Instituto Politécnico Nacional

Escuela Superior de Cómputo

Asignatura:

Desarrollo de sistemas distribuidos

Grupo:4CM3

Tarea 6. Multiplicación de matrices con RMI

Alumno: Garcia Garcia Rafael

Profesor: Carlos Pineda Guerrero

Capturas de la ejecución del programa:

Lo primero que tenemos que realizar en nuestra práctica es ejecutar y compilar nuestros programas de manera local para verificar si funcionan de manera correcta antes de pasarlo a nuestros servidores en Azure. En esta primera parte se compilan los programas haciendo uso en la URL de localhost, además ejecutamos en una terminal de Windows rmiRegistry para que la API empiece la ejecución. Una vez teniendo en nuestra terminal con N=4 el valor de checksum 272 pasamos a montar nuestros servidores de **UBUNTU** en la nube para crear un total de 4 nodos los cuáles van a ejecutar todos RMIregistry y el servidor en la terminal, además vamos a ejecutar en el nodo 0 el cliente para que este establezca la comunicación con los demás nodos para enviarles las partes de la matriz correspondientes.

Configuración de las máquinas Virtuales con Ubuntu

1. Vamos a la plataforma de Microsoft Azure en la parte de “servicios de Azure” seleccionamos máquinas virtuales > Agregar > Máquina Virtual y pasamos a fijar los parámetros de nuestra máquina virtual
2. Seleccionar el grupo de recursos o crear uno nuevo. Un grupo de recursos es similar a una carpeta dónde se pueden colocar los diferentes recursos de nube que se crean en Azure. Luego ingresamos el nombre de la máquina virtual.
3. Seleccionemos la región dónde se creará la máquina virtual.
4. Seleccionemos la Imagen en este caso es un “Ubuntu Server 18.04 LTS -Gen1”
5. Damos click en "Seleccionar tamaño" de la máquina virtual, en este caso vamos a seleccionar una máquina virtual con 1 GB de memoria RAM. Dar click en el botón "Seleccionar".
6. En tipo de autenticación seleccionamos "Contraseña" e ingresamos el nombre del usuario, posteriormente ingresamos la contraseña y confirmamos la contraseña. La contraseña debe tener al menos 12 caractes, debe al menos una letra minúscula, una letra mayúscula, un dígito y un carácter especial.
7. En las "Reglas de puerto de entrada" se deberá dejar abierto el puerto 22 para utilizar SSH (la terminal de secure shell).
8. Dar click en el botón "Siguiente: Discos>" y seleccionar el tipo de disco de sistema operativo, en este caso vamos a seleccionar HDD estándar.
9. Dar click en el botón "Siguiente: Redes>"
10. Dar click en el botón "Siguiente: Administración>" y en el campo "Diagnóstico de arranque" seleccionar "Desactivado".
11. Dar click en el botón "Revisar y crear".
12. Dar click en el botón "Crear".
13. Dar click a la campana de notificaciones (barra superior de la pantalla) para verificar que la maquina virtual se haya creado.
14. Dar click en el botón "Ir al recurso". En la página de puede ver la direción IP pública de la máquina virtual.

Tenemos que repetir este procedimiento 4 veces ya que son el número de máquinas virtuales que ocuparemos en nuestros 4 nodos, donde 4 serán servidores y uno a su vez también cliente que ejercerá una comunicación con las demás máquinas virtuales de la red

Configurando el puerto de salida 1099

Para que nuestras máquinas puedan comunicarse las unas a las otras vamos a tener que agregar algunas reglas de comunicación en las máquinas virtuales y para esto será abrir el puerto de salida 1099 que es el que configuramos para la ejecución de RMIregistry, los pasos que tenemos que seguir son los siguientes:

1. Seleccionar la máquina virtual.
2. Dar click en "Redes".
3. Dar click en el botón "Agregar regla de puerto de entrada".
4. En el campo "Intervalos de puertos de destino" ingresar: 1099
5. Seleccionar el protocolo: TCP
6. En el campo "Nombre" ingresar: Puerto_1099

