A Comparative Performance Analysis of PHP with and without JVM

Abstract: The field of programming language is evolving rapidly.New technology and the burgeoning demand of several application platforms is behind the amelioration of programming languages.Variation in the implementations and applications of programming languages demands the performance measure attribute among them.JPHP , which is simply PHP on the Java VM , was introduced to address the discrepancy of Zend PHP.It's rich features includes JIT, Multi-Threading, Unicode, GUI, Android and Embedded Web Applications.It claims it's high performance 1x - 10x faster than PHP 5.6,PHP 7 because of JIT and Optimizer.Having a new, elegant and powerful API with better standard library, The PHP language can be used not only for web apps but also for GUI, Game, Android.In this report,we have run a comparative study of the performance analysis of JPHP with traditional PHP that runs on server.For this purpose we have explored a few existing benchmarks and in particular, we have found that JPHP is faster than traditional JPHP , as claimed.

Additional Key Words and Phrases: JVM, Bytecode, JPHP,Zend,Benchmark

ACM Reference Format:

1 INTRODUCTION

Benchmark is a standard or point of reference against which things may be compared or assessed.Using benchmark for measuring the performance of any programming languages is a common practice. For a long course of time, PHP is considered as a pragmatic language for server side programming. It was created to fulfill specific needs for quickly making web pages (the name originally stood for Personal Home Pages) and the language was extended as required.The authors of PHP did not intend PHP to become a new programming language, but it grew with time. Features like correct handling of foreign characters / Unicode characters are obviously added on afterwards and not cleanly integrated with the rest of the language.Despite having proper specification until now,it is extremely popular for it's easiness but also criticized for it's bad design.

JVM is the main component of Java architecture and it is the part of the JRE (Java Runtime Enviroment). It provides the cross platform functionality to java.Any code written to target the JVM will run on any platform where JVM is available. In most cases, compilers produce code for a particular system but Java compiler produces code for a virtual machine .Thus, JVM provides security to java.

Dmitriy Zayceff, who is a Russian, is the main developer of JPHP. According to author[12] of JPHP , *JPHP is not a replacement for the Zend PHP engine or Facebook HHVM. We don't plan to implement the zend runtime libraries (e.g. Curl, PRCE, etc.) for JPHP.* Their project started in October 2013.HHVM's idea is to compile PHP as an intermediate bytecode and then compile the bytecode into x64 machine code via JIT.

2017. XXXX-XXXX/2017/9-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The main reasons for initiating the project were as follows:

- It was an experiment.
- Using java libraries in PHP.
- Upgrading performance via JIT and JVM
- Replace the inconsistent and ugly Zend PHP runtime with something more decent.
- Allow to write on PHP not only under the Web
- Implement Unicode support and multithreading.

JPHP is quite rich in features, which include :

- PHP 5.6+ (some language features from PHP 7).
- JIT (~ 2.5 faster PHP 5.6, ~1.1 faster PHP 7), Optimizer
- Using java libraries and classes in PHP code.
- Unicode for strings (UTF-16, like in Java)
- Threading, Sockets, Environment architecture (like sandbox objects in the runkit zend extension).
- GUI (Swing or JavaFX)
- Embedded cache system for classes and functions
- Optional Hot Reloading for classes and functions

JPHP has own elegant standard library with many classes and functions for all.

In the later sections of the report we have covered the following things. In Section 2 we have discussed the motivation of our work for better understanding of this paper easily. Methodology,approach adopted is conferred in Section 3 and Section 4 has the full details of topics discussed in Section 3.In section 5 our design and implementation procedure of our work,experimental setup environment and findings from the work is narrated elaborately. Section 6 summarizes our findings and future works in this respect. Section 7 concludes the report.

