

Planteamiento del problema

Desarrollar un en Java el cual calculará la multiplicación de dos matrices cuadradas en forma distribuida sobre cuatro nodos.

Sean A, B y C matrices cuadradas con elementos de tipo double, N renglones y N columnas, N par y $C=AxB$.

Se deberán ejecutar dos casos:

1. N=8, desplegar las matrices A, B y C y el checksum de la matriz C.
2. N=1000, desplegar el checksum de la matriz C.

El checksum de la matriz C se calculará como la suma de todos los elementos de la matriz.

El checksum deberá ser de tipo Double.

$$\text{checksum} = \sum C[i][j], i=0,\dots, N-1, j=0,\dots, N-1.$$

Se deberá inicializar las matrices de la siguiente manera:

$$A[i][j] = i + 5*j$$

$$B[i][j] = 5*i - j$$

Donde $A[i][j]$ y $B[i][j]$ son los elementos A_{ij} y B_{ij} , respectivamente.

El programa deberá ser ejecutado en cuatro máquinas virtuales con Ubuntu (1 CPU, 1GB de RAM y disco HDD estándar) en cada máquina virtual se pasará como parámetro al programa el número de nodos, a saber: 0, 1, 2 y 3.

El nombre de cada máquina virtual deberá ser: "Tarea-3-", concatenando el número del equipo, un guión y el número de nodo. Por ejemplo si el equipo es 12 entonces el nodo 0 deberá llamarse: Tarea-3-12-0, el nodo 1 deberá llamarse Tarea-3-12-1, y así sucesivamente. No se admitirá la tarea si los nodos no se nombran como se indicó anteriormente.

Recuerden que deben eliminar las máquinas virtuales cuando no las usen, finalidad de ahorrar el saldo de sus cuentas de Azure.

¿Cómo realizar la multiplicación de matrices en forma distribuida?

Suponga que divide la matriz A en las matrices A1 y A2. El tamaño de las matrices A1 y A2 es N/2 renglones y N columnas.

La matriz B se divide en las matrices B1 y B2.

El tamaño de matrices B1 y B2 es N renglones y N/2 columnas.

Entonces la matriz $C=A \times B$ se compone de las matrices C1, C2, C3 y C4, tal como se muestra en la siguiente figura:



Donde:

$$C1 = A1 \times B1$$

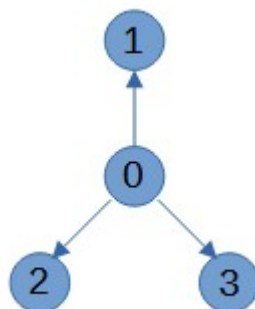
$$C2 = A1 \times B2$$

$$C3 = A2 \times B1$$

Debido a que las matrices se guardan en memoria por renglones, es más eficiente transponer la matriz B y dividirla de la siguiente manera:



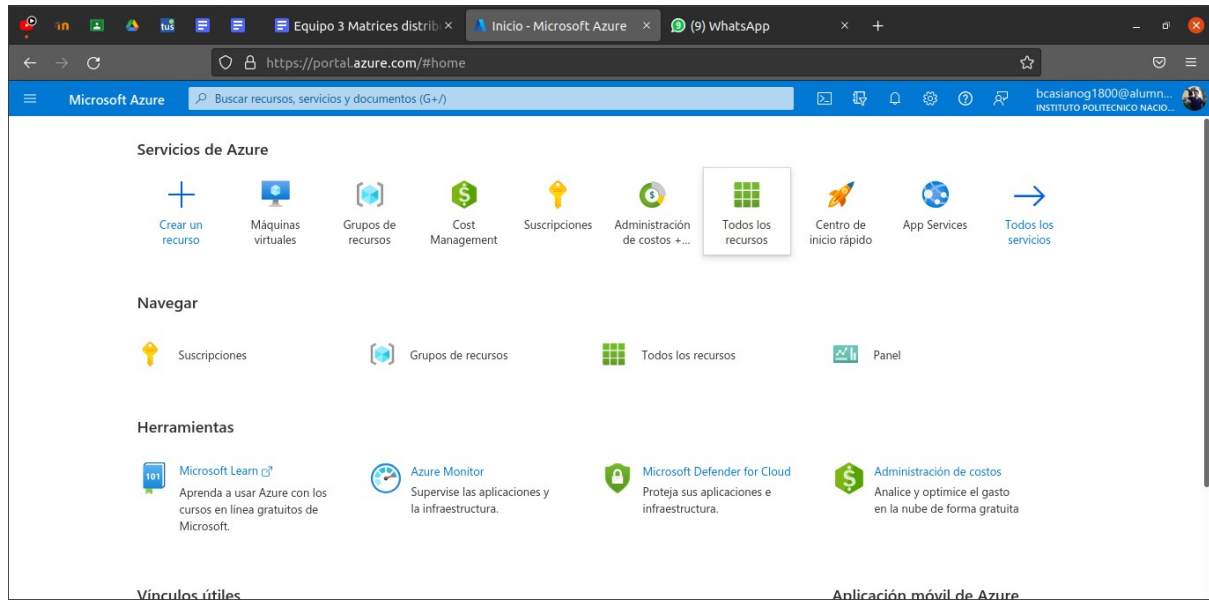
Ahora supongamos que tenemos cuatro nodos identificados con los números 0, 1, 2 y 3, el nodo 0 actuará como cliente y los nodos 1, 2 y 3 como servidores.



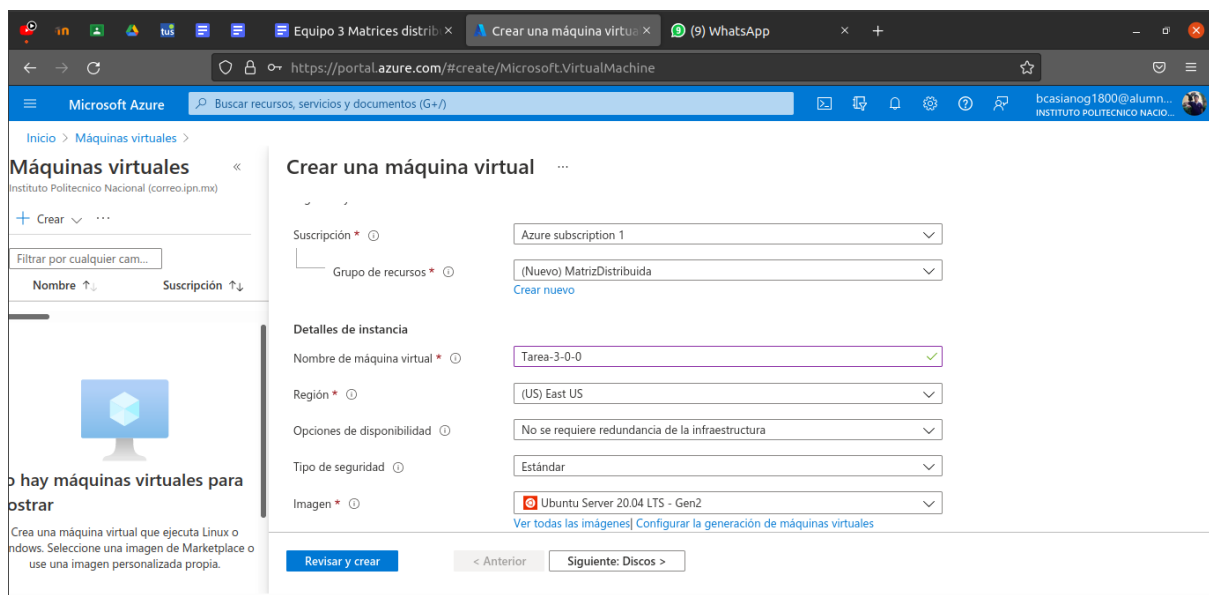
Desarrollo de la tarea

Creación de la máquina virtual de Nodo 0 con Ubuntu

- En esta imagen de abajo está el menú principal de Azure, se selecciona la opción de Máquinas Virtuales.



- Se coloca el nombre de la máquina virtual en este caso es "Tarea-3-0-0", la dejamos en Estados Unidos y escogemos Ubuntu Server.
- Nota "Ya está seleccionado 1 Gb de Ram como se especifica en el documento de entrega".



- Para la autenticación con la máquina virtual será por medio de usuario y contraseña, los cuales se colocan en sus respectivos campos, después damos clic en “Siguiente: Discos”.

Microsoft Azure

Inicio > Máquinas virtuales >

Máquinas virtuales

Instituto Politécnico Nacional (correo.ipn.mx)

+ Crear

Filtrar por cualquier cam...

Nombre ↑ Suscripción ↓

¿No hay máquinas virtuales para mostrar?

Crea una máquina virtual que ejecuta Linux o Windows. Seleccione una imagen de Marketplace o use una imagen personalizada propia.

Crear una máquina virtual

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 >

- En discos seleccionamos el “HDD estándar”, como se especificó en el documento, posterior se da clic en “siguiente: Redes” y “Siguiente: Administración”.

Microsoft Azure

Inicio > Máquinas virtuales >

Máquinas virtuales

Instituto Politécnico Nacional (correo.ipn.mx)

+ Crear

Filtrar por cualquier cam...

Nombre ↑ Suscripción ↓

¿No hay máquinas virtuales para mostrar?

Crea una máquina virtual que ejecuta Linux o Windows. Seleccione una imagen de Marketplace o use una imagen personalizada propia.

