Tarea 2: Comparación exclusión mutua centralizada y distribuida

Nicolás Pino - 201673534-8
Benjamín Camblor - 201673505-4

Noviembre 2020

Procedimiento

La labor realizada tuvo como objetivo observar y cuantificar el comportamiento de 2 algoritmos distintos de exclusión mutua: centralizada y distribuida. Con este fin se realizaron ciertos experimentos sobre una infraestructura compuesta de 3 nodos de datos, un nodo de nombres, y nodos clientes. Este conjunto de nodos simula el funcionamiento de una biblioteca donde los nodos de datos almacenan libros que los clientes suben a la red, mientras que el nodo de nombres se encarga de mantener un registro con la ubicación de los libros, para así posibilitar su descarga de parte de los clientes. Para la primera fase de experimentación se considera la configuración estándar para los nodos. Se levanta el servidor de namenode en conjunto con los 3 datanodes. Finalmente se ejecuta el cliente para subir los libros a la red y otro cliente para descargalos. A continuación se presenta una tabla que resume la información mencionada:

Alias del host	Dirección	Función	Ruta absoluta
dist105	10.6.40.246:50053	Datanode	/home/tarea2/Datanode/
dist106	10.6.40.247:50053	Datanode	/home/tarea2/Datanode/
dist107	10.6.40.248:50053	Datanode	/home/tarea2/Datanode/
dist108	10.6.40.249:50055	Namenode	/home/tarea2/Namenode/
dist10x	10.6.40.24x	Cliente de subida	/home/tarea2/main.go
dist10x	10.6.40.24x	Cliente de descarga	/home/tarea2/ClientDownloader/

Tabla 1: Configuración inicial - La 'x' en las últimas filas indica que los clientes pueden ser ejecutados desde cualquiera de los hosts anteriores, por lo que x puede tomar los mismos valores que los 4 hosts listados.

Libro	Tamaño	Número de chunks
Los 120 días de Sodoma	1,29[MB]	6
El arte de la guerra	293[KB]	2
Drácula	1,69 [MB]	7

Tabla 2: Información sobre los libros ocupados para las pruebas.

Los pasos para obtener el estado inicial de prueba son los siguientes:

1. Establecer conexión remota via ssh a los host de datanode.
2. Acceder a la ruta absoluta para nodos de datos de la Tabla 1.
3. Ejecutar comando make
4. Establecer conexión remota via ssh al host de namenode.
5. Acceder a la ruta absoluta para el nodo de nombre de la Tabla 1.
6. Ejecutar comando make
7. Establecer conexión remota via ssh a cualquiera de los 4 hosts.

8. Acceder a la ruta absoluta para el cliente de subida de la Tabla 1.
9. Ejecutar comando make
10. Aguardar el proceso de subida de los libros
11. Establecer conexión remota via ssh a cualquiera de los 4 hosts.
12. Acceder a la ruta absoluta para el cliente de descarga de la Tabla 1.
13. Ejecutar comando make
14. Seleccionar el libro deseado via consola

Luego de iniciar los nodos de nombre y datos se ocupa una segunda conexión ssh para ejecutar el cliente de subida en uno de los host listados. El cliente sube 3 libros diferentes, pues cada vez selecciona un datanode distinto como inicial: cada ejemplar es dividido en n chunks los cuales son enviados a uno de los data nodes. Aquí la ejecución difiere dependiendo del algoritmo de exclusión mutua, ya sea distribuido o centralizado. Para el centralizado, este elabora una propuesta para distribuir los chunks y la envía al namenode, el cual revisa que los datanodes propuestos estén operativos, y los reemplaza por nodos disponibles cuando correponda. Luego de esto, el namenode retorna la distribución final al datanode que realizó la propuesta. Para el caso distribuido, luego de que el datanode genera una propuesta, esta se envía a todos los demás namenodes, esperando que se apruebe por todos ellos. De ser rechazada por uno o más nodos, se genera una nueva propuesta hasta que se tenga la aceptación total. A continuación se procede a informar a namenode de la distribucion acordada, donde se utiliza el algoritmo de Ricart Agrawala para coordinar el acceso a la sección crítica, es decir, la comunicación al log de namenode. Finalmente, para ambos algoritmos, el datanode distribuye los chunks según la distribución aprobada. El cliente de descarga puede ser ejecutado en este punto para corroborar la integridad de los libros después de descargarlos.

