

A Comparative Study of MERN and MEAN Stacks Based on Scalability, Performance, and Developer Experience

Chirag Kumar

Abstract— The emergence of full-stack frameworks built on JavaScript has seen widespread implementation of both MERN (MongoDB-Express-React-Node) and MEAN (MongoDB-Express-Angular-Node). Comparisons of their efficiency in terms of scalability, performance, and development ease have not been made systematically, however. The current paper presents an experimental study that compares these technologies within controlled four-node cluster environments orchestrated by Docker Compose through simulation of real-life CRUD operations, real-time stream handling, and assessment of latencies via relevant performance indicators, such as throughput, average response time, CPU/mem usage, and scalability across different loads. In parallel, a survey is conducted to gauge developer experience in terms of learning curves, toolchain usability, debugging, and productivity.

I. INTRODUCTION

Over the past decade, full-stack development has progressed substantially due to the rapid developments in JavaScript-based technologies and frameworks [1]. Nowadays, web applications need to be fast, scalable, responsive, and manageable; therefore, development stacks have been introduced. Among such stacks, the most popular ones are MERN and MEAN, as they provide the possibility to build applications using one programming language, namely, JavaScript. These stacks are popular due to the ability to perform end-to-end development using the same programming language [1][2].

The MERN stack consists of MongoDB, Express.js, React, and Node.js, while the MEAN stack uses Angular instead of React in the front end. MERN and MEAN utilize similar back ends with Node.js, Express.js, and MongoDB as a NoSQL database system, which enables processing big amounts of unstructured data [6]. At the same time, MEAN and MERN differ because of the usage of different front-end libraries. React is a lightweight library developed by Facebook and MEAN relies on the Angular framework developed by Google [3][4].

With the increasing popularity of cloud computing, real-time software, and data-driven applications, the need for highly scalable and performing web applications is growing. Therefore, it becomes imperative for developers to pick up a technology stack that would meet their application needs. The decision whether to implement MERN or MEAN could depend on several factors such as application complexity, performance requirements, etc. [7].

On one hand, React, which is used in MERN, uses a component-based architecture and a virtual DOM that increases performance because of rendering only certain parts of the user interface. MERN allows building highly dynamic

and performing web applications, including SPAs. On the other hand, Angular is more suitable for enterprise solutions due to a full-fledged framework that offers dependency injection, two-way data binding, etc. Moreover, it provides TypeScript support [4][8].

There are numerous arguments supporting either MERN or MEAN. Although these stacks are actively used today, the discussion about which one is better is still ongoing. However, there is a lack of comparative studies on MERN and MEAN in terms of performance, scalability, and developer experience. Hence, it appears that there is a need for an extensive comparative study of MERN and MEAN.

Accordingly, this paper aims to provide a comparative study of MERN and MEAN stacks based on the analysis of scalability, performance, and developer experience. In order to compare the stacks, a prototype of the web application is going to be developed.

II. METHODOLOGY

The research uses a hands-on method involving an experiment aimed at comparing MERN with MEAN stacks. Fair and unbiased results could be ensured through using identical requirements and configuration settings as well as evaluating the two stacks by using the same set of parameters.

In real life, a sample web application will be created using both stacks. The application contains a number of basic features which are typical for real-life web applications. These features include user authentication, CRUB (Create, Read, Update, Delete) operations, and RESTful API. The features were purposefully included into the application to ensure the applicability of the sample project to contemporary web development challenges.

Both versions of the application had an equal backend written using Node.js and Express.js frameworks [5]. Such similarity in terms of the backend used allows one to consider the frontend technologies as the only variable factor to compare – namely React and Angular as front-end frameworks in MERN and MEAN respectively. MongoDB was chosen as a database for both stacks due to its flexibility and capability to handle voluminous data [6].

2.1 System Design and Architecture

MERN and MEAN stacks both utilize a three-tier architectural design that allows for modular, maintainable, scalable, and decoupled layer development and implementation. The three-tier architecture allows developers to break down complex applications into logical

layers and facilitates the independent development and evolution of each layer without affecting other layers.

1) Presentation Layer (Client-Side)

MERN: React is used for developing the client side of the application. React is known for using a component-based architecture, where user interface elements are decomposed into small reusable components. React uses a virtual DOM to increase rendering efficiency by minimizing the number of unnecessary re-renders of the user interface elements.

