SMART STOCK MARKET ANALYSIS USING NODE.JS AND AJAX

[1]K. Anitha Kumari, [2]N. Dharun Kumar, [3]N. Rajeswari, [4]Maria Ceceli Brittinna, [5]R. Shanmuga Priya
[1]Asst. Professor, [2,3,4,5]UG Scholars, Dept. of IT,
PSG College Of Technology, Coimbatore, India

Abstract
The virtual kind of stock exchange electronically composes a network of computers involves traders to make trade at ease. One of the highest challenges in stock exchange market is adverse selection; i.e., asymmetric information transfer among users. Therefore, in this paper, we intend to use Node.js in our application to make it fast and scalable. In addition, with the facilitation of Node.js, server workload is reduced to a greater possible extent as it is a single threaded scripting language. Use of such scripting is very useful in data intensive and multiple client applications such as share market bidding. The notification mechanism (callback) of events using Node.js makes the server not to wait for any operation at any point of time. The resultant values of the application retrieved using Node.js is compared with the multithreaded application.

Keywords: Smart stock market analysis, Node.js, AJAX, Asynchronous thread

I. INTRODUCTION

People today are technology favourable, expects a website to be loaded in less than 2 seconds and they abandon the sites that take more than 3 seconds [1].In this inevitable situation, web applications require an efficient access without degradation in performance. Smart Stock Market Analysis (SSMA) is a website that keeps the user updated with the current trends in the stock market. Users are notified with the current updates, and privilege is given to view various information like Bombay Stock Exchange (BSE) and National Stock Exchange (NSE), charts like Sensex and Nifty. Asynchronous JavaScript and XML (AJAX) provides a better user interface simplifying the task of transferring information from the client to the server, helpful in creating asynchronous web applications [2]. Node.js provides faster access to the website in the client side and reduces the server load on the server side [15]. So, the major focus of the application being developed is on the business class people who participate in stock exchange and related processes. The business class people rely on to the web applications for the correctness of information and they require smoothness of accession pertaining to higher degree continuous performance [3]. In this paper, node's performance is boosted with a stock market application that is compared in terms of speed, reliability and smoothness of access with an application running on Apache web server.

II. RELATED WORK

Beginning from the day of its original release, Node.js has undergone a series of development iterations which made it possible to be used in a wide variety of applications and projects. Right from startups in the SanFrancisco Bay area to Techgiants like Paypal, LinkedIn, Uber, NewYorkTimes, NetFlix, Ebay, etc, Node.js has travelled a long journey. Some of the related works Node.js includes LinkedIn's transition from Ruby on Rails to the faster ever developing technology Node.js [18]. They had to do that because maintaining a long list of data and scrolling it would be hard with a native HTML (HyperText Markup Language) application. By this, they went from running 15 servers with 15 instances (virtual servers) on each physical machine to just 4 instances that would handle double the traffic.

III. MATERIALS AND METHODS

Node.js is a very powerful JavaScript-based framework/platform built on Google Chrome's JavaScript V8 Engine developed by Ryan Dahl in 2009. It is one of the upcoming technologies used to develop fast and scalable applications. Node.js' package ecosystem “npm” is the largest ecosystem of open source libraries in the world [6]. Node.js tries to do asynchronous processing on a single thread to provide more performance and scalability for applications that are supposed to handle too much web traffic [5]. The various features that make Node.js advantageous is that, it is single threaded, asynchronous, event driven, non blocking I/O and highly scalable [12]. Asynchronous and event driven,by the terms briefly explains that node based server never waits for an Application Program Interface (API) to return data; it uses a call and notify mechanism of events that helps the server in getting its previous API’s response [13]. Node uses a single threaded program and that handles multiple server requests. Node.js is open source, completely free, and used by thousands of developers around the world [4]. It is used to develop I/O intensive web applications like video streaming sites, single-page applications, and other web applications [11]. A detailed working of Node.js is shown in figure 1.

The core functionality and the full potential of Node.js can be efficiently utilised only when there is more and more traffic in the websites. This kind of scenarios exists only in certain situations. One such scenario is a Stock market website which faces multiple user requests during bidding.

