National College of Ireland

Smart Circuit Management: Enhancing Kubernetes Service Mesh through Dynamic Circuit Breaker

MSc Research Project
Cloud Computing

Manu Kumar Kudaragundi Channappa
Student ID:23200341

School of Computing
National College of Ireland

Supervisor: Ahmed Makki

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Manu Kumar Kudaragundi Channappa
Student ID:	23200341
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Ahmed Makki
Submission Due Date:	12/12/2024
Project Title:	Smart Circuit Management: Enhancing Kubernetes Service Mesh through Dynamic Circuit Breaker
Word Count:	6599
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Manu Kumar Kudaragundi Channappa
Date:	9th December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	✓
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	✓
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	✓

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Smart Circuit Management: Enhancing Kubernetes Service Mesh through Dynamic Circuit breaker

Manu Kumar Kudaragundi Channappa
23200341

Abstract

This project Enhancing Kubernetes Service Mesh with dynamic circuit breaker which focuses on key abilities of Circuit breaker of Istio Service Mesh providing enhanced performance with resiliency of Micro-service and controlled service latency and error rates under different variable conditions to optimize its maximum ability to reduce service failures and improves overall system stability. The entire project revolves in gaining maximum resiliency and demonstrating its ability to perform under variable stress conditions on adapting dynamic circuit breaking principles. With deploying the AWS EKS cluster in high availability and Istio Service mesh for its great network policies and proxy communication, Continuous real-time traffic is monitored by observability tools like Prometheus and Grafana which gives exact traffic observation to evaluate the circuit management and understanding the Micro-service communication and its resiliency, thus with Cloud Native adaptation bringing up challenges in working with Micro-service traffic management, improving fault tolerance and reduced latency which makes adaptation of Service mesh dynamic circuit breaking principle an ideal choice for managing and maintaining resiliency, real time traffic pattern is provided by injecting Fortio for stressing the traffic for continuous inputs of traffic flow into the application, with Over 40% of high Resiliency obtained in Adapting Dynamic principle On static rule helps in gaining improved latency and high performance contribution to Istio Service Mesh Micro-service architecture

***Keywords*— Kubernetes, Service Mesh, Istio,Circuit Breaker, Metrics Monitoring, Resiliency and Performance**

1 Introduction

1.1 Background:

The increase in the adaptation of micro service Architecture and shift from Monolithic culture has revolutionized the software development system providing a great chance to increase in modularity, Scalability and maintainability to the application development and deployment structure, In adaptation to the new Micro-service paradigm introduced significant new challenges in managing services to service communication mainly ensuring system resiliency and fault tolerance to the micro-service architecture by external environment. A Service Mesh like Istio, Providing a dedicated infrastructure layer to handle the communication across all the micro services with providing features like traffic management, security and Observability, Despite with these available benefits ensuring the system robust fault tolerance in Service mesh is a critical concern, Current Implementation techniques often rely on static configuration rules, which is not adaptable for dynamic and unpredictable traffic patterns which can lead to potential service failures and cascading outages.

1.2 Motivation:

The necessity to enhance the resiliency of micro-services in Kubernetes Service Mesh is demanded as the system often faces frequent occurrences of service failures and its cascading failure effects as micro services are interconnected in operation, This significantly impact the application availability effecting user experience. Studying Dynamic circuit breaker is an imperative way to resolving cascading failure effect because they offer more adaptable and responsive approach for managing service failures with various approaches, unlike static configuration which have less ability to cope with variable traffic conditions, By Investigating and implementing the dynamic circuit breaker in the Service Mesh, Our research aims

to contribute more reliable and fault tolerant micro service architecture, Contributing to overall increase in improving system stability and performance maintaining a standard availability.

Variables or Factors Affecting the Outcome: Several factors can influence the effectiveness of dynamic circuit breakers in a service mesh including:

Factors	**Effects**
Traffic Patterns	Variable difference and Real-time Spikes
Compute Capacity	Resource Size and Computational type
Configuration Parameter	Circuit Breaking Settings and Opening and closing of Circuits
Failures	Resource Exhaustion, Server Crash and Network issues

1.3 Research Question:

Based on above inspiration, Our Project is framed with following research question, which sets the stage for conducting experiments and deriving desired outcomes.

- *What is Dynamic Circuit Breaker on existing Micro-service Architecture?*
- *How does the implementation of a Dynamic Circuit Breaker on Istio Service Mesh impact Error Rates and Concurrent Pending Requests of the Application Traffic?*
- *What is its effect on system Resilience and Reliability for Real-Time Monitoring traffic in a Dynamic Adaptation Architecture?*

1.4 Research Objective:

Investigating the state of art of the Service Mesh architecture,their circuit breaking pattern in reviewing and addressing literature review for identifying the key important points where dynamic circuit breakers can full-fill the its impact with the existing model and their features in effecting the application in micro service environment addressed Weerasinghe and Perera 2022. Designing a dynamic circuit breaker pattern with ability of static circuit breaker in changing the adaptiveness to enhance the adaptive configuration, tailoring services mesh with Istio in Kubernetes, Involving a creation of flexible design which can respond to real time traffic variations.

In this Project "Circuit Breaking" is a Crucial strategy in micro-service architectures which is designed to prevent multiple cascading failures and ensures system stability. Circuit breaking pattern involves monitoring services interaction and stopping the services when the system shows failures beyond certain defined thresholds limits effectively making Circuit Break, Isolating the failed services. This state of art of circuit breaking principle makes the system to contribute to overall requests which impact system resiliency, the circuit breaker ensures these failures do-not propagate and effects the complete system. This mechanism is Implemented with Static threshold limits and adjustments are made according to variable metrics based on system load, Offering robust protection against defined failure scenarios and monitoring overall health and performance in micro-service ecosystem.

Implementation of the designed Dynamic Circuit breaker pattern in real-world micro-service application BookInfo hosted on AWS EKS cluster referencing the Figure 1, Architecture diagram of the Project with complete comprehensive setup, demonstrating the theoretical implementation with practical demonstration integrating with dynamic circuit breaker for a book-info application in live micro service environment. Finalizing the dynamic circuit breaker pattern with evaluating the results obtained on load testing environments like Fortio, Chaos Mesh and visualizing them with observability tools like Prometheus and Grafana, Kiali. The complete evaluation would provide the empirical data on system performance and its resiliency in real time load testing environment on dynamic circuit breaker in place. To address the Dynamic nature of circuit breaker the deployment and evaluation is done for book info sample micro service application deployed on AWS EKS cluster with Service Mesh Istio on it, The following evaluation and performance analysis is obtained in accordance with the book-info application, following results may vary for different application, configurations and depending environment. This limitation can be settled by exploring various complex broader application, vast configuration and tuning the circuit breaker pattern in order to generalize the clear findings.

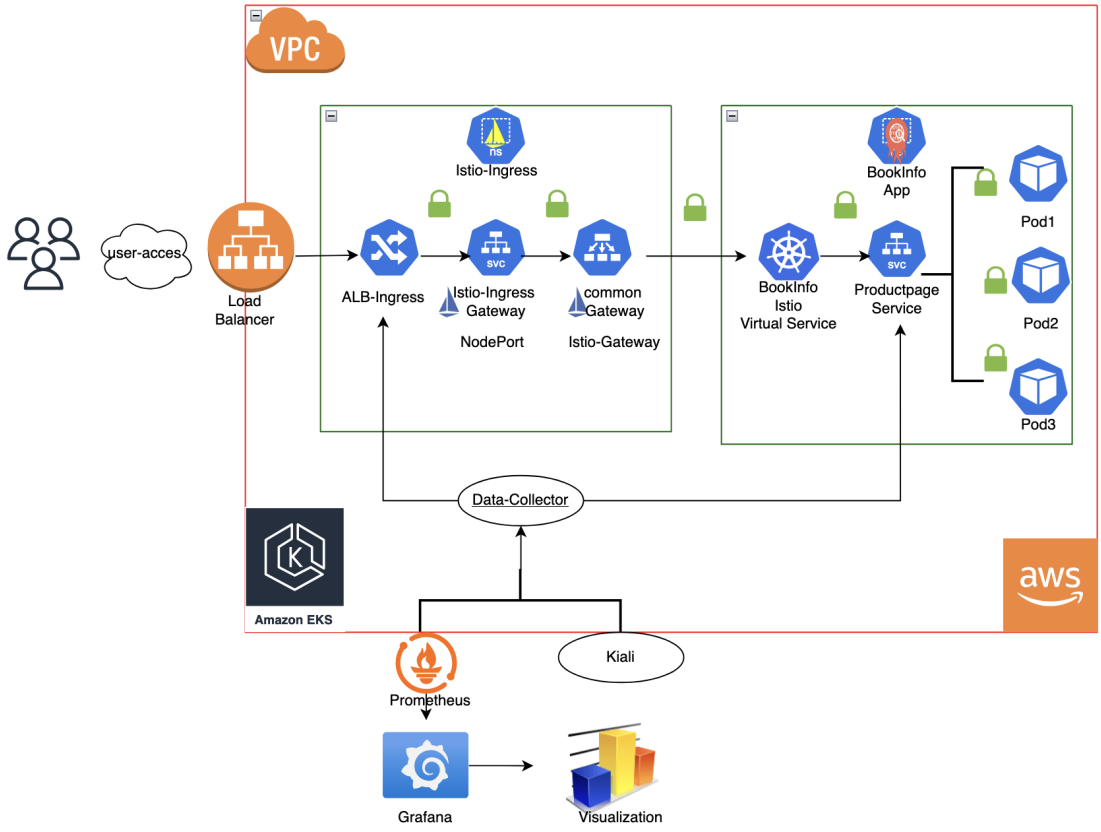


Figure 1: Architecture Diagram of the Project

This framework demonstrates how effective dynamic circuit breakers can significantly help in enhancing the system resiliency and stability of the micro service environment (Kurpad et al. 2023). As Major contribution of this research involved in development and validation of framework for dynamic circuit breaking principle in Kubernetes Istio service mesh Adaptation of existing Architecture into real-time dynamic setup help in accounting improvisation in Cluster management and communication of microservice complexities. Providing the empirical results to support its adaption in real-world scenarios into the system involving cloud native adaptive application development offering value addition in the field for its benefits in fault tolerance mechanism.

