Informe Laboratorio 2
Sistemas Distribuidos INF-343

Fabian Fernández Alfaro 201573028-8
Diego Albornoz Ley 201223536-7

¿Qué hicieron y cómo lo hicieron?

Implementamos un menú de opciones por consola en el cual en primera instancia se debe elegir usando 1 o 2 para las opciones de subir libro (client uploader) y descargar libro (client downloader).

Client uploader: se pide al usuario que ingrese el nombre del archivo del libro incluyendo su extensión. Estos libros son buscados en la carpeta “libros-subidos”. Luego de ingresar el nombre del archivo se le pedirá al usuario indicar el tipo de sistema de exclusión mutua aplicable a la subida del archivo. El archivo se lee con la función “crearChunks()” y luego se los chunks son enviados a un dataNode elegido de forma aleatoria que esté disponible (“vivo”)..

Client downloader: en el caso de que el cliente escoja la opción 2 del menú principal, se le mostrará una lista de los libros disponibles obtenidos con la función “MostrarLibrosDisponibles()” la cual se comunica con el NameNode para obtener todos los libros que se encuentran en el Log.csv. Luego se le pide al cliente que ingrese el nombre del libro que quiere descargar realizando nuevamente una conexión con el NameNode para conocer la ubicación de los chunks. Una vez conocida las ubicaciones se invoca la función localizarChunks() y por último reconstruirArchivo(), el archivo quedará guardado en la carpeta “libros-descargados” de la máquina virtual desde la cual el cliente está haciendo la solicitud.

DataNode: estos reciben la consulta del cliente y la procesan, en caso de ser el client uploader la función grpc GetLibroSubido() es la encargada de estructurar la información recibida y enviarla a la función respectiva según el tipo de algoritmo que se haya elegido. En cualquier caso, ambos algoritmos generan una propuesta la cual tiene que ser aceptada por los demás DataNodes “vivos” en el caso del algoritmo distribuido y por el NameNode en el caso de un algoritmo centralizado. En caso de que un solo DataNode esté “vivo” este puede aceptar la propuesta. Además, un nodo se considera “vivo” cuando responde la función grpc en caso de que haya un error o se cumpla el timeout se considera como “caído”, este mismo criterio se aplica para aceptar o no una propuesta. Cada DataNode contiene una carpeta en la cual se guardarán los archivos asociados a los chunks asignados de una propuesta aceptada.

Estructura

Cada máquina virtual posee un rol determinado, ya sea ser un DataNode o un NameNode. En este caso la asignación es la siguiente:

- DataNode
  - 10.10.28.147:8080
  - 10.10.28.148:8080
  - 10.10.28.149:8080
- NameNode
  - 10.10.28.15:8083
- Cliente
  - En todas las máquinas virtuales

Implementación algoritmos

- **Distribuido**: Para el algoritmo, se utilizó un reloj de Lamport que aumenta en 2 ticks cada 2 segundos (igual para todos los DataNodes), además se utilizó el algoritmo Ricart Agrawala para la toma de decisiones. Para esta implementación en particular primero se hace un Multicast para conocer qué nodos se encuentran “vivos” y que pertenecen a la propuesta generada, donde la propuesta generado utiliza todos los DataNodes, en caso de que uno de los nodos este “caido” se genera una nueva propuesta con los nodos que se encuentran “vivos”. Luego se ejecuta el algoritmo Ricart Agrawala como se vio en clases, utilizando relojes lógicos (Lamport Clock) y el manejo de estados. La sección crítica ,que aparece en el algoritmo de Ricart Agrawala, para esta implementación es acceder al NameNode y escribir en el Log.
- **Centralizado**: Luego de que el DataNode reciba los chunks, este generará una propuesta utilizando los 3 DataNodes considerándolos como “vivos” y se la enviará al NameNode. En el NameNode se comprobarán cuales de los nodos están realmente “vivos” y se generará una lista con los nodos realmente disponibles. En caso de que todos los nodos estén “vivos” la propuesta será considerada aceptada y en caso contrario se generará una nueva propuesta con los nodos que estaban disponibles y se retornará al DataNode la propuesta que finalmente haya sido aprobada. Para el caso de resolver conflictos para editar el log el NameNode lo resuelve con una cola en donde se rellena con las propuestas que fueron aprobadas y usamos mutex para asegurarnos de que solo se intentara escribir un solo elemento de la cola a la vez.
- **Propuestas**: Las propuestas son generadas usando los nodos disponibles y asignan de manera aleatoria los chunks asociados a cada DataNode “vivo”.

Comparación

Tiempo que demora en registrar Log

Este tiempo es medido desde que el usuario ingresa la opción del tipo de algoritmo hasta que el cliente recibe un mensaje resumiendo el registro. Este tiempo incluye : transformarlo

a chunks, enviar el libro al DataNode,aceptar la propuesta, registrar en el Log y retornar las respuestas correspondientes de cada función grpc hasta llegar al Cliente.

