MEMORIA CACHÉ Y RENDIMIENTO

Práctica 3 Arquitectura de Ordenadores

Descripción breve de la práctica

El objetivo de esta práctica es experimentar y controlar el efecto de las memorias caché en el rendimiento de un programa de usuario.

Adrián San Felipe Martín y Carlos Miret Fiuza

Nota: todos los ejercicios han sido probados tanto en el clúster proporcionado por la universidad como en la máquina virtual del laboratorio, sin embargo, el ejercicio 3 no se ejecutaba correctamente en el clúster (y sí en la máquina virtual), y desconocemos la razón por la que ocurría esto (demasiadas personas a la vez probando ejercicios, limitación en el tiempo de ejecución…)

Ejercicio 0: Información sobre la caché del sistema

Para averiguar la información de la memoria caché de nuestra maquina hemos introducido todos los comandos ofrecidos en el enunciado de la práctica. Concluimos que el más útil y sencillo de entender es la información generada por el comando *getconf -a | grep -i cache*, aunque todos los resultados de ejecutar los demás comandos se encuentran en la carpeta de la práctica.

La información que nos aparece al ejecutar dicho comando es la siguiente:

```
LEVEL1_ICACHE_SIZE                 65536
LEVEL1_ICACHE_ASSOC                4
LEVEL1_ICACHE_LINESIZE             64
LEVEL1_DCACHE_SIZE                 32768
LEVEL1_DCACHE_ASSOC                8
LEVEL1_DCACHE_LINESIZE             64
LEVEL2_CACHE_SIZE                  524288
LEVEL2_CACHE_ASSOC                 8
LEVEL2_CACHE_LINESIZE              64
LEVEL3_CACHE_SIZE                  4194304
LEVEL3_CACHE_ASSOC                 16
LEVEL3_CACHE_LINESIZE              64
LEVEL4_CACHE_SIZE                  0
LEVEL4_CACHE_ASSOC                 0
LEVEL4_CACHE_LINESIZE              0
```

Podemos observar que los niveles de la caché van del nivel 1 al nivel 3 y que tienen tamaños diferentes (representados en Bytes).

El único nivel de caché que hace separación entre cachés de datos y de instrucciones es el nivel 1.

De la información podemos averiguar también que el tamaño de línea en todos los niveles de caché es el mismo, 64 Bytes. La asociación de la caché tiene un tamaño de 4 Bytes en el nivel 1 de instrucciones, 8 Bytes en el nivel 1 de datos y en el nivel 2, sin embargo, el tamaño de esta en el nivel 3 de cache es de 16 Bytes.

EL tamaño del nivel 1 de instrucciones es de 65KB, en el nivel 1 de datos es de 32KB, en el nivel 2 de 524KB y en el nivel 3 de 4MB.

Ejercicio 1: Memoria caché y rendimiento

1) Para la realización de este ejercicio hemos hecho uso de los programas "***slow.c***" y "***fast.c***" y en el primero de ellos se puede observar que el acceso de datos se realiza a través de las columnas, mientras que en "***fast.c***" se accede por filas (siendo esta manera de ejecución más eficiente).

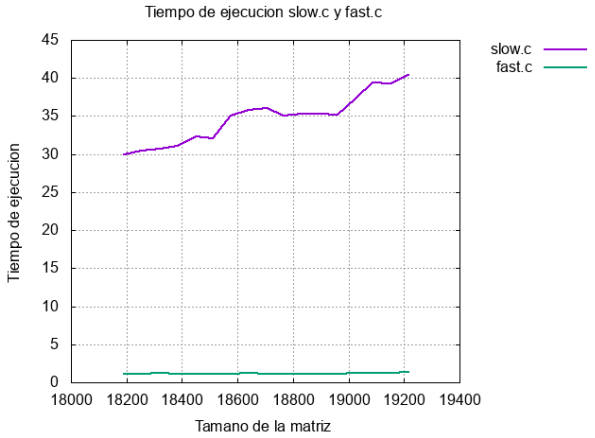
2) En el script es donde llevamos las ejecuciones de forma intercalada ya que así evitamos tomar datos no reales y datos que no se hayan almacenado en la caché en ejecuciones anteriores. En primera instancia hicimos uso de 5 repeticiones intercaladas de ejecución, pero esto nos ha llevado a una saturación de la máquina virtual y hemos procedido a tomar medidas de ejecución de 3 en 3 ("***slow***" y "***fast***").

En las ejecuciones "***slow***" se accede a los datos de la matriz por columnas (posiciones de memoria no contiguas y alejadas) por lo que se deben cargar nuevas páginas de memoria en la caché, mientras que en la opción "***fast***" al cargar una página de memoria a caché, los datos se encuentran contiguos y no hay que realizar continuos accesos de memoria a cache para la carga de dichas páginas.

