



Instituto Politécnico Nacional
Escuela Superior de Cómputo

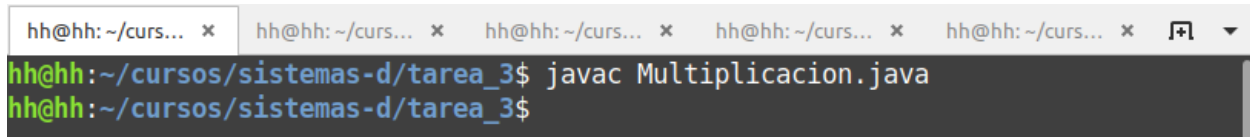
Desarrollo de Sistemas Distribuidos.
Tarea 2, Multiplicación distribuida de matrices usando paso de
mensajes.

Alumno:
Hernández Escudero Luis Hugo

Profesor:
Pineda Guerrero Carlos

15-MAR-2021

Compilación del programa.



```
hh@hh: ~/curs... x hh@hh: ~/curs... x hh@hh: ~/curs... x hh@hh: ~/curs... x hh@hh: ~/curs... x [x] v
hh@hh:~/cursos/sistemas-d/tarea_3$ javac Multiplicacion.java
hh@hh:~/cursos/sistemas-d/tarea_3$
```

Imagen 1. Compilación del programa.

Para compilar el programa únicamente se tiene que llamar al compilador de java, **javac**, pasando como argumento el nombre del archivo a compilar, en este caso el archivo es **Multiplicacion.java**.

> **javac Multiplicacion.java**

Este comando generará un archivo con extensión **.class** llamado **Multiplicacion.class**, que contiene el bytecode listo para ejecutarse en cualquier computadora con un entorno de ejecución java (java runtime environment).

Ejecución del programa.

El programa calculará la multiplicación de dos matrices cuadradas A y B de $N \times N$ elementos de tipo entero, para $N = \{4, 1000\}$ de forma distribuida, el resultado se colocará en la matriz C, también de $N \times N$ elementos, así como también el checksum de la matriz C calculada. Para esto, se utilizarán 5 nodos, el nodo 0 actuará como servidor y será quien envíe una submatriz de A y otra de B a cada nodo cliente, los nodos cliente calculan una submatriz de C al multiplicar las submatrices de A y B que recibieron del servidor y envían el resultado de vuelta servidor.

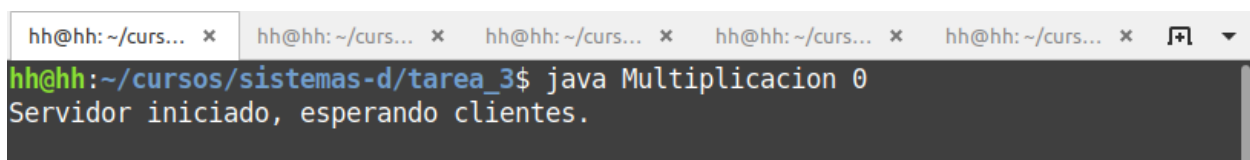
Finalmente, el servidor combinará (de forma correcta) las submatrices enviadas por los nodos cliente, calculará el checksum de la matriz C y lo despliega en la consola.

El programa comienza su ejecución al ejecutar el comando **java** seguido del nombre del archivo (sin la extensión) que contiene el bytecode generado por el compilador de java y de los argumentos del programa, el archivo con el bytecode en este caso se llama **Multiplicacion.class**.

Este programa recibe como argumento el número de nodo, siendo el nodo 0 quien actúe como servidor, y los nodos en el rango [1, 4] quienes actuarán como cliente. Por lo que para ejecutar el programa se ejecuta el siguiente comando.

> **java Multiplicacion <Numero de nodo>**

Para esto, es necesario que inicialmente el servidor esté listo para aceptar a los 4 clientes, es decir el primer nodo que debe ejecutarse es el nodo 0.

