PRÁCTICA 3 – Memoria Caché

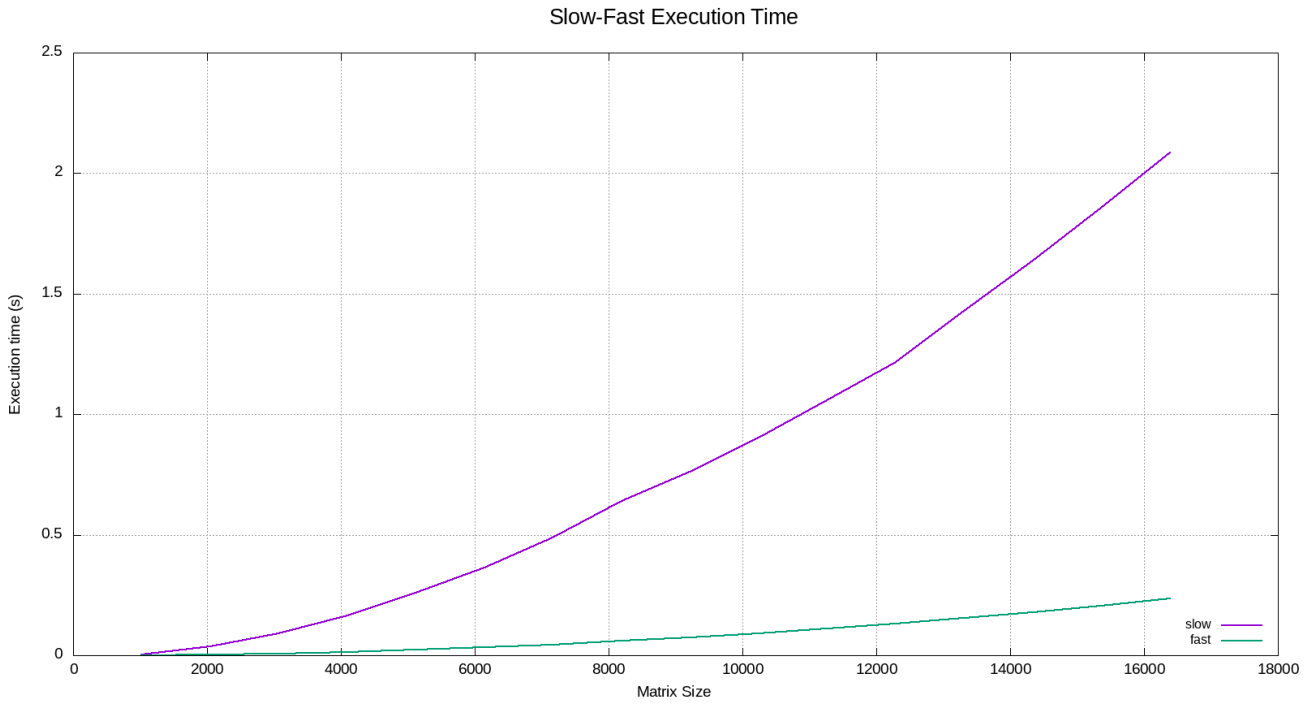
Ejercicio 0 – Ejecución en la máquina virtual de la asignatura

Mediante los comandos getconf –a | grep –i cache, cat /proc/cpuinfo y lstopo podemos observar que distinguimos tres niveles de caché. El primer nivel de caché encontramos separación entre caché de datos e instrucciones, ambas presentan las siguientes características: tamaño 32768 B =32 KB, asociatividad con 8 vías y tamaño de bloque o línea de 64 B. El segundo nivel de caché está unificado y presenta las siguientes características: tamaño 262144 B = 256 KB, asociatividad de 8 vías y tamaño de bloque o línea de 64 B. Por última, el tercer nivel de caché también es unificado con tamaño 8388608 B = 8 MB, asociatividad de 16 vías y tamaño de bloque o línea de 64 B. También podemos observar que el microprocesador tiene dos cores.

Ejercicio 1

En nuestro caso hemos ejecutado 16 veces, una vez por cada tamaño de 1024 a 16384, los dos programas slow y fast intercalados entre sí. Los programas son necesarios ejecutarlos intercalados entre sí variando tamaño de la matriz y ejecución de slow y fast. Esto es debido a que si ejecutamos consecutivamente programas con el mismo tamaño de matriz, es posible que tras la primera ejecución del programa algunos datos o instrucciones necesarios para la ejecución del código se hayan copiado a la caché, por lo que el tiempo de ejecución será menor que la ejecución estándar del programa y no sería representativo respecto del rendimiento real de la ejecución. Para asegurarnos de que la caché de datos e instrucciones no almacena los mismos valores alternamos la ejecución de los programas slow y fast.

Gráfica que muestra el tiempo de ejecución de slow y fast en función del tamaño de matriz.

