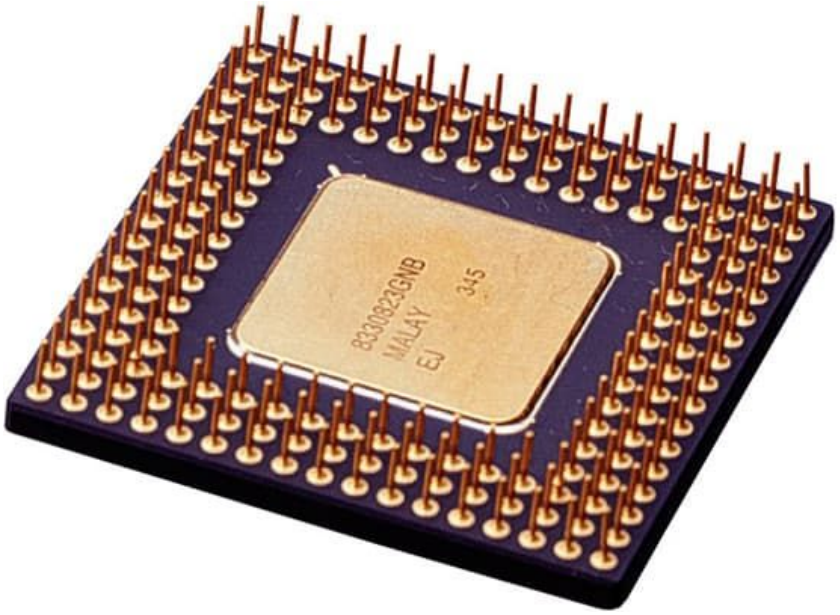
MEMORIA

Práctica 3: Memoria Caché y Rendimiento



Aitor Arnaiz del Val
Andrés Salas Peña

Ejercicio 0

INFORMACIÓN SOBRE LA CACHÉ

Indique en la memoria asociada a la práctica los datos relativos a las memorias caché presentes en los equipos del laboratorio. Se ha de resaltar en qué niveles de caché hay separación entre las cachés de datos e instrucciones, así como identificarlas con alguno de los tipos de memoria caché vistos en la teoría de la asignatura.

```
temporal@localhost:~$ getconf -a | grep -i cache
LEVEL1_ICACHE_SIZE                 32768
LEVEL1_ICACHE_ASSOC                8
LEVEL1_ICACHE_LINESIZE             64
LEVEL1_DCACHE_SIZE                 32768
LEVEL1_DCACHE_ASSOC                8
LEVEL1_DCACHE_LINESIZE             64
LEVEL2_CACHE_SIZE                  262144
LEVEL2_CACHE_ASSOC                 8
LEVEL2_CACHE_LINESIZE              64
LEVEL3_CACHE_SIZE                  3145728
LEVEL3_CACHE_ASSOC                 12
LEVEL3_CACHE_LINESIZE              64
LEVEL4_CACHE_SIZE                  0
LEVEL4_CACHE_ASSOC                 0
LEVEL4_CACHE_LINESIZE              0
```

Tras ejecutar los dos comandos posibles en los equipos del laboratorio, observamos que para responder al ejercicio nos es suficiente con el comando *getconf -a | grep -i cache.* No obstante, en la entrega de la práctica se adjunta una captura de pantalla de la salida de ambas ejecuciones.
Analizando los resultados de la imagen superior, podemos observar que los laboratorios montan tres cachés de diferentes niveles, por lo que siguen un sistema de cachés multinivel:

-Level 1: La caché de nivel 1 posee un tamaño de 65.536 Bytes. Este nivel se encuentra dividido en dos cachés (una para instrucciones y otra para datos). Ambas poseen el mismo tamaño de 32.768 Bytes con una asociatividad de 8 vías.

-Level 2: La caché de nivel 2 posee un tamaño de 262.144 Bytes con una asociatividad de 8 vías.

-Level 3: La caché de nivel 3 posee un tamaño de 3.145.728 Bytes con una asociatividad de 12 vías.

