2023 World Conference on Communication & Computing (WCONF)
Raipur, India. July 14-16, 2023

A Review of Cloud Microservices Architecture for Modern Applications

Gunjan Pathak
Apex Institute of Technology
Chandigarh University
Punjab, India
g.pathak1812@gmail.com

Dr. Monika Singh
Apex Institute of Technology
Chandigarh University
Punjab, India
monika.e11032@cumail.in

2023 World Conference on Communication & Computing (WCONF) | 979-8-3503-1120-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/WCONF58270.2023.10235199

***Abstract*— As a potent architectural paradigm for creating and deploying contemporary applications in the cloud computing environment, cloud microservices have been developed. This review paper offers a thorough examination of the ideas, advantages, problems, and developments related to cloud microservices. We start by outlining the core ideas and traits of the microservices architecture and highlighting its benefits over more conventional monolithic ones. The advantages of using cloud microservices, such as scalability, robustness, and flexibility, are then discussed, along with the difficulties in implementing and managing them. The article compares the features and constraints of several cloud platforms and tools for deploying and orchestrating microservices, such as AWS, Azure, Google Cloud Platform, and Kubernetes. Real-world case studies highlight effective cloud microservices implementations and offer helpful insights and best practices. We also examine current trends in the field and recent research papers to determine where future research should be focused. Additionally, cloud microservice-specific security and governance issues are covered, with an emphasis on methods for encrypting communication, controlling access, and guaranteeing data privacy. This review paper seeks to increase the comprehension of this revolutionary architectural paradigm by providing a thorough examination of cloud microservices for modern applications.**

***Keywords*— cloud, micro-services, cloud platforms, security**

I. Introduction

Modern application development has undergone an enormous shift, thanks to cloud computing [1] which provides scalable and adaptable infrastructure to meet changing business needs. In this setting, microservices [2] architecture has become a potent method for creating and delivering cloud-based systems. Microservices designs, as opposed to conventional monolithic architectures, break down applications into smaller, independent services that interact with one another across a network.

This review paper's goal is to offer a thorough examination of cloud microservices for contemporary applications. We start by outlining the basic tenets and traits of the microservices architecture.

Microservices [3] are independent, loosely connected services that concentrate on particular business functions. Agile and modular application development is made possible by the independent development, deployment, and scaling of each microservice.

Microservices in the cloud provide a number of advantages that are crucial for contemporary applications. They may be scaled horizontally to manage a variety of workloads, which is a major benefit. This guarantees effective resource management and the capacity to adjust to shifting demands. Another critical component is resilience, which increases dependability by ensuring that failures in one microservice do not affect the entire programme. Additionally, the adaptability of microservices enables businesses to select the finest frameworks and technologies for certain services, encouraging innovation and enabling the deployment of best-of-breed solutions.

The use of cloud microservices, however, is not without its difficulties. Among the most important challenges to overcome are managing the complexity of deployment and orchestration, guaranteeing data consistency across services, and effective inter-service communication. Monitoring the general well-being and effectiveness of microservices also becomes essential for preserving system dependability.

Different cloud-native platforms and solutions have evolved to solve these problems. Leading cloud providers like AWS, Azure, and Google Cloud Platform provide specialised services and tools for setting up and maintaining microservices. Microservice packaging and isolation are made possible by containerization technologies like Docker, while orchestration frameworks like Kubernetes give us the tools to effectively manage the scalability and lifetime of these services.

This paper is intended to investigate the advantages, difficulties, and trends related to cloud microservices throughout.

II. Cloud Microservices Architecture

A software architectural strategy called cloud microservices architecture places an emphasis on the creation and deployment of applications as a group of tiny, autonomous services, each of which is in charge of a particular business feature. These services are made to be loosely connected and communicate with one another through a network, usually using messaging systems or standardised protocols like HTTP.

The breakdown of a monolithic programme into a collection of connected and independent services is the fundamental tenet of cloud microservices architecture. Microservices architecture encourages modularity and the separation of responsibilities in contrast to monolithic systems, which consolidate all functions into a single codebase. Each microservice stands for a particular business capacity or functionality, for example order fulfilment, payment processing, user authentication, or product cataloguing.

