UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Máster Universitario en Computación en la Nube y de Altas Prestaciones

Tecnología de la Programación Paralela (TPP).

Ejercicios OpenMP Sesión 4

Ejercicio 1: Paralelización de Cholesky basada en tareas

Este ejercicio se compone de dos versiones paralelas del algoritmo de Cholesky con OpenMP utilizando un **diseño basado en tareas**. Las tareas pueden ser identificadas con las llamadas a las funciones de BLAS/LAPACK.

Version 1 (sin dependencias): Esta versión consiste en identificar aquellas tareas que pueden ser ejecutadas concurrentemente y crearlas mediante la directiva `task`. Hay que observar cuidadosamente si es necesario sincronizar las tareas o no (`taskwait`). Se trata de utilizar una versión de paralelización de bucles equivalente al que se obtiene con la directiva `for` pero con tareas.

Versión 2 (con dependencias): Esta versión trata de la utilización de la cláusula `depend`.

Además de la implemntación anterior es conveniente realizar una comparativa entre las tres versiones del algoritmo paralelo de Cholesky:

- diseño basado en hilos.
- diseño basado en tareas (sin dependencias),
- diseño basado en tareas con dependencias.

Ejercicio 2: Paralelización de un problema de optimización

El problema que se va a paralelizar en esta actividad está implementado en secuencial en el fichero `mimo.c`. Se trata de un problema de optimización en el cuál se ha de buscar el elemento que hace mínima una expresión matemática entre un conjunto de elementos. La búsqueda del valor genera un árbol de posibles soluciones y su recorrido puede realizarse por el método de *Backtracking*, es decir, utilizando la misma técnica que la utilizada en el problema del Sudoku solo que, en este caso, hay que recorrer todo el árbol para encontrar la solución. En este problema se puede utilizar "poda" para reducir la expansión del árbol en exceso. El algoritmo facilitado utiliza esta técnica.

El problema matemático que resuelve este algoritmo sirve para resolver un problema de comunicaciones compuesto por múltiples entradas y múltiples salidas (MIMO – *Multiple Input Multiple Output*). En el anexo de este boletín figura la descripción por si se desea consultar por curiosidad, ya que no es necesario conocer esta información para resolver este problema.

La tarea a realizar consiste en implementar una versión paralela en OpenMP de este problema. Se realizará un estudio de tiempo de ejecución variando el tamaño del problema y el tamaño del conjunto de posibles valores con respecto a la cantidad de hilos que pueden ser utilizando. NOTA: El coste de este problema crece muy rápidamente por lo que hay que llevar cuidado y seleccionar tamaños de problema crecientes y no excesivamente grandes.

Algoritmo 1 Algoritmo para resolver la Eq. 1 (*Fuerza bruta*).

```
1: for all x ∈ I^n do
2:     m ← ||Rx − b||_2
3:     if m < min then
4:         min ← m
5:     end if
6: end for
```

A. Definición del problema de optimización MIMO

El problema de optimización MIMO (*Multiple Input Multiple Output*) se define como el siguiente problema de Álgebra Lineal Numérica que consiste en encontrar un valor de $x \in \mathrm{I}^n$ tal que

$$\min_x ||Rx - b||_2 \ . \tag{1}$$

En el problema que vamos a tratar la matriz $R \in \mathbb{R}^{n \times n}$ es triangular superior, $b \in \mathbb{R}^n$ es un vector independiente de números reales, y $x \in \mathrm{I}^n$ es el vector solución. Los elementos del vector solución pertenecen a un conjunto discreto y finito de elementos $x_i \in \mathrm{I}$. Sea t un valor entero positivo y par, entonces este conjunto se puede definir, por ejemplo, de la siguiente manera:

$$I = \left\{ \ \tfrac{1-t}{2}, \quad \tfrac{3-t}{2}, \quad \tfrac{5-t}{2}, \quad \ldots \quad \tfrac{t-3}{2}, \quad \tfrac{t-1}{2} \ \right\} \ . \tag{2}$$

El cardinal de este conjunto es t. Ejemplos son

$$I_{t=6} = \left\{ \ -\tfrac{5}{2}, \quad -\tfrac{3}{2}, \quad -\tfrac{1}{2}, \quad \tfrac{1}{2}, \quad \tfrac{3}{2}, \quad \tfrac{5}{2} \ \right\} \ ,$$

y

$$I_{t=8} = \left\{ \ -\tfrac{7}{2}, \quad -\tfrac{5}{2}, \quad -\tfrac{3}{2}, \quad -\tfrac{1}{2}, \quad \tfrac{1}{2}, \quad \tfrac{3}{2}, \quad \tfrac{5}{2}, \quad \tfrac{7}{2} \ \right\} \ .$$

Este problema aparece en sistemas MIMO (Multiple Input Multiple Output) de análisis digital de señales, de ahí su nombre.