Crear una máquina virtual

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

Las máquinas virtuales de Azure tienen un disco de sistema operativo y un disco temporal para el almacenamiento a corto plazo. Puede asociar discos de datos adicionales. El tamaño de la máquina virtual determina el tipo de almacenamiento que puede usar y la cantidad de datos que permiten los discos. [Más información](#)

Opciones de disco

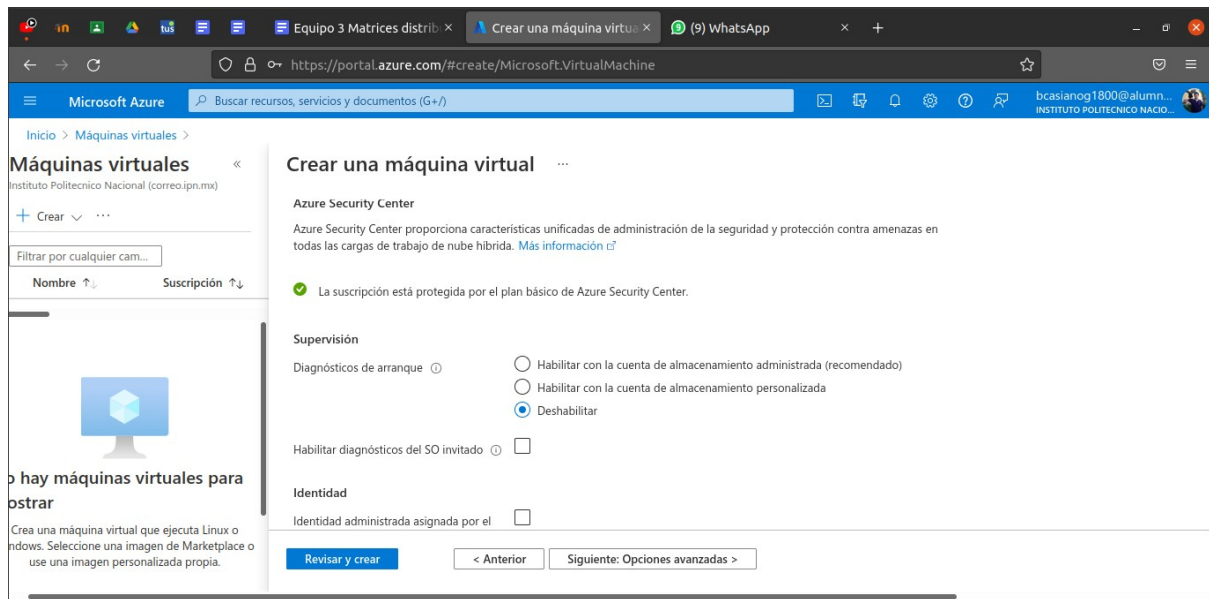
Tipo de disco del sistema operativo *

Elija discos SSD Premium para reducir la latencia, IOPS y el ancho de banda más altos, y expansión de disco. Las máquinas virtuales de instancia única con discos SSD Premium cumplen el SLA de conectividad de 99,9 %. [Más información](#)

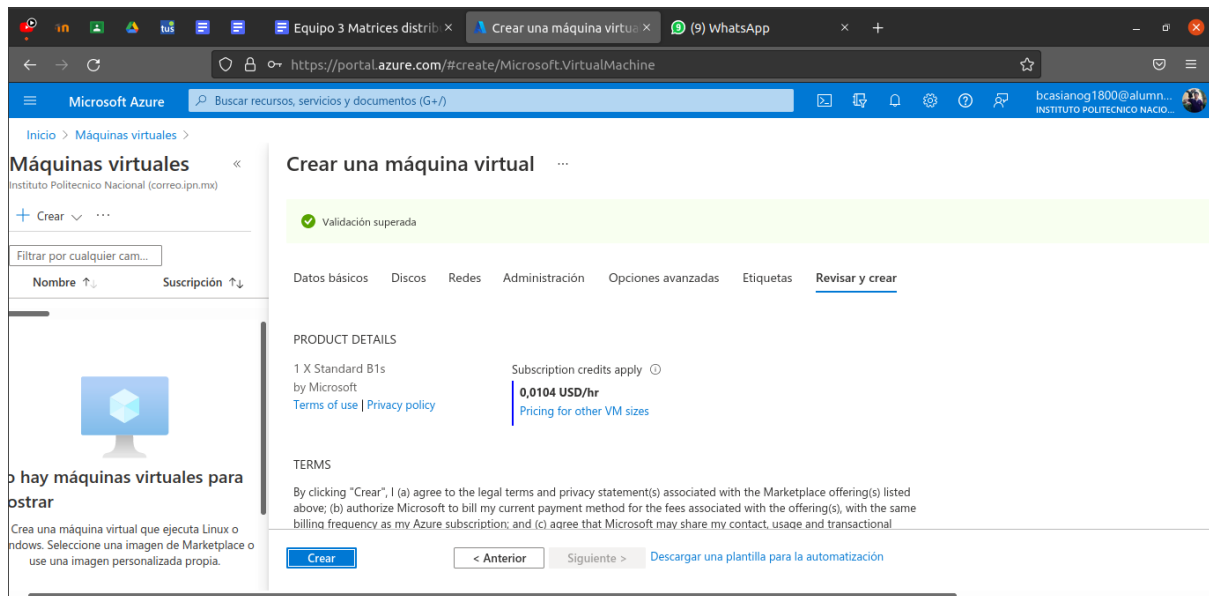
Eliminar con VM ☒

Cifrado en el host ☐

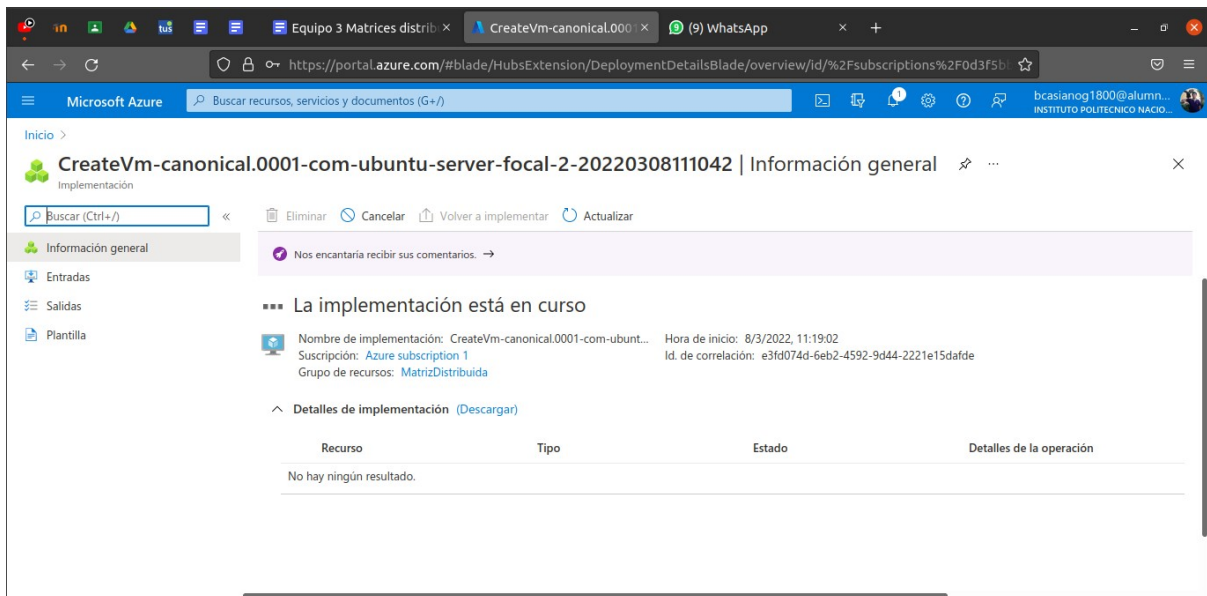
[Revisar y crear](#) < Anterior Siguiente: Redes >



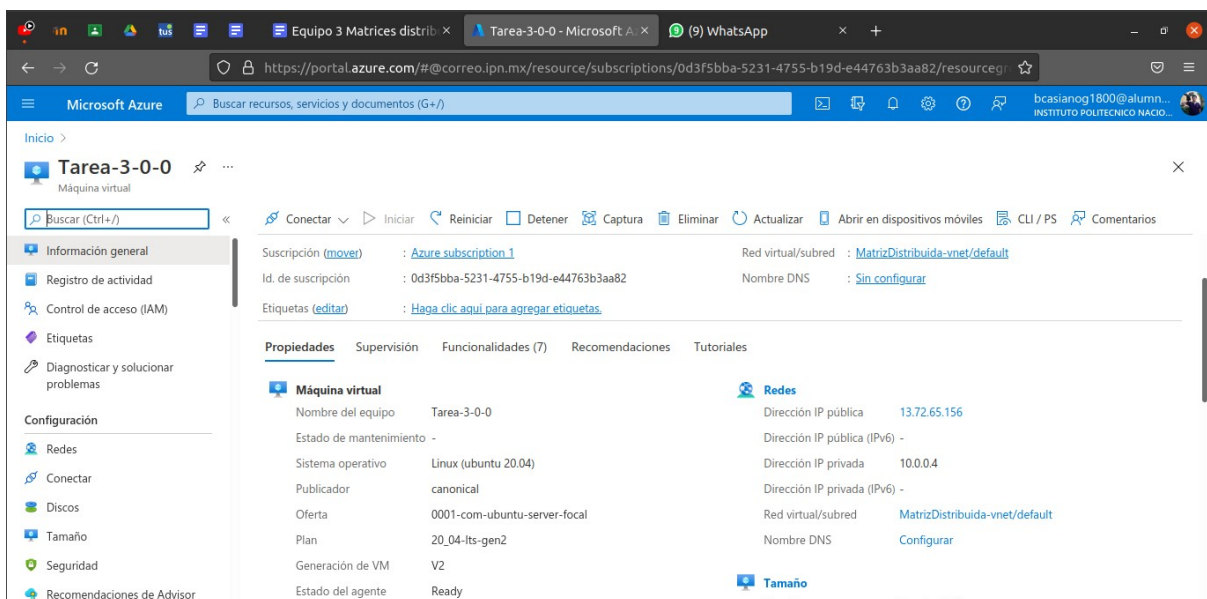
- Deshabilitamos el diagnóstico de arranque, damos clic en “Crear”.