2 MOTIVATION

The basic approaches to implement a programming languages include building VM from scratch or using existing VM. Following the later approach various JVM languages have been designed such as - JRuby and Jython etc[10]. PHP implementations on JVM have been around earlier like Quercus,p8,projectzero .The authors of the project, Quercus, stated that their implementation works at the same speed as Zend PHP + APC. Up to a certain time, there were other PHP implementations for JVM ,but previous ones have fizzled out in favour of the Zend Engine. Another PHP compiler for JVM has born recently: JPHP[12].JPHP compiles PHP sources to JVM bytecode and then can execute the result on the JVM. It doesn't have libararies like PDO , CURL but it enables us to write familiar PHP language syntax for creating applications that run on the JVM, along with the facility to use Java libraries and classes in the PHP code as well as executing it . In one of the features of JPHP includes, it is faster than both PHP 7, and PHP 5.6 .Their website did mention some performance benchmarks , but no proper documents were available to support their claim. So, We decided to evaluate both PHP with and without JVM .Our intend is to test both setup and see the performance whether it support their claim or not.

3 METHODOLOGY

JPHP is quite a new JVM Language. It has it's own website [2] as well as own news twitter [4]. In1, we can see the performance comparison posted by the Author of JPHP.

To evaluate JPHP with PHP, we tried to find if any analysis is done on this topic. The github repo of JPHP[12] does refer to some performance benchmarks , but the benchmarks result wasn't

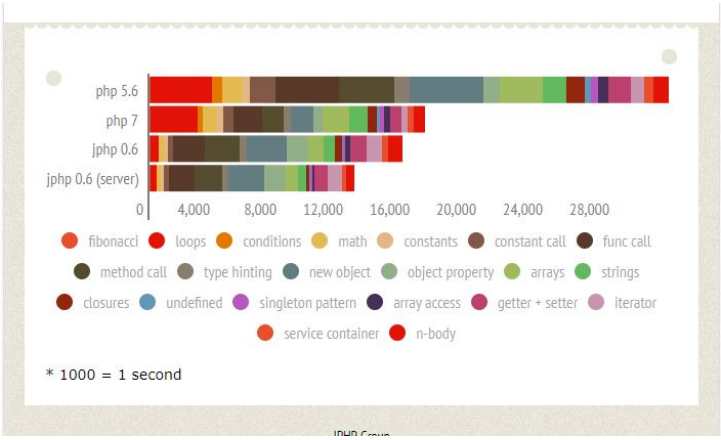

Fig. 1. JPHP vs PHP benchmark[3]

available. We did get access to a set of benchmarks,which were used to test performance of JPHP.A performance analysis of JPHP vs PHP 7 can be found here along with environments and results [7],which is inspired from JPHP provided benchmark with source.We also found hnw's diary[5],Where Zend Benchmarks[8] were used for performance Analysis.This analysis includes PHP 5.6, PHP 7 ,Phalanger,Quercus and JPHP.Hawana didn't just analyze the performance , he also wanted to contribute to the project by making several pull requests.

After taking a tour to existing benchmark testing , we are motivated to analyze PHP and JPHP on our own. For this to happen , we wanted to create a Benchmark set- which combines existing benchmarks as well as our implemented benchmark.For inspiration to implement our benchmark , we explore some of PHP benchmarks,which includes Maettig's collection of benchmarks[6],The PHP benchmark[1] . We also viewed the article of Christian Vigh[9], which discussed about the evolution of PHP over 21 years. This article also included a comparison of pure CPU benchmark results among PHP versions.

To start a new project we followed the instructions given in [2], in Getting started page. To collect a test project , we were needed to install a build tool Gradle in our PC.

4 DIFFERENT TECHNIQUES & OPTIMIZATION

The entire language of JPHP is written from scratch in Java using the ASM library to generate bytecode, it is used by all popular JVM languages, for example Groovy. Gradle was chosen as the assembly system.

The questions arises,Why JPHP doesn't support many functions and classes from original Zend PHP? Answer to this question comprises several reasons, that are stated as follows:

- Zend PHP classes and functions are poorly designed.
- It's one of the project goals - replacing the ugly zend runtime library with a better runtime library.
- To implement this support, it's necessary to spend a lot of time and effort.
- Many PHP functions are too universal and complex to implement them.