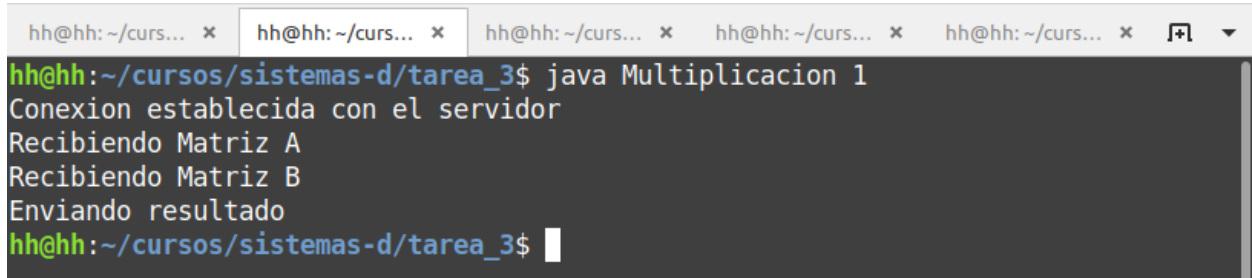


```
hh@hh: ~/curs... x hh@hh: ~/curs... x hh@hh: ~/curs... x hh@hh: ~/curs... x hh@hh: ~/curs... x [x] v
hh@hh:~/cursos/sistemas-d/tarea_3$ java Multiplicacion 0
Servidor iniciado, esperando clientes.
```

Imagen 2. Ejecución del programa pasando como parámetro el número 0.

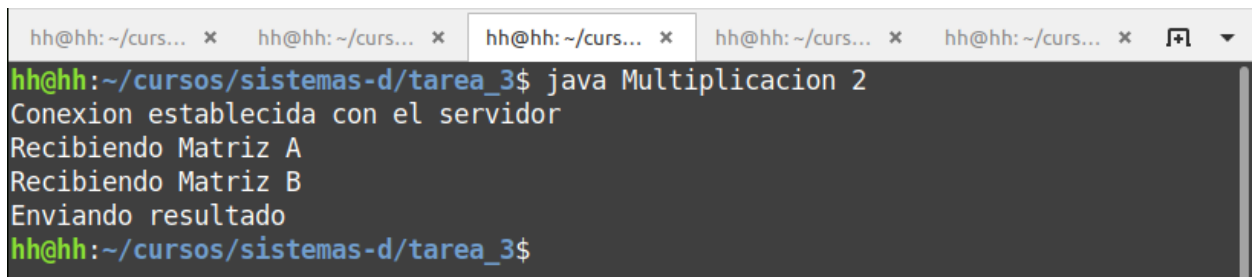
El servidor abre el puerto 5000 y se pone en espera de peticiones al ejecutar la directiva **listen**. Durante 4 iteraciones se aceptan 4 clientes y cada uno se atiende en un hilo de ejecución diferente, espera a que todos los hilos terminen su ejecución, calcula la suma de comprobación (checksum) y lo imprime.

Para que el programa actúe como cliente, el parámetro que recibe es un numero en el rango [1,4]. Se necesitan ejecutar exactamente 4 programas cliente, con su nodo correspondiente, para que el servidor termine su ejecución de manera exitosa.

A terminal window with a dark background and light green text. The prompt is 'hh@hh:~/cursos/sistemas-d/tarea_3\$'. The command 'java Multiplicacion 1' has been executed. The output shows a successful connection to the server, receipt of matrices A and B, and sending of the result. The prompt is now 'hh@hh:~/cursos/sistemas-d/tarea_3\$' with a cursor.

