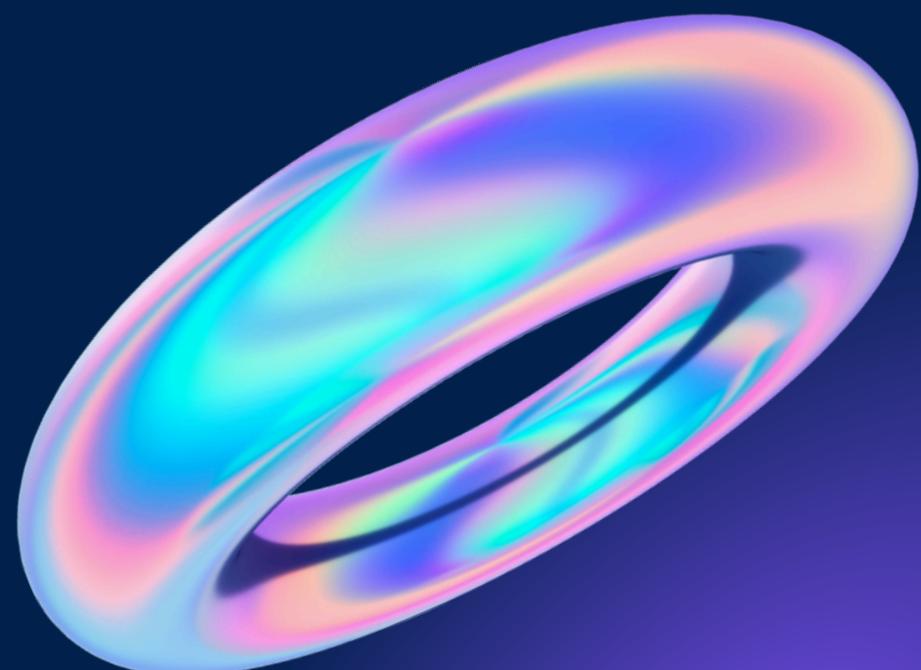




Taller de rendimiento



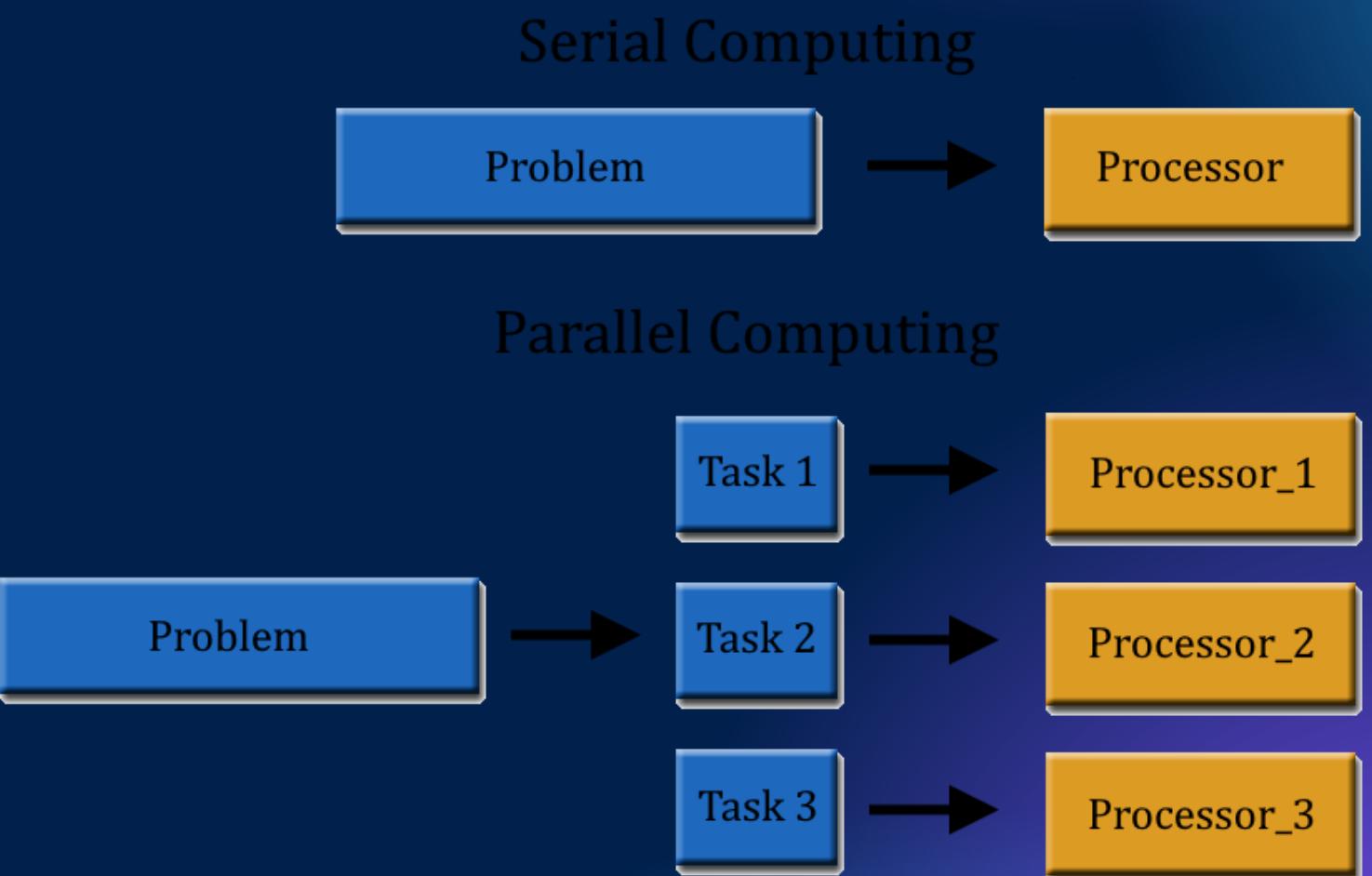
Sistemas Operativos

Juan David Rincón
Juan Felipe Fernández
Juan Felipe Morales

PARADIGMAS DE PROGRAMACIÓN SERIE Y PARALELO

Serie: Se basa en la ejecución secuencial de instrucciones, donde cada tarea se realiza en orden y una a la vez.

