Pontificia Universidad Javeriana

Facultad de Ingeniería

Curso: Sistemas Operativos

Profesor: John Corredor, Ph.D.

Fecha: 1 de mayo de 2025

Informe de Evaluación de Rendimiento: Multiplicación de Matrices

Autores:

Juan Sebastián Álvarez

Daniel Hoyos

Carlos Santiago Pinzón Caicedo
Samuel Jerónimo Gantiva Garzón
Jorge Enrique Olaya Liévano

Introducción

- **Objetivo del taller**: Este informe tiene como objetivo comparar el rendimiento de un algoritmo de multiplicación de matrices, usando diferentes configuraciones de sistemas de cómputo. Además, se busca proporcionar un análisis detallado de los resultados obtenidos mediante experimentación.
- **Motivación**: La multiplicación de matrices es una operación fundamental en muchas aplicaciones de cómputo de alto rendimiento, y este análisis ayuda a entender cómo el número de hilos y las características del sistema afectan su desempeño.

Métricas de Desempeño

- **Métricas comunes**: Para evaluar el desempeño de los algoritmos, se utilizan métricas como el tiempo de ejecución, el uso de CPU, la eficiencia del paralelismo y la escalabilidad. En este taller, la métrica principal utilizada fue el **tiempo de ejecución promedio**, dado que refleja directamente el rendimiento de los algoritmos en diferentes configuraciones.
- **Justificación de la métrica**: El tiempo de ejecución promedio fue seleccionado porque proporciona una medida clara de la eficiencia de los algoritmos y puede capturar la variabilidad inherente a los sistemas de cómputo.

Descripción de las Plataformas de Hardware y Software

- **Sistema 1**:
    - Procesador: Intel Core i7-7600U
    - CPU Hz: 2.80GHz
    - RAM: 16.27 GB
    - SO: Ubuntu 18.04.6 LTS
    - CPUs : 4
    - Threads: 2
    - Cores: 2
- **Sistema 2**:
    - Procesador: AMD Ryzen 5 5600H with Radeon Graphics
    - CPU Hz: 3.293GHz
    - SO: Ubuntu 24.04.2 LTS x86_64
    - RAM: 7.64 GB
    - CPUs: 12
    - Threads: 2
    - Cores: 6
- **Sistema 3**:
    - Procesador: AMD Ryzen 7 7435HS

    - CPU Hz: 4500MHz
    - SO: ArchLinux x86_64
    - RAM: 16 GB
    - CPUs: 16
    - Threads: 2
    - Cores: 8

- **Sistema 4**:
    - Procesador: AMD EPYC 7B13
    - CPU Hz: 3049.998 MHz
    - SO: Ubuntu 20.04.2 LTS
    - RAM: 64.31 GB
    - CPUs: 8
    - Threads: 2
    - Cores: 4
-
- **Sistema 5**:
    - Procesador: Intel Xeon Gold 6240R
    - CPU Hz: 2.4GHz
    - SO: Ubuntu 22.04.5 LTS x86_64
    - RAM: 12 GB
    - CPUs: 4
    - Threads: 1
    - Cores: 1

Procedimiento Experimental

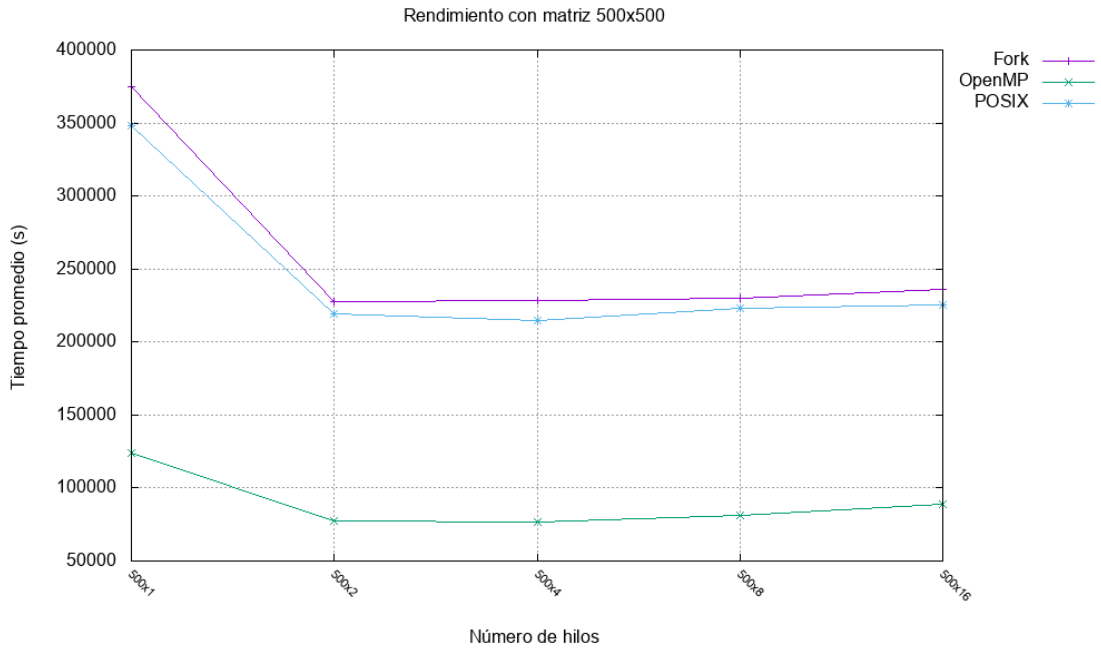
- **Configuración de los experimentos**:y
    - **Tamaño de las matrices (tamMatriz)**: Se seleccionaron tres tamaños de matrices: 500x500, 1000x1000 y 2000x2000. Estos tamaños fueron elegidos para cubrir diferentes rangos de complejidad, permitiendo evaluar el comportamiento del algoritmo en matrices de tamaño pequeño, mediano y grande.
    - **Número de hilos (NumHilos)**: Se probaron configuraciones con 1, 2, 4, 8 y 16 hilos. Esto permitió evaluar cómo la escalabilidad del algoritmo mejora al aumentar el número de hilos en función del número de núcleos disponibles.
    - **Repeticiones de los experimentos**: Cada experimento fue repetido 30 veces para obtener un promedio representativo y minimizar el error de medición, aplicando la ley de los grandes números.

