Group 13
Pablo Almarza Marques
Jorge González Gómez

Task 3: Cache and Performance

Group 13
Pablo Almarza Marques
Jorge González Gómez

Exercise 0: System Cache

There are 3 different cache levels in the machines from the labs.

In the first level, there are also two different types: one for the instructions and another for the data.
The level 1 cache is an 8 way associative cache and it has 64KB per core (in our case, 6 cores, totalling a maximum size of 384 KB)
The level 2 cache is a 4 way associative cache and it has 256KB per core (in our case, 6 cores, again totalling an amount of 1536 KB).
The level 3 cache is a 12 way associative cache, and it's shared by every CPU. It has a capacity of 9216 KB.

Group 13
Pablo Almarza Marques
Jorge González Gómez

Exercise 1: Cache and Performance

The measurements must be taken multiple times because the scheduler may pick another job instead of the one that we are running, so the time can vary based on other processes (or the O.S.). Also, the cache memory can reduce the execution time based on how many cache misses happen (either due to other processes or a poorly coded algorithm).

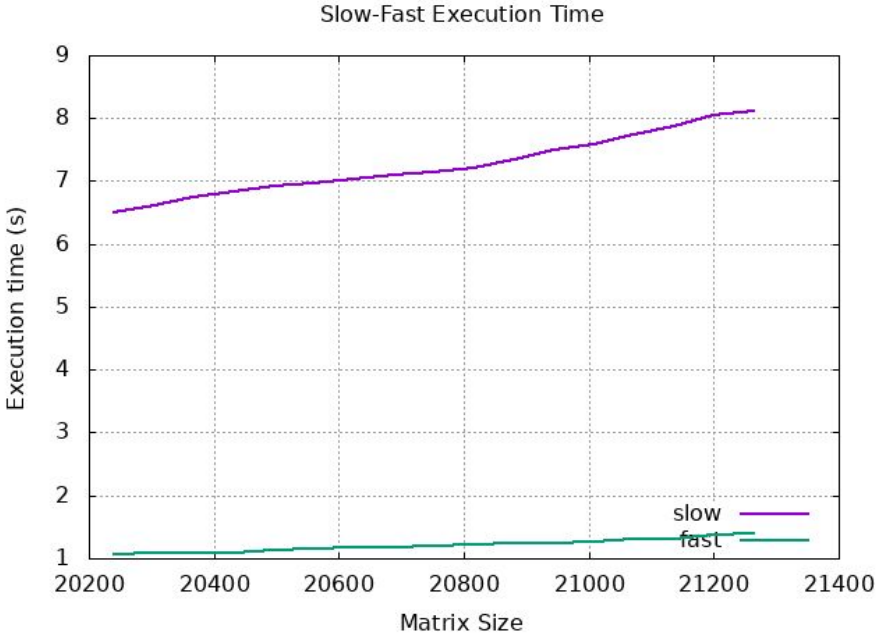
In this program, the interleaved process is done as follows:

```
./slow N1
./slow N2
./slow ...
./slow Nfinal
./fast N1
./fast N2
./fast ...
./fast Nfinal
```

Explanation about the results:

We observe there's a few spikes, this leads us to believe we have done it right. We can also appreciate there's a tiny spike at the start, but it's insignificant. We ran it 15 times, we believe this was the correct amount of times to run it.

