

# Design & Implementation of Jmeter Framework for Performance Comparison in Ruby, PHP, & Python Web Applications

Varun Kumar<sup>1</sup>, Dr. Vinay Chopra<sup>2</sup>

Web Developer & Graphic Designer, Ghanchi Media Pvt. Ltd, India<sup>[1]</sup>

Asst. Professor, CSE Department, DAVIET, Punjab Technical University, India<sup>[2]</sup>

**ABSTRACT:** The absence of the broad performance comparison for websites in multiple frameworks has been a major deterrent in this area. There is a need of preemptive charting which then can be referred to before selecting a framework for web development. The use of best practices is crucial and worthwhile, albeit the performance of an application can't be emasculated. While the former, in general, employs DRY (Don't Repeat Yourself) approach, thus reducing the management of the 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 frameworks, Django, which is a Python framework and Ruby on Rails, which is a Ruby 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 5 KPI's (Key Performance Indicators). At the end, it concludes that in terms of Response Time, Throughput, Efficiency & Latency, Django outperformed other frameworks based on the number of samples processed over time. On the other hand, Rails utilized the least CPU Processing among other frameworks for the same kind of task. It was also found that the PHP applications – Symfony & YII – took the least number of bytes to process the same task in contrast to other frameworks. In terms of KPI's, Django application was found to be more efficient in comparison to other frameworks in the most scenarios.

**KEYWORDS:** WAMP, performance comparative analysis, MVC, KPI, Jmeter, PHP, Python, Django, Ruby on Rails, Symfony, Sample Time/Response Time/Load Time, Latency, Efficiency, CPU Utilization, Throughput, Response Time, Threads/Users, Samples, Auto-scaling.

## 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 adapt 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, there 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 PHP and Apache forms a web development stack that integrates well with each other.

# International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirset.com](http://www.ijirset.com)

Vol. 6, Issue 8, August 2017

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

## A. 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 a 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.

The Model part in the MVC deals with the data and the business rules. The View, on the other hand, holds user interface elements. The 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, Django& Rails are all MVC based Frameworks (more or less). All of them consists of an ORM (Object Relational Mapper) that allows object based processing of Relations/Tables. 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 without of the box web 2.0 standards support [8]. Django, on the other hand, is an open source Web Application Framework built on Python that follows Model-View-Template (MVT) architectural pattern. Rails is a Web Application Framework written in Ruby Licensed under MIT. Some of the best-known practices that these frameworks employees are Active Records, Convention over configuration, Templating engines, Scaffolding – quick generation of configurable code such as CRUD(create, read, update, delete) applications, etc. All the code in these frameworks compiles to HTML, which is rendered by the Web browsers.

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

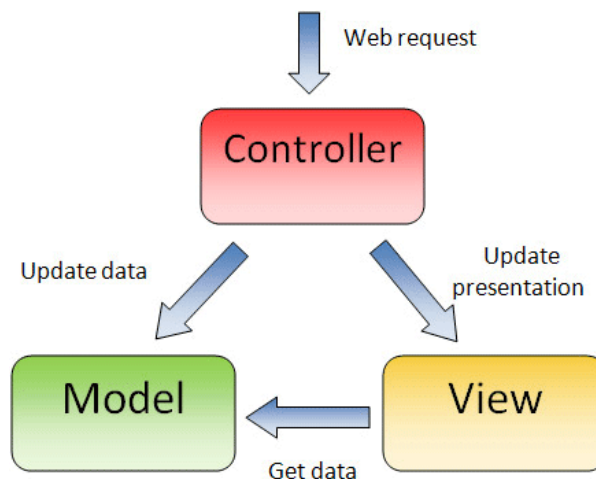


Fig. 1 Model View Controller [27]

## B. 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.

# International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirset.com](http://www.ijirset.com)

Vol. 6, Issue 8, August 2017

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].

## 1) *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].

## 2) *Key Performance Indicators:*

- 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 the application level. But, there has been a substantial amount of work done to analyze performance in-terms of the response time of applications. Mariadel Pilar Salas-Zárteet et al (2015) in the paper discussed 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 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]. The major objectives of this paper are (1) Reduction in development time, (2) Reusability & Maintainability of code.

VarshaApte 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 an application. Design patterns that improve reliability are discussed and applied to 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 Arlitz 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, AnupamChanda, 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

# International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirset.com](http://www.ijirset.com)

Vol. 6, Issue 8, August 2017

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].

AE-Commerce performance benchmarkis 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].