Paralelo: Busca distribuir tareas entre múltiples procesadores o núcleos de CPU, permitiendo que múltiples operaciones se ejecuten simultáneamente.



Funciones en nuestro fichero

-Función llenar_matriz: Inicializa las matrices mA y mB con valores aleatorios utilizando la función srand48 y un ciclo for.

-Función print_matrix: Imprime una matriz en la consola si su tamaño no es mayor a 12. La función organiza los elementos en filas y columnas con tres decimales de precisión.

-Función inicial_tiempo: Obtiene el tiempo actual utilizando la función gettimeofday y lo almacena en la variable start.

-Función final_tiempo: Obtiene el tiempo actual utilizando gettimeofday, calcula la diferencia de tiempo en microsegundos entre stop y start, y la imprime en la consola.

-Función mult_thread: Esta función es ejecutada por cada hilo y realiza la multiplicación de la sección de la matriz asignada al hilo.

Explicación Lanza.pl

```
#!/usr/bin/perl
#####
# Pontificia Universidad Javeriana
# Autor: J. Corredor
# Fecha: Febrero 2024
# Materia: Sistemas Operativos
# Tema: Taller de Evaluación de Rendimiento
# Fichero: script automatización ejecución por lotes
#####

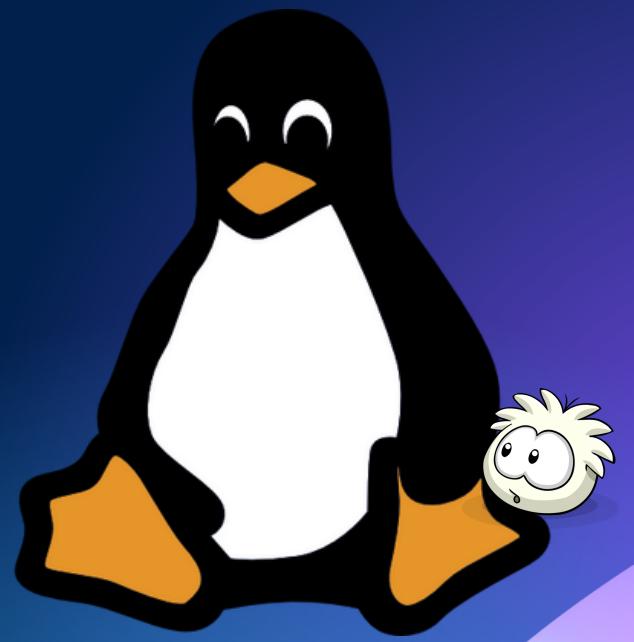
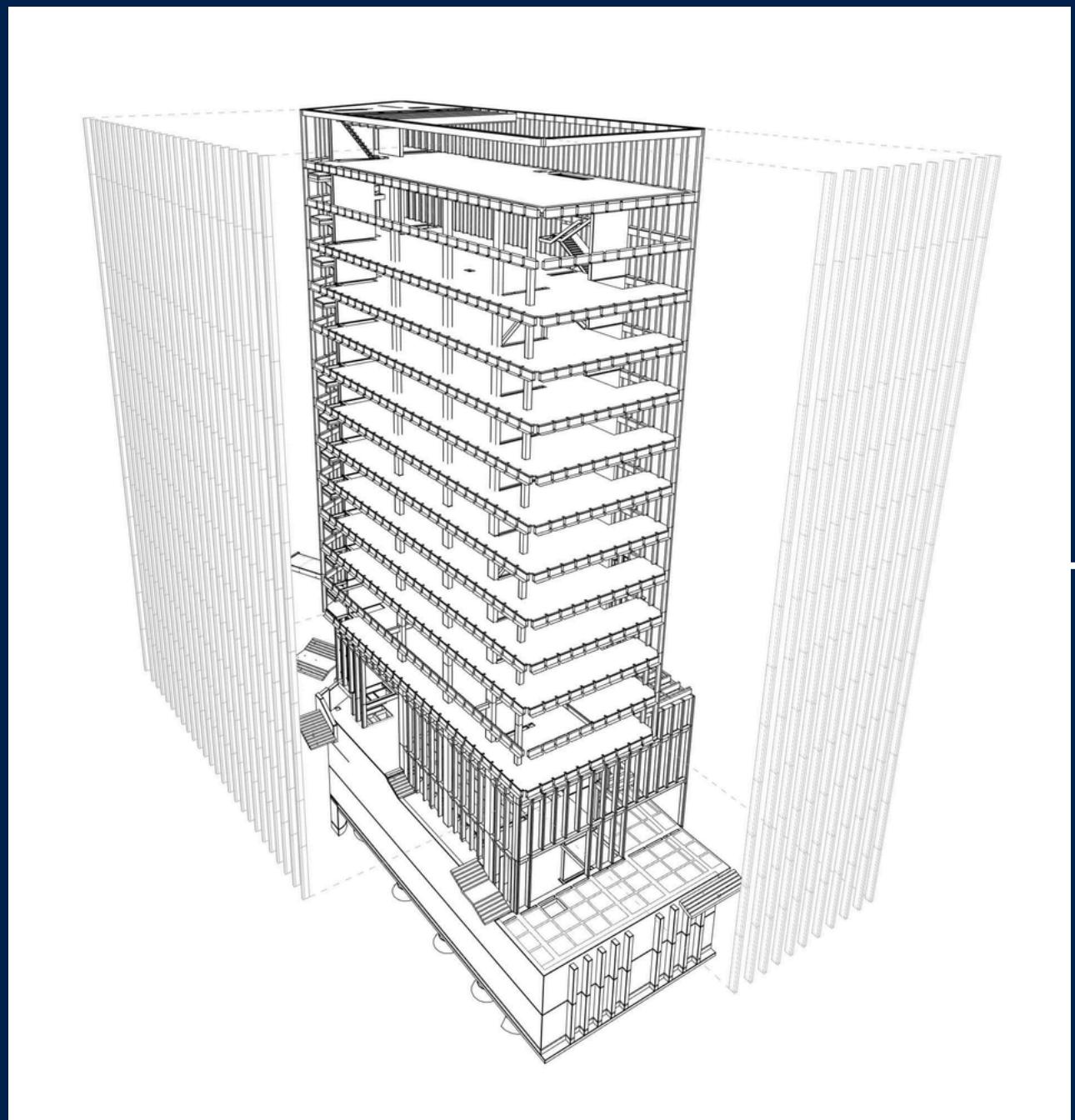
$Path = `pwd`;
chomp($Path);

$Nombre_Ejecutable = "MM_ejecutable";
@Size_Matriz = ("16", "300");
@Num_Hilos = (1,2,4,8);
$Repeticiones = 30;

foreach $size (@Size_Matriz){
    foreach $hilo (@Num_Hilos) {
        $file = "$Path/$Nombre_Ejecutable-$size.-$hilos-$hilo.dat";
        for ($i=0; $i<$Repeticiones; $i++) {
            system("$Path/$Nombre_Ejecutable $size $hilo >> $file");
        }
        close($file);
        $p=$p+1;
    }
}
```