La explicación del porqué los sitemas de los laboratorios montan este sistema multinivel se ha visto en las clases de teoría y fundamentado en el libro “Estructura y Diseño de Computadores” de Patterson y Hennessy. Una vez explicados y expuestos los datos de la captura de pantalla procedemos a las causas que han llevado a esta implementación multinivel.
Por un lado, en relación a los tamaños, la caché de menor nivel posee el tamaño más pequeño porque su principal tarea es la de minimizar el tiempo de acierto para conseguir un

ciclo de reloj lo más pequeño posible, dejando más de lado la reducción de la frecuencia de fallos. Las cachés que se encuentran más orientadas a la reducción de la frecuencia de fallos son las de nivel 2 y 3, por tanto, estas son cada vez más grandes y se orientan a la reducción de la penalización por fallos (debida a los largos tiempos que se requieren para acceder a la memoria principal). También los objetivos de la primera caché y de las otras dos de niveles superiores, explican que la única caché que se divide en instrucciones y datos es la primera. Esta división minimiza el tiempo de acierto, las otras cachés no se dividen porque lo que buscan es reducir la frecuencia de fallos y esto se consigue mejor sin realizar la separación entre instrucciones y datos.

Por otro lado, en relación al número de vías de asociatividad, es frecuente encontrarse con un mayor número de vías de asociatividad en las cachés de niveles superiores porque su objetivo es reducir la frecuencia de fallos y el tiempo de acceso a la caché es menos crítico.

Ejercicio 1

MEMORIA CACHÉ Y RENDIMIENTO

Explica razonadamente por qué hay que realizar múltiples veces la toma de medidas de rendimiento para cada programa y tamaño de matriz.

La ejecución se repite un total de 20 repeticiones para que no se aprecien fuertes picos en la gráfica adjuntada.
Al repetirse múltiples veces, se consigue un valor medio experimental con un menor error con respecto a la media esperada, haciendo que el valor se aproxime mejor a la media teórica buscada.
Los tiempos de ejecución de los programas dependen del resto de operaciones que el ordenador esté llevando a cabo, por eso la ejecución se ha llevado a cabo en el ordenador sin haber abierto ningún programa más en el equipo aunque sigue siendo inevitable que el ordenador realice periódicamente ciertas operaciones a nivel sistema operativo que hacen que los tiempos de ejecución varíen.

Explica el método seguido para la obtención de datos y una justificación del efecto observado.

El método realizado para obtener los datos se puede ver claramente explicado en el script adjuntado en la práctica. Primero se ejecuta el script, alternando entre slow y fast, para cada uno de los valores del tamaño de la matriz en ambos programas, guardando en un fichero *Ejecución$j.txt* los tiempos resultantes de la ejecución slow y fast para cada tamaño de matriz. Este proceso se repite durante 20 veces para evitar picos en las gráficas,

guardándose cada vez en un fichero distinto. Tras esto, se concatenan todos los ficheros y se calculan las medias usando para ello una tabla asociativa de awk para los tiempos slow y otra para los tiempos fast. Por último, se realiza una unión de ambos ficheros resultantes de las tablas asociativas dando lugar al definitivo fichero *time_slow_fast.dat* . Para finalizar, solo queda pintar la gráfica pedida a partir de los datos de este último fichero, para ello se sigue el ejemplo de GNUPlot explicado en clase y dado como ejemplo en el material de la práctica

¿Por qué para matrices pequeñas los tiempos de ejecución de ambas versiones son similares, pero se separan según aumenta el tamaño de la matriz?

Para matrices pequeñas, el tiempo de ejecución entre dos programas que calculan las matrices no difiere mucho debido a que la operación para valores pequeños es muy rápida. Sin embargo, conforme la matriz va aumentando de tamaño, empieza a cobrar mayor importancia el cómo se realizan las operaciones puesto que empiezan a ser más costosas. Este es el motivo del porqué se empiezan a diferenciar más los tiempos de ejecución entre los programas slow y fast.