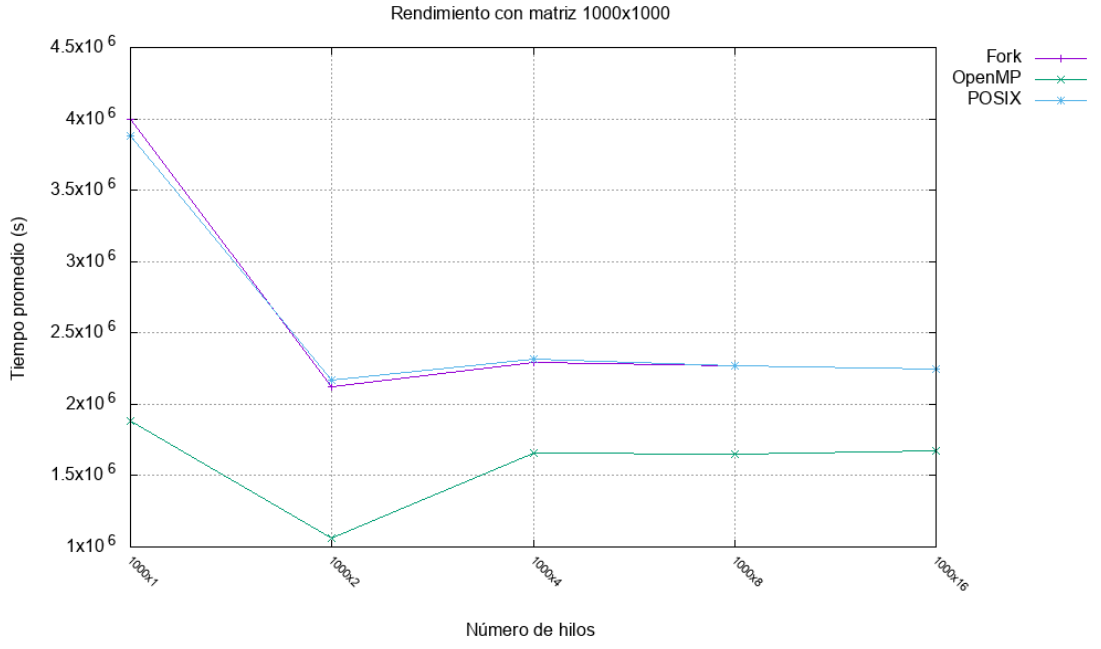
Resultados y Análisis

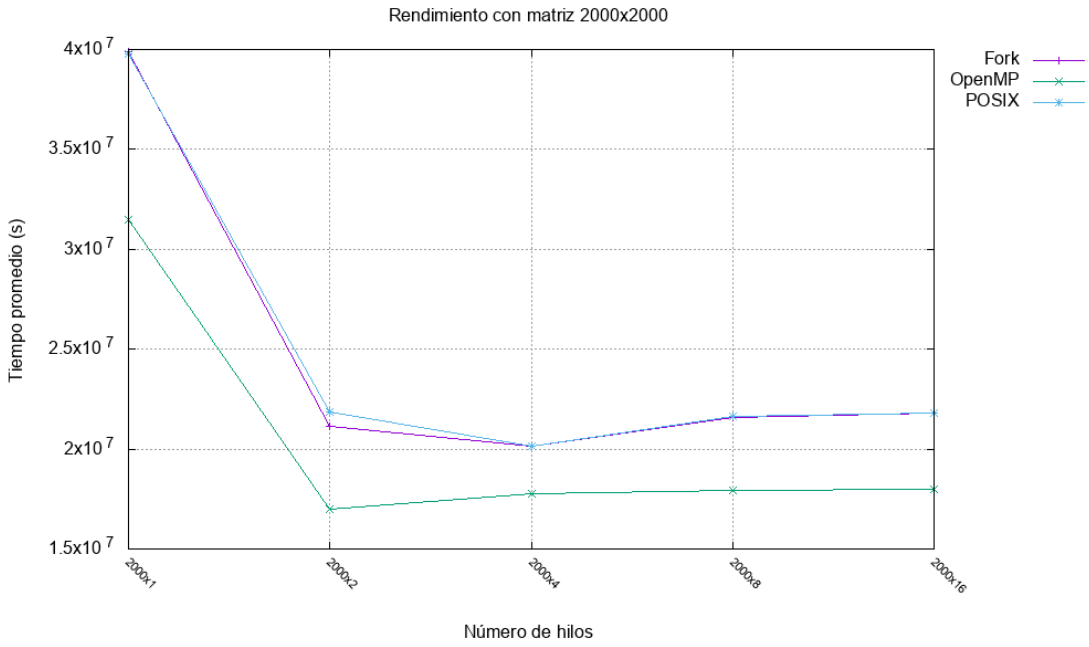
- **Presentación de los resultados**: Se presenta una tabla con los tiempos de ejecución obtenidos para cada combinación de tamMatriz y NumHilos en los 5 sistemas.

- **Sistema 1**:

Configuración	Promedio fork (ms)	Promedio OpenMP (ms)	Promedio Posix (ms)
500x1	375081	124287	348193
500x2	228023	77666.7	219651
500x4	228756	76787.6	214410
500x8	229725	81013.3	223147
500x16	235936	88520.1	225361
1000x1	4.00012e+06	1.88147e+06	3.87731e+06
1000x2	2.12348e+06	1.06197e+06	2.16799e+06
1000x4	2.28966e+06	1.65526e+06	2.3142e+06
1000x8	2.2685e+06	1.65185e+06	2.27337e+06
1000x16	2.24614e+06	1.67077e+06	2.250318e+06
2000x1	3.99151e+07	3.14675e+07	3.97581e+07
2000x2	2.11414e+07	1.6967e+07	2.18708e+07
2000x4	2.01224e+07	1.77672e+07	2.01391e+07
2000x8	2.16065e+07	1.79497e+07	2.16567e+07
2000x16	2.17968e+07	1.79861e+07	2.18017e+07

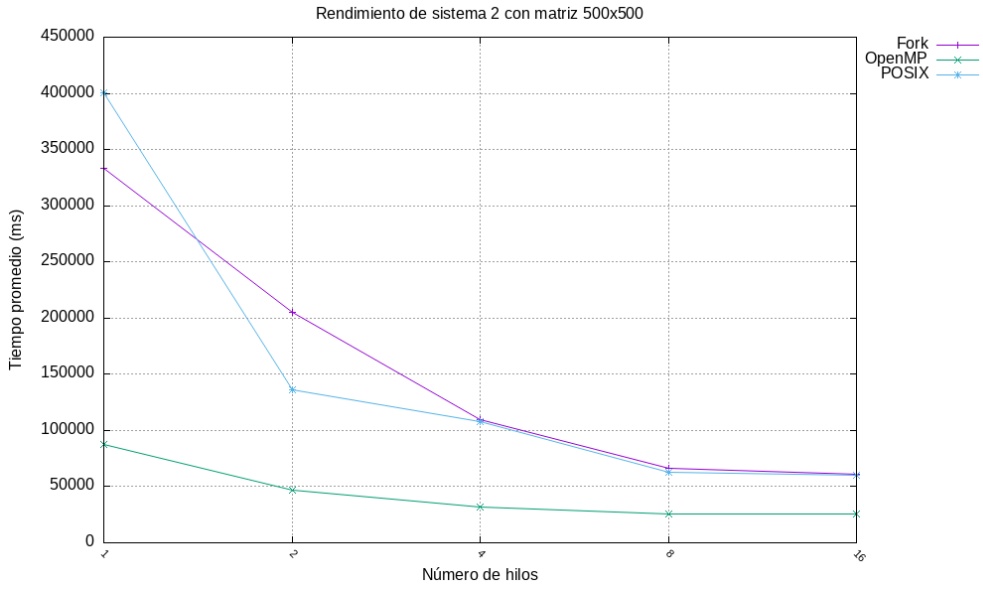

Rendimiento con matriz 500x500
Tiempo promedio (s)
400000
350000
300000
250000
200000
150000
100000
50000
500x1
500x2
500x4
500x8
500x16
Número de hilos
Fork
OpenMP
POSIX

