



**Instituto Politécnico Nacional  
Escuela Superior de Cómputo**



# **Tarea 4: Multiplicación de matrices utilizando objetos distribuidos**

**Olay Silis Jose Eduardo  
4CV12  
Prof. Carlos Pineda Guerrero**

# Introducción

Durante el desarrollo de esta práctica se implementó un programa en Java RMI en el cual el cliente inicializa y divide 2 matrices rectangulares para que mediante un método remoto pueda calcular la multiplicación de la matriz resultante a través de una interfaz RMI.

## Desarrollo

### Fase 1

El desarrollo del código de esta tarea lo realice utilizando ChatGPT y el programa de multiplicación de matrices proporcionado por el profesor y el cual se vio en clase. En primera instancia le fui preguntando al chat los pasos para ir creando las operaciones que realizaba el cliente las cuales eran inicializar las matrices, calcular la transpuesta y dividir dichas matrices en 4 partes iguales obteniendo lo que se muestra en las imágenes siguientes.

The screenshot shows a Microsoft Edge browser window with the following details:

- Address Bar:** chat.openai.com/c/e5040271-0201-4257-ac7b-eb1242f2bc00
- Page Title:** Java RMI Multiplicación Matrices
- Content Area:** A conversation with "Default (GPT-3.5)".
  - User message: "Hola, buenos días necesito que me ayudes a desarrollar un sistema que calcule el producto de dos matrices rectangulares utilizando Java RMI el cual tendrá un cliente y un servidor. Primero comenzaremos a codificar el cliente el cual se llamará ClienteRMI."
  - ChatGPT response: "Necesito que el cliente inicialice las matrices A y B de la siguiente forma:  
La matriz A con N renglones y M columnas de tipo double se inicializa de la siguiente forma:  
 $A[i][j] = 6 * i - 2 * j$ "
  - User message: "La matriz B con M renglones y N columnas de tipo double. se inicializa de la siguiente forma:  
 $B[i][j] = 8 * i + 3 * j$ "
  - ChatGPT response: "N y M se pasaran como parametros al programa"
  - User message: "Claro, puedo ayudarte a desarrollar un sistema que calcule el producto de dos matrices rectangulares utilizando Java RMI. Primero, crearemos el cliente RMI que inicializará las matrices A y B como lo has descrito y luego las enviará al servidor RMI para calcular el producto de matrices. Aquí tienes el código del cliente:"
  - Code block (highlighted in red):

```
java
```
  - Message input field: "Envía un mensaje"
  - Small note at the bottom: "Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version"
- Left Sidebar:** A sidebar with a list of recent chats and sections.
  - Hoy: RMI Paralelo Matrices Java, Java RMI Matrices Multiply, Java RMI Multiplicación, Multiplicación Matrices con R
  - Últimos 7 Días: Compilar interfaz Java RMI
  - Últimos 30 Días: Servidor multithread para peti, Calculadora huella carbono, Calcular Producto de Matrices, Comando Minecraft Educati
- Bottom Status Bar:** Shows the date (30/10/2023), time (04:48 p.m.), and weather (21°C, Mayorm. soleado).

The screenshot shows a Microsoft Edge browser window with the following tabs:

- Correo: Eduardo Olay Sili - Outlook
- SISDIS-4CV12: Tarea 4. Multiplicación
- Curso: Desarrollo de Sistemas Distribuidos
- Java RMI Multiplicación Matrices

The main content area displays a ChatGPT conversation:

**java**

```
import java.rmi.Naming;
public class ClienteRMI {
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Uso: java ClienteRMI <N> <M>");
            System.exit(1);
        }
        int N = Integer.parseInt(args[0]);
        int M = Integer.parseInt(args[1]);
        try {
            // Inicializar las matrices A y B
            double[][] A = new double[N][M];
            double[][] B = new double[M][N];
            for (int i = 0; i < N; i++) {
                for (int j = 0; j < M; j++) {
                    A[i][j] = 6 * i - 2 * j;
                    B[j][i] = 8 * j + 3 * i;
                }
            }
        }
    }
}
```

**Envía un mensaje**

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version

04:48 p. m.  
30/10/2023

The sidebar on the left lists recent chats:

- + Nuevo chat
- Hoy
- RMI Paralelo Matrices Java
- Java RMI Matrices Multiply
- Java RMI Multiplicación
- Multiplicación Matrices con R
- Últimos 7 Días
- Compilar interfaz Java RMI
- Últimos 30 Días
- Servidor multithread para peti
- Calculadora huella carbono.
- Calcular Producto de Matrices
- septiembre
- Comando Minecraft Educatio
- Actualizar a Plus
- Jose Eduardo Olay Siliis ...

21°C  
Mayorm. soleado

The screenshot shows a Microsoft Edge browser window with the same tabs as the previous screenshot.

The main content area displays a ChatGPT conversation:

**Default (GPT-3.5)**

```
// Conectar al servidor RMI
String servidorURL = "rmi://localhost/ServidorMatrices";
ServidorMatrices servidor = (ServidorMatrices) Naming.lookup(servidorURL);

// Llamar al método remoto para calcular el producto de matrices
double[][] resultado = servidor.multiplicarMatrices(A, B);

// Mostrar el resultado
System.out.println("Matriz A:");
imprimirMatriz(A);
System.out.println("Matriz B:");
imprimirMatriz(B);
System.out.println("Resultado del producto de matrices:");
imprimirMatriz(resultado);
} catch (Exception e) {
    e.printStackTrace();
}

private static void imprimirMatriz(double[][] matriz) {
    for (int i = 0; i < matriz.length; i++) {
        for (int j = 0; j < matriz[0].length; j++) {
            System.out.print(matriz[i][j] + " ");
        }
    }
}
```

**Envía un mensaje**

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version

04:48 p. m.  
30/10/2023

The sidebar on the left is identical to the first screenshot.

21°C  
Mayorm. soleado

The screenshot shows a web-based AI chat interface. On the left, there is a sidebar with a list of recent conversations and a weather widget indicating 21°C and sunny conditions. The main area is a dark-themed window titled "Default (GPT-3.5)". It contains Java code for matrix multiplication:

```
        imprimirMatriz(resultado);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private static void imprimirMatrix(double[][] matriz) {
    for (int i = 0; i < matriz.length; i++) {
        for (int j = 0; j < matriz[0].length; j++) {
            System.out.print(matriz[i][j] + " ");
        }
        System.out.println();
    }
}
```

Below the code, there is a note: "Asegúrate de que el servidor RMI esté configurado y escuchando en la dirección "rmi://localhost/ServidorMatrices". A continuación, necesitarás implementar el servidor RMI y la interfaz "ServidorMatrices" en el servidor para realizar el cálculo del producto de las matrices." At the bottom right of the window is a button labeled "Volver a generar".

This screenshot shows the same AI chat interface as the previous one. The sidebar and weather widget are identical. The main window now displays a message from the AI: "ok, ahora necesito que una vez que haya inicializado las matrices A y B me ayudes a implementar la obtención de la matriz BT la cual es la transpuesta de B". Below this, the AI provides Java code to obtain the transpose of matrix B:

```
Claro, puedes obtener la matriz traspuesta 'BT' de la matriz 'B' en Java de la siguiente manera:
```

```
java
// Obtener la matriz traspuesta BT de la matriz B
double[][] BT = new double[N][M];

for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        BT[j][i] = B[i][j];
    }
}

// Imprimir la matriz traspuesta BT
System.out.println("Matriz B Traspuesta (BT):");
imprimirMatriz(BT);
```

At the bottom of the window, there is a note: "Agrega este código después de la inicialización de la matriz 'B'. El bucle anidado intercambia las filas y columnas de la matriz 'B', creando así la matriz trasposta 'BT'." There is also a "Volver a generar" button.

Una vez proporcionado esto le pedí que me diera una idea para implementar los threads dentro del cliente para que de la matriz que se dividieran se fueran multiplicando paralelamente en los nodos correspondientes.

Correo: Eduardo Olay Sili - Outlook | SISDIS-4CV12: Tarea 4. Multiplicación | Curso: Desarrollo de Sistemas Distribuidos | RMI Paralelo Matrices Java

chat.openai.com/c/2ecd696c-30ef-4859-aafe-a0f7da90bf7c

+ Nuevo chat

Hoy

- RMI Paralelo Matrices J
- Java RMI Matrices Multiply
- Java RMI Multiplicación Matr
- Multiplicación Matrices con R

Últimos 7 Días

- Compilar interfaz Java RMI
- Servidor multithread para peti
- Calculadora huella carbono.
- Calcular Producto de Matrices

septiembre

- Comando Minecraft Educati
- Actualizar a Plus
- José Eduardo Olay Sili ...