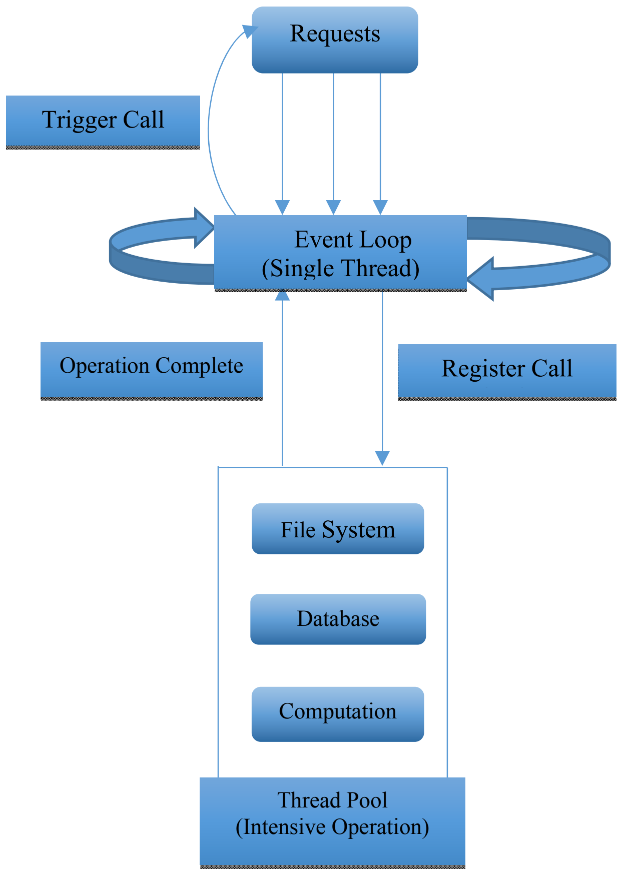

Figure1: Asynchronous working of Node.js server

Many investors who have inadequate knowledge about the share market can gain knowledge about investing wisely in shares through simple and that is a perfect question which also answers the question “Why we have chosen stock market analysis with Node.js?” With multiple users accessing the stock market website simultaneously during auction, bidding, etc; since the traffic flowing through the website is high and the server load may increase which increases the probability of crashing of server [7]. This may be disappointing to potential users of share market, which may also affect the reliability of that stock market website.

This kind of situation can be very well handled by the Node.js backed stock market website which handles users request efficiently in such a way that the probability of server crashing is reduced to a greater extent [9].

IV. NODE.JS IN STOCK MARKET

Node.js plays a major role in the Smart Stock Market Analysis (SSMA) website alias Stockie Smart. Node.js is a JavaScript runtime that uses the v8 engine developed by Google for use in chrome [14]. V8 compiles and executes JavaScript at lightning speeds mainly due to the fact that v8 compiles JavaScript into native machine code [20]. Hence, our website works at a faster rate. The users don't have to wait for a long time and can get the information within a glimpse.

In addition to lightning fast JavaScript execution, the real magic behind Node.js is the event loop. The event loop is a single thread that performs all I/O operations asynchronously. When a Node application needs to perform an I/O operation, it sends an asynchronous task to the event loop, along with a callback function, and then continues to execute the rest of its program [20]. Node allows us to build fast, scalable network applications capable of handling huge number of simultaneous connections with high throughput. The Smart stock market analysis website includes various users. Many users access the website at the same time and the results have to be displayed constantly without any delay. Thus, huge user requests are handled efficiently by Node.js.

The watchlist maintained in the website is constanly updated with real time information of Nifty, Sensex, BSE and NSE charts. These details have to be updated with respect to time and the users have to be updated at a lightning speed. We use Node.js in our website that helps us achieve the better result with high speed.

V. EFFICIENCY AND PERFORMANCE

Since JavaScript is used at the client and server side, the translational overhead is reduced, which reduces the response time to a greater extent [10]. To add on, Node.js uses a single threaded event loop model which performs all the tasks with a single thread that eventually reduces the server load [19]. Thus, efficiency and performance of Node.js powered stock market website is at its peak. The Table 1 demonstrates a simple program in Node.js and PHP/Apache. The advantage of Node.js over PHP/Apache is that it does not require an Apache server to run and it can run on any port of its own (in this case port is 8080).

The Table 1 describes an instance of explaining the efficiency and performance of Node.js versus PHP/ pache. This instance explains the connectivity of Node.js with MongoDB and PHP/Apache with MySQLi. Both the code does connection with their respective databases.