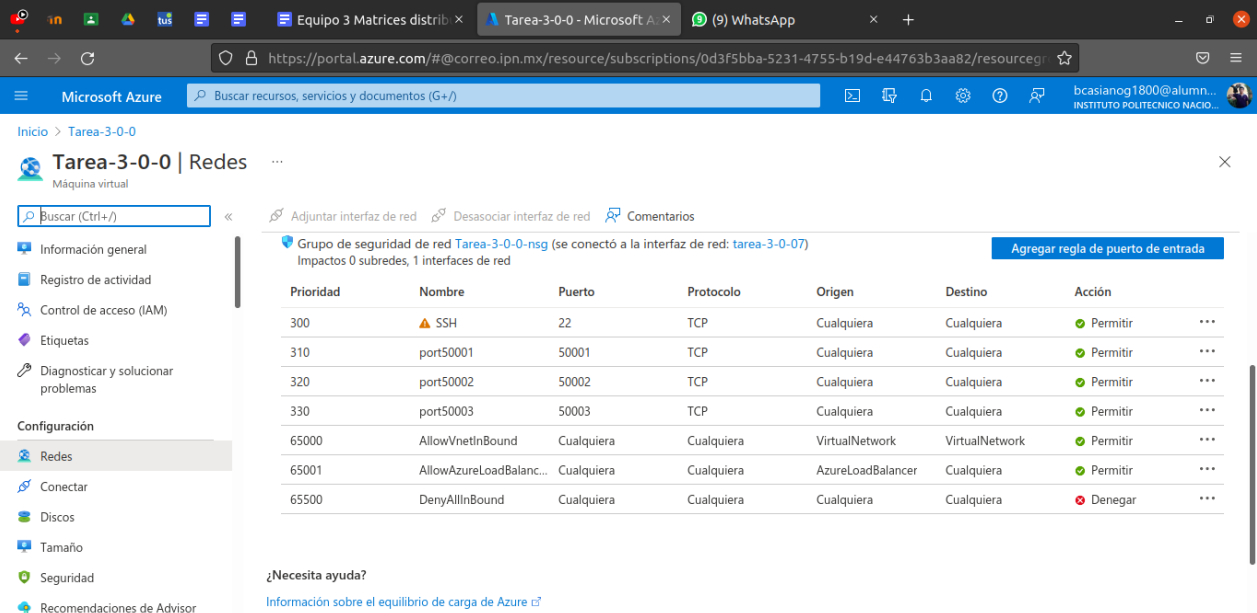
- Imagen donde se puede ver la creación de la máquina virtual en Azure.



- Podemos ver que la máquina se creó correctamente y está corriendo.



1. Nota: por defecto solo está abierto el puerto ssh para conexiones remotas, por lo que se deben abrir los puertos pertinentes para que se puedan crear las conexiones con sockets en este programa son los puertos son 50000+nodo, es decir, el nodo 1 abra el puerto 50001, el nodo 2 abrirá el puerto 50002 y el nodo 3 abrirá el puerto 50003. En esta imagen podemos ver como se debe de abrir un puerto, de la siguiente manera:
 - 1.1. Se debe colocar en la opción de “Redes” de la máquina virtual.
 - 1.2. Se da clic en “Agregar regla de puerto de entrada”.
 - 1.3. En la ventana nueva, se coloca que será de tipo TCP.
 - 1.4. En la misma ventana se coloca el puerto de salida, indicando el número.
 - 1.5. se agrega.
 - 1.6. Se deben de ver como el de la imagen siguiente para la apertura de todos los nodos como ejemplo.



Microsoft Azure portal showing the configuration for a virtual machine named "Tarea-3-0-0". The "Redes" (Network) section is selected, displaying the "Grupo de seguridad de red" (Network Security Group) configuration. The table below shows the configured rules for the network security group.

Prioridad	Nombre	Puerto	Protocolo	Origen	Destino	Acción
300	SSH	22	TCP	Cualquiera	Cualquiera	Permitir
310	port50001	50001	TCP	Cualquiera	Cualquiera	Permitir
320	port50002	50002	TCP	Cualquiera	Cualquiera	Permitir
330	port50003	50003	TCP	Cualquiera	Cualquiera	Permitir
65000	AllowVnetInBound	Cualquiera	Cualquiera	VirtualNetwork	VirtualNetwork	Permitir
65001	AllowAzureLoadBalanc...	Cualquiera	Cualquiera	AzureLoadBalancer	Cualquiera	Permitir
65500	DenyAllInBound	Cualquiera	Cualquiera	Cualquiera	Cualquiera	Denegar

Explicacion del codigo

Para empezar con la explicacion del codigo, primero es necesario conocer cual es el funcionamiento de cada una de las variables globales que en esta práctica serían las siguientes:

Nota: Para probar el código ya sea con matrices de dimensiones $N=8$ o $N=1000$, es necesario modificar el valor de N en el código, para después compilarlo y finalmente ejecutarlo.

- ❖ `int N`: Es para establecer el número de filas y columnas que tendrán las matrices A, B y C.
- ❖ `ServerSocket servidor`: Será el socket para iniciar el servidor del nodo 1, 2 y 3.
- ❖ `Socket Cliente`: Es el socket del cliente para establecer conexión con los servidores.
- ❖ `DataOutputStream salida`: Es el canal de salida para que los servidores puedan enviar el resultado de su sumatoria al cliente.
- ❖ `double Nodos[]`: Este arreglo de longitud 4 lo usaremos para guardar la sumatoria de cada servidor una vez que sea enviada al cliente.
- ❖ `DataInputStream entrada`: Es el canal de entrada para que nuestro cliente pueda recibir las sumatorias por parte de los servidores.
- ❖ `double A[][]` y `B[][]`: Son las matrices que se usarán para la multiplicación, cuyo número de columnas y filas es igual, tomando el valor de N .
- ❖ `double A1, A2, B1, B2`: Estos arreglos bidimensionales son el resultado de dividir la matriz A y B por la mitad, haciendo que estas tengan N número de columnas pero solo la mitad de N filas.
- ❖ `double C1, C2, C3, C4`: Recordando que antes de dividir B sacamos su transpuesta para asignarle esos nuevos valores a B, pues al realizar las operaciones $A1 \times B1, A2 \times B1$, etc., tenemos que el resultado es una matriz de $(N/2) \times (N/2)$ dimensiones, es el porqué de las dimensiones de estos arreglos, los cuales contendrán el resultado de dichas operaciones.
- ❖ `double AS` y `BS`: Estos arreglos bidimensionales son para que el servidor pueda mandar a llamar al método `multiplicar` mandando como parámetro estas dos matrices que representan una mitad de A y una de B, dependiendo del número de nodo que se trate.
- ❖ `double Mult[][]`: Se usa en el método `multiplicar` para guardar el valor de las multiplicaciones de las matrices $A1, A2, B1$ y $B2$, para que en el caso del servidor pueda enviar el resultado al cliente y en el caso del cliente conservar el resultado de la multiplicación que le tocó hacer ya que este no debe de ser enviado, ya que solo el cliente tiene acceso.

- ❖ String host[]: Debido a que cada máquina virtual es independiente es necesario asignarles un host al momento de tratar de conectarnos a las máquinas que hacen de servidores (nodo 1,2 y 3), pidiendo al inicio de ejecución del cliente (nodo 0) que se ingrese el host de cada nodo servidor.

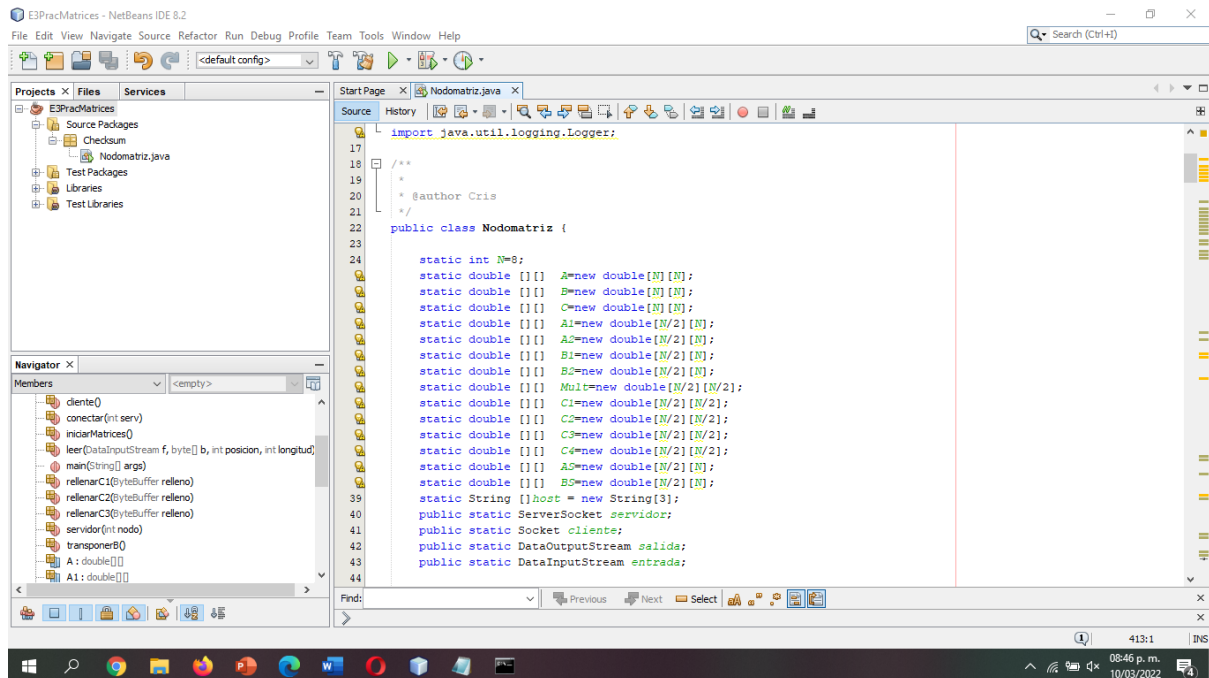


Imagen 1: Captura de las variables globales del programa

Metodo main ()

El método main se encarga de recuperar los argumentos e número de nodo que a partir de este determinara se trata de un cliente (0) o un servidor (1,2 o 3), una vez que se haya determinado si es cliente o servidor se procede a ejecutar sus métodos correspondientes cliente() o servidor(). En caso de ingresar argumentos el programa muestra un mensaje denotando esto y el programa se cierra, caso de que el número de nodo no sea 0,1,2 o 3 el programa dará otro mensaje de error y procederá a terminar su ejecución.

