A Static Site Generator in PHP, Faster Than Hugo or Zola

Elmar Klausmeier, 30-Dec 2020

Agenda

- Why yet another static site generator?
- PHP8 + FFI
- Profiling PHP Programs with XHProf
- Reducing dependencies
- Making a generator 500-times faster: Benchmarking Saaze vs Hugo vs Zola

Why yet another static site generator?

- https://staticsitegenerators.net/ lists more than 450, https://jamstack.org/generators/ lists more than 300
- Some abandonded
- Some quite complex
- Some use their own complex template engine
- Those, that are easy to use, are overly simplistic

Easy to use: Saaze

- Easy to run: Only PHP+composer
- Easy to host: no external JS libraries required, even no PHP required
- Easy to edit: plain Markdown / CommonMark
- Easy to theme: Initially Blade Template Engine, now plain PHP
- Easy to understand, therefore easy to extend
- Reference: https://saaze.dev/

Extensions to Saaze

- Adding MathJax support (inline+display)
- YouTube, Vimeo, Twitter, CodePen, WpVideo, Mermaid
  - [youtube]YpX8DoS2hr8[/youtube]
  - [vimeo]126529871[/vimeo]
- Ampersand glitch fix in Markdown
  - abc [uvw](../iop&a) xyz
  - abc uvw xyz
- Story could have ended here

Using Simplified Saaze

- php.ini: extension=yaml
- Enable FFI in php.ini
- composer create-project eklausme/saaze-example
- pacman -S md4c
- cc -fPIC -Wall -O2 -shared php_md4c_toHtml.c -o php_md4c_toHtml.so -lmd4c-html
- Edit config.php in vendor/eklausme/saaze

PHP8 + FFI

- PHP was always quite easy to extend:
  - Download PHP source
  - ./configure needs 2m
  - make -j8 needs another 2m
- FFI (=Foreign Function Interface) new in PHP 7.4: greatly simplifies adding C routines to PHP, or accessing C structures in PHP
- Developed by Dmitry Stogov from Zend
- C routines are the general carrier for all sorts of languages, like Julia, Go, Lua, Fortran, COBOL, etc.

FFI general observations

- FFI routines cannot easily be bundled with composer. There are no “custom installers” or generic “vendor binaries” for FFI.
- FFI documentation misleading: “Currently, accessing FFI data structures is significantly (about 2 times) slower than accessing native PHP arrays and objects. Therefore, it makes no sense to use the FFI extension for speed; however, it may make sense to use it to reduce memory consumption.” Speed is one of the main motivations for using FFI!

Calling MD4C from PHP via FFI

- MD4C is a C library to convert Markdown to HTML written by Martin Mitas
- Almost always 2-10 times faster than other converters
- PHP code to call MD4C:
  - `$ffi = FFI::cdef("char *md4c_toHtml(const char*);","/srv/http/php_md4c_toHtml.so");`
  - `$html = FFI::string( $ffi->md4c_toHtml($markdown) );`
- C compilation:
  - `cc -fPIC -Wall -O2 -shared php_md4c_toHtml.c -o php_md4c_toHtml.so -lmd4c-html`
  - C code contains no calls to external libraries, no special types or typecasts
- Config in php.ini file has to be changed:
  - `extension=ffi`
  - `ffi.enable=true`

Calling COBOL from PHP via FFI

- GnuCOBOL + PHP work great together

```
<?php
$cbl = FFI::cdef("int phpsqrt(void); “.
  “void cob_init_nomain(int,char**); “.
  “int cob_tidy(void);","/srv/http/phpsqrt.so");
$ffj0 = FFI::cdef("double j0(double);","libm.so.6");

$cbl->cob_init_nomain(0,null);
$ret = $cbl->phpsqrt();
printf("\tret = %d<br>\n",$ret);
echo "Before calling cob_tidy():<br>\n";
echo "\tReturn: ", $cbl->cob_tidy(), "<br>\n";
printf("j0(2) = %f<br>\n", $ffj0→j0(2));
?>
```

Is PHP fast enough?

