



Instituto Politécnico Nacional Escuela Superior de Cómputo

Desarrollo de sistemas distribuidos

Reporte Tarea 3.
"Multiplicación de matrices distribuida utilizando paso de mensajes"

Grupo: 4CV13

Integrantes: Erick Eduardo Ramírez Arellano Elisa Ramos Gomez Omar Ramos Herrera

Desarrollo.

Para esta tarea desarrollamos la multiplicación de matrices de manera distribuida donde bueno por tener una semana de exámenes la mayoría de la lógica la realizamos en el main donde lo separamos de la siguiente forma:

- 1. Cuando ingresan valor de 0 se ejecutará la porción de código designado al cliente.
- 2. Cuando se ingresa un valor de 1 a 3, se ejecutará la porción de código asignado al servidor.
- 3. Cualquier otra cosa ingresada mandará una excepción de RunTime, con el mensaje "valor de nodo incorrecto".

1. Cliente

Esta clase pues tiene mucha similitud con las anteriores ya que es una clase estática que hereda de la clase Threads, donde lo que hacemos el la función de las flechas dentro de la imágen de la topología, donde tenemos un puerto, una ip y partes de la matriz global que son las que enviamos a los servidores para realizar el cálculo, en la función run lo que hacemos es el re-intento de conexión donde le mandamos la dirección de las IPs públicas de las máquinas virtuales y los puerto que configuramos también dentro de ellas, una vez establecida la conexión enviamos la parte de la matriz correspondiente, donde cada servidor pues hace su trabajo que lo describimos abajo y una vez realizado esto recibimos la matriz con su resultado, en código esto se ve de la siguiente manera:

```
static class Cliente extends Thread{
   int puerto;
   String ip;
   double[][] X;
   double[][] M;
   double[][] C;
   Cliente(String ip, int puerto, double[][] X, double[][] M) {
     this.puerto = puerto;
     this.ip = ip;
     this.X = X;
     this.M = M;
   }
   public void run() {
     Socket conexion = null;
     for(;;) {
       try{
           conexion = new Socket(ip, puerto);
            DataOutputStream salida = new
            DataOutputStream(conexion.getOutputStream());
```

```
DataInputStream entrada = new
          DataInputStream(conexion.getInputStream());
         enviarMatriz(salida, X, N);
         enviarMatriz(salida, M, N);
         C = recibirMatriz(entrada, N);
         Thread.sleep(1000);
         conexion.close();
         break;
     }catch(Exception e) {
         try {
             Thread.sleep(100);
         } catch (InterruptedException e2) {
             System.out.println(e2.getMessage());
     }
  }
}
```

2. Funciones de las matrices

Transponer una matriz: Mediante el uso de dos for, se invierte el orden de las listas mandando a llamar i y j, para estos después ser invertidos guardándolo en una nueva variable para posteriormente el nuevo resultado ser mandado a llamar obteniendo así la matriz transpuesta.

```
static double[][] transponerMatriz(double[][] M, int N) {
    double [][] bTAux = new double[N][N];
    for(int i = 0; i < N; i++) {
        bTAux[i][j] = M[j][i];
    }
}

for(int i = 0; i < N; i++) {
    for(int j = 0; j < N; j++) {
        M[i][j] = bTAux[i][j];
    }
}
return M;
}</pre>
```

Multiplicar matriz: mediante el arreglo de tres for se hace el cálculo de la matriz c, multiplicando la matriz a y b, dichos for mantienen la de las multiplicaciones de matrices recorriendo i y j sin estos pasarse de un valor establecido (4), ya que la multiplicación de matrices se hace con una horizontal (2x4) y una vertical (4x2) obteniendo así una matriz 4x4 manteniendo a k como la variable que rige las otras dos matrices.

```
int I, int N) {
   for(int i = 0; i < I; i++) {
     for(int j = 0; j < I; j++) {
        for(int k = 0; k < N; k++) {
            C[i][j] += A[i][k]*B[j][k];
        }
    }
   return C;}</pre>
```