Varun Kumar and Dr. Vinay Chopra et al (2016)in the first part of this paper demonstrates the performance comparative analysis of the PHP frameworks – Symfony and Yii – on three parameters, Response Time, Throughput and CPU Utilization. It concludes that Symfony is better when throughput is of main concern, while Yii gave good Response time for 300 samples[28].

Varun Kumar and Dr. Vinay Chopra et el (2017) in the second part of this concluding paper performed comparative analysis on performance for three frameworks – Django, Yii, and Symfony. In this paper, the frameworks were evaluated on 3 parameters - Throughput, Response Time, and CPU Utilization. It concludes that Django is better in comparison to PHP frameworks. It was also found that Django uses more bytes for the same type of task in comparison to PHP frameworks [29].

## APPLICATION BENCHMARK

We investigate the performance of the applications built on Symfony and Yiibacked by PHP 7 and deployed on WAMP- a Web development stack that comprising Apache, PHP and MySQL[17];RoRbacked byRuby2.3.3p222; Django backed by Python 2.7.12.

Hrnoide (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, four separate applications were developed, each with an aforementioned framework having similar user interface.They also share a common MySQL database. These four applications were then used to identify the performance of their respective frameworks.

TABLE I  
HRNOID APPLICATION TESTING PLATFORM

Framework	Language	Database	OS
Symfony2	PHP 7	MySQL	Windows
Yii			
Django	Python		
Ruby on Rails	Ruby		

### C. 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 Indicator) results.

# International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirset.com](http://www.ijirset.com)

Vol. 6, Issue 8, August 2017

### D. 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, 4GB 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.

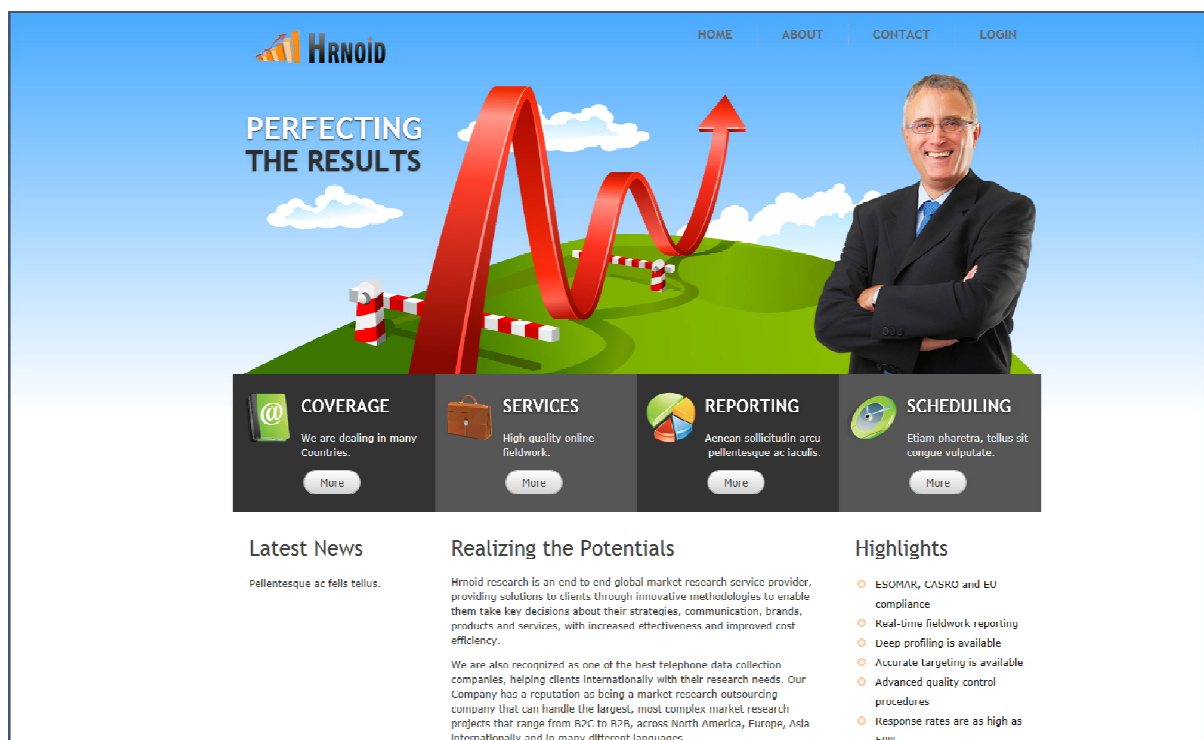
### E. Sample Configuration

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 with the Ramp-up period of 15 sec. This means that every 1.5 second a new thread starts, giving 10 running threads after 15 seconds. Each among 10 users will repeat 30 times in Loop Count. (Table-2).