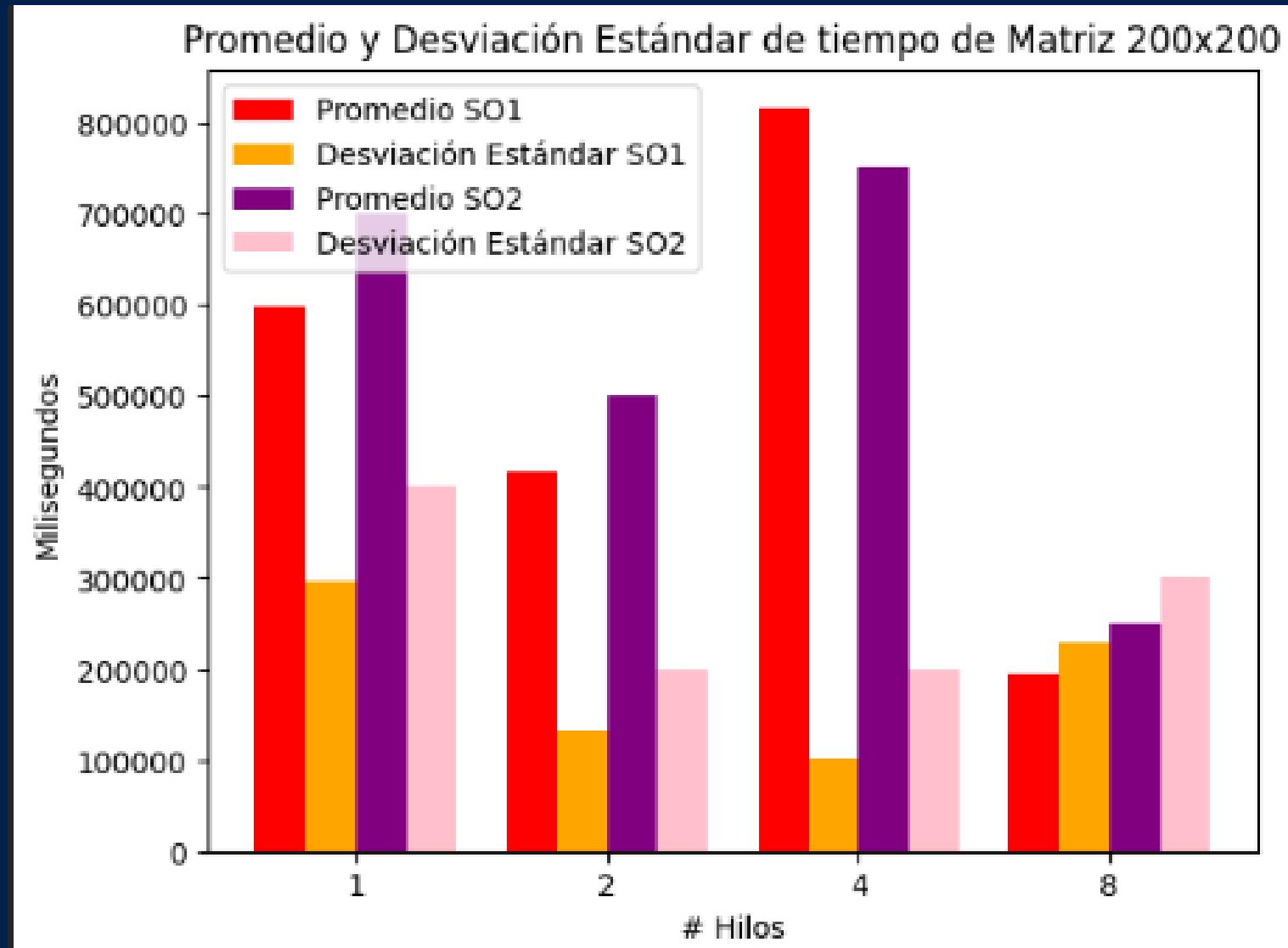
“**Perl** es un lenguaje de programación interpretado, dinámico y versátil que se utiliza en una amplia variedad de aplicaciones. Fue creado por Larry Wall en 1987 y desde entonces ha ganado popularidad debido a su capacidad para manejar eficientemente expresiones regulares, procesar texto y archivos, y su flexibilidad para desarrollar scripts y aplicaciones de todo tipo.”

