International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org
Volume 6, Issue 3, March 2017 ISSN 2319 - 4847

MEAN-vs-JAX RS for RESTful Web Services: Performance and Efficiency Analysis

Shakir Jameel[1], Stuti Khanna[1], Sakshi Rangappa[1], Prof. Sheetal Patil[2], Prof. R.G Khalkar[2]

1 Bharati Vidyapeeth Deemed University Pune, India Department of Computer Engineering

1 Bharati Vidyapeeth Deemed University Pune, India Department of Computer Engineering

1 Bharati Vidyapeeth Deemed University Pune, India Department of Computer Engineering

2 Bharati Vidyapeeth Deemed University Pune, India Department of Computer Engineering

2Bharati Vidyapeeth Deemed University Pune, India Department of Computer Engineering

ABSTRACT

This paper examines the components of the MEAN Stack, and analyzes the benefits and efficiency of MEAN in comparison to JAX RS for RESTful applications. We compare the flow of information between these two application frameworks by creating a simple web application (Messenger Application) using MEAN and JAX RS. We also make performance reviews of servers on client-requests due to the difference in the type of data transferred between client and server.

Keywords- REST, MEAN, web programming, MongoDb, Express.js, Angular.js, Node.js

1. INTRODUCTION

Representational State Transfer (known as REST or RESTful) model for web-services application uses native HTTP operations like POST, GET, PUT and DELETE to map it to four fundamental database operations of CREATE, READ, UPDATE and DELETE (CRUD operations). All web-service applications use database to store user data and their activity logs to deliver services. Therefore, there is a need to make sure service response is quick, efficient, and accurate keeping a check on hardware requirements and cost of infrastructure. That is why there is a dire need to choose such frameworks that are performance as well as cost efficient. Regular JAX RSs like Spring and Hibernate shift the entire load of the application ranging from the client-view to client-response, to the server granting the request. However, frameworks like MEAN and MERN divide the load between client and server to optimize performance and scalability.

2. THE MEAN STACK

Software and Web Developers have been using Java Web Apps and stacks like LAMP (comprising Linux Operating System, the Apache Web Server, MySQL for database management and PHP for rendering view and control) to create dynamic web applications and services for a long time. However, a new development stack known as MEAN or MEAN stack has emerged a few years ago.

MEAN is a collection of four tools that offer both client-side and server-side interactive applications. Mongodb (latest version Mongodb 3.2) provides the object database, Express.js (latest version Express 4.15.2) provides the routing framework, Angular 2 (latest version Angular 2 cli (command line interface)) is used for rendering the web-application and Node.js (latest version v7.7.3) is a web server component and a JavaScript engine that is extremely useful in injecting dependencies in Angularjs and Express [1].

All the constituents of this stack are managed by JavaScript which was initially a client side web programming language. The JavaScript language is lightweight and easy to write and the fact that it is used in all the components of the MEAN Stack makes it extremely easy to understand the data flow through the components from client to server.

2.1. Node.js

Node.js is an event-driven asynchronous JavaScript runtime which was primarily designed to make scalable network applications and APIs. Node Js was built on Chrome V8 Engine which is an open-source JavaScript engine developed on C++ and V8 implements the ECMAScript. Node.js uses an asynchronous event-driven [2], non-blocking I/O model that makes it lightweight, efficient and high-performance [3].

Although the Node is mentioned as the last tool of the MEAN stack, it is the most important one. Node.js' package ecosystem, npm (node package manager), is the largest ecosystem of open source libraries in the world which provides all the modules and devDependencies for Angular CLI along with all the Dependencies for Express. It acts as the dependency injector of the stack.

2.2. Express.js

Express.js is a minimal and flexible Node.js web application framework which provides a robust set of fundamental web application features without obscuring Node.js features. Express makes it easy to create APIs because of access to middleware. Middleware functions are functions which have access to the request and response objects along with a next function, which when invoked, executes the middleware succeeding the current middleware.

Express.js provides a similar functionality to that of what Spring offers to Java applications, that is, an easy to use web framework.

