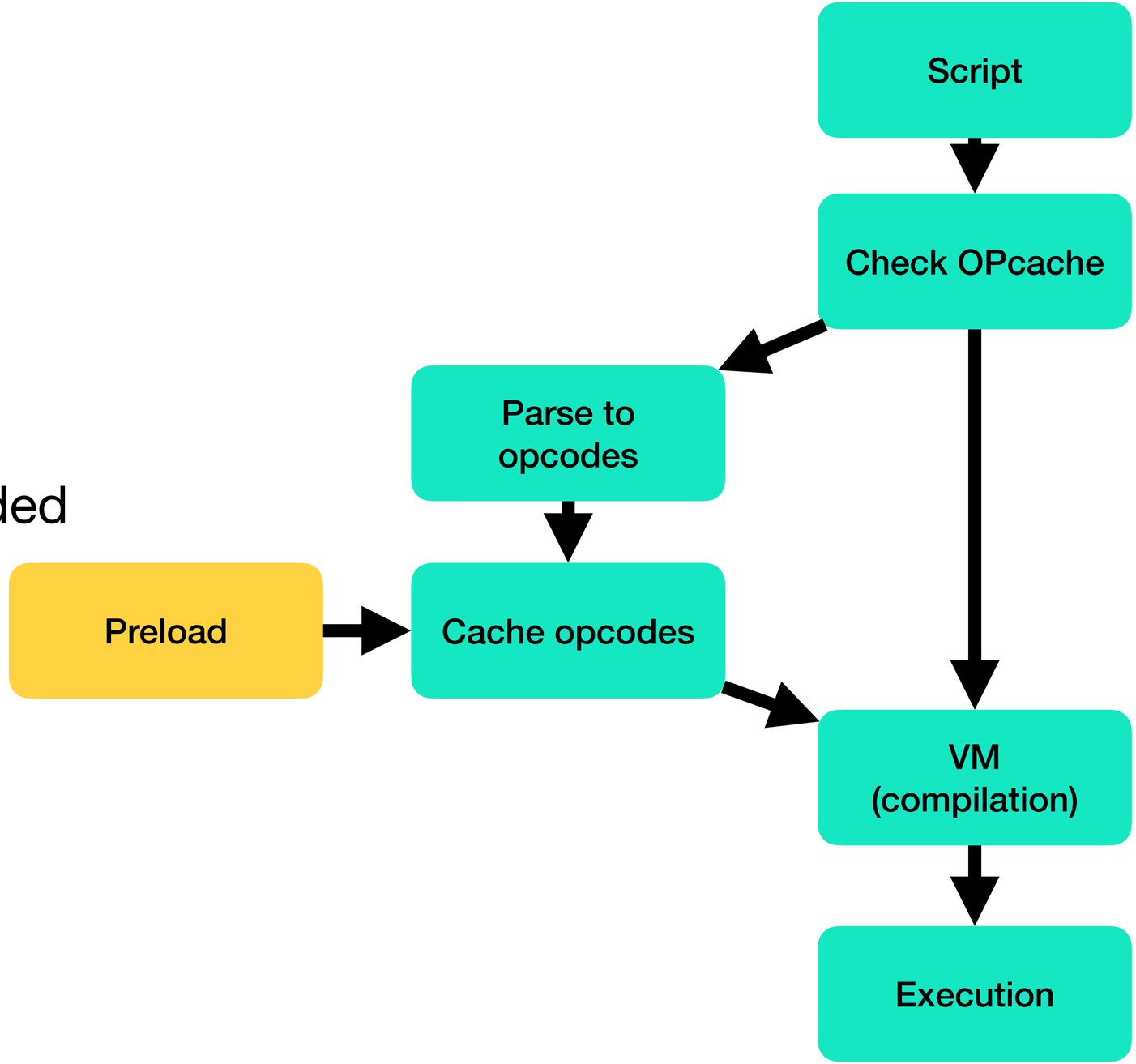
- parser refactoring
- AST optimizations