- Many of the static site generators are written in Go, Rust, JS – largely reflecting that new software is often written in new trendy languages
- PHP8 roughly 8-times slower than JS

Language	NUC	Ryzen	Odroid
C	0.17	0.15	0.44
Java 1.8.0	0.31	0.22	108.17
node.js 16.4	0.34	0.21	1.67
LuaJIT	0.49	0.33	2.06
PyPy3 7.3.5	1.57	0.86	n/a
PHP 8	3.23	2.35	42.38
Python 3.9.6	12.29	7.65	168.17
Perl 5.34	25.87	21.14	209.47

Profiling PHP with XHProf, #1

- https://www.php.net/manual/en/book.xhprof.php hierarchical profiler
- Installing:
  - cd extension
  - phpize
  - configure
  - make
  - Copy so file to PHP libraries (/usr/lib/php/modules/)
  - Change php.ini
  - Copy JavaScript libraries (/usr/share/webapps/xhprof/)

Profiling PHP with XHProf, #2

- Start

```
xhprof_enable(XHPROF_FLAGS_CPU + XHPROF_FLAGS_MEMORY);
```

- Stop

```
function stopXhprof() {
    $xhprof_data = xhprof_disable();
    include_once "/usr/share/webapps/xhprof/xhprof_lib/utils/xhprof_lib.php";
    include_once "/usr/share/webapps/xhprof/xhprof_lib/utils/xhprof_runs.php";
    $xhprof_runs = new \XHProfRuns_Default();
    $run_id = $xhprof_runs->save_run($xhprof_data, "saaze");
    echo "http://<xhprof-ui-address>/index.php?run=$run_id&source=saaze\n";
}
```

Profiling PHP with XHProf, #3

Screen output from https://eklausmeier.goip.de/blog/2021/10-18-profiling-php-programs/

Function Name	Calls	Calls%	Incl. Wall Time (microsec)	IWall%	Excl. Wall Time (microsec)	EWall%	Incl. CPU (microsecs)
main()	1	0.0%	950,667	100.0%	2,000	0.2%	941,893
Saaze\BuildCommand::buildEntry	341	0.1%	680,565	71.6%	6,809	0.7%	674,717
Saaze\TemplateManager::render	357	0.1%	642,288	67.6%	2,348	0.2%	636,502
Saaze\TemplateManager::renderEntry	341	0.1%	634,506	66.7%	4,154	0.4%	629,049
Jenssegers\Blade\Blade::render	357	0.1%	353,641	37.2%	1,610	0.2%	350,347
Jenssegers\Blade\Blade::__construct	357	0.1%	284,532	29.9%	3,728	0.4%	282,212
Illuminate\View\View::render	357	0.1%	279,927	29.4%	1,799	0.2%	277,393
Illuminate\View\View::renderContents	357	0.1%	273,861	28.8%	2,527	0.3%	271,428
Illuminate\View\View::getContents	357	0.1%	259,843	27.3%	1,597	0.2%	257,546
Illuminate\View\Engines\CompilerEngine::get	357	0.1%	256,008	26.9%	2,637	0.3%	253,825
Illuminate\View\Engines\PhpEngine::evaluatePath	357	0.1%	241,397	25.4%	2,922	0.3%	239,303
Illuminate\Container\Container::resolve	1,428	0.5%	240,809	25.3%	17,845	1.9%	239,322
Illuminate\Filesystem\Filesystem::getRequire	357	0.1%	236,356	24.9%	1,790	0.2%	234,362
Illuminate\Filesystem\Filesystem::Illuminate\Filesystem\{closure}	357	0.1%	233,381	24.5%	19,007	2.0%	231,435
Illuminate\Container\Container::get	714	0.2%	229,098	24.1%	1,645	0.2%	227,392

Dependencies in old Saaze

- adbario/php-dot-notation (*)
- erusev/parsedown-extra (*)
- jenssegers/blade
- php-di/php-di (*)
- ralouphie/mimey (*)
- spatie/yaml-front-matter (*)
- symfony/console (*)
- symfony/finder (*)
- symfony/http-foundation (O)
- symfony/process
- symfony/routing
- symfony/yaml (S)
- vlucas/phpdotenv (*)

- *-entries can easily be disposed, they are unnecessary
- S-entries very simple
- O-entries very much overkill
- The other at least have a concrete functionality

Reducing dependencies

- There seems to be a trend to:
  - re-program PHP builtin functionality with new libraries, then hosted on GitHub increasing the karma of its authors
  - these libraries in turn depend on even more libraries, and so on
- “Malware found in coa and rc, two npm packages with 23M weekly downloads”
- Some of those libraries are just abandoned: no response to issues, no fixes, no upgrades, not accepting patches

Reducing dep’s: PECL's Yaml Way Faster Than Symfony's Yaml

- 40% of runtime in Saaze is spent in yaml parsing
- Installing PECL’s Yaml
  - Download latest yaml package from PECL
  - tar zxf yaml...
  - Run phpize
  - Run ./configure
  - Run make. Library is here modules/yaml.so.
  - Run make test. Conducted 74 tests, which all succeeded.
  - As root copy modules/yaml.so to /usr/lib/php/modules
  - Edit /etc/php/php.ini and add extension=yaml
- Check with “php -m”

