ARQUITECTURA DE ORDENADORES
2018/2019

Práctica 3

Alberto Ayala, Mihai Blidaru

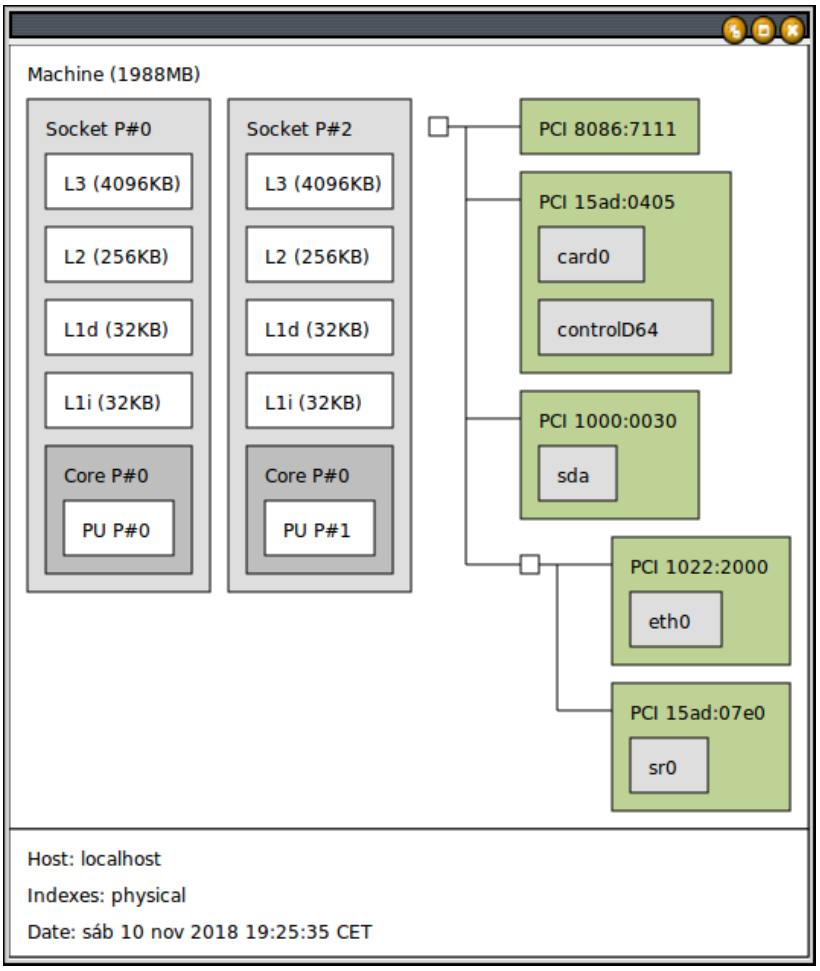
Grupo **1311**

Ejercicio 0: información sobre la caché del sistema

Usando el comando indicado en el enunciado obtenemos la siguiente salida:

```
mihai@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                 4
 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
mihai@localhost:~$
```

Así mismo, usando **lstopo** obtenemos la misma información en un formato más gráfico:



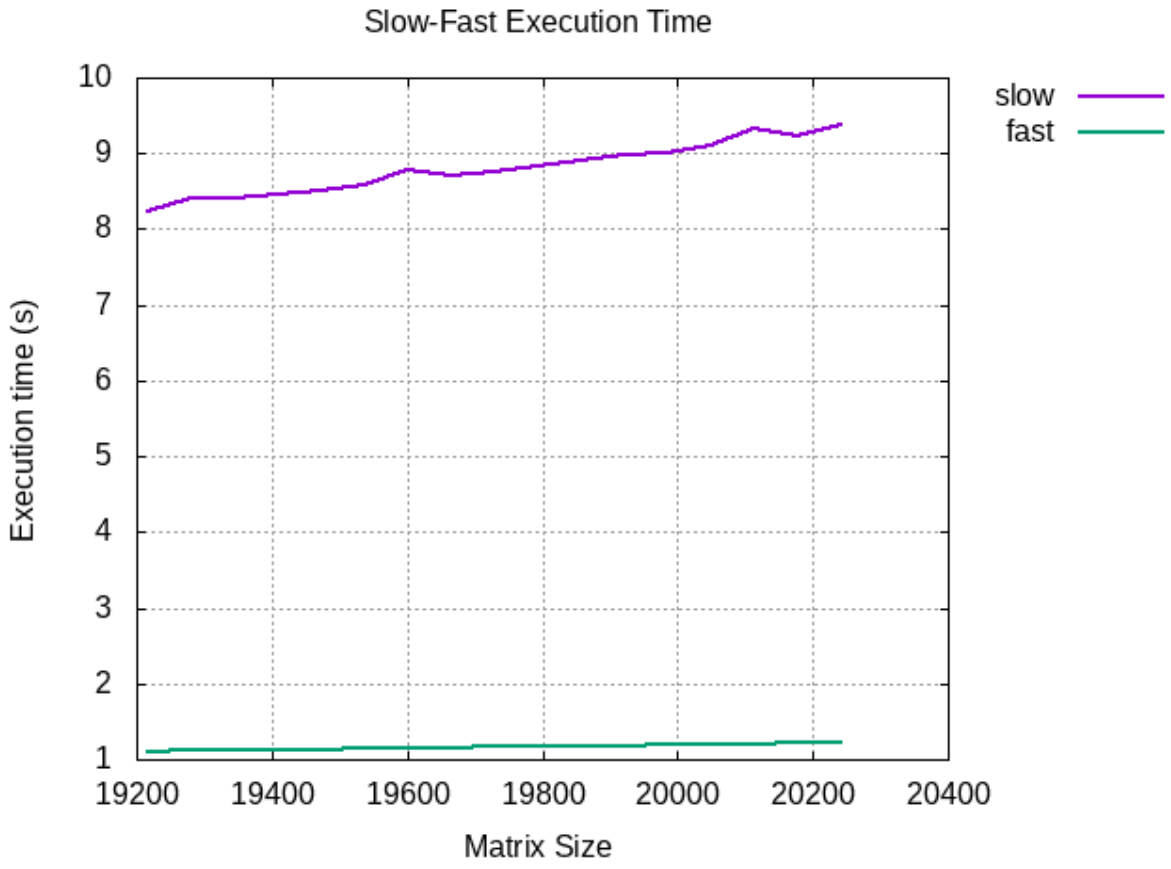
El procesador de la máquina en la que trabajamos por cada núcleo del procesador tiene los siguientes niveles de caché

Nivel	Tamaño
L1 Instrucciones	32Kb
L1 Datos	32Kb
L2	256Kb
L3	4Mb

Ejercicio 1: Memoria caché y rendimiento

Como bien dice el enunciado en este ejercicio nos valdremos de dos programas que utilizaremos para comprobar empíricamente como al cambiar el patrón de acceso a los datos se puede aprovechar mejor las memorias caché.

En nuestro caso con P = 9, y siguiendo las especificaciones de la práctica hemos obtenido los siguientes resultados:



Para la obtención de los datos hemos realizado múltiples veces la toma de medidas ya que al hacer la media de todas ellas podemos obtener unos resultados más exactos.

El procedimiento realizado en nuestro ejercicio consiste en llamar en bucle al programa slow para cada N especificado en el enunciado teniendo en cuenta nuestro número de pareja, siendo N el tamaño de la matriz a sumar, de la misma manera llamaremos en bucle al programa fast con los mismos parámetros. Este procedimiento lo realizamos en bucle las veces para las que consideramos que hemos obtenido datos que nos proporcionan valores muy acercados a la media real.

