



UNL

Universidad
Nacional
de Loja

1859

FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y
LOS RECURSOS NATURALES NO RENOVABLES

Carrera de Computación

PRÁCTICA DE LABORATORIO / TALLER Nro. 2

Carrera de Computación

A. DATOS INFORMATIVOS		
Asignatura: Algoritmos, Análisis y Programación Paralela	Ciclo / Semestre: Séptimo	Paralelo: A
Docente: Roberto Gustavo Figueroa Díaz	Período Académico: Septiembre 2025 – Febrero 2026	
Alumnos: <ul style="list-style-type: none">● Gerardo Herrera● Alexis Ludeña● Luis Medina● Byron Jimenez● Nayely Ramirez		

B. INFORMACIÓN GENERAL		
Unidad: 2. Algoritmos naturalmente paralelos y programación concurrente		
Tema: Algoritmos naturalmente paralelos usando OpenMP y MPI		
Fecha: Loja, 12 de Diciembre de 2025		Nro. horas: 3h
Objetivos: <ul style="list-style-type: none">● Algoritmos naturalmente paralelos y programación concurrente		
Corresponde al resultado de aprendizaje: Identifica tareas independientes en un programa que podría ser paralelizado, bajo los principios de solidaridad, transparencia, responsabilidad y honestidad.		
Recursos y/o materiales: <ul style="list-style-type: none">● Computador. – Computadora con acceso a internet.● Navegador. – Revisión de información por parte del estudiante en la web.● Bibliografía. – Textos físicos, virtuales y materiales facilitados en el Sílabo.● Biblioteca virtual. – Acceso a recursos digitales como libros y artículos.● EVA . – Entorno virtual de aprendizaje de la UNL.		

C. DESARROLLO		
Instrucciones: <ol style="list-style-type: none">1. Revisar el contenido facilitado por el profesor: diapositivas, libro base, sílabo de la asignatura, recursos digitales, use el libro base e información complementaria; así como, internet para reforzar los contenidos estudiados.2. Revise detalladamente el material indicado en clase.3. Realizar la lectura del texto base y complementarios sobre las temáticas abordadas.4. Para el desarrollo de los ejercicios puede usarse las herramientas de su interés tales como lenguaje Python, Java o C++, a su elección.		
Preguntas a responder:		



1. Elabore un informe de la actividad realizada en grupo. Se debe utilizar los archivos adjuntos y verificar el funcionamiento en los ambientes:

a. Secuencial (local)

Para la ejecución del programa de multiplicación de matrices utilizando la configuración local y secuencial, se documentan los resultados obtenidos (Tabla 1) que servirán como la línea base de rendimiento ya que establece la medición base de tiempo para las comparaciones con las implementaciones de memoria compartida (OpenMP) y memoria distribuida (MPI).

Nro. Procesos	Nro. Núcleo	Tiempo	Checksum
16	8	7.50043 segundos	-336.504

Tabla 1. Resultados ejecución secuencial

```
alexis@alexis-ubuntu:~/Descargas/MulMatrizCPP/MulMatrizCPP$ g++ -o matmul_seq matmul_seq.cpp
alexis@alexis-ubuntu:~/Descargas/MulMatrizCPP/MulMatrizCPP$ ./matmul_seq 1000
Secuencial: generando matrices 1000x1000...
Tiempo secuencial: 7.50043 s
Checksum (suma diag): -336.504
alexis@alexis-ubuntu:~/Descargas/MulMatrizCPP/MulMatrizCPP$
```

Figura 1. Resultados de la ejecución secuencial.

b. Memoria compartida (local)

Para la ejecución del problema de multiplicación de matrices utilizando memoria compartida, se hizo uso de la librería OpenMP (Open multi-process), el uso de dicha librería facilitó el paralelismo de la aplicación sin manejar hilos manualmente ideal para CPU multinúcleo, los hilos comparten variables y memoria.

En el ejercicio propuesto, OpenMP permite dividir el trabajo de la multiplicación de matrices entre varios hilos, ejecutando la multiplicación en paralelo.

El objetivo principal de la ejecución con memoria compartida es acelerar el cálculo aprovechando múltiples núcleos del procesador, haciendo que varios hilos trabajen simultáneamente sobre un mismo espacio de memoria.

