int.ParallelComputingDeliverable1 – README PDF VERSION

here is the repository for int.PARCO H1/D1

You can find:

- some example of implementations of main (proj.cpp -> works for serial and implicitly parallelized solution, selects the option by reading if a -true is passed by the command line or not; OmpProj1 and OmpProj1_TEST both shows results calling the functions parallelized with OpenMp, but the one ending with test does it for more times in a row.

- directory for personal tests results and analysis attached to the document at https://drive.google.com/drive/folders/1XiznFH5u935mtcfHnHqpiyWMsDdW9wru?usp=drive_link

- official document IEEE also submitted via e-mail

-PBS files, that can be used in order to execute codes on the HPC cluster of UniTN in the same manner I did:

{

testp1.pbs is for executing proj1.cpp (serial part)

testpImp.pbs is for executing proj.cpp (implicitly parallelized part)

testp3.pbs is for executing OmpProj1.cpp more times with different numbers of threads

testp8.pbs and testpOmp.pbs do the same thing, executing parallel solutions with just 8 threads. One does just for one time, the other for more times.

!!! ATTENTION: Please change the final directory with your name and the address you keep the cpp program stored.

pbs files can be launched with qsub *name*.pbs and job status can be checked with qstat *name_account*

}

-header files (Do not change anything, excluded the size N)

The main can also be created autonomously by the user that can customize an execution for different needs.

be aware that:

-transposing more times the same matrix can give misleading results as data could already be in the cache or in faster memory.

-the program can run everywhere, but not every N for matrix size could work in every system (calculate memory usage by calculating N*N*4 bytes for each matrix present in the code).

- Timers are effective but do not perform well on very small amounts of time

-include all libraries reported

-take in consideration to follow the general idea that checkSym() should be used to avid matrix transposition, as a symmetric matrix is equal to its transposal. The recommended order is: create new Timer(), allocate and create matrix/ matrices, start timer, check symmetry, stop timer, print time, if symmetric you can end program, if not symmetric start timer again compute the transpose, stop timer, print time and check result, if check is correctly completed, store the transpose in a new matrix.