Assignment 2

1. Summary

In this assignment, we used the number of threads (network traffic) as the primary source of uncertainty for Acme Air. This is because the number of threads represent the effective number of requests at a time and puts overload on the system which causes it to crash and restart. Our motive is that Acme Air needs to ensure that the number of times the pods crashing, and restart does not exceed once every 10 minutes for a varying load of network traffic, and the request failure errors due to system performance issues do not exceed 5%. For this purpose, we will use a **self-optimizing** mode to allow the system to optimize itself, ensuring the system can meet the targets by increasing and decreasing CPU and memory. We will attempt to describe the relationship between CPU, memory, and thread count by running Acme Air under different load conditions. We will try to build a utility function and display the analysis approach.

2. Program Architecture & Design

The initial configuration of the services was made such that the CPU cores and Memory assigned to each service are:

Service name	Memory allocated	CPU allocated
Flight service	512 MB	1.5 cores
Main service	512 MB	0.5 cores
Auth service	512 MB	1 core
Booking service	512 MB	2 cores
Customer service	512 MB	1 core

This configuration was found after many trials of running different loads. The JMeter simulates the network traffic such that **40 threads** are initialized when the test is run. Then after every 3 minutes 40 more threads are added until a total of **120 threads** are reached. This helps us in showing how our analyzer responds to the changing network traffic and the suggestions based on it.

The function **selfOptimize()** is used for adaptively adjusting the size of CPU and memory. It will check whether it is necessary to increase or decrease CPU and memory usage through the utility function. When the CPU usage is below 20%, it will reduce CPU that is allotted; when the CPU usage is above 80%, it will increase CPU that is allotted. When the memory usage is below 20%, it will reduce memory assigned by making it half; when the memory usage is above 80%, it will increase memory assigned by doubling it. Once this function is called, it will run indefinitely until interrupted manually. In case a request fails due to issues like pod crashes and restarts, it prints out the error correspondingly. When the function operates normally, it will summarize Acme Air every 30 seconds, adjust the CPU and memory usage service by service, and print out the system status and adjustment reports.

Utility Function

$$U = \begin{cases} -1, when\ CPU\% < 20\%\ or\ memory\% < 20\% \\ 1, when\ CPU\% > 80\%\ or\ memory\% > 80\% \\ 0, otherwise \end{cases}$$

3. Analysis Approach

The detailed steps are shown in the figure below. We adopted the ECA (Event Condition Action) approach. Every 30 seconds, the program collects data for each of the microservice that is running. With this data, it checks for conditions, that is the percentage of **CPU usage** and the percentage of **memory usage**. Different actions will be taken based on different situations. If the CPU% is higher than **80%**, it will tell us to increase the CPU allocation; if it's less than **20%**, it will tell us to decrease the CPU allocation. If the memory% is higher than **80%**, it will tell us to increase the memory allocation; if it's less than **20%**, it will tell us to reduce the memory allocation. After the action is completed, it will enter the next cycle. By making the adjustments in above manner, it will effectively reduce the risk of pods crashing and restarting and reduce request failure errors due to performance problems. Also, during system idle times, it will automatically lower the system performance used, thereby reducing the overall performance requirements of the system.

```
15:45:00----------------------------------------------------------------------------------------
Service acmeair-flightservice restarted 0 times!
Service acmeair-flightservice -> Memory: 53.2982032% -> Memory is good
Service acmeair-flightservice -> CPU: 40.7670002% -> CPU is good
Service acmeair-mainservice restarted 0 times!
Service acmeair-mainservice -> Memory: 28.878022% -> Memory is good
Service acmeair-mainservice -> CPU: 1.2082% -> Decrease CPU
Service acmeair-bookingservice restarted 0 times!
Service acmeair-bookingservice -> Memory: 75.6759646% -> Memory is good
Service acmeair-bookingservice -> CPU: 32.4182% -> CPU is good
Service acmeair-customerservice restarted 0 times!
Service acmeair-customerservice -> Memory: 79.4749762% -> Memory is good
Service acmeair-customerservice -> CPU: 36.1641003% -> CPU is good
Service acmeair-authservice restarted 0 times!
Service acmeair-authservice -> Memory: 40.1258854% -> Memory is good
Service acmeair-authservice -> CPU: 11.2666% -> Decrease CPU
```

