Implementación de multiplicación de matrices con concurrencia utilizando procesos e hilos

Presentado por: Juan Camilo Rojas Cortés

Explicación del algoritmo:

El algoritmo al que era necesario aplicarle concurrencia es el de multiplicación de matrices. Este proceso consiste en generar una matriz resultante en la que cada celda sea el producto punto de la fila de su subíndice en la primera matriz y la columna de su subíndice en la segunda matriz. Para generar este producto punto es necesario recorrer completamente una fila y una columna de la matriz para hacer la suma y la multiplicación, repitiéndose este proceso para cada una de las celdas de la matriz resultante. Por ese motivo, este es un algoritmo con una complejidad de n^3, siendo n las dimensiones de la matriz en el caso de que se trabaje con matrices cuadradas. Por ejemplo, si se multiplican 2 matrices de 5X5, el algoritmo secuencial deberá hacer 125 iteraciones, aumentando cúbicamente el número de estas conforme se incrementan las dimensiones de la matriz.

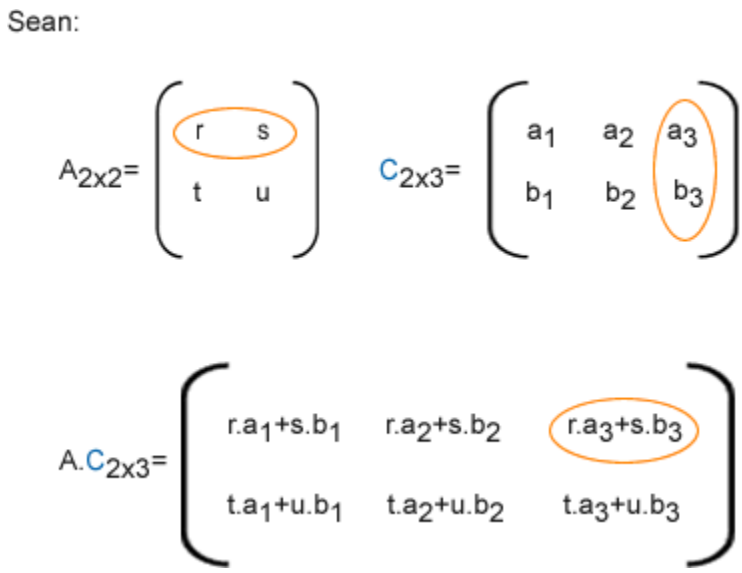

Figura 1. Representación gráfica de la multiplicación de matrices

Algoritmo paralelo:

Para el presente trabajo, se optó por hacer una paralelización por procesos. En este caso, con el objetivo de disminuir la porción de código que se ejecuta de forma secuencial, se le pasa como parámetro a cada proceso un número de fila. De esa forma, de manera secuencial solamente se realizan n iteraciones, siendo n el número de filas y columnas de la matriz. Así, a cada proceso le corresponde hacer n^2 iteraciones, pues cada uno de estos se encarga de calcular los resultados para una fila de la matriz. En esta implementación, el número de procesos tiene relación directa con las dimensiones de la matriz. En una matriz de NXN, el algoritmo generará N procesos.

Además, en la pasada entrega se realizó la paralelización de este algoritmo con hilos. Por lo tanto, en el presente trabajo se compara el rendimiento de ambas técnicas, teniendo en cuenta que algorítmicamente son equivalentes, pero que en cuanto a recursos de hardware funcionan distinto.

Análisis de tiempos y speed up:

Las pruebas realizadas a la paralelización del presente algoritmo se hicieron en una máquina con las siguientes características:

- Memoria RAM: 4 GB.
- Procesador: AMD A8-7410 APU with AMD Radeon R5 Graphics × 4
- Nucleos de procesamiento: 4
- Sistema operativo: ubuntu 16.04 LTS de 64 bits.
- Rendimiento: 27.712,00 Gflop/s (rendimiento obtenido a través de la herramienta System Profiler and Benchmark con el algoritmo de las N Reinas).

