Repositorio Datos Asignatura

Asignatura: Sistemas Distribuidos

Práctica: Práctica 2

Fecha: 31-10-2019

Autores: José Manuel Vidarte Llera, 739729

Diego Marco Beisty, 755232

**INTRODUCCIÓN**

En esta práctica se implementa una infraestructura para alojar un repositorio y acceder al mismo siguiendo el esquema de los lectores / escritores distribuidos.

Para ello se ha optado por una solución descentralizada con un mutex distribuido implementado en elixir siguiendo el algoritmo de Ricart-Agrawala generalizado.

A continuación se explican los detalles de implementación y las pruebas de la corrección del sistema.

**IMPLEMENTACIÓN**

Diagrama de procesos del sistema



Procesos del sistema

Partiendo del algoritmo propuesto de Ricart-Agrawala se ha decidido particionar la implementación en los siguientes procesos. Cada cliente se compone de cinco procesos, tres de ellos se encargan de implementar el algoritmo y los restantes gestionan el almacenamiento y el acceso a las variables que componen el estado de la ejecución del cliente.

El proceso variables globales almacena en qué estado se encuentra el programa (fuera, dentro o intentando leer/escribir), el clock actual, el clock del momento en el que se ha solicitado leer/escibir al resto de clientes y una lista con los clientes que están intentando leer/escribir cuyo clock es superior al del cliente que recibe ó en el caso en el que coincidan, su identificador es mayor, o la matriz de exclusión exclude[op1, op2] = true.

El proceso mutex se encarga de que los procesos accedan en exclusión mutua a las variables globales.

Por otro lado, los otros tres procesos componen las partes del algoritmo que se describen a continuación.

El proceso cliente se encarga, en primer lugar, de coger el mutex, leer y actualizar las variables globales state y clock y mandar la petición de lectura/escritura al resto de clientes. Posteriormente, queda a la espera de recibir confirmación para leer/escribir, esta notificación estará gestionada por el proceso permission, que es el encargado de recibir las confirmaciones de acceso a lectura/escritura del resto de clientes. Una vez el proceso cliente recibe la confirmación, éste solicita al servidor el acceso a la parte del repositorio que le toque en ese momento. Justo después de acabar el acceso al repositorio, éste modifica las variables globales de state y lee los clientes que tenía en la lista, los resetea y les envía un mensaje para confirmarles el acceso.

El proceso restante, request, es el encargado de recibir las peticiones del resto de clientes. Éste junto con los datos recibidos, lee su clock, pide el mutex y modifica las variables globales de state y clock (máximo entre el recibido y el leído), lee su lrd y devuelve el mutex. En función del ldr recibido y el leído, el estado, las prioridades y la exclusión mútua entre operaciones, el cliente decide si le envía el permiso de acceso o lo añade a la lista de los clientes pendientes en sus variables globales.

La ejecución de todos estos procesos se realiza de forma concurrente para gestionar todas las posibles peticiones y estados de ejecución del programa como corresponde en el algoritmo.

