Abstract— The absence of the extensive performance comparison for websites in multiple framework has been a major hindrance in this area. There is a need of proactive charting which then can be referred to before selecting a framework for web development. The use of best practices is imperative and worthwhile, notwithstanding the performance of an application can’t be undermined. While the former, in general, tackles DRY (Don’t Repeat Yourself) approach, thus reducing the management of plethora of code; the later, on contrary, deals with minimizing the perceived delay in page load time.

In this paper we investigate the performance of Yii & Symfony, which are PHP based frameworks and Django, which is a Python framework. Different applications are developed for different frameworks, but each with a unique user interface. Also, each application shares a common MySQL database.

These results are further used to conduct performance comparative analysis, which is to identify that which among other frameworks perform better in-terms of 3 KPI’s (Key Performance Identifiers). At the end it concludes that in terms of throughput, Symfony outperformed Yii framework based on the number of samples processed over time. On the other hand, Yii utilized lesser CPU Processing than symfony for the same kind of task and it was also found to be more responsive. Django application, on the other hand, was found to be more efficient as compare to PHP frameworks in all the scenarios.

Keywords— WAMP, performance comparative analysis, MVC, KPI, Jmeter, PHP, Python, Django, RoR, Symfony.

Introduction

A Web framework is a collection of packages or components that allow developers to write Web applications or services without having to handle such low-level details and reinventing the wheel. Most if not all web frameworks are built on the DRY (Don’t Repeat Yourself) approach. Anything that is repeated in two or more places is more difficult to maintain. Every time a change or correction is made, multiple locations must be updated, which increases the chance of errors and inconsistencies. To avoid this, programmer’s follow the DRY Principle, for “don't repeat yourself,” which applies to both data and code [21]. The framework aims to ease the overhead associated with the common tasks achieved in web
development. The overall goal of the web frameworks is to decrease the Development time.

Each framework in general adopts a single architectural pattern and multiple design patterns. The architectural pattern may be modified to suite the inherent requirements of the web framework. Each framework comprises libraries for database access, session management, templating by promoting the code reusability etc.

A large number of websites are developed on PHP. As per Netcraft’s survey, their exists at minimum 244 Million PHP based websites with 2.1 Million unique IP addresses [26]. MySQL on the other hand is an open source database management system, which is also being used extensively on the Internet. Ruby, Python and PHP are the dominant technologies together holding 82.30% of all the websites [16] on the Internet. MySQL along with with PHP and Apache forms a web development stack that integrates well with each other.

In this paper we present the results we obtained while investigating some performance issues related to Yii, Symfony, & Django applications built with these frameworks. Afterwards, the performance comparative analysis on the results of these framework’s is performed.

**MVC Architectural patterns**

An architectural pattern is a design pattern but with a much wider scope. It gives a reusable solution to a common problem. Some of the problems that are addressed are availability, performance, SQL abstraction (embedding SQL in programming language). The MVC pattern: MVC architectural pattern is used to link the gap between user interface on one end and programming code on the other end thus performing a separation of tasks for the web developers and designers. The MVC pattern distributes the responsibilities of any web development tasks into three main roles thus allowing more efficient Development Effort. These are design, development and integration [22]. Development role is taken care by the programmers. The design role is taken care by the designers who design the look and feel of the application. Integration role integrates the roles of designers and developers to create an application in its entirety.

Model part in the MVC deals with the data and the business rules. View on the other hand holds user interface elements. Controller provides interaction between Model and View Layers. There is also a Router component involved that receives user request and passes it to the controller. It helps URL formation. Symfony, Yii and Django are all MVC based Frameworks (more or less). The Symfony is a set of reusable PHP components developed by Sensio Labs. These PHP components are more cohesive and less coupled thus increasing the reusability to other custom applications [7]. The Yii framework is a high performance PHP framework with out of the box web 2.0 standards support [8]. Django on the other hand is an open source web framework built on Python that follows Model-View-Template (MVT) architectural pattern.

The following diagram in (Figure-1) represents a typical workflow of the PHP [20] MVC framework called Yii application framework.

