The One Billion Row Challenge:

je PHP rychlejší než Java?

27.11.2024 v 18:00

Restaurace U Salzmannů

Salonek v 1. patře

Vstup a občerstvení zdarma

PeoplePath

Few facts about me...

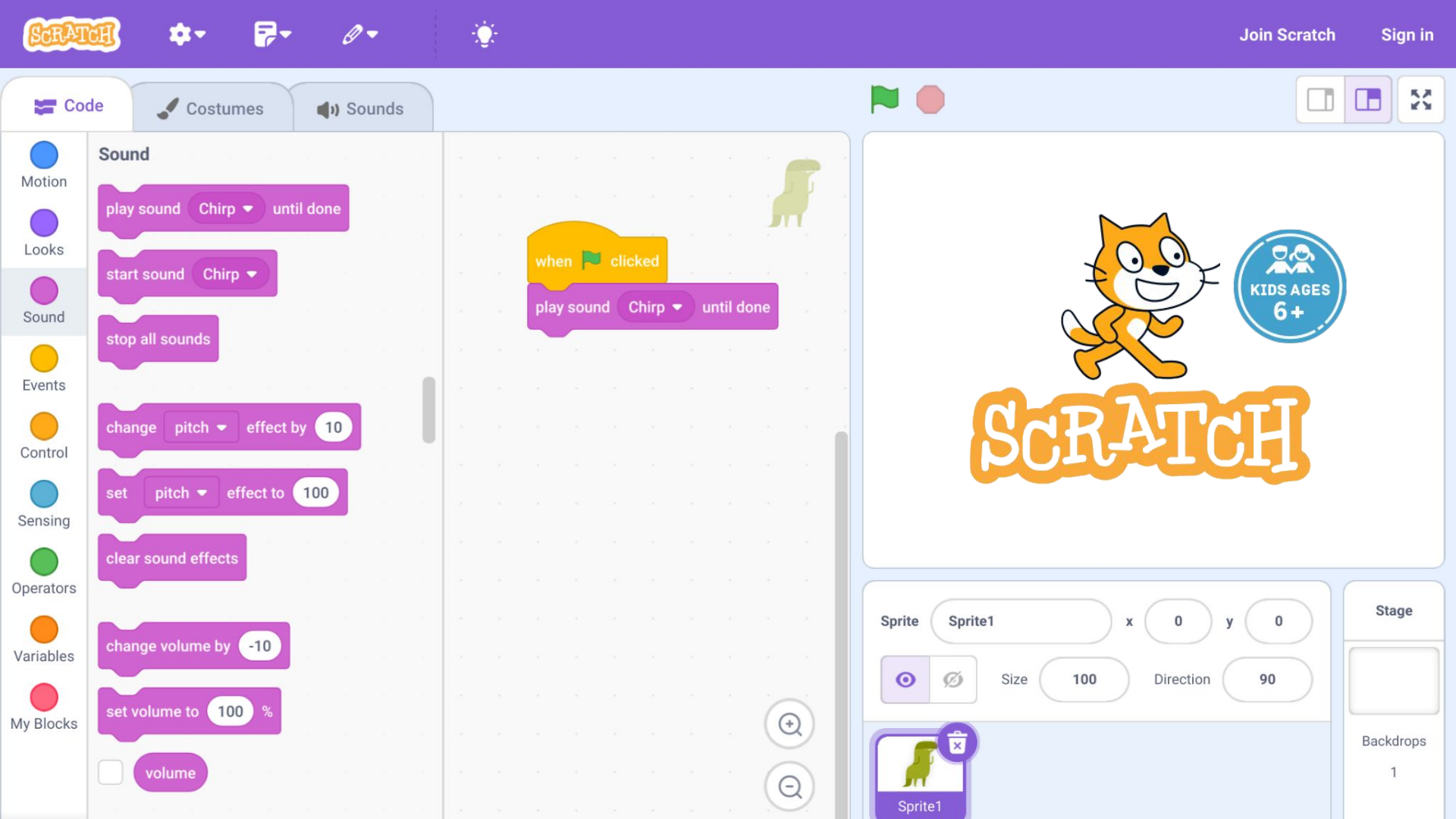
- Software architect
- +25 years experience in IT.
- Contributor to PHP magazines.
- Conference speaker.

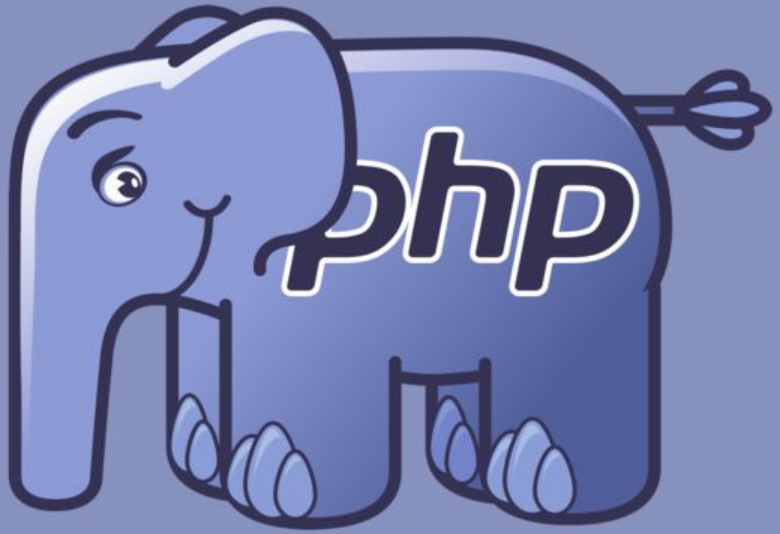
Few facts about me...

- Demotivational speaker.
- Loves crazy projects.
- Tried to patent QR payments in the USA.

Languages that I use the most...


C++
Italiano
Polski
Python
R#
Javascript
JAVA
TypeScript
PHP
Gherkin
Kotlin
Bash
Matlab
Ruby
C#
czech english bash
Czech
js
Nix
GPT
HTML
Plzeňština
CSS


Join Scratch
Sign in
Code
Costumes
Sounds
Motion
Looks
Sound
Events
Control
Sensing
Operators
Variables
My Blocks
Sound
play sound Chirp until done
start sound Chirp
stop all sounds
change pitch effect by 10
set pitch effect to 100
clear sound effects
change volume by -10
set volume to 100 %
volume
when clicked
play sound Chirp until done
KIDS AGES 6+
SCRATCH
Sprite
Sprite1
x
0
y
0
Size
100
Direction
90
Sprite1
Stage
Backdrops
1


php

Error 502 - Google Chrome

Error 502

php.net

Error 502

Bad Gateway

What happened?

The communication with the origin web server timed out or was invalid. Therefore the web page cannot be displayed.

What can I do?

If you are a visitor of this website:

Please try again later.

If you are the owner of this website:

Please make sure that your origin server is not under heavy load. Sometimes a crashing program can also be the root cause.

`opcache.jit_debug` int

A bit mask specifying which JIT debug output to enable. For possible values, please consult » zend_jit.h (search for macro definitions beginning with `ZEND_JIT_DEBUG`).

`opcache.jit_bisect_limit` int

Debugging option that disables JIT compilation after compiling a certain number of functions. This may be helpful to bisect the source of a JIT miscompilation. Note: this option only works when JIT trigger is set to 0 (compile on script load) or 1 (compile on first execution), e.g. `opcache.jit=1215`. See more in opcache.jit option.

`opcache.jit_prof_threshold` float

When using the "profile on first request" trigger mode, this threshold determines whether a function is considered hot. The number of calls to the function divided by the number of calls to all functions must be above the threshold. For example, a threshold of 0.005 means that functions that made up more than 0.5% of all calls will be JIT compiled.

`opcache.jit_max_root_traces` int

Maximum number of root traces. The root trace is an execution flow taking one path through the code firstly, which is a unit of JIT compilation. JIT will not compile new code if it reaches this limit.

`opcache.jit_max_side_traces` int

Maximum number of side traces a root trace may have. The side trace is another execution flow that does not follow the path of compiled root trace. Side traces belonging to the same root trace will not be compiled if it reaches this limit.

`opcache.jit_max_exit_counters` int

Maximum number of side trace exit counters. This limits the total number of side traces there may be, across all root traces.

`opcache.jit_hot_loop` int

After how many iterations a loop is considered hot. Valid value range is `[0,255]`; for any setting out of this range, e.g. `-1` or `256`, default value will be used instead. `0` will disable JIT to trace and compile any loops.

`opcache.jit_hot_func` int

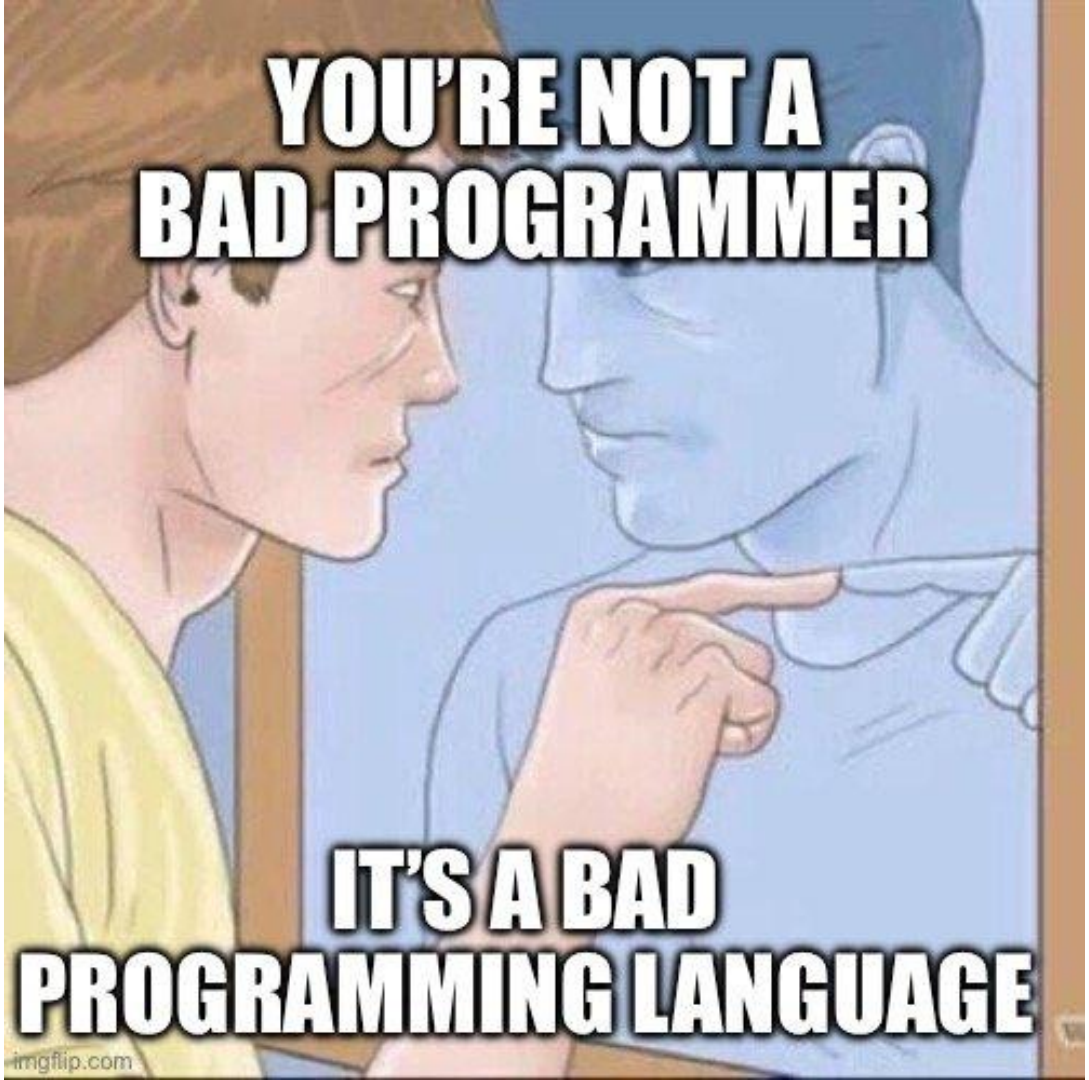
After how many calls a function is considered hot. Valid value range is `[0,255]`; for any setting out of this range, e.g. `-1` or `256`, default value will be used instead. `0` will disable JIT to trace and compile any functions.

