Trabajo Final - Redes y Comunicación

Jazmine Alfaro Llerena
Manuel Álvarez Sánchez
Renato Postigo Avalos
Paulo Rodríguez Rodríguez

1 Introducción

En el siguiente informe se describirá la arquitectura que tendrá nuestra base de datos no relacional, basada en grafos, distribuida en diferentes servidores todos ellos están ligados por un sistema de comunicaciones, esto hace posible que un usuario al acceder desde cualquier parte de esta red tenga los mismos resultados que obtendría al acceder desde una base de datos centralizada.

2 Objetivo

En la actualidad las grandes empresas han aprovechado el poder de las bases de datos basadas en grafos, ya que estas nos ayudan a encontrar relaciones entre los datos y darles un sentido, además de almacenar efectivamente las relaciones entre datos, son flexibles y escalables porque se van ajustando a las necesidades cambiantes, facilitando así agregar nuevos datos y conexiones. Una de las más conocidas es Neo4j, que es un servicio implementado en Java. Sin embargo las bases de datos basadas en grafos actuales no cuentan con un sistema distribuido, el objetivo de este trabajo es crear una base de datos no relacional que funcione con un sistema distribuido.

3 Propuesta

Al ser nosotros cuatro integrantes en el grupo, distribuiremos la base de datos en tres máquinas de las cuales una será un servidor maestro. Aplicándole una función hash al nombre del nodo, nosotros podremos saber en cuál de las cuatro máquinas estará almacenada la conexión. En la figura 1 podemos ver un gráfico que explica las conexiones a ser realizadas. Cada una de las máquinas que almacene la información tendrá un identificador, que irá de 1 a 3. El cómo conseguir dicho identificador será explicado en la sección 3.1, todas las conexiones se harán en el puerto 50000 del lado del servidor.

Además de esto, una de las cuatro máquinas actuará como el servidor principal, al que el cliente le hará las consultas y el que le dará también una respuesta.

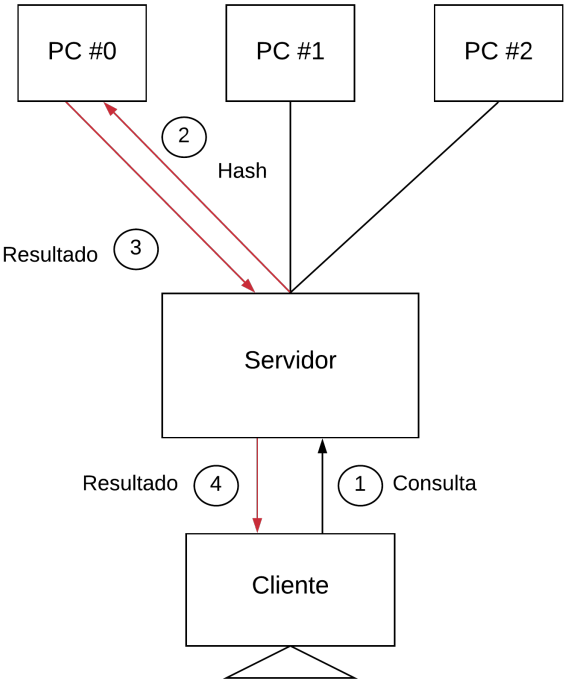

Figure 1: Arquitectura General

3.1 Conexión de las máquinas al servidor

Como se menciona en la sección 3, cada máquina que almacene información tiene un identificador. El servidor no está prendido 24 horas al día, 7 días a la semana, por lo que es necesario que cada máquina pueda saber cuál es su identificador.

Para saber este identificador, se revisa la primera conexión almacenada en la máquina y se aplica la función hash al nodo inicial. El resultado de dicha función será el identificador. En caso no haya elementos, el identificador puede ser cualquier número que no haya sido ya utilizado por otra máquina.

3.2 Estructura del protocolo

Para realizar las distintas operaciones en la base de datos, vamos a implementar el siguiente protocolo para manejar la base de datos no relacional.

3.2.1 Iniciando la conexión

El cliente es el que inicia la conexión, conectándose al servidor, esta secuencia es similar a un three-way handshaking: el cliente envía un mensaje con el texto 'Requesting access', seguidamente el servidor le envía un 'OK', y el cliente le contesta con un 'OK'. En la figura 2 podemos ver su respectivo diagrama de secuencia.

Es importante notar que toda conexión entre el cliente y el servidor se hará en el puerto 40000.