Repetir este procedimiento con cada una de las máquinas virtuales creadas para que no tengamos problemas de comunicación entre las máquinas

En la **imagen 1** vemos nuestras 4 máquinas virtuales de UBUNTU en Azure y en la **imagen 2** vemos las Ip's públicas y privadas de estas máquinas que se adicionan en nuestro archivo **ClienteRMI.java** en la imagen 3 se ve la parte de código en donde se adicionan estas ip´s públicas

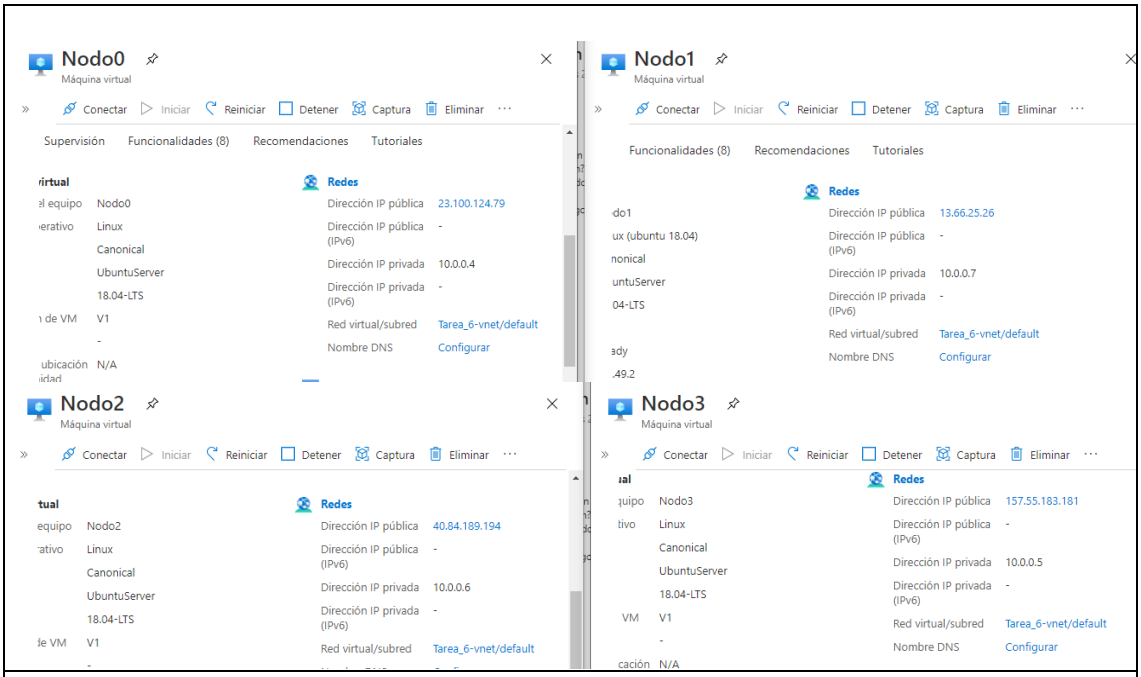


Imagen 1. Máquinas virtuales con Ubuntu y sus Ip's

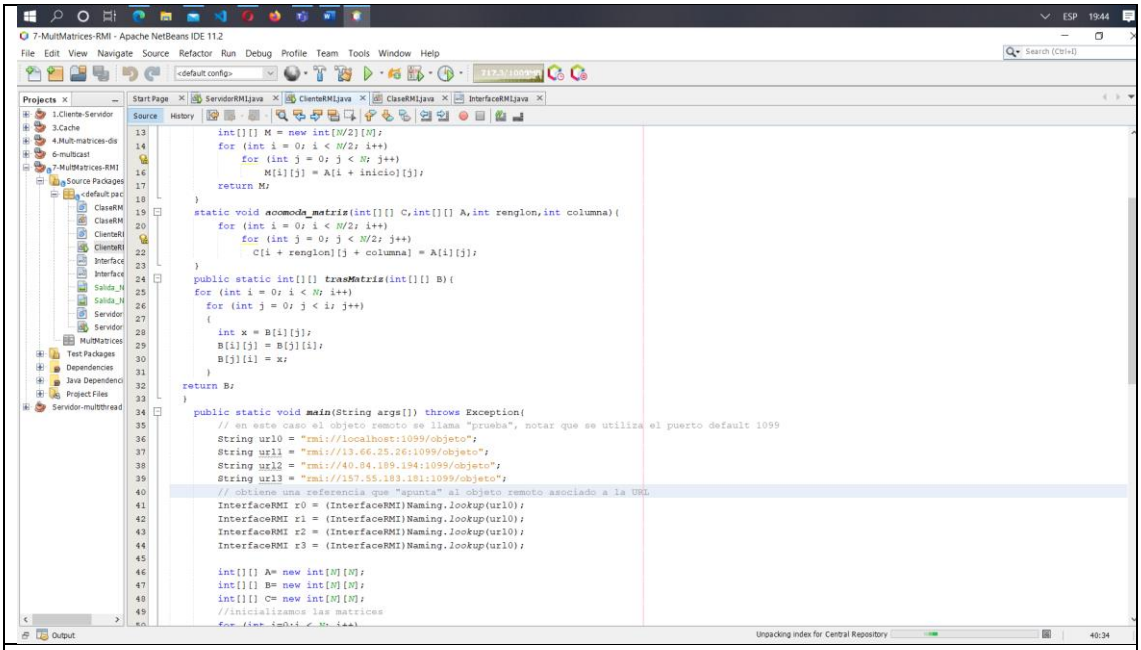
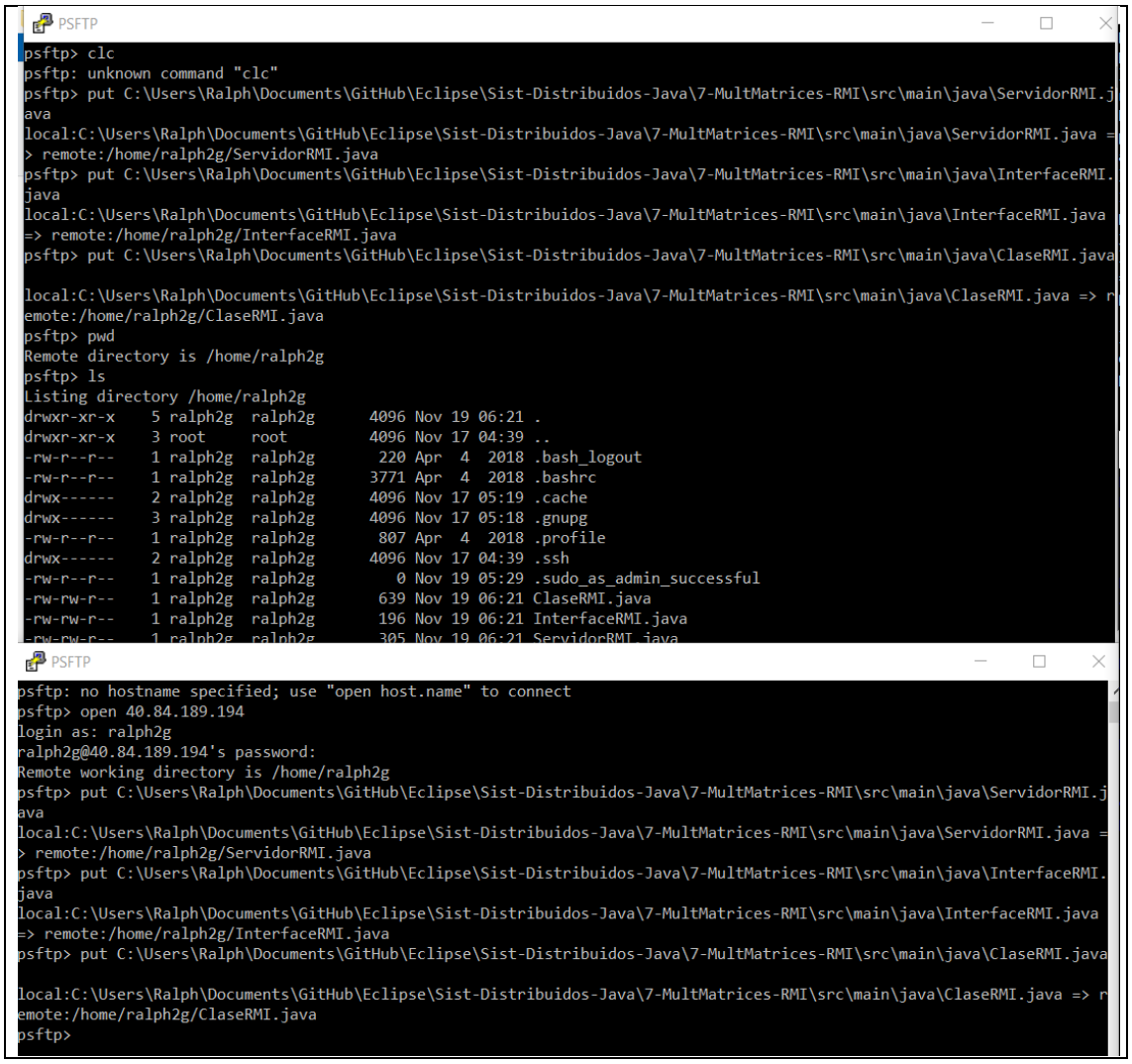


