



HPC Tools

Final Assignment

Haytam EL MERABETI

HPC Master student

elmerabeti@cy-tech.fr

FACULTY OF COMPUTER SCIENCE

December 23, 2022

HPC Tools : Final Assignment

In the conclusion of my report in task 2, I spoke about how the O3 optimization flag was the most appropriate one for my code, paired with the profile guided optimization and some code modifications compared to task 1. In this final task, I will try to use different tools to analyze my code on different aspects (memory, parallelization, vectorization, cache, etc), while always measuring performance using the forementioned compiling configuration (O3 flag with PGO).

Memory :

Using Valgrind's Memcheck tool, I was able to detect some memory issues which led me to the modifications below :

- Freeing the space dynamically allocated to the pointer augMatrix.
 - This led me to realise that I didn't really need a dynamically allocated pointer, given that I already know the size that my array needs, so I changed augMatrix's type to a static double array of size $n*2*n$.
 - In the same time, I choose to use a static double array "ratio" instead of only a variable, in order to avoid concurrency.
- Adding several memory freeing functions to the main part of the code, although it is outside the scope of my implementation of dgesv.

These changes resulted in the following performance improvement (O3 + PGO):

```
[curso356@c206-1 HPCTools]$ ./dgesv_gcc 1024
Time taken by OpenBLAS LAPACK: 0.21s
Time taken by my sequential implementation: 0.40s
Result is ok!
[curso356@c206-1 HPCTools]$ ./dgesv_gcc 2048
Time taken by OpenBLAS LAPACK: 1.06s
Time taken by my sequential implementation: 4.63s
Result is ok!
[curso356@c206-1 HPCTools]$ ./dgesv_gcc 4096
Time taken by OpenBLAS LAPACK: 6.39s
Time taken by my sequential implementation: 55.06s
Result is ok!
```

```
[curso356@c206-1 HPCTools]$ ./dgesv_icc 1024
Time taken by OpenBLAS LAPACK: 0.12s
Time taken by my sequential implementation: 1.74s
Result is ok!
[curso356@c206-1 HPCTools]$ ./dgesv_icc 2048
Time taken by OpenBLAS LAPACK: 0.64s
Time taken by my sequential implementation: 6.12s
Result is ok!
[curso356@c206-1 HPCTools]$ ./dgesv_icc 4096
Time taken by OpenBLAS LAPACK: 4.05s
Time taken by my sequential implementation: 55.50s
Result is ok!
```

The final outputs of Memcheck for gcc and icc without compiler opts and with param 256 (to make it kind of faster...) are in the file valgrind_output_gcc.txt & valgrind_output_icc.txt .

The results show no “definitely lost” or “indirectly lost” leaks for both gcc and icc, and the remaining leaks are not linked to my implementation of dgesv.

Parallelization :

As parallelization wasn't the main goal of this final task, I only added two basic "#pragma omp for" directives in the following code parts :

- The gauss jordan elimination.
- The computation of the unique solution b.

