Universidad Técnica Federico Santa María
Campus Santiago San Joaquín
Departamento de Informática
2° semestre 2020

# Tarea 2
# Transmisión de chunks para biblioteca virtual

Renato Bassi 201773521-K
Joaquin Concha 201773569-4

2 de diciembre de 2020

## 1. Qué se hizo y Cómo se hizo

Para desarrollar la "Tarea 2" se separo entre las maquinas virtuales los cargos de DataNode 1, DataNode 2, DataNode 3 y NameNode. Además se estableció que el DataNode 1 (desde ahora DN1 y misma abreviación para sus pares) tendrá el cliente Uploader y Downloader. Cuando un DN se ejecuta este crea el servidor y el cliente a través de go routines, al momento de ejecutar DN1 este se conectará con los otros servers ubicados en DN para seguir con un menú el cual primeramente solicitará si se desea "subir" o "bajar" un archivo, al seleccionar "subir" solicitara el algoritmo que desea usar "Distribuido" o Çentralizado", para terminar solicitando el nombre del archivo.

Al momento de Subir un archivo y haber elegido "Distribuido", se verificarán los servidores que estén respondiendo (con la función SayHello2) se seleccionará uno al azar, se le enviarán los çhunks"del archivo ya dividido a través de protocol buffers(con la función SayHello), para posteriormente generar una propuesta (que puede incluir los nodos caídos, la función encargada de esto es GenerarPropuesta), que dependiendo de los DN que esta incluya y de los DN que estén respondiendo podrá ser aceptada o no (en este caso por los otros DN, solicitando confirmación con la función PedirConfirmacion2), en caso de no ser aceptada, se generará una hasta que sea aceptada (con la función GenerarPropuesta). Al tener la propuesta aceptada, el DN seleccionado usará la función EscribirPropuestaDis, para decirle al NameNode(NM) para que escriba en el Log.txt (Archivo generado al iniciar el NM) con la propuesta aprobada, se reparten los chunks a sus nodos correspondientes con la función Repartir y agregará a los títulos disponibles con la función AgregarTitulo

Al momento de Subir un archivo y haber elegido ¨ Centralizado", se verificarán los servidores que estén respondiendo (con la función SayHello2) se seleccionará uno al azar, se le enviarán los çhunks"del archivo ya dividido a través de protocol buffers (con la función SayHello), para posteriormente generar una propuesta (que puede incluir los nodos caídos, la función encargada de esto es GenerarPropuesta), que dependiendo de los DN que esta incluya y de los DN que estén respondiendo podrá ser aceptada o no (en este caso por los otros NM, solicitando confirmación con la función PedirConfirmacionNM), en caso de no ser aceptada, se generará una hasta que sea aceptada (con la función GenerarPropuesta2). Al tener la propuesta aceptada, el DN seleccionado usará la función EscribirPropuesta, para decirle al NameNode(NM) para que escriba en el Log.txt

(Archivo generado al iniciar el NM) con la propuesta aprobada, se reparten los chunks a sus nodos correspondientes con la función Repartir y agregará a los títulos disponibles con la función AgregarTitulo

Al seleccionar Bajar, se solicitará el nombre del archivo a bajar, el cual se buscará en los títulos disponibles, devolviendo las ubicaciones de todas sus partes gracias a la función ObtenerUbicaciones, así mismo, con las ubicaciones de los chunks, la función BuscarChunks solicitará a cada DN correspondiente el chunk que posee, (si no esta disponible dará un mensaje indicando que no es posible recuperarla), al momento de terminar la recolección de chunks, se usará la función Unir, para rearmar el archivo y entregarlo al usuario.

# 2. Resultados

Dado que los algoritmos solicitados (tanto centralizado como Ricart y Agrawala) no fueron implementados de la mejor manera en el desarrollo de la tarea, (asumiendo el descuento indicado en las filas 29 y 39 de la rubrica detallada) preferimos analizar estos algoritmos desde un punto de vista teórico, estableciendo ejemplos y como estos algoritmos lo resuelven.

