



INTRODUCTION

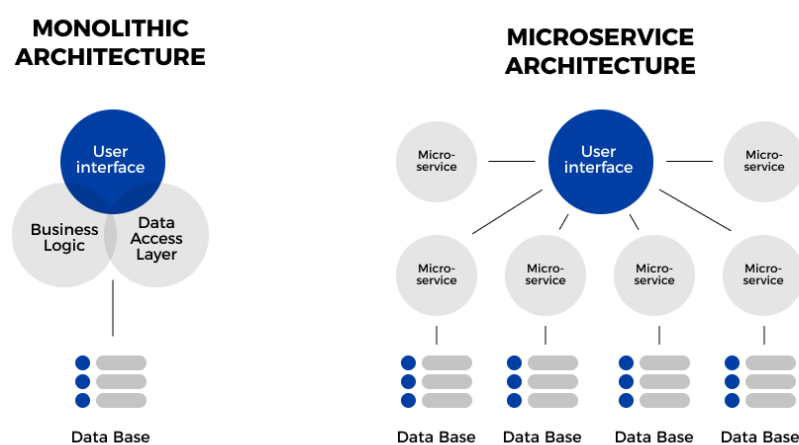


Figure 1: Monolith vs Microservices Architecture

Monolithic vs. Microservices: A Performance Comparison

The Problem: Choosing the right architecture is difficult. Monolithic is simple but hard to scale. Microservices are scalable but complex.

Our Goal: We tested both under high concurrency to see which performs better.

What We Measured: Response time, throughput, error rate, and resource use.

Why It Matters: Our results provide data-driven guidance for this critical design decision.

EXPERIMENT AND METHODOLOGY

Experimental Setup

Hardware: MacBook Pro (M2 Pro, 10-core, 16 GB RAM)

Application: Spring Pet Clinic Management System

Deployment:

- **Monolith:** Single Docker container
- **Microservices:** Multiple containers via Docker Compose

Test Scenarios

Workload: Simulated common user tasks (owner searches, new pet submissions, etc.).

- **Concurrency:** Scaled from 5 to 100 virtual users.
- **Duration:** Each test ran for 3 minutes to observe sustained performance.

Performance Metrics

- **Data Collection:** Results averaged over 5 runs per concurrency level.

Key Metrics:

Total Requests
Average Response Time (Latency)
Throughput (Transactions/sec)
Error Rate (%)
CPU & Memory Usage

RESULTS

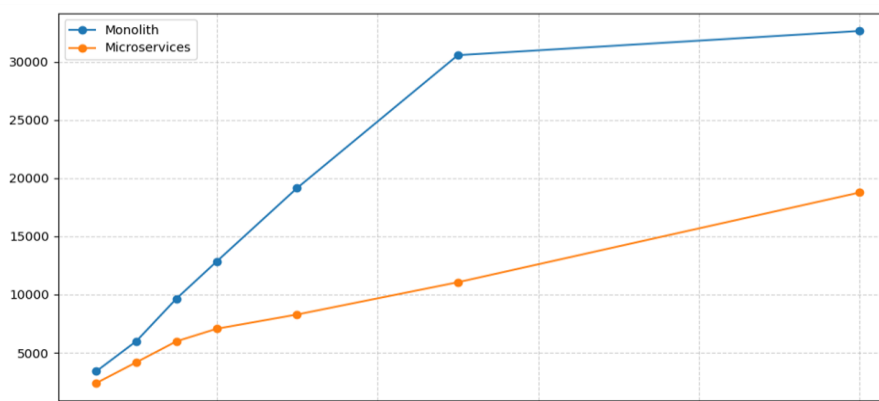


Figure 2: Total number of requests handled by each architecture under increasing load.