1.5 Report Structure:

The section is defined into each steps helping in breaking down the Research topic and distinctive developments are conducted on each topics, Literature Review on each familiar and related papers on discussing the existing gaps and future scope required,Followed by Research Methodology outlining the technique used and approach with the theoretical development and tools employed. Next section discuss about Design and Implementation specification performed on the research project involving architecture design and integration of Istio Service Mesh covers BookInfo application deployment and Destination rule creation for effective circuit breaking. With obtaining results of the implementation, Evaluation of the research Methods and assessing system performance and tolerance in building more resilient and reliable system by effective Dynamic Circuit Breaking strategy is done. The report concludes with evaluation outcomes and Discussion about the strategies and methodologies involved in the project and effective ways of obtaining results, sharing future scope and potential improvements that can be implemented in future research on this topic with Referencing all the cited sources and acknowledge of previous work.

2 Literature Review:

The Literature Review is done with the progress in Kubernetes Architecture development and evolution of micro-service deployment models and new inventions that made changes in cluster deployment ecosystem, below are the descriptive explanation on each paper discussed with the respective section helping in identifying the gaps between each research and current invention principles:

2.1 Container Orchestration using Kubernetes

The paper "Enabling HPC Workloads on Cloud Infrastructure using Kubernetes Container Orchestration Mechanisms"Beltre et al. 2019 providing the evidence of potential of using Kubernetes to manage High performance Computing(HPC)loads on Cloud environment. The authors also compares with other orchestration tools like Docker Swarm and bare-metal execution, focusing on configuring containerized applications, including the device access, inter-container communication and performance over the ethernet , the outcome of the performance Comparision showed Kubernetes and Docker Swarm can achieve near bare-metal efficiency in performance with RDMA communication, also Kubernetes incurs overhead with TCP/IP. The author practically helped in making decision with selecting Kubernetes and confirming HPC workloads and has different views in considering performance consideration across various enterprise needs.
The Paper "A Comparative Analysis of Container Orchestration Tools in Cloud Computing by Malviya and Dwivedi 2022" evaluates the performing strengths and weakness of four popularcontainer orchestration tools like Kubernetes, Docker Swarm, Apache Mesos and Redhat OpenShift. The author highlights the importance of study in containerization for managing applications with cloud native or derived application types, It assess these tools based on the scalability, Stability and security deployments with overall installation and maintaining a learning curve, The authors findings indicate that Kubernetes excels in scheduling and out performing, Docker Swarm is user friendly, and comparing overall achievability Kubernetes performs great in compared to all other Orchestration tools, from the overall analysis of author aims to guide professionals in selecting suitable orchestration tool based upon their specific custom requirement.
The study on Kubernetes Scheduler Optimization for the Real World helps in understanding enhancements to Kubernetes Scheduler improving resource utilization and load balancing for the containerized environmentsCheng et al. 2023. Compared to the default schedular prioritizes nodes upon resource requests and the rates, which eventually lead to resource wastage and uneven distribution causing discrepancies for the actual and requested resources. In this article the author proposes an optimized scheduler that works with real-time data, which was inspired by crane-scheduler from the open-source project Gocrane. By leveraging the real load metric monitoring, the new scheduling mechanism aims to enhance load balancing and resource efficiency across Kubernetes cluster, this paper provides the significance of implementing process and demonstrates the utility of cluster and load distribution through practical applications and result analysis.

The Study of "An Overview of Container Security in a Kubernetes Cluster by German and Ponomareva 2023" provides a great insight into understanding widespread adaptation of Container technology and need for robust technology measures for this vast growth in Kubernetes environments. The article also talks about challenges how it can be exploited by attackers to access sensitive data and infrastructure, the author has reviewed several papers on real-time security monitoring tools for Kubernetes, including Aqua Security, neuVector and many more, the author compares various models and their benefits of security principles they offered, in that AquaSecurity offers comprehensive security for containers, virtual machines and CI/CD pipelines with dynamic scanning capabilities. NeuVector focuses on vulnerability scanning, packet filtering and allowing zero trust approach with defense against zero day attacks and data leaks, the technology Sysdig Secure integrates cloud and container security enabling real time monitoring with forensics, leveraging detailed and user activity data to detect any potential threats to block them, this study also underscores the prime importance of integrating real-time monitoring tools in enhancing Kubernetes security and protect against Cyber threats.

2.2 Service Mesh Architecture

From the Paper "Analyzing and Monitoring Kubernetes Microservices based on Distributes Tracing and Service Mesh by Y.-T. Wang et al. 2022" will explore the the understanding of adoption of Microservice system architecture(MSA) from traditional monolithic architecture, focusing on improved maintainabil-

ity and observability of the architecture, Scalability and fault tolerance are the key areas author focuses on, The paper also talks about challenges in monitoring microservices, mainly with technological heterogeneity, which provides a nonintrusive monitoring solution, KMamiz which uses Istio and Zipkin for distributes tracing and Service mesh installation, KMamiz enables the construction of detailed service dependency with graphs, endpoints request chains and analysis of service study and coupling, enhance in the system quality and aiding in error identification, this study provides insights in demonstrating combining service mesh with distributes tracing can provide comprehensive, low-intrusion metrics monitoring systems with overall improving operational insights and system reliability.
The author tell us about the introduction of Service Mesh-Based systems for Deep learning Applications, where Load Balancing and Task Scheduling are the key focus areaby XIE and Govardhan 2020, Integration of Kubernetes Mesh with Istio to enhance the deployment strategies help in operating with deep learning applications, the article provides the practical approach using Kubernetes for container orchestration and Istio service Mesh integration to the system, By using Flask based deep learning model deployed into Docker container and Kubernetes Cluster, they enable the functionalities like load balancing, task scheduling and real-time metrics monitoring for resource utilization, the paper demonstrates vast improvements in scalability, reliability and efficiency for deep learning applications, strengthening robust load balancing and scalability provided by Istio and Kubernetes, with Comprehensive real-time monitoring enabled inbuilt, the study done on extreme variable load conditions are limited time interval and with tools for various conditions. Future works are suggested by the author to work on optimizing resource allocation and providing enhanced work models and system capabilities with more complex models helps in valuable insights for future research from this area.

"Compliance Validation in the Service Mesh Architecture byAlboqmi and Gamble 2024" working with dynamically validating compliance in microservice architecture (MSA) applications, The proposed method integrating Kubernetes and Istio Service Mesh with compliance validation capabilities facilitating dynamic adjustments of access control policy, Author also depicts the picture with experiments conducted on real time picture, This approach involves Open Security Controls Assessment Language (OSCAL) and trestle, this OSCAL-based software development kit would automate the compliance management, including MAPE-K (monitor-analyze-plan-execute-knowledge a typical control loop with self adaptive functionality with providing real-time compliance validation involved. The key important results demonstrated the feasibility an effectiveness in integrating compliance validation with Service Mesh, involving security and governance of MSA applications, this paper mainly focuses on innovation in use of service Mesh for Dynamic Compliance and practical evaluation with the real world problem, the author however also gives information about the limitations of the system including the complexity and time consuming nature in adopting OSCAL, some of the challenges like static policy enforcement and runtime adaptability, author also gives future work to proceed with optimizing compliance validation processes and extending the approach for more complex MSA environments which can have impact on present traditional approaches.

Research paper "A Web Based Orchestrator for Service Function Chaining in a Cloud-Native Environment by Z. Wang et al. 2022" provides great insights of advanced Network Function Virtualization(NFV) by proposing a web based orchestrator for deploying Service Functions Chaining (SFC) with Cloud –native Network Functions (CNF) within a Kubernetes Cluster using Network Service Mesh (NSM), The author provides a SFC framework with cloud-native which allows users to dynamically create SFCs with container based systems rather than opting traditional practices like VMs with inbuilt NFV/SDN approaches. Provided with the Python based front end web server connected to the Kubernetes API and Prometheus for traffic validation, the system enables dynamic SFC creation by drag dropping preconfigured services in container, with deploying in Kubernetes manifest files, The overall results highlights the system ability in providing reliable SFC path and collect performance metrics providing minimal overhead with Prometheus, Strengths from the study include for its dynamic SFC creation and comprehensive multiple technology ability, working with potential performance bottlenecks, the author also provides future work to focus on optimizing cluster Orchestration system with high validation mechanisms for further ensuring network reliability and efficiency in the system.

2.3 Istio Service Mesh and Circuit breaking pattern

From the paper "A Micro-Service Tracing system based on Istio and Container Technology" the author Song, Liu and E. 2019proposes a tracing system for the container systems using micro-services in

Kubernetes and Istio service Mesh with minimizing code intrusion among the components. This approach uses proxy layer with intercepting and process tracing information, which reduces the need for changing in the original code, implementing Google Dappers trace mechanism and span methodology, the system effectively highlights the dependencies involved with services and call duration. The results demonstrate the tracing system which has little impact on original systems efficiency and performance providing much high insight into services interactions. Strengths of the system include easy integration and cross-language compatibility, however the main challenges involved in managing with potential challenges in complexity with implementing the proxy layer and maintaining metric monitoring, the author also proposes with the future work to focus more into optimizing the proxy layer and extending an arm support for larger for more complex micro service architecture.