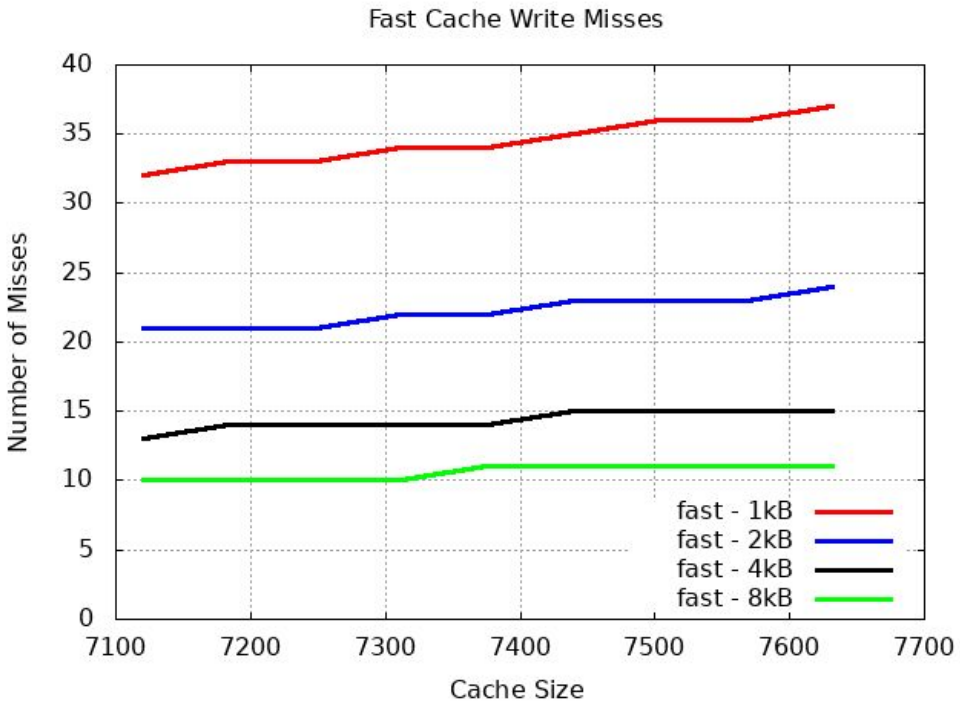
When the size increases, the program that computes the matrix per rows (fast.c) will take much less time to operate than the program that operates them per columns (slow.c). This is because the number of cache misses are proportional to the size of the matrix. The matrix is stored in the memory per rows, indicated by this statement:
`matrix[i][j] = (1.0*rand()) / (RAND_MAX/10);`

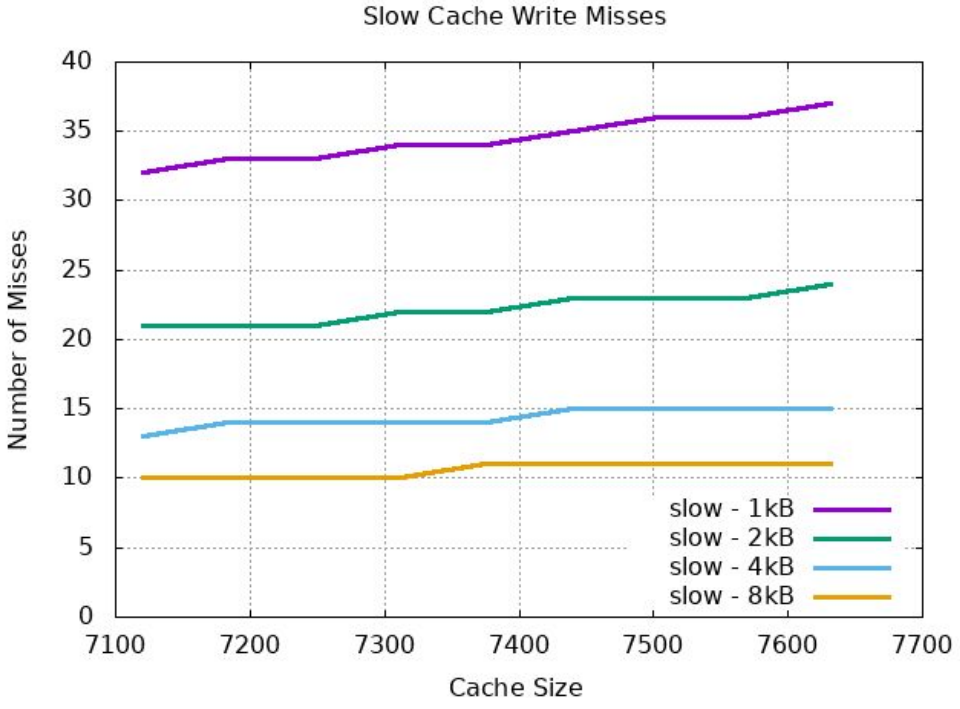


Measurements taken with our script slow_fast_time.sh.

