



Instituto Politécnico Nacional
Escuela Superior de Cómputo

Desarrollo de sistemas distribuidos

Reporte Tarea 5.
"Multiplicación de matrices utilizando objetos distribuidos"

Grupo: 4CV13

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

Marzo, 2022.

Desarrollo.

Para esta tarea desarrollamos un otra versión del programa que multiplica matrices cuadradas, en este caso se ocupó la tecnología RMI (Remote Method Invocation), para hacer uso de métodos remotos que viven en otra computadora, esto es muy útil ya que nos permite enviar varios tipos de dato sin problemas de comunicación como es el caso de los sockets.

1. Servidor RMI

El ServidorRMI utiliza el paquete `java.rmi.Naming`, esta nos permite llamar a método `rebind`, este recibirá la ip para registrar el objeto sobre una instancia del objeto que se está siendo exportado, en este caso es `ClaseRMI`, esta instancia contendrá los métodos remotos. Este se ejecutará en un nodo servidor con una máquina virtual en azure.

```
import java.rmi.Naming;
public class ServidorRMI {
    public static void main(String[] args)
        throws Exception {
        String url = "rmi://localhost/prueba";
        ClaseRMI obj = new ClaseRMI();
        Naming.rebind(url, obj);
    }
}
```

2. ClienteRMI

En el ClienteRMI se continua haciendo uso del paquete `java.rmi.Naming`, del cual se invocará el método `lookup` para obtener la referencia del objeto remoto prueba, y ahora, para que se pueda invocar el objeto remoto registrado se debe de hacer uso nuevamente del método `lookup`. Ahora bien, el metodo cliente lo primero que hace es inicializar las matrices, una vez hecho eso pues por simplicidad las imprimimos antes de mandar a hacer la transpuesta de B ya que usamos la misma matriz para de alguna manera “optimizar” memoria y reducir código que en la práctica 3 nos vimos en la necesidad de hacer copias, una vez finalizando eso ejecutamos la clase Cliente, subclase de la clase Thread ya que en la tarea se pide que una vez iniciado el cliente se tienen que crear hilos para la topología estrella y lo que hacemos es básicamente indicar el número de nodo con su IP publica para continuar con la separación de las matrices y así realizar la multiplicación correspondiente con cada servidor/nodo y guardando los respectivos resultados en la matriz final de nombre C, esta vez finalizados estos procesos, mandamos a imprimir la matriz final ya sea para el $N = 8$ con su checksum o solo el checksum para $N = 4000$, cabe aclarar que para hacer las respectivas operaciones invocamos a los métodos remotos, en código se ve de la siguiente manera:

```
import java.rmi.Naming;
import java.util.jar.Attributes.Name;
import javax.swing.plaf.synth.SynthStyleFactory;
public class ClienteRMI {
    static int N = 8;
```

```

static float [][] A = new float[N][N];
static float [][] B = new float[N][N];
static float [][] C = new float[N][N];
static void imprimirMatriz(float [][] 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();
    }
    return;
} // Impresion matriz
public static class Cliente extends Thread {
    int nodo;
    Cliente(int nodo){
        this.nodo = nodo;
    }
    public void run() {
        try {
            String url = "";
            if(nodo == 1){
                url = "rmi://20.231.19.203/prueba";
            }else if(nodo == 2){
                url = "rmi://20.25.64.209/prueba";
            }else if(nodo == 3){
                url = "rmi://20.231.55.64/prueba";
            }else if(nodo == 4){
                url = "rmi://20.228.220.68/prueba";
            }
            InterfaceRMI r = (InterfaceRMI)Naming.lookup(url);

            float [][] A1 = r.separa_matriz(A, 0);
            float [][] A2 = r.separa_matriz(A, A.length/4);
            float [][] A3 = r.separa_matriz(A, A.length/2);
            float [][] A4 = r.separa_matriz(A, (3*A.length)/4);

            float [][] B1 = r.separa_matriz(B, 0);
            float [][] B2 = r.separa_matriz(B, B.length/4);
            float [][] B3 = r.separa_matriz(B, B.length/2);
            float [][] B4 = r.separa_matriz(B, (3*B.length)/4);

            if(nodo == 1){

                float [][] C1 = r.multiplica_matrices(A1, B1);
                float [][] C2 = r.multiplica_matrices(A1, B2);
                float [][] C3 = r.multiplica_matrices(A1, B3);
                float [][] C4 = r.multiplica_matrices(A1, B4);

                for(int i = 0; i < N/4; i++){

                    for(int j = 0; j < N/4; j++){

                        C[i][j] = C1[i][j];
                        C[i][j+N/4] = C2[i][j];
                        C[i][j+N/2] = C3[i][j];
                        C[i][j+(3*N)/4] = C4[i][j];
                    }
                }
            }
        }
    }
}

```

```

    }
}

} else if(nodo == 2){

    float [][] C5 = r.multiplica_matrices(A2, B1);
    float [][] C6 = r.multiplica_matrices(A2, B2);
    float [][] C7 = r.multiplica_matrices(A2, B3);
    float [][] C8 = r.multiplica_matrices(A2, B4);

    for(int i = 0; i < N/4; i++){

        for(int j = 0; j < N/4; j++){

            C[i+N/4][j] = C5[i][j];
            C[i+N/4][j+N/4] = C6[i][j];
            C[i+N/4][j+N/2] = C7[i][j];
            C[i+N/4][j+(3*N)/4] = C8[i][j];

        }

    }

} else if(nodo == 3){

    float [][] C9 = r.multiplica_matrices(A3, B1);
    float [][] C10 = r.multiplica_matrices(A3, B2);
    float [][] C11 = r.multiplica_matrices(A3, B3);
    float [][] C12 = r.multiplica_matrices(A3, B4);

    for(int i = 0; i < N/4; i++){

        for(int j = 0; j < N/4; j++){

            C[i+N/2][j] = C9[i][j];
            C[i+N/2][j+N/4] = C10[i][j];
            C[i+N/2][j+N/2] = C11[i][j];
            C[i+N/2][j+(3*N)/4] = C12[i][j];

        }

    }

} else {

    float [][] C13 = r.multiplica_matrices(A4, B1);
    float [][] C14 = r.multiplica_matrices(A4, B2);
    float [][] C15 = r.multiplica_matrices(A4, B3);
    float [][] C16 = r.multiplica_matrices(A4, B4);

    for(int i = 0; i < N/4; i++){

        for(int j = 0; j < N/4; j++){

            C[i+((3*N)/4)][j] = C13[i][j];
            C[i+((3*N)/4)][j+N/4] = C14[i][j];
            C[i+((3*N)/4)][j+N/2] = C15[i][j];
            C[i+((3*N)/4)][j+(3*N)/4] = C16[i][j];