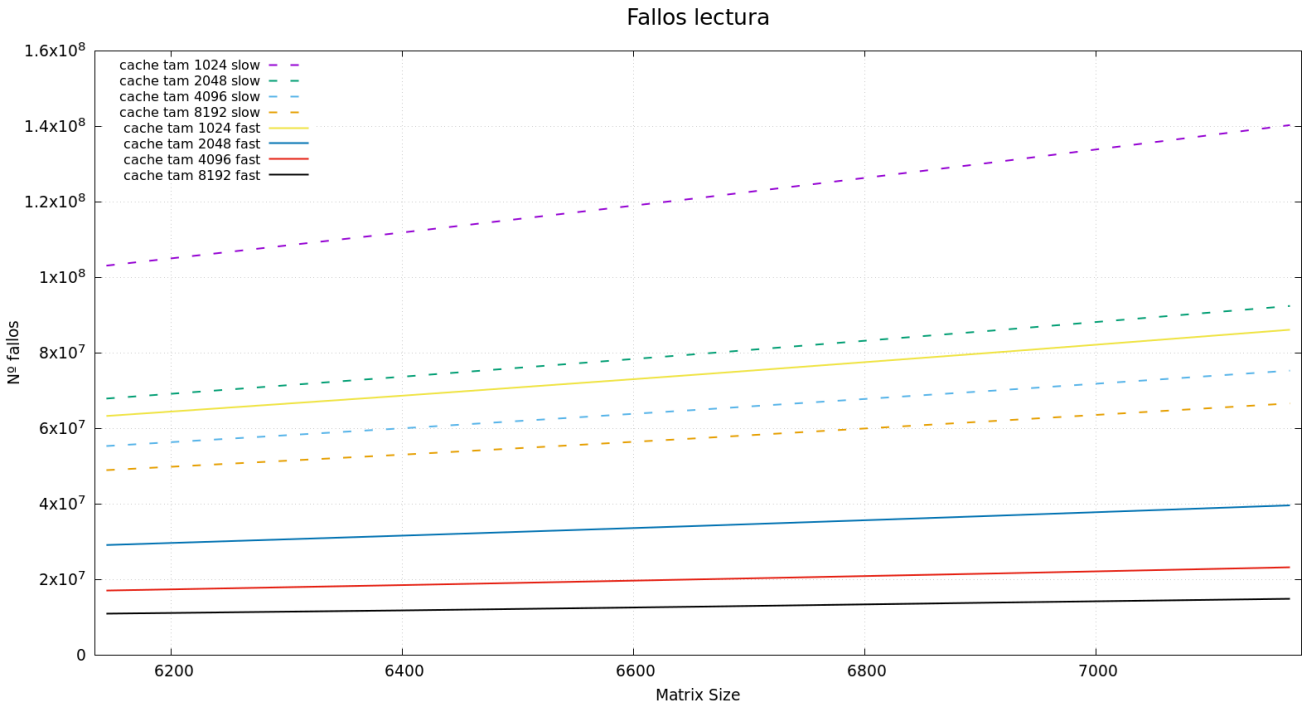


Hemos usado 10 repeticiones para cada tamaño de matriz en los programas slow y fast para evitar anomalías y picos asociados a diferentes configuraciones de la caché de datos e instrucciones que hagan variar el tiempo de ejecución. Podemos observar que la disposición de los bloques de datos en la caché tiene una gran relevancia a la hora de establecer el tiempo de ejecución. De esta forma, la ejecución de slow tiene un gran número de fallos de caché ya que los datos en un mismo bloque no se distribuyen en columnas y sí en filas. Por el contrario, fast realiza la suma por filas, aprovechando el bloque de datos cargado en caché para acceder a datos sin tener tantos fallos de caché. Al tratarse de tamaños de matrices grandes, no caben todas las páginas necesarias para obtener los datos de las matrices en caché, por lo que se producen fallos de caché inevitables a causa del tamaño reducido de la caché. De esta forma comprobamos la importancia de una tener en cuenta la localidad espacial y la disposición de los datos a la hora de ejecutar un programa.

