International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2019): 7.583

Comparison and Evaluation of Web Development Technologies in Node.js and Django

Dr. Anupam Sharma[1], Archit Jain[2], Ayush Bahuguna[3], Deeksha Dinkar[4]

[1, 2, 3, 4]Dr. Akhilesh Das Gupta Institute of Technology & Management Affiliated to Guru Gobind Singh Indraprastha University, Dept. of Computer Science Engineering, Shastri Park, Delhi, India
[1]*anupam.sharma[at]adgitmdelhi.ac.in*
[2]*archit.abj[at]gmail.com*
[3]*ayushbahuguna217[at]gmail.com*
[4]*deeksha.dinkar[at]gmail.com*

Abstract: ***Highly scalable with effective concurrency and able to manage an enormous amount of data is the need and an expectation from this generation of website technology. Node.js and Django both have been highly admired and have been proven successful in the rapid development of effective and scalable web applications. To study and compare the advantages and the differences between both the technologies, using testing techniques, basic facts and figures. Node.js is a lightweight and efficient technology for building data-intensive web applications whereas Django is a developer-friendly technology, used for rapid development and providing well organized and clean code. The open-source, fast and easily manageable nature of both technologies brings up a little more difficulty in comparing them. The result will provide valuable information regarding both of the technologies and will contrast on some of the major features of both.***

Keywords: Web Development; Node.js; Django; Comparison Evaluation; Benchmark Testing

1. Introduction

Development of the Web is advancing at an aggressive rate. Due to rapid advancement, many websites face the problem of concurrency and multiuser requests. In web there is 80 to 20 ratio that it takes most of the time for the frontend to load rather than backend but when there is a spike in traffic or concurrent users visit the website so this ratio changes to 60 to 40. So, the emphasis should also be given to backend while developing websites. In recent times, Python has gained so much popularity and is now the most widely used language in the world. Django is a Python technology for doing web development. Django is a free and open-source technology. With Django web apps can be developed within hours as it is easier to get started and an enormous amount of libraries make the development easier. In Django, we can focus on developing as it takes cares of Web Development without any need for reinventing the wheel. Django is fully loaded as it provides user authentication, content administration and many more tasks. Also, the important features of Django are its security, versatility and scalability. JavaScript is also the widely used language in Web development from providing various functionalities in the front end to developing back end. Node.js is used for developing the backend of websites. Node.js is a platform which is built on Chrome's JavaScript runtime to create scalable network applications. Node.js is lightweight and efficient as it makes use of an event-driven model perfect for real-time applications. Many researchers have done the work related to the evaluation of Web technologies performance. But our works differs from others as we are going to perform various loading tests to evaluate the performance of different Web technologies. This paper mainly focusses on the impact on Web performance from the two different technologies which are widely used Node.js and Python's Django. We are mainly concerned with the performance of supporting concurrent users. The security and scalability issues are not considered in this paper.

2. Experimental Environment

a) Hardware Environment

We set up a test bed consisting of a single machine including both client and server. The machine in our test runs on Windows 10 Home with an Intel® Core™ i7-7700HQ CPU @2.80GHz 2.81GHz processor, 8GB 2400MHz of RAM and a 1TB SSH& 256GB SSD storage hard drives. In addition, all non-essential processes on the machine were disabled to minimize the consumption of resources and kept fairness in our test.

b) Server Configuration

The primary objective of our study is to compare the Web development technologies, Django and Node.js. So, we choose several different modules to build these environments as follows. The testing tool Apache 2.4.4.1.

- *Apache:* Apache 2.4.4.1 uses a hybrid thread and process model in an attempt to improve the server's performance, and it's more scalable. The stable version that recently released.
- *Django:* Django 3.0.2 is a Python technology for doing web development. Django is a free and open-source technology. With Django web apps can be developed within hours as it is easier to get started and many libraries make the development easier.
- *Node.js:* Node.js 10.16.3 is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient and perfect for data-intensive real-time applications that run across distributed devices. We have chosen the "Express" framework to perform the tests in our experiment.

c) Testing Tools

We performed benchmark testing. We have used two testing tools in our experiment to make results more accurate. As to

benchmark tests, we use ApacheBench (Ab), a tool in Apache. Ab can make requests in a local Web server to ensure that the time is just processing time, not including data transmission time on the Internet or calculation time in local machine.