21°C Mayor m. soleado

Entiendo tu solicitud. Para lograr que el cliente utilice threads para invocar el método remoto `multiplica_matrices()` en los servidores RMI y calcular los productos en paralelo, puedes seguir estos pasos:

- Define la interfaz RMI:

Primero, define la interfaz RMI que contiene el método remoto `multiplica_matrices()`. Asegúrate de que tanto el servidor como el cliente tengan acceso a esta interfaz.

```
javaimport java.rmi.Remote;
import java.rmi.RemoteException;

public interface MatrizMultiplicadora extends Remote {
    int[][] multiplica_matrices(int[][] matrizA, int[][] matrizB)
}
```

Volver a generar

Envía un mensaje

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version

04:52 p. m.  
30/10/2023

Correo: Eduardo Olay Sili - Outlook | SISDIS-4CV12: Tarea 4. Multiplicación | Curso: Desarrollo de Sistemas Distribuidos | RMI Paralelo Matrices Java

chat.openai.com/c/2ecd696c-30ef-4859-aafe-a0f7da90bf7c

+ Nuevo chat

Hoy

- RMI Paralelo Matrices J
- Java RMI Matrices Multiply
- Java RMI Multiplicación Matr
- Multiplicación Matrices con R

Últimos 7 Días

- Compilar interfaz Java RMI
- Servidor multithread para peti
- Calculadora huella carbono.
- Calcular Producto de Matrices

septiembre

- Comando Minecraft Educati
- Actualizar a Plus
- José Eduardo Olay Sili ...

21°C Mayor m. soleado

1. Implementa la interfaz en los nodos:

En cada nodo del servidor, implementa la interfaz RMI en una clase que proporcione la lógica para multiplicar matrices.

En el "Nodo 1":

```
javaimport java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class Nodo1MatrizMultiplicadora extends UnicastRemoteObject implements
    public Nodo1MatrizMultiplicadora() throws RemoteException {
        super();
    }

    @Override
    public int[][] multiplica_matrices(int[][] matrizA, int[][] matrizB) throws RemoteException {
        // Realiza la multiplicación de las matrices C1 a C8 de manera paralela
        int[][] resultado = new int[matrizA.length][matrizB[0].length];

        // Implementa la lógica de multiplicación en paralelo aquí.

        return resultado;
    }
}
```

Volver a generar

Envía un mensaje

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version

04:52 p. m.  
30/10/2023

Correo: Eduardo Olay Silis - Outlook SISDIS-4CV12: Tarea 4. Multiplicación Curso: Desarrollo de Sistemas Distribuidos RMI Paralelo Matrices Java

+ Nuevo chat

Hoy

- RMI Paralelo Matrices J
- Java RMI Matrices Multiply
- Java RMI Multiplicación Matric
- Multiplicación Matrices con R

Últimos 7 Días

- Compilar interfaz Java RMI
- Servidor multithread para peti
- Calculadora huella carbono.
- Calcular Producto de Matrices

septiembre

- Comando Minecraft Educati

Actualizar a Plus

Jose Eduardo Olay Silis ...

21°C Mayorm. soleado

chat.openai.com/c/2ecd696c-30ef-4859-aafe-a0f7da90bf7c

En el "Nodo 2":

```
java

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class Nodo2MatrizMultiplicadora extends UnicastRemoteObject implements
    public Nodo2MatrizMultiplicadora() throws RemoteException {
        super();
    }

    @Override
    public int[][] multiplica_matrices(int[][] matrizA, int[][] matrizB) throws RemoteException {
        // Realiza la multiplicación de las matrices C9 a C16 de manera paralela.
        int[][] resultado = new int[matrizA.length][matrizB[0].length];

        // Implementa la lógica de multiplicación en paralelo aquí.

        return resultado;
    }
}
```

1. Inicializa los nodos y registra los objetos remotos:

En cada nodo del servidor, inicializa el objeto de su respectiva implementación y regístralos en el registro RMI.

Volver a generar

Envía un mensaje

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version

04:52 p. m.  
30/10/2023

Correo: Eduardo Olay Silis - Outlook SISDIS-4CV12: Tarea 4. Multiplicación Curso: Desarrollo de Sistemas Distribuidos RMI Paralelo Matrices Java

+ Nuevo chat

Hoy

- RMI Paralelo Matrices J
- Java RMI Matrices Multiply
- Java RMI Multiplicación Matric
- Multiplicación Matrices con R

Últimos 7 Días

- Compilar interfaz Java RMI
- Servidor multithread para peti
- Calculadora huella carbono.
- Calcular Producto de Matrices

septiembre

- Comando Minecraft Educati

Actualizar a Plus

Jose Eduardo Olay Silis ...

21°C Mayorm. soleado

chat.openai.com/c/2ecd696c-30ef-4859-aafe-a0f7da90bf7c

1. Inicializa los nodos y registra los objetos remotos:

En cada nodo del servidor, inicializa el objeto de su respectiva implementación y regístralos en el registro RMI.

```
java

import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;

public class Nodo1Servidor {
    public static void main(String[] args) {
        try {
            MatrizMultiplicadora nodo1MatrizMultiplicadora = new Nodo1MatrizMultiplicadora();

            // Inicia el registro RMI en el puerto 1099 (puedes cambiar el puerto).
            LocateRegistry.createRegistry(1099);

            // Registra el objeto remoto en el registro RMI.
            Naming.rebind("Nodo1MatrizMultiplicadora", nodo1MatrizMultiplicadora);

            System.out.println("Nodo 1 RMI listo.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Volver a generar

Envía un mensaje

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version

04:52 p. m.  
30/10/2023

1. En el cliente, llama a los nodos remotos utilizando threads:  
En el cliente, puedes utilizar threads para llamar a los nodos remotos y obtener las matrices C1 a C8 y C9 a C16 de manera paralela.

```
java
import java.rmi.Naming;

public class Cliente {
    public static void main(String[] args) {
        try {
            MatrizMultiplicadora nodo1 = (MatrizMultiplicadora) Naming.lookup("rmi://localhost:1099/nodo1");
            MatrizMultiplicadora nodo2 = (MatrizMultiplicadora) Naming.lookup("rmi://localhost:1099/nodo2");

            int[][] matrizA = obtenerMatrizA(); // Define tu propia función
            int[][] matrizB = obtenerMatrizB();

            // Crea threads para obtener las matrices C1 a C8 y C9 a C16 en paralelo
            Thread thread1 = new Thread(() -> {
                int[][] resultado1 = nodo1.multiplica_matrices(matrizA, matrizB);
                // Realiza las operaciones necesarias con resultado1
            });
            Thread thread2 = new Thread(() -> {
                int[][] resultado2 = nodo2.multiplica_matrices(matrizA, matrizB);
                // Realiza las operaciones necesarias con resultado2.
            });

            thread1.start();
            thread2.start();

            // Espera a que ambos threads terminen.
            thread1.join();
            thread2.join();

            // Combina los resultados finales si es necesario.
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Este código utiliza threads para llamar a los nodos remotos y obtener las matrices C1 a C8 y C9 a C16 en paralelo. Asegúrate de ajustar la lógica de multiplicación y combinar los resultados según tus necesidades.

```
Thread thread2 = new Thread(() -> {
    int[][] resultado2 = nodo2.multiplica_matrices(matrizA, matrizB);
    // Realiza las operaciones necesarias con resultado2.
});

thread1.start();
thread2.start();

// Espera a que ambos threads terminen.
thread1.join();
thread2.join();

// Combina los resultados finales si es necesario.
} catch (Exception e) {
    e.printStackTrace();
}
}
```

Del código anterior lo que utilice para la codificación del programa fue la ultima parte del cliente el cual mandaba llamar a los nodos y el método remoto utilizando threads. Una vez teniendo esas ideas base comencé por mi propia cuenta a implementar el código de los programas basándome en lo proporcionado por el Chat y lo visto en clase.