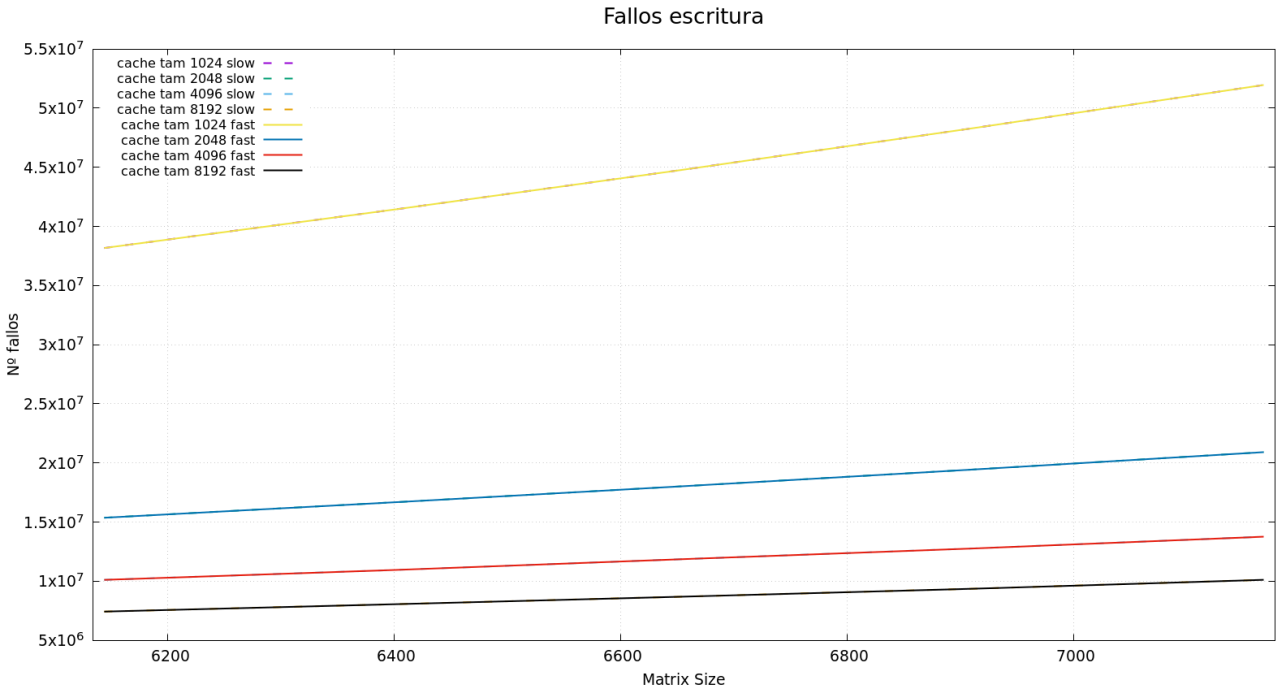
Para matrices de tamaño pequeño apenas hay diferencia en los tiempo de ejecución debido a que se necesitan pocos bloques para acceder a todos los datos en caché, de forma que todas pueden estar en caché al mismo tiempo, y el número de fallos de caché es prácticamente el mismo accediendo en filas que por columnas.

Ejercicio 2

Gráfica del número de fallos de lectura en función del tamaño de la matriz dependiendo del tamaño de la caché, en ejecución de programas slow y fast.



Gráfica del número de fallos de escritura en función del tamaño de la matriz dependiendo del tamaño de la caché, en ejecución de programas slow y fast.