Reducing dep’s: Blade templates #1

- Having its own syntax
- Difficult to pass variables, except global variables
- Difficult to check correctness of templates: errors pop-up at the latest point, i.e., during final execution
- Trying to mimic PHP syntax+functionality: nice, but PHP is way more powerful
- Slow (compiler, cache)
- Alternative: Use plain PHP for templating – you have full programmability, way faster

Reducing dep's: Blade templates #2

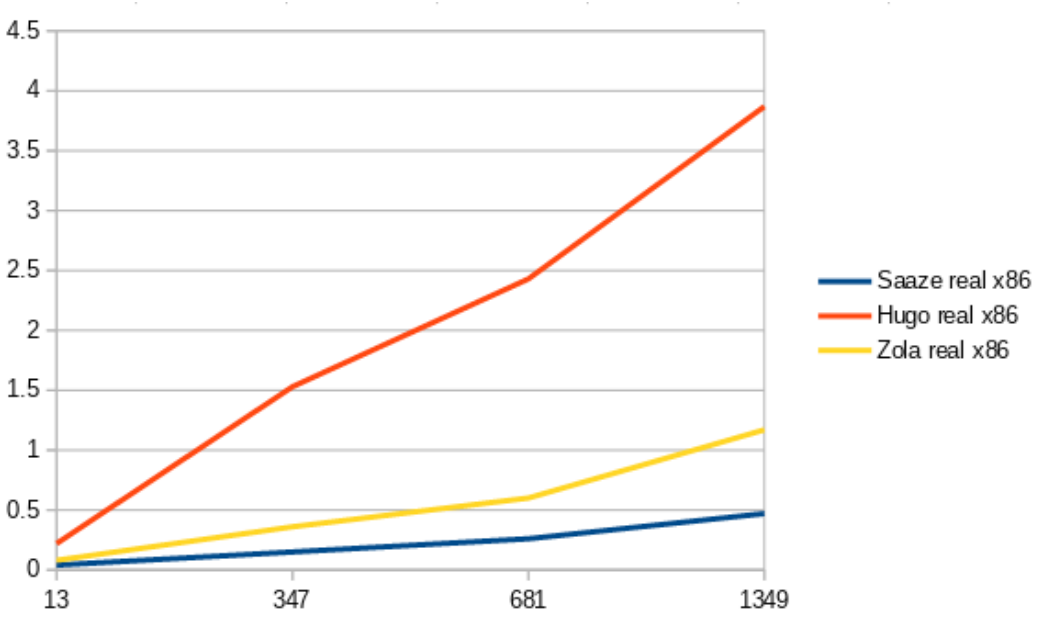
```
        <footer>  <p class="blogarea
dimmedColorSansSerif"><br><br>Generated <?= date('d-M-y
H:i') ?> GMT<br><br></p>  </footer>

<?php if (isset($GLOBALS['Fatty_Javascript'])) { ?>
    <!-- Yandex.Metrika counter -->
    <script type="text/javascript" >
    (function(m,e,t,r,i,k,a){m[i]=m[i]||function(){
    (m[i].a=m[i].a||[]).push(arguments)};
    Ym(..., "init", { … });
    </script>
    <noscript><div><img src=https://mc.yandex.ru/.../>
    </div></noscript>
    <!-- /Yandex.Metrika counter -->
<?php } ?>

</body>
```