Arranque del sistema

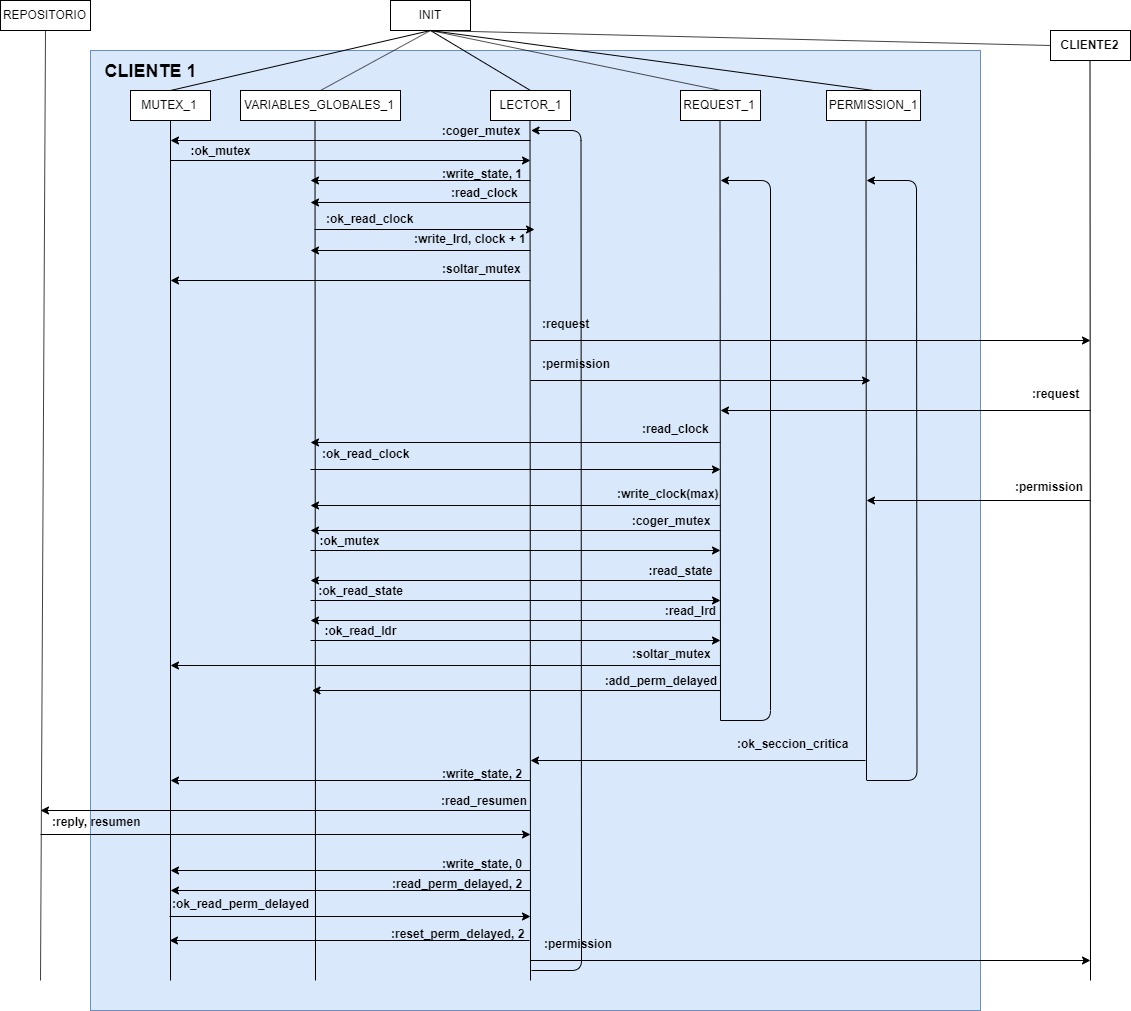
Para iniciar todos los clientes con sus respectivos procesos, a un proceso init se le pasa un número de lectores y escritores junto con una lista de máquinas cliente disponibles. El proceso cliente ejecuta Node.spawn/4 de las funciones permission request e initCliente\*\* en cada máquina cliente disponible.



\*\* initCliente en lectores corresponde a la función initLector y en escritores initEscritor.

**DIAGRAMA DE SECUENCIAS**

Caso un lector (CLIENTE 1), de identificador 1 y un escritor (CLIENTE 2) de identificador 2 , quieren acceder al servidor (REPOSITORIO).

En el diagrama se puede observar como el lector y el escritor se mandan una petición de acceso al servidor(:request). El tiempo de petición de ambos procesos es el mismo (lrd = 1) pero lector tiene un identificador menor por lo que escritor le manda el permiso (:permission) y lector accede al servidor. 

