Performance Comparison and Evaluation Of Node.Js And Traditional Web Server (IIS)

Lakshmi Prasanna Chitra
Department Of Computer Science
GITAM University
Visakhapatnam, India
prassu2490@gmail.com

Ravikanth Satapathy
Department Of Computer Science
GITAM University
Visakhapatnam, India
raviks.mtech@gmail.com

Abstract – ***As the technological needs increase there is an increase in the necessity for information of users using the internet that has lead to the development of real-time web services. This paper analyses the differences of Node.js compared to traditional web server- IIS. Node provides a high performance, asynchronous event-based server. Node.js is event-based and uses single thread for its event loop, which allows developers to use asynchronous interfaces to do I/O operations. Traditional Web servers (IIS) follow the multiple thread request model. This paper shows how the architectural choices of Node.js and traditional web server brings an affect in the way applications perform that run on them. Web services operate over network throughput which is considered as a metric parameter for evaluation. They are developed both in node.js as well as in .NET which are hosted in Internet Information Services (IIS Server). Systematic tests are performed presenting different scenarios to make a comparison of the performance of Node and IIS. The performance evaluation results show some valuable performance data of Node and IIS Server in a certain time. The results show that lightweight and efficient nature of Node.js. A web server environment which can perform in heavily can be built with node. Hence it is good for developing web-service APIs.***

***Index Terms*—IIS, Node.js, Performance Evaluation, Throughput, Web Services.**

I. INTRODUCTION

With the growing need for data from the internet, the multiple users problem of requesting the server at a high concurrency level. Node.js is a new technology in JavaScript. It is a platform built on Chrome's JavaScript runtime used for constructing fast and growing network applications [5]. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. It is best suited for CPU intensive applications in real time that run across many devices across [5]. Large companies from the web industry like Yahoo [4] have contributed to the development of the platform, others like Microsoft [7] have integrated the support of Node.js into their cloud platform. Even though the project is in its initial stages and the software has not been widely used yet, the way it can be used in the development of network applications, by making use of popular technologies it has become a good alternative to the conventional platforms like IIS, Apache+PHP, Python [8,9] and Java application servers. This paper focuses on the performance comparison and evaluation of Node.js and traditional web server- IIS.

The Node.js platform comprises of four main parts: a language used in the application development, a runtime environment, a set of standard libraries and modules.

II. THE NODE.JS PLATFORM

This section contains a brief technical overview of the Node.js platform.

A. Architecture

The Node.js architecture is event based using a single thread executing application code. There is no application level concurrency as it executes on a single thread which means that all requests coming in are run by that thread itself. The development of application is simplified by this approach. The complex calculations and blocking input and output calls are executed in the event thread which do not allow the applications to handle other requests. In order to handle this problem Node.js' uses asynchronous I/O. Node.js uses the same pattern to block the input output functions, like requesting the database or some other sources like files. Callback function can be taken as an another argument and for which values are returned immediately. There is a difference in the execution model of Node.js and the execution model based on multiple threads where each connection of the client is handled by a separate thread. Multithreaded model is followed by traditional web platforms like IIS+C#. The threaded and event-based models for server applications are equivalent in their resource usage and performance, given efficient implementations. While the one-thread-per-request model may be a more natural way for the developers to think about the flow of their applications, the scalability of currently available thread-based mainstream server platforms is less than optimal and they are incapable of handling tens of thousands client connections efficiently without consuming extensive amounts of system resources. This kind of process allows Node.js to use resources of system in a more efficient way when compared to platforms with multi-threaded architecture.

B. Language

The platform's common language is JavaScript. JavaScript is a basic client side scripting language for web pages [5]. It is used in building dynamic web pages and has received many enhancements.

C. Runtime

There is a runtime environment provided by Node.js , just like how the JVM is present for Java. Node.js is

implemented in C++ . V8 is fast because of the features like just in time compilation to native code and in memory management. This looks after the programs that are running on the event thread. Internally it consists pool of threads as if the applications that are working under non-blocking I/O.

D. Libraries and Modules

A programming language without any libraries is useless for practical applications. Therefore Node.js has an API for IO operations, basic networking, HTTP server modules, compression and for many other required tasks. The libraries of Node.js that are available can be extended with additional packages. The packages are made available via public or private package registries. They can be installed using NPM package manager and are organized according to the Common JS package format. We can say that the unique combination of the JavaScript language along with an efficient runtime and the architecture based on events make node.js a very good platform for application developers.

III. PROPOSED WORK

To choose a suitable method for building web services. They are accessed over the network [1]. The performance parameter for evaluation is taken as throughput. Performance is calculated and comparison of Node.js and traditional web server- IIS is done. However, Lance and Martin experimentally evaluated the effect of three web technologies (Perl, PHP, and Java) on Web server performance [2].

