Instituto Politécnico Nacional
Escuela Superior de Cómputo

Desarrollo de Sistemas Distribuidos
Actividad: Multiplicación de matrices utilizando objetos distribuidos
Curso impartido por el profesor: Pineda Guerrero Carlos
Grupo: 4CV1
21/01
Alumno: Adrian González Pardo

Ultima fecha modificado: 17 de noviembre de 2020

1. Desarrollo

Para esta practica es necesario conocer las ip's de nuestras maquinas virtuales las cuales actuaran como nuestro cliente y nuestros servidores de Java RMI, por ello a continuación se mostrara el código fuente y el como se realizo la tarea de enviar los archivos.

2. Código fuente:

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

/*
 * @author Adrian Gonzalez Pardo
 */

public interface InterfaceRMI extends Remote{
  public void setN(int N) throws RemoteException;
  public int[][] multiplica_matrices(int[][] A,int[][] B) throws RemoteException
    ;
}
```

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

/*
 * @author Adrian Gonzalez Pardo
 */

public class ClaseRMI extends UnicastRemoteObject implements InterfaceRMI{
  // es necesario que el contructor ClaseRMI() invoque el constructor de la
    superclase
  public int N;
  public ClaseRMI() throws RemoteException{
    super( );
  }

  public void setN(int N) throws RemoteException{
    this.N=N;
  }

  public int[][] multiplica_matrices(int[][] A,int[][] B) throws RemoteException
    {
    int[][] C = new int[N/2][N/2];
    for (int i = 0; i < N/2; i++)
      for (int j = 0; j < N/2; j++)
        for (int k = 0; k < N; k++)
          C[i][j] += A[i][k] * B[j][k];
    return C;
  }
}
```

```
import java.rmi.Naming;

/*
 * @author Adrian Gonzalez Pardo
 */

public class ServidorRMI{
  public static void main(String[] args) throws Exception{
    String url="rmi://0.0.0.0/matrices";
    /* 0.0.0.0 es un comodin para la interfaz de red en el que permita
     * escuchar en cualquier interfaz lo cual permite que no importa si
     * el server esta en intranet o extranet
     **/
```

```
    ClaseRMI obj=new ClaseRMI();
    // registra la instancia en el rmiregistry
    Naming.rebind(url,obj);
  }
}
```

```
import java.rmi.Naming;

/*
 * @author Adrian Gonzalez Pardo
 */

public class ClienteRMI{

  static int N=4,
    A[][]=new int[N][N],
    B[][]=new int[N][N],
    C[][]=new int[N][N];
  static void acomoda_matriz(int[][] C,int[][] A,int renglon,int columna){
  for (int i = 0; i < N/2; i++)
    for (int j = 0; j < N/2; j++)
      C[i + renglon][j + columna] = A[i][j];
  }

  public static void muesta_matriz(int[][] y){
    for(int[] i:y){
      for(int j:i){
        System.out.print("\t"+j+"\t");
      }
      System.out.print("\n");
    }
  }

  static int[][] parte_matriz(int[][] A,int inicio){
  int[][] M = new int[N/2][N];
  for (int i = 0; i < N/2; i++)
    for (int j = 0; j < N; j++)
      M[i][j] = A[i + inicio][j];
  return M;
  }

  static int[][] transponer(int[][] A){
    int [][] Bt=new int[N][N];
    for(int i=0;i<N;i++){
      for(int j=0;j<N;j++){
        Bt[j][i]=A[i][j];
      }
    }
    return Bt;
  }

  public static void llena_matriz(int[][] a,int[][] b,int[][] c){
    int i,j;
    for(i=0;i<N;i++){
      for(j=0;j<N;j++){
        a[i][j]=2*i-j;
        b[i][j]=2*i+j;
        c[i][j]=0;
      }
    }
  }

  public static long checksum(int[] a){
    long c=0;
    for(int i=0;i<a.length;i++){
      c+=a[i];
    }
    return c;
```

```
  }

  public static long checksum(long[] a){
    long c=0;
    for(int i=0;i<a.length;i++){
      c+=a[i];
    }
    return c;
  }

  public static void main(String args[]) throws Exception{
    if(args.length<1){

    }
    String url_local="rmi://localhost/matrices", /* Nodo 0 */
    url_nube1="rmi://52.185.206.51/matrices",    /* Nodo 1 */
    url_nube2="rmi://52.171.214.166/matrices",   /* Nodo 2 */
    url_nube3="rmi://13.84.203.145/matrices";    /* Nodo 3 */

    InterfaceRMI r=(InterfaceRMI)Naming.lookup(url_local),
    r1=(InterfaceRMI)Naming.lookup(url_nube1),
    r2=(InterfaceRMI)Naming.lookup(url_nube2),
    r3=(InterfaceRMI)Naming.lookup(url_nube3);
    r.setN(N);
    r1.setN(N);
    r2.setN(N);
    r3.setN(N);
    llena_matriz(A,B,C);
    int [][] Bt=transponer(B);
    int[][] A1=parte_matriz(A,0);
    int[][] A2=parte_matriz(A,N/2);
    int[][] B1=parte_matriz(Bt,0);
    int[][] B2=parte_matriz(Bt,N/2);
    int[][] C1=r.multiplica_matrices(A1,B1);
    int[][] C2=r1.multiplica_matrices(A1,B2);
    int[][] C3=r2.multiplica_matrices(A2,B1);
    int[][] C4=r3.multiplica_matrices(A2,B2);
    acomoda_matriz(C,C1,0,0);
    acomoda_matriz(C,C2,0,N/2);
    acomoda_matriz(C,C3,N/2,0);
    acomoda_matriz(C,C4,N/2,N/2);
    if(N==4){
      System.out.println("Matriz A");
      muesta_matriz(A);
      System.out.println("Matriz B");
      muesta_matriz(B);
      System.out.println("Matriz C");
      muesta_matriz(C);
    }
    long checksum_t[]=new long[N],checksum_f=0;
    for(int i=0;i<N;i++){
      checksum_t[i]=checksum(C[i]);
    }
    checksum_f=checksum(checksum_t);
    System.out.println("El checksum de la matriz C es: "+
      Long.toHexString(checksum_f));
  }
}
```

```
# Archivo Makefile
# @author Adrian Gonzalez Pardo
JVC=javac
SRCC=$(wildcard *.java)
OBJS=$(SRCC:.java=.class)
all: ${OBJS}

%.class: %.java
   ${JVC} $<
```

```
10
11 .PHONY: clean
12
13 clean:
14    rm *.class
```