From the paper "Managing Multi-Cloud deployments on Kubernetes with Istio, Prometheus and Grafana" as the author Sharma 2022explores the benefits in Kubernetes with Istio service Mesh in managing multi-cloud deployments, with Metrics Monitoring like Prometheus and Grafana for Visualization. This setup allows administrators to effectively manage multi-cloud deployments, reducing the single vendor dependency while enhancing reliability and availability. The overall results highlighted the main advantages of multi-cloud environments, increased in innovation, risk mitigation and minimal vendor lock-in policy, with built in increased complexity and integrating higher implementation costs. With the built in Observability tools like Prometheus and Grafana with the potential weakness of involving higher complexity and cost of managing multiple cloud services. The author proposes future work could focus on further optimizing multi-cloud integration and exploring additional tools for streamlining process management and advanced monitoring process.
The most relevant paper "Boosting Micro-service Resilience: An Evaluation of Istio's Impact on Kubernetes Clusters Under Chaos" Author Singh, Muntean and Gupta 2024 have clearly investigated the role of Istio Service Mesh in maintaining in resilience of the Kubernetes deployed, with variable conditions in load from Chaos mesh, with investigating how Istio improving system stability and fault tolerance with injecting failures and understanding the metrics by analyzing metrics like response time, error rates, resource usage and requests per second. This process involves deploying micro-services in a Kubernetes cluster with Istio and subjecting them to chaotic tests for measuring its system resiliency. The author explains how Istio provides results in improvements in system stability, offering better response time with resilient to failures in comparison to traditional architecture practices, which provides a Istio a ideal choice in effectively managing micro-service communication with enforcing its policies, with its robust performance under variable environment.
"Breaking the Vicious Circle: Self Adaptive Micro-service Circuit Breaking and Retry"Saleh Sedghpour et al. 2023 with very informative explanation on system resiliency for its efficiency in making calculative decisions in terms of retry mechanisms and circuit breaking policy, with service mesh architecture, the paper sensitively talks how adaptive retry and circuit breaking mechanisms works in various scenario for making a stable system, thus development of dynamic retry controller helps with adaptive circuit breaking pattern, the controller adjusts the retry attempts and scenario for transient workloads which result in enhanced throughput up-to 12X which is compared to static configurations, The system dynamic controller ability and implementing its architecture increases the robust performances, the author speaks about future work about focusing on optimizing controller and exploring its application in more diverse micro-service systems with dynamic approach in real time, which is taken as a part of our project methodology taken from it.
From the author Rupesh Raj and Others has given "Automated Testing and resilience of Microservice's Network Link using Istio Service Mesh" where the paper by Karn et al. 2022 has described about the systematic resilience testing of the environment with Istio Service Mesh and this uses unique approach in micro-service communication between them and monitor communication for automated testing and various resilient strategy, the paper shown that they have injected various fault injection technique provided by Istio, also with Locust load testing tool, to significantly simulate micro-service loads and identify faulty links. To visualize the framework it uses Grafana and Jaeger dashboards to identify the issues and monitor the changes, the technique involves in deploying redundant services, and implementing service scaling with circuit breakers for mitigating network congestion, with using the setup Hipster shop e-com application deployed in Kubernetes Cluster, the author speaks about various strengths and weaknesses which include detailed monitoring with fault injection capabilities of Istio, with challenges involve the complexity of setting up and integrating with multiple tools, the future works given by author should focus on optimizing automated resilience mechanisms and exploring various applicability with diverse micro-service environments.

Research Paper	**Existing Research-gaps and Our Research Focusing on Research Gaps**
An Overview of Container Security in a Kubernetes Cluster by German and Ponomareva 2023 German and Ponomareva 2023	Author focuses on Kubernetes Security and applied principles in storing sensitive data. Secure Micro-service communication with Advanced mTLS certificate technology , Istio Service Mesh with inbuilt security helps with its Citadel, TLS certs, and Advanced Security
Analyzing and Monitoring Kubernetes Microservices based on Distributes Tracing and Service Mesh Y.-T. Wang et al. 2022	Author focuses on improved microservice in communication with Istio Service Mesh, less focus on Istio's Observability features ,Observability in Istio Service Mesh, Kiali-Grafana-Prometheus
Service Mesh-Based systems for Deep learning Applications XIE and Govardhan 2020	Existing Research talks about Deep learning usage in microservice architecture, did not focus on Performance improvement in the real-time MSA ,Our Research on Istio Service Mesh with gaining maximum resiliency and Reliability focuses on performance as a factor.
A Web Based Orchestrator for Service Function Chaining in a Cloud-Native Environment Z. Wang et al. 2022	Author focuses on NFV (Network Function Virtualization) improved architecture approach for Cloud-Native application using Service Mesh , Our Research Focuses on Future work of the Research in enhancing the Cloud Native application with advanced networking strategies
A Micro-Service Tracing system based on Istio and Container Technology Song, Liu and E. 2019	Author Clearly explains the strengths in implementing Istio Service Mesh and challenges in exploring the Service Mesh integration , This Project works on different approach in finding the research objectives and attain expected results with Istio Service Mesh
Managing Multi-Cloud deployments on Kubernetes with Istio, Prometheus and Grafana Sharma 2022	Author Focuses on Benefits of Istio Observability features in integrating Prometheus-Grafana for Visualization , Our Research includes practical demonstration of Observability features enhancing Visualization of the Architecture in Real-time
Boosting Micro-service Resilience: An Evaluation of Istio's Impact on Kubernetes Clusters Under Chaos Singh, Muntean and Gupta 2024	Author explains the role of Istio Service Mesh and its Mechanism in maintaining Reliable system under Fault tolerant conditions , Author Failed to explain Real-time Monitoring system with variations in Load like Fortio Integration along with Chaos Mesh
Automated Testing and resilience of Microservice's Network Link using Istio Service Mesh Karn et al. 2022	Author works on filling earlier gaps on Research, Introduces Istio Service Mesh with its advanced capabilities and its automated approach , Author explains the automated approach of the Istio Service Mesh, but fails to address its effects on maintaining Dynamic Circuit Breaking strategy and its policies

Table 1: Table Representing Literature Review of Existing Research conducted on Micro-service and Istio Service Mesh Architecture.

3 Research Methodology:

The Methodology involves practical approach with structural way to setting the deployment and evaluation of the project. Referencing the architecture diagram from figure 1: Setting up the AWS EKS cluster using eksctl, followed by configuring kubectl for Kubernetes cluster management, deploying the BookInfo application consisting multiple micro-services is deployed with side car injection enabled for managing inter service communication and maintaining the traffic balance.

3.1 Istio and Its Capabilities in Real-Time Traffic Handling:

Stand-out feature of Istio Service Mesh for its ability to make inter-service communication by utilizing Envoy proxy. Envoy proxy which is high performing proxy managing all inbound-outbound traffic of services inside mesh. Supporting dynamic service discovery, Load-Balancing, TLS termination,HTTP/1 and HTTP/2,gRPC proxy, health checking and stage rollouts in traffic splitting with its important feature of circuit breaking. This extrordinary features sets complete control over traffic routing and spliiting services, ensuring security and reliable communication in services. the figure2: representing Istio Service Mesh Envoy proxy integration with control plane for interservice communication.

Selecting Istio Service Mesh remained an ideal choice for its more comprehensive security features with mutual TLS and zero-trust networking, fine grained access control came into accountability. Excelling in high Observability, distributed tracing, metrics logging are very crucial for monitoring and debugging in complex microservice communication. Istio with its ability to simplify microservice management, consistent and powerful set of tools in traffic management, Side-car injection approach are the highlighted features of integration, The ability of Istio Service Mesh with its interservice communication and without requiring to change the code makes Istio flexible and powerful solution tool for Dynamic adaptation tool.

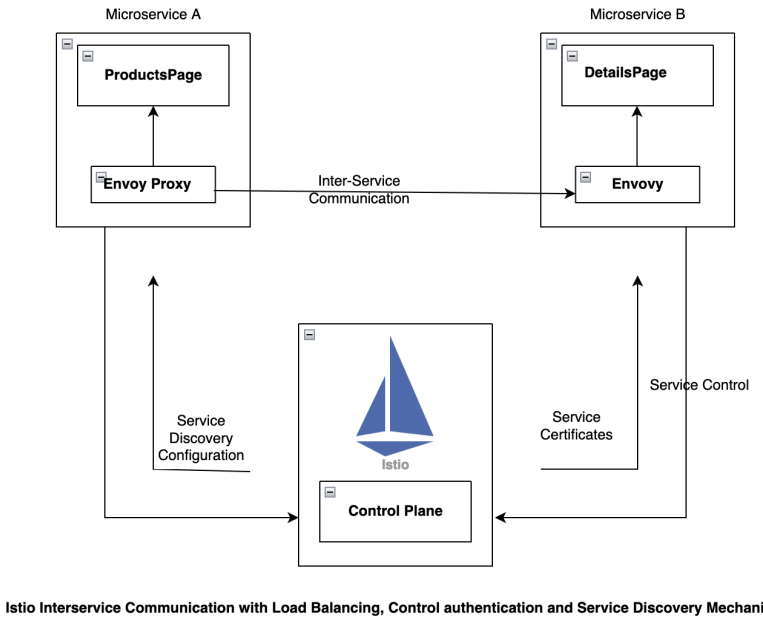


Figure 2: Architecture Diagram of inter-service communication and intelligent mechanism in Istio

