https://doi.org/ 10.33472/AFJBS.6.Si2.2024.3563-3570

Optimizing Scalability and Load Balancing for Microservices in DevOps - Driven Systems.

K.KARUNAKAR
Department of Computer Science and Engineering
Koneru Lakshmaiah Education Foundation
Guntur, Andhra Pradesh, India
2100030993cseh@gmail.com

N.V. MANOJ SAI
Department of Computer Science and Engineering
Koneru Lakshmaiah Education Foundation
Guntur, Andhra Pradesh, India
2100031080cseh@gmail.com

N. PAVAN KRISHNA PRASAD
Department of Computer Science and Engineering
Koneru Lakshmaiah Education Foundation
Guntur, Andhra Pradesh, India
2100031447cseh@gmail.com

B.UDAY
Department of Computer Science and Engineering
Koneru Lakshmaiah Education Foundation
Guntur, Andhra Pradesh, India
2100031526cseh@gmail.com

K.V.V.SATHYANARYANA
Department of Computer Science and Engineering
Koneru Lakshmaiah Education Foundation
Guntur, Andhra Pradesh, India
koparti@kluniversity.in

Abstract:

Many companies are now using smaller, independent software parts called microservices. These parts help in quicker deployments, work well together, and make it easier to adapt to changes. They're often put inside containers and managed using systems like Kubernetes. We're exploring how these microservices handle tasks, especially in dealing with lots of data (like Big Data) and managing lots of users. We found they can be more effective when they're balanced well across different computers and when their connections are managed carefully. They're good for fast-paced changes in what customers want, yet they additionally present certain challenges. For example, they require more complex ways of managing them and ensuring their proper operation together. In our research, we've seen that these microservices work better when they're organized and managed thoughtfully. They can make systems faster and more flexible but need careful handling to work their best. This summary focuses on the advantages, challenges, and key considerations when using microservices in a simple manner.

Keywords: *Quicker Deployments, Containerization, Kubernetes management, Scalability and load balancing, Continuous Deployment, DevOps, Microservices.*

Introduction:

In recent years, the evolution from traditional monolithic architectures to the acceptance of microservices has redefined the area of software development. There have been spurred shifts. by the inherent advantages microservices offer, such as streamlined application deployments, continuous integration, and the facilitation of independent development. However, this transformation introduces a network-centric dimension, fragmenting monolithic systems into independently deployable components. Microservices, often implemented as containers within Kubernetes-managed clusters, have sparked innovations and challenges in load balancing and architecture. Traditional load balancing algorithms, like Least Connection, face new hurdles in a distributed microservices environment, where multiple

instances operate independently, posing challenges in maintaining accurate metadata caches. This paradigm shift aligns with the growth trajectory observed in DevOps and microservices, where DevOps principles find efficient application in the microservices framework, emphasizing smaller, specialized teams and diverse platforms. Case studies, exemplified by Backstory's transition, showcase how migrating from monolithic to microservices architectures aligns with DevOps, enhancing deployment frequency and scalability across diverse user bases. Furthermore, Big Data applications demand specialized environments and effective load balancing solutions. Integrating containerized environments, notably Docker a Docker Swarm has emerged as a novel approach to address load balancing needs in Microservice-based Big Data applications. Research into microservices' performance, scalability, and resource management emphasizes their advantages in dividing up difficult tasks into smaller, manageable components. But these investigations also highlight challenges in CPU balancing, efficient task scheduling, and understanding the result of task interconnections on overall system efficiency. While the transition to microservices offers promising advantages, it also presents complexities in managing diverse services, evolving relationships between services, technological diversity, and testing challenges. This research aims to explore and elucidate the multifaceted impact of microservices on software architecture, load balancing, resource management, and the overall dynamics of modern software systems.

Literature Survey:

[1] The rise of microservices has gained friction because of several benefits they offer such as easy deployments of applications; it continuously integrates and the ability for independent development, but this strategy also presents a network aspect into the architecture by breaking down a monolithic system into several autonomously deployed components. In these-days microservices are often deployed as containers within clusters managed by Kubernetes.