Mostrar matriz: En esta parte del código se imprime la matriz mediante el uso de dos for que irán arrastrando los valores de la matriz a imprimir.

```
static void mostrarMatriz(double[][] M, int N, int I) {
   for (int i = 0; i < N; i++) {
      for (int j = 0; j < I; j++) {
          System.out.print(M[i][j] + "\t ");
      }
      System.out.println("");
   }
   System.out.println("");
}</pre>
```

Dividir una matriz: En esta parte de la función se divide la matriz en mitad para esta posteriormente ser multiplicada con la otra mitad de la matriz con la cual será dividida, esto mediante el empleo de la función declarada en la variable Limit, la cual será arrastrada por el método for obteniendo así el resultado.

```
static double[][] dividirMatriz(double[][] M, int N, int I) {
    double[][] X = new double[N/2][N];
    int Limit = (I != 0) ? N : N/2;

    for(int i = I, k = 0; i < Limit; i++, k++) {
        for(int j = 0; j < N; j++) {
            X[k][j] = M[i][j];
        }
    }
    return X;
}</pre>
```

Envío de matriz: Gracias al uso de dos for, se irá arrastrando los valores de la matriz a enviar, para está mediante la función salida.write.Double ser escrita en el lugar solicitado.

```
static void enviarMatriz(DataOutputStream salida, double[][] M, int N) throws
Exception {
    for(int i = 0; i < N/2; i++) {
        for(int j = 0; j < N; j++) {
            salida.writeDouble(M[i][j]);
        }
    }
}</pre>
```

Recibir matriz: Mediante esta función, se hace la recepción de la matriz enviada por la función explicada anteriormente, almacenando en una nueva variable que será llamada posteriormente sin antes haber estado declarado en una nueva matriz, las listas de esta matriz serán arrastradas mediante los dos for empleados.

```
static double[][] recibirMatriz(DataInputStream entrada, int N) throws
Exception {
         double[][] X = new double[N/2][N];
         for(int i = 0; i < N/2; i++) {</pre>
```

```
for(int j =0; j < N; j++) {
      X[i][j] = entrada.readDouble();
    }
}
return X;
}</pre>
```

Checksum de una matriz: Con la función de esta parte del código se podrá realizar el checksum de la matriz, lo cual consiste en sumar todas las variables de dicha matriz, esto se puede realizar gracias a la variable declarada sum que se sumará e igualara con las variables arrastradas de la matriz mediante los dos for.

```
static double checksum(double[][] M, int N) {
    double sum = 0;
    for(int i = 0; i < N; i++) {
        for(int j = 0; j < N; j++) {
            sum += M[i][j];
        }
    }
    return sum;
}</pre>
```

3. Main

En el método main como en los anteriores programas busca primero el argumento que se envía desde la terminal de comandos, busca si es cliente o servidor para empezar con el proceso. Si es cliente entonces se creará la transpuesta de la matriz B y se envían las matrices a multiplicar, al término regresa la matriz cuadrada. Definimos las ip públicas de las máquinas virtuales en azure con su puerto previamente configurado, creamos 3 clientes que se conectarán a los 3 servidores y les enviaran las matrices A1, A2, B1, B2.

Cada nodo se ejecutará en una máquina virtual, los servidores recibirán 2 matrices y las multiplicarán, regresando una matriz que formará parte de la matriz final.

