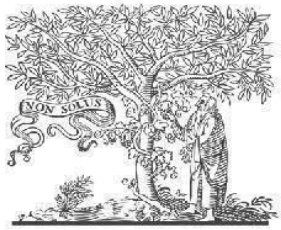
International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

COPY RIGHT



ELSEVIER SSRN

IJIEMR Transactions, online available on 30th Oct 2024. Link

https://ijiemr.org/downloads.php?vol=Volume-13&issue= Issue10

DOI:10.48047/IJIEMR/V13/ISSUE10/62

Title: " LEVERAGING LOGGING AND MONITORING TOOLS FOR PROACTIVE ISSUE DETECTION IN MICRO-SERVICES"

Volume 13, ISSUE 10, Pages: 495- 504

Paper Authors
Venkat Marella

USE THIS BARCODE TO ACCESS YOUR ONLINE PAPER

To Secure Your Paper as Per UGC Guidelines We Are Providing A ElectronicBar code

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

LEVERAGING LOGGING AND MONITORING TOOLS FOR PROACTIVE ISSUE DETECTION IN MICRO-SERVICES

Venkat Marella

Independent Researcher, USA.

Abstract

Performance uncertainty is a major deterrent to cloud adoption, with consequences for cost, revenue, and performance. Predictable performance becomes increasingly more crucial when cloud services transition from monolithic architectures to microservices. In microservices systems, discovering QoS breaches after they occur results in long recovery periods since spots propagate and amplify across dependent services. However, their usage in industrial datasets has been less frequent. Additionally, the logging statements in the open-source datasets being studied are often rather large and do not change much over time. For a brand-new dataset from an industrial service, this may not be the case. This study tests many state-of-the-art anomaly detection techniques on the industrial dataset from the project partner, which is much smaller and less structured than most large-scale open-source benchmark datasets. Therefore, microservices provide heterogeneity, scalability, agility, and a fair degree of fault tolerance by breaking up modular programs into several services. However, there are a number of issues and challenges with this architectural paradigm that businesses must deal with. The purpose of this article is to outline the primary advantages and possible drawbacks of the microservices architecture and to propose an intelligent microservices structure that leverages AI and ML to support automation, flexibility, and optimization. This is to ensure that there is a rationale for why a certain activity should be performed and how it may achieve the process's desired goals.

Keywords: - Performance, automation, and optimization, Detecting QoS, leverages AI, scalability, open-source, cost, fault tolerance, benchmark dataset, machine learning.

I. INTRODUCTION

Cloud computing services are governed by strict quality of service (QoS) constraints in terms of throughput, and more critically tail latency. Breaking these rules has serious financial repercussions, degrades the end experience for consumers, and results in decreased availability and dependability [1]. Cloud services have recently seen a significant change from complex monolithic designs, which contain all functionality in a single binary data towards graphs of hundreds of loosely-coupled, single-concerned microservices in an attempt to meet these performance constraints and enable frequent application updates [1]. Because each microservice is written in the programming language or framework that best fits its needs, microservices are attractive for a number of reasons, including speeding up development and deployment, making correctness debugging easier because errors can be isolated in specific tiers, and enabling a rich software ecosystem [1, 2]. However, microservices present a number of system issues and mark a fundamental shift from the architecture of typical cloud systems.

In particular, the tail latency needed for each individual microservice is far more stringent than for conventional cloud applications, despite the fact that the end-to-end application's quality-of-service (QoS) requirements are comparable for microservices and monoliths [3]. Because microservices are dependent on one another, a single misbehaving microservice might result in cascade QoS breaches across the system, increasing the need to guarantee predictable performance [3, 4].

Cyber-Physical Systems (CPS) driven by Internet of Things (IoT) devices are being used more and more for a variety of purposes as Industry 4.0 develops, including environmental condition detection, maintenance monitoring, and security enhancement [4]. In order to detect failure circumstances and abnormalities, these IoT devices often gather vast volumes of multi-variate time-series data, which may be analysed offline or in real-time. However, the sheer amount of data makes manual analysis almost impossible, and manual analysis of large data is time-consuming and needs specialist expertise [4, 5]. We have log files that are specified and organized for a specific device. However, log files for heterogeneous networked devices lack structure [P4]. Unstructured log files don't have a set structure and may have different amounts of formatting and information than structural log files, which have a pre-set format [5].