The Least Connection algorithm, commonly for load balancing in microservices architecture, originates from the join the shortest Queue approach, known for efficiency in approximating optimal load balancing within a single node centralized system. However, in the microservices landscape, the load balancer no longer operates as a single centralized node, but rather as multiple instances linked to each client service. In this distributed setup, Least Connection encounters challenges maintaining accurate metadata cache which can decrease rapidly.

[2] Based on Google trends both DevOps and microservices have shown similar growth trajectories. While DevOps principles can be applied to monolithic systems, microservices offer an efficient framework for implementing DevOps by emphasizing the significance of smaller, focused teams. These services can run on diverse platforms and utilize different technological stacks, communicating through lightweight methods such as RESTFUL or RPC-based APIs. In this setup, each service represents a distinct business capability and can employ various programming languages and data stores, typically developed by compact teams.

In this author said that the migration from monolithic architectures to microservices, as exemplified by Backstory's platform transition, has showcased numerous advantages aligning with DevOps principles. This shift enabled Backstory to enhance feature deployment frequency and scale effectively for a diverse user base across various mobile applications. Overall, the experiences shared through backstory's incremental migration highlight the substantial advantages of adopting microservices in conjunction with DevOps methods. This combination fosters agility, scalability, and accelerated development, significantly benefiting both the platform and its user base**.[3]** Big Data applications demand substantial resources and specialized resources and specialized environments for storing, processing, and analyzing vast volumes of data across distributed systems. While containerization through cloud computing offers a solution, it is a suitable load-balancing system. As resources increase, there's an exponential rise in server load, highlighting the critical need for effective load balancing. Furthermore, the rapid and accurate adjustment of containers according to service loads stands as a crucial factor in Big Data Applications.

This study delves into the integration of containerized environments like Docker in addressing the loadbalancing needs of Big Data applications. It proposes a novel scheduling mechanism leveraging Docker Swarm and Microservice architecture tailored for Big Data application environments. This study emphasizes the advantages of employing containerization in Microservice-based Big Data applications. Docker containers, paired with the orchestration tool Docker Swarm, effectively manage the critical aspects of load balancing and service discovery within Microservices. **.[4]** Microservices are comparable to the building blocks for businesses in the digital world. divide up large tasks into smaller, easier-to-handle pieces, which helps companies adapt quickly to new technologies and changes in what customers want. These smaller parts also make it easier for businesses to work with other outside services, making operations smoother and improving how customers experience their products or services.

In our study, we introduced a new way of organizing how a business uses its technology called the MicroservicesDriven Enterprise Architecture Model. We tested this model on different types of computer processors to see how well it works and how it can make things run better. What we found is that this model can grow or shrink to handle more work as needed, no matter the type of processor. It also used its resources more effectively in contrast to the other type of processor, making the tasks it handles run more efficiently. **[5]** Small, independent microservices are becoming more popular instead of big, all-in-one applications. These microservices possess a few distinct clear advantages over the older, larger applications. They make it easier for different teams to work together in the

same project, and they're simpler to put into action. Big companies like AWS and Alibaba are now offering ready-made microservices setups for people to use, which makes it easier for them to launch their own apps.

Our research into how microservices perform while running has some important findings for how we schedule them and manage resources in groups. We discovered that most of these microservices are affected more by other tasks using the CPU rather than by how much memory is being used. This means we should focus on making sure the CPU is balanced well between different computers to schedule these tasks better. Also, we noticed that the time it takes for online services to respond depends a lot on how different tasks are connected. This means we need to consider these connections when we're trying to make the scheduling of microservices more efficient. But figuring this out can be quite tricky. **[6]** A lot of developers are now using microservices to address the difficulties of building monolithic applications. A recent trend involves deploying these microservices using containers, especially across distant cloud locations. Containers, a more lightweight alternative to virtual machines (VMs), have gained popularity in the industry. Unlike VMs, they are simpler to use and require less space, making them a favored choice for many developers. As more people use the framework and more data piles up, things get more complicated. When an excessive number of people are using it, the system slows down, making it a bad experience for them. The way the system was set up originally starts to get messy and weak.

This study focused extensively on load balancing and auto-scaling issues concerning Cloud-based Container microservices. It thoroughly explored topics like Microservice architecture, cloud-based microservices, and container microservices, shedding light on the specific challenges related to load balancing and autoscaling within microservices.**[7]** This study looks at how problems (called faults) affect how well online software services can handle lots of users at once. They used a special test to see how faults in the software affect its ability to handle many people using it. Other studies helped understand how well these services can handle lots of users, so this study could dig deeper into that.