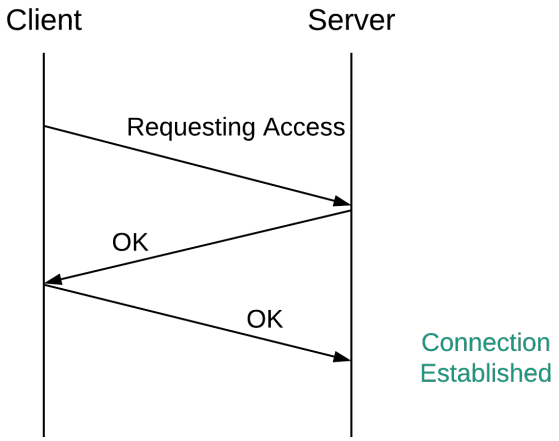

Figure 2: Diagrama de secuencia del inicio de conexión

3.2.2 Inserción un dato

Para insertar una conexión de la forma NODO_1 $->$ NODO_2, el cliente debe enviar un mensaje con la siguiente sintaxis:

INSERT <NODE_1> $->$ <NODE_2> <0|1>

Donde <NODE_1> y <NODE_2> son los nombres respectivos de los nodos, y el flag del final es 0 si queremos que la conexión sea bidireccional y también se guarde la conexión <NODE_1> $->$ <NODE_2>, y 1 si queremos que la conexión sea unidireccional. Por ejemplo:

$$INSERT\ A -> B\ 1$$

En este caso, **solo** se inserta la conexión $A -> B$. Para esto, aplicamos la función hash $f(h)$ en A ($f(A)$) para saber en qué máquina vamos a almacenar dicha conexión. Una vez tengamos el identificador, almacenamos el dato.

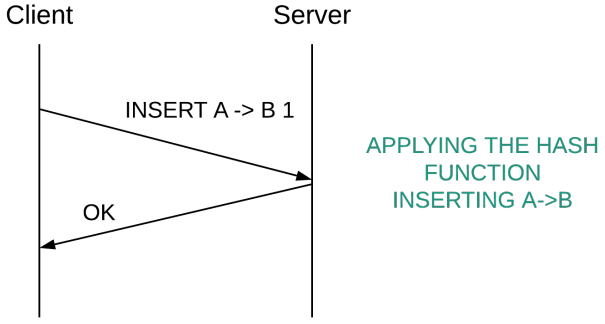

Figure 3: Diagrama de secuencia de la inserción de una conexión unidireccional

Si por el contrario, ocurriera una consulta del tipo:

$$INSERT\ A->B\ 0$$

En este caso, tendríamos que insertar las conexiones $A->B$ y $B->A$. Repetimos el proceso explicado anteriormente para almacenar $A->B$, y posteriormente realizamos lo mismo con $B->A$: aplicamos la función hash $f(h)$ en B ($f(B)$) para saber en qué máquina se almacenará dicha conexión.

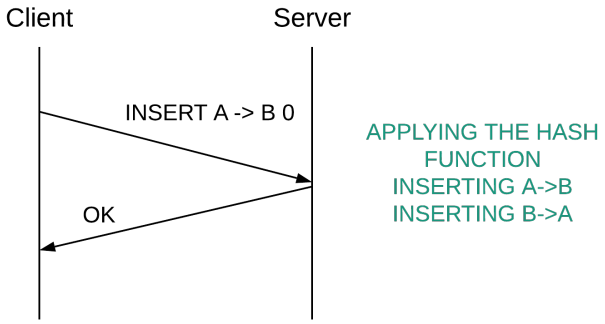

Figure 4: Diagrama de secuencia de la inserción de una conexión bidireccional

En ambos casos almacenamos la conexión en la máquina cuyo identificador sea el valor que nos devuelve la función hash.

La sucesión de mensajes sería en esta sucesión:

1. El cliente envía el INSERT como se menciona arriba.
2. El servidor le aplica la función hash al nodo inicial.
3. Se manda la conexión del grafo a la máquina cuyo identificador sea igual a lo que devuelva la función hash.
4. Esa máquina le manda un 'OK' al servidor.
5. (Si la conexión es bidireccional) El servidor aplica la función hash al nodo final.
6. (Si la conexión es bidireccional) Se manda la conexión del grafo a la máquina cuyo identificador sea igual a lo que devuelva la función hash.
7. (Si la conexión es bidireccional) Esa máquina le manda un 'OK' al servidor.
8. El servidor manda un 'OK' al cliente.