---

**Fig. 1 Model View Controller [27]**
Analyzing Web Application Performance

Performance in simple terms is finding the acceptable page load time. From the click on a URL to the point at which the page is completely displayed on the destination PC is termed as page load time. Page load time depends on the discovery (DNS lookup) and transfer (sending request & response to the client and server respectively) [11].

The overall goal of improving performance is to minimize the perceived delay the user experiences between the moment he clicks on the link and the page is finally displayed.

Over the last couple of years performance of web applications became more important to businesses as search engines (such as Google) factor in performance into their ranking. This ultimately leads to Performance == Better Visibility == More Users == More Revenue [18].

Strategies for improving performance:

- Decreasing the number of requests. This can be done by reducing the number of resources by using sprites [14] and combining JavaScript [15].
- Optimizing the rendering speed. This can be done, for example, by using more efficient CSS selectors, a better page layout, and optimizing the JavaScript code [12].
- Making the loading time appear shorter. Leverage how humans perceive information to make the delay appear less than it really is by adding loading indicators [13].

Key Performance Identifiers:

- by identifying its number of responses per second.
- by identifying the response time.
- by identifying the CPU utilization where the server is located.

Related Work

Even though PHP, Python and Ruby are very popular programming languages, there has been very little work done relative to the performance comparative analysis of their frameworks on multiple parameters at application level. But, there has been a substantial amount of work done to analyze performance in-terms of response time of applications. María del Pilar Salas-Zárate et al (2015) in the paper discussed about the best practices of different web application framework by comparing a scala based framework Lift and concludes that lift supports relatively more features compare to other discussed frameworks [2].

As a proof of concept Lift based web applications were developed by applying best practices such as actors, lazy loading, Comet support, SiteMap, Wiring, HyperText Markup Language, version 5 (HTML5) support, and parallel rendering.

Dragos-Paul Pop et al (2014) discussed about the MVC architecture for rapid application development. Many other components were discussed like security, form generation and validation, database access and routing. This paper encourages the separation of presentation from logic and data storage in an application [3]. Major objective of this paper are (1) Reduction in development time, (2) Reusability & Maintainability of code.

Varsha Apte et al (2002) in the paper conducted a performance comparison of four web programming technologies, namely, Java Servlets, Java Server Pages, CGI/Cpp and FastCGI/Cpp. In this paper, the comparison is based on two cases: one is a case study of a complex application that was deployed in an actual Web-based service; the other is a ‘trivial’ application [4]. The methodology of performance analysis was stress testing and measurement. The performance analysis parameters were Response time, CPU Utilization, Concurrent Users, Throughput.

Md Umar Khan and Dr T.V. Rao et al. (2014) discussed the architectural pattern XWADF and
compared it with MVC pattern by identifying design patterns that can improve scalability and availability. Latency and throughput parameters were used to measure reliability in terms of scalability. It was stated that reliability attributes, Scalability and Availability, directly impact the performance of application. Design patterns that improve reliability are discussed and applied on XWADF architectural pattern. The empirical results revealed that XWADF architectural pattern can improve reliability of web applications and make them robust in terms of scalability and availability [5].

Lance Titchkosky, Martin Arlittz and Carey Williamson et al. (2004) evaluated the impact of three different dynamic content technologies (PHP, Perl and Java) on the basis of web server performance. Static and dynamic cases are discussed on the basis of with and without database access. Results showed that Java server technologies outperformed both PHP and Perl scripting technologies for the dynamic content generation [6]. Following four parameters were evaluated empirically Response Rate, CPU Utilization, Failed TCP connection attempts, Active server threads.

Cristiana Amza, Anupam Chanda, Alan L. Cox et al. (2002) described and evaluated three benchmarks for evaluating the performance of websites with dynamic content. Three types of applications were discussed namely, online bookstore, an auction site and the bulletin board. Online bookstore was evaluated with TPC-W benchmark specification while auction site and bulletin board were discussed by proposing the new benchmark specifications for each. Bottlenecks on various servers namely, Web server, application server and database server were identified. It was found that there was a bottleneck in the database CPU in online bookstore. It was also found that webserver bottleneck existed in both auction and bulletin board implementations. The overall goal of the paper was to present an approach for the benchmarking of the dynamic web technologies [18].

