ARQO

Memoria Práctica 3

Roberto Martín Alonso

Diego Forte Jara

Pareja 11

ÍNDICE

Ejercicio 1……………………………………………………………………………………...…………4

Ejercicio 2……………………………………………………………………………….…….…………5

Ejercicio 3……………………………………………………………………………….……………….6

Ejercicio 4……………………………………………………………………………….……………….7

Ejercicio 5……………………………………………………………………………….……………….8

# Aclaraciones previas:

La práctica actual se va a desarrollar haciendo uso del subsistema de Windows para Linux (WSL), con las siguientes especificaciones:

Texto

Descripción generada automáticamente

Los ficheros entregados están distribuidos de la siguiente forma:

* En el directorio “Codigo” están todos los ficheros fuente base de los programas desarrollados en la práctica, ficheros .c, .h y makefile.
* En los directorios Ex (Donde x es el número de ejercicio) se encuentran los ficheros .c, .dat, .png y los scripts correspondientes a cada ejercicio. **Para ejecutar un script se debe pasar dicho script al interior del directorio “Codigo” junto a los ficheros .c (solo aplica para los ejercicios 4 y 5)**
* En el directorio “Docs” se encuentra la memoria de la práctica.

# Ejercicio 1: Información sobre la caché del sistema

Tras ejecutar los comandos getconf -a | grep -i cache y lstopo tal y como se pide en el enunciado se obtienen los siguientes resultados:

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

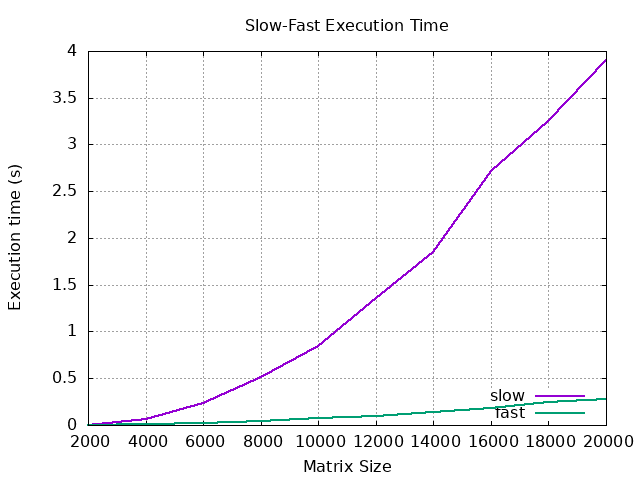
Diagrama

Descripción generada automáticamente

Se observa que los valores para la cache L3 mostrados por getconf -a | grep -i cache y lstopo no coinciden, lo cual es esperable ya que se está ejecutando en una máquina virtual.

# Ejercicio 2: Memoria caché y rendimiento

1. Para la realización de este apartado se crea un script bash, a partir del script suministrado de partida. Este script se llama “script\_ej2.sh” y realiza de forma intercalada la ejecución de slow y fast tal y como se pide en el enunciado.
2. Hay que realizar múltiples veces la toma de medidas para cada programa y tamaño de matriz debido a que en cada ejecución los tiempos varían, lo que provoca anomalías en las gráficas generadas. Para mitigar este efecto se realizan varias medidas en cada tamaño de matriz y se hace la media.
3. El script “script\_ej2.sh” genera automáticamente el fichero “time\_slow\_fast.dat” en el formato pedido, el cual se adjunta en los ficheros entregados.
4. El script “script\_ej2.sh” genera automáticamente el fichero “time\_slow\_fast.dat”, el cual se adjunta en los ficheros entregados y se muestra a continuación:



Como puede apreciarse en la imagen, no hay picos anómalos, esto es debido a la ejecución reiterada de los programas slow y fast para cada tamaño de matriz y el posterior cálculo de su media.

