INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Desarrollo de sistemas distribuidos

Profesor: Carlos Pineda Guerrero

Grupo: 4CV4

Tarea 2. Uso eficiente de la memoria caché

Alumno: Vladimir Azpeitia Hernández

Tarea 2. Uso eficiente de la memoria caché

Desarrollo

MultiplicaMatriz.java

Para este ejercicio realizamos una multiplicación de matrices estándar (fila por columna) de A * B, utilizando tres ciclos *for* anidados.

```
class MultiplicaMatriz {
  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];

  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] = i + 2 * j;
        C[i][j] = 0;
      }
    }

    // multiplica la matriz A y la matriz B, el resultado queda en la matriz C
    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();
    System.out.println("Tiempo: " + (t2 - t1) + "ms");
  }
}
```

MultiplicaMatriz_2.java

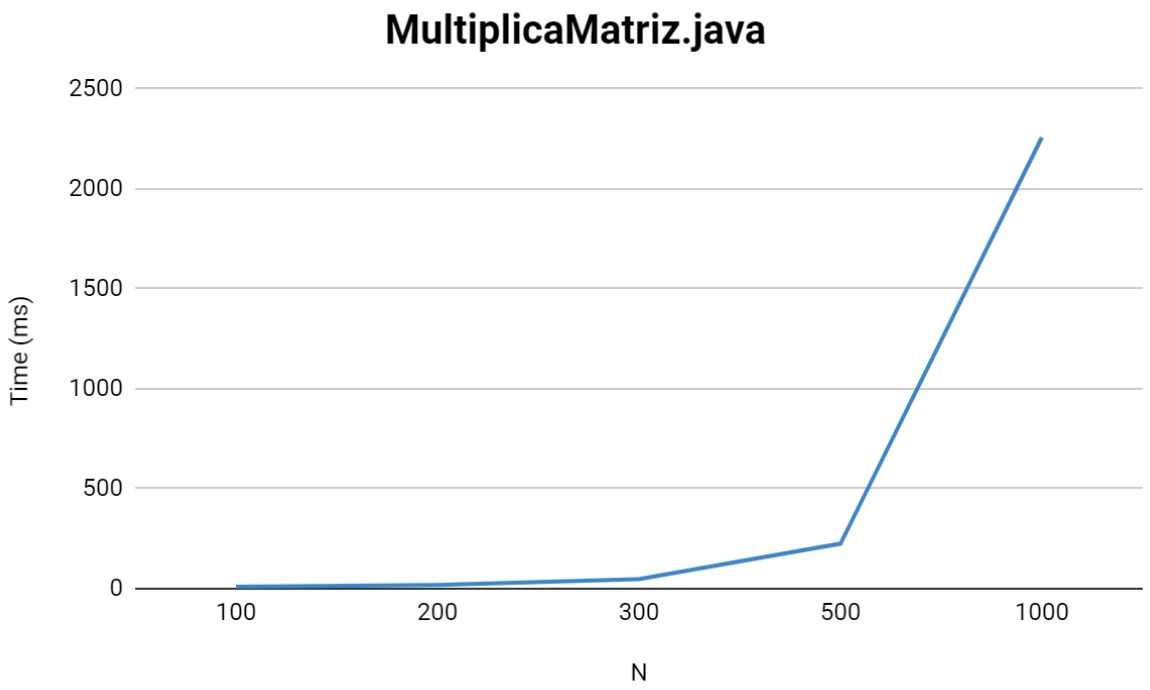
Para el segundo programa, sacamos la matriz transpuesta de B. Luego multiplicamos la matriz A por la matriz transpuesta de B, invirtiendo los índices de la matriz B

```
class MultiplicaMatriz2 {
  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];

  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] = i + 2 * j;
            C[i][j] = 0;
        }
    }

    // transpone la matriz B, la matriz traspuesta queda en B
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < i; j++) {
            int x = B[i][j];
            B[i][j] = B[j][i];
            B[j][i] = x;
        }
    }

    // multiplica la matriz A y la matriz B, el resultado queda en la matriz C
    // notar que los indices de la matriz B se han intercambiado
    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[j][k];
            }
        }
    }

    long t2 = System.currentTimeMillis();
    System.out.println("Tiempo: " + (t2 - t1) + "ms");
  }
}
```

Resultados obtenidos.
La ejecución de los programas se realizó en una máquina con un procesador intel i7 7700, con 8MB de caché, y 16GB de memoria RAM.

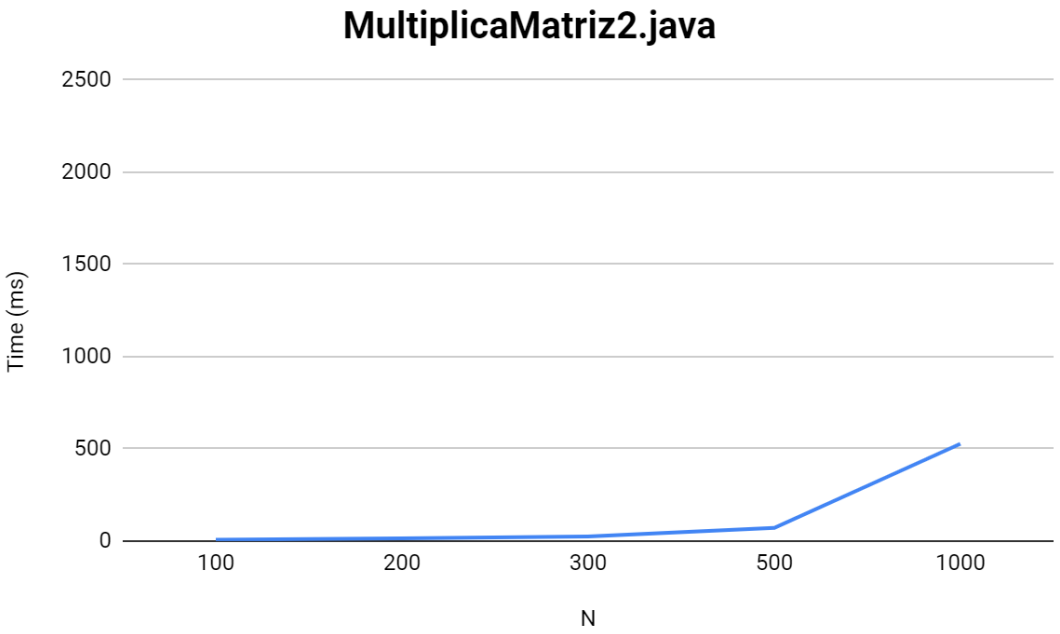
Para el primer programa se obtuvieron los siguientes resultados

N	Time (ms)
100	10
200	19
300	48
500	225
1000	2257



Para el segundo programa se obtuvieron los siguientes resultados

N	Time (ms)
100	7
200	15
300	25
500	71
1000	527



Conclusiones

El programa **MultiplicaMatriz2.java** es más eficiente que el primer programa, debido a que Java almacena las matrices como renglones y el primer programa accede a la matriz B por columnas, y esto se vuelve más ineficiente cuando la matriz B crece. Por otro lado, el segundo programa tiene un acceso más eficiente a los elementos de la matriz B, debido a que ahora se leen los elementos de B en forma secuencial, lo cual aumenta la localidad espacial y temporal de los datos.