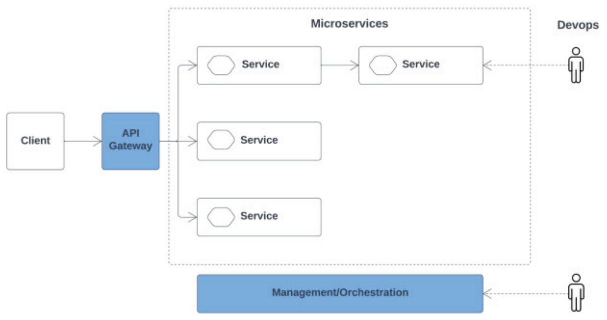

Fig. 1. Microservices architecture.

Due to the independent creation, deployment, and scaling of these microservices, organisations can increase the flexibility and agility of their application development process. Depending on the particular needs of each service, development teams can work on several microservices at once while utilising a variety of technologies and programming languages. Teams can adopt the finest frameworks and tools for each microservice because to this decoupling, which boosts creativity and development efficiency.

The capacity of cloud microservices architecture to extend horizontally is one of its main benefits. Even if only one particular feature or service was seeing greater demand, growing the entire application under conventional monolithic architectures [4] would necessitate scaling all of its components. Individual services can scale independently with microservices based on their own usage patterns and resource needs. Organisations may optimise resource allocation and guarantee effective use of computing resources in the cloud thanks to this granular scalability.

The cloud microservices design encourages robustness and fault separation in addition to scalability. Failures or difficulties in one microservice do not cascade to other services since microservices are independently deployable and communicate through clear interfaces. This isolation reduces the effects of failures and raises the overall reliability of the system. The rest of the application can continue to run without interruption if one microservice fails, minimising downtime and enhancing user experience.

An essential component of cloud microservices architecture is service discovery and communication. Effective discovery and communication between services are required. Microservices can find one other and connect to each other dynamically with the use of service discovery techniques, such as service registries or DNS-based methods. Asynchronous messaging platforms like RabbitMQ or Apache Kafka or synchronous protocols like HTTP/REST are generally used to communicate between services. The interface between services is facilitated by well-defined APIs and contracts, allowing for loose coupling and independent evolution and upgrades.

Although the cloud microservices architecture has many advantages, it also has problems that businesses must solve. Some of the issues that come with cloud microservices architecture include managing the complexity of inter-service communication, guaranteeing data consistency in distributed environments, managing service orchestration and choreography, and monitoring the health and performance of several services.

Organisations use cloud-native technologies and platforms to address these issues. Microservices are packaged and deployed using lightweight, portable containerization tools like Docker. The management of containers is automated by orchestration frameworks like Kubernetes, providing functions like service scalability, load balancing, and rolling deployments. The infrastructure and tools required by these technologies and platforms make it easier to deploy and orchestrate microservices in the cloud.

Cloud microservices architecture is a paradigm that encourages the creation and distribution of applications as a group of tiny, independent services. Modularity, scalability, resilience, and adaptability are just a few advantages it has. Communication, data consistency, orchestration, and monitoring issues are also presented. Organisations may successfully deal with these issues and realise the full potential of cloud microservices architecture in contemporary application development by utilising cloud-native technologies and platforms.

III. Benefits of Microservices Architecture

1. **Scalability:** The horizontal scalability [4] provided by the microservices architecture allows for the independent scaling of individual services in accordance with their unique demand patterns. This adaptability guarantees efficient resource allocation, smart management of shifting workloads, and the capacity to scale only the essential components, improving performance and reducing costs.

2. **Flexibility and agility**: Microservices encourage application development's agility. Because each service can be created, tested, and deployed individually, release cycles are shortened and it is possible to react swiftly to shifting business needs. Additionally, separate teams can work on various services simultaneously while utilising the best frameworks and technologies for each service, encouraging innovation.