So, a total number of 300 samples will be processed by each framework for different Key Performance Indicators. It's useful for testing **auto-scaling** - e.g. for most websites, it's unlikely that a website goes from 0 to 100 concurrent users instantly. It's more likely that concurrent users will gradually increase (and thereby give the auto-scaling system time to respond). Ramp-up enables this scenario to be tested

TABLE II  
SAMPLE CONFIGURATION

Number of Threads (users)	10
Ramp-Up Period (in Seconds)	15
Loop Count	30



F.

Fig.2HrnoId Application



# International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirset.com](http://www.ijirset.com)

Vol. 6, Issue 8, August 2017

## G. Findings

### 3) Response Time:

**F1.** Django took the least Response Time among other frameworks.

Total response time is the aggregate response of every sample. Symfony, Yii, Django & Rails applications took 96.9, 49.5, 9.3, and 36.3 seconds of response time (Figure-3) to process 300 samples. Django took the least Response time i.e., 9.3 seconds to service 10 users with 300 requests. Thus Django is around 4, 5 and 10 times faster than Rails, Yii and Symfony applications respectively.

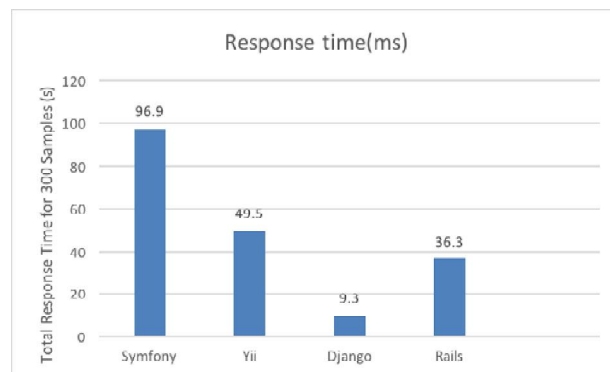


Fig.3 Total Response Time

**F2.** Django took the least Average Response Time among other frameworks.

Average response time is the Total response time divided by total no. of samples. As can be seen in (Figure-4), the total of 300 samples was sampled by 10 users. The average response time in Symfony, Yii, Django, and Rails was 323, 165, 31 and 121 ms respectively. Django application outperformed by the factor of 10, 5 and 4; the Symfony, Yii and Rails applications respectively.

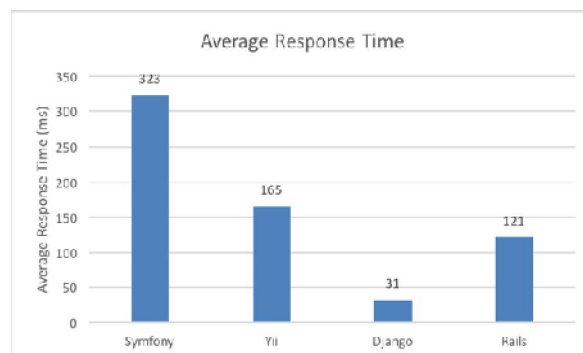


Fig.4 Average Response Time

**F3.** Rails' Maximum Response Time, out of all the samples, was lower as compared to other frameworks.

Maximum response time is the highest response time that was identified among the response times of 300 samples. (Figure-5) shows the maximum response time for Symfony was 1626 ms. While Yii had a Maximum Response Time of 611 ms. Django had a Maximum Response time of 494 ms. Among all these frameworks, Rails had the least Maximum Response time of 231 ms. So, Rails' Maximum Response Time outperformed Symfony, Yii, and Django by the factors of 7, 3 and 2 respectively.

# International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirset.com](http://www.ijirset.com)

Vol. 6, Issue 8, August 2017

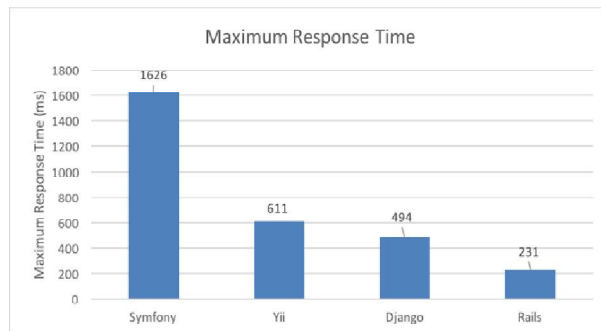


Fig.5 Maximum Response Time

**F4.** With every increase in the thread size (No. of concurrent users), Yii outperformed symfony. Rails outperformed both Symfony and Yii. Django outperformed all the 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 is increased. Response Time changes with anumber of parallel threads. Naturally, a server takes longer to respond when a lot of users request it simultaneously. As the no. of users are scaled, this parameter is used to identify the response times.

