Multiplicación de matrices

Autor 1: David Buitrago Rodriguez
Autor 2: Federico Pérez Ramirez
Autor 3: Nicolas Duque Gutierrez

Universidad Tecnológica de Pereira

***Resumen*— En el presente documento se analiza el rendimiento de la multiplicación de matrices cuadradas de tamaño nxn. Para ésto se implementa dicha multiplicación de manera secuencial y paralela programadas en el lenguaje C, el propósito es comparar el rendimiento presentado a la hora de ejecutar un programa secuencial, con hilos y procesos.**

***Palabras clave*— Rendimiento, matrices, secuencial, C, hilos, procesos, ejecución.**

I. INTRODUCCIÓN

La multiplicación de matrices es una operación fundamental tanto en el ámbito de las matemáticas como en la informática. Su aplicación se extiende a diversas áreas, como la visión por computadora, el procesamiento de señales y, especialmente, en la inteligencia artificial, donde desempeña un papel crucial en numerosos algoritmos y modelos.

La multiplicación de matrices de forma secuencial implica multiplicar cada par de elementos de las matrices en un orden predeterminado, todo dentro de un solo hilo de ejecución. Este enfoque es simple y fácil de implementar, pero puede volverse ineficiente cuando se manejan matrices de gran tamaño, ya que requiere un tiempo considerable.

Por otro lado, la multiplicación de matrices de forma paralela se basa en la idea de descomponer la tarea en múltiples subtareas que se ejecutan simultáneamente en varios hilos de ejecución o procesadores. Cada hilo se encarga de calcular una porción de la multiplicación de matrices, lo que acelera significativamente el proceso y reduce el tiempo de ejecución total. Sin embargo, este enfoque consume más recursos de la máquina, ya que involucra la coordinación y gestión de múltiples hilos de ejecución para lograr una mejora notable en el rendimiento, especialmente al lidiar con matrices de gran tamaño.

II. CONTENIDO

1. High Performance Computing

Campo de la informática que se centra en el uso de sistemas informáticos altamente potentes y avanzados para realizar cálculos y procesamientos de datos extremadamente complejos y demandantes en términos de recursos computacionales.

En un sistema de HPC, se utilizan múltiples procesadores (CPUs o, en algunos casos, GPUs) interconectados en paralelo para resolver tareas que requieren una gran cantidad de cálculos en un período de tiempo relativamente corto. Estos sistemas están diseñados para manejar una amplia gama de aplicaciones científicas, técnicas y empresariales que incluyen simulaciones de física, modelado climático, análisis de datos genéticos, análisis financiero, renderizado de gráficos en 3D, y muchas otras tareas intensivas en cómputo.

Algunas características distintivas de la computación de alto rendimiento incluyen:

- Paralelismo: Los sistemas HPC se basan en la idea de dividir tareas en partes más pequeñas que se pueden procesar en paralelo, lo que permite un procesamiento más rápido de datos.
- Gran capacidad de almacenamiento: Dado que muchas aplicaciones HPC generan y procesan grandes cantidades de datos, estos sistemas suelen estar equipados con unidades de almacenamiento masivo de alta velocidad.
- Redes de alta velocidad: La comunicación eficiente entre los nodos de procesamiento es esencial en HPC. Por lo tanto, estos sistemas suelen contar con redes de alta velocidad para garantizar una transferencia rápida de datos.
- Recursos compartidos: Los sistemas HPC suelen ser utilizados por múltiples usuarios o proyectos simultáneamente, por lo que se necesita una gestión eficiente de recursos y programación para garantizar un uso óptimo de los recursos disponibles.

2. Procesamiento paralelo

Es una técnica de computación que utiliza múltiples procesadores o núcleos de procesamiento para trabajar en un

problema en paralelo, en lugar de en serie, para así acelerar el tiempo de cálculo y mejorar la eficiencia en la computación. El procesamiento paralelo divide un problema en tareas más pequeñas y se distribuyen estas tareas a múltiples procesadores o núcleos de procesamiento para que trabajen simultáneamente, combinando los resultados de cada tarea para producir un resultado final del problema.

