# *Why everyone hates PHP*

*(survey project)*

Wangcheng Yuan ,Sida zhong

Computer science dept. Class 252

San Jose State University

San jose, United States

wangcheng.yuan@sjsu.edu

sida.zhong@sjsu.edu

***Abstract*—PHP has always been a very popular scripting language, Approximately 25% of the world's programming languages are using it. When the browser was born in 1996, PHP had a very high reputation, and countless websites were born."PHP is the best language in the world", this is how the earliest official document was written.But after 2015, PHP's popularity began to decline, and more and more people found that PHP has serious hidden problems.What was worse, many people began to question the necessity of PHP. Many people predict that PHP will disappear in the next 10 years.However, PHP survived the doubt. In the past few years, countless languages have disappeared, but not only did PHP in 2021 not disappear, it still occupies nearly 80% of the Internet market.In this research paper, It will first introduce the history of PHP, PHP's security issues, then test the performance of PHP, discuss some new improvements in PHP, and finally look forward to the future of PHP.**

***Keywords*—*component; PHP history; Security issues; Performance and testing; PHP improvements; PHP future;***

## I PHP history

In 1995, the newly born PHP was not a backend language, or not even a programming language. Rasmus Lerdorf, the creator of PHP, said in many speeches that his original intention of designing PHP was a simple HTML tool. However, PHP's flexible features and simple learning curve quickly catched the eyes of a large number of programmers. More and more people were devoted to the development of PHP, and the requirements of PHP were also increasing day by day. Three years later, two Israeli engineers, Zeev Suraski and Andi Gutmans, re-written the PHP compiler in C and named it Zend Engine. So far, PHP has become a very successful and flexible front-end language, which can add dynamic interactive operations to HTML static websites. After fastCGI was applied to PHP, PHP was becoming a background language. PHP-fastCGI is a communication protocol that can connect PHP script files and web servers like nginx and apache. In 2004, Andrei Nigmatulin developed PHP-fpm. PHP-fpm is a process management system used to coordinate the process of the PHP-fastCIG communication protocol. This greatly improves the stability between PHP and the web server. PHP ushered in its most glorious era.

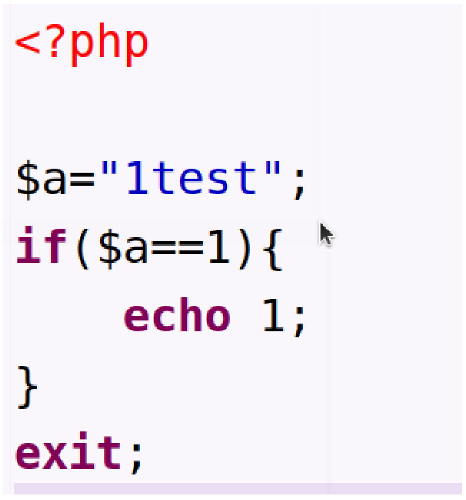
After 2010, due to the rise of smartphones, people started to focus on small programs on mobile phones, and the entire market demand for websites began to shrink. At this time, the problem of PHP was exposed. Since PHP was born on the Internet, PHP can only be used in the website application. In other fields of development, PHP will not be used at all. This is completely different from some universal programing languages like JAVA. JAVA has always occupied a place in network development, and it is also very handy in mobile phone development and other fields. This makes JAVA's libraries and communities very active, which is one of the reasons why JAVA is still popular for so many years. After 2015, the rise of artificial intelligence has further compressed the living space for website development. More and more people will choose to learn Python or some languages related to system and intelligent development, to catch up with the trend of the new era. And website development has been considered a low-tech job. So the whole history of the rise and fall of PHP is actually the history of the website.

Due to the rapid development of PHP in the early days, many low level structures were not well designed and expanded, which carried a heavy historical burden. For example, some of PHP internal functions look normal (strpos()), some use underscores (str_replace()), and some use numbers (nl2br()), some inverted nouns and verbs (substr(); strrev();). The PHP code is not elegant at all. Nowadays, many potential PHP problems are not unsolvable, but if solved, many small and medium website companies will be greatly impacted. These companies do not have enough money and time to refactor the code. PHP is now in a very embarrassing stage, stuck in the wall of the website, unable to reinvent itself to catch up with the trend of the new era.