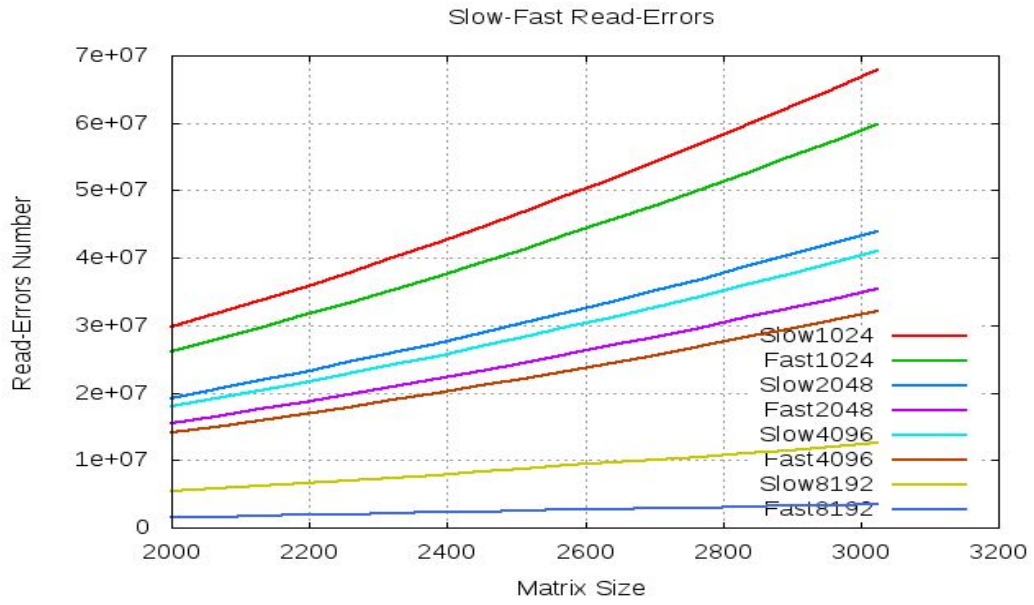
¿Cómo se guarda la matriz en memoria, por filas (los elementos de unas filas están consecutivos) o bien por columnas (los elementos de una columna son consecutivos)?

Tal y como se puede intuir, la matriz se guarda en memoria por filas. Haciendo que su guardado sea más rápido y su reconstrucción más sencilla.

Ejercicio 2

TAMAÑO DE LA CACHÉ Y RENDIMIENTO

En este caso, no ha sido necesario ejecutar varias repeticiones del programa ya que el número de fallos de lectura y escritura en las cachés es constante y no depende de la ejecución de otros programas.



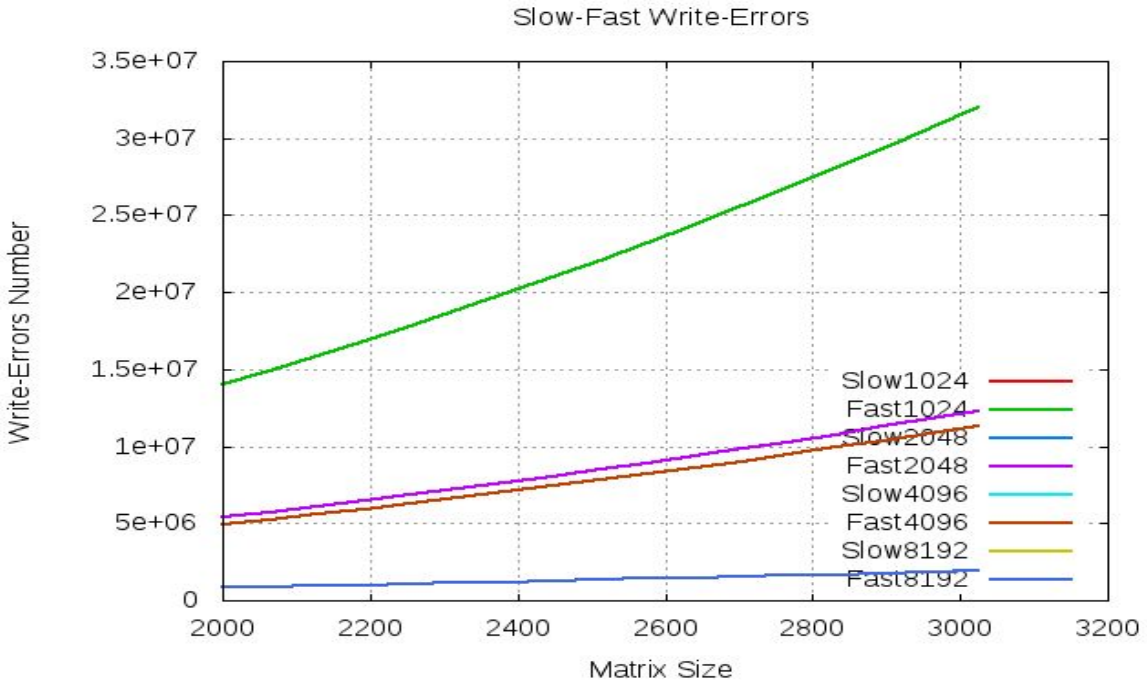
¿Se observan cambios de tendencia al variar los tamaños de caché para el programa slow? ¿Y para el programa fast?

