Programación Concurrente y de Tiempo Real
Grado en Ingeniería Informática
Asignación de Prácticas Número 3

Se le plantean a continuación un conjunto de ejercicios sencillos de análisis de rendimiento de programas en versiones secuencial y concurrente, que debe resolver de forma individual como complemento a la tercera sesión práctica. Para cada uno, debe desarrollar un programa independiente que lo resuelva. El objetivo de la asignación es aprender a efectar paralelismo de datos con división manual de la nube de datos. **Documente todo su código con etiquetas (será sometido a análisis con `javadoc`)**. Si lo desea, puede también agrupar su código en un paquete de clases, aunque no es obligatorio.

1. Ejercicios

1. Queremos efectuar el producto escalar de dos vectores reales de 10^6 componentes. Comience por escribir un programa secuencial que desarrolle el cálculo y guárdelo en `prodEscalar.java`. Ahora escriba un programa que efectúe el cálculo de forma paralela, utilizando división manual de los datos. Para ello, escriba una clase `prodEscalarParalelo.java` que modele a las hebras mediante herencia de la clase `Thread`. El constructor de clase podría ser similar a la siguiente:

```
public prodEscalarParalelo(int idHebra, int inicio, inf final)
```

donde los parámetros del constructor le indican a cade hebra cuál es su identificador (un número distinto para cada hebra, asignado desde el programa principal), y dónde comienzan y terminan los subvectores de datos que les corresponde procesar. El resultado de ese procesamiento será almacenado por cada hebra en una ranura `productoParcial[idHebra]` de un array común a todas hebras. El programa principal creará y lanzará concurrentemente las hebras, esperará a que concluyan, y sumará todas las ranuras del vector `productoParcial` para obtener el resultado final. Tome tiempos para el programa secuencial, y para el programa paralelo con un número de hebras igual a $2, 4, 8, 10$ y escriba sus resultados en formato tabular, en el documento `tiemposProdEscalar.pdf`

2. Se desea disponer de un programa que realice de forma paralela el producto de una matriz cuadrada de $n \times n$ componentes por un vector $A \cdot b = y$ también de n componentes de acuerdo al siguiente esquema:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Ambas estructuras de datos se rellenarán con datos aleatorios enteros obtenidos a través de una instancia de la clase `Random`. Escriba primero un programa que solucione el problema de forma secuencial, y llámelo `matVector.java`. Reescriba ahora su programa para realizar el producto de forma paralela mediante concurrencia por implementación de la interfaz `Runnable`, utilizando paralelismo de datos por división manual del dominio (cada hebra será reponsable de un número determinado de filas), con diferente número de tareas paralelas $n = 2, 4, 8, ..,16$. Cada tarea necesitará saber de cuántas filas es responsable (por ejemplo, vía constructor). El programa principal creará y lanzará las hebras, esperando posteriormente a que terminen. Guarde su trabajo en `matVectorConcurrente.java`. Tome tiempos para la versión secuencial y las diferentes versiones paralelas, y construya una curva $tiempo = f(hebras)$. Tome nota también de los picos de CPU y construya una curva $\%_{CPU} = f(hebras)$

NOTAS:

- Esta aproximación que proponemos al producto paralelo de matrices es «ingenua», y no responde en absoluto al algoritmo estándar utilizado para multiplicar matrices de forma paralela. Únicamente pretende introducir el modelo de paralelismo de datos en arrays bidimensionales.
- Para observar mejoras de rendimiento con la aproximación paralela, n debe ser «grande» (del orden de varios miles). Si se requiere, la cantidad de memoria de trabajo de la JVM puede aumentarse utilizando el *flag* `-Xmx`.

3. Repita todo lo anterior cambiando de sistema operativo, por ver qué pasa. Si utilizó Linux ahora empleará Windows, o al revés. Trace nuevamente las curvas, e intente determinar si el cambio de sistema operativo influye en los tiempos que se obtienen, y en cómo se utiliza la CPU. Escriba un corto documento que integrará toda la información generada en los ejercicios 2 y 3; esto es, curvas de tiempo y uso de CPU para ambos sistemas operativos, y sus reflexiones sobre todo el asunto. Guárdelo en `analisis.pdf`.