PROGRAMACIÓN PARALELA Y COMPUTACIÓN DE ALTAS PRESTACIONES

TRABAJO AUTÓNOMO: ENTORNOS DE PROGRAMACIÓN PARALELA

PRÁCTICAS DE PPCAP 17/18

José Luis Cánovas Sánchez

Algoritmos Matriciales Paralelos

# CUESTIÓN 4

Incluir paralelización de bucles (en los bucles más externos) en una versión de la multiplicación de matrices por bloques y comparar los tiempos de ejecución con la versión por bloques no paralela y con la versión paralela en la que se paraleliza el trabajo dentro de cada bloque.

En el directorio 4/ adjunto, o en mpp17-45/algmatpar/4/ de luna, se encuentra el código fuente de las versiones paralelizadas en los bucles externos e internos. Comparamos con la tabla de tiempos (segundos) del código secuencial por bloques ejecutado en marte:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algoritmo | N=500 | N=1000 | N=2000 | N=3000 | N=4000 |
| **Secuencial** |  |  |  |  |  |
| Tamaño de bloque: 25 | **0.177865** | 1.453277 | 11.751641 | 39.685550 | 93.973494 |
| Tamaño de bloque: 50 | 0.180716 | **1.361554** | **10.562337** | **35.292006** | **83.937251** |
| Tamaño de bloque: 100 | 0.188849 | 1.376489 | 10.777013 | 36.056393 | 84.532981 |
| **Paralelización bucle interno** |  |  |  |  |  |
| Tamaño de bloque: 25 | 0.074525 | 0.567204 | 4.449876 | 14.790170 | 35.241169 |
| Tamaño de bloque: 50 | 0.053506 | 0.350413 | 2.735158 | 9.259358 | 21.996043 |
| Tamaño de bloque: 100 | **0.047331** | **0.315868** | **2.395529** | **8.036172** | **18.861149** |
| **Paralelización bucle externo** |  |  |  |  |  |
| Tamaño de bloque: 25 | **0.049491** | **0.303321** | 2.387386 | 7.748638 | 18.723906 |
| Tamaño de bloque: 50 | 0.051893 | 0.329063 | **2.177237** | **6.901120** | **16.996734** |
| Tamaño de bloque: 100 | 0.054400 | 0.328252 | 2.454313 | 7.003805 | 17.380692 |

Se utiliza la función omp\_get\_max\_num\_threads() para asignar el máximo número de threads admitidos por marte. El detalle más importante a tener en cuenta es reservar un array de matrices s, donde se acumularán las multiplicaciones de matrices con suma dentro de cada hilo, para que no interfieran entre sí. Vemos que en todos los casos mejoramos más del doble al mejor tiempo secuencial.

Entre las versiones paralelizadas, a mayor tamaño de las matrices se observa la ventaja de paralelizar los bucles externos, donde repartiendo más trabajo de cómputo entre los hilos aprovechamos la división en bloques en las múltiples cachés de los cores, así como evitar el overhead de fork-join de hilos en cada multiplicación.

# CUESTIÓN 14

Comparar el comportamiento de distintas versiones de la multiplicación de matrices con OpenMP en forma offload en venus. Experimentar con schedule, collapse y simd los tiempos de las multiplicaciones matriciales.

En el directorio 14/ adjunto, o en mpp17-45/algmatpar/14/ de luna, se encuentra el código fuente de la multiplicación de matrices paralela utilizando la transpuesta de b. Se descarga el trabajo de la multiplicación al Xeon Phi de Venus con la línea:

#pragma offload target(mic) in(a:length(fa\*ca)) in(bt:length(fb\*cb)) inout(c:length(fc\*cc))

Dentro del coprocesador, paralelizamos con OpenMP, probando con distintas cláusulas. Los resultados (en segundos) se muestran abajo:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tamaño de Matriz = | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 |
| Sin cláusulas extra | 2.921431 | 4.335710 | **6.903049** | 12.213944 | 19.344727 | 30.088533 |
| Static | 2.917755 | 4.537175 | 6.978585 | 12.350229 | 18.973087 | 29.916222 |
| Static 5 | 2.977286 | 4.357692 | 7.292801 | 12.337511 | 19.444749 | 29.858921 |
| Dynamic | 2.939719 | 4.299595 | 7.069046 | 11.723230 | 18.649555 | **29.300287** |
| Collapse | **2.744677** | **3.840926** | 6.963857 | **11.362626** | **18.400358** | 29.407477 |
| Simd | 3.594959 | 6.200148 | 17.879717 | 20.294112 | 30.960325 | 70.273822 |
| Simd 2 | 2.897114 | 4.431727 | 6.941834 | 12.246327 | 19.320921 | 30.047152 |

En la tabla de arriba, simd y simd 2 hacen referencia al bucle donde se añade la vectorización. El primero corresponde al bucle que recorre las columnas de b (filas de bt), y simd 2 hace referencia al bucle del índice k donde se hacen las operaciones de multiplicación y suma.

Los mejores tiempos los obtiene en general la versión que utiliza la cláusula collapse, seguido de cerca del resto de versiones.