International Journal of Research in Advent Technology (E-ISSN: 2321-9637) Special Issue
National Conference "NCPCI-2016", 19 March 2016
Available online at www.ijrat.org

Using Node.Js to Build High Speed and Scalable Backend Database Server

S. L. Bangare[1], S. Gupta[2], M. Dalal[3], A. Inamdar[4]
Department of Information Technology[1, 2, 3, 4],
sunil.bangare@gmail.com[1] , *guptashubham293@gmail.com*[2]

Abstract- With the increasing number of Application and Website development, it is quite mandatory for the developers to use a Database server for efficient storage and transfer of data. There are various 3rd party Cloud Database providers (Google, AmazonAWS, Mongolab) for developers to start building their application without worrying about the backend services (server-side scripting). This method is not very efficient and affordable for small-scale developers. Also there is always a concern for the privacy of data as the data is not stored in our machines. In order to overcome this problem for the developers, there needs to be a way for developer to build their application along with hosting their own local machine as a database server. Also making them free from tedious server-side coding. Node.Js is a server-side platform mainly used for real-time application because of it 'event-driven architecture' and 'non-blocking I/O'. Node.js is found to be 10 times faster in I/O operations.

Index Terms- Website development, database server, Node.js, server-side platforms, non-blocking I/O

1. INTRODUCTION

There is rapid increase in the development of Applications and Websites with the rapid development of web today. The developers always require a database server for storage of their application data. These data can be very large and often requires real-time access anytime from anywhere. At present, building an app is very convenient if there is backend service available which can be interfaced with their apps across all platforms. But the main struggle for the developers is in picking/selecting/building a flexible, high-concurrency backend for their app. Many factors are responsible for building the backend server like scripting language, client handling mechanism, data transfer process etc. Mobile phones are nowadays far more than merely devices to communicate with. Especially, Smartphone's are products that help to make our work and everyday life easier [13].

At present, a developer often use following couple of methods to connect its application with backend database:

- **Third Party Cloud Server**: This services often referred as PaaS (Platform as a Service) provides application developers with various data storage plans as per their need and support different platforms.
- **Implementing Backend server**: A developer can sometime deploy its data by building his own server using different scripting languages like PHP, Ruby on Rails etc.

2. LITERATURE SURVEY

The most important thing with a web server is its ability to handle multiple users efficiently. This has a lot to do with the programming language used to write its script. Hence performance of server-side scripting languages like PHP, Python were taken into consideration with comparison with Node.js [8]. P. S. Bangare et al has proposed the novel secure encryption mechanism which is a combination of chaotic logistic mapping and RC4 stream cipher [10].

Node.js is an excellent tool if you want some kind of live interaction, real-time results. It is capable of very quickly delivering data to/from a web server. Traditionally, there has always been a big problem with computers where the CPU can only do one thing at a time. It was solved long ago with multi-threading, allowing us to have multiple 'threads' on a single CPU. It switches between them all the time, and while it's pretty fast, the switching has a ton of overhead. To avoid this overhead node.js solves this problem by running in a single, event-driven thread. Rather than have a new thread get created on each request, there is one thread for every single request. When a new one comes in, it fires an event that runs some code. When you make a call to a database, for example, rather than block until it's returned, you just run a call-back function after the call is complete. Any number of call-backs can respond to any event, but only one call-back function will ever be executing at any time. Everything else your program might do—like waiting for data from a file or an incoming HTTP request—is handled by Node, in parallel, behind the scenes. Your application code will never be executed at the same time as the most important thing with a web server is its ability to handle multiple users efficiently. This has a lot to do with the programming language used to write its script. Hence performance of server-side scripting languages like PHP, Python were taken into consideration with comparison with Node.js [8].

Node.js is an excellent tool if you want some kind of live interaction, real-time results. It is capable of very quickly delivering data to/from a web server.
Traditionally, there has always been a big problem with computers where the CPU can only do one thing at a time. It was solved long ago with multi-threading, allowing us to have multiple 'threads' on a single CPU. It switches between them all the time, and while it's pretty fast, the switching has a ton of overhead. To avoid this overhead node.js solves this problem by running in a single, event-driven thread. Rather than have a new thread get created on each request, there is

one thread for every single request. When a new one comes in, it fires an event that runs some code.

When you make a call to a database, for example, rather than block until it's returned, you just run a call-back function after the call is complete. Any number of call-backs can respond to any event, but only one call-back function will ever be executing at any time. Everything else your program might do—like waiting for data from a file or an incoming HTTP request—is handled by Node, in parallel, behind the scenes. Your application code will never be executed at the same time as anything else. It will always have the full attention of Node's JavaScript engine while it's running.

Heavy I/O applications benefit well from this, whereas CPU intensive applications will not. However, a backend database server is more I/O intensive, so that's generally an effective trade-off. Many middleware tasks are I/O-bound, just like client-side scripting and databases. These server-side programs often have to wait for things like a database result, feedback from a third-party web service, or incoming connection requests. Node.js is designed for exactly these kinds of applications.