They ran tests on a cloud platform called EC2 using a real software service. They made the software slowdown in two different ways and tested it with different levels of demand. They describe what they did on these tests in detail. This paper talks about a way they tested how well online software can handle lots of users. They used a special method called ALFI. They describe what they did by using four things: a tool to create lots of work, a problem in the software, ways to measure how well it handles lots of users, and the actual software and its setup. They tested this with a program called Orangery on EC2, a cloud service. They made the software slowdown in two different ways and tested it with different levels of people using it. They discovered that their approach was good at showing how these

problems affect how well the software can handle lots of users**.[8]** Companies today are aiming to meet customer demands faster than ever. To achieve this, many are adopting a methodology called DevOps and embracing Continuous Delivery (CD). I worked on implementing these practices at Paddy Power, a huge betting and gaming company, for four years. During this time, I discovered that the structure of software can sometimes be a big challenge. To tackle these challenges, we explored a newer way of designing software known as Microservices. What I noticed was that using Microservices brought advantages like making it easier to deploy, modify, and safeguard against the breakdown of the software's original design.

In conclusion, the adoption of DevOps and Continuous Delivery (CD) as a reaction to the fast-paced customer demands led to the investigation of Microservices as an emerging software architecture approach. While Microservices offered advantages such as improved deploy ability, flexibility, and resilience against design issues, it also introduced complexities related to managing a larger number of services, evolving service relationships, technological diversity, and testing challenges**.[9]** Cloud services have recently shifted from large, all-in-one applications to a more fragmented approach called microservices. These microservices consist of numerous smaller, interconnected components, offering both opportunities and difficulties in ensuring top-notch service quality and efficient cloud usage. This paper explores how microservices affect datacenter server design and bottlenecks in the system. Specifically, we reexamine the age-old debate of using powerful 'brawny' cores versus more modest 'wimpy' cores within the microservices framework**.** We assess how much of a burden these microservices put on instruction caches (I-cache) and measure how much time is spent computing as opposed to communicating across services via Remote Procedure Calls (RPCs). **[10]** Imagine putting together smaller, interconnected computer programs that can handle a lot of data at once. These programs use tools like Apache Flink, Apache Kafka Streams, or others to process huge amounts of data smoothly. Although these tools claim they can handle a lot, we don't have much real-world proof comparing how well each of them really scales up to big tasks. Usually, when working with large amounts of data, we use one big system to handle all the data processing tasks. But now, with microservice setups, we can place smaller systems for handling data processing within each tiny service. It's like having mini data processors inside each service. This way, we can pick lighter and more fitting data processing tools for each specific job. We looked at our findings without focusing on how much work each small service could handle, how much computing it did, or which cloud setup we used. These data processing systems can also manage additional work if one of its powers is increased. In summary: We found that all the frameworks we tested could be made scalable. But the specific framework chosen and how it's set up can significantly affect how much it costs to run.

[11] Microservices architecture is a service-oriented architecture structure that produces software system as a collection of little services, each alone conveyable on a

different floor and software stack. Microservices is also called as a microservices architecture. It is important for the migrating monolithic architecture to microservices gives many advantages, but it is not pliability to modify to the technology converts to keep away from technology demonstration, and more seriously, lower time-to-market. Microservices are helped in various ways, mostly in posting new features more regularly. DevOps is a list of practices in the relocation process towards microservices. DevOps practices can be utilized for monoliths, but microservices enables a successful execution of the DevOps through increasing the significance of small teams**.[12]** Hence, there is an increasing need for additional research to simplify the challenges that come with working on big data projects. A promising avenue for addressing this challenge is focusing on data architecture. Data architecture provides the framework for a versatile and scalable BD system, capable of adapting to evolving requirements. An effective approach to assimilating the abundance of knowledge in data architecture is through Reference Architectures. In the context of this study, the first
Systematic Literature Review is conducted to gather all Microservices patterns present in the existing body of knowledge. The second SLR aims to identify and compile all existing Big Data Reference Architectures, highlighting their architectural components and limitations. The outcomes of these Systematic Literature Reviews are then systematically gathered and combined using thematic synthesis.

