

# Software and Programming of High Performance Systems

## 3rd Exercise

Papageorgiou Spyridon

Maximos Frountzos

The experiments were carried out in the Krylov system. It is a server with 24 Intel Xeon cores, 128 GB RAM, and a single NVIDIA Tesla V100 GPU (5120 CUDA cores, 32 GB of graphics memory).

### Question 1 (OpenMP & CMM)

b)

The basic elements of the implementation are as follows. First, the input matrices (A, B, C, D) are defined on the host where they are initialized (in a deterministic way for debugging) with the function ***fill\_matrix***. Then the function ***gpu\_complex\_matrix\_multiply*** is called which defines the intermediate calculation matrices (AC, BD, AD, BC) and calculates them with the function ***gpu\_matrix\_multiply***. Then the matrices E and F are calculated with the functions ***gpu\_matrix\_subtract*** and ***gpu\_matrix\_add*** respectively.

The ***gpu*** functions (of type  $C = A \text{ op } B$ ) are the same as those for serial execution on CPUs, except that OpenMP directives are used for execution on GPUs. Initially, to limit data transfers between the host and the device, the ***data map to*** directive was used for the matrices A and B and the ***data map from*** directive was used for the matrix C. Then for the loops, we used the directives ***teams***(like CUDA blocks), ***distribute***(iteration distribution to the teams), ***parallel for***(parallel execution in each team) and ***collapse(2)***(combine 2 for loop into one for better parallelization).

c)

In order to compare performances, a serialized version of the problem was implemented on a CPU that follows the same logical order. A `get_wtime` function was also used for timekeeping. The following variables are used in the main function to execute the code with different options : *singleExec*, *maxExp*, *minExp*, *cpuExec*, *verbose*, *checkResults*. If *singleExec* == 1 then the calculation is done only once and in the way defined by *cpuExec*, i.e. if *cpuExec* == 1 then the CPU implementation is executed, while if it is 0 the GPU implementation is executed. In case *singleExec* == 0 then both implementations are executed for problem size  $N = 2^{\text{minExp}}$  to  $N = 2^{(\text{maxExp} - 1)}$  and if *checkResults* == 1 then it is checked if the results are the same. If *verbose* == 1 then only in the case of the single execution are the results displayed.

Below are the results for *singleExec* == 0, *minExp* == 1, *maxExp* == 11 and *checkResults* == 1.

```
CPU and GPU outputs are equal for N=2
CPU and GPU outputs are equal for N=4
CPU and GPU outputs are equal for N=8
CPU and GPU outputs are equal for N=16
CPU and GPU outputs are equal for N=32
CPU and GPU outputs are equal for N=64
CPU and GPU outputs are equal for N=128
CPU and GPU outputs are equal for N=256
CPU and GPU outputs are equal for N=512
CPU and GPU outputs are equal for N=1024
```

# Software and Programming of High Performance Systems

## 3rd Exercise

Papageorgiou Spyridon

Maximos Frountzos

Below are the results for *singleExec* == 0, *minExp* == 10 and *maxExp* == 13 and *checkResults* == 0.

```
running on CPU
  CPU finished with computation time:47.9939seconds for N=1024
running on GPU
  GPU finished with computation time:1.43835seconds for N=1024
running on CPU
  CPU finished with computation time:936.011seconds for N=2048
running on GPU
  GPU finished with computation time:13.2367seconds for N=2048
running on CPU
  CPU finished with computation time:8574.93seconds for N=4096
running on GPU
  GPU finished with computation time:109.593seconds for N=4096
```

Below are the results for *singleExec* == 0, *minExp* == 2 and *maxExp* == 11 and *checkResults* == 0.

```
running on CPU
  CPU finished with computation time:1.40462e-05seconds for N=4
running on GPU
  GPU finished with computation time:0.170745seconds for N=4
running on CPU
  CPU finished with computation time:1.35032e-05seconds for N=8
running on GPU
  GPU finished with computation time:0.00422771seconds for N=8
running on CPU
  CPU finished with computation time:9.69223e-05seconds for N=16
running on GPU
  GPU finished with computation time:0.00624151seconds for N=16
running on CPU
  CPU finished with computation time:0.000675713seconds for N=32
running on GPU
  GPU finished with computation time:0.00637013seconds for N=32
running on CPU
  CPU finished with computation time:0.00527554seconds for N=64
running on GPU
  GPU finished with computation time:0.00658512seconds for N=64
running on CPU
  CPU finished with computation time:0.0446089seconds for N=128
running on GPU
  GPU finished with computation time:0.00956931seconds for N=128
running on CPU
  CPU finished with computation time:0.397204seconds for N=256
running on GPU
  GPU finished with computation time:0.0334677seconds for N=256
running on CPU
  CPU finished with computation time:4.46371seconds for N=512
running on GPU
  GPU finished with computation time:0.142632seconds for N=512
running on CPU
  CPU finished with computation time:49.1602seconds for N=1024
running on GPU
  GPU finished with computation time:1.30635seconds for N=1024
```

We notice that for small values of N the execution on a CPU is much faster than on a GPU, which is due to the transfer of data for processing to the GPU. The larger the N, the more the previous behavior is compensated, since now the cost of data transfer is no longer as significant compared to that of the operations that need to be done.