3. Hilos

Los hilos se refieren a la unidad más pequeña de procesamiento que se puede ejecutar en un sistema multiprocesador o multinúcleo, un hilo es una secuencia de instrucciones que se pueden ejecutar simultáneamente con otros hilos en el mismo procesador o núcleo.
Los hilos se utilizan en HPC para aprovechar el paralelismos nivel de tareas, se dividen las tareas en subtareas que se ejecutan en hilos diferentes en el mismo núcleo, de esta manera se puede aprovechar la capacidad de procesamiento paralelo del hardware sin necesidad de distribuir la carga de trabajo en diferentes núcleos.

4. Procesos

Un proceso hace referencia a una instancia de un programa en ejecución en un sistema paralelo siendo ejecutados en múltiples núcleos o procesadores, permitiendo así acelerar el tiempo de cálculo y mejorar la eficiencia en la computación. Los procesos pueden comunicarse con otros procesos a través de una red de comunicación, siendo esencial para el procesamiento paralelo, ya que los procesos deben compartir información y cooperar entre sí para resolver problemas complejos.
Los procesos en un sistema de HPC se gestionan mediante un planificador o un sistema de gestión de tareas que coordina la ejecución de múltiples procesos del sistema, el planificador determina qué procesos se ejecutan en qué núcleos o procesadores en un momento dado, y cómo se comunican los procesos entre sí para resolver un problema en particular.

5. Speedup

El speedup es una medida que evalúa la mejora del rendimiento al emplear un sistema de procesamiento en paralelo en contraste con uno en serie. Se calcula mediante la división del tiempo de ejecución de un programa en un sistema serie entre el tiempo de ejecución del mismo programa en un sistema paralelo, como se expresa matemáticamente:

Aunque el speedup ideal sería igual al número de procesadores o núcleos utilizados en el sistema paralelo, esto no siempre es factible debido a las restricciones en la comunicación y coordinación entre los procesadores y núcleos. Por lo tanto, un alto speedup indica que el sistema paralelo está aprovechando eficazmente los recursos disponibles y logrando una mejora sustancial en el rendimiento.

III. IMPLEMENTACIÓN

Para la implementación de este programa se realizó la multiplicación de matrices nxn primero de manera secuencial para ver el rendimiento, después de eso se implementaron hilos y procesos para tomar datos de ejecución, se implementó un script el cual ejecutaba diez veces el programa con cierta cantidad de hilos o procesos y así sacar un promedio de los datos y presentarlos en una tabla para la realización de los gráficos.

1. Características de la máquina

Las pruebas fueron realizadas en el lenguaje de programación C, con el sistema operativo Ubuntu con las siguientes especificaciones:

Características del PC donde se realizaron las pruebas:

- Arquitectura: 64-bit
- Procesador: AMD Ryzen 5 3400G
- Cantidad de núcleos: 4
- Cantidad de subprocesos/hilos: 8
- Frecuencia básica del procesador: 3.70GHz
- Ram 8GB DDR4 @ 3200 MHz
- SSD: 512GB - R: 550 MB/s - W: 490 MB/s
- Sistema operativo: Ubuntu 22.04.3 LTS

2. Implementación secuencial

Para la implementación de la multiplicación de matrices de forma secuencial se ejecutó un script diez veces para cada de los siguientes tamaños: [200, 400, 800, 1600, 3200, 6400] para obtener un tiempo de ejecución promedio.