Nro. Procesos (Hilos)	Nro. Núcleo	Tiempo	Checksum
16	8	2.30024 segundos	-336.504

Tabla 2. Resultados ejecución Memoria Compartida

Tal como se puede observar en la tabla 2, los resultados obtenidos evidencia que la ejecución del programa utilizando memoria compartida presenta un incremento significativo en la velocidad de procesamiento en comparación con los resultados de la ejecución del programa secuencial, manteniendo el mismo tamaño de las matrices se registra una reducción del aproximada del 60% en el tipo de ejecución, lo cual se debe al aprovechamiento del paralelismo mediante múltiples hilos, permitiendo que las operaciones de multiplicación se ejecuten de forma simultánea y sea más eficiente sobre los recursos del sistema.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\LENOVO\Desktop\Actividad> g++ matmul_shared.cpp -o matmul_shared.exe -std=c++17 -fopenmp
PS C:\Users\LENOVO\Desktop\Actividad> ./matmul_shared.exe 1000
Paralelo memoria compartida: generando matrices 1000x1000...
Tiempo paralelo (shared memory): 2.30024 s
Checksum (suma diag): -336.504
PS C:\Users\LENOVO\Desktop\Actividad>
```

Figura 2. Resultado de la ejecución

**c. Memoria distribuida (clúster)**

La implementación del clúster se llevó a cabo sobre el sistema operativo Linux, específicamente utilizando la distribución Ubuntu, seleccionada por su estabilidad y compatibilidad nativa con entornos de computación distribuida.

Para garantizar el desarrollo, la administración y el correcto funcionamiento del clúster, se integraron las siguientes tecnologías y herramientas:

- SSH (Secure Shell): Empleado para establecer acceso remoto seguro y gestionar la administración de los nodos del clúster.
- MPI (Message Passing Interface): Estándar utilizado para la programación paralela, facilitando la comunicación y coordinación eficiente entre los procesos distribuidos.
- GCC (GNU Compiler Collection): Compilador usado para la construcción de los programas en C/C++ dentro del entorno MPI.
- Herramientas de red y monitoreo: Se utilizó ping para verificar la conectividad entre nodos , scp para la transferencia segura de archivos y htop/top para el monitoreo en tiempo real del uso de CPU, memoria y gestión de procesos.

A continuación, se presentan los tiempos de ejecución obtenidos al distribuir la carga de trabajo entre los nodos del clúster (Maestro y Esclavo).

Pc	Tipo Nodo	Proceso	Tiempo (segundos)
Luis	Maestro	1	0:14.39
Luis	Maestro	2	0:14.41
Luis	Maestro	3	0:14.42
Luis	Maestro	4	0:14.42
Luis	Maestro	5	0:14.41
Luis	Maestro	6	0:14.41
Luis	Maestro	7	0:14.41
Luis	Maestro	8	0:14.42
Bayron	Esclavo	9	0:40.71
Bayron	Esclavo	10	0:39.85
Bayron	Esclavo	11	0:38.51
Bayron	Esclavo	12	0:38.21

Tabla 3. Resultados ejecución memoria distribuida



UNL

Universidad
Nacional
de Loja

1859

FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y
LOS RECURSOS NATURALES NO RENOVABLES

Carrera de Computación

Vista nodo Maestro:

```
luis@ubuntu:~$ mpirun -np 12 ./matmul_mpi 1000
Authorization required, but no authorization protocol specified
MPI root: checksum diag = -336.504
luis@ubuntu:~$
```

Figura 3. Resultados ejecución de la memoria distribuida

```
luis@ubuntu:~$ mpirun -np 12 ./matmul_mpi 1000
Authorization required, but no authorization protocol specified
MPI root: checksum diag = -336.504
luis@ubuntu:~$
```

Figura 4. Procesos del nodo maestro de matmul

Como se evidencia en las Figuras 3 y 4, el nodo maestro gestionó la orquestación del proceso mediante el comando mpirun. Se observa que el *checksum* de la diagonal (-336.504) es consistente con las ejecuciones secuenciales y de memoria compartida, validando la corrección del cálculo distribuido.



UNL

Universidad
Nacional
de Loja

1859

FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y
LOS RECURSOS NATURALES NO RENOVABLES

Carrera de Computación

Vista del nodo esclavo:

```

12 de dic 10:51
bayron@bayron-ubuntu: ~

Descargas Escritorio Imágenes Música Público sumaVector
Documentos hola matmul_mpi Plantillas snap Videos
bayron@bayron-ubuntu: ~ sudo apt-get install libopenmpi-dev
[sudo] contraseña para bayron:
Lo siento, prueba otra vez.
top - 10:50:52 up 3:16, 2 users, load average: 2,93, 2,23, 1,45
Tareas: 265 total, 6 ejecutar, 0 detener, 0 zombie
%Cpu(s): 36,4 us, 62,7 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,8 st, 0,0 st
Mem Mem : 7638,7 total, 1556,6 libre, 3624,2 usado, 3471,0 búf/caché
Mib Intercambio: 4096,0 total, 4095,7 libre, 0,2 usado. 4014,5 dispon Mem