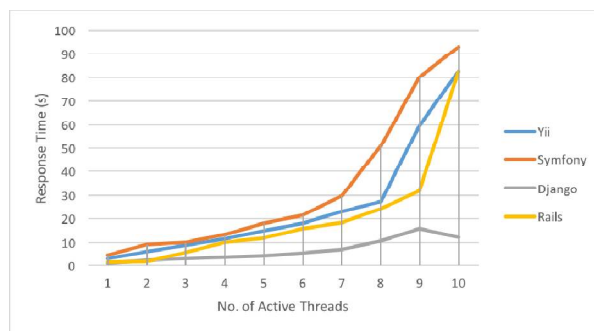


Fig.6 Response Time vs Active Threads

Threads in this scenario (Figure-6) 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 threads, the efficient an application is. Response time directly impacts the page load time of an application

TABLE III  
RESPONSE TIME VS ACTIVE THREADS

No. of Active Threads	Yii (ms)	Symfony (ms)	Django (ms)	Ruby on Rails (ms)
1	2.94	4.5	0.81	1.59
2	6.06	9.12	2.4	1.86
3	8.91	10.08	2.97	5.85
4	11.76	13.32	3.72	9.96
5	15	18.15	4.35	12

# International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirset.com](http://www.ijirset.com)

Vol. 6, Issue 8, August 2017

6	18.18	21.6	5.58	15.84
7	22.89	29.82	7.14	18.48
8	27.36	51.12	10.8	24.24
9	59.67	80.19	15.93	31.86
10	82.8	92.7	12.3	82.784

From Table III, it can be observed that when the No. of the active thread was 1, Django took the least, 0.81ms to fulfill the requesting sample. Yii, on the other hand, took the maximum response time of 2.94ms, to fulfill the same requesting sample of active thread 1. As the No. of active threads increased to 10, Django took the least Response time of 12.3ms and outperformed other frameworks, Yii, Symfony & Rails by the factor of 7, 8 & 7 respectively. The Response times of Rails and Yii were same i.e., 82.3ms when the active threads were 10.

#### 4) Throughput:

**F1.** Django processed more samples per second as compared to other frameworks.

Throughput identifies the maximum capacity of an application to handle the requests. As can be seen in (Figure-7), Symfony, Yii, Django & Rails processed 12.8, 16, 20.5 & 17.7 samples per second respectively. Symfony scored the least throughput in comparison to other frameworks, thus it can be safely said that it can handle lesser No. of Responses per second than other frameworks.

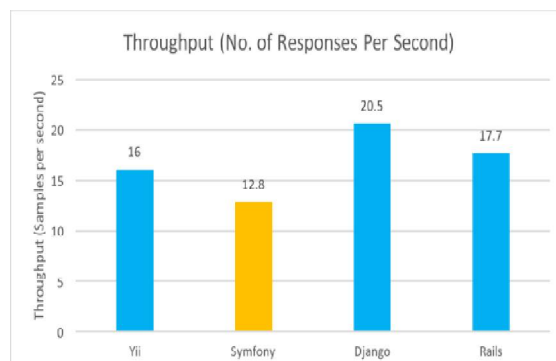


Fig. 7 Throughput

**F2.** Django scales well when the No. of active threads increases.

In this finding, the total throughput of the samples with respect to the increase in the thread size or when the number of users is increased is identified. It was found that the Django application outperformed other frameworks when the No. of active threads were increased. It can also be said that Rails performed better when the No. of active threads was less as compared to other frameworks. Symfony and Yii by an average factor of 5.25 and 7 respectively (Figure-8). Please refer Table-IV.



