HPC Tools – Final Assignment

Thomas DUSSAUT – EISTI Pau – USC

GitHub : https://github.com/dussauttho/hpctools

The source code for the task3 solver is available here:
https://github.com/dussauttho/hpctools/blob/master/Task3_solver.c .

Section 0 – Valgrind

Valgrind found a total of 126.616 MB of memory leak (for a matrix size of 300x300), on 19 spots of the source code.
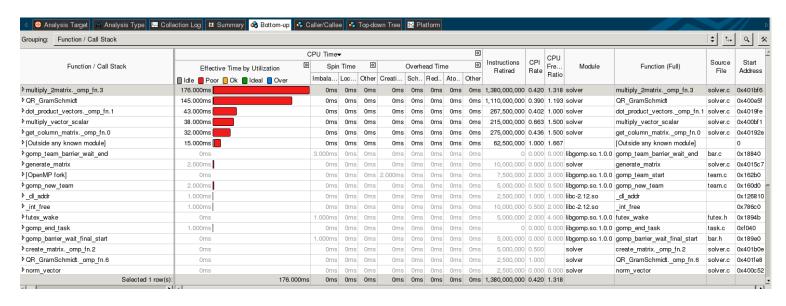
To reduce memory leaks, I modified the *get_column_matrix* function. Now, I pass the existing *double* v* as an argument of the new *get_column_matrix* and modify it directly in it, without creating a new vector and then return the new value. I applied the same principle everywhere else in the program in order to reduce memory leaks. Moreover, by freeing all the variables when they became useless everywhere in the program, I also reduced the memory leaks.

Section 1 – OpenMP parallelization

In order to parallelize the algorithm, I simply put "*#pragma omp parallel*" sections and "*#pragma omp for*" before each for loop that was suitable for parallelization. Here are the execution time and speed-up I got for a 20x20 matrix:

Number of processors	Execution Time	Speed-Up
1	0.135 sec	/
4	0.032 sec	4.22
8	0.014 sec	9.64

Section 2 – Intel VTune Amplifier

Like we can see on this screenshot, the two main hotspots are the *multiply_2matrix* and *QR_GramSchmidt* functions. With more than 1.3 million iterations, we can be surprised that the matrix product function is actually the most used function in the program, instead of the core function *QR_GramSchimdt.*

Section 3 – Intel Advisor

I wasn't able to run Intel Advisor because the software kept crashing on my personal computer, even if I was following the instructions on the tutorial video. When launching Advisor directly on ft6.cesga.es, I was kicked from the resources affected to my session after a few minutes, also causing Advisor to crash.

Section 4 – Extrae / Paraver

N/D