Multiplicación de Matrices: Clasica vs Bloques

Blas David Cruz Huaman
Escuela de Ciencia de la Computación
Universidad Catolica San Pablo
Email: blas.cruz@ucsp.edu.pe

April 1, 2021

GitHub : `https://github.com/blasdch18/Computacion_Paralela_-Dist.git`

Abstract

Se implementará los algoritmos de Multiplicación de matrices en forma clásica y modelo por Bloques, la dinamica es usar matrices de dimensiones como 100 x 100 hasta 800 x 800. Todo esto para el analisis de nuestro presente trabajo.

1 INTRODUCCION

Se realizará el analisis del Comportamiento de la Memoria Caché mediante el algoritmo de multiplicación de matrices clasico vs por Bloques. Para esto desarrollaremos ambos metodos que se utilizaron para realizar el siguiente trabajo.

2 MULTIPLICACION CLASICA

Solo se puede multiplicar dos matrices si sus dimensiones son compatibles, lo que significa que el número de columnas en la primera matriz es igual al número de filas en la segunda matriz.

El algoritmo clásico de multiplicación de matrices tiene una complejidad de $O(n^3)$. Se ha conjeturado que el algoritmo (clásico) óptimo debe alcanzar $O(n^2)$ en tiempo.

El algoritmo tiene $2n^3 = O(n^3)$ operaciones aritméticas (n elementos de la i- ésima fila de A multiplicada por la j- ésima columna de B) * (n columnas de B) * (n filas de A) * 2. 2 es tener en cuenta cada adición de C [i, j]) opera en 3 * n^2 palabras de memoria, para A, B y res cada una de n^2 tamaño

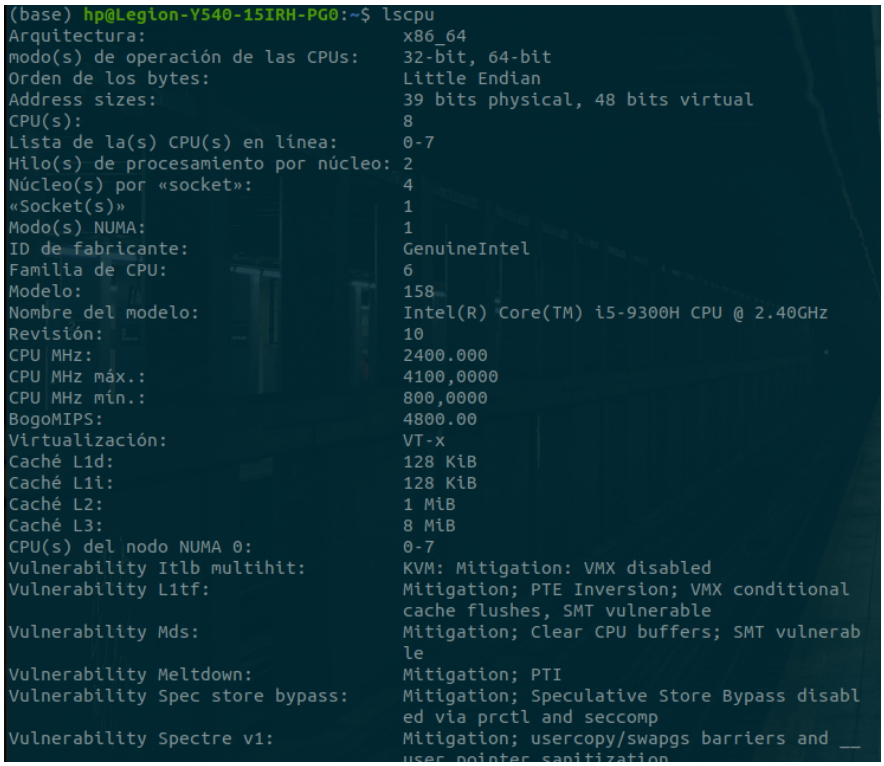

Figure 1: Caracteristicas del Computador

2.1 Algoritmo

```
Matrix res(height, width);
    // Lectura rapida en fila i de A
 for (int i=0 ; i<height ; i++)
    // Lectura rapida en res[i][j]
    // Lectura rapida en Col j de B
  for (int j=0 ; j<width ; j++)
   for (int k = 0; k < width; k++)
   // Escritura Lenta en res[i][j]
     res.array[i][j] += arrayA[i][k] * m.arrayB[k][j];
```