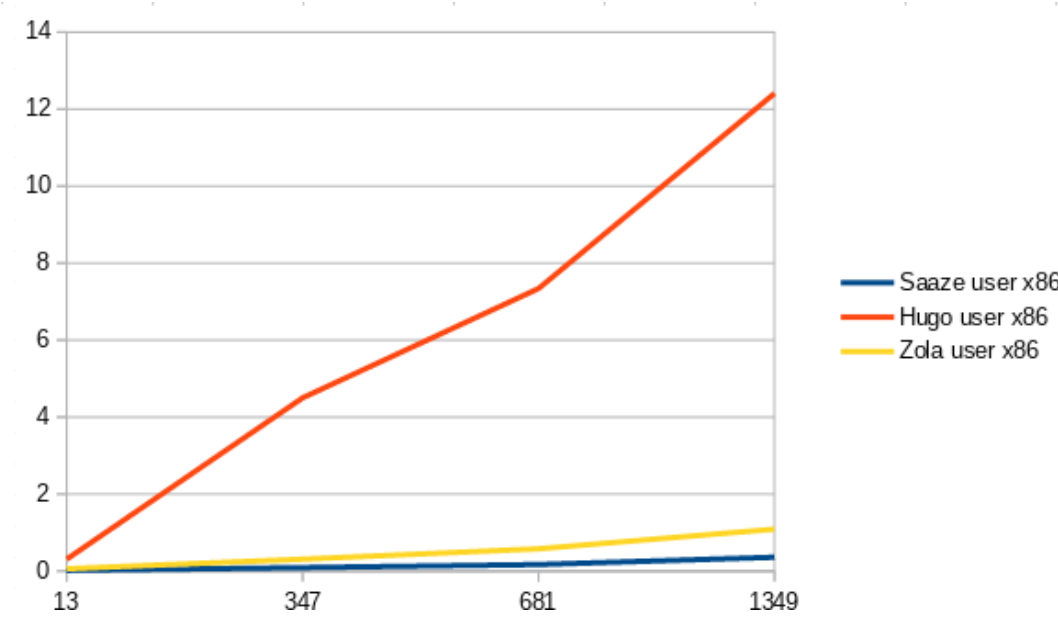
```
<?php if (isset($entry['MathJax'])) { ?>
        <script src="...polyfill.min.js?features=es6"></script>
        <script id="MathJax-script" async src=".../es5/tex-mml-
chtml.js"></script>
        <script id="MathJax-script" async src=".../es5/tex-mml-
chtml.js"></script>
<?php } ?>
<?php if (isset($entry['Twitter'])) { ?>
        <script async src="https://platform.twitter.com/widgets.js"
charset="utf-8"></script>
<?php } ?>
<?php if (isset($entry['Mermaid'])) { ?>
        <script src="https://.../mermaid.min.js"></script>
        <script>mermaid.initialize({startOnLoad:true});</script>
<?php } ?>
<?php if (!isset($pagination) && isset($entry['prismjs'])) { ?>
        <script src="/jscss/prism.js"></script>
<?php } ?>

</html>
```

Performance comparison

Generator	#posts	real x86	user x86	real ARM	user ARM
S. Saaze	13	0.04	0.02	0.09	0.06
S. Saaze	347	0.15	0.09	0.37	0.25
S. Saaze	681	0.26	0.17	0.64	0.41
S. Saaze	1349	0.47	0.36	1.32	0.86
Hugo	13	0.22	0.31	0.76	1.47
Hugo	347	1.53	4.5	4.39	17.73
Hugo	681	2.43	7.34	6.76	29.88
Hugo	1349	3.87	12.41	11.83	52.08
Zola	13	0.08	0.06	0.27	0.23
Zola	347	0.36	0.31	1.16	1.1
Zola	681	0.6	0.58	2.05	1.99
Zola	1349	1.17	1.09	3.75	3.67