Cecchet et al. examine more current dynamic web content construction technologies, by using two very diverse benchmarks [23]. Lance Titchkosky, Martin Arlitt and Carey Williamson’s work on “A Performance Comparison of Dynamic Web Technologies” utilizes these benchmarks in a more simpler workloads and finds not so ad hoc performance tradeoffs applicable to any Web Content Construction dynamic site.

UV Ramana & TV Prabhakar et al. (2005) evaluated the performance of the LAMP(Linux, Apache, MySQL, PHP) architecture. A website was built and deployed in LAMP, WAMP and WIMP for performance comparison. It was found that in the benchmarking test, LAMP outperformed both WAMP and WIMP. It was also found that WAMP just marginally performed better than LAMP. Among all, WIMP underperformed in the benchmark test. This paper also focuses on the performance comparison of PHP and C. It concluded that C outperformed by a factor of around 400 than PHP [19].

A E-Commerce performance benchmark is TPC-W. It is a Web Performance Benchmark developed by the Transaction Processing Performance Council. The Workload used in TPC-W intends to demonstrate the activities of a business oriented transactional Web server. A detailed evaluation of TPC-W is given in [24].

Gaurav Banga and Peter Drushel et al. demonstrates a client emulator for the burst traffic generation that temporarily go beyond the capacity of the server. It also accentuates the problems that one faces in the measurement of Web server capacity. They have
constructed a tool for traffic generation by using a two process architecture [25].

**Application Benchmark**

We investigate the performance of the applications built within WAMP for PHP based application frameworks. WAMP is Web development stack that comprises Apache, PHP and MySQL in a single software package [17].

Hrnoid (Figure 3) is a web application developed for a company who is into market surveys. This application fetches data to its homepage from database management system (MySQL). In this benchmark, two separate applications were developed, each with an aforementioned framework along with similar user interface. They also share a common MySQL database. These two applications were then used to identify the performance of their respective frameworks.

**TABLE I Hrnoid Application Testing Platform**

<table>
<thead>
<tr>
<th>Framework</th>
<th>Language</th>
<th>Database</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symfony2</td>
<td>PHP</td>
<td>MySQL</td>
<td>Windows</td>
</tr>
<tr>
<td>Yii</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Django</td>
<td>Python</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Application Performance Measurement**

The experiment setup consists of a client emulator called Jmeter which is an open source project that emulates multiple clients. These multiple clients, also called as threads, can then be used to request a single server for the identification of various KPI’s (Key Performance Identifier) results.

**Test Bed**

The machine on which the webserver and database server are running is a 2.7GHz dual-core Intel Core i5 processor (Turbo Boost up to 3.1GHz) with 3MB shared L3 cache, 8GB of 1866MHz LPDDR3 onboard memory, 128GB PCIe-based flash storage SSD and onboard Intel Iris 6100 graphics. The webserver used is Apache web server version 2.4.9 with PHP module version 5.5.12. The database server is MySQL version 5.6.17.

**Sample Configuration**

JMeter is used as an emulator for identifying and calculating the performance results based on various parameters [9]. In Jmeter, each thread is considered a user that emulates for a single sample [10]. Multiple samples are used for the ideal results. In our test case, 10 users (threads) will try to access the webpage simultaneously within 15 seconds. Each among 10 users will repeat 30 times. (Table-1). So, a total number of 600 samples will be processed by each framework for different Key Performance Identifiers.

**TABLE II Sample Configuration**

<table>
<thead>
<tr>
<th>Number of Threads (users)</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ramp-Up Period (in Seconds)</td>
<td>15</td>
</tr>
<tr>
<td>Loop Count</td>
<td>30</td>
</tr>
</tbody>
</table>

**Fig. 3 Hrnoid Application**

**D.Findings**

1. **Response Time:**

F1. Django was found to be more efficient than Symfony & Yii in terms of total response time.
Total response time is the aggregate response of every sample. In this case the total of 600 samples were sampled. Symfony & Yii and Django applications took 1415.4, 1359, 128.4 seconds of response time to service 600 requests of 10 users respectively (Figure-4). Thus Django is around 11 times faster than Symfony application and around 10 times faster than Yii application.

