SISTEMAS DISTRIBUIDOS

Práctica 5

Curso 2018-2019

Autores:
Ignacio Palacios Gracia 739359
Rubén Rodríguez Esteban 737215

1. Introducción

La finalidad de esta práctica ha sido el diseño y la implementación un servicio de almacenamiento distribuido clave/valor en la memoria RAM (Random Access Memory) que sea resiliente en cuanto a la tolerancia a fallos empleando el esquema de Primario/Copia implementado en la anterior práctica.

Para atender todas estas tareas el servidor se encuentra en un bucle esperando la recepción de mensajes. De esta forma cuando se produce la llegada de una tarea, ésta se procesa, se cambia el estado en el que se encuentra el sistema para efectuar la secuencia de acciones acordes a la tarea recibida. Una vez que la tarea ha sido efectuada el sistema retorna al estado inicial. La secuencia de acciones y cómo se han implementado se describen en el apartado siguiente.

2. Solución planteada

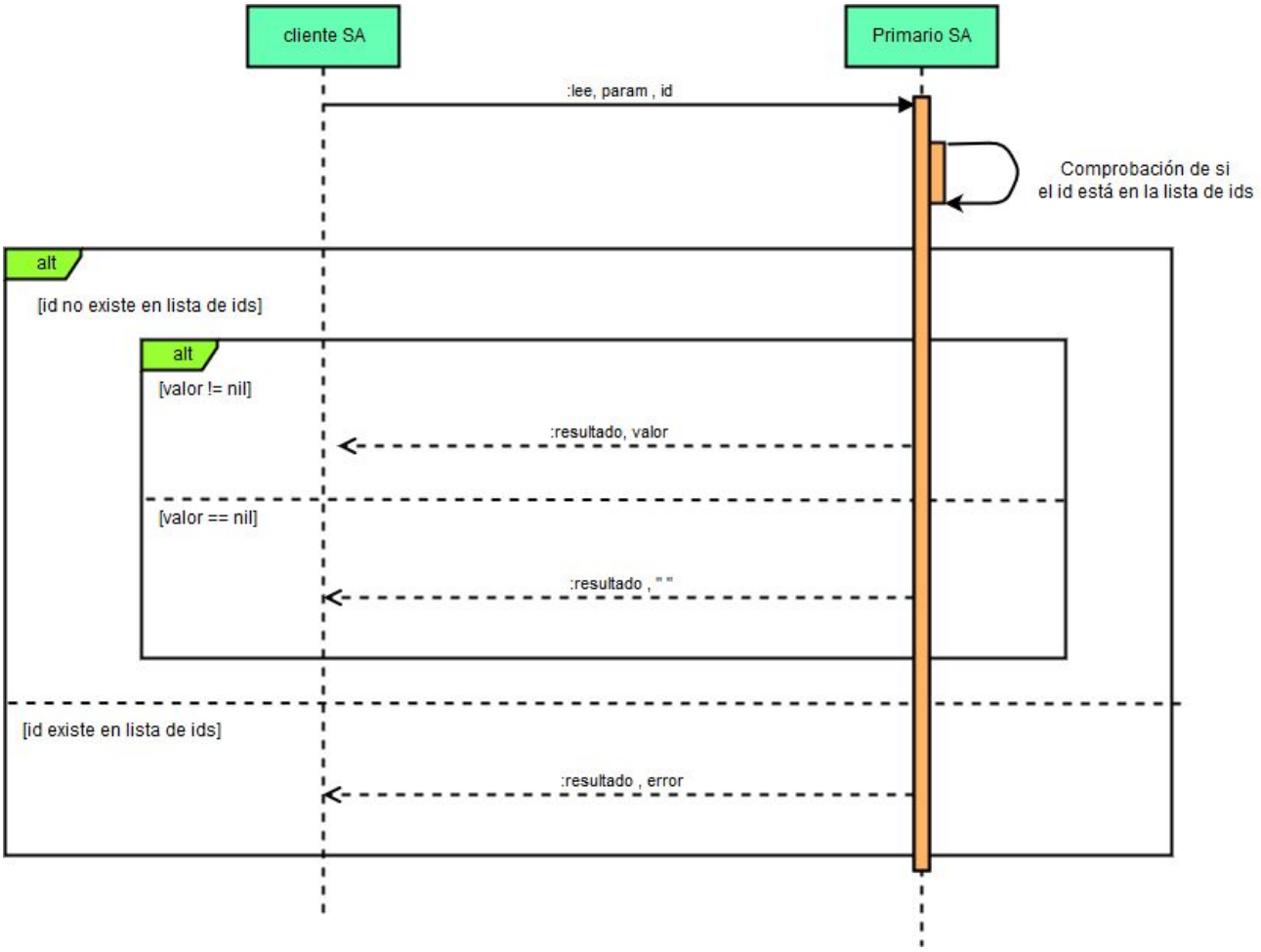
En el diseño del servicio de almacenamiento se ha definido una estructura de datos, compuesta por el número de vista actual, el nodo primario, el nodo copia, una estructura base de datos tipo map para guardar las parejas clave-valor, y por último una lista de operaciones, en la que se almacenan las operaciones ya realizadas en parejas id operación-pid nodo.