# International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirset.com](http://www.ijirset.com)

Vol. 6, Issue 8, August 2017

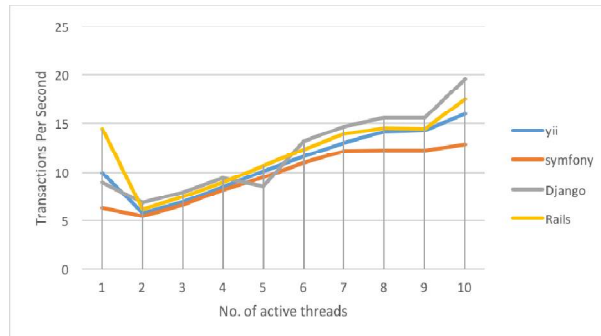


Fig.8 Throughput vs Threads

It was found that when no. of the thread was 1, Rails had given the best throughput. With the increase in active threads, in all the cases Django outperformed other frameworks except when the active threads were 5. Symfony's throughput, although increased with the increase in the threads, it was still nowhere in comparison to other frameworks. Yii performed equally well in comparison to Rails and Django.

TABLE IV  
SCALABILITY: THROUGHPUT VS ACTIVE THREADS

No. of Active Threads	Yii	Symfony	Django	Ruby on Rails
1	10	6.4	9	14
2	5.8	5.5	6.9	6.2
3	7	6.7	8	7.5
4	8.5	8.2	9.4	9
5	10	9.5	8.6	10
6	11	11	13.1	12
7	13	12.1	14.7	13
8	14	12.2	15.6	14
9	14	12.2	15.6	14
10	16	12.8	19.6	17

5)

6) CPU Utilization:

**F1.PHP Applications:** Symfony & Yii, utilized more CPU in comparison to Django & Rails applications.

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. Yii's CPU Utilization was relatively lower than Symfony when the number of users was less than 4, but as the concurrent users exceed the threshold of 4, Symfony's CPU utilization was equivalent to Yii application as shown in (Figure-9), (Table V). Both Django & Rails performed better and utilized lesser CPU than PHP applications. Between Rails and Django, former takes the lead.

# International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirset.com](http://www.ijirset.com)

Vol. 6, Issue 8, August 2017

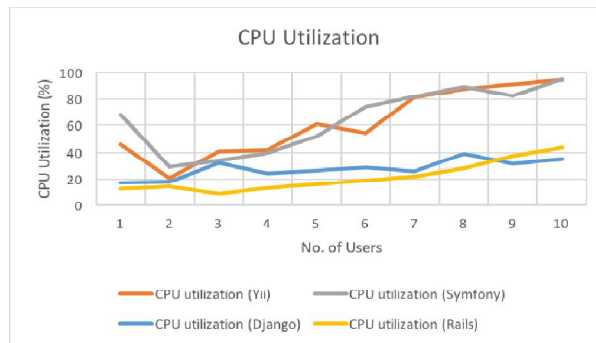


Fig.9 CPU Utilization

Rails' Utilization was found to be relatively lower in all cases (Active Thread increase), hence better. At the end, the average CPU utilization of Symfony, Yii, Django & Rails was 65%, 62%, 27% and 21% out of the total 100% processing power of CPU respectively. In all the cases, the increased concurrent user requests may cause a 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. In our test case, PHP applications shown CPU bottlenecks as the number of users were increased, while there weren't any bottlenecks for Rails and Django.

TABLE V  
CPU UTILIZATION VS ACTIVE THREADS

CPU Utilization	Yii	Symfony	Django	Ruby on Rails
1	45.9	68	17. 4	12. 8
2	20.7	28.9	17. 7	14. 6
3	40.7	33.9	32. 2	8.5
4	41.6	39.2	23. 9	13
5	61	51.7	26. 3	15. 9
6	54.1	74.5	28. 7	18. 9
7	81.4	82.4	25. 6	22. 1
8	88	89.4	39. 1	27. 8
9	91.3	83	31. 2	37. 4
10	94.8 2	95.5	34. 8	17. 5

### 7) Efficiency:

**F1.** Django and Rails both performed better in Efficiency as compare to PHP Applications: Symfony & Yii. Efficiency [12] is usually defined as throughput divided by utilization. When comparing two components, if one has a higher throughput at the same level of utilization, it is regarded as more efficient. If both have the same throughput but one has a lower level of utilization that one is regarded as more efficient.

# International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirset.com](http://www.ijirset.com)

Vol. 6, Issue 8, August 2017

Another way of identifying Efficiency in any application is by calculating its number of Bytes/s that it can process in request response cycle.