## II Security issues

The PHP language itself does not have any loopholes, and I don't believe that any language has security loopholes. Because PHP is a dynamic language type, it has some strange features. If someone does not understand these factors, strange bugs or security risks will occur. These are man-made results, not PHP internal functions or compiler vulnerabilities.

Fig. 1. Unsecurity code



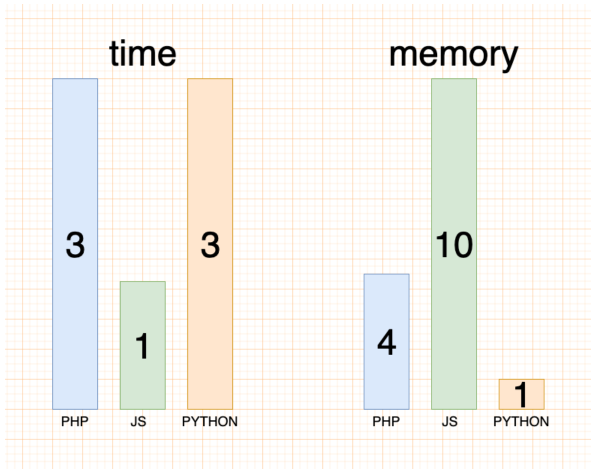
```
<?php

$a="1test";
if($a==1){
    echo 1;
}
exit;
```

To give a most common example, in Figure 1, although the variable a is a string, and the condition of if statement is an integer, the program will still pass the statement, and the final output result is "1". This is because when the variable types are different, PHP will automatically intercept the first string as the condition. The solution to this situation is to use "=== ", the conditional statement will determine the variable type. Not only PHP has this problem, javascript also has the same problem, all dynamic languages have this problem. Since PHP is very popular on websites, many problems are misunderstood as only occurring in the PHP language. And these security issues are very easy to solve.

In the first 100 popular github PHP projects[1], there is no SQL injection problem at all, which is almost as I expected. MVC framework [2] is used for most of PHP now. These frameworks strictly block front-end HTML rendering and back-end data control, and SQL injection will not happen at all. Currently, the most attacked PHP website is XSS (Cross Site Scripting), which accounts for more than half. Most XSS attacks occur on websites that allow users to upload code, such as custom blogger sites. Hackers will upload JS or other scripting language codes. When other users browse the hacker's personal homepage, the illegal code will be automatically executed to steal the current user's browser information. Of course, the solution is also very simple, as long as you filter illegally uploaded codes. Research also shows that the activities of the PHP project are not directly related to security vulnerabilities. This shows that even PHP open source projects that many people participate in still have errors. I think it is because the focus of PHP development projects is demand and speed, not maintenance. PHP can develop projects faster than any language, which is also one of the biggest reasons for vulnerabilities.

