Laboratorio 5: Sistemas Distribuidos

Profesor: Jorge Díaz
Ayudantes de Lab: Iñaki Oyarzun M. & Debora Alayo

Junio 2024

1 Objetivos del laboratorio

- Familiarizarse con los conceptos asociados a la **replicación y consistencia de datos** en Sistemas Distribuidos.
- Aplicar consistencia eventual en un sistema distribuido.
- Aplicar modelos de consistencia centrados en el cliente.
- Profundizar el uso de **Golang y gRPC**.

2 Introducción

Dentro de los sistemas distribuidos, la replicación pasa a ser una buena alternativa para manejar la tolerancia a fallos, además de beneficiarse en cuanto al rendimiento debido a la división de la carga entre las diferentes réplicas que se manejen del sistema, o también acercar geográficamente los datos a los clientes que van a acceder a ellos. Sin embargo, como gran tradeoff de su implementación es que se producen problemas de consistencia, ya que, al actualizarse una réplica, esta se vuelve diferente a las otras. Es por ello que, para solucionarlo, se han propuesto los modelos de consistencia.

Para poner esto en práctica, se propone el siguiente problema en donde haciendo uso de gRPC y golang para llevar a la práctica la replicación, en donde también se tendrán que aplicar modelos de consistencia centrados en los datos y el cliente. Para este laboratorio, se hara uso del modelo de consistencia eventual y Monotonic Reads junto a Read Your Writes.

3 Tecnologías

- El lenguaje de programación a utilizar es **Go** y **Docker**
- Para las comunicaciones se utilizará **gRPC**

4 Laboratorio

4.1 Contexto

El Imperio T'au, guiado por el ideal del "Bien Supremo", ha identificado el planeta Vior'la como un objetivo estratégico crucial en su expansión galáctica. Vior'la, un planeta fortaleza, está bajo el control de una fuerza enemiga bien organizada y fuertemente armada. Su conquista es vital para asegurar rutas de suministro y expandir la influencia del Imperio T'au en la región.

La tarea de preparar el asalto recae sobre el comandante Shas'o T'au Kais, un líder militar conocido por su astucia y habilidad en el campo de batalla. Sin embargo, la clave para una operación exitosa radica en la información precisa y oportuna sobre las defensas enemigas. Aquí es donde entran los ingenieros T'au de la Casta de la Tierra, Fio'ui Jeth'kann y Fio'vre Mal'kor.

Los ingenieros han desplegado una red avanzada de drones de reconocimiento y sensores en la órbita de Vior'la. Estos dispositivos están recolectando datos críticos sobre las posiciones enemigas, incluyendo fortalezas, puntos de suministro y patrullas.

El éxito de la operación depende de la capacidad de los ingenieros para transmitir esta información al comandante Kais de manera precisa y oportuna. Kais, con esta información en mano, podrá planificar el asalto con precisión quirúrgica, asegurando que las fuerzas T'au ataquen en los puntos más vulnerables del enemigo.

Su objetivo como ingenieros novatos dentro de la linea de batalla será llevar a cabo el despliegue de la red que facilite este intercambio de mensajes de forma continua entre estos ingenieros y el comandante Kais. Su desarrollo determinará el éxito de esta misión.

4.2 Explicación

4.2.1 Arquitectura del sistema

El sistema distribuido para este laboratorio se compone de 5 entidades por un lado los ingenieros Jeth y Malkor, el comandante Kais, el Broker y el conjunto de servidores fulcrum.