Imagen 2. Adición de las ip's públicas a nuestro archivo **ClienteRMI.java**

Continuando con la ejecución de nuestro programa volvemos a compilar ClienteRMI.java para que los cambios hechos sean efectivos. Posterior a esto pasamos mediante un servidor FTP (en nuestro caso PSFTP) a los nodos **0,1,2,3** nuestros archivos *ClaseRMI.class, InterfaceRMI.class y servidorRMI.class* para ejecutarlos en las máquinas virtuales. Por último pasamos a nuestro **nodo 0** el archivo *Cliente.class* para también ejecutarlo en la máquina virtual y establezca la comunicación con los demás nodos.

N° de Nodo	Ip pública	Ip privada
0	23.100.124.79	10.0.0.4
1	13.66.25.26	10.0.0.7
2	40.84.189.194	10.0.0.6
3	157.55.183.181	10.0.0.5

En la **imagen 3** se puede ver la conexión y transferencia de archivos a nuestra máquina virtual por medio de FTP (PSFTP) a las máquinas virtuales mediante el comando ***put*** + ***la dirección del archivo en la computadora local***


PSFTP
psftp> clc
psftp: unknown command "clc"
psftp> put C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\7-MultMatrices-RMI\src\main\java\ServidorRMI.j
ava
local:C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\7-MultMatrices-RMI\src\main\java\ServidorRMI.java =
> remote:/home/ralph2g/ServidorRMI.java
psftp> put C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\7-MultMatrices-RMI\src\main\java\InterfaceRMI.
java
local:C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\7-MultMatrices-RMI\src\main\java\InterfaceRMI.java
=> remote:/home/ralph2g/InterfaceRMI.java
psftp> put C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\7-MultMatrices-RMI\src\main\java\ClaseRMI.java
local:C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\7-MultMatrices-RMI\src\main\java\ClaseRMI.java => r
emote:/home/ralph2g/ClaseRMI.java
psftp> pwd
Remote directory is /home/ralph2g
psftp> ls
Listing directory /home/ralph2g
drwxr-xr-x 5 ralph2g ralph2g 4096 Nov 19 06:21 .
drwxr-xr-x 3 root root 4096 Nov 17 04:39 ..
-rw-r--r-- 1 ralph2g ralph2g 220 Apr 4 2018 .bash_logout
-rw-r--r-- 1 ralph2g ralph2g 3771 Apr 4 2018 .bashrc
drwx------ 2 ralph2g ralph2g 4096 Nov 17 05:19 .cache
drwx------ 3 ralph2g ralph2g 4096 Nov 17 05:18 .gnupg
-rw-r--r-- 1 ralph2g ralph2g 807 Apr 4 2018 .profile
drwx------ 2 ralph2g ralph2g 4096 Nov 17 04:39 .ssh
-rw-r--r-- 1 ralph2g ralph2g 0 Nov 19 05:29 .sudo_as_admin_successful
-rw-rw-r-- 1 ralph2g ralph2g 639 Nov 19 06:21 ClaseRMI.java
-rw-rw-r-- 1 ralph2g ralph2g 196 Nov 19 06:21 InterfaceRMI.java
PSFTP
psftp: no hostname specified; use "open host.name" to connect
psftp> open 40.84.189.194
login as: ralph2g
ralph2g@40.84.189.194's password:
Remote working directory is /home/ralph2g
psftp> put C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\7-MultMatrices-RMI\src\main\java\ServidorRMI.j
ava
local:C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\7-MultMatrices-RMI\src\main\java\ServidorRMI.java =
> remote:/home/ralph2g/ServidorRMI.java
psftp> put C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\7-MultMatrices-RMI\src\main\java\InterfaceRMI.
java
local:C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\7-MultMatrices-RMI\src\main\java\InterfaceRMI.java
=> remote:/home/ralph2g/InterfaceRMI.java
psftp> put C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\7-MultMatrices-RMI\src\main\java\ClaseRMI.java
local:C:\Users\Ralph\Documents\GitHub\Eclipse\Sist-Distribuidos-Java\7-MultMatrices-RMI\src\main\java\ClaseRMI.java => r
emote:/home/ralph2g/ClaseRMI.java
psftp>

