Universidad Nacional Autónoma de México
Facultad de Ciencias

Computación Concurrente

Práctica 5

Bonilla Reyes Dafne - 319089660
García Ponce Jose Camilo - 319210536
Juárez Ubaldo Juan Aurelio - 421095568

Snapshots y Collects: Backups

Descripción de los programas

- `ExecuteSnapshot.java:`

  Programa que implementa la ejecución de *WFSnapshot* usando la clase `WFSnapshot<T>` (código de la profesora).

- `ExecuteSnapshotQ.java:`

  Programa que implementa la ejecución de *WFSnapshotH* usando la clase `WFSnapshotH<T>` (código de la profesora).

- `ExecuteSnapshotS.java:`

  Programa que implementa la ejecución de *SimpleSnapshotH* usando la clase `SimpleSnapshotH<T>` (el Snapshot obstruction-free).

- `ExecuteSnapshotSC.java:`

  Programa que implementa la ejecución de *SimpleSnapshotHC* usando la clase `SimpleSnapshotHC<T>` (el Snapshot obstruction-free con collect).

- `RunnableQ.java:`

  Programa que implementa el runnable para el programa `ExecuteSnapshotQ.java` (código de la profesora).

- `RunnableS.java:`

  Programa que implementa el runnable para el programa `ExecuteSnapshotS.java`.

- `RunnableSC.java:`

  Programa que implementa el runnable para el programa `ExecuteSnapshotSC.java`.

- `SimpleSnapshot.java:`

  Programa que implementa el Snapshot obstruction-free (código de la profesora).

- `SimpleSnapshotH.java:`

  Programa que implementa el Snapshot obstruction-free pero con los cambios para poder tener una lista de valores.

- `SimpleSnapshotHC.java:`

  Programa que implementa el Snapshot obstruction-free pero con los cambios para poder tener una lista de valores y solo usando collect en el scan.

- `SnapshotHC.java:`

  Interfaz para Snapshots (código de la profesora).

- `StampedSnap.java:`

  Código para representar snap con una marca (código de la profesora).

- `StampedSnap.java:`

  Código para representar snap con una marca, que tiene una lista para valores y otro para snaps (código de la profesora).

- `StampedValue.java:`

  Código para representar valor con una marca (código de la profesora).

- `StampedValueH.java:`

  Código para representar valor con una marca, que tiene una lista para valores.

- `ThreadID.java:`

  Código para representar IDs de hilos (código de la profesora).

- `WFSnapshot.java:`

  Programa que implementa el Snapshot wait-free (código de la profesora).

- `WFSnapshotH.java:`

  Programa que implementa el Snapshot wait-free pero con los cambios para poder tener una lista de valores y snaps (código de la profesora).

Reporte de ejercicios

1. **Implementa el mismo modelo que en el programa (2), reemplaza el Snapshot Wait-free por un Snapshot Obstruction-free.**

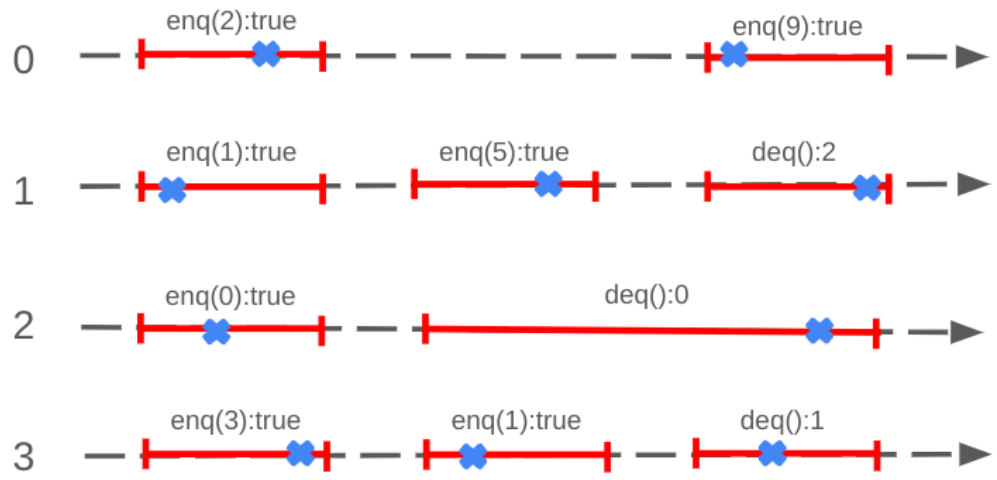
   Este ejercicio fue implementado en los archivos `StampedValueH.java`, `SimpleSnapshotH.java`, `RunnableS.java` y `ExecuteSnapshotS.java`

2. **A partir de tu implementación, dibuja o describe una ejecución de 10 operaciones. ¿Observas alguna diferencia al utilizar un Snapshot obstruction-free en vez de uno wait-free? Argumenta porque crees que sucede así.**

