Implementing Secure and Reliable Server By Node.js yet very simple compare to over PHP

Mohammed Jahangir Alom

R00144214

Cork Institute of Technology

Abstract

Node.js (also termed Node) is a platform built on Google Chrome V8 JavaScript runtime engine for easily building fast, scalable, and lightweight applications. V8 and Node are mostly implemented in C and C++ focusing on performance and low memory consumption. In this paper, I will provide an overview of Node by comparing it to a conventional server-side scripting programming language PHP. Initially, I focus on Node's modularity, its in-built package manager labeled Node Package Manager and Node's working architecture. The main feature of Node is its use of non-blocking event-driven I/O with an asynchronous programming model to remain lightweight and efficient in handling concurrency. With Node.js, complex real-time applications can be built that can scale to millions of client connections. I also discuss factors supporting choosing Node over PHP and why developers should use it. I describe some of the security holes in Node with solutions to handle them. I conclude by highlighting some of the limitations of Node and I discuss the current developments in process to remediate Node's deficiencies.

Table of Contents

Chapter **Page**

Chapter 1: Introduction

Node.js (Node) [1] is a cross platform runtime environment originally developed in ***2009 by Ryan Dahl*** for developing server-side applications. It can be regarded as server-side JavaScript. It was created to address the issues platforms can have with the performance in network communication time dedicating excessive time processing web requests and responses. Node has become very popular as it makes creating high performance, real-time web applications easy. Node allows JavaScript to be used end to end, both on the server and on the client. Node is event-based rather than thread-based. Node uses an event loop within a single thread instead of multiple threads, and can scale to millions of concurrent connections. In Node, a single thread can accomplish a high concurrency. Every I/O operation in Node is carried out in an asynchronous fashion, meaning that the server can continue to process incoming requests while the I/O operation is taking place [2].

Objectives of our Study

Node has not just strengthened server-side JavaScript, but also has been competitive with other popular server-side scripting languages with respect to performance and scalability. In this paper, I will describe the advantageous features of ***Node over PHP***. The underlying features of Node: single-threaded, event-driven I/0, and asynchronous programming are discussed with sufficient examples to give better insight into the working architecture of Node that led to Node's success. For further development and enhancement, I also highlight the existing limitations and deficiencies of Node.

Simple HTTP Server with Node.js

One of the common uses of Node is to build servers. Node can be used to create different types of servers [3]. A simple HTTP (Hyper Text Transfer Protocol) web server that responds, "Hello Node!" to every request it receives can be created with very few lines of code.

Node.js Modules

Modules are plugins, add-ons, and extensions for Node to help with the development process. The Node module exposes a public API (Application Programming Interface) that one can use after the module is imported into the current script. Node modules can be categorized as core modules, third party modules, and local modules.

NPM: Node Package Manager

NPM is a built-in tool that is included by default with every installation of Node. NPM helps in easily managing modules in Node projects by downloading packages, resolving dependencies, running tests, and installing command line utilities [4].

How Node.js Works?

The main distinctive features of the Node architecture are the usage of non-blocking, event driven, asynchronous I/O calls that operate in a single thread. Node is lightweight, efficient, and different. It can support tens of thousands of concurrent connections because of its unique features. Even with limited memory and a single thread, Node can achieve high concurrency rate without having to perform context switching between threads [5].

Chapter 2: PHP vs Node.js

What is PHP?

PHP (Hypertext Pre-processor) is a general-purpose scripting language that quickly became the de facto server-side language of choice for web developers after its initial release in 1995.

PHP vs Node.js Major Difference

While both are server-side technologies, there are a few differences between a PHP back-end and a JavaScript-powered back-end via Node.js. [10]

- **Runtime environments**. While both JavaScript and PHP can be embedded directly into HTML, they both need an interpreter to run. PHP has long been readily straightforward to install and use on the server-side, and is powered by the Zend engine. Node.js is a runtime environment for JavaScript on the server side, powered by Google's V8 JavaScript engine.
- **PHP is simpler**. PHP is conceptually much simpler to use than Node.js. When setting up a server, all you need is a ". php" file with some code wrapped between the tags, enter the URL into your browser, and you're done. Behind the scenes, a web server like MySQL with PHP installed will be able to interpret the file and display your web page in your

browser. Setting up a Node.js server, while not difficult, usually requires more lines of code, and a basic understanding of how closures and call back functions work.

- **Concurrency**. PHP, like most server-side languages, uses multi-threaded, blocking I/O to carry out multiple tasks in parallel. JavaScript is unique in that it uses a few tricks (event loop + Node clustering) to achieve an event-driven, non-blocking I/O execution model that uses a single main thread of execution.
- **JSON**. JSON "JavaScript Object Notation" is a lightweight data format that gives Node.js an edge when dealing with JSON. While PHP can work with JSON, it's more situational.

Performance

PHP is no slouch and there are projects and options which make it faster. Even the most demanding PHP developer rarely worries about speed but **Node.js** performance is generally better. Of course, performance is largely a consequence of the experience and care taken by the development team but Node.js has several advantages… [11]

Fewer Dependencies

All requests to a PHP application must be routed via a web server which starts the PHP interpreter which runs the code. Node.js doesn't need so many dependencies and, while you'll almost certainly use a server framework such as Express, it's lightweight and forms part of your application.

A smaller, Fast Interpreter

Node.js is smaller and nimbler than the PHP interpreter. It's less encumbered by legacy language support and Google has made a huge investment in V8 performance.