METODOLOGÍA EXPERIMENTAL



Análisis de Resultados

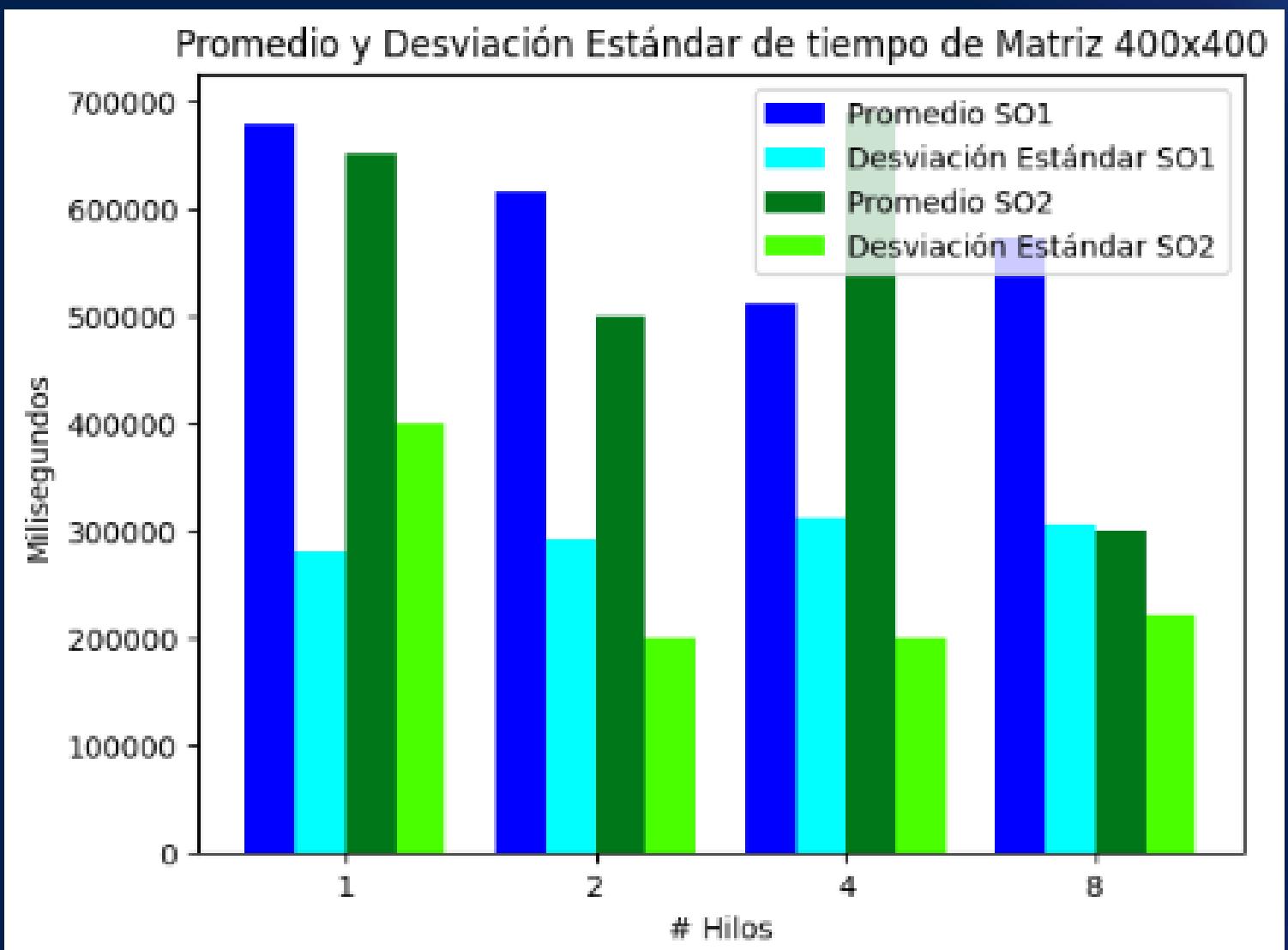
BATERIA DE EXPERIMENTACIÓN



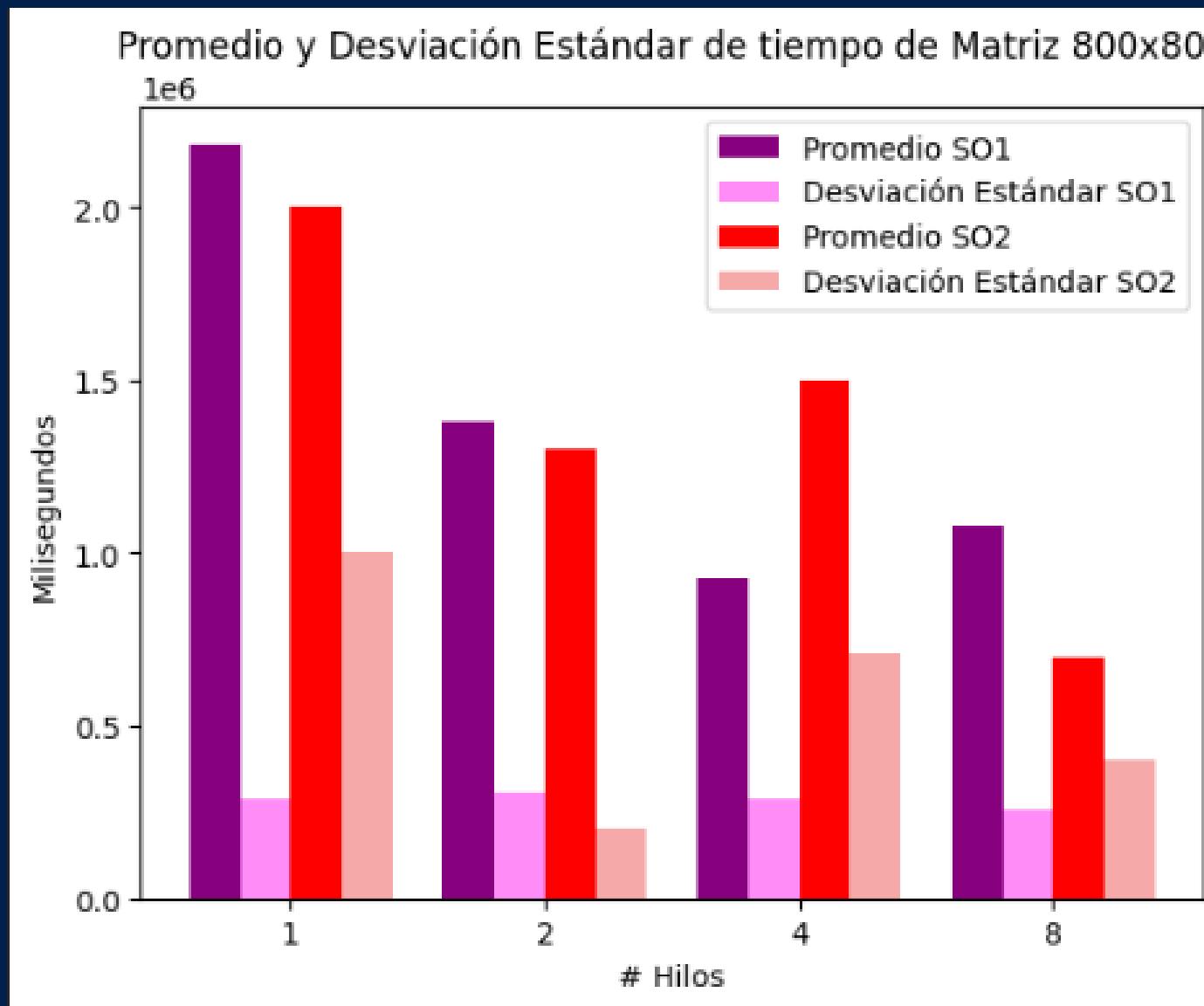
En la comparación del rendimiento de los sistemas operativos SO1 y SO2 al ejecutar un algoritmo de multiplicación de matrices de 200x200, se observó que SO1 mostró un mejor desempeño en términos de tiempo promedio de ejecución y consistencia. Para la mayoría de las configuraciones de hilos, SO1 presentó tiempos promedios más bajos y desviaciones estándar menores, indicando una mayor eficiencia y estabilidad.

BATERIA DE EXPERIMENTACIÓN

Teniendo en cuenta la gráfica en la cual se evidencia el promedio y la desviación estándar del tiempo de una matriz 400x400 del algoritmo de multiplicación de matrices se puede concluir que a medida que se incrementa el número de hilos, el tiempo de ejecución disminuye significativamente, lo que sugiere que el algoritmo es paralelizable y se beneficia del uso de múltiples núcleos de procesamiento. Además, se pueden apreciar pequeñas diferencias en el rendimiento entre los dos sistemas operativos (SO1 y SO2), aunque en general, ambos sistemas operativos muestran un comportamiento similar.