3.2.3 Realizando la consulta

Para poder realizar una consulta y ver los vecinos en diferentes niveles de los nodos, el cliente debe hacer una consulta con la siguiente sintaxis:

$$SELECT \text{ <NODE\_NAME> LEVEL <LVL>}$$

Donde <NODE_NAME> es el nombre del nodo a ser consultado y <LVL> es el nivel en el que queremos consultar. Por ejemplo, si tenemos las siguientes relaciones guardadas en la base de datos:

Relaciones guardadas en el servidor 1

Nodo Origen	Nodo Destino	Dirigido
A	B	1
A	C	0

Table 1: Relaciones del servidor 1

Relaciones guardadas en el servidor 2

Nodo Origen	Nodo Destino	Dirigido
B	C	0

Table 2: Relaciones del servidor 2

Relaciones guardadas en el servidor 3

Nodo Origen	Nodo Destino	Dirigido
C	A	0
C	B	0

Table 3: Relaciones del servidor 3

Teniendo la siguiente consulta:

$$SELECT\ A\ LEVEL\ 1$$

El resultado serían todas las relaciones que tiene A tal como se muestra en la siguiente tabla:

Nodo Origen	Nodo Destino	Dirigido
A	B	1
A	C	0

Table 4: Resultado de la consulta $SELECT\ A\ LEVEL\ 1$

Por otro lado, si se realiza la siguiente consulta:

$$SELECT\ A\ LEVEL\ 2$$

Como resultado se obtendrían todas las relaciones que tiene A y todas las relaciones que tienen los nodos conectados a A tal como se ve en esta tabla:

Nodo Origen	Nodo Destino	Dirijido
A	B	1
A	C	0
B	C	0
C	A	0
C	B	0

Table 5: Resultado de la consulta $SELECT\ A\ LEVEL\ 2$

Para obtener los resultados mostrados en las tablas 4 o en 5 se realizara el siguiente intercambio de mensajes con el servidor:

1. El cliente envía la consulta SELECT tal y como se especifica arriba.
2. El servidor envía 'OK' al cliente significando que recibió su mensaje.
3. El servidor le aplica la función hash al nodo de origen.
4. Dependiendo del resultado de función hash se decide en que servidor buscar el resultado, si el $LEVEL$ es estrictamente mayor a 1 este proceso de repite hasta que se alcance el nivel de resultados deseados.
5. Los datos recolectados por el servidor maestro se envían al cliente.
6. El cliente responde con un mensaje de 'OK' significando que ha recibido la data enviada por el servidor.
7. El cliente muestra los resultados en el PROMPT.

3.2.4 Cerrando la conexión

El cliente es el que decide cuando cierra la conexión, mandando un mensaje que diga 'Closing Connection.' al servidor. El servidor le manda un 'OK.', y el cliente le manda un mensaje 'Confirming connection ending.'.

Para confirmar que se cierra la sesión, el servidor envía un mensaje al cliente con el mensaje 'Are you sure?', y espera la respuesta del cliente, si esta es 'Yes.', entonces el servidor le responde con 'Closing connection confirmed.'. Recién en este momento se procede a cerrar los sockets del cliente.

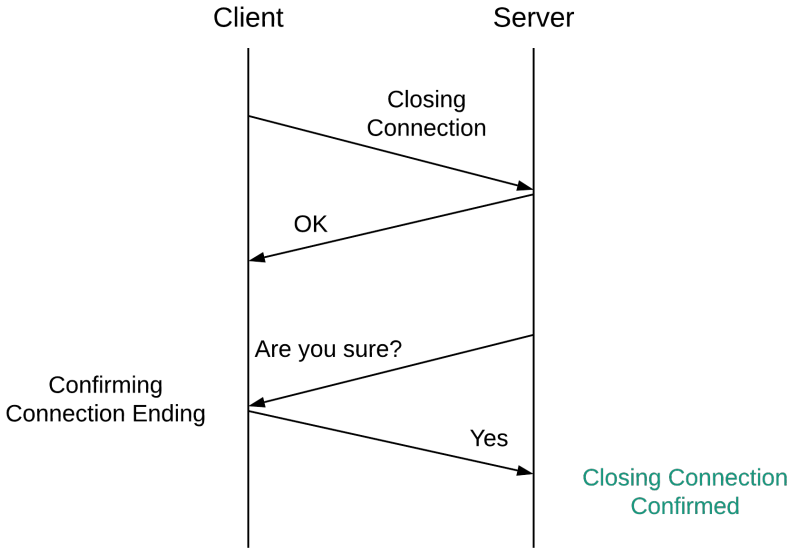

Figure 5: Diagrama de secuencia del cierre de conexión