REST based Web- Services are developed in Node.js version 6.2.2 using the Express framework in order to showcase the different situations like IO Intensive situation, CPU Intensive situations. The database used for IO Intensive situation is MySQL. Each service consists of its own Uniform Resource Indicator (URI).

A. REST Based Web- Services in .NET

REST based Web-Services are developed in .NET platform using C# Web API. Similar functionalities are developed to demonstrate the IO Intensive situations and CPU Intensive situation. The database used for IO Intensive situation is same as that used for Node.js i.e. MySQL.

B. Testing Tool

The performance is compared on doing systematic tests. The testing tool used to evaluate the performance is JMeter. This creates a group of users sending requests to a main server in which the application is deployed and it shows the performance results of the server/application. [12]

JMeter: Apache JMeter 2.13. It is a software that is used to test performance both on any type of resources. More stress on the server or network can be put by imposing load to make an analysis of the entire performance under different types of load.

IV. PERFORMANCE COMPARISON TEST METHOD

The performance comparison test method used for Node.js and IIS is Systematic tests. The results are evaluated and compared. In performing the result analysis, each module is taken at a time to make sure that the tests are effective and accurate. According to this type of design method, systematic tests are done, keeping the number of requests at 1000, the users are increased from 10 to 1000. This is done for three modules under the systematic tests. First one is to develop a Web server. Second module is to compare performance of Node.js and IIS in the IO-intensive situation by interacting with the database. Third module is developed to find the value of Fibonacci and compare performance in CPU intensive situation. For the second and the third module web services are developed in Node.js platform as well as in C# .NET. For all the modules, the users are initially taken as 10 and they are increased to 100,200,500 and 1000. The performance is compared for Node.js and IIS.

V. PERFORMANCE METRICS

Throughput is a metrics parameter for evaluation for Web services in a network. As discussed in the previous section, the number of requests is 1000 and the number of users making the requests is being increased from 10 to 1000. As the numbers of users requesting the web service go on increasing the change in the throughput is evaluated and the performance is compared in different situations.

Throughput is an important performance metric in the evaluation of performance of applications. It shows the number of transactions or requests that can be served in a certain period of time. It is used to check the load capacity of the server [13]. The throughput of Node.js server and traditional server- IIS is compared under different situations by increasing the number of users. In Fig 1 to Fig 3 the horizontal axis represents the throughput in bytes/minute and the vertical axis represents the number of users.

Response Time is the time it takes to respond to a request for service from an application like querying some data from the database, or to load a requested web page[13]. In this paper response time is calculated for web server when the user requests for a service.

VI. RESULTS AND ANALYSIS

A. Results and Analysis

1) Building a Web Server

As the users increase, the performance of Node.js and IIS server is analysed. When the throughput of them is compared, it can be clearly seen that the node.js has more throughput than IIS server in serving the web page requested by the users. Initially both the servers are tested with 10 users, the throughput of IIS is calculated as 34032.898 bytes/minute whereas Node.js the throughput is calculated as 34684.686 bytes/minute. As the number of users gradually increases, it can be clearly seen that Node.js throughput is greater than IIS. Fig.1 shows the comparative performance results in terms if throughput of Node.js and IIS .

2) IO Intensive Situation

In order to evaluate Node.js and IIS in an IO Intensive Situation, such a situation is designed in which a web service is developed in both the platforms which returns the requested record from the database. Hence we perform a database operation. For 10 users Node.js shows a throughput of 14363.001 bytes/minute whereas the web service hosted in IIS shows a throughput of 12526.881 bytes/minute. As the number of users increase to 100 the throughput is 21519.646 bytes/minute for Node.js. For the same number of users 10 and 100 , the throughput of IIS is calculated as 12526.881 bytes/minute and 19512.386 bytes/minute which is lesser than that of Node.js. Fig.2 shows the comparative performance results in terms if throughput of Node.js and IIS . It can be seen from the graph results that Node.js shows a greater throughput than that of IIS for data intensive situation.

3) CPU Intensive Situation

Now, Node.js and IIS are evaluated in a CPU Situation, where a Fibonacci value is calculated and returned to the user on request. Even in this scenario a REST web service is developed and performance is evaluated and compared. For 10 users Node.js shows a throughput of 24772.915 bytes/minute whereas the web service hosted in IIS shows a throughput of 28199.464 bytes/minute. As the number of users increase to 100 the throughput is 29063.149 bytes/minute for Node.js.

For the same number of users 10 and 100, the throughput of IIS is calculated as 28199.464 bytes/minute and 39434.768 bytes/minute which is lesser than that of Node.js.

