Laboratorio 3: Sistemas Distribuidos

Profesor: Jorge Díaz
Ayudantes de Lab: Iñaki Oyarzun M. & Javiera Cárdenas

Noviembre 2023

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.

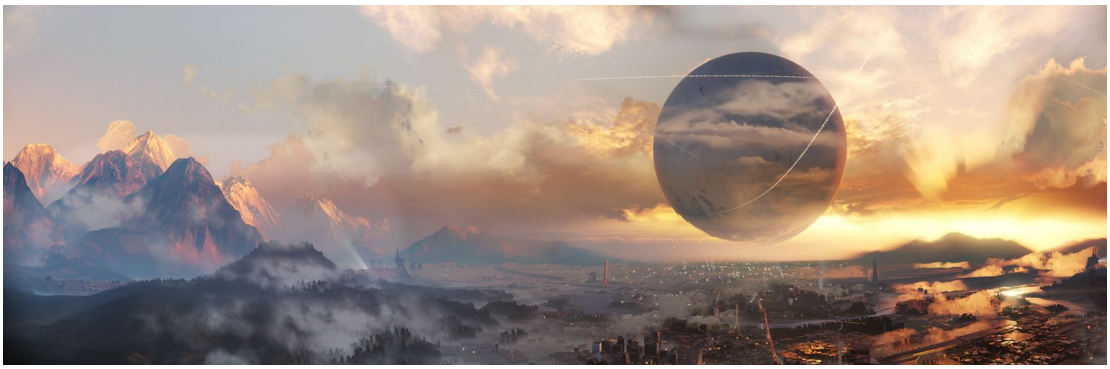
Para poner esto en práctica, se propone el siguiente problema en donde haciendo uso de gRPC y golang para llevar a la práctica la replicación, en donde también se tendrán que aplicar modelos de consistencia centrados en los datos y el cliente. Para este laboratorio, se hara uso del modelo de consistencia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



Lo llamamos, 'El Viajero', y su llegada nos cambió para siempre. Construimos grandes ciudades en Marte y en Venus. Mercurio se convirtió en un vergel, la esperanza de vida se triplicó. Fue una época llena de milagros. Observábamos la galaxia y sabíamos que nuestro destino era caminar bajo la luz y otras estrellas. Pero el Viajero tenía un enemigo, una Oscuridad que llevaba eones persiguiéndolo a través de los sombríos golfos del espacio, siglos después del inicio de nuestra edad de oro. Esta oscuridad nos encontró. Y fue el final de todo...

Pero también fue, un comienzo. Los guardianes, hijos de la luz y férreos protectores de la vida nacen de la mano del Viajero para proteger los restos de la humanidad ante las amenazas, en conjunto con sus aliados de otras especies que han sufrido por el mismo destino de la Oscuridad.

En estos momentos de guerra, la Vanguardia ha mantenido constantes guerrillas contra las fuerzas de la Oscuridad en Neptuno, donde se ha encontrado recientemente una ciudadela humana conformada por aquellos que lograron escapar del desastre de la edad de Oro.

Debido a la distancia, se han visto dificultadas las formas de comunicación de forma consistente, es por ello que, su equipo como famosos desarrolladores de tecnología post edad de oro, son llamados a coordinar estas comunicaciones entre los guardianes dentro del campo y la mesa de mando de la vanguardia para recibir informaciones acerca de supervivientes en cada una de los sectores de Neptuno.

El éxito de esta campaña determinará un gran paso en la obtención de terreno importante para prevenir futuras invasiones de la Oscuridad, esta oportunidad no debe desaprovecharse, buena suerte...

4.2 Explicación

4.2.1 Arquitectura del sistema

El sistema distribuido para este laboratorio se compone de 5 entidades por un lado la emperatriz Caiatl de la raza Cabal, Comandante Osiris, La mesa de comando de la Vanguardia, el Broker Luna y el conjunto de servidores fulcrum.