Application are Permanently On

PHP follows the typical client-server model. Every page request initiates your application; you load configuration parameters, connect to a database, fetch information and render HTML. A Node.js app runs permanently, and it need only initialize once.

An Event-driven, Non-Blocking I/O

PHP and most other server-side languages use an obvious blocking execution model. When you issue a command such as fetching information from a database, that command will complete execution before progressing to the next statement. Node.js doesn't (normally) wait. Instead, you provide a call back function which is executed once the action is complete.

Chapter 3: Why Node.js?

Node has been popular among developers and with its success has attracted many high-tech companies who have adopted Node replacing their existing technologies. There are many reasons for Node's popularity and why one should use Node for their application development.

High Performance Web-Servers.

Popularity of JavaScript.

One Language Multiple Functionality.

Simple Development Environment.

Good Reputation.

Chapter 4: Limitations of Node.js

We know Node's benefits in terms of performance and scalability. In this paper, I showed Node as the superior and neat platform for application development. Node's true potential in handling concurrency with limited resources. However, some of Node's benefits have become the reasons for its weaknesses. Part of the utility of Node is that there are limited weaknesses in the typical sense of the word [5]. Having been developed only in 2009, Node is currently still in the development life cycle.

Poor handling of heavy server-side computation

Node struggles in handling of very high computationally intensive tasks, because whenever it executes long-running task, Node will queue all remaining incoming requests, because it implements single-threaded architecture with an event loop [9].

Server-side application with relational database

Node integrates quite well with NoSQL databases which are non-relational in nature. However, relational database tools for Node are still in their early stages and are rather immature [5].

Complexity with callback function

Asynchronous I/O combined with a callback function is the salient feature of Node that allows it to handle multiple concurrencies. A callback function in Node is an anonymous function that is

usually nested together with some other factory methods. When the logic of the code becomes complex, Node might suffer from the problem termed callback hell [7].

Ecosystem in Development

Node's ecosystem is currently in its development stage. Although, the number of public libraries or modules in Node online repository is increasing, and the community and support is expanding, there is still a long way to go, before its ecosystem is sufficiently established.

Adherence to JavaScript

Node is benefited with its base on JavaScript and how Node suffers in terms of the security aspects that it inherited from JavaScript. Additionally, JavaScript has not been developed for use on servers until recently. JavaScript is still very new in the server-side environment

Chapter 8: Limitations and recommendations for further Study

Node is a very new application platform. Node's features and benefits have attracted many developers worldwide. As a result, the scope of Node has increased significantly. Due to the wide scope of Node issues, ***I covered a limited number of aspects of Node in this paper that includes, what Node is, how it is installed, how it works, what a Node Module and NPM is, how Node is different from JavaScript and PHP, what security holes it has, and what its limitations are.***

There are some limitations and shortcomings in our study. First, we don't provide detailed information on popular in-built Node modules available in NPM repository. There are many popular Node modules that are used in Node application development. For instance, modular frameworks such as Express.js and Jade are not discussed in this paper A further investigation of **Node application development using MEAN would provide a better understanding of the application development process in Node.**

Chapter 9: Conclusion

In this paper I showed that Node has transformed the usability of JavaScript, making Node a complete programming language. From browsers to server-side scripting outside of browsers, Node has made possible the availability of a runtime environment, a library full of free useful modules that can be imported by using an in-built tool named NPM. Node uses nonblocking, event-driven I/O asynchronous programming to be lightweight and be efficient. I showed that setting up a Node environment is simple, and Node is available to all major operating systems.

Node's performance with respect to another server-side scripting language PHP is quite remarkable. Apart from its benefits, Node does have some security holes. If Node applications are not programmed with good error handling and input validation, then those applications can be vulnerable to attacks. Therefore, it becomes the responsibility of Node developers to make Node applications secure. With all its advantages, Node plays a critical role in the technology stack of many high-profile companies who depend on its unique benefits. One can exploit the benefits of Node while experiencing fast and scalable applications development with the power of Node.

References

[1] Kurniawan, A. (2014). Node.js Succinctly. Synfusion Inc.

[2] Govett, D. (2010, March). Learning Server-Side JavaScript with Node.js. Retrieved from Envato Tuts+: http://www.webcitation.org/6ePoNkZwD

[3] Mike Cantelon, T. H. (2013). Node.js in Action. Manning Publications.

[4] Rauch, G. (2012). Smashing Node.JS JavaScript Everywhere. John Wiley & Sons Inc.

[5] Joseph Delaney, C. G. (n.d.). Node.js at a glance. Whale Path Inc.

[6] Kamali, B. (2015, July). How to Create a Node.js Cluster for Speeding Up Your Apps. Retrieved from Sitepoint: http://www.webcitation.org/6ew832ZZA

[7] Callback Hell (n.d.). Retrieved from Callback Hell: http://www.webcitation.org/6ew8DXVK0

[8] McIlvenna, S. (January, 2014). Retrieved from Evolving Software: http://www.webcitation.org/6ew8PX8Dh

[9] Posa, R. (March, 2015). Retrieved from JournalDev: http://www.webcitation.org/6ew9SMBVa

[10] PHP vs Node.js: Major Differences
https://www.upwork.com/hiring/development/php-vs-node-js/

[11] Performance
https://www.sitepoint.com/sitepoint-smackdown-php-vs-node-js/