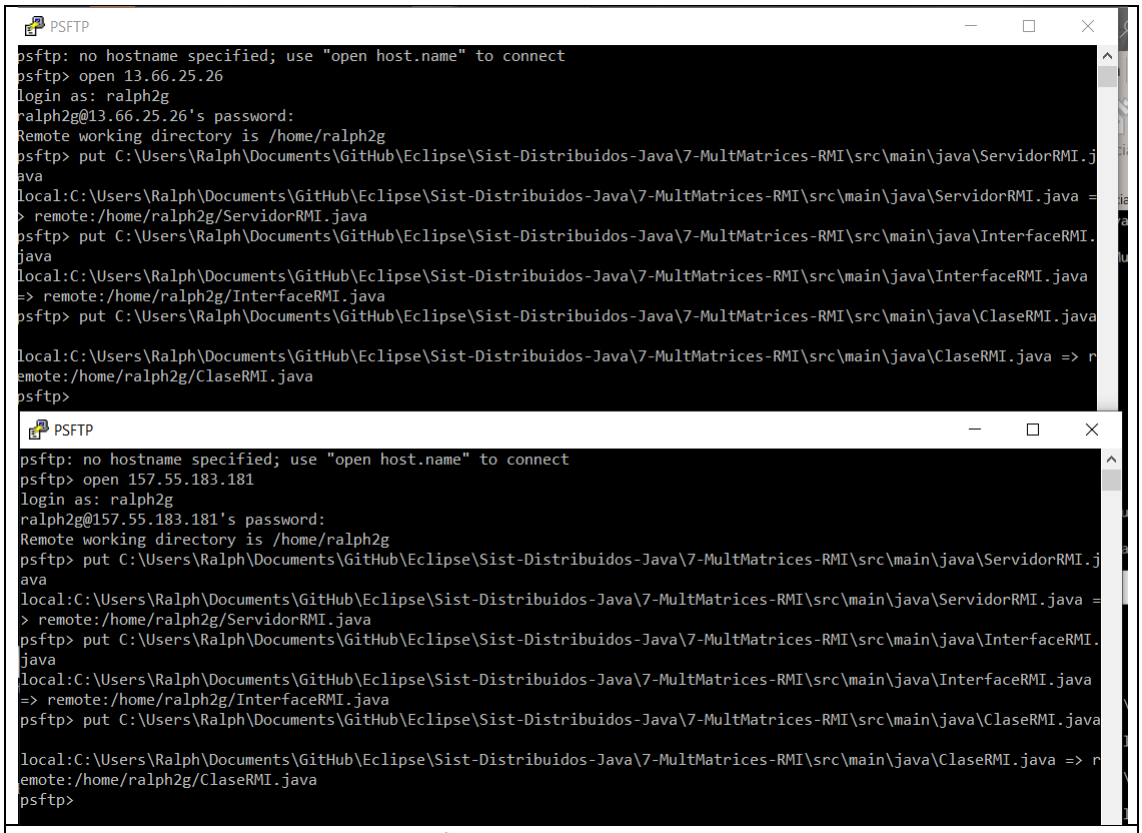


Imagen 3. Transferencia de archivos por medio de FTP

Nos conectamos por medio de SSH (en nuestro caso el programa que usamos es putty) a cada una de nuestras máquinas virtuales, para esto tenemos que ingresar el usuario y contraseña que definimos para la máquina virtual cuando la estuvimos creando. En la **imagen 4** se muestra el enlace de la conexión con estas máquinas