Comencé codificando la **InterfazRMI.java** la cual contiene el método `multiplicaMatrices()` la cual recibe como parámetros las 2 matrices y el número de líneas y columnas.

```

import java.rmi.Remote;
import java.rmi.RemoteException;

/*
    Declarar firma de métodos que serán sobreescritos
*/
public interface InterfazRMI extends Remote {
    public double[][] multiplicaMatrices(double[][] A, double[][] B,int N, int M)
throws RemoteException;
}

```

Posteriormente comencé con la codificación del ClienteRMI.java el cual implemente el Restrigy de manera diferente a la vista en clase debido a que mediante la URL no sabía que estaba realizando mal que al momento de ejecutar el programa no funcionaba, debido a esto la IP y el numero de puerto los declare en una variable privada y mediante la interfaz Registry y el nombre del registro referencia los 2 Nodos.

```

public class ClienteRMI {

    private static final String IP = "localhost"; // Puedes cambiar a localhost
    private static final String IP2 = "localhost"; // Puedes cambiar a localhost
    private static final int PUERTO = 8080; //Si cambias aquí el puerto, recuerda
cambiarlo en el servidor
    public static void main(String[] args)
        throws MalformedURLException, RemoteException, NotBoundException,
InterruptedException {
        if (args.length != 2) {
            System.out.println("Uso: java ClienteRMI <N> <M>");
            System.exit(1);
        }

        int N = Integer.parseInt(args[0]);
        int M = Integer.parseInt(args[1]);

        try {

            Registry registry = LocateRegistry.getRegistry(IP, PUERTO);
            Registry registry2 = LocateRegistry.getRegistry(IP2, PUERTO);
            InterfazRMI Nodo1 = (InterfazRMI) registry.lookup("Nodo1"); //Buscar en
el registro...
            InterfazRMI Nodo2 = (InterfazRMI) registry2.lookup("Nodo2"); //Buscar en
el registro...
            // Listo para ocupar los metodos
        }
    }
}

```

Por la parte del servidor (**ServidorRMI.java**) igual cambio un poquito la implementación ya que utilice la interface Remote el cual exporta un objeto remoto de la clase “ClaseRMI” registrándolo en el registro RMI en el puerto 8080, bajo los nombres “Nodo1” y “Nodo2”. Haciendo que el cliente pueda conectarse

a este servidor RMI y realizar llamadas remotas a los métodos del objeto remoto utilizando los nombres "Nodo1" y "Nodo2".

```
import java.rmi.Remote;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;

public class ServidorRMI {
    private static final int PUERTO = 8080; //Si cambias aquí el puerto, recuerda
cambiarlo en el cliente
    public static void main(String[] args) throws Exception {
        Remote remote = UnicastRemoteObject.exportObject(new ClaseRMI(), 0);
        Registry registry = LocateRegistry.createRegistry(PUERTO);
        System.out.println("Servidor escuchando en el puerto " +
String.valueOf(PUERTO));
        registry.bind("Nodo1", remote); // Registrar Nodo
        registry.bind("Nodo2", remote); // Registrar Nodo
    }
}
```

Posteriormente regresando al client Inicializamos las matrices A y B y calculamos su Transpuesta para posteriormente dividir tanto la Matriz A como la matriz BT en 4 partes iguales, quedando submatrices de N/4 renglones por M columnas.

```
// Inicializar las matrices A y B
double[][] A = new double[N][M];
double[][] B = new double[M][N];

System.out.println("Inicializando Matrices..");
for (int i = 0; i < N; i++) {
    for (int j = 0; j < M; j++) {
        A[i][j] = 6 * i - 2 * j;
    }
}

for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        B[i][j] = 8 * i + 3 * j;
    }
}

// Obtener la matriz traspuesta BT de la matriz B
double[][] BT = new double[N][M];
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        BT[j][i] = B[i][j];
    }
}
```

```

    }

    // Dividimos las matrices A y BT en N/4 renglones por M columnas
    System.out.println("Dividiendo Matrices....");

    double[][] A1 = separa_matriz(A, 0);
    double[][] A2 = separa_matriz(A, (N / 4));
    double[][] A3 = separa_matriz(A, (N / 4) * 2);
    double[][] A4 = separa_matriz(A, (N / 4) * 3);

    double[][] BT1 = separa_matriz(BT, 0);
    double[][] BT2 = separa_matriz(BT, (N / 4));
    double[][] BT3 = separa_matriz(BT, (N / 4) * 2);
    double[][] BT4 = separa_matriz(BT, (N / 4) * 3);

```

Una vez hecho esto, en 2 threads comenzamos a multiplicar cada uno de los renglones de la matriz A y BT invocando la método remoto multiplicaMatrices() dando como resultado una sección de la matriz resultante C la cual mediante el método acomoda\_matriz() se va colocando en su renglón y columna correspondiente para que de esta manera se vaya creando la matriz resultante C.

```

// Utilizamos hilos para comenzar a calcular las multiplicaciones mediante
double[][] C = new double[N][N];

System.out.println("\nCalculando la multiplicacion de Matrices....");
// threads y el metodo remoto
Thread thread1 = new Thread(() -> {
    try {
        double[][] C1 = Nodo1.multiplicaMatrices(A1, BT1,N,M);
        double[][] C2 = Nodo1.multiplicaMatrices(A1, BT2,N,M);
        double[][] C3 = Nodo1.multiplicaMatrices(A1, BT3,N,M);
        double[][] C4 = Nodo1.multiplicaMatrices(A1, BT4,N,M);
        double[][] C5 = Nodo1.multiplicaMatrices(A2, BT1,N,M);
        double[][] C6 = Nodo1.multiplicaMatrices(A2, BT2,N,M);
        double[][] C7 = Nodo1.multiplicaMatrices(A2, BT3,N,M);
        double[][] C8 = Nodo1.multiplicaMatrices(A2, BT4,N,M);

        // Obtenemos la matriz C de las submatrices obtenidas por los
métodos remotos
        acomoda_matriz(C, C1, 0, 0,N);
        acomoda_matriz(C, C2, 0, (N/4),N);
        acomoda_matriz(C, C3,0,(N/4)*2,N);
        acomoda_matriz(C, C4,0, (N/4)*3,N);
        acomoda_matriz(C, C5, (N/4),0,N );
        acomoda_matriz(C, C6, (N/4), (N/4),N );
        acomoda_matriz(C, C7, (N/4), (N/4)*2,N);
        acomoda_matriz(C, C8, (N/4), (N/4)*3,N);
    } catch (RemoteException e) {

```

```

        // Manejo de excepciones específico para el thread 1.
        e.printStackTrace();
    }
});

Thread thread2 = new Thread(() -> {
    try {
        double[][] C9 = Nodo2.multiplicaMatrices(A3, BT1,N,M);
        double[][] C10 = Nodo2.multiplicaMatrices(A3, BT2,N,M);
        double[][] C11 = Nodo2.multiplicaMatrices(A3, BT3,N,M);
        double[][] C12 = Nodo2.multiplicaMatrices(A3, BT4,N,M);
        double[][] C13 = Nodo2.multiplicaMatrices(A4, BT1,N,M);
        double[][] C14 = Nodo2.multiplicaMatrices(A4, BT2,N,M);
        double[][] C15 = Nodo2.multiplicaMatrices(A4, BT3,N,M);
        double[][] C16 = Nodo2.multiplicaMatrices(A4, BT4,N,M);

        // Obtenemos la matriz C de las submatrices obtenidas por los
métodos remotos
        acomoda_matriz(C, C9, (N/4)*2, 0,N);
        acomoda_matriz(C, C10, (N/4)*2, (N/4),N);
        acomoda_matriz(C, C11, (N/4)*2, (N/4)*2,N);
        acomoda_matriz(C, C12, (N/4)*2, (N/4)*3,N);
        acomoda_matriz(C, C13, (N/4)*3 ,0,N );
        acomoda_matriz(C, C14, (N/4)*3, (N/4),N);
        acomoda_matriz(C, C15, (N/4)*3, (N/4)*2,N);
        acomoda_matriz(C, C16, (N/4)*3, (N/4)*3,N);;
    } catch (RemoteException e) {
        // Manejo de excepciones específico para el thread 2.
        e.printStackTrace();
    }
});

thread1.start();
thread2.start();

thread1.join();
thread2.join();

```

Al final se calcula el checksum de la matriz C y para el primer caso que se menciona dentro de la práctica se imprimen las matrices resultantes.

```

System.out.println("\nObteniendo Cheksum....");
//Se calcula el checksum de la matriz C
double checksum =0;
for(int i=0;i<N;i++){
    for(int j=0;j<N;j++){

```

```
        checksum += C[i][j];
    }
}
// Mostrar el resultado
if(N==8 && M==4){
    System.out.println("Matriz A:");
    imprimirMatriz(A);
    System.out.println("\nMatriz B:");
    imprimirMatriz(B);
    System.out.println("\nMatriz c:");
    imprimirMatriz(C);
    System.out.println("\nEl Checksum del producto de matrices es:
"+checksum);
} else{
    System.out.println("\nEl Checksum del producto de matrices es:
"+checksum);
}
} catch (Exception e) {
    e.printStackTrace();
}
}
```