# PHP 7.4

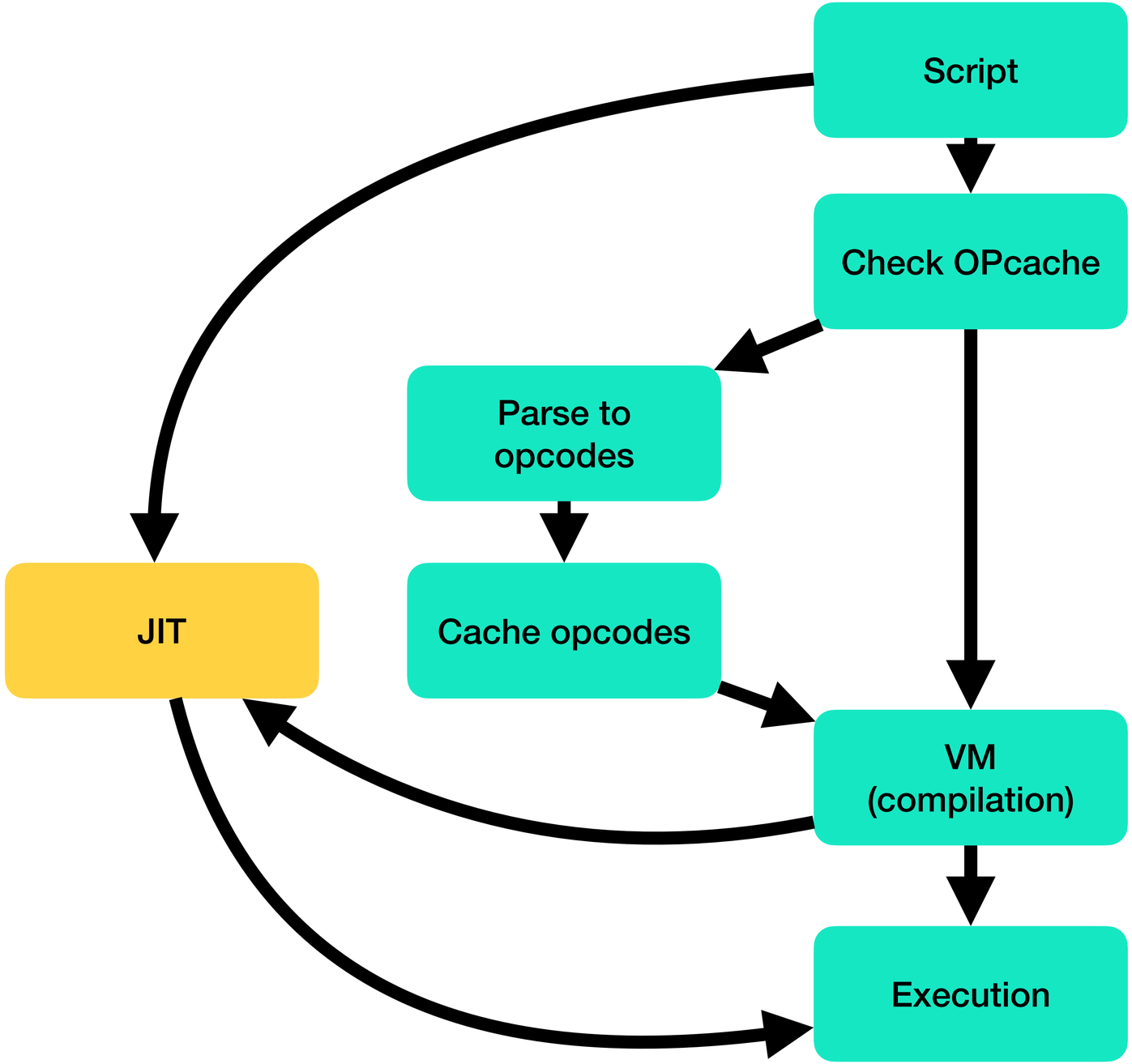
## useful hack

- preload can be used for populating opcache
- autoloading can be avoided completely



