International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-1, October, 2019

# Making the Web 2.0 Faster for Next Generation

**Brijesh Reddy, Pranav Chandran T**

***Abstract:*** ***Undeniably the most favored web scripting language is PHP. Almost 80% of the internet's server-side web applications are written in PHP which includes big giants like WordPress, Wikipedia, and Facebook. In present-day, at an accelerating pace, the quantity of digital content is burgeoning. A heterogeneous set of users' devices is being amassed by these contents and administering these contents manually is an infeasible solution engendering an increasing set of problems. A solution to this problem would be to switch to a web programming language, which can be compiled. We are describing an easy to deploy and a continuous conversion mechanism for converting existing Web 2.0 PHP application systems into Facebook's HHVM supported Hack server-side application systems. We are trying to use the power of Hack language and amplify the performance of existing PHP server-side applications. Instead of interpreting all of your code Hack translates it to assembly and runs that instead, which can lead to an immense amount of increase in performance. We are using Hacktificator, a tool developed by Facebook Developers and our demo web application running on HHVM to test and convert user's existing PHP codebase to Hack language.***

***With this proposed methodology we do not have to make any change to existing codebase manually or hire new engineers for the conversion, nor do we have to take down our live systems. Conversion can be done on the fly and will result in approximately 2x to 20x better performance. The availability of this tool can save costs for manual conversion, save time as well as improve the user experience of websites with better performance.***

***Keywords: Hack;HHVM;PHP; Web 2.0***

## I. INTRODUCTION

During recent years PHP - a server-side scripting language has emerged as a prominent language for website development. Over millions of websites have used PHP as their core language to develop server-side web applications. As of January 2018, PHP was used by nearly 86% internet websites which are running on PHP 5.0 and above [1]. PHP is also used as a primary server-side development language for organizations including the Facebook and Wikipedia.

The HipHop Virtual Machine (HHVM) is a JIT compiler and runtime for PHP. While PHP values are dynamically typed, real programs often have latent types that are used for optimization once discovered [6]. Some types can be proven through static analysis, but limitations in the ahead of time approach leave some types to be discovered at runtime. And even though many values have latent types, PHP programs can also contain lots of polymorphic variables and expressions, which must be handled without catastrophic slowdown. Hack is a programming language for HHVM. Hack reconciles the fast development cycle of a dynamically typed language with the discipline provided by static typing while adding many features commonly found in other modern programming languages. Hack provides instantaneous type checking by incrementally checking your files as you edit them. It typically runs in less than 200 milliseconds, making it easy to integrate into your development workflow without introducing a noticeable delay [3]. Web 2.0 basically refers to the transition from static HTML Web pages to a more dynamic Web that is more organized and is based on serving Web applications to users. Increasingly, websites enable community-based input, interaction, content-sharing and collaboration. Types of social media sites and applications include forums, microblogging, social networking, social bookmarking, social curation, and wikis.

Revised Manuscript Received on October 30, 2019.
* Correspondence Author
**Brijesh Anand,** Department of Computer Engineering, SIES Graduate School of Technology, Mumbai, India
**Pranav Chandran T,** Department of Computer Engineering, SIES Graduate School of Technology, Mumbai, India



## II. BACKGROUND

### A. Background of PHP

Given that C++ well represents the class of static, high performing languages, while PHP is representative of the much worse performing scripting languages, it is worth pointing out the main differences between the two languages [7]. This brief goes through issues of the language in terms of compiling and its performance as in comparison to its usage. In PHP, there is a tendency to mix HTML and language syntax inside HTML files, which quickly results in the poorly maintainable code where views and business logic are not separated. As a result, it's hard to extend PHP applications with new functionality and manage applications with a large code base. As explained in the research during the construction of HipHop Compiler for PHP, there is a severe slow down due to dynamic lookups. There are also several other issues such as dynamic name binding, dynamic name resolution, dynamic code evaluation and dynamic symbol inspection. The loading of PHP programs is not as simple as required for the easy execution [7].

### B. Need for JIT Complier and HHVM

The HipHop compiler (HPHPc) had greatly improved the performance of PHP applications before work began on HHVM. HPHPc uses ahead-of-time techniques to translate PHP into C++ and then produce a native binary. While HPHPc's performance gains over interpreted implementations are impressive, the ahead-of-time model introduces both performance limitations and operational complexity which motivated the development of HHVM [6].