```

```

        }
    }

    }

    } catch (Exception e){
        System.out.println(e.getMessage());
    }
}

}

public static void main(String[] args) throws Exception {

    String url = "rmi://20.231.19.203/prueba";
    InterfaceRMI r = (InterfaceRMI)Naming.lookup(url);

    System.out.println("Soy el cliente");

    //int N = 8;
    //float [][] A = new float[N][N];
    //float [][] B = new float[N][N];

    for(int i = 0; i < N; i++){
        for(int j = 0; j < N; j++){
            A[i][j] = i+2*j;
            B[i][j] = 3*i-j;
        }
    }

    if(N == 8) {

        System.out.println("La matriz A es:");
        imprimirMatriz(A);
        System.out.println("La matriz B es:");
        imprimirMatriz(B);
    }

    B = r.transpuesta(B);

    Cliente cliente1 = new Cliente(1);
    cliente1.start();
    cliente1.join();
    System.out.println("\nhilo 1");

    Cliente cliente2 = new Cliente(2);
    cliente2.start();
    cliente2.join();
    System.out.println("\nhilo 2");

    Cliente cliente3 = new Cliente(3);
    cliente3.start();
    cliente3.join();
    System.out.println("\nhilo 3");
}

```

```

        Cliente cliente4 = new Cliente(4);
        cliente4.start();
        cliente4.join();

        System.out.println("\nhilo 4");

        if(N == 8) {
            System.out.println("La matriz C es:");
            imprimirMatriz(C);
        }

        System.out.println("El checksum es: " + r.checksum(C));
    }
}

```

3. InterfaceRMI

En la Interface se declararán todos los métodos que serán invocados de manera remota, esta interfaz los métodos a exportar. para poder usar RemoteException en necesario que la interfaz extienda de remote.

El código se ve de esta manera

```

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

public interface InterfaceRMI extends Remote {

    public float checksum(float[][] m) throws RemoteException;
    public float [][] multiplica_matrices(float [][]A, float[][] B)
    throws RemoteException;
    public float [][] separa_matriz(float [][] A, int inicio) throws
    RemoteException;
    public float [][] traspuesta (float [][] A) throws
    RemoteException;

}

```

4. ClaseRMI

La ClaseRMI que implementa InterfaceRMI define los métodos dentro de la interfaz, serán los métodos que podrán ser invocados remotamente, se definieron los métodos de traspuesta, pásara de las fila a columnas de la matriz, separa_matriz, dividir la matriz de NxN a una de N/4xN multiplica matriz, está multiplicará dos matrices renglon por renglon y regresa una matriz cuadrada, y checksum, la cual sumará cada elemento de la matriz C. todos los métodos remotos utilizan RemoteException ya que puede ocurrir una excepción en la ejecución del programa. El programa extiende a UnicatRemoteObject se utiliza para exportar un objeto remoto con Java.

El código se ve de esta manera:

```

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

public class ClaseRMI extends UnicastRemoteObject implements InterfaceRMI {

    public ClaseRMI () throws RemoteException {
        super();
    }

    public float checksum(float[][] m) throws RemoteException {
        float s = 0;

        for(int i = 0; i < m.length; i++){
            for(int j = 0; j < m[0].length; j++){
                s += m[i][j];
            }
        }
        return s;
    }

    public float [][] traspuesta (float [][] A) throws RemoteException {

        //Crear la traspuesta de la matriz
        float [][] aT = new float[A.length][A[0].length];

        for(int i = 0; i < A.length; i++){
            for(int j = 0; j < A[0].length; j++){
                aT[i][j] = A[j][i];
            }
        }
        //Asignar la traspuesta
        for(int i = 0; i < A.length; i++){
            for(int j = 0; j < A[0].length; j++){
                A[i][j] = aT[i][j];
            }
        }

        return A;
    }

    // 0
    // N/4
    // N/2
    // 3N/4

    public float [][] separa_matriz(float [][] A, int inicio) throws
    RemoteException {

        float [][] C = new float[A.length/4][A[0].length];

        for(int i = 0; i < A.length/4; i++){
            for(int j = 0; j < A[0].length; j++){
                C[i][j] = A[i+inicio][j];
            }
        }

        return C;
    }

    public float [][] multiplica_matrices(float [][]A, float[][] B) throws
    RemoteException{

        //La multiplicacion de matrices
        float [][] C = new float[A.length][A.length];

```

```

        //int tam = N/2;
        //System.out.println("Lenght = " + A.length + ", [0] lenght = " +
A[0].length);

        for(int i = 0; i < A.length; i++){
            for(int j = 0; j < B.length; j++){
                for(int k = 0; k < A[0].length; k++){

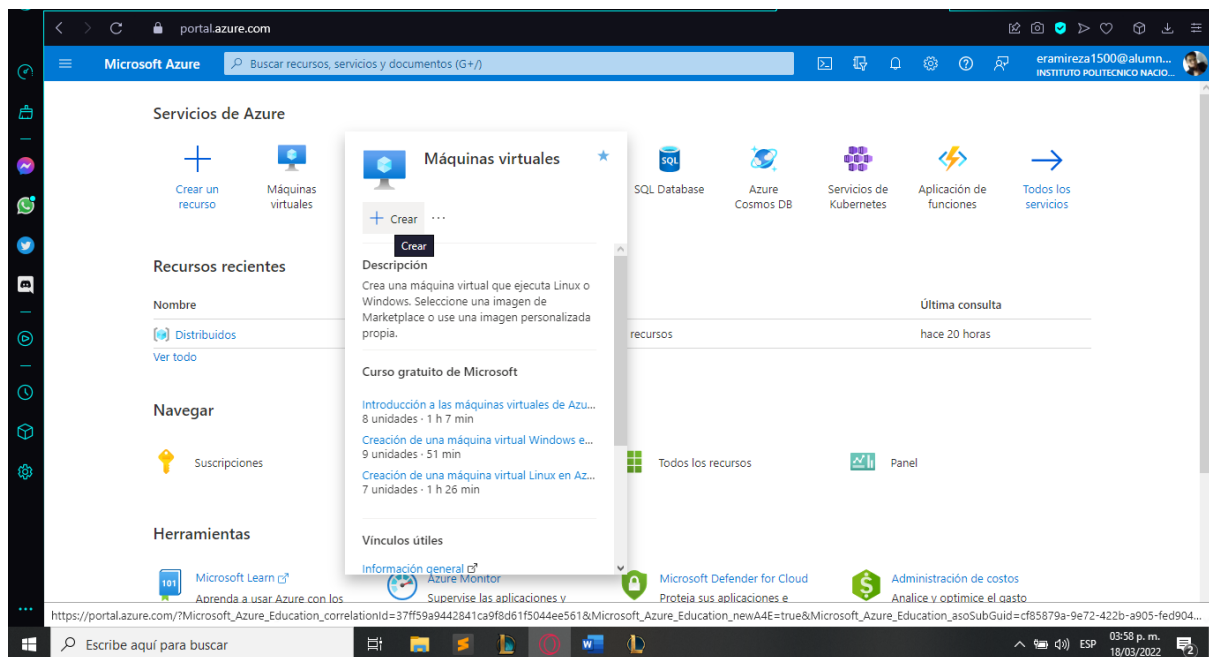
                    C[i][j] += ( A[i][k] * B[j][k] );

                }
            }
        }
        return C;
    }
}