# PHP 7.4
## with JIT?

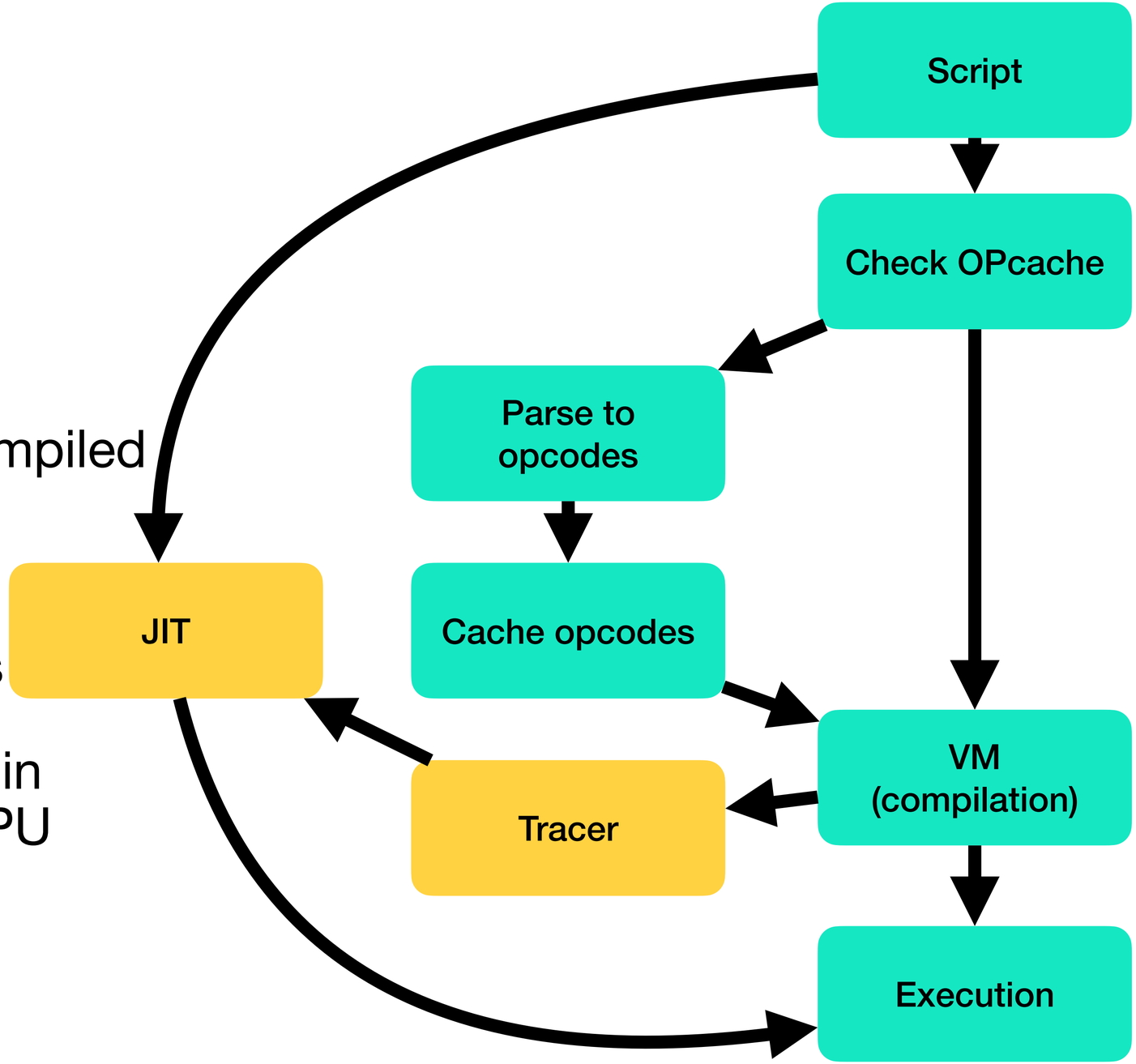
- function based JIT



# PHP 8.0

## JIT

- tracer JIT
- performance close to compiled languages like C
- small performance gains in common PHP projects
- huge performance gains in applications heavy on CPU and memory