As the main objective of the project is to show the effectiveness of circuit breaking principle Istio Service mesh, the Destination rule is configured to for effective circuit breaking technique within the services, with specifying the policies for connection pool and outlier detection. Introducing the external load to the application deployed on EKS cluster to check the performance of the system by external load testing tools like Fortio with variable traffic conditions, engaging continuous metric monitoring with Prometheus and Visualizing the incoming load to the Istio Service Mesh by Grafana for overall error rates and latency report for the real time analysis, the ability of the system for its self-healing capability and resilient system is measured and sensitively analyzed with collecting its defined release pattern of

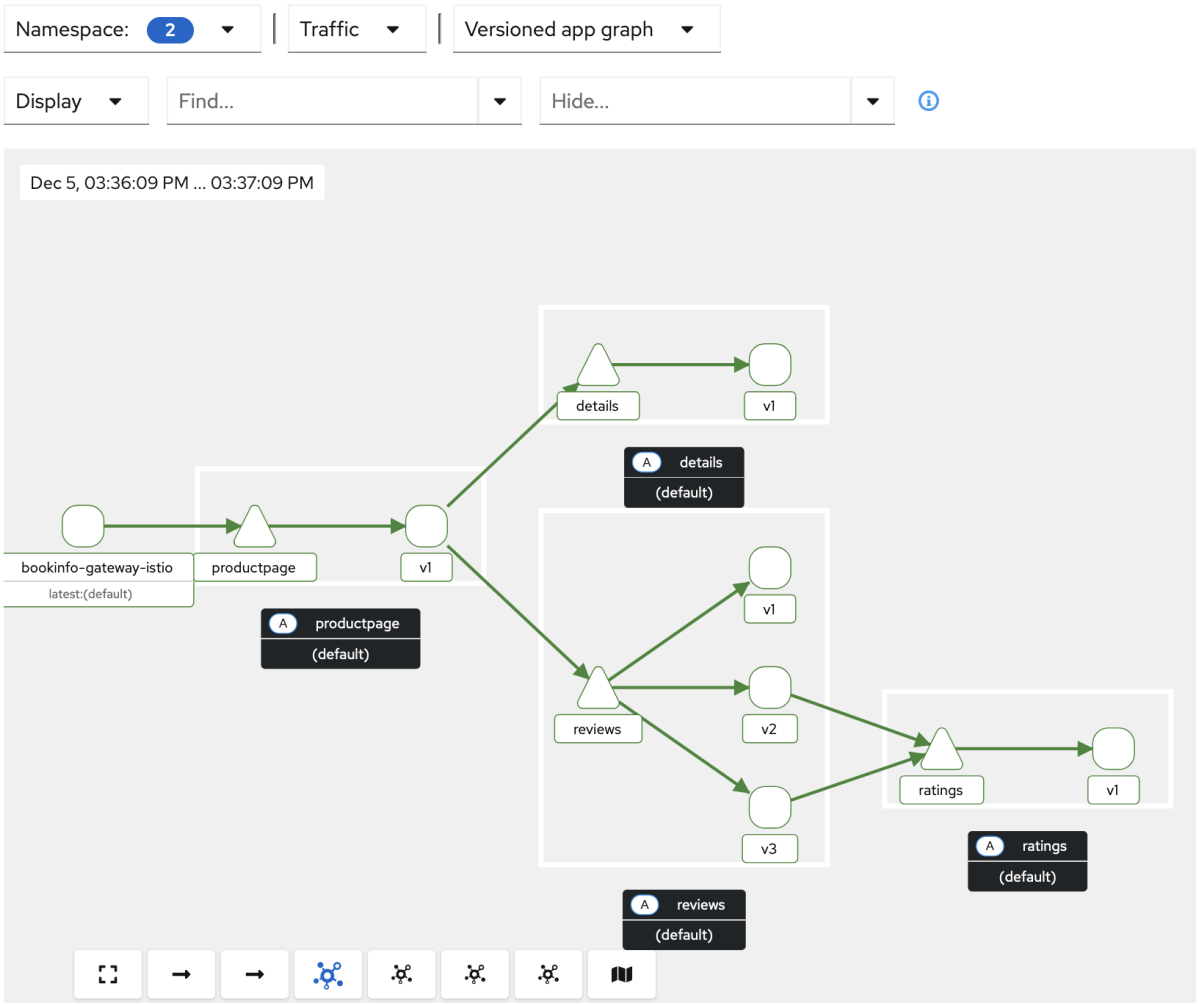


Figure 3: Internal Architecture of BookInfo Micro-service Application

load, this comprehensive methodology is aimed to enhance the system resiliency and improve performance for the micro-service communication within a Kubernetes cluster mesh environment.

From the Figure3:Above, The internal architecture diagram of BookInfo Application in Istio Service Mesh with an exemplary demonstration for managing micro-services, The application mainly consists of 4 distinct micro-services: ProductPage, details, reviews and ratings pages, Each micro-service functions are deployed in separate pods isolate from each other, the ProductPage is the frontend of all the pods, which aggregates all the data from details, reviews and ratings pods which are present in BookInfo application. Each pods have a distinct features while the details microservice provides additional book information, reviews provide services to offer customers reviews with ratings in combining connected feature to this microservice, This interconnected microservice architecture in Kubernetes cluster demonstrates how microservice communicate seamlessly with Istio Service Mesh, facilitating efficient fault tolerance, traffic management and Observability features.

3.2 From Selection to Execution:

Implementation of dynamic circuit breaking rule for enhancing the reliability and resiliency of Kubernetes microservice application in Istio Service Mesh configuration. This approach of developing system architecture for automatically detecting and isolate faults in the systems process, continuous operation in stress conditions withstands by its fault tolerant principle of circuit breaking, transition from static rule to dynamic circuit breaker principle helps in dynamically adaptive environment real-time, The testing of the system with Fortio for load testing shown to break with existing static circuit breaking rule, where the system fault tolerance is evaluated, With Istio's Service Mesh which helps in understanding dynamic circuit breaking preventing cascading failures and self-healing mechanism is maintained, thereby guar-

anteeing the system resilience nature.
Fortio's effective load producing strategySingh, Muntean and Gupta 2024 simulates high traffic trends and collects detailed metrics such as latency, throughput and error rates. This simulation enables us to produce real-time system monitoring experience for evaluating crucial scalability and performance. With the effective load bearing capacity system bears to a certain limit and breaks at point when the static rule breaks, which cannot sustain the real time load metrics, thus creating a open state configuration disallowing all the requests and increasing the pending request section, Thus the need for dynamic conditions and effective efficiency is at most priority to the architecture creating a robustness in the micro-service system, configuring Destination Rule , with effective fault isolation and helps in preventing cascading failures by outlier pre-configured. As this rule set a criteria-based setup on when to break circuit and trip on reaching number of failed requests or exceeding limit in threshold. With the dynamic approach gives fine grain tuning on performance of maintaining on all the request to optimum level helping in quick recovery of the services and set to half-open state of incoming traffic configuration.

From Selection of Istio Service mesh circuit breaking technique with its dynamic ability to execution of this framework, dynamic circuit breakers have presented a great ability in maintaining the system resiliency and providing a reliable solution which has the capability to respond to real time fluctuating conditions, ensuring robust and resilient micro-service environment.

4 Design Specification:

The essential requirements and equipments for this project are as follows:

4.1 AWS EKS cluster:

AWS EKS (Elastic Kubernetes Service) is a managed Kubernetes service platform which simplifies the process for running Kubernetes setup on AWS Public cloud platform without the need for installation and operating On-premises cluster and managing control plane. EKS with built in capability to automate key tasks as patching, provisioning node and cluster scaling making the sure the high availability and secure Kubernetes environment. Using AWS CLI tool eksctl, creating and managing EKS cluster was made easy, with deploying instance in a cluster by giving required necessary commands specifying region, node type and size of the node groups. Created cluster using kubectl command line tool is used to interact with Kubernetes, which allows to manage cluster with effectively enabling doe deploying application and manage other resource, allowing monitoring cluster health. Combination of both eksctl and kubectl command streamlines the operational management of Kubernetes Cluster, making it easy process to leverage the Kubernetes on AWS for its scalability and resiliency of the building application with it.

4.2 Istio Service Mesh:

Istio is an open-source mesh which acts an additional infrastructure layer with its robust technology in managing micro-services by simplifying the complexities involved in inter-service communication allowing features like network proxying, sidecar injection and circuit breaking, the need for Istio Service mesh arises from associated challenges in managing large scale container deployment's in micro-service, mainly traffic management with benefits of security and observability, With Side car injection where Istio automatically injects Envoy proxy for each pod in handling network functions such as load balance, network traffic routing, security establishment without effecting application code. this project involved huge traffic incoming spikes which makes the system to showcase the circuit breaking technology which prevents the cascading failures as each micro-service is interconnected, with its intelligent technology for isolating its unhealthy services which is effective reason for choosing Istio Service Mesh for their benefits in high availability, resiliency and streamlining complex operations.

4.3 Metric Monitoring:

Metrics monitoring and observability for the Isto Service Mesh traffic is significantly enhanced by tools Kiali, Prometheus and Grafana, As Istio integrates effortlessly with Prometheus for gathering detailed metrics data produced by the Istio in-terms of performance and network traffic, which then be integrated to Grafana for visualizing with comprehensive dashboard and help in monitoring performance indicators like error rates, latency and request volumes, Kiali enhances the in-depth insights architecture of service

mesh's role in the BookInfo application for its internal working, with integrating all these tools create powerful framework which help in real-time monitoring and observability, helps in managing application traffic and network performance for maintaining continuous reliability and resiliency of the micro-service architecture system.