Como se puede apreciar en la gráfica de errores de lectura y de escritura, se observa un cambio de tendencia tanto en los programas slow como en los programas fast, para cachés más pequeñas los errores siguen una función lineal con mayor pendiente mientras que para las cachés más grandes, ambos programas siguen una función lineal con una pendiente mucho menor.
La explicación de este fenómeno está en los distintos tamaños de la caché. Cuanto más grande es el tamaño de caché, las matrices que entran en las mismas pueden ser más grandes, luego los fallos de acceso serán mucho menores.

¿Varía la tendencia cuando se fija en un tamaño de caché concreto y compara el programa slow y fast? ¿A qué se debe cada uno de los efectos observados?

Comparando ahora los errores de lectura y de escritura para una misma caché según se ejecute slow o según se ejecute fast, se puede observar en la gráfica de fallos de escritura que no hay ninguna diferencia entre ambos. Sin embargo, en la gráfica de fallos de lectura sí que se observa una clara diferencia entre ambos programas. Tras ver esto, podemos decir que la diferencia de tiempo entre ambos programas y por tanto la eficiencia con la que se ejecuta uno y otro reside en la lectura de las matrices y no en su escritura.

La lectura de matrices de ambos programas es la siguiente: el programa slow realiza la lectura por columnas lo que hace que el programa vaya mucho más lento que el programa fast que lo hace primero por filas. Explicando esto, la diferencia de fallos de lectura tan elevada entre un programa y otro.