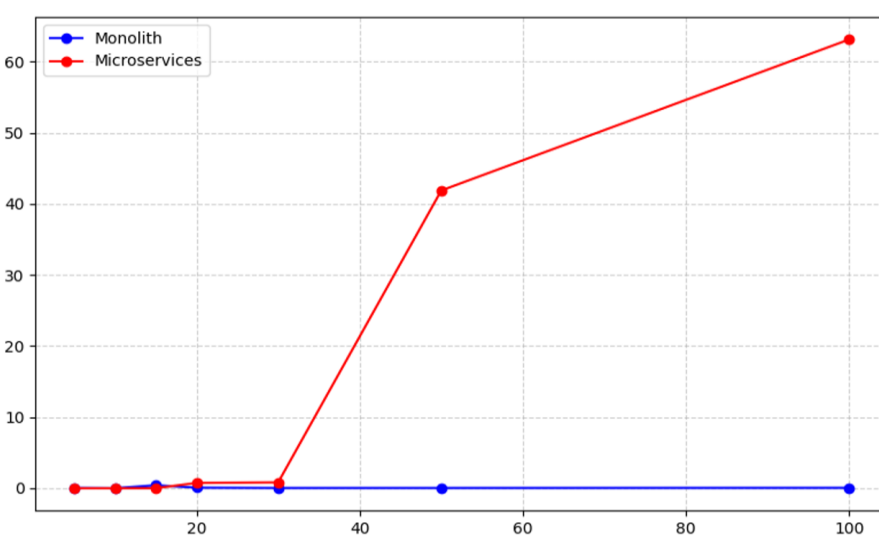


Figure 3: Average response time comparison between both architectures under increasing load.

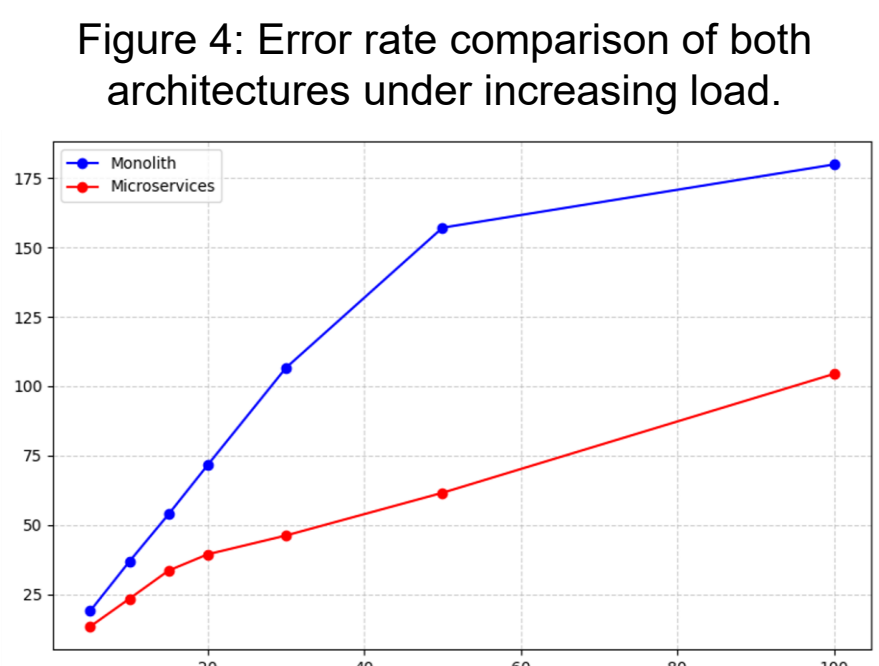


Figure 4: Error rate comparison of both architectures under increasing load.