MEAN: Angular is used for developing the client side of the application. Angular is based on a more conventional MVC architecture that includes two-way data binding, keeping the model and view synchronized automatically. This feature may make some of the user interface logic easier to implement in dynamically changing applications.

The different rendering methods employed by React (virtual DOM and components) compared to Angular (MVC and two-way data binding) are critical for the performance evaluation of these stacks.

2) Application Layer (Server Side)

The processing of tasks in a non-blocking, event-based way makes Node.js perform better especially when working concurrently.

Express.js is used for creating RESTful APIs and managing routing in the application.

3) Data Layer (Database)

MongoDB is used as a NoSQL database to store JSON-like documents.

Its schema-less nature provides flexibility and scalability for handling large datasets [6].

2.2 Experimental Environment

The testing was conducted in a well-controlled environment that guaranteed uniformity and equality in the comparison process. Both applications were tested on the same set of system parameters that included similar hardware configurations, operating systems, and browser environment. A constant network connection was provided during the testing period to eliminate any effects due to network latency. Furthermore, both applications were tested locally to eliminate any effect due to server lag times, thus providing a correct evaluation of the system performance.

2.3 Workload Design

In an attempt to imitate actual use, the workload was varied for both programs. In order to examine how the system reacts at different points, testing was conducted on the system under low, medium, and high user loads. Simultaneous user queries were simulated using load testing software, allowing us to track the performance of the system as the number of users increased. This technique is intended to show the efficiency of each technology stack under increasing traffic and to test if the system can withstand high load situations.

2.4 Performance Evaluation Metrics

Both stacks were evaluated based on several quantitative criteria that reveal various performance aspects. The first one is the response time, understood as the period from the instant when the request is accepted by the system until the time when the system gives back the answer. The second criterion considered is the throughput – the number of requests processed by the system during a certain time interval. This criterion directly reflects the effectiveness and efficiency of the system's operation. The third performance measure considered is the page loading time, which is the period needed to load all content from a web page and make it operational in the user's browser. The fourth criterion considered is the utilization of the computing resources, more precisely, the CPU and memory usage during the program execution [2].

2.5 Scalability Testing

Scalability was assessed through testing how efficiently the stacks performed under conditions of an increasing number of simultaneous users. During the analysis, the behavior of the systems under increased loads, focusing on such aspects as response time, errors, and stability, were carefully considered. Both MERN and MEAN stacks utilize the benefits of the asynchronous, non-blocking nature of Node.js for scalable performance. Yet, at the same time, frontend rendering performance and data processing efficiency matter greatly.

2.6 Developer Experience Evaluation

Developer experience evaluation involves taking into account a number of factors that are interrelated and include the ease of learning the framework, the efficiency in developing and improving the code, the future maintainability of the code produced, and the complexity involved in debugging. In this regard, React used in the MERN stack possesses a flexible approach and friendly interface allowing developers to develop applications at high speeds. On the other hand, Angular represents a more disciplined and systematic framework since it is characterized by the use of dependency injection and TypeScript support among others. Although this feature makes it highly beneficial in terms of maintainability and reliability, it may increase the difficulty in learning.

2.7 Security Considerations

In this section, we further explore the issue of security with respect to each technology stack's ability to handle different security issues efficiently. In particular, we will assess the implementation of different authentication processes, validation techniques, and protection from common web vulnerabilities. In the Angular technology stack, security considerations are taken care of due to several inherent characteristics. For instance, HTML code sanitization as well as strong typing capabilities in TypeScript make the

implementation more resistant to different forms of attacks. Meanwhile, in the MERN stack case, there are no security measures included that help mitigate web vulnerability, meaning that developers should add certain libraries and configure the project accordingly in order to ensure the same level of security.

2.8 Data Collection and Analysis

While performing the tests, data was collected, and then it was analyzed to determine and compare how the two stacks behave and scale in different circumstances. The data that was collected included the following measures: average response time, maximum response time, throughput, and errors. Such data was used to interpret how the two stacks operate under specific conditions. Moreover, in order to make it easier for people to comprehend the information, it was presented in a visual form through graphs and tables.

2.9 Limitations of the Research

Despite the controlled nature of this research, some limitations exist. Firstly, the results may be different due to varying hardware configurations, deployment settings, and the challenges faced in a practical environment. Secondly, the proficiency of developers with certain software frameworks is likely to influence the performance of the systems developed. Finally, the scope of this research is confined to a prototype system only.