Through this process, the study generates and validates various design theories by seeking expert opinions. In essence, the research collects insights from the literature on both Microservices and Big Data Reference Architectures, creating a foundation for developing and confirming design principles in these domains**.[13]** Certainly! Kubernetes is a free and open-source platform designed to streamline the deployment, scaling, and management of applications packaged in containers. Containers enable developers to package an application along with its dependencies, ensuring consistent performance across various environments.

Kubernetes simplifies the deployment process by allowing developers to focus on building and deploying their applications without the need to concern themselves with the intricate details of the underlying infrastructure. It operates on a declarative model, where users define the desired state of their applications, and Kubernetes takes care of maintaining that state.

One key feature of Kubernetes is its ability to automatically detect and recover from failures, providing a self-healing mechanism for applications. This enhances the reliability and availability of applications in production environments.

In summary, the survey on Kubernetes scheduling offers a thorough examination of the present landscape in this field. It delves into the goals, approaches, algorithms, experiments, and findings from various research endeavors. The report emphasizes how important scheduling is to Kubernetes and how crucial it is to have excellent scheduling algorithms.

Notably, the experimental results indicate that there is still room for enhancement in this domain. The findings suggest that future efforts should concentrate on devising novel algorithms and refining existing ones to further optimize Kubernetes scheduling. In essence, the survey serves as a valuable resource, shedding light on the current state of Kubernetes scheduling and suggesting promising avenues for future research**.[14]** The concept of auto-scaling becomes crucial for both cost efficiency and enhancing Quality of Service (QoS). This study aims to explore solutions related to load balancing and auto-scaling that can dynamically adjust the allocation of materials for cloud services based on incoming workloads. The focus is particularly on cloudbased container microservices, which are components of a larger system that utilizes containers to deliver its services. This paper reviews the challenges associated with load balancing and auto-scaling in the context of cloud-based container microservices. It delves into topics such as microservices in a cloud-based system, containerized microservices, and the key objectives and issues involved in load balancing and auto-scaling within microservices. The primary motivation behind this research is to address current issues related to these techniques, aiming to mitigate problems like server overload, sudden spikes in traffic, and service failures. The goal is to enhance the overall performance of services, thereby providing a better Quality of Service to end-users. This paper thoroughly examined the significant challenges associated with load balancing and auto-scaling in Cloud-based Container microservices. The detailed exploration covered various aspects, including microservice architecture, cloud-based microservices, containerized microservices, additionally specific issues related to load balancing and auto-scaling within the context of microservices. The study delved deeply into understanding and addressing these challenges to contribute valuable insights to the field**.[15]** Microservices represent a modern approach to software architecture, breaking down a single program into a set of small, independent services This differs from customary monolithic apps, where all functions are bundled into a single entity. With microservices, each program operates independently in its own process. These services communicate and collaborate with each other, working together to maximize the overall effectiveness of the system. In today's computing system architecture, microservices play a crucial role as a paradigm and infrastructure, offering a more flexible and scalable way to design and manage complex software systems.

Problem Statements and Solutions:

Problem Statement	**Potential Solutions**	**Contextual Evidence**
Maintaining accurate metadata cache for Least Connection load balancing in distributed microservices architecture.	Implement gossip protocols like epidemic or anti-entropy algorithms to propagate service status updates among different load balancer instances, ensuring consistent metadata across the distributed system.	Microservices architecture's distributed nature: The text highlights the shift from centralized monolithic systems to distributed. [1] microservices, increasing network complexity and introducing challenges for traditional load balancing algorithms like Least Connection.
Network complexity of distributed microservices.	Utilize Docker Swarm and container orchestration for efficient management and communication.	Backstory's experience [2] and Least Connection challenges highlight the need for orchestration.
Maintain accurate metadata cache for Least Connection load balancing.	Implement gossip protocols or distributed key-value stores for consistent data propagation.	Study on containerized Big Data applications [3] emphasizes distributed data management.
Microservices as building blocks for businesses to adapt to changes.	Easier for teams to work, faster deployments, smoother operations, and improved customer experience.	Study on Microservices-Driven Enterprise Architecture Model [4] highlights these benefits.
Schedule and manage resources in microservices based on CPU usage and task dependencies.	Focus on CPU balancing and consider connections between tasks for scheduling efficiency.	Research on performance measurement of running microservices [5] provides these insights.
Challenges of deploying containerized microservices across cloud locations.	Increased complexity and slowdown as user base grow.	Study on load balancing and auto-scaling issues in cloud-based container microservices [6] addresses these challenges.
Fault tolerance testing for online software services.	Developed a method to test how faults affect service performance with different user demands.	Study on online software service performance under faults [7] describes this method.