`opcache.jit_hot_return` int

After how many returns a return is considered hot. Valid value range is `[0,255]`; for any setting out of this range, e.g. `-1` or `256`, default value will be used instead. `0` will disable JIT to trace and compile any returns.

`opcache.jit_hot_side_exit` int

After how many exits a side exit is considered hot. Valid value range is `[0,255]`; for any setting out of this range, e.g. `-1` or `256`, default


YOU'RE NOT A
BAD PROGRAMMER
IT'S A BAD
PROGRAMMING LANGUAGE
imgflip.com

1BRC

 The One Billion Row Challenge

1BRC - objective

- Calculate stats from 1B-row file.

```
Hamburg;12.0
Bulawayo;8.9
Palembang;38.8
Hamburg;5.0
London;0.8
```

1BRC - output

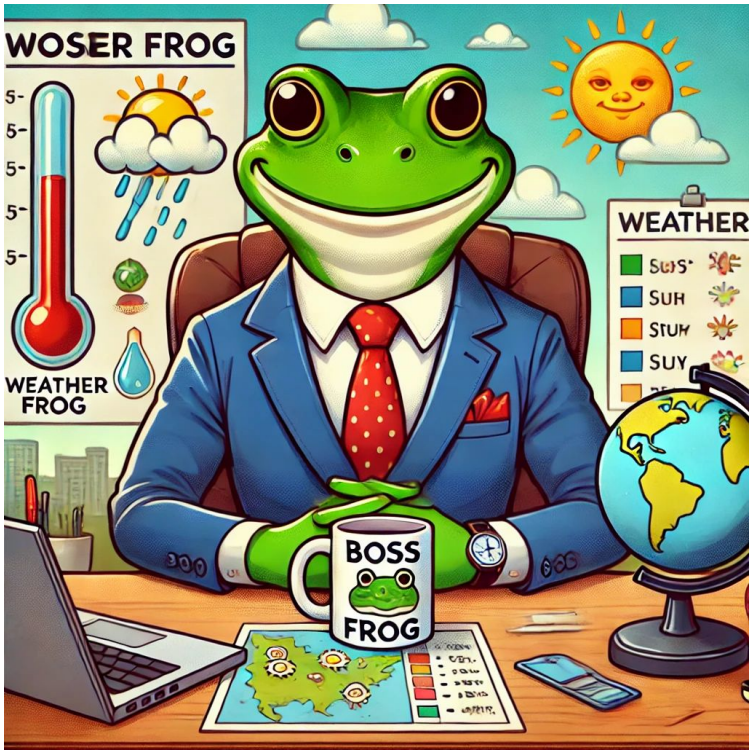
- Print min, mean, and max temps per station alphabetically.

```
{Abha=5.0/18.0/27.4, Abidjan=15.7/26.0/34.1, ...}
```

1BRC - background

- Originally in JAVA.
- Other languages C, Python, RUST, etc.
- In PHP done by Florian Engelhardt.

TL;DR - funny challenge for optimisation



Measure methodology 1 BRC

- Using wall clock, on Linux.
- Results differ across hardware.
- Flushing vmdata caches after each run.
- CLI tool hyperfine

“ Talk is cheap. Show me the code. ”

Linus Torvalds

```php
<?php
$stations = []; // create empty array
$fp = fopen('measurements.txt', 'r'); // open file for read
while ($data = fgetcsv($fp, null, ';')) { // get line from file and parse it
  if (!isset($stations[$data[0]])) {
    $stations[$data[0]] = [
      $data[1], // minimum value
      $data[1], // maximum value
      $data[1], // sum of values
      1         // count of measurements
    ];
  } else {
    $stations[$data[0]][3] ++; // Increment the count
    $stations[$data[0]][2] += $data[1]; // Add to the sum
      if ($data[1] < $stations[$data[0]][0]) {
        $stations[$data[0]][0] = $data[1]; // Update the minimum value
      }
      if ($data[1] > $stations[$data[0]][1]) {
        $stations[$data[0]][1] = $data[1]; // Update the maximum value
     }
  }
}
ksort($stations); // Sort an array by key in ascending order
echo '{';
foreach($stations as $k=>&$station) {
  // Calculate the average value
  $station[2] = $station[2]/$station[3];
  echo $k, '=', $station[0], '/', $station[2], '/', $station[1], ", \n";
}
echo '}';
```

```php
<?php
$stations = [];
$fp = fopen('measurements.txt', 'r');
while ($data = fgetcsv($fp, null, ';')) {
```

- Array is an ordered map (KV, **hash table**).
- fgetcsv - get line from file and parse it.
  - Eg.: berlin;5.87 will return array of two elements: berlin and 5.87

```
4  while ($data = fgetcsv($fp, null, ';')) {
5    if (!isset($stations[$data[0]])) {
6      $stations[$data[0]] = [
7        $data[1], // minimum value
8        $data[1], // maximum value
9        $data[1], // sum of values
10       1         // count of measurements
11     ];
12   } else {
```

```
12    } else {
13      $stations[$data[0]][3] ++;
14      $stations[$data[0]][2] += $data[1];
15        if ($data[1] < $stations[$data[0]][0]) {
16          $stations[$data[0]][0] = $data[1];
17        }
18        if ($data[1] > $stations[$data[0]][1]) {
19          $stations[$data[0]][1] = $data[1];
20       }
21    }
22  } // while loop end
```

```
22  } // while loop end
23  ksort($stations);
24  echo '{';
25  foreach($stations as $k=>&$station) {
26   // Calculate the average value
27   $station[2] = $station[2]/$station[3];
28   echo $k, '=', $station[0], '/', $station[2], '/',
$station[1], ", \n";
29  }
30  echo '}';
```

What are arrays in PHP?

A. A collection of random values stored in memory.

B. A mystical portal that stores data in parallel universes.

C. An ordered map, which is a type of hash table.

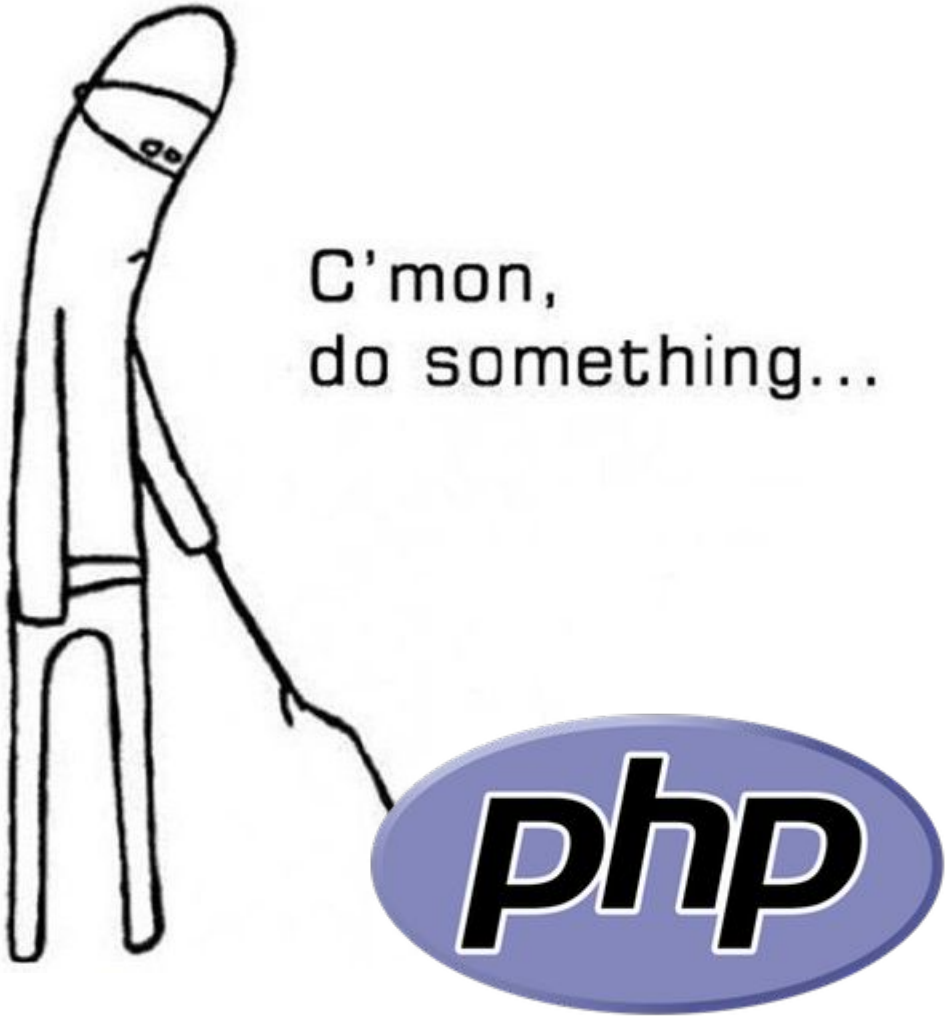
D. A fixed-size collection of integers.

What are arrays in PHP?

- A collection of random values stored in memory. — 8%
- A mystical portal that stores data in parallel universes. — 4%
- An ordered map, which is a type of hash table. — 83%
- A fixed-size collection of integers. — 4%

Baseline

902.70 sec


C'mon,
do something...
php

What to do?

What to do?

A. Use other language?

B. Consult with more experience developer?

C. Don’t change anything, check configuration?

D. Act like nothing is happening!

What to do?

