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 debe 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 en una co-rutina 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 su sesultados en una tabla `tiemposProdEscalar.pdf`

2. Se desea disponer de un programa que realice de forma paralela, supuesto un procesador *multi-core,* el producto de una matriz cuadra de $10^6 \times 10^6$ componentes por un vector $A \cdot b = y$ también de 10^6 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 diferente número de tareas igual $2, 4, 8, ..,16$. Cada tarea necesitará saber de cuántas filas es responsable. El programa principal activará una co-rutina donde creará las hebras 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)`

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`

2. Procedimiento y Plazo de Entrega

Se ha habilitado una tarea de subida en *Moodle* que le permite subir cada fichero que forma parte de los productos de la práctica de forma individual en el formato original. Para ello, suba el primer fichero de la forma habitual, y luego siga la secuencia de etapas que el propio *Moodle* le irá marcando. Recuerde además que:

- Los documentos escritos que no sean ficheros de código deben generarse **obligatoriamente** utilizando Latex, a través del editor *OverLeaf*, disponible en la nube. Tiene a su disposición en el Campus Virtual un manual que le permitirá desarrollar de forma sencilla y eficiente documentos científicos de alta calidad. Puede encontar el citado manual en la sección dedicada a Latex en el bloque principal del curso virtual. El url de *OverLeaf* es: `https://www.overleaf.com/`

- No debe hacer intentos de subida de borradores, versiones de prueba o esquemas de las soluciones. *Moodle* únicamente le permitirá la subida de los ficheros por **una sola vez**.
- La detección de plagio (copia) en los ficheros de las prácticas, o la subida de ficheros vacíos de contenido o cuyo contenido no responda a lo pedido con una extensión mínima razonable, invalidará plenamente la asignación, sin perjuicio de otras acciones disciplinarias que pudieran corresponder.
- El plazo de entrega de la práctica se encuentra fijado en la tarea de subida del Campus Virtual.
- Entregas fuera de este plazo adicional no serán admitidas, salvo causa de fuerza mayor debidamente justificadas mediante documento escrito.
- Se recuerda que la entrega de todas las asignaciones de prácticas es recomendable, tanto un para un correcto seguimiento de la asignatura, como para la evaluación final de prácticas, donde puede ayudar a superar esta según lo establecido en la ficha de la asignatura.
- Se recuerda que la entrega de todas las asignaciones de prácticas es obligatoria, y requisito indispensable para poder superar las prácticas de la asignatura, y por tanto a la asignatura en sí misma.