The Java Virtual Machine (JVM) is quite a powerful tool. How the JVM bytecode works can be found from the documentation for the ASM library. The capabilities of the VM can be described briefly as follows:

- Virtual stacking machine.
- It is possible to store local variables by indexes (something like registers).
- GC (garbage collector) implemented at the VM level .
- Objects and classes implemented at the level VM .
- A large number of standard operations - POP, PUSH, DUP, INVOKE, JMP, etc.
- For Try Catch there are bytecode instructions, for finally - partially .
- For VM there are several types of values: int32, int64, float, double, objects, arrays of scalars, arrays of objects, for bool, short, byte, char int32 is used.

The Java technology stack provides very convenient conditions for writing a JVM language. We do not need to write our VM with JIT, the garbage collector and the class system are already implemented, the head does not have any problems with cross-platform, and the bytecode JVM itself is very easy to understand. JIT allowed to increase productivity by 1-10 times, depending on the tests, on average 1.5-3 times on real code.

The main problem with PHP performance is the global space for variables, the magic of variables, just magic when you can access a variable by name at runtime. For this reason, JPHP can compile the same code in different ways. Whereas the truth is, there is no magic with variables, the compiler just converts variables into indices and at runtime will immediately access them by index.

For the sake of getting idea about different benchmarks,we explored JPHP[7],which is performance test(PHP 7 vs JPHP) using the official JPHP benchmark.Here they have used Java version "1.8.0_92" and PHP version 7.0.6. The result is as given below:

- PHP7: CLI test: 11.1s
- PHP7: Apache test: 11.3s
- jPHP: java client test: 16s
- jPHP: java server test: 13s

Further in this work, in the diary of HNW [5] ,we found PHP source code benchmark test included Zend bench[8] run by the author.

	PHP 7	PHP 5.6	Phalanger	Quercus	JPHP
simple	0.094	0.112	0.069	0.195	0.058
simplecall	0.028	0.116	0.019	0.179	0.009
simpleucall	0.054	0.112	0.021	0.212	0.061
simpleudcall	0.053	0.116	0.021	0.229	0.075
mandel 2	0.347	0.356	0.584	0.769	0.315
ackermann (7)	0.078	0.140	0.044	0.264	0.132
ary (50000)	0.008	0.023	0.025	0.097	0.073
ary2 (50000)	0.008	0.019	0.017	0.044	0.049
ary 3 (2000)	0.136	0.152	0.338	0.394	0.138
fibo (30)	0.181	0.374	0.151	0.609	0.183
hash 1 (50000)	0.017	0.029	0.116	0.114	0.077
hash 2 (500)	0.017	0.035	0.073	0.062	0.105
heapsort (20000)	0.067	0.092	0.116	0.310	0.216
matrix (20)	0.069	0.086	0.136	0.164	0.100
nestedloop (12)	0.156	0.199	0.115	0.256	0.076
sieve (30)	0.041	0.089	0.114	0.151	0.089
strcat (200000)	0.010	0.013	0.013	0.041	0.033
Total	1.363	2.061	1.969	4.089	1.787

Fig. 2. Run 1

Although JPHP was slower than PHP 7 , it was equal to or slightly faster than PHP 5.6 / Phalanger here in the 2. This is because it includes time to compile Java bytecode and JIT compilation . In case of the operation on the Web server , such processing can be done in advance. For measuring

the true ability, the author called the benchmark function twice and measured only the second execution time Saw.