Para las pruebas de ejecución, se tomaron 7 valores de n distintos (10, 100, 200, 500, 800, 1000 y 2000) y para cada uno de estos se hizo un total de 10 pruebas, evaluando los tiempos de ejecución en cada uno de los algoritmos, tanto serial como parlelos. A continuación se pueden ver estos resultados:

IMPLEMENTACIÓN SECUENCIAL (tiempo dado en segundos)							
Iteración	**10**	**100**	**200**	**500**	**800**	**1000**	**2000**
1	0,00001	0,008707	0,066717	1,696583	11,025918	23,05065	198,588116
2	0,000022	0,011321	0,063909	1,777635	11,217768	23,003422	198,731774
3	0,000011	0,007936	0,063588	1,55296	10,97958	22,790453	197,164891
4	0,000022	0,010614	0,063801	1,529325	11,007149	22,826133	196,296044
5	0,000021	0,008507	0,064053	1,914466	11,095712	22,893029	196,564942
6	0,000022	0,006972	0,059075	1,454648	10,027709	21,131893	185,382501
7	0,000022	0,009318	0,059398	1,553169	10,16939	21,132597	184,00702
8	0,000009	0,011053	0,061804	1,539698	10,355836	21,554958	185,116232
9	0,00001	0,007345	0,058828	1,5348	10,662544	21,563793	182,589572
10	0,000009	0,011026	0,059724	1,55803	10,313104	21,778125	184,630674
Promedio	0,0000158	0,0092799	0,0620897	1,6111314	10,685471	22,1725053	190,9071766

