FIT5225-S1 2024 Assignment 1

The CloudDetect Project

Name: Peichun Shih

Student ID: 33475881

Tutor: Jay Zhao, Qifan Deng, Jinchun Du

Table of Content

I. The following table demonstrates the experiment results.

	Nectar		Master	
# of pods	Max Users	Avg. Response Time (ms)	Max Users	Avg. Response Time (ms)
1	1	972.56	1	2014.14
2	1	932.25	2	1963.32
4	1	930.77	2	1951.69
8	2	969.11	3	1963.03

II. Results Explanation and Experiments Observation

The aim of the experiments is to evaluate the performance of Kubernetes distributed system under the Nectar virtual machine and the Master local machine across various scenarios. The experiments test the system by assessing their ability to handle the maximum number of users and measuring the average response time with different configurations. The table above shows the results of the experiments. The following insights and observations are gained from the results.

1. **Enhanced Performance in Average Response Time Perspective:**

   It is stated that pod deployment metrics significantly influence the performance of Kubernetes cluster (Ihuman, 2024). In both Nectar and Master systems, the experiments demonstrate the trend that adding more pods to the cluster results in better performance. Specifically, in the Nectar experiment, it indicates a consistent decrease in average response time, from 972.56ms, 932.25ms, to 930.77ms, as the number of pods increases from 1, 2, to 4 respectively, even though the maximum number of users remains at 1. From the local machine, the comparison between 2 pods and 4 pods reveals that although the maximum number of users is the same, the performance of the 4-pod setup is better than the performance of the 2-pod setup. This implies that scaling up the system leads to improved performance.

2. **Enhanced Performance in Maximum Number of User Perspective:**

   Previous research has concluded that increased user access to the same system in Kubernetes is more likely to cause system breakdowns, and a greater number of pods within a cluster can accept more user access (Learnk8s, n.d.), which aligns with the experiment results. The results suggest that increasing the number of pods generally allows the systems to accommodate more users. The maximum number of users in the Nectar system remains at 1 for 1-pod, 2-pod, and 4-pod setup, while it creases to 2 when 8 pods are utilised. Similarly, in the Master system, the maximum number of users ranges from 1 to 3 as the number of pods increases.

3. **The Comparison between the Nectar Virtual Machine and Local Machine:**

   A notable difference between running the application in Nectar and the Master local machine is their average response time. According to prior study, one of the advantages of using virtual machines is that it is independent from developers' hardware, and it is not restricted by the speed of the hardware (Eğilmez, 2022).

Consequently, the Nectar system signifies lower average response time, which suggests better resource utilisation. In addition, when examining the web service in the Nectar system, locust is the only running application, resulting in less interference from other applications or software and further leading to less response time. On the other hand, the Master system demonstrates the higher average response time, which may result from two reasons. First, when assessing the performance, there are other application or website running in the local machine, which is likely to extend the average response time. Moreover, the increased overhead due to managing more users through pod scaling may also slightly lengthen the response time. In contrast, the capacity to handle additional users is more limited in the Nectar system. This trade-off emphasises the importance of evaluating distributed systems regarding both performance and scalability to satisfy specific use cases and requirements of workload.

Task allocation system, namely the load balancing mechanism, is highly concerned and focused when it comes to cloud computing (Ullah et al., 2021). However, based on the experiments, the load balancing mechanism in the local machine is superior to that in Nectar virtual machine since the local machine dynamically adjusts to the changing number of users and workloads and further causes fewer pod crashes and failures compared to the Nectar virtual machine. This suggests that the Master system is more flexible and scalable. Additionally, it also implies that the local machine is equipped with better stability and reliability in managing different workloads and user demands.

The findings from the experiment highlights the difference between running the distributed systems on the local machine and Nectar virtual machine. It provides valuable implications and insights to the real-world scenarios. The requirements for scalability, performance, and resource utilisation must be thoroughly considered when choosing a distributed system architecture. For instance, the Nectar virtual machine will be a better choice when the application needs stable performance with predictable or fixed workloads. It is reliable under steady usage patterns. On the contrary, a local machine might be preferable for the application with fluctuating workloads that demand dynamic scaling to accommodate varying user demands. It ensures optimal performance with less possibility to compromise resource efficiency even during peak periods. As a result, these experiments demonstrate the potential strategies to optimise performance and resource utilisation under various circumstances to meet specific needs and demands.

III. The challenges in the CloudDetect Project

In this project, three critical challenges have been identified— scalability, security, and failure handling. These challenges are vital for guaranteeing a smooth and reliable operation of the object detection web service. The illustration of each challenge and its corresponding solution is as follows.

1. Scalability

Scalability is commonly defined as the system's ability to manage a growing workload by allocating additional necessary resources efficiently and effectively

(Lehrig et al., 2015). In the CloudDetect project, scalability is essential to ensure that the system can navigate through varying rates of demands or fluctuating number of users. Accordingly, it is imperative for the distributed system to enhance scalability, which may mitigate the occurrence of failures. For example, as more users upload images for object detection, the system must scale to accommodate the increased requests dynamically and seamlessly. It involves the system's capability to quickly allocate additional resources, such as computing power or memory, to process the growing workload without compromising performance.