Si el valor de N es 8 entonces se mostrará la matriz A y B. También el checksum, ahora si N es 1000 entonces solo se mostrará el checksum. El código se ver de esta manera:

```
public static void main(String[] args) {
    if (Integer.parseInt(args[0]) == 0) {
        //inicializacion de las matrices
        for(int i = 0; i < N; i++) {
            for(int j = 0; j < N; j++) {
                A[i][j] = i+5*j;
                B[i][j] = 5*i-j;
                C[i][j] = 0;
        }
    }

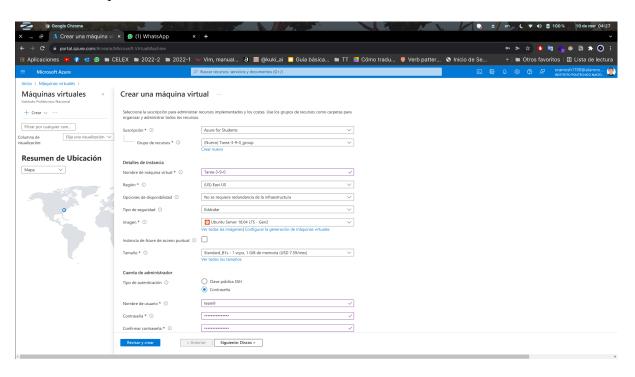
B = transponerMatriz(B, N);
    Matrices.add(dividirMatriz(A,N, 0));
    Matrices.add(dividirMatriz(B,N, N/2));

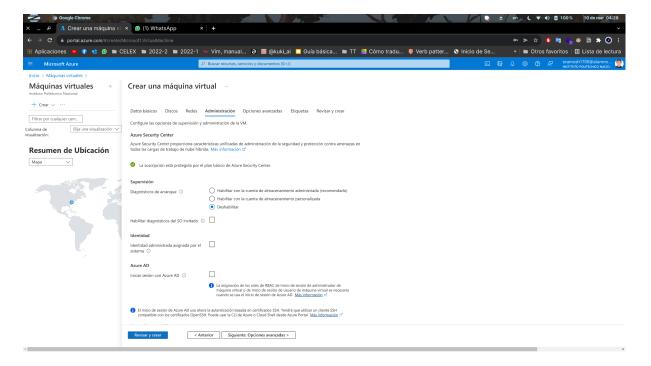
Matrices.add(dividirMatriz(B,N, N/2));</pre>
```

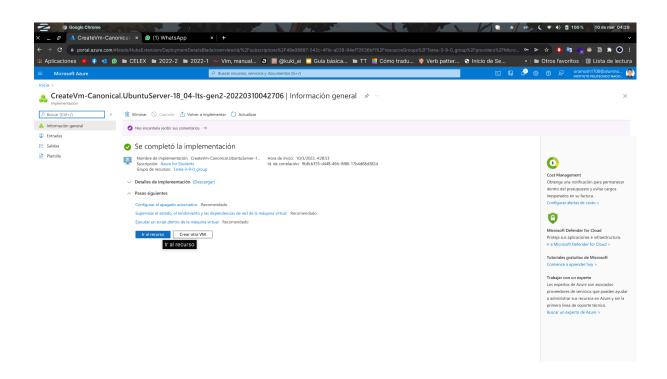
```
try {
             // Cliente Cli1 = new
            Cliente ("localhost", 50001,
            Matrices.get(0), Matrices.get(2));
            // Cliente Cli2 = new
            Cliente ("localhost", 50002,
            Matrices.get(0), Matrices.get(3));
            // Cliente Cli3 = new
            Cliente ("localhost", 50003,
            Matrices.get(1), Matrices.get(2));
            Cliente Cli1 = new
            Cliente ("52.234.149.97", 50001,
            Matrices.get(0), Matrices.get(2));
            Cliente Cli2 = new
            Cliente("20.25.74.41", 50002,
            Matrices.get(0), Matrices.get(3));
            Cliente Cli3 = new
            Cliente ("20.25.15.253", 50003,
            Matrices.get(1), Matrices.get(2));
            Cli1.start();
            Cli2.start();
            Cli3.start();
            Clil.join();
            Cli2.join();
            Cli3.join();
        aux =multiplicaMatriz(Matrices.get(1),
Matrices.get(3), aux, N/2,N);
        C = crearMatrizC(C, Cli1.C, 1);
        C = crearMatrizC(C, Cli2.C, 2);
        C = crearMatrizC(C, Cli3.C, 3);
        C = crearMatrizC(C, aux, 4);
        // mostrarMatriz(C,N,N);
        if (N \le 8) {
          mostrarMatriz(A, N, N);
          mostrarMatriz(B, N, N);
          mostrarMatriz(C, N, N);
        System.out.println("Checksum: " + checksum(C,
N));
      } catch (Exception e) {
       System.out.println(e.getMessage());
    } else if (Integer.parseInt(args[0]) >= 1 &&
Integer.parseInt(args[0]) <= 3) {</pre>
      try {
        ServerSocket server = new ServerSocket(50000
+ Integer.parseInt(args[0]));
        Socket conexion = server.accept();
        DataOutputStream salida = new
DataOutputStream(conexion.getOutputStream());
        DataInputStream entrada = new
DataInputStream(conexion.getInputStream());
        double[][] a = recibirMatriz(entrada, N);
        double[][] b = recibirMatriz(entrada, N);
        double[][] c = new double[N/2][N];
        c = multiplicaMatriz(a,b,c,N/2,N);
        enviarMatriz(salida,c,N);
```