Tabla 1. Mediciones de tiempos para la implementación secuencial

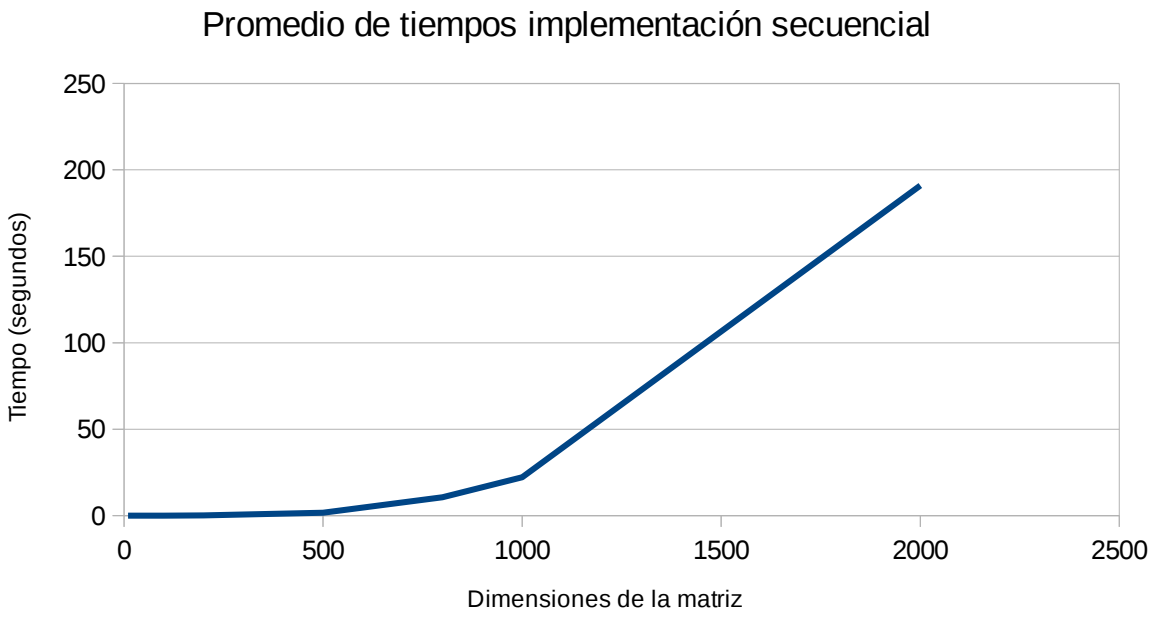

Gráfico 1. Promedio de tiempos para la implementaación secuencial

IMPLEMENTACIÓN CON PROCESOS (tiempo dado en segundos)							
Iteración	**10**	**100**	**200**	**500**	**800**	**1000**	**2000**
1	0,000473	0,008779	0,039613	0,819163	6,076385	11,233157	95,64709
2	0,000601	0,008754	0,04905	0,917474	5,769686	11,32583	95,422326
3	0,000417	0,008732	0,044463	0,9229	5,276212	11,073209	101,18951
4	0,000523	0,009856	0,053547	0,940168	5,596793	11,179473	96,268104
5	0,000396	0,008316	0,047111	0,926069	5,324141	10,960025	95,487048
6	0,000391	0,00886	0,045155	0,921572	5,303858	10,806473	96,245024
7	0,000403	0,009082	0,046421	0,932895	5,358065	10,696624	95,009629
8	0,000415	0,00858	0,045466	0,938342	5,292186	10,636114	95,201766
9	0,000461	0,009446	0,04688	0,929484	5,692629	11,575546	100,170502
10	0,000409	0,009482	0,044062	0,932934	5,324141	11,32583	95,64709
Promedio	0,0004489	0,0089887	0,0461768	0,9181001	5,5014096	11,0812281	96,6288089