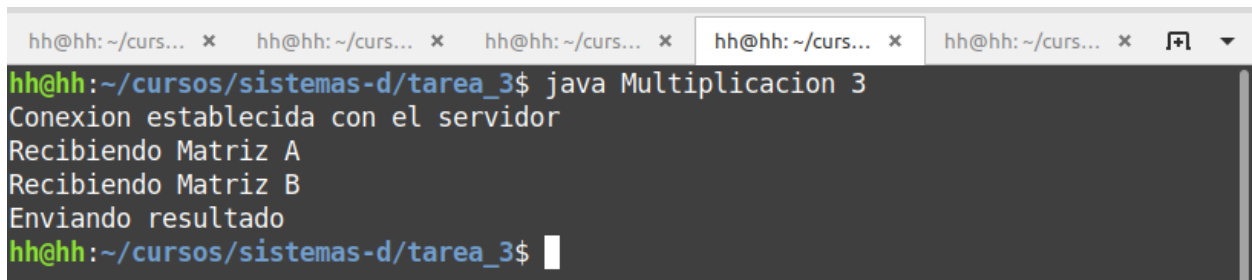
```
hh@hh:~/cursos/sistemas-d/tarea_3$ java Multiplicacion 1
Conexion establecida con el servidor
Recibiendo Matriz A
Recibiendo Matriz B
Enviando resultado
hh@hh:~/cursos/sistemas-d/tarea_3$
```

Imagen 3. Ejecución del programa cliente 1.

A terminal window with a dark background and light green text. The prompt is 'hh@hh:~/cursos/sistemas-d/tarea_3\$'. The command 'java Multiplicacion 2' has been executed. The output shows a successful connection to the server, receipt of matrices A and B, and sending of the result. The prompt is now 'hh@hh:~/cursos/sistemas-d/tarea_3\$' with a cursor.

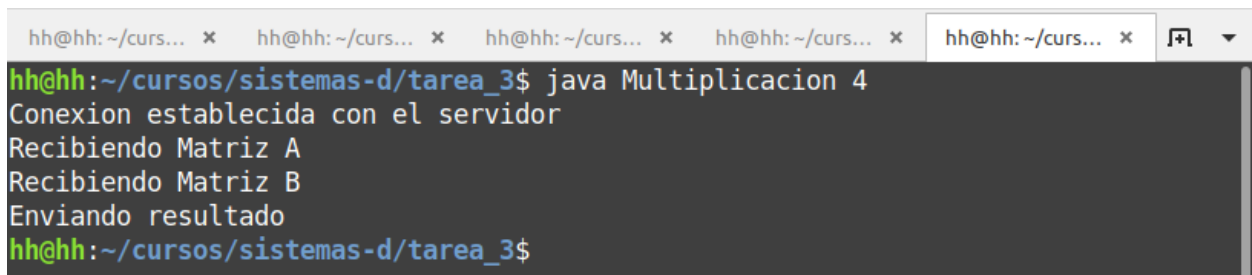
```
hh@hh:~/cursos/sistemas-d/tarea_3$ java Multiplicacion 2
Conexion establecida con el servidor
Recibiendo Matriz A
Recibiendo Matriz B
Enviando resultado
hh@hh:~/cursos/sistemas-d/tarea_3$
```

Imagen 4. Ejecución del programa cliente 2.

A terminal window with a dark background and light green text. The prompt is 'hh@hh:~/cursos/sistemas-d/tarea_3\$'. The command 'java Multiplicacion 3' has been executed. The output shows a successful connection to the server, receipt of matrices A and B, and sending of the result. The prompt is now 'hh@hh:~/cursos/sistemas-d/tarea_3\$' with a cursor.

```
hh@hh:~/cursos/sistemas-d/tarea_3$ java Multiplicacion 3
Conexion establecida con el servidor
Recibiendo Matriz A
Recibiendo Matriz B
Enviando resultado
hh@hh:~/cursos/sistemas-d/tarea_3$
```