**F2.** Django was found to be more efficient than symfony and Yii in average response time.

Average response time is the Total response time divided by total no. of samples. As can be seen in (Figure-5), the total of 600 samples were sampled by 10 users. The symfony’s average response time for 600 samples was 2359 ms, while that of Yii’s Average Response time was 2265 ms. Django application outperformed with an average response time of 214 ms.

**F3.** Django’s Maximum Response Time, out of all the samples, was lower as compare to Symfony & Yii. Maximum response time is the highest response time that was identified among the response times of 600 samples. (Figure-6) shows the maximum response time for symfony was 4312ms. While, Yii had a Maximum Response Time of 5458ms. Django had a Maximum Response time of 2050ms. Out of 1800 samples of all the applications, Maximum Response Time was that of Yii. So, Django’s Maximum Response Time outperformed both Symfony & Yii by the factors of 2.01 and 2.6 respectively.

**F4.** With every increase in the thread size (No. of concurrent users), Yii outperformed symfony. Django outperformed both the PHP frameworks.

In this parameter, it is identified the total response time of the samples with respect to the increase in the thread size or when the number of users are increased. Response Time changes with amount of parallel threads. Naturally, server takes longer to respond when a lot of users requests it simultaneously. As the no. of users are scaled, this parameter is used to identify the response times.
Threads in this scenario are the no. of users that access the shared resource in this case the webserver where the respective frameworks are operative. The lower the Response time is with respect to active thread, the efficient an application is. Response time directly impacts the page load time of an application.

**TABLE III**

<table>
<thead>
<tr>
<th>No. of Active Threads</th>
<th>Yii (ms)</th>
<th>Symfony (ms)</th>
<th>Django (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>297.000</td>
<td>452.000</td>
<td>35.4</td>
</tr>
<tr>
<td>2</td>
<td>340.916</td>
<td>605.625</td>
<td>99.05</td>
</tr>
<tr>
<td>3</td>
<td>622.285</td>
<td>775.818</td>
<td>551.36</td>
</tr>
<tr>
<td>4</td>
<td>628.333</td>
<td>925.500</td>
<td>262.93</td>
</tr>
<tr>
<td>5</td>
<td>794.650</td>
<td>1249.692</td>
<td>274.65</td>
</tr>
<tr>
<td>6</td>
<td>978.100</td>
<td>1528.750</td>
<td>518.9</td>
</tr>
<tr>
<td>7</td>
<td>1108.540</td>
<td>1813.200</td>
<td>443.56</td>
</tr>
<tr>
<td>8</td>
<td>1292.377</td>
<td>1928.510</td>
<td>564.4</td>
</tr>
<tr>
<td>9</td>
<td>1475.243</td>
<td>2083.440</td>
<td>688.8</td>
</tr>
<tr>
<td>10</td>
<td>1563.000</td>
<td>2244.367</td>
<td>700.8</td>
</tr>
</tbody>
</table>

From Table 2 it can be observed that when the No. of active thread was 1, Django outperformed symfony & Yii, but when threads were increased to 3, All the curves converged to some degree. All overall scenario, Django outperformed all the frameworks, so it proves that Django takes lesser time to process the same request than other application frameworks.

**2. Throughput:**

**F1.** Symfony processed more samples per second as compare to Yii.

Throughput identifies the maximum capacity of an application to handle the requests. As can be seen in (Figure-8), Symfony, Yii & Django processed 7.4, 6.1 & 16 samples per second respectively. While, Django processed 16 samples per second respectively. Thus Django’s throughput was more than double the performance of both the PHP frameworks.

![Fig. 8 Throughput](image)

**F2.** With the increase in active threads, in all the cases Django outperformed PHP frameworks.

In this finding, the total throughput of the samples with respect to the increase in the thread size or when the number of users are increased is identified. It was found that the Django application outperformed Symfony and Yii by an average factor of 5.25 and 7 respectively. (Figure-9).