Tabla 2. Mediciones de tiempos para implementación con procesos

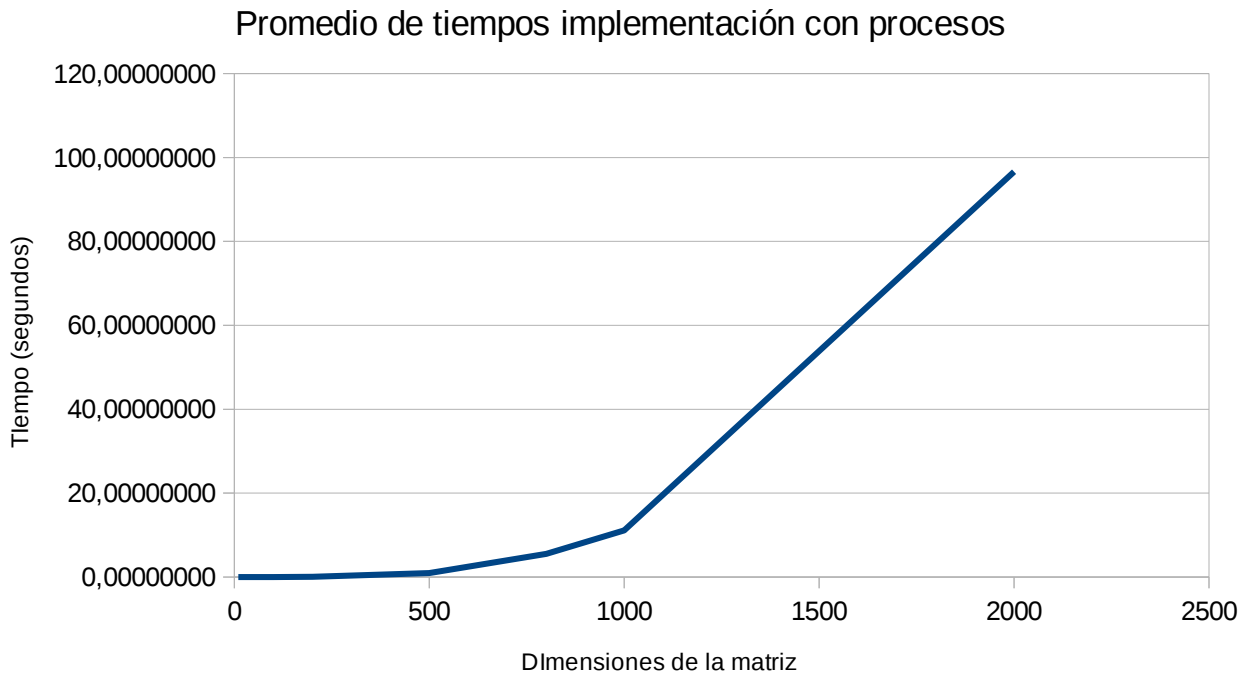

Gráfico 2. Promedio de tiempos para la implementación con procesos

Para tener una mayor claridad de los resultados comparativos entre ambas implementaciones, se realizó el cálculo del speed up rate. A continuación se pueden ver los resultados:

Tabla 3. Speed up rate

Dimensiones	Promedio implementación secuencial (segundos)	Promedio implementación con procesos (segundos)	Speed up rate implementación con procesos
10	0,0000158	0,00044890	-0,9648028514
100	0,0092799	0,0089887	0,0323962308
200	0,0620897	0,0461768	0,3446081149
500	1,6111314	0,9181001	0,7548537463
800	10,685471	5,5014096	0,9423151114
1000	22,1725053	11,0812281	1,000906858
2000	190,9071766	96,6288089	0,9756755648

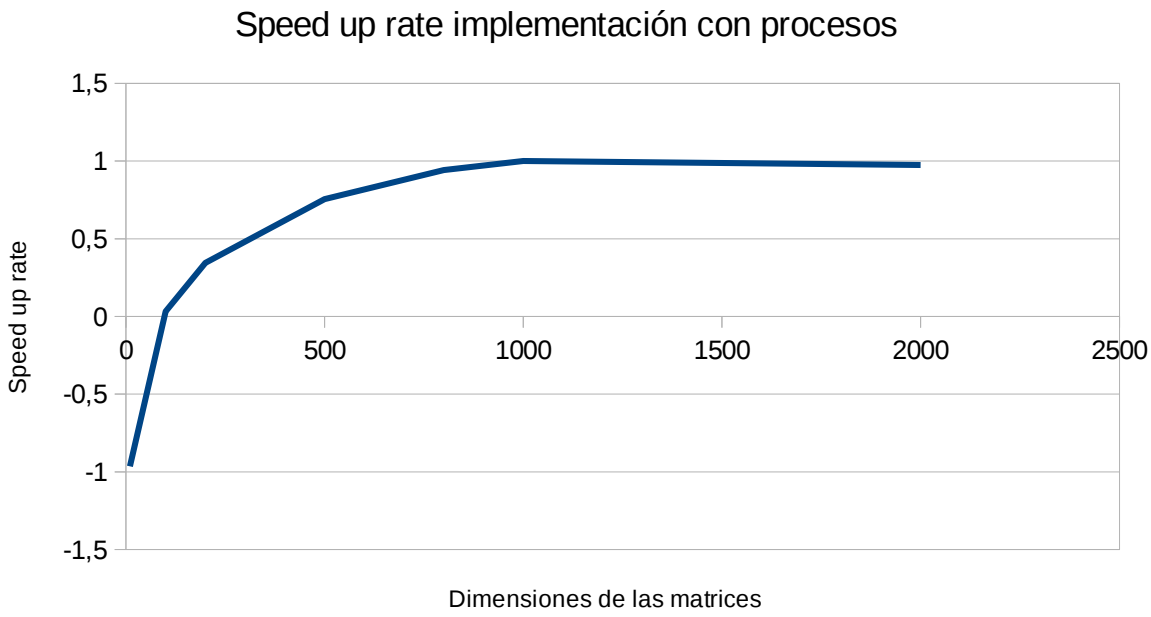

Gráfico 3. Speed up rate

Como es posible observar en los datos presentados anteriormente, cuando se trabaja con conjuntos de datos grandes y se crean procesos según el número de filas de las matrices que se multiplican, el tiempo de ejecución puede disminuír incluso en valores cercanos al 100%.