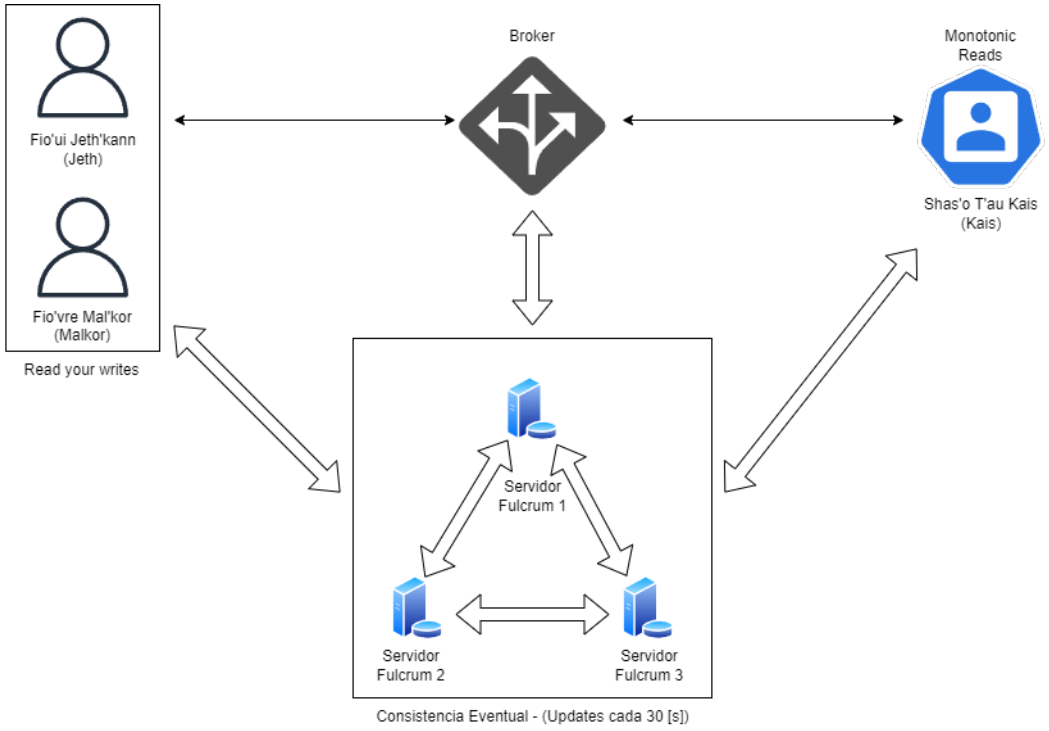

Figure 1: Arquitectura del sistema distribuido

4.3 Servidores Fulcrum

Serán 3 servidores los cuales comparten las mismas funciones. Son réplicas que ayudan a mejorar la tolerancia a fallas del sistema ante eventuales tormentas solares. Estos servidores deberán hacer lo siguiente:

- Almacenar registros de soldados enemigos dentro de los sectores de Vior, los cuales se guardan en el siguiente formato:

  `nombre_sector nombre_base cantidad_de_enemigos`
  Ejemplo:
  `SectorAlpha Campamento1 5`

  Nota: Debe crearse un archivo por cada sector, es decir, el ejemplo anterior modifica o crea un archivo 'SectorAlpha.txt'

- Cada uno de estos archivos, esta asociado a un reloj de vector. Este reloj debe tener la forma **[x,y,z]**, donde la posición en el vector indicará el servidor Fulcrum al que se referencia y el valor en la posición indicará la cantidad de cambios hechos a esa posición:
  - **X :** Servidor Fulcrum 1
  - **Y :** Servidor Fulcrum 2
  - **Z :** Servidor Fulcrum 3

- Cada servidor Fulcrum deberá llevar un **Log de registro** junto a los registros de enemigos, la función que cumplirá este log será la de registrar los cambios (AgregarBase - BorrarBase - RenombrarBase - ActualizarValor) que se hayan hecho en el. Una vez que se hayan propagado los cambios entre los servidores fulcrum, estos **deben** ser borrados y nuevamente creados.
  La razón detrás de estos archivos es que pueda ser de ayuda para la resolución de eventuales conflictos durante la propagación de los cambios. El log lleva la siguiente estructura:

  `<acción> <SectorAfectado> <BaseAfectada> <Nuevo Valor>`
  Ejemplo:
  `AgregarBase SectorAlpha Campamento2 13`
  `RenombrarBase SectorAlpha Campamento2 CampamentoAlpha`
  `ActualizarValor SectorAlpha CampamentoAlpha 25`
  `BorrarBase SectorAlpha CampamentoAlpha`

  Nota: El comando BorrarBase no tiene por qué recibir un nuevo valor ya que la base afectada es la que se borra.

- Mantener **consistencia eventual**, realizando propagación de los cambios cada **30 segundos**.

- Cuando se presenten problemas de consistencia entre las versiones de las réplicas, se deberá realizar un **Merge** entre los servidores, lo cuál se debe llevar a cabo de forma automática. Para ello, serán útiles los registros de log antes mencionados.

4.4 Broker

Es el balanceador de la carga para los servidores Fulcrum, realiza las siguientes tareas:

- Redirige a los Ingenieros y al Comandante a uno de los servidores Fulcrum de forma **aleatoria** por medio de una dirección hacia el servidor fulcrum a conectar para obtener la información.
- Cuando alguna entidad informa de una inconsistencia, indicará una réplica en específico basado en el reloj entregado al Broker por la entidad.