Resultados versión secuencial:

T_n

	200	400	800	1600	3200	6400
0	0.029700	0.227301	1.952989	15.466263	118.278389	939.136882
1	0.030178	0.238613	1.997104	14.931243	117.314435	935.287559
2	0.029998	0.254739	2.063474	14.957889	119.760437	935.735884
3	0.028522	0.261676	1.929972	15.022960	118.108436	941.384527
4	0.059079	0.244561	1.965212	15.034723	117.964628	937.217924
5	0.029705	0.264673	2.001673	14.876862	117.488067	935.036017
6	0.030344	0.274548	2.037526	14.951660	117.043289	937.441753
7	0.029423	0.234248	1.929067	14.948540	118.542097	937.322257
8	0.029764	0.233092	2.086985	15.108062	118.370987	940.467287
9	0.030067	0.252788	1.963180	15.120637	119.192614	936.127180

Promedio	0.032678	0.248624	1.992718	15.041884	118.206338	937.515727

$$Speedup = \frac{Ts}{}$$

Donde Ts es el tiempo de ejecución en serie y Tp el tiempo de ejecución en paralelo.

3. Implementación con hilos

Para la implementación con hilos se hizo uso de la librería POSIX Threads la cual permite la creación y el manejo de hilos, cada hilo es el encargado de realizar la multiplicación de un segmento de la matriz, para así dividir la carga total de procesado entre los diferentes hilos y conseguir un mejoramiento en el tiempo de ejecución de todo el programa.

Se ejecutó un script para realizar seis pruebas con diferentes cantidades de hilos con el objetivo de comparar sus tiempos: [2, 4, 8, 16, 32, 64], y al igual que con la ejecución secuencial se ejecutó diez veces para cada de los siguientes tamaños de matrices: [200, 400, 800, 1600, 3200, 6400] para obtener un tiempo de ejecución promedio.

Resultados con 2 hilos:

	200	400	800	1600	3200	6400
0	0.017864	0.113159	1.108867	8.718152	67.743919	547.653070
1	0.033265	0.143024	1.021751	8.666452	66.916420	512.384182
2	0.015976	0.137339	0.920126	9.250356	69.768830	527.998757
3	0.013394	0.138833	1.079763	7.877796	66.019887	524.295418
4	0.013328	0.154664	0.938181	8.503334	69.620959	505.514250
5	0.013766	0.135465	0.990477	8.836030	70.877031	551.152603
6	0.015288	0.133719	1.177069	8.006726	62.912829	518.850536
7	0.036347	0.137654	1.228550	7.891151	70.684221	529.471167
8	0.044336	0.134435	0.941363	8.848298	63.404304	509.899894
9	0.063294	0.142555	0.920119	9.296894	63.366327	517.320141
Promedio	**0.026686**	**0.137085**	**1.032627**	**8.589519**	**67.131473**	**524.454002**
Speedup	**1.224546**	**1.813652**	**1.929757**	**1.751191**	**1.760818**	**1.787603**

Resultados con 4 hilos:

	200	400	800	1600	3200	6400
0	0.016447	0.092819	0.551952	5.035581	31.705926	250.271575
1	0.017889	0.088512	0.498812	4.143292	31.833621	242.747213
2	0.016610	0.069741	0.558480	4.292156	32.476815	258.534445
3	0.017110	0.076765	0.566860	4.155222	31.130643	252.471555
4	0.019083	0.081197	0.613856	4.145516	32.931419	240.361337
5	0.018997	0.083757	0.509862	3.814114	31.455671	246.136628
6	0.016904	0.084254	0.510225	4.867326	32.055069	243.782873
7	0.023286	0.076627	0.637000	4.033649	32.444943	244.273971
8	0.034420	0.076419	0.497412	4.403582	32.824860	244.678912
9	0.018358	0.072111	0.529251	4.371432	31.903608	244.020598
Promedio	**0.019910**	**0.080220**	**0.547371**	**4.326187**	**32.076258**	**246.727911**
Speedup	**1.641253**	**3.099268**	**3.640526**	**3.476938**	**3.685166**	**3.799796**

Resultados con 8 hilos:

	200	400	800	1600	3200	6400
0	0.015840	0.054651	0.449578	3.494414	27.481053	220.012488
1	0.016754	0.051545	0.418319	3.457704	27.529816	219.724832
2	0.015837	0.051604	0.418916	3.433856	27.493648	219.310477
3	0.016070	0.053792	0.429185	3.433741	27.599041	219.562956
4	0.016072	0.053031	0.425855	3.425608	27.629378	219.952454
5	0.016055	0.052062	0.416227	3.440212	27.580761	219.815546
6	0.016363	0.052053	0.444444	3.476621	27.582373	220.683696
7	0.016472	0.054279	0.428946	3.468140	27.572141	220.703779
8	0.012277	0.053255	0.428022	3.507988	27.634310	220.366450
9	0.016205	0.052622	0.436368	3.475028	27.692273	220.288710
Promedio	**0.015795**	**0.052889**	**0.429586**	**3.461331**	**27.579479**	**220.042139**
Speedup	**2.068948**	**4.700827**	**4.638694**	**4.345693**	**4.286025**	**4.260619**

Resultados con 16 hilos:

	200	400	800	1600	3200	6400
0	0.009811	0.053132	0.425365	3.447604	27.707515	219.994358
1	0.008166	0.054089	0.424958	3.457278	27.616375	220.124751
2	0.006917	0.053723	0.429169	3.436643	30.586280	220.512503
3	0.013719	0.055269	0.436077	3.427885	27.683415	220.194866
4	0.008414	0.059768	0.426050	3.433129	27.669987	222.549420
5	0.007021	0.054648	0.425854	3.468023	27.794165	221.682747
6	0.007041	0.053929	0.427757	3.453714	28.247147	220.519429
7	0.008356	0.054164	0.430661	3.447984	28.424903	219.611521
8	0.007353	0.052934	0.440501	3.420969	29.013516	220.733240
9	0.008059	0.056989	0.433892	3.464890	27.608363	220.301878
Promedio	**0.008486**	**0.054865**	**0.430028**	**3.445812**	**28.235167**	**220.622471**
Speedup	**3.850949**	**4.531599**	**4.633922**	**4.365266**	**4.186493**	**4.249412**

Resultados con 32 hilos:

	200	400	800	1600	3200	6400
0	0.006870	0.052474	0.428336	3.452776	27.791757	221.286997
1	0.007101	0.053301	0.420934	3.461810	28.283597	221.181639
2	0.007851	0.053498	0.422917	3.451876	28.721639	221.676415
3	0.007860	0.052915	0.423476	3.439788	27.886055	220.848056
4	0.007509	0.052864	0.422264	3.451026	27.780398	221.012833
5	0.007036	0.052554	0.430701	3.453786	27.730238	223.106823
6	0.007464	0.053579	0.421070	3.440378	27.787811	220.927865
7	0.007661	0.053176	0.424107	3.444913	27.809659	221.486463
8	0.007231	0.052451	0.422395	3.420739	27.816104	222.349172
9	0.006973	0.053676	0.422013	3.421331	27.696489	221.226075
Promedio	**0.007356**	**0.053049**	**0.423821**	**3.443842**	**27.930375**	**221.510234**
Speedup	**4.442602**	**4.686702**	**4.701789**	**4.367762**	**4.232179**	**4.232381**

Resultados con 64 hilos:

	200	400	800	1600	3200	6400
0	0.007636	0.053983	0.420388	3.460064	28.313733	222.669921
1	0.007675	0.053497	0.419036	3.433500	27.781776	222.400896
2	0.007252	0.052926	0.426435	3.419380	27.786697	223.124558
3	0.007075	0.053529	0.424479	3.453012	27.795203	222.327829
4	0.007270	0.053683	0.418635	3.473621	27.852252	222.762566
5	0.007711	0.054475	0.421014	3.431936	27.884975	222.314615
6	0.007153	0.053588	0.425457	3.442467	27.888360	223.417526
7	0.007334	0.052287	0.428737	3.483859	27.867309	222.285137
8	0.007702	0.052752	0.423497	3.418084	27.750984	222.435128
9	0.007574	0.053375	0.425151	3.456406	27.901179	223.064208
Promedio	**0.007438**	**0.053410**	**0.423283**	**3.447233**	**27.882247**	**222.680238**
Speedup	**4.393267**	**4.655050**	**4.707769**	**4.363466**	**4.239484**	**4.210143**