4.4 Load testing:

This project uses Fortio and for performing comprehensive load-testing for the BookInfo application deployed under Istio Service Mesh, As Fortio is a Versatile tool simulating high traffic loads and allows to collect detailed performance report in metrics like Service latency, throughput and error rates, this offers real-time system monitoring when encountered under stress, enabling the efficiency of micro-service scalability and performance.This Methodology includes configuring static rule and Dynamic rule with effective breaking strategy in Istio, leveraging the use of real-time metrics monitoring and making traffic policies dynamically. Fortio load testing tool provides real insights on how these load effects on resiliency and reliability ensuring the service mesh handles real-time workloads.

4.5 Istio Service Mesh and Destination Rule

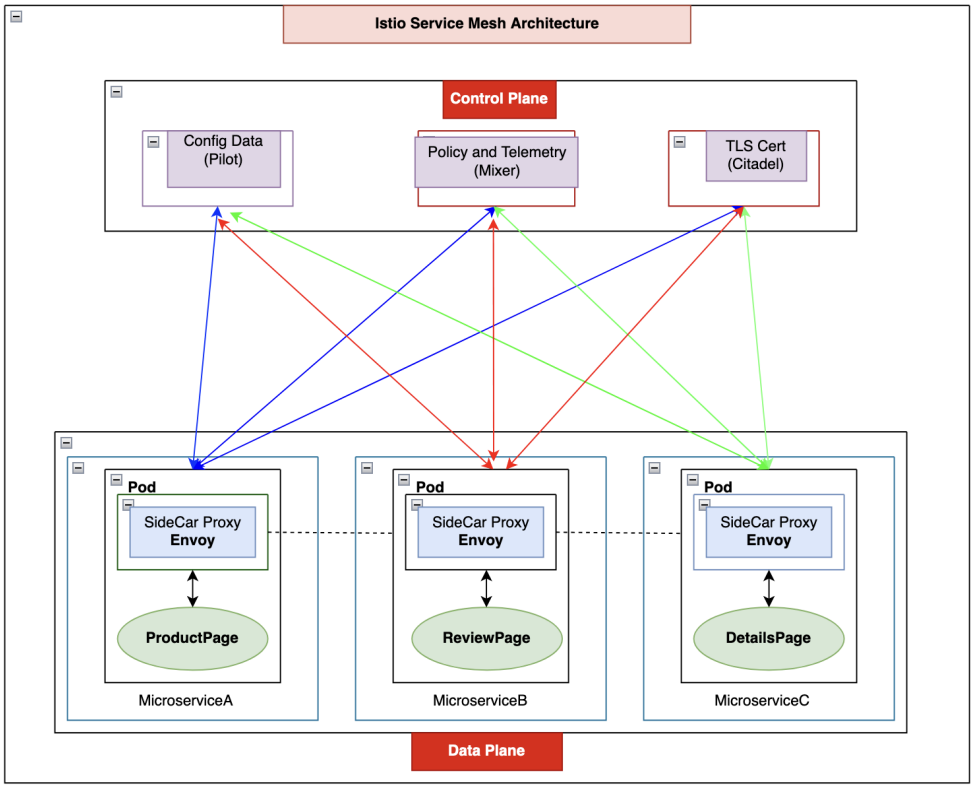


Figure 4: Istio Service Mesh Architecture

From Selection of Istio Service mesh circuit breaking technique with its dynamic ability to execution of this framework, dynamic circuit breakers have presented a great ability in maintaining the system resiliency and providing a reliable solution which has the capability to respond to real time fluctuating conditions, ensuring robust and resilient micro-service environment. Above figure 4 is the descriptive representation of Istio Service mesh architecture inside Kubernetes cluster pod deployment. With Dynamic Circuit breaking principle framework that underlie the implementation of Service mesh in Microservice communication, that adaptiveness of Circuit breaking rule which helped to be more fault tolerant of the system to be more effective than with the static circuit breaking rule implementation by looking at the Istio Service Mesh architecture.

Istio service mesh with indispensable managing microservices and its robust architecture and advance

features, The Istio service mesh comprises of control plane, component of control plane including Pilot for service discovery and routing service, Mixer used for Policy control and Citadel component for security, the data plane consists of Envoy proxy component it is added to all the pods in the cluster, helps in managing flow and enforcing policies, As in our project implementation the most powerful component of Istio service mesh is dynamic circuit breaking with its capabilities to adapt real time changing traffic conditions and service health.

Fortio's effective load producing strategySingh, Muntean and Gupta 2024 simulates high traffic trends and collects detailed metrics such as latency, throughput and error rates. This simulation enables us to produce real-time system monitoring experience for evaluating crucial scalability and performance. With the effective load bearing capacity system bears to a certain limit and breaks at point when the static rule breaks, which cannot sustain the real time load metrics, thus creating a open state configuration disallowing all the requests and increasing the pending request section, Thus the need for dynamic conditions and effective efficiency is at most priority to the architecture creating a robustness in the micro-service system, configuring Destination Rule , with effective fault isolation and helps in preventing cascading failures by outlier pre-configured. As this rule set a criteria-based setup on when to break circuit and trip on reaching number of failed requests or exceeding limit in threshold. With the dynamic approach gives fine grain tuning on performance of maintaining on all the request to optimum level helping in quick recovery of the services and set to half-open state of incoming traffic configuration.

```
manukumarkc@Manus-MacBook-Air ~ % kubectl apply -f - <<EOF
apiVersion: networking.istio.io/v1
kind: DestinationRule
metadata:
  name: httpbin
spec:
  host: httpbin
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 1
      http:
        http1MaxPendingRequests: 1
        maxRequestsPerConnection: 1
    outlierDetection:
      consecutive5xxErrors: 1
      interval: 1s
      baseEjectionTime: 3m
      maxEjectionPercent: 100
EOF
```

Figure 5: Pictorial Representation of Destination Rule with Static Circuit Breaking Configuration.

Unlike principles in static circuit breakers where relying on constant thresholds dynamic circuit breakers continuously monitor the system and adjust according to real time, preventing cascading failures and making sure the seamless operation of service mesh responsibility, Tools like Fortio plays a crucial role in setup for generating realistic traffic patterns to load testing the services, Selection of Istio Service mesh with dynamic circuit breaker is chosen over static circuit breaker for its superior ability in adaptability and responsiveness for traffic conditions, ultimately allowing the system architecture with robust and resilient micro-service environment. This ability of dynamically adjust and self healing role is making an ideal choice for Site reliability engineers to choose Istio for maintaining high reliability in complex Kubernetes deployment architecture Sedghpour, Klein and Tordsson 2021.

The Image in Figure 5, provides the clear representation of Destination Rule defined for the Istio Service Mesh to BookInfo application apply to traffic intended services when routing has occurred. Maintaining

the specified configuration with load balancing, Sidecar connection pool size with Outlier detection settings enabled for predicting and evict unhealthy hosts from the load balancer connection. The Key components of Destination Rule written in YAML file are as follows:

Host: Defines the service to which rule is specified.
Traffic Policy: defining parameters for load balancing, Connection pool settings, outlier detections for circuit breaking settings.

Connection Pool Settings:
1)tcp:maxConnections: this sets the maximum number of TCP connections baseline setup, this value can be Dynamically adjusted based on metrics produced by real-time monitoring on Prometheus.
2)http:http1MaxPendingRequests: this Defines the maximum pending request from HTTP/1.1 version request, Adjusted based on Monitoring metrics.

Outlier Detection:
1)Consecutive5xxErrors:pod returning more than set threshold of 5xx errors will be considered unhealthy.
2)Interval: The Rule checks every period of set interval.
3)BaseEjectionTime: The pod set to eject for 30seconds, if it has seemed unhealthy.
4)MaxEjectionPercent: Rule that is set to maximum percentile of pods which can be ejected any time Simultaneously.
with Deploying all the Services and connection to Istio Network Management, the metrics monitoringY.-T. Wang et al. 2022 is observed in real-time with CPU usage rate and incoming traffics and Memory usage for time interval this is shown in the figure 6: Below.