3 MULTIPLICACION POR BLOQUES

Al multiplicar dos matrices por bloques, hay que dividir cada una de las matrices en una retícula rectangular de submatrices. Si queremos calular C = A*B, de lo que nos tenemos que asegurar es de que la división de las n columnas de A coincida con la división de las n filas de B.

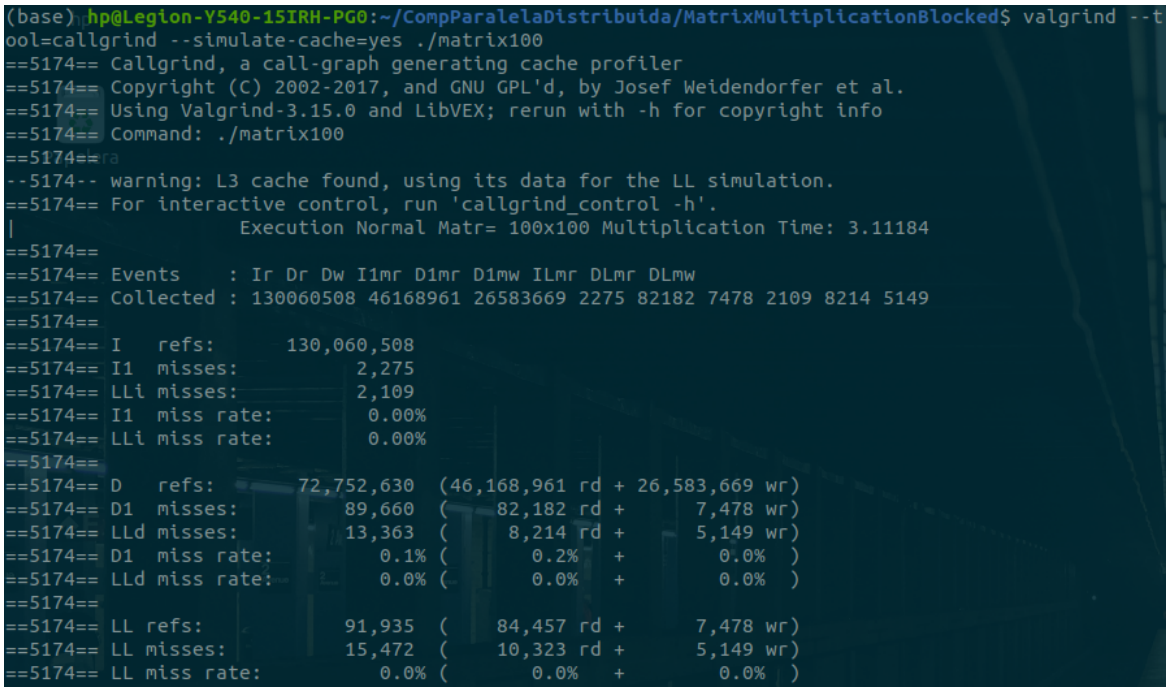

Figure 2: Cache Monitor Mult. Matriz 100x100

```
Matrix res (size,size);
    // Lectura Rapida de Bloque res[jj][kk]
 for (int jj=0; jj<size; jj+=block_size)
  for (int kk=0; kk<size; kk+=block_size)
    // Lectura Rapida de Bloque A[jj][kk]
    // Lectura Rapida de Bloque B[kk][i]
   for (int i=0; i<size; i++)
    for (int j=jj;j<((jj+block_size)>size ? size:(jj+block_size));j++)
        tmp = 0;
        for (int k=kk; k<((kk+block_size)>size ? size: (kk+block_size));k++)
            // Escritura Rapida de Bloque res[i][jj]
         tmp += A.array[i][k] * B.array[k][j];
          res.array[i][j] += tmp;
 return res;
```

