International Journal on Advances in Engineering Technology and Science
Volume: 5, Issue: 1, 2024, e-ISSN: 2455-3131
DOI: 10.5281/zenodo.10709588

Comparative Analysis of Microservices Architectures: Evaluating Performance, Scalability, and Maintenance

Mahesh Kumar Bagwani[1], Gaurav Kumar Shrivastava[2]

[1]MTech Scholar, [2]Associate Professor
Sanjeev Agrawal Global Educational University, Bhopal

[1]maheshbagwani7@gmail.com, [2]gourav.s@sageuniversity.edu.in

***Abstract*— Microservices have become a key architectural paradigm in the ever-changing field of web application development. This study compares and contrasts microservices architectures in great detail, paying close attention to each one's scalability, maintenance, and performance. This research analyses a variety of microservices frameworks and reveals the subtleties of their architecture through a methodical assessment. Through an examination of critical performance indicators like response times, scalability under different workloads, and ease of ongoing maintenance, the study seeks to identify best practices and draw attention to potential issues related to each architecture. The knowledge gathered from this research will help architects and developers choose or optimize microservices frameworks with confidence. This paper not only contributes to the academic discourse but also offers pragmatic guidance for real-world applications, ensuring that the chosen architecture aligns seamlessly with the specific needs of a project. Embracing a holistic approach, this research provides a nuanced understanding of the trade-offs inherent in diverse microservices approaches, fostering a more robust and informed development community.**

***Keywords*— Microservices, Monolithic, Jenkins, JMeter, KVM**

I. INTRODUCTION

In the fast-evolving landscape of software development, organizations seek robust technological solutions. Over time, various architectural approaches have been crafted by software developers to enhance resource efficiency and meet functional requirements. The traditional monolithic architecture, widely successful in small and large-scale projects, faces performance challenges with increasing data volumes. Solutions have emerged, such as technology migration and the adoption of more powerful servers, but these may lead to higher resource expenses if not chosen wisely.

In recent decades, innovative architectures, notably microservices, have gained prominence, replacing monolithic systems. Microservices offer a distributed approach with isolated services, but their implementation, especially in scaling to the Cloud, introduces challenges. Automation tools like DevOps, Docker, Chef, and Puppet streamline processes but require additional development, migration, and integration efforts. Companies adopting microservices face concerns about infrastructure costs, orchestration in production, and organizational challenges.

This study addresses the lack of precision in measuring the migration process from monolithic to microservices architectures. It emphasizes compiling research results on the evaluation of both architectures, focusing on performance metrics like CPU, memory consumption, network performance, and development complexities. The study evaluates two scenarios: a monolithic architecture on a virtual server with KVM and a microservices architecture running in containers. Stress tests under similar conditions allow for a quantitative comparison using a nonparametric regression model.

Key contributions of the study include a critical analysis of research on monolithic versus microservices performance, a review of variables impacting migration performance, and an evaluation of response times and resource consumption in microservices architecture. The article concludes with insights into the state-of-the-art, theoretical framework, experimental design, findings, and future work considerations.

II. LITERATURE REVIEW

In this section, we dive into the current state of research in the field. Stubbs et al. [1] explore container technology and propose Serf Node for service discovery in microservices architectures. Villamizar et al. [2] compare the cost and performance of web applications in different architectures, revealing microservices' cost-effectiveness. Al-Debagy and Martinek [3] compare microservices and monolithic architectures, emphasizing performance under various loads. Guaman et al. [4] focus on migrating a monolithic application to microservices and analysing performance metrics.

Akbulut and Perros [5] provide insights into microservices' performance, considering factors like query response time and hosting costs. Singh and Peddoju [6] compare microservice and monolithic deployments, showcasing the former's advantages in terms of deployment time and continuous integration. Various studies, including Ponce et al. [7], Taibi et al. [8], and Mazlami et al. [9], explore migration techniques and technical debt reduction in transitioning from monolithic to microservices.