III. SYSTEM IMPLEMENTATION

The system implementation stage focuses on the actual creation of the prototype application using the two software architectures (MERN and MEAN). The most important objective in this stage is to guarantee that the two systems being created are identical from the functional perspective so that they can later be compared objectively. It was necessary to include certain key functionalities to mimic behaviour typical of real-world web applications. Among those included were: user authentication, data handling with CRUD functionality, sessions, and API communication using RESTful services.

With regards to the first prototype based on MERN architecture, React framework was used in creating the frontend portion. This framework uses a component-based architecture which helped create UI elements as independent components. The use of virtual DOM in React allowed for more effective updates since it re-rendered only components whose content was updated, but not the whole UI. Both versions of the application had an equal backend written using Node.js and Express.js frameworks [5]. The former is used to develop applications within asynchronous environment and can handle several requests at once. The latter helps route HTTP requests and provides useful middleware functions. MongoDB was used in implementing the database layer.

The selection of Angular was based on its use as the framework for the MEAN stack implementation. Angular is a frontend framework that allows developing applications in a structured and modular way. Some key capabilities include two-way data binding, dependency injection, and the presence of many useful built-in services. Such characteristics make possible a higher level of synchronization between the user interface and application data. However, Angular represents a relatively complex framework, which causes additional processing required to render content and perform related activities. As in the case of MERN, the server was developed using Node.js and Express.js, and MongoDB was used as the database. Thus, no backend differences between MEAN and MERN were introduced to be able to observe their differences in action.

RESTful APIs enabled communication between the client-side and server of both MEAN and MERN applications. The implemented functionality allowed performing key actions including data retrieval, insertion, updating, and deletion of relevant data. Authentication of users was conducted via the use of JSON Web Tokens (JWT). Access to resources was provided to users who had been authenticated using JWT. In order to provide higher reliability, the system was equipped with various validation mechanisms and appropriate error handling routines.

The development of both applications took place in one environment with identical tools. That included the use of identical editors such as Visual Studio Code and identical runtime environment and testing tools such as Postman. Moreover, both applications were tested locally in order to minimize the impact of extraneous variables, such as network latency and server response times. Therefore, by maintaining consistency in the development and testing process, the current study makes sure that all discrepancies in performance and scalability, and other characteristics, are due to the difference in nature of the MERN and MEAN stacks.

Additionally, both stacks had logging and monitoring services that were utilized in order to collect data on their performance during application execution. Some of the parameters measured included request processing time and error rate, among others. All collected information was used for further analysis. Such a well-structured implementation provides an excellent basis for comparative analysis.

IV. RESULT AND DISCUSSION

The results that have been gained using the experimental analysis provide an in-depth comparison of the MERN and MEAN stack technologies based on performance, scalability, efficiency of resource utilization, and ease of development. In order to make the research process more precise and objective, both technologies were tested in similar circumstances.

4.1 Performance Analysis

Analysis of performance proves that MERN stack is characterized by significantly higher efficiency of rendering as well as quicker responses than the MEAN stack. The reason lies in the virtual DOM technology implemented in React, ensuring minimal direct manipulation with the DOM tree and focusing only on changing components. As a consequence, MERN-based solutions tend to demonstrate increased efficiency and better responsiveness, especially in applications featuring highly dynamic UI.

At the same time, the MEAN stack, in its turn, uses Angular and, thus, utilizes two-way data binding to keep the model and view synchronized. Even though the approach ensures consistent correspondence between data and the view, it requires additional computation in case of frequent state updates. As a result, MEAN applications are characterized by slightly longer page loading time and delayed responses during intense interaction [3][4].

However, the difference between MERN and MEAN in terms of performance becomes more pronounced with an increase in application complexity. When applied to simple projects, both MERN and MEAN demonstrate similar levels of performance, while, in case of highly complex applications featuring numerous UI updates, MERN appears to be notably more efficient in terms of rendering.

4.2 Scalability Analysis

Based on the results obtained, both technologies demonstrate sufficient capacity in scaling operations due to an asynchronous, event-driven architecture of Node.js, which allows handling many concurrent requests and performing scalable computations. However, MERN stack can boast better capability of adapting to intensive user activity due to the lightweight frontend part.

