Dissertation on

"µScale-A scalable microservices benchmark"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE20CS390A – Capstone Project Phase - 1

Submitted by:

Vismaya R	**PES1UG20CS511**
Abhishek Patil	**PES1UG20CS531**
G Ranjitha Rani	**PES1UG20CS549**
Tushar N Borkade	**PES1UG20CS608**

Under the guidance of

Prof.Prafullata Kiran Auradkar
Assistant Professor, Department of Computer Science and Engineering, PES University

January - May 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100 Feet Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'µScale-A scalable microservices benchmark'

is a bonafide work carried out by

Vismaya R	**PES1UG20CS511**
Abhishek Patil	**PES1UG20CS531**
G Ranjitha Rani	**PES1UG20CS549**
Tushar N Borkade	**PES1UG20CS608**

in partial fulfilment for the completion of sixth semester Capstone Project Phase - 1 (UE19CS390A) in the Program of Study - **Bachelor of Technology in Computer Science and Engineering** under rules and regulations of PES University, Bengaluru during the period Jan. 2022 – May. 2022. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 6th semester academic requirements in respect of project work.

Signature	Signature	Signature	External
Prof.Prafullata Kiran Auradkar Assistant Professor, Department of Computer Science and Engineering,	Dr. Shylaja S S Chairperson	Dr. B K Keshavan Dean of Faculty	viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

DECLARATION

We hereby declare that the Capstone Project Phase - 1 entitled **"μScale-A scalable microservices benchmark"** has been carried out by us under the guidance of **Prof.Prafullata Kiran Auradkar, Assistant Professor, Department of Computer Science and Engineering, PES University** and submitted in partial fulfilment of the completion of sixth semester of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester January – May 2023. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES1UG20CS511	**Vismaya R**
PES1UG20CS531	**Abhishek Patil**
PES1UG20CS549	**G Ranjitha Rani**
PES1UG20CS608	**Tushar N Borkade**

ACKNOWLEDGEMENT

We would like to express our gratitude to Prof.K.V.Subramanian, Department of Computer Science and Engineering, PES University, for his continuous guidance, assistance, and encouragement throughout the development of this UE20CS390A - Capstone Project Phase-1.

We would like to express our gratitude to Prof.Prafullata Kiran Auradkar, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE20CS390A - Capstone Project Phase-1.

We are grateful to the project coordinator, Dr. Priyanka H., all the panel members & the supporting staff for organizing, managing, and helping the entire process.

We take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support we have received from her.

We are grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, Dr. B.K. Keshavan, Dean of Faculty, PES University for providing us various opportunities and enlightenment during every step of the way.

Finally, this project could not have been completed without the continual support and encouragement we have received from our family and friends.

ABSTRACT

µScale: A scalable microservices benchmark is a benchmarking tool that facilitates the performance and analysis of microservices-based applications. Initially, we are setting up the complete implementation and working of a similar tool viz, µBench. This tool generates prototype microservice applications that can run on a Kubernetes cluster. It allows users to manage key characteristics of the microservices it generates, including the service-to-service, and the behaviour of microservices using a variety of stress functions (for instance, CPU, memory, I/O, network).

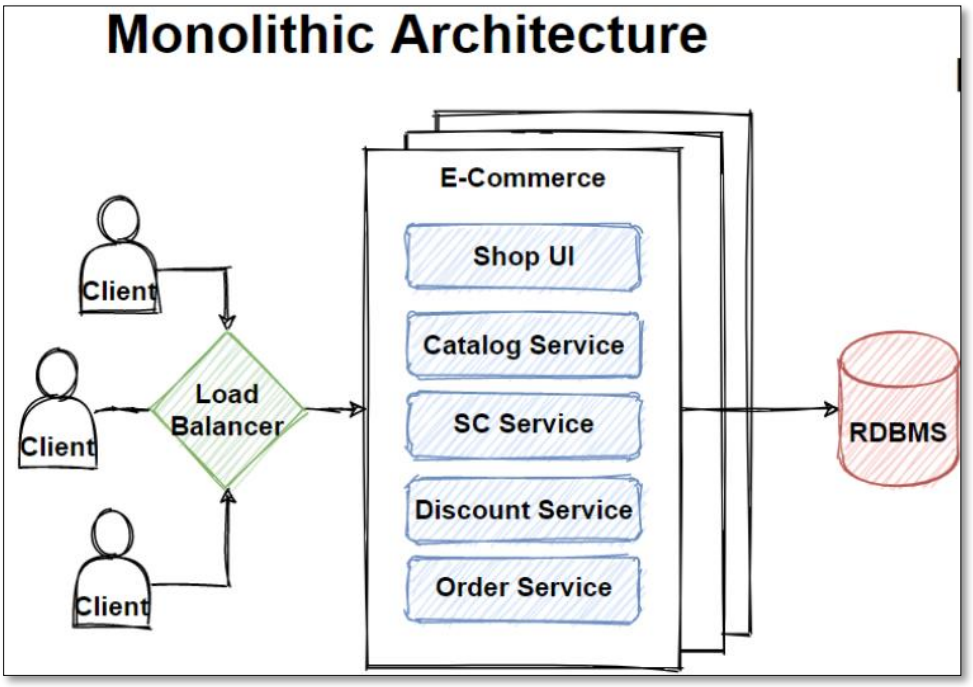
With the working of the µBench, our main aim is to scale and extend the features of µBench i.e., making it serverless and we will analyse traces from Alibaba to identify microservices that constitute an application and then convert those into the model.