Como es sabido, la 2-norma de un vector y de n elementos se expresa así $||y||_2$ y se calcula como

$$||y||_2 = \sqrt{y^T \cdot y} = \sqrt{\sum_{i=0}^{n-1} y_i^2} \ , \tag{3}$$

por lo tanto, para obtener el valor de $||Rx - b||_2$, se realiza el producto matriz-vector Rx, se resta el vector b y se calcula la norma según la Eq. 3. La 2-norma de un vector se puede calcular utilizando la función de BLAS `dnrm2`. Un ejemplo de la expresión a minimizar (Eq. 1) para una matriz R de 4×4 es

$$\min_x \left\| \begin{pmatrix} r_{00} & r_{01} & r_{02} & r_{03} \\ & r_{11} & r_{12} & r_{13} \\ & & r_{22} & r_{23} \\ & & & r_{33} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} - \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} \right\|_2 \ . \tag{4}$$

El método básico para obtener la solución x es un método de prueba y error por fuerza bruta. Este método consistiría en probar con todos los elementos del conjunto I^n y quedarse con el que minimiza la Eq. 1. El Algoritmo 1 resume este método.

El Algoritmo 1 es sencillo pero tiene un coste de $O(t^n)$ flops dado que t^n es el número de elementos del conjunto I^n. Este coste puede hacer inviable el algoritmo para valores no demasiado grandes de t y/o n.

Algoritmo eficiente basado en *Ramificación y Poda*

El problema planteado en la Eq. 1 es un problema de optimización en el cuál se busca una solución dentro de un conjunto dado de posibles soluciones. Esta búsqueda puede realizarse en árbol siguiendo ciertos criterios. El recorrido de este árbol puede hacerse de diferentes maneras y, dado que la extensión del mismo es grande, se pueden aplicar técnicas de *Ramificación y Poda* con objeto de limitar el

recorrido exhaustivo de todo el árbol. De cara a utilizar una técnica de *Ramificación y Poda*, existe una propiedad/metodología que puede ser útil ya que permite "cortar" la exploración de alguna rama en concreto.

Retomemos el ejemplo de la matriz de tamaño 4×4. Si x es un vector que minimiza la Eq. 4, entonces también minimiza su cuadrado. Además, operando un poco tenemos que la Eq. 4 es igual a

$$\min_x \left(\left\| \begin{pmatrix} r_{00} & r_{01} & r_{02} & r_{03} \\ & r_{11} & r_{12} & r_{13} \\ & & r_{22} & r_{23} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} - \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} \right\|_2^2 + \|r_{33}x_3 - b_3\|_2^2 \right) , \tag{5}$$

que, a su vez, es igual a

$$\min_x \left(\left\| \begin{pmatrix} r_{00} & r_{01} & r_{02} \\ & r_{11} & r_{12} \\ & & r_{22} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} \bar{b}_0 \\ \bar{b}_1 \\ \bar{b}_2 \end{pmatrix} \right\|_2^2 + \|r_{33}x_3 - b_3\|_2^2 \right) , \tag{6}$$

donde

$$\begin{pmatrix} \bar{b}_0 \\ \bar{b}_1 \\ \bar{b}_2 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} - x_3 \begin{pmatrix} r_{03} \\ r_{13} \\ r_{23} \end{pmatrix} .$$

La propiedad anterior se puede utilizar de la siguiente manera. Supongamos que elegimos un posible valor para x_3, entonces, podemos calcular el segundo término de la Eq. 5, es decir, $\|r_{33}x_3 - b_3\|_2^2$. Si este valor es mayor que un cierto valor o cota que tenemos seleccionado como mínimo actual, es seguro que no encontraremos ninguna solución por ese camino y, por tanto, ya se pueden descartar todas las soluciones con dicho x_3. En caso contrario, podemos seguir con la tesis de que la solución final puede tener como x_3 el valor elegido y continuar buscando. Para ello, utilizaremos el primer término de la Eq. 6 y probaremos con valores tentativos para x_2, y de manera recursiva, llegaremos hasta la elección del x_0.