4. Implementación con procesos

Para la implementación con procesos se hizo uso de la librería unistd la cual permite la creación y el manejo de procesos, cada proceso es el encargado de realizar la multiplicación de un segmento de la matriz, para así dividir la carga total de procesado entre los diferentes procesos y conseguir un mejoramiento en el tiempo de ejecución de todo el programa.

Se ejecutó un script para realizar seis pruebas con diferentes cantidades de procesos con el objetivo de comparar sus tiempos: [2, 4, 8, 16, 32, 64], y al igual que con la ejecución secuencial se ejecutó diez veces para cada de los siguientes tamaños de matrices: [200, 400, 800, 1600, 3200, 6400] para obtener un tiempo de ejecución promedio.

Resultados con 2 procesos:

	200	400	800	1600	3200	6400
0	0.064564	0.268339	2.129064	16.710073	133.613314	1065.760423
1	0.05797	0.29902	2.136497	16.638627	133.759294	1059.320293
2	0.035527	0.29278	2.116713	16.872579	139.106475	1057.113326
3	0.050968	0.308938	2.180032	17.379172	137.640188	1057.409026
4	0.062126	0.296826	2.216654	17.316675	137.550909	1059.963941
5	0.049967	0.289849	2.239238	17.465206	132.080639	1056.99777
6	0.058172	0.274485	2.189809	16.578613	137.880572	1056.424415
7	0.063555	0.282469	2.265779	16.531093	134.55145	1058.424316
8	0.056835	0.29265	2.120108	17.048771	136.599588	1056.176287
9	0.034699	0.285687	2.146897	16.763706	133.793095	1061.877834
Promedio	**0.053438**	**0.289104**	**2.174079**	**16.930452**	**135.657552**	**1058.946763**
Speedup	**0.611509**	**0.859980**	**0.916580**	**0.888451**	**0.871358**	**0.885328**

Resultados con 4 procesos:

	200	400	800	1600	3200	6400
0	0.041691	0.167923	1.086827	8.580074	67.162939	537.974557
1	0.018031	0.15086	1.05149	8.433829	67.378038	539.431879
2	0.015963	0.135096	1.074222	8.397975	67.582689	539.987541
3	0.040079	0.138939	1.082018	8.456724	67.871539	540.785947
4	0.044166	0.159204	1.082223	8.41903	67.644547	540.362048
5	0.020504	0.14451	1.086543	8.484007	67.614213	540.338665
6	0.016753	0.170384	1.080619	8.431882	67.757854	540.827939
7	0.036383	0.137852	1.087282	8.466313	67.747652	540.637908
8	0.043447	0.182388	1.095392	8.466148	67.605069	541.351241
9	0.04822	0.163929	1.077477	8.468232	67.679128	541.425197
Promedio	**0.032524**	**0.155109**	**1.080409**	**8.460421**	**67.604367**	**540.312292**
Speedup	**1.004744**	**1.602903**	**1.844410**	**1.777912**	**1.748502**	**1.735137**

Resultados con 8 procesos:

	200	400	800	1600	3200	6400
0	0.029651	0.119988	0.950009	7.608867	60.862941	486.995256
1	0.016632	0.124924	0.952674	7.628961	60.919198	487.206405
2	0.019171	0.124222	0.961902	7.630734	61.693693	487.485346
3	0.022872	0.127739	0.989245	7.612795	61.005698	487.316821
4	0.01677	0.121304	0.950758	8.029466	60.920257	487.170616
5	0.015788	0.121691	0.952711	7.955992	60.899206	487.940354
6	0.01505	0.132846	0.950566	7.726207	60.92083	487.287342
7	0.015046	0.121685	0.955708	7.680901	61.283072	487.585484
8	0.015621	0.122228	0.963993	7.632638	60.952882	487.340814
9	0.019323	0.122463	0.948489	7.670429	60.983156	488.213672
Promedio	**0.018592**	**0.123909**	**0.957606**	**7.717699**	**61.044093**	**487.454211**
Speedup	**1.757600**	**2.006504**	**2.080939**	**1.949011**	**1.936409**	**1.923290**