3. **Fault isolation and resilience**: Microservices architecture isolates services and uses well defined interfaces for communication. The rest of the application can continue operating even if one service fails, reducing the effect of failures and enhancing system resilience overall. By ensuring fault isolation, it is prevented for failures in one service to cascade and affect the entire application.

4. **Maintainability and Modularity**: Microservices design encourages modularity and the division of duties. Services are simpler to comprehend, create, and manage since they are self-contained and encompass particular business functionalities. Smaller codebases allow developers to concentrate, which improves the quality of the code, the testability, and the ease of adding new features or addressing bugs.

5. **Technology Heterogeneity**: The usage of various technologies and programming languages for distinct services is made possible by the microservices architecture. Because of their ability to be flexible, organisations can select the tools and frameworks that are most suited for each service, utilising the advantages of many technologies and promoting innovation. As each service may communicate with

external systems using the right protocols, it also makes it easier to integrate old systems or third-party services.

IV. Challenges of Microservices Architecture

1. **Distributed System Complexity**: A distributed system environment with several services interacting over a network is introduced by the microservices architecture. It can be difficult to control the intricacies of inter-service communication, including network latency, message formats, and maintaining dependable connection. To effectively handle these issues, organisations must put in place reliable communication methods and mechanisms.

2. **Data Consistency and Transactions:** In a distributed microservices context, maintaining data consistency across numerous services can be challenging. Implementing and designing transactions that span several services carefully is necessary. To guarantee data integrity across services, organisations must implement tactics like distributed transactions, event-driven architectures, or eventual consistency patterns.

3. **Service orchestration and choreography**: Managing the interconnections between numerous microservices and coordinating their activities can be difficult. Organisations must choose between decentralised choreography patterns and centralised service orchestration. In terms of complexity, adaptability, and scalability, both strategies have trade-offs. Critical factors to take into account include picking the right pattern and putting in place efficient service coordination systems.

4. **Operational Overhead**: The architecture of microservices adds to operational complexity. Each service needs to be independently installed, watched over, scaled, and maintained. To reduce operational overhead, strong infrastructure, efficient monitoring tools, and efficient DevOps procedures are required. To deal with the operational difficulties brought on by managing many services, organisations must invest in automation and infrastructure-as-code solutions.

5. **Testing and Debugging**: Comparing a monolithic design to a distributed microservices system, testing and debugging can be more difficult. Comprehensive testing methodologies are needed to ensure end-to-end functionality and identify problems across multiple services. To ensure proper behaviour and uphold quality assurance, organisations must use testing strategies including integration testing, contract testing, and canary releases.

6. **Monitoring and Observability**: It might be challenging to keep track of the performance, health, and availability of numerous microservices. To acquire insights into the behaviour of the system and quickly identify faults, organisations must employ efficient monitoring and observability practises, including distributed tracing, logging, and centralised monitoring tools. For system upkeep, it is essential to ensure thorough monitoring across all services. Various cloud service providers like AWS, GCP and Azure provide robust monitoring services.

V. Cloud Platforms

1. **Amazon Web Services**: Microservice deployment and maintenance are made easier by Amazon Web Services (AWS), [5] which provides a full range of cloud services. The serverless computing architecture offered by AWS Lambda allows programmers to deploy distinct functions as microservices. Microservice deployments and management using containers are supported by Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS). Additionally, AWS offers tools like AWS App Mesh for service mesh functionality and Amazon API Gateway for building APIs.

2. **Microsoft Azure**: Microservice deployment and management options are available through Microsoft Azure[6]. Serverless microservice deployments are made possible by Azure Functions, which frees developers from infrastructure management to concentrate on creating code. Deployments of containerized microservices are supported by Azure Container Instances (ACI) and Azure Kubernetes Service (AKS). Other Azure platform capabilities that facilitate creating and managing microservice architectures are Azure API Management and Azure Service Fabric.