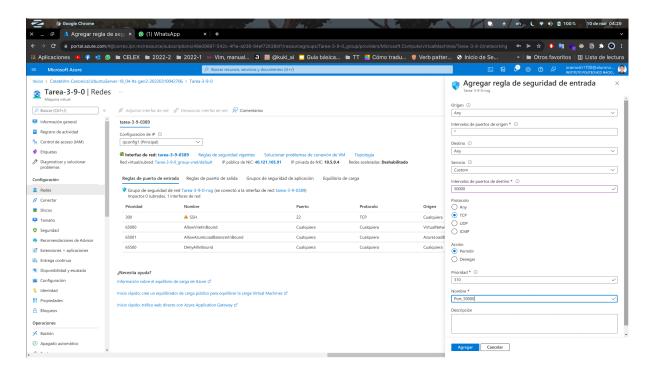
```
server.close();
    salida.close();
    entrada.close();
} catch (Exception e) {
    System.out.println(e.getMessage());
}
} else {
    throw new RuntimeException("un valor de nodo incorrecto");
}
}
```

Para la creación de la máquina virtual seguimos los pasos que estamos adjuntando a continuación de la creación de las máquinas virtuales, paso a paso la selección de las características para realizar esta tarea.



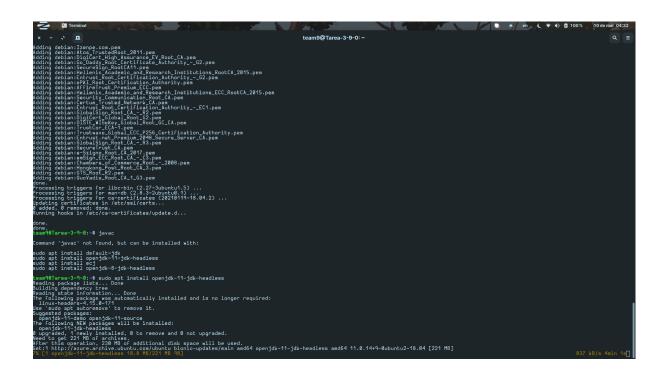




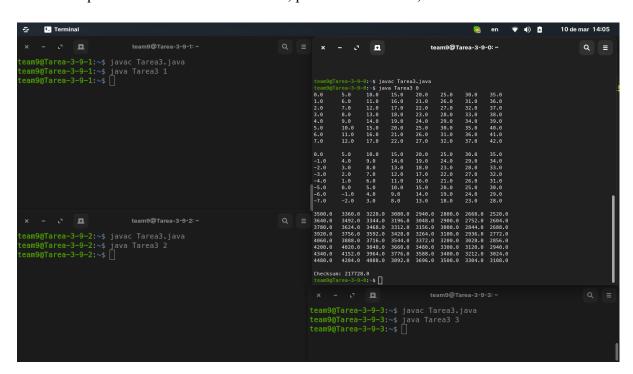


```
team@@Tarea-3-9-0:-

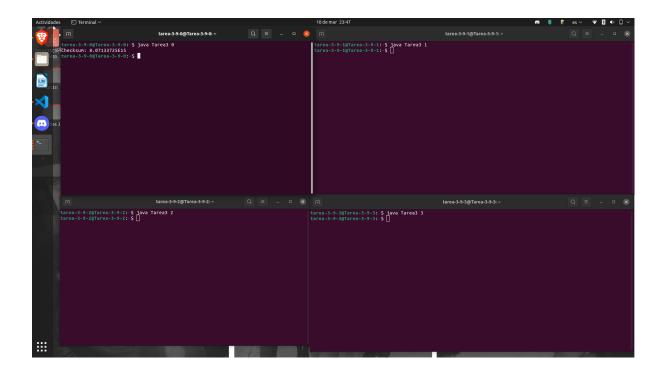
mode spit install default-jr.
mode spit insta
```



Ahora adjuntamos las capturas de pantalla correspondientes donde vamos apreciar el checksum que es la suma de toda la matriz, para un valor n = 8, el resultado obtenido es:



Para un valor de n = 1000, tenemos el siguiente resultado:



Conclusiones.

Erick Eduardo Ramírez Arellano: Mediante la elaboración de esta práctica comprendí más a fondo la comunicación que se puede desarrollar de un cliente servidor, aunque se me complicó más el aspecto de las matrices y sus funciones, en especial con el checksum pude comprender, gracias a los apuntes del profesor, los cálculos que se deben desarrollar para este, igual comprendí más a fondo la diferencia que se puede tener al momento de realizar la comunicación entre cliente-servidor si se hace en mi computadora a si se desarrolla en una máquina virtual.

Un punto más a destacar fue el empleo de los for y el impacto que se tiene al desarrollar la práctica, si bien el uso de estos no puede ser tan factible, pero, acordé a las comunicaciones y el cómo se fue tratando a la matriz con las diversas funciones, agiliza los tiempos de trabajo y visualización de las matrices.