Large-scale distributed systems often employ unstructured log files, which are essential for tracking system health, identifying problems, and spotting irregularities. In the manufacturing or production industries, unstructured log files are analysed by combining formal product information with domain expert knowledge [4, 5]. There are several methods for analysing log files. The multivalent system that was suggested in a research is fault-tolerant. It tackles the issue of preserving system operation in the face of disruptions or malfunctions, which is a crucial component of fault tolerance. Another research produced a more thorough control rule for multivalent systems. Through self-supervised training procedures, these methods may assist IoT devices in identifying abnormalities; nevertheless, they often lack the ability to fully explain the importance and underlying reasons of the anomalies they identify.

The management of cloud resources is a conventional subject with a wide range of associated activities. Monitoring and auto scaling are features offered by major cloud providers including Google, Microsoft, Amazon, and others. The EC2 Auto Scaling (EC2-AS) service is provided by AWS.4. For autonomously managing cloud resources, EC2-AS performs well [5]. Amazon Elastic Container Service (ECS), which has limitations with regard to EC2-AS, is the recommended solution for using services published in containers; in this instance, AWS Fargate is the best choice [4, 6].

Amazon Elastic Containers Service (ECS) and Amazon's Elastic Kubernetes Service (EKS) are compatible with Fargate, a server-less computing engine for containers [6]. However, our approach seeks to proactively minimize resource usage while preserving application service performance, while Fargate seeks to optimize the cost of using cloud resources. Based on workload requirements, Google Kubernetes Engine's (GKE) Cluster Autoscaler controls the number of nodes in a node pool. The cluster autoscaler scales down to a minimal size intended to limit expenses [6, 7] if the demand is low [6]. Furthermore, Google offers the Autopilot option, which is an autoscaler that increases threshold setting accuracy and optionally provides

a complex algorithm based on reinforcement learning that has enabled notable incremental benefits.

The idea behind microservices is to divide an application into several readily modifiable and controllable tiny services. HTTP or message queuing protocols are used by the interfaces of each service, which is dedicated to a single business feature [8]. This architectural design enables CI/CD and is in line with contemporary development methodologies like Agile and DevOps. The scale, resilience, handling of information, testing, and deployment of microservices architectures are all problematic. In addition to the DevOps technique, there could be problems with services communication, service registration and discoveries, and general management of connections between services that are difficult to track and record [8, 9].

Furthermore, because each standard service often contains a database by default, managing scattered data and the absence of uniform data across different instances of the standard service may be complex. More and more businesses are considering using AI and ML in order to get beyond the aforementioned obstacles and fully use microservices [9, 10]. AI/ML can effectively complement microservices architecture by offering decision support, automation, and improvement. For example, supply chain data and real-time service performance are input into AI/ML algorithms in fraction intelligent service delivery, which then reinvents and optimizes the service to load and scale resources accordingly. The best utilization of resources, increased system dependability, and application performance improvement might result from this ongoing learning and adaptation [7, 8].

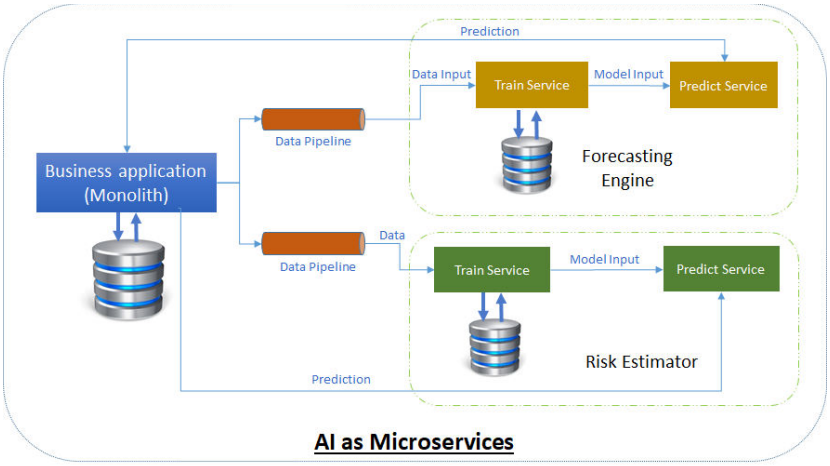


Fig. 1 Flowchart demonstrating the advantages of integrating AI/ML with microservices architecture. [8, 9]