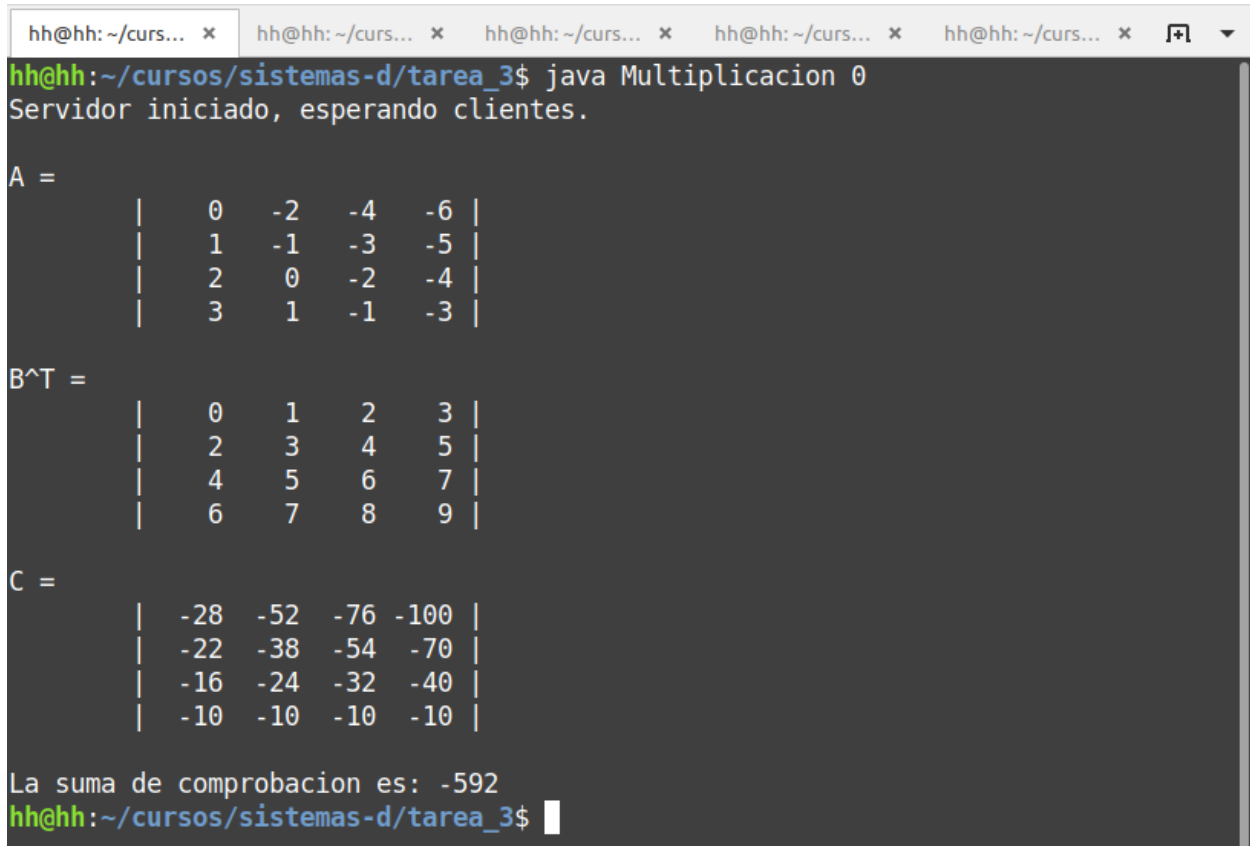
Imagen 5. Ejecución del programa cliente 3.

A terminal window with a dark background and light green text. The prompt is 'hh@hh:~/cursos/sistemas-d/tarea_3\$'. The command 'java Multiplicacion 4' has been executed. The output shows a successful connection to the server, receipt of matrices A and B, and sending of the result. The prompt is now 'hh@hh:~/cursos/sistemas-d/tarea_3\$' with a cursor.

```
hh@hh:~/cursos/sistemas-d/tarea_3$ java Multiplicacion 4
Conexion establecida con el servidor
Recibiendo Matriz A
Recibiendo Matriz B
Enviando resultado
hh@hh:~/cursos/sistemas-d/tarea_3$
```

Imagen 6. Ejecución del programa cliente 4.