Para ejecutar la primera iteración del experimento, se accede a cada uno de los hosts remotos para luego llegar a la ruta absoluta de cada nodo de datos y del nodo de nombre. En cada caso se ejecuta el comando make para inicializar el nodo. La segunda iteración del proceso involucra el mismo procedimiento, pero solo se ejecutan 2 nodos de datos. Se realizaron 3 intentos para cada caso, por lo que el tiempo será reportado para cada iteración. La cantidad de mensajes es constante, por lo tanto se incluye una única vez en cada caso.

Resultados

Se realiza la prueba ejecutando un cliente de subida que crea 3 hebras paralelas para subir cada libro en la tabla 2 a un datanode distinto. Los datos a considerar son el tamaño del libro, la cantidad de chunks en los que fue dividido, la cantidad de mensajes que fueron enviados desde la ejecución del algoritmo de exclusión mutua hasta la distribución de los chunks en los datanodes, y la cantidad de tiempo que el envío de dichos mensajes tomó.

Algoritmo centralizado

		Tiempo de ejecución		
Libro	**# mensajes**	**Iteración 1**	**Iteración 2**	**Iteración 3**
120 días de Sodoma	12	66,96 [ms]	54,28 [ms]	63,32 [ms]
El arte de la guerra	5	28,56 [ms]	33,52[ms]	40,05 [ms]
Drácula	14	69,92 [ms]	65,19 [ms]	64,40 [ms]

Tabla 3: Algoritmo centralizado - Número de mensajes y tiempo de ejecución para las 3 iteraciones con 3 datanodes operativos.

		Tiempo de ejecución		
Libro	**# mensajes**	**Iteración 1**	**Iteración 2**	**Iteración 3**
120 días de Sodoma	16	3,22 [s]	3,27 [s]	3,20 [s]
El arte de la guerra	7	835,17 [ms]	830,17[ms]	837,32 [ms]
Drácula	18	2,44 [s]	2,43 [s]	2,44 [s]

Tabla 4: Algoritmo centralizado - Número de mensajes y tiempo de ejecución para las 3 iteraciones con 2 datanodes operativos.

Algoritmo distribuido

		Tiempo de ejecución		
Libro	**# mensajes**	**Iteración 1**	**Iteración 2**	**Iteración 3**
120 días de Sodoma	26	71,06 [ms]	75,53 [ms]	56,14 [ms]
El arte de la guerra	21	39,55 [ms]	50,81[ms]	71,34 [ms]
Drácula	27	64,09 [ms]	62,40 [ms]	64,33 [ms]

Tabla 5: Algoritmo distribuido - Número de mensajes y tiempo de ejecución para las 3 iteraciones con 3 datanodes operativos.

		Tiempo de ejecución		
Libro	**# mensajes**	**Iteración 1**	**Iteración 2**	**Iteración 3**
120 días de Sodoma	34	6,50 [s]	6,47 [s]	6,48 [s]
El arte de la guerra	17	2,44 [s]	2,44 [s]	2,45 [s]
Drácula	29	6,33 [s]	5,650 [s]	5,69 [s]

Tabla 6: Algoritmo distribuido - Número de mensajes y tiempo de ejecución para las 3 iteraciones con 2 datanodes operativos.

Análisis

En las tablas 3 y 4 se observan los resultados obtenidos en los experimentos sobre el algoritmo centralizado. Como se esperaba libros más ligeros disfrutan de tiempos de transferencia más cortos, pues una menor cantidad de chunks implica menos iteraciones para corroborar la propuesta de distribución, lo que reduce la labor del namenode; además se disminuye el tiempo de transferencia hacia el resto de los datanodes. En la tabla 4 se observa como al disminuir a 2 los datanodes, los tiempos de ejecución aumentan. Esto obedece a la naturaleza del algoritmo, pues las propuestas originales de distribución deben ser enmendadas cada vez que se intente contactar al nodo que se encuentra fuera de línea.
En cuanto a la cantidad de mensajes, en cada caso se contemplan 2 mensajes correspondientes a la comunicación entre el primer datanode y el namenode; luego 1 mensaje por cada chunk del archivo, pues el namenode ocupa 1 mensaje para verificar la disponibilidad de cada nodo candidato a almacenar un chunk; finalmente se considera 1 mensaje adicional por cada chunk que es destinado a un datanode distinto al inicial. Al quitar uno de los namenodes la cantidad de mensajes aumenta en un factor de 2, pues el namenode ocupa 1 mensaje para detectar que uno de los nodos propuestos esta indispuesto, y otro mensaje adicional para verificar que el siguiente nodo candidato está disponible. Mientras mayor sea la cantidad de chunks asignados al datanode caído, mayor será el aumento de mensajes adicionales, como se observa en la tabla 4 el archivo de 2 chunks aumentó en 2 mensajes, y ambos libros de mayor tamaño tomaron 4 mensajes adicionales en su proceso.