3. Ejecución en red

Para la ejecución en red es necesario hacer uso de distintas VM's las cuales ejecutaran en distintas Maquinas Virtuales, el cual se presenta con hostname:ip

- ubuntu-azure:52.183.250.226
- ubuntu-azure1:52.185.206.51
- ubuntu-azure2:52.171.214.166
- ubuntu-azure3:13.84.203.145

Para comunicarse rápidamente con los equipos se realizo el siguiente script el cual se comunica inicialmente con el e instalar rapidamente los archivos se escribio el siguiente script y en la interfaz de red de la VM aun sea el caso o no de que cada VM este dentro del entorno de trabajo o no es bueno que inicialmente en la configuración de red le configuremos la aceptación del puerto 1099 vía tcp o en cualquier caso cualquiera para que este no sea el problema habilitar tanto el puerto TCP como el UDP para la aplicación.

```
1  #!/usr/bin/env bash
2  # @author Adrian Gonzalez Pardo
3  echo "Conectandose por primera vez a las VM's"
4  ssh adrian@52.171.214.166
5  ssh adrian@52.183.250.226
6  ssh adrian@52.185.206.51
7  ssh adrian@13.84.203.145
8  echo "Ejecutando instruccion de instalacion ya con pass predeterminado"
9  sshpass -p "adrianPardo_99" ssh adrian@52.171.214.166 "sudo apt update && sudo
       apt install openjdk-11-jdk cmake -y"
10 sshpass -p "adrianPardo_99" ssh adrian@52.183.250.226 "sudo apt update && sudo
       apt install openjdk-11-jdk cmake -y"
11 sshpass -p "adrianPardo_99" ssh adrian@52.185.206.51 "sudo apt update && sudo
       apt install openjdk-11-jdk cmake -y"
12 sshpass -p "adrianPardo_99" ssh adrian@13.84.203.145 "sudo apt update && sudo
       apt install openjdk-11-jdk cmake -y"
```