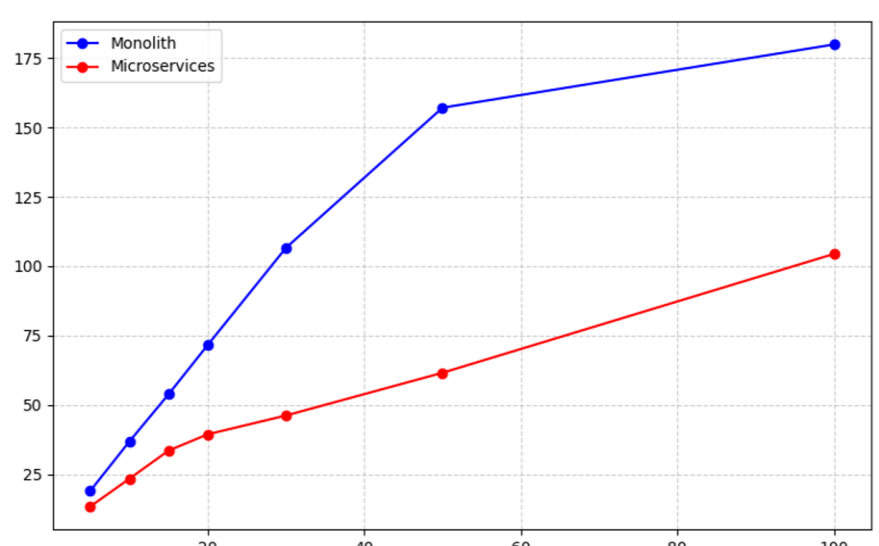


Figure 5: CPU Usage Comparison of both architectures under increasing load.

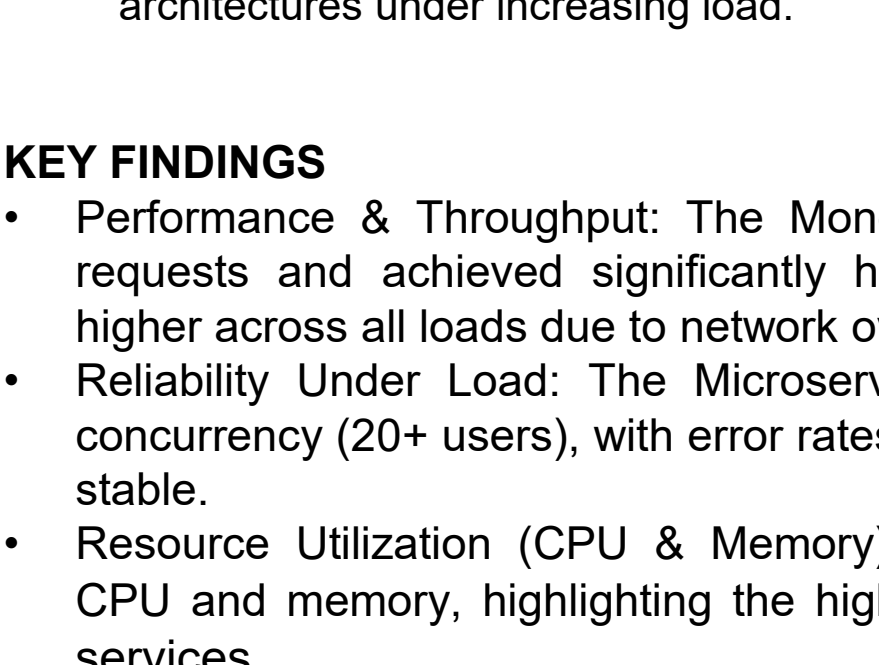


Figure 6: Throughput comparison of both architectures under increasing load.

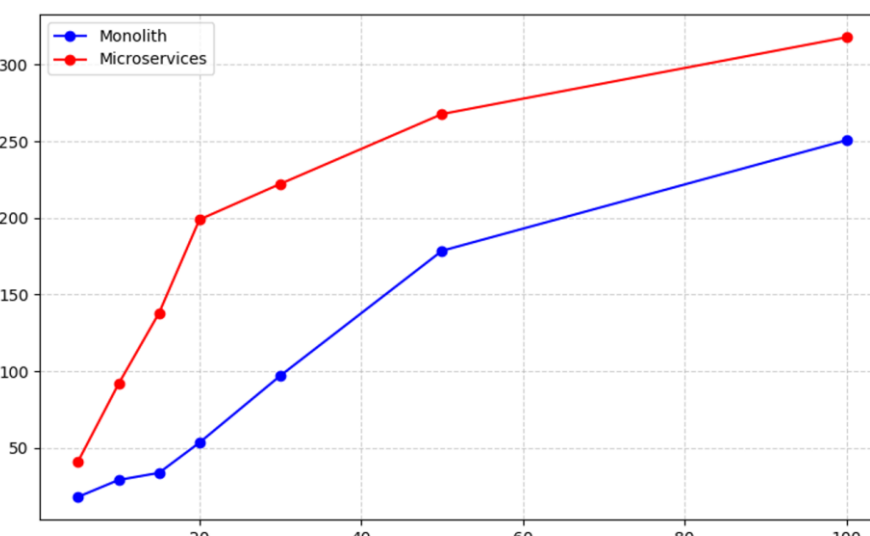


Figure 7: Memory Usage Comparison of both architectures under increasing load.