This led to improving the performance as follows :

```
[curso356@c206-1 HPCTools]$ time ./dgesv_gcc 1024
Time taken by OpenBLAS LAPACK: 0.20s
Time taken by my sequential implementation: 0.37s
Result is ok!

real    0m0.452s
user    0m0.604s
sys      0m0.011s
[curso356@c206-1 HPCTools]$ time ./dgesv_gcc 2048
Time taken by OpenBLAS LAPACK: 1.05s
Time taken by my sequential implementation: 4.54s
Result is ok!

real    0m4.963s
user    0m5.703s
sys      0m0.037s
[curso356@c206-1 HPCTools]$ time ./dgesv_gcc 4096
Time taken by OpenBLAS LAPACK: 6.38s
Time taken by my sequential implementation: 54.51s
Result is ok!

real    0m56.686s
user    1m1.354s
sys      0m0.130s
```

```
[curso356@c206-1 HPCTools]$ time ./dgesv_gcc 1024
Time taken by OpenBLAS LAPACK: 0.22s
Time taken by my parallel implementation: 0.803214s
Result is ok!

real    0m0.190s
user    0m1.065s
sys      0m0.031s
[curso356@c206-1 HPCTools]$ time ./dgesv_gcc 2048
Time taken by OpenBLAS LAPACK: 1.07s
Time taken by my parallel implementation: 15.656177s
Result is ok!

real    0m2.459s
user    0m16.859s
sys      0m0.069s
[curso356@c206-1 HPCTools]$ time ./dgesv_gcc 4096
Time taken by OpenBLAS LAPACK: 6.45s
Time taken by my parallel implementation: 112.382234s
Result is ok!

real    0m16.386s
user    1m59.296s
sys      0m0.208s
```

```
[curso356@c206-1 HPCTools]$ time ./dgesv_icc 1024
Time taken by OpenBLAS LAPACK: 0.12s
Time taken by my sequential implementation: 1.85s
Result is ok!

real    0m0.559s
user    0m1.940s
sys      0m0.082s
[curso356@c206-1 HPCTools]$ time ./dgesv_icc 2048
Time taken by OpenBLAS LAPACK: 0.64s
Time taken by my sequential implementation: 6.36s
Result is ok!

real    0m5.325s
user    0m7.055s
sys      0m0.112s
[curso356@c206-1 HPCTools]$ time ./dgesv_icc 4096
Time taken by OpenBLAS LAPACK: 4.07s
Time taken by my sequential implementation: 55.43s
Result is ok!

real    0m55.440s
user    0m59.787s
sys      0m0.340s
```

```
[curso356@c206-1 HPCTools]$ time ./dgesv_icc 1024
Time taken by OpenBLAS LAPACK: 0.13s
Time taken by my parallel implementation: 0.803868s
Result is ok!

real    0m0.188s
user    0m0.972s
sys      0m0.041s
[curso356@c206-1 HPCTools]$ time ./dgesv_icc 2048
Time taken by OpenBLAS LAPACK: 0.64s
Time taken by my parallel implementation: 6.752788s
Result is ok!

real    0m1.117s
user    0m7.547s
sys      0m0.102s
[curso356@c206-1 HPCTools]$ time ./dgesv_icc 4096
Time taken by OpenBLAS LAPACK: 4.06s
Time taken by my parallel implementation: 98.328030s
Result is ok!

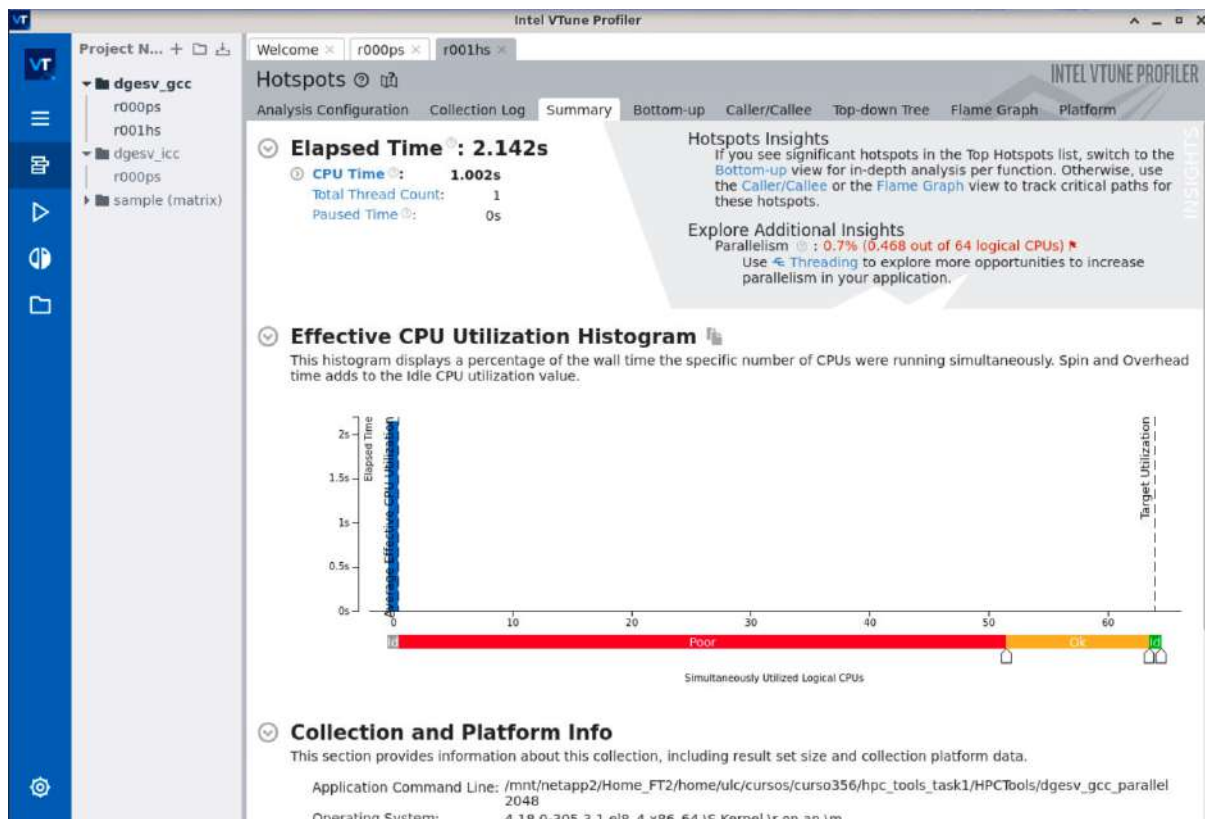
real    0m13.499s
user    1m43.005s
sys      0m0.338s
```