En la tabla 5 y 6 se observan los datos obtenidos de la ejecución del algoritmo distribuido. Nuevamente se observa un menor tiempo de ejecución y cantidad de mensajes para el arte de la guerra como se esperaría. Lo más notorio de los resultados es al comparar los algoritmos, el distribuido tiene una mayor cantidad de mensajes y mayor tiempo de ejecución, lo cual es consecuente ya que los datanodes deben coordinarse entre sí tanto para aceptar la propuesta de distribuición, como para permitir el acceso al namenode. Nuevamente sucede que el tiempo de ejecución aumenta considerablemente cuando se ejecuta con dos nodos, esto es el comportamiento esperado, ya que la propuesta inicial es rechazada si el nodo caído formaba parte, teniendo que evaluar una segunda distribución.

Discusión

Gran parte de la discrepancia que se observa entre los tiempos de ejecución entre las tablas 3 y 4 se debe al lapso de timeout asignado al namenode. Cuando uno de los datanodes se encuentra fuera de servicio, el namenode espera exactamente 800 milisegundos antes de determinar que el datanode no se encuentra disponible. Para los casos previamente discutidos donde el namenode debe consultar al mismo datanode 2 veces, este proceso se repite 2 veces, agregando más de 1 segundo al tiempo total de ejecución. Por lo anterior, este valor "infla" el tiempo en estos casos, y su modificación se vería directamente reflejada en los resultados reportados. Es posible observar dicho efecto en los tiempos reportados para "El arte de la guerra": teniendo solo 2 chunks, y sabiendo que solo es posible que uno de ellos sea inicialmente asignado al datanode caído, el namenode debe esperar 800 milisegundos para poder proponer a un nuevo candidato, el cual siempre estará disponible dadas las condiciones de la prueba. Al observar las tablas 3 y 4, las diferencias entre los tiempos reportados para el mencionado libro son casi exactamente 800 [ms]. Un segundo factor relevante a los tiempos de ejecución radica en el punto de partida de la medición: el tiempo en la cola de cada Datanode, es decir de cada libro en las tablas 3 y 4, no fue considerado. Dado que la ejecución del cliente de subida fue realizada en 3 hebras creadas secuencialmente en el mismo orden de los libros en la tabla 2, es razonable inferir que el tiempo de espera para el n-ésimo libro es aproximadamente igual al tiempo de ejecución de sus $n-1$ antecesores. De manera similar, el algoritmo distribuido también se ve afectado por el timeout del nodo caído, pero esta vez se ve más marcado, ya que son el resto de los datanodes los que tratan de establecer conexión al nodo faltante, lo cual se aplica tanto para la evaluación de las propuestas como para la coordinación al acceso al namenode, aumentanto considerablemente el tiempo de ejecución. En el caso del arte de la guerra ocurre un caso peculiar donde toma una menor cantidad de mensajes para la ejecución de dos nodos, pero esto se explica dado el pequeño tamaño del libro, la propuesta inicial no alcanza a considerar el nodo caído, teniendo que evaluar una sola propuesta, con una menor cantidad de nodos a coordinar. Como se espera para los otros libros de mayor extensión, esto no ocurre.

Conclusión

En base a lo observado, cabe destacar el impacto que tiene la caída de un nodo en el tiempo de ejecución, esto esta directamente relacionado al tiempo de timeout utilizado, aplicándose cada vez que se intenta conectar, dado que esté efectivamente caído. Otro aspecto que se pudo evidenciar es el impacto por tipo de algoritmo, el distribuido sufre un mayor incremento en su tiempo de ejecución dado que los nodos toman un rol más activo que en el centralizado, formando parte de la aceptación de la propuesta, y coordinación a acceso a secciones críticas, el namenode en este caso. A pesar de lo señalado anteriormente, es importante tener en consideración la escala del experimento, si bien el algoritmo distribuido muestra peor rendimiento en este escenario, en un situación a escala real, el cuello de botella que conlleva el algoritmo centralizado se vuelve relevante, situación que no se ve reflejada en nuestras mediciones.