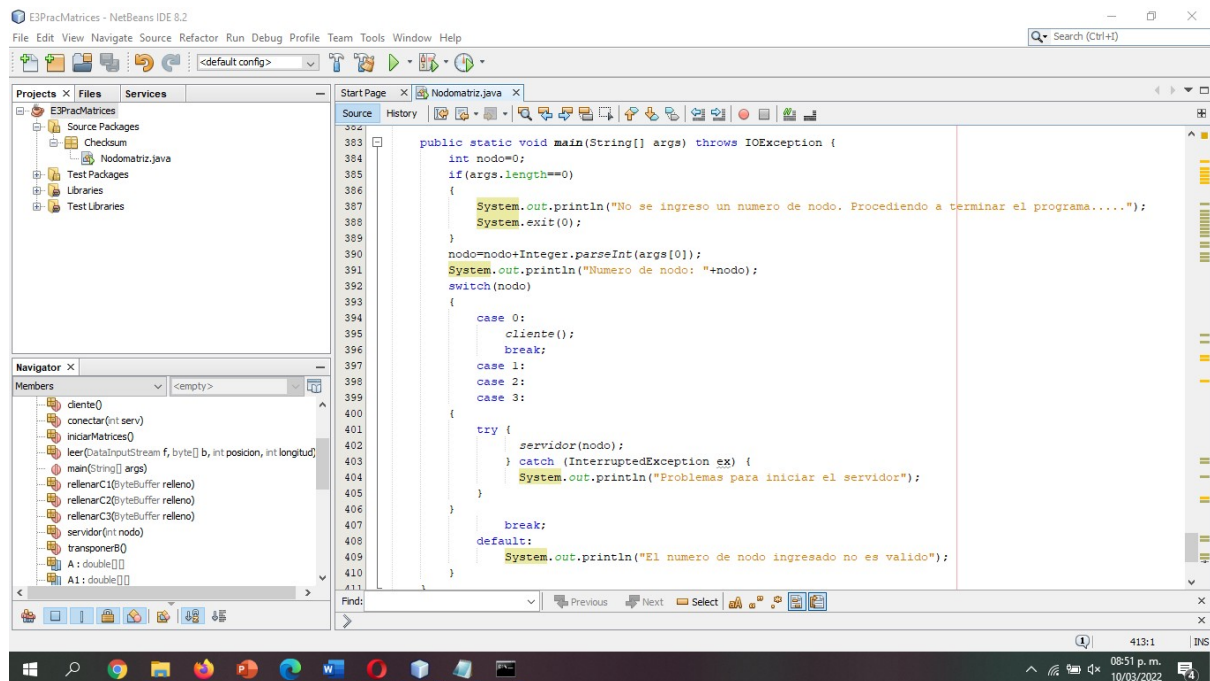


Imagen 2: Captura del método main

Método iniciarMatrices()

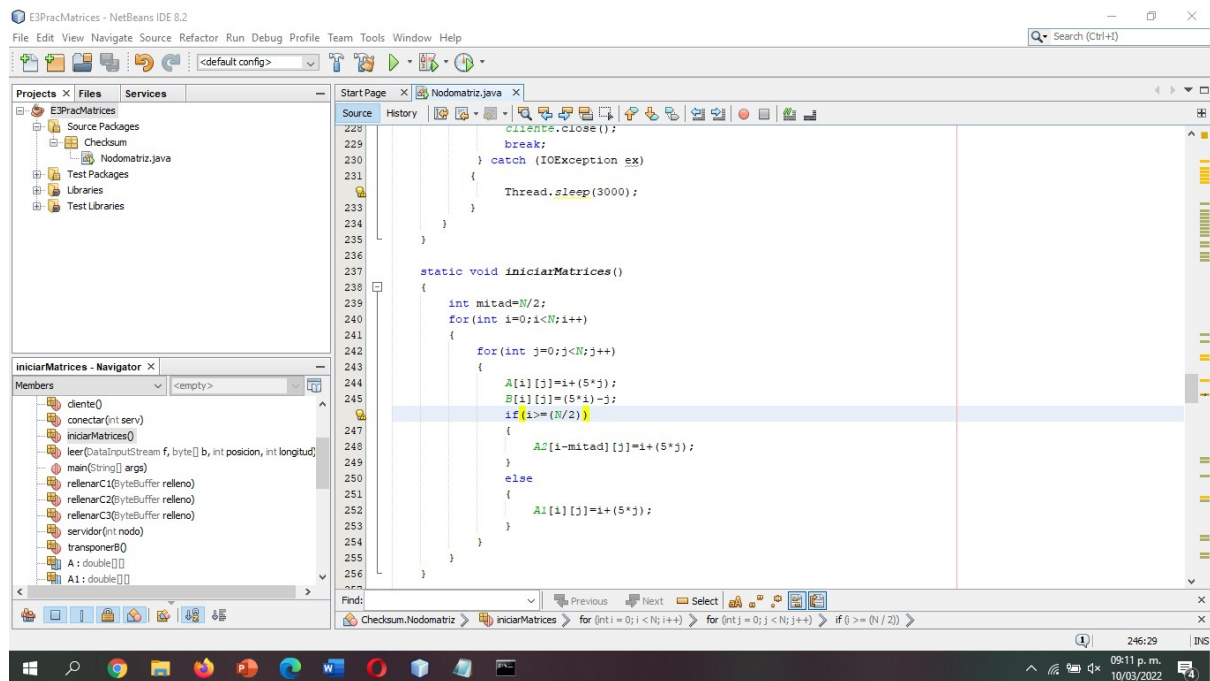
Este método usa la variable global N para crear un ciclo anidado, donde el ciclo principal es para las filas y el ciclo anidado es para las columnas, ambos teniendo en común que el contador inicia en 0, incrementa en 1 y la condición para terminar es que el contador debe de ser menor a N. Esto permite crear las matrices A y B de dimensiones NxN lo único que hay que hacer dentro de este ciclo anidado es asignar a cada elemento de ambos arreglos bidimensionales el valor que les corresponde de acuerdo a la fórmula que se establece en la tarea, donde:

$$A[i][j] = i + (5*j);$$

$$B[i][j] = (5*i) - j;$$

Además, debido que para sacar A1 y A2 solo necesitamos dividir A a la mitad del lado de las filas, por lo que podemos en este mismo método obtener A1 y A2, solo hay que verificar cuando i es mayor a 3 ya que se podría considerar que va en la segunda mitad de la matriz A por lo que dicho valor de la celda que se iniciando en ese momento no solo aplica a A sino también a A2, en caso de sea menor a (N/2) (es decir nos encontramos en la primera mitad) además de A también asignamos el valor a A1, este análisis de tenemos que iniciar un valor en A1 o A2 lo hacemos gracias a los if que verifica la condición antes mencionada. A parte hay que recordar que A2 su última fila es la de índice [(N/2)-1], por lo que para evitar un desbordamiento ya que su caso se activa cuando $i > (N/2)$, es decir la

mitad del número de filas) a dicho índice de fila de A2 le restamos ($N/2$), momento de asignar el valor.



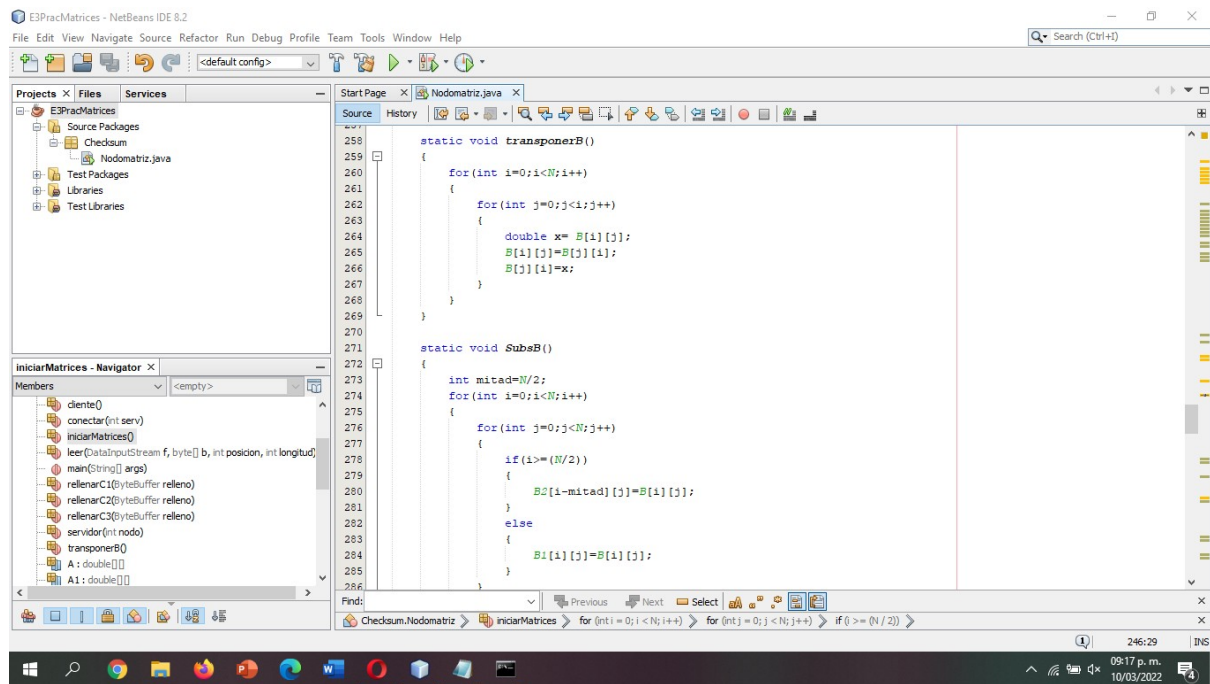


Imagen 3: Captura del método transponer.