TABLE VI  
EFFICIENCY

	Yii	Symfony	Django	Ruby on Rails
Sample Size	300	300	300	300
Bytes	10603	10741	12778	12356
Throughput	16	12.8	20.5	17.7
Average Utilization	0.6195	0.6465	0.2769	0.2142
Efficiency (Throughput/Utilization)	25.83	19.8	74.03	82.63
Efficiency (KB/s)	165.67	134.26	255.81	213.58

Bytes/s of every framework is then compared with one another for performing Performance Comparative Analysis.

The results stated that despite being higher in throughput, Django came out to be less efficient in contrast to Rails in terms of Throughput/Utilization due to its 27% of CPU utilization to fulfill the assigned task, whereas Rails took 21% of CPU for the same task. Both PHP frameworks performed least Efficiently due to the CPU bottlenecks that arose as the number of active threads increased. (fig. 10)

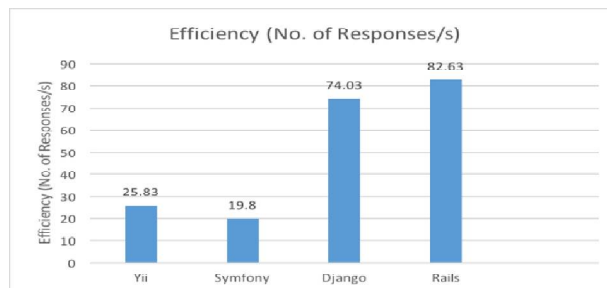


Fig. 10 Efficiency (Throughput/Utilization)

Django on the other hand performed better in contrast to Rails due to its higher Throughput in terms of Efficiency in Bytes/s. In this scenario, utilization was not taken into consideration. (fig. 11)

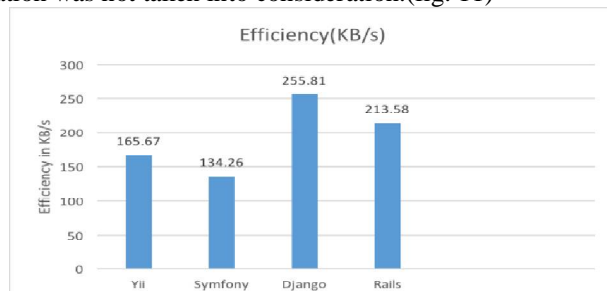


Fig. 11 Efficiency (KB/s)

# International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirset.com](http://www.ijirset.com)

Vol. 6, Issue 8, August 2017

## 8) Latency:

**F1.**Django and Rails both performed better in Efficiency as compare to PHP Applications: Symfony & Yii.

Latency is a difference between the time when the request was sent and time when the response has started to be received. Response time (= Sample time = Load time = Elapsed time) is a difference between time when the request was sent and time when the response has been fully received. So Response time always  $\geq$  latency. Therefore, Response Time = Latency + Processing Time.

In this test, Django took the least amount of time in latency per second to server 300 samples. Whereas Symfony took maximum time to serve the same number of samples in latency per second.

Thus Django performed better in-terms of Latency among all other frameworks.

