FIT5225 – Cloud computing and security

Creating and Deploying an Image Object Detection Web Service(iWebLens) within a Containerised Environment

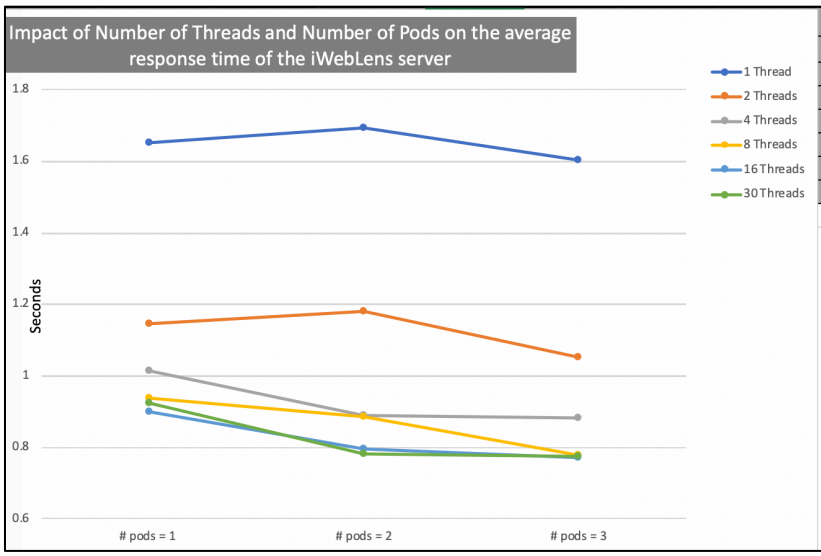
Full Name: Farhad Ullah Rezwan
Tutor Name: Shashikant Ilager
Login Name: frez0003
Student Number: 30270111

The below line chart depicts the impact of the average response time of the iWebLens server for processing images when client requests for the number of threads and the number of pods for a single worker node in the Kubernetes cluster.

With a single thread and irrespective of the number of pods, the response time average for all the images processed together was the poorest, which is more than 1.60 seconds per image. The reason behind poor performance for the single thread is the upload time of images. The pods sit idle for a portion of time while the images are being uploaded, because each image is uploaded first then processed, and then returns to the client-side, and then the second image is uploaded and processed.

The number of thread equals 1 and the number of thread equals 2 appears highly co-related. Even though there is a similar trend, the average response time has a significant difference which is below 1.2 seconds for 2 threads.

For single pod deployment as the number of threads increases from 1 thread to 30 threads, the average response time of the processes decreases from 1.65 seconds to 0.92 seconds. The single pod deployment was not able to handle the load of such thread intensive tasks and pods tend to hang due to overutilization of processes during the test when the thread number was 32.



Average response time data			
# Threads	# pods = 1	# pods = 2	# pods = 3
1	1.6517465	1.6920277	1.6035306
2	1.1443443	1.1805537	1.0513144
4	1.0130594	0.8904087	0.8808597
8	0.9362003	0.88601	0.7767007
16	0.9003718	0.7952421	0.7700346
30	0.9233101	0.7827534	0.7748213
32	Error	0.7724535	0.7656893

From the above experiments it is evident that the image recognition process for the iWebLens server in the Kubernetes cluster is not suitable for the load of client-side requests when the number of threads is 1 or the number of threads is 2 and the number of pods is 1. So now we can give emphasis on when thread number is 4 or more and the number of pods is 2 and 3.

When the server scales up with number of pods of 2, the performance of the service is best at 32 threads as the average processing time for the images is 0.77 seconds, and, we can see a similar performance when the number of thread is 30 and 16 at 0.79 and 0.78 seconds respectively. In addition, when the number of pods is 3, the average response time is at its peak at 16 threads with 0.77 seconds and we can see a similar performance for 8 threads, and 30 threads with 0.78 and 0.76 seconds respectively. So, we can say that even though we increase the number of threads, the performance remains same.

The experiments proof the Amdahl's Law as well, we can see adding pods does not mean we are getting efficiency. For example, for 16 and 30 threads, when we scale up or down between 2 and 3 pods, the performance of average response time remains almost same at 0.77 to 0.79.

So, from the above experiment, in my analysis I would use pod size of two and number of threads per 128 requests to 16 to balance out proper resource utilisation and processor utilisation.