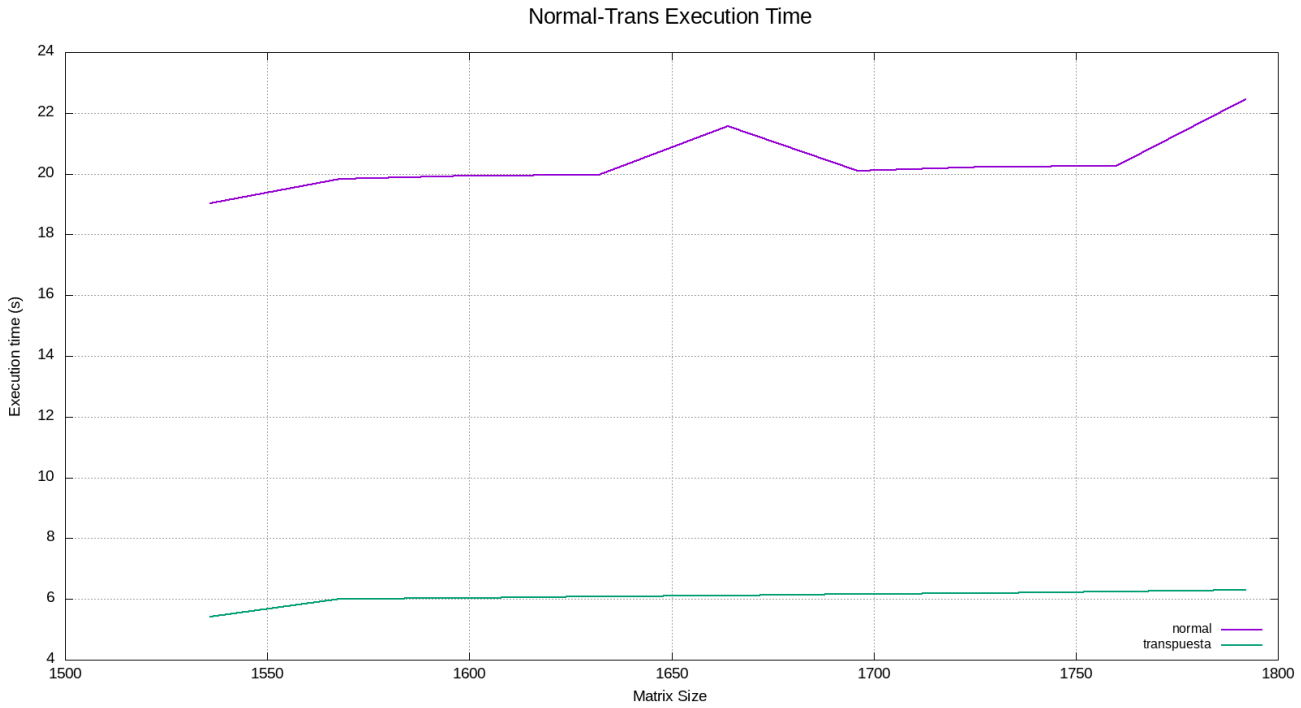


En ambos programas slow y fast reduciendo el tamaño de la caché aumentamos el número de fallos tanto en lectura como en escritura, y también aumentan los fallos conforme aumentamos el tamaño de la matriz. Podemos observar que los fallos de lectura en los la ejecución de programas slow son considerablemente de mayor número que en los fast. Sin embargo, en los fallos de escritura prácticamente no hay diferencia en la ejecución de ambos programas (el número de fallos es muy parecido).

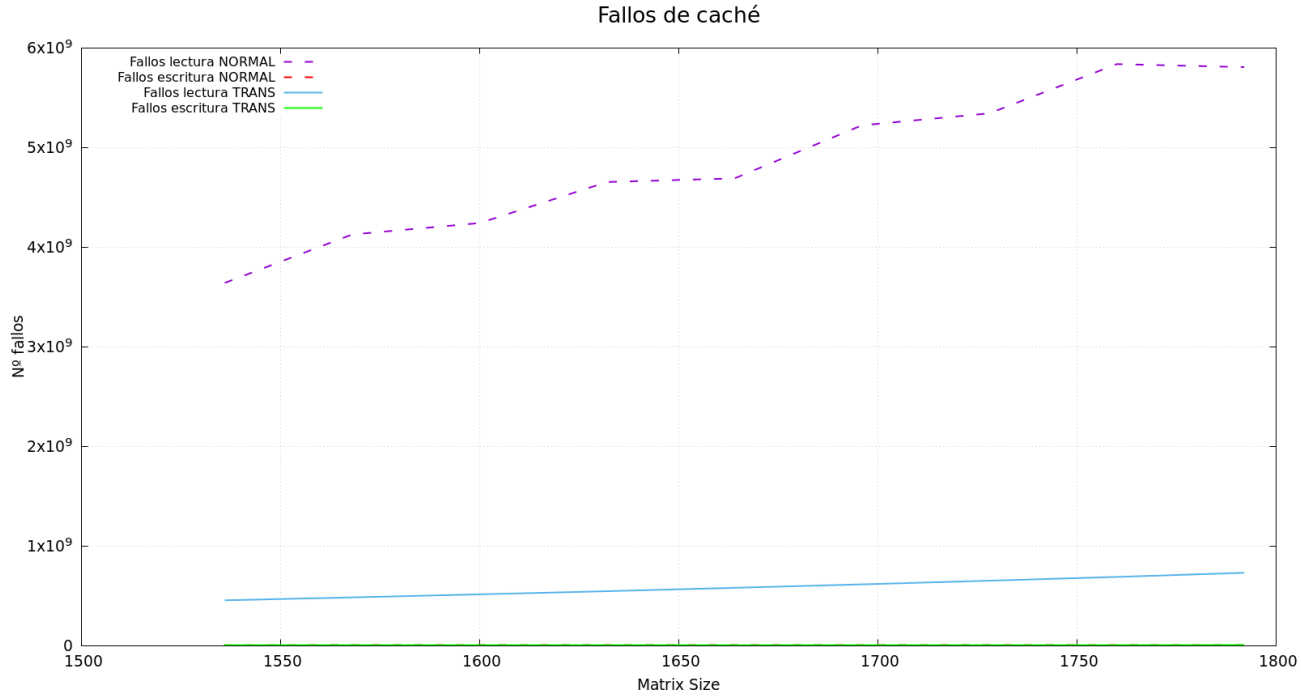
Al aumentar el tamaño de caché es posible almacenar más direcciones reales con bloques de datos asociados en caché, por lo que se producirá un menor número de fallos de caché, y consecuentemente disminuirá el tiempo de ejecución. Si mantenemos fijo un tamaño de caché y aumentamos el tamaño de la matriz, aumentará el tiempo de ejecución a la hora de ejecutar el programa linealmente al tamaño de matriz. Un tamaño de matriz mayor supondrá que esté formada por un mayor número de datos y por tanto necesitará cargar más bloques en caché a la hora de sumar sus componentes en los programas slow y fast. Cuanto mayor sea el tamaño de la matriz sin variar el tamaño de la caché, menos bloques y por tanto datos podrá almacenar en caché, derivando en un mayor número de fallos de caché que ralentizan la ejecución del programa.