TABLE OF CONTENT

CHAPTER 1: INTRODUCTION

1.1Monolithic services:

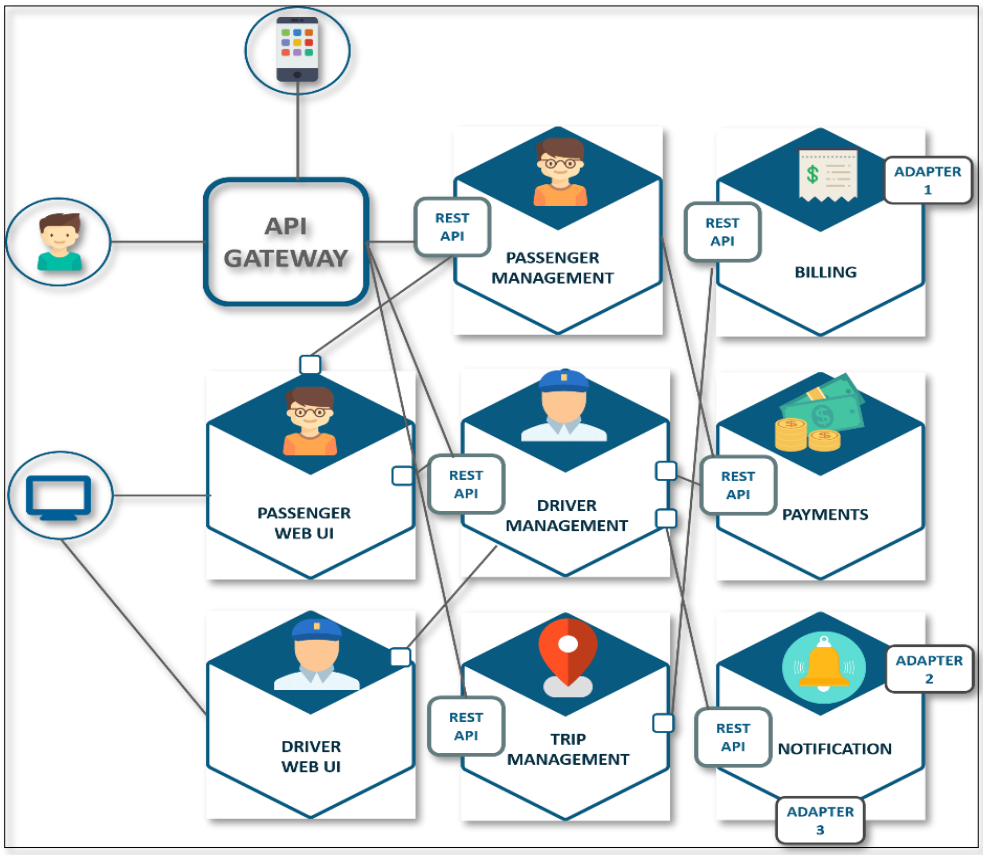
A monolithic application is a single, cohesive unit that contains all the functionalities of a project within one codebase. However, this approach poses numerous challenges and limitations. For instance, there is a high degree of interdependence among components, which makes it challenging to scale or modify specific parts of the system. Additionally, this architecture limits agility and innovation, as updates or changes are difficult to implement. Ultimately, monolithic applications are difficult to scale, and certain portions of the system cannot be scaled independently, resulting in time-consuming and effort-intensive processes. With the increasing prevalence of large-scale data centres hosting online cloud services across diverse industries, adopting a more flexible and adaptable architecture has become imperative.



1.2. Microservices:

Microservices architecture, on the other hand, is fast growing to overcome the drawbacks of monolithic systems.

Cloud services have transitioned away from single-tiered software programmes and moved towards graphs of many loosely linked, single-concerned microservices. Microservices are fine-grained, loosely connected, and independently deployable application tiers that assemble to perform more complex functionality. They enable for rapid and easy scaling of systems, reducing the time it takes to implement new features. Some of its advantages include cost-effectiveness, minimal failure impacts, and simple management, adaptability, and reliability.



Microservices benefit from language and programming framework heterogeneity because they only require a single cross-application interface API. Microservices, on the other hand, restrict the amount of programming languages that can be used for development.