## III Performance and testing

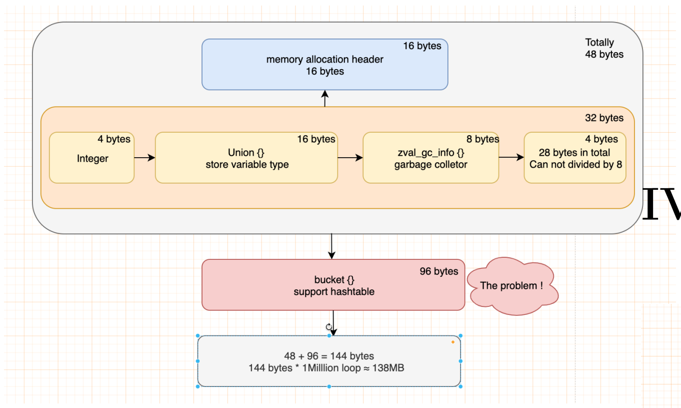
Fig. 1. Unsecurity code



- This test compares the performance of PHP, JS and Python in parsing 100MB json files. The result is that the performance of PHP is very poor [3]. As shown in Figure 2, PHP neither takes advantage of space nor time, in other words, it is very low cost performance. For example, the execution speed of JS is more than 3 times that of PHP, but the memory consumption is only twice that of PHP. The execution speed of Python is as slow as PHP, but the memory consumption of Python is only one-fourth that of PHP. The prerequisite for this test is to use the language version before 2017, the tested PHP version is 5.6, nodejs is 10, and Python is 2.3. After testing with the new PHP7.2 and node14, PHP is 1.4 times faster than JS, and its memory is 2 times faster than JS. Very surprised, PHP7.2 has surpassed the running speed of JS in some respects. JS has always been considered the fastest dynamic language before. The performance of PHP7 has been greatly improved, which will be introduced in the future of PHP.

- This test compares the comprehensive performance of PHP and ASP. The result is that the total ASP test score is higher than PHP [4]. This test measures from 5 aspects, static web page loading speed, algorithm processing, database management, file upload and file reading and writing. ASP's performance in file upload and database operations falls behind PHP, and wins in all other aspects. Especially in terms of algorithm speed, it is 20 times ahead of PHP. Is ASP really better than PHP? I have a different view. I think in website development, the database is the most important indicator and the biggest bottleneck. The website is different from other personal applications. The website is mainly to display information, This leads to the biggest bottleneck that the server reads and writes to the database during periods of high concurrency. The algorithm of the website is usually relatively simple, even if there is a complicated algorithm, it can be solved by other methods such as cronjob tasks, or memcache delay. So the least important aspect is algorithm processing. Although the speed of static page ASP is ahead of PHP, the loading speed of static pages is inherently very fast, and it is usually supported by DNS static caching support. ASP running speed leading does not bring much advantage, but the disadvantage of the database is fatal. All in all, I think ASP is ahead of PHP in overall performance, but falls behind PHP in network performance.

Fig. 3. How PHP allocated its memory


Totally 48 bytes
16 bytes
memory allocation header 16 bytes
32 bytes
4 bytes
Integer
16 bytes
Union {} store variable type
8 bytes
zval_gc_info {} garbage colletor
4 bytes
28 bytes in total Can not divided by 8
96 bytes
bucket {} support hashtable
The problem !
48 + 96 = 144 bytes
144 bytes * 1Milllion loop ≈ 138MB

- This test compares the usage and allocation of large memory by PHP, JS and Ruby. This test was shown in the presentation, and the result is that PHP uses 4 times more memory than JS, and the execution time is about 8 times slower. Ruby's memory usage is the same as JS, and it runs 3 times slower than JS. JS has a big win in array allocation. This test exposed the biggest problem in early versions of PHP, and it is also one of the most direct reasons for PHP's slow speed. PHP arrays are processed by hashtable in the compilar. As shown in Figure 3, based on the dynamic language of C compilar, variables need extra space to store the variable type, and also need to store the address allocated by the memory. However PHP uses the hashtable data structure at the end, which doubles the original 48byte memory space. PHP array is very powerful, it can directly convert the array into a hash table without any declaration. This is actually one of the major features of PHP, combining array and hashtable into an associated array. Not all situations need to be handled by hashtable, which is very inefficient. In the new PHP7, this problem has been resolved. The solution is rather obscure. The overall concept is to cancel the data structure of the doubly linked list, and the new method adopted is to use pointers for small memory allocation and sorting. This aspect involves too much low-level knowledge, which is beyond my comprehension.

## IV PHP improvements

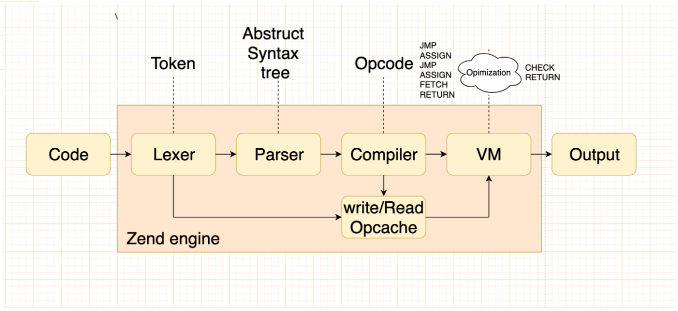
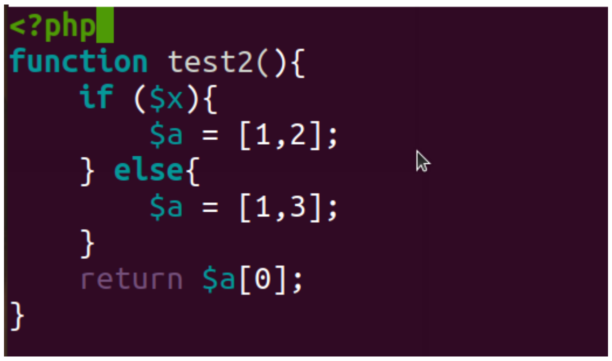