Since this project has already implemented Kubernetes, which serves as the container orchestration system that allows to automatically provision additional pods to handle the workload, another potential solution to improve scalability is the application of Horizontal Pod Autoscaler (HPA), as explained by Tiutiu & Gupta (2024). HPA adjusts the number of replica pods in a deployment based on its observation of CPU utilisation or custom metrics defined by users. The custom metrics can be specifically tailored to align with the workload patterns of CloudDetect, such as request volumes. By configuring HPA, the system is equipped with the ability to scale pods more precisely to satisfy demand, and further optimising resource allocation and reducing errors.

2. Security

Ensuring data security is essential in distributed systems when managing sensitive and private information, such as images. It is associated to maintaining the confidentiality, integrity and availability of data while protecting it from malicious attacks or unauthorised access (Sun et al., 2019). Therefore, security measures should be implemented to protect image data and the system integrity in CloudDetect project.

To address the challenge, robust access control mechanism should be utilised within Kubernetes. The mechanism is adopted to restrict access to sensitive resources and forbid unauthorised users from entering the system (K & Priya S, 2012). By employing access controls, CloudDetect can defend itself against potential security breaches and unauthorised access attempts. This proactive approach to security assists to ensure the data integrity and confidentiality of sensitive images in the CloudDetect project.

3. Failure Handling

Failure handling is related to effectively managing system failures, such as hardware crashes or software errors, to make sure the system is always available and reliable (Gill & Buyya, 2018). In CloudDetect project, avoiding interrupted service is crucial to offer users a seamless experience. Hence, implementing strategies for failure mitigation and recovery are vital to maintain the reliable system and user satisfaction.

Although Kubernetes already offers built-in capabilities to deploy multiple replicas and automatically redistribute workload, there are alternative approaches to tackle this challenge. For instance, a comprehensive disaster recovery plan can be applied to the project (Jay, 2023). The plan includes actions, such as backing up critical data, replicating data to dispersed locations, and restoring services in the event of an outage. This will not only evaluate the effectiveness of the plan but ensure

CloudDetect's readiness to respond to unforeseen incidents. Moreover, failure recovery mechanisms can be conducted through custom scripts. The scripts can be designed to set up health checks for the pods and define recovery actions that should be taken while failures are detected. The recovery plan improves the system's resilience and reduces downtime.

Overall, the CloudDetect project involves three challenges identified: scalability, security, and failure handling. By adopting HPA for dynamic scaling, access control mechanism to improve security, and the recovery plan to handle failures, the web service can manage different rates of requests, protect user data, maintain uninterrupted service, and deliver a reliable and secure object detection application.

Appendix

The attempt/utilisation of ChatGPT

- object_detection.py
  Attempt 1: Help me to encode an image into text is using the base64 method.
  Attempt 2: Provide instruction to complete the main method.
  Attempt 3: How to prepare for the response?

- Dockerfile
  Attempt 1: How to specify python base image.
  Attempt 2: Provide the instruction for the CMD.

- my-development.yml
  Attempt 1: How to specify CPU and memory limits?

- service.yml
  Attempt 1: How to specify NodePort?

- locust.py
  Attempt 1: How to specify the host of ImageUser?
  Attempt 2: How to decode data? How to define headers?
  Attempt 3: How to catch exception, print the result and reduce the failures?

References

Eğilmez, İ. (2022, August 1). Developing in the Cloud vs. Developing in Local. Medium. https://medium.com/@i.egilmez/developing-in-the-cloud-vs-developing-in-local-14b6f31bc07a

Gill, S. S., & Buyya, R. (2018). Failure Management for Reliable Cloud Computing: A Taxonomy, Model and Future Directions. Computing in Science & Engineering, 1–1. https://doi.org/10.1109/mcse.2018.2873866

Ihuman, A. (2024, January 22). Monitoring Kubernetes Performance: Key Metrics to Know. Medium. https://medium.com/@Anita-ihuman/monitoring-kubernetes-performance-key-metrics-to-know-19f5452d28a6

Jay. (2023, November 30). Ensuring Resilience: A Guide to Fault Tolerance, High Availability, and Disaster Recovery. Medium. https://medium.com/@seaflux/ensuring-resilience-a-guide-to-fault-tolerance-high-availability-and-disaster-recovery-daf425e32124

K, P., & Priya S, J. (2012). Analysis of Different Access Control Mechanism in Cloud. International Journal of Applied Information Systems, 4(2), 34–39. https://doi.org/10.5120/ijais12-450660

Learnk8s. (n.d.). Architecting Kubernetes clusters — how many should you have? Learnk8s. Retrieved April 15, 2024, from https://learnk8s.io/how-many-clusters

Lehrig, S., Eikerling, H., & Becker, S. (2015). Scalability, Elasticity, and Efficiency in Cloud Computing. Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures - QoSA '15. https://doi.org/10.1145/2737182.2737185

Sun, Y., Zhang, J., Xiong, Y., & Zhu, G. (2019). Data Security and Privacy in Cloud Computing. International Journal of Distributed Sensor Networks, 10(7), 190903. Sagepub. https://doi.org/10.1155/2014/190903

Tiutiu, & Gupta. (2024, February 1). Scaling applications in Kubernetes using the HorizontalPodAutoscaler. DigitalOcean | Cloud Infrastructure for Developers. https://www.digitalocean.com/community/developer-center/scaling-applications-in-kubernetes-using-the-horizontalpodautoscaler

Ullah, A., Nawi, N. M., & Ouhame, S. (2021). Recent advancement in VM task allocation system for cloud computing: review from 2015 to2021. Artificial Intelligence Review, 55(3), 2529–2573. https://doi.org/10.1007/s10462-021-10071-7

OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. https://chat.openai.com/chat