Método SubsB()

Funciona de de forma similar a inicializarMatrisélelo que en este caso ya contamos con los valores de la matriz B y solo habrá que asignar el valor de la celda de B en la que nos encontremos de acuerdo al ciclo a una de las submatrices ya sea B1 o B2, por lo que solo es necesario verificar si el contador del ciclo principal es menor a la mitad de N ($N/2$) para saber que nos encontramos en la primera mitad o sea B1 y en caso contrario se trataría de B2 y así conservamos los valores de la matriz B pero ahora divididos en dos matrices de ($N/2$) x N dimensiones.

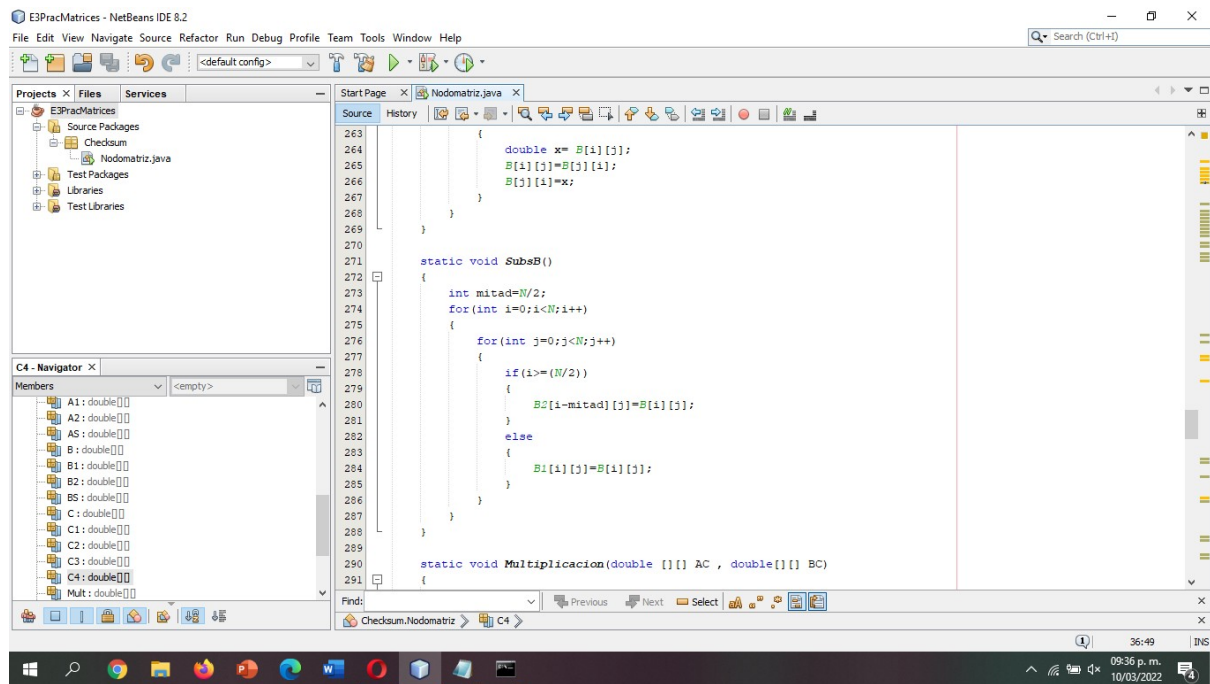


Imagen 4: Captura del método SubsB.

Método Multiplicar()

Debido a que sacamos la transpuesta de B antes de dividirla nos es posible realizar la multiplicación de las submatrices ($A1 \times B1$, $A2 \times B2$, $A1 \times B2$ o $A2 \times B1$) multiplicando cada renglón de una submatriz sea A1 o A2 por cada renglón de la submatriz ya sea la B1 o la B2. Nos apoyamos de una variable de tipo entero que calcula la mitad de N ya que recordemos que estas submatrices solo cuentan con la mitad de filas y el producto de dichas submatrices nos da una matriz de dimensiones $(N/2) \times (N/2)$ y dependiendo de la multiplicación es el arreglo bidimensional que usaremos (puede ser C1, C2, C3 o C4).

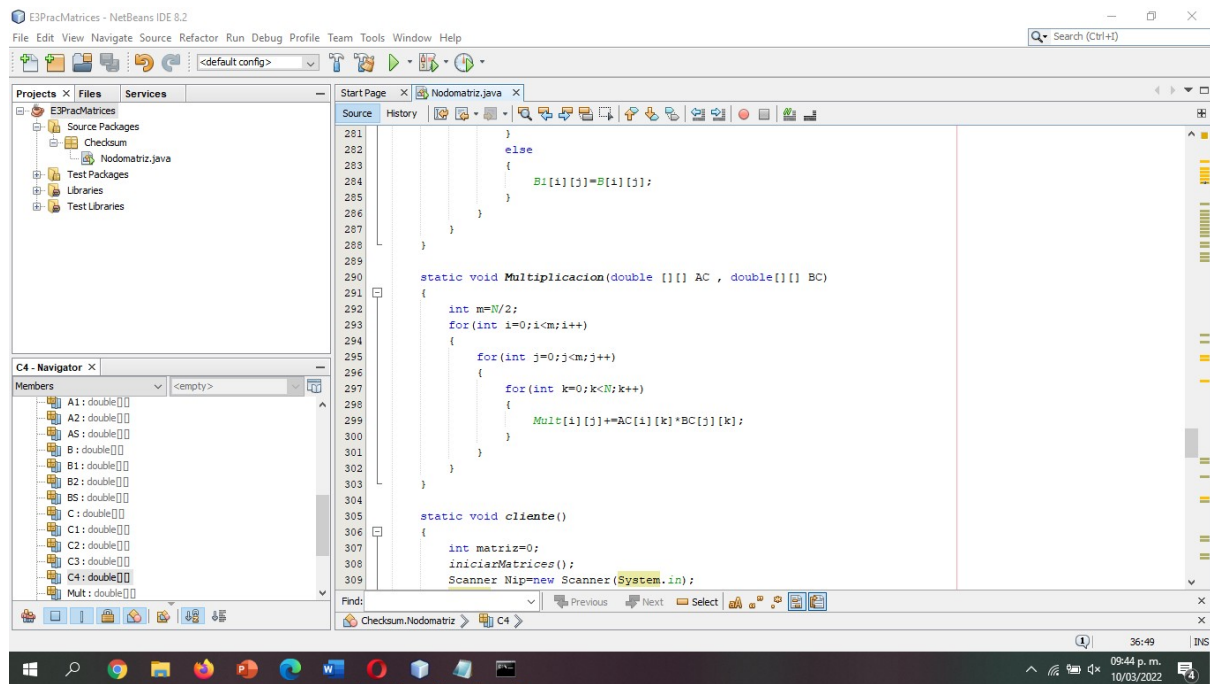


Imagen 5: Captura del método Multiplicación.

Métodos rellenar C1() , rellenar C2() y rellenar C3()

Estos métodos funcionan de forma idéntica, que se cuenta con un ByteBuffer de tipo double como parámetro, el cual contiene el resultado de haber multiplicado alguna submatriz de A (A1 o A2) con alguna submatriz de B (B1 o B2), lo único que se hacemos es sacar cada valor del buffer e insertarlo en la celda de la submatriz (C1, C2 o C3) insertando los valores mediante los renglones, es decir, que una vez terminado con insertar los valores de cada renglón de la fila 0 pasamos con los renglones de la fila 1. Para determinar que submatriz de C deseamos rellenar debemos estar conscientes de que multiplicación se hizo (A1 x B1, A1 x B2 o A2 x B1) y a partir de eso elegir ya sea rellenar C1, C2 o C3. C4 no se rellena ya que esa multiplicación la realiza el cliente y basta con asignarle a C4 los valores de la matriz Mult.