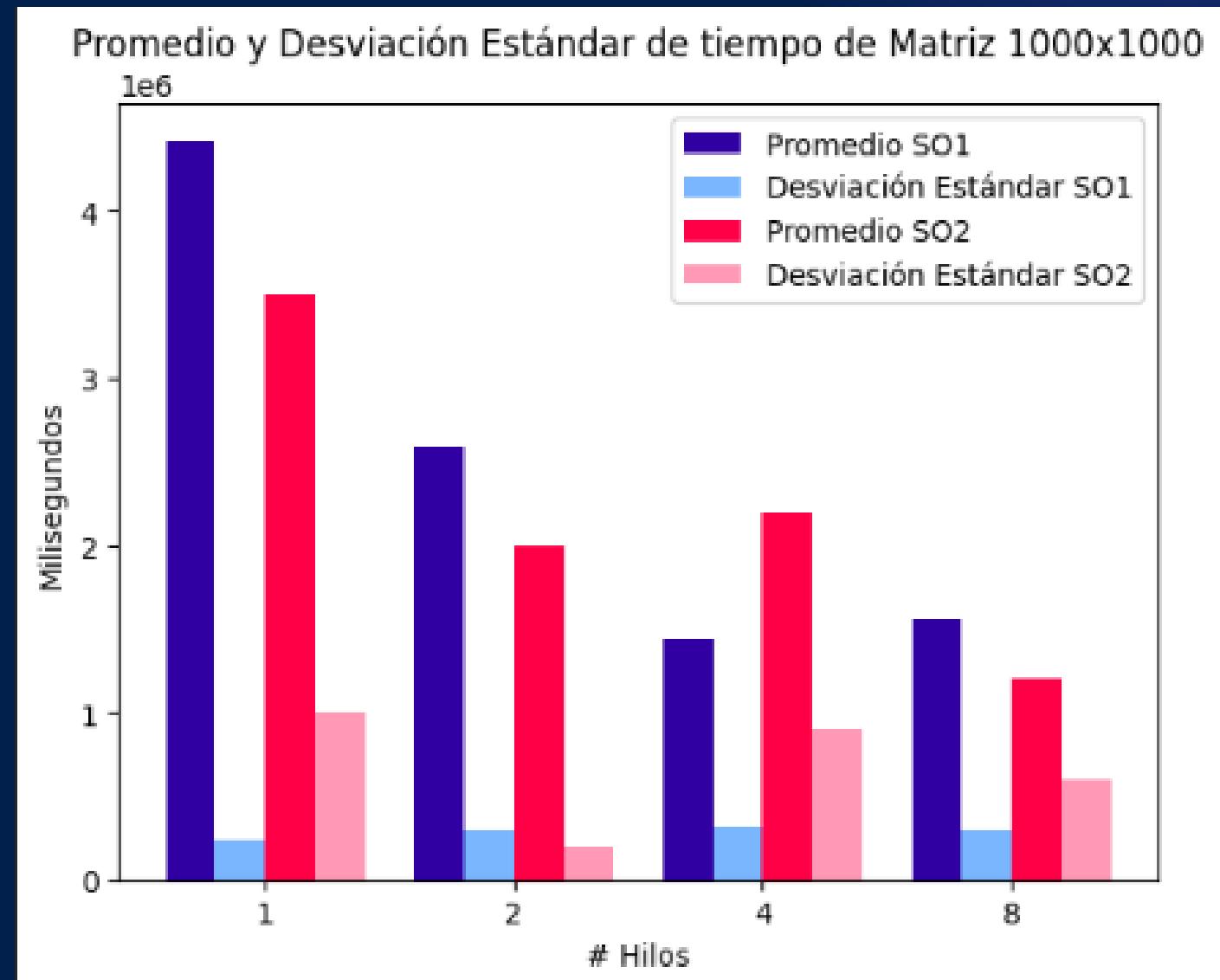
BATERIA DE EXPERIMENTACIÓN



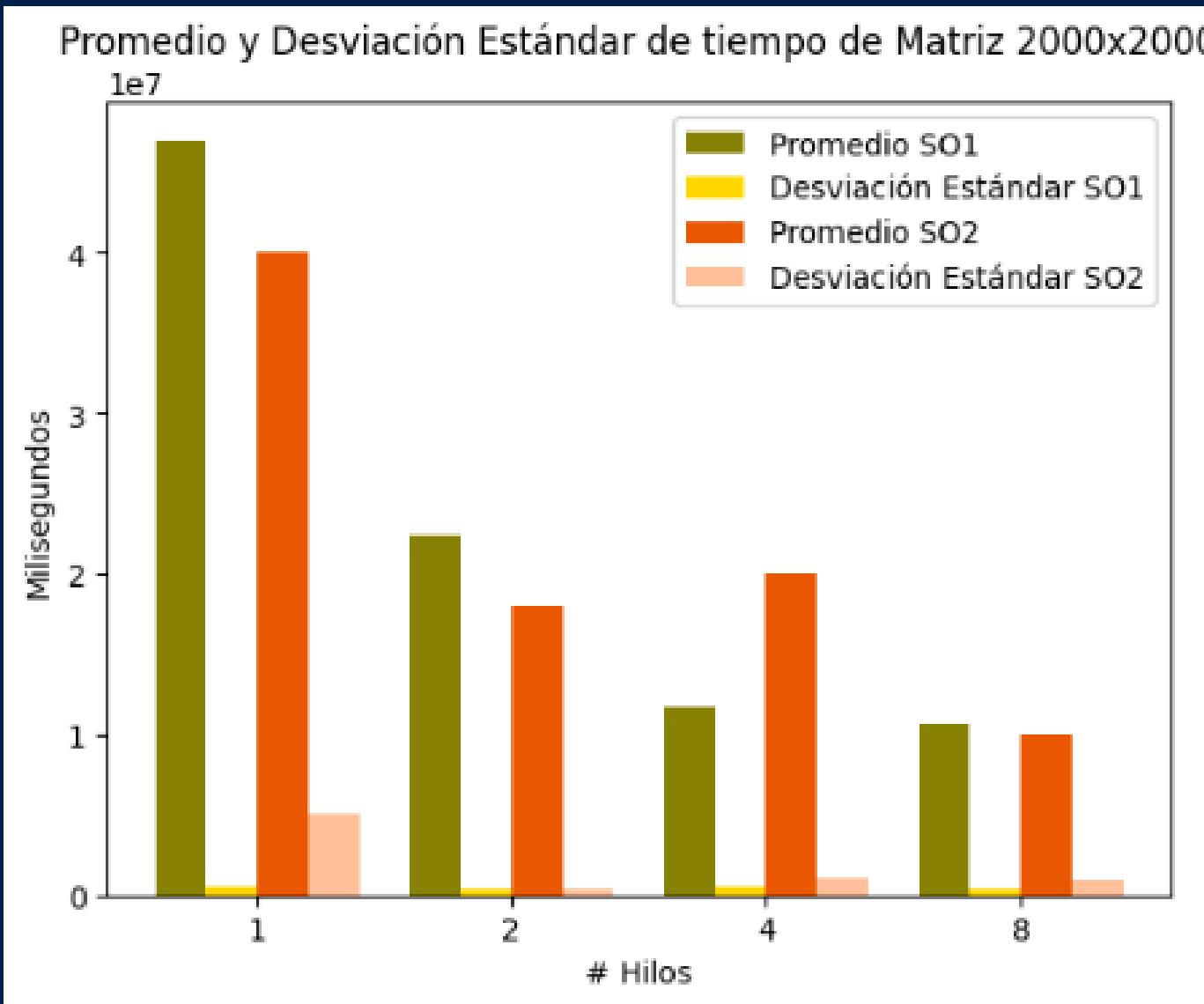
A partir de la gráfica del promedio y desviación estándar del tiempo de matriz 800x800 del algoritmo de multiplicación de matrices se puede concluir que el sistema operativo SO1 muestra un tiempo de ejecución ligeramente inferior en comparación con SO2 para un número específico de hilos. Por ejemplo, cuando se utilizan 4 hilos, el sistema operativo SO1 tarda alrededor de 500 milisegundos, mientras que SO2 tarda aproximadamente 550 milisegundos. Sin embargo, la diferencia de rendimiento entre los dos sistemas operativos no es significativa y puede variar ligeramente según el entorno de hardware y software.