3.1 Algoritmo

El algoritmo por bloques de multiplicación de matrices tiene una complejidad de $O(n^3)$ igual que la forma clasica; sin embargo en este caso se esta ejecutando un algoritmo paralelo ya que se divide cada bloque para asi ser operado, y asi nos da menor tiempo en sus resultados. Fig. 2

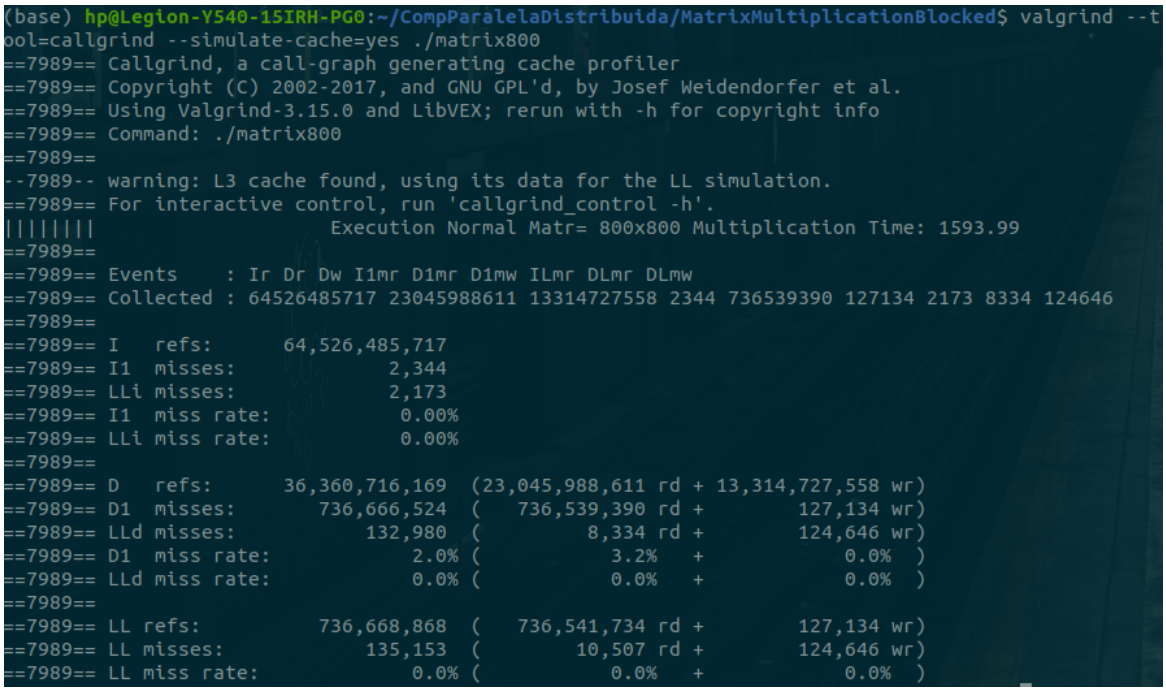

Figure 3: Cache Monitor Mult. Matriz 800x800

4 Conclusión

Este resultado es importante ya que muestra que podemos aumentar la intensidad computacional eligiendo un valor de tamaño grande para cada bloque y puede ser más rápido que la multiplicación clasica.

NxN Matriz	Tiempo	I1	LLi	D1	LLd
100	3.11184	2275	2109	89660	13363
200	24.6026	2305	2133	564758	19195
300	80.5403	2337	2153	2778571	28761
400	193.071	2345	2158	71565584	42080
500	378.123	2345	2158	180084582	59149
600	647.207	2341	2156	311244345	79968
700	1069.47	2341	2156	382855178	104536
800	1593.99	2344	2173	736666524	132980

Table 1: Multiplicación Clasica : Dimensión y Misses

NxN Matriz	I1 miss rate%	LLi miss rate%	D1 miss rate%	LLd miss rate%
100	0.00	0.00	0.1	0.0
200	0.00	0.00	0.1	0.0
300	0.00	0.00	0.1	0.0
400	0.00	0.00	1.6	0.0
500	0.00	0.00	2.0	0.0
600	0.00	0.00	2.0	0.0
700	0.00	0.00	1.6	0.0
800	0.00	0.00	2.0	0.0

Table 2: Multiplicación Clasica : Dimensión y Misses Rate

References

[1] Pacheco P.: An Introduction to Parallel Programming Ed. MK, San Francisco, USA 2011.