3. **Google Cloud Platform**: Google Cloud Platform (GCP) [7] is a platform that provides a number of services for managing and deploying microservices. Microservices can run serverless thanks to Google Cloud Functions. For the deployment of containerized microservices, Google Kubernetes Engine (GKE) offers a managed Kubernetes environment. Additionally, GCP provides services like Cloud Endpoints for API administration and authentication and Cloud Run for the deployment of containerized microservices without maintaining clusters.

VI. Tools for Microservices Management

1. **Kubernetes**: An open-source container orchestration technology called Kubernetes [8] makes it easier to build, scale, and manage microservices. It offers functions including rolling updates, service discovery, load balancing, and intelligent scaling. Kubernetes is a popular option for managing microservices since it enables businesses to execute microservices in a durable and scalable manner and supports different cloud platforms.

2. **Docker**: Using Docker [9], developers can package microservices and their dependencies into small, portable containers. Docker is a containerization platform. Microservice distribution and deployment are made easier by Docker, which also ensures consistency between various environments. In order to effectively manage containerized microservices, organisations can use it in conjunction with orchestration solutions like Kubernetes.

3. **Istio**: Istio [10] is a service mesh that is open-source and offers cutting-edge features for managing traffic routing, observability, security, and microservices communication. It aids in tackling issues with service discovery, load balancing, fault tolerance, and distributed tracing that are typical in microservices

systems. Istio makes common cross-cutting issues easier to implement and offers fine-grained control over microservice interactions.

4. **Apache Kafka**: For microservices architectures, Apache Kafka [11] is a distributed streaming technology that offers dependable, scalable, and real-time data streaming. Through publish-subscribe messaging patterns, it facilitates asynchronous communication across microservices, providing secure data transfer and service decoupling. Kafka is ideal for managing large volumes of data in distributed microservices scenarios because it is extremely scalable and fault-tolerant.

5. **Netflix OSS**: A collection of tools and frameworks created by Netflix [12] to support their microservices architecture is known as the Netflix Open Source Software (OSS) suite. For service discovery, fault tolerance, client-side load balancing, and API gateway capabilities, it contains tools like Netflix Eureka, Netflix Hystrix, Netflix Ribbon, and Netflix Zuul. These tools give builders of durable and scalable microservices architectures crucial skills.

These cloud computing tools and platforms provide a wide range of capabilities to make it easier to install, scale, manage, and communicate microservices in the cloud. Depending on their unique needs and preferences, organisations can use these platforms and technologies to efficiently create and run microservices architectures.

VII. Case Studies and Real World Examples

Listed below are a few case studies and real-world examples that illustrate the successful implementation of microservices architectures:

A. NETFLIX

In order to create a highly scalable and resilient streaming platform, Netflix [12] is a well-known example of a business that has embraced microservices design. The architecture of Netflix is made up of thousands of microservices, each of which handles a different function, including user authentication, recommendation engines, content delivery, and billing. Netflix has increased availability, fault tolerance, and the capacity to expand various components independently in response to demand by implementing microservices. With the help of this infrastructure, Netflix can offer millions of viewers a flawless viewing experience.

B. UBER

Another well-known company that has embraced microservices design is the ride-hailing service Uber [13]. The architecture of Uber is made up of a number of microservices that are in charge of various facets of the platform, such as trip management, driver management, user administration, and real-time position tracking. Uber can dynamically scale their platform to support millions of concurrent users and deliver real-time changes thanks to microservices. Additionally, they enable quick product development and rollout, providing continual innovation and a better user experience.

C. AIRBNB

To support their quickly expanding platform, Airbnb, [14] an online marketplace for accommodations, successfully used microservices architecture. Numerous microservices that handle various functionalities, including search, booking, payments, reviews, and messaging, make up their architecture. Airbnb can expand its services autonomously, manage sophisticated business logic, and seamlessly interface with external services thanks to microservices. With the help of this architecture, Airbnb has been able to grow internationally, manage high traffic volumes, and offer their users a dependable and personalised booking experience.

