Práctica 2
Repositorio datos asignatura

Eduardo Gimeno Soriano 721615

Sergio Álvarez Peiró 740241

5 de noviembre de 2019

Introducción

El sistema desarrollado es un repositorio de datos distribuido utilizando el algoritmo de exclusión mútua de Ricart-Agrawala y los relojes lógicos de Lamport.

Sección Principal

1. Resumen

En el sistema se identifican dos roles, lector y escritor, existe un único programa en el cual se diferencian por un argumento introducido y un identificador único.

Para la implementación del algoritmo de Ricart-Agrawala, se ha traducido el algoritmo modificado del enunciado de la forma más exacta posible.

2. Arquitectura del sistema

Cada proceso del sistema está formado por cinco subprocesos. Un proceso principal que ejecuta el protocolo de acceso a la sección crítica, así como el acceso a la misma, un proceso encargado de recibir request (request process) y un proceso encargado de recibir permission (permission process).

Para la coordinación de las variables compartidas de estos tres procesos del algoritmo, cada nodo también ejecuta otros dos procesos: un proceso de semáforo binario (mutex) para el acceso a variables que se leen y escriben del proceso shared data. Es decir, el proceso principal y los dos procesos que reciben request y permission tienen que solicitar el acceso al proceso mutex y leer y escribir las variables enviado y recibiendo mensajes del proceso shared_data.

Todos los nodos del sistema conocen la lista de nodos profesores (escritores) y alumnos (lectores) además del repositorio que es el servidor que se ejecuta en un nodo aparte.

2.2 Proceso principal

En primer lugar se ejecuta la operación init del modulo, que hace el spawn de todos los subprocesos iniciales y recibe como argumentos el rol y el identificador del nodo para luego llamar a la función protocol que se ejecuta cada pocos segundos.

En la función protocol se realiza, haciendo uso del mutex y shared data para asegurar la consistencia de las variables, el código correspondiente a acquire_mutex() del algoritmo de Ricart-Agrawala, la sección crítica que se comunica con el repositorio mandándole una operación aleatoria según el rol y una vez recibida la respuesta el código correspondiente a release_mutex() de Ricart-Agrawala.

2.4 Request process

Implementa when REQUEST is recieved del algoritmo de Ricart-Agrawala. Antes de eso, se registra su nombre como :request_process para que el proceso principal de los otros nodos puedan enviarle mensajes.

Cuando se recibe una request, se tiene que decidir si añadir el nodo que la ha hecho a la lista perm_delayed (implementada como una lista de nombres de nodos) o darle el permiso. Para ello se calcula la prioridad, que tiene en cuenta si las operaciones de lectura o escritura son compatibles, (solo se permite una lectura con otra), si el identificador y el reloj del proceso que hace la request es mayor o si el proceso principal del nodo no está intentando entrar también a la sección crítica (cs_state != :out).

2.3 Permission process

Implementa when PERMISSION is recieved del algoritmo de Ricart-Agrawala. Antes de eso, se registra su nombre como :permission_process para que el proceso principal de los otros nodos puedan enviarle mensajes.

Cuando un proceso recibe un permiso, actualiza la variable waiting_for, que se ha implementado como una lista de booleanos que representa si se está esperando a cada elemento, usando el identificador de los nodos como índice. Inicialmente todos están a falso menos el correspondiente al identificador propio. Solo se mandará el acceso a la sección crítica cuando waiting_for esté todo a true.

2.5 Mutex process

Para poder enviar mensajes y recibir mensajes del shared data process, los procesos tienen que pedir el acceso a este proceso. El proceso escucha mensajes de wait y signal a modo de semáforo binario. Contiene una lista de procesos bloqueados esperando una respuesta y un contador que puede tomar valores 0 o 1.

2.6 Shared data process

Se encarga de recibir mensajes (read o write) para las variables, waiting_for, perm_delayed, cs_state, op_type, clock y lrd que son las variables empleadas en el algoritmo de Ricart-Agrawala.

Validación Experimental

Las pruebas realizadas de la ejecución han sido con 5 nodos: un repositorio, 2 alumnos que leen y 2 profesores que leen. También se han ejecutado todos en la misma máquina o en 3 máquinas distintas para lectores, escritores y repositorio en los ordenadores del laboratorio.

Se quiere asegurar la porpiedad de exclusión mutua en las operaciones de escritura, de manera que a la vez solo puedan leerse datos y la escritura sea de manera individual. Para comprobarlo se introducen mensajes que aparecen por pantalla en los que los nodos notifican cuando entran y salen de la sección crítica, la operación y su resultado.

Para asegurar la consistencia de las variables dentro de cada nodo y comprobar que el mutex funciona correctamente, también se añade un modo debug para que cada nodo imprima por pantalla la variable que se cambia. Además, el algoritmo necesita que se cumpla para funcionar correctamente.

Por último, para hacer que la solución sea escalable, los procesos request y permission hacen un spawn para cada mensaje que reciben, generando otros procesos para atender a cada request o permission.

Conclusiones

El algoritmo de Ricart-Agrawala es eficaz a la hora de garantizar la exclusión mútua si no se utiliza un proceso coordinador. El problema más representativo de este algoritmo es que si cualquier nodo del sistema falla se produce un bloqueo del sistema.

Bibliografía

[1] Material de la asignatura de Sistemas Distribuidos del grado de Ingeniería Informática de UNIZAR.

[2] Documentación del lenguaje Elixir https://elixir-lang.org/docs.html