```

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.

1.- Se ingresa al portal de Azure y seleccionar la opción de Máquina Virtual - Crear



2.- Se ingresan los datos correspondientes en el apartado de datos básicos poniendo énfasis en nombre de la máquina, imagen (en este caso será Ubuntu Server 18.04 LTS), tamaño, nombre de usuario y contraseña.

portal.azure.com

Microsoft Azure

Buscar recursos, servicios y documentos (0+)

Inicio > Máquinas virtuales >

Crear una máquina virtual

⚠ Al cambiar opciones básicas se pueden restablecer las selecciones realizadas. Revise todas las opciones antes de crear la máquina virtual.

Detalles del proyecto
 Seleccione la suscripción para administrar recursos implementados y los costos. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción *

Grupo de recursos * [Crear nuevo](#)

Detalles de instancia

Nombre de máquina virtual *

Región *

Opciones de disponibilidad

Tipo de seguridad

Imagen * [Ver todas las imágenes](#) [Configurar la generación de máquinas virtuales](#)

Instancia de Azure de acceso puntual ☐

Tamaño * [Ver todos los tamaños](#)

[Revisar y crear](#) < Anterior Siguiente: Discos >

portal.azure.com

Microsoft Azure

Buscar recursos, servicios y documentos (0+)

Inicio > Máquinas virtuales >

Crear una máquina virtual

⚠ Al cambiar opciones básicas se pueden restablecer las selecciones realizadas. Revise todas las opciones antes de crear la máquina virtual.

Cuenta de administrador

Tipo de autenticación ☐ Clave pública SSH ☒ Contraseña

Nombre de usuario *

Contraseña *

Confirmar contraseña *

Reglas de puerto de entrada
 Seleccione los puertos de red de máquina virtual que son accesibles desde la red Internet pública. Puede especificar acceso de red más limitado o granular en la pestaña Red.

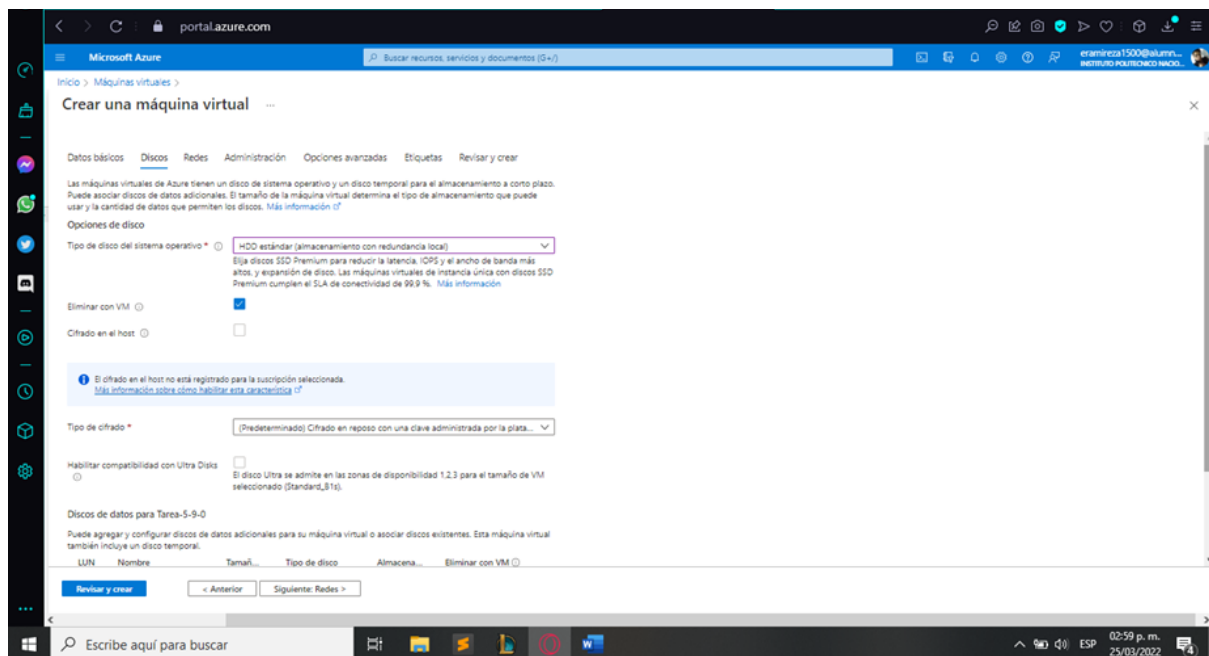
Puertos de entrada públicos * ☐ Ninguno ☒ Permitir los puertos seleccionados

