Table of Contents

Literature Review

The purpose of this literature review is to investigate some of the current trends and past innovations in relation to microservice-based applications. And use the findings, to help develop a more modern and innovative application.

High Availability and Fault Tolerance

When building a microservice-based system, the most important consideration is ensuring that the system can achieve high availability. In order to achieve high availability, it is important to select appropriate frameworks for the microservice-based system that is in development. As each framework may have various impacts on each microservice-based system. An explorative study by Marquez et al. (2020) looked into open-source frameworks, with the intention of providing a study of the positive and negative impacts of those frameworks on the ability for microservices to be highly available. With the idea that this knowledge would be used by those working with microservices, to make more informed decisions about some of the frameworks they may be considering using in their microservice-based system.

While useful for looking at open-source frameworks, it is limited to only those frameworks, it does not study any other kinds.

While high availability is an important issue that developers should try and aim for, it does not mean that microservice-based application will never experience any issues. Therefore, developers are also aiming to make their applications fault tolerant, so that if they do have an issue, despite trying to prevent it, the application can withstand the issue and continue to function. In a paper by Wu, Zuo and Zhang (2018) they present a custom framework for testing microservice-based applications. In which users can run various test cases to see how fault tolerant an application is.

As their solution was only tested in one use case further research may be required into just how effective the solution is.

Kubernetes and Stateful Applications

With Kubernetes being the leader in container deployment, it is the go-to for many microservice-based applications when deploying microservice containers. As it reduces the complexity of orchestrating the containers for the microservices and their availability. And allows for microservices to be stateful. However, it doesn’t seem to be without its issues. A paper by Abdollah Vayghan et al. (2019) argues that the complexity of stateful microservices can be too much for what the Kubernetes controllers can handle. Therefore, a State Controller was proposed, which works by having two Kubernetes pods one labelled ‘active’ which is the pod in use and another labelled ‘standby’ which can take over if the ‘active’ pod fails. The paper claims their solution can improve stateful microservices recovery time by 55% to 99%.

However due to the scope of their experiment, further research would be required into the solution's effectiveness in larger scale.

Monitoring and Management

Due to the nature of microservice-based systems, there can be a lot of individual systems that are working with each other, making them quite complex. Because of this complexity it is not easy to identify issues, as developers have to look through large log files to try and identify issues. And due to the system being online a lot of issues may only arise while the system is online. A literature review by Soldani, Andrew Tamburri and Van Den Heuval (2018) identified logging size and complexity to be the biggest issue when monitoring microservice-based applications. They further found that due to each microservice having its own log file, combined with the general nature of microservices being separate from each other, it can make problem identification difficult. As for ways to try and make problem identification easier, Mayer and Weinrich (2017) proposed a custom monitoring and management dashboard, for microservice-based systems in which the user could view important information through various data visualisations. Some of which include:

- An overview of the system
- Runtime information of each microservice
- How services are interacting with each other

However, it does not appear to consider the inclusion of a dashboard page to present logs of the system. Having a way to easily access logs of various microservices or other parts of the system would be very helpful and make the process of accessing the logs lot more efficient.

Methodology

Throughout the papers referenced in the literature review, a variety of research methods were used.

In the paper by Marquez et al. research was conducted to find out the high-availability properties of open-source microservices. Using the list of properties they created, a survey was conducted among people active in the industry, which provided them with an idea of the impacts of open-source frameworks on microservices from actual IT industry members, working with the technologies. However, being a survey there is still the possibility that members may not be truthful, and as it was a limited selection of participants it may not accurately reflect the overall opinions of the entire industry.

Soldani, Andrew Tamburri and Van Den Heuval (2018) in their paper conducted a broad literature review into what exactly were the current pros/cons of microservices. The review covered a broad array of areas of microservices, allowing it to convey a largescale picture of the pros/cons. However, as a lot of their sources were found online, which they did acknowledge, this may impact their validity.