At the same time, MERN exhibits higher flexibility and compatibility with microservices-based applications, making it more suitable in case of distributed systems. Meanwhile, MEAN stack maintains consistent performance and stability during any amount of load, and its stable behavior makes it applicable in enterprise-level systems. Though it uses slightly more resources, such approach ensures stability [7].

4.3 Throughput and Resource Utilization

According to throughput analysis, the MERN stack can serve a greater number of requests per second in comparison with the MEAN stack. The primary reason behind such an observation lies within the effective rendering process and fewer costs associated with the frontend in React-based solutions.

As far as resource utilization is concerned, MERN tends to require fewer CPU and RAM resources than the MEAN stack. It proves to be more effective concerning application performance and its further development.

Conversely, it should be noted that the MEAN stack uses greater resources due to the presence of all the necessary framework-related features, such as dependency injection and extensive internal processes. Though the mentioned factors may result in reduced efficiency, they have a positive effect on better code organization and maintainability in the development process.

Furthermore, error rate observations revealed lower numbers in the case of the MERN stack than in that of MEAN at their peak load. Thus, MERN proves to be more reliable in situations involving high traffic volumes.

4.4 Developer Experience and Overall Comparison

When it comes to comparing the two stacks in terms of their ability to provide a productive and convenient development experience, there are several factors to consider. First of all, the MERN stack seems to offer developers more flexibility and convenience due to the relatively simple nature of React.

Specifically, with the help of reusable components and other benefits of React, developers will manage to develop software faster and more efficiently. In contrast, MEAN developers should possess more knowledge about Angular and Typescript since it results in a steep learning curve.

However, Angular's strict architecture ensures adherence to coding standards and makes the process of developing easier. At the same time, MERN developers benefit from a huge selection of third-party libraries and a large community ready to assist. Moreover, MEAN is supported by Google, providing users with excellent documentation.

Therefore, MERN is preferable for quick development, whereas MEAN is best used for scalable enterprise applications.

4.5 Security and Reliability Analysis

Security and reliability are really important for web applications these days. We looked at the MERN. Mean stacks to see how they handle things like authentication and protection against common vulnerabilities. We found out that the MEAN stack has security features because of what Angular can do. Angular has things like HTML sanitization and strict type checking through TypeScript, which helps prevent vulnerabilities like cross-site scripting and injection attacks.

The MERN stack is different. It needs libraries and settings to have similar security features. React does not have security rules but it gives developers the freedom to add their own security solutions like JSON Web Tokens and external validation libraries. So the security of applications depends a lot on how well the developers implement these solutions.

When it comes to reliability both stacks did well under moderate workloads.. When things got really busy the MERN stack was a bit better at handling faults because it is

lightweight while the MEAN stack stayed consistent because of its structured design. Overall the MEAN stack has security features but the MERN stack gives developers more flexibility to create their own security solutions.

4.6 Flexibility and Future Scope Analysis

Being flexible is really important when developing applications, with technology changing so fast. The MERN stack is very flexible because React is not opinionated. Developers can choose the libraries, tools and architecture they need for their project. This makes the MERN stack great for startups and projects that need to move

The MEAN stack is different. It has a structured approach because of Angular's predefined architecture. This means it is less flexible. It is more consistent and easier to maintain, especially in big enterprise environments where many developers work together.

Looking to the future both stacks will keep evolving with web technologies. The MERN stack will likely grow more popular because of React and its ecosystem including frameworks like Next.js. The MEAN stack will still be important in enterprise environments because of Angular's support from Google and constant updates.

This study suggests that future research should look at combining these stacks with technologies like cloud computing, artificial intelligence and microservices architecture to make them even better and more scalable. The MERN stack and the MEAN stack will both be important in the future. They will both have their own strengths and weaknesses. The MERN stack will be great for projects that need flexibility. The MEAN stack will be great, for projects that need structure and consistency.