TABLE VI  
LATENCY

Frameworks	Latency (sec)
Yii	47.03
Symfony	93.54
Django	28.4
Ruby on Rails	8.2

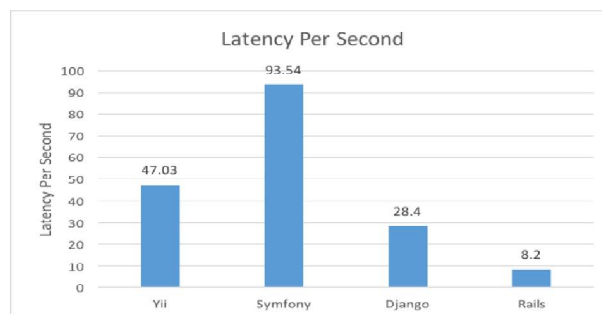


Fig. 12 Latency Per Second

## CONCLUSION AND FUTURE SCOPE

### H. Conclusion

Performance comparative analysis was done between four well-known PHP frameworks: Symfony & Yii; a Python Framework named Django; and Ruby on Rails, which is a ruby based framework. It was found that Yii outperformed Symfony in Total Response time, Average Response Time and Maximum Sample Response time. With all these results, it can be said that, as far as PHP frameworks are concerned, Yii has the higher capacity to process more samples per second which make it an obvious choice for enterprise applications. Yii application turns out to be an obvious choice when page load time and higher throughput is of main concern among PHP frameworks. Rails, on the other hand, outperformed symfony by 10 seconds for about 3 MB of data, whereas, in terms of throughput, the difference among both isn't much. The main difference between Rails and Yii comes in case of the CPU utilization; the former outperformed the latter by a factor of 3. Django outperformed both Rails and PHP frameworks in all the parameters, thus making it an obvious choice over other frameworks. In case of Response time, Django outperformed Ruby by a factor of 3. The maximum Response time recorded was lowest for Rails. Ruby also utilized least CPU among other frameworks while giving almost equal throughput as Django had. In-terms of Efficiency (throughput/utilization), Rails was more efficient to Django, while for Efficiency (bytes/s), Django outperformed Rails. As far as Latency is concerned, Django took only 8.2 seconds to fulfill the requests of 300 sample users and outperformed other frameworks with a minimum factor of 3.

At the end, it can be concluded that in terms of major KPI's – Response time, Latency, Efficiency, Throughput, Utilization – Django and Rails outperformed PHP frameworks with the exception that Yii performed some what comparative to Rails in many performance parameters. While it also cannot be neglected that PHP frameworks took the least number of bytes to perform the same task as Django and Rails did, in simple terms, their application was relatively smaller in size to Rails and Django. As far as

# International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirset.com](http://www.ijirset.com)

Vol. 6, Issue 8, August 2017

scalability in-terms of throughput is concerned, all the frameworks performed equally well where Django outperformed others with a meager margin.

## I. Future Scope

In this study, Key Performance Indicators, Response time, Throughput, CPU Utilization, Efficiency, and Latency are used for the performance comparative analysis. In future, Scalability, Availability, Processing time etc., can be used for a more robust performance identification. Scalability identifies how good a framework performs when no. of concurrent users is increased. Processing time or Processing speed is an indicator of the power of a processor and the resources allocated to it. Availability is the less downtime. As the availability increases so do the reliability. So it can be said the availability is the factor of reliability. In future, more frameworks, such as struts, Spring MVC, ASP.NET MVC etc., along with aforementioned Key Performance Indicators can be used for more extensive performance comparative analysis.

TABLE VII  
KEY PERFORMANCE INDICATORS: PERFORMANCE COMPARATIVE ANALYSIS CHART

Key Performance Indicators	Description	Frameworks			
		Yii	Symfony	Ruby on Rails	Django
No. of Samples	Total No. of requests from clients to a server	300	300	300	300
Bytes (Lower is better)	AverageByte transmitted per sample	10603	10741	12356	12778
Total Bytes (MB) (Lower is better)	TotalBytes transmitted by 300 samples: (No. of Samples * Bytes)/(1024*1024)	3.03M B	3.07M B	3.5MB	3.6MB
Total Response Time (Lower is better)	Gross of Total Response Time that each of 300 samples took	49.5 Sec	96.9 Sec	36.3 Sec	9.3 Sec
Average Response Time (Lower is better)	Total Response Time/No. of samples	165ms	323ms	121ms	31ms
Maximum Response Time (Lower is better)	Maximum response time of a sample in 300 samples	611ms	1626ms	231ms	494ms
Throughput (Higher is better)	No. of Responses per Second	16	12.8	17.7	20.5
Scalability: Throughput vs Active Threads (average) (Higher is better)	Throughput as the No. of Samples/Users Increases	11	10	12	12
CPU Utilization (Lower is better)	A measure of the amount of work a processor can handle in different load scenarios	61.952 %	64.65%	21.42%	27.69%
Efficiency (Higher is better)	Throughput/Utilization	25.83	19.8	82.63	74.03
Efficiency (KB/s) (Higher is better)	(Average Total Response Time * Throughput)/1024	165.67	134.26	213.58	255.81
Latency (Lower is better)	Latency = Response Time- Processing Time.	47.03 sec	93.54	8.2	28.4