Seleccionar puertos de entrada *

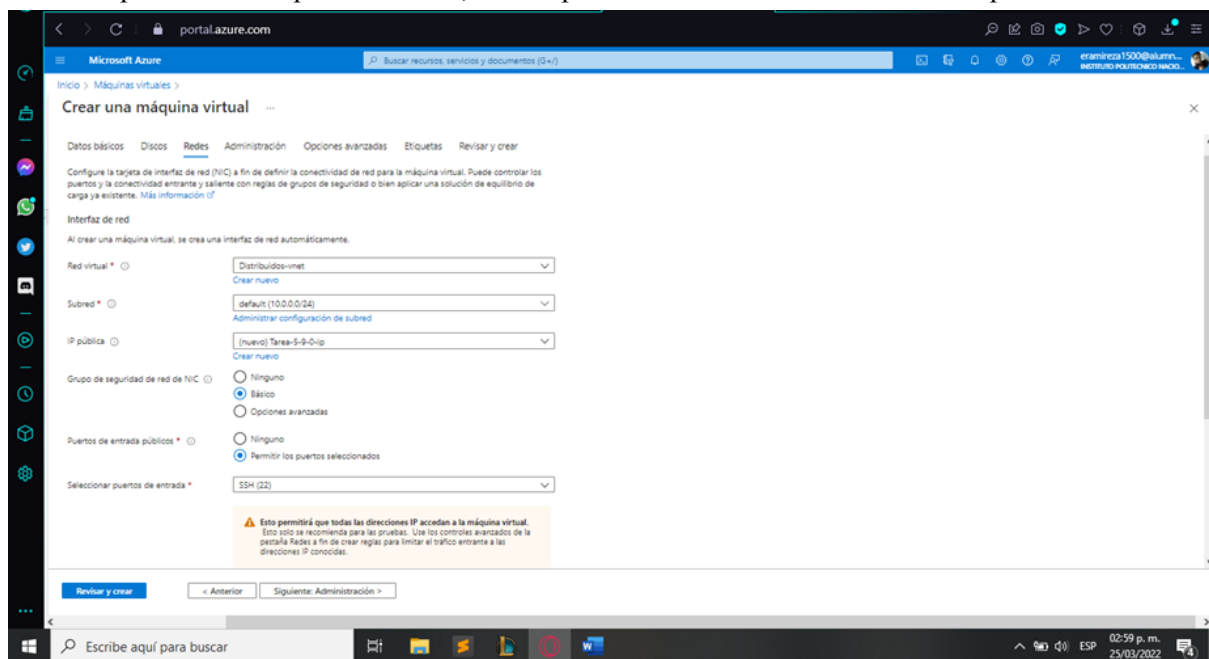
⚠ Esto permitirá que todas las direcciones IP accedan a la máquina virtual. Esto solo se recomienda para las pruebas. Use los controles avanzados de la pestaña Redes a fin de crear reglas para limitar el tráfico entrante a las direcciones IP conocidas.

[Revisar y crear](#) < Anterior Siguiente: Discos >

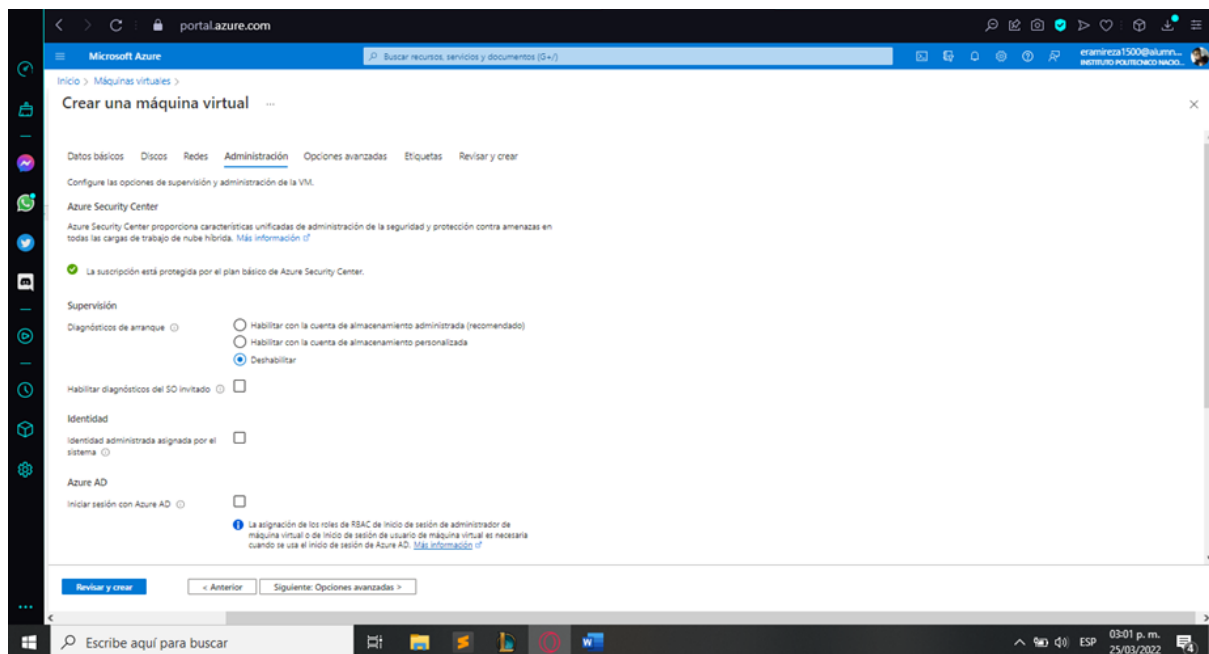
3.- Se continúa ingresando los datos con respecto al disco, indicando el disco duro como HDD estándar.



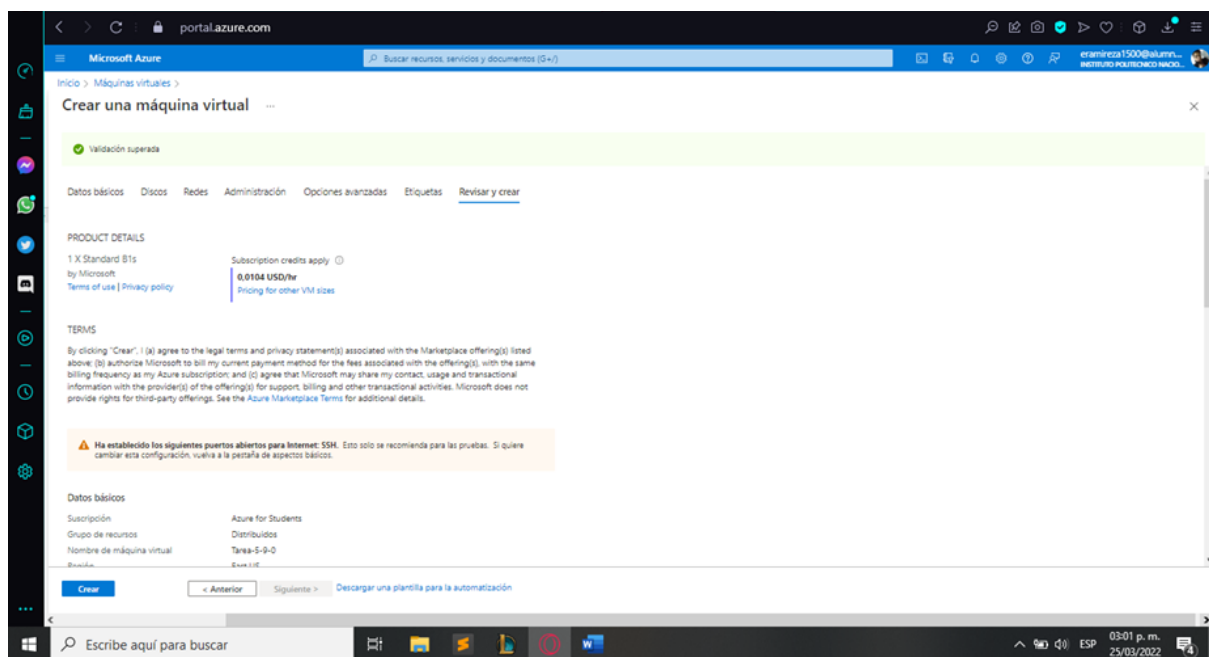
4.- Para proceder con la creación de la máquina virtual se continúa ingresando la información necesaria para llenar el apartado de red, dicho apartado es llenado automáticamente por Azure.



