# HPC, Parallel Programming: Homework 4

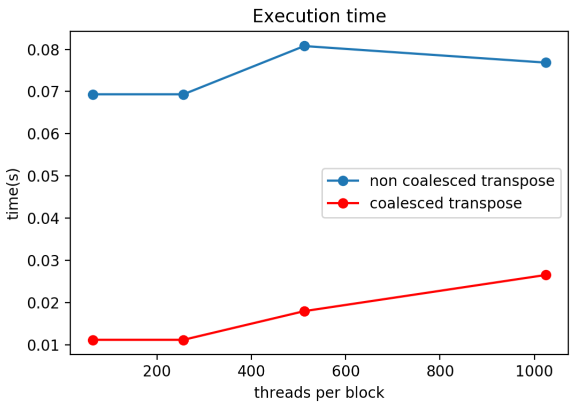
Simone Amadio

## General informations

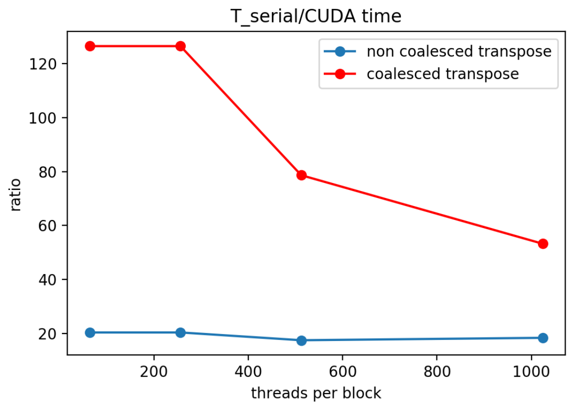
For clarity and brevity, the results will be presented as graphs. The code and files containing the results are provided separately in this same folder. Only peculiar results will be analyzed singularly.

## 1 Coalesced Transpose

In this exercise I recorded times for the transposition problem in three cases: serial, CUDA coalesced (exploiting shared memory) and CUDA non coalesced. In order to record the device execution time, I exploited the function cudaDeviceSynchronize(), as suggested on CUDA documentation.
In particular as requested I transposed a matrix of dimension $8192^2$ and a number of threads per block nth$\in$ $\{64, 512, 1024\}$. For further sampling, I also recorded times for nth$=$ 256.
When possible, the whole matrix has been divided in square blocks of dimension nth, while for values of nth s.t. $\sqrt{(nth)} \notin \mathbb{N}$ (i.e. 512), I supposed to have an algorithm which returns a couple (a,b)$\in$ $\mathbb{N}$ such that $ab = nth$ and $|b - a|$ is minimum. So I used a rectangular matrix block as close as possible to a square matrix.
For each value of nth the measurement has been repeated 5 times, in order to get a mean and check results consistency. A naive auxiliary function has also been used in order to check results correctness.
Once again, the data has little to no oscillation, so the standard error is negligible.
As expected, the coalesced transpose procedure times are always less than the naive ones.



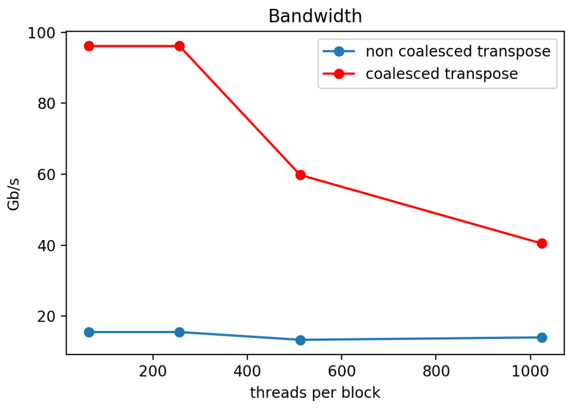
The scaling factor with respect to the serial time has also been calculated and plotted. It's interesting to note that the ratio seems to be decreasing as a function of nth. This is probably due to the synchronization procedure, which becomes more challenging and expensive as nth grows.



The time measurement has also been used to estimate the device memory bandwidth, as

$$\text{bandwidth}{=}8192^2 * \texttt{sizeof(size\_t)} * 2/time$$

The result is again decreasing with respect to nth, but we see that for nth=64,256 we obtained a bandwidth value similar to the nominal one, which is $\approx 110Gb/s$.



## Code and file references

- Results are obtained through the script `script.sh`. The code in `transpose.c` has been used to obtain an executable.
- Times for each value of nth are contained in the `results_nth.txt` file. It contains the execution time and a check for correctness. If correct==1, then the matrix has been correctly transposed.