II. BENEFITS OF MICROSERVICES

2.1 Architecture Improved Scalability

Scalability is an additional advantage of using a microservices architecture [9, 10]. Depending on the needs of a component, resources may be allocated more efficiently since each service is separated and can be scaled out by generating additional instances [11]. Additionally, it allows

companies to reduce superfluous squad expenditures by expanding the number of particular services according on utilization rates.

2.2 Increased Agility and Flexibility

Microservices design contributes to a development process's increased agility and flexibility. It is possible to refactor any service that is included [11, 12]. Since they are distinct parts, it is possible to implement a new feature or update in one without having an impact on the program as a whole. Development cycles and the capacity to adjust to business changes are accelerated as a result.

2.3 Fault Isolation

Because each service in a microservices design functions separately, the system as a whole is unaffected when one fails [14]. Better system dependability results from this kind of fault isolation, and developers find it easier to fix issues when they are aware of the faulty service [10].

2.4 Technology Diversity

Utilizing various technologies and languages of programming is another benefit of microservices architecture [1, 6]. To increase productivity and merge outdated systems, developers may choose the best framework and tools for each service. This technology group optimizes performance and development by giving businesses the finest technology for a given job [9].

2.5 Enhanced Deployment and Maintenance

Microservices facilitate updates and maintenance operations by enabling autonomous deployment and administration. Services may be launched or terminated individually and don't need the whole program to be stopped and resumed because of its modularity [8]. This independence helped the HTML process since it allowed for more frequent updates that were less dangerous.

2.6 Improved Target, Goal, and Need Co-Ordination

Because it makes it possible to create services that represent distinct business skills, micro-services architectures is more strategic for businesses [8, 9]. Additionally, this alignment improves technical teams' understanding of and ability to react to business needs.

2.7 Facilitated Continuous Improvement

Microservices' loose coupling encourages continuous improvement techniques since it allows for incremental enhancements. Teams may continuously improve and modify their specific services as needed without waiting for a full system roll-out cycle since services are created and maintained independently.

2.8 Enhanced Security and Compliance

Because each service is independent and inter-service communication is tightly regulated, microservices design may enhance security and compliance. Depending on its unique requirements, a service may offer its permission and authentication [9, 10]. Because of this isolation, the system is less accessible, which prevents the breach from spreading swiftly [11].

III. AI/ML-BASED SERVICE-MANAGEMENT SOLUTIONS

3.1 AI/ML-Powered Service Orchestration and Management

The AI/ML-based service orchestrating and management system that is part of the suggested solution is another important consideration. It's important to emphasize once again that this system would be based on artificial intelligence and machine learning, which would explain how the relational database of service information about performance would analyse current patterns in real time before making logical decisions about load allocation and, if required, load balancing [11]. The efficacy and efficiency of the application may also be enhanced by this system, which may also optimize the resources that are available and the system's resilience [13]. The system may pre-allocate extra instances to address anticipated surges in consumption and schedules the additional instances according to a specific web application's usage pattern. However, when demand is low, it might cut down on resources. The orchestration system can also detect and redistribute to maintain optimal efficiency, and load balance services so that no service is overwhelmed [15].

3.2 Intelligent monitoring and anomaly detection

The e-monitoring and identification of anomalies system, which need to be integrated with an intelligent system, is another essential element [16]. To find early indicators of issues, inefficiencies, and possible failures, this system would sift logs, metrics, and other information about monitoring from various services using machine learning techniques.