The following speedup values were calculated using 8 cores, and based on the “real” time of the function time, in order to have a common function for sequential and parallel code.

Speedups	gcc	icc
----------	-----	-----

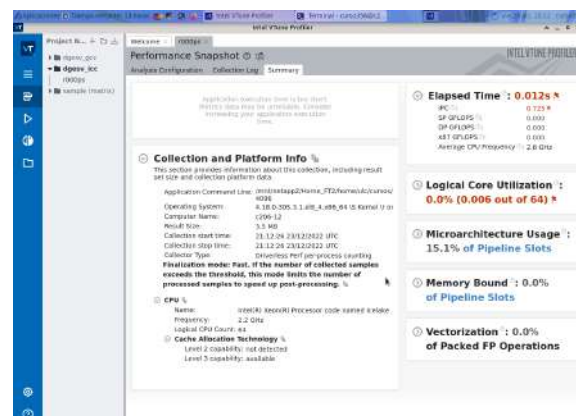
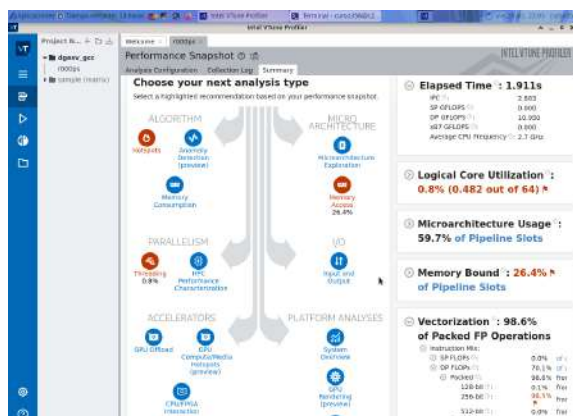
Speedups	gcc	icc
Small	2.38	2,97
Medium	2.02	4,77
Large	3.46	4,11

The output of Intel Vtune Amplifier is as follows :



Vectorization :

In this part, I used Intel Vtune Amplifier utility to further analyze my code, especially to know the status of vectorization in it.



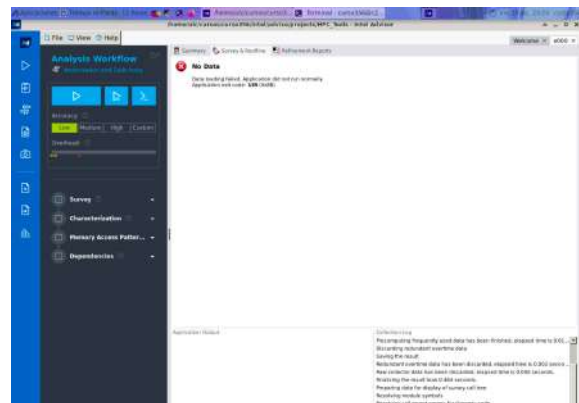
The results, as shown in the above image, present a score of nearly 99% in vectorization when compiling with gcc (icc version apparently had a short execution time, even when i used 8192 as param... so I couldn't get much info for this version).