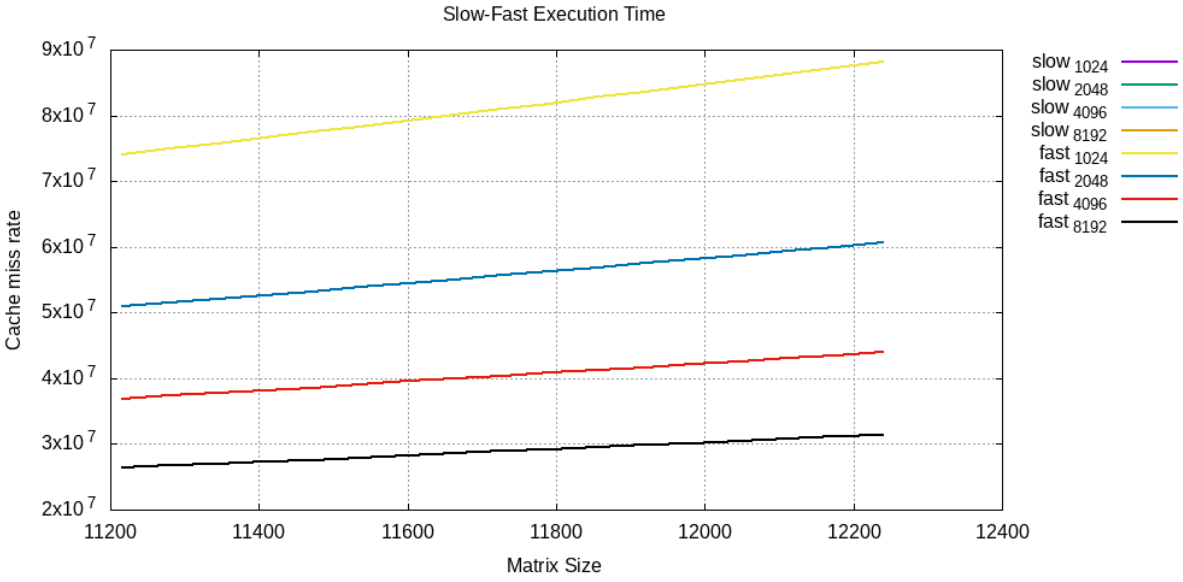
Como podemos ver en la gráfica, la diferencia entre los dos programas para valores pequeños es mínima, esto es debido a que, aunque la forma en la que accedemos a los datos sea distinta, el tiempo que tarda en hacer la operación de suma para valores pequeños es muy corto, sin embargo, a medida que aumentamos el tamaño de la matriz podemos observar cómo se hace cada vez más evidente la diferencia.

Ejercicio 2: Tamaño de la caché y rendimiento

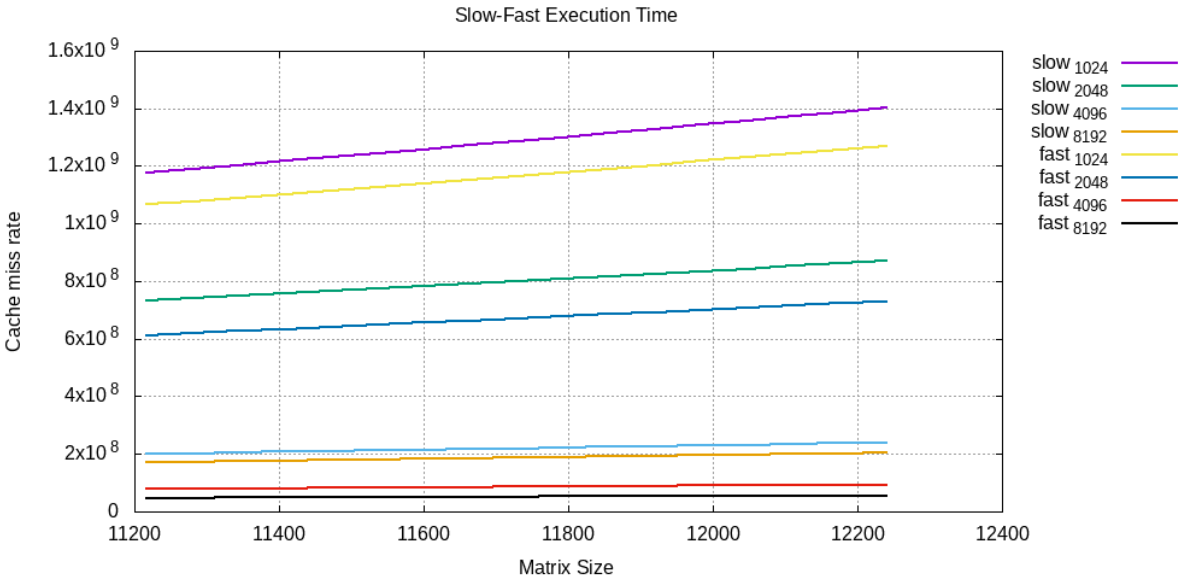
En cuento a la estructura de este programa es un poco distinta del anterior debido a que podemos lanzar los programas slow y fast seguidos y una sola iteración para obtener los datos requeridos por el enunciado.

