SOFE 4790U: Distributed Computing

Date: October 9th, 2022

Lab 2: Scalability & Traffic Management

Zachary Carman

Discussion:

Summarize the problem, the solution, and the requirements for the pattern given in part 1. Which of these requirements can be achieved by the procedures shown in parts 2 and 3?

The main problem is when clients need to be able to handle multiple services and change service instances. The client will need to be updated are services are added and removed as the user entered information changes. The client will also be updated when instances are created or destroyed, as well as be able to handle multiple versions of the same service.

The solution to this problem is to use a gateway as an intermediate between the customer and the services. This makes it so the client will only need to know about one endpoint. This gateway can be able to interact with multiple different services allowing the client to not have to be updated. This is a similar case for both multiple instances of the same service or different versions of the same service. The gateway will handle the interaction with these services so the clients don't need to be updated. There are a few requirements for this pattern. First is the client will need to be able to handle many services behind a gateway and only be accessible by a single endpoint. The requests need to be ran from external addressable endpoints then to internal virtual endpoints. The client will need to be able to consume a service running in many regions or the same region. Finally, the clients will need to be able to handle multiple versions of the service at one time.

Design:

Autoscaling is another pattern that can be implemented by GKE. Your task is to configure a Yaml file that autoscaling the deployment of a given Pod. Why autoscaling is usually used? How autoscaling is implemented? How autoscaling is different than load balancing and request splitter?

Auto-scaling is used to dynamically increase resources as required to commute an operation. This is done to ensure availability during spikes in traffic. Auto-scaling also has better fault tolerance by creating new instances upon failure. Auto-scaling as has better availability as it will increase capacity upon increase in demand. This dynamically changing use also ends up being more cost effective as there is never any wasted resources. This also then helps performance as the system can scale to meet and performance indicator.

- Auto-scaling is implemented by increasing resources and replicas as demand raises. This allows resource to be dynamic to handle spikes and dips in demand. Auto-scaling uses virtualization which is where a user defines a virtual instance. The main difference between auto-scaling and load balancing and request splitter is that auto-scaling is dynamically creating and removing resources whereas the other two are not. Load balancing just evens out the computational load between resources and the request splitter splits the request across establish resources. This is different from auto-scaling which will dynamically change resource requirements are the load changes.

Deliverables:

1. A report that includes the discussion and the design parts.
See above

2. An audible video of about 5 minutes maximum showing the final results of following the lab steps. It should include showing the deployments, services, and pods created in the lab with their test cases (step 6 in part 2, step 3 in part 3).
https://drive.google.com/file/d/1_iVjcgdpKKQmBOZcOvSY_-qP3PhaRHGk/view?usp=sharing

3. Another audible video of 3 minutes maximum shows the configuration and implementation of autoscaling using GKE.

https://drive.google.com/file/d/1Wm2egnBYgi4v_pDPCaNchME7BMrcOizJ/view?usp=sharing