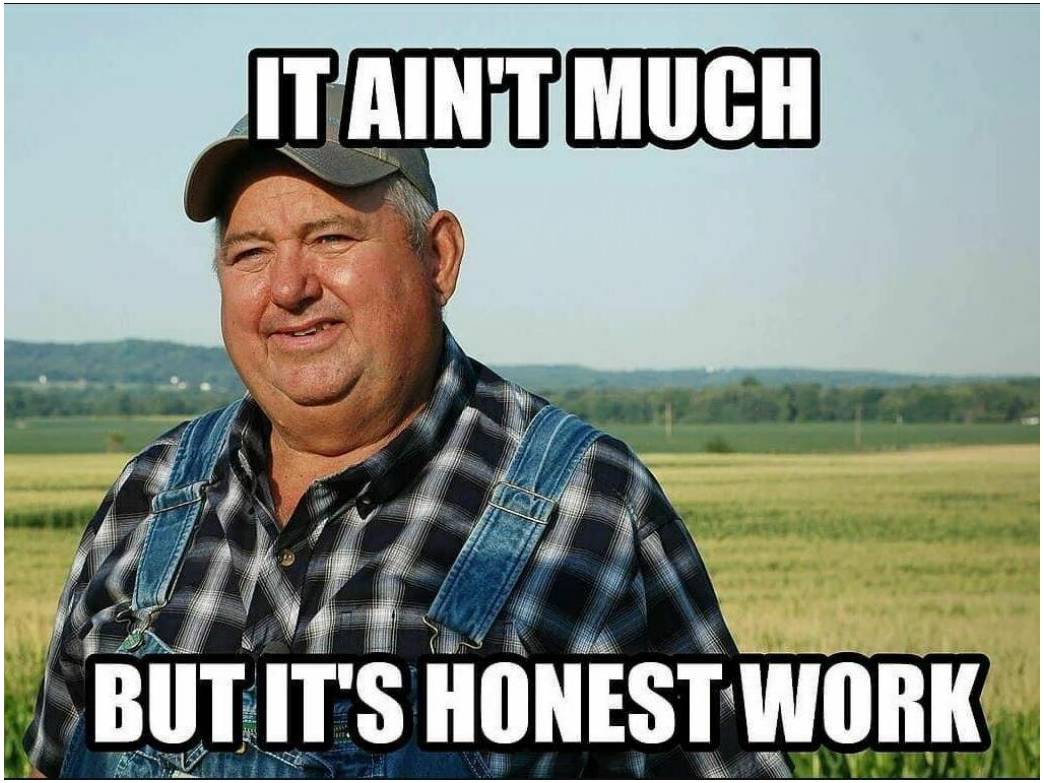
- Use other language? 44%
- Consult with more experience developer? 59%
- Don’t change anything, check configuration? 41%
- Act like nothing is happening! 26%

Don’t change anything, check configuration!

Don’t change anything

Version	Time (sec)	Raw difference	% improvement
Baseline	902.70	n/a	n/a
OpCache	882.97	19.73	2.19%
JIT	850.38	32.59	3.69%

- 5.8% (52.32 sec) just by configuration.


IT AIN'T MUCH
BUT IT'S HONEST WORK

OpCache

- Can be enabled also for CLI.
- Up to 6 optimisation passes.
- JIT engine.

TL;DR Check configuration and runtime

What’s next?

What’s next?

A. Randomly changing things?

B. Do the profiling?

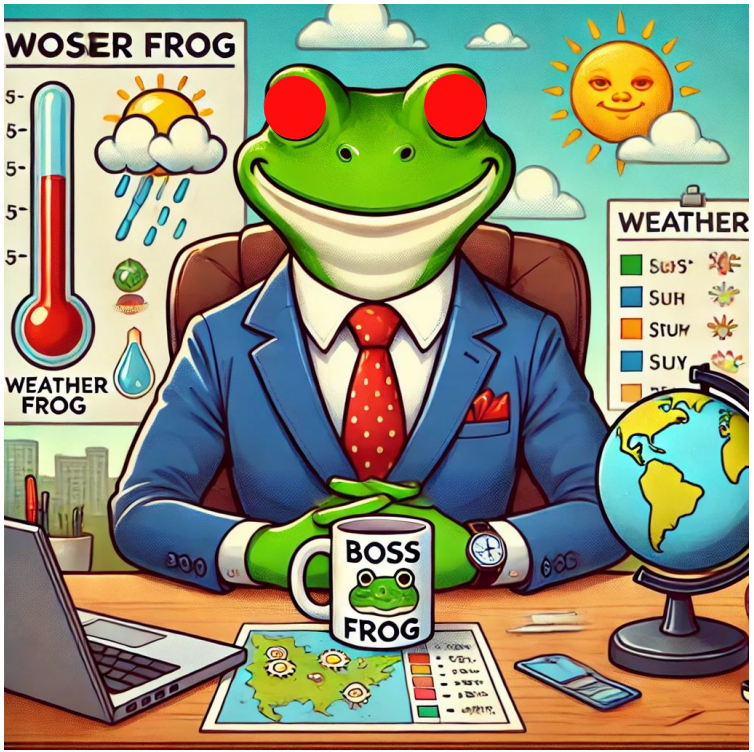
C. C’mon 850 sec? It’s fast enough!

D. Gave up programing, start farming?

What's next?


Randomly changing things?
7%
Do the profiling?
72%
C'mon 850 sec? It's fast enough!
3%
Gave up programing, start farming?
17%

C’mon 850 sec? It’s fast enough!



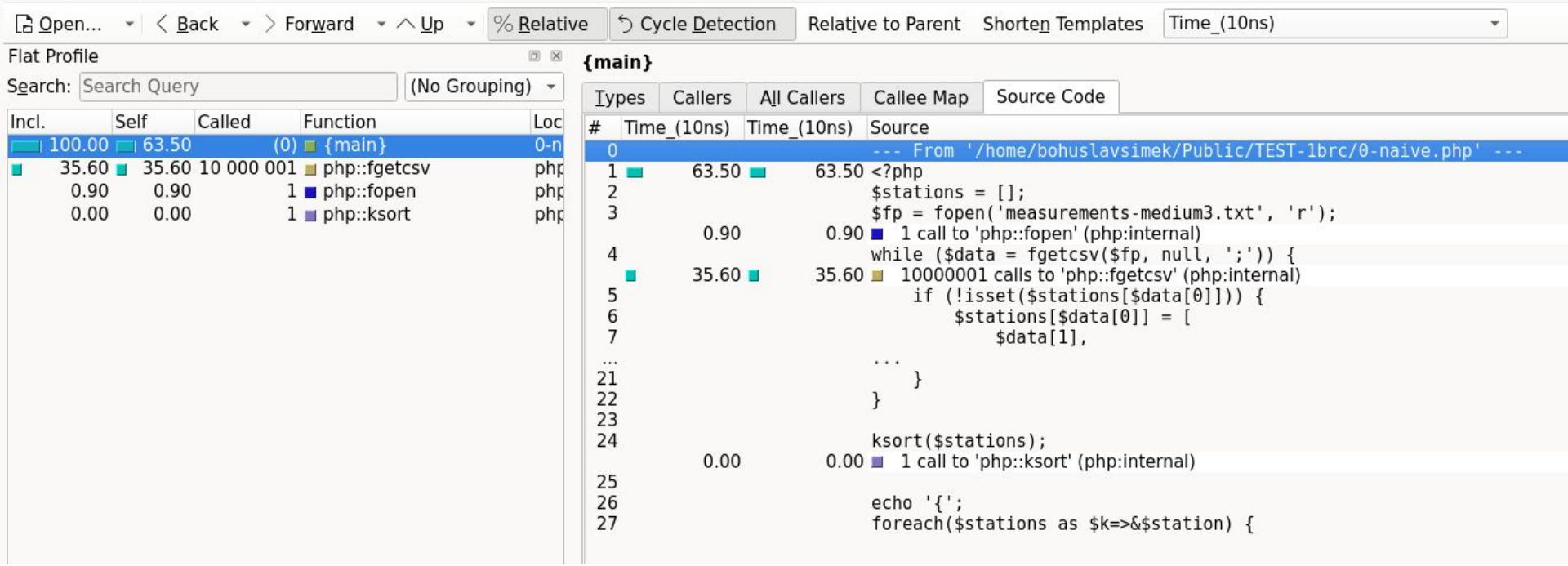
Croak NO!
Fix it, croak!

Profiling

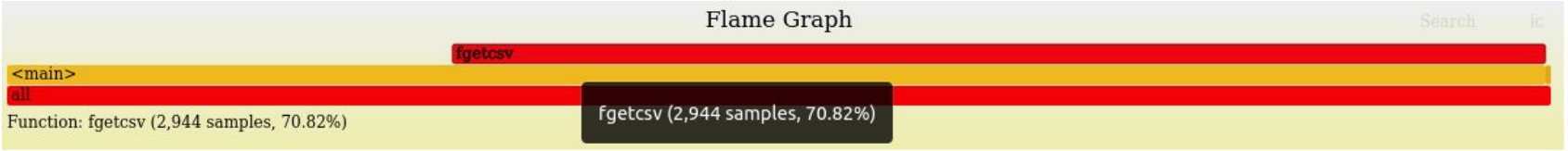
Profiling - PHP specific

- Xdebug (tracing profiler)
- Reli-prof (sampling profiler)
- Tideways (SASS)

