# I Concurso de programación paralela: Problema D

Víctor Moreno Martínez, Diego Sánchez Román, Germán Retamosa de Agreda

victor.moreno@uam.es, d.sanchez@uam.es, german.retamosa@uam.es

Escuela Politécnica Superior, Universidad Autónoma de Madrid

## Resumen

En este documento se presenta la metodología utilizada para resolver el problema D presentado en el I Concurso de Programación Paralela celebrado el 8 de Septiembre de 2011 en La Laguna. Este problema consistía en la *Multiplicación de matrices densas*.

## 1. Enunciado del problema

Dadas cuatro matrices cuadradas densas se quiere obtener su producto.

## 2. Resolución del problema

A lo largo de esta sección se utliza la siguiente notación:

- $T$ = tamaño de las matrices (cuadradas) a multiplicar
- $N_n$ = número de nodos en el clúster
- $N_c$ = número de cores en cada nodo del clúster (en este caso se trata de un sistema homogéneo, por lo que todos los nodos tienen las mismas características)

### 2.1. Distribución de carga entre los nodos del *clúster*

Para la realización de la multiplicación de las cuatro matrices en el sistema se ha utilizado una estructura de árbol binario, de modo que se multiplican cada una de las dos matrices originales por parejas, generando resultados intermedios, que posteriormente serán multiplicados entre sí para obtener el resultado final.

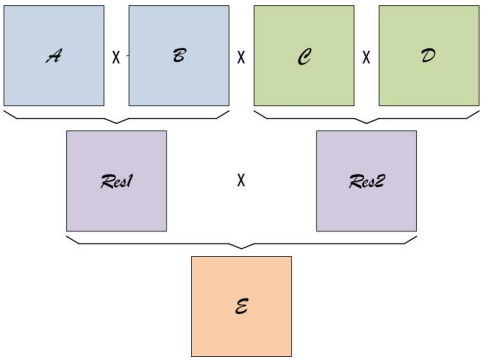


Figura 1: Estructura en árbol binario del problema

En primer lugar se ha hecho una distribución del trabajo entre los nodos del sistema en términos de las filas de la matriz A, de modo que a cada uno de estos nodos se les enviarán $T/N_n$ filas de la matriz A. La matriz B será enviada por completo a cada uno de los nodos del sistema. De este modo, cada nodo generará una submatriz de la matriz C resultante (ver Fig. 2).

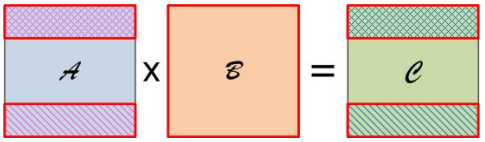


Figura 2: Reparto de trabajo por filas de la matriz A

Para hacer el reparto de trabajo entre los nodos del sistema se ha utilizado el mecanismo de paso de mensajes basado, MPI [1].

Una vez que se ha repartido el trabajo a realizar entre los nodos del sistema, se procede

a repartir el trabajo que debe llevar a cabo cada uno de los *cores* de cada nodo. Para llevar a cabo esta labor, se ha ujtilizado OpenMP [2].

**Nota:** El cluster donde se realiza el concurso consta de cuatro nodos, cada uno con 8 cores, del equipo Arabí del Centro de Supercomputación de la Fundación Parque Científico de Murcia.

### 2.2. Optimización de los accesos a memoria

Aparte de las mejoras en términos de paralelización del trabajo a realizar, también se llevó a cabo una optimización relativa al patrón de acceso de los datos en la memoria.

La forma más común para almacenar los datos de una matriz en memoria es hacerlo por filas. Como se puede observar en la Fig. 3 esto favorece el acceso a los datos cuando se hace a lo largo de una misma fila, ya que al acceder al primero de los datos, los datos de la misma fila (o parte de ellos) se traslandan a la caché en cuanto se pide el primero de ellos, permitiendo de este modo maximizar el rendimiento de acceso a los datos. Esto es justo lo que queremos para acceder a la matriz A en el proceso de multiplicación (que recordamos es "filas de A"por çolumnas de B".