- ApacheBench 2.3: The stress testing tool in Apache 2.4.4.1. It is a black box of web server performance testing.
- Siege 3.0.5: Siege is a load testing and benchmark utility. It let the Web Developers measure the code under traffic to see how it perform while loading.

3. Test Methodology

The experiment evaluated the results based on benchmark testing using two different testing tools. In all the tests, we must follow one factor at a time experimental design to ensure the accuracy and effectiveness of tests.

a) Benchmark Test Methodology

According to one-factor-at-a-time experimental design, we make three fundamental tests – "Hello World", "Loading an Image" and "Calculating Harmonic Progression Summation". "Hello World", a fundamental module, is used to display 'Hello World'on a web server and helps to distinguish between the two technologies. "Loading an Image" module uses HTML to display the image on a web server and evaluate the performance of the Html page and image URL under their loading time. "Calculating Harmonic Progression Summation" is a module used for evaluating and comparing the performance based on computation performed for a series consisting of a random amount of numbers (greater than 10,000).

- Apache Bench Benchmark Test Methodology: Under this benchmark test for all the modules, we keep requests number to 10,000 and then we change concurrent users from 10 to 1000. TABLE I summarizes all the factors in our experiment.
- Siege Benchmark Test Methodology: Under this benchmark test for all the modules, we run the tests for 100 seconds and keep the delay of 1 second and then we change concurrent users from 10 to 100. TABLE II summarizes all the factors in our experiment.

Table I: Apache Bench Benchmark Test Factors

Requests	10,000
Concurrent Users	10, 100, 200, 500, 1000
Benchmark Test Module	Hello World, Loading an Image and Calculating Harmonic Progression Summation
Web Development Technology	Node.js (JavaScript), Django (Python)

Table II: Siege Benchmark Test Factors

Time Period	100 seconds
Delay	1 second
Concurrent Users	10, 25, 50, 75, 100
Benchmark Test Module	Hello World, Loading an Image and Calculating Harmonic Progression Summation
Web Development Technology	Node.js (JavaScript), Django (Python)

4. Experimental

a) Results and Analyses of Apache Bench Benchmark Tests

1) "Hello World" module

With the growth of concurrent users, the performance of Node.js first increases then decreases whereas Django shows a steep decrease in performance when jumping from 10 to 100 users, after that its performance remains almost constant. We have got highest "requests (mean) per second" in Node.js at 200 users i.e. 1845.52 and for Django at 10 users i.e. 462.73. The "time (mean) per request" is almost constant for Node.js i.e. around 0.55 for all users whereas there is a steep jump again in Django from 2.16 at 10 users to 29.9 at 100 users and then remaining almost constant afterwards.