**Sebastian Bergmann**
@s_bergmann

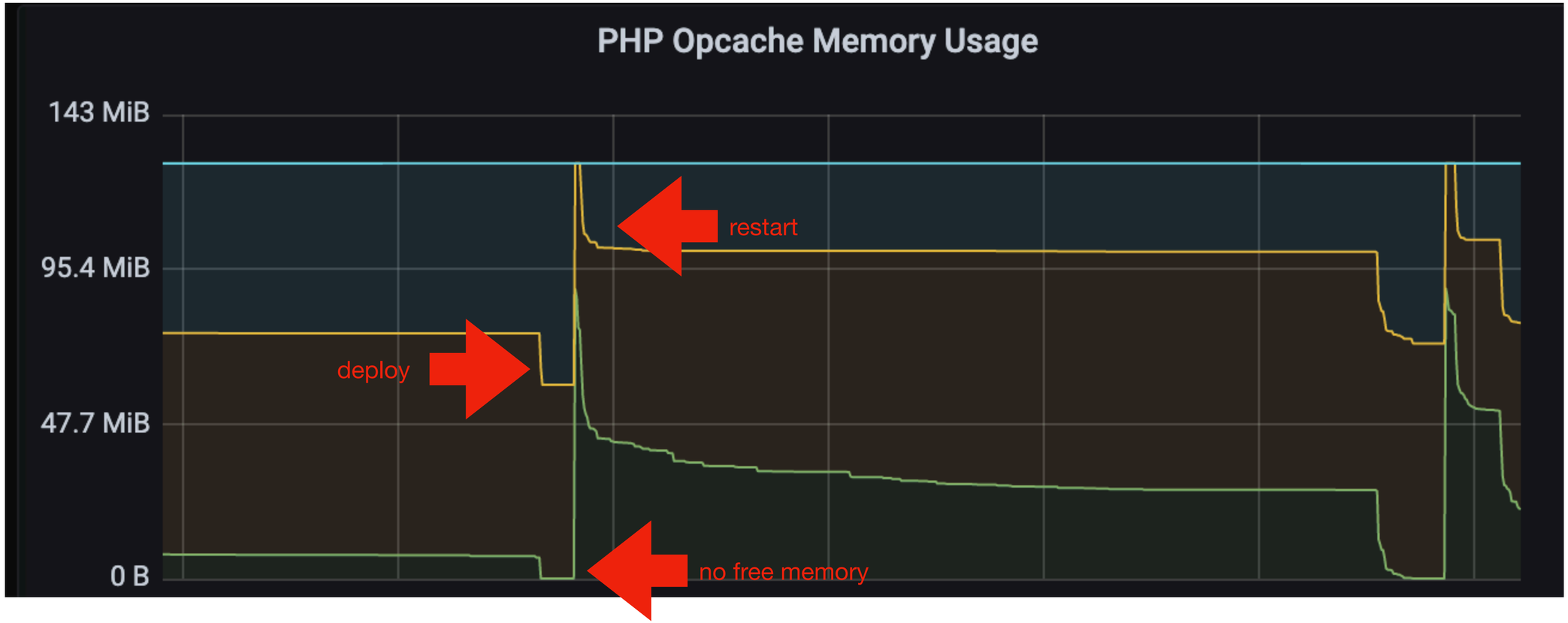
Before somebody asks: yes, PHP's Just-in-Time compiler makes a difference.

```
raytracer  master ✔  ./tools/phpunit --filter test_chapter_6 --repeat 10
PHPUnit 9.5.6 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.0.7
Configuration: /usr/local/src/raytracer/phpunit.xml

..........                                                        10 / 10 (100%)

Time: 01:04.739, Memory: 14.00 MB

OK (10 tests, 20 assertions)
raytracer  master ✔  php -d opcache.enable=1 -d opcache.enable_cli=1 -d opcache.optimization_level=-1 -d opcache.jit=
1255 -d opcache.jit_buffer_size=32M ./tools/phpunit --filter test_chapter_6 --repeat 10
PHPUnit 9.5.6 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.0.7
Configuration: /usr/local/src/raytracer/phpunit.xml

..........                                                        10 / 10 (100%)

Time: 00:42.184, Memory: 14.00 MB

OK (10 tests, 20 assertions)
```

source: https://twitter.com/s_bergmann/status/1408402073901842440

# OPcache lifecycle

## cache, invalidate, restart



opcache.enable_cli

*useful only for debug*

# opcache.memory_consumption

*memory limit for opcache*

NOTE: opcache is not classic LRU cache, it cannot garbage collect old entries. Opcache can be freed only by restart

opcache.max_accelerated_files

*at least double of all files in the app*

# opcache.max_wasted_percentage

# opcache.validate_timestamps

# *disable it to avoid any access on filesystem*

NOTE: it’s good strategy to disable it but any file change will require either server restart or call of opcache_invalidate()