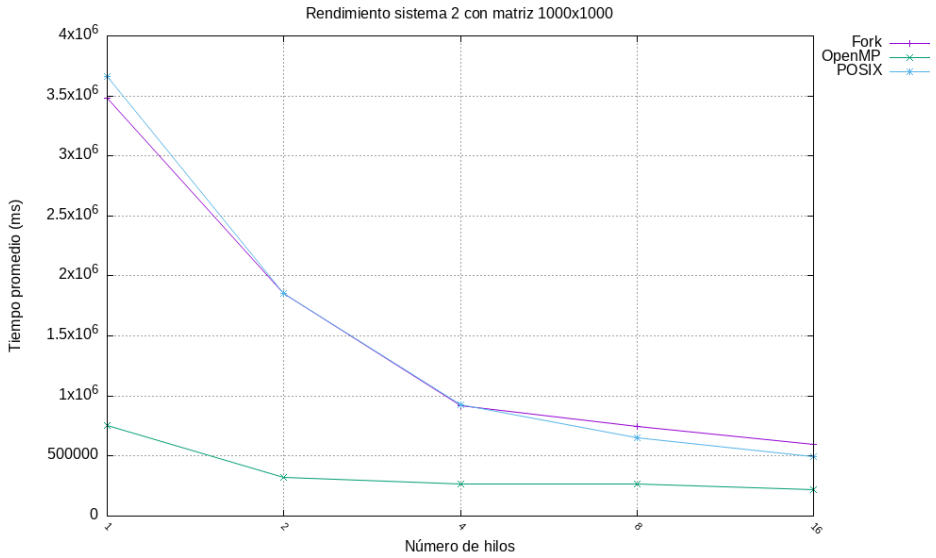

Rendimiento con matriz 1000x1000
Tiempo promedio (s)
4.5x10 6
4x10 6
3.5x10 6
3x10 6
2.5x10 6
2x10 6
1.5x10 6
1x10 6
1000x1
1000x2
1000x4
1000x8
1000x16
Número de hilos
Fork
OpenMP
POSIX

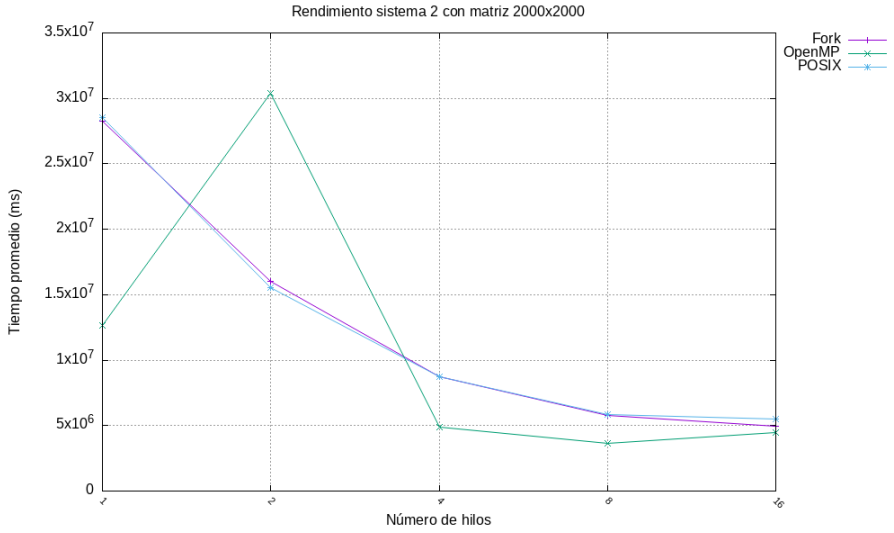


- **Sistema 2:**

Configuración	Promedio fork (ms)	Promedio OpenMP (ms)	Promedio Posix (ms)
500x1	333459	87371.1	400238
500x2	204840	46601.2	136203
500x4	109204	31916	108232
500x8	66744.1	25892.1	62849.3
500x16	61212	25385.7	60077.1
1000x1	3,48E+11	757059	3,66E+11
1000x2	1,86E+11	325864	1,86E+11
1000x4	915525	265446	930162
1000x8	748179	269462	654970
1000x16	594176	219941	494165
2000x1	2,83E+12	1,27E+12	2,85E+12
2000x2	1,60E+12	3,04E+12	1,55E+10
2000x4	8,72E+11	4,86E+11	8,73E+10
2000x8	5,78E+11	3,62E+11	5,85E+11
2000x16	4,94E+11	4,49E+11	5,53E+11

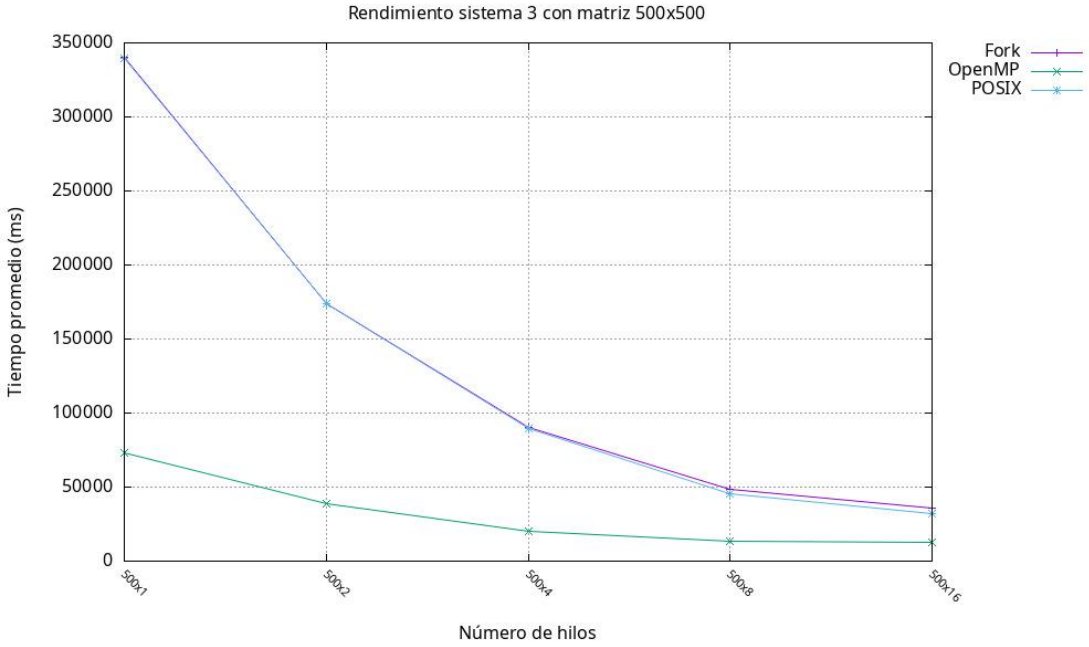

Rendimiento de sistema 2 con matriz 500x500
Tiempo promedio (ms)
Número de hilos
Fork
OpenMP
POSIX
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
1
2
4
8
16