Ejercicio 3

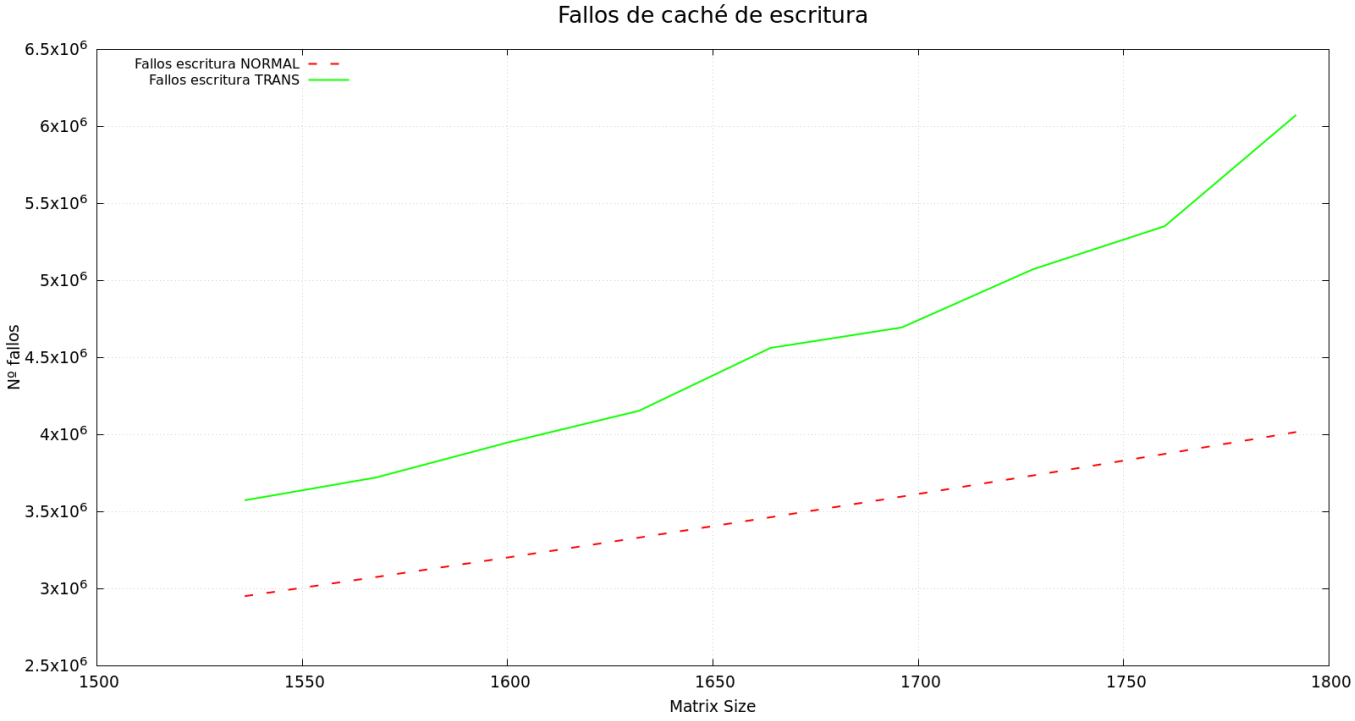
Gráfico del tiempo de ejecución de las multiplicaciones normal y traspuesta en función del tamaño de la matriz.



Gráfica de los fallos de caché de las multiplicaciones normal y traspuesta en función del tamaño de la matriz.



Gráfica de fallos de caché de escritura de las multiplicaciones normal y traspuesta en función del tamaño de la matriz; se trata de las dos funciones de menor valor de la gráfica anterior, que parecen ser constantes de valor 0, dada la diferencia de magnitud entre los fallos de lectura y escritura.



A la vista de los resultados, podemos determinar que la multiplicación normal es considerablemente más lenta que la traspuesta y que presenta un mayor número de fallos de caché. No obstante, los fallos de caché de escritura son mayores en la multiplicación traspuesta.