Al haberse hecho previamente una implementación de concurrencia con hilos para el mismo problema, se comparó el desempeño de las dos implementaciones (hilos y procesos). Los resultados pueden verse en la siguiente gráfica:

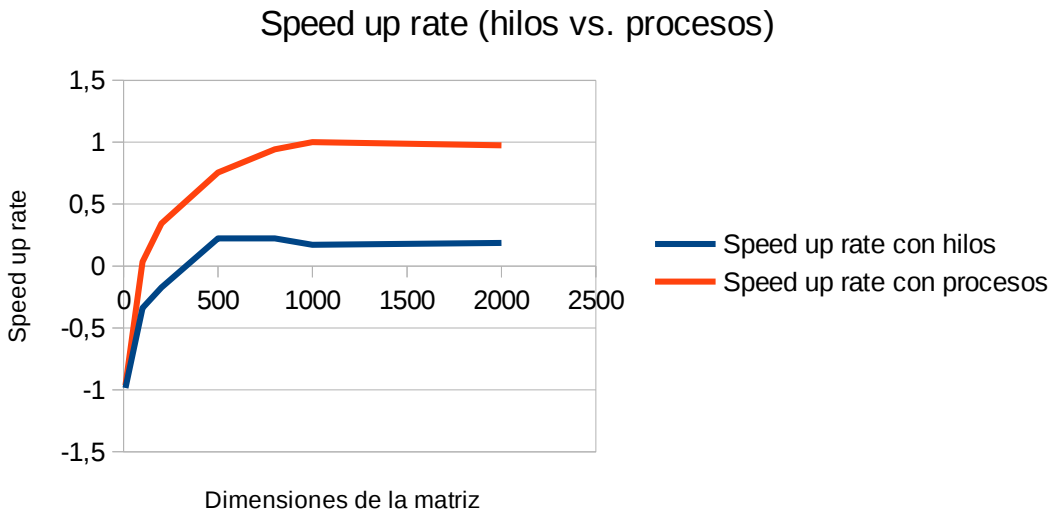

Gráfico 4. Comparación de rendimiento entre implementación con hilos y con procesos

Conclusiones:

- Para valores de N (siendo N el número de filas y columnas de las matrices) pequeños, la paralelización, tanto con procesos como con hilos, tiende a tener un rendimiento inferior a la implementación secuencial. Esto ocurre porque termina siendo mayor el desgaste de la creación de hilos o procesos que la ejecución del algoritmo por sí mismo. Sin embargo, al incrementar el valor de N, la paralelización termina incrementando notablemente el desempeño de ejecución. Por ejemplo, el speed up rate de la paralelización por procesos llegó a valores cercanos a 1, lo que quiere decir que esta ejecución puede llegar a incrementar su rendimiento en un 100% respecto a la implementación secuencial.
- Tanto en paralelización con hilos como con procesos, el speed up rate llega a un valor máximo y se estabiliza allí. No es un crecimiento lineal con las dimensiones del dataset.
- Como es posible observar en el Gráfico 4, el rendimiento alcanzado con la paralelización con procesos es bastante superior al alcanzado con la paralelización con hilos. Mientras el speed up rate máximo de la paralelización con hilos es cercano a 0.2, con procesos este valor llega a 1,0. Esto puede deberse a que el procesador en el que se realizaron las pruebas solamente cuenta con 4 hilos, por lo que su rendimiento no será óptimo cuando se crean, por ejemplo, 2000 hilos. En estos casos la paralelización por procesos tiene ventaja, pues estos son administrados por el sistema operativo y no tienen una dependencia marcada respecto a las características del hardware, como es el caso de los hilos.