Por último creamos la claseRMI la cual contiene el método remoto multiplicaMatrices() en este caso como la clase implementa directamente la Interfaz no es necesaria agregar la SuperClase debido a la implementación anterior realizada tanto en el cliente como en el Servidor por lo que simplemente se codifica de la siguiente manera.

```
import java.rmi.RemoteException;

public class ClaseRMI implements InterfazRMI {

    public double[][] multiplicaMatrices(double[][] A, double[][] B, int N,int M)
throws RemoteException{
    double[][]C=new double[N/4][N/4];
    for(int i=0;i<N/4;i++)
        for(int j=0; j<N/4;j++)
            for(int k=0;k<M;k++)
                C[i][j] += A[i][k] * B[j][k];

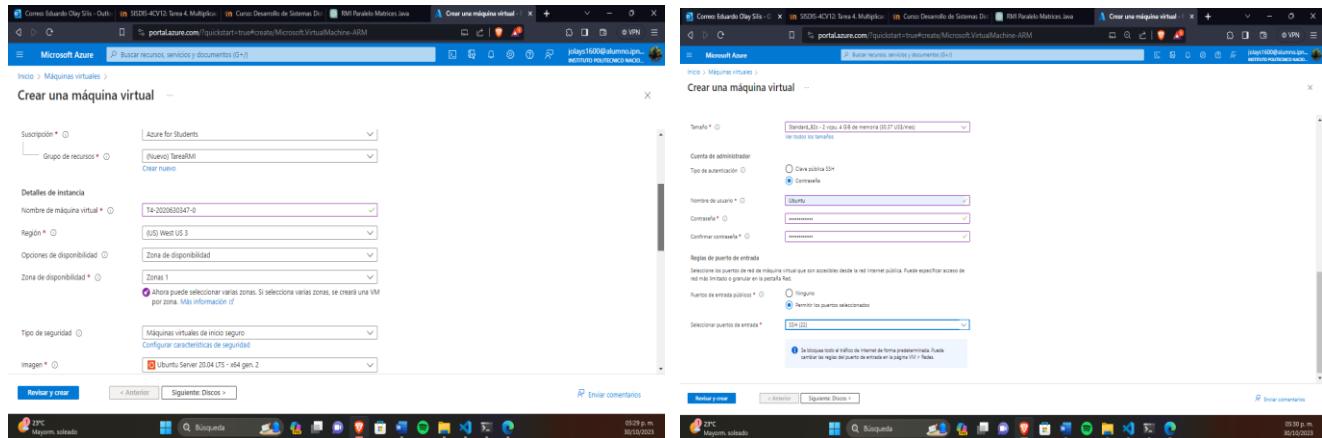
    return C;
}
}
```

Y ya con estos programas queda listo nuestro sistema.

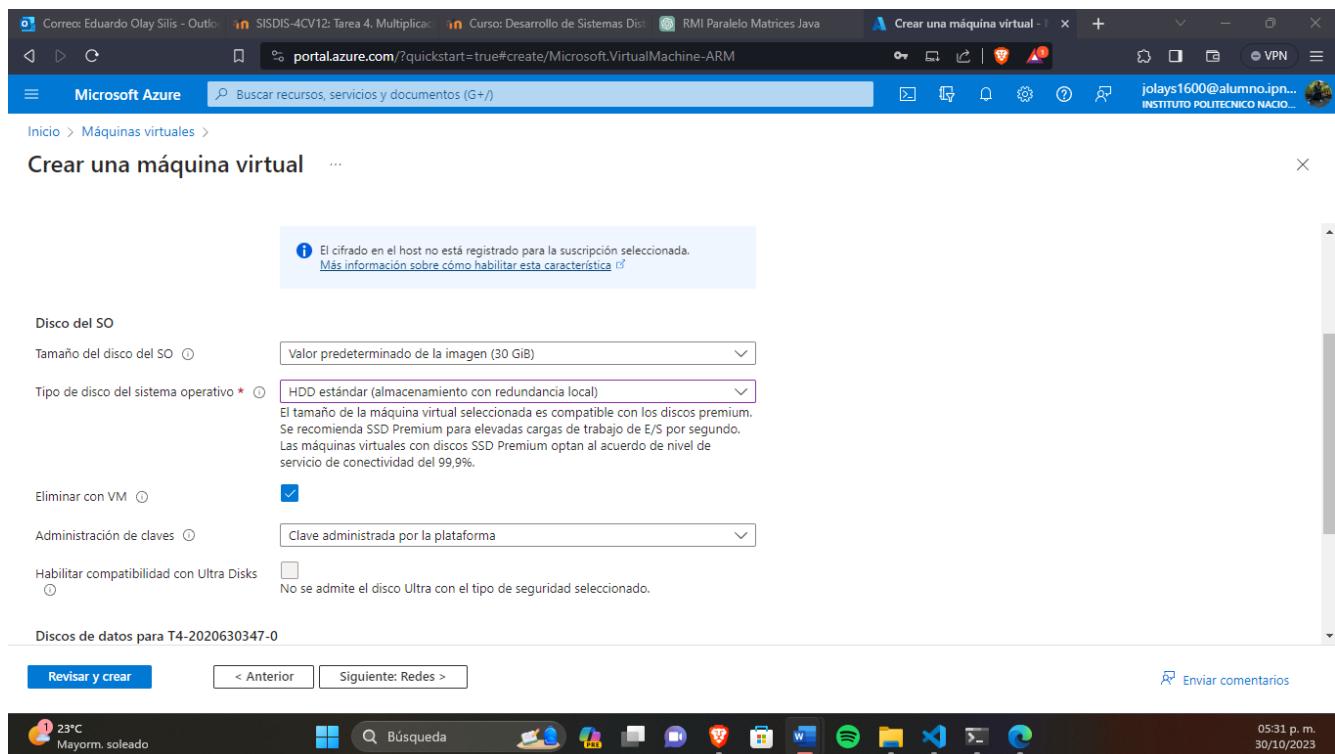
## Fase 2

Una vez teniendo listo el programa comenzamos el proceso de creación de las máquinas virtuales los cuales simularán los nodos. A continuación, se pondrá el proceso de creación de la máquina virtual que simulará el Nodo 0.

Creamos un grupo de recursos para que los 3 nodos estén en el mismo, colocamos su nombre, la Imagen la cual será Ubutnu 20,el tipo de memoria el cual será de 4GB debido a la matriz grande del segundo caso que habrá que calcular, el usuario y contraseña incluyendo el puerto de entrada SSH.



Seleccionamos el HHD estándar



Comprobamos que se haya generado por default la Ip Pública

Correo: Eduardo Olay Silis - Outlook | SISDIS-4CV12: Tarea 4. Multiplicacion | Curso: Desarrollo de Sistemas Distribuidos | RMI Paralelo Matrices Java | Crear una máquina virtual - | +

Microsoft Azure | Buscar recursos, servicios y documentos (G+) | portal.azure.com/?quickstart=true#create/Microsoft.VirtualMachine-ARM

Inicio > Máquinas virtuales > Crear una máquina virtual ...

Interfaz de red

Al crear una máquina virtual, se crea una interfaz de red automáticamente.

Red virtual \* ⓘ (nuevo) T4-2020630347-0-vnet  
Crear nuevo

Subred \* ⓘ (nuevo) default (10.2.0.0/24)

IP pública \* ⓘ (nuevo) T4-2020630347-0-ip  
Crear

Grupo de seguridad de red de NIC ⓘ  Ninguno  Básico  Opciones avanzadas

Puertos de entrada públicos \* ⓘ  Ninguno  Permitir los puertos seleccionados

Seleccionar puertos de entrada \* ⓘ SSH (22)

**Revisar y crear** < Anterior Siguiente: Administración > Enviar comentarios

Correo: Eduardo Olay Silis - Outlook | SISDIS-4CV12: Tarea 4. Multiplicacion | Curso: Desarrollo de Sistemas Distribuidos | RMI Paralelo Matrices Java | Crear una máquina virtual - | +

Microsoft Azure | Buscar recursos, servicios y documentos (G+) | portal.azure.com/?quickstart=true#create/Microsoft.VirtualMachine-ARM

Inicio > Máquinas virtuales > Crear una máquina virtual ...

Datos básicos Discos Redes Administración **Supervisión** Opciones avanzadas Etiquetas Revisar y crear

Configure las opciones de supervisión de la máquina virtual.

Alertas  Habilitar reglas de alerta recomendadas

Diagnóstico

Diagnósticos de arranque ⓘ  Habilitar con la cuenta de almacenamiento administrada (recomendado)  Habilitar con la cuenta de almacenamiento personalizada  Deshabilitar

Habilitar diagnósticos del SO invitado ⓘ

Estado

Habilitar supervisión de estado de la aplicación ⓘ

**Revisar y crear** < Anterior Siguiente: Opciones avanzadas > Enviar comentarios

Y le damos en revisar y crear, para que una vez haya finalizado la revisor le demos en crear y esta se pueda visualizar dentro del grupo de recursos.

Mostrando de 1 a 1 de 1 registros.