DevOps and Continuous Delivery leading to exploration of microservices for tackling challenges.	Microservices offered advantages like improved deploy ability, flexibility, and resilience.	Paddy Power's experience implementing DevOps and microservices [8] highlights these benefits.
Microservices impact on system bottlenecks and datacenter server design.	Reexamining the use of 'brawny' vs. 'wimpy' cores within microservices framework.	Study on cloud services and microservices impact on system bottlenecks [9] explores these aspects.
Scalability of different data processing frameworks for microservices.	Specific framework choice and setup can significantly impact cost.	Study on comparing big data processing frameworks within microservices [10] highlights this finding.
Difficulty in adapting microservices architecture to changing technology while maintaining rapid deployment and time-to-market.	Focus on loosely coupled services and **APIs:** Design microservices with welldefined and stable APIs that minimize dependencies.	The text states that while microservices offer advantages like faster deployments and easier feature updates, it can be inflexible when adapting to technological changes due to technology lock-in and the complexity of transitioning between different software stacks [11].
Complexities in Big Data projects due to evolving requirements and lack of clear design principles.	Focus on data architecture as a foundation: Develop a well-defined data architecture that outlines the system's components, data flows, and governance protocols. This provides a clear roadmap for managing and evolving the Big Data system.	The passage states that Big Data projects often face difficulties due to the need for adaptability and a lack of established design principles [12]. Existing approaches tend to rely on monolithic architectures or fragmented, poorly understood patterns.
Optimizing scheduling algorithms within Kubernetes to enhance resource utilization, performance, and overall efficiency.	Facilitate cross-cluster scheduling: Allow for scheduling across multiple Kubernetes clusters to leverage shared resources and improve resource utilization in distributed environments.	The survey highlights the critical role of scheduling in Kubernetes and the ongoing need for improvement in this area [13]. Research findings indicate that current scheduling algorithms have room for optimization to better meet diverse application needs and resource constraints.
Ineffective load balancing and autoscaling strategies for cloud-based container microservices, leading to bottlenecks, service failures, and poor QoS.	Implement adaptive load balancing algorithms: Utilize machine learningpowered algorithms that can dynamically adjust load distribution based on real-time metrics and application-specific requirements.	The paper highlights the critical role of load balancing and auto-scaling in optimizing resource utilization and ensuring service availability in cloudbased container microservices environments [14]. Current approaches face challenges in handling sudden traffic spikes, preventing server overload, and efficiently scaling resources to meet dynamic workload demands.

Taming the Load Balancing Beast in Modern Microservices Architectures.	Consistent Hashing: Maps services and requests to specific servers based on pre-defined hash functions, ensuring predictable distribution but limiting flexibility in workload adjustments.	The rise of complex microservices architectures has brought new challenges in managing service load and maintaining optimal capacity. Ensuring continuous availability while minimizing request delays is paramount.[15]

Conclusion:

The study highlights the need for more investigation, particularly in the areas of load balancing, auto-scaling, and resource management in cloud-based container microservices and Kubernetes scheduling algorithm optimization. It draws attention to how important these areas are to improving the effectiveness and quality of services. The shift from monolithic software development to microservices is significant. It offers scalability, flexibility, and the capacity to work with emerging technologies. However, further study is required to improve these systems even more for the future because there are still issues to be resolved. Microservices must be carefully organized and managed to realize their benefits. In general, microservices offer speed and flexibility, but to fully realize their potential, they must be handled carefully.