Kalske et al. [10] examine the evolution challenges of transitioning to microservices, emphasizing its benefits in handling complexity. Bures et al. [11] focus on identifying business applications' transactional contexts for microservices design. Sarkar et al. [12] analyse the architectural features of an industrial application for migration to microservices. Debroy and Miller [13] discuss the challenges and infrastructure changes in adopting microservices at VARIDESK.

In comparison, this study stands out by conducting a comprehensive performance evaluation of both hardware resources and applications in monolithic and microservices architectures. The unique approach includes mathematical modelling for accurate results during service execution, shedding light on productivity, cost reduction, and efficiency in hardware resource utilization. This comparative analysis, especially in a real-world context, distinguishes our study from previous research efforts.

III. MONOLİTHİC VS MİCROSERVİCES

Monolithic architecture relies on a single development technology, limiting flexibility as changes in one part of the system require building and deploying a new version of the entire system. This approach integrates presentation, processing, and storage into a single component running on a server. While offering stability and full system control, monolithic architectures have drawbacks, including rigidity and difficulty adapting to new needs. Major players like IBM and Sun Microsystems have utilized this approach, but its proprietary nature and high costs pose challenges.

As technology evolves, the complexity of modern systems demands improvements in software production and performance. Monolithic architectures, with their inherent defects, are giving way to more contemporary solutions like microservices. While some applications, like firmware and certain security tools, may still find efficiency in monolithic structures, the shift toward more adaptable architectures is evident.

As Martin Fowler puts it, "A microservice architectural style is an approach to developing a single application as a suite of small services, each running in its process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery." In simpler terms, microservices break down a large application into small, independent services that run their processes and communicate through APIs, allowing for flexibility, scalability, and easier maintenance [14].

Unlike monolithic architectures, microservices operate as a collection of individual services that can tolerate failures and enhance availability. This approach embraces a culture of automation, with decentralized processes that permit independent deployments. The application's structure differs significantly from a monolithic one, as shown in Figure 1. Microservices encapsulate complex business scenarios and offer the advantage of updating only specific components, rather than the entire system. This architectural shift brings forth a new era of adaptability and efficiency in software development [5,32,33].

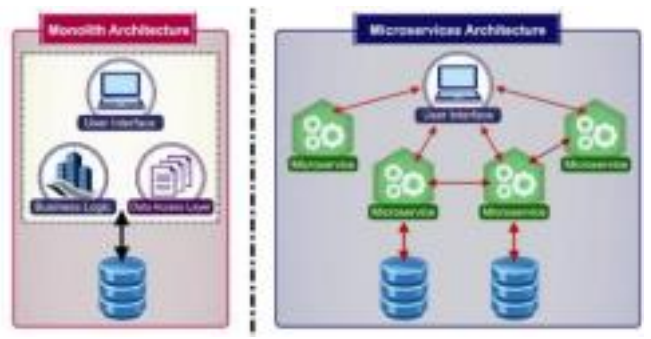

Fig-1. Visual representation of a monolithic versus Microservices architecture.

In our study, we explored two application architectures: monolithic, running on a Kernel-based Virtual Machine (KVM), and microservices orchestrated with Docker containers. Here's a brief overview of the tools we employed for research, performance measurements, and implementation:

Kernel-based Virtual Machine (KVM):

- Purpose: Virtualization technology for Linux systems.
- Usage: Implemented as a hypervisor, allowing support for multiple operating systems and virtualization of hardware.
- Associated Application: QUEMU, an operating system emulator.
- Contribution: Provided a stable platform for running the monolithic application with ease of use through a graphical interface.

Docker:

- Client and Server: An open platform for building, transporting, and running distributed applications.
- Docker Compose: Used to define configurations for the Microservices environment.
- Contribution: Enabled the creation of lightweight, portable containers for individual services, fostering independence and ease of deployment.

Other Tools:

- Sublime: A text editor for code and configurations.
- Git: A version control system for maintaining application integrity.
- Java8 (JRE), NodeJS, NPM: Runtime environments for application execution.
- MySQL and MongoDB: Databases used for storing information generated in stress tests.
- Performance Measurement Tools:

- Apache JMeter: A Java-based open-source tool for load testing and performance measurement.
- Server Agent: A component of JMeter for collecting and interpreting data during stress tests.
- New Relic: A performance analysis service focusing on real-time data collection and analysis during stress tests.

