UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

SISTEMAS OPERATIVOS 1

ING. SERGIO ARNALDO MENDEZ AGUILAR

AUX. LEONEL AGUILAR

AUX. CARLOS RAMIREZ

Manual Técnico

Nombre	Carné
Cristian Francisco Meoño Canel	201801397
Josue Guillermo Orellana Cifuentes	201801366
César Alejandro Sosa Enríquez	201800555

Contenido

Modelo de Base de Datos

No se implementó un modelo de base datos ya que solamente se implementaron bases de datos NoSQL.

Descripción de Herramientas

Gorrutinas

Podemos definirlas como hilos ligeros que son administrados por Go en tiempo de ejecución.

Ingress

Permite definir que una aplicación va a responder en una determinada URL y se va a encargar de gestionar el balanceo entre las distintas instancias. El Ingress puede ser implementado de diversas formas pero una de las más utilizadas es a través de nginx.

React

Es una librería open source de JavaScript para desarrollar interfaces de usuario. Fue lanzada en el año 2013 y desarrollada por Facebook, quienes también la mantienen actualmente junto a una comunidad de desarrolladores independientes y compañías.

Kafka

Apache Kafka es una plataforma distribuida de transmisión de datos que permite publicar, almacenar y procesar flujos de registros, así como suscribirse a ellos, de forma inmediata.

RabbitQM

Es un software de encolado de mensajes llamado bróker de mensajería o gestor de colas. Dicho de forma simple, es un software donde se pueden definir colas, las aplicaciones se pueden conectar a dichas colas y transferir/leer mensajes en ellas.

Pubsub

Es un sistema de búsqueda que notifica la noticias e información nueva que concuerda con la búsqueda.

Linkerd

Es un proxy de red open source diseñado para ser desplegado como Service Mesh y que está basado en finagle y netty. Su principal cometido es hacer de link, como su nombre indica, entre las diferentes piezas de sistemas distribuidos y es un buen compañero para nuestras arquitecturas de microservicios, muy en boga en estos días.

Prometheus

Es un sistema de monitoreo de código abierto basado en métricas. Recopila datos de servicios y hosts mediante el envío de solicitudes HTTP en puntos finales de métricas. Luego, almacena los resultados en una base de datos de series de tiempo y los pone a disposición para análisis y alertas.

Grafana

Es una herramienta para visualizar datos de serie temporales. A partir de una serie de datos recolectados obtendremos un panorama gráfico de la situación de una empresa u organización.

Go

Es un lenguaje de programación creado en el año 2007 por Google. Go nació sobre todo para mejorar la concurrencia que otros lenguajes ya existentes como Python, Java o C/C++ no eran capaces de manejar correctamente. El software es cada vez más complejo y tiene que hacer más cosas simultáneamente.

Redis

Que significa Remote Dictionary Server, es un rápido almacén de datos clave-valor en memoria de código abierto.

MongoDB

Es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico.

GRCP

Es un moderno sistema de llamada a procedimiento remoto que procesa la comunicación en estructuras cliente-servidor distribuidas de manera especialmente eficiente gracias a una innovadora ingeniería de procesos.

Kubernetes

Es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios. Kubernetes facilita la automatización y la configuración declarativa.

Chaos Mesh

Es un proyecto alojado en Cloud Native Computing Foundation (CNCF) . Es una plataforma de ingeniería del caos nativa de la nube que organiza el caos en los entornos de Kubernetes.

Preguntas de Reflexión

1. **Cómo funcionan las métricas de oro, cómo puedes interpretar las 7 pruebas de faulty traffic, usando como base los gráficos y métricas que muestra el tablero de Linkerd Grafana.**
    a. Tasa de éxito: es el porcentaje que representa el éxito de los request que se hacen.
    b. Solicitudes por segundo: la cantidad de solicitudes que se envían por segundo, también llamado concurrencia.
    c. Latencia: El tiempo que se demora en ser resuelta la request.

    1. Prueba 100% del tráfico en Kafka con una concurrencia de 25:
        - La tasa de éxito se mantiene en 100%.
        - Las solicitudes por segundo llegan como máximo a 15.
        - La latencia tiene como valor máximo 2 segundos.

        Prueba 100% del tráfico en Kafka con una concurrencia de 50:
        - La tasa de éxito se mantiene en 100%.
        - Las solicitudes por segundo llegan como máximo a 38.
        - La latencia tiene como valor máximo 2 segundos.
    2. Prueba 100% del tráfico en Pubsub con una concurrencia de 25:
        - La tasa de éxito se mantiene en 100%.
        - Las solicitudes por segundo llegan como máximo a 23.
        - La latencia tiene como valor máximo 250 milisegundos.

        Prueba 100% del tráfico en Pubsub con una concurrencia de 50:
        - La tasa de éxito se mantiene en 100%.
        - Las solicitudes por segundo llegan como máximo a 43.
        - La latencia tiene como valor máximo 270 milisegundos.
    3. NO TENEMOS RABBIMQ
    4. Prueba 50% del tráfico en Kafka y 50% de error con una concurrencia de 25:
        - La tasa de éxito se mantiene en 50% aproximadamente.
        - Las solicitudes por segundo llegan como máximo a 7.
        - La latencia tiene como valor máximo 2 segundos.

