Servicio de almacenamiento distribuido

Sergio Herrero Barco 698521

Alex Oarga Hategan 718123

Diciembre de 2017

1. Introducción

El sistema diseñado consiste en un almacenamiento de datos distribuidos. Los datos van a ser pares clave, valor. Dicho sistema va a ser tolerante a fallos teniendo los datos duplicados en 1 copia, por lo que el sistema es tolerante a un fallo de caída/perdida de datos.

2. Sección principal

Implementación del servidor

Para la implementación del servidor de almacenamiento distribuido, se ha definido un tipo de dato struct compuesto de un numero de vista, un dato del tipo map para almacenar las parejas clave valor, un nodo primario, un nodo copia y un booleano para saber si la copia es válida o no.

El servidor debe almacenar una serie de datos en todo momento, es decir, debe almacenar un estado. El estado se va a componer de un tipo de dato del tipo struct anteriormente citado llamado atributos.

Para la escritura en la base de datos, se tiene que asegurar que un solo cliente es el que está escribiendo. Para ello, se ha creado un patrón mutex. Cuando un cliente quiere escribir en la base de datos, el servidor le da el mutex, y tras la escritura, el servidor se encarga de devolver el mutex.

Cuando un nodo primario detecta que se ha caído una copia, y un nodo en espera ha sido promocionado, debe copiar los datos al nuevo nodo. Esto se hace, enviándole un mensaje con los datos al nodo copia. Cuando se realiza la copia, el primario no debe dar ningún servicio hasta que no haya copiado todo, por lo que antes de copiar los datos, coge el mutex y cuando lo ha copiado, lo devuelve.

La lectura se puede hacer de manera concurrente, pero el sistema no debe de estar copiando. Esto se ha conseguido, pidiendo el mutex y una vez que se lo han cedido, lo devuelve. Así se asegura, que no estaba copiando los datos.

Para la realización de todas estas funciones, el servidor se encuentra en todo momento esperando la recepción de un mensaje. Cuando llega un mensaje, procesa el mensaje, cambia de estado y finalmente vuelve al estado inicial. Se pueden ver los estados posibles en la figura 1.

Los mensajes que puede recibir (Véase *Figura 2*) y las acciones que debe hacer se muestran a continuación:

- `{:lee, clave, nodo origen}`: Al llegar este mensaje, el servidor debe comprobar si es el nodo primario. En caso de que no lo sea, le envía un mensaje a *nodo_origen* diciéndole que no es nodo primario y vuelve al estado inicial. Si, por el contrario, es el nodo primario, coge y deja el mutex para asegurarse de que no está copiando los datos y busca la *clave* que le envía *nodo_origen*. Si encuentra la clave le envía el valor asociado, y si no le envía una cadena vacía.
- `{:escribe generico, {clave, nuevo valor, es hash}, nodo origen}`: Al llegar este mensaje, el servidor debe comprobar si es el nodo primario. En caso de que no lo sea, le envía un mensaje a *nodo_origen* diciéndole que no es nodo primario,

y vuelve al estado inicial. Si, por el contrario, es el nodo primario, coge el mutex y busca el valor asociado a *clave.* Si la clave existe, sustituye su valor por *nuevo_valor, o* por *hash(valor_antiguo<>valor_asociado)* si es_hash es true. Si la clave no existe. Si la clave no existe, crea una nueva pareja cuyo valor será *nuevo_valor,* o *hash(“ ”<>nuevo_valor)* si *es_hash* es true. Después de la inserción en los datos, copia la nueva base de datos al nodo copia, envía los datos a *nodo_origen* y devuelve el mutex.

- `{:copia_datos, nuevos_datos}`:Este mensaje solo puede ser recibido por un nodo copia. Al llegar este mensaje, cambia sus valores del struct por los que le han enviado.
- `:envia_latido`: Al llegar este mensaje envía un latido al Servidor_GV con la vista que tiene. Recibe un mensaje con una vista, y un booleano. El booleano representa si la vista enviada es válida o no. Si la vista es válida, se actualiza el número de vista y se envía un latido confirmando. Si la vista enviada no es válida, se comprueba si el número de vista es 1, si es así y el que la recibe es primario de su vista, es el caso inicial, así que se le envía un latido -1. Si, por el contrario, el número de vista no coincide con el número de vista del servidor, se actualizan los primarios y copias y el número de vista. Se comprueba si es el primario designado, en caso afirmativo coge el mutex, copia los datos al nodo copia si lo hay, devuelve el mutex y confirma la vista valida. Si el número de vista coincide, se envía un latido con el número de vista.