5.- Por último, se procede a seleccionar en el apartado de administración la opción de deshabilitar el diagnóstico de arranque para después dar click en revisar y crear.



6.- Una vez ingresado todas las características de nuestra máquina virtual y haber seleccionado revisar y crear se observa los datos generales de nuestra máquina virtual y el costo que este tendrá.



portal.azure.com

Microsoft Azure

Búsqueda de recursos, servicios y documentos (G+7)

Inicio > Máquinas virtuales >

Crear una máquina virtual

Validación superada

Datos básicos

Suscripción	Azure for Students
Grupo de recursos	Distribuidos
Nombre de máquina virtual	Tarea-5-9-0
Región	East US
Opciones de disponibilidad	No se requiere redundancia de la infraestructura
Tipo de seguridad	Estándar
Imagen	Ubuntu Server 18.04 LTS - Gen2
Tamaño	Standard B1s (1 vcpu, 1 GiB de memoria)
Tipo de autenticación	Contraseña
Nombre de usuario	Equipop9
Puertos de entrada públicos	SSH
Azure de acceso puntual	No

Discos

Tipo de disco del sistema operativo	LRS de HDD estándar
Usar discos administrados	Si
Eliminar disco de SO con VM	Habilitado
Disco de SO efímero	No

Definir

Crear < Anterior Siguiente > Descargar una plantilla para la automatización

Escribe aquí para buscar

03:02 p. m. 25/03/2022

portal.azure.com

Microsoft Azure

Búsqueda de recursos, servicios y documentos (G+7)

Inicio > Máquinas virtuales >

Crear una máquina virtual

Validación superada

Redes

Red virtual	Distribuidos-vnet
Subred	default (10.0.0.0/24)
IP pública	(nuevo) Tarea-5-9-0-ip
Redes aceleradas	Desactivado
¿Quieres colocar esta máquina virtual como subyacente respecto a una solución de equilibrio de carga existente?	No
Eliminar IP pública y NIC cuando se elimine la VM	Deshabilitado

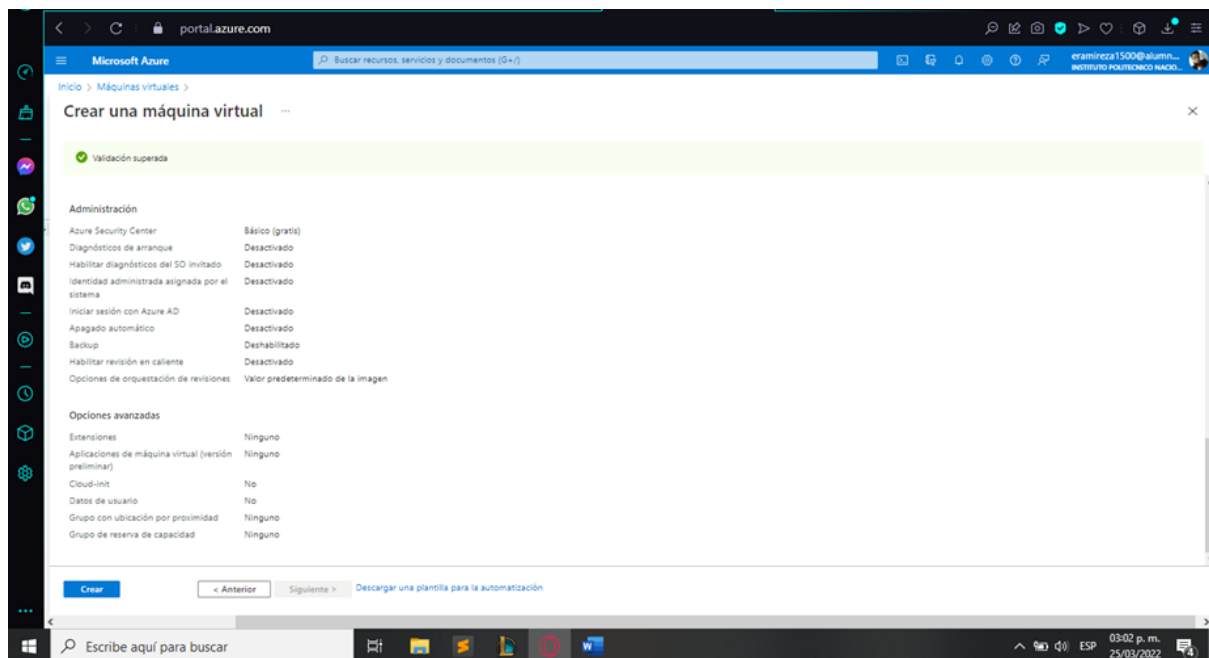
Administración

Azure Security Center	Básico (gratis)
Diagnósticos de arranque	Desactivado
Habilitar diagnósticos del SO invitado	Desactivado
Identidad administrada asignada por el sistema	Desactivado
Iniciar sesión con Azure AD	Desactivado
Apagado automático	Desactivado
Backup	Deshabilitado
Habilitar revisión en caliente	Desactivado
Opciones de orquestación de revisiones	Valor predeterminado de la imagen