Finalmente se escribio un script para enviar los archivos fuente para su ejecución

```
1  #!/usr/bin/env bash
2  # @author Adrian Gonzalez Pardo
3
4  echo "Envia datos a la nube"
5  echo "Nodo 0"
6  sshpass -p "adrianPardo_99" scp *.java Makefile adrian@52.171.214.166:~/
7  echo "Nodo 1"
8  sshpass -p "adrianPardo_99" scp *.java Makefile adrian@52.183.250.226:~/
9  echo "Nodo 2"
10 sshpass -p "adrianPardo_99" scp *.java Makefile adrian@52.185.206.51:~/
11 echo "Nodo 3"
12 sshpass -p "adrianPardo_99" scp *.java Makefile adrian@13.84.203.145:~/
13
14 echo "Compila datos java ya preconfigurados"
15 echo "Nodo 0"
```

```
16 sshpass -p "adrianPardo_99" ssh adrian@52.171.214.166 "make"
17 echo "Nodo 1"
18 sshpass -p "adrianPardo_99" ssh adrian@52.183.250.226 "make"
19 echo "Nodo 2"
20 sshpass -p "adrianPardo_99" ssh adrian@52.185.206.51 "make"
21 echo "Nodo 3"
22 sshpass -p "adrianPardo_99" ssh adrian@13.84.203.145 "make"
```

4. Capturas y descripción del programa

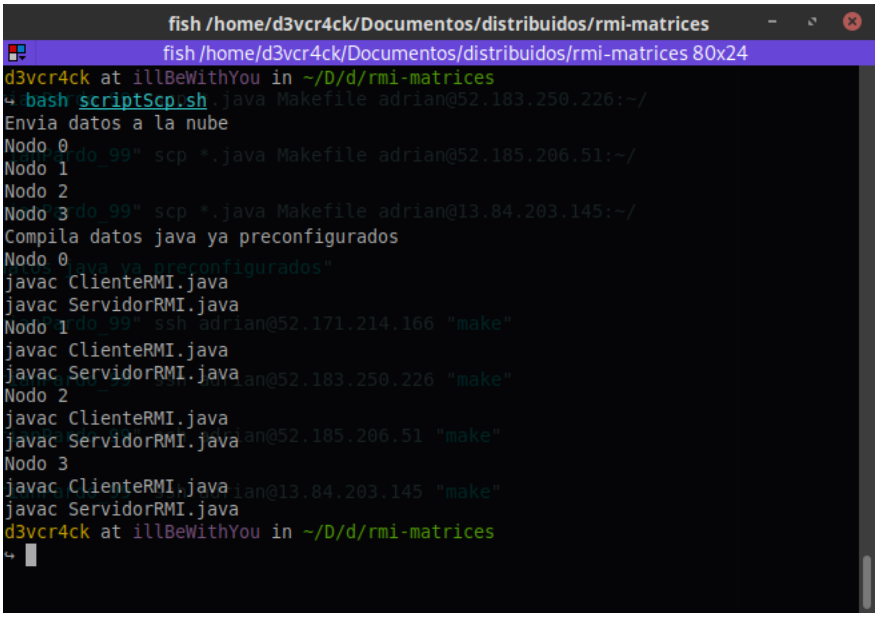

Figura 1: Ejecución del script para crear enviar los datos a las maquinas virtuales