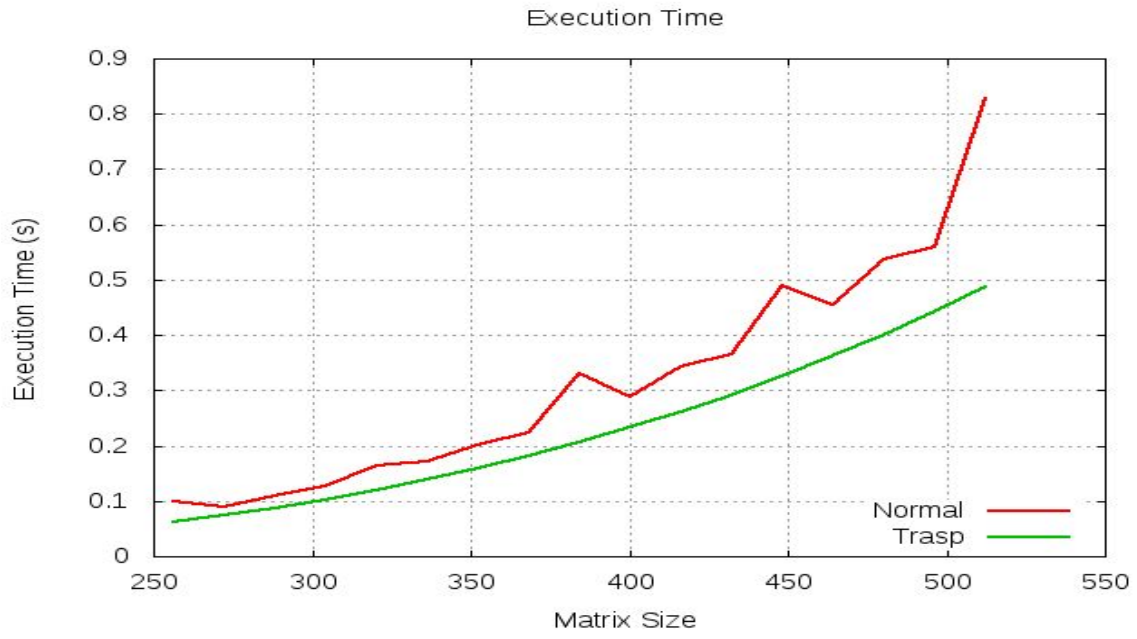


Ejercicio 3

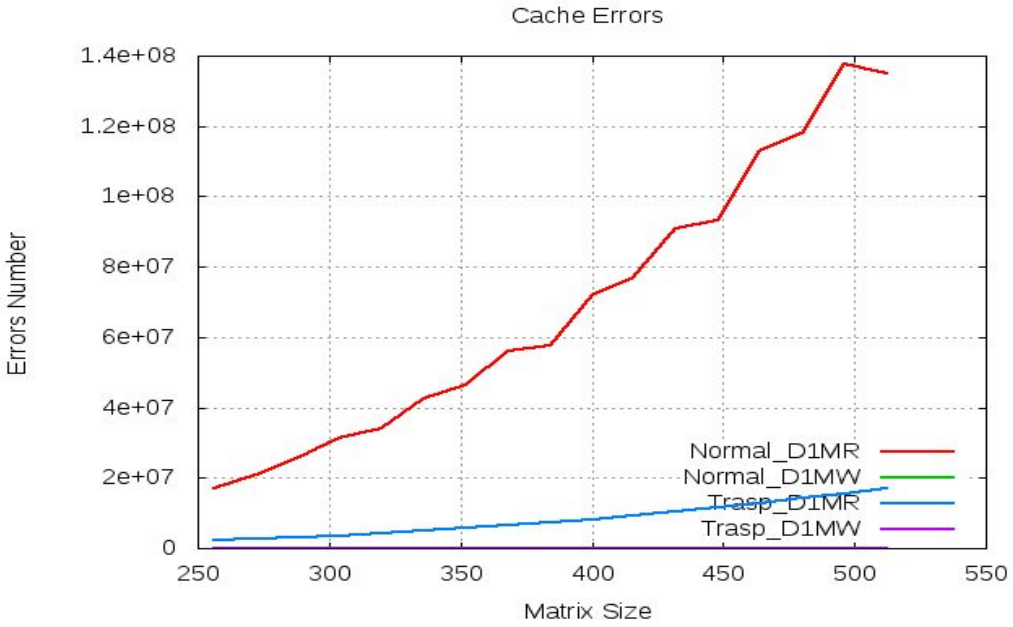
CACHÉ Y MULTIPLICACIÓN DE MATRICES

Justifique el efecto observado. ¿Se observan cambios de tendencia al variar los tamaños de las matrices? ¿A qué se deben los efectos observados?

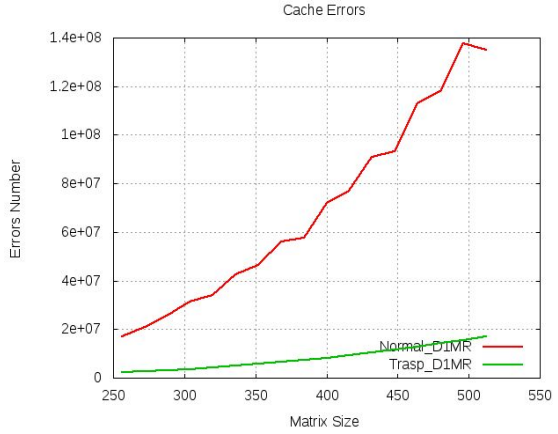
Este ejercicio se ha ejecutado realizando solamente 10 repeticiones en los laboratorios de la escuela puesto que si no tardaría mucho tiempo en ejecutarse (entendiendo que el objetivo no es esperar 3 horas a que acabe la ejecución si no un análisis de unos resultados coherentes) .
Las dos primeras gráficas que se presentan son las pedidas por los enunciados, las otras dos siguientes permiten hacer un mejor análisis de los fallos de lectura y de escritura por separado.

