**Instituto Politécnico Nacional.**

**Escuela Superior De Cómputo.**





**Materia:**

**Desarrollo de Sistemas distribuidos.**

**Tema:**

**Multiplicación de matrices usando objetos distribuidos.**

**(Tarea 06)**

**Profesor:**

**Carlos Pineda Guerrero.**

**Alumno:**

**Mario Alberto Miranda Sandoval.**

**Grupo:**

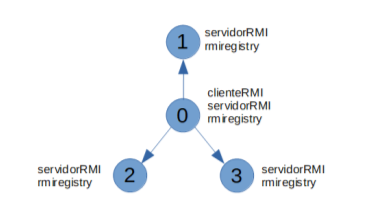
**4CM5**

**Objetivo.**

El alumno deberá desarrollar un sistema que calcule el producto de dos matrices cuadradas utilizando Java RMI.

**Desarrollo.**

Al igual que la tarea 3, se hará la multiplicación de matrices distribuidas, pero esta vez se ejecutará usando RMI, primeramente, se deberá implementar la siguiente topología de red.



Posteriormente, cada nodo es una máquina virtual de Azure, en la cual el sistema operativo que usaremos será Ubuntu.

Para poder realizar la práctica es importante destacar que las máquinas virtuales deben crearse en el mismo grupo de recursos, a modo que se pueda utilizar su dirección IP privada para comunicarse entre ellas.

Una captura de pantalla de una computadora

Descripción generada automáticamente

*Se puede observar como las 4 máquinas virtuales están en el mismo grupo de recursos.*

Una vez que tenemos las máquinas listas para usarse, se procede a explicarse el proceso que se hizo en cada máquina virtual con una máquina de ejemplo.

Una captura de pantalla de una computadora

Descripción generada automáticamente

Como primer paso abrimos una terminal en Linux, donde nos conectaremos de manera remota a la máquina virtual usando ssh, con el siguiente comando:

**ssh <nombre usuario>@<dirección IP pública>**

Nos pedirá la contraseña con la cual creamos la máquina virtual, una vez introducida nos dejará tener acceso a la máquina virtual desde la terminal.

Ahora para obtener el código, se uso el wget donde se hará la petición al repositorio de GitHub, para esto primero se necesita copiar el enlace del repositorio, este puede ser encontrado al entrar al repositorio como se ve en la siguiente imagen.

Una captura de pantalla de una computadora

Descripción generada automáticamente

Una vez copiado el enlace, procedemos a la máquina virtual para usar el wget.

Una captura de pantalla de una computadora

Descripción generada automáticamente

En la imagen se puede observar el comando, que tiene la siguiente estructura.

**wget <enlace al repositorio (quitar el .git)>/archive/<nombre y formato en el que se va a descargar>**

Una captura de pantalla de una computadora

Descripción generada automáticamente

Una captura de pantalla de una computadora

Descripción generada automáticamente

*Se puede ver la descarga del código del repositorio, si usamos el comando ls, podremos observar que tenemos el archivo tar.gz que se especificó.*

Ahora para extraer los archivos usamos el siguiente comando:

**tar -xvf <nombre\_archivo.tar.gz>**

**Una captura de pantalla de una computadora

Descripción generada automáticamente**

Una vez ejecutado el comando tar, nos quedaran los archivos que descargamos del repositorio en la máquina virtual, así como se ve en la siguiente imagen.

Una captura de pantalla de una computadora

Descripción generada automáticamente

Ahora al ser máquinas virtuales recién creadas estas no poseen java instalado, por lo que correremos los dos siguientes comandos en el orden que son mostrados.

**sudo apt update**

**sudo apt-get install openjdk-11-jdk-headless**

El primer comando es para actualizar los paquetes que tiene Ubuntu, mientras que el segundo comando es usado para instalar el jdk con la configuración del compilador.

En la siguiente imagen se aprecia como la máquina virtual cuenta ya con Java y su compilador.

Una captura de pantalla de una computadora

Descripción generada automáticamente

Ese proceso es el que se repetira para cada máquina virtual que tengamos.

Ahora, en las cuatro máquinas virtuales que hemos creado y configurado previamente, en el archivo **ServidorRMI.java, se colocará la dirección IP privada correspondiente a cada máquina virtual.**

Captura de pantalla de computadora

Descripción generada automáticamente

Y en el archivo **ClienteRMI.java, en el método Naming.lookup se colocará la misma cadena que se colocó a cada servidor.**

Texto

Descripción generada automáticamente

*Nota: En la imagen hay un error, debido a que donde se hace el cast debe ser con la interfaz no con el objeto remoto, este error fue corregido para el funcionamiento de la práctica.*

Captura de pantalla de computadora

Descripción generada automáticamente Captura de pantalla de computadora

Descripción generada automáticamente

Antes de pasar a la compilación y ejecución, mediante un ping se comprueba la conectividad de las máquinas virtuales.

Ahora, procedemos a levantar el servicio de rmiregistry en cada máquina con el comando **rmiregistry &,** además, compilamos y ponemos en ejecución los programascomo se ve en las siguientes imágenes.

Texto

Descripción generada automáticamente Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente Texto

Descripción generada automáticamente

*Los servidores deben correrse primero, mientras que el cliente se correra después.*

Texto

Descripción generada automáticamente

Como el cliente se corre en el nodo 0, es necesario crear una nueva terminal para conectarse al nodo 0, pero al ser la misma máquina, se puede omitir la compilación de los archivos.

Texto

Descripción generada automáticamente

*Ejecución del cliente, con N = 4.*

Texto

Descripción generada automáticamente

Ajustamos el cambio a probar con N = 500.

Una captura de pantalla de una computadora

Descripción generada automáticamente

*Recompilamos y ejecutamos para el programa con N = 500, se puede ver el comando nano, que nos sirve para abrir y modificar archivos desde terminal.*

**Conclusiones.**

En esta práctica se puede notar a la perfección la diferencia de usar objetos remotos, ya que cuando esta misma actividad se hizo en la tarea 3, usar sockets hizo que fuera una tarea más complicada ya que él envió de los datos lo hacían una tarea complicada.

El usar objetos remotos nos garantiza una forma más sencilla de realizar la tarea además de que la integridad de los datos se garantiza de una mejor manera en contraste a enviar datos por sockets, incluso esto se ve reflejado en la cantidad de líneas de código que se usaron para el programa.

Aunque se documento en el reporte del error en su respectiva imagen, decidí incluirlo ya que cuando me salto ese error y buscar como solucionarlo, vi que era un error algo común por eso tome la decisión de dejar la imagen con el error y hacer su respectiva observación.