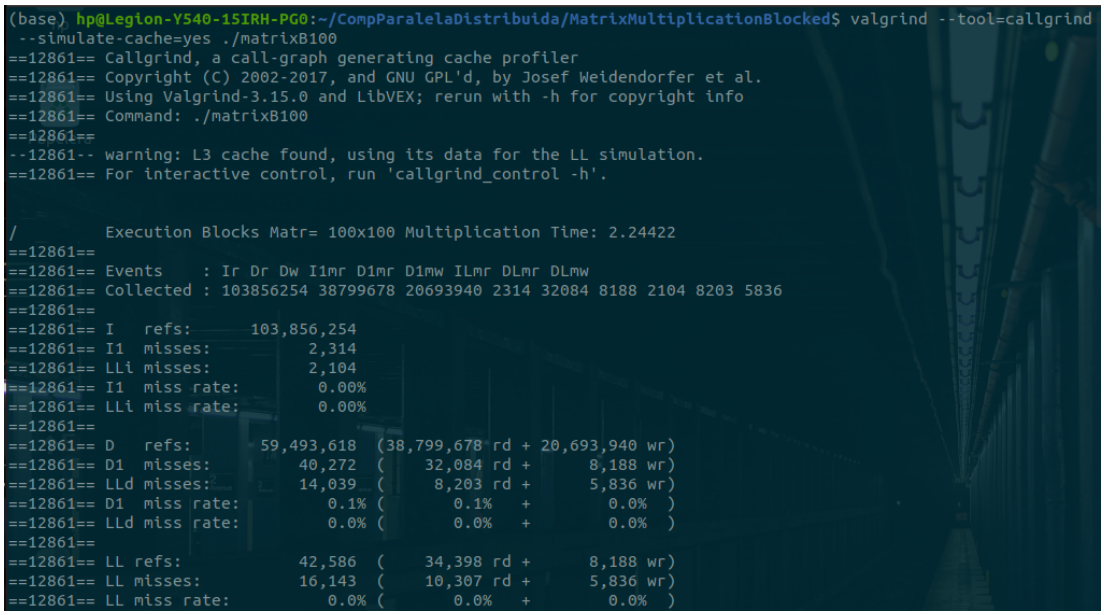

Figure 4: Cache Monitor Mult.Bloques 100x100

NxN Matriz	Tiempo	I1	LLi	D1	LLd
100	2.24422	2314	2104	40272	14039
200	19.0681	2355	2129	672678	21809
300	62.9421	2362	2135	2510103	34565
400	148.410	2371	2143	5929912	52322
500	286.986	2368	2140	11607945	75078
600	492.369	2368	2140	19733445	102834
700	771.523	2371	2197	30176106	135807
800	1152.33	2371	2280	46993632	254302

