COMPARATIVE ANALYSIS OF CODEIGNITER AND LARAVEL IN RELATION TO OBJECT-RELATIONAL MAPPING, LOAD TESTING AND STRESS TESTING

Ushawu Ibrahim [1], J. B. Hayfron-Acquah [2], Frimpong Twum[3]

[1,2,3] *Department of Computer Science - Kwame Nkrumah University of Science and Technology, Kumasi, Ghana.*

Abstract - *As developers endeavour to improve on solving world issues through software technology so are they interested in building robust applications with greater performance. PHP application maintenance becomes difficult and performance compromised, if the applications are developed as a core without frameworks. With the increasing number of frameworks, it is difficult to choose which framework has good performance for development. Two similar web applications were developed in two PHP frameworks (Laravel and CodeIgniter) and Jmeter used to test the performance (Load and Stress) of the applications. Assessment indicated that there is significant variation in performance between Laravel and CodeIgniter on a local setup regarding response time and throughput with mean values of response time as 1,441.5 ms and 5,778.4 ms for CodeIgniter and Laravel respectively, and mean value for throughput as 199.66 kb/sec and 61.64 kb/sec for CodeIgniter and Laravel respectively. On the live set up however, CodeIgniter performed better for small loads but as load increased there was no significant difference in the two. In conclusion, CodeIgniter performed better on local conditions and should be chosen for small and medium scale applications for local use whereas Laravel should be a preferred choice for large applications intended for the worldwide web since data abstraction and other inbuilt libraries will help reduce development time and also performance is enhanced at larger loads.*

Key Words: Performance, Framework, Laravel, CodeIgniter, Load, Jmeter, data abstraction layer, Object Relational Mapping

1. INTRODUCTION

Advancement in technology and its awareness has aided today's internet to become popular as a result of growing number of clients who use it for their daily activities. Internet has evolved and is now the need of all spheres of life as our daily activities are dependent on it. The internet houses web applications which are used for several electronic transactions ranging from, social networking, payment of utility bills, emailing, shopping, banking etc. These applications are coded in different programming and scripting languages some of which include Ruby on Rails, PHP (Hypertext pre-processor), ASP.NET and EJB (Enterprise Java Beans) (Singh et al, 2011). Owing to the fast evolution of technology and the related significance, for all facet of businesses, web development is now a dominant field in technology. Rise in request of web applications has also brought about high request for scalability, reliability, efficiency and maintainability. PHP has several features which gives it added advantages over other languages; cross platform compatibility, open source, scalable, efficient execution and support for SQL, for connection and manipulation of databases (Cui et al, 2009). PHP is a server domain language designed purposely for web application development, it is mainly designed for creating interactive and dynamic content. Developing in core PHP, means combining business logic with queries of a database and presentation views. This kind of mixture in development makes scalability and maintenance of these applications very difficult. This study will investigate two popular PHP frameworks taking into consideration load and stress test on both a local and server environment.

1.2 Problem Domain

Web applications deliver a multifarious collection of functionalities to a huge number of varied clients. To maintain users' interest on a web application it is imperative to start web pages quickly (Ortiz et al, 2012). There is therefore a greater need for developers to choose suitable web development platform for coding, which is a time consuming task. In the past, developers loved CodeIgniter for the related comprehensive documentation together with its simplicity even though it lacked some functionalities that Taylor Otwell, deemed critical in web application development. Laravel which has built in Object Relational Mapping (ORM) was released to cure the growing challenges of CodeIgniter (surguy, 2014), but it is however important to investigate whether solving the critical issues of CodeIgniter in Laravel compromises performance or not. The research work thus calculates and compares the performance of two PHP frameworks taking into consideration load and stress testing.