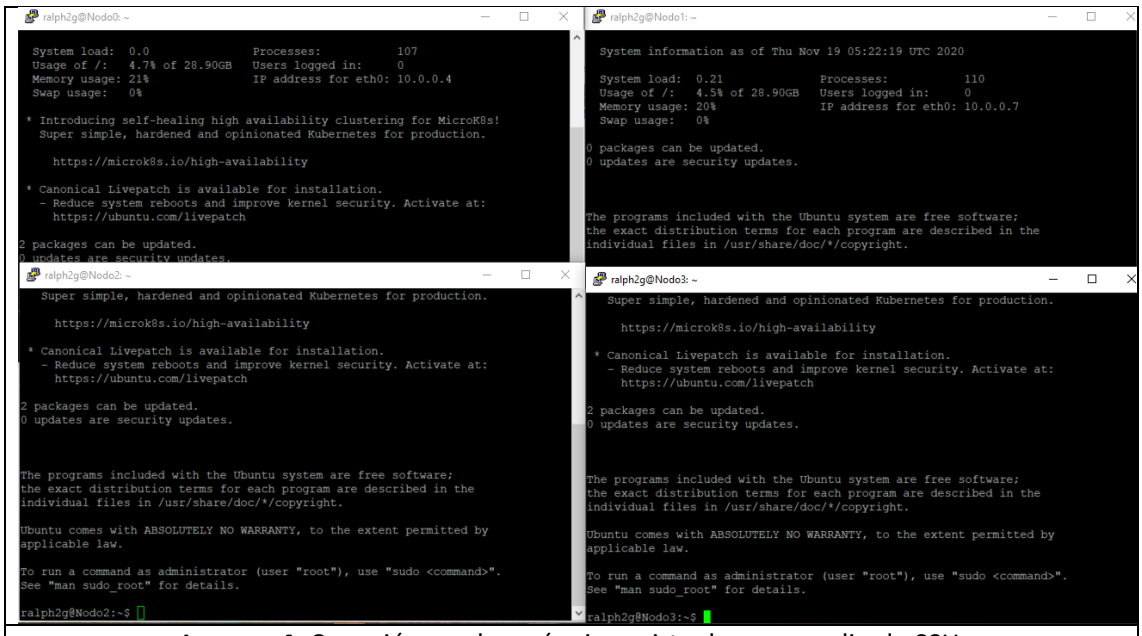


Imagen 4. Conexión con las máquinas virtuales por medio de SSH

En la **imagen 5** vamos a ejecutar en cada uno el programa **rmiregistry** y también vamos a ejecutar el **servidor RMI** en cada uno de los nodos. Además podemos notar que en el nodo 0 también ejecutamos un cliente que será el encargado de mandar la matriz a los demás servidores.