Table 3: Multiplicación Por Bloques : Dimensión y Misses

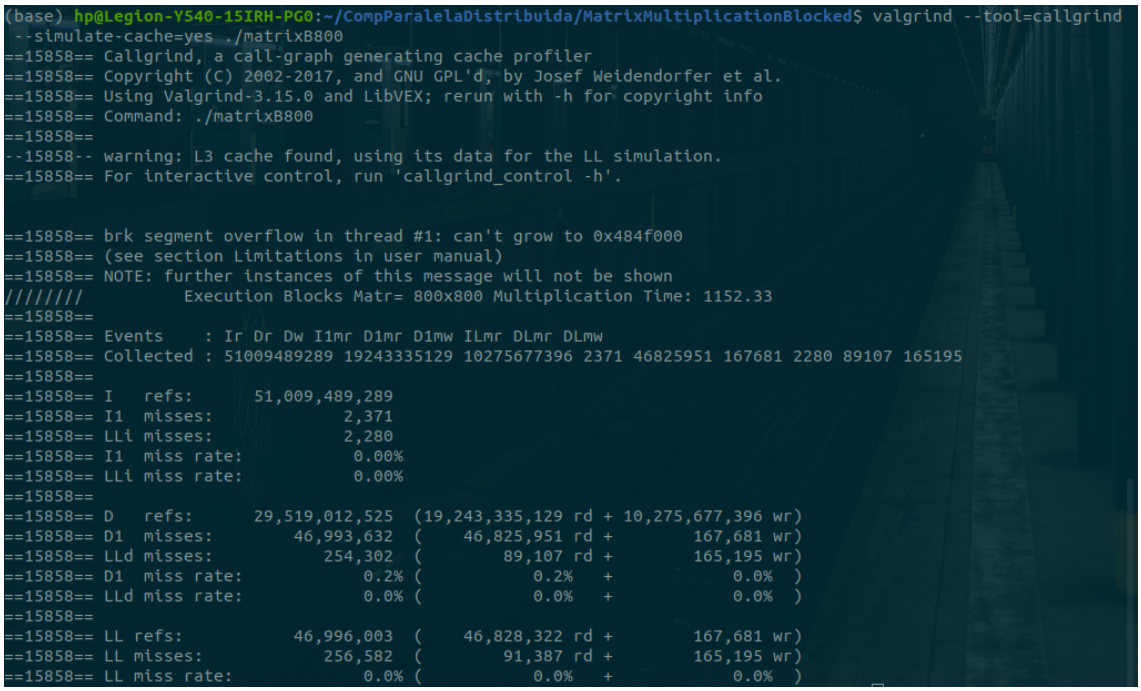

Figure 5: Cache Monitor Mult. Bloques 800x800

NxN Matriz	I1 miss rate%	LLi miss rate%	D1 miss rate%	LLd miss rate%
100	0.00	0.00	0.1	0.0
200	0.00	0.00	0.1	0.0
300	0.00	0.00	0.2	0.0
400	0.00	0.00	0.2	0.0
500	0.00	0.00	0.2	0.0
600	0.00	0.00	0.2	0.0
700	0.00	0.00	0.2	0.0
800	0.00	0.00	0.2	0.0