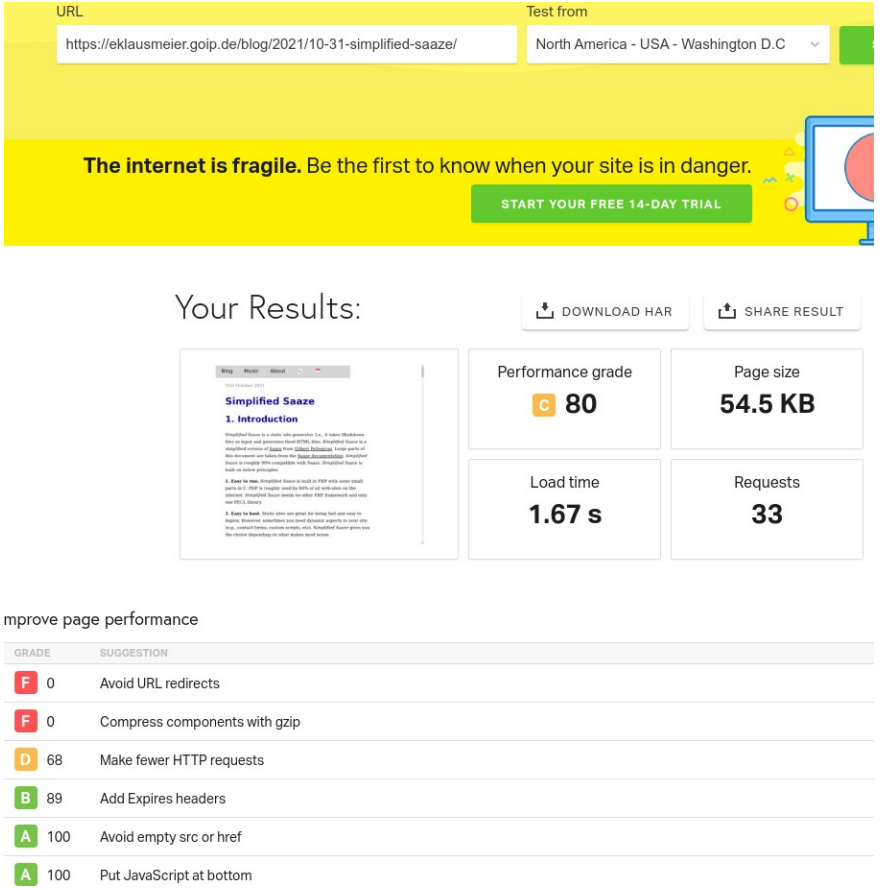
Performance comparison #2





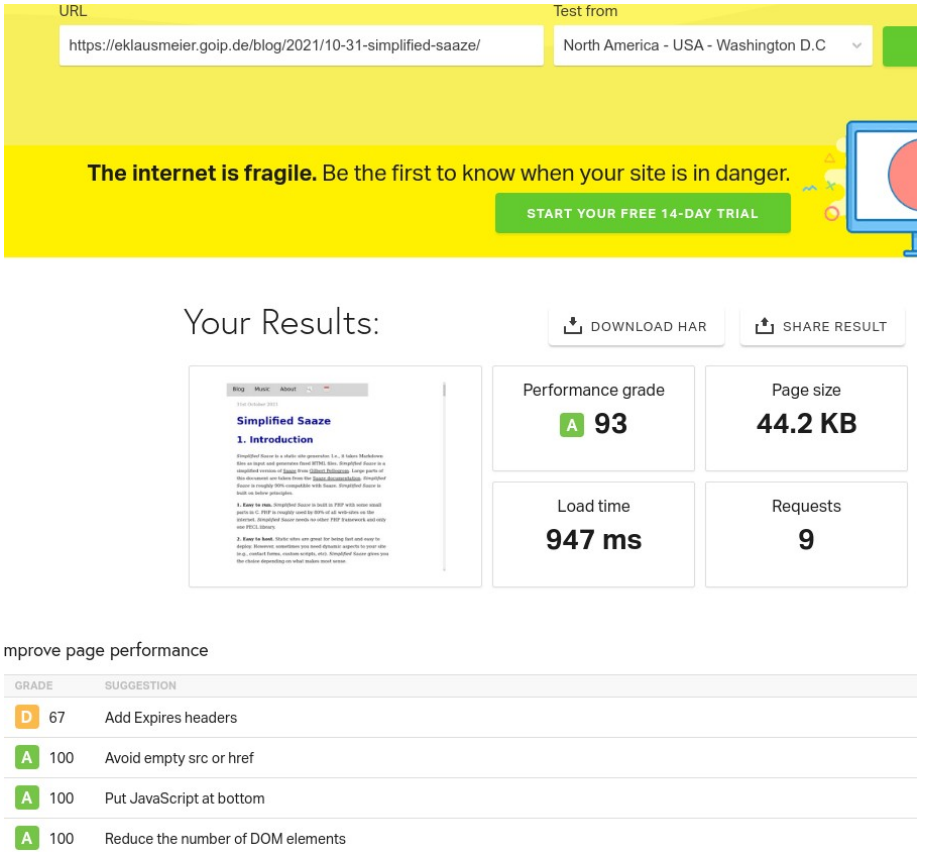
Reducing Requests #1

- Reduce number of requests the web-browser has to do to fully load your page



Reducing Requests #2

- Downloading JS library, bundling it into one file



End

References

- Gilbert Pellegrom: https://saaze.dev
- Simplified Saaze: https://eklausmeier.goip.de/blog/2021/10-31-simplified-saaze
- PHP-FFI: https://phpconference.com/blog/php-ffi-and-what-it-can-do-for-you
- XHProf: https://www.php.net/manual/en/book.xhprof.php
- Yaml in PECL: https://pecl.php.net/package/yaml
- yaml_parse(): https://www.php.net/manual/en/function.yaml-parse
- Blade templates: https://laravel.com/docs/8.x/blade
- Open Source PHP Template Engines: https://ourcodeworld.com/articles/read/847/top-7-best-open-source-php-template-engines
- Pingdom tests: https://eklausmeier.goip.de/blog/2021/11-16-accelerating-page-load-times-by-reducing-requests
- Saaze vs. Hugo vs. Zola: https://eklausmeier.goip.de/blog/2021/11-13-performance-comparison-saaze-vs-hugo-vs-zola