Sistemas Distribuidos

Laboratorio 2: Exclusión Mutua Distribuida

Harold Caballero, 201773602-k
Nicolás Castillo, 201773561-9

En este laboratorio se pide implementar una distribución de recursos para una red de bibliotecas y sus usuarios, ya que debido a las restricciones necesarias estos últimos no se han podido acercar a las dependencias. Para esto los libros serán divididos en chunks y estos repartidos en distintos nodos, se tienen tres tipos de nodos: Cliente, DataNode y NameNode. A grandes rasgos Cliente tendrá funciones de cargar y descargar archivos del sistema, DataNode son los encargados de almacenar los chunks y NameNode registra los metadatos de las partes de cada libro en un archivo log, además de mostrar el listado de libros disponibles al cliente. En ningún caso, más de un nodo debe acceder al registro log, por lo que se ha pedido que se implementen dos algoritmos de exclusión mutua: El algoritmo centralizado y el algoritmo distribuido. Para el laboratorio se debe utilizar el lenguaje de implementación GO[1] y para la comunicación se usará gRPC[2]

A continuación se muestran los procedimientos suponiendo que todos los sistemas están arriba y funcionando..

Cliente

Tiene dos tipos de clientes. Cliente Uploader se encarga de cargar los libros en la red, para ello los divide en chunks los cuales son enviados a un DataNode en particular. Cliente Downloader: Se encarga de recuperar los libros de la red, se conecta al NameNode para obtener la ubicación de los chunks, los cuales descarga de los DataNodes que le son indicados y reconstruye el libro a partir de los chunks.

Para la implementación de cliente se comenzó probando dividir un libro y volver a juntarlo[3], también se realizaron pruebas de comunicación por medio de gRPC gracias al tutorial que se dejó anteriormente. Luego se creó un menú en donde se elige la funcionalidad a utilizar, uploader o downloader. Para el modo uploader se hace una solicitud por pantalla del archivo a subir, el cual se divide en chunks gracias a la prueba realizada anteriormente y se envían los datos necesarios por medio de comunicación gRPC a un DataNode al azar, en esta oportunidad se consideró que estos datos eran: titulo del libro, cantidad total de chunks, número actual del chunk que se está enviando y los bytes correspondientes al chunk. Para el modo downloader se comienza pidiendo el listado de los libros disponibles por medio de gRPC al NameNode, luego se solicita por pantalla el libro requerido y se

[1] https://www.javatpoint.com/go-tutorial
[2] https://tutorialedge.net/golang/go-grpc-beginners-tutorial/
[3] https://www.socketloop.com/tutorials/golang-recombine-chunked-files-example

pide nuevamente al NameNode las ubicaciones de los chunks correspondientes a ese archivo. A partir de la respuesta obtenida se comienza a solicitar a los DataNode los chunks correspondientes al libro, para esto se les envía el título del libro y el número del chunk solicitado. A medida que se van recibiendo los chunk se van escribiendo en un nuevo archivo el cual se llamó “DescargaNOMBRELIBRO”.

NameNode (Exclusión Mutua Distribuida)
Será el encargado de almacenar el LOG dentro del cual se registran los metadatos de las partes de cada archivo, además de mostrar el listado de libros disponibles al cliente.

Se comenzó probando la escritura y lectura del archivo log el cual es de tipo .txt, en donde para cada libro se sigue el siguiente formato:

NombreLibro CantidadPartes
NombreLibro_1 ipmaquina
NombreLibro_2 ipmaquina
...
NombreLibro_CantidadPartes ipmaquina

Posteriormente se establecieron las funciones para la comunicación con Cliente, para lo cual se consideró solo una función. Si el mensaje de Cliente corresponde a “” (vacío) se está solicitando el listado de libros, el cual se obtiene de leer el leer una estructura auxiliar dentro del nodo que almacena la misma información que el log, pero de una forma más rápida de acceder, respondiendo un string con los nombres separados por espacios. Mientras que si el mensaje enviado corresponde a “Nombrelibro” se buscará en la estructura antes mencionada las direcciones, al haber coincidencia se responde con la distribución correspondiente, ip de las máquinas separados por espacios y en orden de acuerdo al chunk almacenado. Por otro lado, al recibir un mensaje de DataNode se procede a escribir en el LOG, el mensaje ya viene con el formato correspondiente.

También para implementar Ricart-Agrawala se tiene una variable global que permite saber si se está escribiendo o no en el log.

DataNode (Exclusión Mutua Distribuida)
Son los encargados de almacenar los chunks que fueron generados por el cliente uploader. En primera instancia, uno de los DataNodes se encarga de distribuir los fragmentos entre sus pares, sin embargo, previo a eso debe producir una propuesta de cómo se repartirán las partes. Esta propuesta debe ser aprobada por los otros DataNodes. En cada DataNode existe una variable global que permite saber si se quiere escribir en el log.

Cada vez que se desee escribir en el log, se cambia la variable global antes mencionada y se procede a consultar a los otros nodos si se puede escribir en el log. Junto con la petición se manda el id del nodo que quiere escribir,cuando todos los nodos responden que sí se procede a consultar con el namenode si se puede escribir. Cuando un nodo recibe un mensaje de permiso, este da respuesta positiva cuando no esté intentando escribir en el log o en caso de que esté intentado, si la id del que realiza la petición es menor, también se da respuesta afirmativa.

Discusión

En esta ocasión no se pudo desarrollar el método centralizado, sin embargo se pudo observar que con el método distribuido implementado la escritura en el log es segura ya que sólo se puede escribir cuando nadie más quiere o puede escribir.
Debido a que es un sistema pequeño no se hallaron problemas de tiempos, saturación ni nada por el estilo, todas las transferencias eran casi instantáneas. Sin embargo, prevemos que probablemente para escalar el sistema sea mejor implementar una cola para las escrituras en el log.

Conclusión

Con el desarrollo de este laboratorio se nos permitió comprender el funcionamiento de un sistema distribuido que considera un algoritmo de exclusión mutua distribuida, para esto se tuvieron en cuenta diferentes tipos de nodos que cumplen un rol dentro de este sistema. Además como se dijo anteriormente no existieron problemas de tiempo al ser un sistema pequeño.