And as seen in the literature review, several papers proposed their own ideas to solve a gap that they had discovered from prior research. This included the papers by Wu, Zuo and Zhang (2018), Abdollah Vayghan et al. (2019), and Mayer and Weinrich (2017). The methodology of which is discussed below:

- Wu, Zuo and Zhang (2018) broke down their proposal into three levels: user interface, scheduling, and execution. They also provided a use case to demonstrate how they're proposal might work. However, it is only one example, so it only gives a narrow picture of the system.
- Abdollah Vayghan et al. (2019) conducted an experiment using the solution they proposed. By using experiment conditions that were close to a real environment, they were able to get more accurate results, allow their solution to have more validity. The only issue was that due to the limited scope of their experiment, the results may not be as accurate in larger scall microservice application deployments.
- Mayer and Weinrich (2017) conducted a survey with fifteen industry experts in twelve different companies to gain a strong idea of the required information for their dashboard solution. The study was limited to Germany and Austria. As half of the participants were interviewed face-to-face or online through Skype there is the possibility that the other half may not have had the chance to provide as much insight as they could have as they completed a questionnaire instead.

Background of the Project

The client wants to train an agent-based software system to be resilient and adaptable to changes in a server-side system environment. To do this, it must learn what to look out for, so that it can recognize any failures or attacks and attempt fix them, to try and maintain high availability for the system. A server-side application has been proposed, which simulates microservices that behave incorrectly. By behaving incorrectly, the agent-based software system will be able to monitor the system and dynamically learn how best to meet the needs of the server-side microservice application. This will allow the agent-based software to be capable of resolving issues that may be similar to those that arise from attacks or failures. And thus, be capable of resolving them in a real situation.

Project Goals and Objectives

The main goals of this project that are required to be fulfilled in order to complete the project include:

1. To create a microservice-based application where some microservices will be setup to not work correctly with bad or slow response times to simulate faults or attacks, for the client's agent-based system to learn from.

2. Utilise Dapr, a service mesh, Docker, and Kubernetes to deploy and orchestrate the microservice-based application across containers.

In order to achieve these main project goals a list of objectives have been devised that will layout a more specific process for meeting the main requirements of the project.

- Create a web application consisting of a number of generic microservices
- Setup microservice simulator configuration database and supply initial configuration
- Integrate web application with Dapr
- Implement interface for monitoring the system
- Implement workload generator interface
- Configure state management for simulator and microservices
- Provide functional interface to enable communication between workload generator and the system
- Implement event driven design pattern within the simulator e.g., message broker
- Implement database for the collection of logs of the simulator
- Implement a monitoring client and interface for monitoring the simulator

Desired Outcomes and Benefits

The desired outcome of the project is that the team is able to produce a working prototype that can meet the requirements of the client. That is, the prototype is capable of simulating a server-side application that can simulate one that is experiencing a fault. This will allow the client to then test their agent-based system to learn how to respond to and try and fix issues that may arise in a real server-side application.

This simulator will also provide a safe environment to give the agent a chance to learn without the worry of it breaking something, as it is not a real server-side application. As the prototype will be configurable, the client can expose the agent to a variety of server-side application environments.

Learning Issues/Problem

The learning issue that I have been assigned with is state management. Specifically, how does state management work with Dapr and Kubernetes?

This is an important issue to resolve as Dapr and Kubernetes are a major new technology to the team, that will be used with the microservice-based web application. And state management is required to know in order to have the simulator be able to save state. Without it the simulator can’t be stateful or long running and users would have to re-set up the simulator each time they use it. And microservices would not be able to save or retrieve state data.

Project Scope and Exclusions

In order to meet the goals and objectives detailed above, a microservice simulator will be created that will consist of the following features:

- The system will include a number of dummy microservices that behave either correctly or incorrectly, to simulate a real microservice that may be experiencing issues
- The system will collect and display logs about the simulator which can be viewed in a monitoring interface
- The system will allow users to supply fake data through a workload generator interface, to test the simulator with

There are also some features that can be in-scope but due to constraints such as time and knowledge between the team, may not be achievable. Therefore, the team will aim to prioritise them as secondary tasks once the main features detailed above have been implemented, and if the team feels there is enough time. These features include

- A management interface and client which will allow users to configure different settings for the simulator
- A perturbation interface and generator which will allow users to insert bugs or disruptions to the simulator to make it behave poorly

Finally, any sort of hosting of the project on a server or cloud infrastructure of any kind is not in-scope for this project. The simulator project will only be run on local machines.