Básicamente consiste en un programa que nos muestra en las siguientes gráficas el número de fallos de acceso a las memorias cache tanto al ejecutar slow como fast de lectura y de escritura, el tamaño de la matriz a sumar en nuestro caso varía entre (2000 +1024 * P) y (2000 +1024 * (P+1)) donde P es 9, además para cada N variaremos el tamaño de las cache tal y como se especifica en el enunciado de este ejercicio.

Como podemos observar, en la gráfica que nos muestra los fallos de escritura las rectas de fast y slow se superponen, esto se debe a que solo existe este tipo de fallos a la hora de generar la matriz no al realizar la suma, en este caso ambos programas tienen el mismo algoritmo por lo que para cada tamaño de la cache el número de fallos es el mismo. Además, observamos que cuanto mayor es el tamaño de las memorias el número de fallos es menor ya que podemos almacenar más datos en cada nivel.



En el caso de la gráfica que nos muestra los fallos de lectura podemos ver que al igual que la gráfica anterior los fallos son menores cuanto más grande es el tamaño de las memorias, sin embargo y en este caso, el número de fallos del programa slow es mayor que el del programa fast, esto se debe a que de acceso a los datos de fast se hace mediante filas, es decir cuando cargamos por ejemplo el primer dato de una fila de la matriz cargamos una línea entera de caché que contiene más datos de esta fila , y como vamos sumando por filas no va a haber ningún fallo de lectura porque esos elementos ya están cargados, sin embargo al leer por columnas esos datos no nos sirven ya que el algoritmo de slow no los va a usar en ese momento implicando el aumento de los fallos de lectura.