Nombre	Tipo	Suscripción	Grupo de recursos	Ubicación	Estado	Sistema operativo	Tamaño	Dirección
T4-2020630347-0	Máquina virtual	Azure for Students	TAREARMI	West US 3	Creando	Linux	Standard_B2s	20.168.1

< Anterior Página 1 de 1 Siguiente >

Enviar comentarios

Una vez observada la Máquina virtual creada le damos en la configuración de red y creamos 2 nuevas reglas de entrada, la primera una del puerto 8080 ya que en el código de nuestra aplicación es el que colocamos y otra regla ICMP para que se pueda realizar una conexión sin problema entre las máquinas virtuales que creamos.

Prioridad	Nombre	Puerto	Protocolo	Origen
300	SSH	22	TCP	Cualquiera
310	Port8080	8080	Cualquiera	Cualquiera
320	ICMPRule	Cualquiera	ICMP	Cualquiera
65000	AllowVnetinBound	Cualquiera	Cualquiera	VirtualNetwork
65001	AllowAzureLoadBalancing	Cualquiera	Cualquiera	AzureLoadBalancing
65500	DenyAllInBound	Cualquiera	Cualquiera	Cualquiera

< Página 1 de 1 >

Enviar comentarios

Dicho proceso se repetirá para el Nodo 1 y 2 el quedando al final el grupo de máquinas virtuales de la siguiente manera.

Filtrar por cualquier cosa... Suscripción es igual a todo Tipo es igual a todo Grupo de recursos es igual a todo Ubicación es igual a todo Agregar filtro

Mostrando de 1 a 3 de 3 registros.

Nombre ↑↓	Tipo ↑↓	Suscripción ↑↓	Grupo de recursos ↑↓	Ubicación ↑↓	Estado ↑↓	Sistema operativo ↑↓	Tamaño ↑↓	Dirección
T4-2020630347-0	Máquina virtual	Azure for Students	TareaRMI	West US 3	En ejecución	Linux	Standard_B2s	20.168.1
T4-2020630347-1	Máquina virtual	Azure for Students	TareaRMI	South Central US	En ejecución	Linux	Standard_B2s	40.124.1
T4-2020630347-2	Máquina virtual	Azure for Students	TareaRMI	East US	En ejecución	Linux	Standard_B2s	20.51.25

< Anterior Página 1 de 1 Siguiente >

https://portal.azure.com/?quickstart=true#@correo.ipn.mx/resource/subscriptions/44ecfc53-60b2-410f-b59c-a9e26531219d/resourceGroups/TareaRMI/providers/Microsoft.Compute/virtualMachines/T4-2020630347-2

Enviar comentarios 05:47 p.m.  
30/10/2023

20°C Soleado

Ahora procedemos a instalar el jdk en cada una de las máquinas virtuales mediante los siguientes comandos:

```
sudo add-apt-repository ppa:openjdk-r/ppa
```

```
sudo apt-get update
```

```
sudo apt install openjdk-8-jdk-headless
```

Página 16 de 16 1600 palabras Español (Méjico) Predicciones de texto: activado Accesibilidad: es necesario investigar

16 : 35 : 14 : 11 : 10 : 9 : 8 : 7 : 6 : 5 : 4 : 3 : 2 : 1

Autoguardado Tareas

Archivo Inicio Insertar Dibujar Diseño Disposición Referencias Correspondencia Revisar Vista Ayuda

Ubuntu@T4-2020630347-0:~

```
Pegar to provide /usr/bin/jstated (jstated) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jmap t
o provide /usr/bin/jmap (jmap) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jdb to
provide /usr/bin/jdb (jdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/serial
ver to provide /usr/bin/serialver (serialver) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jfr to
provide /usr/bin/jfr (jfr) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/wsgen
to provide /usr/bin/wsgen (wsgen) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jcmd t
o provide /usr/bin/jcmd (jcmd) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jarsig
ner to provide /usr/bin/jarsigner (jarsigner) in auto mode
Processing triggers for libgcj-bin (2.31-0ubuntu9.12) ...
Processing triggers for systemd (245.4-4ubuntu3.22) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for ca-certificates (20230311ubuntu0.20.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...

done.
done.
Ubuntu@T4-2020630347-0:~$ java -version
openjdk version "1.8.0_382"
OpenJDK Runtime Environment (build 1.8.0_382-8u382-ga-1~20.04.1-b05)
OpenJDK 64-Bit Server VM (build 25.382-b05, mixed mode)
Ubuntu@T4-2020630347-0:~$
```

Ubuntu@T4-2020630347-1:~

```
r) in auto mode
Processing triggers for libc-bin (2.31-0ubuntu9.12) ...
Processing triggers for systemd (245.4-4ubuntu3.22) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for ca-certificates (20230311ubuntu0.20.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...

done.
done.
Ubuntu@T4-2020630347-1:~$ java -version
openjdk version "1.8.0_382"
OpenJDK Runtime Environment (build 1.8.0_382-8u382-ga-1~20.04.1-b05)
OpenJDK 64-Bit Server VM (build 25.382-b05, mixed mode)
Ubuntu@T4-2020630347-1:~$
```

Ubuntu@T4-2020630347-2:~

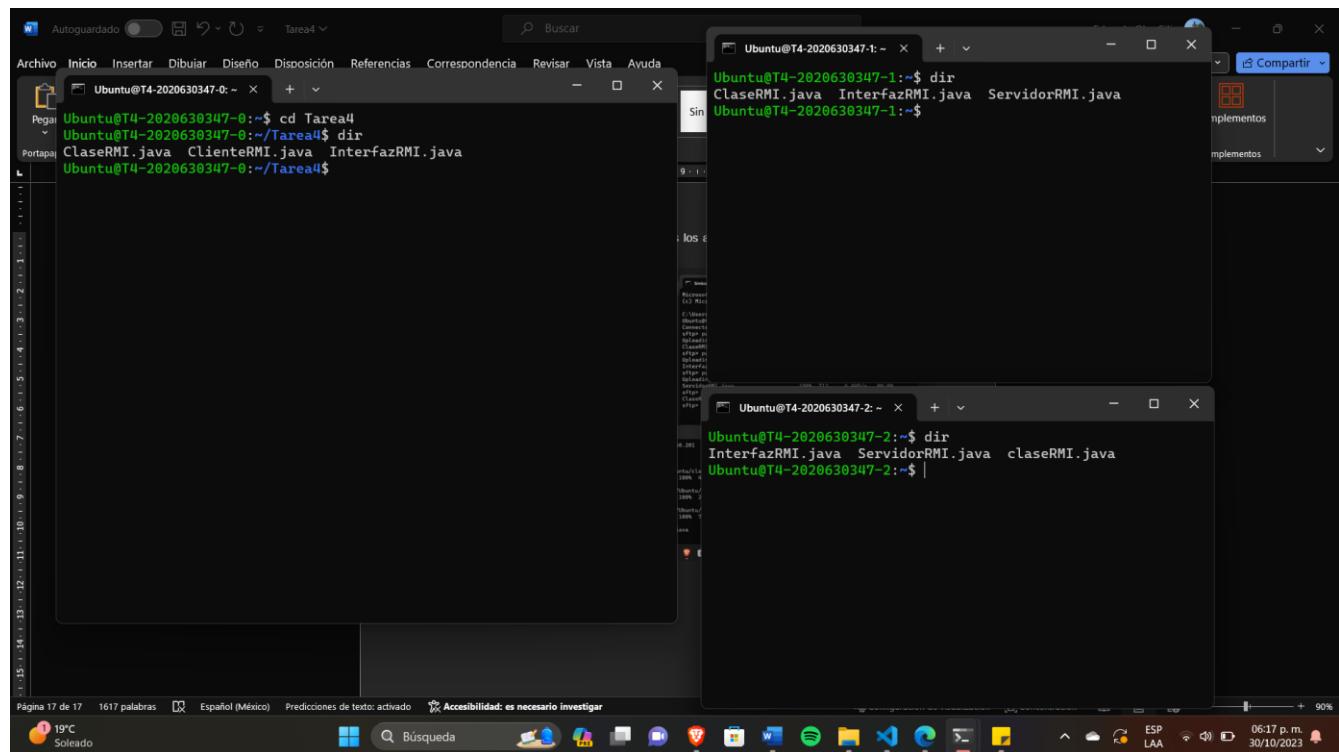
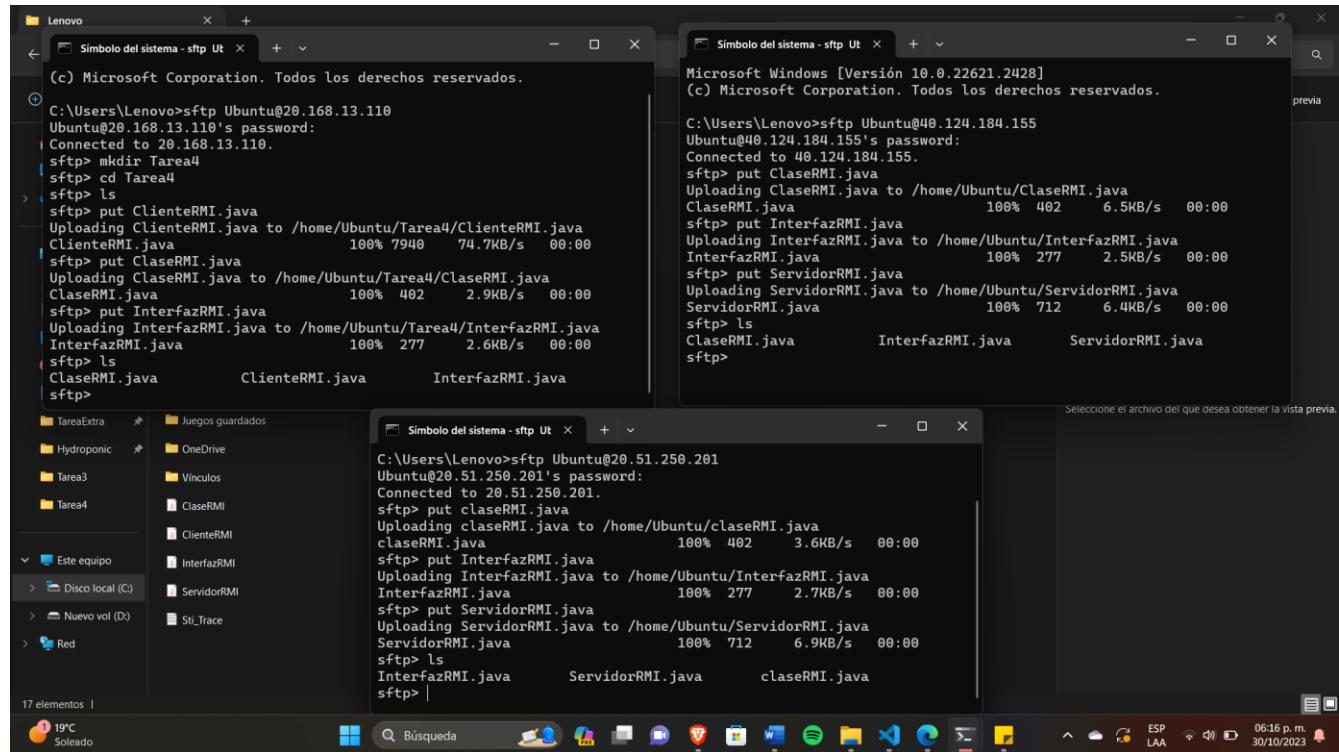
```
Processing triggers for systemd (245.4-4ubuntu3.22) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for ca-certificates (20230311ubuntu0.20.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...

done.
done.
Ubuntu@T4-2020630347-2:~$ java -version
openjdk version "1.8.0_382"
OpenJDK Runtime Environment (build 1.8.0_382-8u382-ga-1~20.04.1-b05)
OpenJDK 64-Bit Server VM (build 25.382-b05, mixed mode)
Ubuntu@T4-2020630347-2:~$
```