4.5 Los Ingenieros

Los ingenieros son 2: Jeth y Malkor. Son los encargados de agregar información a los registros de enemigos de los servidores Fulcrum, además de actualizar o eliminar dichos registros. Para lo anterior disponen de los siguientes 4 comandos:

- **AgregarBase** `<nombre_sector> <nombre_base> [valor]`**:** Este comando creará una nueva linea en el registro del sector correspondiente. Si el sector aún no posee un archivo de registro de enemigos, se deberá crear uno nuevo. Este comando **puede no recibir un valor** indicando en ese caso que el registro para la base deberá ser con una cantidad 0 de soldados.
- **RenombrarBase** `<nombre_sector> <nombre_base> <nuevo_nombre>`**:** Con este comando se cambia el nombre a una base dentro del registro del sector.
- **ActualizarValor** `<nombre_sector> <nombre_base> <nuevo_valor>`**:** Con este comando se actualiza la cantidad de soldados de la oscuridad en la base indicada dentro del registro del sector.
- **BorrarBase** `<nombre_sector> <nombre_base>`**:** Con este comando se borra la linea de dicha base dentro del archivo de movimientos del sector.

Los comandos se envían al Broker Luna, el cual responde con la dirección de uno de los servidores Fulcrum aleatoriamente seleccionado. Posteriormente, los informantes se conectan al servidor Fulcrum y vuelven a enviar el mismo comando que le envió al Broker. La respuesta del servidor Fulcrum es el reloj de vector del registro de sector del sector modificado

Ambos ingenieros utilizarán el modelo de consistencia **Read your Writes**. Para poder llevar a cabo esta consistencia, los ingenieros pueden mantener en memoria la información de los registros de sectores que han modificado, junto con el reloj de vector del archivo del sector y la dirección del servidor Fulcrum al que se conectó por última vez a ese registro para corroborar esta consistencia.

4.6 Comandante Kais

Es quien realiza las consultas al Broker para obtener información sobre la cantidad de enemigos en las bases de los sectores que se le consulten. De esa forma, se define el siguiente comando:

- **GetEnemigos** `<nombre_sector> <nombre_base>`**:** Este mensaje se envía al Broker y recibe por respuesta la dirección de uno de los servidores Fulcrum para reenviar el comando y obtener la cantidad de enemigos que se encuentran en dicha base del sector consultado, junto al reloj vectorial del registro del servidor Fulcrum consultado.

El comandante tiene que hacer uso del modelo de consistencia **Monotonic Reads**, Para poder llevar a cabo esta consistencia, El comandante puede mantener en memoria la información de los sectores que ha consultado anteriormente, junto con el reloj de vector obtenido del sector y el servidor Fulcrum que le respondió por última vez al solicitar por esa base en ese sector para tener formas de corroborar la consistencia según la respuesta del servidor Fulcrum.

4.7 Consistencia Eventual

Para llevar a cabo las consistencias indicadas por el Laboratorio se hará uso de los "relojes de vectores" indicados anteriormente. Estos tienen el formato [x,y,z], donde la posición en el vector indica a cual servidor Fulcrum se hace referencia y el valor en la posición indica la cantidad de cambios que se han aplicado a dicho servidor en un archivo de sector en particular. Cada vez que se detecte que hubo cambios concurrentes en los archivos a partir de los relojes de vectores al momento de propagar los cambios, se deberá someter a un proceso de merge.

4.7.1 Merge

Al momento de hacer la replicación de los cambios entre los servidores Fulcrum, estos deben de asimilar los cambios que se han generado entre sí, denominándose aquello como un proceso de merge. Para realizar este proceso, se debe definir un nodo (servidor Fulcrum) dominante que sea de base para la aplicación de los cambios que se hayan efectuado en los otros nodos (servidores Fulcrum), pudiendo ser este proceso definido entre los tres servidores Fulcrum o por el Broker.