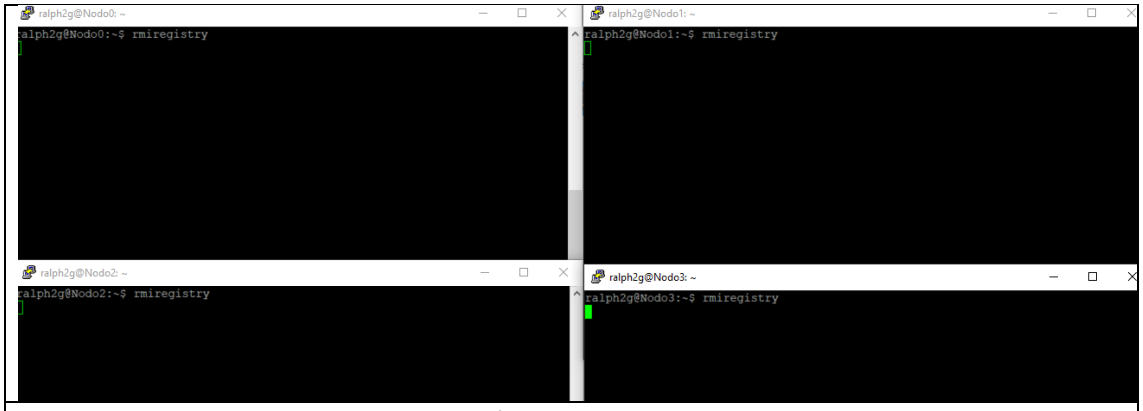


Imagen 5. Ejecución de *rmiregistry* en cada nodo

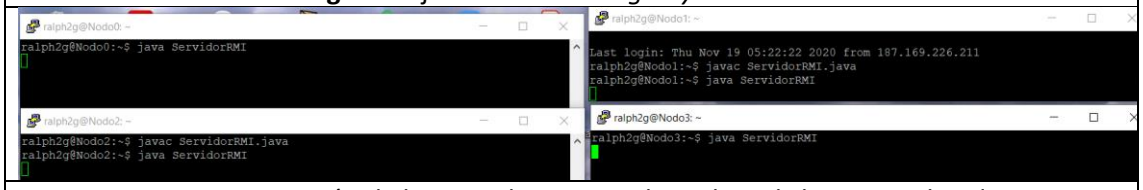


Imagen 6. Ejecución de los servidores en cada nodo y el cliente en el nodo 0

Ejecución del cliente con N=4

Como primera instancia ejecutaremos nuestro cliente con un valor del tamaño de la matriz N=4, como se muestra en la **imagen 7** vamos a tener de salida la matriz C de tamaño NxN la cuál va a tener como salida de **checksum** 272.