2.3. Angular.js

Angular is an open source web application framework, to create single page web application. It is a Javascript framework which can be added to an HTML page or any web page, with a <script> tag. Angularjs uses ng-directives to extend HTML. The Angularjs directives are simply HTML attributes to which a prefix is attached- ng. Angular forms the front end of the MEAN stack. It implements the MVC pattern to differentiate between the client side and server side of an application and thus reduce the load on the server [4]. The initial launch version of Angular was Angular 1.x which used the concept of creating modules and registering them on HTML for responsiveness. The latest version of Angular which is Angular 2, uses the concept of creating Components that help in code reuse in any deployment target.

2.4. Mongodb

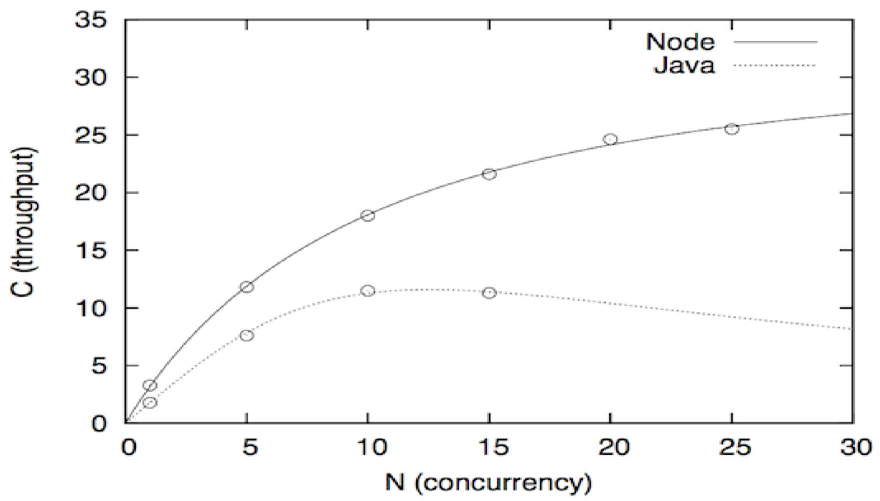
Mongodb is an open-source, document-based database. It is also called as a noSQL database program. It makes use of JSON like documents along with schemas. Because of the tremendous increase in the volume and variety of data over the past few years, there has been a need for a non-relational database. Mongodb fulfills this criteria, as it stores its data in a document-oriented, file structure. Each database can have multiple collections and data in those collections is stored in the form of array of JSON objects.

3. EFFICIENCY ANALYSIS OF MEAN AND JAVA WEB-APPLICATION FRAMEWORKS

3.1. On server-side

In a regular Java web application and Apache web server is used to respond to the requests made by the client. Without using any framework this is implemented creating the http operation methods in the servlet and transferring the request to the service layer and repository layer consecutively. The transfer of request from the URL must be mapped in the web.xml file to make the browser understand the data flow through the URLs. To optimize these operations we can use Spring Framework that automates these operations and we can use them in our application by using Annotations. Even the server doesn't have to be installed using the Spring Framework which automatically does that using Maven. Maven is the dependency provider for Spring. All the dependencies are declared in an xml file by the name pom.xml which enables the Maven to understand what dependencies to install for running the application.

However, in a MEAN Stack application Express.js provides the server and routing framework. Note that these features can be achieved by node alone but Express makes the basic features required for a web-app more easy to use. Like the Maven in a JAX RS, Node provides us with npm (node package manager) that can install the dependencies for the application and like Spring has pom.xml file that stores the declarations of dependencies, Node refers to the package.json file for understanding what dependencies to install [5].Large commercial organizations have adopted the shift of moving products from Java to Node Framework. One of the most popular organizations to do it is PayPal along with Walmart and many more[6].



(**Fig. 1** Modeling with the Universal Scalability Law. Analysis of PayPal's Node vs Java Benchmarks).