Individual microservices can be easily upgraded or switched out and replaced by newer modules without causing major changes to the application's architecture. In contrast, monolithic makes regular modifications difficult and error-prone.

While microservices are extremely beneficial, there are a few drawbacks. Higher communication costs are one of them.

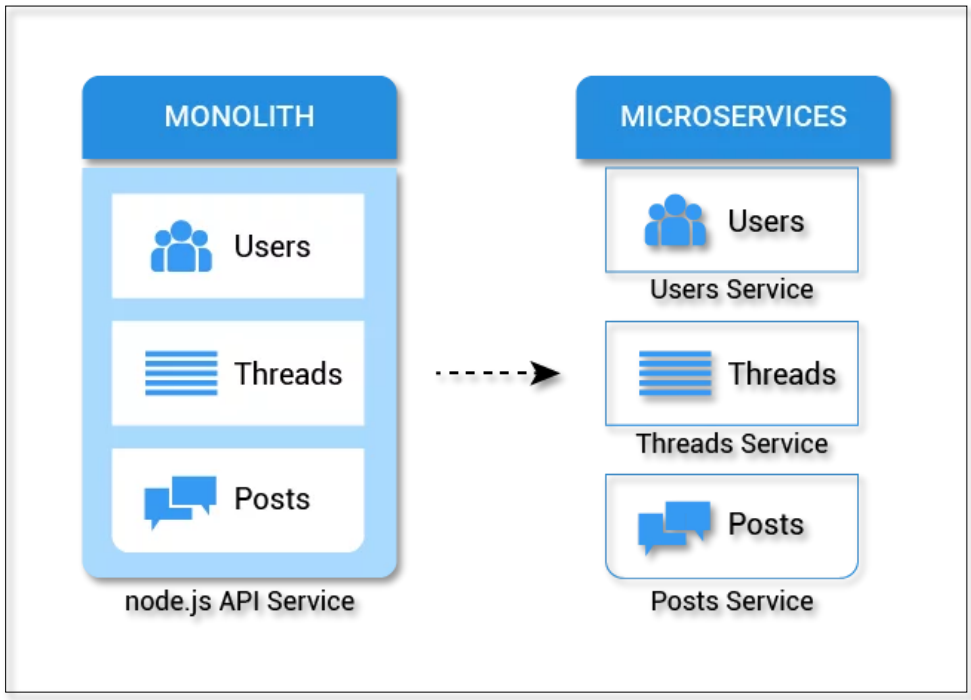
Another significant issue with microservices is their highly unpredictable behaviour when thousands of microservices are involved.

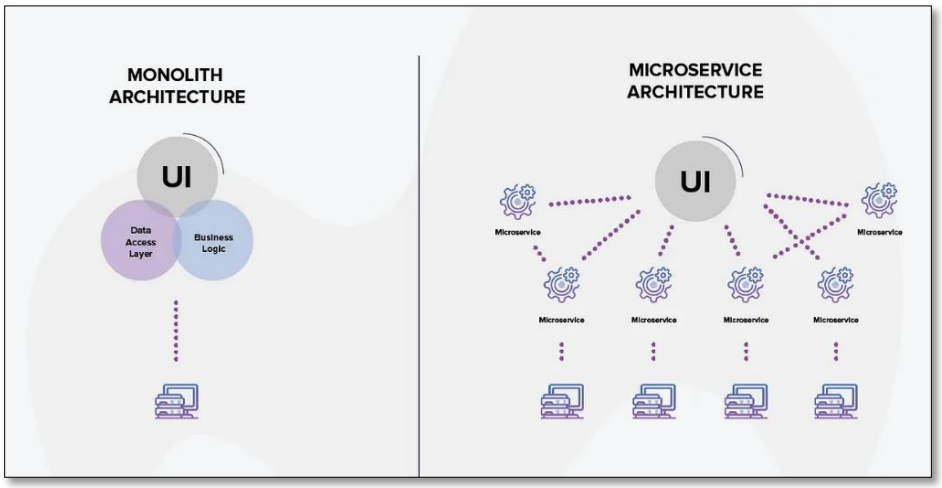
1.3. When should you move to microservices architecture from monolithic?

When you've decided to embrace microservices, the question is when you should start using them. When your organisation is growing and experiencing productivity challenges, it is time to convert monolithic apps to microservices.

Working on smaller projects is simpler with the monolithic system. Larger projects, on the other hand, necessitate the transition from monolithic architecture to microservices. Also, when communication between different teams is disrupted, it is time to act.