2. LITERATURE REVIEW

Performance of applications is very significant to users and developers. Chandran and Angepat [1] compared two web development technologies by developing a real estate application using PHP and ASP.NET as programming languages. They concluded that ASP.NET had more efficiency and reliability than PHP and also ASP.NET with C# provides controls, functionalities and built-in tools which aids quicker code development. Lamos [2] dealt with the performance and speed optimisations of an underlying server and a WordPress-based CMS application operating on it. The work

enabled an average web application developer the ability to mount and configure his/her web server and refactoring the source code of his/her PHP web application, devoid of making research on what to change or avoid from other sources, thereby saving time and resources. Vasar, Srirama and Dumas [3] research work focused on setting up a test bed for cloud that will run web applications. The study gives a proper picture on how servers in the cloud are scaled whilst the whole process remains transparent for the end user, as it sees the web application as one server. The work of Wang [4] was based on Speeding up Mobile Browsers without Infrastructure Support, Wang concluded that Mobile browsers were known to be slow and they characterized the performance of mobile browsers and concluded that resource loading is the bottleneck. Zarate and Hernandez [5] deliberated on best norms in diverse platforms including Ruby on Rails, CakePHP, Struts, JSF, Lift. Comparatively, CakePHP demonstrated the same rate of performance as seen in other frameworks though Lift indicated better outcomes for web development. Pena-Ortiz and Sahuquillo [6] analysed the performance of a web server. They explored the changes of web server presentation using dynamic workload in web applications. The outcome indicated that dynamic workload exerts more pressure on a web server relative to using the normal workload and also, there was a rise in CPU utilization up to 20 percent whilst response time reduced from 40 percent to 50 percent, when diverse user dynamic behaviours were replicated in the same website. Patel, Rathod and Parikh [7] performed comparison of diverse content management systems of PHP. They established that Drupal was best suited for informative websites, whereas Joomla is preferable for quick response and compound objects whilst WordPress is suited for caching. For the live set up, it was concluded that WordPress performance was the greatest in all areas though Joomla achieved better performance after caching pages. Dinh [8] compared two frameworks Concrete5 and Refinery CMS by building a web application with them. Concrete5 required less effort and time for installation due to its countless features such as plug-in availability, packages and templates, and thus was recommended as a better option for building web applications. Fayyaz and Munir [9] compared and analysed the effect of data abstraction layer (ORM) on CakePHP and CodeIgniter performance. They finalised that CAKEPHP is suitable for enterprise level and large applications, whereas CodeIgniter was suited for medium and small size applications. With the inception of several programming languages, it is tedious if not impossible to determine the advantage of one language in a specific scope and structure over another. Dwarampudi, Dhillon, Shah and Sebastian [10] performed comparative analysis of the main programming languages like PHP, Perl, and VB .NET, Ruby, AspectJ, Haskell, C++, Scala, Java and Scheme with the core factors in program development. Ruby and PHP is seen as the most ideal languages for web development. Ruby is mostly ideal in situations where flexibility, readability, performance and security are of great concern. Ruby and Java are the most preferred for situations where applications developed has high demand for robustness and security without having much concern for performance. C++ should be used for developing standalone applications where performance is of utmost concern.

3. METHODOLOGY

In order to achieve the experimental object, similar applications were developed in both frameworks that is Laravel and CodeIgniter. The applications had the same scope, design and best practices were also adhered to. A basic ecommerce web application is developed with the following components/functionalities:

- ✓ User can login and logout
- ✓ Customers can see about us page, contact us and store interface
- ✓ Users can add products, edit products and delete products
- ✓ User can add other users
- ✓ Customers can purchase items by selecting products and their quantities
- ✓ Check out page where customers can check out to pay for items
- ✓ Payment gateway where customers will be expected to enter payment details and pay for products

The applications use MySQL database to keep application information. HTML, Twitter bootstrap CSS and JavaScript for design and styling purposes respectively. AJAX is used for developing a customized cart system. Both applications were deployed in local and live environments. Local environment was run on a windows 8 operating system where as live environment was Linux operating system.

Table 1: Test Plan in JMETER

Online				
	Online User			
		Java Request Defaults	Thread Group	
		Store list	Name:	Online User
		Login		
		About us		
		Contact	Thread Properties	
		Store list	Number of Thread:	200
		Store list	Ramp-Up Period (in seconds):	20
		Store list		
		Check_out_cart	Loop Count:	3
		Check_out		
		Summary Report		
Work Bench				

4. IMPLEMENTATION

Both applications were verified in diverse testing scenarios. Starting repetition from 200 threads and reiterated both set ups up to 2,000 threads (users). Loop count was set to 3 to allow average value when the experiment was done 3n times even though each user will perform one set of events at a time when the site is visited. Testing had an incremental value of 200 for each cycle making a total of 10 counts per test on both local and liver server, except in the case of Laravel on a live set up which could not take values beyond 1000 threads, threads beyond 1000 resulted in system freeze and hence experiment was aborted at that level.