D. SOUNDCLOUD

To provide a scalable and responsive user experience, the music streaming service SoundCloud [15] has made use of microservices architecture. The architecture of SoundCloud is made up of a number of microservices that handle audio streaming, user administration, content distribution, and recommendation engines. SoundCloud has enhanced performance, scalability, and the capacity to manage high volumes of audio content and user interactions by implementing microservices. SoundCloud has been able to deliver new features separately, iterate swiftly, and scale their infrastructure to accommodate their expanding user base thanks to the use of microservices.

E. AMAZON

Amazon.com Amazon Prime, and Amazon Web Services (AWS) [5] are just a few of the numerous services that are powered by the microservices architecture that one of the biggest e-commerce corporations, Amazon, has adopted. With the help of their microservices design, Amazon is able to grow their services throughout the globe, manage peak traffic during promotional periods, and keep high availability. In order to provide their customers with a seamless and customized shopping experience, they have developed a highly scalable architecture that enables autonomous service development and deployment.

These case studies serve as an example of the advantages of microservices architecture for large-scale platforms and applications by providing scalability, robustness, quick development, and innovation. They serve as examples of how microservices can be successfully applied to create intricate, highly available systems that satisfy the requirements of contemporary applications across diverse industries.

VIII. Research in Microservices

1. **Performance Optimisation**: Methods to improve the performance of microservices architectures are being investigated by researchers. To improve the overall performance and scalability of microservices-based applications, it requires researching effective communication protocols, load balancing approaches, caching mechanisms, and resource allocation algorithms.

2. **Fault Tolerance and Resilience**: Research in this field is focusing on creating reliable fault tolerance techniques for microservices systems. To provide high availability and fault tolerance in distributed

microservices systems entails researching techniques including distributed consensus algorithms, replication schemes, and fault detection and recovery procedures.

3. **Security and Privacy**: Privacy and security issues are being addressed by researchers in microservices architectures. This entails investigating authentication and authorisation procedures, secure communication channels, data encryption techniques, and access control mechanisms to guard microservices-based applications against security risks and weaknesses.

4. **DevOps and Continuous Deployment**: Continuous Deployment and DevOps: Research is being done to enhance DevOps practises and tooling for developing and deploying microservices. In order to enable quicker and more effective development, testing, and deployment of microservices, this requires investigating automated deployment methodologies, continuous integration and delivery pipelines, container orchestration, and infrastructure-as-code techniques.

IX. Emerging Trends in Microservices

1. **Serverless Microservices**: Serverless computing for microservices is becoming more and more popular. AWS Lambda, Azure Functions, and Google Cloud Functions are a few serverless systems that offer an easy way to deploy microservices without having to manage servers. This trend offers more precise pricing and scalability while allowing developers to concentrate entirely on building code.

2. **Service Mesh**: To handle the complexity of microservices communication, service mesh technologies like Istio and Linkerd are becoming more and more popular. Advanced features including traffic routing, load balancing, service discovery, fault tolerance, and observability are offered by service meshes. They aid in addressing issues with distributed microservices architectures' observability, security, and inter-service communication problems.

3. **Event-Driven Architecture**: In the world of microservices, event-driven designs are gaining popularity. By enabling microservices to communicate via events and asynchronous messaging systems, event driven architecture supports loose coupling and scalability. Microservices architectures frequently leverage technologies like Apache Kafka and RabbitMQ for event-driven communication, enabling real-time data streaming and event processing.

4. **Microservices and Edge Computing**: Microservices and edge computing are becoming more popular. In order to reduce latency and enable quicker response times, edge computing moves computation closer to the data source. Applications can manage data-intensive activities and react quickly when microservices are deployed at the edge because they can offer localized processing and decision-making capabilities.

5. **AI and Machine Learning in Microservices**: Artificial intelligence (AI) and machine learning (ML) in microservices is a new trend that involves incorporating these technologies into microservice architectures. Microservices can be created to tackle particular AI/ML tasks, such as image identification, fraud detection, recommendation systems, and natural language processing. This development trend makes it possible to create intelligent microservices that improve the usability and customization of apps.

