

Tarea tres -

Multiplicación distribuida de matrices.

Daniela Cortes Castillo

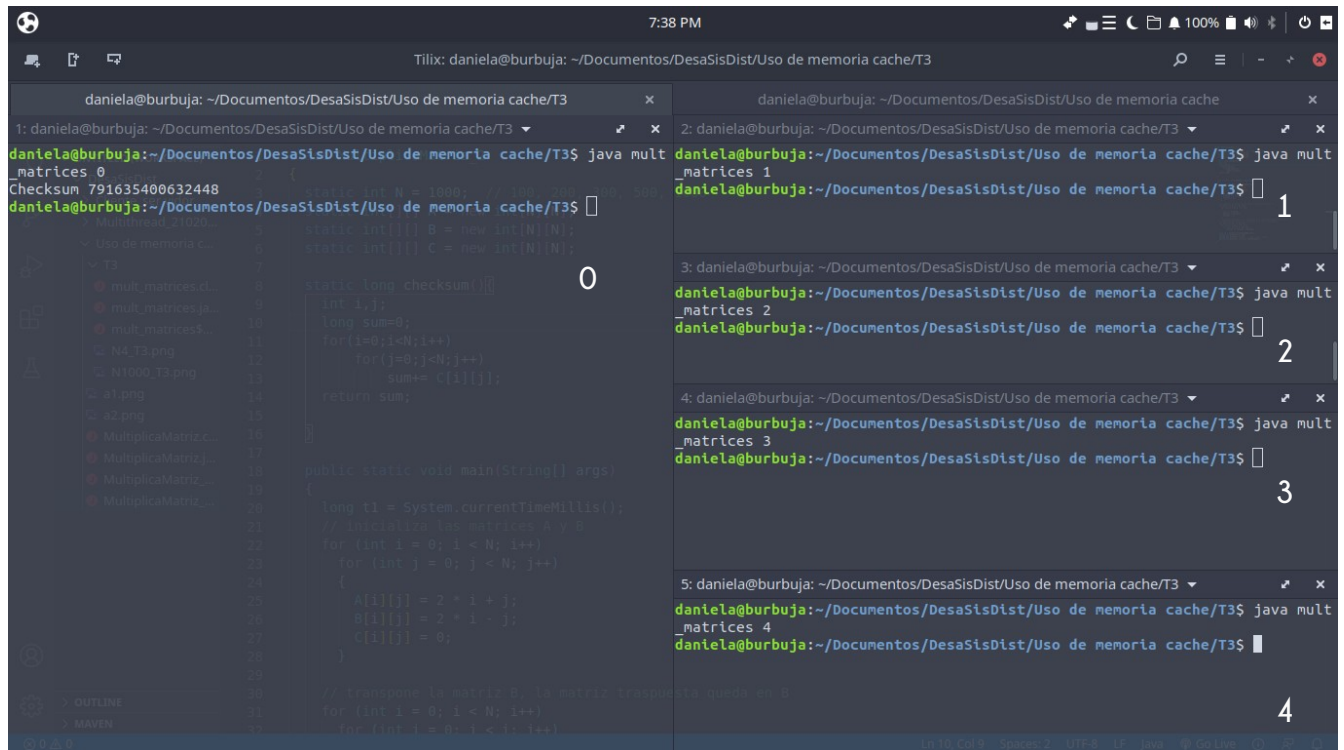
Prueba 1. Multiplicación de matrices de 4x4

```
1: daniela@burbuja: ~/Documentos/DesaSisDist/Usode memoria cache/T3$ javac mult_matrices.java
daniela@burbuja:~/Documentos/DesaSisDist/Usode memoria cache/T3$ java mult_matrices 0
Checksum 592
MATRIZ C
28 22 16 10
52 38 24 10
76 54 32 10
100 70 40 10
daniela@burbuja:~/Documentos/DesaSisDist/Usode memoria cache/T3$

2: daniela@burbuja: ~/Documentos/DesaSisDist/Usode memoria cache/T3$ java mult_matrices 2
3: daniela@burbuja: ~/Documentos/DesaSisDist/Usode memoria cache/T3$ java mult_matrices 4
4: daniela@burbuja: ~/Documentos/DesaSisDist/Usode memoria cache/T3$ java mult_matrices 1
5: daniela@burbuja: ~/Documentos/DesaSisDist/Usode memoria cache/T3$ java mult_matrices 3
```

En la captura de pantalla se puede observar la ejecución desde terminal de los 5 nodos. El número 0 es el que recibe los resultados de los demás e imprime el checksum de la matriz resultado C que en este caso es **592** y además como nuestra matriz es de 4X4 imprime el resultado.

Prueba 2. Multiplicación de matrices de 1000x1000



```
7:38 PM
Tilix: daniela@burbuja: ~/Documentos/DesaSisDist/Us... de memoria cache/T3

1: daniela@burbuja: ~/Documentos/DesaSisDist/Us... de memoria cache/T3
daniela@burbuja:~/Documentos/DesaSisDist/Us... de memoria cache/T3$ java mult_matrices 0
Checksum 791635400632448
daniela@burbuja:~/Documentos/DesaSisDist/Us... de memoria cache/T3$

2: daniela@burbuja: ~/Documentos/DesaSisDist/Us... de memoria cache/T3
daniela@burbuja:~/Documentos/DesaSisDist/Us... de memoria cache/T3$ java mult_matrices 1
daniela@burbuja:~/Documentos/DesaSisDist/Us... de memoria cache/T3$

3: daniela@burbuja: ~/Documentos/DesaSisDist/Us... de memoria cache/T3
daniela@burbuja:~/Documentos/DesaSisDist/Us... de memoria cache/T3$ java mult_matrices 2
daniela@burbuja:~/Documentos/DesaSisDist/Us... de memoria cache/T3$

4: daniela@burbuja: ~/Documentos/DesaSisDist/Us... de memoria cache/T3
daniela@burbuja:~/Documentos/DesaSisDist/Us... de memoria cache/T3$ java mult_matrices 3
daniela@burbuja:~/Documentos/DesaSisDist/Us... de memoria cache/T3$

5: daniela@burbuja: ~/Documentos/DesaSisDist/Us... de memoria cache/T3
daniela@burbuja:~/Documentos/DesaSisDist/Us... de memoria cache/T3$ java mult_matrices 4
daniela@burbuja:~/Documentos/DesaSisDist/Us... de memoria cache/T3$

daniela@burbuja: ~/Documentos/DesaSisDist/Us... de memoria cache/T3
static int N = 1000;
static int[][] A = new int[N][N];
static int[][] B = new int[N][N];
static int[][] C = new int[N][N];

static long checksum() {
    long sum = 0;
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            sum += C[i][j];
    return sum;
}

public static void main(String[] args) {
    long t1 = System.currentTimeMillis();
    // Inicializa las matrices A y B
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
        {
            A[i][j] = 2 * i + j;
            B[i][j] = 2 * i - j;
            C[i][j] = 0;
        }

    // Transpone la matriz B. La matriz transpuesta queda en B
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            B[j][i] = B[i][j];

    // Multiplica las matrices A y B
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            for (int k = 0; k < N; k++)
                C[i][j] += A[i][k] * B[k][j];

    long t2 = System.currentTimeMillis();
    long time = t2 - t1;
    System.out.println("Time: " + time);
    checksum();
}
```

En la captura de pantalla se puede observar la ejecución desde terminal de los 5 nodos. El número 0 es el que recibe los resultados de los demás e imprime únicamente el checksum de la matriz resultado C que es 791635400632448.