Implementación y análisis de algoritmos de exclusión mutua

- **Introducción**

Los algoritmos de exclusión mutua son algoritmos que permiten el acceso a diferentes secciones críticas de un sistema distribuido.

A lo largo de los años el acceso a secciones compartidas o críticas, han traído consigo bastantes problemas y se ha vuelto un tema importante de investigación en el área de informática, trayendo consigo dos algoritmos bastantes robustos.

El primer algoritmo se denomina Algoritmo de Exclusión Mutua Centralizada, el cual enfoca 1 nodo o punto central, que se encarga de coordinar todos los procesos dentro del sistema a trabajar, por otro lado se encuentra el Algoritmo de Exclusión Mutua Distribuida, el cual trabaja dentro de todos nodos configurados elige uno inicial al azar, el cual se encarga de enviar un mensaje de admisión a todos los otros nodos, los nodos que confirmen el mensaje entregado, serán los participantes del proceso del sistema.

- **Qué se hizo y cómo se hizo**

Se hizo un sistema de biblioteca basado en un sistema de archivos distribuidos, se consideraron 3 tipos de nodos, estos nodos se denominan Cliente, DataNode y NameNode, estos nodos tendrán comportamientos diferentes dependiendo del algoritmo que se implementa.

1. Cliente:
    a. El Cliente es el encargado de particionar los libros en tamaños de 250 [kb], subir y descargar archivos del sistema de la Biblioteca.
2. DataNode:
    a. Los DataNodes, son los que guardan las diferentes partes del libro que fue particionado anteriormente.
    b. Estos tendrán un comportamiento distinto en base al algoritmo a desempeñar.
        i. Distribuida:
            1. En el caso de AEMD, el nodo seleccionado debe levantar una propuesta de envío a los chunks, si los otros nodos aceptan o estan disponibles, se realizará el envío de los chunks, sin embargo, en caso de fallar, es decir que un nodo no responda, el nodo inicial debe crear una nueva propuesta que satisfaga a todos los nodos disponibles.
        ii. Centralizada
            1. En el caso de AEMC, el NameNode que será el nodo central se encargará de verificar la propuesta inicial generada por el dataNode inicial, y dependiendo de la respuesta generará una propuesta nueva o aceptará la propuesta.

3. NameNode:
    a. El NameNode será el encargado de almacenar el LOG, es decir, guardará el nombre del libro, la cantidad de partes asociadas al libro, el nombre de cada una de las partes y la IP del nodo que lo almacena.
    b. La Estructura del NameNode es la siguiente:

```
Nombre_Libro_1 Cantidad_Partes_1
parte_1_1 ip_maquina
...
parte_1_n ip_maquina
Nombre_Libro_2 Cantidad_Partes_2
parte_2_1 ip_maquina
...
parte_2_n ip_maquina
...
```

Figura 1:Estructura del Log que se guardara en el NameNode

Para iniciar el Lenguaje utilizado para la investigación fue Golang y el sistema de comunicación fue gRPC con protocol buffer.

Continuando se inició indicando a cada máquina virtual su rol a desempeñar dentro del sistema, luego se coordinó los mensajes de forma centralizada, para luego comenzar con la coordinación del sistema de forma distribuida.

Luego se realizaron las separación de archivos en chunks, para luego comenzar con la distribución de estos a cada 1 de las máquinas virtuales.

Después de estabilizar la separación de archivos, se comenzó con la recombinación de los chunks separados, solo de forma local, una vez logrado, siguió el avance del NameNode para lograr la escritura del Log y conseguir las direcciones de los chunks distribuidos.

Luego se realizó la recombinación a nivel de sistema, dejando de lado el formato local, para que finalmente, se lograra estabilizar la recombinación por medio de máquinas virtuales.

Finalmente una vez conseguido lo anterior, se comenzó el trabajo con los algoritmos de comunicación planteados verificando los mensajes enviados y recibidos, y una vez conseguida la estabilización se realizaron pruebas generales para comenzar con la toma de datos para la elaboración del presente informe.