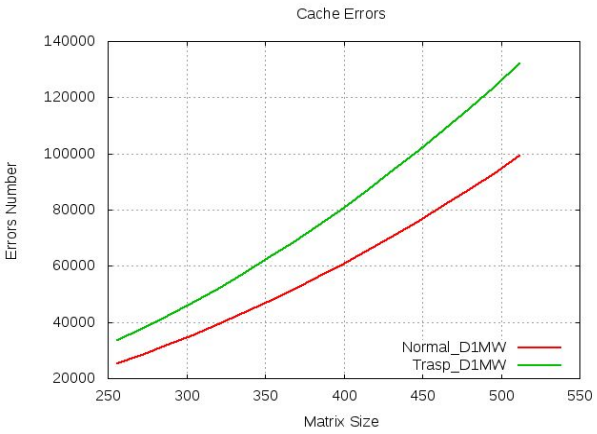


Tal y como se observa en esta primera gráfica, al ejecutarse el programa traspuesta, el tiempo de ejecución es constante mientras que el programa normal es más inestable, esto es debido a que multiplicar filas por filas es menos costoso que filas por columnas (multiplicación usual) y por tanto mucho más eficiente. Al igual que en el ejercicio 1, conforme se hace más grande el tamaño de las matrices, el tiempo de ejecución de ambos programas se separa más puesto que empieza a ser un factor muy importante el modo de ejecutar la multiplicación ya que no entra todo en caché.



En esta segunda gráfica y, ayudándonos de las dos presentadas debajo de esta explicación, se aprecian el número de errores de escritura y lectura para ambos programas.
Para los errores de lectura, observamos con más detenimiento la gráfica de abajo a la izquierda, en ella vemos que la diferencia entre el programa normal y el de la traspuesta es muy elevada puesto que cuando se mete una matriz en caché, se mete por filas y no por columnas, explicando esto el hecho de que el programa normal obtenga un elevadísimo número de errores de lectura en comparación con los de la traspuesta.
Para los errores de escritura, observamos con más detenimiento la gráfica de abajo a la derecha, en ella vemos que la diferencia entre ambos programas es menor aunque sigue siendo menor el número de errores en la traspuesta que en la normal. El motivo de este hecho es que en las traspuestas se necesita menos memoria caché para realizar las operaciones y por tanto hay más memoria caché libre a posteriori para ir guardando los datos. Además se observa que ambas crecen de una manera constante, lo que explica que una vez realizada la operación ambas guardarán en memoria de una manera más o menos constante.





Por último los fallos de escritura son menores que los de lectura porque se escribe con escritura retardada para no penalizar tanto los fallos, entonces solo se producen fallos cuando se va a sobreescribir en su posición de memoria caché.