opcache.revalidate_freq

*useful if opcache.validate_timestamps is true*

opcache.save_comments

*enable it for annotation based code (eg. Doctrine)*

opcache.file_cache

*may help with often restarts*

# opcache.preload

*initialization of PHP engine and OPcache specially*

opcache.jit_buffer_size

*enables JIT compilation*

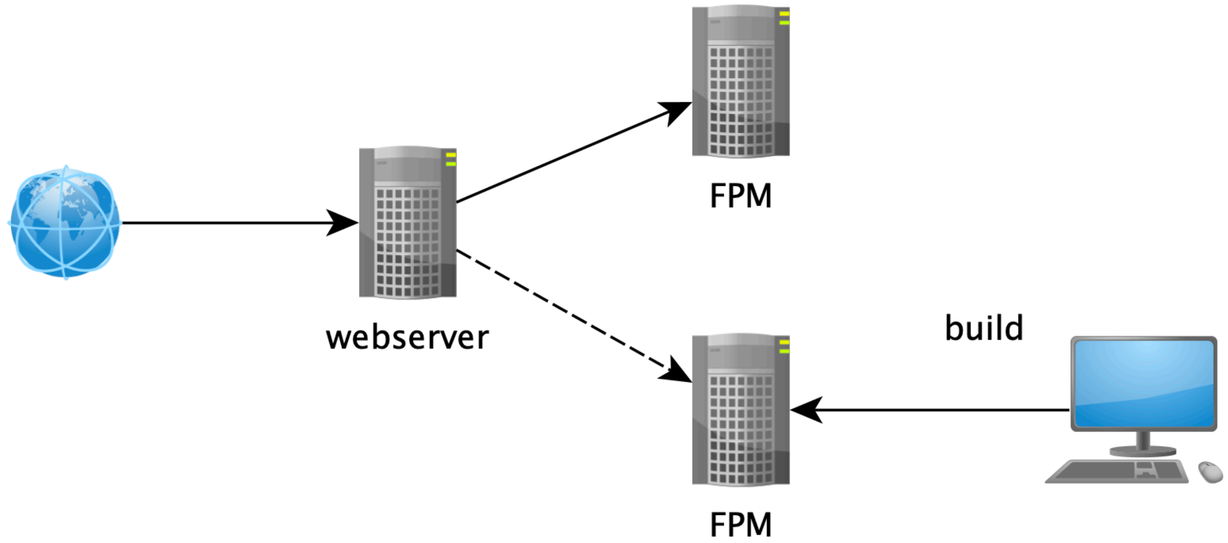
# opcache_get_status()

```
array (size=9)
  'opcache_enabled' => boolean true
  'cache_full' => boolean false
  'restart_pending' => boolean false
  'restart_in_progress' => boolean false
  'memory_usage' =>
    array (size=4)
      'used_memory' => int 24559880
      'free_memory' => int 109650360
      'wasted_memory' => int 7488
      'current_wasted_percentage' => float 0.0055789947509766
  'interned_strings_usage' =>
    array (size=4)
      'buffer_size' => int 6291008
      'used_memory' => int 2444672
      'free_memory' => int 3846336
      'number_of_strings' => int 37445
  'opcache_statistics' =>
    array (size=13)
      'num_cached_scripts' => int 1298
      'num_cached_keys' => int 2584
      'max_cached_keys' => int 16229
      'hits' => int 1409
      'start_time' => int 1637766159
      'last_restart_time' => int 0
      'oom_restarts' => int 0
      'hash_restarts' => int 0
      'manual_restarts' => int 0
      'misses' => int 1302
      'blacklist_misses' => int 0
      'blacklist_miss_ratio' => float 0
      'opcache_hit_rate' => float 51.973441534489
  'scripts' =>
    array (size=1298)
      '/Users/esler/iw/github.com/peoplepath/fine-tuning-php/var/www/html/vendor/psr/http-factory/src/StreamFactoryInterface.php' =>
        array (size=6)
          'full_path' => string '/Users/esler/iw/github.com/peoplepath/fine-tuning-php/var/www/html/vendor/psr/http-factory/src/Str…
          'hits' => int 1
          'memory_consumption' => int 4184
          'last_used' => string 'Wed Nov 24 16:02:43 2021' (length=24)
          'last_used_timestamp' => int 1637766163
          'timestamp' => int 1556627896
```

# Atomic deploy

## seamlessly without ANY side effects

1. build source code
2. fire up FPM server
3. wait for green
4. switch traffic to webserver to the new FPM server
5. gracefully stop old FPM server




NOTE: practical demonstration of this approach can be topic of the next PeoplePath workshop. Let us know :-)

# Live demo

https://github.com/peoplepath/fine-tuning-php

# Sources

- https://www.zend.com/blog/exploring-new-php-jit-compiler
- https://support.cloud.engineyard.com/hc/en-us/articles/205411888-PHP-Performance-I-Everything-You-Need-to-Know-About-OpCode-Caches
- https://www.php.net/manual/en/book.opcache.php