

**Pontificia Universidad Javeriana**

Taller de Evaluación de Rendimiento



Juan Guillermo Pabon Vargas  
Tomas Alejandro Silva Correal  
Sergio Pardo Hurtado  
Juan Felipe Rodriguez Amador  
Arley Bernal

Sistemas Operativos

3 de Mayo de 2025

# Índice:

|  |          |
|--|----------|
| <b>Objetivos.....</b>                      | <b>3</b> |
| <b>Introducción.....</b>                   | <b>3</b> |
| <b>Desarrollo.....</b>                     | <b>3</b> |
| <b>Resultados.....</b>                     | <b>3</b> |
| Datos (explicados):.....                   | 3        |
| Recursos de HW y SW del experimento:.....  | 4        |
| Métricas de rendimiento de programas:..... | 4        |
| Resultados y comparaciones:.....           | 5        |
| <b>Conclusiones.....</b>                   | <b>5</b> |
| <b>Link al repositorio.....</b>            | <b>5</b> |
| <b>Bibliografía.....</b>                   | <b>5</b> |

# Objetivos

El objetivo de este taller es comparar la ejecución de un algoritmo en serie y en paralelo. Esto se va realizar en diferentes sistemas de cómputo y se comparará los resultados para poder llegar a una conclusión y poder dar recomendaciones sobre las diferentes formas de ejecutar un algoritmo y cual de las opciones de concurrencia es mejor.

## Introducción

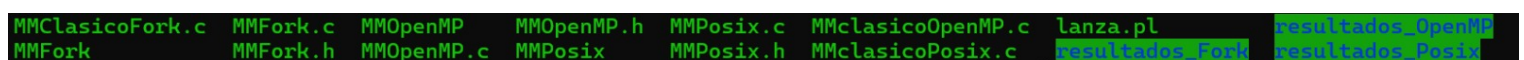
En este taller se van a modularizar los ficheros principales y se documentaran para ser ejecutados en diferentes sistemas de cómputo. Esto se hará para comparar los resultados de los tiempos de ejecución y conseguir un promedio entre ellos para poder acercarnos al valor real.

Se elaborará una tabla que represente la batería de experimentación a realizar, para poder hacer el análisis de rendimiento del algoritmo de Multiplicación de Matrices en serie y en paralelo. Con el fichero en PERL se automatizará el proceso de ejecución para que automáticamente capture los resultados de todas las ejecuciones para todos los valores de su batería de experimentación.

## Desarrollo

Primero se documentaron los ficheros y se modularon los ficheros en librerías e interfaces. Cada fichero principal se encuentra en su propia carpeta con las librerías y su ejecutable.

A continuación se mostrarán los diferentes programas, ejecutables creados, librerías y el archivo PERL generado para automatizar el proceso de las ejecuciones. Ese archivo PERL también crea 3 carpetas donde se guardaran los resultados de las ejecuciones.



|                 |          |            |            |           |                   |                 |                   |
|-----------------|----------|------------|------------|-----------|-------------------|-----------------|-------------------|
| MMClasicoFork.c | MMFork.c | MMOpenMP   | MMOpenMP.h | MMPosix.c | MMclasicoOpenMP.c | lanza.pl        | resultados_OpenMP |
| MMFork          | MMFork.h | MMOpenMP.c | MMPosix    | MMPosix.h | MMclasicoPosix.c  | resultados_Fork | resultados_Posix  |

Imagen 1, directorio de archivos para el experimento

## Resultados

A continuación están los promedios de las ejecuciones junto a los datos a utilizar y una explicación para su utilización, además de detalles de los equipos utilizados, el software donde se utilizó el programa, y detalles respecto a las métricas de rendimiento que existen, al igual que la seleccionada para este taller:

### Datos (explicados):

**Repeticiones:** Se realizaron 30 repeticiones para cada elemento, esto dado no sólo al enunciado, sino que además dado a la ley de los grandes números, de la teoría de la probabilidad, que explica como un incremento entre el número de repeticiones y un

promedio implica un acercamiento más preciso al valor real de lo que se espera medir, siendo 30 lo que inicia una distribución normal y es el mínimo para un acercamiento extenso al valor real.