Much of HPHPc's power to accelerate PHP comes from type inference, which allows static binding of many dynamic invocation sites. The undecidability of type inference limits the power of this technique in an ahead-of-time setting. Since the directions of many program branches are undecidable, unknown types frequently taint the data-flow graph, preventing optimization. As a just-in-time compiler, HHVM has access to the runtime values flowing through the program.

### C. Adoption of HHVM and Hack

By the end of 2015-16, Facebook has tremendously gained a phase shift in terms of performance as compared to increase in their traffic. With HHVM and Hack, Facebook shifted from 35% of their server-side codebase to 90% of their server-side codebase into Hack. Soon, several giants like Wikipedia shifted to HHVM and gained a performance boost as shown in fig. 1 below [10].

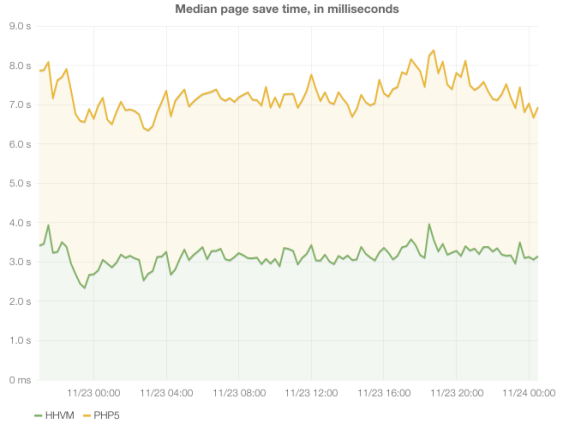


**Fig 1.** ***Wikipedia's performance boost from PHP5 to HHVM***

### D. Performance of Hack

According to popular benchmarks, Hack beat PHP almost in all high computations. The site gave statistics based on CPU load, gzip compression of LOC of both codebase, memory usage and time taken in seconds. Following figure 2 shows the performance in terms of time taken (in seconds) [11].

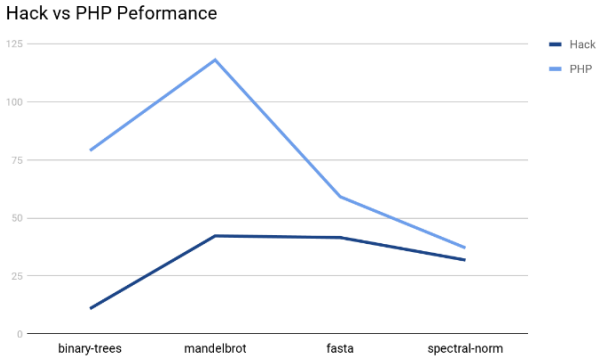


**Fig 2.:** ***Computer Language Benchmarks Game, Hack vs PHP***

## III. PROPOSED APPROACH AND TOOL SUPPORT

### A. *Available tools from Facebook Developers*

#### *i. hh_server*

The Hack server works behind the scenes to keep our codebase in constant sync for the hh_client typechecker. However, the server hh_server can also be used for some primary standalone functionality as well [8]. If we want to typecheck a file or a directory of files without the overhead of a constant server running combined with hh_client, we can do a quick and dirty check with hh_server.

***hh_server --check <path>***

#### *ii. hactificator*

The Hackificator is a tool to use to begin converting your PHP codebase to Hack. It ships in default with HHVM package for all major operating systems [8].

***hackificator <directory or file path>***

#### *iii. hack-codegen*

Hack Codegen is a library for easily generating Hack code and writing it into signed files that prevent undesired modifications. The idea behind writing code that writes code is to raise the level of abstraction and reduce coupling [9]. We can use our own way of describing a problem and generate the corresponding code.

### B. *Proposed Conversion Tool*

We have generated a very simple overview of the proposed tool. The *Hack Web 2.0 Grid*, will consist of HHVM server running an application. Grid consists of the main conversion mechanism. Our process consists of *3 phases* of the overall conversion.

*The first phase* consists of making HHVM server live up to switching the new codebase from PHP to Hack. This can be done prior to the conversion process or after the conversion.

*The second phase* is making the existing PHP codebase available on any *Version Control System (VCS)* like GitHub.

*The third phase* consists of the actual conversion mechanism. The application on which the Grid runs will be able to extract all the existing code base using APIs provided by the VCS platform. The whole PHP codebase will be passed through a stream channel to *hackificator* [4]. Logs will be maintained for recording minute changes made to the codebase via stream channel. Once the PHP codebase is properly converted into Hack codebase *(at least of* ***decl*** *type)*, the application will start testing via *hh_server*. The testing of annotations, compiling and performance checkups can be made during this process. Statistics, Log outputs can be made available to the user at this stage and with an option to replace the newly converted codebase with the original codebase hosted on any VCS.

