A COMPARATIVE ANALYSIS OF NODE.JS AND PHP FRAMEWORK

Mahendra Banaj[1], Mahendra Banaj[2], Rajendra Koras[3], Aastha Tiwari[4]

[1,2,3,4]Student, Computer Science & Engineering,
[1,2,3,4]Govt.Engineering College Bilaspur

Abstract: One of the additional intriguing improvements, as of late, acquiring prominence with regard to the server- side JavaScript space is Node.js. It's a structure for growing high-performance, concurrent programmers that don't depend on the standard multi-threading approach yet utilize offbeat. I/O with an event-driven programming model. The ongoing climate of web applications requests execution and adaptability. A few past methodologies have carried out stringing, occasions, or both. However, expanding traffic requires new answers for working on simultaneous assistance. Node.js is another web system that combines server-side JavaScript and event-driven I/O. Tests will be performed against two tantamount systems that think about help demand times across various centers. The outcomes will show the presentation of JavaScript as a server-side language and the effectiveness of the non-hindering non concurrent model. The paper demonstrates that node.JS is an appropriate system for the advancement of versatile web servers and can be scaled and disseminated across numerous hubs utilizing bunching and replication components.

Keywords: Node.js, JavaScript, performance, scalable, function, asynchronous, threading.

I. INTRODUCTION

In this era of computer and internet, dynamic web is the ubiquitous source for socializing; news updates to name afew amongst many other uncountable usages. The Internet is continually evolving and presents a number challenges to web application designers [1].High traffic demands services to better handle concurrent sessions [2].web based applications are increasing its popularity as they become easier reachable to the clients and does not require additional installations in most cases and are quickly customizable [3].

In this paper, test driven development of a web application has been carried out using relatively new programming language called node.JS (server side JavaScript technology). Node.js which is suitable language for development of scalable network application has been used to develop a prototype. Node.js is an open-source, cross- platform, back-end JavaScript runtime that allows to build scalable network applications. Implement the event model through the entire stack. Developed in 2009 by Ryan Dahl, Node.js (or just Node) is a single-threaded server-side JavaScript environment implemented in C and C++ [4]. Nodes architecture makes it easy to use as an expressive, functional language for server- side programming that's popular among developers [5]. Node utilizes the JavaScript V8 engine, developed by Google [6], a fast and powerful implementation of JavaScript [7] that helps Node achieve top performance. In the following experiments, Node will be compared to Apache to evaluate practical web frameworks for the challenges posed by the current web. JavaScript layer is allowed to access the main thread only whereas C layer is open for multithreading [8]. Multithreaded programming approach is easy and efficient way to make use of multiple cores for concurrency, but it has some pitfalls like deadlocks, accessibility of shared resources and frequent switching of processes [9]. Whereas Event driven model of node.JS provides more scalable solution of switching between tasks making use of event notification functionalities of underlying OS like select(), poll() along with epoll, kqueue, kevent calls [10].

II. LITERATURE SURVEY

A literature survey was done throughout this thesis work. Scientific and research papers[10] were studied, however since node.js is yet a neoteric technology has been delicately providing a services in Development, Digital marketing, Corporate Staffing and Payroll services. There were certain papers available on the topic. Therefore, online documentation and blog posts were also deliberated. In recently published article on IEEE author Tilkov, Stefan summarizes[11] that node is one of the better-known frameworks and environments that support server-side JavaScript development.

Similarly, the author of [Til10] summarizes that node.js can be used to build high performance network programs due to its evented asynchronous style of programming. It bridges the gap between JavaScript being used only on the client side and dependency of other platforms in server side by making a single platform usable in both environments where data precision and consistency are not major concerns (like bank transactions) compared to its scalability and performance.

III. APPROACH & TECHNIQUES USED

Node.js uses an event-driven model where the web server accepts the request, spins it off to be handled, and then goes on to service the next web request. When the original request is completed, it gets back in the processing queue and when it reaches the

front of the queue the results are sent back (or whatever the next action is). This model is highly efficient and scalable because the web server is basically always accepting requests because it's not waiting for any read or writes operations. (This is referred to as "non-blocking I/O" or "event-driven I/O".)