After this migration PayPal openly released the performance analysis of their product before and after migration to Node.js. Fig. 1 depicts the quantifiable scalability of computer systems [7]. It is clearly evident that Node outperforms Java. and it is also worth noting that Java is bottlenecked less on serialization, and Node.js is bottlenecked less on coherency delays. This is exactly what one should expect from their architectures (multi-threaded versus single-threaded with event loop) and their blog post ("using a single core for the node.js application compared to five cores in Java").

3.2. On client-side

In regular web-applications that do not store user information and activity log the view pages can be made more interactive using plain old JavaScript. However, most interactive web-apps perform CRUD operations which requires fetching data from the database and rendering it in the view. The most convenient way of rendering data from the server on the client is by using JSP (Java Server Pages) using scriplets. But the problem with JSPs is that they are eventually converted into Servlets which are processed at the server-side.

AngularJs uses a rather uses a more convenient way of handling and rendering data on the view. Angular creates an MVC structure for the front-end that handles all the data received from the server. Therefore by using ng directives we can easily manipulate the DOM based on the result received from the server. This is highly efficient method to distribute load of rendering view from the server to the client's device. Therefore using MEAN stack makes us figuratively divide the entire application into into a client side application and server side application.

In reality most of the responsive web-app front-end consists of jQuery instead of pure JavaScript because it is fast, small and feature rich JavaScript Library. Comparing Angular with a tool like jQuery we can understand better the feature differences these two offer. jQuery helps us in abstracting the DOM and provides us features like Unit Test Runner, Deferred Promises, Cross-Module Communication, Animation Support and finally AJAX and JSON support. Angular, however, not only provides all the features that jQuery provides but also provides us with RESTful APIs, Integration Test Runner, MVC Pattern Support, Templating, Two-Way Data Binding, Dependency Management, Deep-Link Routing, Form Validation and Localization.

Even after so many feature addition the difference between the size of similar files generated by both Angular and jQuery is hardly even 5KBs. Adding to that the concept of selectors in Angular gives us the opportunity to define our own custom tags with their functionality defined in @component. We can also import various modules from different sources to maximize code reusability.

Due to this efficient mechanism of code-reusability Angular apps are extremely scalable and memory efficient. The new version of Angular 2 cli (Command Line Interface) makes the creation of components even easier. Another advantage that the Angular update provides, i.e, Angular 2 cli is that it is based on TypeScript. TypeScript is the superset of JavaScript. JavaScript was initially made for complementing the view of web-applications, however, since JavaScript is used in Full Stack Development it can not offer all features of a full-fledged, statically line by line developed object oriented programming language like Java. Therefore TypeScript was developed which uses the latest ECMAScript to offer the main pillars of object oriented language. The only disadvantage of this update from Angular 1.x to Angular 2 is that Angular is no longer backward compatible. That is, code written in Angular 1.x cannot be executed using Angular 2 cli.

3.3. Databases

A given mongo database is broken up into a series of BSON files on disk. BSON is a format built specifically for Mongodb. Mongodb stores the data on the disk as BSON in our data path directory. A conventional relational database stores data in the form of tables but Mongodb stores data in documents in JSON format [8].

Mongodb is not magically faster. If we store the same data, organised in basically the same fashion, and access it exactly the same way, then we really shouldn't expect our results to be wildly different.

For example, consider a design that persisted a lot of information about a complicated entity in a normalised fashion. This could easily use dozens of tables in MySQL to store the data in normal form, with many indexes needed to ensure relational integrity between tables [9].

Now consider the same design with a document store. If all of those related tables are subordinate to the main table (and they often are), then we might be able to model the data such that the entire entity is stored in a single document. In Mongodb we can store this as a single document, in a single collection. This is where Mongodb starts enabling superior performance.

In Mongodb, to retrieve the whole entity, we have to perform:

- One index lookup on the collection (assuming the entity is fetched by id).
- Retrieve the contents of one database page (the actual binary json document).

In MySQL with 20 tables, we have to perform:

- One index lookup on the root table (again, assuming the entity is fetched by id).
- With a clustered index, we can assume that the values for the root row are in the index.
- 20+ range lookups (hopefully on an index) for the entity's pk value.
- These probably aren't clustered indexes, so the same 20+ data lookups once we figure out what the appropriate child rows are.