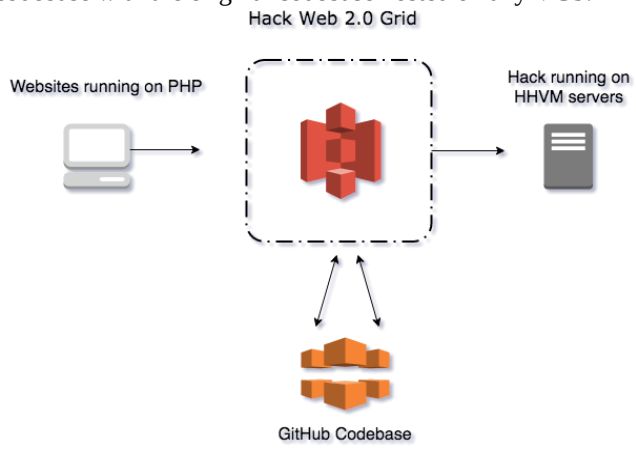


**Fig 3. Overview of proposed tool**

## IV. LIMITATIONS AND FUTURE SCOPE

Our paper deals with the conversion mechanism of PHP coded projects or libraries which are written in the Object-Oriented way. Hack being gradually typed language currently it does not render views as easy as PHP. The scope of this conversion tool is to add support for XHP as well as bring more features such as Async, Lambdas, and Callables, etc. Hack is supported on HHVMs hence, it is necessary for the developers to switch from their existing only Apache or nginx servers to HHVM supported servers and run on Proxygen or a FastCGI-based server on top of nginx or Apache [2].

## V. CONCLUSION

Overall, we have presented a mechanism that is applicable to any web 2.0 PHP server-side project which is written using an Object-Oriented methodology. Improved support for XHP and Databases will lead to better performance and close to 100% adoption of HHVM and Hack. This improves the performance of businesses, blogs, and many websites as well as saves time for manual conversion of PHP to Hack. Project managers and developers of current project do not have to take down their live servers down for making up the new Hack system, this can be done on the fly. All of these techniques will save cost as well time for training current developers to learn Hack.

To accomplish our aims, we have introduced an idea which depicts the flow and design of conversion tool. We need full fledge systems which can convert large codebases with LOCs comparable to small blogs and libraries. With the adoption of this tool, we can achieve improved performance of all the websites and which in turn will lead to making almost 80% of the internet faster.

## REFERENCES

1. Haiping Zhao, Iain Proctor, Minghui Yang, Xin Qi, Mark Williams, Qi Gao, Guilherme Ottoni, Andrew Paroski, Scott MacVicar, Jason Evans, Stephen Tu, “The HipHop Compiler for PHP”, *ACM International Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA), October 2012.*
2. Keith Adams, Jason Evans, Bertrand Maher, Guilherme Ottoni, Drew Paroski, Brett Simmers, Edwin Smith, Owen Yamauchi, “The HipHop Virtual Machine”, *ACM International Conference on Object-Oriented Programming Systems, Languages, and Applications, October 2014.*
3. The Hack Programming Language, *http://hacklang.org/*
4. HHVM, *https://hhvm.com/*
5. Facebook Code, *https://code.facebook.com/*
6. The HipHop Virtual Machine - Facebook Research, *https://research.fb.com/publications/the-hiphop-virtual-machine*
7. The HipHop Compiler for PHP - Facebook Research, *https://research.fb.com/publications/the-hiphop-compiler-for-php/*
8. Hack tools, *https://docs.hhvm.com/hack/tools/introduction*
9. Hack Codegen - library for easily generating Hack code and writing it into signed files that prevent undesired modifications, *https://hhvm.github.io/hack-codegen/*
10. Inside Wikipedia’s transition to HHVM, *https://code.facebook.com/posts/1553091884935685/inside-wikipedia-s-transition-to-hhvm/*
11. Hack vs PHP, *https://benchmarksgame.alioth.debian.org*

## AUTHORS PROFILE

**Brijesh Reddy** has completed his Graduate degree in B.E. Computer Engineering from SIES Graduate School of Technology (University of Mumbai). His research interests are Web Development, Big Data, Data Mining.

**Pranav Chandran T** has completed his Graduate degree in B.E. Computer Engineering from SIES Graduate School of Technology (University of Mumbai). His research interests are Web Development, Data Mining, Machine Learning.