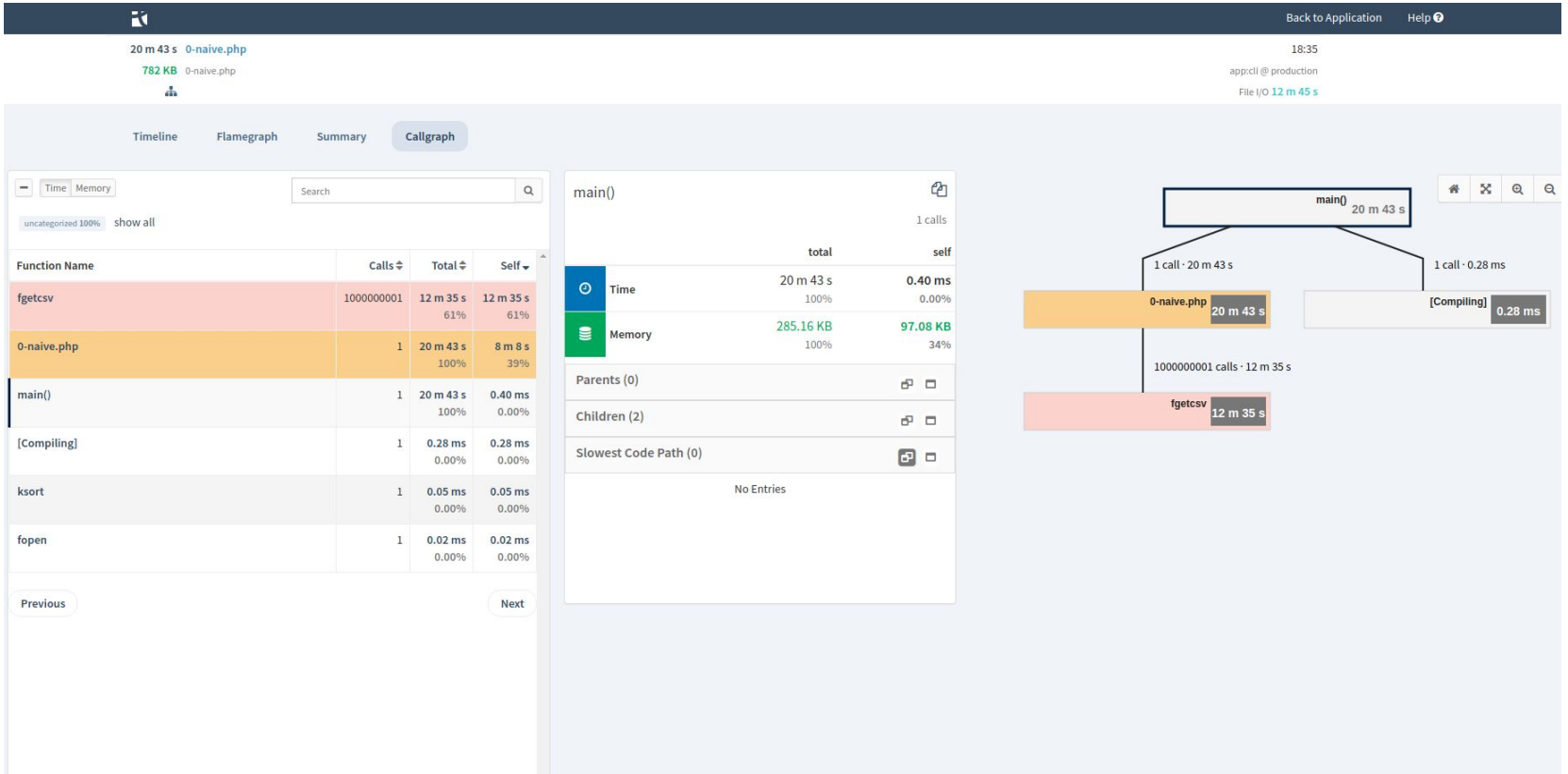
XDebug


Open...
Back
Forward
Up
Relative
Cycle Detection
Relative to Parent
Shorten Templates
Time_(10ns)
Flat Profile
Search:
Search Query
(No Grouping)
Incl.
Self
Called
Function
Loc
100.00
63.50
(0)
{main}
0-n
35.60
35.60
10 000 001
php::fgetcsv
php
0.90
0.90
1
php::fopen
php
0.00
0.00
1
php::ksort
php
{main}
Types
Callers
All Callers
Callee Map
Source Code
Time_(10ns) Time_(10ns) Source
0 --- From '/home/bohuslavsimek/Public/TEST-1brc/0-naive.php' ---
1 63.50 63.50 <?php
2 $stations = [];
3 $fp = fopen('measurements-medium3.txt', 'r');
0.90 0.90 1 call to 'php::fopen' (php:internal)
4 while ($data = fgetcsv($fp, null, ';')) {
35.60 35.60 10000001 calls to 'php::fgetcsv' (php:internal)
5 if (!isset($stations[$data[0]])) {
6 $stations[$data[0]] = [
7 $data[1],
... ...
21 }
22 }
23
24 ksort($stations);
0.00 0.00 1 call to 'php::ksort' (php:internal)
25
26 echo '{';
27 foreach($stations as $k=>&$station) {

Reli-prof (sampling profiler)



Tideways (SASS)



Profiling - universal tools

- Spread time function - “shotgun” method.
- Blackfire (SASS)
- Datadog (SASS)
- Perf (Linux)

```
Samples: 3K of event 'cycles:P', Event count (approx.): 145919026547
Overhead  Com  Shared Object      Symbol
  31,62%  php  libc.so.6          [.] __gconv_transform_utf8_internal
  21,26%  php  libc.so.6          [.] __mbrtowc
   6,27%  php  php                [.] zend_strtod
   5,66%  php  php                [.] php_fgetcsv_lookup_trailing_spaces
   4,21%  php  php                [.] _is_numeric_string_ex.part.0
   4,05%  php  php                [.] php_fgetcsv
   2,62%  php  libc.so.6          [.] _dl_mcount_wrapper_check
   1,84%  php  php                [.] zendi_smart_strcmp
   1,53%  php  php                [.] _php_stream_get_line
   1,46%  php  php                [.] zend_hash_find
   1,15%  php  php                [.] zif_fgetcsv
   0,94%  php  php                [.] add_function_slow
   0,90%  php  php                [.] zend_string_equal_val
   0,89%  php  php                [.] zend_hash_next_index_insert
   0,85%  php  php                [.] _efree
   0,76%  php  libc.so.6          [.] __memmove_avx_unaligned_erms
   0,76%  php  php                [.] zend_compare
   0,72%  php  php                [.] _emalloc
   0,64%  php  php                [.] mbrtowc@plt
   0,61%  php  opcache.so         [.] zend_jit_symtable_lookup_rw
   0,60%  php  php                [.] zend_hash_func
   0,54%  php  php                [.] ZEND_PRE_INC_SPEC_VAR_RETVAL_UNUSED_HANDLER
   0,53%  php  php                [.] _zendi_try_convert_scalar_to_number.isra.0
   0,45%  php  php                [.] add_function
   0,40%  php  php                [.] zend_array_destroy
   0,33%  php  php                [.] php_stream_locate_eol
   0,28%  php  php                [.] _zend_mm_alloc
   0,28%  php  php                [.] _zend_new_array_0
   0,22%  php  php                [.] zend_fetch_resource2
   0,22%  php  libc.so.6          [.] __ctype_b_loc
   0,22%  php  libc.so.6          [.] __memchr_avx2
   0,20%  php  php                [.] zend_is_true
   0,20%  php  php                [.] _emalloc_160
   0,20%  php  php                [.] _erealloc
   0,11%  php  php                [.] zval_ptr_dtor
   0,11%  php  php                [.] memcpy@plt
   0,11%  php  php                [.] _emalloc_56
   0,09%  php  [JIT] tid 31918    [.] 0x00005555c4600bf5
```

```
Samples: 3K of event 'cycles:P', Event count (approx.): 145929078431, Thread: php, DSO: php
  Children      Self  Com  Symbol
+   13,97%     5,98%  php  [.] php_fgetcsv_lookup_trailing_spaces
+   12,67%     5,87%  php  [.] zend_strtod
+    8,96%     4,36%  php  [.] php_fgetcsv
+    6,22%     4,08%  php  [.] _is_numeric_string_ex.part.0
+    5,56%     0,00%  php  [.] zend_execute
+    4,87%     1,60%  php  [.] zend_hash_find
+    3,57%     1,61%  php  [.] zendi_smart_strcmp
+    2,81%     0,00%  php  [.] php_stdiop_read
+    2,44%     1,51%  php  [.] _php_stream_get_line
+    2,44%     0,91%  php  [.] _efree
+    2,28%     0,37%  php  [.] _emalloc
+    1,95%     1,27%  php  [.] zend_hash_next_index_insert
+    1,88%     1,00%  php  [.] zif_fgetcsv
+    1,51%     0,77%  php  [.] zend_string_equal_val
+    1,43%     0,50%  php  [.] zend_compare
+    1,28%     0,77%  php  [.] add_function_slow
     1,12%     0,66%  php  [.] zend_array_destroy
     1,02%     0,99%  php  [.] mbrtowc@plt
     0,93%     0,54%  php  [.] php_stream_locate_eol
     0,86%     0,46%  php  [.] zend_hash_func
     0,79%     0,37%  php  [.] _zendi_try_convert_scalar_to_number.isra.0
     0,65%     0,36%  php  [.] ZEND_PRE_INC_SPEC_VAR_RETVAL_UNUSED_HANDLER
     0,62%     0,20%  php  [.] _emalloc_160
     0,60%     0,40%  php  [.] _zend_new_array_0
     0,56%     0,39%  php  [.] add_function
     0,49%     0,26%  php  [.] _erealloc
     0,40%     0,17%  php  [.] _emalloc_56
     0,37%     0,12%  php  [.] _zend_mm_alloc
     0,26%     0,06%  php  [.] _efree_56
     0,23%     0,11%  php  [.] zend_is_true
     0,23%     0,09%  php  [.] zend_fetch_resource2
     0,20%     0,09%  php  [.] rc_dtor_func
     0,15%     0,09%  php  [.] zval_ptr_dtor
     0,14%     0,08%  php  [.] memcpy@plt
     0,12%     0,12%  php  [.] __ctype_b_loc@plt
     0,09%     0,03%  php  [.] zend_string_hash_func
     0,08%     0,05%  php  [.] php_file_le_stream
     0,03%     0,03%  php  [.] php_file_le_pstream
```

add_function_slow 😕🤯😖

Perf - slowest functions

1. php_fgetcsv_lookup_trailing_spaces
2. zend_strtod
3. php_fgetcsv
4. is_numeric_string_ex.part.0
5. zend_execute
6. zend_hash_find

TL;DR Find Out what is slow.

How to fix this?

A. Use a higher PHP version to improve performance.

B. Compress the CSV file to reduce size before reading.

C. Rewrite the fgetcsv functionality.

D. Increase the memory limit for the PHP.

How to fix this?

- Use a higher PHP version to improve performance. 4%
- Compress the CSV file to reduce size before reading. 15%
- Rewrite the fgetcsv functionality. 78%
- Increase the memory limit for the PHP. 4%

Replace fgetcsv with fgets

fgets

(PHP 4, PHP 5, PHP 7, PHP 8)

fgets — Gets line from file pointer

Description

```
fgets(resource $stream, ?int $length = null): string|false
```

Gets a line from file pointer.

Parameters

stream

The file pointer must be valid, and must point to a file successfully opened by fopen() or fsockopen() (and not yet closed by fclose()).

Replace fgetcsv with fgets

```
- while ($data = fgetcsv($fp, null, ';')) {
-     if (!isset($stations[$data[0]])) {
-         $stations[$data[0]] = [
```

```
+ while ($data = fgets($fp, null)) {
+     $pos = strpos($data, ';');
+     $city = substr($data, 0, $pos);
+     $temp = substr($data, $pos+1, -1);
+     if (!isset($stations[$city])) {
+         $stations[$city] = [
```

Replace fgetcsv with fgets

Before
850.38 sec

After
294.80 sec

65.33% faster!

Any downside to this replacement?

A. More code, replacing a function with more lines.

B. Lacks robust CSV support for edge cases.

C. A built-in LLM to fix typos in your CSV file.

D. Higher risk of bugs in custom implementation.

Any downside to this replacement?

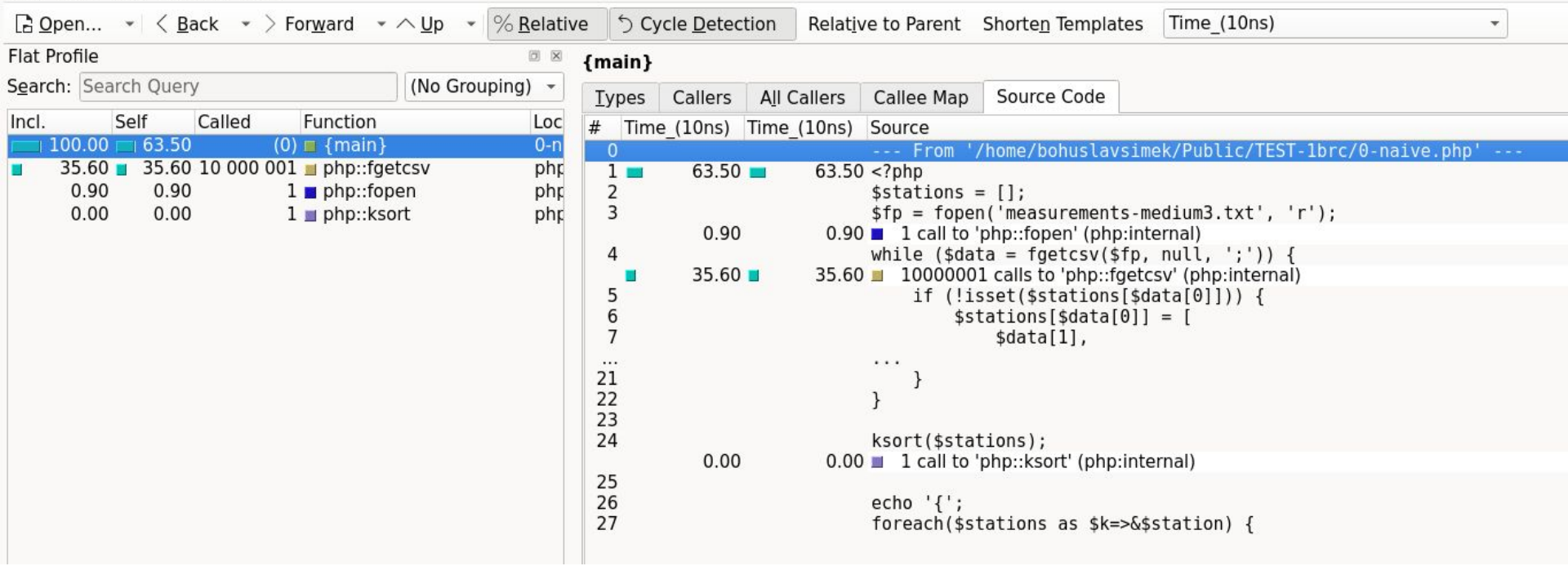
- More code, replacing a function with more lines. 39%
- Lacks robust CSV support for edge cases. 57%
- A built-in LLM to fix typos in your CSV file. 4%
- Higher risk of bugs in custom implementation. 75%

TL;DR
Find out what's essential.