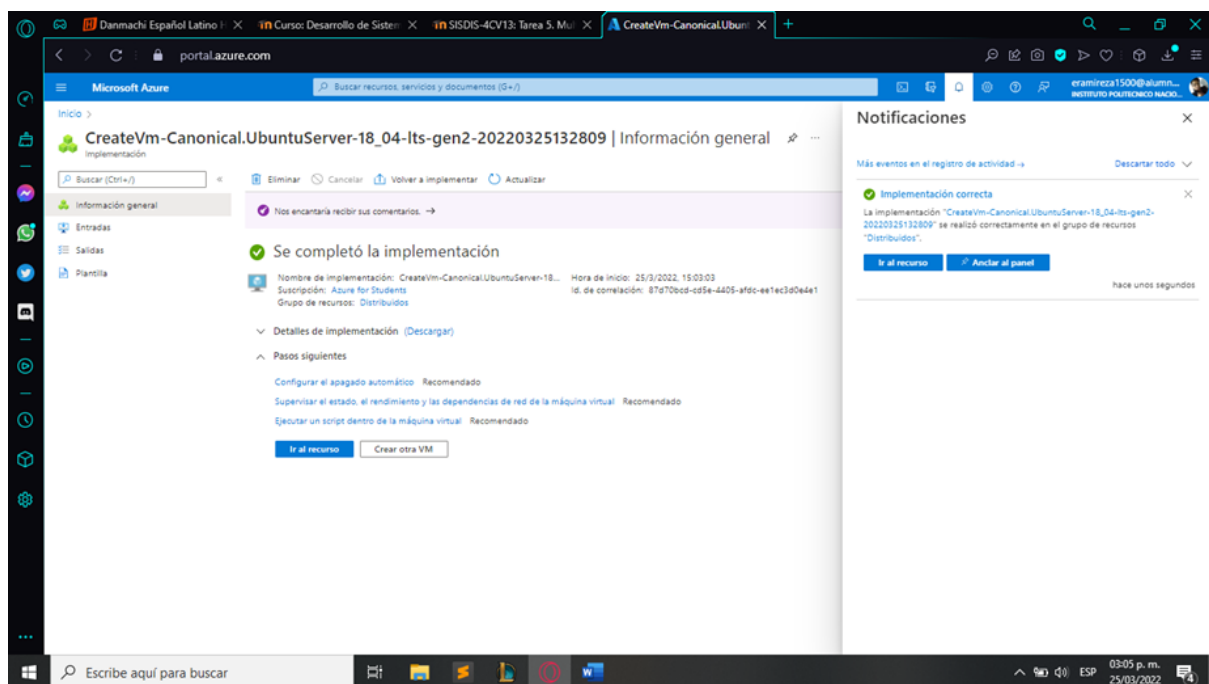
Crear < Anterior Siguiente > Descargar una plantilla para la automatización

Escribe aquí para buscar

03:02 p. m. 25/03/2022



7.- Ya verificado los datos, se procederá a darle crear, para iniciar con el proceso de creación de la máquina virtual.



8.- Ya creada la máquina virtual, se procederá a “ir al recurso” para visualizar la IP pública, dicha dirección puede cambiar cada que se apague y encienda la máquina virtual.

The screenshot shows the Azure portal interface for a virtual machine named 'Tarea-5-9-0'. The left sidebar contains navigation options like 'Información general', 'Registro de actividad', 'Control de acceso (IAM)', 'Etiquetas', 'Diagnosticar y solucionar problemas', 'Configuración', 'Redes', 'Conectar', 'Discos', 'Tamaño', 'Seguridad', 'Recomendaciones de Advisor', 'Extensiones + aplicaciones', and 'Entrega continua'. The main content area is divided into 'Información esencial' and 'Propiedades'.

Información esencial:

- Grupo de recursos: [\(mover\)](#) Distribuidos
- Estado: En ejecución
- Ubicación: East US
- Suscripción: [\(mover\)](#) Azure for Students
- Id. de suscripción: cf85879a-9e72-422b-a905-fed904b1dfbd
- Etiquetas: [\(editar\)](#) Haga clic aquí para agregar etiquetas.
- Sistema operativo: Linux (ubuntu 18.04)
- Tamaño: Standard D51 v2 (1 vcpu, 3.5 GiB de memoria)
- Dirección IP pública: [20.25.35.108](#)
- Red virtual/subred: [Distribuidos-vnet/default](#)
- Nombre DNS: [Sin configurar](#)

Propiedades:

- Máquina virtual:**
 - Nombre del equipo: Tarea-5-9-0
 - Estado de mantenimiento: -
 - Sistema operativo: Linux (ubuntu 18.04)
 - Publisher: Canonical
 - Oferta: UbuntuServer
 - Plan: 18_04-lts-gen2
 - Generación de VM: V2
- Redes:**
 - Dirección IP pública: 20.25.35.108
 - Dirección IP pública (IPv6): -
 - Dirección IP privada: 10.0.0.5
 - Dirección IP privada (IPv6): -
 - Red virtual/subred: [Distribuidos-vnet/default](#)
 - Nombre DNS: [Configurar](#)

NOTA: Es importante aclarar que la RAM del nodo cliente fue modificada posteriormente cambiando de 1 GB de RAM a 3.5 GB de RAM dado a que por cada región solo se permite utilizar hasta 4 máquinas por 1 GB.