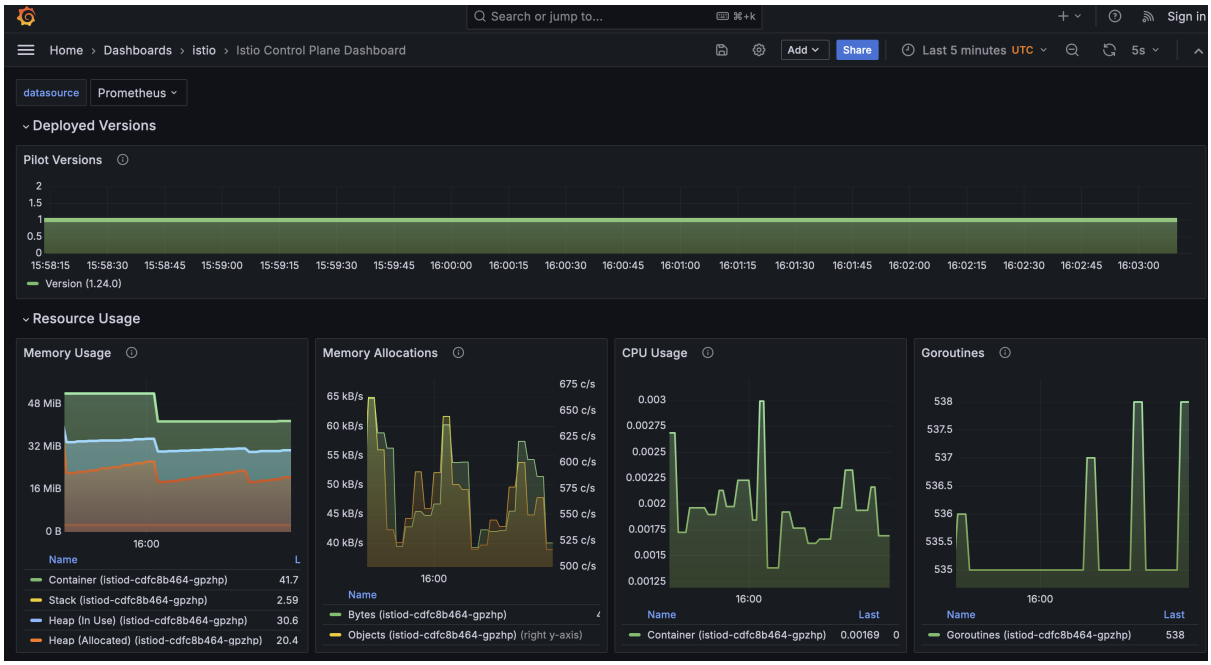


Figure 6: Real-Time Monitoring with Grafana Visualization

Istio Service Mesh with its advanced observability enables real-time monitoring harnessing the capabilities of Prometheus and Grafana for ensuring data collection and Visualization on system performance. Prometheus a powerful metric collection tool, collect data from various components including microservices, Istio Side car poxies and istio control plane,This data collecetd include resource utilization, request latencies and error rates. Visualizing in Grafana which is known for its flexible and dynamic dashboards, the representation of data , transformation and raw metric insights on graphs and charts.

The Real-time Metrics is viewed in Istio control plane dashboard available in Grafana, Comprehensive view of health and performance highlighting the metrics related operation of Istio components such as Pilot, Mixer and Citadel. Along with the visualization operator which tracks the traffic flows, detecting anomalies and gain insights of service mesh behavior in real-time actions, Optimizing resource allocation

and maintenance in reference system reliability. Through Grafana integration to Istio Service mesh gives advance monitoring dashboard which is available in the above figure 6.

5 Implementation and Solution development:

Implementing Istio service mesh in our project is the pivotal part with providing robust infrastructure support for managing micro-services, The critical feature in implementing dynamic circuit breaking principle for destination rule in the service mesh leveraging the critical benefits of service mesh operability, relying on predefined thresholds and flexibility of response rates for the current generating traffic conditions and service health in real time, ensuring the faults are kept isolated and preventing cascading service failures from the faults in internal communication helps the system maintain its resilience and reliable architecture for micro-service system in introducing load testing and fault injection.

5.0.1 Fortio Load testing Tool:

Load testing tool Fortio is a comprehensive tool which is designed specifically to test the microservice architecture allowing a precise load distribution on multiple pods. The Fortio pod uses QPS- Query-per-second rate which maintains consistent load in testing period. Distributing load on every pod evenly ensuring controlled and balanced testing environment, By this load testing approach enables detailed collection of performance metrics, latency, throughput and error rates are depicted from this, clear picture of systems behavior under varying load conditions are given from the Fortio system. Thus with load testing tool Fortio, helps in identifying potential bottlenecks and optimize resource allocation which makes it as asset for tetsing performance in distributed load system.

5.1 Scenario 1: Fortio Load Testing: Static Circuit Breaking

Load testing by Fortio Pod, where static rules are defined as set by Istio mesh rules (Figure5) with tcp:MaxConnection:1 ,MaxPendingConnection:1and Consecutive5xxErrors:1 where the circuit is set to break if the defined rules are passed by incoming traffic, Load to this request involves by setting up continuous 100 connections by MaxConnection:1 continuously no changes in the Circuit and the Web Application runs without any failure,

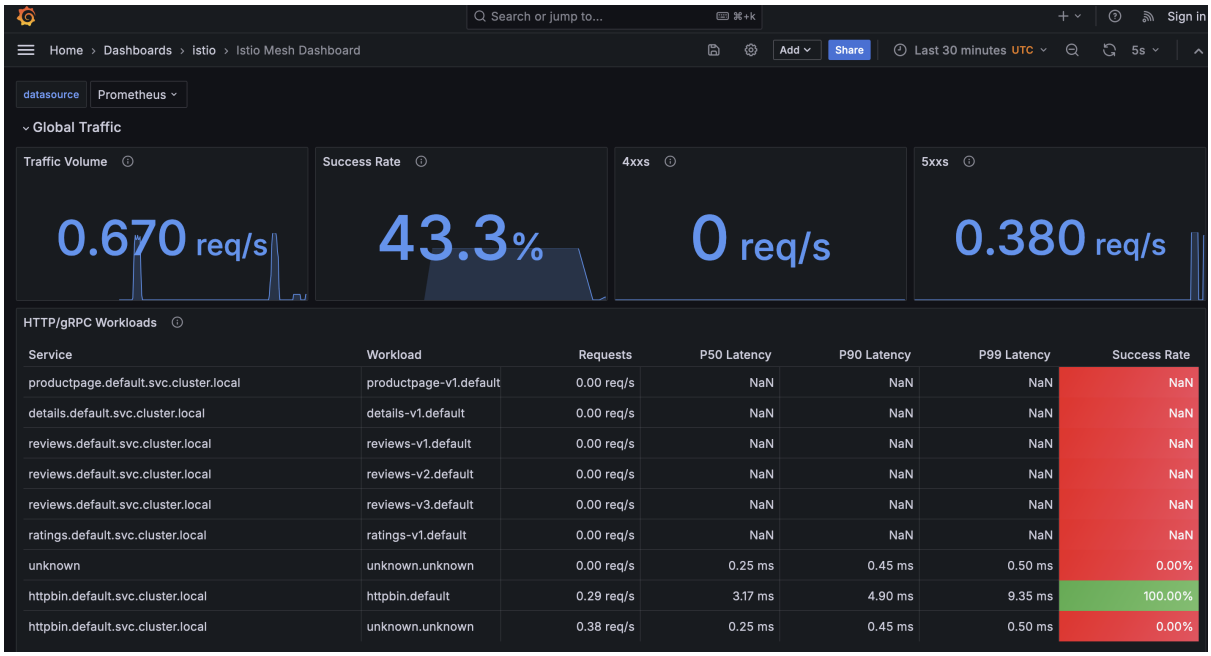


Figure 7: Real-Time Metrics Monitoring of Static Circuit Breaker

Load testing by Fortio Pod, where static rules are defined as set by Istio mesh rules (Figure5) with tcp:MaxConnection:1 ,MaxPendingConnection:1and Consecutive5xxErrors:1 where the circuit is set to break if the defined rules are passed by incoming traffic, Load to this request involves by setting up

continuous 100 connections by MaxConnection:1 continuously no changes in the Circuit and the Web Application runs without any failure, Next Fortio loading by executing command to produce 20 requests with 2 concurrent connections which set to put load on the application from AWS ELB loadBalancer connection by Fortio Pod, which breaks the circuit for –c2 and 20 requests and the MaxPendingRequests increases in time, on increasing the load with producing 30 requests on 3 concurrent requests at every connection will depict the exact real-time application environment which is facing high spike of requests to the website, where the BookInfo Application deployed here with broken circuit will continuously disallow the traffic to pass through as the Breaker is Open and all the requests are set to pending, which is set to Increase the MaxPendingRequest, During this transition Circuit is tripped to prevent further failing of the services with consecutive 5xx errors, This system effectively exhibits and isolates the faults making system resilient with new healthy pod created in a open state loop of Circuit breaking and the results of continuous testing are showing in the table

5.2 Scenario 2: Fortio Load Testing: Dynamic Circuit Breaking

In this Implementation step, Viewing the Results of Static circuit breaker Adjusting the Dynamic principle by considering several factors such as **historical data, traffic patterns, Service health, Compute Capacity and adaptive principles in Real-time monitoring** were considered to re-define the circuit breaking rule in Destination YAML file, with necessary adaptive changes in according to Fortio load testing scenario which simulate exact real-time load for application, the configuration change in Dynamic rule helps in implementing, enhancing the system reliability and Fault tolerance capacity to the microservice,As Fortio Load is distributing continuous load in -c2,-c3,-c4,-c5, the compute capactiy of the node is t2.medium size of cluster, Collecting al the effecting features and creating Destination Rule according to continuos testing of predictable load, we changed the tcp:MaxConnections:2, MaxPendingRequest:2 with variable required system reliability, Outlier detection of Consecutive5xxErrors: 5 as per the allowable limit per active request, Showing a tremendous result in providing no errors in system connections for –c1 for 100 consecutive request, since the load was static and was not high in traffic, And same result are obtained as the request for –c2 with 20 requests where the circuit did not break because of new **Destination Rule** created, On increasing the load with Fortio –c3 and 30 Requests the circuit