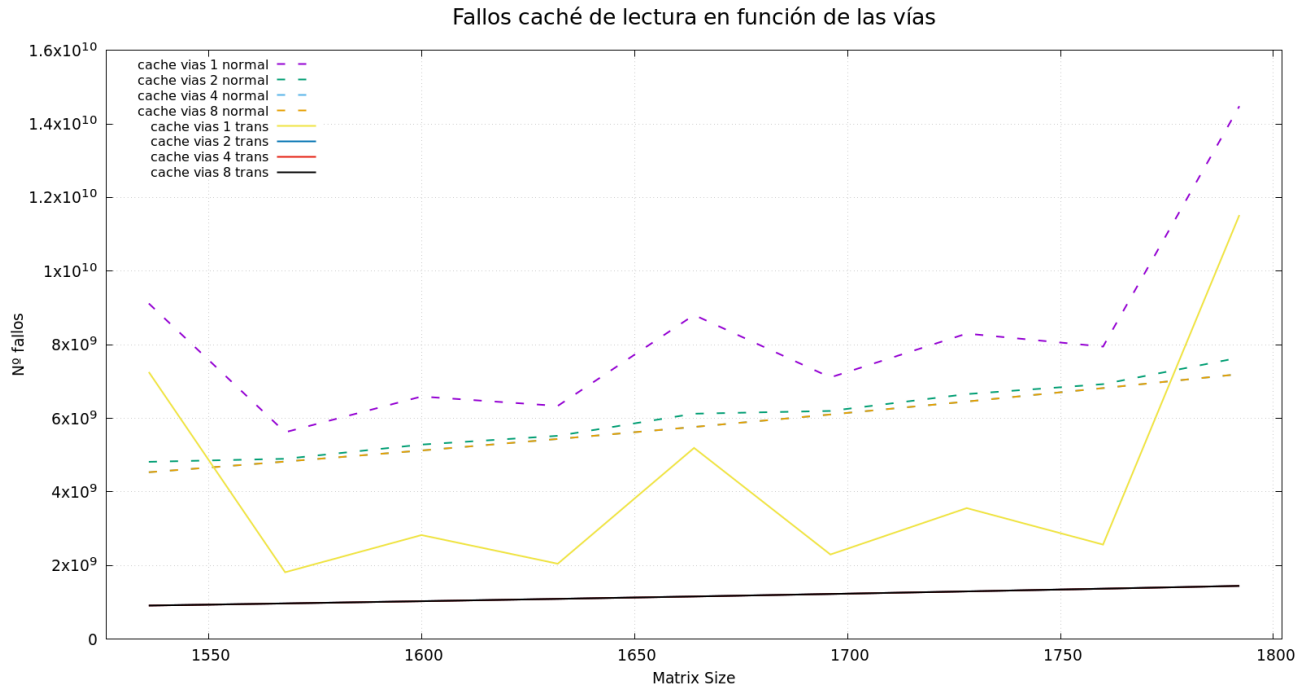
Al trasponer la matriz, establecemos que los datos de posiciones consecutivas se sitúen en una misma fila de la matriz. De esta manera, al recorrer la fila para la multiplicación muchos de los datos se encuentran en el mismo bloque, por lo que no supondrán tantos fallos de caché como leerlos por columnas. Esto reduce el tiempo de ejecución y el número de fallos de caché.

Al aumentar el tamaño de las matrices, aumenta el número de datos a multiplicar. Como mantenemos constante el tamaño de la caché, al aumentar el número de bloques necesarios para realizar la multiplicación, se producen más fallos de caché, que harán que aumente el tiempo de ejecución del programa.