When compared on multi users the performance of Node.js is better. The performance of Node.js is two times larger than PHP and six to seven times larger than Python. Hence making it highly concurrent and real-time [8].

The choice of database is very important when considering a web server. It should go with the server framework selected. Node.js is written in JavaScript environment and since MongoDB also works on JavaScript it can perfectly synchronize and work together. MongoDB is a document database. The concept of rows still exists but columns are removed from the picture. Rather than a column defining what should be in the row, each row is a document, and this document both defines and holds the data itself. MongoDb stores documents as BSON, which is binary JSON. In short, JSON is a JavaScript way of holding data, hence why MongoDB fits so well into our JavaScript centric framework.

3. RELATED WORK

There has been lots of implementation of Node.Js based program lately. The systems have been implemented by integrating Node with other tools and making web server for application [7].

Stefan Tilkov and Steve Vinoski studied how JavaScript based Node.js can be used to build high performance programs. They studied and compared the performance of a multi-thread over a single thread process [7].

An open source community linnovate developed a way of using Node.Js with MongoDB, Express and Angular.Js to create a full stack called MEAN stack. This shows how can it is possible to integrate this all technologies to make server for website and application.

4. PROPOSED SYSTEM

This paper aims to implement a framework for application developers which include software bundles like Node.js and MongoDB and create APIs to connect their application with their database easily without worrying of the server-side coding. Also this APIs can be used to create drivers for various platforms like Android, .Net, IOS etc.

There needs to be a more convenient and simple way for an application developer to connect their applications and website with a database server. This backend server should be such that it could handle multiuser request and should be of high concurrency. Also it should take into consideration the privacy and security of data.

Node.js is currently a new and trending technology in JavaScript. It is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Using Node's core functionality and integrating it with one of the fastest and scalable database i.e. MongoDB, we intent to implement a framework for a backend server for our database which can help easily connect our frontend with the MongoDB. This framework will have the following advantage over the above mentioned ways:

1. It will make a developer free from writing the server side script and hence making him concentrate only on the frontend to the application.
2. This framework uses Node.js with is than traditional scripting language like PHP, Ruby on Rails etc.
3. All the layers on this framework use only one language throughout the stack i.e. JavaScript, hence making it well fit and in sync.
4. As the database server is deployed on the developers system, he has full control over the privacy and security of it.

Node.js can be used with many databases such as Mysql, MongoDB etc but due to various advantages of MongoDB over other databases, we are using Mongodb in our project. MongoDB is an increasingly popular document-based, high-performance NoSQL database. In MongoDB, data is stored as a document as a set of key-value pairs. It can define multiple databases in MongoDB where each database can have many collections, and those collections are simply a set of documents that consist of data stored as a key-value pair. The data structure that defines a MongoDB document is called BSON (Binary JSON). BSON is binary representation of JSON and also supports data types such as Date, which is not supported in plain JSON format. MongoDB internally converts JSON to BSON and vice versa for performance benefits, although the user can save, query and retrieve as JSON.

MongoDB offers very good performance for situations containing very high write loads, but where data integrity isn't a pressing concern; a good example are the comments sections of large, busy websites like Craigslist or The New York Times – by the way, these aren't theoretical by the way: both of these use MongoDB [4]. Our main aim is to implement a single framework integrating node.js and its modules such as mongojs,express.js .

The deployment of Mongodb as a remote database is taken into account after evaluation of its performance. This paper aims to implement framework using restfull API's over conventional API's such as SOAP or CORBA. REST is an architectural style that uses simple HTTP calls for inter-machine communication instead of more complex options like CORBA, COM+, RPC, or even SOAP. Using REST means your calls will be message-based and reliant on the HTTP standard to describe these messages [5]. Using the HTTP protocol means REST is a simple request/response mechanism. Each request returns a subsequent response [5]. P. A. Kotwal et al have shown in their work to use App Server and its role in their research work [14].

5. DETAILS

Node.js is a software platform that allows you to create your own webserver and build web applications on top of it. Node.js is not itself a webserver, nor is it a language. It contains a built-in HTTP server library, meaning that you don't need to run a separate web server program such as Apache or IIS. This ultimately gives you greater control over how your web server works, but does increase the complexity of getting it up and running particularly in a live environment. With PHP for example, you can easily find a shared-server webhost running Apache, send some files up over FTP and all being well your site is running. This works because the webhost has already configured Apache for you and others to use. With Node.js this is not the case, as you configure the Node.js server when you create the application.