Project Deliverables

The final project deliverables include:

- The project source code
- Presentation demoing the prototype in action
- Documentation to explain the functionality of the prototype, how to use it and provide some understanding what has been done

Project Management Plan

Timeline

The overall timeline of the project is as follows:

Microservice Simulator Project	**Week 6**	**Week7**	**Week 8**	**Week9**	**Week 10**	**Week 11**
Build generic microservices	🟦	🟦				
Develop functional interface		🟦				
Implement event driven pattern		🟦				
Build simulator configuration component		🟦				
Implement monitoring system backend			🟦			
Build monitoring interface frontend				🟦		
Develop observability service interface				🟦		
Develop perturbation generator					🟦	
Management interface					🟦	
Testing		🟦	🟦	🟦	🟦	🟦

Table 1: Project Development Gantt Chart

Goals and Milestones

Goal: Build generic web application and incorporate with Dapr

a. Milestone: Implement several microservices
b. Milestone: Deploy microservice within Dapr environment
c. Milestone: Setup sidecars for microservices
d. Milestone: implement a message queue
e. Milestone: implement a message broker and setup pub-sub between relevant microservices

Goal: Setup a workload generator

a. Milestone: Implement workload generator web interface
b. Milestone: Implement functional interface for interaction between workload generator web interface and microservices

Goal: Build simulator configuration component

a. Milestone: Develop simulator configurator and configuration database

Goal: Implement monitoring system backend

a. Milestone: Set up database for storing logs
b. Milestone: Set up monitoring tools and store logs in database
c. Milestone: Develop observability service interface

Goal: Build monitoring interface frontend

a. Milestone: Create basic monitoring client web page
b. Milestone: implement ability to view logs

Stage 2: Would like to do, if we have time

Goal: Develop perturbation generator

a. Milestone: Implement basic frontend for inserting perturbations
b. Milestone: Implement perturbation interface which interacts with the configuration DB
c. Milestone: Create various perturbations

Goal: Management interface

a. Milestone: develop management client
b. Milestone: write system state to Configuration DB

Team breakdown and duties

Scrum Team Member	**Role**
Chuan He	Scrum Master, Developer
Luke Parker	Product Owner, Developer
Cindy Tao	Developer
Jiangli Shi	Developer
Mingyang Hou	Developer
Sukhkaran Singh	Developer

Table 2: Team Roles

As seen in the table above, my role in the team is to be the product owner as well as a developer. As the product owner I will be in charge of setting up and maintaining the project's GitHub repository and managing the Trello board. And as a developer I will be involved with working on various tasks, and implementing features, some of which will likely be related to my learning issue, during the project.

Word Count:

References

1. Márquez, G., Soldani, J., Ponce, F. & Astudillo, H. 2020, "Frameworks and high-availability in microservices: An industrial survey", *23rd Iberoamerican Conference on Software Engineering, CIbSE 2020*.
2. Wu, N., Zuo, D. & Zhang, Z. 2018, "An extensible fault tolerance testing framework for microservice-based cloud applications", *ACM International Conference Proceeding Series*, pp. 38.
3. Abdollahi Vayghan, L., Saied, M.A., Toeroe, M. & Khendek, F. 2019, "Microservice Based Architecture: Towards High-Availability for Stateful Applications with Kubernetes", *Proceedings - 19th IEEE International Conference on Software Quality, Reliability and Security, QRS 2019*, pp. 176.
4. Soldani, J., Tamburri, D.A. & Van Den Heuvel, W.-. 2018, "The pains and gains of microservices: A Systematic grey literature review", *Journal of Systems and Software,* vol. 146, pp. 215-232.
5. Mayer, B. & Weinreich, R. 2017, "A dashboard for microservice monitoring and management", *Proceedings - 2017 IEEE International Conference on Software Architecture Workshops, ICSAW 2017: Side Track Proceedings*, pp. 66.