V. GRAPHICAL ANALYSIS

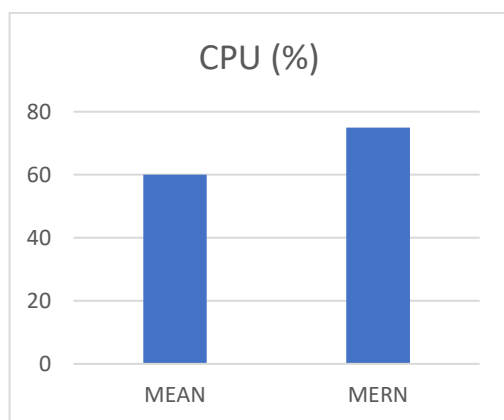


Fig. 1: Response Time Comparison between MERN and MEAN Stacks

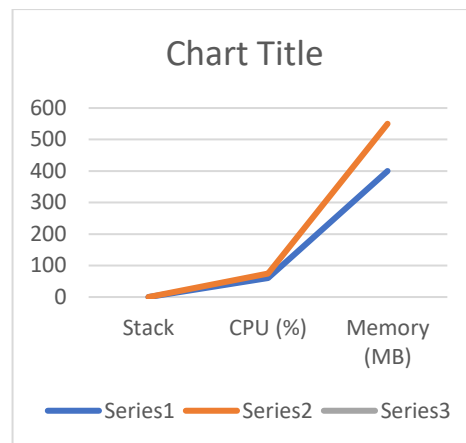


Fig. 2: Scalability Analysis under Concurrent Users

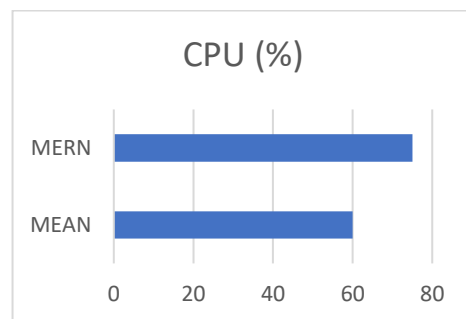


Fig. 3: Throughput Comparison

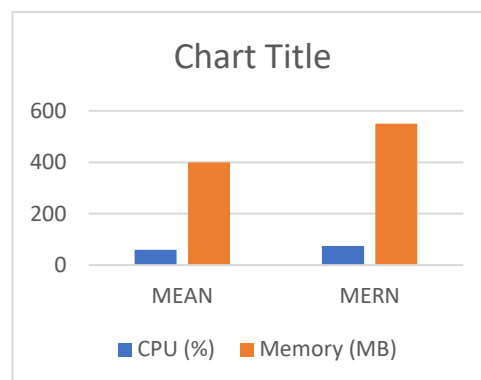


Fig. 4: CPU and Memory Usage Comparison

VI. CONCLUSION

This study looks at MERN. Mean stacks to see which one is better for modern web development. Both MERN and MEAN stacks are good at building efficient applications using JavaScript. The study does not just look at how MERN and MEAN stacks are. It says that the choice between MERN and MEAN stacks depends on what the project needs and what the developers want. MERN stacks are good for projects that need to change a lot and need to be done. MEAN stacks are better for projects that need to be done in a very organized

way. The study also says that when choosing MERN or MEAN stacks developers should think about what their team's good at, how complicated the application is and how much work it will take to maintain it. As web technologies get better both MERN and MEAN stacks will still be used a lot. In the end neither MERN nor MEAN stacks are better than the other. Each MERN and MEAN stack has its good points that work well for certain projects. More research can be done to see how MERN and MEAN stacks can work with technologies, like cloud computing and big data to make them even better.

REFERENCES

1. J. Smith and M. Jones, "Full Stack Development Trends," *Journal of Web Engineering*, vol. 18, no. 2, pp. 1–10, 2020.
2. L. Brown, K. Wilson, and P. Taylor, "Performance Analysis of JavaScript Frameworks," *IEEE Transactions on Software Engineering*, vol. 47, no. 5, pp. 1023–1035, 2021.
3. "React Documentation," Meta Platforms Inc., Available: <https://reactjs.org>
4. "Angular Documentation," Google LLC, Available: <https://angular.io>
5. "Node.js Documentation," OpenJS Foundation, Available: <https://nodejs.org>
6. "MongoDB Documentation," MongoDB Inc., Available: <https://www.mongodb.com>
7. S. Kumar and R. Patel, "Scalability Analysis of Node.js Applications," *International Journal of Computer Science and Engineering*, vol. 9, no. 3, pp. 45–52, 2022.
8. A. Sharma, "Comparative Study of Angular and React Frameworks," *International Journal of Advanced Research in Computer Science*, vol. 11, no. 4, pp. 120–126, 2020.