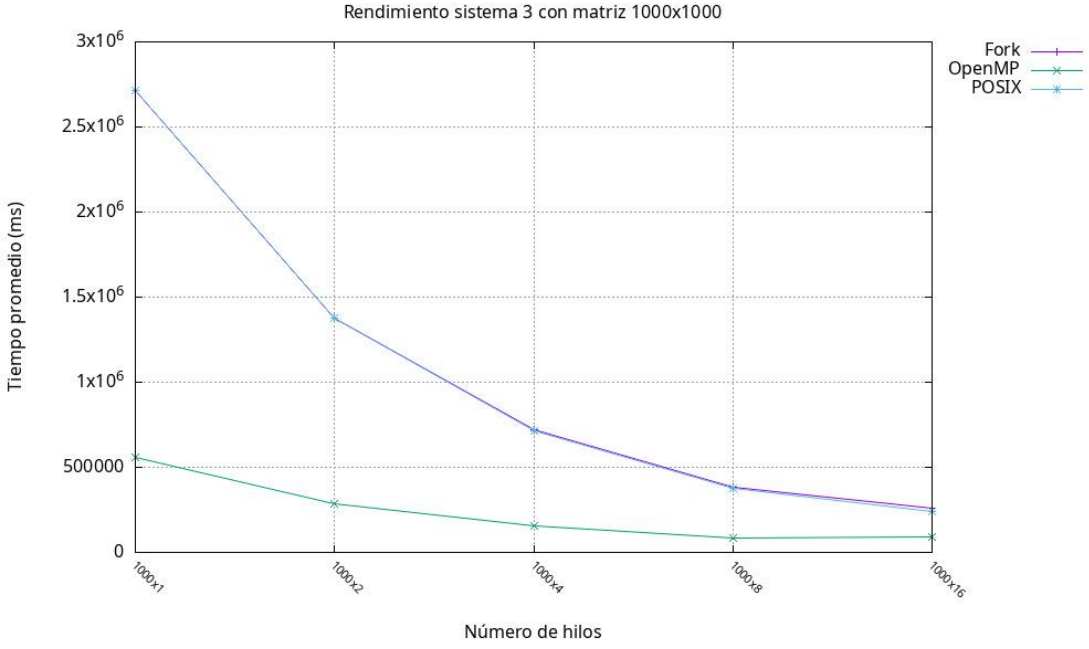

Rendimiento sistema 2 con matriz 1000x1000
Tiempo promedio (ms)
Número de hilos
Fork
OpenMP
POSIX
4x10^6
3.5x10^6
3x10^6
2.5x10^6
2x10^6
1.5x10^6
1x10^6
500000
0
1
2
4
8
16

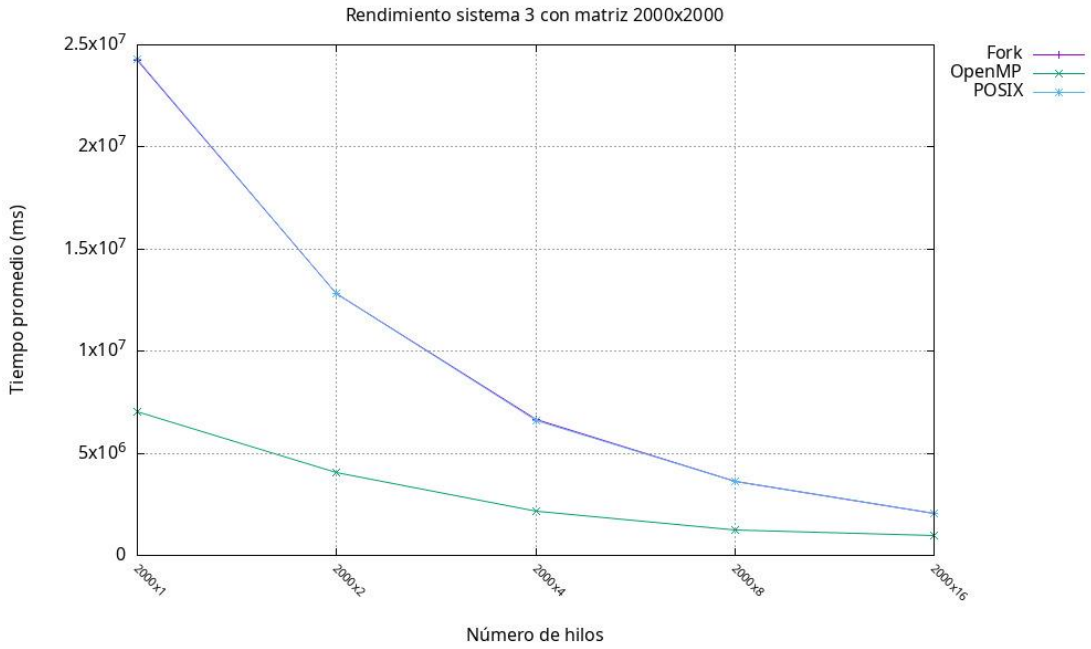


- **Sistema 3:**

Configuración	Promedio fork (ms)	Promedio OpenMP (ms)	Promedio Posix (ms)
500x1	340191	73109.3	339791
500x2	174012	38701.2	173509
500x4	90116.5	20200.5	89286.6
500x8	48601	13730.8	45880.1
500x16	35883.9	12548	31771.6
1000x1	2,71E+11	558444	2,72E+11
1000x2	1,38E+11	286036	1,38E+11
1000x4	721326	152835	715809
1000x8	382485	82739	377432
1000x16	258734	93608.8	237560
2000x1	2,43E+12	7,03E+11	2,43E+12
2000x2	1,28E+12	4,06E+10	1,28E+12
2000x4	6,65E+11	2,17E+11	6,62E+11
2000x8	3,64E+11	1,23E+11	3,62E+09
2000x16	2,03E+11	947484	2,04E+11