Fig. 4. PHP compiler

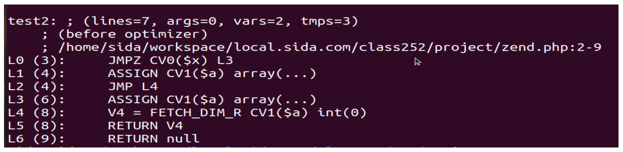
PHP7.2 was released at the end of 2017, and its performance has been greatly improved, as mentioned in the chapter on performance and testing. The PHP compiler is C, which was first completed in 1998 by two Israeli engineers, Zeev Suraski and Andi Gutmans, and named it Zend Engine. The new version of PHP not only redesigns the compilar structure of hashtable, but also improves the performance of compilar. As shown in Figure 4, its newly added opcache module can detect whether there is repeated code execution. If there are duplicate codes, The compiled opcode in the memory will directly call the VM (virtual machine) to execute, skip it through the lexical analyzer, parser and compiler, thus greatly improving the running speed.

Fig. 5. Bad PHP code


```
<?php
function test2(){
    if ($x){
        $a = [1,2];
    } else{
        $a = [1,3];
    }
    return $a[0];
}
```

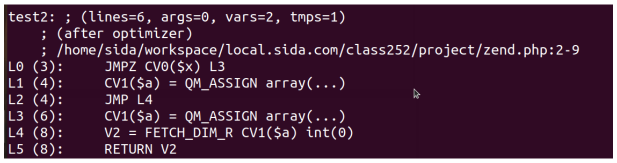
In his speech in May 2018 [6], the creator of PHP also introduced a new concept called DCE (dead code elimination). This technology can greatly reduce the compiled opcode, which means that the execution speed of the VM can be directly accelerated. As shown in Figure 5, test2() is a very "bad" function, probably written by a PHP programmer. The variable $ x in the if condition is undefined, so the following code will never be executed. The array $ a is defined twice. The return result is the first element of the array $ a, which means that no matter what the situation, the final result will only return 1.

Fig. 6. Unoptimized opcode


```
test2: ; (lines=7, args=0, vars=2, tmps=3)
    ; (before optimizer)
    ; /home/sida/workspace/local.sida.com/class252/project/zend.php:2-9
L0 (3):     JMPZ CV0($x) L3
L1 (4):     ASSIGN CV1($a) array(...)
L2 (4):     JMP L4
L3 (6):     ASSIGN CV1($a) array(...)
L4 (8):     V4 = FETCH_DIM_R CV1($a) int(0)
L5 (8):     RETURN V4
L6 (9):     RETURN null
```

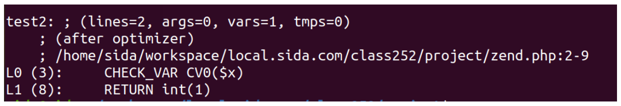
As shown in Figure 6, this is an unoptimized opcode with a total of 7 lines.

Fig. 7. PHP7.2 optimized opcode


```
test2: ; (lines=6, args=0, vars=2, tmps=1)
    ; (after optimizer)
    ; /home/sida/workspace/local.sida.com/class252/project/zend.php:2-9
L0 (3):     JMPZ CV0($x) L3
L1 (4):     CV1($a) = QM_ASSIGN array(...)
L2 (4):     JMP L4
L3 (6):     CV1($a) = QM_ASSIGN array(...)
L4 (8):     V2 = FETCH_DIM_R CV1($a) int(0)
L5 (8):     RETURN V2
```

As shown in Figure 7, this is an opcode optimized by the PHP7.2 version, with a total of 6 lines.