El servicio de almacenamiento tiene un bucle principal, que es el que recibe todos los mensajes, tanto del cliente como de sí mismo, y los procesa acordemente. En este bucle principal se encuentra el estado actual del servidor, compuesto por la estructura definida anteriormente. Este estado se irá modificando constantemente según los mensajes que se reciban.

Al arrancar el servidor se servicio de almacenamiento, también ponemos en marcha un proceso aparte que mande un mensaje al bucle principal cada @intervalo latidos, para indicar al servidor de mandar un latido al gestor de vistas, y si la vista que le devuelve es válida, actualiza el estado actual del servicio de almacenamiento.

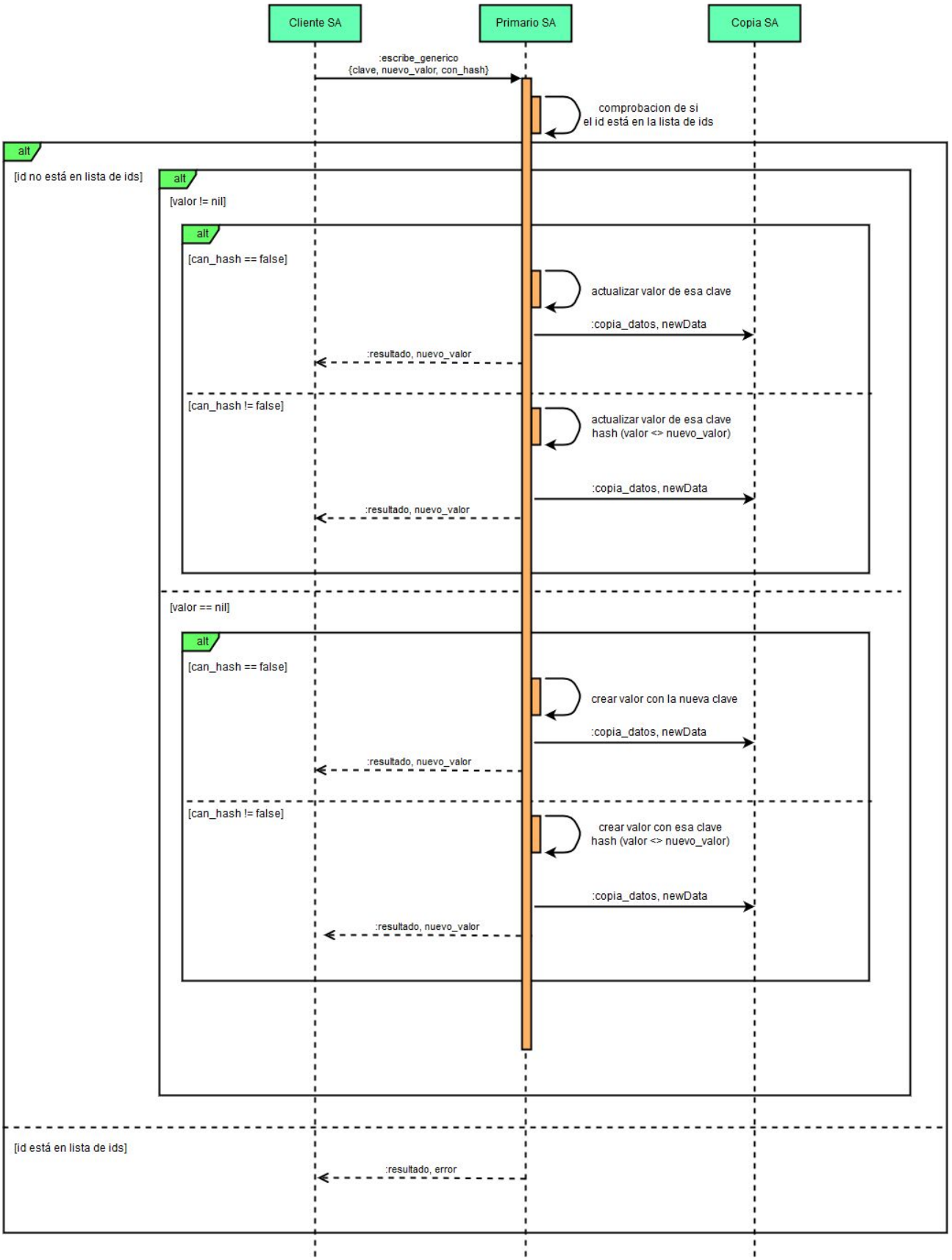
Los mensajes que se pueden recibir, y las acciones que toma el servidor de almacenamiento, son las siguientes:

- **{ :lee, param, nodo_origen, id_operacion}**
  Al llegar este mensaje, se comprueba primero si el servidor que lo ha recibido es el primario. Si no lo es, avisa al cliente_sa de que no es el primario. Si se trata del primario, comprueba que una operación con el mismo id y el mismo nodo de origen no se ha realizado ya, y si se ha realizado mandamos al cliente_sa un aviso de error. Si no se ha realizado, cogemos el valor de la base de datos del estado actual en el que la clave coincida con param, y miramos si tiene un valor asociado o no. Si lo tiene, le mandamos ese valor al cliente_sa, y si no le mandamos un carácter vacío.
  Al terminar la operación, guardamos la pareja id_operacion-nodo_origen en la lista de operaciones del estado actual.



- **{:escribe_generico, {clave, nuevo_valor, con_hash}, nodo_origen, id_operacion}**
  Al llegar este mensaje comprobamos tanto si somos el nodo primario, como si la operación recibida ya se había realizado. Si no se cumplen estas condiciones, mandamos al cliente_sa un aviso de que no soy el primario, o un error, respectivamente. Si existe una valor asociado a la base de datos del estado actual asociado con la clave con la que se va a escribir, se sustituye ese valor por el nuevo valor recibido del cliente, y si el booleano con_hash es true, también se hashea el valor a almacenar. Una vez terminado el almacenamiento de dicho valor en la base de datos, se le manda la base de datos actualizada a la copia del estado actual. Si no existe un valor asociado a la clave recibida, se añade una nueva entrada en la base de datos con esa clave y ese valor, hasheado o no.
  Una vez terminado el proceso de escritura en la estructura de la base de datos, se almacena la pareja id_operacion - nodo_origen a la lista de operaciones realizadas, y se manda al cliente una respuesta con el valor insertado.