![Fig. 9 Throughput vs Threads](image)

It was found that when no. of thread was 7, both symfony and Yii performed equally well. With the increase in active threads, in all the cases Django outperformed PHP frameworks.
TABLE IV
THROUGHPUT VS ACTIVE THREADS

<table>
<thead>
<tr>
<th>Number of active threads</th>
<th>Yii</th>
<th>Symfony</th>
<th>Django</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.808</td>
<td>1.474</td>
<td>17.8</td>
</tr>
<tr>
<td>2</td>
<td>1.436</td>
<td>3.732</td>
<td>22.6</td>
</tr>
<tr>
<td>3</td>
<td>1.602</td>
<td>3.520</td>
<td>14.8</td>
</tr>
<tr>
<td>4</td>
<td>2.836</td>
<td>3.664</td>
<td>22.7</td>
</tr>
<tr>
<td>5</td>
<td>3.944</td>
<td>4.011</td>
<td>31.8</td>
</tr>
<tr>
<td>6</td>
<td>3.764</td>
<td>4.242</td>
<td>14.5</td>
</tr>
<tr>
<td>7</td>
<td>4.005</td>
<td>4.202</td>
<td>25.6</td>
</tr>
<tr>
<td>8</td>
<td>3.801</td>
<td>3.688</td>
<td>22.5</td>
</tr>
<tr>
<td>9</td>
<td>3.638</td>
<td>4.113</td>
<td>14.6</td>
</tr>
<tr>
<td>10</td>
<td>4.101</td>
<td>4.236</td>
<td>19.6</td>
</tr>
</tbody>
</table>

CPU Utilization:

**F1. PHP Applications: Symfony & Yii, utilized more CPU as compare to Django application.**

CPU Utilization is a measure of the amount of work a processor can handle in different load scenarios. This parameter can be used effectively for comparative analysis of two or more applications. Symfony’s CPU Utilization was relatively lower than Yii when the number of users were less than 4, but as the concurrent users exceed the threshold of 4, Symfony’s CPU utilization exceeds that of Yii application as shown in (Figure-10).

![CPU Utilization graph](image)

**Fig. 10 CPU Utilization**

Django’s Utilization was found to be relatively lower in the all cases (Active Thread increase). At the end, the average CPU utilization of Symfony, Yii and Django was 80%, 76% and 46% out of the total 100% processing power of CPU respectively. In all the cases, the increased concurrent user requests may cause bottleneck due to the lack of processing power. In the real life scenario, the concurrent user access hardly manifests in a shorter ramp-up time.

**Conclusion And Future Scope**

**Conclusion**

Performance comparative analysis was done between two well known PHP frameworks: Symfony & Yii, and a Python Framework named Django. It was found that Yii outperformed Symfony in Total Response time, Average Response Time and Maximum Sample Response time. Yii also outperformed by a factor of around 1 in all the cases when active threads were compared with Response time. While, on the other hand, Symfony outperformed Yii in the throughput scenario where number of samples per second were calculated. Symfony also outperformed Yii when Active threads are compared with Throughput by a factor of around 1.2 samples per second. With all these results, it can be said that symfony has the higher capacity to process more samples per second which makes it an obvious choice for enterprise applications, where as Yii performs better in terms of response time. Yii applications turns out to be an obvious choice when page load time is of main concern. Django outperformed PHP frameworks in all the parameters, thus making it an obvious choice over PHP frameworks.

**Future Scope**

In this study, Key Performance identifiers, Response time, Throughput & CPU Utilization are used for the performance comparative analysis. In the future Latency, scalability & efficiency can be used for a more robust performance identification. Scalability
identifies how good a framework performs when no. of concurrent users is increased. Efficiency is directly proportional to throughput and inversely related to utilization. Latency is a time duration between when the request is received to the server and when the first response is replied back to the requesting client by the server. In future, more frameworks along with aforementioned Key Performance Identifiers can be used for more extensive performance comparative analysis.

**References**


https://en.wikipedia.org/wiki/Don%27t_repeat_yourse

If