You use your web browser to make a request for "/about.html" on a Node.js web server. The Node server accepts your request and calls a function to retrieve that file from disk. While the Node server is waiting for the file to be retrieved, it services the next web request. When the file is retrieved, there is a callback function that is inserted in the Node server's queue. The Node server executes that function which in this case would render the "/about.html" page and send it back to your web browser. Now, sure, in this case, it may only take microseconds for the server to retrieve the file, but Microseconds matter!

Particularly when you are talking about highly-scalable web servers! This is what makes Node.js different and of such interest right now. Add in the fact that it also uses the very common language of JavaScript, and it is a very easy way for developers to create very fast and very scalable servers.

TABLE I. A simple HTTP file server. Events trigger anonymous functions that execute input or output operations.

```
var sys = require("sys"), http = require("http"), url =
        require("url"), path = require("path"),fs =
                        require("fs");
          http.createServer(function(request, response) {var uri =
url.parse(request.url).pathname; var filename = path.join(process.cwd(), uri);
                    path.exists(filename, function(exists) {
                                        if(exists) {
          fs.readFile(filename, function(err, data) {response.writeHead(200);
                                response.end(data);
                                        });
                                    } else {
                    response.writeHead(404);response.end();
                                }
                            });
                                    }).listen(8080);
                    sys.log("Server running at http://localhost:8080/");
```

IV. PERFORMANCE BENCHMARK

This section focuses on the results obtained after benchmarking the node.js against PHP with some very simple tasks. These benchmarks are obtained as a result of load testing using a highly distributed, concurrent benchmarking testing tool. I have done few very basic performance tests to see how node.js server behaves compared to Apache when serving very simple pages. For comparison I have used node.js 0.1.103 on one side, and Apache 2.2.14 with prefork MPM and PHP 5.2.10 on the other side. I am discussing all the tests one by one following are test:-

Test 1

I have used node.js 0.1.103 on one side, and Apache 2.2.14 with prefork MPM and PHP 5.2.10 on the other, hitting them with Apache Bench 2.3 and total of 100,000 requests with 1,000 concurrent requests during first test:

ab -r -n 100000 -c 1000 http://127.0.0.1:8000/ And then with total of 1,000,000 requests and 20,00 concurrent requests during the second one:

ab -r -n 1000000 -c 20000 http://127.0.0.1:8000

Total Request: 100,000; Concurrency Level: 1,000 Node.js result

TABLE II Node. Js and PHP code.

Node.js Code	**PHP Code**

var sys = require('sys'),
http = require('http');
var body="Hello World"
http.createServer(**function**(req, res) {
res.writeHead(200, {'Content-Type'**:**
'text/html'});
res.end();
}).listen(8000);
sys.puts(„Server running at
http://127.0.0.1.8000/")
	<?php
	echo ' Hello
	World ';

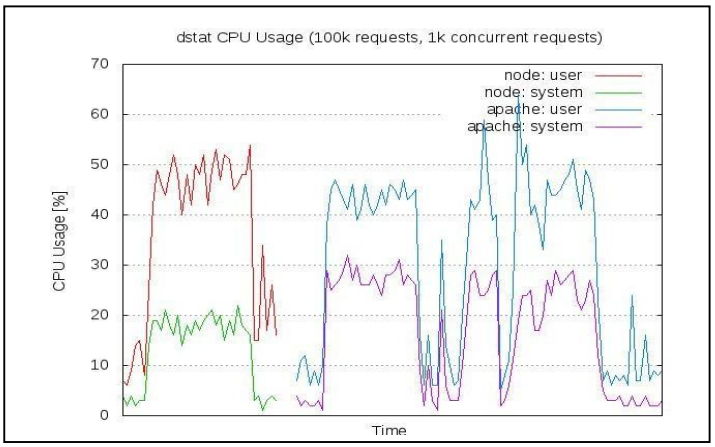

Fig.1. CPU Usage Node.js vs PHP in Apache Bench -100k request,1k concurrent requests.