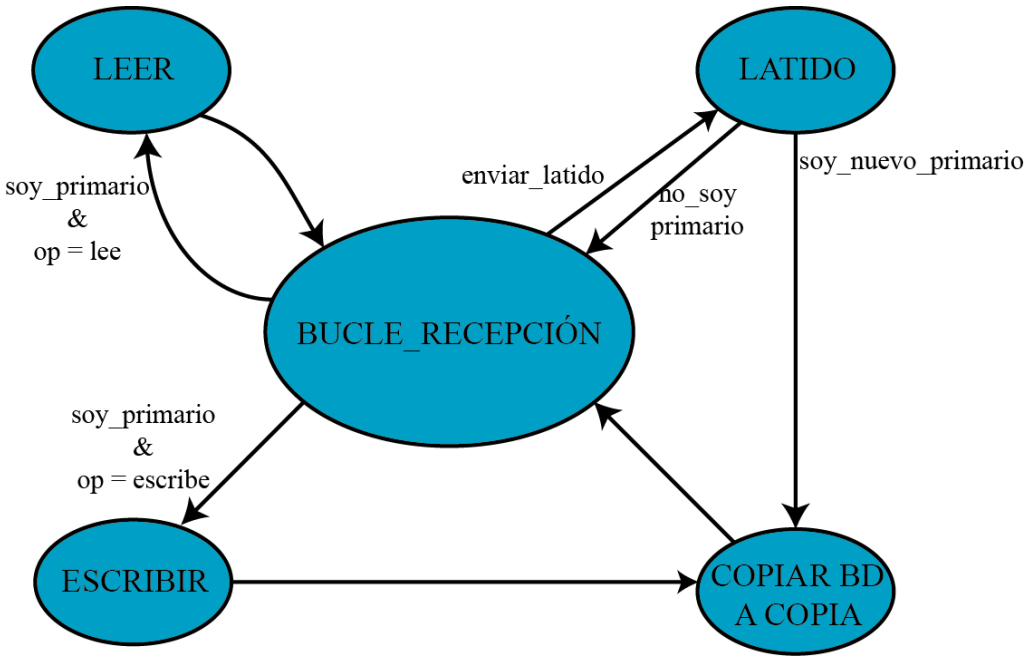

Figura 1: Estados

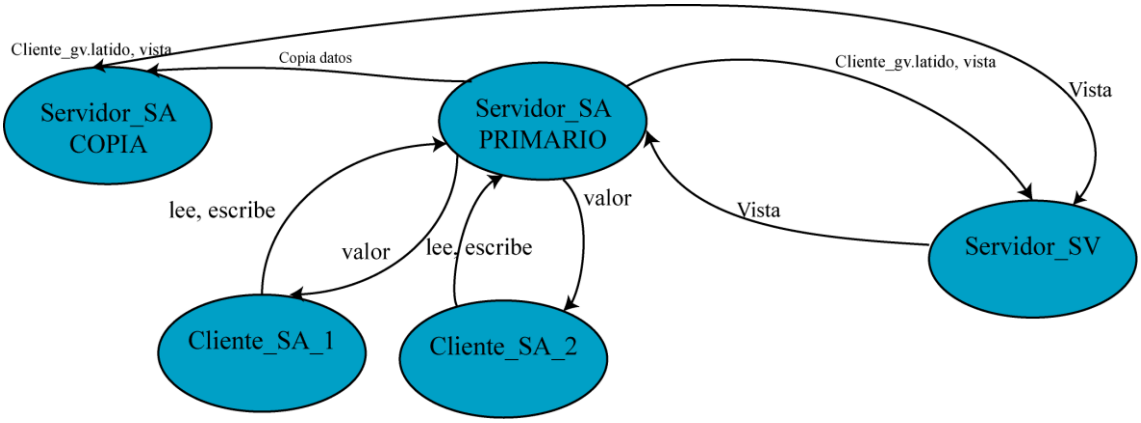

Figura 2: Diagrama de comunicación

3. Validación

Para asegurarse de la correcta implementación, se ha usado el fichero de test de elixir. Al realizar las pruebas en una red Lan, se ha cambiado las direcciones dentro del fichero de las maquinas.

Para que funcione todo de manera correcta, se ha tenido que copiar la llave publica para poder acceder al ssh sin tener que poner la contraseña, y asegurarse que los ficheros están en el mismo *path* en las diferentes maquinas.

Dado que el fichero estaba incompleto, a continuación, se mostrará las acciones para superar el test completo:

- Test 4: Se inician las maquinas, se espera a que se configure nodo primario, copia y esperas. Se comprueba que el nodo primario sea ca1 y la copia ca2. Se escriben dos valores, y se guardan las parejas de clave-valor que se han escrito. Se hace que caiga el nodo primario, se espera a que el sistema vuelva a estar operativo, y se leen los datos de las claves almacenadas. Estos datos estarían almacenados en ese momento, en el anterior nodo copia que ahora ha sido promocionado a primario. Si el sistema es correcto, los valores anteriores deberían ser iguales a los valores obtenidos después. Si es así, **TEST SUPERADO.**