Containers:

- Definition: Similar to virtual machines but without the overhead of a separate kernel.
- Benefits: Fast startup, resource efficiency, and independence for each workload.
- Challenges: Poor visibility of processes and potential cybersecurity issues.
- Docker: Widely adopted for container management, providing solutions for large scale container projects.

Our approach involved an array of open-source tools, allowing us to implement, configure, and stress-test both monolithic and microservices architectures effectively. These tools played a crucial role in enabling flexibility, scalability, and ease of deployment in our research endeavors.

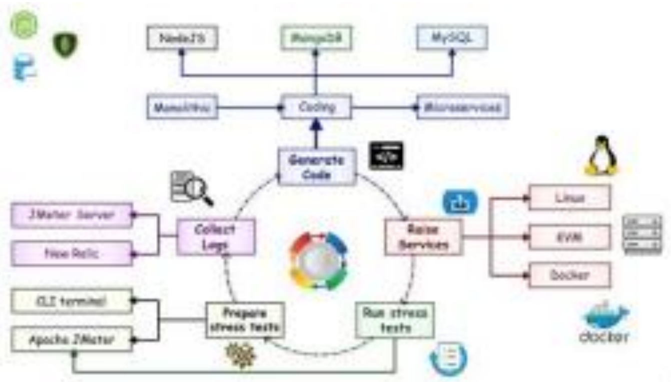

Fig-2. Tools for performance analysis.

IV. IMPLEMENTATİON

4.1. Application Architecture Design

This section outlines the research process for comparative performance analysis between monolithic and microservices architectures. It covers the application architecture design, data collection, and preliminary results.

4.1.1. Monolithic Architecture (KVM)

The monolithic application, running on KVM, involves a Node.js cluster, load balancer, and target groups. The application handles forums, chats, comments, and notifications. A hierarchical structure and key files are described.

4.1.2. Microservices Architecture (Docker Containers) The microservices application uses Docker containers with AWS's example as a foundation. Each service (user, thread, post) runs independently. The directory structure is more organized. Key files and Docker configurations are detailed.

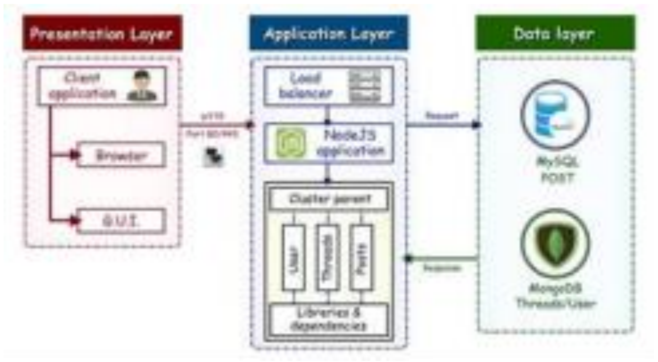

Fig-3. Monolithic application architecture implemented

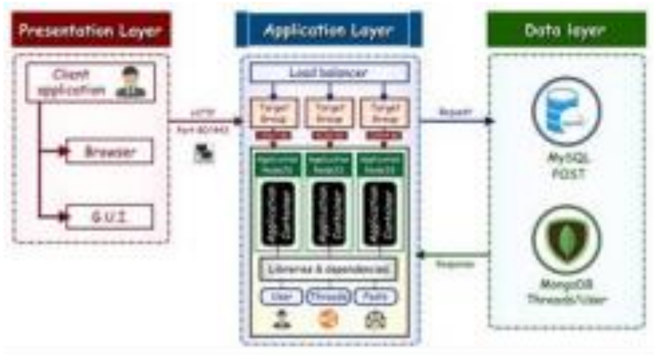

Fig-4. Microservices application architecture implemented.

4.2. Performance Testing Scenarios

Two scenarios are presented:

Monolithic Architecture (KVM): Involves a Node.js cluster, load balancer, and databases running on KVM.