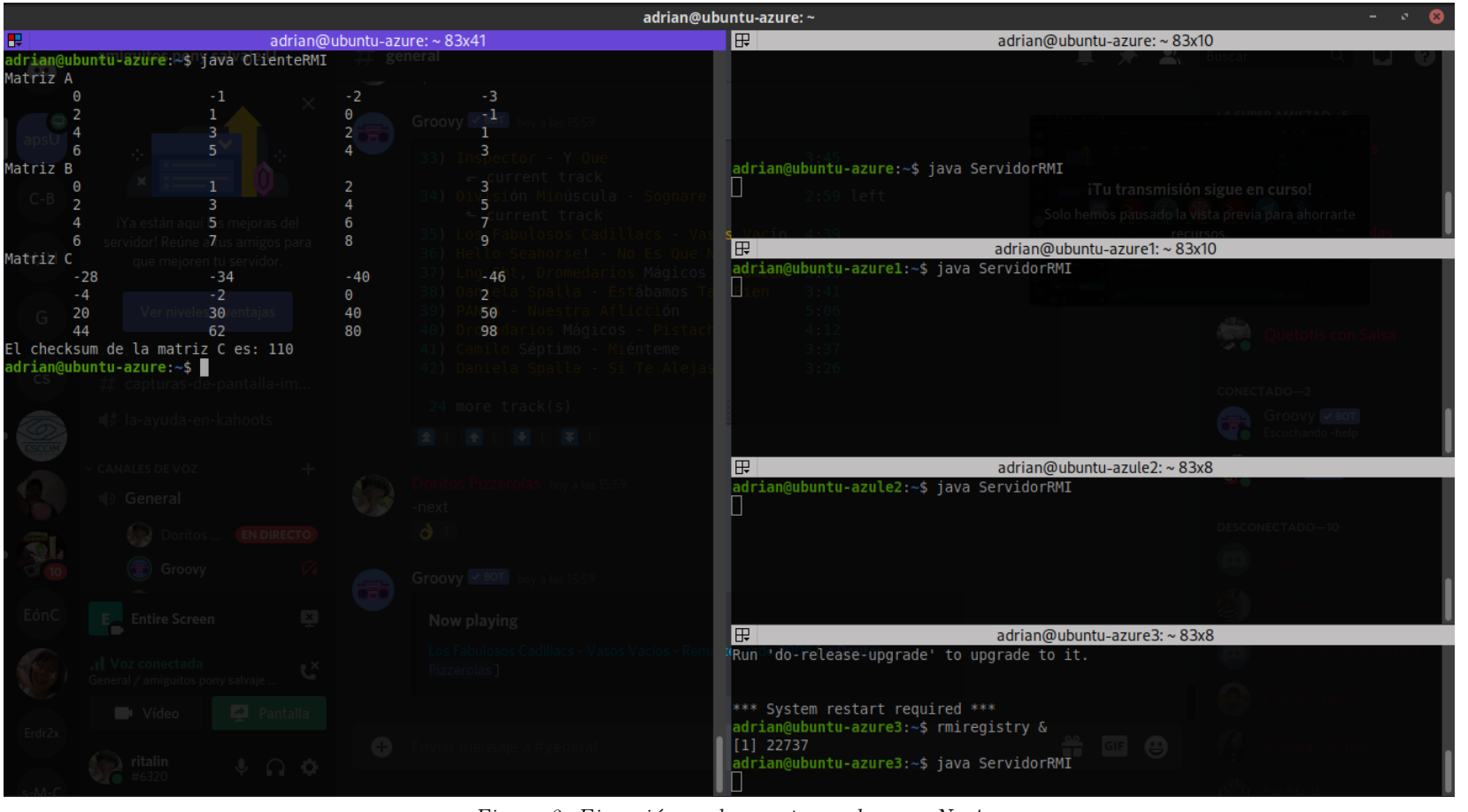

Figura 2: Ejecución en los cuatro nodos con N=4

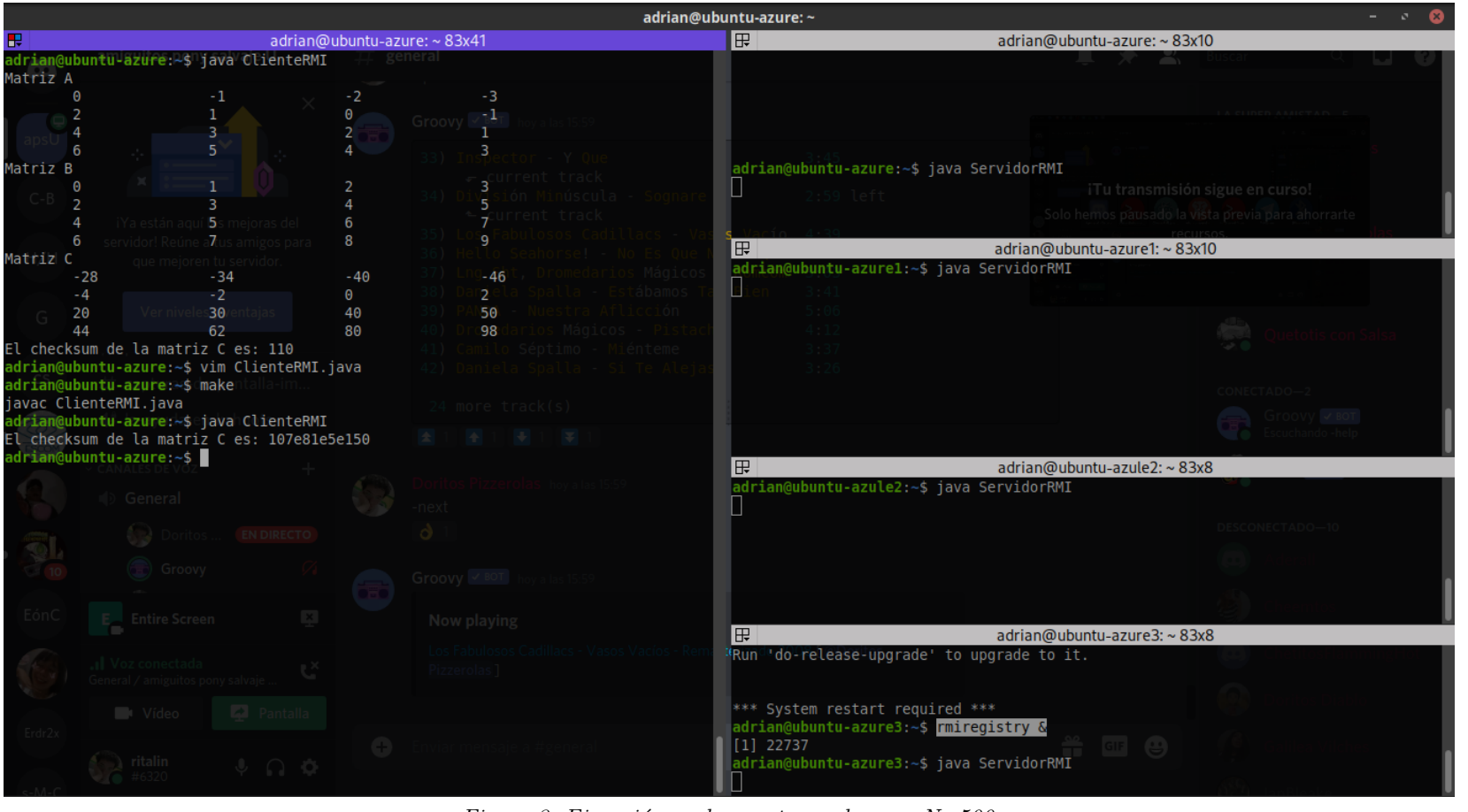

Figura 3: Ejecución en los cuatro nodos con N=500

5. Conclusiones

El implementar RMI en esta practica nos permite darnos cuenta de lo sencillo que es implementar el calculo de la multiplicación de matrices que se realizo vía hilos y sockets, que fue más sencillo de implementar con respecto al otro al menos en líneas de código.