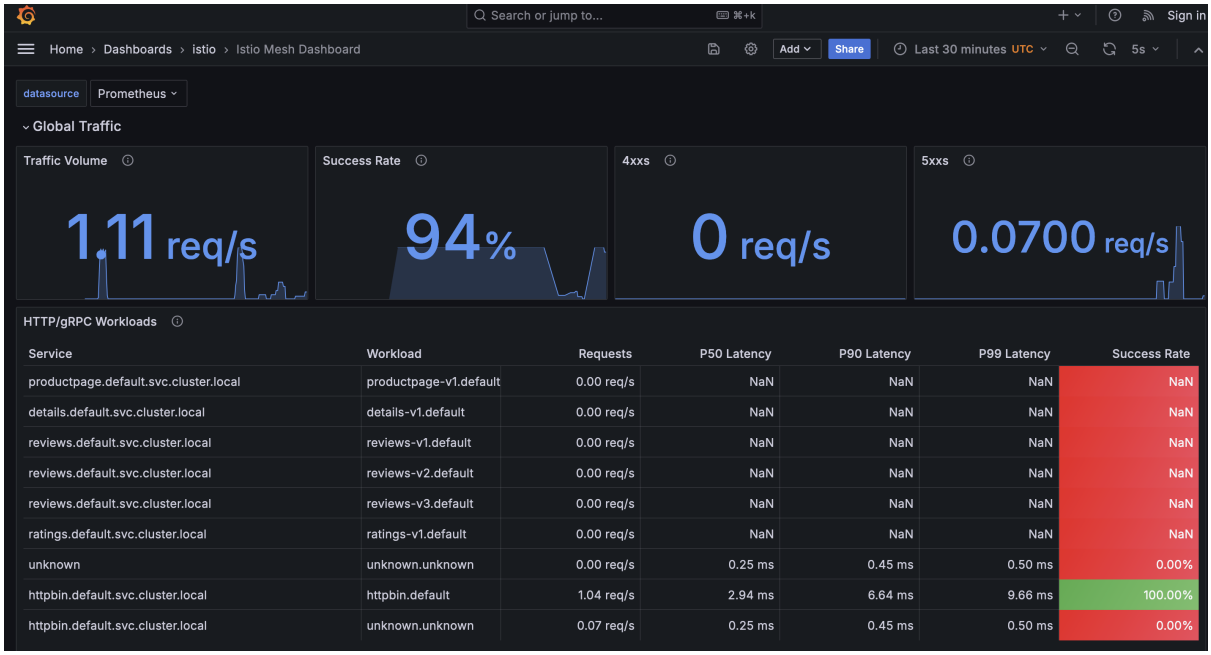


Figure 8: Real-Time Metrics Monitoring of Dynamic Circuit Breaker

breaks and shows the principle of Dynamic Adaptive nature, The Dynamic Circuit Breaker detects the threshold limits and trips when exceeds, but it breaks the circuit with half-Open state which permits certain limits per request until all the services and requests are recovered. This effective measure in tripping the circuit with half-Open state and allowing limited number of requests for BookInfo application, this intelligent mechanism of Istio Dynamic rule help to set Resilient and avoid Fault and provides more

Reliable system architecture, The results obtained from the Dynamic Circuit Breaker rule is discussed in the results Section with comparative analysis with Static rule.

6 Results and Critical Analysis:

On Evaluating the implementation of Fortio load testing for the Istio Service Mesh which yielded in the significant results into microservice architecture performance and its nature to get resilient, with Fortio simulating high traffic loads, Analyzing real-time traffic metrics as latency, throughput and error rates, with the help of these metrics the system robust performance revealed the results for variable traffic load,

In the table 2: The consistent pending request from static circuit breaker from normal load of –1 100 request to –c5 50 Request describe the nature of circuit breaker in being closed state, Meaning the closed of the circuit breaker not accepting any more requests and increasing the count for PendingRequestPerConnection which increases the rate of delay and error rates displayed to Clients accessing the application, this marker indicates the need of Dynamic principles in which during variable conditions adaptive environment for circuit breaker in decreasing the PendingRequestPerConnection , As Kubernetes AutoScaling Principles takes time to attain Horizontal Pod Scaling (HPA) with input allowing to increase pod count from Load Balancer actions, The count for number of PendingRequest remain unattended, Dynamic Circuit Breaking Rule with increasing the count for Maximum TCP:maxConnections and httpmaxPendingRequest to the Real-time Metric count, with Increasing the count of OutlierDetection: Consecutive5xxerrors to the reasonable count will with stand the variable environment, This changing in Destination Rule mechanism proves the need for Dynamic nature of Circuit Breaker. collective results obtained from table 1 shows the effective strength of change in rule of Circuit Breaker.

The below tabular column represent the results performed for both the Implementation on Static Circuit Breaker and Dynamic Circuit Breaker, with its breaking point note down, In the Comparitive Analysis of Circuit Breaking pattern, the Implementation of Dynamic Breaking principle showed us a better results and has less persistent PendingRequestPerConnections: Which is a important deciding factor for choosing dynamic adaptation, In Real-time environment user accessing the application will not be held for PendingRequests As seen in the Static Rule, but eventhough the circuit is experiencing high load to the system, through Istio Service intelligent mechanism and its adaptability to provide real-time resiliency and managing maximum possible request, which lowers the count of PendingRequestperConnections: rate enabling system to be more Resilient and Enhancing performance even in high load environment.

The findings below directly address the research question on how effective dynamic circuit breaker help in enhancing micro-service reliability. The nature of dynamic circuit breaker and its ability to adjust real-time metrics condition isolating faults will underscore superiority over static circuit breaker's nature for relying on predefined thresholds.

While the constant increase in concurrent load represent the increase in the load on the application traffic, which is huge spike on the system, From the graphical Figure:9 representation shows that there is large amount of Consecutive5xxerrors: Errorr Rate continuously increasing with static breaker, while the same nature is shown in Dynamic Breaker as well but the rate at which the count is increasing is approximately over 40% lower in comparison to the Static Breaker counts in total, which clearly indicates the decrease in the level of Error Rates by Dynamic breaker introduction being main turn point of our research objective in this project .

		Static			**Dynamic**	
	Code200	**Code500**	**PendingRequest**	**Code200**	**Code500**	**PendingRequest**
-c1 20Req	100	0	0	100	0	0
-c2 20Req	65	35	9	100	0	0
-c3 30Req	43.3	56.6	17	90	10	3
-c4 40Req	30	70	28	75	25	13
-c5 50Req	24	76	38	62	38	19

Table 2: Results of Static And Dynamic Circuit Breaker Analysis

Further Experimenting the Dynamic circuit breaking rule, continuous testing of the request continuous

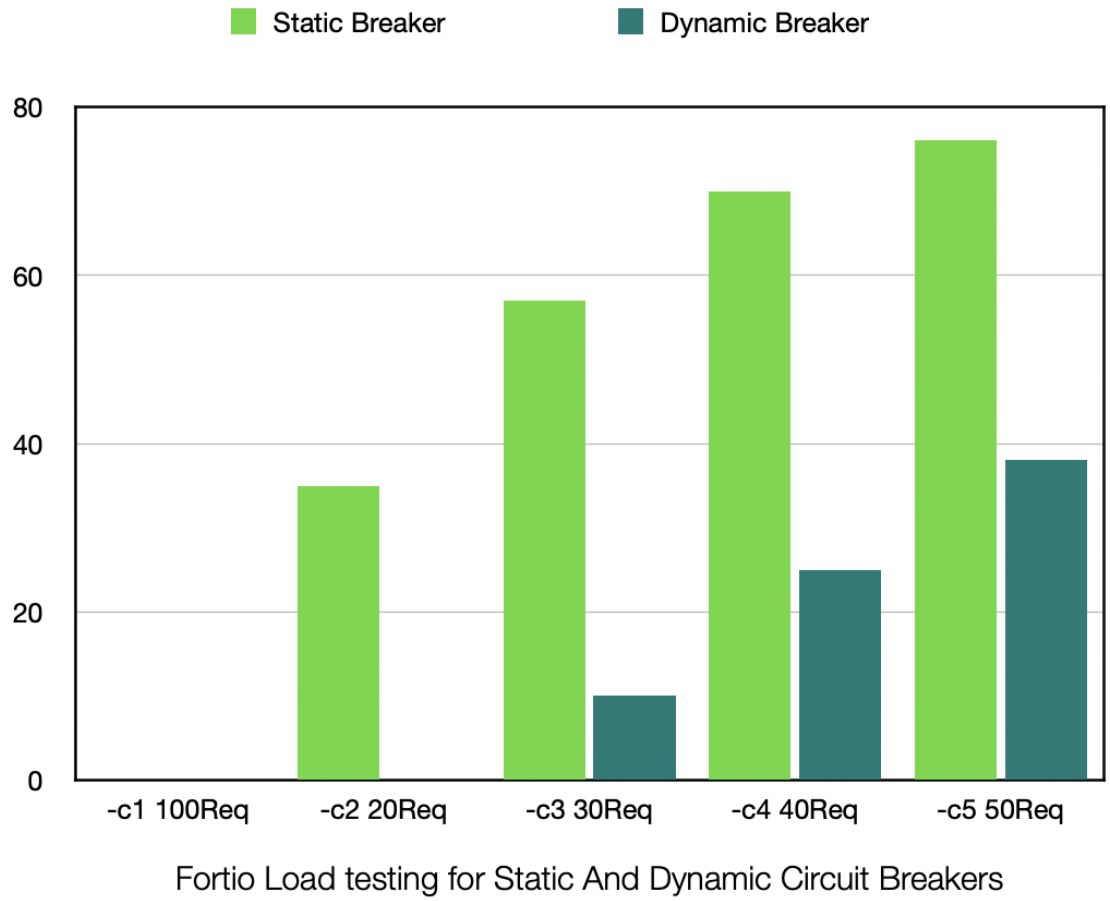


Figure 9: Graphical representation of data with Fortio Load testing of Static and Dynamic Circuit Breakers with 500Errors output

load with circuit breaking –c4 40 requests and –c5 50 requests five times will let us know the performance of dynamic DestinationRule created in reference with static rule breaker. The below tabular column from table 2 represent the continuous testing of application services with testing –c4 and –c5 concurrent requests checks the PendingRequestPerConnections: count.