Ejercicio 4 - opcional

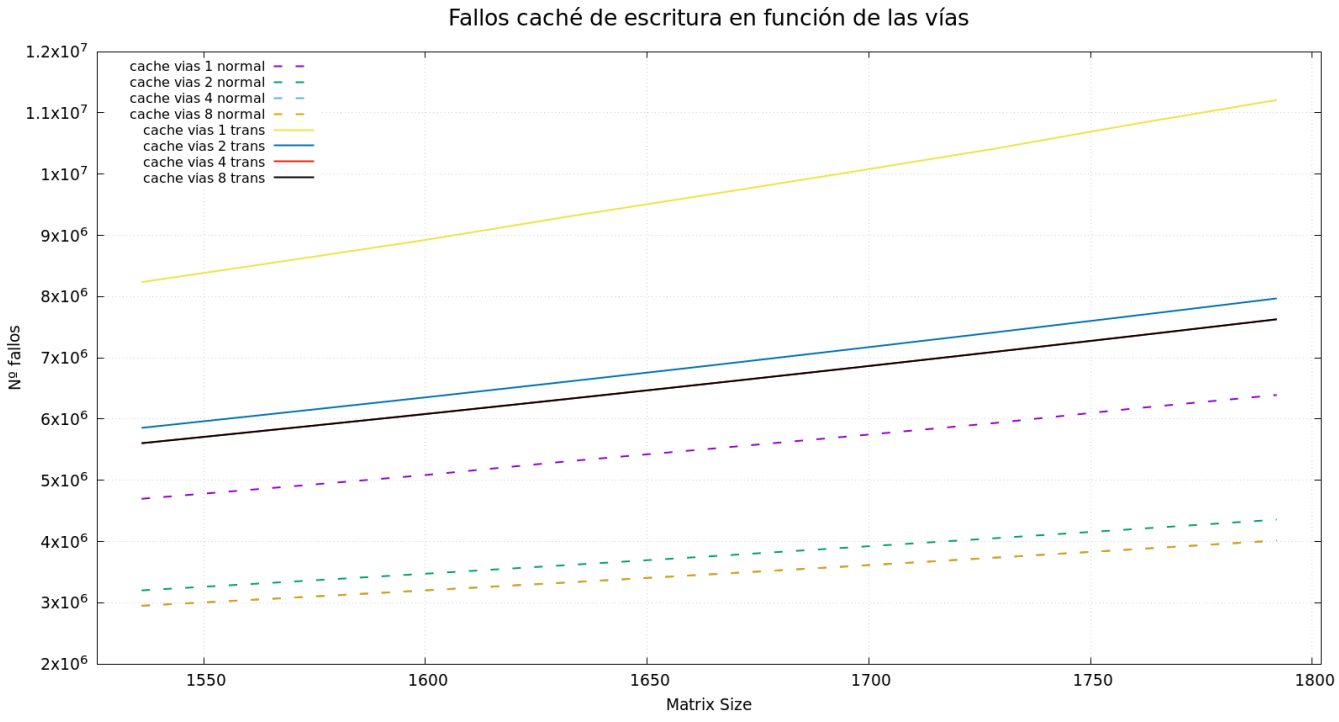
Para el ejercicio 4, hemos decidido tomar medidas de los fallos de escritura y lectura de caché para diferentes valores de vía (1, 2, 4, 8) fijando un tamaño de caché de 2048 B y el resto de los valores iguales a los del ejercicio 2. Asimismo, hemos tomado medidas de los fallos variando el tamaño de palabra de caché (32, 64, 128, 256). Cabe destacar que algunos resultados son incongruentes, ya que la gráfica no define una función clara y lógica. Más concretamente, *los fallos de lectura en función del tamaño de palabra* definen funciones no monótonas y poco coherentes. A este respecto, simplemente hemos podido concluir un rango en el que se encontraría la verdadera función que parametrizase el fenómeno estudiado, como se explica más adelante.

Gráfica de fallos de caché de lectura en función del número de vías en relación con el tamaño de la matriz



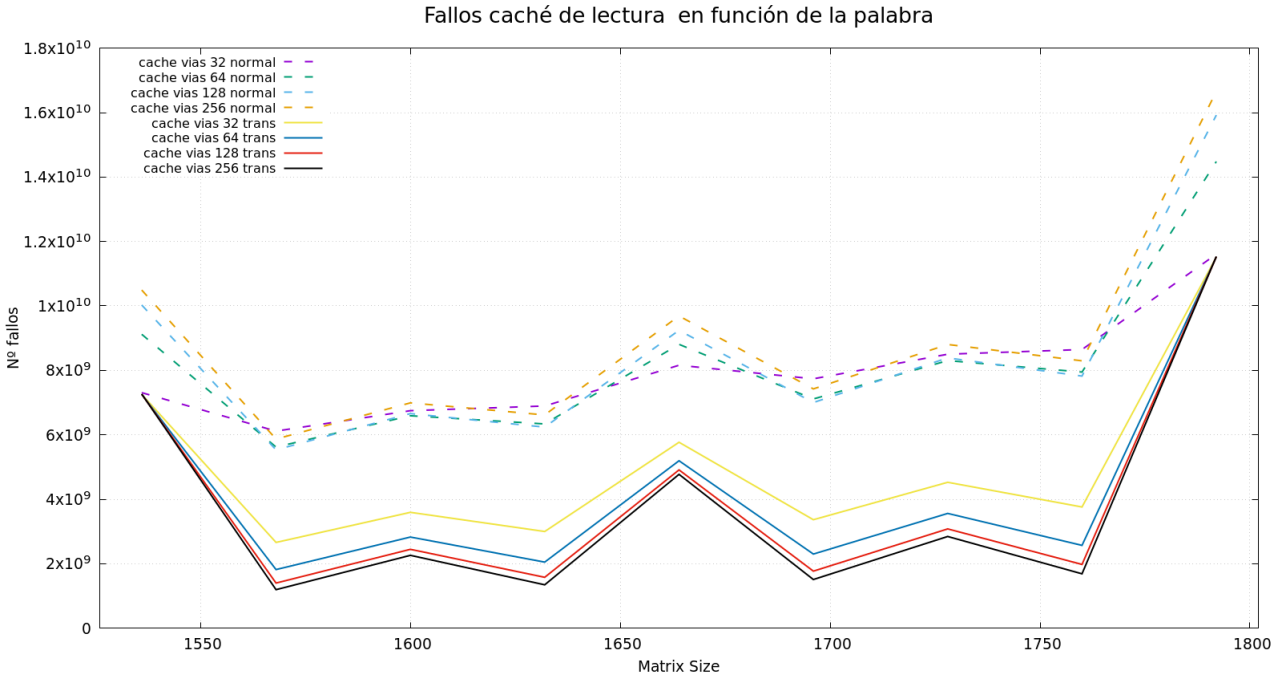
Los fallos de lectura disminuyen conforme aumentamos el número de vías de asociatividad, y manteniéndose en valores inferiores las ejecuciones de traspuesta respecto de la multiplicación normal. En las ejecuciones de multiplicaciones normal y traspuesta con 1 vía no podemos determinar una clara relación entre el número de fallos y es tamaño de la matriz, pero podemos asegurar que se encuentran en un rango mayor que con un número más elevados de vías de asociatividad. En el resto de casos se ve una clara relación en el aumento del tamaño de la matriz y aumento del número de fallos de lectura una vez fijado el número de vías. Cabe destacar que las gráficas se superponen en ambos programas para 4 y 8 vías.