**Hilos:** Se tomaron 4 opciones de hilos: 1, 2 y 4. Estos valores se eligieron para probar la diferencia entre el procesamiento en serie (1 hilo) y en paralelo (2 y 4), con los últimos 2 valores implican un aumento exponencial de la cantidad de hilos lo cual involucra un posible incremento que afecta los resultados obtenidos, por igual se tomó en cuenta la necesidad de una medición apropiada, y por tanto, se utilizó como máximo el mínimo de hilos entre todos los dispositivos de prueba.

**Tamaños de matriz:** Se aplicaron 3 tamaños de matriz principalmente: 50, 100 y 300. Estos valores se escogieron dado que muestran tres variantes de tamaño de muestra, pequeño (50), mediano (100) y grande (300), que implican cambios potencialmente radicales en los tiempos de ejecución que mostraran diferencias dependiendo del número de hilos y demás factores.

## Recursos de HW y SW del experimento:

A continuación, se presenta la información de hardware (procesador) y software (sistema operativo) de cada equipo, junto a su respectivo dueño:

### **Equipo 1** - Sergio Pardo Hurtado

- Hardware: Intel(R) Core(TM) i7-3537U 2.50GHz
- Software: Ubuntu terminal environment / Version Ubuntu 24.04.1 LTS

### **Equipo 2** - Juan Pabon Vargas

- Hardware: AMD Ryzen 9 5900hs
- Software: Sistema operativo Ubuntu 24.04.2 LTS

### **Equipo 3** - Tomas Alejandro Silva Correal

- Hardware: Procesador de doceava generación, Intel(R) Core(TM) i5-1235U 1.30GHz
- Software: Sistema operativo Ubuntu 24.04.2 LTS

## Métricas de rendimiento de programas:

A continuación se explicaran las diferentes métricas de rendimiento de programas y cual de estas fue la seleccionada para la comparativa de este taller.

- Tiempo de ejecución: Mide el tiempo que tarda el software en compilar y completar un programa con todos sus procesos.
- Rendimiento: Refleja la cantidad de trabajo que el software puede manejar en un periodo de tiempo determinado, como el número de transacciones por segundo.
- Uso de CPU y memoria: Evalúa cuánta carga de CPU y memoria consume el software durante su ejecución. Un uso eficiente de recursos es esencial para mantener un buen rendimiento.
- Tasa de errores: Mide la frecuencia de errores o fallos en la ejecución del software. Un bajo número de errores es un indicador de estabilidad y fiabilidad.

- Tiempo de disponibilidad: Representa el porcentaje de tiempo que el software está disponible y funcionando correctamente. Un alto tiempo de disponibilidad es vital para servicios críticos.
- Escalabilidad: Mide la capacidad del software para manejar un aumento en la carga de trabajo sin afectar negativamente el rendimiento. [1]

En este taller se optó por utilizar el *tiempo de ejecución* como métrica de rendimiento, esto dado a su cambio respecto a tamaño de matrices y cantidad de hilos respectivamente

## Resultados y comparaciones:

Los resultados obtenidos fueron organizados en tablas comparativas que muestran el tiempo de ejecución (en milisegundos) registrado por cada equipo para cada combinación de algoritmo, tamaño de matriz y número de hilos. A partir de estos datos, se calculó el promedio de tiempo por configuración, con el propósito de comparar objetivamente el desempeño general de cada enfoque.

|       |           | Tiempo de ejecución promedio (ms) |          |          |          |
|-------|-----------|-----------------------------------|----------|----------|----------|
| Hilos | Algoritmo | Equipo 1                          | Equipo 2 | Equipo 3 | Promedio |
| 1     | POSIX     | 1165.2                            | 1521.6   | 1480.6   | 1389.1   |
| 1     | Fork      | 836.2                             | 1371.5   | 1112.7   | 1106.8   |
| 1     | OpenMP    | 4850.4                            | 6214.1   | 305.1    | 3789.9   |
| 2     | POSIX     | 857.8                             | 1082.1   | 886.2    | 942.0    |
| 2     | Fork      | 773.5                             | 1046.7   | 890.1    | 903.4    |
| 2     | OpenMP    | 4459.0                            | 5839.8   | 411.2    | 3570.0   |
| 4     | POSIX     | 939.4                             | 983.0    | 718.7    | 880.4    |
| 4     | Fork      | 788.5                             | 1129.2   | 827.5    | 915.1    |
| 4     | OpenMP    | 4872.2                            | 5837.7   | 594.8    | 3768.2   |