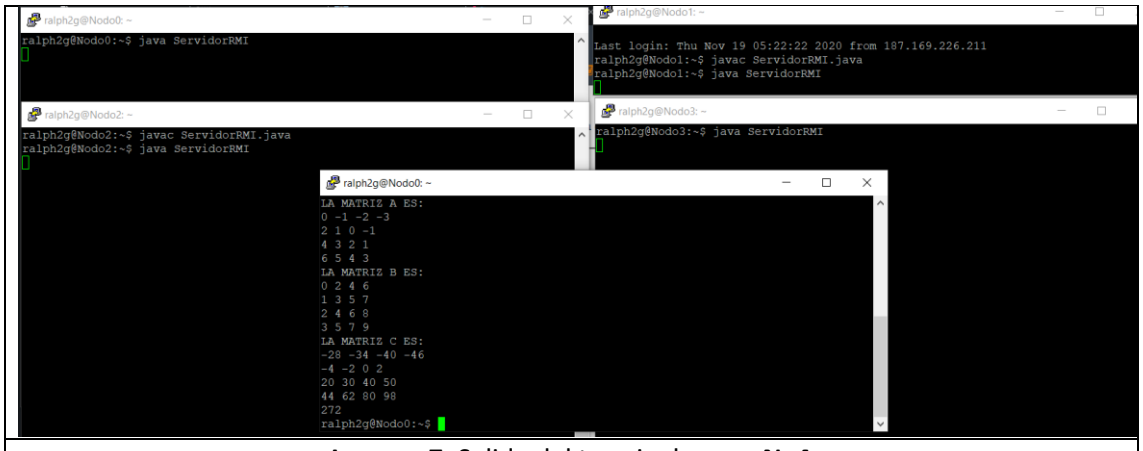


Imagen 7. Salida del terminales con **N=4**

Una vez hayamos terminado de ejecutar nuestro cliente con una matriz de N=4 toca ejecutar el cliente con **N=500**, En la imagen 8 se muestra en la terminal que se abrió el editor de texto Vi para editar el archivo de cliente, se compiló y finalmente se ejecutó para para ver el resultado de la multiplicación de la matriz de 500x500 el cual fue: **18135531250000**

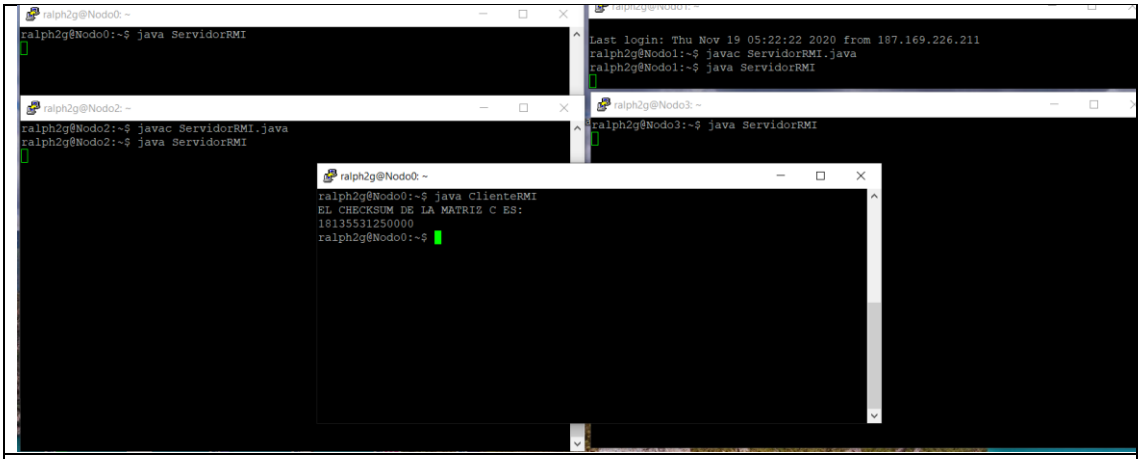


Imagen 8. Salida del terminales con **N=500**

Conclusiones:

La implementación de RMI es un parteaguas en el desarrollo de aplicaciones distribuidas rápido ejecutar todos estos programas en sus respectivos servidores. Creo que el uso de RMI es beneficioso para las aplicaciones distribuidas además de que genera una capa mucho más abstracta que permite que muchas computadoras se comuniquen entre sí.

Respecto a la práctica fue muy padre ver como nuestros servidores ejecutaban un programa de manera distribuida usando simplemente objetos mediante RMI. Creo que el ejemplo manejado en clase fue muy clara para entender el tema y espero que manejemos mucho más esta API para la generación de aplicaciones.