Apart from this, when the need for frequent testing arises, you must consider migrating to microservices. Many big players like Amazon, Uber, Netflix, eBay and Google have already transitioned from a monolithic architecture to microservices





1.4. Benefits of microservices

Microservices architecture provides practical advantages for businesses compared to centralized monolithic applications. Advantages of shifting to microservices architecture:

- Accelerated time-to-market.
- improved app and customer data security
- Application downtime has been reduced.
- A systematic strategy to scaling
- App maintenance and debugging are simplified.
- Increased client retention
- Increased staff productivity

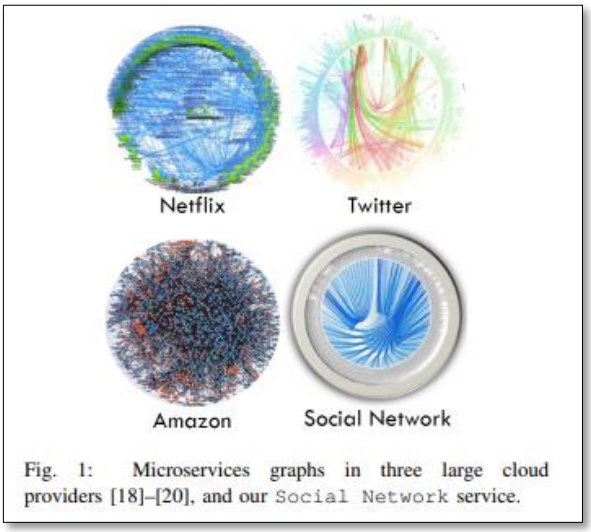


Fig. 1: Microservices graphs in three large cloud providers [18]–[20], and our `Social Network` service.

1.5. The evolution continues!

While microservices architecture is not a new concept, it is continually evolving through API and container technology advancements. This presents new opportunities for business benefits across various functional areas.

CHAPTER 2: PROBLEM STATEMENT

2.1 Description

The aim of this project will be to design and implement a benchmarking tool that runs microservice applications to measure the performance and analysis using different parameters viz, CPU, memory, etc. This helps researchers and cloud platform developers who lack real microservice applications to benchmark their findings.

2.2 Scope

The scope of this project spans three objectives. The first and most important of these is to build services with the required mesh topology. The second objective is to measure the performance and check the workload capacity of the different services. And the final objective is to implement the serverless and analyse traces from Alibaba to identify microservices that constitute an application and then convert those into the model.

CHAPTER 3: LITERATURE REVIEW

A comprehensive literature survey was carried out on different systems that have been developed for benchmarking microservice applications. Past research has involved the use of a limited number of services that an application can have and is not serverless and works in synchronous style.

3.1. Characterizing Microservice Dependency and Performance: Alibaba Trace Analysis

- **Year:** 2021
- **Description:**
    - A thorough examination of runtime performance in large-scale production clusters was conducted by analysing over 109 call traces from roughly 20,000 microservices in the Alibaba cluster within seven days.
- **Techniques/ Tool:**
    - Aimed to simulate the impact of request fanout on tail latency at scale and evaluate the effects of power management on the performance of latency-sensitive microservices.
    - It is event driven.

- **Inputs:**
    - Alibaba cluster traces.

- **Result/ output:**
    - Characterization of structural properties of microservices call graphs.

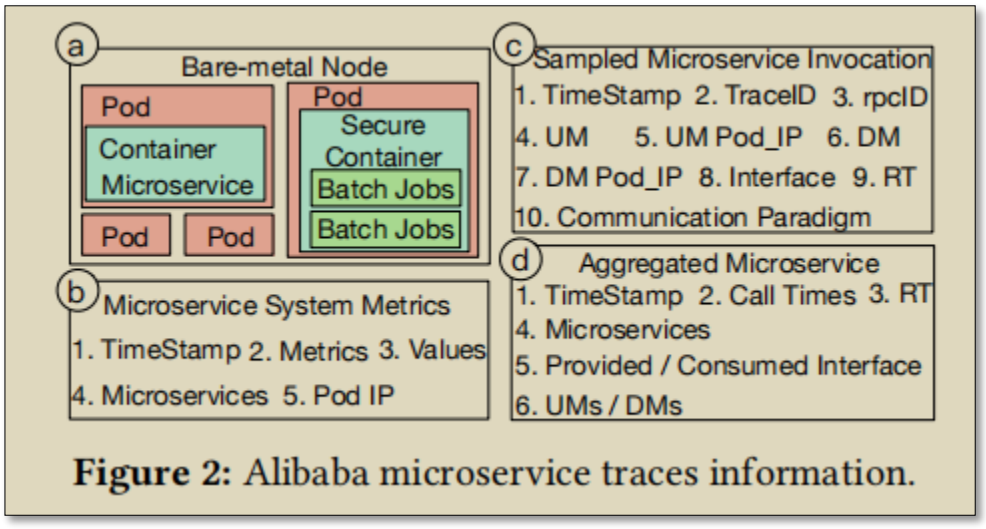
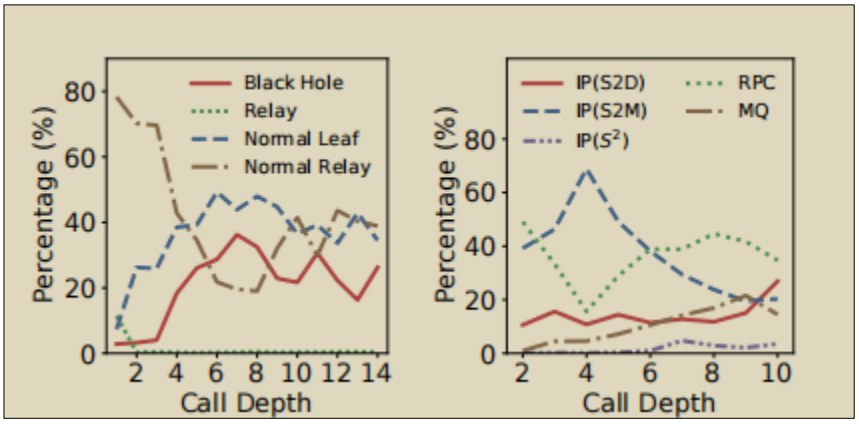