So the total for mysql, even assuming that all indexes are in memory (which is harder since there are 20 times more of them) is about 20 range lookups.

These range lookups are likely comprised of random IO — different tables will definitely reside in different spots on disk, and it's possible that different rows in the same range in the same table for an entity might not be contiguous (depending on how the entity has been updated, etc).

This is how Mongodb can boost performance in some use cases.

4. HORIZONTAL VS VERTICAL SCALING

Horizontal Scaling may be called one of the most important feature of using MEAN Stack because of the cost efficiency it provides for developing product.While making production ready application it is important to understand the infrastructure required to manage the client-requests.

When we increase the number of systems by getting more systems of the same type/configuration to manage increase in request load, we perform horizontal scaling. However, when we buy a new system of better configuration to manage increase in request load, we perform vertical scaling.

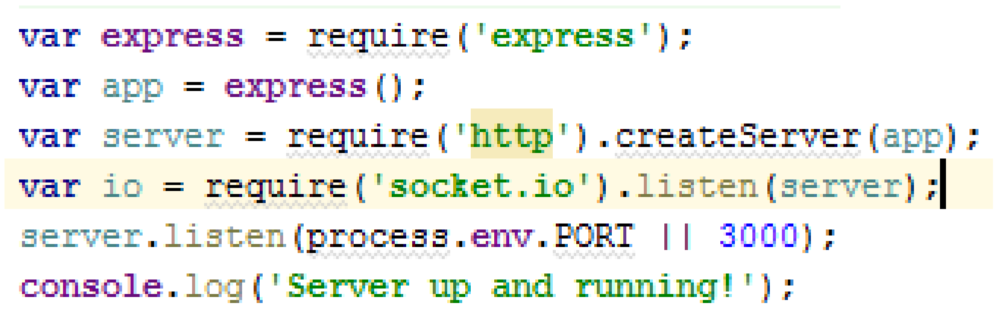
For Example, if a system with a configuration, i5 processor, 4GB RAM and 1TB Drive manages the client request. Then getting more systems with the same configuration is known as Horizontal Scaling and getting a system with a configuration, let’s say i7 processor, 16GB RAM and 2TB Drive to manage the increase in client- request would be known as Vertical Scaling.

In real life, production ready applications require much bigger infrastructure to maintain the product service. Therefore it is extremely important to cut costs wherever possible. That is why it is preferred to get large number of low performing machines as compared to a few high performing machines.

5. REQUEST, RESPONSE EFFICIENCY AND CODE MODULARITY

The implementation of MEAN stack makes an application extremely modular. That is the most important reason why MEAN apps are horizontally scalable. JAX RS can also be made modular but the cohesion of these applications is relatively more than MEAN apps. Handling http requests can easily be done by making router modules that deliver responses to the client.