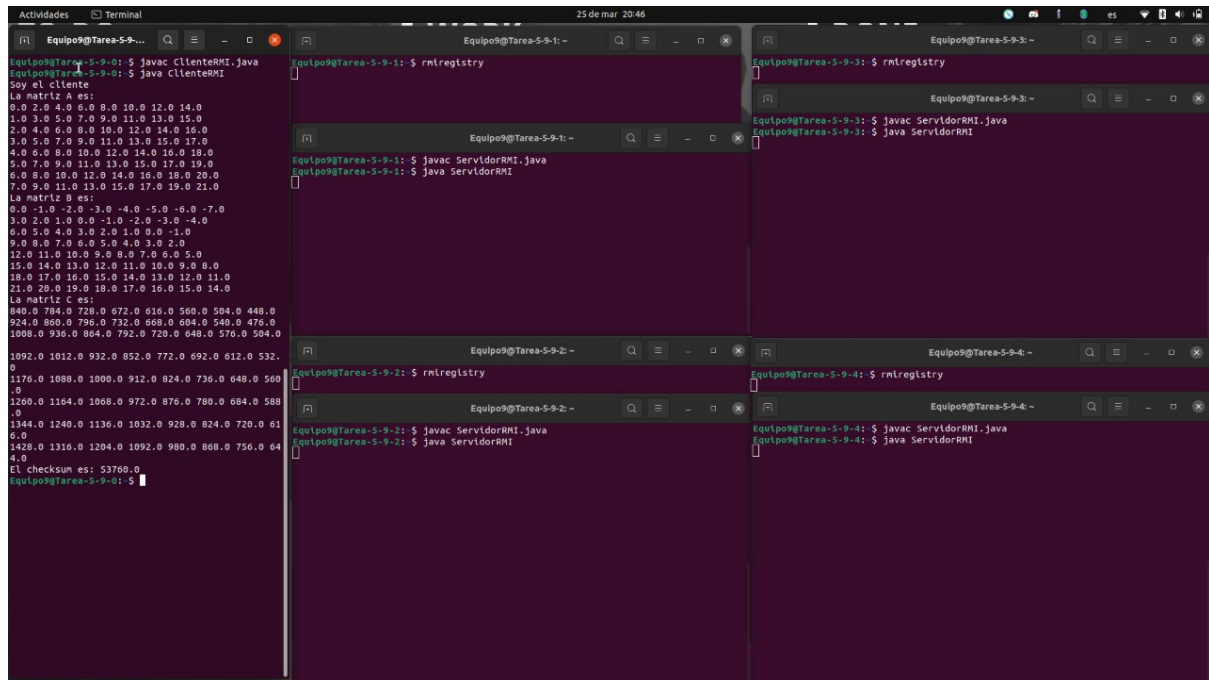
The screenshot shows the 'Tamaño' (Size) configuration page for the virtual machine 'Tarea-5-9-0'. It displays a list of available VM sizes with columns for 'Tamaño de VM', 'Familia', 'vCPU', 'RAM (GiB)', 'Discos de datos', 'E/S máxima por s...', and 'Almacenamiento'. The current size is 'Standard_DS1_v2'.

Tamaño de VM	Familia	vCPU	RAM (GiB)	Discos de datos	E/S máxima por s...	Almacenamiento
DS1_v2	Uso general	1	3.5	4	3200	7
D2s_v3	Uso general	2	8	4	3200	16
D2as_v4	Uso general	2	8	4	3200	16
DS2_v2	Uso general	2	7	8	6400	14

Los precios que se muestran son precios estimados en su moneda local que incluyen solo el costo de la infraestructura de Azure y los descuentos aplicables a la suscripción y ubicación. Los precios no reflejan los costos de software aplicables. Los cargos definitivos se mostrarán en su moneda local en las vistas de facturación y análisis de costos. [Vea la calculadora de precios de Azure.](#)

Ahora adjuntamos las capturas de pantalla correspondientes a la conversación realizada para mostrar la compilación y ejecución del programa:

Podemos ver el resultado para $N = 8$, donde imprimimos las matrices de cada una como se nos pidió en los requerimientos y el checksum.



```
Actividades Terminal 25 de mar 20:46
Equipo9@Tarea-5-9-0: $ javac ClienteRMI.java
Equipo9@Tarea-5-9-0: $ java ClienteRMI
Soy el cliente
La matriz A es:
0.0 2.0 4.0 6.0 8.0 10.0 12.0 14.0
1.0 3.0 5.0 7.0 9.0 11.0 13.0 15.0
2.0 4.0 6.0 8.0 10.0 12.0 14.0 16.0
3.0 5.0 7.0 9.0 11.0 13.0 15.0 17.0
4.0 6.0 8.0 10.0 12.0 14.0 16.0 18.0
5.0 7.0 9.0 11.0 13.0 15.0 17.0 19.0
6.0 8.0 10.0 12.0 14.0 16.0 18.0 20.0
7.0 9.0 11.0 13.0 15.0 17.0 19.0 21.0
La matriz B es:
0.0 -1.0 -2.0 -3.0 -4.0 -5.0 -6.0 -7.0
3.0 2.0 1.0 0.0 -1.0 -2.0 -3.0 -4.0
6.0 5.0 4.0 3.0 2.0 1.0 0.0 -1.0
9.0 8.0 7.0 6.0 5.0 4.0 3.0 2.0
12.0 11.0 10.0 9.0 8.0 7.0 6.0 5.0
15.0 14.0 13.0 12.0 11.0 10.0 9.0 8.0
18.0 17.0 16.0 15.0 14.0 13.0 12.0 11.0
21.0 20.0 19.0 18.0 17.0 16.0 15.0 14.0
La matriz C es:
840.0 784.0 728.0 672.0 616.0 560.0 504.0 448.0
924.0 860.0 796.0 732.0 668.0 604.0 540.0 476.0
1008.0 936.0 864.0 792.0 720.0 648.0 576.0 504.0
1092.0 1012.0 932.0 852.0 772.0 692.0 612.0 532.0
1176.0 1088.0 1000.0 912.0 824.0 736.0 648.0 560.0
1260.0 1164.0 1068.0 972.0 876.0 780.0 684.0 588.0
1344.0 1240.0 1136.0 1032.0 928.0 824.0 720.0 616.0
1428.0 1316.0 1204.0 1092.0 980.0 868.0 756.0 644.0
El checksum es: 53760.0
Equipo9@Tarea-5-9-0: $

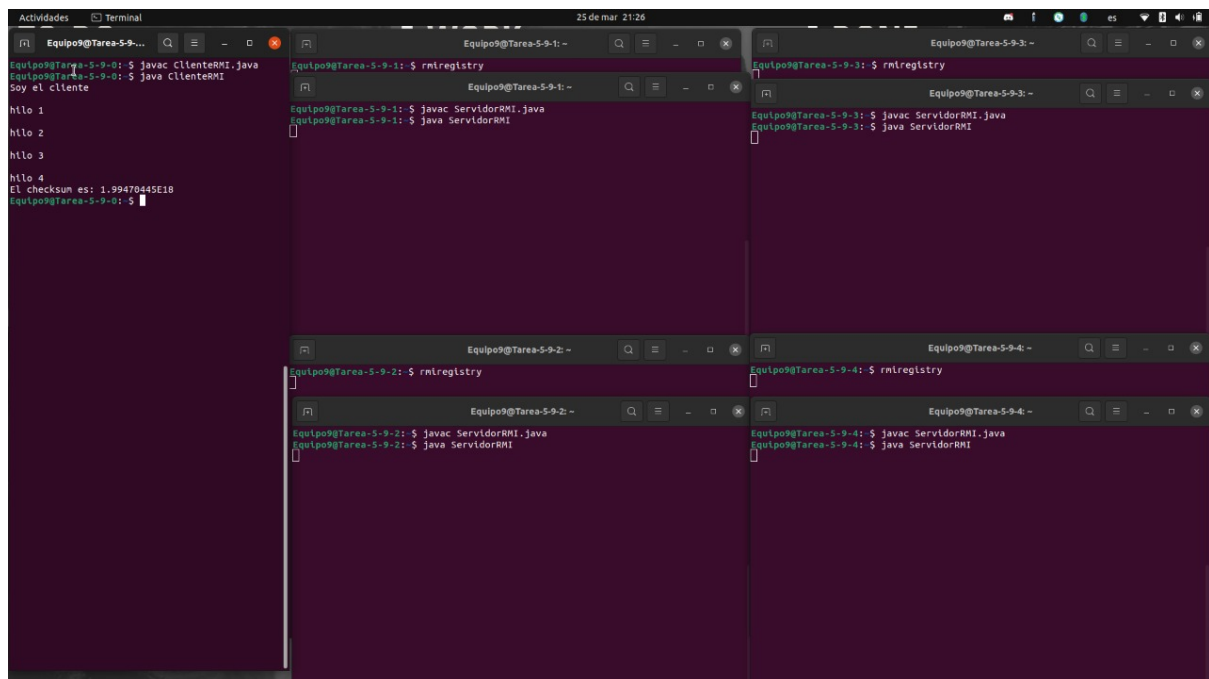
Equipo9@Tarea-5-9-1: $ rmiregistry
Equipo9@Tarea-5-9-1: $ javac ServidorRMI.java
Equipo9@Tarea-5-9-1: $ java ServidorRMI

Equipo9@Tarea-5-9-3: $ rmiregistry
Equipo9@Tarea-5-9-3: $ javac ServidorRMI.java
Equipo9@Tarea-5-9-3: $ java ServidorRMI

Equipo9@Tarea-5-9-2: $ rmiregistry
Equipo9@Tarea-5-9-2: $ javac ServidorRMI.java
Equipo9@Tarea-5-9-2: $ java ServidorRMI

Equipo9@Tarea-5-9-4: $ rmiregistry
Equipo9@Tarea-5-9-4: $ javac ServidorRMI.java
Equipo9@Tarea-5-9-4: $ java ServidorRMI
```