Se calcula el tiempo promedio, se envio 3 veces cada libro ejecutando cada algoritmo asumiendo que todos los DataNodes se encontraban “vivos”, además el cliente se ejecutó en la maquina virtual 10.10.28.149

- Drácula de Stoker Bram, 1.7 MB
  - Centralizado: 278,463 ms
  - Distribuido: 215,332 ms
- Ivanhoe Walter Scott, 1.39 MB
  - Centralizado: 189,042 ms
  - Distribuido: 176,607 ms

Cantidad de mensajes intercambiados

Para esta métrica, se considera como 2 mensajes distintos, la ejecución de la función grpc y la respuesta de esta.

Cliente Uploader

Nodos vivos\Tipo de algoritmo	Centralizado	Distribuido
1 Datanode	N+1 mensajes para subir los chunks al DataNode (N chunks más el mensaje de vuelta), 3+1 para identificar los nodos disponibles (1 mensaje de ida para comprobar el nodo y 1 de respuesta para el nodo con vida), 2 para enviar la propuesta al NameNode, 2*N chunks para enviarlos a cada DataNode “vivo”.	N+1 mensajes para subir los N chunks al DataNode (N chunks más el mensaje de vuelta). 2 para identificar nodos disponibles de la propuesta,2*N chunks para enviarlos a cada DataNode “vivo”. 2+2 en algoritmo Ricart Agrawala (2 para ver estado del resto de los DataNodes + respuesta y 2 para entrar a la sección Crítica)
2 Datanode	N+1 mensajes para subir los chunks al DataNode (N chunks más el mensaje de vuelta), 3+2 para identificar los nodos disponibles (1 mensaje de ida para comprobar el nodo y 1 de respuesta para el nodo con vida), 2 para enviar la propuesta al NameNode, 2*N chunks para enviarlos a cada DataNode “vivo”.	N+1 mensajes para subir los N chunks al DataNode (N chunks más el mensaje de vuelta).4 para identificar nodos disponibles de la propuesta,2*N chunks para enviarlos a cada DataNode “vivo”. 4+2 en algoritmo Ricart Agrawala (4 para ver estado del resto de los DataNodes +

		respuesta y 2 para entrar a la sección Crítica)
3 Datanode	N+1 mensajes para subir los chunks al DataNode(N chunks más el mensaje de vuelta), 3+3 para identificar los nodos disponibles (1 mensaje de ida para comprobar el nodo y 1 de respuesta para el nodo con vida), 2 para enviar la propuesta al NameNode, 2*N chunks para enviarlos a cada DataNode “vivo”.	N+1 mensajes para subir los N chunks al DataNode (N chunks más el mensaje de vuelta). 6 para identificar nodos disponibles de la propuesta,2*N chunks para enviarlos a cada DataNode “vivo”.6+2 en algoritmo Ricart Agrawala (6 para ver estado del resto de los DataNodes + respuesta y 2 para entrar a la sección Crítica)

Análisis y Discusión

Dada la cantidad de mensajes intercambiados utilizando ambos algoritmos, se aprecia que el algoritmo DIstribuido intercambia más mensajes (en situaciones ideales) dado que debe tanto verificar los nodos de la propuesta como también verificar los estados de los demás nodos para poder acceder a la sección crítica. En cambio, el centralizado solo envia 2 *X siendo X la cantidad de DataNodes vivos para que le acepten la propuesta y en el mejor de los casos esta propuesta es aceptada de inmediato por lo que intercambia menos mensajes. Si bien, estos resultados teóricos pueden dar a entender que el algoritmo Centralizado es más eficiente en cuanto a cantidad de mensajes intercambiados que el Distribuido esto puede deberse a la implementación de cada algoritmo más que al algoritmo en sí.
Otra métrica a considerar es el tiempo que demora en realizar el Log en el NameNode que en ambos casos el algoritmo Distribuido tuvo mejores tiempos promedios para los libros probados lo que pudo haber afectado es el hecho de que el algoritmo centralizado utiliza una cola para escribir en el log por lo que aumenta el retraso un poco. Otros factores pueden ser: distancia entre los nodos, la cantidad de nodos del sistema, el tamaño de los archivos, entre otros.

Conclusiones

Dado el análisis realizado previamente y los resultados obtenidos, no se puede concluir que un algoritmo sea mejor que otro, ya que esto depende de la implementación.SIn embargo, para este caso en particular el algoritmo Distribuido presentó mejores tiempos de respuesta lo que se puede considerar como una ventaja importante a la hora de elegir un sistema. Además no posee “cuellos de botellas” que puedan perjudicar su rendimiento, a pesar de eso, la elección entre los algoritmos va a depender de las necesidades de cada persona.