05:47 p.m.  
30/10/2023

20°C Soleado

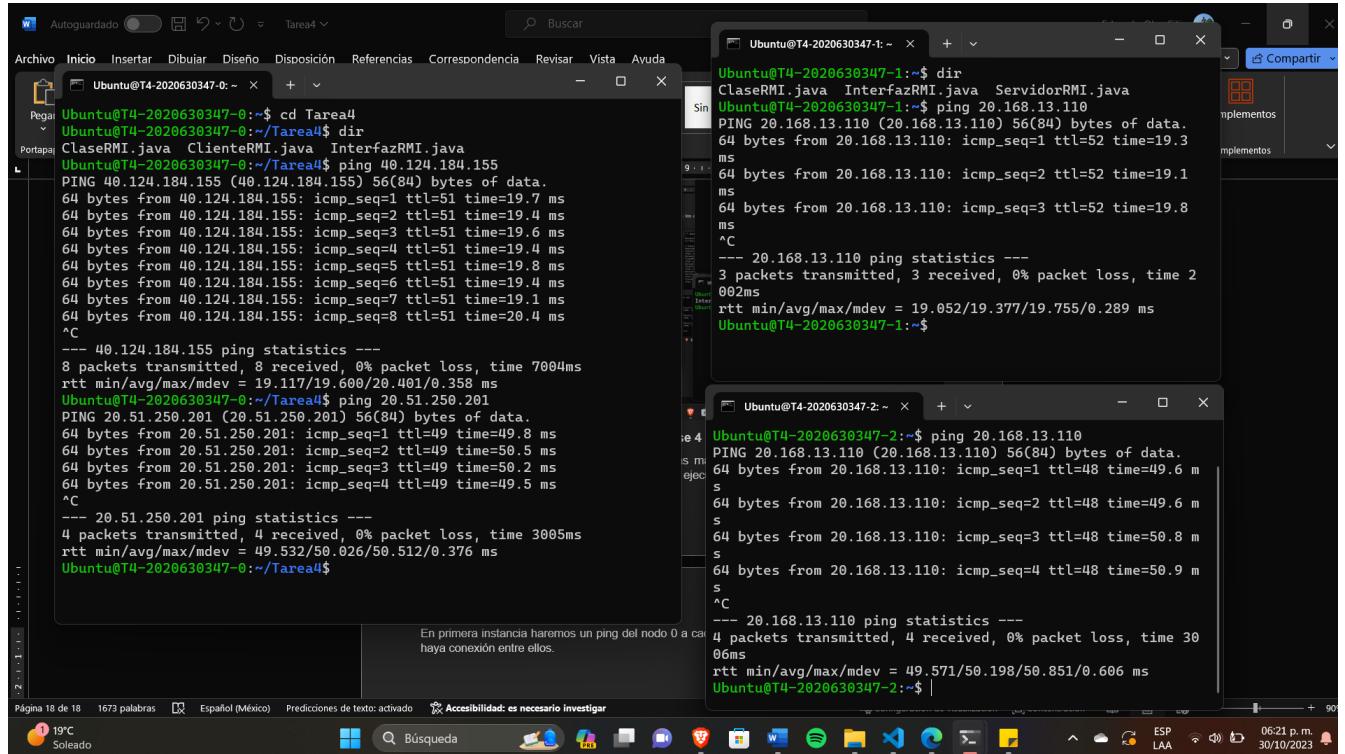
Por último, mediante el protocolo sftp le pasamos los archivos necesarios a compilar a la máquina virtual.



## Fase 4

Ya que pasamos los archivos a cada una de las máquinas virtuales mediante el Protocolo sftp, procedemos a realizar el proceso de compilación y ejecución de los programas.

En primera instancia haremos un ping del nodo 0 a cada uno de los nodos 1 y 2 para poder ver que haya conexión entre ellos.



```
Ubuntu@T4-2020630347-0:~$ cd Tarea4
ClaseRMI.java ClienteRMI.java InterfazRMI.java
Ubuntu@T4-2020630347-0:~/Tarea4$ ping 40.124.184.155
PING 40.124.184.155 (40.124.184.155) 56(84) bytes of data.
64 bytes from 40.124.184.155: icmp_seq=1 ttl=51 time=19.7 ms
64 bytes from 40.124.184.155: icmp_seq=2 ttl=51 time=19.4 ms
64 bytes from 40.124.184.155: icmp_seq=3 ttl=51 time=19.6 ms
64 bytes from 40.124.184.155: icmp_seq=4 ttl=51 time=19.4 ms
64 bytes from 40.124.184.155: icmp_seq=5 ttl=51 time=19.8 ms
64 bytes from 40.124.184.155: icmp_seq=6 ttl=51 time=19.4 ms
64 bytes from 40.124.184.155: icmp_seq=7 ttl=51 time=19.1 ms
64 bytes from 40.124.184.155: icmp_seq=8 ttl=51 time=20.4 ms
^C
--- 40.124.184.155 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7004ms
rtt min/avg/max/mdev = 19.117/19.600/20.401/0.358 ms
Ubuntu@T4-2020630347-0:~/Tarea4$ ping 20.51.250.201
PING 20.51.250.201 (20.51.250.201) 56(84) bytes of data.
64 bytes from 20.51.250.201: icmp_seq=1 ttl=49 time=49.8 ms
64 bytes from 20.51.250.201: icmp_seq=2 ttl=49 time=50.5 ms
64 bytes from 20.51.250.201: icmp_seq=3 ttl=49 time=50.2 ms
64 bytes from 20.51.250.201: icmp_seq=4 ttl=49 time=49.5 ms
^C
--- 20.51.250.201 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 49.532/50.026/50.512/0.376 ms
Ubuntu@T4-2020630347-0:~/Tarea4$
```