# International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirset.com](http://www.ijirset.com)

Vol. 6, Issue 8, August 2017

## REFERENCES

- [1] Jones, Paul M. 'Pmjones/Adr'. *GitHub*. N.p., 2015. Web. 23 Oct. 2015.
- [2] Salas-Zárate, María del Pilar et al. 'Analyzing Best Practices On Web Development Frameworks: The Lift Approach'. *Science of Computer Programming* 102 (2015): 1-19. Web.
- [3] Pop, Dragos-Paul, and Adam Altar. 'Designing An MVC Model For Rapid Web Application Development'. *Procedia Engineering* 69 (2014): 1172-1179. Web.
- [4] Apte, Varsha, Tony Hansen, and Paul Reeser. 'Performance Comparison Of Dynamic Web Platforms'. *Computer Communications* 26.8 (2003): 888-898. Web.
- [5] Khan, Umar, and T.V. Rao. 'XWADF: Architectural Pattern For Improving Performance Of Web Applications'. *IJCSI International Journal of Computer Science* 11.2 (2014): n. pag. Print.
- [6] Titchkosky, Lance, Martin Arlitt, and Carey Williamson. 'A Performance Comparison Of Dynamic Web Technologies'. *ACM SIGMETRICS Performance Evaluation Review* 31.3 (2003): 2-11. Web.
- [7] Wikipedia,. 'Symfony'. N.p., 2015. Web. 23 Oct. 2015.
- [8] Yiiframework.com,. 'The Definitive Guide To Yii 2.0'. N.p., 2015. Web. 23 Oct. 2015.
- [9] Jmeter.apache.org,. 'Apache Jmeter - User's Manual: Glossary'. N.p., 2015. Web. 23 Oct. 2015.
- [10] Jmeter.apache.org,. 'Apache Jmeter - Apache Jmeter™'. N.p., 2015. Web. 23 Oct. 2015.
- [11] Sevcik, P., and J. Bartlett. 'Understanding Web Performance'. *Business Communications Review* 31.10 (2015): 28-36. Print.
- [12] Mozilla Developer Network,. 'Writing Efficient CSS'. N.p., 2015. Web. 24 Oct. 2015.
- [13] Community.dynatrace.com,. 'Best Practices On Web Site Performance Optimization - Public - Dynatrace Community'. N.p., 2015. Web. 24 Oct. 2015.
- [14] W3schools.com,. 'CSS Image Sprites'. N.p., 2015. Web. 24 Oct. 2015.
- [15] Sexton, Patrick. 'How To Combine External Javascript For Faster Pageloads'. *Varvy.com*. N.p., 2015. Web. 23 June 2015.
- [16] W3techs.com,. 'Usage Statistics And Market Share Of Server-Side Programming Languages For Websites, October 2015'. N.p., 2015. Web. 24 Oct. 2015.
- [17] WampServer,. 'Windows, Apache, Mysql& PHP Stack'. N.p., 2015. Web. 24 Oct. 2015.
- [18] A. C. and C. A., 'Specification and implementation of dynamic Web site benchmarks', *Workload Characterization, 2002. WWC-5. 2002 IEEE International Workshop on*, 2015.
- [19] U. Ramana and T. Prabhakar, 'Some experiments with the performance of LAMP architecture', *The fifth international conference on computer and information technology*, 2005.
- [20] Php.net, 'PHP: Hypertext Preprocessor', 2015. [Online]. Available: <https://www.php.net>. [Accessed: 31- Oct- 2015].
- [21] Don't repeat yourself - Wikipedia [online]. Available: [https://en.wikipedia.org/wiki/Don%27t\\_repeat\\_yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself)
- [22] Kevin McArthur, *Pro PHP: Patterns, Frameworks, Testing and More*, Apress, 2008.
- [23] E. Cecchet, A. Chanda, S. Elinikety, J. Marguerite, and W. Zwaenepoel, "Performance Comparison of Middleware Architectures for Generating Dynamic Web Content", *Proceedings of 4<sup>th</sup> Middleware Conference*, Rio de Janeiro, Brazil, June 2003.
- [24] D. Garcia and J. Garcia, "TPC-W E-Commerce Benchmark Evaluation", *IEEE Computer*, Vol. 36, No. 2, pp. 42-48, February 2003.
- [25] Banga and Drushel, Measuring the Capacity of a Webserver, *USENIX Symposium on Internet Technologies and Systems*, Monterey, California, December 1997.
- [26] Netcraft | Internet Research and PCI Security Services [online]. Available: <https://www.netcraft.com>
- [27] The Definitive Guide to Yii | Yii PHP Framework [online]. Available: [www.yiiframework.com/doc/guide](http://www.yiiframework.com/doc/guide)
- [28] Kumar, V., &Chopra, V. (2016). Design & Implementation Of Jmeter Framework For Performance Comparison In Web Applications (ISBN. 9788193137383, Pp. 403-412). Goa: International Interdisciplinary Conference On Engineering Science &Management.
- [29] Kumar, V., &Chopra, V. (2017). Design & Implementation of Jmeter Framework for Performance Comparison in PHP & Python Web Applications (ISBN. 9780998900001, Pp. 85-94). Goa: International Interdisciplinary Conference on Science Technology Engineering Management Pharmacy and Humanities.