Hypothesis for local set up:

1. Null Hypothesis, H0: There exist no variance of performance concerning CodeIgniter and Laravel in the perspective of load testing (throughput and response time) on a local set up.

2. Alternate Hypothesis, H1: There exist a variance of performance concerning CodeIgniter and Laravel in the perspective of load testing (throughput and response time) on a local set up.

Hypothesis for live set up:

1. Null Hypothesis, H0: There exist no variance of performance concerning CodeIgniter and Laravel in the perspective of load testing (throughput and response time) on a live set up.

2. Alternate Hypothesis, H1: There exist a variance of performance concerning CodeIgniter and Laravel in the perspective of load testing (throughput and response time) on a live set up.

Table 2: Datasets for Local Server

Iterations	Number of threads (users)	Ramp up time
1	200	20
2	400	20
3	600	20
4	800	20
5	1,000	20
6	1,200	20
7	1,400	20
8	1,600	20
9	1,800	20
10	2,000	20

Table 3: Datasets for Live Server

Iterations	Number of threads (users)	Ramp up time
1	200	20
2	400	20
3	600	20
4	800	20
5	1,000	20

5.0 PRESENTATION OF RESULTS AND ANALYSIS

5. 1 Cross Evaluation of Response Interval on Local Set Up

The two applications had a test up to 2000 users, however, beyond 2000 virtual users, the error rate went over 50% and the system used for the testing also experienced a hang up. Even though on a local server apache allows up 10,000 clients of connection within a time period. Figure 1 shows the evaluation of mean response time concerning Laravel and CodeIgniter applications. Response interval was detailed through executing test plan using varying number of virtual clients for the two applications. CodeIgniter had improved performance for all the load situations on a local environment.

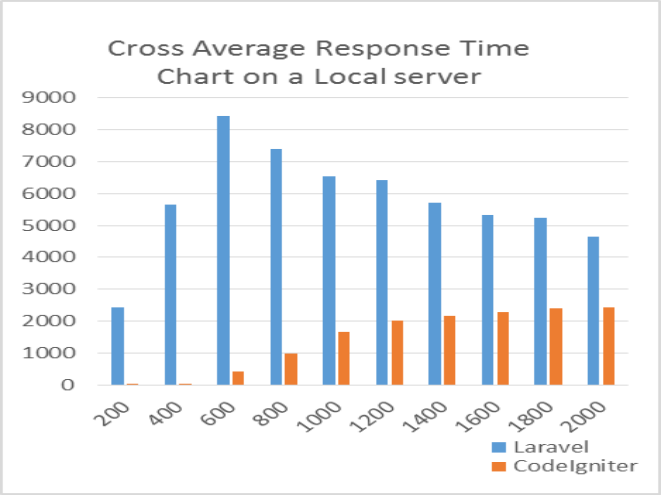

Figure 1: Mean response time of CodeIgniter and Laravel on local environment

The null hypothesis is rejected and the alternate hypothesis "There exist a variance of performance concerning Laravel and CodeIgniter in the perspective of load testing (response time) on a local set up" accepted.

5. 2 Cross Evaluation of Throughput on Local Set Up

Figure 2 shows the assessment of throughputs for Laravel and CodeIgniter on local environment. It is evident

from the chart that the throughput of CodeIgniter is expressively improved as compared to Laravel in every load condition.