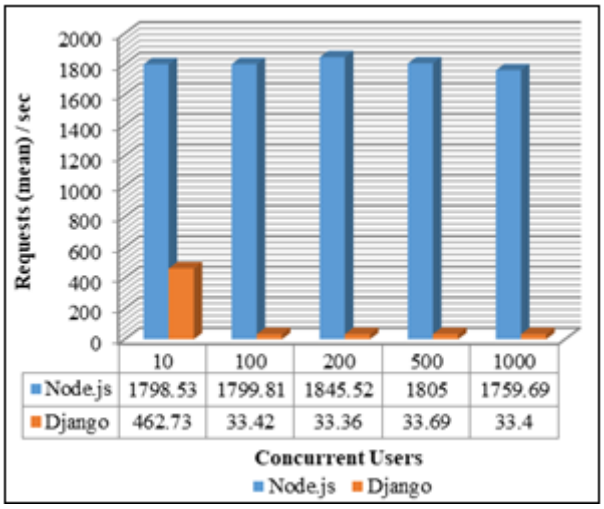

Figure 1: Results depicting "Hello World" module's requests (mean) per second

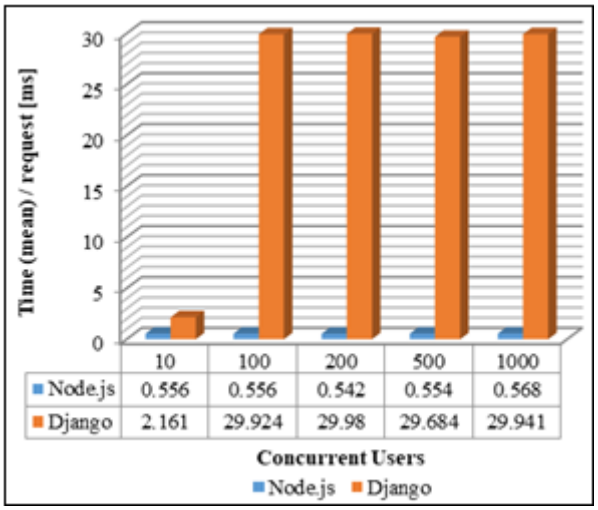

Figure 2: Results depicting "Hello World" module's time (mean) per request

2) "Loading an Image" module

This module has a similar trend as seen in the "Hello World" module. Here again, we have got highest "requests (mean) per second" in Node.js at 200 users i.e. 1645.34 and for Django at 10 users i.e. 340.70. The "requests (mean) per second" varies in the range of 1580 to 1650. The "time (mean) per request" also follows a similar trend where its

highest value for Node.js is 0.632 at 1000 users and for Django, it is 32.231 at 100 users.

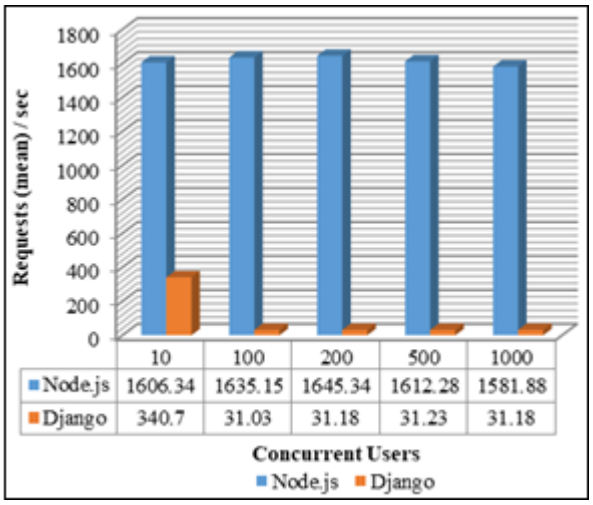

Figure 3: Results depicting "Loading an Image" module's requests (mean) per second

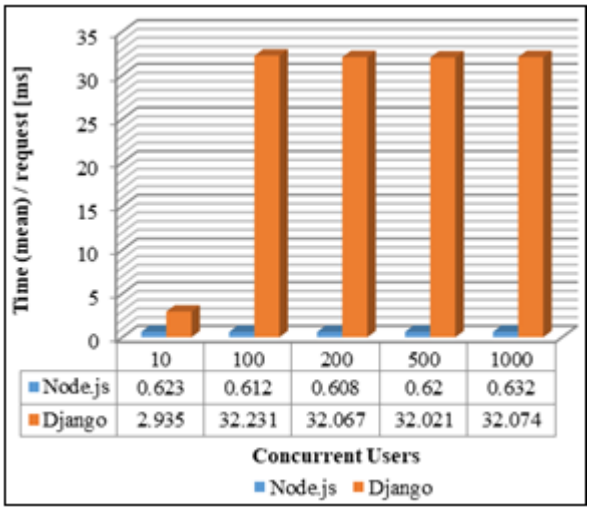

Figure 4: Results depicting "Loading an Image" module's time (mean) per request

3) "Calculating Harmonic Progression Summation" module

This module again shows the same trend as followed in "Hello World" and "Loading an Image" modules. The highest "requests (mean) per second" in Node.js goes again at 200 users i.e. 1672.51 and for Django again at 10 users i.e. 337.58. Similar trends are followed for "time (mean) perrequest" having their highest value for Node.js at 1000 users i.e. 0.606 and for Django at 100 users i.e. 36.438.