        Prueba 50% del tráfico en Kafka y 50% de error con una concurrencia de 50:
        - La tasa de éxito se mantiene en 50% aproximadamente.
        - Las solicitudes por segundo llegan como máximo a 18.
        - La latencia tiene como valor máximo 2.1 segundos.
    5. Prueba 100% del tráfico en Pubsub y 50% de error con una concurrencia de 25:
        - La tasa de éxito se mantiene en 50% aproximadamente.
        - Las solicitudes por segundo llegan como máximo a 11.
        - La latencia tiene como valor máximo 250 milisegundos.

        Prueba 100% del tráfico en Pubsub y 50% de error con una concurrencia de 50:
        - La tasa de éxito se mantiene en 50% aproximadamente.
        - Las solicitudes por segundo llegan como máximo a 21.

- La latencia tiene como valor máximo 280 milisegundos.

6. NO TENEMOS RABBIMQ
7. Prueba 50% Kafka y 50% Pubsub con una concurrencia de 25
   - La tasa de éxito se mantiene en 100%
   - Las solicitudes por segundo llegan como máximo a un promedio de 100 milisegundos.

   Prueba 50% Kafka y 50% Pubsub con una concurrencia de 50
   - La tasa de éxito se mantiene en 100%
   - Las solicitudes por segundo llegan como máximo a un promedio de 110 milisegundos.

2. Menciona al menos 3 patrones de comportamiento que hayas descubierto en las pruebas de faulty traffic.

- Kafka: La latencia parece tener como límite superior los 2 segundos y fracción ya que en ninguna de las pruebas que realizamos se llegó a los 3 segundos.
- Pubsub: La latencia comienza a subir hasta que se la concurrencia es mayor a 50.
- En el 50% Kafka y 50% Pubsub tienen un mejor desempeño en conjunto.

3. ¿Qué sistema de mensajería es más rápido? ¿Por qué?

- Pubsub tuvo el mejor de los desempeños en las pruebas realizadas, esto podría deberse a que no está dentro del Cluster de Kubernetes por lo que no debe compartir los recursos con los demás pods y este desempeño está administrado por Google.

4. ¿Cuántos recursos utiliza cada sistema de mensajería?

- CPU: (en miliunidades)
  - PubSub consume 40m por Pod de la API y el Worker entre 125 y 175
  - Kafka consume 10 m por Pod de la API y el Worker 150 m adicional se le suman 75 de Strimzi
- RAM:
  - PubSub consume 70mb por Pod de la API y el worker 100mb
  - Kafka consume 20mb por Pod de la API y el Worker 325 mb adicional 800mb de Strimzi, esto irá creciendo don los mensajes
- Conclusión:
  - PubSUb y Kafka en terminos de CPU son similares.
  - En RAM Kafka consume muchísimos más recursos que PubSub.

5. ¿Cuáles son las ventajas y desventajas de cada servicio de mensajería?

	Kafka	Pubsub
Ventajas	Tiene mejor persistencia de datos. Es fácil instalarlo en Kubernetes con Strimzi.	Es administrado por Google. Hay bastante documentación con la que nos podemos apoyar para hacer la implementación.
Desventajas	La latencia se ve afectada directamente por lo recursos del Cluster. Consume muchos recursos del Cluster.	Depende de GCP, o sea que no se puede migrar.

6. **¿Cuál es el mejor sistema de mensajería?**
    - Pubsub debido al rendimiento superior que nos ofrece sin la necesidad alta de recursos que se nos requiere para tener un rendimiento parecido desde Kafka.
7. **¿Cuál de las dos bases de datos se desempeña mejor y por qué?**
    - Redis porque es una base de datos que está montada sobre la ram entonces tiene acceso a los datos de manera más rápida a diferencia que Mongo que está montado sobre el disco y debe de utilizar las estructuras tradicionales de bases de datos.
8. **¿Cómo se reflejan en los dashboards de Linkerd los experimentos de Chaos Mesh?**
    - Pod Kill: Elimina el pod del dashboard de linkerd y luego de unos segundos aparece un nuevo pod.
    - Pod failure: No elimina el pod del dashboard pero lo deja sin métricas, lo cual lo hace inaccesible por un tiempo.
    - Container Kill: tiene el mismo efecto que pod kill pero kubernetes tarda más en identificarlo porque lo verifica mediante el livenessProbe.
    - Network Emulation (inyección de latencia): Se observa un aumento en el tiempo de respuesta para las request.
    - DNS Chaos: Se observan fallas esporádicamente, es como si fuese un request kill. O se hace bien o se hace mal.
9. **¿En qué se diferencia cada uno de los experimentos realizados?**
    - Pod Kill: mata un pod y kubernetes automáticamente lo reemplaza.
    - Pod Failure: inhabilita un pod durante un tiempo definido.
    - Container Kill: mata un contenedor dentro de un pod, si el container tiene livenessProbe este se reinicia de lo contrario se mantiene muerto.
    - Network Emulation: agrega un delay a la request.
    - DNS Chaos: devuelve una IP random de forma aleatoria cuando se solicita algo al servidor DNS.
10. **¿Cuál de todos los experimentos es el más dañino?**
    - Container Kill si es que está mal configurado el livenessProbe Kubernetes no tendría forma de saber que ese contenedor no existe dentro del pod, por lo cual no lo reiniciaría.
    - DNS chaos: puede llegar a provocar un error fatal dentro del container. Por ejemplo que regrese una IP random para Pubsub y que no se pueda establecer la conexión.