	PHP 7	PHP 5.6	Phalanger	Quercus	JPHP
simple	0.090	0.109	0.061	0.128	0.026
simplecall	0.027	0.117	0.018	0.125	0.004
simpleucall	0.050	0.115	0.019	0.170	0.051
simpleudcall	0.052	0.113	0.019	0.235	0.036
mandel 2	0.336	0.336	0.575	0.770	0.205
ackermann (7)	0.074	0.133	0.041	0.208	0.113
ary (50000)	0.008	0.020	0.018	0.015	0.022
ary2 (50000)	0.006	0.018	0.020	0.020	0.033
ary 3 (2000)	0.112	0.165	0.332	0.362	0.156
fibo (30)	0.179	0.349	0.147	0.524	0.185
hash 1 (50000)	0.017	0.030	0.116	0.034	0.035
hash 2 (500)	0.016	0.033	0.066	0.047	0.038
heapsort (20000)	0.067	0.087	0.105	0.180	0.116
matrix (20)	0.069	0.083	0.127	0.131	0.068
nestedloop (12)	0.158	0.190	0.108	0.244	0.082
sieve (30)	0.040	0.088	0.106	0.111	0.038
strcat (200000)	0.010	0.013	0.012	0.018	0.006
Total	1.309	2.000	1.891	3.321	1.213

Fig. 3. Run 2

Here, 3 in the second run,the code is already pre-compiled and as expected JPHP wins. It was the best result when JPHP looked at the total of all the tests.

In [6], some micro benchmarks were also explored . The author created this comparison to learn something about PHP and how the PHP compiler works.These benchmarks are advised not to use in comparison of PHP versions.

PHPBench.com[1] was constructed as a way to open people's eyes to the fact that not every PHP code snippet will run at the same speed.

The purpose of article [9], is to learn how performance improved across the latest PHP versions starting from PHP 5 up to the latest developments, including the recent version 7.1 with opcache optimization, as well as the experimental JIT branch that will be become part of PHP 8 or PHP 7.2 the next version. The experimental JIT branch mentioned in the article is JPHP,when the article was written it was still experimental. The author of JPHP also contributed to clarify and review information presented in this article, so it is clear and accurate.

The get-started project, downloaded from [2], has this directory structure-

```
Get-Started
├── build.gradle
└── src
    ├── Bootstrap.php
    └── JPHP-INF/
        ├── .bootstrap.php
        └── launcher.conf
```

We can any php code in the `JPHP-INF/.bootstrap.php` Such as

```
<?php echo "Hello World";
```

After manually configuring buld.gradle, we need to use the command line to run your app : **Gradle run**

5 EXPERIMENT DESIGN AND PERFORMANCE ANALYSIS

This section presents the experimental design that we used to evaluate JPHP and PHP.

Experimental Setup

We use the following experimental methodology.

Benchmarks. We draw 25 benchmarks -which combines the Zend Benchmark[8], JPHP benchmarks[7], own inspired benchmarks.

The benchmark consists of:

- Fibonacci -simple fibonacci function with recursion.
- Loops -it's general language features like " for, foreach, while "
- Conditions - it's general language features like "if, else,elseif, switch-case, ".
- Math - testing performance of math operators and functions.
- Constants - performance of getting simple constants and class constants.
- Type hinting - performance of calling methods and functions with type hinting.
- New object - performance of creating new instances of classes.
- Arrays - performance of arrays, adding, foreach-ing, creating, etc.
- Strings - performance of string operations and basic functions.
- Object property - performance of getting/setting object properties.
- Closures - performance of closures.
- Func call - performance of calling functions.
- Method call - performance of calling dynamic and static methods.
- Design Patterns - performance of singleton,
- getter + setter - getter and setter in a class
- Ackerman - the simplest example of a well-defined total function where its value grows rapidly, even for small inputs.
- EmptyString -performance of all ways of checking if a string is empty
- Encryption - performance of Caeser and Vigenere Encryption
- Matrix - simple matrix operations
- NBody -Nbody simulation
- Prime - check a number is prime
- Quicksort - PHP implementation of popular sorting algorithm, in worst,best and average case
- Regex - Utilizing the regular expression evaluation in PHP and JPHP
- Array Searching - Binary and linear search
- Mandal - Mandal's second law

We omit two *Mandal* bench from our analysis as it was causing the script to hang, but keeping it in benchmark class for completeness.Among the benchmarks we found Ackerman and Encryption to contribute most in total runtime, due to it's complex nature.