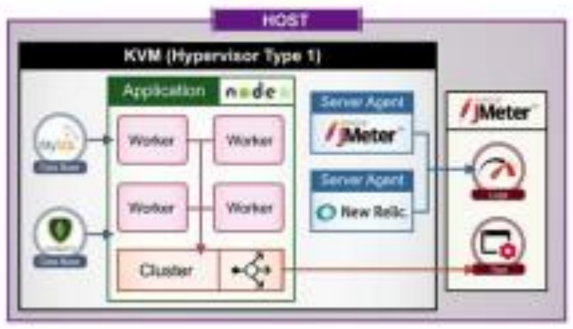

Fig-5. First Case: Monolithic application on a KVM

Microservices Architecture (Docker):

Docker orchestrates containers with separate services for users, threads, and posts. The databases run in containers, and a sync volumes container keeps data synchronized.

4.3. Data Collection, Experiments, and Preliminary Results

4.3.1. Data Collection and Experiments

Stress tests are configured with JMeter for both scenarios. Two scenarios involve generating and selecting data. Tests focus on server-side performance. Results include total requests, errors, duration, and requests per second.

4.3.2. Comparative Analysis and Preliminary Results

• Case 1: Monolithic - 273 requests (2.5/s),

Microservices - 1053 requests (3.1/s).

• Case 2: Monolithic - 2 errors, Microservices - 0 errors. Microservices perform better in requests per second. • Microservices exhibit better performance in processing requests and duration.

• A mathematical model is applied to analyze CPU, memory, disk, and network performance. Examples of recorded data are provided for microservices and monolithic architectures.

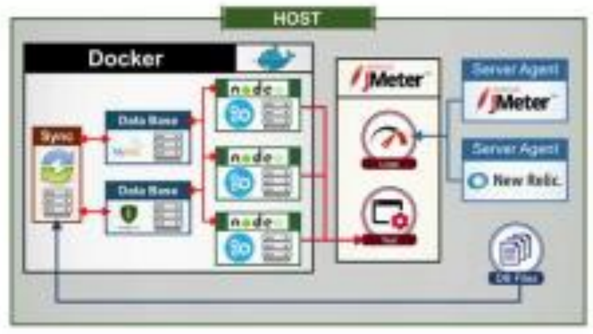

Fig-6. Second Case: Application with microservices on containers

4.4. Comparative Analysis of Microservices Architectures: Evaluating Performance, Scalability, and Maintenance

4.4.1. Data Collection Process and Experiments

The stress tests were conducted using JMeter, comparing two scenarios: Microservices and Monolithic architectures. Both scenarios underwent identical configurations, with Case 1 generating GET requests to create databases and data, while Case 2 involved GET requests to select information from MySQL and MongoDB. The server-side performance was the focal point, terminating the process upon receiving a server response.

4.4.2. Comparative Analysis and Preliminary Results

The analysis, validated with New Relic, focused on performance, scalability, and maintenance.

Request Processing:

Microservices processed all 1053 requests successfully, monolithic encountered two errors in Scenario 1.

Execution Time:

Microservices demonstrated quicker execution times:

- Case 1: Microservices (00:01:29) vs. Monolithic (00:01:50)
- Case 2: Microservices (00:12:08) vs. Monolithic (00:14:17).

Requests/s Enhancement:

Microservices outperformed in requests processed per second:

- Case 1: Microservices (3.1/s) vs. Monolithic (2.5/s).
- Case 2: Microservices (1.4/s) vs. Monolithic (1.2/s).

Resource Utilization:

Microservices exhibited efficient CPU, memory, disk, and network performance compared to monolithic.