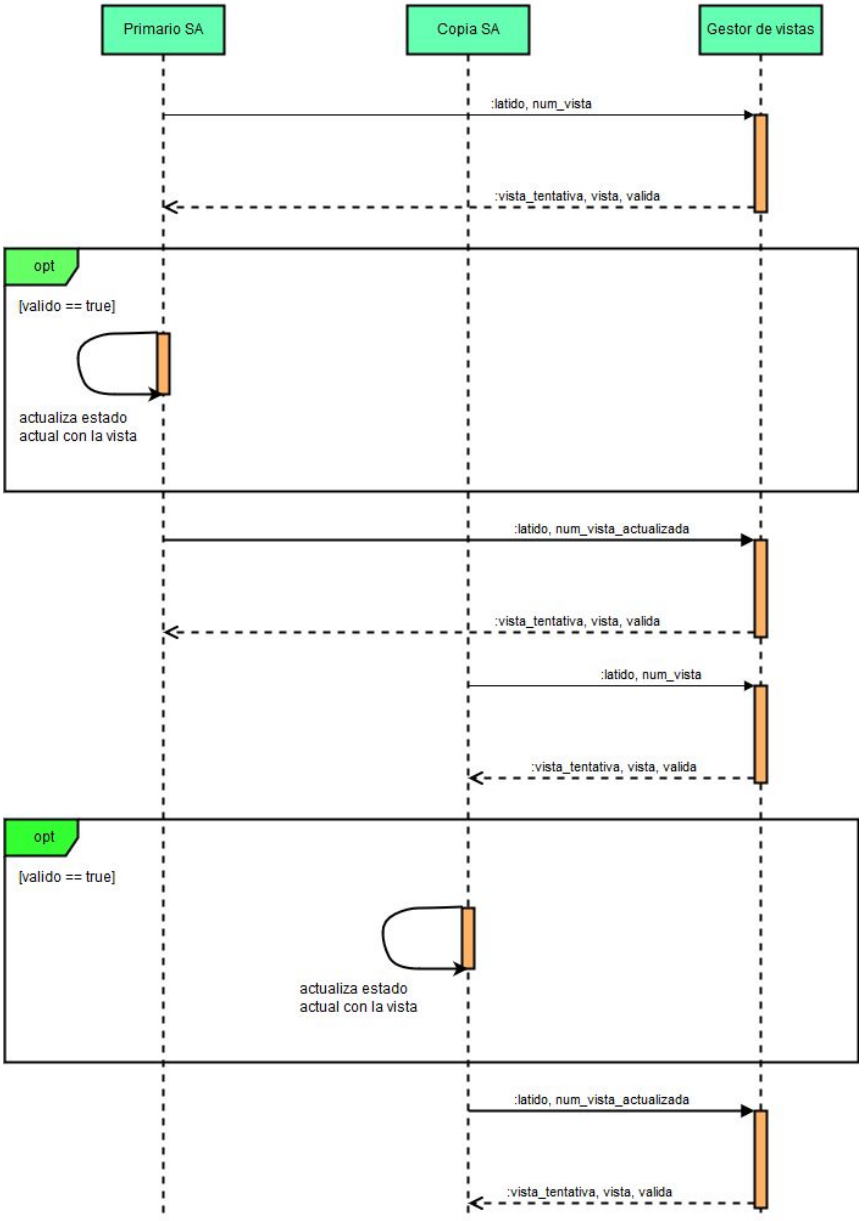
- **{ :copia_basedatos, nuevos_datos}**
  Al llegar este mensaje se actualiza la base de datos del estado actual (solo si soy el nodo copia) con los nuevos datos escritos por el nodo primario en su base de datos.
  El diagrama de secuencia de esta operación puede reflejarse en el diagrama anterior cuando después de llevarse a cabo la actualización o la creación de la clave se efectúa la copia de la información. Por lo que no se ha considerado necesario elaborar un diagrama independiente.

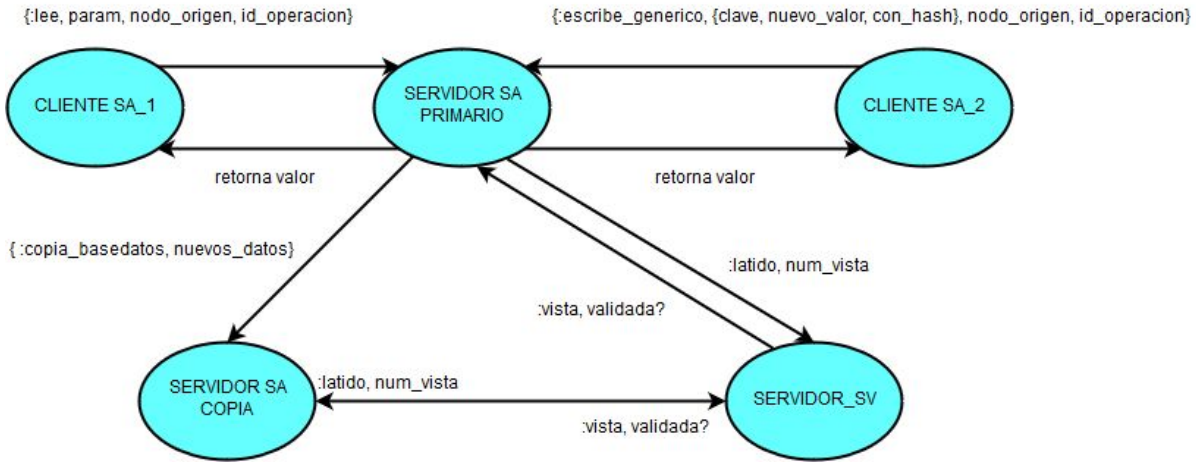

Cliente SA
Primario SA
Copia SA
:escribe_generico
{clave, nuevo_valor, con_hash}
comprobacion de si
el id está en la lista de ids
alt
[id no está en lista de ids]
alt
[valor != nil]
alt
[can_hash == false]
actualizar valor de esa clave
:copia_datos, newData
:resultado, nuevo_valor
[can_hash != false]
actualizar valor de esa clave
hash (valor <> nuevo_valor)
:copia_datos, newData
:resultado, nuevo_valor
[valor == nil]
alt
[can_hash == false]
crear valor con la nueva clave
:copia_datos, newData
:resultado, nuevo_valor
[can_hash != false]
crear valor con esa clave
hash (valor <> nuevo_valor)
:copia_datos, newData
:resultado, nuevo_valor
[id está en lista de ids]
:resultado, error