Group 13
Pablo Almarza Marques
Jorge González Gómez

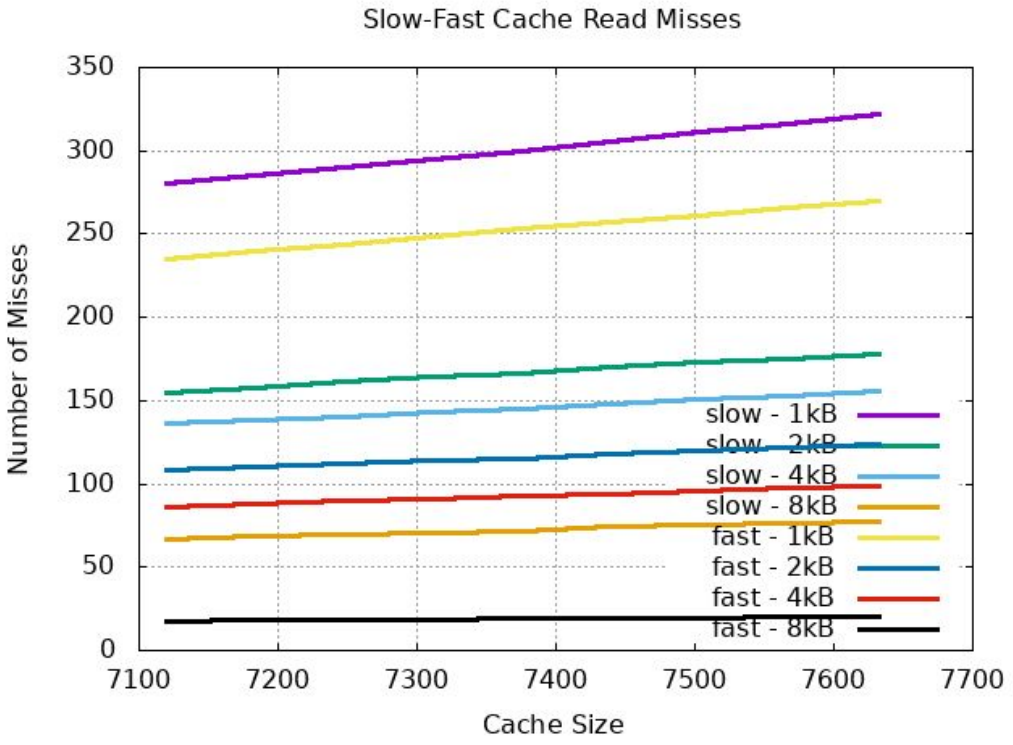
Exercise 2: Cache size and Performance





We can observe that there are the same write misses for both slow program and fast program either if the cache size is different or, in the same cache size, we compare fast and slow. This happens because the two programs write at the same time using the same algorithms, so there is no program that writes more or less times than the other.