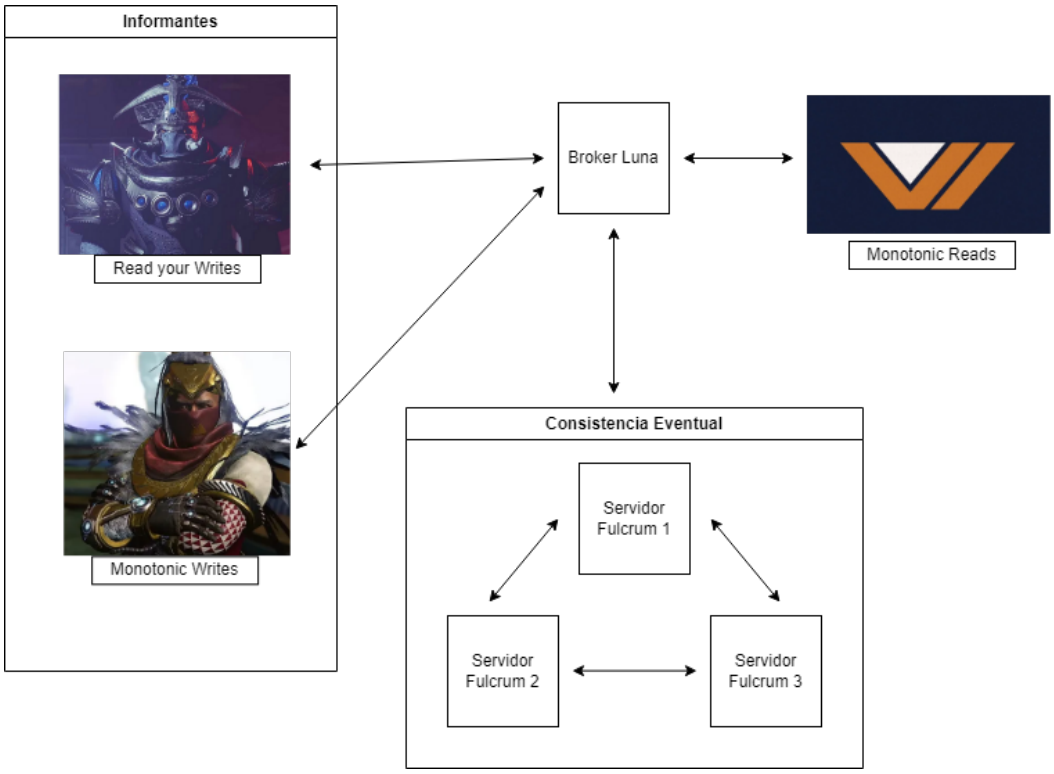


Figure 1: Arquitectura del sistema distribuido

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 Neptuno, los cuales se guardan en el siguiente formato:

  `nombre_sector nombre_base cantidad_de_soldados_oscuridad`
  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 soldados 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 **1 minuto**.

- 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 - Luna

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

- Redirige a los Informantes a uno de los servidores Fulcrum de forma **aleatoria**. En el caso de La Vanguardia, toma el rol de intermediario entre los servidores Fulcrum y los Informantes, por lo que este Broker se encargará de transmitir los mensajes enviados entre ambos componentes.
- Redirige a los Informantes y a La Vanguardia a una réplica en específico cuando estos presenten un conflicto entre las versiones de los registros de sectores.

4.5 Los Informantes

Los informantes son 2: La Caiatl y el comandante Osiris. Son los encargados de agregar información a los registros de soldados de los servidores Fulcrum, además de actualizar o eliminar dichos registros. Para lo anterior dispone 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 soldados, 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 Luna de manera automática. La respuesta del servidor Fulcrum es el reloj de vector del registro de sector del sector modificado

Uno de los informantes (Caiatl) utilizará el modelo de consistencia **Read your Writes** mientras que el otro (Osiris) utilizará **Monotonic Writes**. Para poder llevar a cabo la consistencia, los informantes deben 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.

4.6 La Vanguardia

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

- **GetSoldados** `<nombre_sector> <nombre_base>`**:** Este mensaje se envía al Broker Luna y recibe por respuesta la cantidad de soldados de la oscuridad que se encuentran en dicha base del sector consultado, junto al reloj vectorial del registro del servidor Fulcrum al que Luna consultó.

La Vanguardia tiene que hacer uso del modelo de consistencia **Monotonic Reads**, Para poder llevar a cabo esta consistencia, La Vanguardia deberá 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.

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).

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 Luna debe estar en un servidor diferente a los servidores Fulcrum
- La Vanguardia, Informante 1, Informante 2 y Broker Luna 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.
- 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 **Javiera** o **Iñaki** con el asunto **Consulta grupo XX - Lab 3**
- Las librerías de **Golang** permitidas son las mencionadas durante los dos laboratorios anteriores (cualquier otra librería externa deberá ser consultada con los ayudantes por medio de aula).
- **Sobre Github:** Cada grupo tendrá acceso a un repositorio privado (se ha creado uno nuevo dentro de la organización a la que fueron agregados) el cual le permitirá actualizar y entregar el código de su laboratorio. Para ello, el día de la entrega, deberá dejar en la rama principal (main) todos sus archivos finales.
- **Docker:** El uso de Docker para este laboratorio es **obligatorio** por ende, todos los códigos deberán ejetutarse con esta app. **El no uso de Docker significará que la tarea no será revisada.**

7 Reglas de Entrega:

- El laboratorio se debe presentar en el Laboratorio de Redes con una fecha a definir la cual será informada por aula.
- La fecha de entrega es el día **20 de Noviembre 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. 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.
- Debe dejar un **MAKEFILE** o similar en cada máquina virtual asignada a su grupo para la ejecución de cada entidad. Este debe manejarse de la siguiente forma:
  - `make docker-broker`: Iniciará el código hecho en Docker para el Broker Luna
  - `make docker-vanguardia`: Iniciará el código hecho en Docker de La Vanguardia
  - `make docker-f1`: Iniciará el código hecho en Docker para el servidor Fulcrum 1
  - `make docker-f2`: Iniciará el código hecho en Docker para el servidor Fulcrum 2
  - `make docker-f3`: Iniciará el código hecho en Docker para el servidor Fulcrum 3
  - `make docker-i1`: Iniciará el código hecho en Docker para el servidor de Informante 1
  - `make docker-i2`: Iniciará el código hecho en Docker para el servidor de Informante 2
- Debe dejar un **README** en el repositorio, que indique las instrucciones de ejecución y lo asumido para el desarrollo del laboratorio.
- No se aceptan entregas que no se encuentren dentro del repositorio, 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.**