Programación Concurrente

Trabajo Práctico

Se desea implementar en Java usando métodos *synchronized* una clase "monitor" que encapsule el comportamiento de un vector de números garantizando un correcto acceso concurrente. El vector incluye una operación para calcular la derivada de una función numéricamente mediante el método de diferencias finitas. Además debe proveer las operaciones provistas en el prototipo que acompaña el enunciado. El funcionamiento de cada operación debe ser equivalente al dado de forma secuencial, pero con el agregado de que deben poder resolverse concurrentemente. Para esto durante la creación del vector se deben tomar los siguientes parámetros (todos estrictamente mayor a cero):

- `dimension`, que indica la cantidad de elementos que puede almacenar el vector
- `threads`, que indica la cantidad máxima de threads a utilizar para realizar las operaciones concurrentemente
- `load`, que indica la cantidad de elementos en la que puede diferir la asignación a cada thread

Por ejemplo, un vector de `dimension` 11, con 5 `threads` y `load` 1 realizando la operación `abs` puede distribuir la carga entre los 5 threads trabajando con 2 elementos en 4 threads y 3 elementos en el thread restante. Por el contrario si el `load` fuera 11, todos los elementos podrían ser procesados en un sólo thread.

Tenga en cuenta que la derivada por el método de diferencias finitas a implementar considera los dos elementos adyacentes a una posición a la hora de aproximar la derivada. Por lo tanto aquellas posiciones que por estar cerca de un extremo del vector no tengan valores alrededor son descartadas y en consecuencia el vector resultado es de una dimensión menor que el pasado por parámetro.

1. Escribir un *test suite* que logre un 100 % de cobertura de sentencias de la versión secuencial. Para escribir el test suite se recomienda:
   - Testear la operación de derivada considerando funciones bien conocidas, como constantes, rectas, funciones cuadráticas y otras funciones polinomiales que considere pertinente.
   - Medir la cobertura de sentencias usar los plugins de eclipse clover[1] o eclemma[2].
   - Generar casos de tests usando el plugin de Eclipse para generación automática de casos de tests EvoSuite[3].
2. Escribir la versión concurrente de la clase secuencial. El test suite debe pasar (es decir, ningún test debe fallar) en ambas versiones (secuencial y concurrente)

La entrega debe incluir el test suite creado y la versión concurrente de la clase suministrada.

[1] https://confluence.atlassian.com/display/CLOVER/
[2] http://eclemma.org
[3] http://www.evosuite.org