Resultados con 16 procesos:

	200	400	800	1600	3200	6400
0	0.01727	0.126261	0.971801	7.690545	60.988637	488.062752
1	0.01719	0.1311	0.998278	7.648151	60.999493	487.484346
2	0.01532	0.130142	0.979008	7.673797	60.985677	487.573178
3	0.017023	0.125544	0.988946	7.684485	61.02615	488.070394
4	0.018696	0.126904	0.973623	7.659532	60.959537	487.957731
5	0.015577	0.120722	0.986362	7.672482	61.004189	488.049233
6	0.015419	0.126499	0.96411	7.695701	61.473111	488.657015
7	0.015483	0.121655	0.966713	7.875577	61.084784	488.054442
8	0.016332	0.128976	0.972978	7.710117	61.0636	488.066117
9	0.015478	0.132657	0.968135	7.692598	61.195068	488.051643
Promedio	**0.016379**	**0.127046**	**0.976995**	**7.700299**	**61.078025**	**488.002685**
Speedup	**1.995140**	**1.956960**	**2.039639**	**1.953416**	**1.935333**	**1.921128**

Resultados con 32 procesos:

	200	400	800	1600	3200	6400
0	0.017024	0.122015	0.955663	7.65967	59.542142	488.995281
1	0.01649	0.123573	0.959609	7.676057	60.117397	488.507998
2	0.016385	0.122021	0.953288	7.678766	60.357528	488.417907
3	0.018241	0.122386	0.970357	7.672554	60.538613	488.100296
4	0.016068	0.123416	0.957169	7.666693	60.657588	488.166691
5	0.016708	0.124052	0.96252	7.66891	60.806709	488.738183
6	0.016218	0.124207	0.953317	7.673763	60.750145	488.795104
7	0.016099	0.129285	0.96061	7.678998	60.826942	488.193252
8	0.016992	0.129516	0.961244	7.693188	60.905119	488.280799
9	0.017314	0.124542	0.952675	7.672046	60.90148	488.203054
Promedio	**0.016754**	**0.124501**	**0.958645**	**7.674065**	**60.540366**	**488.439857**
Speedup	**1.950471**	**1.996958**	**2.078682**	**1.960093**	**1.952521**	**1.919409**

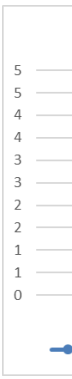
Resultados con 64 procesos:

	200	400	800	1600	3200	6400
0	0.019367	0.129568	0.959227	7.681175	61.444052	489.324041
1	0.017122	0.124805	0.960043	7.67917	60.994443	489.242453
2	0.017072	0.124543	0.961405	7.678719	60.969348	489.587798
3	0.016903	0.123095	0.958211	7.72202	60.986802	489.667331
4	0.017256	0.124573	0.985155	7.676213	61.028126	489.312678
5	0.017144	0.128169	0.968562	7.663276	61.131575	489.553411
6	0.016989	0.123438	0.961622	7.664649	61.042479	489.462123
7	0.016898	0.123439	0.95998	7.671043	61.079374	489.358974
8	0.017003	0.122692	0.962855	7.664939	61.005714	489.611022
9	0.017127	0.123394	0.958936	7.697209	61.035522	489.284556
Promedio	**0.017288**	**0.124772**	**0.963600**	**7.679841**	**61.071744**	**489.440439**
Speedup	**1.890202**	**1.992632**	**2.067994**	**1.958619**	**1.935532**	**1.915485**