Lector termina acceso al servidor y manda permiso al escritor(:permission).

**BATERÍA DE PRUEBAS**

Para probar la corrección del sistema hemos ejecutado los siguientes escenarios:

1E= Un escritor. 1L =Un lector. 2L= Dos lectores. 4E= 4 escritores. 4L= 4 lectores

|  |  |  |
| --- | --- | --- |
| ESCENARIO | PRUEBAS | DESCRIPCIÓN |
| 1E, 1L | exclude[read, write]  exclude[write, read] | Dentro del servidor accede 1L o 1E. |
| 1E, 2L | exclude[read, read] | Si 1E en servidor -> exclusión mútua con los 2L.  Si 1L en servidor -> exclusión mútua con 1E.  Probamos |
| 4E, 4L | exclude[write, write] | Si 1E en servidor -> exclusión mútua con 3E y 4L  Si 1L en servidor -> exclusión mútua con 4E |

ESCENARIO 1E, 1L ESCENARIO 1E, 2L ESCENARIO 4E, 4L

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Servidor | Clientes |  | Servidor | Clientes |  |  |  |
| update\_resumen  #PID<11111.118.0>  read\_resumen  #PID<11110.118.0>  update\_principal  #PID<11111.118.0>  read\_principal  #PID<11110.118.0>  update\_entrega  #PID<11111.118.0> | lector 2: REQUEST  escritor 1: REQUEST  escritor 1: ENTRAR  escritor1: SALIR  escritor 1: REQUEST  lector 2: ENTRAR  lector2: SALIR  lector 2: REQUEST  escritor 1: ENTRAR  escritor1: SALIR  escritor 1: REQUEST  lector 2: ENTRAR  lector2: SALIR  lector 2: REQUEST  escritor 1: ENTRAR  escritor1: SALIR |  | update\_resumen  #PID<11110.118.0>  read\_resumen  #PID<11112.118.0>  read\_resumen  #PID<11111.118.0>  update\_principal  #PID<11110.118.0>  read\_principal  #PID<11112.118.0>  read\_principal  #PID<11111.118.0> | ector 3: REQUEST  escritor 1: REQUEST  lector 2: REQUEST  escritor 1: ENTRAR  escritor1: SALIR  escritor 1: REQUEST  lector 2: ENTRAR  lector 3: ENTRAR  lector3: SALIR  lector 3: REQUEST  lector2: SALIR  lector 2: REQUEST  escritor 1: ENTRAR  escritor1: SALIR  escritor 1: REQUEST  lector 2: ENTRAR  lector 3: ENTRAR  lector2: SALIR  lector 2: REQUEST  lector3: SALIR |  | update\_resumen  #PID<11114.118.0>  update\_resumen  #PID<11112.118.0>  update\_resumen  #PID<11113.118.0>  update\_resumen  #PID<11109.118.0>  read\_resumen  #PID<11107.118.0>  read\_resumen  #PID<11111.118.0>  read\_resumen  #PID<11110.118.0>  read\_resumen  #PID<11108.118.0> | escritor 4: REQUEST  lector 7: REQUEST  lector 5: REQUEST  escritor 1: REQUEST  escritor 2: REQUEST  lector 8: REQUEST  lector 6: REQUEST  escritor 3: REQUEST  escritor 1: ENTRAR  escritor1: SALIR  escritor 1: REQUEST  escritor 2: ENTRAR  escritor2: SALIR  escritor 2: REQUEST  escritor 3: ENTRAR  escritor3: SALIR  escritor 3: REQUEST  escritor 4: ENTRAR  escritor4: SALIR  escritor 4: REQUEST  lector 5: ENTRAR  lector 7: ENTRAR  lector 6: ENTRAR  lector 8: ENTRAR  lector7: SALIR  lector 7: REQUEST  lector6: SALIR  lector 6: REQUEST  lector8: SALIR  lector 8: REQUEST  lector5: SALIR |