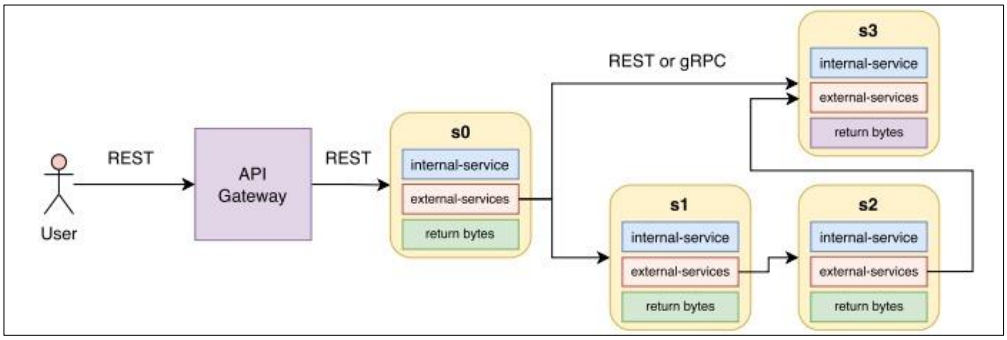
Figure 2: Alibaba microservice traces information.

- **Limitation:**
    - The graph model is unable to accommodate node sharing across different graphs, where a single node may appear in multiple graphs.

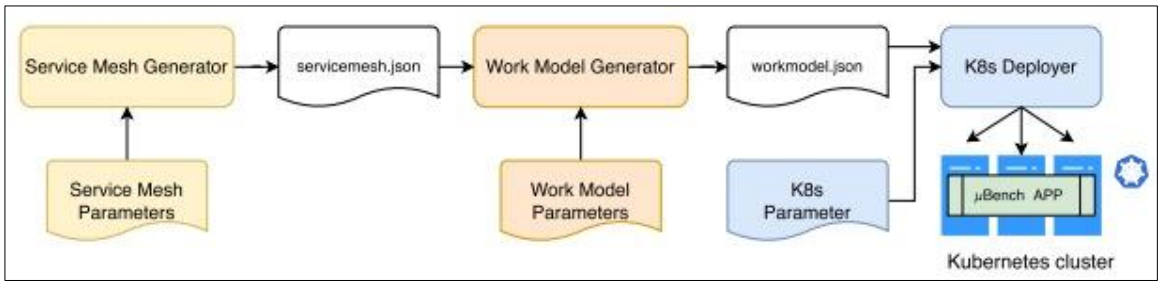


3.2. µBench: An Open-Source Factory of Benchmark Microservice Applications

- **Year:** 2022
- **Description:**
    - µBench is an open-source tool for benchmarking cloud/edge computing platforms that run microservice applications.



    - It solved the issues that were not addressed by Deathstar Bench:
        - Can work with any number of applications and microservices that the user wishes to.
        - They consist of microservices whose quantity and work model can be configured.
- **Toolchain:**
    - They work in Toolchain which includes: ServiceMeshGenerator, WorkModelGenerator and K8sDeployer.
    - Makes it easy to work with changing the configuration wherever required according to the user configuration.



3.3 Scaling up a cloud microservices simulator

- **Year:**2022
- **Description:**
  - The goal is to enhance and expand µsim to simulate a social media application from the Deathstar benchmark suite. Specifically, the "User Login," "search service", and "recommender service" will be replicated.
- **Techniques/tools:**
  - Dhrystone and STREAM - to modify µsim