References:

[1] BLOC: Balancing Load with Overload Control In the Microservices Architecture Ratnadeep Bhattacharya Department of Computer Science George Washington University ratnadeepb@gwu.edu Timothy Wood Department of Computer Science George Washington University timwood@gwu.edu

[2] Microservices: Architecting for Continuous Delivery and DevOps Lianping Chen Lianping Chen Limited Dublin, Ireland lianping.chen@outlook.com

[3] Load balancing and service discovery using Docker Swarm for microservice based big data applications Neelam Singh1 , Yasir Hamid2 , Sapna Juneja3 , Gautam Srivastava4,5,6, Gaurav Dhiman1,7,8,9, Thippa Reddy
Gadekallu9,10 and Mohd Asif Shah11,12*

[4] Microservices-driven enterprise architecture model for infrastructure optimization A. M. Abd-Elwahab1 , A. G. Mohamed2* and E. M. Shaaban3

[5] Characterizing Microservice Dependency and Performance: Alibaba Trace Analysis Shutian Luo ∗ ‡ Shenzhen Institute of Advanced Technology, CAS Univ. of CAS, Univ. of Macau st.luo@siat.ac.cn Huanle Xu ∗ ‡ University of Macau huanlexu@um.edu.mo Chengzhi Lu‡ Shenzhen Institute of Advanced Technology, CAS Univ. of CAS cz.lu@siat.ac.cn Kejiang Ye‡ Shenzhen Institute of Advanced Technology, CAS kj.ye@siat.ac.cn Guoyao Xu Alibaba Group yao.xgy@alibaba-inc.com Liping Zhang Alibaba Group liping.z@alibaba-inc.com Yu Ding Alibaba Group shutong.dy@alibaba-inc.com Jian He Alibaba Group jian.h@alibaba-inc.com Chengzhong Xu†‡ University of Macau czxu@um.edu.mo

[6] A Cloud-Based Container Microservices: A Review on Load Balancing and Auto-Scaling Issues Shamsuddeen Rabiu a,1,*, Chan Hauh Yong a,2 , Sharifah Mashita Syed Mohamad a,3 a School of Computer Sciences, University Sains Malaysia,11800 USM, Pulau Pinang Malaysia. 1 shamsrabiu@student.usm.my; 2 hychan@usm.my; 3 mashita@usm.my * corresponding author

[7] Scalability resilience framework using applicationlevel fault injection for cloud based software services Amro Al-Said Ahmad1,2* and Peter Andras3

[8] Microservices: Architecting for Continuous Delivery and DevOps Lianping Chen Lianping Chen Limited Dublin, Ireland lianping.chen@outlook.com

[9] The Architectural Implications of Cloud Microservices Yu Gan and Christina Delimitrou Cornell University {yg397,delimitrou}@cornell.edu

[10] Microservices Architecture Enables DevOps: An Experience Report on Migration to a Cloud-Native Architecture Armin Balalaie Abbas Heydarnoori Pooyan Jamshidi

[11] Application of microservices patterns to big data systems Pouya Ataei1* and Daniel Staegemann2

[12] A survey of Kubernetes scheduling algorithms Khaldoun Senjab1 , Sohail Abbas1*, Naveed Ahmed1 and Atta ur Rehman Khan2

[13] A Cloud-Based Container Microservices: A Review on LoadBalancing and Auto-Scaling Issues Shamsuddeen Rabiu a,1,*, Chan Hauh Yong a,2 , Sharifah Mashita Syed Mohamad a,3 a School of Computer Sciences, University Sains Malaysia,11800 USM, Pulau Pinang Malaysia. 1 shamsrabiu@student.usm.my; 2 hychan@usm.my; 3 mashita@usm.my * corresponding author

[14] Load Balancing in Microservices Architecture Shetty Srinidhi Udaya Department of Information Technology (MSc. IT Part I) Sir Sitaram and Lady Shantabai Patkar College of Arts and Science, Mumbai, India

[15] Challenges and Solution Directions of Microservice Architectures: A Systematic Literature Review Mehmet Söylemez 1 , Bedir Tekinerdogan 2,* and Ayça Kolukısa Tarhan 1 1 Department of Computer Engineering, Hacettepe University, 06800 Ankara, Turkey; mehmetsoylemez@hacettepe.edu.tr (M.S.); atarhan@hacettepe.edu.tr (A.K.T.) 2 Information Technology Group, Wageningen University & Research, 6706 PB Wageningen, The Netherlands * Correspondence: bedir.tekinerdogan@wur.nl