Hardware and Operating System. We use three hardware platforms:

(1) Processor Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz, 2400 MHZ, 2 Core(s), 4 Logical Processor(s)
(2) Processor Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz, 3201 MHZ, 4 Core(s), 4 Logical Processor(s)

(3) Processor Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz, 2501 Mhz, 2 Core(s), 4 Logical Processor(s)

We used Windows 10 Pro distribution and a 64-bit (x86 64) in all platforms .

Gradle. For this project,We downloaded and installed the Gradle distributive.Gradle is simply a distribution plugin. It is a build tool which replaces XML based build scripts with an internal DSL which is based on **Groovy programming language**.The **Launcher** in Gradle uses a special class loader to load all the classes from the **src** directory.The Java class file contains Java bytecode (highly optimized set of instructions) which is executed by Java Virtual Machine (JVM).The class loader in Gradle loads any PHP files in the **src** directory automatically and this class files are converted into bytecode which is run by JVM.

Other Experiment Setup. We have used **xampp** as apache server. We have run these scripts against PHP 7.1and PHP 5.6. We run the bench code several times, and took the minimum time. IntelliJ IDEA is used as IDE for bulding Gradle Project, which we have downloaded from [2] as a starting point of our experiment.

The java and PHP verisons of each platform are given below-

(1) For Platform 1 a.k.a PC1

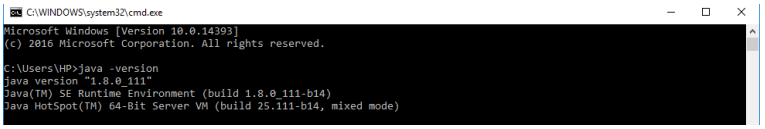

Fig. 4. Java configuration of PC1

We have run 2 Apache Servers in PC1. Their configurations are -

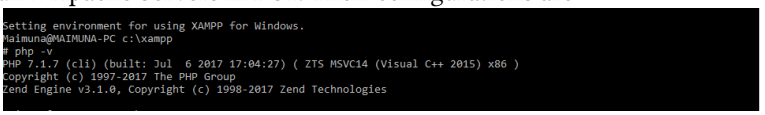

Fig. 5. PHP 7.1 in PC1

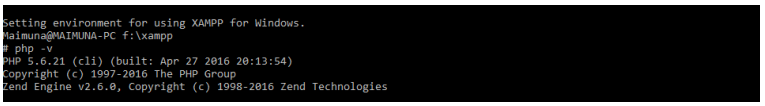

Fig. 6. PHP 5.6 in PC1

(2) For PC2

Fig. 7. Java configuration of PC2

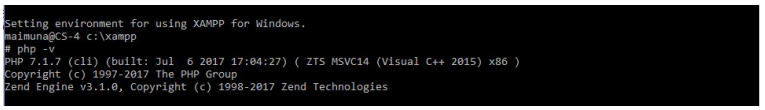

Fig. 8. PHP 7.1 in PC2

(3) For PC3

Fig. 9. Java configuration of PC3

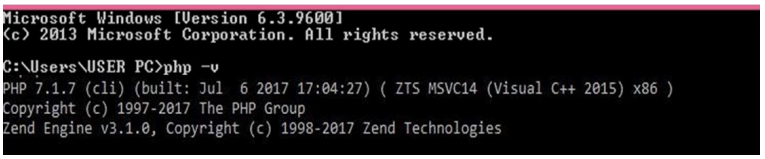

Fig. 10. PHP 7.1 in PC3

Performance Analysis

This section analyses the results of our experiment.Our sole purpose of this experimentation was to achieve first-hand experience to witness the effectiveness of JVM. With this in mind ,we have designed our benchmark which consists of both complex and simple procedures.