I would consider that the good vectorization of gcc version is thanks to :

- The O3 flag.
- The use of the static arrays.

For some unknown reasons to me, I wasn't able to use Intel Advisor on my executable.

I had this screen :



Cache :

Using Valgrind's Cachegrind, I analysed my code to check whether there is any cache misses on the different levels. Using gcc, the output of Cachegrind is in the file Cachegrind_output.txt . I couldn't find any special improvement in my implementation.

Final performances :

Several tools and optimizations later, my program performs as follows (the given executables are the most optimized ones) :

- O3 with PGO and Parallelization :


```
[curso356@c206-1 HPCTools]$ time ./dgesv_gcc 1024
Time taken by OpenBLAS LAPACK: 0.22s
Time taken by my parallel implementation: 0.803214s
Result is ok!

real    0m0,190s
user    0m1,065s
sys      0m0,031s
[curso356@c206-1 HPCTools]$ time ./dgesv_gcc 2048
Time taken by OpenBLAS LAPACK: 1.07s
Time taken by my parallel implementation: 15.656177s
Result is ok!

real    0m2,459s
user    0m16,859s
sys      0m0,069s
[curso356@c206-1 HPCTools]$ time ./dgesv_gcc 4096
Time taken by OpenBLAS LAPACK: 6.45s
Time taken by my parallel implementation: 112.382234s
Result is ok!

real    0m16,386s
user    1m59,296s
sys      0m0,208s
```

```
[curso356@c206-1 HPCTools]$ time ./dgesv_icc 1024
Time taken by OpenBLAS LAPACK: 0.13s
Time taken by my parallel implementation: 0.803868s
Result is ok!

real    0m0,188s
user    0m0,972s
sys      0m0,041s
[curso356@c206-1 HPCTools]$ time ./dgesv_icc 2048
Time taken by OpenBLAS LAPACK: 0.64s
Time taken by my parallel implementation: 6.752788s
Result is ok!

real    0m1,117s
user    0m7,547s
sys      0m0,102s
[curso356@c206-1 HPCTools]$ time ./dgesv_icc 4096
Time taken by OpenBLAS LAPACK: 4.06s
Time taken by my parallel implementation: 98.328030s
Result is ok!

real    0m13,499s
user    1m43,005s
sys      0m0,338s
```

- O3 with PGO without Parallelization :

```
[curso356@c206-1 HPCTools]$ time ./dgesv_gcc 1024
Time taken by OpenBLAS LAPACK: 0.21s
Time taken by my sequential implementation: 0.40s
Result is ok!

real    0m0,487s
user    0m0,634s
sys      0m0,020s
[curso356@c206-1 HPCTools]$ time ./dgesv_gcc 2048
Time taken by OpenBLAS LAPACK: 1.06s
Time taken by my sequential implementation: 4.63s
Result is ok!

real    0m5,070s
user    0m5,797s
sys      0m0,048s
[curso356@c206-1 HPCTools]$ time ./dgesv_gcc 4096
Time taken by OpenBLAS LAPACK: 6.37s
Time taken by my sequential implementation: 55.63s
Result is ok!

real    0m57,819s
user    1m2,465s
sys      0m0,136s
```

```
[curso356@c206-1 HPCTools]$ time ./dgesv_icc 1024
Time taken by OpenBLAS LAPACK: 0.12s
Time taken by my sequential implementation: 1.85s
Result is ok!

real    0m0,588s
user    0m1,926s
sys      0m0,094s
[curso356@c206-1 HPCTools]$ time ./dgesv_icc 2048
Time taken by OpenBLAS LAPACK: 0.63s
Time taken by my sequential implementation: 6.41s
Result is ok!

real    0m5,385s
user    0m7,082s
sys      0m0,133s
[curso356@c206-1 HPCTools]$ time ./dgesv_icc 4096
Time taken by OpenBLAS LAPACK: 4.07s
Time taken by my sequential implementation: 56.22s
Result is ok!

real    0m56,279s
user    1m0,554s
sys      0m0,362s
```

While I am sure I could further improve my code and optimize it, especially through a better use of the remaining tools that we discovered through this course, I am satisfied by what I was able to accomplish so far.