Figure 1 : 40 threads

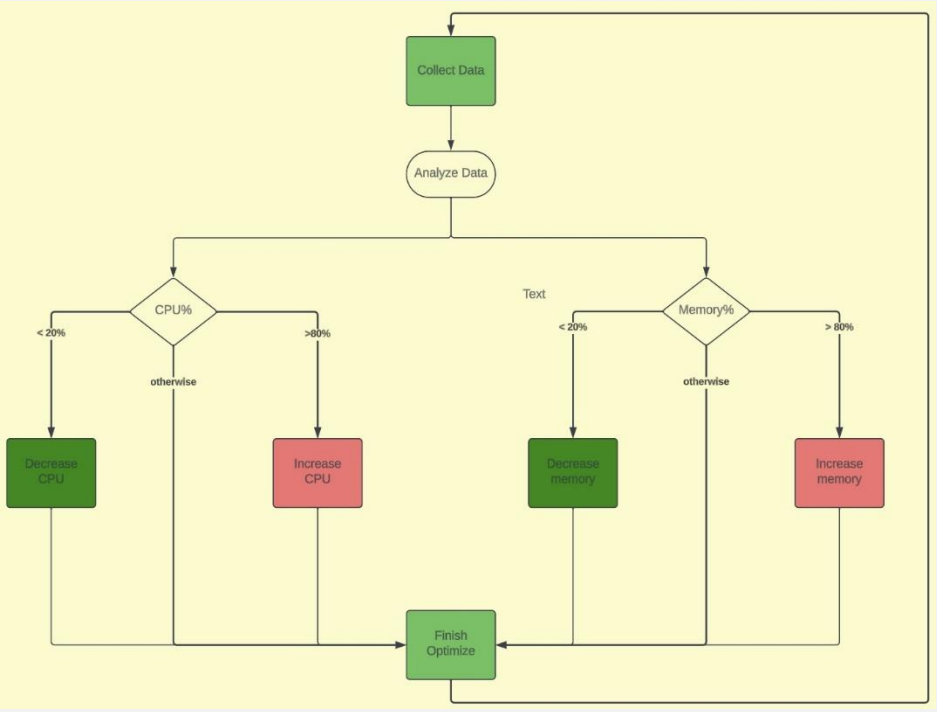
```
15:48:00----------------------------------------------------------------------------------------
Service acmeair-flightservice restarted 0 times!
Service acmeair-flightservice -> Memory: 51.8814245% -> Memory is good
Service acmeair-flightservice -> CPU: 64.06790000000001% -> CPU is good
Service acmeair-mainservice restarted 0 times!
Service acmeair-mainservice -> Memory: 28.880311% -> Memory is good
Service acmeair-mainservice -> CPU: 1.2985002% -> Decrease CPU
Service acmeair-bookingservice restarted 0 times!
Service acmeair-bookingservice -> Memory: 83.6680837% -> Increase memory
Service acmeair-bookingservice -> CPU: 54.1890001% -> CPU is good
Service acmeair-customerservice restarted 0 times!
Service acmeair-customerservice -> Memory: 79.1084982% -> Memory is good
Service acmeair-customerservice -> CPU: 62.4462% -> CPU is good
Service acmeair-authservice restarted 0 times!
Service acmeair-authservice -> Memory: 40.0307928% -> Memory is good
Service acmeair-authservice -> CPU: 18.368199999999998% -> Decrease CPU
```

Figure 2 : 80 threads

```
15:53:11----------------------------------------------------------------------------------------
Service acmeair-flightservice restarted 0 times!
Service acmeair-flightservice -> Memory: 53.3969503% -> Memory is good
Service acmeair-flightservice -> CPU: 90.66380050000001% -> Increase CPU
Service acmeair-mainservice restarted 0 times!
Service acmeair-mainservice -> Memory: 28.880311% -> Memory is good
Service acmeair-mainservice -> CPU: 1.2991% -> Decrease CPU
Service acmeair-bookingservice restarted 0 times!
Service acmeair-bookingservice -> Memory: 84.1402059% -> Increase memory
Service acmeair-bookingservice -> CPU: 78.0102% -> CPU is good
Service acmeair-customerservice restarted 0 times!
Service acmeair-customerservice -> Memory: 79.4565204% -> Memory is good
Service acmeair-customerservice -> CPU: 86.1808005% -> Increase CPU
Service acmeair-authservice restarted 0 times!
Service acmeair-authservice -> Memory: 40.2501532% -> Memory is good
Service acmeair-authservice -> CPU: 24.576900000000002% -> CPU is good
```

Figure 3 : 120 threads

An example of how the program informs the current usage of CPU and memory for different number of threads allocated is shown. With these suggestions given by the analyzer, we will be able to apply new changes such that the system does not crash with increasing load.



Thank you.

Group 11

All the plots and raw data along with the scripts used have been provided separately. The codebase used has been updated to [GitHub repository] and is made up to date. In case of any discrepancy, kindly reach out at n2chhabr@uwaterloo.ca.