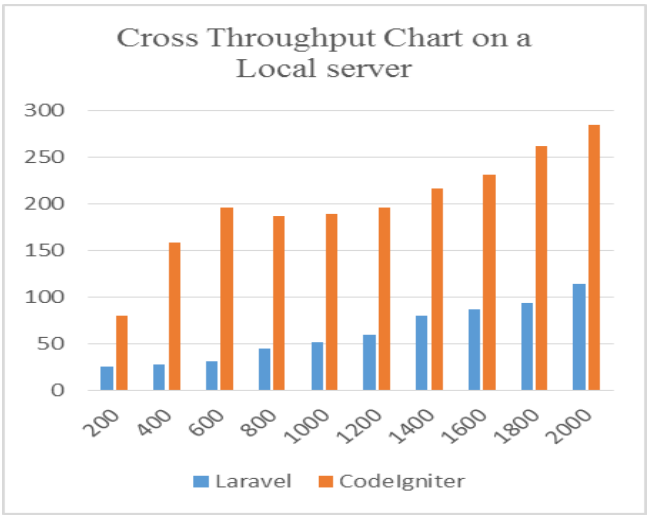

Figure 2: Throughput of CodeIgniter and Laravel on local environment

From figure 2, there is significant variation among the values of throughput for Laravel and CodeIgniter on a local server. The null hypothesis is therefore rejected and the alternate hypothesis accepted.

5. 3 Cross Evaluation of Response Interval on Live Set Up

Figure 3 is the assessment of mean time for response between CodeIgniter and Laravel applications on a live set up. There is an insignificant deference between the response times of both application with Laravel having a better response time in one instance. On extreme load situations, CodeIgniter performed better than Laravel in only one instance of that execution.

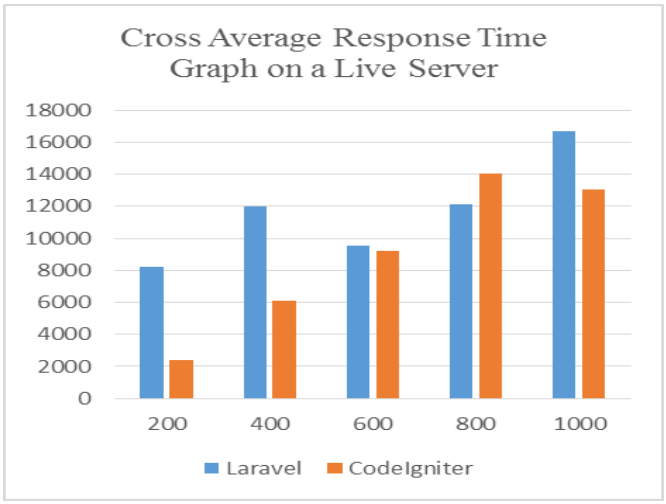

Figure 3: Average response Time between CodeIgniter and Laravel on live set up

The T-test used on the time period of response for CodeIgniter and Laravel on a live set indicates that total **t** value that is 1.76 is lesser compared to the value of **t** critical with significance level of 0.05. There is no significant variation in the time period of response data for the two frameworks. So the Null hypothesis cannot be rejected that is "There exist no variance of performance concerning Laravel and CodeIgniter in the perspective of load testing (response time) on a live set up",

5. 4 Cross Evaluation of Throughput on Live Set Up

Figure 4 shows the assessment of throughputs for CodeIgniter and Laravel on a live set up. The CodeIgniter application throughput considerably outweighs that of Laravel on small loads for the majority number of test runs but on higher load conditions the throughput of both applications is almost the same.