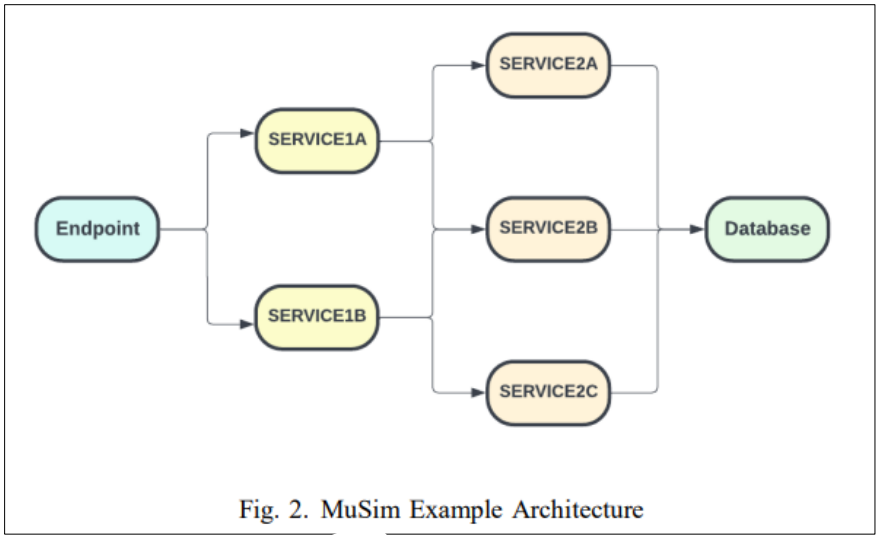


Fig. 2. MuSim Example Architecture

- **Input:**
    - YAML file having all the required configuration and connection, traces of user login and search application.
- **Output:**
    - Measure the time it takes for loops that exceed CPU cache capacity to estimate the effective memory and bandwidth of contemporary CPU-based architectures.
- **Limitations:**
    - Extended μsim only for two applications i.e., login and search.

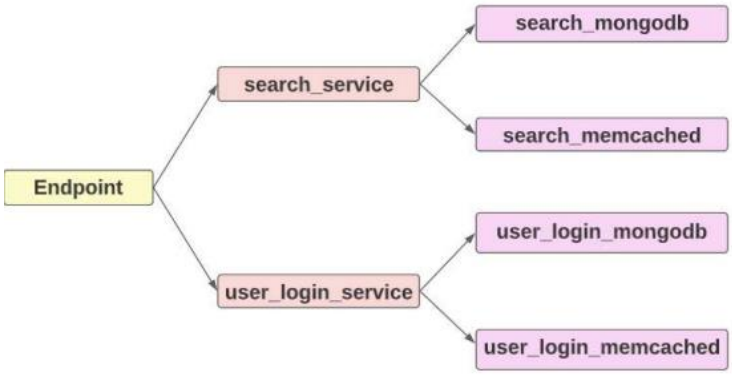
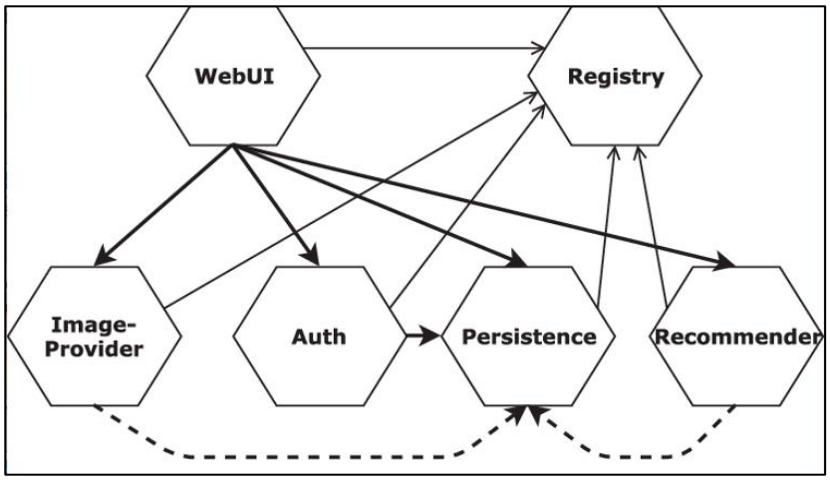


Fig. 4. Search and User Login Service

3.4 TeaStore-A microservice reference application

- **Year:**2019
- **Description:**
    - Using the most recent technological stack, teastore is a microservice application for benchmarking. It represents an online store for tea and consists of five services with diverse performance behaviour used to evaluate the applicability of self-management approaches for microservices.

- **Techniques /tool:**
    - Five services: Registry, Web UI, Auth, Image provider, Recommender, Persistence
    - Three different deployment models: Application server, docker container, Kubernetes



- **Input:** Requests sent by user is executed by an idle user from the pool user the user perform the action and return to pool.

- **Output:**
    - System-level metrics [CPU utilization, memory usage and network traffic]
    - Application-level metrics
- **Limitations:**
    - Focuses on a single use case of online tea store with five services which may not be the representative of other types of microservices-based applications

CHAPTER 4: SYSTEM REQUIREMENTS SPECIFICATION

• Functional Requirements:

1. Microservice Benchmarking: A tool that can benchmark microservices should be able to assess their functionality, scalability, and fault tolerance.