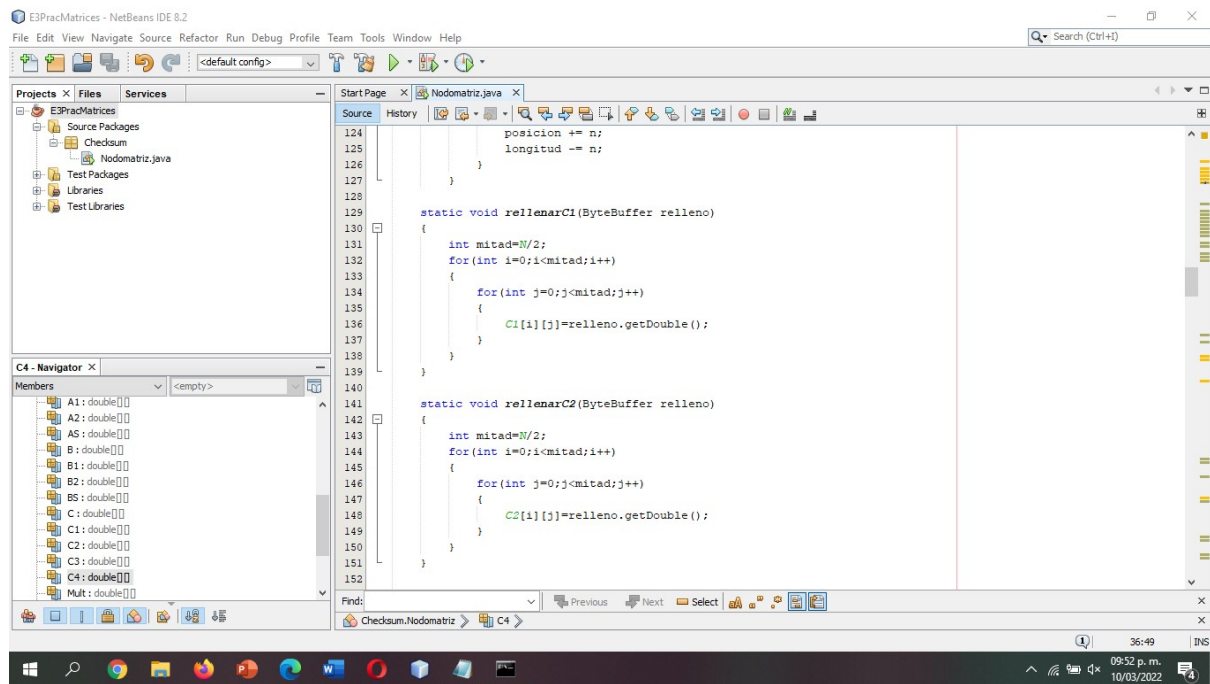
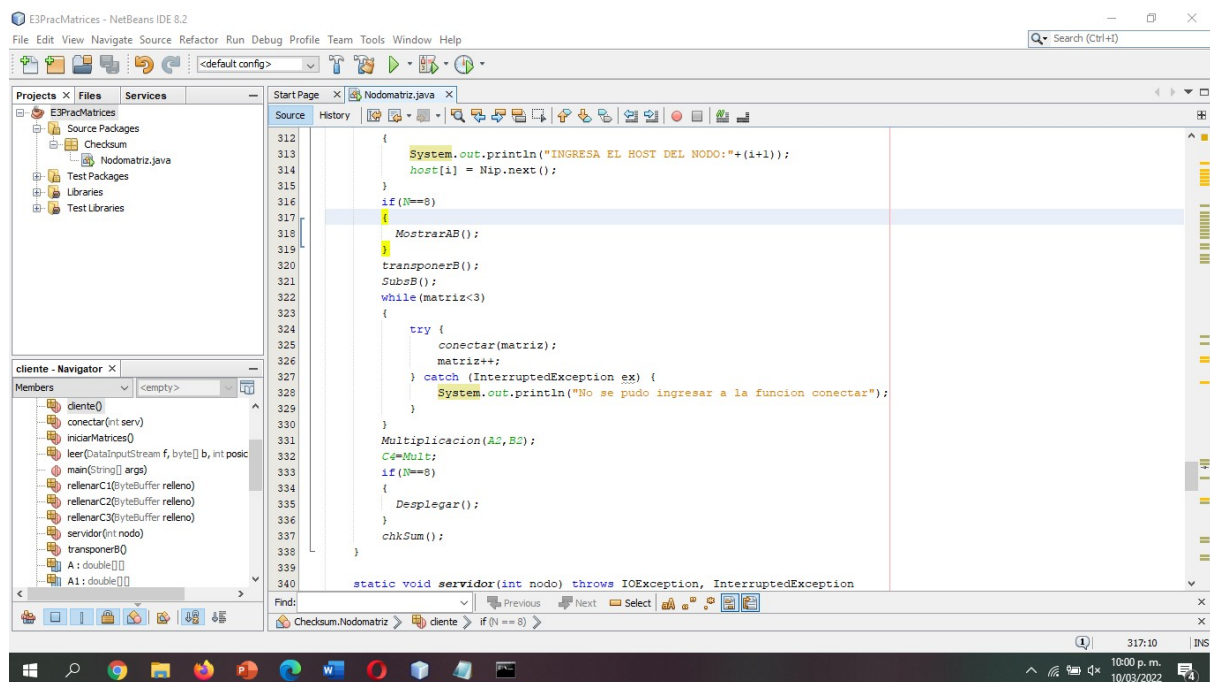
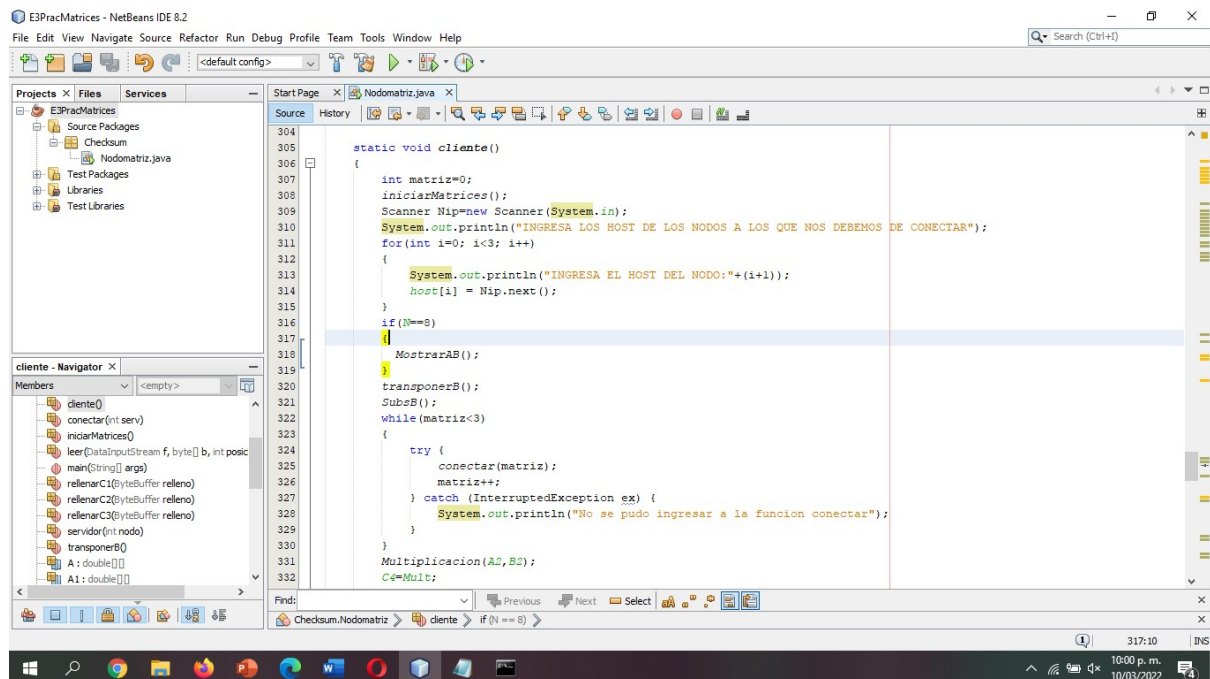


Imagen 6: Captura del método rellenarC1 y rellenarC2.

Metodo cliente()

Este método se manda a llamar cuando se trata de un nodo cliente (0). Tan solo iniciar el método mandamos a llamar al método `iniciarMatrices()` para inicializar las matrices, después de esto procedemos a pedir los datos del nodo 1, 2 y 3, ya que recordemos que estos nodos se encuentran en máquinas independientes y por lo tanto no se encuentran en el localhost. Posterior a esto verificamos las dimensiones de las matrices A y B son de 8 (N=8) ya que en caso de ser afirmativo, procedemos a mostrar A y B antes de que saqueños la transpuesta de B ya que esta sustituye sus valores en la matriz B, y después de sacar la transpuesta sacamos las submatrices B1 y B2 mandando a llamar al método `sumaB`. Una vez hecho esto nos anclamos en un bucle while donde mandaremos a llamar al método `conectar` para establecer conexión con cada servidor, poderle enviar las submatrices correspondientes a la multiplicación que debe hacer cada servidor, también poder recibir el resultado de dicha multiplicación, la cual se trata de una matriz de dimensiones (N/2) x (N/2) y guardar los datos de esta matriz en la submatriz de C correspondiente. Una vez que hayamos logrado conectar con cada servidor (matriz `prec3`) procedemos a salir del ciclo para realizar la multiplicación que le corresponde al cliente (A2 x B2), a lo que una vez obtenido la matriz resultante la cual se trata de una submatriz C4 le asignamos los valores de esta otra matriz, ya que el cliente ya cuenta con estos datos y solo necesitamos pasarlos a C4 para poder realizar el checksum. Antes de realizar el checksum verificamos si el valor de las dimensiones de A y B es 8, ya que en caso de ser cierto, no es posible desplegar la tabla resultante de todas las multiplicaciones, las cuales ya tendremos hasta este punto, lo que procedemos a mandar a llamar al método `desplegar` que mostrará la matriz C.

después de esto realizamos checksum mandando a llamar al método `chkSum` en caso contrario solo se hace el checksum.



Imágenes 7 y 8: Capturas del método cliente.

Método servidor()

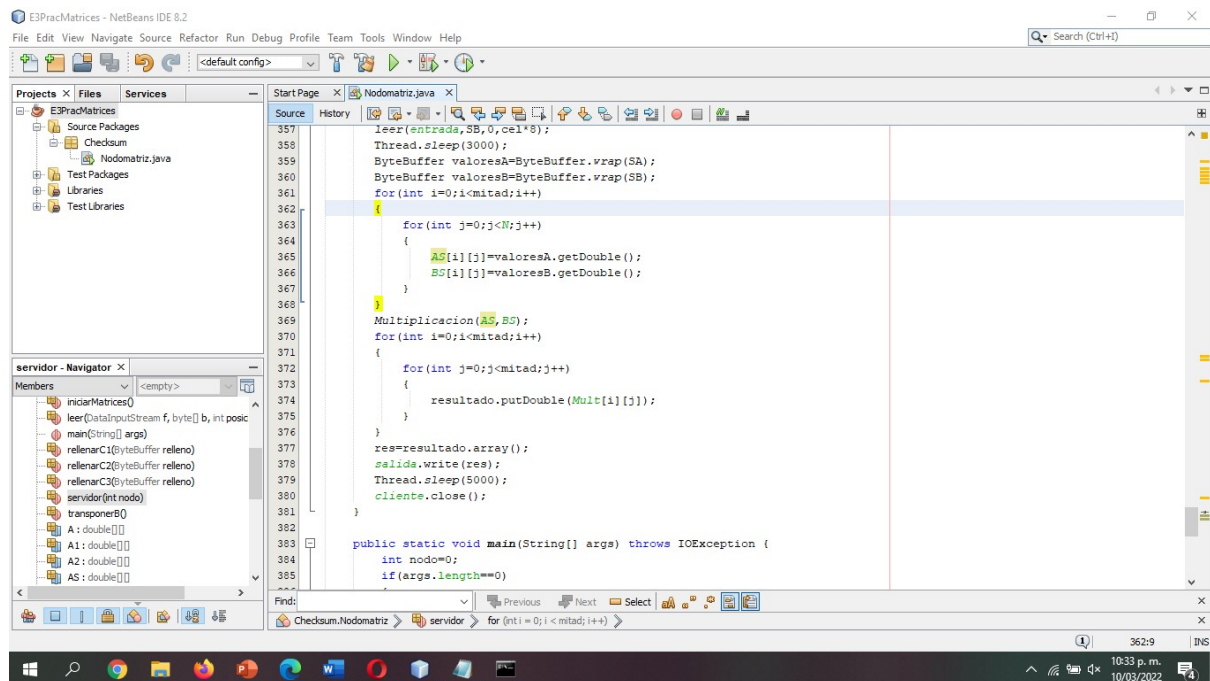
Para el método servidor requerimos un `ByteBuffer` para ingresar los valores de cada celda de la matriz resultante (Mult) de multiplicar las submatrices de A y B correspondientes al número de nodos así como arreglos de bytes para recibir las matrices correspondientes al