3.3 AI-Driven Testing and Debugging

Incorporating AI-based testing and debugging capabilities into the complex is another aspect of the suggested design [12]. To anticipate issues, create more effective tests, and concentrate testing efforts in the best places, machine learning models may be developed using test case outcomes and historical data.

3.4 ML-Based Personalization and Recommendation

Information and suggestions for users, which may be provided with the use of microservices [16], are bundled as features in various application situations. Targeted content, goods, or services may be created by analysing user behaviour, preferences, and context via the integration of machine learning models into the microservices architecture [17]. E-commerce platforms, for example, may use machine learning (ML) models to anticipate which goods a customer would be interested in based on their past purchases, online searches, and similar consumer behaviour patterns [18].

IV. CHALLENGES OF MICROSERVICES

4.1 Architecture Increased Complexity

Because microservices design is a distributed system, it has also added complexity. It may be difficult to handle SVS interactions, recognize and handle requests, orchestrate, log, monitor, and implement good DevOps practices [19]. Microservices thus need specific protocols, usually REST, gRPC APIs, or message brokers like Rabbit MQ or Kafka, while monolithic programs include components that are directly dependent on one another [15, 17].

4.2 Data Management and Consistency

When the data has to be synced to maintain integrity across many services, it becomes difficult. Separate records are necessary for a single service, which encourages independence and expandability while creating problems with data organization [16]. Among the things developers must do to enhance the services are methods for preserving data consistency and handling cross-service interactions [18].

4.3 Testing and Debugging Difficulties

Compared to monolithic apps, microservices-based solutions may be more difficult to test and debug. To trace issues and their causes across many services, it is essential to take into account a variety of integration patterns and use distributed investigation and logging correlations [19]. Individual service testing is still straightforward, but initiative testing involves simulating events involving the services in question, scenarios that may be created and maintained by many teams [19, 20]. Changes to one service might improve a system's overall performance since integration testing must be carried out on automated testing platforms that must be extended to include additional end-to-end solutions [20].

4.4 Deployment and Versioning Challenges

Managing several deployments from diverse sources and addressing the problem of disparate versions across different services, which leads to compatibility issues, are difficult tasks [20, 21]. To avoid service delays, these modifications must be planned and automated. In CI/CD pipelines, microservices need two additional configurations: controlling inter-service communication and addressing dependencies.

4.5 Service Discovery and Management

Another crucial element that raises the difficulty levels in a microservices design is service discovery. Finding and managing appropriate instances becomes essential to attaining reliability in interactions as the number of services increases. By maintaining an up-to-date list of available services and their locations, service discovery tools like Consul, EUREKA, [20,21], or a Kubernetes service discovery component help.

V. FUTURE DIRECTIONS AND TRENDS

5.1 Advanced AI/ML Techniques

Thanks to contemporary AI/ML technologies, there are even additional options to improve microservices design as the practice develops [21]. Higher degrees of intelligence are now being seen, along with the optimization of microservices and associated technologies like as natural language processing (NLP), deep learning, and reinforcement learning.

- **Deep Learning:** Neural networks known as deep learning models may be integrated into microservices to perform intricate tasks like prediction, emotion recognition, and image processing [22]. In order to help microservices create better trends, they analyse data to identify patterns or correlations within a massive data collection.
- **Reinforcement Learning:** Microservices can function even in situations with large-scale flows, and decision-making based on continuously changing data can be done using reinforcement learning. In this context, [21], reinforcement learning enables microservices to modify their actions in response to input from the system's environment system depending on the outcomes of several tests.
- **Natural Language Processing (NLP):** When NLP is used to support microservices, the applications get natural language processing capabilities that enable them to understand human speech [21]. Because microservices may scan text, provide some responses, or even classify the material based on context, this is particularly fantastic for chatbots, customer interaction-based apps, and content analysis [23, 24].

5.2 Edge Computing and IoT

IoT-enabled edge computing is expected to be a breakthrough method for expanding the capabilities of more intricate microservices [25]. Processing near the network's edge lowers latency, which is a basic need for latency-low applications. This is known as edge computing.