PID USUARIO PR NI VIRT RES SHR S %CPU %MEM HORA+ ORDEN
32258 bayron 20 0 227624 24732 12220 R 100,0 0,3 0:40.71 matmul_mpi
32249 bayron 20 0 226972 24072 12192 R 92,5 0,3 0:39.85 matmul_mpi
32248 bayron 20 0 226972 23980 12120 R 85,0 0,3 0:38.51 matmul_mpi
32247 bayron 20 0 227660 24648 12124 R 82,5 0,3 0:38.21 matmul_mpi
2067 bayron 20 0 4782168 342464 151772 S 12,1 4,4 7:06.38 gnome-shell
32304 bayron 20 0 784160 89240 46100 S 9,6 1,1 0:03.27 eog
517 root -51 0 0 0 0 S 6,7 0,0 2:06.50 irq/158-rtw88_pci
25868 bayron 20 0 629572 60712 47572 R 6,7 0,8 0:06.66 gnome-terminal-
284 root 19 -1 165772 97284 95644 S 1,7 1,2 0:46.45 systemd-journal
24667 bayron 20 0 1963496 221184 117796 S 0,8 2,8 1:56.50 gnome-control-c
39 root 20 0 0 0 0 S 0,4 0,0 0:01.44 ksoftirqd/3
138 root 0 -20 0 0 0 I 0,4 0,0 0:02.07 kworker/0:1H-l915_cleanup
183 root -51 0 0 0 0 S 0,4 0,0 0:10.16 irq/57-ELAN1200:00
831 message+ 20 0 12284 7732 4788 S 0,4 0,1 0:12.40 dbus-daemon
2201 bayron 20 0 386632 12864 7528 S 0,4 0,2 0:07.03 ibus-daemon
2391 bayron 20 0 419732 30864 19352 S 0,4 0,4 0:03.66 ibus-extension-
2565 bayron 20 19 1010702 34484 19474 S 0,4 0,4 0:08.90 tracer/meson_f

```

Figura 5. Procesos del nodo esclavo de matmul

En la figura 5 se puede observar el funcionamiento del clúster desde la parte del nodo esclavo. En el primer comando se enlistan los documentos en el directorio y se observa a “*sumaVector*” y “*matmul_mpi*” que fueron enviados por el nodo maestro a través de *ssh*.

También se realizó la instalación de *libopenmpi-dev* para habilitar la ejecución de archivos que utilicen MPI. Finalmente con el comando *top* se visualiza los procesos en ejecución y entre los primeros destaca “*matmul_mpi*” que fue ejecutado por el maestro en este nodo.

Conclusiones:

- La práctica permitió validar la utilidad de la programación paralela tanto en memoria compartida (OpenMP) como distribuida (MPI), utilizando los códigos desarrollados para medir y comparar el rendimiento en cada escenario.
- Mediante la implementación de los códigos *matmul_seq.cpp*, *matmul_openmp.cpp* y *matmul_mpi.cpp*, se comprobó que la parallelización con OpenMP reduce considerablemente el tiempo de ejecución frente al enfoque secuencial (7.50043 segundos - Secuencial a 2.30024 segundos - OpenMP), aprovechando múltiples núcleos de procesador.
- El uso del código MPI permitió distribuir la carga de trabajo de la multiplicación de matrices entre los nodos Maestro y Esclavo, demostrando la eficacia de la librería para coordinar y optimizar el procesamiento en sistemas de múltiples nodos, confirmando la utilidad del MPI en arquitecturas de computación distribuida.

D. RÚBRICA DE EVALUACIÓN

Informe de trabajo:	2 puntos
<ul style="list-style-type: none"> Contenido: pertinente y concreto. Estructura y organización: Elementos vinculados y estructurados coherentemente. Originalidad y creatividad: trabajo inédito, presentación de nuevas ideas. 	
Resolución de Preguntas:	6 puntos
<ul style="list-style-type: none"> Proceso de resolución de preguntas: con originalidad y creatividad 	
Conclusiones:	2 puntos
<ul style="list-style-type: none"> Redacción Originalidad y creatividad: conclusiones inéditas en base a su experiencia y objetivos 	
Total	10 puntos



UNL

Universidad
Nacional
de Loja

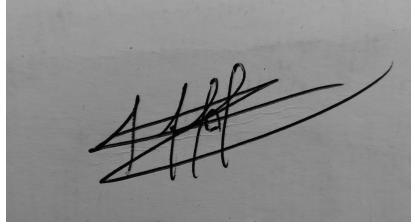
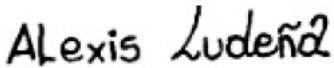
1859

FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y
LOS RECURSOS NATURALES NO RENOVABLES

Carrera de Computación

(Ponderado en Aprendizaje Práctico Experimental)

E. FIRMAS DE RESPONSABILIDAD:

Estudiante(s):	Firma
Gerardo Israel Herrera Campoverde	
Byron Alejandro Jimenez Cango	
Alexis Gradyn Ludeña Cueva	
Luis Alberto Medina Chamba	
Nayely Cruzcaya Ramirez Herrera	