Fig.3 shows the comparative performance results in terms if throughput of Node.js and IIS. With this, a conclusion can be made that Node.js is used mainly for IO-intensive applications and not for CPU intensive tasks because such kind of applications do not show much increase in the efficiency of Node.js.

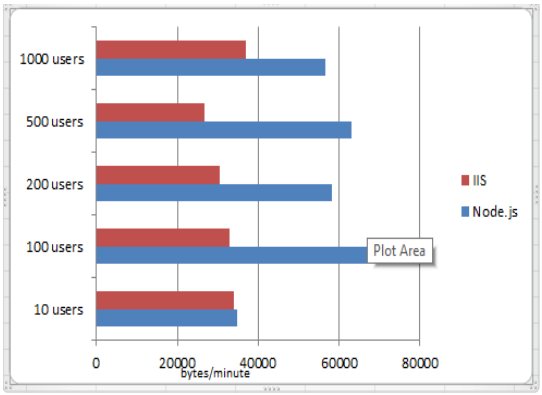

Fig. 1: Throughput trend

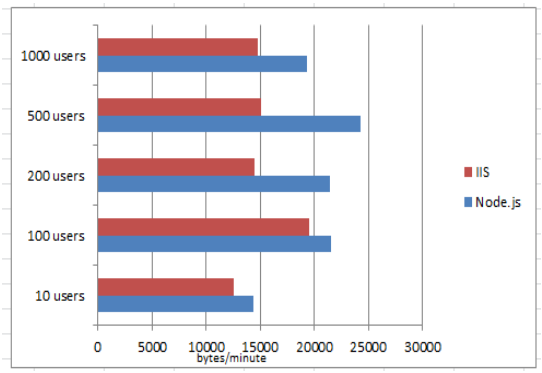

Fig 2: Throughput Trend for IO Intensive Situation

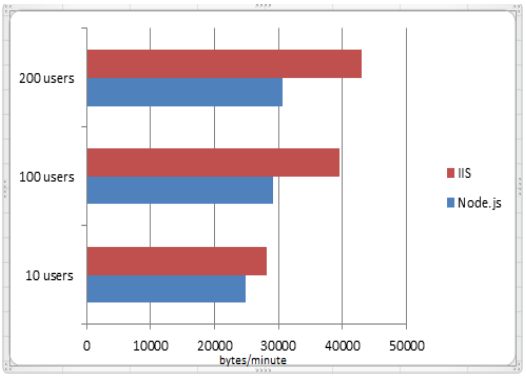

Fig 3: Throughput Trend for CPU intensive situation

VII. CONCLUSION

This paper shows a comparison and evaluation of the performance of Node.js which is the latest technology and IIS from objective systematic tests. Node.js shows a good performance than the traditional server IIS in the execution of IO intensive tasks. Besides, Node.js is used preferably in the IO-intensive situation, not CPU intensive services. By looking at the performance results in terms of throughput. Node.js is a technology that is now being used by most of the organizations because of its advantages in IO-intensive situation.

VIII. FUTURE WORK

Only the systematic tests have been used in this paper to make the comparison of the performance of Node.js and IIS by evaluating them in different scenarios. We have also considered the architecture design in the paper. Our future scope of work is focused on performing more tests like realistic user behaviour tests in order to evaluate the performance of Node.js and IIS. The paper mainly concerns the comparison of performance of both node.js and traditional web server – IIS.

IX. REFERENCES

[1] Toyotaro Suzumura, Scott Trent, Michiaki Tatsubori, Akihiko Tozawa and Tamiya Onodera," Performance Comparison of Web Service Engines in PHP, Java, and C", 2008 IEEE International Conference on Web Services

[2] T.Lance, A.Martin and W.Carey, "Performance Comparison of Dynamic Web Technologies", ACM SIGMETRICS

Performance Evaluation Review, Volume 31 Issue 3, December 2003.
[3] S.Tilkov, S.Vinoski, “Node.js: Using JavaScript to Build High-Performance Network Programs”, IEEE Internet Computing, 2010.
[4] Yahoo Developer Network. (2012) Multi-Core HTTP Server with NodeJS.
[5] http://Node.js.org/
[6] http://strongloop.com/developers/Node-jsinfographic/?utm_source=ourjs.com#3. Https://github.com/search?l=JavaScript&o=desc&q=stars%3A%3E1&s= stars&type=Repositories.
[7] http://www.PHP.net/.
[8] https://www.Python.org/.
[9] Google, Inc. V8 JavaScript Engine. [Online]. Available: http://code.google.com/p/v8/
[10] Z. Schlueter. The NPM and Registry. [Online]. Available: https://npmjs.org/
[11] http://jmeter.apache.org/
[12] https://en.wikipedia.org