- **Real-time Processing:** Among them are edge-based localized microservices that operate on streaming feeds in the spirit of adaptive streaming, which allows choices to be taken in milliseconds or even nanoseconds without requiring instructions to be sent back to the cloud. This feature is essential for industries where low latency is advantageous, such manufacturing, self-driving cars, and careful city/urban planning [21].
- **Scalability and Efficiency:** In order to guarantee scalability and lessen the strain on a central server room, edge computing for microservices facilitates compute on tiny devices at the periphery. Additionally, it offers dispersal, which improves speed, makes the system friendlier, and boosts dependability since there won't be a single point of failure.
- **Integration with IoT Devices:** IoT smart devices use a range of sensors to generate a substantial volume of data that microservices may locally handle. For example, wearable technology may monitor patients' health condition in real time in the health sector [23, 23]. Microservices quickly analyse this data to provide medical advice or alerts at the same time.

5.3 AI-Enhanced DevOps Utilizing

Microservices development and deployment benefit from the use of AI in DevOps processes, sometimes known as AI-DevOps or MLOps.

- **Automated Code Reviews:** Before the program is released to the market, intelligent automation technologies may analyze the code and identify problems or perhaps incorrect sections.

- **Predictive Maintenance:** AI systems can detect the likelihood of microservice failures or degradations by learning from telemetry data [25]. This proactive approach allows teams to schedule maintenance tasks in advance to minimize system downtime and improve system availability, even if the definition of preventive and corrective maintenance may have evolved in recent years.
- **Intelligent CI/CD Pipelines:** AI may assist in analyzing dependencies, choosing the optimal Continuous Integration/Continuous Deployment techniques, and determining how changes affect performance [21, 23] [24].

VI. CONCLUSION

The foundation for a Kubernetes-based cloud was presented in this post, with an emphasis on automating scaling processes, gathering detailed analytics about agnostic services with various needs, and employing a forecasting module to anticipate demand spikes. In order to maintain the intended quality of service (QoS) while addressing the effective use of resources during periods of low consumption, the monitoring and administration system was expanded to incorporate auto scaling and a load prediction service.

When creating intricate, adaptable, and scalable software systems, the microservices architecture offers several benefits. But it also brings with it additional difficulties in handling data, complexity, testing, and deployment. As was previously said, using microservices in large-scale software systems has some difficulties that may be resolved by enhancing the microservices environment with AI and ML. Prominent real-world examples bolster this analysis and demonstrate the advantages of this approach. The intelligent microservices architecture proposed in this paper suggests using AI/ML for microservices' orchestration, with monitoring, evaluation, debugging, and personalization for intelligent decision-making and automation. When creating intelligent, flexible, and high-performing apps, microservices architecture will eventually include AI and ML even more.

VII. REFERENCES

[1] L. Chen, Y. Xu, Z. Lu, J. Wu, K. Gai, P.C.K. Hung, M. Qiu, IoT microservice deployment in edge-cloud hybrid environment using reinforcement learning, IEEE Internet Things J. 8 (16) (2021) 12610–12622.

[2] H. Zhao, H. Lim, M. Hanif, C. Lee, Predictive container auto-scaling for cloud-native applications, in: 2019 International Conference on Information and Communication Technology Convergence, ICTC, 2019, pp. 1280–1282.

[3] S. Zhou, J. Li, K. Zhang, M. Wen, Q. Guan, An accurate ensemble forecasting approach for highly dynamic cloud workload with VMD and R-transformer, IEEE Access 8 (2020) 115992–116003.

[4] B. Liu, J. Guo, C. Li, Y. Luo, Workload forecasting based elastic resource management in edge cloud, Comput. Ind. Eng. 139 (2020) 106136.

[5] Y. Zhu, W. Zhang, Y. Chen, H. Gao, A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment, EURASIP J. Wireless Commun. Networking 2019 (1) (2019).

[6] B. Thurgood, R.G. Lennon, Cloud computing with kubernetes cluster elastic scaling, in: Proceedings of the 3rd International Conference on Future Networks and Distributed Systems, ICFNDS '19, Association for Computing Machinery, New York, NY, USA, 2019.