Table 4: Multiplicación Por Bloques : Dimensión y Misses Rate

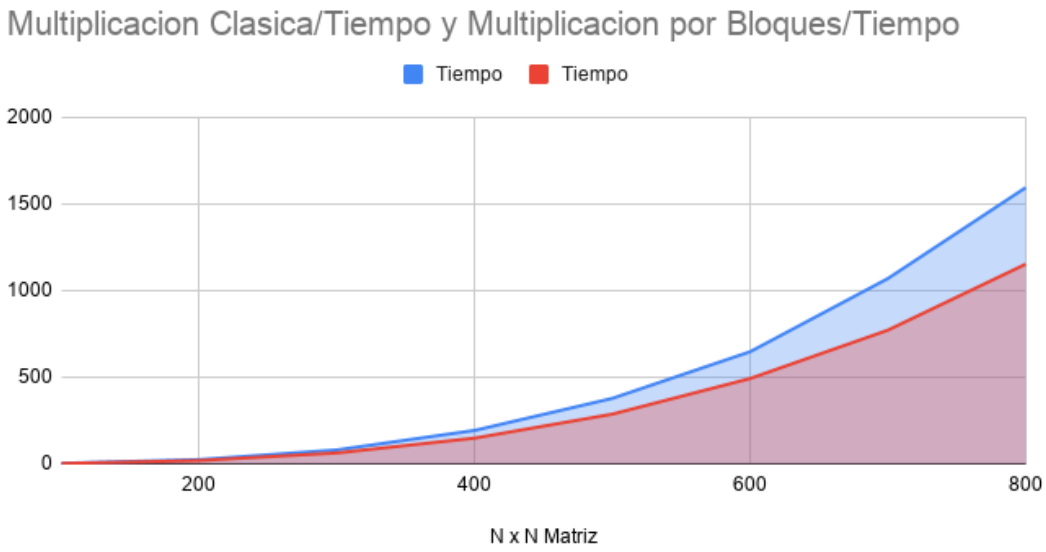

Figure 6: Ejecución de Resultados: Tiempos

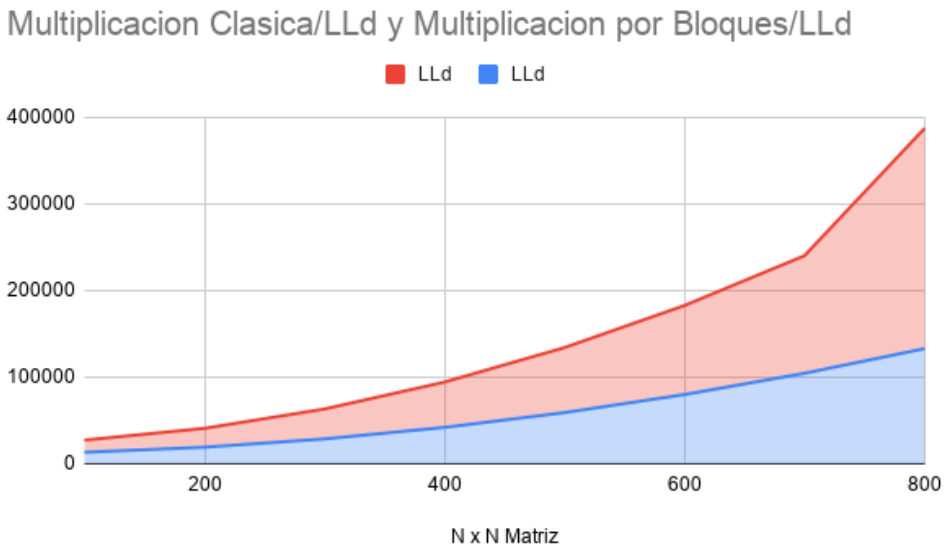

Figure 7: Ejecución de Resultados: Instruction Cache Misses

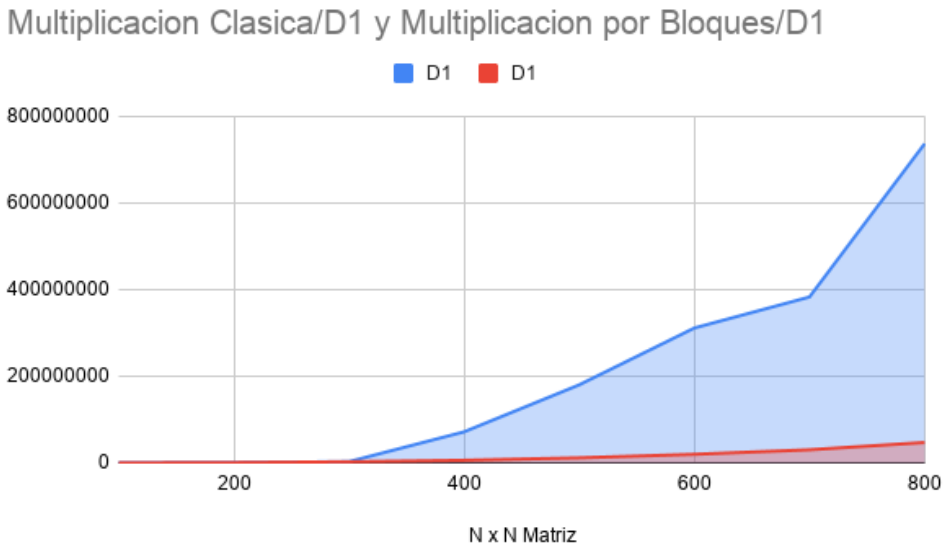

Figure 8: Ejecución de Resultados: Data Cache Misses