TABLE 1.CODE FOR NODE.JS Vs PHP

Testing code using Node.js server	**Testing code using PHP file**
var express = require('express');	<?php
var path = require('path');	$servername = "localhost";
var MongoClient =	$username = "stockieDatabase";
require('mongodb').MongoClient;	$password = "sh@rEm@rkeT";
var assert = require('assert');	$dbname = "StockDatabase";
var mongoose = require('mongoose');	$connect = new mysqli($servername,
console.log("Database Connectivity:");	$username, $password, $dbname);
var url = 'mongodb://localhost:27017/mydb';	if($connect->connect_error){
MongoClient.connect(url, **function** (err, db) {	die("Connection failed: ". $conn-
if (err) {	>connect_error);
console.log('Unable to connect to the	}
mongoDB server. Error:', err);	$result = $conn->query("SELECT id,
} **else** {	firstname, lastname FROM Users");
	if($result->num_rows >0) {
console.log('Connection established	while($row = $result->fetch_assoc()){
to', url);	echo "id: ". $row["id"]." - Name: ".
var collection =	$row["firstname"]."
db.collection('userDetails');	". $row["lastname"]."
";
collection.find({name:	}
'priya'}).toArray(**function** (err, result) {	}else{
if (err) {	echo "0 results";
console.log(err);	}

} **else if** (result.length) {
console.log('Found:', result);
} **else** {
console.log('No document(s) found with defined "find" criteria!');
}
db.close();
});
}
});
	$conn->close();
	?>

INSTANCE 1 :(Node.js and MongoDB):To connect with MongoDB,an inbuilt package called "mongodb" is used,which simplifies the connection complexity with MongoDB. Then, the port on which the MongoDB runs is specified to imply where to find the database for Node.js

INSTANCE 2(PHP/Apache and MySQLi): At first, the required details such as username, password, the server and the database name are stored and then passed to the MySQLi database. On successful connection and authentication, the details of a list of users are retrieved and displayed.

TABLE 2. COMPARISON BETWEEN NODE.JS & APACHE

Node.js efficiency	**Apache efficiency**
Concurrency Level: 1000 Time taken for tests: 21.162 seconds Complete requests: 100000 Failed requests: 147 (Connect: 0, Receive: 49, Length: 49, Exceptions:49) Total transferred: 8096031 bytes Requests per second: 4725.43 [#/sec] (mean) Time per request: 211.621 [ms] (mean) Transfer rate: 373.61 [Kbytes/sec] received Connection Times (ms) min mean[+/-sd] median max Connect: 0 135 821.9 0 9003 Processing: 1 40 468.5 25 21003 Waiting: 1 30 64.1 25 12505 Total: 2 175 949.1 26 21003 Percentage of the requests served within a certain time (ms) 50% 26 66% 33 75% 36 80% 39 90% 55 95% 94 98% 3030 99% 3090	Concurrency Level: 1000 Time taken for tests: 121.451 seconds Complete requests: 100000 Failed requests: 879 (Connect: 0, Receive: 156, Length: 567, Exceptions: 156) Total transferred: 29338635 bytes Requests per second: 823.38 [#/sec] (mean) Time per request: 1214.510 [ms] (mean) Transfer rate: 235.91 [Kbytes/sec] received Connection Times (ms) min mean[+/-sd] median max Connect: 0 38 321.8 20 9032 Processing: 0 565 5631.0 51 121380 Waiting: 0 262 2324.1 41 52056 Total: 29 603 5641.7 73 121431 Percentage of the requests served within a certain time (ms) 50% 73 66% 78 75% 82 80% 83 90% 89 95% 105 98% 4251 99% 13205

The Table 2 showcases the performance comparison between Node.js and PHP/Apache during the phase of execution. The concurrency level which indcates the concurrency at which Node.js and PHP/Apache operates on. As given, for a 1, 00,000 requests from the clients which also comprise 20,000 concurrent requests, the time taken by Node.js is absolutely very minimal when compared with Apache/PHP. The rate of failure is also considerably reduced in the case of the former. The time taken to complete the requests is about 5 times smaller in Node.js when compared with PHP/Apache. The transfer rate is maximum in Node.js while minimizing the total transferred bytes. Thus, Node.js has higher efficiency and performance when compared with other server side scripting languages [8].
CPU and memory usage is very clearly portrayed in figure 2 and 3.