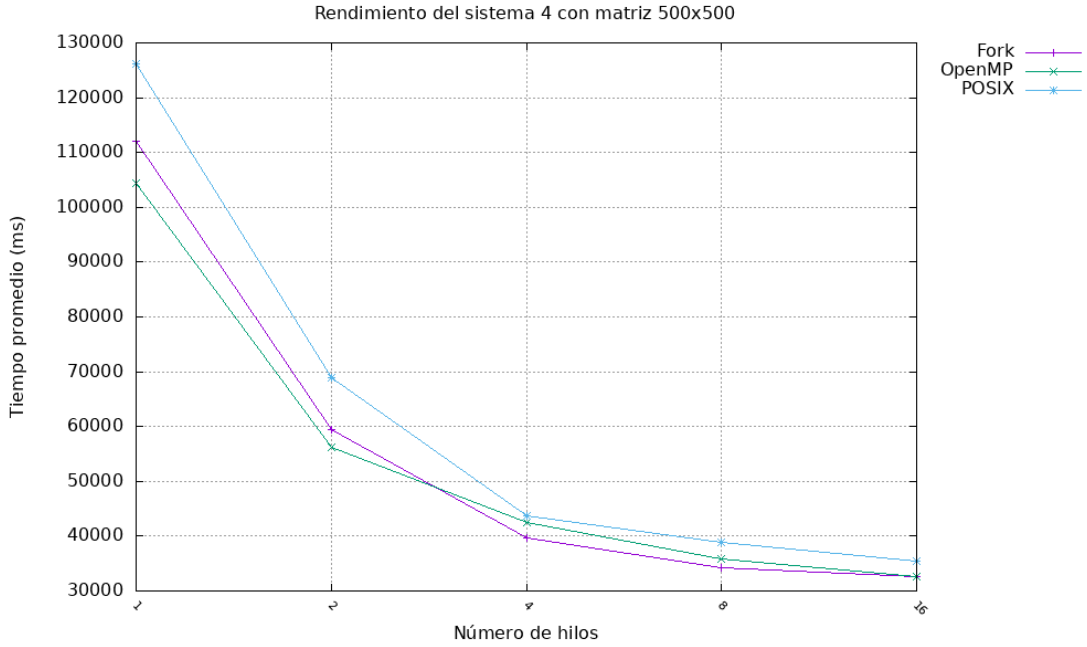

Rendimiento sistema 3 con matriz 500x500
Tiempo promedio (ms)
350000
300000
250000
200000
150000
100000
50000
0
500x1
500x2
500x4
500x8
500x16
Número de hilos
Fork
OpenMP
POSIX

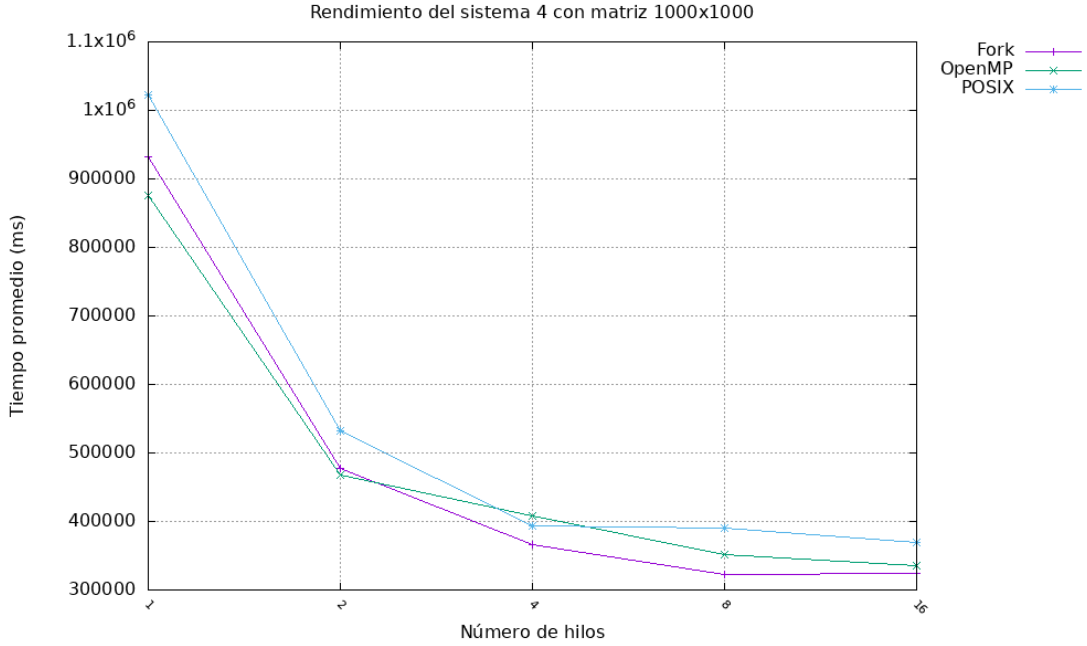

Rendimiento sistema 3 con matriz 1000x1000
Tiempo promedio (ms)
3x10^6
2.5x10^6
2x10^6
1.5x10^6
1x10^6
500000
0
1000x1
1000x2
1000x4
1000x8
1000x16
Número de hilos
Fork
OpenMP
POSIX