TL;DR
Do only what's essential.

What’s next?
Rinse and repeat!

XDebug


Open...
Back
Forward
Up
Relative
Cycle Detection
Relative to Parent
Shorten Templates
Time_(10ns)
Flat Profile
Search:
Search Query
(No Grouping)
Incl.
Self
Called
Function
Loc
100.00
63.50
(0)
{main}
0-n
35.60
35.60
10 000 001
php::fgetcsv
php
0.90
0.90
1
php::fopen
php
0.00
0.00
1
php::ksort
php
{main}
Types
Callers
All Callers
Callee Map
Source Code
Time_(10ns) Time_(10ns) Source
0 --- From '/home/bohuslavsimek/Public/TEST-1brc/0-naive.php' ---
1 63.50 63.50 <?php
2 $stations = [];
3 $fp = fopen('measurements-medium3.txt', 'r');
0.90 0.90 1 call to 'php::fopen' (php:internal)
4 while ($data = fgetcsv($fp, null, ';')) {
35.60 35.60 10000001 calls to 'php::fgetcsv' (php:internal)
5 if (!isset($stations[$data[0]])) {
6 $stations[$data[0]] = [
7 $data[1],
... ...
21 }
22 }
23
24 ksort($stations);
0.00 0.00 1 call to 'php::ksort' (php:internal)
25
26 echo '{';
27 foreach($stations as $k=>&$station) {

```
Samples: 23K of event 'cycles:P', Event count (approx.): 96664162627
  Children      Self  Command  Shared Object                  Symbol
+   35,23%    18,12%  php      php                            [.] zend_strtod
+   22,28%     0,00%  php      [unknown]                      [.] 0000000000000000
+   19,17%    12,62%  php      php                            [.] _is_numeric_string_ex.part.0
+   13,74%     4,78%  php      php                            [.] zend_hash_find
+   10,75%     0,00%  php      php                            [.] zend_execute
+    9,68%     0,00%  php      [unknown]                      [.] 0x0000001600000001
+    8,97%     5,02%  php      php                            [.] zendi_smart_strcmp
+    7,45%     4,49%  php      php                            [.] zif_substr
+    7,32%     2,35%  php      php                            [.] _emalloc
+    6,91%     0,00%  php      [unknown]                      [.] 0xfa1e0ff327ff41ff
+    5,98%     2,69%  php      php                            [.] add_function_slow
+    4,84%     1,87%  php      php                            [.] zend_compare
+    4,52%     2,28%  php      php                            [.] zend_string_equal_val
+    4,34%     2,97%  php      php                            [.] _php_stream_get_line
+    4,26%     0,00%  php      [unknown]                      [.] 0xfffffa78000000d0
+    4,15%     0,01%  php      [kernel.kallsyms]              [k] do_syscall_64
+    4,14%     0,00%  php      [kernel.kallsyms]              [k] entry_SYSCALL_64_after_hwframe
+    4,11%     0,01%  php      libc.so.6                      [.] read
+    4,10%     0,00%  php      [unknown]                      [k] 0x0000002400000006
+    3,99%     0,02%  php      [kernel.kallsyms]              [k] __x64_sys_read
+    3,85%     1,08%  php      libc.so.6                      [.] __memchr_avx2
+    3,40%     2,48%  php      php                            [.] zif_fgets
     3,21%     2,84%  php      php                            [.] _efree
+    3,20%     1,84%  php      php                            [.] zif_strpos
+    2,85%     1,75%  php      libc.so.6                      [.] __memmove_avx_unaligned_erms
+    2,83%     1,59%  php      php                            [.] ZEND_ASSIGN_DIM_OP_SPEC_VAR_CONST_HANDLER
+    2,73%     0,00%  php      [unknown]                      [.] 0x4cf7014c0c776349
+    2,68%     0,00%  php      [kernel.kallsyms]              [k] x64_sys_call
+    2,63%     0,01%  php      [kernel.kallsyms]              [k] ksys_read
+    2,63%     0,03%  php      [kernel.kallsyms]              [k] vfs_read
+    2,41%     0,00%  php      [unknown]                      [.] 0x0000000000000005
+    2,39%     1,49%  php      php                            [.] zend_hash_func
+    2,37%     0,08%  php      [kernel.kallsyms]              [k] filemap_read
+    2,37%     0,01%  php      [kernel.kallsyms]              [k] ext4_file_read_iter
+    2,36%     1,52%  php      opcache.so                     [.] zend_jit_symtable_lookup_rw
+    2,35%     0,01%  php      [kernel.kallsyms]              [k] generic_file_read_iter
+    2,35%     1,16%  php      php                            [.] _zendi_try_convert_scalar_to_number.isra.0
+    2,33%     0,76%  php      php                            [.] _erealloc
```

Perf - slowest functions

1. ~~php_fgetcsv_lookup_trailing_spaces~~
2. zend_strtod
3. ~~php_fgetcsv~~
4. is_numeric_string_ex.part.0
5. zend_execute
6. zend_hash_find

How can I find out if this is connected to something I can change?

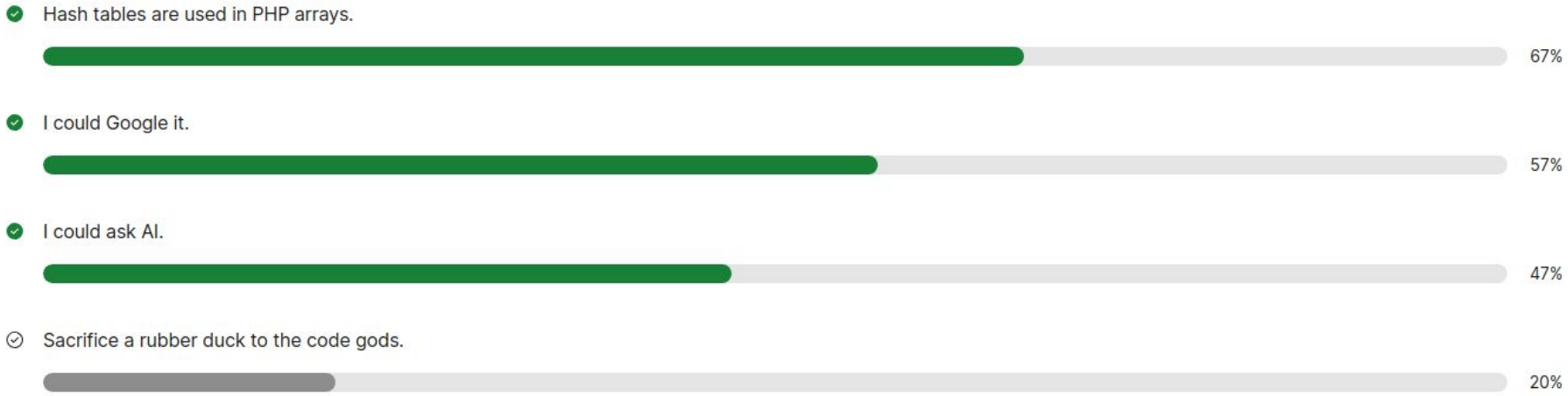
A. Hash tables are used in PHP arrays.

B. I could Google it.

C. I could ask AI.

D. Sacrifice a rubber duck to the code gods.

How can I find out if this is connected to something I can change?



zend_hash_find?
Hash table?

Can I, as a PHP programmer, limit the number of calls to the *zend_hash_find* function?

Can I, as a PHP programmer, limit the number of calls to the zend_hash_find function?

Yes, as a PHP programmer, you can indirectly limit the number of calls to `zend_hash_find` by optimizing how you interact with PHP arrays and hash tables. The `zend_hash_find` function is a low-level C function in the Zend Engine, which PHP uses to look up values in arrays (hash tables). While you cannot directly modify the engine to limit calls to `zend_hash_find`, you can optimize your PHP code to minimize unnecessary lookups. Here are some strategies:

---

1. Cache Results of Expensive Lookups

If you are performing multiple lookups on the same key, store the result in a variable instead of repeatedly querying the array:

php Copy code

```php
// Inefficient
for ($i = 0; $i < 100; $i++) {
    $value = $array['key']; // zend_hash_find is called every time
    // Process $value
}

// Efficient
$value = $array['key']; // zend_hash_find called once
for ($i = 0; $i < 100; $i++) {
    // Process $value
```

```
12      } else {
13        $stations[$data[0]][3] ++;
14        $stations[$data[0]][2] += $data[1];
15          if ($data[1] < $stations[$data[0]][0]) {
16            $stations[$data[0]][0] = $data[1];
17          }
18          if ($data[1] > $stations[$data[0]][1]) {
19            $stations[$data[0]][1] = $data[1];
20         }
21      }
22  } // while loop end
```

Lets try it!