These research initiatives and new trends demonstrate the microservices industry's ongoing development and advancement. In order to create scalable, robust, and effective microservices architectures, they seek to fix issues, enhance performance, strengthen security, and investigate novel ways.

When it comes to establishing and managing cloud microservices, security and governance are crucial factors to take into account.

X. Security in Cloud Microservices

1. **Authentication and Authorization**: Microservices architectures need reliable techniques for authenticating users and granting permission to access services. In order to do this, secure authentication mechanisms are implemented, such as OAuth or JSON Web Tokens (JWT), and fine-grained access controls based on user roles and permissions must be enforced. To further safeguard inter-service communication, secure communication channels (like HTTPS) and service-to-service authentication are used.

2. **Data Protection**: In microservices architectures, data security is essential. Data at rest and in transit are secured using encryption techniques. This includes using secure key management procedures, encrypting databases, and implementing transport layer security (TLS) protocols. Sensitive data within microservices are anonymized via data masking and tokenization.

3. **Secure Service-to-Service Communication**: Secure service-to-service communication is crucial since microservices frequently communicate over networks. For inter-service communication, service mesh technologies like Istio or Linkerd can offer secure communication channels, authentication, and encryption. Additionally, network segmentation and the adoption of mutual TLS (mTLS) increases the security of service-to-service communication.

4. **Threat Detection and Monitoring**: For the identification and mitigation of security threats in microservices architectures, it is essential to have strong logging, monitoring, and intrusion detection systems. Individual service health, performance, and security can be seen through the use of centralised logging and monitoring platforms. Techniques for real-time monitoring, anomaly detection, and log analysis assist in quickly identifying and mitigating hazards.

5. **Infrastructure and Secure Deployment**: To safeguard the infrastructure that supports microservices, secure deployment practices are adopted. The underlying infrastructure are made more secure by using secure container registries, infrastructure-as-code (IaC) techniques, and secure configuration management procedures. To find and fix potential security holes, regular vulnerability assessments, penetration tests, and security audits are also carried out.

XI. Governance in Cloud Microservices:

1. **Service Discovery and Registry**: Integrating a service discovery and registry mechanism into microservices architectures is essential for governance. Services can dynamically find and interact with one another because of it. Effective service registration, discovery, and management are made possible by tools like Consul, Eureka, or Kubernetes Service Discovery, facilitating governance and control over the services deployed in the architecture.
2. **API Gateway and Management**: API gateways offer a single point of entry for microservices, enabling governance over their use, security, and monitoring. They offer analytics and usage information in addition to enforcing restrictions, rate limiting, and authentication techniques. Apigee, Kong, or Azure API Management are a few API management tools that can help with successfully managing and governing microservices APIs.
3. **Service versioning and lifecycle management**: Microservice versioning and lifecycle management are essential for effective governance. Adopting semantic versioning methodologies, for example, enables backward compatibility and easy migration. Release management techniques, service contract definition, and management of service dependencies all contribute to proper microservice lifecycle governance.
4. **Compliance and Regulatory Requirements**: Cloud microservices must adhere to all applicable rules and regulations, including those governing data privacy. Organisations are responsible for making that the microservices architecture complies with all relevant standards and laws, such as GDPR, HIPAA, and PCI DSS. Governance and compliance are ensured by putting security measures, privacy rules, and data access limits into place in accordance with these criteria.
5. **Documentation and Change Management**: Upholding change management procedures and documentation is essential for governance in microservices architectures. The architecture is better understood by developers and operators when service APIs, dependencies, and configurations are clearly documented. Governance and control are facilitated by the implementation of change management procedures, such as version control, release management, and testing.

XII. Future scope of work

FaaS platforms and serverless computing are growing in popularity because they offer a more granular method of deploying microservices. In the future, organisations will be able to create and deploy individual microservices as functions while only paying for the real execution time owing to the convergence of serverless computing and microservices architecture.

The integration of AI and ML capabilities within microservices architectures will become increasingly important. In order to add highly intelligent functionality to their apps, businesses will use machine learning models and AI algorithms as microservices. The creation of data-driven, intelligent microservice architectures will be enabled due to this integration.