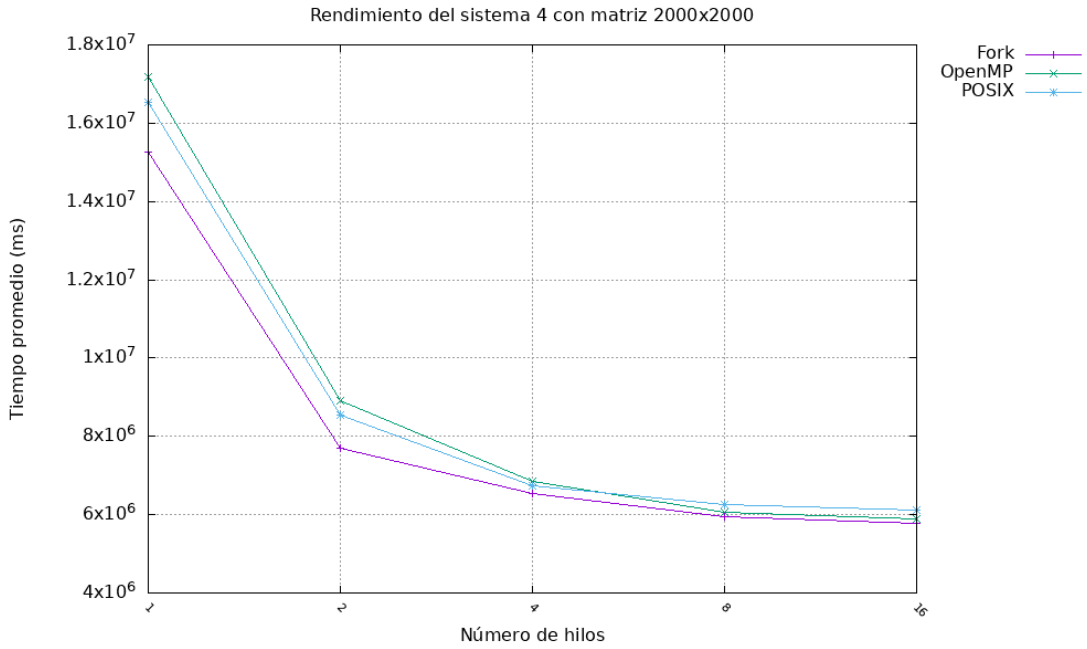


- **Sistema 4:**

Configuración	Promedio fork (ms)	Promedio OpenMP (ms)	Promedio Posix (ms)
500x1	112115	104380	126192
500x2	59512.2	56190.6	68859.6
500x4	39769.8	42513.3	43726.9
500x8	34143.4	35872.2	38778.3
500x16	32639.8	32536.9	35452.6
1000x1	932071	875351	1,02E+11
1000x2	476706	467572	532075
1000x4	365363	407998	393908
1000x8	323192	351595	389933
1000x16	323441	335386	368966
2000x1	1,53E+12	1,72E+12	1,65E+12
2000x2	7,69E+11	8,91E+10	8,54E+11
2000x4	6,55E+11	6,86E+11	6,73E+10
2000x8	5,96E+11	6,05E+11	6,26E+11
2000x16	5,77E+11	5,88E+11	6,11E+10

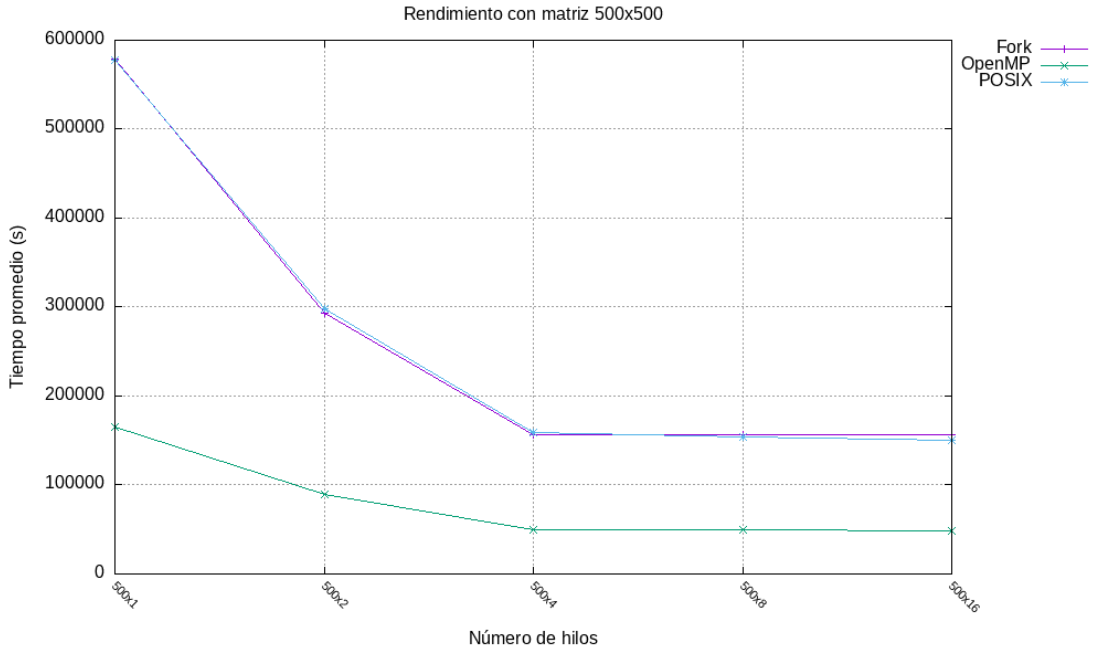

Rendimiento del sistema 4 con matriz 500x500
Tiempo promedio (ms)
130000
120000
110000
100000
90000
80000
70000
60000
50000
40000
30000
1
2
4
8
16
Número de hilos
Fork
OpenMP
POSIX

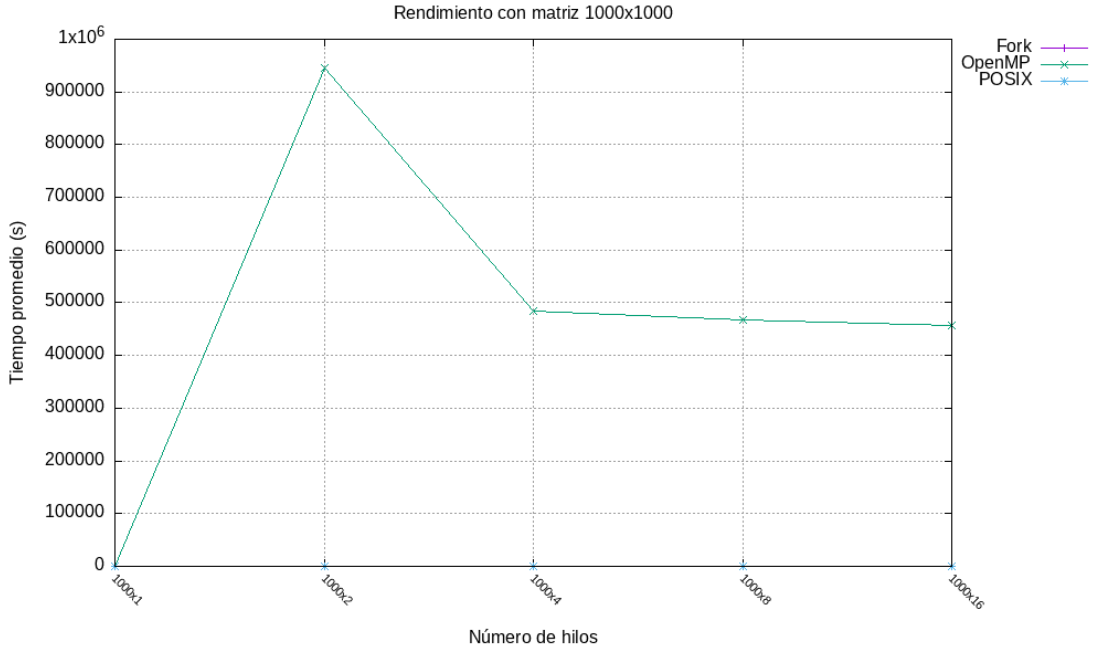

Rendimiento del sistema 4 con matriz 1000x1000
Tiempo promedio (ms)
1.1x10^6
1x10^6
900000
800000
700000
600000
500000
400000
300000
1
2
4
8
16
Número de hilos
Fork
OpenMP
POSIX