Use references

```
-           $stations[$city][3] ++;
-           $stations[$city][2] += $temp;
-           if ($temp < $stations[$city][0]) {
-               $stations[$city][0] = $temp;
```

```
+           $station = &$stations[$city];
+           $station[3] ++;
+           $station[2] += $temp;
+           if ($temp < $station[0]) {
+               $station[0] = $temp;
```

Use references

Before
294.80 sec

After
276.06 sec

6.36% faster!

```
Samples: 22K of event 'cycles:P', Event count (approx.): 93830033974
  Children      Self  Command  Shared Object                    Symbol
+   39,71%    18,94%  php      php                              [.] zend_strtod
+   23,89%     0,00%  php      [unknown]                        [k] 0000000000000000
+   21,59%    13,03%  php      php                              [.] _is_numeric_string_ex.part.0
+   10,75%     0,00%  php      php                              [.] zend_execute
+    9,32%     5,33%  php      php                              [.] zendi_smart_strcmp
+    7,69%     4,77%  php      php                              [.] zif_substr
+    7,30%     2,45%  php      php                              [.] _emalloc
+    6,37%     0,00%  php      [unknown]                        [.] 0x0000001600000001
+    5,61%     0,00%  php      [unknown]                        [.] 0xfa1e0ff327ff41ff
+    5,14%     2,10%  php      php                              [.] zend_compare
+    5,09%     3,35%  php      php                              [.] _php_stream_get_line
+    4,94%     0,00%  php      [unknown]                        [.] 0x0000001a00000001
+    4,32%     1,14%  php      libc.so.6                        [.] __memchr_avx2
+    3,96%     2,17%  php      php                              [.] ZEND_ASSIGN_REF_SPEC_CV_VAR_HANDLER
+    3,94%     1,36%  php      php                              [.] zend_hash_find
+    3,85%     2,38%  php      php                              [.] add_function_slow
+    3,60%     2,10%  php      php                              [.] zif_strpos
+    3,50%     2,62%  php      php                              [.] _efree
+    3,19%     0,00%  php      [unknown]                        [.] 0xfffffa78000000b0
+    3,17%     2,16%  php      php                              [.] zif_fgets
+    2,98%     1,71%  php      php                              [.] zend_hash_func
+    2,92%     1,88%  php      libc.so.6                        [.] __memmove_avx_unaligned_erms
+    2,77%     0,01%  php      [kernel.kallsyms]                [k] entry_SYSCALL_64_after_hwframe
+    2,76%     0,02%  php      libc.so.6                        [.] read
+    2,75%     0,00%  php      [kernel.kallsyms]                [k] do_syscall_64
+    2,74%     0,00%  php      [unknown]                        [.] 0x0000002400000006
+    2,72%     0,00%  php      [unknown]                        [.] 0x0f03087b83f3014c
+    2,72%     0,00%  php      [unknown]                        [.] 0xfffffa00000000d0
+    2,70%     0,00%  php      [unknown]                        [.] 0x0000704a43c141b0
+    2,68%     1,47%  php      php                              [.] ZEND_ASSIGN_DIM_OP_SPEC_CV_CONST_HANDLER
+    2,56%     0,02%  php      [kernel.kallsyms]                [k] __x64_sys_read
+    2,51%     1,40%  php      php                              [.] _zendi_try_convert_scalar_to_number.isra.0
```

Perf - slowest functions

1. ~~php_fgetcsv_lookup_trailing_spaces~~
2. zend_strtod
3. ~~php_fgetcsv~~
4. is_numeric_string_ex.part.0
5. zend_execute
6. ~~zend_hash_find~~

TL;DR
Minimise expansive code calling

TL;DR
"try it and see"

Check for redundancy...

```
12      } else {
13        $station = &$stations[$city];
14        $station[3] ++;
15        $station[2] += $temp;
16          if ($temp < $station[0]) {
17            $station[0] = $temp;
18          }
19          if ($temp > $station[1]) {
20            $station[1] = $temp;
21          }
22      }
23  } // while loop end
```

Check for Redundancy

- Temperature can't exceed max or fall below min simultaneously.
- Both conditions are mutually exclusive.

Remove extra comparison

```
        $station[2] += $temp;
        if ($temp < $station[0]) {
            $station[0] = $temp;
-       }
-       if ($temp > $station[1]) {
+       } else if ($temp > $station[1]) {
            $station[1] = $temp;
        }
```

Remove extra comparison

Before
276.06 sec

After
250.58 sec

9.23% faster!

Perf - slowest functions

1. ~~php_fgetcsv_lookup_trailing_spaces~~
2. zend_strtod
3. ~~php_fgetcsv~~
4. is_numeric_string_ex.part.0
5. zend_execute
6. ~~zend_hash_find~~

zend_strtod and is_numeric_string_ex.part.0

In PHP's Zend Engine, `zend_strtod` and `is_numeric_string_ex.part.0` are key parts of its handling of numeric values and string conversion. Here's a breakdown:

`zend_strtod`

- **Purpose**: This function is responsible for converting a string to a floating-point number (`double`) in a locale-independent way.
- **Key Use Case**: It's utilized internally by PHP when a string needs to be parsed as a floating-point number. For example:

```php
$number = (float)"3.14"; // zend_strtod likely plays a role here.
```

- **Implementation Notes**:
  - It bypasses locale-specific behavior to ensure consistency regardless of system settings.
  - This is crucial for avoiding issues where, for example, a locale might use commas (`1,23`) instead of dots (`1.23`) for decimals.

`is_numeric_string_ex.part.0`

- **Purpose**: This is an internal function used to determine whether a string can be interpreted as a numeric value (either an integer or a float). It may also indicate how the string should be interpreted.

Dynamic Typing

- Variables in PHP do not have fixed types.
- A variable's type can change at runtime.
- Is often dynamic - is_numeric_string_ex
- Can be expansive.

add_function_slow 😕🤯😖


```c
static zend_never_inline zend_result ZEND_FASTCALL add_function_slow(zval *result, zval *op1, zval *op2)
{
   // parts omitted for clarity
   zval op1_copy, op2_copy;
   if (UNEXPECTED(zendi_try_convert_scalar_to_number(op1, &op1_copy) == FAILURE)
         || UNEXPECTED(zendi_try_convert_scalar_to_number(op2, &op2_copy) == FAILURE)) {
      zend_binop_error("+", op1, op2);
      if (result != op1) {
         ZVAL_UNDEF(result);
      }
      return FAILURE;
   }

   if (add_function_fast(result, &op1_copy, &op2_copy) == SUCCESS) {
     return SUCCESS;
   }
   // parts omitted for clarity
}
```

How to limit impact of dynamic types in this case?

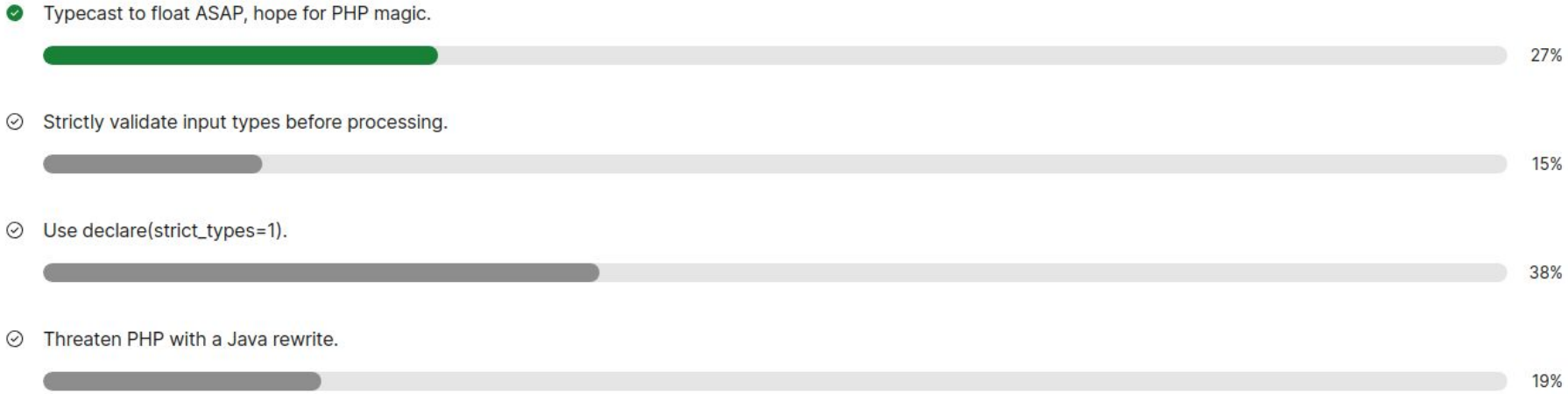
A. Typecast to float ASAP, hope for PHP magic.

B. Strictly validate input types before processing.

C. Use declare(strict_types=1).

D. Threaten PHP with a Java rewrite.

How to limit impact of dynamic types in this case?



Typecasting

```
  while ($data = fgets($fp, null)) {
      $pos = strpos($data, ';');
      $city = substr($data, 0, $pos);
-     $temp = substr($data, $pos+1, -1);
+     $temp = (float) substr($data, $pos+1, -1);
      if (!isset($stations[$city])) {
          $stations[$city] = [
              $temp,
```

Typecasting

Before
250.58 sec

After
159.45 sec

36.37% faster!

```
Samples: 12K of event 'cycles:P', Event count (approx.): 52597824389
  Children      Self  Command  Shared Object                    Symbol
+   23,08%     0,00%  php      [unknown]                        [.] 0000000000000000
+   17,84%     0,00%  php      php                              [.] zend_execute
+   17,81%     9,51%  php      php                              [.] zend_strtod
+   15,39%     0,00%  php      [unknown]                        [.] 0xfa1e0ff327ff41ff
+   12,73%     7,98%  php      php                              [.] zif_substr
+   12,00%     4,21%  php      php                              [.] _emalloc
+    9,23%     0,00%  php      [unknown]                        [.] 0xfffffa58000000b0
+    7,75%     2,30%  php      php                              [.] zend_hash_find
+    7,68%     4,96%  php      php                              [.] _php_stream_get_line
+    7,04%     3,64%  php      php                              [.] ZEND_ASSIGN_REF_SPEC_CV_VAR_HANDLER
+    6,88%     4,71%  php      php                              [.] _efree
+    5,87%     1,68%  php      libc.so.6                        [.] __memchr_avx2
+    5,74%     3,76%  php      php                              [.] zif_fgets
+    5,59%     3,02%  php      php                              [.] zend_hash_func
+    5,46%     2,97%  php      php                              [.] zif_strpos
+    5,23%     3,16%  php      libc.so.6                        [.] __memmove_avx_unaligned_erms
+    4,68%     2,60%  php      php                              [.] ZEND_ASSIGN_DIM_OP_SPEC_CV_CONST_HANDLER
+    4,41%     1,75%  php      php                              [.] zend_hash_lookup
+    4,39%     0,00%  php      [unknown]                        [.] 0x0000002400000006
+    4,39%     0,04%  php      [kernel.kallsyms]                [k] entry_SYSCALL_64_after_hwframe
+    4,38%     0,03%  php      [kernel.kallsyms]                [k] do_syscall_64
+    4,38%     0,02%  php      libc.so.6                        [.] read
+    3,96%     0,00%  php      [unknown]                        [.] 0xfffffa88000000b0
+    3,90%     1,90%  php      php                              [.] zend_string_equal_val
+    3,68%     0,03%  php      [kernel.kallsyms]                [k] __x64_sys_read
+    3,53%     2,58%  php      php                              [.] ZEND_CAST_SPEC_VAR_HANDLER
+    2,78%     0,92%  php      php                              [.] ZEND_PRE_INC_SPEC_VAR_RETVAL_UNUSED_HANDLER
+    2,62%     1,10%  php      php                              [.] _erealloc
+    2,49%     0,00%  php      [unknown]                        [.] 0xffffffff0000010a
+    2,45%     0,02%  php      [kernel.kallsyms]                [k] x64_sys_call
+    2,45%     0,00%  php      [unknown]                        [.] 0x0f03087b83f3014c
+    2,45%     0,00%  php      [unknown]                        [.] 0xfffff9c0000000d0
+    2,40%     1,70%  php      php                              [.] php_stream_locate_eol
+    2,39%     1,21%  php      php                              [.] zend_fetch_dimension_address_inner_RW_CONST
+    2,21%     0,00%  php      [unknown]                        [.] 0x0000001a00000001
+    2,18%     0,98%  php      php                              [.] _zend_mm_alloc
+    1,97%     0,09%  php      [kernel.kallsyms]                [k] vfs_read
```

```
Samples: 12K of event 'cycles:P', Event count (approx.): 52597824389
  Children      Self  Command  Shared Object                                 Symbol
-   23,08%     0,00%  php      [unknown]                                     [.] 0000000000000000
   - 0
      - 17,84% zend_execute
           2,09% ZEND_ASSIGN_DIM_OP_SPEC_CV_CONST_HANDLER
           1,87% ZEND_PRE_INC_SPEC_VAR_RETVAL_UNUSED_HANDLER
           1,74% 0x58510ce012ea
           0,83% 0x58510ce011e5
           0,80% 0x58510ce012ec
           0,70% add_function
           0,69% 0x58510ce01370
           0,56% zend_is_true
           0,55% 0x58510ce012ac
        2,59% zend_hash_func
      - 1,55% 0x2820776100000106
           0,87% zval_get_double_func
```

Perf - slowest functions

1. ~~php_fgetcsv_lookup_trailing_spaces~~
2. ~~zend_strtod~~
3. ~~php_fgetcsv~~
4. ~~is_numeric_string_ex.part.0~~
5. ze 

ecute
6. ~~zend_hash_find~~

Use explode

```
  while ($data = fgets($fp, null)) {
-     $pos = strpos($data, ';');
-     $city = substr($data, 0, $pos);
-     $temp = (float) substr($data, $pos+1, -1);
+     [$city, $temperature] = explode(';', $data, 2);
+     $temp = (float) $temperature;
      if (!isset($stations[$city])) {
```

Use explode

Before
159.45 sec

After
158.73 sec

0.45% “faster” :-/

What to do next?