3)

18192	29.9771950000	1.0694170000
18256	30.4766990000	1.0903505000
18320	30.7200580000	1.2056840000
18384	31.1370480000	1.0702065000
18448	32.2617790000	1.0595135000
18512	32.0933080000	1.0716785000
18576	35.1038375000	1.1279140000
18640	35.8217915000	1.1946385000
18704	36.0662155000	1.1242450000
18768	35.0392770000	1.1491160000
18832	35.2814920000	1.1213855000
18896	35.3385025000	1.1319055000
18960	35.1217325000	1.1531580000
19024	37.2334025000	1.2180800000
19088	39.3786135000	1.2225615000
19152	39.3168170000	1.2180155000
19216	40.3932190000	1.3992265000

4)



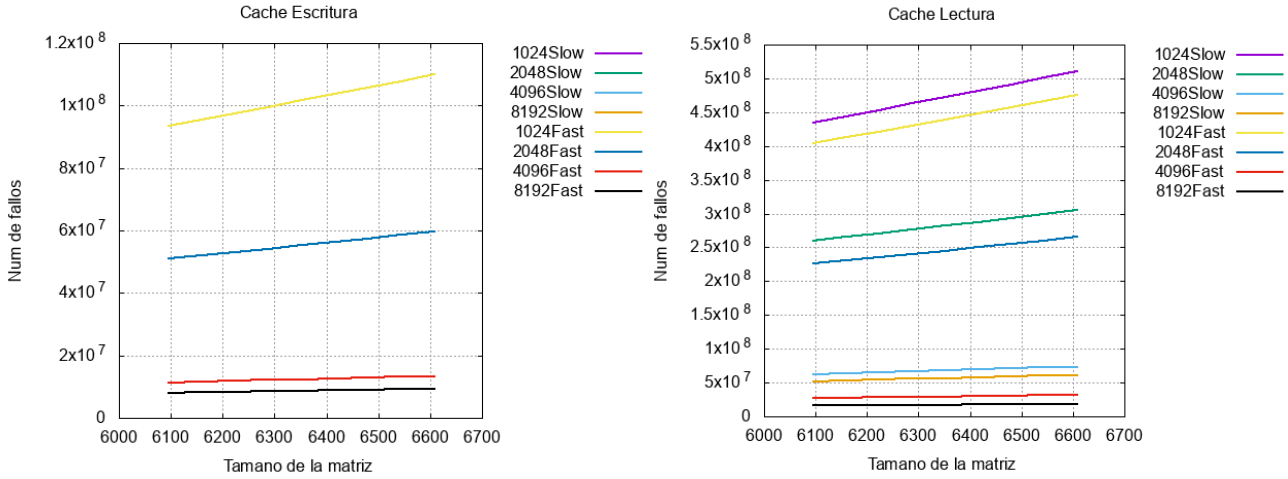
5) Para tamaños de matriz pequeños los tiempos de ejecución son similares para ambos programas, es decir, no se haría uso completo de la memoria caché, en cambio si se utiliza un tamaño grande de la matriz, el número de fallos de página aumenta y por consiguiente aumenta el tiempo de ejecución del programa, ya sea "**slow**" o "**fast**"

Ejercicio 2: Tamaño de la caché y rendimiento

1) En este apartado utilizamos slow y fast con distintos tamaños de caché de datos e instrucciones. Como se nos indica, utilizamos tamaños de caché de 1024, 2048, 4096 y 8192 bytes respectivamente, y los programas serán ejecutados a través de valgrind. El incremento de salto es de 64 unidades y el tamaño de línea es 64 bytes.

2) Guardamos en varios archivos .dat los valores de los resultados obtenidos al ejecutar cada cache de tamaños diferentes (cache_1024.dat, cache_2048.dat…). Los valores de cada línea (de izquierda a derecha) se refieren a: La N ejecutada, el nivel D1 de lectura del programa slow, el nivel D1 de escritura del programa slow, el nivel D1 de lectura del programa fast y el nivel D1 de escritura del programa fast. (Incluidos en la carpeta del ejercicio)

3)



4) Como podemos observar en la grafica de los fallos de caché en escritura, no hay diferencias notorias de fallos entre los programas slow y fast ya que el método de escritura que realizan es el mismo. Sin embargo, si que podemos observar una tendencia: cuando el tamaño de la caché es mayor, el numero de fallos se reduce considerablemente. También comprobamos gracias a la gráfica que cuando el tamaño de la caché es pequeño y el tamaño de la matriz va en aumento, el numero de fallos también se incrementan, y cuando la caché es mas grande el numero de fallos se estabiliza, variando mínimamente por el tamaño de la matriz.
Por otra parte, en la gráfica de los fallos de caché en lectura, podemos observar una constante: cuanto menor es el tamaño de la caché, mayor es el número de fallos en escritura y con el mismo tamaño de caché, el programa fast causa menos cantidad de fallos. También ocurre que la pendiente de la relación fallos-tamaño de matriz aumenta con los tamaños más pequeños de caché.