Ejercicio 4

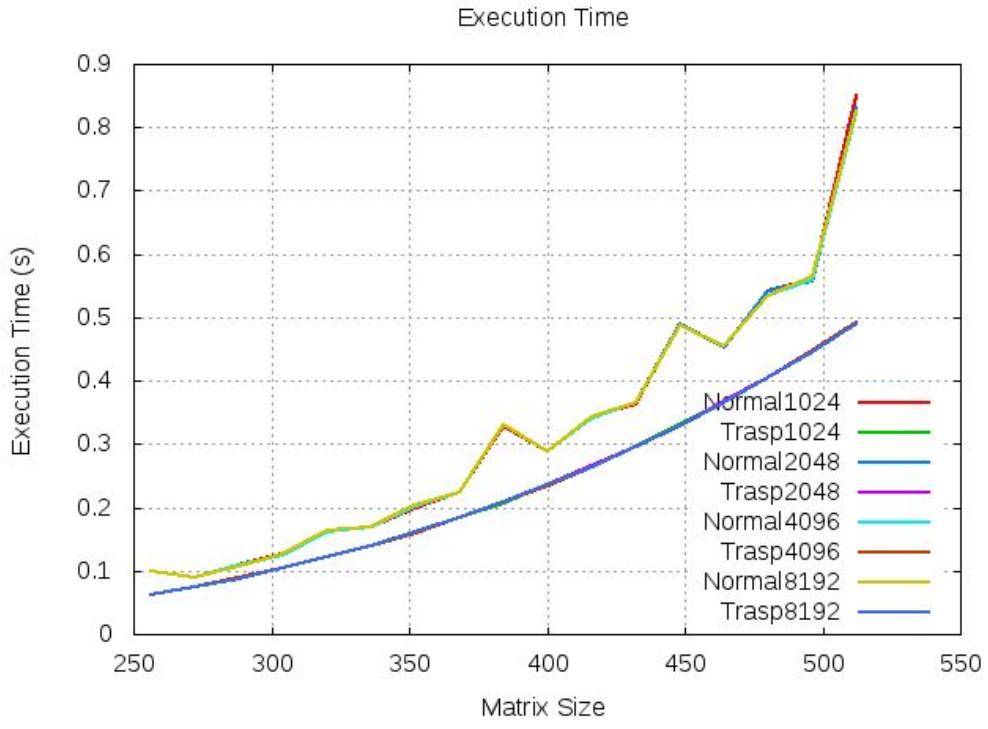
CONFIGURACIONES DE CACHÉ EN LA MULTIPLICACIÓN DE MATRICES

Explicación del experimento realizado y conclusiones adquiridas tras el experimento.

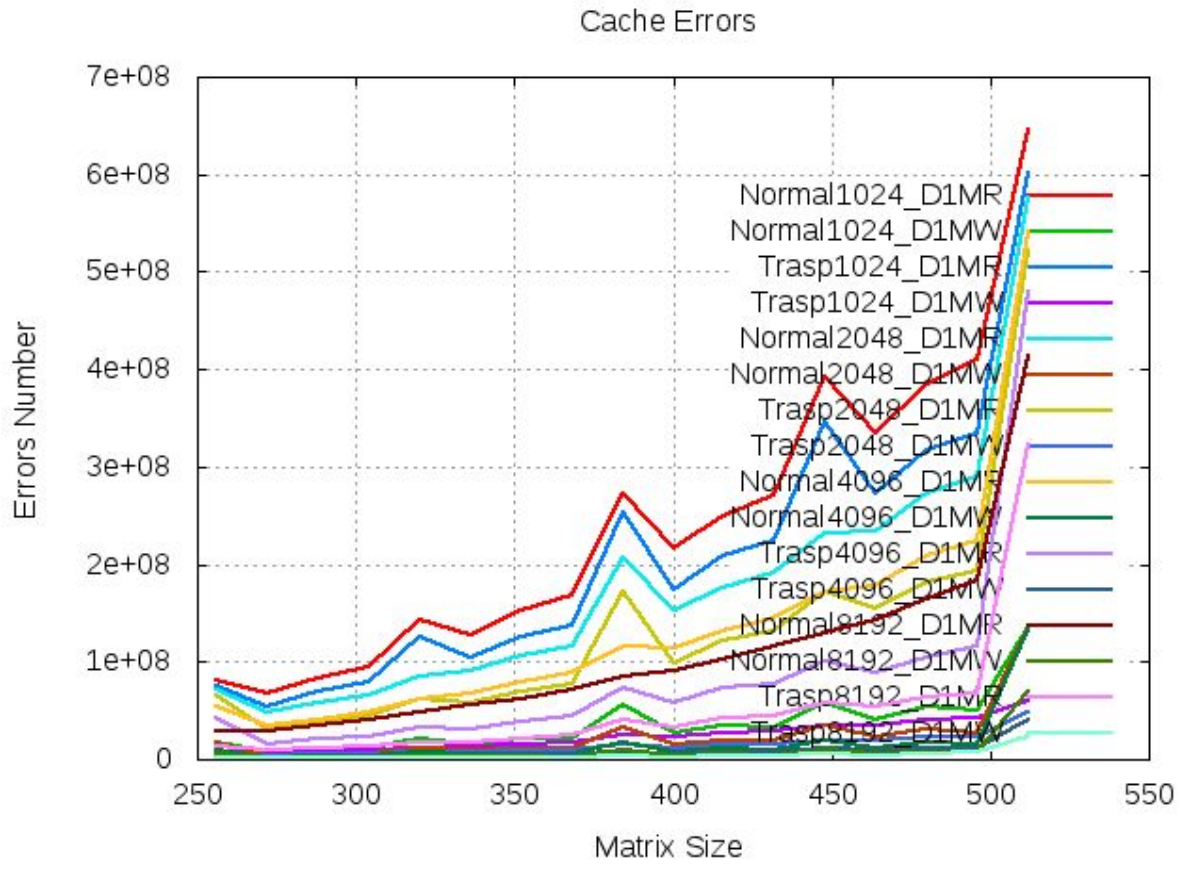
El experimento realizado es el mismo que en el ejercicio 2 pero con los programas normal y traspuesta. Este ha consistido en definir las cachés e ir realizando ambos programas obteniendo sus resultados tanto de tiempo de ejecución como de fallos de lectura y escritura.