**Tabla 1. Tamaño de matriz 50**

|       |           | Tiempo de ejecución promedio (ms) |          |          |          |
|-------|-----------|-----------------------------------|----------|----------|----------|
| Hilos | Algoritmo | Equipo 1                          | Equipo 2 | Equipo 3 | Promedio |

|   |        |        |         |        |        |
|---|--------|--------|---------|--------|--------|
| 1 | POSIX  | 4030.3 | 7811.4  | 5199.8 | 5680.5 |
| 1 | Fork   | 3762.4 | 7650.1  | 4237.9 | 5216.8 |
| 1 | OpenMP | 8438.5 | 11297.9 | 3332.9 | 7689.8 |
| 2 | POSIX  | 2760.2 | 4360.3  | 2580.8 | 3233.8 |
| 2 | Fork   | 3096.5 | 4198.9  | 2319.7 | 3205.0 |
| 2 | OpenMP | 8529.8 | 11174.7 | 2388.6 | 7364.4 |
| 4 | POSIX  | 2253.8 | 2677.5  | 2049.4 | 2326.9 |
| 4 | Fork   | 2415.0 | 2745.7  | 2952.5 | 2704.4 |
| 4 | OpenMP | 8485.3 | 11012.7 | 1303.8 | 6933.9 |

**Tabla 2, Tamaño de matriz 100**

|       |           | Tiempo de ejecución promedio (ms) |          |          |          |
|-------|-----------|-----------------------------------|----------|----------|----------|
| Hilos | Algoritmo | Equipo 1                          | Equipo 2 | Equipo 3 | Promedio |
| 1     | POSIX     | 144220.2                          | 198695.6 | 63960.2  | 135625.4 |
| 1     | Fork      | 96599.2                           | 200559.6 | 65017.0  | 120725.3 |
| 1     | OpenMP    | 72865.0                           | 38209.9  | 55594.5  | 55556.5  |
| 2     | POSIX     | 91820.7                           | 98987.0  | 32050.2  | 74286.0  |
| 2     | Fork      | 63756.7                           | 100828.0 | 31995.3  | 65526.7  |
| 2     | OpenMP    | 116921.6                          | 40151.1  | 27276.7  | 61449.8  |
| 4     | POSIX     | 79663.3                           | 51202.8  | 25871.8  | 52246.0  |
| 4     | Fork      | 52669.4                           | 52041.1  | 25905.8  | 43538.8  |
| 4     | OpenMP    | 143278.0                          | 41698.5  | 17768.9  | 67581.8  |

**Tabla 3. Tamaño de matriz 300**

#### **Análisis de resultados:**

En base a estas tablas se generaron los análisis a continuación:

La cantidad de hilos afecta de forma considerable el tiempo de ejecución, siendo los valores con menos hilos los más pequeños en tiempo de respuesta y aumentando mediante la cantidad de hilos disminuye. Es posible notar como de los tres tipos de concurrencia existentes, fork tiende a ser el algoritmo más eficiente a nivel de tiempo entre los tres, esto

con la excepción de cuatro hilos, donde posix llega a ser mejor en algunas ocasiones, con tamaños menores (como es visto con 50 y 100 en tamaños de matrices). De los tres equipos el equipo número tres resultó el más veloz en tiempos de ejecución, potencialmente debido a cómo el equipo maneja sus recursos. Esto permite definir al equipo 3, usando tamaños pequeños de matrices y 4 o más subprocesos fork como la solución o la opción más adecuada para el algoritmo de multiplicación de matrices.

## Conclusiones

En base a lo trabajado a lo largo del taller es posible concluir que cuando la cantidad de hilos incrementa, el trabajo es distribuido y hace que los procesos terminen más rápido mediante concurrencia.

También podemos concluir que el algoritmo donde se hace fork es el algoritmo que en la mayoría de casos tiende a contar con un menor de tiempo de ejecución que POSIX o OpenMP, viendo al igual al equipo 3 como el más eficiente en ejecución.

Este taller nos permite conocer a profundidad diversas opciones para concurrencia dentro de la programación en c, y el cómo diversos factores se toman en cuenta para mejorar el rendimiento de programas a la hora de ejecutarlos en algún dispositivo.

## Link al repositorio

<https://github.com/JohnWilliams2050/Sistemas-Operativos/tree/main/TallerRendimiento>

## Bibliografía

[1]Valtx, "VALTX - Tecnología que genera resultados," Valtx, Sep. 19, 2024. <https://www.valtx.pe/blog/metricas-para-el-desarrollo-de-software-conoce-cuales-son-las-mas-relevantes>