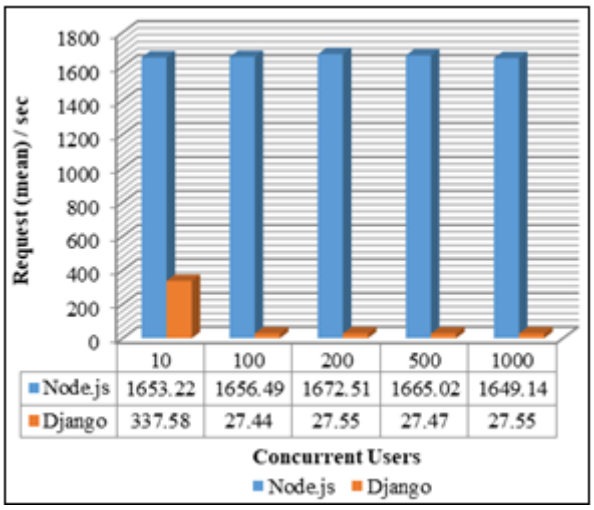

Figure 5: Results depicting "Calculating Harmonic Progression Summation" module's requests (mean) per second

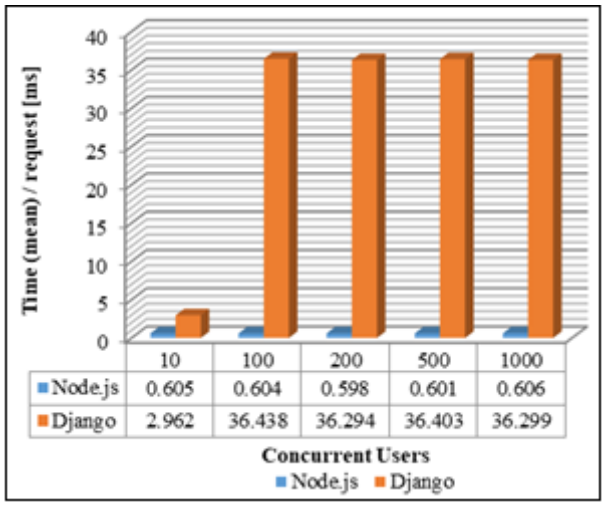

Figure 6: Results depicting "Calculating Harmonic Progression Summation" module's time (mean) per request

In short, the performance of Node.js is much better than Django in term of both "requests (mean) per second" and "time (mean) per request" parameters.

b) Results and Analyses of Siege Benchmark Tests

1) "Hello World" module

With the growth of concurrent users, there is an increase in performance for both Node.js and Django technologies. In terms of "transaction rate per second" Node.js has a slightly large number of transactions whereas in terms of "concurrency" clearly Django is a winner. The highest "transaction rate per second" is at 100 concurrent users i.e. 196.72 for Node.js and 192.1 for Django and the highest "concurrency" rate at again at 100 users is 0.79 for Node.js and 2.78 for Django.

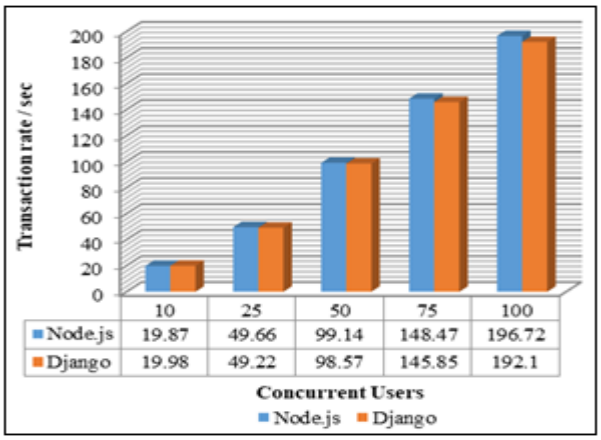

Figure 7: Results depicting "Hello World" module's transaction rate per second

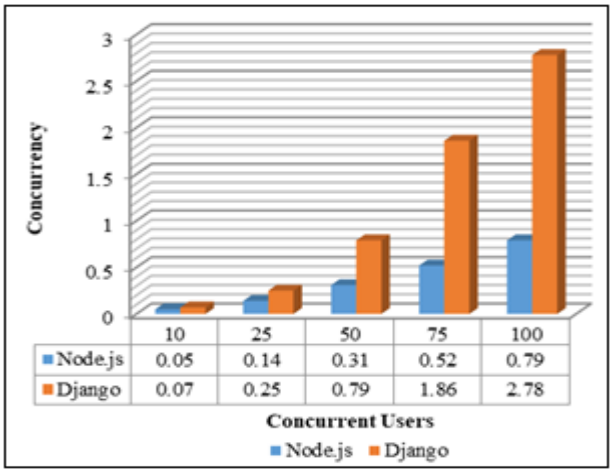

Figure 8: Results depicting "Hello World" module's concurrency

2) "Loading an Image" module

There is a similar trend followed as in "Hello World" module. The performance increases with the increase in the number of concurrent users. Again the highest "transaction rate per second" is at 100 concurrent users i.e. 196.97 for Node.js and 190.03 for Django and the highest "concurrency" rate at again at 100 users is 1.02 for Node.js and 4.52 for Django.