Utilizando los logs de cambios de los demás servidores Fulcrum, se podrá realizar una comparación de las operaciones que se hayan realizado por servidor, para que de esta forma, se puedan aplicar los cambios en el archivo del nodo dominante, el cual será la nueva versión que debe ser propagada a todas las réplicas (servidores Fulcrum). Cabe agregar además, que cuando ocurre un proceso de merge, los relojes de vectores deben ser actualizados para coincidir con esta nueva versión, por ejemplo, si se tienen los vectores:

```
[1,0,0] - [0,1,0] - [0,0,0]
```

Luego del proceso de merge los relojes de todos los servidores Fulcrum debieran ser:

```
[1,1,0]
```

Puesto que es posible que se puedan presentar conflictos por casos particulares, un ejemplo puede ser que se tenga un mismo registro de base y sector pero con valores distintos entre dos servidores fulcrum; En caso de presentarse, el método para la resolución es libre, sin embargo, tiene que obedecer el principio de consistencia (y ser registrado en caso de realizarse en el README), es decir, que finalizado el proceso de merge, no sigan estando diferencias entre servidores fulcrum a nivel de estructura de los registros de sectores. Puesto que se esta trabajando con el paradigma de consistencia eventual, el hecho que se "pierda" información no es un inconveniente.

En caso de la ocurrencia de otros casos particulares, estos deberán ser registrados como parte del README del entregable, ya que, son parte de decisiones de diseño que se hayan realizado como grupo.

5 Restricciones

- Todo uso de librerías externas que no se han mencionado en el enunciado debe ser consultado en aula.

- La distribución de las máquinas debe seguir las siguientes reglas:
  - Los 3 procesos de servidores Fulcrum deben estar en 3 máquinas diferentes
  - El broker debe estar en un servidor diferente a los servidores Fulcrum
  - El comandante, Malkor, Jeth y El broker deben estar en máquinas diferentes

6 Consideraciones

- **Prints por pantalla:** Para ver el desarrollo y a la vez como apoyo para el debugging dentro del laboratorio, se solicitará que se realice como mínimo el print del mensaje recibido en cada entidad del proceso.
- Se utilizarán las 4 máquinas virtuales entregadas a cada equipo del laboratorio anterior.
- El código en cada máquina virtual deberá estar en una carpeta aparte para ser diferenciada de la entrega de laboratorio anterior.
- Se realizará una ayudantía para poder resolver dudas y explicar la tarea. Será notificado por aula.
- Consultas sobre la tarea se deben realizar en Aula o enviar un correo a **Iñaki** con el asunto **Consulta grupo XX - Lab 5 Distribuidos**
- Las librerías de **Golang** permitidas son las mencionadas durante los laboratorios anteriores (cualquier otra librería externa deberá ser consultada con los ayudantes por medio de aula).
- **Sobre Aula:** Una vez definido el código final para cada una de las entidades será requisito poder subir estos archivos en un .zip junto al README para poder informar sobre el proceso de ejecucion en cada una de las máquinas.
- **Docker:** El uso de Docker para este laboratorio es **obligatorio** por ende, todos los códigos deberán ejecutarse con esta app, estando cada entidad en un contenedor (Ingenieros, broker, servidores fulcrum y Comandante). **El no uso de Docker significará que la tarea no será revisada.**

7 Reglas de Entrega:

- La fecha de entrega es el día **28 de Junio a las 23:59hrs.**
- La tarea se revisará en las máquinas virtuales, por lo que, los archivos necesarios para la correcta ejecución de esta, deben estar en ellas, además de la copia mencionada en la entrega de Aula. Recuerde que el código debe estar identado, comentado, sin Warnings ni errores.
- Se aplicará un descuento de **5 puntos** al total de la nota por cada Warning, Error o Problema de Ejecución.
- Opcionalmente puede dejar un **MAKEFILE** en cada máquina virtual para asegurar el funcionamiento de los comandos deseados, indicando en el README con cual comando iniciar los códigos
- Debe dejar un **README** en la entrega de aula, que indique las instrucciones de ejecución y lo asumido para el desarrollo del laboratorio.
- No se aceptan entregas que no se encuentren dentro de aula, tampoco de las máquinas y que no puedan ser ejecutadas desde una consola de comandos. Incumplimiento de estas reglas, significa **nota 0**.
- No se revisará de manera local los archivos, solo los almacenados en las máquinas virtuales.
- Cada hora o fracción de atraso se penalizará con un descuento de **5 puntos**.
- Copias serán evaluadas con **nota 0** y serán notificadas al profesor y autoridades pertinentes.

8 Consultas:

Para hacer las consultas, recomendamos hacerlas por medio del foro del ramo en Aula. De esta forma los demás grupos pueden beneficiarse en base a la pregunta. **Se responderán consultas hasta 48 hrs. antes de la fecha y hora de entrega.**