In this experiment, To get the results simultaneously,we have executed a '.bat' file which consists of simple loop for executing a script multiple times. Script outputs were written into a file . After several executions, We can estimate the total runtime of the bench codes. The number of iterations inside each bench code also effect the run time - as input size increases, the execution time also rises. To achieve the effective output we have kept the number of iterations moderate.

Table 1 shows the execution time of each benchmark as well as the total time which is measured in *seconds*. In PC1, more than one PHP servers were used for two versions of PHP (PHP 7.1 and PHP 5.6) and JPHP beats them both.Other two platforms also follow these results.These results indicate that our experiment model was well-designed and well-thought, And for this we are getting faster execution in JPHP.In all platforms we can observe that, the claim that JIT (~ 2.5 faster than PHP 5.6, ~ 1.1 faster than PHP 7) in JPHP is true .But if we look at the individual benchmarks, and not the summed benchmark,then PHP7 seems to be quicker in a lot of important scenarios (method calls, object creation...) ; and has better feature support. From [5] , we already know that there is some start up delay in the first run of JPHP beacuse of JVM.So, to keep the comparison simple we have avoided the result of first time run of JPHP code.

	Benchmarks	PHP 7.1(ms)	JPHP(ms)	PHP 5.6(ms)
	Fibonacci (25)	26	16	66
	Prime number (100)	1880	1298	2987
	Ackermann Function(3,3)	1893	2143	5453
	Loop (1000)	444	251	780
	Math Methods (100)	4370	8368	10405
	Empty String Check (25)	11	31	39
	Control Structure (100)	279	79	500
	String Manipulation (25)	1174	389	1170
PC 1	Array Search(linear,binary)(10)	60	124	175

	Quick Sort (1)	341	189	846
	Regex (25)	1094	2148	4224
	Fetch constants (18)	212	201	256
	Closure (100)	483	204	771
	Getter_Setter (8)	401	327	966
	Simple func call(100)	4112	4106	10906
	New object (100)	1102	1152	3445
	n-body (1)	418	481	683
	object property (18)	379	1385	642
	simple method call (100)	816	1405	2200
	Type hinting (100)	183	235	489
	Array (25)	1122	2571	4284
	singleton pattern (100)	102	62	286
	Matrix (8)	7292	5832	11086
	Encryption (10)	8934	2992	15331
	Total Time	**37.13**	**36**	**77.99**
PC 2	Fibonacci (25)	71	56	x
	Prime number (100)	1495	856	x
	Ackermann Function(3,3)	1519	1625	x
	Loop (1000)	337	161	x
	Math Methods (100)	3445	6438	x
	Empty String Check (25)	9	11	x
	Control Structure (100)	217	25	x
	String Manipulation (25)	861	374	x
	Array Search(linear,binary)(10)	47	72	x
	Quick Sort (1)	265	217	x
	Regex (25)	880	1438	x
	Fetch constants (18)	161	138	x
	Closure (100)	386	159	x
	Getter_Setter (8)	319	312	x
	Simple func call(100)	3316	2842	x
	New object (100)	906	1109	x
	n-body (1)	332	372	x
	object property (18)	301	854	x

	simple method call (100)	671	867	x
	Type hinting (100)	147	171	x
	Array (25)	897	1901	x
	singleton pattern (100)	82	65	x
	Matrix (8)	5836	4509	x
	Encryption (10)	6887	2349	x
	Total time	**29.39**	**27**	x
PC3	Fibonacci (25)	31	62	x
	Prime number (100)	2002	1106	x
	Ackermann Function(3,3)	2019	2236	x
	Loop (1000)	482	188	x
	Math Methods (100)	4672	8846	x
	Empty String Check (25)	15	62	x
	Control Structure (100)	305	16	x
	String Manipulation (25)	1280	438	x
	Array Search(linear,binary)(10)	69	47	x
	Quick Sort (1)	361	235	x
	Regex (25)	1199	2018	x
	Fetch constants (18)	235	173	x
	Closure (100)	520	394	x
	Getter_Setter (8)	445	390	x
	Simple func call(100)	4375	4197	x
	New object (100)	1198	1469	x
	n-body (1)	458	472	x
	object property (18)	417	1278	x
	simple method call (100)	873	1452	x
	Type hinting (100)	201	184	x
	Array (25)	1225	2943	x
	singleton pattern (100)	110	93	x
	Matrix (8)	7864	7105	x
	Encryption (10)	9674	3286	x
	Total Time	**40.03**	**39**	x

Table 1. Performance Evaluation of benchmarks in JPHP and PHP.Bench Codes were run several times and the minimum value is taken. Total times is measured in seconds.