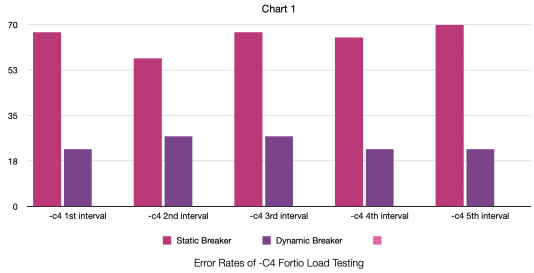
Further Experimenting on Both Static and Dynamic Circuit Breakers for Continuous load testing by -C4 and -C5 Concurrent loads, the Experiment shown the result with proving Dynamic Breaker Principle accepting fewer requests even when the circuit breaks and works in Half-Open state of Circuit Principle, passing fewer requests disallowing only limited number of requests in the Consecutive5xxerrors: which users experience in Real-Time for accessing the application and keeping the application Up-time persistently.

Critically Analyzing the Circuit Breaking pattern of Both Static and Dynamic Nature, Highlighting the benefit of Service Mesh adaptation to microservice architecture, Influence of Dynamic circuit breaker with its efficiency of 40% lesser MaxPendingRequestConnection: Connections that wait outside of the request and lowering the number of Consecutive5xxerrors: showed linear progress in increasing the linear load on the system, this effects the resilience of the microservice communication of the system, allowing time for the system to recover and trace services at a time. This overall features of Dynamic Circuit Breaking principles have contributed to increase in performance percentage of Service Mesh Adaptability in Cluster deployment, Concluding the overall enhancement of system efficiency and reliability answering research questions.

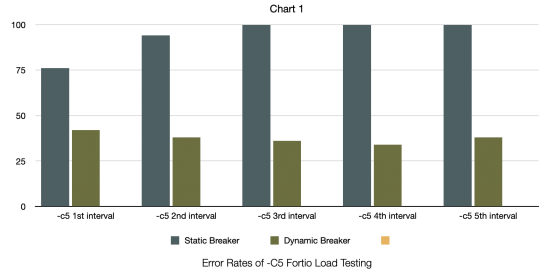
The continuation of Theoretical demonstration of Fortio and Chaos Mesh frameworks which was previously discussed in research (Veeri n.d.) and (Singh, Muntean and Gupta 2024) are used to apply its practical application emphasizing on wavering load traffic and robust fault injection methods in all

Load	Static5xx	Dynamic5xx	Load	Static5xx	Dynamic5xx
-c4	67.5	22.5	**-c5**	76	42
-c4	57.5	27.5	**-c5**	94	38
-c4	67.5	27.5	**-c5**	100	36
-c4	65	22.5	**-c5**	100	34
-c4	70	22.5	**-c5**	100	38

Table 3: Results of Continuous Load to System by Fortio with -c1,-c2,-c3,-c4,-c4,-c5



(a) Concurrent load testing of -C4 and 40 Requests



(b) Concurrent load testing of -C5 and 50 Requests

Figure 10: Comparison of Static and Dynamic Circuit Breakers with concurrent Fortio Load testing with 500Errors output

way to test system resiliency, The research recommendation is critically implemented. Above findings help in validating prior recommendations and showcase practical benefits of dynamic circuit breaker for real-world micro-service system.

Analyzing the results of both implementation evaluation, the Results obtained in implementing practical implementation of dynamic circuit breaker working on executing the performance results shown us the reduction in Pending Request by 40% less overall than Static Circuit Breaking configuration(reference from Table:2) which is directly proportional to high allowable traffic to the application enhancing resiliency in fluctuating conditions this contributed to the advancement of micro-service architecture best practice in managing real-time traffic practices.

7 Conclusion and Future Work:

The Conducted experiment for load testing with Fortio tool on Istio service mesh yielded in impacting results showing systems performance on resiliency. During the Fortio load testing, On employing Circuit Breaker Rule Micro-service architecture resulted in maintaining high throughput and low latency for varying traffic loads, Proving the nature of breaking Static Rule principle could not withstand much load and showed constant Open state of the circuit,Validating from Results (Reference from Table 2 and Table 3)PendingRequest: per every connections have successfully lowered 40% relatively than Static Circuit Breaking Rule. This dynamic circuit breaker showed result in improving better resilience and proving adaptable nature with Half-Open State with its effective principle rules and conducted experiment, this metrics has validated the effectiveness of dynamic nature in circuit breaker for managing traffic spikes and ensuring system stability in Real-World environment.

The results obtained showed us confidence in strengthening existing literature review and theoretical model developed with the rigorous testing methodology and detailed observability of tools employed showed realistic impact on architecture, highlighting the advantages of dynamic circuit breakers and its ability to recognize fault tolerance intelligently with its mechanism focusing on real world conditions were set to examine real-world conditions. Thus the results are broadly applicable for similar micro-service architectures with additional research exploring scenarios in this environment.

With the Implications of our finding significance in suggesting dynamic circuit breakers offer real-world practical solution for improving micro-service resilience and reliability, Future work considering the practical application of this system by extending the fault tolerance scenarios exploring integration with other

available service mesh platforms in Hybrid cloud environment or testing through Multi-Cloud environment, investigating on long term effects on dynamic circuit breakers on system performance, Also tuning the circuit breaker destination rule with Machine learning model setting up with Automated adaptability with historical data and processing the system with real-time metrics injected could help in further more fine grain optimizing the real-time system responses. The subsequent experiments on micro-service architecture with circuit breaker policy on Istio service mesh provides a robust framework for micro-service management for any traffic conditions, contributing to the advancement of best practices in Service mesh architecture and micro-service communication management.

References

Alboqmi, Rami and Rose F. Gamble (2024). "Compliance Validation in the Service Mesh Architecture". In: *2024 IEEE Symposium on Product Compliance Engineering - (SPCE Bloomington)*, pp. 1–5. DOI: 10.1109/IEEECONF63668.2024.10739633.

Beltre, Angel M. et al. (2019). "Enabling HPC Workloads on Cloud Infrastructure Using Kubernetes Container Orchestration Mechanisms". In: *2019 IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC)*, pp. 11–20. DOI: 10.1109/CANOPIE-HPC49598.2019.00007.

Cheng, Xuelin et al. (2023). "Research on Kubernetes Scheduler Optimization Based on real Load". In: *2023 3rd International Conference on Electronic Information Engineering and Computer Science (EIECS)*, pp. 711–716. DOI: 10.1109/EIECS59936.2023.10435549.

German, Kazenas and Olga Ponomareva (2023). "An Overview of Container Security in a Kubernetes Cluster". In: *2023 IEEE Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pp. 283–285. DOI: 10.1109/USBEREIT58508.2023.10158865.

Karn, Rupesh Raj et al. (2022). "Automated Testing and Resilience of Microservice's Network-link using Istio Service Mesh". In: *2022 31st Conference of Open Innovations Association (FRUCT)*, pp. 79–88. DOI: 10.23919/FRUCT54823.2022.9770890.

Kurpad, Supreeth et al. (2023). "Microarchitectural Analysis and Characterization of Performance Overheads in Service Meshes with Kubernetes". In: *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, pp. 1–6. DOI: 10.1109/ASIANCON58793.2023.10270428.

Malviya, Anshita and Rajendra Kumar Dwivedi (2022). "A Comparative Analysis of Container Orchestration Tools in Cloud Computing". In: *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 698–703. DOI: 10.23919/INDIACom54597.2022.9763171.

Saleh Sedghpour, Mohammad Reza et al. (2023). "Breaking the Vicious Circle: Self-Adaptive Microservice Circuit Breaking and Retry". In: *2023 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 32–42. DOI: 10.1109/IC2E59103.2023.00012.

Sedghpour, Mohammad Reza Saleh, Cristian Klein and Johan Tordsson (2021). "Service mesh circuit breaker: From panic button to performance management tool". In: *Proceedings of the 1st Workshop on High Availability and Observability of Cloud Systems*, pp. 4–10.

Sharma, Vivek (2022). "Managing Multi-Cloud Deployments on Kubernetes with Istio, Prometheus and Grafana". In: *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*. Vol. 1, pp. 525–529. DOI: 10.1109/ICACCS54159.2022.9785124.

Singh, Shubham, Cristina Hava Muntean and Shaguna Gupta (2024). "Boosting Microservice Resilience: An Evaluation of Istio's Impact on Kubernetes Clusters Under Chaos". In: *2024 9th International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 245–252. DOI: 10.1109/FMEC62297.2024.10710237.

Song, Meina, Qingyang Liu and Haihong E. (2019). "A Mirco-Service Tracing System Based on Istio and Kubernetes". In: *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, pp. 613–616. DOI: 10.1109/ICSESS47205.2019.9040783.

Veeri, Veeranjaneyulu (n.d.). "Modern Kubernetes Ingress Solutions: An In-depth Comparison of Contour and Istio Architectures". In: ().

Wang, Yu-Te et al. (2022). "Analyzing and Monitoring Kubernetes Microservices based on Distributed Tracing and Service Mesh". In: *2022 29th Asia-Pacific Software Engineering Conference (APSEC)*, pp. 477–481. DOI: 10.1109/APSEC57359.2022.00066.

Wang, Ziqiang et al. (2022). "A Web-based Orchestrator for Dynamic Service Function Chaining Development with Kubernetes". In: *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, pp. 234–236. DOI: 10.1109/NetSoft54395.2022.9844086.

Weerasinghe, L.D.S.B and I Perera (2022). "Evaluating the Inter-Service Communication on Microservice Architecture". In: *2022 7th International Conference on Information Technology Research (ICITR)*, pp. 1–6. DOI: 10.1109/ICITR57877.2022.9992918.

XIE, Xiaojing and Shyam S. Govardhan (2020). "A Service Mesh-Based Load Balancing and Task Scheduling System for Deep Learning Applications". In: *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pp. 843–849. DOI: 10.1109/CCGrid49817.2020.00009.