Security will remain a critical concern for microservices architectures. Enhancing security methods like encryption, access control, and authentication within microservices will be the primary focus of future advancements. Circuit breakers and chaos engineering are two methods for creating robust systems that will become increasingly popular.

XIII. Conclusion

This paper covered many elements of cloud microservices, including their architecture, advantages, difficulties, and the platforms and tools that may be used for their deployment and maintenance. Additionally, emphasis here is on the case studies and actual instances of businesses that have effectively utilised microservices. Microservices design has a number of benefits, including enhanced fault tolerance, continuous deployment, and independent scaling. But it also comes with difficulties, such as harder dispersed system management, more secure communication, and consistency across services. In order to effectively address these difficulties, organisations must carefully weigh their implications and implement the right technologies and methods. In conclusion, Microservices architecture is a distributed design approach intended to overcome the limitations of traditional monolithic architectures. For building large and complex systems, industries are moving towards adopting agile style of building. With microservices it has become fairly easy for businesses to adapt rapidly to changing requirements with software components by avoiding extensive recoding and retesting that is generally required in the monolithic applications. Microservices architecture modules go through continuous delivery and testing processes, hence the delivery of applications which are error free is improved significantly. Increase in efficiency of microservices reduce downtime as well as reduces infrastructure costs. Adoption of microservices is becoming essential as small- and large-scale enterprises modernise their application portfolios. It also paves a way for the applications and services to reach the market quicker.

References

[1] R. Buyya, "Cloud computing: The next revolution in information technology," in Parallel Distributed and Grid Computing (PDGC), 2010 1st International Conference on, pp. 2–3, Oct 2010

[2] J. Lewis and M. Fowler, "Microservices," 2014.

[3] Rong ZENG , Xiaofeng HOU , Lu ZHANG , Chao LI , Wenli ZHENG , Minyi GUO , "Performance optimization for cloud computing systems in the microservice era: state-of-the-art and research opportunities", 2022 Front. Comput. Sci., 2022

[4] G. Blinowski, A. Ojdowska and A. Przybyłek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," in IEEE Access, vol. 10, pp. 20357-20374, 2022, doi: 10.1109/ACCESS.2022.3152803.

[5] Vural, Hulya & Koyuncu, Murat & Guney, Sinem "A Systematic Literature Review on Microservices" International Conference on Computational Science and Its Applications, 2017

[6] Carutasu, George "Cloud Computing and Windows Azure" ECAI 2016 - International Conference – 8th EditionAt: Ploiesti, ROMÂNIA, 2016

[7] E. Roloff, F. Birck, M. Diener, A. Carissimi, and P. Navaux,"Evaluating high performance computing on the windows azure platform, IEEE 5th International Conference on, 2012

[8] A. Pereira Ferreira and R. Sinnott, "A Performance Evaluation of Containers Running on Managed Kubernetes Services," 2019 IEEE

International Conference on Cloud Computing Technology and Science (CloudCom), 2019

[9] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment", Linux Journal, vol. 2014, no. 239, pp. 2, 2014.

[10] Karn, Rupesh & Das, Rammi & Pant, Dibakar & Heikkonen, Jukka & Kanth, Rajeev "Automated Testing and Resilience of Microservice's Network-link using Istio Service Mesh" Proceedings of the XXth Conference of Open Innovations Association, 2022

[11] B. R. Hiraman, C. Viresh M. and K. Abhijeet C., "A Study of Apache Kafka in Big Data Stream Processing," 2018 International Conference on Information , Communication, Engineering and Technology (ICICET), Pune, India, 2018

[12] Asaye Bobmanuel, Adetokunbo "THE STRATEGIC ANALYSIS OF NETFLIX", 2022

[13] https://www.uber.com/en-IN/blog/microservice-architecture/

[14] https://quastor.substack.com/p/airbnbs-architecture

[15] https://developers.soundcloud.com/blog/inside-a-soundcloud-microservice