Finalmente, después de que todos los hilos terminaron su ejecución, el servidor calcula la suma de comprobación, si $N = 4$ se imprimen las matrices A , B^T y C , seguidas de la suma de comprobación, si $N = 1000$ únicamente se imprime la suma de comprobación.



```
hh@hh:~/curs... x hh@hh:~/curs... x hh@hh:~/curs... x hh@hh:~/curs... x hh@hh:~/curs... x
hh@hh:~/cursos/sistemas-d/tarea_3$ java Multiplicacion 0
Servidor iniciado, esperando clientes.

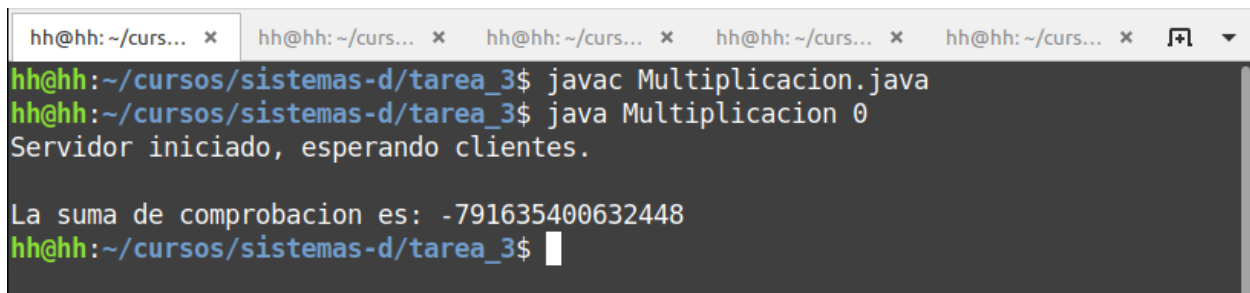
A =
| 0 -2 -4 -6 |
| 1 -1 -3 -5 |
| 2 0 -2 -4 |
| 3 1 -1 -3 |

B^T =
| 0 1 2 3 |
| 2 3 4 5 |
| 4 5 6 7 |
| 6 7 8 9 |

C =
| -28 -52 -76 -100 |
| -22 -38 -54 -70 |
| -16 -24 -32 -40 |
| -10 -10 -10 -10 |

La suma de comprobacion es: -592
hh@hh:~/cursos/sistemas-d/tarea_3$
```

Imagen 7. Salida del programa servidor para $N = 4$.



```
hh@hh:~/curs... x hh@hh:~/curs... x hh@hh:~/curs... x hh@hh:~/curs... x hh@hh:~/curs... x
hh@hh:~/cursos/sistemas-d/tarea_3$ javac Multiplicacion.java
hh@hh:~/cursos/sistemas-d/tarea_3$ java Multiplicacion 0
Servidor iniciado, esperando clientes.

La suma de comprobacion es: -791635400632448
hh@hh:~/cursos/sistemas-d/tarea_3$
```

Imagen 8. Salida del programa servidor para $N = 1000$.

Conclusión.

Realizar esta práctica, me ayudó a poder combinar las técnicas de programación anteriormente vistas y adicionalmente agregar la parte de programar eficientemente para la cache, de esta forma, no solo estamos aprovechando las capacidades de concurrencia y paralelismo del procesador, sino también aprovechamos nuestra memoria cache al máximo.

Adicionalmente, esta práctica me ayudo a reconocer que cuando se desea enviar una gran cantidad de datos, es mejor enviarlos en conjunto como un arreglo de bytes en lugar de uno por uno.

La primera versión del programa la implemente con los métodos `readInt` y `writeInt` de las clases `DataInputStream` y `DataOutputStream`, en el caso de $N = 2$ el programa parecía tener fluidez, sin embargo, para $N = 1000$ el programa comenzaba a tardar en cada nodo cliente. Por esa razón decidí cambiar la técnica de escribir dato por dato por la de escribir un conjunto de datos empaquetados en un arreglo de bytes, esto aumento la velocidad del programa significativamente.