2. Test Suite Generation: A tool creates test suites for microservices based on predefined workloads, scenarios, and data types

3. Microservice Configuration: Microservices should be able to be configured with a variety of options, including resource distribution, scaling rules, and fault injection parameters.

4. Experiment Execution: A tool should be able to do tests to assess how well microservices function under various workloads and scenarios.

5. Results Collection and Analysis: The tool should collect and analyse experiment results (e.g., response time, throughput, error rates) and generate reports and graphs.

- **Non-Functional Requirements:**

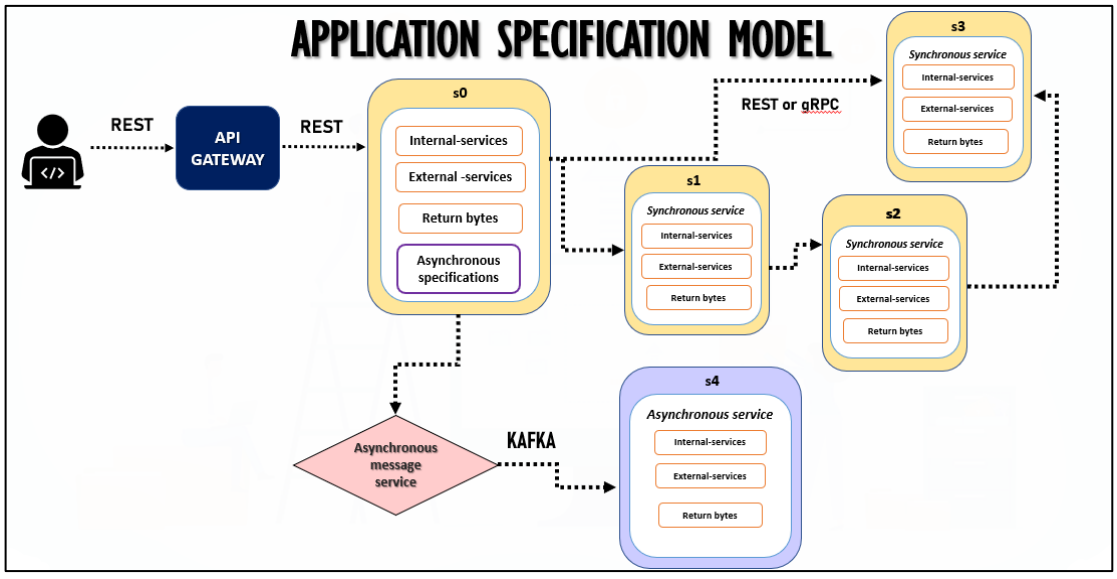
1. **Performance**– Able to perform accurately under high-loads.
2. **Scalability–** Scalable and able to handle many microservices.
3. **Understandability-** Easily understandable for the user
4. **Reliability –** Reliable and able to produce consistent results over multiple test runs.