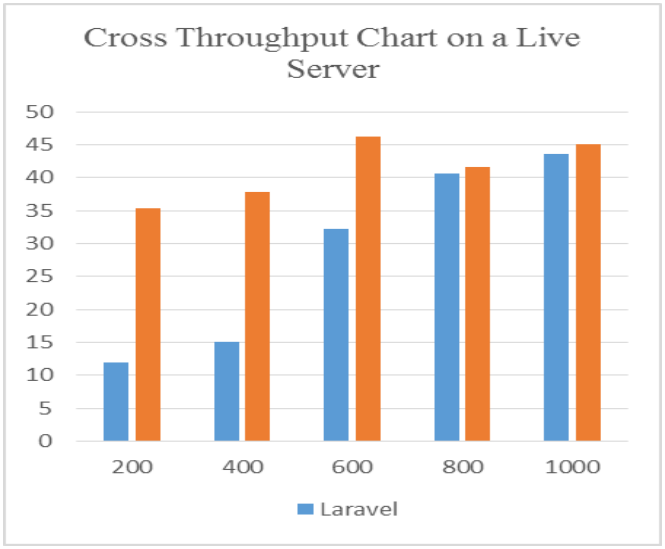

Figure 4: Throughput between CodeIgniter and Laravel and on live set up

It is established that the **t- value** that is 2.55 is slightly less than **t-critical** value for two-tail and slightly more than t-critical for on-tail considering level of significance of 0.05. There isn't a substantial variation in the throughput data for the two frameworks. Therefore the Null hypothesis cannot be rejected. That is "There exist no variance of performance concerning Laravel and CodeIgniter in the perspective of load testing (throughput) on a live set up".

6. CONCLUSIONS

After carefully analysing the outcomes from all testing environments, it is evident that, generally CodeIgniter performed significantly well on the local server with respect to throughput and response time, thus the null hypothesis is rejected for the local set up. Performance difference on a live set up is not very significant although CodeIgniter still performed better in smaller load conditions but there was an

instance of mixed performance results with Laravel slightly performing better than CodeIgniter at that instance with respect to response time, the null hypothesis is thus accepted in this instance. The inbuilt provision of Object-Relational Mapping in Laravel creates an overhead in the presentation of the Laravel application in normal and in stress conditions on a local set up. For the live set up, in smaller load conditions ORM provision in Laravel generated an overhead in performance but in higher load conditions ORM provision creates an affirmative effect on the performance of Laravel that is why there was no much difference between the performances of both applications at bigger load conditions. It is therefore concluded that the presence of ORM in Laravel is not useful in its performance for normal/smaller conditions of load but useful for the performance in stress/bigger load conditions. ORM raises the level of the maintainability, productivity and code reusability. Therefore ORM embedded frameworks should be utilized when developing large scale applications, which require extensive database communications.

REFERENCES

[1] Chandran Sneha Prabha, Angepat Mridula, (2011), "Comparison between ASP.NET and PHP – Implementation of a Real Estate Web Application". LAP Lambert Academic Publishing , Germany ©2012, pp. 37.

[2] Lamoˇs Rastislav (2015), "Performance and speed optimizations of Wordpress-based Web Application and its underlying server stack", Comenius University in Bratislava Faculty of Mathematics, Physics and Informatics. pp. 41.

[3] Vasar Martti, Srirama Satish Narayana, Dumas Marlon (2012). "A Framework for Verifying Scalability and Performance of Cloud Based Web Applications", WICSA/ECSA '12 Proceedings of the WICSA/ECSA 2012 Companion, pp. 53-60.

[4] Wang Zhen, (2012). "Speeding Up Mobile Browsers without Infrastructure Support", Rice University, Houston, Texas April, 2012, pp. 68.

[5] Zarate M. d. P. S., Hernandez G. A., A. R., (2012). "Developing Lift-based Web Applications Using Best Practices", Procedia Technology 3, The 2012 Iberoamerican Conference on Electronics Engineering and Computer Science, Volume 3, pp. 214–223.

[6] Ortiz R. P., Gil J. A., Sahuquillo J., Pont A., (2013). "Analyzing web server performance under dynamic user workloads", Computer Communications, Volume 36, Issue 4, 15 February 2013, IEEE, pp. 386-395.

[7] Patel S. K., Rathod V.R., Parikh S., (2011) "Joomla Drupal and WordPress - A Statistical Comparison of Open Source CMS", 3rd International Conference on Trendz in Information Sciences & Computing (TISC2011), IEEE, pp. 182 - 187.

[8] Dinh Melinda, (2015), "Ruby and PHP Development: A Comparative study of Development and Application using Content Management Systems Refinery CMS and Concrete5", pp. 22-24.

[9] Fayyaz Ali Raza, Munir Madiha, (2013). "Performance Evaluation of PHP Frameworks (CakePHP and CodeIgniter) in relation to the Object-Relational Mapping, with respect to Load Testing", pp. 32-34.

[10] Dwarampudi Venkatreddy, Dhillon Shahbaz Singh, Shah Jivitesh, Sebastian Nikhil Joseph (2008). "Comparative study of the Pros and Cons of Programming languages". Revision 1.0, pp. 19.