

Programación Concurrente y de Tiempo Real

Grado en Ingeniería Informática

Asignación de Prácticas Número 5

En la asignación número tres desarrolló soluciones a los problemas planteados utilizando paralelismo de datos con división manual del espacio de datos. En esta asignación utilizará paralelismo de datos particionando un problema y definiendo el número de hilos necesarios para su resolución en función de ecuaciones de balanceado de carga. También aprenderá el concepto de *speed up* como medida de la aceleración que una solución paralela ofrece. **Documente todo su código con etiquetas (será sometido a análisis con javadoc).**

1. Ejercicios

1. En la carpeta de la práctica dispone de código para realizar de forma paralela la búsqueda de números primos en un rango de naturales dados, y para descargar paralelamente diversas páginas web con escritura de su código `html` a disco; son ejemplos de tareas con diferentes coeficientes de bloqueo. Descárguelos. Observe que ambos utilizan la ecuación de *Subramanian*:

$$N_t = \frac{N_{nld}}{1 - C_b}$$

para determinar el número de hilos necesarios en función del número de núcleos lógicos disponibles y del coeficiente de bloqueo del problema. Experimente con ambos códigos, y redacte utilizando \LaTeX sobre *OverLeaf* un documento llamado `analisis.pdf` donde deberá distinguir para cada problema su tipología, el tipo de solución y cómo la ecuación anterior define el número de hilos paralelos que se deben tener. En particular, para el programa de descarga de páginas web se le pide que estime de forma aproximada el coeficiente de bloqueo mediante un análisis empírico de rendimiento.

2. En la asignación número tres escribió una solución secuencial y otra paralela con división manual del espacio de datos, para el producto matriz-vector. Adapte ambas versiones para multiplicar matrices cuadradas de números enteros de $m \times m$ componentes utilizando la ecuación de Subramanian para decidir cuántas tareas necesita, dividiendo el espacio de datos de forma automatizada. Soporte las tareas mediante la interfaz `Runnable` y procéselas con un ejecutor de capacidad fija. Los códigos se guardarán en `prodMatricesSecuencial.java` y `prodMatricesParalelo.java`. NOTA: m debe ser suficientemente grande (del orden de varios miles).

3. El *speedup* se define por la siguiente ecuación

$$SpeedUp = \frac{T(1)}{T(n)}$$

donde $T(1)$ es el tiempo de cálculo de la mejor solución secuencial a un problema, y $T(n)$ el tiempo de cálculo de una solución paralela de n tareas o hebras que se desea estudiar. Normalmente debe ser mayor que uno y menor que el número de *cores* disponibles. A partir de los programas que ha desarrollado en el anterior apartado, elabore una curva $speedup = f(n)$ siendo n el número de tareas que irá creciendo (1, 2, 4, ..., 16). Guarde las gráficas y sus comentarios en un documento `analisisProdMatricial.pdf` que deberá escribir utilizando \LaTeX sobre *OverLeaf*.

4. Un área habitual de uso de la programación paralela es el tratamiento digital de imágenes, que aplica operadores de tipo diferencial o integral discretos sobre cada uno de los píxeles que la forman. En particular, el operador de resaltado se puede definir -entre otras formas- a partir de la siguiente ecuación:

$$x_{i,j} = (4x_{i,j} - x_{i+1,j} - x_{i,j+1} - x_{i-1,j} - x_{i,j-1})/8$$

Suponga que abstraer su imagen mediante una matriz de $k \times k$ píxeles con 256 niveles de gris (0-255) que inicialmente estarán determinados de forma aleatoria para cada pixel (si quiere probarlo con una imagen real, en la carpeta de la práctica dispone de los recursos para ello). Se pide:

- Escribir un programa de resaltado mediante un algoritmo secuencial, en `resImagen.java`.
- Escribir un programa de resaltado mediante un algoritmo paralelo en `resImagenPar.java`. La versión paralela debería utilizar una matriz de entrada y otra matriz de salida.

A partir de los programas anteriores, elabore una gráfica $speedup = f(n)$ siendo n el número de tareas. Guarde las gráficas y sus comentarios en un documento `analisisOpResultado.pdf` que deberá escribir utilizando \LaTeX sobre *OverLeaf*. NOTA: k debe ser suficientemente grande (del orden de varios miles), y necesitará aplicar una condición de frontera, que tratándose de una imagen, debería ser nula.