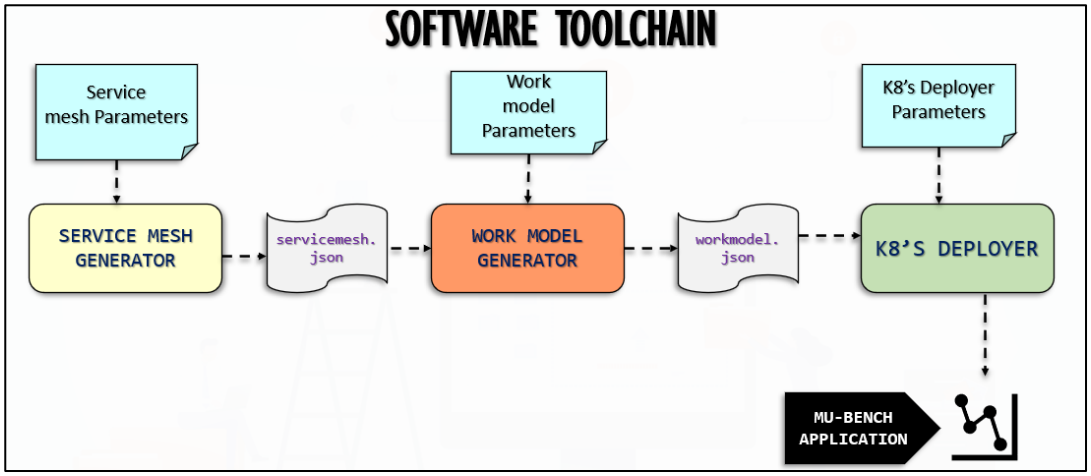
CHAPTER 5: SYSTEM DESIGN

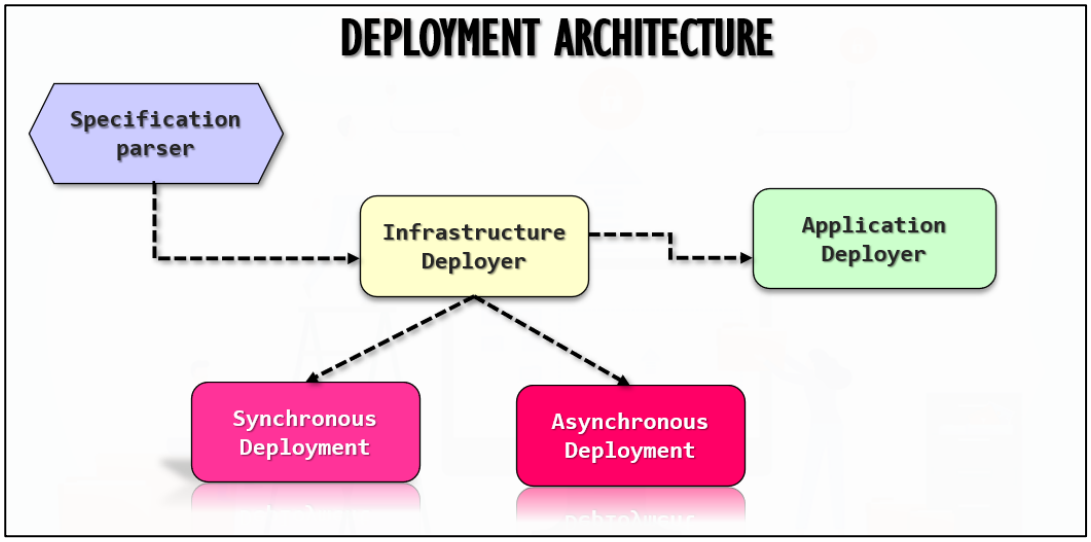
- **Design Approach:**
  - Specifying the asynchronous communication parameters in the configuration files and feeding to the infrastructure deployer for the deployment of those services and thus benchmarking the microservices which use these methods of communication.

- **Possible solution to this approach:**
  - Specify the asynchronous communication parameters.
  - Use an asynchronous messaging service such as Kafka/RabbitMQ to deploy these services.
  - After these services are created, deploy them on the Kubernetes cluster and benchmark using the μScale application.

- **High-level Design:**







- **Service Mesh Parameters:** Required user configurations which include number of services, databases, probabilities, external service groups etc.
- **Service Mesh Generator:** Generates topology (service-to-service communication) for a given user specification.
- **Work model parameters:** Includes the function which is running as internal-service (such as CPU, memory etc). These functions called by respective external-service groups and give how many bytes to return.

- **Work model generator:** The work model for each microservice is generated, outlining its internal service function, external services called (from servicemesh.json), and average bytes sent back.
- **Kubernetes Deployer:** Deploys the microservices application on Kubernetes cluster.
- **Asynchronous specifications:** Specifying the asynchronous communication metrics in the configuration files such as "request_method."
- **Asynchronous messaging service:**
    - Publish-Subscribe type of communication.
    - Publishers create messages and send them to a message broker, while subscribers receive messages from the broker.
        - Publishers do not need to know anything about subscribers
        - Subscribers - subscribe to topics
    - Ex: Kafka, RabbitMQ
- **Specification parser:** A specification parser is a type of software that is designed to examine a specification document that has been written in a particular format or programming language. The parser will then identify and extract important details, such as requirements, limitations, and variables from the document.
- **Infrastructure deployer**: An infrastructure deployer is a software tool or program that automates the process of provisioning, configuring, and deploying infrastructure resources.

- **Application Deployer**: Deploy microservices as deployments that is consisting of PODs where microservice exposed through dedicated services.

CHAPTER 6: CONCLUSIONS FROM PROJECT PHASE-1

- Previous technologies like Deathstar Bench,μqSim worked with only limited number of applications.
- Implementation of the existing tool μBench and understanding it thoroughly
- μBench works for only synchronous style of communication.
- Present world scenarios require both synchronous and asynchronous style of communication e.g.: messaging services, thus extending the tool to perform both.
- Building the required architecture with the extension in our tool - μScale.

CHAPTER 7: PLAN OF WORK FOR PHASE-2

- Exploring and specifying all the required asynchronous communication parameters in the configuration files.
- Benchmarking the asynchronous applications and comparing them with the synchronous applications.
- Implementation of the μScale application which includes both synchronous and asynchronous communications.

ORIGINALITY REPORT

7%	4%	3%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.softwebsolutions.com Internet Source	4%
2	Abdul Mannan Kanji, Ishita Chaudhary, Rithika Lakshmi Shankar, Subramaniam Kalambur. "Scaling Up a Cloud Microservices Simulator", 2022 4th International Conference on Circuits, Control, Communication and Computing (I4C), 2022 Publication	2%
3	Andrea Detti, Ludovico Funari, Luca Petrucci. "μBench: an open-source factory of benchmark microservice applications", IEEE Transactions on Parallel and Distributed Systems, 2023 Publication	1%

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		