nodo para realizar la multiplicación (SA y SB) y el arreglo de bytes para mandarle resultado de dicha operación (res). Antes que nada procedemos a iniciar el servidor con el número de puerto que se trata de puerto 50000 más el número de nodo a fin de que podamos identificar al servidor por su puerto y después de esto lo ponemos a esperar al cliente para establecer conexión. Una vez establecida creamos los canales de entrada y salida para poder recibir las submatrices así como poder enviar la matriz resultante, posteriormente procedemos a llamar al método leer el cual se trata de la clase especial read que vimos en las primeras clases para manejar arreglos de bytes, finalidad de poder recibir las submatrices que el cliente nos envió sin temor de no leer todos los valores, en este caso mandaremos a llamar a leer 2 veces (primero para la submatriz de A y luego para la submatriz de B) pero dejando entre estos un descanso de 3 segundos y otro posterior a la segunda lectura. Después con ayuda del método wrap de clase ByteBuffer obtenemos los valores de los arreglos de bytes en formato double, los cuales guardaremos respectivamente en ByteBuffer valoresA y valoresB. Posterior a esto usamos un ciclo para sacar cada valor de los ByteBuffers y asignarlos a las matrices globales A y B, los cuales enviaremos como parámetros al método Multiplicar para obtener la matriz resultante (Mult), cuyos valores de cada celda se acumularán en el ByteBuffer resultado para posteriormente convertirlo en un arreglo de bytes (res) y poder enviarle estos datos al cliente el cual se encargará de asignarlos a la submatriz de C correspondiente. Antes de finalizar la conexión esperamos un lapso de 5 segundos para asegurar que se han mandado por completos los datos al cliente.

```

339
340 static void servidor(int nodo) throws IOException, InterruptedException
341 {
342     int mitad=N/2;
343     int cel=mitad*N;
344     int prod=mitad*mitad;
345     ByteBuffer resultado= ByteBuffer.allocate(prod*8);
346     byte [] SA= new byte[cel*8];
347     byte [] SB= new byte[cel*8];
348     byte [] res= new byte[prod*8];
349     servidor=new ServerSocket(50000+nodo);
350     System.out.println("Servidor "+nodo+" en espera de un cliente");
351     cliente= servidor.accept();
352     System.out.println("Cliente conectado con éxito");
353     entrada= new DataInputStream(cliente.getInputStream());
354     salida= new DataOutputStream(cliente.getOutputStream());
355     leer(entrada,SA,0,cel*8);
356     Thread.sleep(3000);
357     leer(entrada,SB,0,cel*8);
358     Thread.sleep(3000);
359     ByteBuffer valoresA=ByteBuffer.wrap(SA);
360     ByteBuffer valoresB=ByteBuffer.wrap(SB);
361     for(int i=0;i<mitad;i++)
362     {
363         for(int j=0;j<N;j++)
364         {
365             AS[i][j]=valoresA.getDouble();
366             BS[i][j]=valoresB.getDouble();
367         }
368     }
369 }

```



Imágenes 9 y 10: Capturas del método servidor.

Metodo conectar()

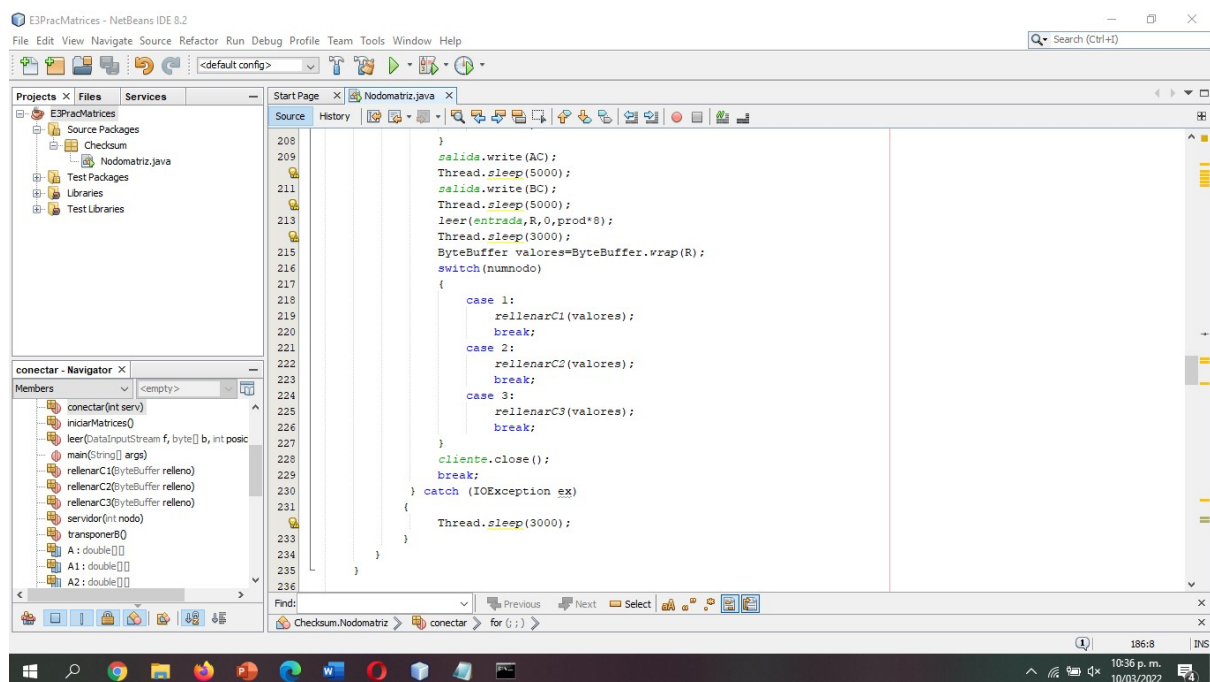
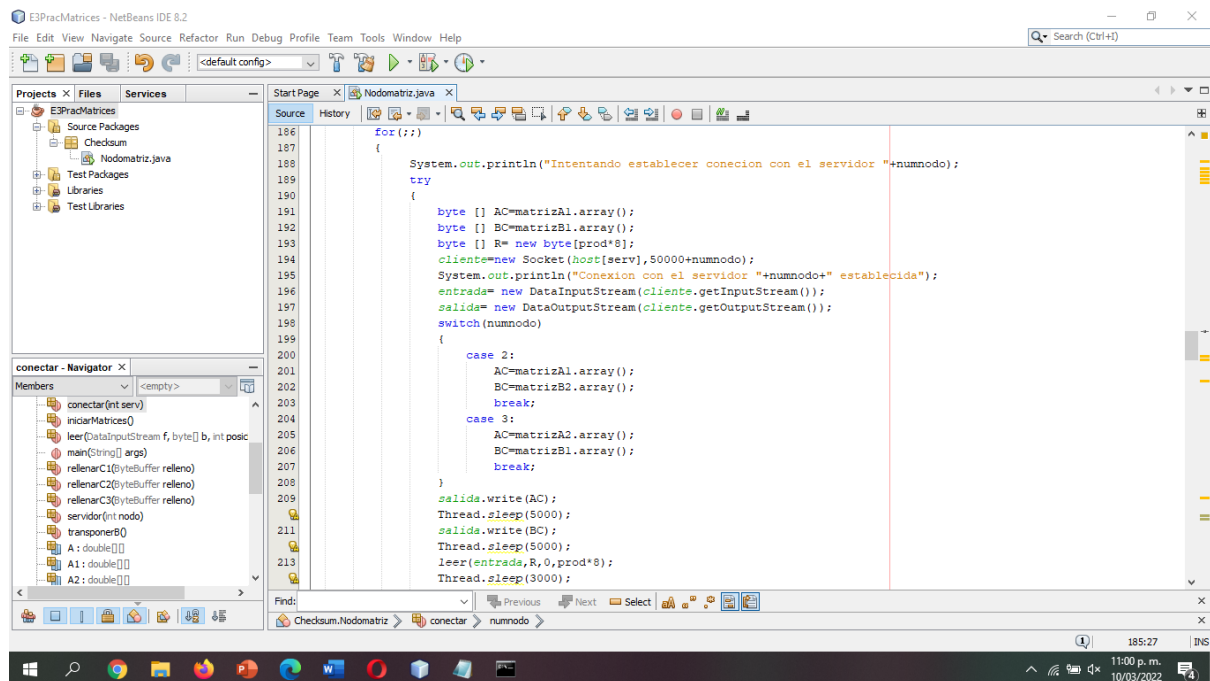
El metodo conectar usa 4 ByteBuffers para poder almacenar todos los valores de las submatrices de A y B (matrizA1, matrizA2, matrizB1 y matrizB2) mediante un ciclo el cual va almacenando los valores de cada submatriz por renglones, y luego se hace esto creamos una variable entera la cual es el número de nodo vuelta del ciclo del metodo cliente (parámetro que mandamos llamar al método conectar) incrementado en 1, ya que sino habria problemas con los puertos ya que el primer valor que adopta el parámetro serv es un 0. Declaramos un ciclo for infinito para poder complementar reintentos de conexión, cuyo si en todo caso es exitoso procedemos a declarar dos arreglos de bytes (AC y BC) cuyo valor inicial es el ByteBuffer matrizA1 y matrizB2 correspondientemente. Posteriormente procedemos a iniciar la conexión con el servidor que como dijimos, en este caso fue exitoso, pero hay que mencionar que a parte del número de puerto, también es necesario el host de dicho servidor ya que recordemos que se tratan de máquinas independientes, cuyo host lo obtenemos del arreglo host que rellenamos en el método cliente. Al igual que el servidor creamos los canales de entrada y salida para enviar y recibir datos, después de esto tenemos un switch para verificar con que servidor nos conectamos y a partir de ahí serán los ByteBuffers que convertiremos en arreglos de bytes y asignaremos a (AC y BC), mencionar que si no se encuentra el caso del nodo 1, se debe a que son los valores iniciales de nuestros arreglos de bytes por lo que si se tratara de dicho nodo no habría problema de no analizar su caso. Después de esta verificación le mandamos estos arreglos de bytes al servidor con 2 llamadas al método write, las cuales tienen de separación un lapso de descanso de 5 segundos para asegurar que se mande completamente el arreglo de bytes y

