Tarea #2
Redes de Computadores

Profesor: Pablo Serrano
Ayudantes: Ignacio Cofré ignacio.cofre.14@sansano.usm.cl
Javier Ramos javier.ramos.14@sansano.usm.cl
Benjamín Riquelme benjamin.riquelme@sansano.usm.cl
Catherin Vargas catherin.vargas.13@sansano.usm.cl

26 de abril de 2019

1. Objetivos

- Simular el comportamiento de un sistema de almacenamiento distribuido.

2. Introducción

En la actualidad las demandas de almacenamiento ya no son triviales. Empresas gigantes como Facebook o Google poseen el control de cientos de Petas de datos donde se encuentran tanto el conocimiento como la vida de las personas que residen en el planeta. Dado este escenario, es impensable la idea de tener todos estos datos concentrados en un solo disco.

Por lo mismo, estos gigantes han desarrollado varios algoritmos para distribuir la información a través de máquinas de costo bajo y de reemplazo rápido en caso de falla (comodities). Un ejemplo de este tipo de desarrollos es el proyecto de Apache HDFS, el cual implementa un sistema de almacenamiento que permite construir granjas de servidores que juntos simulan grandes discos resilientes a posibles fallas.

3. Tarea

La interacción inicial entre el cliente y servidor es similar a la implementada en la tarea 1: el servidor escucha a través de un socket, se aceptan los mismos verbos (`ls, get, put` y `delete`), se realiza el handshake cliente-servidor, las conexiones deben ser manejadas a través de un Thread Pool y generar el archivo log.

Ahora, los archivos en vez de ser alojados en el disco del servidor, estos serán distribuidos en fragmentos de tamaño fijo, correspondiente a 64 KB por paquete, entre las N máquinas virtuales, las cuales deben ser capaces de manejar dichos fragmentos correctamente en su propio disco. Para la correcta realización a cada pareja se le asignarán 2 máquinas virtuales.

Nota: colocar todos los fragmentos de un archivo en una misma máquina no es una solución válida.

La tarea del servidor ahora se divide en dos procesos, por una parte deberá mantener un índice de los archivos, para saber en qué máquina se está alojando cada fragmento, por otra parte deberá manejar la coordinación para pasar un archivo del cliente a las máquinas y viceversa. Como el servidor no posee los archivos en su disco, el comportamiento de los verbos solicitados por el cliente variarán para ser compatibles con la nueva arquitectura.

- `ls`: listar los archivos actualmente disponibles. Para que un archivo esté disponible, todas las máquinas que contengan alguno de sus fragmentos deben estar activas. En caso que alguna

de las máquinas que posee alguno de los fragmentos esté offline, el archivo no puede ser accedido.

- `get nombre_archivo`: El servidor debe solicitar los fragmentos a las máquinas que alberguen los fragmentos del archivo. El servidor debe ser capaz de reconstruir el archivo a partir de sus fragmentos y enviárselo al cliente.
- `put nombre_archivo`: El cliente enviará el archivo al servidor y este se encargará de elegir la distribución de los n paquetes de tamaño 64 [KB] (en qué máquina serán albergados) agregando esta información al índice y luego enviando los archivos a las máquinas correspondientes.
- `delete nombre_archivo`: eliminar un archivo del índice en el servidor, y luego el servidor comunicarse con las máquinas para eliminar los fragmentos correspondientes.

Tanto las conexiones cliente-servidor como servidor-máquina deben concretarse realizando un handshake en tres pasos, descrito de manera resumida en la siguiente figura:

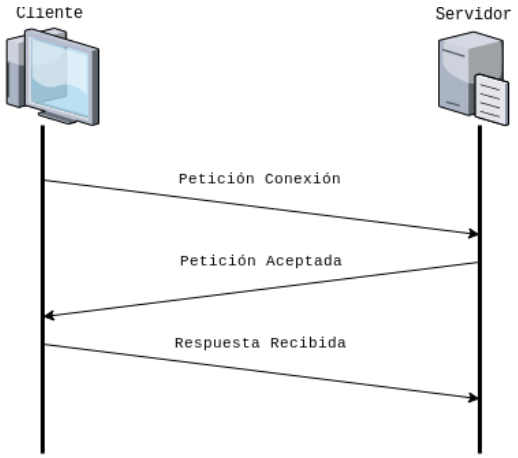

Figura 1: Protocolo de handshake en tres pasos.

3.1. Malla Antigua (ILI)

Deberán agregar la opción de elegir un tamaño de paquete al levantar el servidor, y las máquinas deben ser capaces de conocer el tamaño del paquete definido para el servidor.

3.2. Bonus - 10 puntos

Como se planteó previamente, la caída de una máquina implica que los archivos que posean alguno de sus fragmentos en dicha máquina no estarán disponible. ¿De qué manera, manteniendo el sistema de distribución de fragmentos, pueden evitar este problema? Implemente su solución en la tarea.

4. Restricciones

- No pueden usar la librería de Apache, pero sí utilizar sus interfaces y métodos como guía, también pueden crear sus propias interfaces e implementaciones.
- No se puede usar ninguna implementación de java para la generación del ThreadPool. Ej: Executor, ExecutorService. Ahora sin embargo, pueden ver esas implementaciones e intentar entender cómo funcionan por debajo. Toda otra función definida en la librería default de Java está permitida.
- La única librería externa que está permitido usar es una librería para manejar JSON.
- Los bloques deben ser manejados como bloques de texto codificados en **Base64**.

5. Consideraciones

- Se realizará una ayudantía con el fin de presentar y resolver dudas de la tarea.
- Consultas sobre la tarea se deben realizar en moodle o enviar un correo a **catherin.vargas.13@sansano.usm.cl** o **benjamin.riquelme@sansano.usm.cl**

6. Reglas de entrega

- La tarea se realiza en **grupos de 2 personas**. Si se desea hacer de manera individual, deben contactar tanto a Catherin Vargas y Benjamin Riquelme.
- La fecha de entrega es el día **19 de Mayo a las 23:55**.
- La entrega debe realizarse a través de Moodle, en un archivo comprimido **ZIP** o **TAR.GZ** y debe llevar el nombre **T2-Rol1-Rol2**. Los roles no deben inlcuir el digito verificador. Ejemplo de nombre: `T2-201673507-201573508.zip`.
- Debe entregar todos los archivos fuente necesarios para la correcta ejecución. Recuerde que el código debe estar identado, comentado, sin warnings y sin errores.
- Se aplicará un descuento de 5 puntos al total de la nota por cada Warning, Error o Problema de ejecución.
- Debe entregar un **MAKEFILE** o similar para cada uno de los componentes que utilizará.
- Debe entregar un **README** con nombre y rol de cada integrante, además de la información necesaria para ejecutar los archivos. Se piden instrucciones resumidas, no un informe o testamento.
- No se aceptan entregas que no puedan ser ejecutadas desde una consola de comandos. Incumplimiento de esta regla significa **nota 0**.
- Cada hora o fracción de atraso, se penalizará con un descuento de **10 puntos**.
- Copias serán evaluadas con **nota 0**.
- **Para esta entrega no se realizará ningún cambio de fecha o modificación de los descuentos descritos.**