- Distribuido

  Tenemos una situación, la cual se resuelve con un algoritmo de exclusión mutua distribuida, en donde hay un DataNode que quiere acceder al log pero no ha solicitado entrar, uno ya esta escribiendo en este y uno que ya pidió escribir. Esta situación esta representada en la siguientes estructuras:

  1. DataNode1{id=1, estado="TOMADA", timestamp=3}
  2. DataNode2{id=2, estado="BUSCADA", timestamp=2}
  3. DataNode3{id=3, estado="LIBERADA", timestamp=0}

  Inmediatamente DataNode1 libera la zona critica (1 mensaje) y el DataNode3 pide acceder al log (2 mensajes), pero recibe un mensaje de vuelta solamente (1 mensaje), ya que el DataNode2 tiene un timestamp mayor que el tercero, por lo que queda esperando que este libere el log. Luego de que termina de ocupar la zona critica, avisa al tercer nodo que quedo esperando una respuesta (1 mensaje). Finalmente solo queda DataNode3, por lo que procede a entrar finalmente a ocupar el log. Si lo vemos desde el punto de vista solamente del DataNode3, este recibe y manda 4 mensajes en total.

- Centralizado.

  El coordinador que en este caso es NameNode, que recibe la solicitud de Datanode2, (1 mensaje), acá existen dos opciones: Mandar un mensaje por parte del NameNode (2 Mensaje) hacia Datanode2 indicando que el recurso esta siendo ocupado o no enviar nada, así se añadirán a la cola que constará de (2) al momento de DataNode1 terminar de ocupar el recurso compartido dará aviso al coordinador (3 mensajes) y el coordinador le mandará un mensaje (4 mensajes) al siguiente en la lista para que pueda hacer uso del recurso.

# 3. Análisis

Si analizamos la cantidad de mensajes que necesita el DataNode3 para trabajar en la zona critica, nos daremos cuenta que este necesita 4 mensajes en el caso del sistema distribuido, mientras que en la situación de la implementación con un sistema centralizado, necesitara solo 3 mensajes.

Con respecto al tiempo que se demora para poder escribir en el log, en la implementación distribuida nos encontramos que se demora menos, ya que el sistema centralizado coloca todas las peticiones en una cola manejada por el NameNode, lo cual hace que el DataNode3 espere más en la implementación central.

# 4. Discusión

Primeramente se sabía que el algoritmo de exclusión mutua distribuida (Ricart y Agrawala) manda n-1 mensajes y recibe n-1 respuestas, lo que en total nos daría 2(n-1). También hay que considerar que el tiempo de sincronización es igual al tiempo máximo de transmisión de mensajes, lo cual es una ventaja, ya que otros algoritmos tienen el tiempo de sincronización de un mensaje de ida y otro de vuelta. Una desventaja de este algoritmo, es que en el caso en que cualquiera de los nodos del sistema se caiga o tenga algún error, podría detener el progreso del proceso.

En la implementación con el algoritmo centralizado, se conocía que la cantidad de mensajes que es necesario generar por acceso era 3 siempre, lo que en conjunto con la facilidad de implementación es una ventaja. La desventaja principal, es la implementación de como se administra las peticiones, ya que el NameNode va acumulando en una cola todas las peticione que le lleguen, lo cual implica que se demora más en procesar información que el sistema distribuido. Esto también implica que se puede producir cuellos de botella, ya que debido a la implementación, la comunicación se hace solamente con el NameNode, lo cual implica que se puede generar un colapso por procesar muchas peticiones en poco tiempo.

# 5. Conclusión

El objetivo al final era ver las ventajas y desventajas de cada algoritmo y ver en que situación conviene usar uno u otro. Esto no significa que un algoritmo sea mejor que otro, sino que depende del contexto en que se quieran implementar.

La implementación con el algoritmo de exclusión mutua centralizada dada una conexión, se tendrán menos mensajes para entrar al log, pero más tiempo de espera en comparación con el enfoque de Ricart y Agrawala.