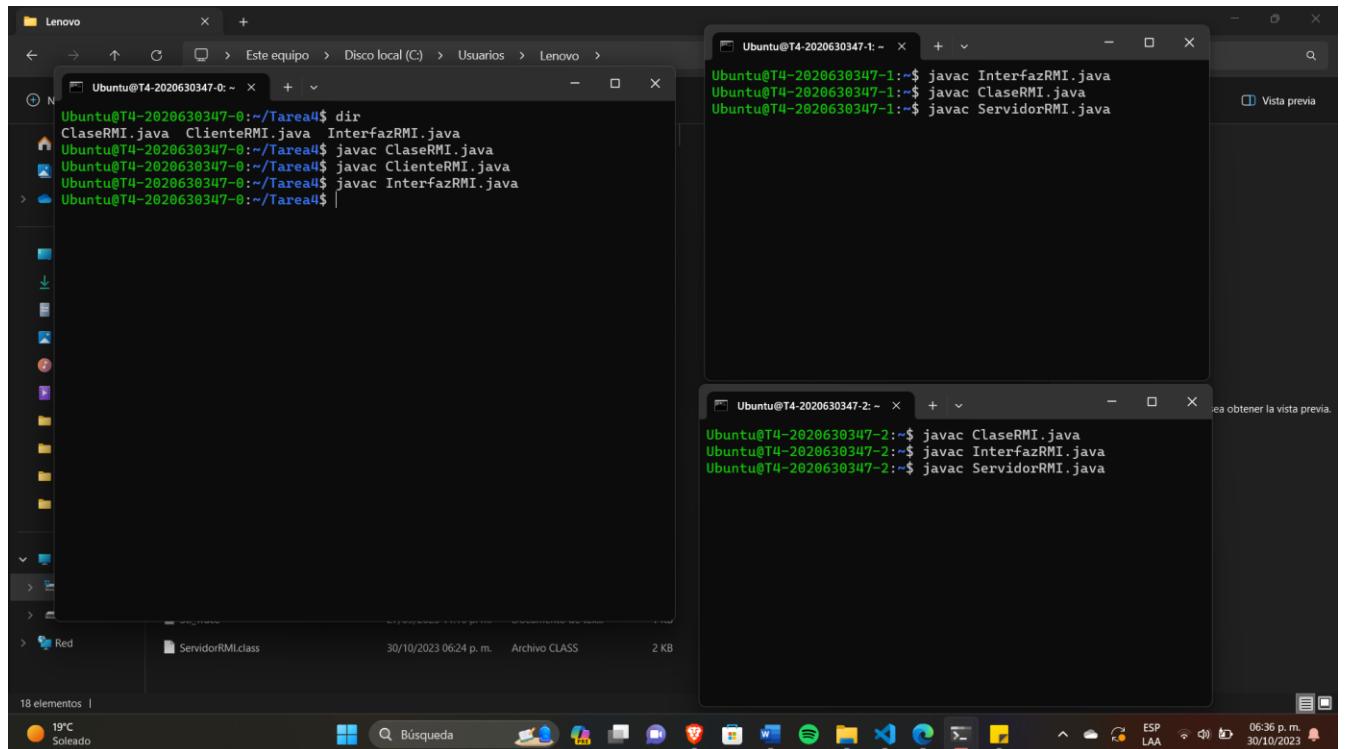
  

```
Ubuntu@T4-2020630347-1:~$ dir
ClaseRMI.java InterfazRMI.java ServidorRMI.java
Ubuntu@T4-2020630347-1:~$ ping 20.168.13.110
PING 20.168.13.110 (20.168.13.110) 56(84) bytes of data.
64 bytes from 20.168.13.110: icmp_seq=1 ttl=52 time=19.3 ms
64 bytes from 20.168.13.110: icmp_seq=2 ttl=52 time=19.1 ms
64 bytes from 20.168.13.110: icmp_seq=3 ttl=52 time=19.8 ms
^C
--- 20.168.13.110 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2 002ms
rtt min/avg/max/mdev = 19.052/19.377/19.755/0.289 ms
Ubuntu@T4-2020630347-1:~$
```

```
Ubuntu@T4-2020630347-2:~$ ping 20.168.13.110
PING 20.168.13.110 (20.168.13.110) 56(84) bytes of data.
64 bytes from 20.168.13.110: icmp_seq=1 ttl=48 time=49.6 ms
64 bytes from 20.168.13.110: icmp_seq=2 ttl=48 time=49.6 ms
64 bytes from 20.168.13.110: icmp_seq=3 ttl=48 time=50.8 ms
64 bytes from 20.168.13.110: icmp_seq=4 ttl=48 time=50.9 ms
^C
--- 20.168.13.110 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 30 06ms
rtt min/avg/max/mdev = 49.571/50.198/50.851/0.606 ms
Ubuntu@T4-2020630347-2:~$
```

Ya verificando que hay conexión entre las 3 máquinas comenzamos a compilar cada uno de los programas correspondientes.



```
Ubuntu@T4-2020630347-0:~/Tarea4$ dir
ClaseRMI.java ClienteRMI.java InterfazRMI.java
Ubuntu@T4-2020630347-0:~/Tarea4$ javac ClaseRMI.java
Ubuntu@T4-2020630347-0:~/Tarea4$ javac ClienteRMI.java
Ubuntu@T4-2020630347-0:~/Tarea4$ javac InterfazRMI.java
Ubuntu@T4-2020630347-0:~/Tarea4$
```

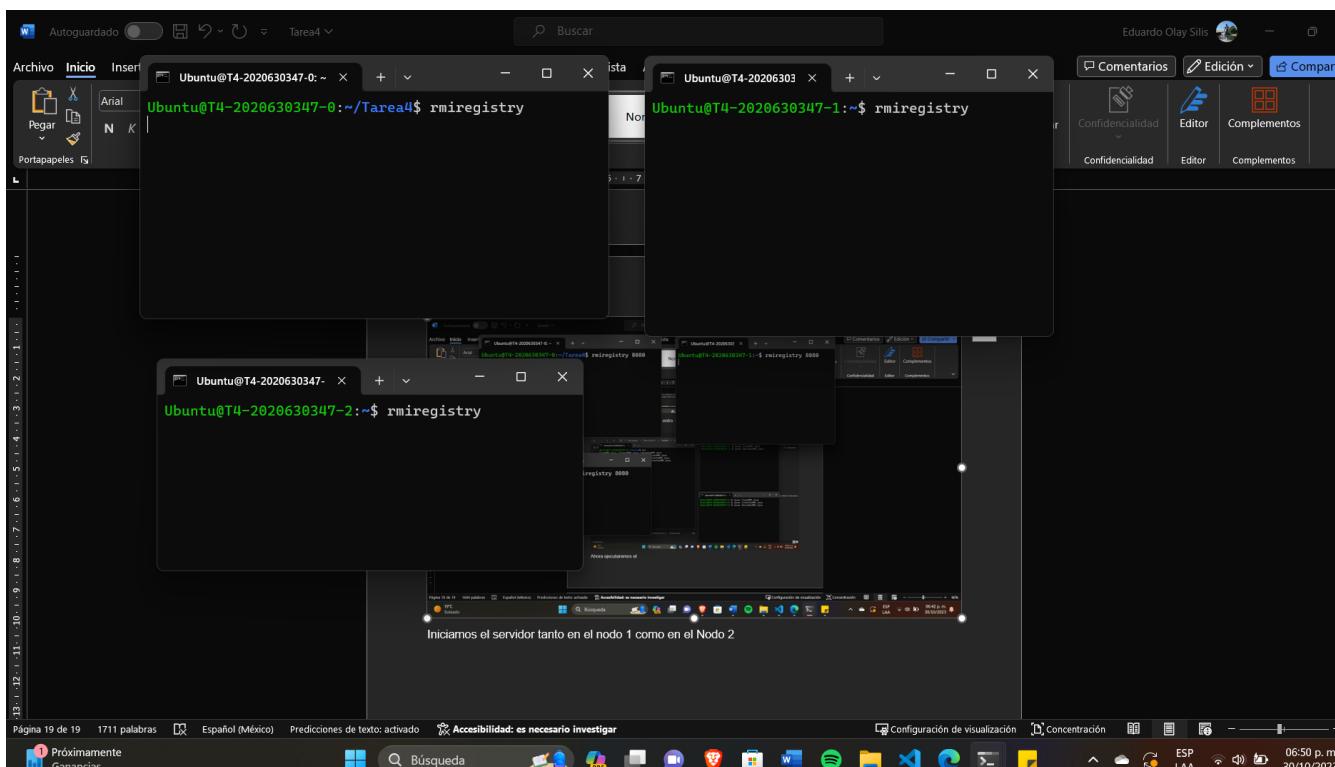
  

```
Ubuntu@T4-2020630347-1:~$ javac InterfazRMI.java
Ubuntu@T4-2020630347-1:~$ javac ClaseRMI.java
Ubuntu@T4-2020630347-1:~$ javac ServidorRMI.java
```