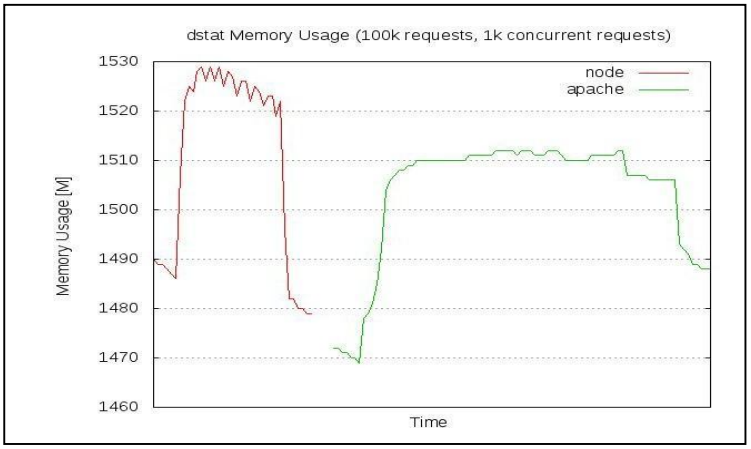

Fig. 2. Memory Usage Node.js vs PHP in Apache Bench -100k request,1k concurrent requests.

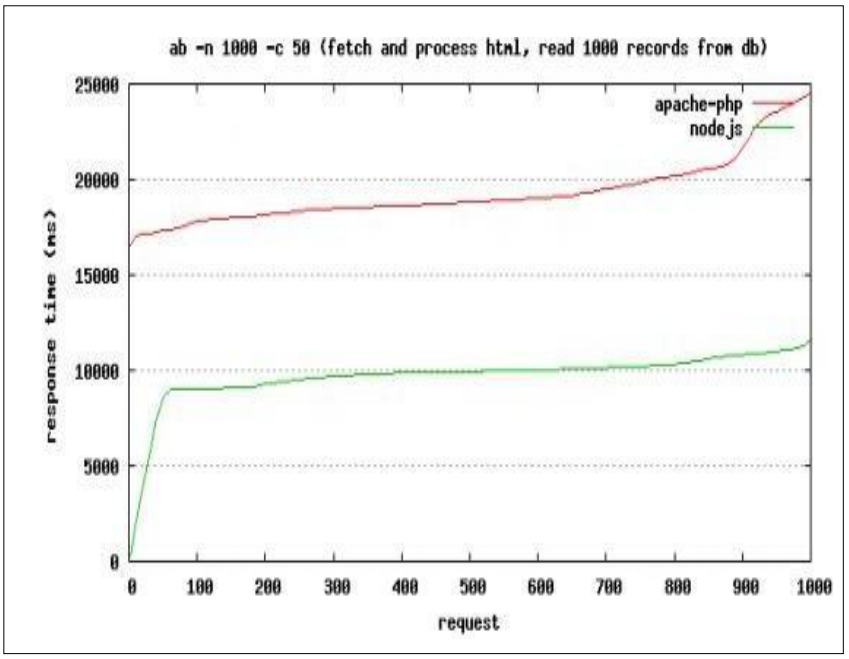

Fig. 3. .ab -r -n 1000 -c 50 http://127.0.0.1:8000/

IV . APPLICATION OF NODE IN THE ENTERPRISE

Voxer is using Node and Riak to build its walkie talkie-style mobile app (and open sourced their in-house Riak client).Yahoo has used Node to develop their Mojito platform and Twitter ,LinkedIn, Netflix , PayPal . Walmart Labs has been using lots of Node in mobile development.