Fig. 8. PHP7.3 optimized opcode


```
test2: ; (lines=2, args=0, vars=1, tmps=0)
    ; (after optimizer)
    ; /home/sida/workspace/local.sida.com/class252/project/zend.php:2-9
L0 (3):     CHECK_VAR CV0($x)
L1 (8):     RETURN int(1)
```

As shown in Figure 8, this is an opcode optimized by the PHP7.3 version, with a total of 2 lines. The first line checks the variable $ x, and the second line returns the integer 1. It can be seen that the compiler of PHP becomes smarter and smarter as the version improves.

## V PHP future

Facebook introduced the static variable language HACK based on PHP syntax in 2014 [7]. But a few years later, this language did not cause great repercussions. The reason is most regular PHP projects have good code optimization, problems caused by dynamic variables are almost zero. No one will define a string variable and then add an integer after it. This makes the advantage of changing PHP to a static language very small. Even if the code of static variables is refactored, there is no big difference, but the cost of refactoring the code is high. Many programmers have adapted to dynamic languages, and programmers, like the PHP language, are also making progress. This brings up the biggest question, will PHP be replaced by other languages? I think it is not within the next 10 years. I think PHP is the only language most suitable for website development. Because of the unique development model of the website, its requirements change very quickly, so a flexible dynamic language is needed to control it. ASP and Ruby are also languages for website development, but they have been declining sharply over the years, so they will not pose a threat to PHP. The most threatening is the rising trend of Python and Javascript. Javascript is more of a client-side language, and frameworks like VUE, ANGULAR, and REACT are all client-side logic. Early Jquery and Ajax are also on client-side. The Python community paid more attention to artificial intelligence and operating system-level applications. Only PHP is completely dedicated to website development. I mentioned at the beginning that PHP can only be applied to website development, which is actually both a disadvantage and an advantage. The dramatic increase in PHP's performance also shows that PHP has a stable and strong community behind it. Many languages have disappeared silently, but the main reason why PHP can go to this day is "everyone hates PHP".

# Bibliography

[1] A. Ibrahim, M. El-Ramly and A. Badr, "**Beware of the Vulnerability! How Vulnerable are GitHub's Most Popular PHP Applications**?," 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), Abu Dhabi, United Arab Emirates, 2019, pp. 1-7, doi: 10.1109/AICCSA47632.2019.9035265.

[2] S. I. Adam and S. Andolo, "**A New PHP Web Application Development Framework Based on MVC Architectural Pattern and Ajax Technology**," 2019 1st International Conference on Cybernetics and Intelligent System (ICORIS), Denpasar, Indonesia, 2019, pp. 45-50, doi: 10.1109/ICORIS.2019.8874912.

[3] H. K. Dhalla, "**A Performance Analysis of Native JSON Parsers in Java, Python, MS.NET Core, JavaScript, and PHP**," 2020 16th International Conference on Network and Service Management (CNSM), Izmir, Turkey, 2020, pp. 1-5, doi: 10.23919/CNSM50824.2020.9269101.

[4] K. Bounnady, K. Phanthavong, S. Pathoumvanh and K. Sihalath, "**Comparison the processing speed between PHP and ASP.NET,**" 2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Chiang Mai, Thailand, 2016, pp. 1-5, doi: 10.1109/ECTICon.2016.7561484.

[5] Nikita Popov, "**How big are PHP arrays (and values) really? (Hint: BIG!)**" 2011, https://www.npopov.com/2011/12/12/How-big-are-PHP-arrays-really-Hint-BIG.html

[6] Coding Tech, "**PHP in 2018 by the Creator of PHP**", https://www.youtube.com/watch?v=rKXFgWP-2xQ& t=45s

[7] L. Eshkevari, F. Dos Santos, J. R. Cordy and G. Antoniol, "**Are PHP applications ready for Hack**?," 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Montreal, QC, Canada, 2015, pp. 63-72, doi: 10.1109/SANER.2015.7081816.