Gráfica de fallos de caché de escritura en función del número de vías en relación con el tamaño de la matriz



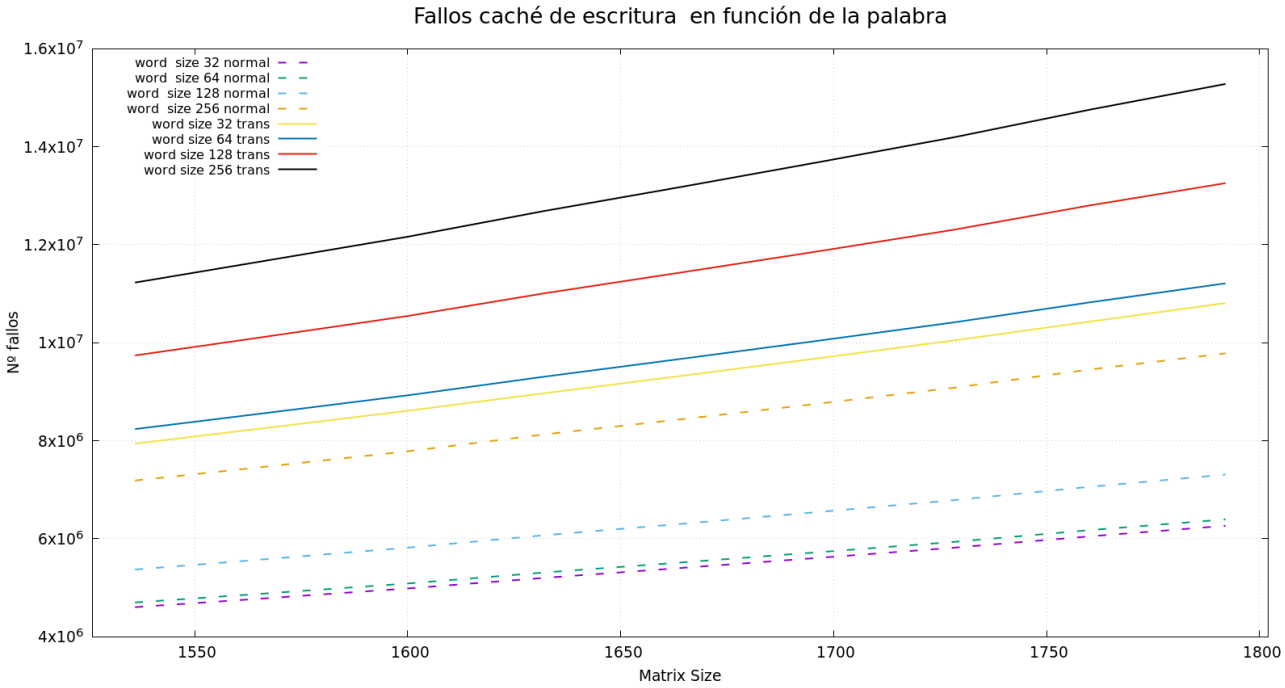
Como podemos observar, al aumentar la asociatividad, disminuye el número de fallos de caché de escritura en ambos programas. Al aumentar el tamaño de la matriz fijando un número de vías aumenta linealmente el número de fallos de escritura.

Gráfica de fallos de caché de lectura en función del tamaño de la palabra en relación con el tamaño de la matriz



En este caso solo podemos concluir que el número de fallos de lectura disminuye en cuanto a su rango al aumentar el tamaño de palabra. Al fijar un tamaño de palabra y aumentar el tamaño de la matriz los resultados no son congruentes por lo que no podemos establecer una clara relación entre el número de fallos y el tamaño de matriz para un tamaño de palabra fijo en ambos programas.

Gráfica de fallos de caché de lectura en función del tamaño de la palabra en relación con el tamaño de la matriz



Nuevamente podemos observar una disminución del número de fallos de escritura conforme aumentamos el tamaño de palabra. Al aumentar el tamaño de la matriz también aumenta el número de fallo de caché de escritura.