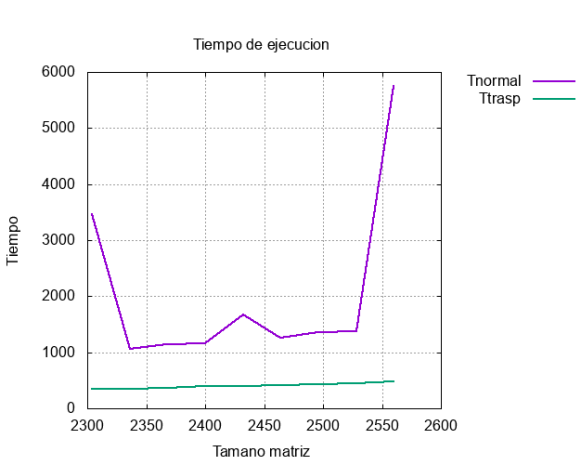
Ejercicio 3: Caché y multiplicación de matrices

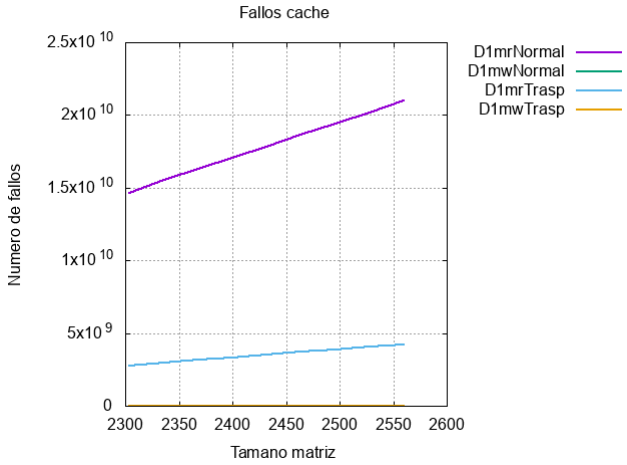
1) Para realizar este ejercicio de la práctica hemos tomado diferentes valores para calcular el tiempo de ejecución de la multiplicación de dos matrices. Para comenzar, la matriz es del tamaño NxN, con N inicial igual a 256+256*P y N final 256+256*(P+1). En nuestro caso, P obtiene el valor de 8 y el incremento de salto es de 32 unidades en cada ejecución. Utilizaremos dos programas de multiplicación de matrices, uno de multiplicación "normal" que obtiene los datos de la segunda matriz por columnas y uno de multiplicación de matrices traspuesta, que genera la matriz traspuesta de la segunda matriz y obtiene los datos por filas.

2) Para este ejercicio de la práctica hemos tomado los valores anteriores de N y P para calcular el numero de fallos de caché al ejecutar la multiplicación normal y traspuesta. Las conclusiones sobre los datos recibidos serán detalladas en el apartado 5 del ejercicio.

3) Hemos almacenado en varios ficheros .dat los tiempos de ejecución de la multiplicación de matrices normal y traspuesta, que se encuentran en la carpeta de este ejercicio.

4)





5) Como podemos observar en la gráfica de los fallos de caché, la lectura de los datos de la multiplicación de matrices "normal" por columnas y la lectura de los datos de la multiplicación de matrices traspuesta tienen un gran numero de fallos, aunque la lectura de la matriz normal supera por creces a la matriz traspuesta, y cuanto mas grande es la matriz, mas numero de fallos genera. Sin embargo, la escritura genera un fallo mínimo (apenas se ve en la gráfica) sea cual sea el tamaño de la matriz.
Por otro lado, tenemos la grafica del tiempo de ejecución de la multiplicación de matrices. El tiempo de la multiplicación traspuesta se podría considerar constante, ya que aumenta muy poco cuando aumenta el tamaño de la matriz, en contraparte con la multiplicación normal, no entendemos muy bien el resultado pintado en la gráfica. Pensamos que la primera bajada es causada por el "esfuerzo inicial" que debe hacer las matrices a la hora de leer los datos por columnas, ya que luego (salvo el pico entre los tamaños 2400 y 2500) se mantiene constante (aunque siempre por encima del tiempo de ejecución de la multiplicación traspuesta) hasta que llega al tamaño de matriz 2528, el cual aumenta el tiempo de ejecución hasta su mayor pico.