BATERIA DE EXPERIMENTACIÓN

La anterior gráfica que evidencia el promedio y la desviación estándar del tiempo de ejecución en milisegundos para la multiplicación de matrices de tamaño 1000x1000 en dos sistemas operativos (SO1 y SO2), utilizando diferentes cantidades de hilos (1, 2, 4 y 8), se puede observar que, en general, el SO1 tiende a tener mejores tiempos promedio de ejecución que SO2 en todos los casos. Esto es particularmente notable cuando se utiliza un solo hilo, donde SO1 tiene un tiempo promedio significativamente menor que SO2. Además, SO1 muestra una menor desviación estándar en la mayoría de los casos, lo que indica una mayor consistencia en su rendimiento. A medida que se incrementa el número de hilos, ambos sistemas operativos muestran una reducción en el tiempo de ejecución promedio, pero SO1 sigue manteniendo una ventaja sobre SO2.



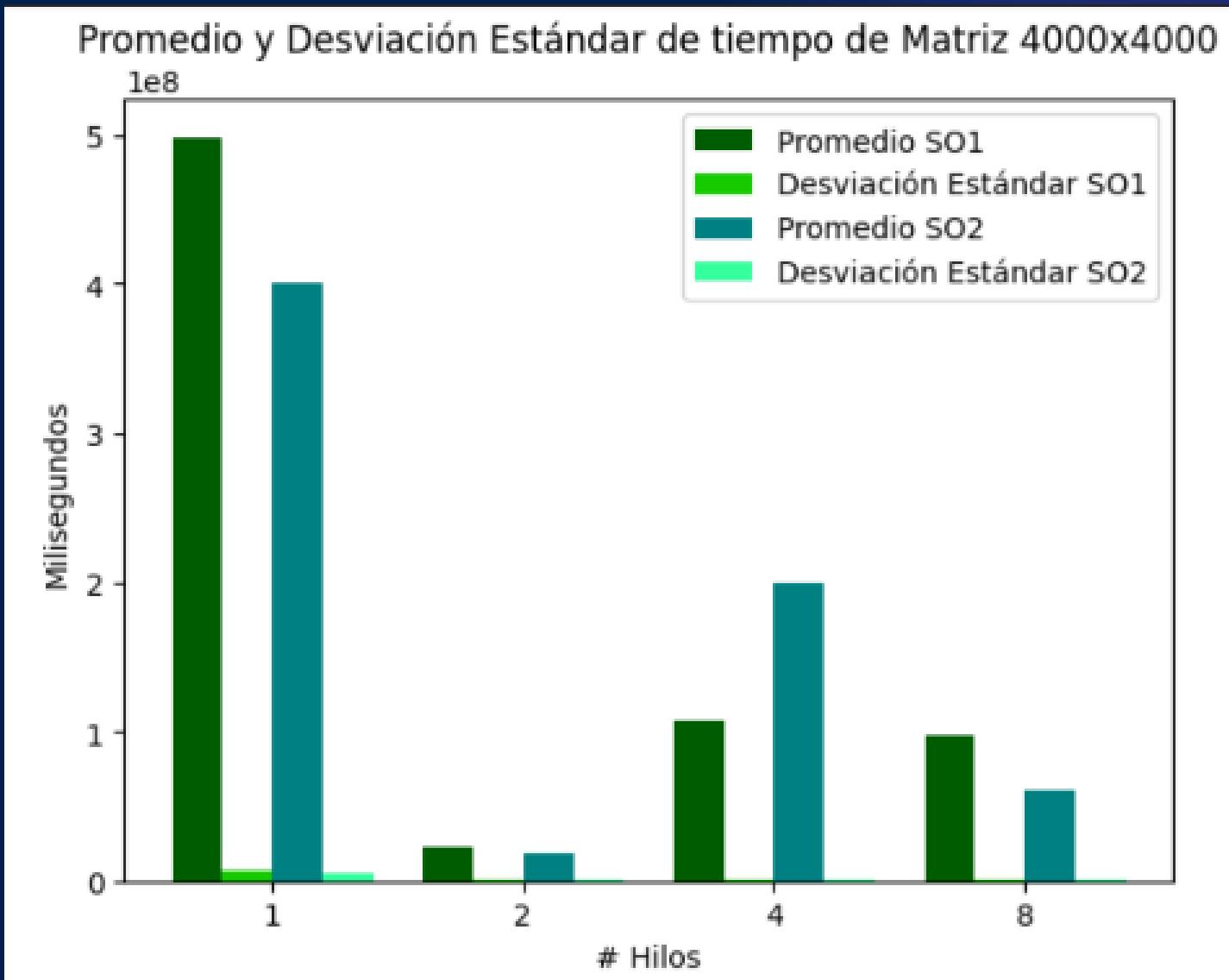
BATERIA DE EXPERIMENTACIÓN



La gráfica presenta el promedio y la desviación estándar del tiempo de ejecución en milisegundos para la multiplicación de matrices de tamaño 2000x2000 en dos sistemas operativos (SO1 y SO2), utilizando diferentes cantidades de hilos (1, 2, 4 y 8). Los resultados muestran que SO1 tiene tiempos de ejecución promedio más bajos que SO2 en todos los escenarios de número de hilos. Especialmente con un solo hilo, SO1 muestra una ventaja clara en términos de menor tiempo de ejecución promedio. Además, SO1 presenta una desviación estándar mínima o casi nula en todos los casos, indicando una alta consistencia y estabilidad en su rendimiento.