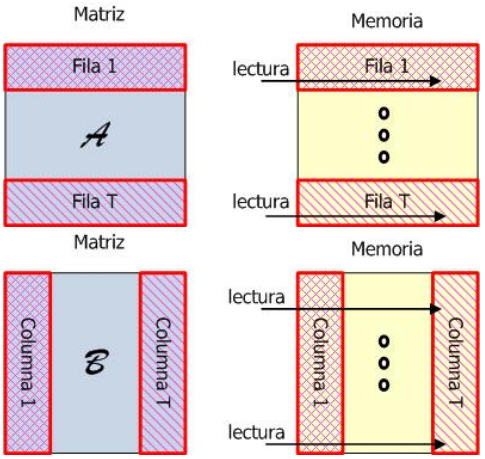


Figura 3: Almacenamiento de matrices por filas

No obstante, para el acceso a los datos de la matriz B en el proceso de la multiplicación de matrices, que se realiza por columnas, esto supone una gran pérdida de rendimiento, ya que cada acceso a cada dato (o a casi todos) contiguo en la misma columna provocará un fallo de caché. Es por esa razón que para la matriz B ha de ser almacenada por columnas (ver FIg. 4) si se quiere optimizar el acceso a sus datos.

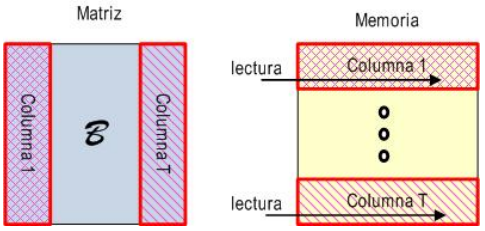


Figura 4: Almacenamiento de una matriz por columnas

De este modo, para nuestra problema nos interesa almacenar la matriz A por filas y la matriz B por columnas. Eso sí, este cambio implica hacer un pequeño ajuste en los bucles que recorren las matrices a la hora de la multiplicación.

En la tabla 1 puede observarse el impacto que tiene la utilización de la "multiplicación de matrices tradicional.º la utilización de la"multiplicación por la traspuesta.en la ejecución realizada **sobre un único core** en un procesador *Intel® Core™ i7-2600* con cuatro *cores* a 3,4*Ghz* y 8MB de caché .

Un análisis más exhaustivo sobre este fenómeno puede obtenerse utilizando herramientas de *profiling* que aportan información sobre el uso de la caché, como la herramienta *Cachegrind* [3].

### 2.3. Posibles mejoras

Como posibles mejoras a la solución presentada a lo largo de este documento se enmarcarían la utilización de algoritmos más óptimos para la repartición del trabajo, como los enunciados en [4]. Otra posible mejora sería la de explotar la jerarquía en árbol binario que se

| Tamaño de la matriz | Versión | Tiempo de ejecución($\mu s$) | Speedup |
|---|---|---|---|
| 1024 | tradicional | 10064 | |
| 1024 | traspuesta | 3361 | 2.99 |
| 2048 | tradicional | 144840 | |
| 2048 | traspuesta | 25602 | 5.65 |

Cuadro 1: Impacto del patrón de acceso de datos en el rendimiento de la multiplicación de 2 matrices

ha desarrollado para hacer las multiplicaciones parciales de modo que en la primera etapa los dos resultados intermedios se generen de forma simultánea. Estas soluciones no pudieron ser implementadas a lo largo del concurso por motivos de limitación del tiempo disponible.

## Agradecimientos

Agradecemos en primer lugar a los miembros del comité organizador de esta I edición del Concurso de Programación Paralela por su esfuerzo y dedicación, y en segundo lugar los miembros del grupo de investigación en Redes y Computación de Altas Prestaciones [5] de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid por su apoyo de cara a la participación en este concurso.

## Referencias

[1] *The Message Passing Interface (MPI) standard*, http://www.mcs.anl.gov/research/projects/mpi/.

[2] *The official OpenMP website*, http://www.OpenMP.org.

[3] Valgrind Project, *Cachegrind: a cache and branch-prediction profiler*, http://valgrind.org/docs/manual/cg-manual.html.

[4] Foster, I., *Designing and Building Parallel Programs*, 4.6 Case Study: Matrix Multiplication, http://www.mcs.anl.gov/ itf/dbpp/text/node45.html.

[5] *High Performance Computing and Networking Group*, http://www.hpcn.es.