   Usaremos las siguiente ejecución dada por nuestra implementación

```
Ejecutanto el SimpleSnapshotH
Snapshot Final --- Values

 Thread Owner: 0 Last Stamp: 2 Number of Snaps:
The values are:
[enq( 2 )] + [enq( 1 )] + [enq( 0 )] + [enq( 3 )] |  true ||
[enq( 2 ), enq( 9 )] + [enq( 1 ), enq( 5 ), deq()] + [enq( 0 ), deq()] + [enq( 3 ), enq( 4 ), deq()] |  true ||

 Thread Owner: 1 Last Stamp: 3 Number of Snaps:
The values are:
[enq( 2 )] + [enq( 1 )] + [enq( 0 )] + [enq( 3 )] |  true ||
[enq( 2 )] + [enq( 1 ), enq( 5 )] + [enq( 0 ), deq()] + [enq( 3 ), enq( 4 )] |  true ||
[enq( 2 ), enq( 9 )] + [enq( 1 ), enq( 5 ), deq()] + [enq( 0 ), deq()] + [enq( 3 ), enq( 4 ), deq()] |  2 ||

 Thread Owner: 2 Last Stamp: 2 Number of Snaps:
The values are:
[enq( 2 )] + [enq( 1 )] + [enq( 0 )] + [enq( 3 )] |  true ||
[enq( 2 ), enq( 9 )] + [enq( 1 ), enq( 5 ), deq()] + [enq( 0 ), deq()] + [enq( 3 ), enq( 4 ), deq()] |  0 ||

 Thread Owner: 3 Last Stamp: 3 Number of Snaps:
The values are:
[enq( 2 )] + [enq( 1 )] + [enq( 0 )] + [enq( 3 )] |  true ||
[enq( 2 )] + [enq( 1 ), enq( 5 )] + [enq( 0 ), deq()] + [enq( 3 ), enq( 4 )] |  true ||
[enq( 2 ), enq( 9 )] + [enq( 1 ), enq( 5 ), deq()] + [enq( 0 ), deq()] + [enq( 3 ), enq( 4 ), deq()] |  1 ||
camilo@wowi:~/CC/septimoSemestre/Con/Computacion-Concurrente/Practicas/Practica05$
```

Y obtuvimos esta ejecución



Entonces, las diferencias que podemos observar entre un Snapshot obstruction-free y uno wait-free es que, en el caso de obstruction-free, los hilos pueden llegar a ser bloqueados temporalmente por otros hilos que acceden a los mismos recursos, lo cual impide el progreso continuo si hay demasiada interferencia. En cambio, en el caso de wait-free, se garantiza que cada hilo completará su operación en un número finito de pasos, sin importar el comportamiento de los demás hilos, asegurando un progreso constante para todos. En nuestra ejecución, podemos ver que no se muestra un estado consistente entre los hilos, esto se debe a que el progreso de cada hilo se puede ver interrumpido por otros hilos, lo que causa que un Snapshot obstruction-free muestre resultados inconsistentes en sistemas de alta concurrencia.

3. **Implementa el mismo modelo que en el programa (2), reemplaza el Snapshot Wait-free por un Collect, para ello modifica el *scan()* del Snapshot obstruction-free (código en 3) como en la Tarea 4, en vez de hacer *double - collects* haz solo un *collect*.**

Este ejercicio fue implementado en los archivos `StampedValueH.java`, `SimpleSnapshotHC.java`, `RunnableSC.java` y `ExecuteSnapshotSC.java`

4. **Los collects, a diferencia de los snapshots, no son objetos atómicos, de hecho, ni siquiera tienen especificación secuencial. Sin embargo, analizando lo que se escribió en el collect, ¿crees que es posible obtener la ejecución así como con los snapshots? ¿Qué diferencias observas? Justifica tu respuesta**

   Creemos que si es posible obtener la ejecución de una manera similar a con los snapshots, esto debido principalmente a que en la tarea 4 vimos que la implementación del snapshot con un solo collect seguía siendo linealizable, por lo tanto podríamos generar una ejecución la cual tuviera sentido y fuera linealizable, de una manera similar a lo realizado en el ejercicio 2.
   La principal diferencia seria en la manera en la que tenemos que poner el momento en el que se realizan las operación (los tachecitos), ya que como los collects pueden regresan viejos y nuevos, tenemos que tener mucho cuidado de como acomodarlos para que los valores ya tengan más sentido, esto no pasa cuando se realizan varios collects, ya que como se realizan varios collects podemos tener la seguridad de que los scans van a regresar valores que sean congruentes y tengan sentido, pero al usar un solo collect los valores no pueden tener tanto sentido y ahi es donde entra la decisión de cuando poner el tachecito y que todo se pueda linealizar. En resumen con collects si es posible obtener una ejecución pero existe la posibilidad de que los valores obtenidos sean algo inconsistentes y tal vez no reflejen el estado completo de todos los hilos, a diferencia de los snapshots.

5. **Investiga y escribe de forma breve en qué consiste el lenguaje *TLA+*, cuándo fue inventado y enumera 3 empresas que lo utilicen.**

   TLA+ es un lenguaje de alto nivel para modelar programas y sistemas, especialmente los concurrentes y distribuidos. Fue creado por Leslie Lamport en la década de 1990 y se basa en la idea de que la mejor manera de describir las cosas con precisión es con matemáticas simples. Este lenguaje es usado para encontrar y eliminar errores que pueden ser difíciles de encontrar corregir en el codigo.[1]

   Algunas de las empresas que lo utilizan son:

   - **Amazon Web Services (AWS)**: Usa TLA+ para verificar la corrección de servicios distribuidos críticos.
   - **Microsoft**: Aplica TLA+ en la especificación de algoritmos de sincronización y servicios de nube.
   - **Oracle**: Usa TLA+ para probar y especificar aspectos críticos de sus bases de datos distribuidas.

Referencias

[1] The TLA+ home page. (s.f.). https://lamport.azurewebsites.net/tla/tla.html