Express is used to manage user sessions, with optional support from MongoDB. User authentication will typically use Node.js, Express, MongoDB and Mongoose, but there are a number of third-party modules that you can plug into your application so that you don't have to do all of the hard work. A server on local port can be enabled using following code:

```
var express = require('express');
app.listen(3000); //parameter is port number
console.log("Server running on port 3000");
//This will start the server on the port no. 300 of local host
```

Express abstracts this difficulty away by setting up a web server to listen to incoming requests and return relevant responses. On top of this it also defines a directory structure. One of these folders is setup to serve static files in a non-blocking way the last thing you want is for your application to have to wait when somebody else requests a CSS file! You could configure this yourself directly in Node.js, but Express does it for you.

An asynchronous programming model of events and call backs is well suited for a server which has to wait for a lot of things, such as incoming requests and inter-process communications with other services (like MongoDB).

MongoDB is a low-overhead database where all entities are free-form ***BSON*** — "binary JSON" — documents. This lets you work with heterogeneous data and makes it easy to handle a wide variety of data formats. Since BSON is compatible with JSON, building a REST API is simple — the server code can pass requests to the database driver without a lot of intermediate processing.

Node and MongoDB are inherently scalable and synchronize easily across multiple machines in a distributed model; this combination is a good choice for applications that don't have an evenly distributed load. This asynchronous nature on node coupled couple with its compatibility with MongoDb makes it vital for our server as it's a real time functioning framework. We further have to write all possible operation that is supposed to be in a database server. All CRUD (create, Read, Update, Delete) to and from db.

We build an API for all CRUD operation using callback functionality of Node.js. The APIs when triggered lead to executing its operation in the database and ending with the resulting callback function.

the bellow example of an Insert API:

```
app.post('/register', function (req, res)
{
console.log("Request for Insertion received with this data");
 console.log(req.body);
  db.users.insert(req.body,function(err,docs){
  console.log("Data after insert: ");
   console.log(docs);
   res.json(docs);
  });
});
```

These APIs work on various HTTP protocols. In the above code for example we get a HTTP request from a client with JSON data to be inserted. The code executes it by using the mongodb driver functionality and finaly responds with a HTTP response variable [6]. This APIs is then used to build driver for Android Platform. API testing is also possible, related work has been mentioned by S. L. Bangare et al [11] [12].

6. CONCLUSION

This paper presents the framework using node.js to build highly scalable and high-speed backend database server for web developers as well as application developers. It also demonstrates the use of NoSQL database such as Mongodb in proposed project work over other traditional database such as MySQL.

7. ACKNOWLEDGEMENT

We are thankful to Prof. A. N. Adapanawar, HOD-IT, Dr. K. P. Patil, Vice Principal & Dr. V. M. Wadhai, Principal, Sinhgad Academy of Engineering, Pune for providing the support for this research work. Also we are thankful to Dr. S. T. Patil, Professor, VIT, Pune & Dr. G. Pradeepini, Professor, K. L. University, A.P. for their valuable guidance.

REFERENCES

[1] http://node.js.org/
[2] http://php.net/
[3] http://python.org/
[4] https://www.upguard.com/articles/mysql-vs-mongodb
[5] http://blog.pluralsight.com/representational-state-transfer-tips
[6] https://github.com/mongodb/node-mongodb-native
[7] S. Tilkov, S. Vinoski, "Node.js: Using JavaScript To Built High-performance Network Programs", Internet Computing, IEEE, Page(s): 80-83 Volume: 14, Issue: 6, 01 November 2010.
[8] Kai Lei, Yining Ma, Zhi Tan, "Performance Comparison and Evaluation of Web Development Technologies in PHP, Python and Node,js", Computational Science and Engineering (CSE), IEEE , Page(s): 661-668 ,19 December 2014.
[9] Jim R. Wilson, "Node.js the Right Way:Practical Server Side Javascript that Scales", The Pragmetic express, ISBN-13: 978-1937785734.
[10] P. S. Bangare, S. L. Bangare, "Implementing Separable Reversible Data Hiding In Encrypted Image Using CLM RC4 Method", International Journal of Engineering Research and Technology (IJERT), Volume 3, Issue 04, 2014/4/26.
[11] S. L. Banagre, P. S. Bangare, "Automated API Testing Approach", International Journal of Engineering Science and Technology, Volume 4, Issue 2, Page(s) 673-676, ISSN: 0975-5462, 2012.
[12] S. L. Bangare, P. S. Bangare, "Automated Testing in Development Phase", International Journal of Engineering Science and Technology, Volume 4, Issue 2, Page(s) 677-680, ISSN: 0975-5462, 2012
[13] P. S. Bangare, S. L. Bangare, "The Campus Navigator: An Android Mobile Application", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 3, Issue 3, March 2014, Page(s) 5715-5717, ISSN (Online) : 2278.
[14] P. A. Kotwal, S. L. Bangare, "A Location Tracer With Social Networking Services", International Journal of Engineering and Technology (IJET), Vol 4 No 1 Feb-Mar 2012, Page(s) 19-23, ISSN: 0975-4024.