5a-Para matrices pequeñas el tiempo de cálculo es insignificante, por lo que la diferencia de tiempos es mínima. Cuando estos tamaños empiezan a crecer empieza a tomar relevancia el tiempo de cálculo y los datos empiezan a divergir. Esto es debido a que el programa fast hace las sumas de los elementos por filas y slow por columnas.

5b-La matriz se guarda en memoria por filas ya que la memoria es un array unidimensional.

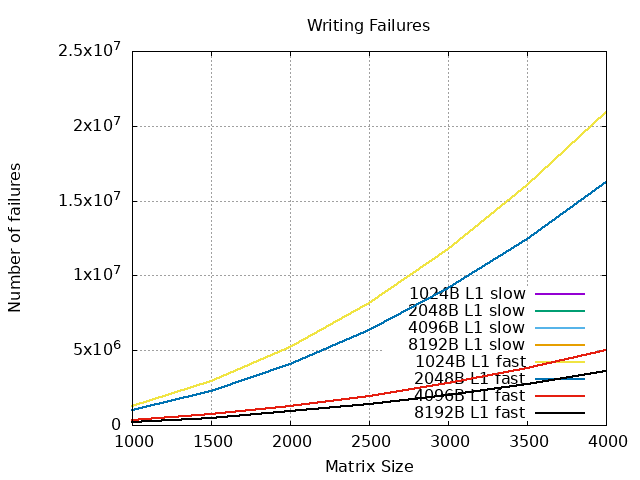
5c- El tamaño máximo de matriz que cabe en la caché L1, siendo esta de 32768 bytes y la arquitectura del procesador de 64 bits y cada double ocupa 8 bytes:

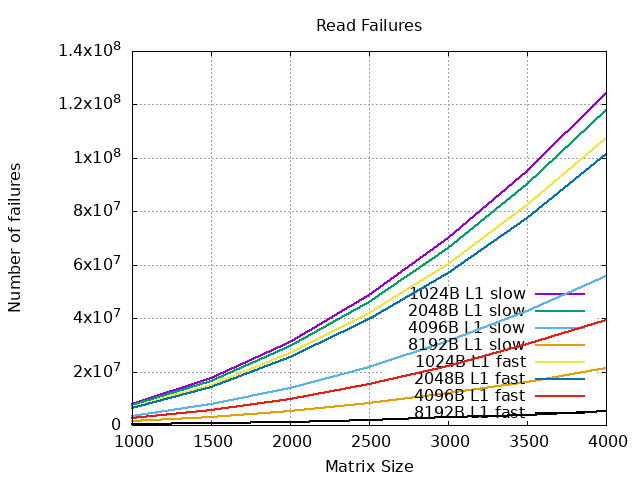
32768 bytes / 8 bytes/elemento = 4096 elementos

Haciendo su raíz cuadrada para obtener la dimensión de la matriz se obtiene que la dimensión máxima de la matriz es de 64.

# Ejercicio 3: Tamaño de la caché y rendimiento

3.1, 3.2, 3.3- Se crea un script llamado “script\_ej3.sh” con los requisitos pedidos y se incluye junto con los ficheros de datos .dat y las imágenes de las gráficas en la entrega de la práctica. Después de ejecutar el script las gráficas que se obtienen son las siguientes:





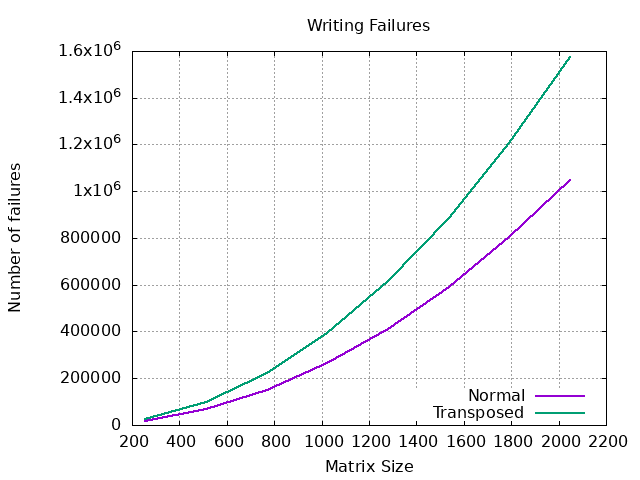
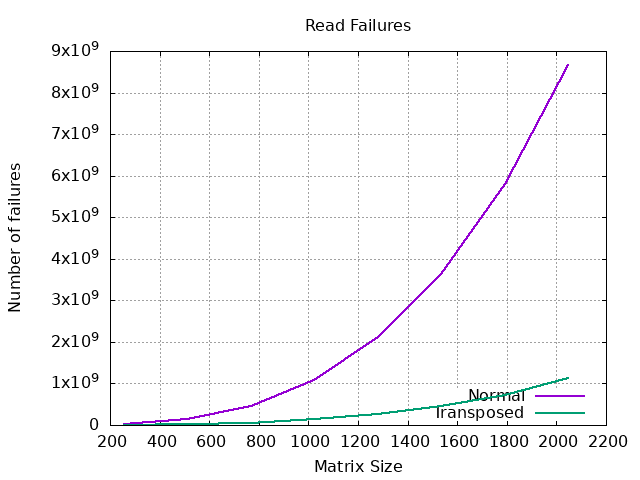
3.4a- Tanto para el programa slow como para fast se observan cambios al variar los tamaños de caché. Se puede apreciar que cuanto mayor es el tamaño de la caché L1 menor es el número de fallos en ambos programas.

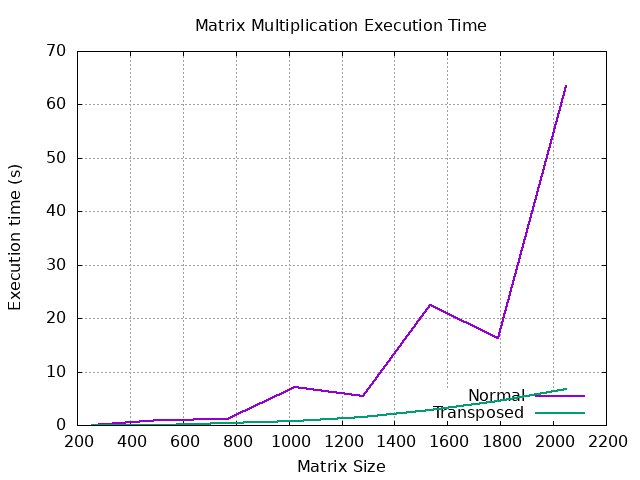
3.4b- Si se fija un tamaño de caché concreto se observa que el número de fallos de lectura en fast para un tamaño de caché dada es siempre menor que los fallos de lectura de slow. Esto se debe a al no encontrarse un dato en la caché, se trae a ésta el bloque que lo contiene y por el principio de localidad en este bloque se encontrarán datos que se usarán posteriormente por lo que el programa fast puede aprovecharse de esto y producir menos fallos de caché (se recuerda que el programa fast suma los elementos por filas en lugar de por columnas)

En lo que respecta a los fallos de escritura, el número de fallos de slow y fast para un tamaño de caché L1 es el mismo debido a que estos se guardan en memoria de forma idéntica en ambos programas.

# Ejercicio 4: Caché y multiplicación de matrices

4.1, 4.2, 4.3, 4.4- Se crea un script llamado “script\_ej4.sh” con los requisitos pedidos y se incluye junto con los ficheros de datos .dat y las imágenes de las gráficas en la entrega de la práctica. Después de ejecutar el script las gráficas que se obtienen son las siguientes:





4.5- Al variar los tamaños de las matrices se observa que:

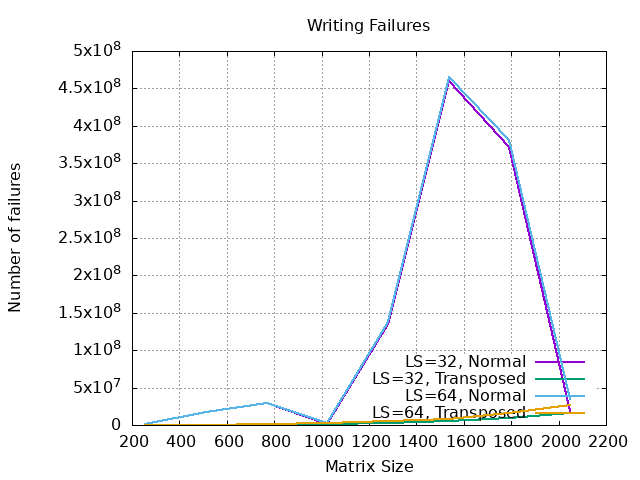
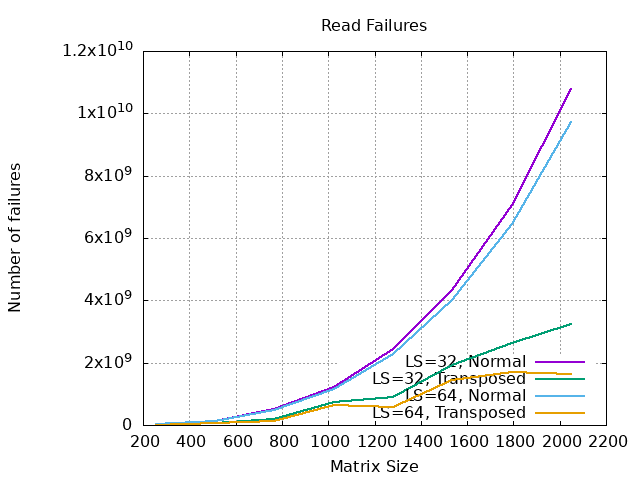
Fallos de lectura: El programa que realiza la multiplicación de matrices normal tiene más fallos de lectura cuando se aumenta el tamaño de la matriz que el que realiza la multiplicación con la traspuesta. Esto se debe a la forma en la que se accede a la matriz al realizar la multiplicación (acceso por filas en lugar de columnas, aprovecha la localidad espacial de la memoria), tal y como sucedía en el ejercicio 2.

Fallos de escritura: El programa que realiza la multiplicación de matrices traspuesta tiene mas fallos de escritura que el que realiza la multiplicación de matrices normal. Esto se debe a que una de las matrices tiene que trasponerse, lo que implica escribir datos en memoria.

Tiempo de ejecución: El programa que realiza la multiplicación de matrices normal tiene un tiempo de ejecución mayor cuanto más se aumenta el tamaño de la matriz que el que realiza la multiplicación con la traspuesta. Esto se debe a la forma en la que se accede a la matriz al realizar la multiplicación (acceso por filas en lugar de columnas, aprovecha la localidad espacial de la memoria), tal y como sucedía en el ejercicio 2.

# Ejercicio 5: Configuraciones de caché en la multiplicación de matrices

En este apartado se ha decidido estudiar los fallos de lectura y escritura al variar el tamaño de línea de la caché (32 y 64 bits) y analizar los resultados. Para ello se crea un nuevo script, “script\_ej5.sh” y se ejecuta, obteniéndose el fichero mult5.dat con los datos obtenidos y las gráficas siguientes (adjuntadas también en los ficheros entregados):



* Fallos de lectura: Se observa que cuanto mayor es el tamaño de matriz mayor es el número de fallos de lectura. Comparando las versiones de multiplicación normal y transpuesta y sus ejecuciones con tamaños de línea 32 y 64 bits encontramos que el número de fallos para las ejecuciones con tamaño de línea 64 bits es menor que su homólogo de 32 bits. Esto sucede debido a que como el tamaño de línea es mayor en 64 bits se realizan menos accesos a memoria para obtener datos, lo que produce un menor número de fallos de lectura.
* Fallos de escritura: Debido a la forma tan extraña de la gráfica de fallos de escritura no se pueden sacar conclusiones.