Group 13
Pablo Almarza Marques
Jorge González Gómez

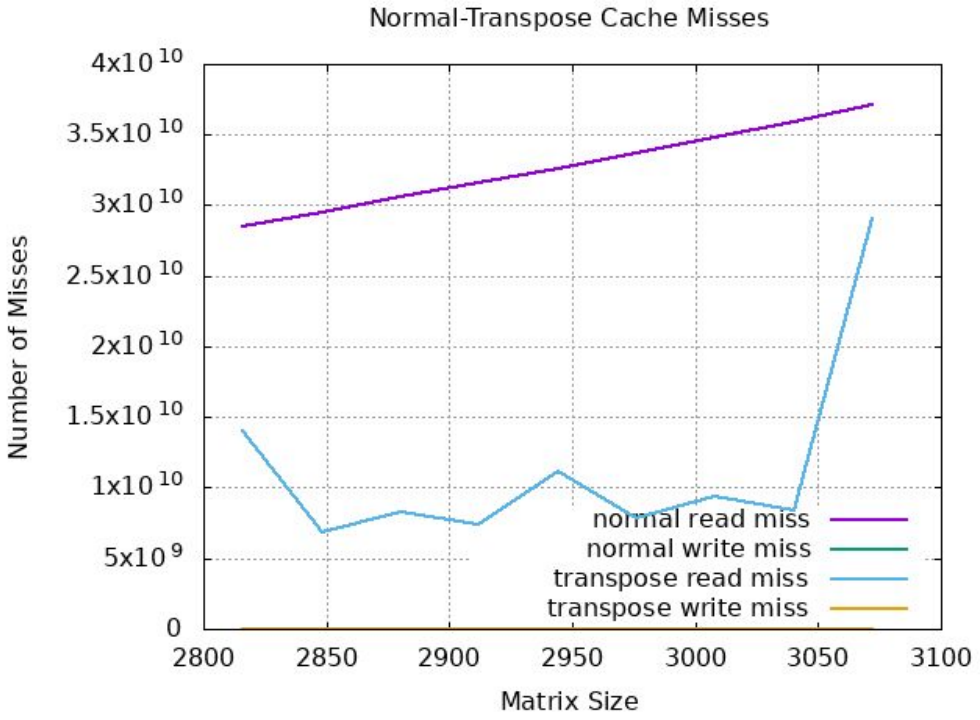


We can observe by varying the size of the caches that, for the slow program, the bigger the cache size is, the less the misses are. This trend can also be seen in the fast program. This happens due to the more the cache size is, the more information will be saved so the program will find it in the cache more times when the cache size is big.

Also, when you look at a same sized cache and compare the slow and fast program, it is clearly seen that the fast program does less read misses than the slow program, because the slow program reads by columns but the memory has been written by rows.