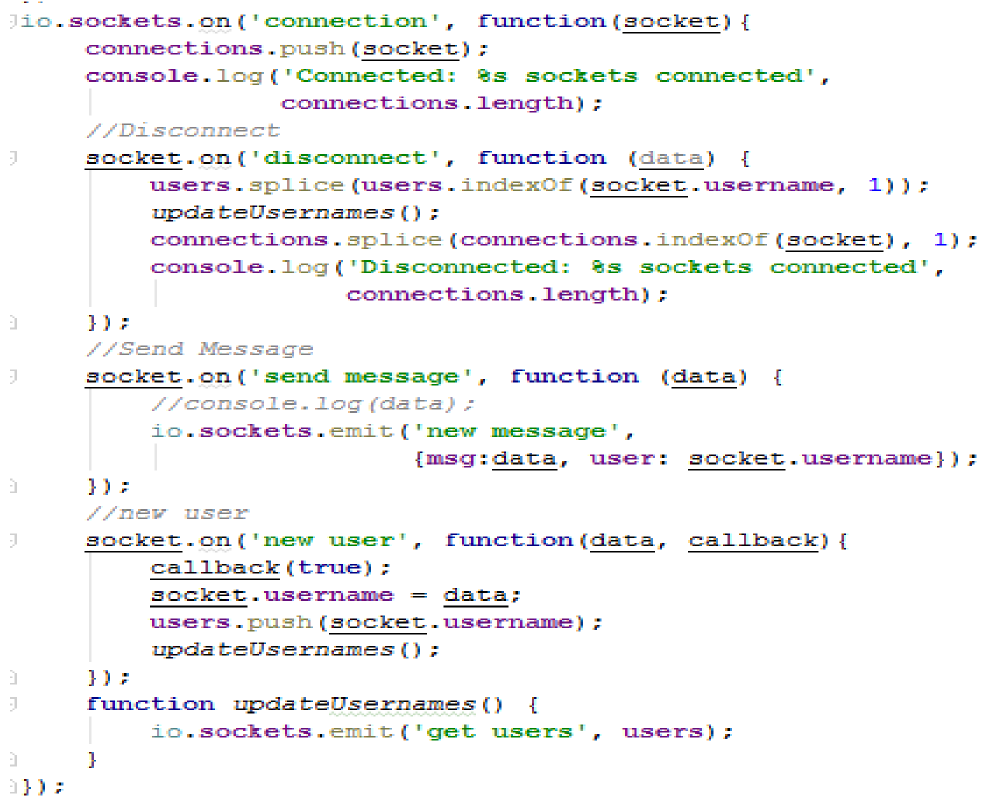
Setting up server is extremely easy using MEAN, Fig. 2 shows how easy it is to set up a server on a custom port using Express.js
.



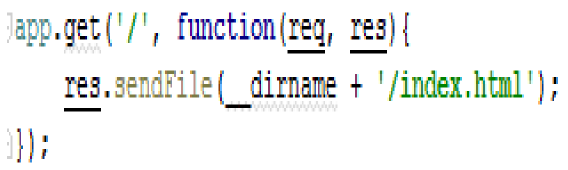
```
var express = require('express');
var app = express();
var server = require('http').createServer(app);
var io = require('socket.io').listen(server);
server.listen(process.env.PORT || 3000);
console.log('Server up and running!');
```

Fig. 2 Server startup at user selected port 3000 using Express.js. The image is an actual screenshot from the messenger app developed using MEAN.

The most important advantage of using a framework that has JavaScript as it's core is that all request and responses that travel between client and server are in the form of JSONs. Since the entire stack can handle and manipulate JSON, the additional overhead of converting xml content is drastically reduced. The JSON data can be directly sent across the web through the request header by setting 'Content-Type' to 'application/json'. Fig. 3 shows Socket.io receiving JSON content and responding to the request by sending JSON data for various functions developed for the messenger application.



```
io.sockets.on('connection', function(socket){
    connections.push(socket);
    console.log('Connected: %s sockets connected',
                connections.length);
    //Disconnect
    socket.on('disconnect', function (data) {
        users.splice(users.indexOf(socket.username, 1));
        updateUsernames();
        connections.splice(connections.indexOf(socket), 1);
        console.log('Disconnected: %s sockets connected',
                    connections.length);
    });
    //Send Message
    socket.on('send message', function (data) {
        //console.log(data);
        io.sockets.emit('new message',
                        {msg:data, user: socket.username});
    });
    //new user
    socket.on('new user', function(data, callback){
        callback(true);
        socket.username = data;
        users.push(socket.username);
        updateUsernames();
    });
    function updateUsernames() {
        io.sockets.emit('get users', users);
    }
});
```



```
app.get('/', function(req, res){
    res.sendFile(__dirname + '/index.html');
});
```

Fig. 3 Server managing user requests by fetching JSON data as a parameter and responding with appropriate JSON data or response. The image is an actual screenshot from the messenger app developed using MEAN.