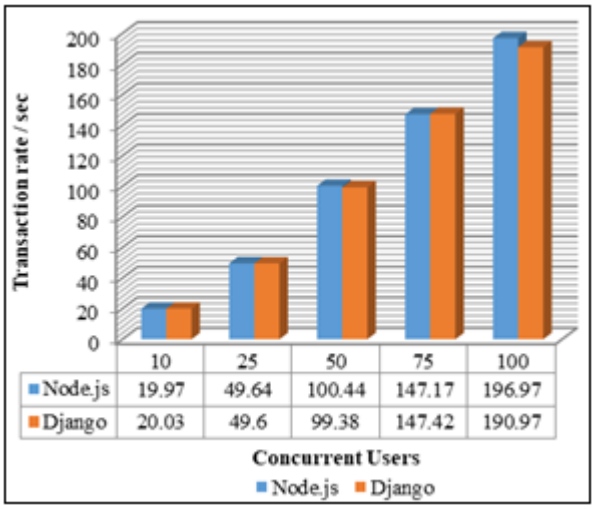

Figure 9: Results depicting "Loading an Image" module's transaction rate per second

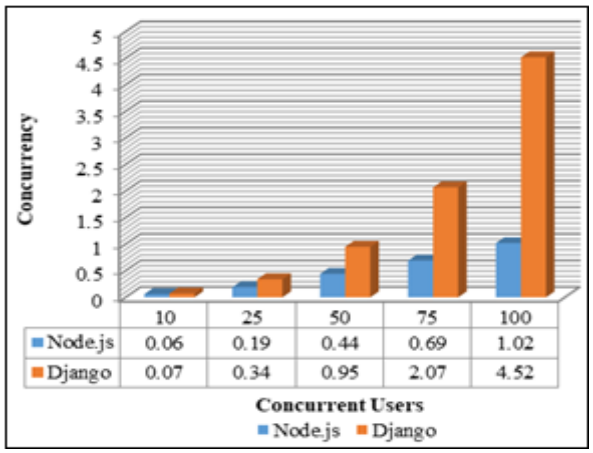

Figure 10: Results depicting "Loading an Image" module's concurrency

3) "Calculating Harmonic Progression Summation" module

There is a similar trend followed as in "Hello World" and "Loading an Image" modules. The performance here also increases with the increase in the number of concurrent users. Again the highest "transaction rate per second" is at 100 concurrent users i.e. 196.98 for Node.js and 188.85 for Django and the highest "concurrency" rate at again at 100 users is 0.76 for Node.js and 4.95 for Django.

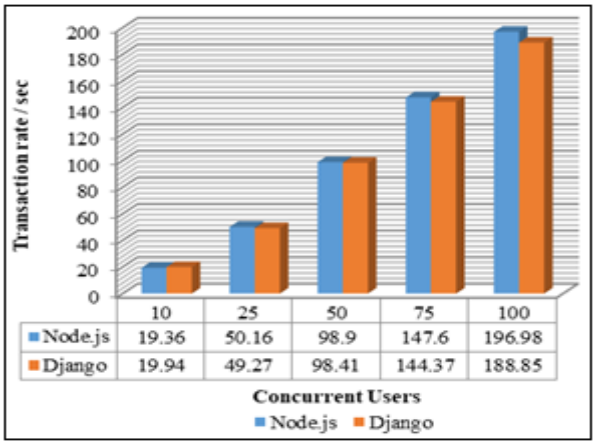

Figure 11: Results depicting "Calculating Harmonic Progression Summation" module's transaction rate per second