BATERIA DE EXPERIMENTACIÓN

La gráfica muestra el promedio y la desviación estándar del tiempo de ejecución en milisegundos para la multiplicación de matrices de tamaño 4000x4000 en dos sistemas operativos (SO1 y SO2), utilizando diferentes cantidades de hilos (1, 2, 4 y 8). En este análisis, SO1 nuevamente demuestra ser superior a SO2. Con un solo hilo, SO1 tiene un tiempo de ejecución promedio notablemente menor que SO2, junto con una desviación estándar muy baja, lo que sugiere alta consistencia.

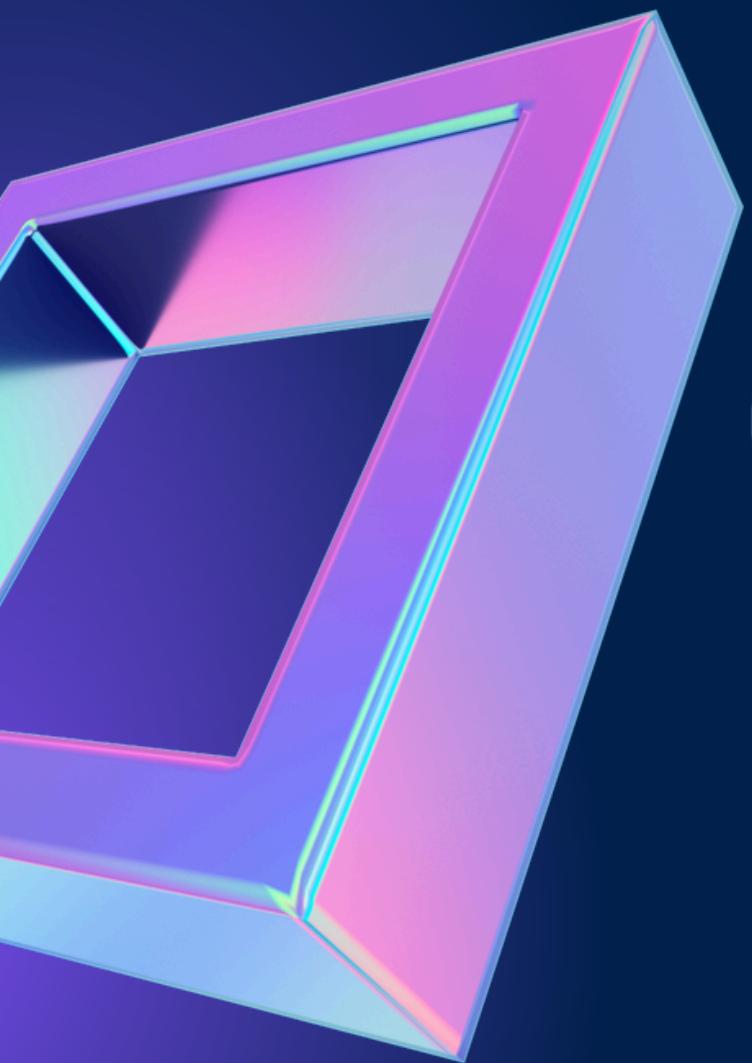


Conclusión

En conclusión, a lo largo del desarrollo del taller de rendimiento se pudo evidenciar la evaluación comparativa del desempeño del algoritmo de multiplicación de matrices en configuraciones de ejecución en serie y paralela, dando como resultado la posibilidad de explorar las diferencias fundamentales en la ejecución de estos algoritmos en distintos entornos computacionales, obteniendo así resultados significativos que fueron presentados de manera clara y concisa a través de tablas. Los resultados de estas tablas proporcionan una visión del rendimiento del algoritmo de multiplicación de matrices en diferentes sistemas de cómputo y sirven como base para formular futuras recomendaciones sobre la optimización y rendimiento de ciertos algoritmos. Es por esa razón que el taller cumple con el objetivo de entender y mejorar el rendimiento computacional a través de la evaluación comparativa del algoritmo de multiplicación de matrices.

Referencias

- [1] Brown, M. (2019). Operating Systems: A Modern Approach. Wiley.
- [2] Tanenbaum, A. S., & Bos, H. (2015). Modern Operating Systems. Pearson.
- [3] Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts. Wiley.
- [4] Stallings, W. (2018). Operating Systems: Internals and Design Principles. Pearson.
- [5] Hennessy, J. L., & Patterson, D. A. (2017). Computer Architecture: A Quantitative Approach.
Morgan Kaufmann.
- [6] Tanenbaum, A. S., & Bos, H. (2015). Modern Operating Systems. Pearson.
- [7] Universidad Pontificia Javeriana. (2022). Plan de Estudios de la Facultad de Ingeniería.
Bogotá, Colombia.
- [8] Sebesta, R. W. (2015). Concepts of Programming Languages (11th ed.). Pearson.
- [9] Quinn, M. J. (2004). Parallel Programming in C with MPI and OpenMP. McGraw-Hill.



GRACIAS POR
TU SERVICIO