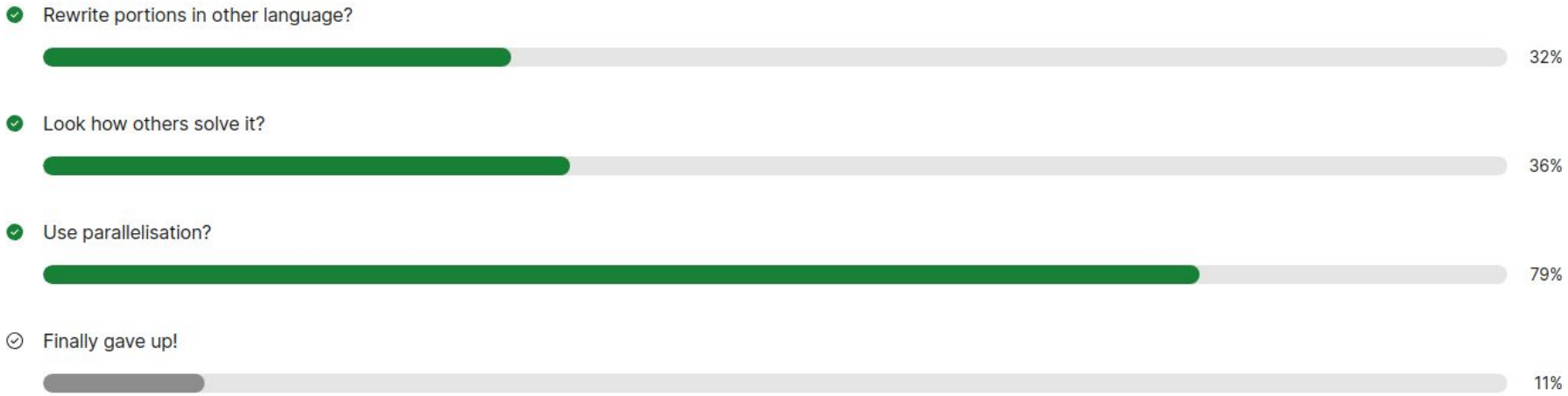
A. Rewrite portions in other language?

B. Look how others solve it?

C. Use parallelisation?

D. Finally gave up!

What to do next?



Own extension in C - fastfileloader

```
    $stations = [];
-   $fp = fopen('measurements.txt', 'r');
-   while ($data = fgets($fp, null)) {
+   $fp = fastfileloader_open(__DIR__.'/'.'measurements.txt');
+   while ($data = fastfileloader_readline($fp)) {
        [$city, $temperature] = explode(';', $data, 2);
```

Naive simple implementation?

Own extension in C - fastfileloader

- It’s 19.17% **slower** :-/
- Before 158.73 sec
- After **189.16** sec
- Reason: naive simple implementation.

Own extension in C - readline

```
PHP_FUNCTION(fastfileloader_readline) {
    // omitted for clarity

    if (fgets(buffer, sizeof(buffer), file) != NULL) {
        RETVAL_STRING(buffer);
    } else {
        if (feof(file)) {
            RETVAL_FALSE;  // Return FALSE on end of file
        } else {
            php_error_docref(
                NULL, E_WARNING, "Error reading from file"
            );
            RETVAL_FALSE;
        }
    }
}
```

Tl;dr - low level file access is need

- Need to redesign extension for peak perf.
- You need to know the OS and system things.
- In Linux mmap

Tl;dr - low level file access is need

```
PHP_FUNCTION(fastfileloader_open) {
   // removed for sake of clarity

   data = mmap(0, fileInfo.st_size, PROT_READ, MAP_PRIVATE, fd, 0);

   // removed for sake of clarity

   RETURN_RES(zend_register_resource(obj, le_file_resource));
}

PHP_FUNCTION(fastfileloader_readline) {
   // removed for sake of clarity

   char *end = memchr(start, '\n', obj->size - offset);

   // removed for sake of clarity

   RETVAL_STRINGL(start, len);
}
```

Own extension in C - second try

- Virtually same (speed up 0.77%) :-/
- Before 158.73 sec
- After 158.50 sec
- This is no worth it...

Own extension in C - second try

Before
158.73 sec

After
158.50 sec

0.77% “faster” :-/

TL;DR
You are not that smart.

Inspiration from others

Parallelisation beat them all...

- Fastest 1BRC solutions uses parallelisation.
- In PHP extension **parallel**.
- Thread support in constraints of CSP model.

Last resort...

Start threads

```php
$chunks = get_file_chunks($file, $threads_cnt); // $chunks = [[0, 123], [124, 256]]

$futures = [];

for ($i = 0; $i < $threads_cnt; $i++) {
    $runtime = new \parallel\Runtime();
    $futures[$i] = $runtime->run(  // $chunks =
        $process_chunk,
        [
            $file,
            $chunks[$i][0],
            $chunks[$i][1]
        ]
    );
}
```

Thread code

```php
$process_chunk = function (string $file, int $chunk_start, int $chunk_end): array {
    $stations = [];
    $fp = fopen($file, 'rb');
    fseek($fp, $chunk_start);
    while ($data = fgets($fp)) {
        $chunk_start += strlen($data);
        if ($chunk_start > $chunk_end) {
            break;
        }

        // regular processing
    }
    return $stations;
};
```

Synchronise and merge result

```
$results = [];

for ($i = 0; $i < $threads_cnt; $i++) {
    $chunk_result = $futures[$i]->value(); // blocks until a result is available,
    foreach ($chunk_result as $city => $measurement) {
        if (isset($results[$city])) {
            $result = &$results[$city];
            $result[2] += $measurement[2];
            $result[3] += $measurement[3];
            if ($measurement[0] < $result[0]) {
                $result[0] = $measurement[0];
            } else if ($measurement[1] > $result[1]) {
                $result[1] = $measurement[1];
            }
        } else {
            $results[$city] = $measurement;
        }
    }
}
```

Print result

```php
ksort($results);

echo '{', PHP_EOL;
foreach($results as $k=>&$station) {
    echo "\t", $k, '=', $station[0], '/',
    number_format($station[2]/$station[3], 1),
    '/', $station[1], ',', PHP_EOL;
}
echo '}', PHP_EOL;
```

Parallelisation beat them all...

Before
158.50 sec

After
29.81 sec

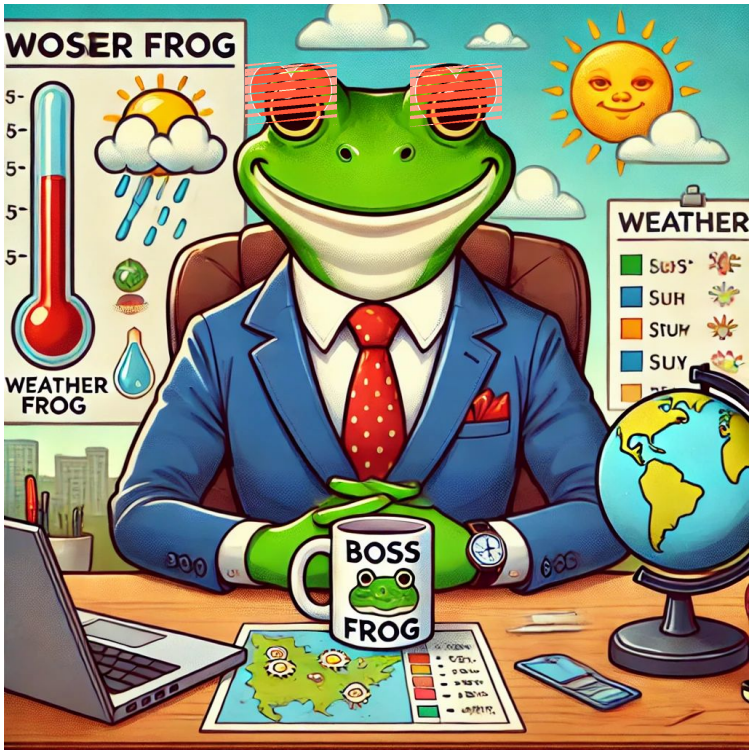
81.19% faster!

Parallelisation beat them all...

- It’s 81.19% faster!
- Before 158.50 sec
- With 20 threads 29.81 sec
- But you need extension and ZTS PHP build.

TL;DR
Take inspiration from others.

Boss seems happy!



Skip directly to parallelisation?

- fgets (2nd) version as a base version
- It’s 116.54% slower!
- 64.55 sec vs 29.81 sec
- macro-optimization/architectural change

Time to address the elephant in the room...

How does PHP compare with others?

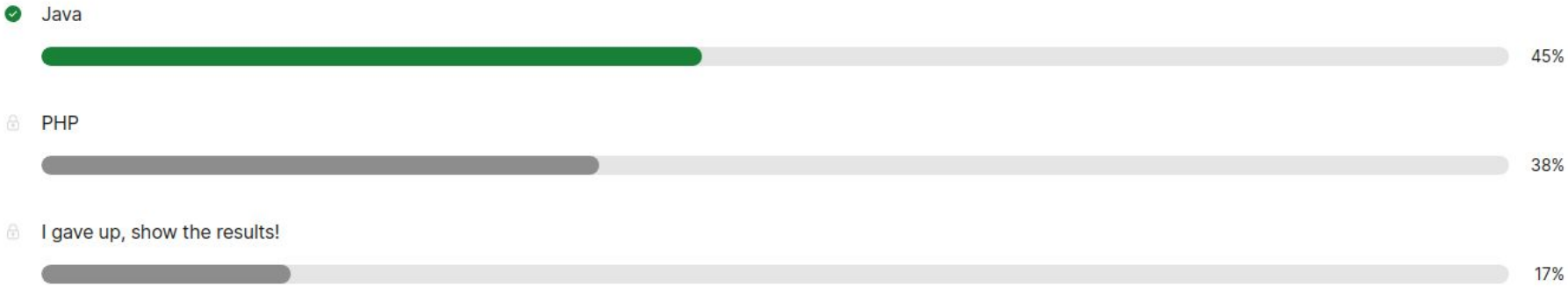
Who will be faster Java or PHP?