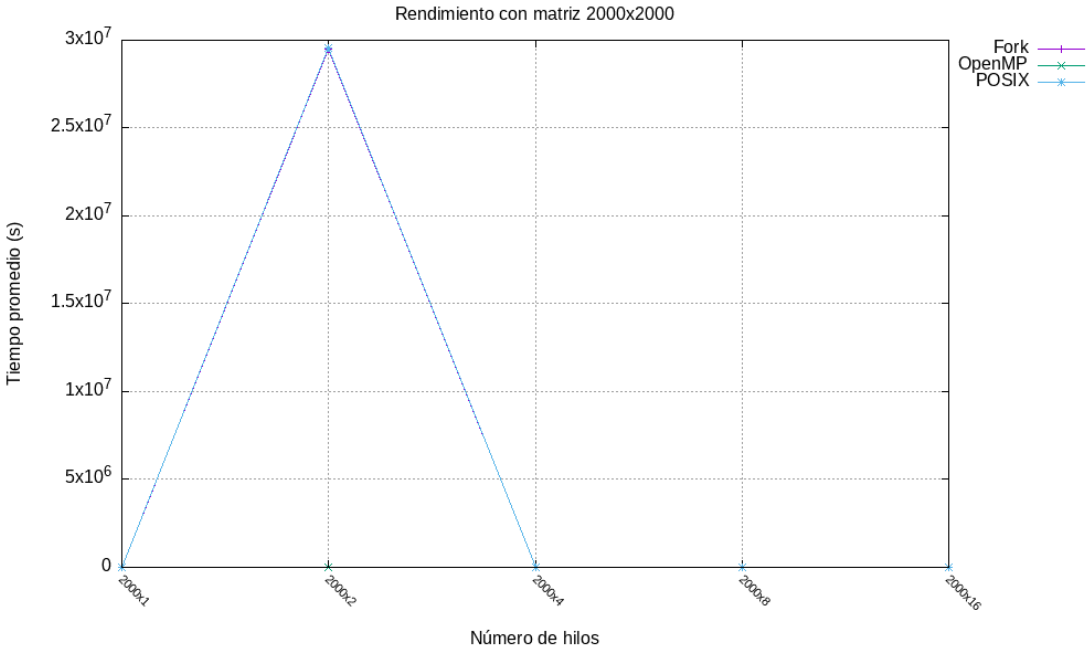


- **Sistema 5:**

Configuración	Promedio Fork (ms)	Promedio OpenMP(ms)	Promedio Posix(ms)
500x1	578747	165231	577243
500x2	292599	89175,7	298032
500x4	156744	49405,5	158697
500x8	156752	49438,5	154133
500x16	157122	48098,2	150363
1000x1	5,35E+06	1,84E+06	5,33E+06
1000x2	2,54E+06	946214	2,55E+06
1000x4	1,17E+06	483623	1,17E+06
1000x8	1,17E+06	468183	1,14E+06
1000x16	1,19E+06	458066	1,13E+06
2000x1	6,13E+07	5,04E+07	6,20E+07
2000x2	29517176	2,33E+07	29582691
2000x4	1,38E+07	1,04E+07	1,31E+07
2000x8	1,32E+07	1,06E+07	1,23E+07
2000x16	1,48E+07	1,12E+07	1,37E+07


Rendimiento con matriz 500x500
Tiempo promedio (s)
600000
500000
400000
300000
200000
100000
0
500x1
500x2
500x4
500x8
500x16
Número de hilos
Fork
OpenMP
POSIX


Rendimiento con matriz 1000x1000
Tiempo promedio (s)
1x10^6
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
1000x1
1000x2
1000x4
1000x8
1000x16
Número de hilos
Fork
OpenMP
POSIX



1. Rendimiento General

- **Versión en Serie (1 Hilo):**
    - La versión en serie (1 hilo) sirve como línea base para evaluar las mejoras del paralelismo. En todos los sistemas, los tiempos de ejecución con 1 hilo son significativamente mayores que con múltiples hilos para matrices grandes (1000x1000 y 2000x2000), lo que refleja la naturaleza intensiva en cómputo de la multiplicación de matrices.
    - Por ejemplo, en el Sistema 2 para matrices de 2000x2000, los tiempos en serie son:
        - Fork: 2.83E+12 ms (~28,300 s)
        - OpenMP: 1.27E+12 ms (~12,700 s)
        - Posix: 2.85E+12 ms (~28,500 s)
    - OpenMP muestra un mejor rendimiento en serie en la mayoría de los sistemas para matrices grandes, probablemente debido a optimizaciones internas del compilador o mejor gestión de caché.
- **Versiones Paralelas (2, 4, 8, 16 Hilos):**

- Las versiones paralelas reducen significativamente los tiempos de ejecución en la mayoría de las configuraciones, especialmente para matrices de 1000x1000 y 2000x2000, donde el paralelismo puede aprovechar múltiples núcleos.
- En el Sistema 3, para matrices de 2000x2000 con 16 hilos:
  - Fork: 2.03E+11 ms (~203 s)
  - OpenMP: 947,484 ms (~0.95 s)
  - Posix: 2.04E+11 ms (~204 s)
  - OpenMP logra una mejora drástica, reduciendo el tiempo en órdenes de magnitud comparado con la versión en serie, lo que sugiere una excelente escalabilidad en este sistema.

2. Anomalía en OpenMP (2000x2000, 2 Hilos):

- Una anomalía notable ocurre en el Sistema 2 para matrices de 2000x2000 con OpenMP, donde el tiempo con 2 hilos (3.04E+12 ms) es mayor que con 1 hilo (1.27E+12 ms). Esta anomalía no se observa en Fork ni Posix, que muestran mejoras consistentes al aumentar los hilos.
- Posibles causas incluyen:
  - **Falso uso compartido (false sharing):** Los hilos escribiendo en elementos adyacentes de la matriz resultado (matrixC) pueden causar invalidaciones de caché.
  - **Contención de ancho de banda de memoria:** Dos hilos compitiendo por acceso a memoria pueden saturar el bus.
  - **Sobrecarga de hilos:** La creación y gestión de hilos en OpenMP introduce un costo que puede superar los beneficios con solo 2 hilos.
- Esta anomalía no aparece en otros sistemas o tamaños de matriz, lo que sugiere una interacción específica entre el hardware del Sistema 2 (AMD Ryzen 5, 6 núcleos) y la implementación de OpenMP.

Impacto del Número de Hilos en el Rendimiento

1. Escalabilidad General

- En todos los sistemas, el aumento del número de hilos (de 1 a 16) generalmente reduce los tiempos de ejecución, especialmente para matrices grandes. Esto es esperado, ya que la multiplicación de matrices es altamente paralelizable, y más hilos distribuyen el trabajo entre los núcleos disponibles.
- **Ejemplo (Sistema 3, 1000x1000):**

- Fork: De 2.71E+11 ms (1 hilo) a 258,734 ms (16 hilos).
- OpenMP: De 558,444 ms (1 hilo) a 93,608.8 ms (16 hilos).
- Posix: De 2.72E+11 ms (1 hilo) a 237,560 ms (16 hilos).
- La mejora es más pronunciada en OpenMP, que logra tiempos significativamente menores con 16 hilos, probablemente debido a una mejor gestión de hilos por el runtime de OpenMP.

