Informe Tarea 2

ALPA-K

Qué se hizo y cómo se hizo

Utilizando Golang y gRPC se crearon y desarrollaron las entidades declaradas en el enunciado:
Usuario Uploader, Usuario Downloader, NameNode, DataNode A, DataNode B, DataNode C

La secuencia de acciones esperadas es la siguiente:

1. El **Cliente Uploader** sube un libro en PDF, eligiendo al comienzo de su ejecución si el sistema utilizará un algoritmo centralizado o descentralizado. Luego, divide el libro en chunks de máximo *250 kB* y los envía a un Datanode aleatorio entre los disponibles.
   Procedimiento: Se crea un buffer, se escribe una parte (chunk) del libro en el buffer, luego se envía el contenido por gRPC al **Datanode** principal. Este, al recibir los bytes por grpc se escriben en otro buffer, y por último dicho buffer se escribe en un archivo sin extensión.

2. El **Datanode** elegido (llámese *Datanode Alpha*) genera una propuesta de distribución de los chunks entre todos los **Datanodes** (incluyéndose a sí mismo), a menos que el libro pese muy poco y tenga menos de 3 chunks, en cuyo caso considerará tan solo uno o dos **Datanodes**, dependiendo de la cantidad de chunks.

3. El **Datanode Alpha** verifica según el tipo de algoritmo, que la propuesta sea aprobada por los nodos participantes:
   En el caso del algoritmo *descentralizado*, los **Datanodes** a los que se envía la propuesta verifican si hay un **Datanode** caído, siempre que esté considerado en la propuesta. En caso de estarlo esta se rechaza y el **Datanode Alpha** crea otra propuesta considerando solo los **Datanodes** activos (bajo el supuesto que a lo más se caerá 1 **Datanode**), la que es nuevamente revisada, pero solo por el nodo no caído, y así sucesivamente hasta ser aprobada.
   En el caso del algoritmo *centralizado*, se le envía la propuesta al **Namenode** para que éste la analice y, en caso de detectar un **Datanode** caído, que a la vez está considerado en la propuesta, el **Namenode** genera una nueva propuesta.

4. Una vez aprobada, en el caso *centralizado*, el **Namenode** escribe en su *log* directamente la propuesta de distribución de chunks, puesto que ya cuenta con la estructura requerida en el archivo (la propuesta es un solo *string*).
   Mientras que en el caso *descentralizado*, el **NameNode Alpha**, envía la propuesta aceptada al **NameNode** para que la escriba en el *log*.

5. En caso de que haya más de nodo intentando escribir en el log de namenode, se maneja el conflicto con exclusión mutua según el tipo de algoritmo:

Exclusión Mutua Centralizada
Se basa en el uso de una cola para hacer esperar a los procesos que quieran escribir en el log del NameNode, en caso de haber otro proceso utilizándolo.

Exclusión Mutua Descentralizada
En el caso *descentralizado* se utiliza el algoritmo de *Ricart y Agrawala,* en el cual, si hay un proceso de escritura en curso, el estado del **Datanode** responsable pasa a *held*, si hay otro proceso concurrente que desea escribir debe esperar (su estado pasa a *wanted*), los **Datanodes alpha** concurrentes esperan la respuesta de todos los otros **Datanodes** interesados en escribir, guiándose por sus timestamps o tiempos de llegada y/o por sus estados actuales. Cuando un **Datanode alpha** recibe la notificación que se terminó de escribir en el log, su estado pasa a *released*, y de ese modo los siguientes **Datanodes alpha** de la cola pueden enviar la propuesta aprobada para que el **Namenode** la escriba en el log.

6. El **Datanode alpha**, entonces, distribuye y envía los chunks a los Datanodes comprometidos en la propuesta aprobada.

7. Los **Datanodes** reciben los chunks y los guardan en disco sin extensión.

8. El **Cliente Downloader** puede elegir por consola si desea ver una lista de los libros disponibles o si desea descargar un libro, en este último caso el cliente debe ingresar el nombre del archivo sin extensión.

9. El **Cliente Downloader** envía el nombre del libro al Namenode para que éste busque en el log la distribución de los chunks del archivo, y le devuelve esa información al Cliente Downloader.

10. El **Cliente Downloader** una vez que recibió la ubicación de los chunks, solicita a cada Datanode correspondiente los chunks necesarios para unificar el libro en un archivo PDF.