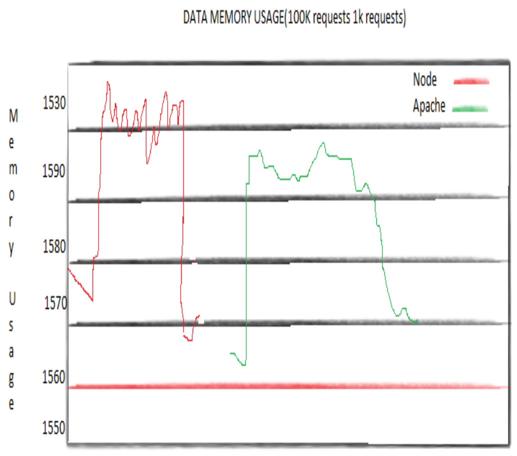

Figure 2: Node.js Vs Apache/PHP CPU Usage

As portrayed in the figure 2, Node.js uses less CPU cycles, reducing the CPU usage by means of a single threaded event loop mechanism. The single threaded event model is possible only because of a concept called "callback", in which the single thread does not wait for an operation to be completed. It fires the respective event and then registering a callback [16]. Once it has completed, the thread will be called and processing is done. Using a single thread definitely reduces the CPU usage, thereby advantageous over multi-threaded server-side scripting languages.

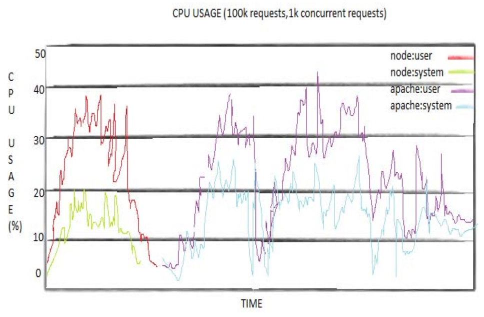

Figure 3: Node.js Vs Apache/PHP Memory Usage

The figure 3 indicates that the memory usage of Node.js is slightly higher than PHP/Apache. This is due to the fact that Node.js uses event loop mechanism and once an event is completed, it has to process it. This is possible only when it maintains a stack of all the functions it has started; so that it can be processed once it is completed, which makes use of memory. But, this is not a major drawback as the real time servers will have humongous amount of memory available [17].

VI. CONCLUSION

Numerous performance tests and efficiency tests were conducted on a smart application in this paper to clearly precribe that Node.js is the most powerful event loop model that performs very efficiently than any other server side scripting languages. Hence, this new technology with no doubt due to its widespread adoption, positively impacts society.

REFERENCES

[1] Santhosh, a Tech talker on upcoming technologies. (2014). Retrieved from http://santhoshthepro.in.
[2] Edmond Woychowsky,"Creating Web Pages with Asynchronous JavaScript and XML", 2007.
[3] Scientific Journal Impact Factor (SJIF): 1.711 International Journal of Modern Trends in Engineering and Research.
[4] Karthik. (2016). Retrieved from www.tutorialspoint.com/node.js/tutorials.
[5] Domecraft. (2016, November 12). Retrieved from https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[6] Austin. (2016, November 29). Retrieved from https://nodejs.org/en.

[7] Node.js: Using JavaScript to Build High-Performance Network Programs , Stefan Tilkov -innoQ, Steve Vinoski – Verivue, IEEE INTERNET COMPUTING,pg:80.

[8] Herb Sutter, A Fundamental Turn Toward Concurrency in Software, March 01, 2005.

[9] Gregor Hohpe, Programming Without a Call Stack – Event-driven Architectures, 2006.

[10] Vivek S. Pai, Peter Druschel, and Willy Zwaenepoel, Flash: An Efficient and Portable Web Server.

[11] Nickolai Zeldovich, Alexander Yip, Frank Dabek, Robert T Morris, David Mazières, Frans Kaashoek, Multiprocessor Support for Event-Driven Programs, 2003.

[12] Paul Krill. (2016, January 08). Retrieved from http://www.infoworld.com.

[13] Dan Shaw. (2016). Retrieved from http://nodesource.com.

[14] Paul Krill. (2014, August 12). Retrieved from http://www.javaworld.com.

[15] Dan Kegel, The C10K problem, 1999.

[16] Gregor Hohpe, Programming Without a Call Stack, 2006.

[17] Flash: an efficient and portable web server, Vivek S. Pai,1999.

[18] Bu Kinoshita (2015, January 10). Retrieved from https://www.quora.com/How-has-the-switch-from-Ruby-on-Rails-to-Node-js-affected-LinkedIn-long-term.

[19] Douglas C. Schmidt, Reactor, 1995.

[20] Gerard Sychay. (2014, September 10). Retrieved from http://blog.modulus.io/top-10-reasons-to-use-node.