- **Límite de Escalabilidad:**
  - En algunos casos, el rendimiento se estabiliza o disminuye ligeramente con 16 hilos, especialmente en sistemas con menos núcleos (e.g., Sistema 1: 2 núcleos, 4 CPUs). Por ejemplo, en el Sistema 1 para 1000x1000:
    - OpenMP: 1.88147e+06 ms (1 hilo) → 1.67077e+06 ms (16 hilos), pero los tiempos con 8 y 16 hilos son similares (~1.65e+06 ms).
    - Esto indica saturación de recursos, como ancho de banda de memoria o contención en los núcleos físicos.

2. Dependencia del Hardware

- **Sistemas con Más Núcleos (Sistemas 2, 3, 4):**
  - El Sistema 3 (AMD Ryzen 7, 8 núcleos, 16 CPUs) muestra la mejor escalabilidad, especialmente con OpenMP, alcanzando tiempos muy bajos para 2000x2000 con 16 hilos (947,484 ms). Esto se debe a la alta capacidad de cómputo y la arquitectura moderna del procesador.
  - El Sistema 2 (AMD Ryzen 5, 6 núcleos, 12 CPUs) también escala bien, pero la anomalía con OpenMP y 2 hilos limita su rendimiento en ciertas configuraciones.
  - El Sistema 4 (AMD EPYC, 4 núcleos, 8 CPUs) muestra buena escalabilidad para matrices pequeñas y medianas, pero los tiempos para 2000x2000 son altos (~5.77E+11 ms con 16 hilos), posiblemente debido a limitaciones de memoria o menor frecuencia del procesador (3.05 GHz).

- **Sistemas con Menos Núcleos (Sistemas 1, 5):**
  - El Sistema 1 (Intel Core i7, 2 núcleos, 4 CPUs) muestra mejoras limitadas con más de 4 hilos, y los tiempos para 1000x1000 y 2000x2000 son similares entre 4, 8 y 16 hilos. Esto sugiere que los 2 núcleos físicos se saturan rápidamente.

- El Sistema 5 (Intel Xeon, 1 núcleo, 4 CPUs) su configuración (1 núcleo físico) implica que el paralelismo estaría severamente limitado, probablemente mostrando poca o ninguna mejora con más hilos.

Diferencias Notables entre los Sistemas

1. Rendimiento Absoluto

- **Sistema 3** destaca como el más rápido en la mayoría de las configuraciones, especialmente para matrices grandes y con OpenMP. Por ejemplo, para 2000x2000 con 16 hilos, OpenMP logra 947,484 ms, comparado con ~2E+11 ms en otros sistemas. Esto se atribuye a:
  - Mayor número de núcleos (8) y CPUs (16).
  - Alta frecuencia (4.5 GHz).
  - Arquitectura moderna del AMD Ryzen 7.
- **Sistema 1** tiene el peor rendimiento junto con el **Sistema 5** para matrices grandes, con tiempos de ~2E+07 ms para 2000x2000 incluso con 16 hilos. Esto se debe a su menor número de núcleos (2) y frecuencia más baja (2.8 GHz).
- **Sistema 4** muestra tiempos competitivos para matrices pequeñas (500x500), pero su rendimiento para 2000x2000 es inferior al de los Sistemas 2 y 3, posiblemente debido a la menor cantidad de núcleos (4) y una frecuencia moderada (3.05 GHz).

2. Impacto del Sistema Operativo

- Los sistemas usan diferentes versiones de Linux (Ubuntu 18.04, 20.04, 22.04, 24.04; ArchLinux). Sin embargo, no se observan diferencias claras atribuibles al sistema operativo, ya que el rendimiento está dominado por el hardware (núcleos, frecuencia, RAM) y la implementación del algoritmo.
- ArchLinux en el Sistema 3 podría ofrecer ligeras ventajas debido a su naturaleza optimizada, pero los datos no permiten concluir esto de forma definitiva.

3. RAM y Memoria

- Los sistemas con más RAM (Sistema 4: 64.31 GB; Sistemas 1, 3: ~16 GB) no muestran una ventaja clara para matrices de 2000x2000, lo que sugiere que el cuello de botella principal es el cómputo y no la capacidad de memoria.
- El Sistema 2, con solo 7.64 GB de RAM, logra un rendimiento competitivo, indicando que la RAM no es un factor limitante para los tamaños de matriz probados.

Análisis Específico de las Implementaciones

- **Fork:**
    - Consistentemente más lento que OpenMP en sistemas con muchos núcleos (e.g., Sistema 3), debido a la sobrecarga de crear procesos en lugar de hilos.
    - Sin embargo, no muestra anomalías como OpenMP y escala bien en todos los sistemas, especialmente en el Sistema 3.
- **OpenMP:**
    - Generalmente la implementación más rápida, especialmente en el Sistema 3, pero la anomalía con 2 hilos en el Sistema 2 para 2000x2000 indica sensibilidad a la configuración del hardware y la planificación de hilos.
    -
    - La directiva #pragma omp for con planificación estática puede ser subóptima para ciertos tamaños de matriz y números de hilos.
- **Posix Threads:**
    - Similar a Fork en términos de escalabilidad, pero más rápida en algunos casos (e.g., Sistema 2, 500x500 con 2 hilos: 136,203 ms vs. 204,840 ms para Fork).
    - La división explícita del trabajo en multiMatrixPosix parece mitigar problemas como el falso uso compartido, lo que explica su rendimiento más estable.

Conclusiones

1. **Ventajas del Paralelismo:** Las versiones paralelas superan significativamente a la versión en serie para matrices grandes, con mejoras más pronunciadas en sistemas con más núcleos (Sistemas 2, 3, 4). OpenMP es la implementación más eficiente en la mayoría de los casos, especialmente en el Sistema 3.
2. **Impacto del Número de Hilos:** Aumentar el número de hilos mejora el rendimiento hasta un punto donde la saturación de núcleos o la contención de memoria limita las ganancias. En sistemas con pocos núcleos (Sistema 1), el beneficio de más de 4 hilos es mínimo.
3. **Diferencias entre Sistemas:** El Sistema 3 (AMD Ryzen 7, 8 núcleos) ofrece el mejor rendimiento debido a su alta capacidad de cómputo, mientras que el Sistema 1 (Intel Core i7, 2 núcleos) es el menos eficiente. La anomalía de OpenMP en el Sistema 2 sugiere interacciones específicas entre el hardware y la implementación.

4. **Recomendaciones:** Para mitigar anomalías como la de OpenMP con 2 hilos, se sugiere probar planificaciones dinámicas (schedule(dynamic)), alinear la memoria para evitar falso uso compartido, y perfilar el uso de caché con herramientas como perf.

Repositorio

- https://github.com/Jeol28/Taller-de-Evaluacion
- https://github.com/Alviz09/SisOpsTallerEvaluacion
- https://github.com/CarlitosPinzon/TallerEvaluaci-n
- https://github.com/Danielhoyos06/Tallerevaluacionrend
- https://github.com/Gantu78/TallerEvaluaciondeRendimiento