Name: Anna Mary Almeida
Tutor: Mohammad Goudarzi
Student ID :32313500

FIT5225: Assignment 1- Report

This assignment was to use a python-based application and create a web service for it. To facilitate this, we use python flask and then use docker to create containers that can be used to deploy the application with ease. The application was then run on a Kubernetes cluster where pods were created to obtain objects from the images present in the list of images that were provided. Kubernetes is an open-source orchestration tool that manages container deployment, scaling and descaling of containers and container load balancing.
The experimental raw data was collected from the average response time taken for the application to run the object detection model on both the local machine and over the cloud. The experiment was run by taking a varying number of pods and worker threads to figure out what configuration is best suited for this application without causing too much overhead. Then the average of this raw data was taken and a graph has been plotted as shown below.

The Average Response time of the web service versus the number of pods for a different number of threads for the local client:

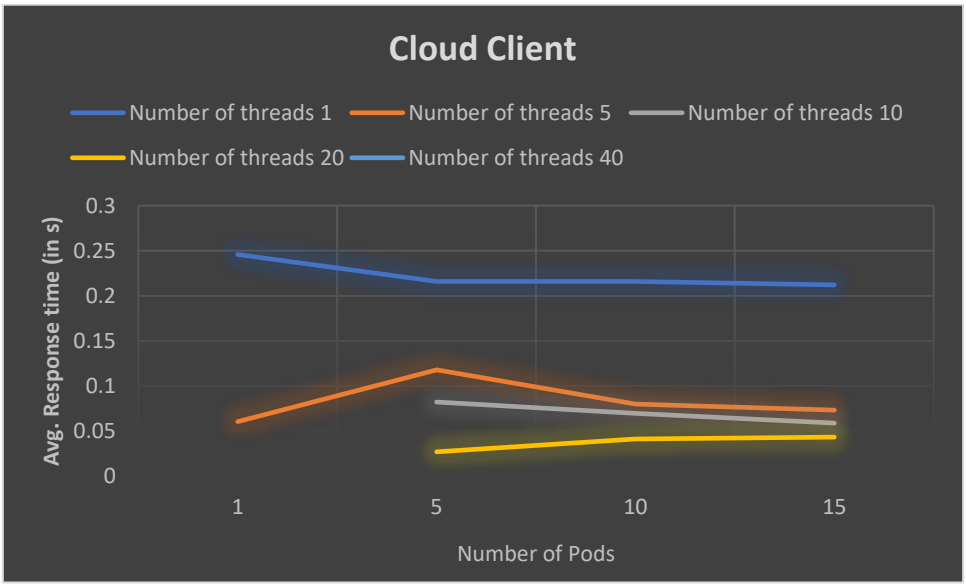
		Number of Pods			
		1	**5**	**10**	**15**
Number of threads	**1**	0.45208989	0.43131829	0.45156428	0.45208741
	5	#N/A	0.14790895	0.10749031	0.10306524
	10	#N/A	#N/A	0.08239143	0.06811572
	20	#N/A	#N/A	0.06487812	0.05829899
	40	#N/A	#N/A	#N/A	0.05846529

When there is a single pod deployed onto the cluster, we can see that the performance of the application is poor, since it is trying to create multiple worker threads which may cause it to crash. On a single pod, the application usually handles requests sequentially and this can increase the waiting time significantly as the application is large. For the local system on a

single thread, we see that the average response time is approximately 0.5 seconds. Consequently, for a multithreaded application, the average waiting time is about 0.1 seconds.

The Average Response Time of the web service versus the number of pods for different number of threads for the cloud client:

		Number of Pods			
		1	**5**	**10**	**15**
Number of threads	**1**	0.24581434	0.21597566	0.21589253	0.21208153
	5	0.06037909	0.1178364	0.07993112	0.07320254
	10	#N/A	0.0821131	0.06944831	0.0587697
	20	#N/A	0.02689238	0.04112122	0.04320747
	40	#N/A	#N/A	#N/A	#N/A



Irrespective of the number of pods, a single-threaded application shows a similar trend in the average response rate. This indicates that performance does not depend solely on the worker threads but on other factors like system resources, number of pods, network traffic and so on. We can see a slight decrease in the average response time in the cloud client which we may attribute to the fact that the client is a part of the server network which reduces traffic and latency.

The range of average response time implies that our system is quite efficient. Even at the highest amount of load on a small number of pods, the average response time did not cross the 0.3-second mark.

During the experiment, as seen in the above table, we were unable to calculate the average waiting time for some configurations as subsequent readings for them were inconsistent and produced more errors than output, making it difficult and so have been marked as "#NA". The results that we have chosen to consider are from configurations that proved to be at least 80 per cent successful.

The worst performance was observed when we significantly increased the number of threads on any number of pods, causing them to crash and reboot on several occasions. This could be due to there being a larger threading overhead than there are jobs available exhausting the resources required. We can also infer from the experiments that the average response rate is inversely proportional to the number of pods as the time reduces for the increasing pods. We also see that after a certain level, the response rate in all of them stays constant which shows that depending on the complexity of the application, after a certain number of pods, adding more will not be of any advantage to the system.

Distributed system challenges:

Distributed systems are necessary for the modern era considering the current state at which the scale of computing performance is increasing alongside the increase in requirements of resources. The performance requirements for the types of applications that are currently present would be immense to be contained within any one single instance. As important as the concept is, there are several factors that pose a challenge to distributed systems and several of them can be considered while building the application for this assignment. Some of these factors of this application that I considered as challenges during deployment were – **Scalability, Failure Handling and Quality of Service**. Scalability refers to the demand for the system features by its users and how they can change rapidly and one of the main things that this application needs to overcome is how to accept multiple client requests without getting overwhelmed. As this application is in the domain of image processing, it also requires a lot of resources as it needs to detect the various different objects in each image and so needs to process the images carefully. To overcome this challenge, we deploy the use of Docker and Kubernetes and run this application on a cloud service provider. This allows our application to be containerized and put into a cluster that has several nodes on which this application can run efficiently. Using this architecture, scaling up an application is a lot easier than traditional systems as nodes can be added or removed as required and various pods are used to run the application. Using this, we can also handle any crashes or failures that may occur. This application is deployed on a cluster than contains a master node and two worker nodes. All three nodes work together to facilitate load balancing, and in case one of the nodes crashes, the other node can still continue to work and the application can run without there being any downtime to fix them thus reducing any inconveniences the users might have faced otherwise. The use of several pods in each node adds another layer of abstraction and isolates any issues that are faced by that pod. Another important challenge is to provide quality service to our users. On a distributed system, it is hard to maintain non-functional requirements that better the Quality of service provided by an application. Some of the aspects of quality of service that the system tries to address are reliability, adaptability, performance and availability. The use of multiple nodes and pods for the deployment of the application provides users with a backup in case of any adversities. The use of docker ensures that the application is adaptable on any platform regardless of the type

of architecture being used as the packages and necessary requirements for the application to run are within the image. The performance of the application can be scaled with ease depending on the usage.

References:

Toosi ,A,N .,(2022).WEEK 1: Introduction to cloud computing from distributed computing to cloud computing [Lecture notes]

Retrieved from https://lms.monash.edu/mod/resource/view.php?id=10060855