también hay un lapso de descanso de la misma duración después del segundo write para asegurar que se envíe completo el arreglo, que ahora tocaría esperar a que el servidor haga la multiplicación. Para recibir los bytes de la multiplicación tenemos un arreglo de bytes R el cual con ayuda del método wrap del ByteBuffer vamos a poder obtener los valores de tipo doble resultantes de multiplicar las 2 submatrices. Vamos a hacer una verificación pero en este caso contamos con el caso del nodo ya que a partir del nodo con el que estemos conectados mandaremos a llamar a sea al método (rellenarC1, rellenarC2 o rellenarC3) con el fin de asignarle los valores a una de las submatrices de C. Para finalizar cerramos la conexión con el servidor y realizamos un break para salir del for infinito. Cabe mencionar que en caso de no poder establecer conexión con el servidor procederemos a esperar 3 segundos antes de volver a intentarlo.

```

165 static void conectar(int serv) throws InterruptedException
166 {
167     int mit=N/2;
168     int cel=mit*N;
169     int prod=mit*mit;
170     ByteBuffer matrizA1= ByteBuffer.allocate(cel*8);
171     ByteBuffer matrizB1= ByteBuffer.allocate(cel*8);
172     ByteBuffer matrizA2= ByteBuffer.allocate(cel*8);
173     ByteBuffer matrizB2= ByteBuffer.allocate(cel*8);
174
175     for(int i=0;i<mit;i++)
176     {
177         for(int j=0;j<N;j++)
178         {
179             matrizA1.putDouble(A1[i][j]);
180             matrizA2.putDouble(A2[i][j]);
181             matrizB1.putDouble(B1[i][j]);
182             matrizB2.putDouble(B2[i][j]);
183         }
184     }
185     int numnodo=serv+1;
186     for(;;)
187     {
188         System.out.println("Intentando establecer conexion con el servidor "+numnodo);
189         try
190         {
191             byte [] AC=matrizA1.array();
192             byte [] BC=matrizB1.array();
193             byte [] R= new byte[prod*8];

```



Imágenes 11, 12 y 13: Capturas del método conectar.

Método chkSum()

Este método realiza la sumatoria de los valores de la matriz que si lo analizamos bien están contenidos en C2, C3 y C4 por lo que bastará con realizar un ciclo anidado para que en un acumulador poder recibir la sumatoria de cada uno de los valores de estas submatrices y una vez terminado este ciclo procedemos a mostrar el resultado del checksum.

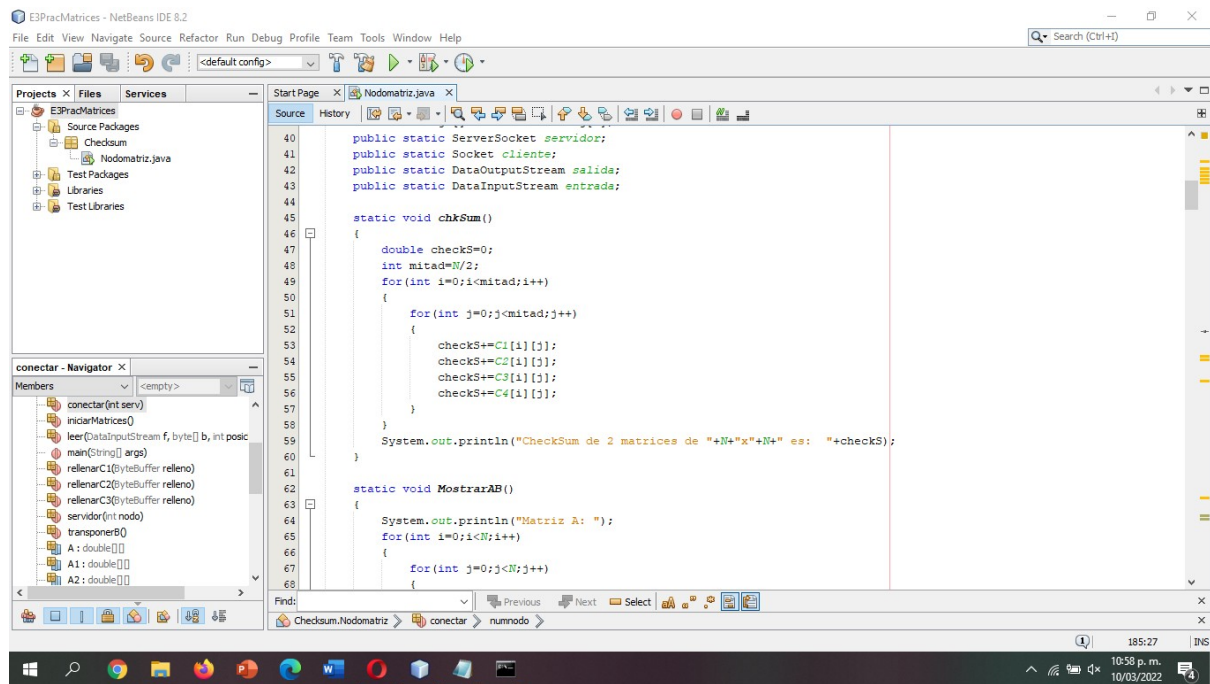


Imagen 14: Captura del método chkSum.

Compilacion y ejecucion del programa

- Nota: Se debe actualizar los repositorios y sistema operativo de los servidores con los comandos “sudo apt update” y “sudo apt upgrade”.
- Nota 2: Se debe instalar jre y jdk para compilar y ejecutar el programa hecho en java, en cada servidor con las instrucciones “sudo apt install default-jre” y “sudo apt install default-jdk”.
- Nota 3: Se debe de enviar el programa creado a las máquinas virtuales con la instrucción “scp ruta archivo origen nombreUser@dominio:rutaDestino”, en este caso este comando conviene ya que solo es un archivo.
- En la imagen se aprecia que cada terminal linux se conecta a una máquina virtual diferente, los nodos 1, 2, 3 se manejan como servidores y el nodo 0 como cliente.
- Se coloca el número de nodo en cada terminal y los host de todos los servidores.
- Los servidores entregan la parte de C correspondiente a su nodo y envían al cliente a través de socket. Como se aprecia en la imagen.

Actividades Terminal 10 de mar 21:25

equipo0@Tarea-3-0-0: ~

equipo0@Tarea-3-0-0:~\$ java Nodomatriz 0
Numero de nodo: 0
INGRESA LOS HOST DE LOS NODOS A LOS QUE NOS DEBEMOS DE CONECTAR
INGRESA EL HOST DEL NODO:1
20.25.85.19
INGRESA EL HOST DEL NODO:2
20.25.32.211
INGRESA EL HOST DEL NODO:3
20.228.154.10

equipo1@Tarea-3-1-1: ~

equipo1@Tarea-3-1-1:~\$ java Nodomatriz 1
Numero de nodo: 1
Servidor 1 en espera de un cliente

equipo2@Tarea-3-2-2: ~

equipo2@Tarea-3-2-2:~\$ java Nodomatriz 2
Numero de nodo: 2
Servidor 2 en espera de un cliente

equipo3@Tarea-3-3-3: ~

equipo3@Tarea-3-3-3:~\$ java Nodomatriz 3
Numero de nodo: 3
Servidor 3 en espera de un cliente

Actividades Terminal 10 de mar 21:26

equipo0@Tarea-3-0-0: ~

Conexion con el servidor 3 establecida
Matriz C:
3500.0 3360.0 3220.0 3080.0 2940.0 2800.0 2660.0 2520.0
3640.0 3492.0 3344.0 3196.0 3048.0 2900.0 2752.0 2604.0
3780.0 3624.0 3468.0 3312.0 3156.0 3000.0 2844.0 2688.0
3920.0 3756.0 3592.0 3428.0 3264.0 3100.0 2936.0 2772.0
4060.0 3888.0 3716.0 3544.0 3372.0 3200.0 3028.0 2856.0
4200.0 4020.0 3840.0 3660.0 3480.0 3300.0 3120.0 2940.0
4340.0 4152.0 3964.0 3776.0 3588.0 3400.0 3212.0 3024.0
4480.0 4284.0 4088.0 3892.0 3696.0 3500.0 3304.0 3108.0

Checksum de 2 matrices de 8x8 es: 217728.0
equipo0@Tarea-3-0-0:~\$

equipo1@Tarea-3-1-1: ~

equipo1@Tarea-3-1-1:~\$ java Nodomatriz 1
Numero de nodo: 1
Servidor 1 en espera de un cliente
Cliente conectado con exito
equipo1@Tarea-3-1-1:~\$

equipo2@Tarea-3-2-2: ~

equipo2@Tarea-3-2-2:~\$ java Nodomatriz 2
Numero de nodo: 2
Servidor 2 en espera de un cliente
Cliente conectado con exito
equipo2@Tarea-3-2-2:~\$

equipo3@Tarea-3-3-3: ~

equipo3@Tarea-3-3-3:~\$ java Nodomatriz 3
Numero de nodo: 3
Servidor 3 en espera de un cliente
Cliente conectado con exito
equipo3@Tarea-3-3-3:~\$