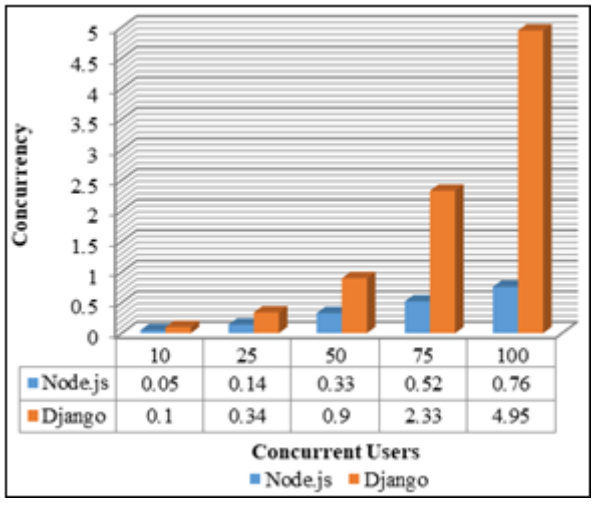

Figure 12: Results depicting "Calculating Harmonic Progression Summation" module's concurrency

In short, the transaction rate per second is a bit higher for Node.js than Django but the concurrency level is quite higher in Django than in Node.js

Comparison

Node.js and Django both are open-source backend technology as well as free to use but some important things which make these technologies differ from each other which are defined in TABLE III.

Table III: Comparison between Features of Node. Js and Django

Node.js	Django
It is a JavaScript runtime platform helps in easily fast building & scalable network application	It is a high-level Python technology that helps in rapid development & clean design
It uses a package manager which install, upgrade &remove libraries. The package manager used by Node.js is called Node Package Manager (NPM)	It follows the Model-Template-View (MTV) architecture pattern
It is more scalable	It is less scalable
It provides fewer security measures while developing. A developer needs to check it again and again	Security is very higher than Node.js. No need to check again and again
It is a less cost-effective and quite slow process	It is a very cost-efficient and fast process
Mainly used in small projects due to the absence of multi-threading operations	Mainly used in very large projects due to the presence of multi-threading operations
It is less complex because it gives the user their own way of developing code	It is more complex because it requires the user to follow a specific structure to solve the problem

5. Conclusion

This paper depicts a performance measurement study and comparison evaluation study of two Web Technologies. Both technologies are good for building web applications, however, there are some cases where one differs from another by some means. Django, for example, is good for users having proper knowledge about python, concerns about the security and fast development of web applications. While Node.js can be used for heavy lifting of client-side processing.

In short, Node.js performs much better than Django in terms of mean time required for requests, the number of requests passed and transactions performed in a specific time period for both small and large number of concurrent users. Django is only useful in obtaining a higher concurrency level.

6. Acknowledgment

We are grateful to all the anonymous reviewers for their valuable feedback and suggestions which helped in improvising this paper. We are also thankful to the Department of Computer Science Engineering of Dr Akhilesh Das Gupta Institute of Technology and Management for providing the opportunity and environment to conduct this research.

References

[1] https://nodejs.org/en/
[2] https://www.djangoproject.com/
[3] https://medium.com/@mihaigeorge.c/web-rest-api-benchmark-on-a-real-life-application-ebb743a5d7a3
[4] Kai Lei, Yining Ma, Zhi Tan, “Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js**”,** conference paper, December 2014.
[5] T.Lance, A.Martin and W.Carey, “Performance Comparison of Dynamic Web Technologies”, ACM SIGMETRICS Performance Evaluation Review, Volume 31 Issue 3, December 2003.
[6] Nimesh Chhetri, “A Comparative Analysis of Node.js (Server-Side JavaScript)”, March 2016.
[7] Prof. B Nithya Ramesh, Aashay R Amballi, Vivekananda Mahanta, “Django the Python Web Framework”, April-June 2018.
[8] Y.Hu, A.Nanda, and Q.Yang, “Measurement, Analysis, and Performance Improvement of the Apache Web Server", Technical Report No. 1097-0001, University of Rhode Island, 1997.
[9] Hezzbullah Shah, Tariq Rahim Soomro, “Node.js Challenges in Implementation”, May 2017.
[10] https://stackshare.io/stackups/django-vs-nodejs
[11] http://httpd.apache.org/docs/2.2/programs/ab.html
[12] https://www.joedog.org/siege-manual/
[13] http://httpd.apache.org/docs/2.4/platform/windows.html
[14] https://dzone.com/articles/nodejs-vs-djangois-javascript-better-than-python
[15] S.Tilkov, S.Vinoski, “Node.js: Using JavaScript to Build High-Performance Network Programs”, IEEE Internet Computing, 2010.