- **:latido**

  Al llegar este mensaje, que lo manda un proceso aparte del servidor de almacenamiento cada @intervalo_latido tal y como se ha explicado antes, se procede a mandar un latido con el número de vista del estado actual al gestor de vistas.

  Al recibir la confirmación del gestor de vistas con la vista y un booleano con su validación, se comprueba si la vista es confirmada. Si lo es, se actualiza el estado actual del servidor de almacenamiento con el número de vista, el primario y la copia recibidos. Si no, no se actualiza el estado.





3. Validación

Para poder llevar a cabo la evaluación y puesta a prueba de la práctica, el programa ha sido ejecutado en múltiples máquinas. Primero se probó lanzando el script de inicialización creando todos los nodos en la misma máquina. Una vez se comprobó que todos los test facilitados por el profesorado de la asignatura, y los test realizados por los alumnos de este grupo, funcionaban correctamente, se procedió a lanzar cada uno de los clientes y el servidor en máquinas diferentes, confirmando así que funcionaban correctamente tanto en local como de forma remota.

Los test realizados para poner a prueba el programa son los siguientes:

1. Solo arranque y parada.
   No hubo problemas en este test.

2. Algunas escrituras:
   - Comprobar escritura con primario, copia y espera.
   - Comprobar escritura después de fallo de nodo copia.

Fue en este test el que dio más problemas, ya que el código del servidor de almacenamiento le faltaba alguna corrección referido a la escritura, y que la gestión del latido no se realizaba correctamente. Todo esto provocaba que no superase el test, tanto en la parte que se comprobaba que el primario y la copia fuesen las correctas, como en las partes posteriores.

3. Escrituras mismos valores clientes concurrentes.
   No hubo grandes problemas en este test.

4. Escrituras concurrentes y comprobación consistencia.
   No hubo grandes problemas en este test.

5. Petición escritura inmediatamente después de caída del nodo copia.
   No hubo grandes problemas en este test.

6. Petición de escritura duplicada por pérdida de respuesta.
   En este test hubo que incorporar un cambio al código tanto del servidor_sa, como del cliente_sa, y parte de los tests del servicio de almacenamiento. Se incorporó a todos estos ficheros las ids de la operaciones que se ejecutaban, que se pasaba como parámetro. En los tests se incorporó este cambio en los tests anteriores, actualizando la forma en la que leían o escribían. En el cliente_sa se añadió la id en los parámetros de las funciones, y en los mensajes que manda al servidor_sa. En el servidor_sa se utiliza este id para almacenarlo junto con el pid del nodo del cliente que lo manda en una lista de operaciones, para comprobar que no se había ejecutado anteriormente esa misma operación.

4. Conclusión

A lo largo de esta práctica se ha diseñado y puesto en funcionamiento un servidor de almacenamiento, utilizando para ello una estructura de datos compuesta por el número de vista actual, el nodo primario, el nodo copia, una lista con las operaciones realizadas con parejas id_operacion - nodo_emisor, y otra estructura de datos, simulando una base de datos, con las parejas clave-valor escritas por los clientes.
Se ha modificado la estructura original proporcionada por los profesores, especialmente en el fichero cliente_sa, para poder mandar los mensajes e las operaciones con un id de dicha operación.
El servidor_sa recibe cuatro tipos de mensajes diferentes, uno de leer el valor asociado a una clave, otro de escribir un valor asociado a una clave (donde el valor puede ser hasheado o no) si es primario, actualizar la base de datos con el nuevo valor si es copia, y de enviar un latido al gestor de vistas, y recibir la vista actual. Las operaciones de leer y escribir se mandan con un id de operación, y si el cliente ya había mandado una petición con ese id, se le responde con un error.
Se han realizado para la comprobación de su funcionamiento los tests del 1 al 6.