Table 1. Results of the Stress Test for Both Scenarios:

	Monolithic - Case 1	Microservice - Case 1	Monolithic - Case 2	Microservice - Case 2
Total Requests	273	273	1053	1053
OK	273	273	1051	1053
Error	0	0	2	0
Duration time	0:01:50	0:01:29	0:14:17	0:12:08
Requests/s (average)	2.5/s	3.1/s	1.2/s	1.4 /s
Min	7 ms	4 ms	22 ms	8 ms
Max	3677 ms	2793 ms	35,784 ms	10,832 ms
Average	1150 ms	936 ms	3934 ms	3411 ms
Median	695 ms	312 ms	2548 ms	715 ms
Standard deviation	1094.66 ms	1082.4 ms	38.05 ms	4220.21 ms
Total data Received	85,014.43 KB/s	105,193.29 KB/s	86,342.29 KB/s	102,433.12 KB/s
Sent	0 KB/s	0 KB/s	0 KB/s	0 KB/s

In conclusion, the comparative analysis Favors microservices, showcasing superior reliability, faster execution, and efficient resource utilization. These findings highlight the advantages of microservices over monolithic architectures in terms of performance, scalability, and maintenance. The adoption of microservices is recommended for optimizing system performance and ensuring future scalability.

REFERENCES

[1] Stubbs, J.; Moreira, W.; Dooley, R. Distributed systems of microservices using docker and serfnode. In Proceedings of the IEEE 2015 7th International Workshop on Science Gateways, Budapest, Hungary, 3–5 June 2017; pp. 34–39.

[2] Villamizar, M.; Garces, O.; Ochoa, L.; Castro, H.; Salamanca, L.; Verano, M. Infrastructure cost comparison of running web applications in the cloud using AWS lambda and monolithic and microservice architectures. In Proceedings of the 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Cartagena, Colombia, 16–19 May 2016; pp. 179–182.

[3] Al Debagy, O.; Martinek, P. A Comparative Review of Micr oservices and Monolithic Architectures. In Proceedings of the 2018 IEEE 18th International Symp osium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, 21–22 November 2018; pp. 149– 154.

[4] Guaman, D.; Yaguachi, L.; Samanta, C.C.; Danilo, J.H.; Soto, F. Performance evaluation in the migration process from a monolithic application to microservices. In Proceedings of the IEEE 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), Caceres, Spain, 13–16 June 2018; pp. 1–8.

[5] Akbulut, A.; Perros, H.G. Performance Analysis of Microservices Design Patterns. IEEE Internet Comput. 2019, 23, 19–27. [CrossRef]

[6] Singh, V.; Peddoju, S.K. Container-based microservice architecture for cloud applications. In Proceedings of the IEEE 2017 International

Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 5–6 May 2017; pp. 847–852.

[7] Ponce, F.; Márquez, G.; Astudillo, H. Migrating from monolithic architecture to microservices: A Rapid Review. In Proceedings of the 2019 IEEE 38th International Conference of the Chilean Computer Science Society (SCCC), Concepcion, Chile, 4–9 November 2019; pp. 1–7.

[8] Taibi, D.; Lenarduzzi, V.; Pahl, C. Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. IEEE Cloud Comput. 2017, 4, 22–32. [CrossRef]

[9] Mazlami, G.; Cito, J.; Leitner, P. Extraction of microserv ices from monolithic software architectures. In Proceedings of the 2017 IEEE International Conference on Web Services (ICWS), Honolulu, HI, USA, 25–30 June 2017; pp. 524–531.

[10] Kalske, M.; Mäkitalo, N.; Mikkonen, T. Challenges when moving from monolith to microservice architecture. In International Conference on Web Engineering; Springer: Cham, Switzerland, 2017; pp. 32–47

[11] Bures, T.; Duchien, L.; Inverardi, P. (Eds.) Software Architecture—Proceedings of the 13th European Conference, ECSA 2019, Paris, France, 9–13 September 2019; Springer Nature: Cham, Switzerland, 2019; Volume 11681.

[12] Sarkar, S.; Vashi, G.; Abdulla, P.P. Towards Transforming an Industrial Automation System from Monolithic to Microservices. In Proceedings of the 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Turin, Italy, 4–7 September 2018; pp. 1256– 1259.

[13] Debroy, V.; Miller, S. Overcoming Challenges with Continuous Integration and Deployment Pipelines When Moving from Monolithic Apps to Microservices: An experience report from a small company. IEEE Softw. 2019, 37, 21–29. [CrossRef]

[14] Kratzke, N. About Microservices. Contain. Their Underestim. Impact 2015, 961, 165–169.