Omar Ramos Herrera: Con el desarrollo de esta tarea ocupamos ya más conceptos que se han tomado en clase, las máquinas virtuales, como abrir un puerto y conectarlas con sockets, la creación de las máquinas se me hizo fácil, gracias a los procedimientos en la plataforma. Aprendí cómo configurar una máquina virtual me visto de su sistema operativo, cpu, ram, hdd entre otros.

Lo que más se me dificulta fue entender cómo afectaba a la memoria que la matriz no estuviera transpuesta. Me parece algo muy interesante que se pueda optimizar el tiempo de respuesta de esa manera. El uso de for anidados pensé que sería mucho más lento pero de acuerdo a las marcas de tiempo no fue tan tardado. Me gustaron las iteraciones de arreglos bidimensionales, y la conexión con las Máquinas de Azure fue sencilla con los sockets.

Elisa Ramos Gomez: Está tarea de forma personal me emocionó mucho realizarla ya que fue mi primer acercamiento con Azure y creación de máquinas virtuales, conectarse a ellas, seleccionar las características, la verdad fue una impresión pues padre todas las cosas que ofrecen, entonces bueno la creación después de gastarme como medio dolar del crédito por andar creando a prueba y error la creación de las máquinas ya entendí como funcionaba, afortunada o desafortunadamente mi computadora tenía Windows y me había borrado todas mis carpetas del sistema y las mando a dependencias OneDrive y luego el disco decidió decirme adiós entonces la mande a reparar y le instale Ubuntu, me quede impactada de la facilidad con la que nos podemos conectar a las máquinas virtuales sin necesidad de tanta complicación; por otra parte la tarea fue un reto para mi porque tuve que practicar haciendo ejercicios con matrices para poder seccionar la matriz para que la caché no haga saltos de fila, entonces igual poco a poco fueron siendo consistentes los resultados, pero me gustó mucho el poder practicar matrices y distribuir cálculos.