A. Java

B. PHP

C. I gave up, show the results!

Who will be faster Java or PHP?

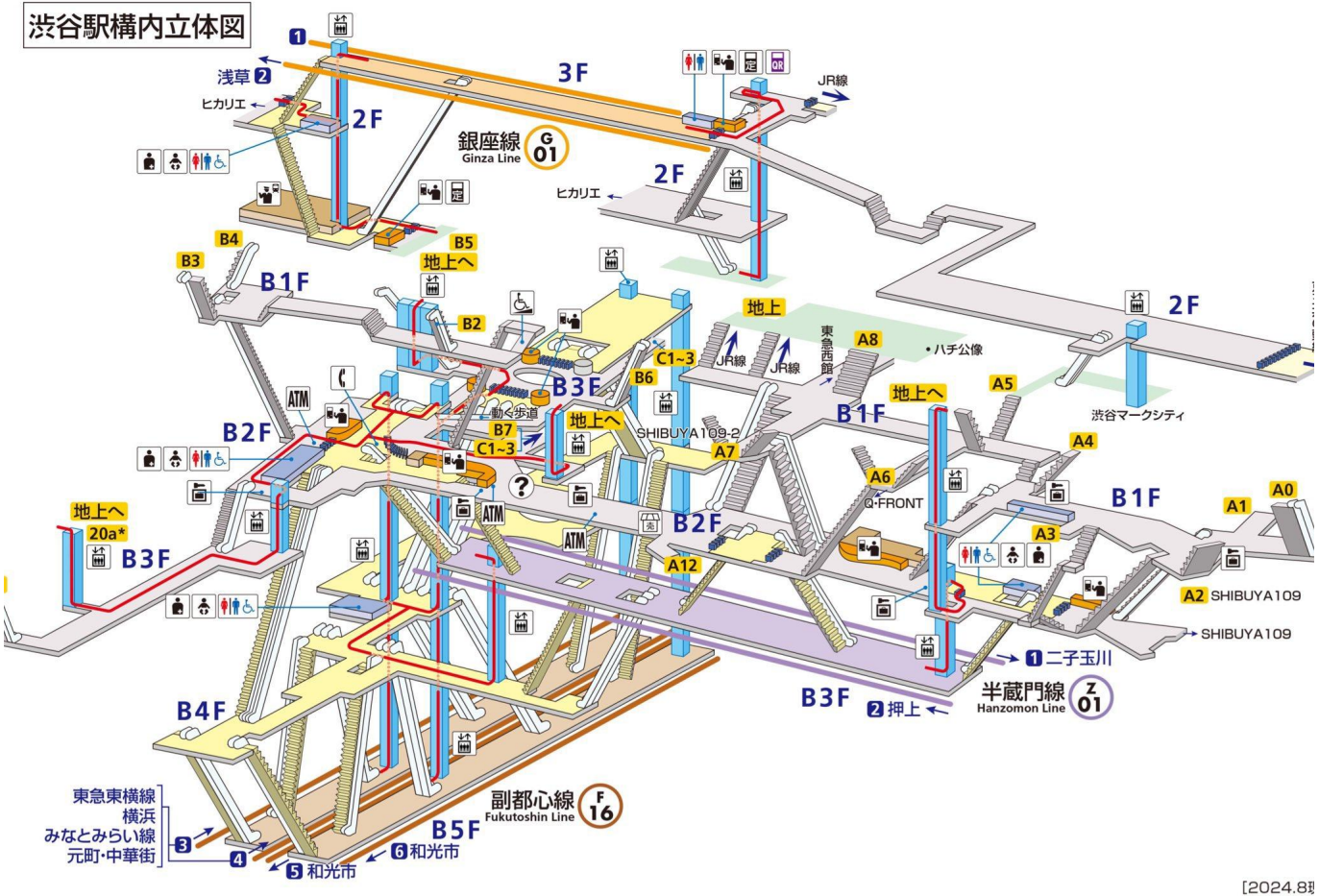


In broad context...

- Java
  - Baseline - 131.85 sec
  - Fastest - 5.2 sec
- PHP - 29.81 sec
- C - 5.25 sec

How best solution in JAVA looks?

渋谷駅構内立体図



[2024.8現

```
 private static long scanNumber(Scanner scanPtr) {
   long numberWord = scanPtr.getLongAt(scanPtr.pos() + 1);
   int decimalSepPos = Long.numberOfTrailingZeros(~numberWord & 0x10101000L);
   long number = convertIntoNumber(decimalSepPos, numberWord);
   scanPtr.add((decimalSepPos >>> 3) + 4);
   return number;
}

private static long convertIntoNumber(int decimalSepPos, long numberWord) {
   int shift = 28 - decimalSepPos;
   long signed = (~numberWord << 59) >> 63;
   long designMask = ~(signed & 0xFF);
   long digits = ((numberWord & designMask) << shift) & 0x0F000F0F00L;
   long absValue = ((digits * 0x640a0001) >>> 32) & 0x3FF;
   return (absValue ^ signed) - signed;
}
```

Why is Java version so fast?

- Developed by lead of GraalVM.
- Native AOT binary.
- Parallelize I/O
- Uses integers for calculation.

Why is Java version so fast?

- Custom hash table.
- Uses sun.misc.Unsafe.
- Branchless conversion between formats.
- Branch prediction.

My pragmatic solution?

Personal choice, DuckDB, ~33.96 sec

```
WITH src AS (
  SELECT station_name,
  MIN(measurement) AS min_measurement,
  CAST(AVG(measurement) AS DECIMAL(8,1)) AS mean_measurement,
  MAX(measurement) AS max_measurement
  FROM READ_CSV(
    'measurements.txt', header=false, columns= {'station_name':'VARCHAR','measurement':'double'}, delim=';', parallel=true
   )
  GROUP BY station_name
)

SELECT '{' ||
  ARRAY_TO_STRING(
     LIST_SORT(
       LIST(station_name || '=' || CONCAT_WS('/',min_measurement, mean_measurement, max_measurement))
     ),', '
  ) || '}' AS "1BRC"
FROM src;
```

Recapitulation

Recapitulation

- Check configuration and runtime.
- Get to know your tools.
- Find out what is slow.

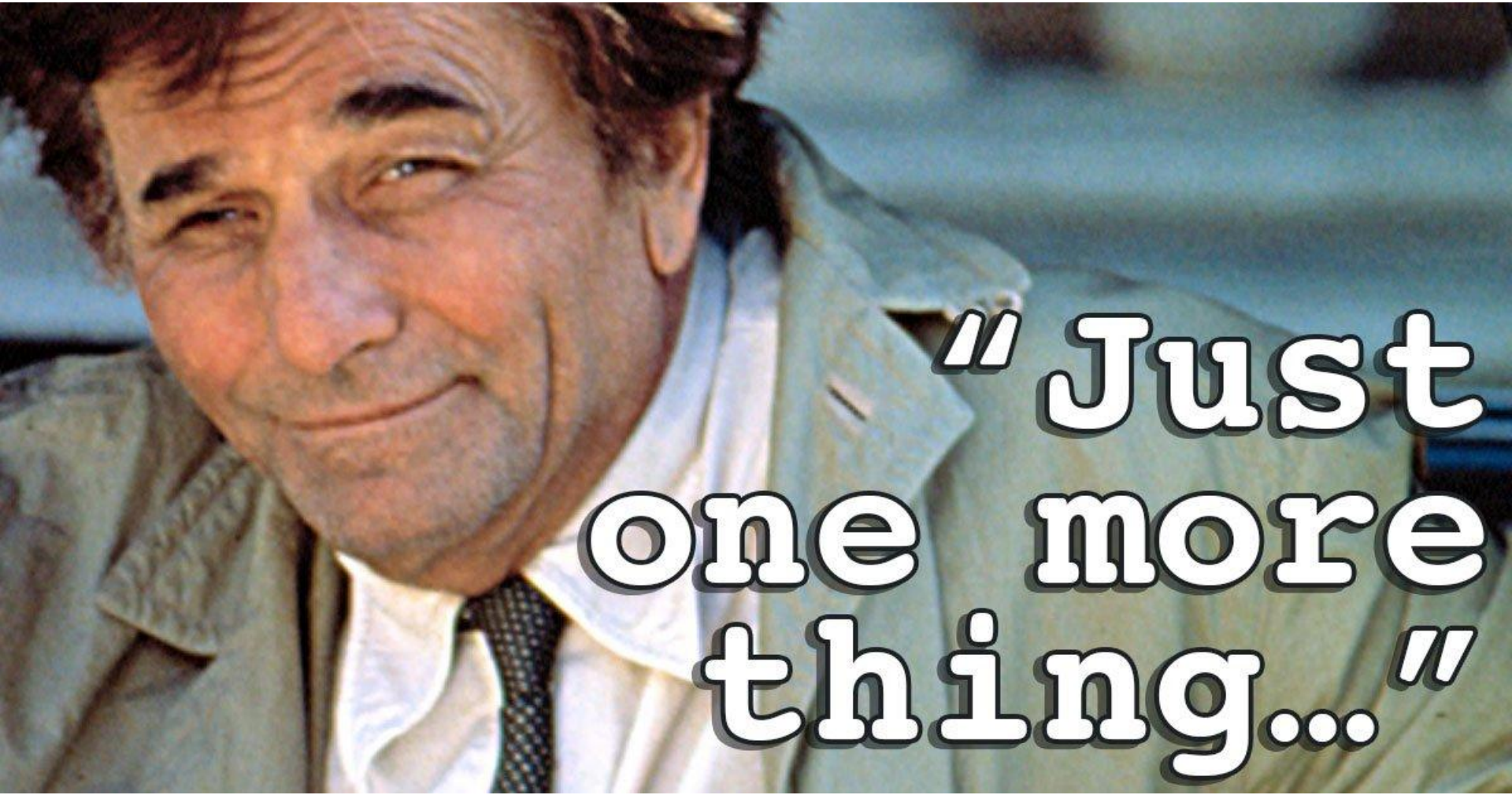
Recapitulation

- Find out what's essential.
- Do only what's essential.
- "Try it and see" – great for AI.

Recapitulation

- Minimise expansive code calling.
- You are not that smart.
- Take inspiration from others.

Was it worth it?


"Just one more thing..."

Quiz result

Quiz result

1	Vojta	7/9	4:03
2	Karel	5/9	4:10
3	Maňas	5/9	4:48
4	Your Boss	5/9	4:58
5	Šáník	5/9	5:33
6	Jiří	4/9	3:14
7	Miroslav Hatina	4/9	3:39
8	Perry	4/9	3:58
9	Marx	4/9	4:02
10	Jiří PPH	4/9	4:37
11	Prochi	4/9	4:38


"That's all Folks!"
Isberg©

Moderní PHP: Využijte sílu nových funkcí!

- **Co je nového v PHP?** – Představíme klíčové novinky z posledních verzí PHP, rychlostní testy, moderní syntax a nové funkcionality
- **Tipy na moderní kód** – Praktické ukázky, jak psát čitelnější, efektivnější a udržitelnější kód
- **Jak na migraci na nové verze?** – Rady a postupy, jak efektivně přejít na nejnovější verze PHP ve vašich projektech

Osteria Garage
Salonek v 1. patře
Částkova 56, Plzeň

29. 01. 2025
18:00 hod

Martin Flora