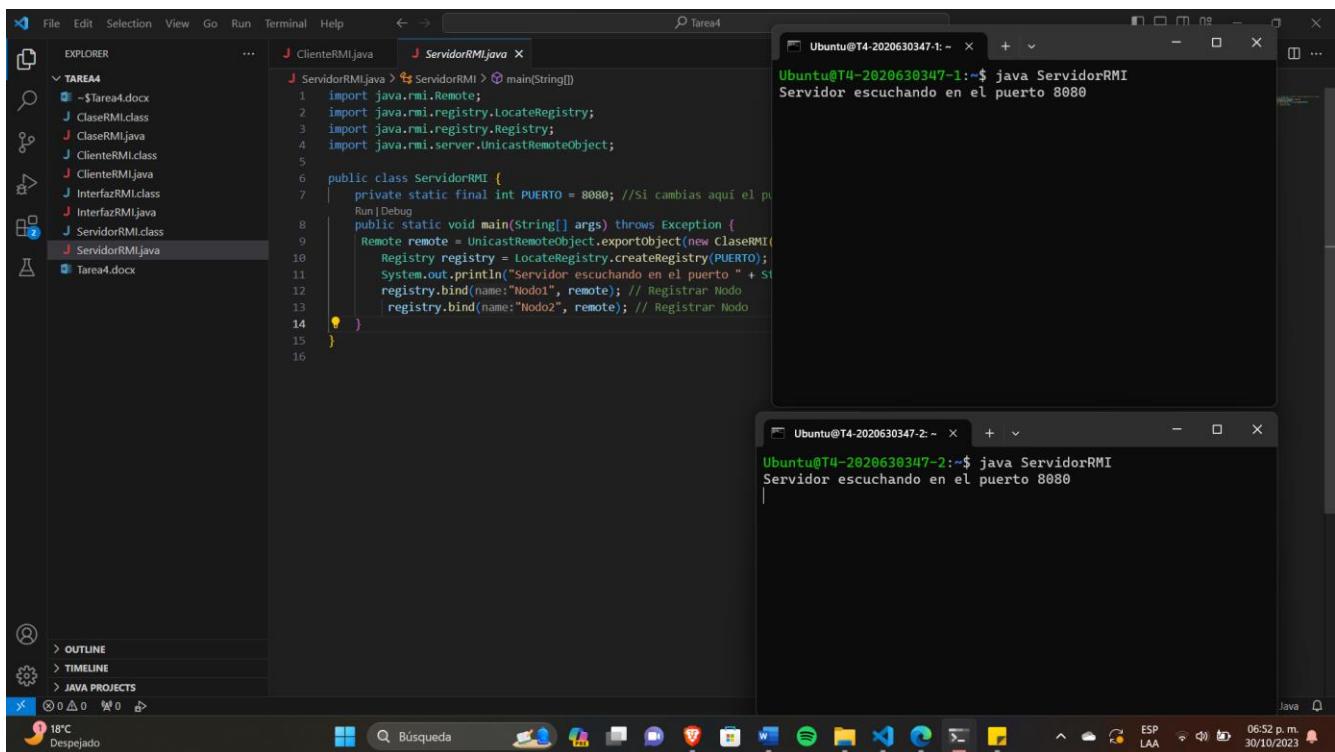
  

```
Ubuntu@T4-2020630347-2:~$ javac ClaseRMI.java
Ubuntu@T4-2020630347-2:~$ javac InterfazRMI.java
Ubuntu@T4-2020630347-2:~$ javac ServidorRMI.java
```

Ahora ejecutaremos el comando rmregistry para iniciararlo



Iniciamos el servidor tanto en el nodo 1 como en el Nodo 2



Ahora inicializamos el cliente con los primeros casos que es con **N=8** y **M=4** dando como resultado lo siguiente.

The screenshot shows a Linux desktop environment with several terminal windows open. The main window is titled 'Lenovo' and contains the output of the 'ClienteRMI' client. It shows the initialization of matrices, division of matrices, calculation of matrix multiplication, and obtaining checksums. The final output is 'El Checksum del producto de matrices es: 98560.0'. Other terminal windows show the server side ('ServidorRMI') listening on port 8080.

```
Ubuntu@T4-2020630347-0:~/Tarea4$ java ClienteRMI 8 4
Inicializando Matrices..
Dividiendo Matrices...
Calculando la multiplicacion de Matrices...
Obteniendo Checksum...
Matriz A:
0.0 -2.0 -4.0 -6.0
6.0 4.0 2.0 0.0
12.0 10.0 8.0 6.0
18.0 16.0 14.0 12.0
24.0 22.0 20.0 18.0
30.0 28.0 26.0 24.0
36.0 34.0 32.0 30.0
42.0 40.0 38.0 36.0

Matriz B:
0.0 3.0 6.0 9.0 12.0 15.0 18.0 21.0
8.0 11.0 14.0 17.0 20.0 23.0 26.0 29.0
16.0 19.0 22.0 25.0 28.0 31.0 34.0 37.0
24.0 27.0 30.0 33.0 36.0 39.0 42.0 45.0

Matriz c:
-224.0 -260.0 -296.0 -332.0 -368.0 -404.0 -440.0 -476.0
64.0 100.0 136.0 172.0 208.0 244.0 280.0 316.0
352.0 460.0 568.0 676.0 784.0 892.0 1000.0 1108.0
648.0 820.0 1000.0 1180.0 1360.0 1540.0 1720.0 1900.0
928.0 1180.0 1432.0 1684.0 1936.0 2188.0 2440.0 2692.0
1216.0 1540.0 1864.0 2188.0 2512.0 2836.0 3160.0 3484.0
1564.0 1980.0 2296.0 2692.0 3088.0 3484.0 3880.0 4276.0
1792.0 2260.0 2728.0 3196.0 3664.0 4132.0 4600.0 5068.0

El Checksum del producto de matrices es: 98560.0
Ubuntu@T4-2020630347-0:~/Tarea4$
```

```
Ubuntu@T4-2020630347-1:~$ java ServidorRMI
Servidor escuchando en el puerto 8080
```

```
Ubuntu@T4-2020630347-2:~$ java ServidorRMI
Servidor escuchando en el puerto 8080
```

Ahora procedemos a ejecutar el segundo caso donde N=4000 y M=800 saliendo los resultados siguientes:

The screenshot shows a Windows desktop environment with two terminal windows. The left window is a PowerShell session where the 'ClienteRMI' client is run with parameters 4000 and 800. The output is identical to the Linux version, showing the process of initializing, dividing, calculating, and obtaining the checksum of the matrix multiplication result. The right window shows the 'ServidorRMI' server running on port 8080.

```
Ubuntu@T4-2020630347-0:~/Tarea4$ java ClienteRMI 4000 800
Inicializando Matrices..
Dividiendo Matrices...
Calculando la multiplicacion de Matrices...
Obteniendo Checksum...
El Checksum del producto de matrices es: 1.3069654911999007E18
Ubuntu@T4-2020630347-0:~/Tarea4$
```

```
Ubuntu@T4-2020630347-1:~$ java ServidorRMI
Servidor escuchando en el puerto 8080
```

```
Ubuntu@T4-2020630347-2:~$ java ServidorRMI
Servidor escuchando en el puerto 8080
```

## Conclusión

Durante el desarrollo de esta práctica pude comprender más el tema de la transferencia de objetos distribuidos mediante Java RMI el cual es un API el cual invoca métodos remotos. Dicho método que utilizamos fue el de multiplicaMatrices por lo que al momento de desarrollarlo e implementarlo dentro del código, pude comprender un poco más acerca de este y como es que se instancia dentro de un programa.

Algo importante en el desarrollo de esta práctica fue la división de matrices la cual se realizó renglón por renglón reforzando lo visto en la clase de “jerarquía de memoria” utilizando esta implementación pude comprender mejor como es que se iban realizando dichas multiplicaciones y los resultados de estas como irlos acomodando, dependiendo el renglón y la columna.

Considero que esta implementación es un poco más sencilla que la realizada en la tarea 3 el cual fue por paso de mensajes ya que simplemente se puede mandar llamar al método remoto y puedes dejar de preocuparte si se pasó o no bien el mensaje ya que simplemente se manda llamar al método.

Estas y otras herramientas sé que me serán de gran ayuda em un futuro y seguirán complementando mi conocimiento y mis habilidades como programador.