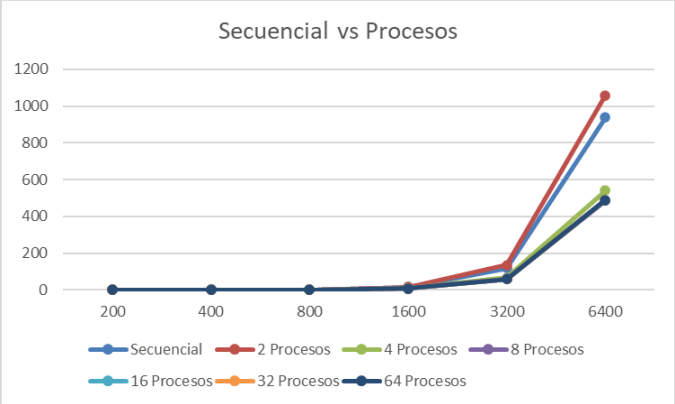
5. Gráficos:

Ejecución secuencial vs hilos:

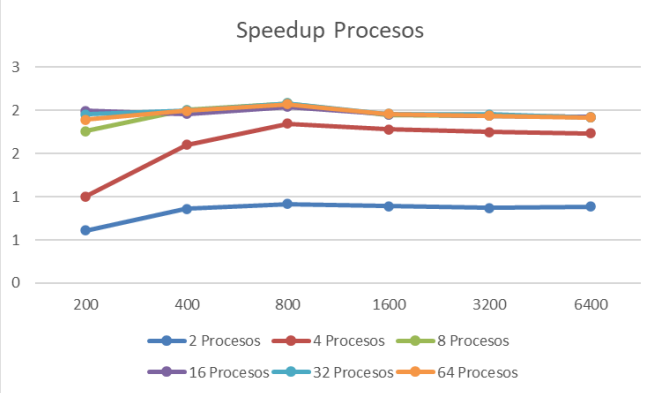
Speedup de hilos en función de las dimensiones de la matriz:



Ejecución secuencial vs procesos:



Speedup de procesos en función de las dimensiones de la matriz:



IV. CONCLUSIONES

La multiplicación de matrices es una operación computacionalmente intensiva. Su naturaleza cuadrática implica que a medida que aumenta el tamaño de las matrices, el tiempo requerido para completar la multiplicación crece de manera exponencial. Esto la convierte en un excelente candidato para la paralelización, buscando dividir el trabajo entre múltiples hilos o procesos para acelerar la operación.

- Secuencial: La versión secuencial muestra un incremento en el tiempo de ejecución conforme crece el tamaño de las matrices. Sin beneficios de paralelización, la eficiencia se mantiene constante a medida que el tamaño de las matrices aumenta.

- Paralelización con hilos: Al introducir la paralelización mediante hilos, se observa una mejora significativa en los tiempos de ejecución. Sin embargo, hay un límite en los beneficios de añadir más hilos. Al alcanzar 8 hilos, que coincide con el total de hilos disponibles en el procesador (4 núcleos con 2 hilos cada uno), el tiempo de ejecución se estabiliza. Más allá de este punto, agregar más hilos no aporta beneficios significativos y puede introducir overheads no deseados.

- Paralelización con procesos: Aunque los procesos son más pesados que los hilos y tienen un overhead asociado, la paralelización mediante procesos también muestra beneficios en términos de tiempo de ejecución. Además, al igual que con los hilos, al llegar a 8 procesos, que es el número óptimo dado el hardware disponible, el rendimiento tiende a estabilizarse.

REFERENCIAS

[1]. LLNL HPC Tutorials, POSIX Threads Programming. Disponible: https://hpc-tutorials.llnl.gov/posix/

[2]. StackOverflow, Cómo funciona la función fork() Disponible: https://es.stackoverflow.com/questions/179414/como-funciona-la-funci%C3%B3n-fork

[3]. Procesos e hilos en C: https://www.um.es/earlyadopters/actividades/a3/PCD_Activity3_Session1.pdf