6 FINDINGS/SUGGESTIONS ETC.

JPHP is not the first attempt to improve PHP performance, in the end how effective, we will have to wait and see.It has elegant API of it's own but if one wants to use a function that is not implemented in JPHP, a migration guide is there to migrate the jPHP std library. Using PHP with android is now possible with JPHP but it's not stable and in development. By the way, JPHP can do GUI programming using Swing , examples are given in their website [2] .In this way, it is one of JPHP's aims to write programs that are used by non - Web servers in the PHP grammar . It seems that one can also use other ways to set up a daemon to listen for TCP connections, and to write Android code

As the source of JPHP is open for all, anyone can contribute to it. But it seems like production of JPHP is now not that active. The creators of JPHP are not interested in implementation PHP standard libraries (pcre, pdo, etc). So running Joomla, Drupal or another CMS on JPHP is impossible. But despite these shortcomings , there are several things that can be planned for the future work. In short,JPHP seems neat and might have a promising future.If the authors plan to implement the language with zend runtime libraries,then it might gain popularity like HHVM. They should put more focus on increasing support in writing PHP for Android apps, because *Faster* tag won't attract most developers to switch entire deployment process to cut off a bit of execution time.

7 CONCLUSION

JPHP was implemented to compile from PHP source code to Java bytecode , and according to the benchmark test results it turned out that it is a faster processing system than other implementations.It can be said that JPHP has fully drawn out the power of JVM.
However,JPHP still is in active development. Due to the project goals it does not have the mass appeal that HHVM.If anyone interested to adopt this language for performance , may keep an eye on their github repository.[11].
PHP was a capricious candidate, but in the end the operation was successful.For quickly creating a prototype, a front end for something, using PHP, is conspicuous.In this report, we tried to measure the performance of JPHP and PHP by taking blended combination of functions for benchmarking.

REFERENCES

[1] [n. d.]. The PHP Bench. ([n. d.]). http://www.phpbench.com/
[2] 2014. JPHP - an alternative to PHP. (2014). http://j-php.net/
[3] 2014. JPHP vs PHP benchmark. (2014). https://infogram.com/jphp_vs_php_benchmark
[4] 2014. jphp@jphpcompiler. (2014). https://twitter.com/jphpcompiler
[5] Hanawa. 2015. JPHP of PHP processing system written in Java was fast. (2015). http://d.hatena.ne.jp/hnw/20150117
[6] Thiemo MÃďttig. 2017. My PHP Performance Benchmarks. (2017). http://maettig.com/code/php/php-performance-benchmarks.php
[7] Shigebeyond. 2016. jphp-bench-test. (2016). https://github.com/shigebeyond/jphp-bench-test
[8] Dmitry Stogov. 2010. The PHP Interpreter -Zend Benchmark. (2010). https://github.com/php/php-src/blob/master/Zend/bench.php
[9] Christian Vigh. 2017. PHP Performance Comparison. (2017). https://www.phpclasses.org/blog/post/493-php-performance-evolution.html
[10] Wikipedia. 2017. List of JVM languages. (2017). https://en.wikipedia.org/wiki/List_of_JVM_languages
[11] Dmitry Zaitsev. 2014. jphp. (2014). https://github.com/jphp-compiler
[12] Dmitry Zaitsev. 2014. JPHP - an implementation of PHP. (2014). https://github.com/dim-s/jphp