Group 13
Pablo Almarza Marques
Jorge González Gómez

Exercise 3: Cache and Matrix multiplication

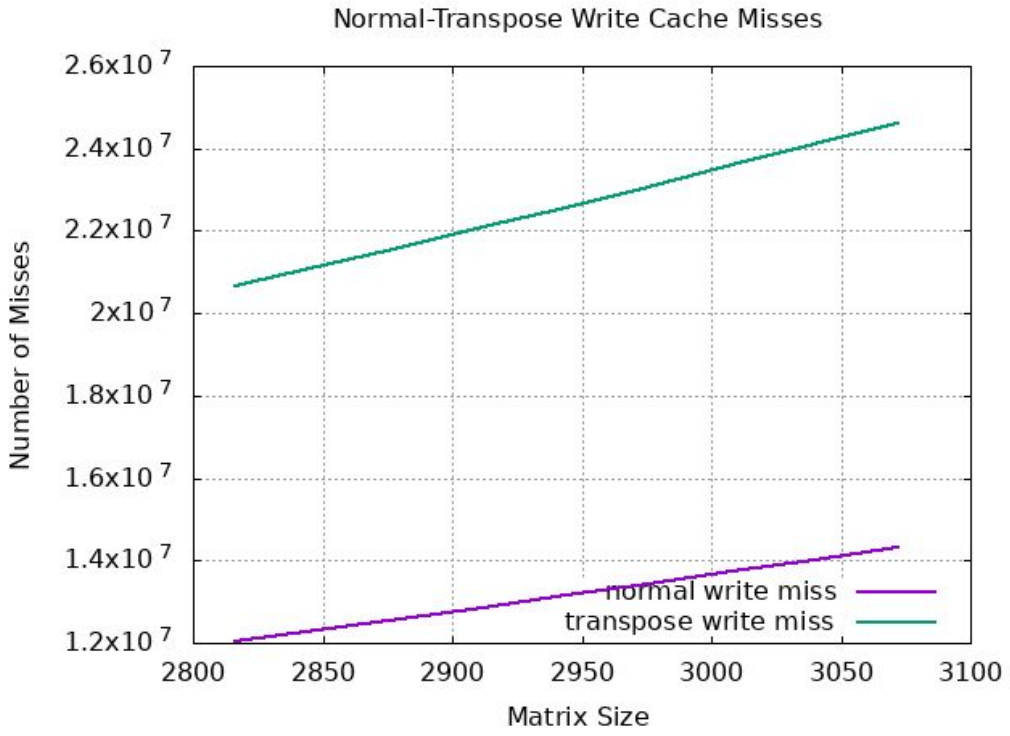


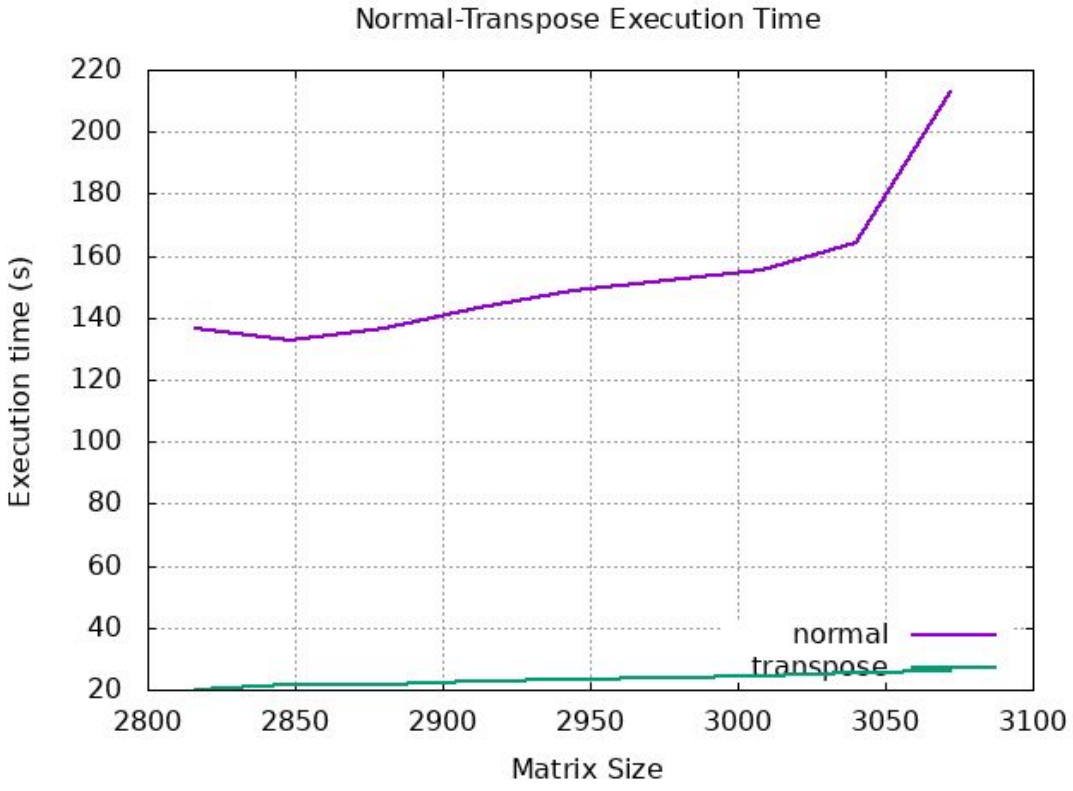
If we look at the plot, we can clearly see that the normal read miss is higher than the transpose. That is because the normal multiplication is read by columns on the second matrix while the transpose is read by rows in both the first and the second matrix, so the misses will be less on the transpose.

If we look at write misses, both of them are lower than the plot reflects, that is that normal write miss is of the order of 1*10^7 and the transpose write miss of the order of 2*10^7. The reason behind the transpose program doing more write cache misses is simply because we're writing a new matrix to "convert" the 2nd matrix to a transpose one. We plotted the cache write misses down below to reflect how they truly vary as you change the size of the matrices.

Also, the anomaly in the transpose read miss is due to the fact the matrix row size is bigger than the cache, and thus, since the row would be "more likely" out of our cache blocks, and thus the number of read misses will be much higher. The fact we're taking the second matrix to make a transpose of it and reading that one by columns also contributes to this weird effect.

Group 13
Pablo Almarza Marques
Jorge González Gómez





The execution time is less in the transpose because of the fact that the transpose is read by columns, so it takes less time to do the operations than if it was read by column. Also, this is backed by the fact the cache read misses of “normal” are more than the transpose.