Resultados

Al ejecutar el Namenode y los Datanodes A, B y C, estos inicialmente implementan *Listen* y esperan conexiones entrantes por gRPC. Por su parte el Cliente Uploader pregunta si se quiere trabajar con el algoritmo centralizado o descentralizado, posteriormente pide el nombre del libro pdf como está en la descripción. Cliente Downloader despliega un menú con la opción de ver un listado de libros disponibles y otra opción de descargar un libro, donde se debe ingresar el nombre de este sin extensión.

A continuación se muestran los resultados de un ejemplo con un mismo libro implementando ambos algoritmos.

1) Ejecución Descentralizada

```
2020/12/02 21:33:25 listo
2020/12/02 21:33:25 Cantidad de mensajes: 3
2020/12/02 21:33:25 Conexión a DataNode realizada
2020/12/02 21:33:25 Distribuyendo chunk a otro DataNode

2020/12/02 21:33:25 tiempo Total : 139.943065ms
```

Datanode principal imprime en pantalla el tiempo empleado en la ejecución completa (sin contar la subida o bajada de chunks de los clientes).

Tiempo de ejecución datanode principal: **139,94 milisegundos.**
Tiempo Escritura Log NameNode: **242,993 microsegundos**
Cantidad de mensajes: **3**

2) Ejecución Centralizada

```
2020/12/02 21:35:06 Conexión a DataNode realizada

2020/12/02 21:35:06 Distribuyendo chunk a otro DataNode

2020/12/02 21:35:06 Conexión a DataNode realizada

2020/12/02 21:35:06 Distribuyendo chunk a otro DataNode

2020/12/02 21:35:06 tiempo Total : 97.304627ms
```

Tiempo de ejecución datanode principal: **97,3 milisegundos.**
Tiempo Escritura Log NameNode: **164,27 microsegundos**
Cantidad de mensajes: **1**

3) Descarga de Libro

```
2020/12/02 21:40:20 Obteniendo ubicaciones de chunks del libro Alicia_en_el_pais_
de_las_maravillas-Carroll_Lewis
2020/12/02 21:40:20 Libro encontrado!
2020/12/02 21:40:20 Tiempo de búsqueda : 508.897µs
```

Tiempo de búsqueda del libro en el log: **508,9 microsegundos.**

Acá se encuentran más Capturas de la Ejecución

Análisis

En este ejemplo particular, se observa que tanto la ejecución del datanode principal como la escritura en el log de Namenode, se demoran menos tiempo con el algoritmo centralizado que con el distribuido. Esto se debe esencialmente a que en el caso de algoritmo centralizado el namenode es quién acepta o rechaza la propuesta (y genera nuevas propuestas él mismo en caso de ser rechazada la primera) y la escribe inmediatamente una vez es aprobada. Mientras que en el caso descentralizado, son los otros Datanodes participantes en la propuesta los que deben aceptarla o rechazarla y adicionalmente si es rechazada, el datanode principal debe generar una nueva propuesta y volver a enviarla a los Datanodes, por lo tanto recién una vez que está aprobada la propuesta por los Datanodes involucrados, el namenode puede escribir en el log. Por esta razón también, se observan 3 mensajes en lugar de 1 como el centralizado.

Discusión

En comparación a cómo se esperaba que funcionara el sistema, y analizando los resultados obtenidos se pueden hacer los siguientes comentarios:

Dado que el programa está hecho para las características específicas de la tarea (donde hay 3 Datanodes), en el caso centralizado no se considera que haya más de un Datanode caído, por lo tanto el Namenode genera una nueva propuesta con los dos Datanodes restantes. Esto siguiendo la lógica de que el propósito del programa es distribuir libros en chunks a varias locaciones, si hubieran 2 Datanodes caídos, sólo quedaría el Alpha quien ya contiene todos los chunks y no tiene sentido enviarselos a sí mismo.

Además, al ser específico para 3 Datanodes como se mencionó, no es un programa tan generalizable como podría serlo en un caso ideal, en que se pudiera considerar una cantidad numerosa de Datanodes.

Dado que el sistema está pensado para casos pequeños, no se consideró que más de un proceso de escritura en el log tuvieran la misma timestamp y por tanto el desempate por ID en dicho caso no fue implementado.

Conclusiones

En este laboratorio se desarrolló un sistema que permite distribuir archivos pdf en chunks de bytes utilizando Golang y gRPC, aplicando además algoritmos de exclusión mutua centralizada y distribuida, para luego implementar el sistema en máquinas virtuales. Durante el proceso se estudió la lógica de los algoritmos de exclusión mutua y cómo desarrollarlos acorde a las necesidades del proyecto. Se obtuvieron resultados favorables dentro de los supuestos y condiciones consideradas. De esta manera se pudo aterrizar y concretar los contenidos de la asignatura en cuanto a coordinación y exclusión mutua.