- Resultados

Para los resultados se utilizaron 3 libros los cuales fueron:

1. Dracula-Stoker_Bram - 1737 [Kb]
2. Frankenstein-Mary_Shelley - 1000 [Kb]
3. Don_Quijote_de_la_Mancha-Cervantes_Miguel - 2001 [Kb]
4. La_letra_escarlata-Nathaniel_Hawthorne - 1338 [Kb]

Para el caso del Uploader, se tiene lo siguiente:

- 3 Nodos Disponibles:

Los resultados obtenidos por el algoritmo Distribuido y el Centralizado son los siguientes :

Libro	Tamaño [Kb]	Tiempo [ms]
1	1737	175
2	1000	129
3	2001	182
4	1338	151

Libro	Tamaño [Kb]	Tiempo [ms]
1	1737	87
2	1000	63
3	2001	106
4	1338	86

- 2 Nodos Disponibles:

Los resultados obtenidos por el algoritmo Distribuido y el Centralizado son los siguientes :

Libro	Tamaño [Kb]	Tiempo [ms]
1	1737	168
2	1000	127
3	2001	186
4	1338	147

Libro	Tamaño [Kb]	Tiempo [ms]
1	1737	94
2	1000	70
3	2001	101
4	1338	82

- 1 Nodos Disponibles:

Los resultados obtenidos por el algoritmo Distribuido y el Centralizado son los siguientes :

Libro	Tamaño [Kb]	Tiempo [ms]
1	1737	114
2	1000	91
3	2001	115
4	1338	103

Libro	Tamaño [Kb]	Tiempo [ms]
1	1737	88
2	1000	62
3	2001	92
4	1338	83

- Análisis

Como se puede observar en los resultados, hay una relación directamente proporcional entre el tamaño de los archivos y el tiempo que demora el proceso de repartición de chunks, mientras que al mismo tiempo existe un relación inversamente proporcional entre el número de nodos en lo que se reparten los chunks y el tiempo, es decir el proceso tarda más si los chunks se reparten entre más nodos.

Por otro lado, si se comparan en general los tiempos obtenidos entre el algoritmo distribuido y el algoritmo centralizado, se observa que los tiempos son considerablemente menores para el caso del algoritmo centralizado.

- Discusión

Se afirma que los resultados obtenidos son consistentes con la teoría. Como era de esperar, más tamaño implica más chunks, lo que se traduce en una mayor cantidad de mensajes enviados y por lo tanto más tiempo, mientras que un menor número de nodos implica menor cantidad de mensajes y menos tiempo.

Asimismo, no es de extrañar que el algoritmo centralizado demore menos que el algoritmo distribuido. La gran ventaja del algoritmo centralizado es que hay solo un nodo (el nameNode) que se encarga de aprobar y rechazar las propuestas, para luego enviar esa propuesta a un único dataNode que se encarga de ejecutar las instrucciones. En el caso del algoritmo distribuido, las propuestas deben ser aprobadas por todos los dataNodes, y el dataNode que genera la propuesta tendrá que esperar la aprobación del resto de los dataNodes, lo que obviamente implica un mayor traspaso de mensajes y mayor tiempo de ejecución del proceso.

Sin embargo, el algoritmo distribuido posee una ventaja importante en una categoría distinta: la tolerancia a fallos, ya que en el caso del algoritmo distribuido, si el nameNode falla entonces todo el proceso se cae y deja de funcionar, mientras que el algoritmo distribuido puede sobrevivir aún cuando se caiga al nameNode o algún dataNode (aunque en el caso del nameNode no se podría escribir en el log).

- Conclusión

Gracias a este laboratorio, se pudo llevar a la práctica lo aprendido en clases y así comprobar la teoría asociada a los sistemas distribuidos y específicamente, a los algoritmos de exclusión mutua. Se pudo demostrar que los tamaños de los archivos son relevantes para la repartición en distintos nodos, así como también la cantidad de nodos en los que se van a repartir, y que el algoritmo centralizado al utilizar una menor cantidad de mensajes demora menos tiempo que el algoritmo distribuido, pero este último posee una mayor tolerancia a fallos al no tener un punto único de fallo (SPOF). Entonces, la elección de un algoritmo o de otro va a depender de las necesidades propias del sistema distribuido, por lo que se debe realizar un análisis cuidadoso al momento del diseño de la arquitectura.