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`).

1. Ejercicios

1. Pida a ChatGPT que le ilustre sobre el funcionamiento básico del método `System.nanoTime()` de Java. Pida ahora a ChatGPT que le instruya sobre cómo medir tiempos de ejecución de un programa, utilizando una cadena de solicitud de trabajo como la siguiente: «Escribe un programa en Java para ordenar un vector de números reales de 1000 componentes que muestre cómo medir el tiempo de ejecución.». Compile y ejecute el programa. ¿Hay alguna tarea mejorable en el trabajo proporcionado por el bot?

2. 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 tener una estructura similar a la siguiente:

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

donde los dos 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 lse corresponde procesar. El resultado de ese procesamiento será almacenado por cada hebra en una ranura `productoParcial[idHebra]` de un vector común a todas

hebras. El programa principal creará y lanzará concurrentemente las hebras, esperá 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 $2, 4, 6, 8, 10$ y escriba sus resultados en una tabla `tiemposProdEscalar.pdf`

3. Se desea disponer de un programa que realice de forma paralela, supuesto un procesador *multi-core,* el producto de una matriz cuadrada de $10^3 \times 10^3$ componentes por un vector $A \cdot b = y$ también de 10^3 componentes de acuerdo al siguiente esquema:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

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, con diferente número de tareas paralelas $n = 2, 4, 8, ..,16$. Cada tarea necesitará saber de cuántas filas es responsable. 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(número de tareas)`. Tome nota también de los picos de CPU y construya una curva `CPU=f(número de tareas)`. Para el desarrollo de las curvas, recomendamos el uso de GnuPlot (http://gnuplot.info/).

Esta aproximación que proponemos al producto paralelo de matrices es algo «ingenua», y no responde en absoluto al algoritmo estándar de bloques utilizado para multiplicar matrices de forma paralela. Únicamente pretende introducir el modelo de paralelismo de datos en arrays bidimensionales. Puede ser un buen momento para pedir a ChatGPT que le ilustre sobre el algoritmo de producto paralelo por bloque de matrices, e incluso por su implementación en Java, cuyo rendimiento puede comparar con la desarrollada por usted.

3. Repita todo lo anterior cambiando de sistema operativo. 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; esto es, tablas, curvas, y sus reflexiones sobre todo el asunto. Guárdelo en `analisis.pdf`