Ejercicio 3: Caché y multiplicación de matrices

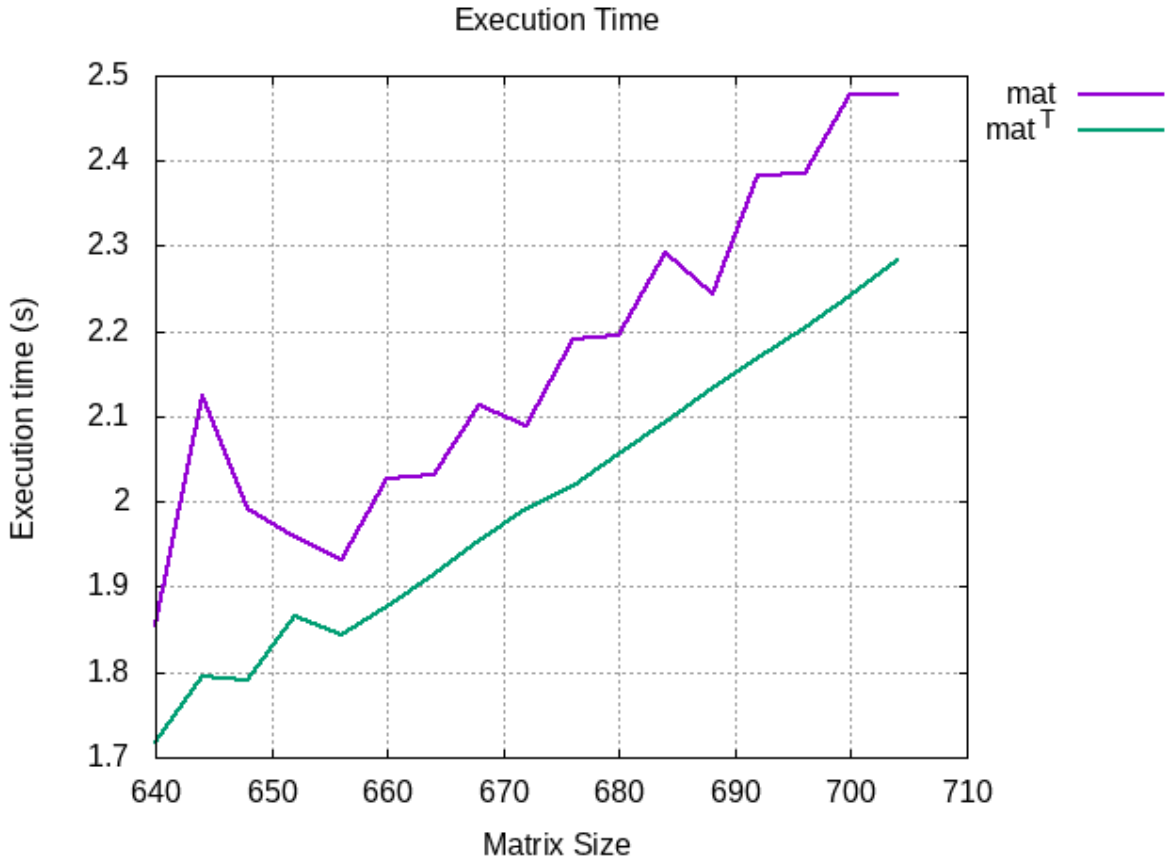
Para este ejercicio hemos implementado los dos programas de multiplicación de matrices en los ficheros **matmul.c** y **matmultrans.c** usando la misma salida que los programas proveídos con el enunciado de la práctica.

El primero de ellos es la multiplicación de una matriz por ella misma y el segundo por su traspuesta.

Al igual que el ejercicio uno, mediremos el tiempo que tardan en realizar la multiplicación de matrices los dos programas nuevos que hemos realizado en función del tamaño de la matriz especificado por el enunciado, para ello tendremos un bucle para cada programa y otro externo que nos permitirá hacer una media de los datos y conseguir una gráfica más exacta.

Al igual que el ejercicio dos obtendremos los fallos de acceso a los datos en cache tanto de escritura como de lectura en ambos programas

Observando los datos obtenidos podemos afirmar que a medida que crece el tamaño de las matrices el tiempo que tarda en ambos programas aumenta, sin embargo, el tiempo que tarda la multiplicación de una matriz por su traspuesta crece menos que la multiplicación de matrices normal por lo que cuanto más grande sea el tamaño de la matriz mayor será la diferencia entre ambas gráficas.



Al igual que en ejercicio dos y observando la gráfica, el número de fallos de escritura en cache de ambos ejercicios es el mismo debido a que usan el mismo

algoritmo para la generación de matrices, a diferencia del ejercicio dos no variamos el tamaño de las cache.

En cuanto a los fallos de lectura y coincidiendo de nuevo con el ejercicio 2, ambos aumentan con el tamaño de la matriz, pero el numero de fallos de la multiplicación normal es mucho mas grande debido al patrón de acceso de datos explicado ya en el ejercicio 2.