6. ADVANTAGES OF MEAN IN REST API

6.1. Node

Whilst there are a multitude of ways to implement a REST API, doing so using the MEAN stack, and specifically the use of Nodes & Express, enhances productivity by reducing the development effort required; whilst delivering an effective, efficient, and highly-scalable implementation. Although presently still less commonly known than the LAMP stack, MEAN appears likely to become influential; especially given Microsoft's increasing involvement with Node. Research has shown that a Node server significantly outperforms both Apache and Nginx for serving dynamic content [10]— and Node's implementation of JavaScript is more than 2.5 times faster than the more traditional PHP approach, by efficiently utilizing available hardware.

6.2. JSON

A REST API needs to consume and produce data, encoded in a consistent manner. Whilst many choices exist (XML, YAML, etc.) a popular choice is JSON. JSON is well suited to data that needs to be both human and machine readable. JSON data is, arguably, easier for a human reader to understand than XML; and can also be easily parsed by a computer. Tools and libraries to parse JSON exist in all major programming languages and environments.

JSON's key-value pair format is ideal for use with regular parametric data — such as may be produced by a sensor device, or transmitted as a command-and-control message to a device or actuator. From Node or another JavaScript implementation JSON data is already in the native format & as such requires no further parsing unlike, for example, XML.

6.3. Mongodb

The advantage of the use of a document-oriented database, such as Mongodb is that is offers a dynamic (rather than fixed, and rigid) schema. This means that if the data structure of the database is required to change, as a result of needing to store additional parameters; new data with a different structure can be accommodated within the database,
alongside earlier data, without the need to perform large scale data manipulation on the whole database. Mongodb is also a good choice for storing JSON encoded data. Mongodb internally stores data in an efficient binary JSON format (BSON) — which allows for quick and easy import and export. It is also is also ideally suited to storing and processing large volumes of data. It can scale to thousands of nodes and petabytes of data.

Mongodb also exhibits better runtime performance for simple operations at a small-scale (single node), than Microsoft SQL Server Express.

7. CONCLUSIONS

The comparison carried out by comparing the same application created using MEAN and JAX RS clearly show that MEAN is a better alternative for creating Web Apps because of it's lightweight, efficient and well structured framework. But the performance of both the applications was similar due to the limited features of the application developed. However, this performance will increase greatly in the case of production ready application because of the numerous features and the size of infrastructure required to maintain the application.

A combination of Node & Express.js, with a Mongodb database provide an excellent implementation of a JSON based web-service; not least because all of the tools within the stack will natively utilise the underlying JSON data. Such a MEAN stack web-service is also highly scalable - thanks to the benefits of both Node and Mongodb.

REFERENCES

[1] MEAN.io. (2015) MEAN — Full-Stack JavaScript Using MongoDB, Express, AngularJS, and Node.js. [Online].

Available: http://mean.io/

[2] S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," IEEE Internet Computing, vol. 14, no. 6, pp. 80–83, 2010. [Online] Available: http://doi.ieeecomputersociety.org/10.1109/MIC.2010.145

[3] I. K. Chaniotis, K.-I. D. Kyriakou, and N. D. Tselikas, "Is Node.js a viable option for building modern web applications? A performance evaluation study," Computing, pp. 1–22, 2014. [Online]. Available: http://dx.doi.org/10.1007/s00607-014-0394-9

[4] A. Leff and J. T. Rayfield, "Web-application development using the Model/View/Controller design pattern," Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference, pp. 118–127, 2001.

[5] npm: the Node package manager. [Online]. Available: https://www.npmjs.com

[6] Github. (2015) Projects, Applications, and Companies Using Node. [Online].Available: https://github.com/joyent/node/wiki/Projects,-Applications,-and-Companies-Using-Node

[7] Baron Schwartz, "Analysis of PayPal's Node-vs- Java Benchmarks" [Online]. Available: https://www.vividcortex.com/blog/2013/12/09/analysis-of-paypals-node-vs-java-benchmarks/

[8] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format Interchange Format," 2014. [Online]. Available: https://tools.ietf.org/html/rfc7159

[9] Z. Parker, S. Poe, and S. V. Vrbsky, "Comparing NoSQL MongoDB to an SQL DB," Proceedings of the 51st ACM

[10] Welcome to NGINX Wiki's documentation. [Online]. Available: https://www.nginx.com/resources/wiki/