PROGRAMACIÓN PARALELA Y COMPUTACIÓN DE ALTAS PRESTACIONES

TRABAJO AUTÓNOMO: ENTORNOS DE PROGRAMACIÓN PARALELA

PRÁCTICAS DE PPCAP 17/18

José Luis Cánovas Sánchez

Librerías (MKL)

# CUESTIÓN 1

Comparar prestaciones de multiplicación secuencial por bloques y sin bloques con la multiplicación de MKL.

En el directorio 1/ adjunto, o en mpp17-45/mkl/matriz\_matriz/ de luna, se encuentra el código fuente de las versiones con y sin bloques, y MKL. El código de las 2 primeras es el usado en las otras prácticas, siendo el sin bloques la versión con transpuesta que dio los mejores tiempos. El código de MKL utiliza la llamada a BLAS:

cblas\_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, fa, cb, ca, alpha, a, lda, b, ldb, beta, c, ldc);

Donde se indica que se ordena la matriz en memoria por filas, que no es la transpuesta, se pasan las dimensiones, y las constantes alpha y beta que indican en este caso, alpha = 1 y beta = 0.

https://software.intel.com/sites/default/files/did_feeds_images/A4D3381E-A201-4EF9-A829-5E1190266252/A4D3381E-A201-4EF9-A829-5E1190266252-imageId=0C7ED481-3A43-482C-8022-DAFD9655243B.gif

En una primera prueba utilicé gcc para las versiones sin MKL, y no indiqué explícitamente que se usara un único hilo en MKL. Por referencia, los tiempos tomados en Saturno de esa primera ejecución son:

|  |  |  |  |
| --- | --- | --- | --- |
| N = | 2000 | 3000 | 4000 |
| Sin bloques (trans) | 22.665726 | 75.494501 | 177.077932 |
| Con bloques |  |  |  |
| TB=25 | 13.496091 | 42.318879 | 100.471517 |
| TB=50 | *12.777352* | *40.772234* | *95.693469* |
| TB=100 | 14.281152 | 45.684251 | 112.802174 |
| MKL | **0.878072** | **1.149646** | **2.310619** |

Vemos que en la versión de bloques, con tamaño de bloque 50 se obtiene el mejor tiempo, pero frente a la implementación de MKL de la multiplicación de matrices BLAS, ésta última llega a ser ~40 veces más rápida que la versión por bloques.

Una vez recompilo con icc todas las versiones, y pongo explícitamente las líneas

1. mkl\_set\_dynamic( 0 );
2. mkl\_set\_num\_threads( 1 );

para forzar el uso de un solo hilo en MKL, tomo los tiempos en Júpiter, pues Saturno estaba ocupado por la cola de investigación.

|  |  |  |  |
| --- | --- | --- | --- |
| N = | 2000 | 3000 | 4000 |
| Sin bloques (trans) | 6.292005 | 22.071910 | 52.246046 |
| Con bloques |  |  |  |
| TB=25 | 8.572358 | 28.787944 | 70.131146 |
| TB=50 | 6.983950 | 23.548761 | 56.315092 |
| TB=100 | 6.790730 | 22.259663 | 52.701967 |
| MKL | **1.293702** | **3.669136** | **7.389529** |

En este caso la diferencia de MKL con el mejor tiempo secuencial es *solo* hasta ~7 veces más rápido que las otras implementaciones secuenciales.

# CUESTIÓN 2

En la rutina de factorización LU por bloques, sustituir las multiplicaciones de matrices por llamadas a BLAS. Comparar tiempos de ejecución. Comparar los resultados con los de la factorización usando directamente MKL.

En el directorio 2/ adjunto, o en mpp17-45/mkl/lu/ de luna, se encuentra el código fuente de las versiones de LU por bloques con llamadas a BLAS de MKL en la multiplicación de matrices, y la LU de MKL.

En la primera versión, sustituimos la multiplicación con resta de matrices por el siguiente código, donde la alfa y beta corresponden a -1 y 1 respectivamente, para restar el producto sobre la matriz c:

https://software.intel.com/sites/default/files/did_feeds_images/A4D3381E-A201-4EF9-A829-5E1190266252/A4D3381E-A201-4EF9-A829-5E1190266252-imageId=0C7ED481-3A43-482C-8022-DAFD9655243B.gif

1. **void** multiplicar\_restar\_matrices(**double** \*a,**int** fa,**int** ca,**int** lda,**double** \*b,**int** fb,**int** cb,**int** ldb,**double** \*c,**int** fc,**int** cc,**int** ldc)
2. {
3. mkl\_set\_dynamic( 0 );
4. mkl\_set\_num\_threads( 1 );
5. **double** alpha = -1.0;
6. **double** beta = 1.0;
7. cblas\_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, fa, cb, ca, alpha, a, lda, b, ldb, beta, c, ldc);
8. }

En la versión de la LU de LAPACK en MKL, sólo hace falta llamar a la siguiente función, en vez de a nuestra función de LU por bloques. La variable ipiv es el array de cambios de filas, pues en verdad se hace la descomposición PA = LU, donde se cambian las filas de A para minimizar los errores al operar con valores pequeños.

LAPACKE\_dgetrf(LAPACK\_ROW\_MAJOR, fa, ca, a, lda, ipiv);

Comparamos con la tabla de tiempos (segundos) del código secuencial por bloques de referencia, la versión con MKL para el producto de matrices, y la versión de MKL. Usamos bloques de tamaño 10, pues en pruebas anteriores son los que mejor tiempo obtenían. Todos compilados con icc. Ejecutado en Júpiter:

|  |  |  |  |
| --- | --- | --- | --- |
| N= | 5000 | 7000 | 9000 |
| LU BLOQUES 10 | 61.293587 | 168.515420 | 359.437214 |
| LU BLOQUES 10 + MKL | 6.558937 | 17.514004 | 36.840648 |
| LU MKL | **5.908207** | **12.960004** | **27.409984** |

Vemos cómo el mayor tiempo de cómputo en la LU por bloques es la multiplicación de matrices. En la versión LU en que se utiliza MKL para las multiplicaciones, vemos una mejora de ~10 veces en el tiempo respecto a la versión de referencia. Esto concuerda con la gran mejora que obteníamos en el ejercicio anterior al usar MKL. Y al usar la versión de MKL de la LU de LAPACK mejoramos a la LU por bloques en varios segundos.

# CUESTIÓN 3

Realizar dos de los siguientes: experimentos con LU con paralelismo OpenMP+MKL, con LU con paralelismo MPI+MKL, LU combinando MKL en CPU con GPU, LU combinando MKL en CPU con Xeon Phi, experimentos de LU con PARDISO.

En el directorio 3/ adjunto, o en mpp17-45/mkl/lu\_mkl\_openmp/ de luna, se encuentra el código fuente de las paralelizaciones.

Las versiones programadas son:

* LU con bloques, MKL en producto de matrices con 24 hilos, OpenMP con 24 hilos en la resolución de los sistemas.
* LU de LAPACK MKL con 24 hilos.
* LU con bloques, MKL en producto de matrices en offload, OpenMP en offload en la resolución de los sistemas.
* LU de LAPACK MKL en offload.

Los tiempos en segundos son los siguientes:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Vemos que los tiempos con 24 hilos y 24 nodos son muy parecidos, pero a pesar de haber tiempo de paso de