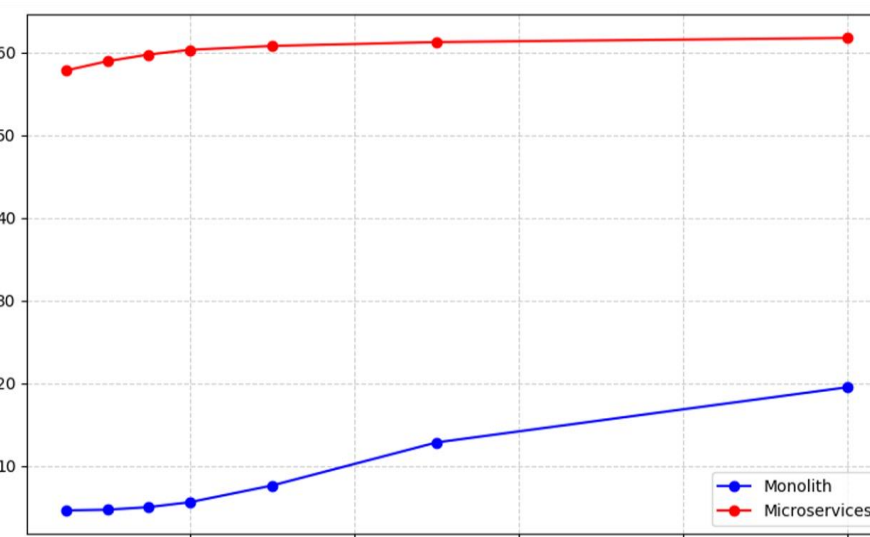


Figure 8: Memory Usage Comparison of both architectures under increasing load.

KEY FINDINGS

- **Performance & Throughput:** The Monolith architecture consistently handled more requests and achieved significantly higher throughput. Microservice latency was higher across all loads due to network overhead.
- **Reliability Under Load:** The Microservices architecture became unstable at high concurrency (20+ users), with error rates spiking to over 60%. The Monolith remained stable.
- **Resource Utilization (CPU & Memory):** Microservices required substantially more CPU and memory, highlighting the higher operational cost of managing distributed services.

CONCLUSION AND FUTURE DIRECTIONS

- Monolith outperformed microservices in requests handled, throughput, and response time.
- Microservices used more CPU/memory, were stable at low load, but failed more often under high load.
- Monolith proved more resilient and efficient on limited hardware.
- Integrate Event-Driven Architecture (Kafka) into both architectures and compare synchronous vs. event-driven throughput under varying load.

ACKNOWLEDGEMENTS

- Thanks to the FUTURES Act program at Lincoln University for research support and resources.
- I would also like to thank Dr Bouchaib Falah and Dr. Olamide Tawose for their mentorship and support.

REFERENCES

- [1]Cabane, H., & Farias, K. (2023). On the impact of event-driven architecture on performance: An exploratory study. *Future Generation Computer Systems*, 153, 52–69. <https://doi.org/10.1016/j.future.2023.10.021>
- [2]Lazzari, L., & Farias, K. (2021). An exploratory study on the effects of event-driven architecture on software modularity. *arXiv* (Cornell University). <https://doi.org/10.48550/arxiv.2110.14699>
- [3]Przemysław Jatkiewicz and Szymon Okrój. (2023). Differences in performance, scalability, and cost of using microservice and monolithic architecture. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23)*. Association for Computing Machinery, New York, NY, USA, 1038–1041. <https://doi.org/10.1145/3555776.3578725>
- [4]Spring (2025) spring-petclinic. <https://github.com/spring-projects/spring-petclinic>