[7] A.A. Khaleq, I. Ra, Intelligent autoscaling of microservices in the cloud for real-time applications, IEEE Access 9 (2021) 35464–35476.

[8] G. Yu, P. Chen, Z. Zheng, Microscaler: Cost-effective scaling for microservice applications in the cloud with an online learning approach, IEEE Trans. Cloud Comput. 10 (2) (2022) 1100–1116.

[9] M.-N. Tran, D.-D. Vu, Y. Kim, A survey of autoscaling in kubernetes, in: 2022 Thirteenth International Conference on Ubiquitous and Future Networks, ICUFN, 2022, pp. 263–265.

[10] D. Ferreira, C. Senna, S. Sargento, Distributed real-time forecasting framework for IoT network and service management, in: NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, IEEE Press, 2020, pp. 1–4.

[11] J.-G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, "Mining invariants from console logs for system problem detection," 01 2010.

[12] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Online system problem detection by mining patterns of console logs," in 2009 Ninth IEEE International Conference on Data Mining, 2009, pp. 588–597.

[13] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in Proceedings of the 38th International Conference on Software Engineering Companion, ser. ICSE '16, 2016, p. 102–111.

[14] S. He, Q. Lin, J.-G. Lou, H. Zhang, M. R. Lyu, and D. Zhang, "Identifying impactful service system problems via log analysis," ser. ESEC/FSE 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 60–70.

[15] Christina Delimitrou, Nick Bambos, and Christos Kozyrakis. [n. d.]. QoS-Aware Admission Control in Heterogeneous Datacenters. In Proceedings of the International Conference of Autonomic Computing (ICAC). San Jose, CA, USA, 2013.

[16] Christina Delimitrou and Christos Kozyrakis. [n. d.]. iBench: Quantifying Interference for Datacenter Workloads. In Proceedings of the 2013 IEEE International Symposium on Workload Characterization (IISWC). Portland, OR, September 2013.

[17] Alves, J. M., Honório, L. M., & Capretz, M. A. (2019). ML4IoT: A framework to orchestrate machine learning workflows on Internet of things data. IEEE Access, 7, 152953-152967.

[18] Macpherson, J. D., de Wardt, J. P., Florence, F., Chapman, C. D., Zamora, M., Laing, M. L., & Iversen, F. P. (2013). Drilling-systems automation: Current state, initiatives, and potential impact. SPE drilling & completion, 28(04), 296-308.

[19] Moreno-Vozmediano, R., Montero, R. S., Huedo, E., & Llorente, I. M. (2019). Efficient resource provisioning for elastic cloud services based on machine learning techniques. Journal of Cloud Computing, 8(1), 1-18.

[20] Nadareishvili, I., Mitra, R., McLarty, M., & Amundsen, M. (2016). Microservice architecture: aligning principles, practices, and culture. "O'Reilly Media, Inc.".

[21] Ndlovu, T., & Mariussen, A. (2015). From competitive agility to competitive leapfrogging: responding to the fast pace of change. In Handbook of Research on Global Competitive Advantage through Innovation and Entrepreneurship (pp. 1-12). IGI Global.

[22] O'Connor, R. V., Elger, P., & Clarke, P. M. (2017). Continuous software engineering—A microservices architecture perspective. Journal of Software: Evolution and Process, 29(11), e1866.

[23] Pianykh, O. S., Langs, G., Dewey, M., Enzmann, D. R., Herold, C. J., Schoenberg, S. O., & Brink, J. A. (2020). Continuous learning AI in radiology: implementation principles and early applications. Radiology, 297(1), 6-14.

[24] Christina Delimitrou and Christos Kozyrakis. [n. d.]. Qualityof-Service-Aware Scheduling in Heterogeneous Datacenters with Paragon. In IEEE Micro Special Issue on Top Picks from the Computer Architecture Conferences. May/June 2014.

[25] Christina Delimitrou and Christos Kozyrakis. [n. d.]. Quasar: ResourceEfficient and QoS-Aware Cluster Management. In Proc. of ASPLOS. Salt Lake City, 2014.