I have also seen all kinds of promising game development done using tools like Node, socket.io, and Redis. I'm excited to see how far this goes and whether or not large gaming companies like Zynga will come to favor Node over Action Script, Erlang, and other languages and frameworks. AppFog has used Node extensively in our console and another crucial parts of our application architecture after various flirtations with Event Machine turned out disastrously for us in testing. We're a new company, but we're making an investment in Node, and we are not alone.

Even Oracle has gotten involved in the Node community. Last fall, they announced that they planned to develop Nashorn, a JVM-based JavaScript engine set to be available in late 2012. They've explicitly talked up the ability to use Nashorn to do crazy things like create Node.jar files and more generally to make the power of Node more readily Java-compatible.

There will always be thousands of developers contributing modules to NPM and providing incremental improvements and additions to Node, but big leaps often require larger projects demanding whole teams of developers. Big shake-ups in the database space, for example, have often followed on the heels of working papers publicized by Google (like this one on its Spanner architecture, which could also have a huge impact outside Google). The Node community could come to function in a similar fashion. As the above graph shows, Node.js performed better than PHP & Apache. The results were encouraging enough for me to make the switch to Node.js (for this particular script).The important result is that Node.JS was around 80% faster than PHP & Apache. I used Apache Benchmark for this test (with Gnu plot to create the image). The results above will certainly vary depending on your system, the resources available, etc.

V. CONCLUSION & FUTURE WORK

Both Node.js and PHP are great server-side technologies for web and mobile app development better. However, it is tough to select the right one in an instant. You should carefully weigh the pros and cons and see if one suits your requirements Node accomplishes its goals of providing highly-scalablservers. It uses an extremely fast JavaScript engine from Google, the V8 engine. It uses an Event-Driven design to keep code minimal and easy-to-read. All of these factors lead to Node's desired goal it's relatively easy to write a massively-scalable solution.

Just as important as understanding what Node is not, it's also important to understand what it is not? Not is not simply a replacement for Apache that will instantly make your PHP web application more scalable.

Popular PHP Frameworks

1. Laravel
2. CodeIgnitor
3. Slim
4. Laminas Project
5.

Popular Node.js Frameworks

1. Express.js
2. Meteor
3. Sails.js
4. Next.js

REFERENCES

1. Labovitz, C., Iekel-Johnson, S., McPherson, D., Oberheide, and Jahanian, F. Internet Inter-Domain Traffic. SIGCOMM ‟10 (2010).
2. L. A. Wald and S. Schwarz. The 1999 Southern California Seismic Network Bulletin. Seismological Research Letters, 71(4), July/August 2000.
3. Matt Welsh, David Culler, and Eric Brewer, ”SEDA: An Architecture for Well-Conditioned, Scalable Internet Services”, ACM Symposium on Operating Systems Principles, 2001.
4. Benchmarking Node.js - basic performance tests against Apache +PHP.
5. John Ousterhout, ”Why Threads are a Bad Idea (for most purposes talk)given at USENIX Annual Conference, September 1995.
6. 2011 Scalable web application using node.JS and CouchDB
7. Tilkov, S., Vinoski, S. Node.js: Using JavaScript to Build High Performance Network Programs. Internet Computing, IEEE, 2010 STRI- EGEL, GRAD O F‟11, PROJECT DRAFT
8. Paruj Ratanaworabhan, Benjamin Livshits, and Benjamin Zorn. JSMeter: Comparing the behavior of JavaScript benchmarks with real web applications. In USENIX Conference on Web ApplicationDevelo-pment (WebApps), June 2010.
9. S. Pai, P. Druschel, and W. Zwaenepoel. Flash: An Efficient and Portable Web Server. In Proceedings of the 1999 Annual Usenix Technical Conference, June 1999.
10. M. Welsh, D. E. Culler, and E. A. Brewer. SEDA: An architecture for well-conditioned, scalable Internet services. In Symposium on Operating Systems Principles, pages 230243, 2001.
11. Sun Microsystems. RPC: Remote Procedure Call Protocol Speciation Version 2. Internet Network Working Group RFC1057,June 1988.