Para $N = 4000$, en este caso pues no se nos pide imprimir las matrices pero sí el checksum; como mensaje adicional imprimimos cada vez que mandábamos a ejecutar cada uno de los hilos, sólo con fines ilustrativos.



```
Actividades Terminal 25 de mar 21:26
Equipo9@Tarea-5-9-0: $ javac ClienteRMI.java
Equipo9@Tarea-5-9-0: $ java ClienteRMI
Soy el cliente
hilo 1
hilo 2
hilo 3
hilo 4
El checksum es: 1.99470445E18
Equipo9@Tarea-5-9-0: $

Equipo9@Tarea-5-9-1: $ rmiregistry
Equipo9@Tarea-5-9-1: $ javac ServidorRMI.java
Equipo9@Tarea-5-9-1: $ java ServidorRMI

Equipo9@Tarea-5-9-3: $ rmiregistry
Equipo9@Tarea-5-9-3: $ javac ServidorRMI.java
Equipo9@Tarea-5-9-3: $ java ServidorRMI

Equipo9@Tarea-5-9-2: $ rmiregistry
Equipo9@Tarea-5-9-2: $ javac ServidorRMI.java
Equipo9@Tarea-5-9-2: $ java ServidorRMI

Equipo9@Tarea-5-9-4: $ rmiregistry
Equipo9@Tarea-5-9-4: $ javac ServidorRMI.java
Equipo9@Tarea-5-9-4: $ java ServidorRMI
```

Conclusiones.

Erick Eduardo Ramírez Arellano: Mediante la elaboración de esta práctica comprendí más a fondo las funciones más completas de una máquina virtual, así como la creación y forma de ejecución de una máquina virtual pero con un sistema operativo Ubuntu, ya que esto cambia, en la máquina virtual de Windows, solo bastaba con descargar el recurso .rdp e ingresar el usuario, pero con Ubuntu se debe de acceder al cmd para ingresar ssh usuario@ip pública y posteriormente ingresar el usuario y contraseña, cosa que se me complicó ya que el cmd no me detectaba ssh como una herramienta, cosa que pude solucionar, de igual forma aprendí a habilitar nuevos puertos para permitir la comunicación entre los nodos (siendo el puerto 1099) y también, entendí más a fondo lo que involucra las funciones RMI y el cómo intentar la comunicación entre ellos.

Omar Ramos Herrera: En la tarea ocupamos los métodos remotos mediante RMI, es muy ocupar métodos que se ejecutarán en otra computadora y te regresaran los datos o información ya procesada, hicimos uso de esta herramienta para hacer un cálculo de manera distribuida y paralela, para esto ocupamos 5 máquinas en azure con ubuntu, y aun así 1GB de ram no fue suficiente, por lo cual se subió la capacidad de las máquinas para que el programa pueda ejecutarse con una matriz de 4000 x 4000. La ambientación de las máquinas es muy sencilla debido al instalador de paquetes de ubuntu. a diferencia de la tarea 3 fue más sencillo conectar las máquinas virtuales ya que el funcionamiento es muy similar al de una API, diciendo que estamos consumiendo un servicio web, dando datos y recibiendo una respuesta sin hacer muchas configuraciones de conexión.

Elisa Ramos Gomez: La tarea fue todo un mar de complicaciones porque para empezar no me había quedado 100% claro cómo es que existe la transparencia en RMI, ya que no está de manera explícita lo que estábamos acostumbrados de llamar a métodos como enviar y recibir, pero de alguna forma al acceder a los métodos remotos en cliente suceden las cosas, de igual manera fue mucha prueba y error ya que esta vez me tocó a mí hacer algo un poco digamos menos tedioso al momento de manejar las matrices ya que en un principio hacíamos copias completamente digamos innecesarias ahorrando espacio y de igual manera tiempo ya que la división se volvió un poco más fácil al momento de solo pasarle índices y no más que eso, de igual manera me sorprendió que no era suficiente los recursos que a un inicio habíamos seleccionado y teníamos que añadir recursos ya que para el cálculo de la matriz de 4000 pues necesitaba un poco más para poder funcionar de igual manera tuvimos que cuidar detalles importantes como abrir el puerto 1099 para RMI, ejecutar el rmiregistry en cada una de las máquinas, pero al final fue muy agradable ver que todo salió correctamente y que de manera abstracta sí estamos haciendo conexiones interesantes, tuve que de igual manera desvelarme pero cada vez me causa más satisfacción que las cosas funcionen de manera correcta y que mi equipo es muy colaborativo, aprendemos cada vez más en distribuir las cosas para que todo salga a la perfección.