Para realizar un análisis del experimento realizado se presentan, al igual que ocurriera en el ejercicio 3, dos gráficas principales: una de tiempos de ejecución y otra de fallos de lectura y escritura.

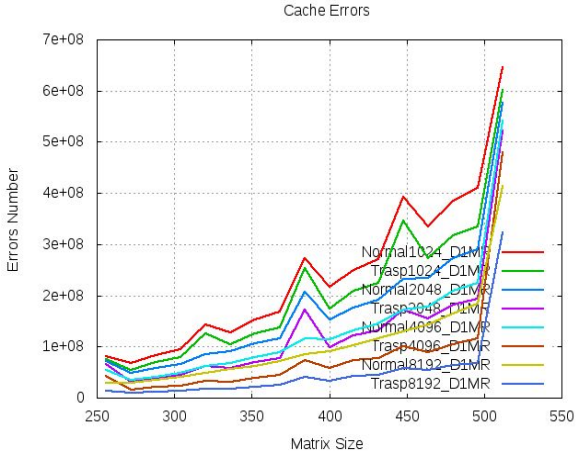
De nuevo, como en el ejercicio 3, la ejecución del programa tardaba del orden de varias horas así que se ha ejecutado en los laboratorios realizando cinco repeticiones por caché.

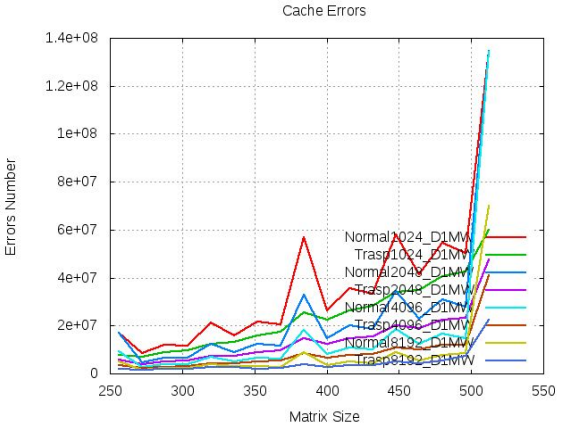


En esta primera gráfica sobre los tiempos de ejecución se observa que se solapan todos los tiempos de ejecución del programa normal y por otro lado, todos los tiempos de ejecución del programa traspuesta. Por tanto, se puede concluir que el tiempo de ejecución del programa no va a depender tanto del tamaño de caché como de la eficiencia del programa. Además se observa que es mucho más irregular para el programa normal y es más fijo para el programa traspuesta.



En esta segunda gráfica sobre el número de fallos tanto de lectura como de escritura, es difícil explicar nada, por eso recurriremos a las gráficas auxiliares de abajo.
Los resultados en estas gráficas y las del ejercicio 2 son exactamente los mismos para el caso de fallos de lectura puesto que el número de fallos se va solapando idénticamente (de mayor a menor número de fallos, encontramos Normal1024, Trasp1024, Normal2048, Normal4096, Trasp2048, Normal8192, Trasp4096 y Trasp8192). Luego el análisis realizado será el mismo que anteriormente. Sin embargo, para el número de fallos de escritura no se produce el solapamiento completo entre normal y traspuesta como sí ocurría en el ejercicio 2, la explicación de este hecho, es la misma que la del ejercicio 3.





Aclaraciones

Todas las ejecuciones se han realizado con P=0 ya que tal y como se indica, el valor de P se calcula sumando el número de la pareja y el número del grupo y haciendo módulo 10 del resultado. 1301+29=1330 mod 10 = 0

Adjuntos se encuentran (junto a los ficheros .dat y las gráficas .png pedidas) la captura de pantalla de la ejecución de ambos comandos del ejercicio 0, los scripts usados para cada ejercicio, los programas .c realizados para los ejercicios 3 y 4, el makefile modificado y una serie de gráficas auxiliares para el ejercicio 3 y el ejercicio 4 que permiten un mejor entendimiento y razonamiento analizando las mismas.