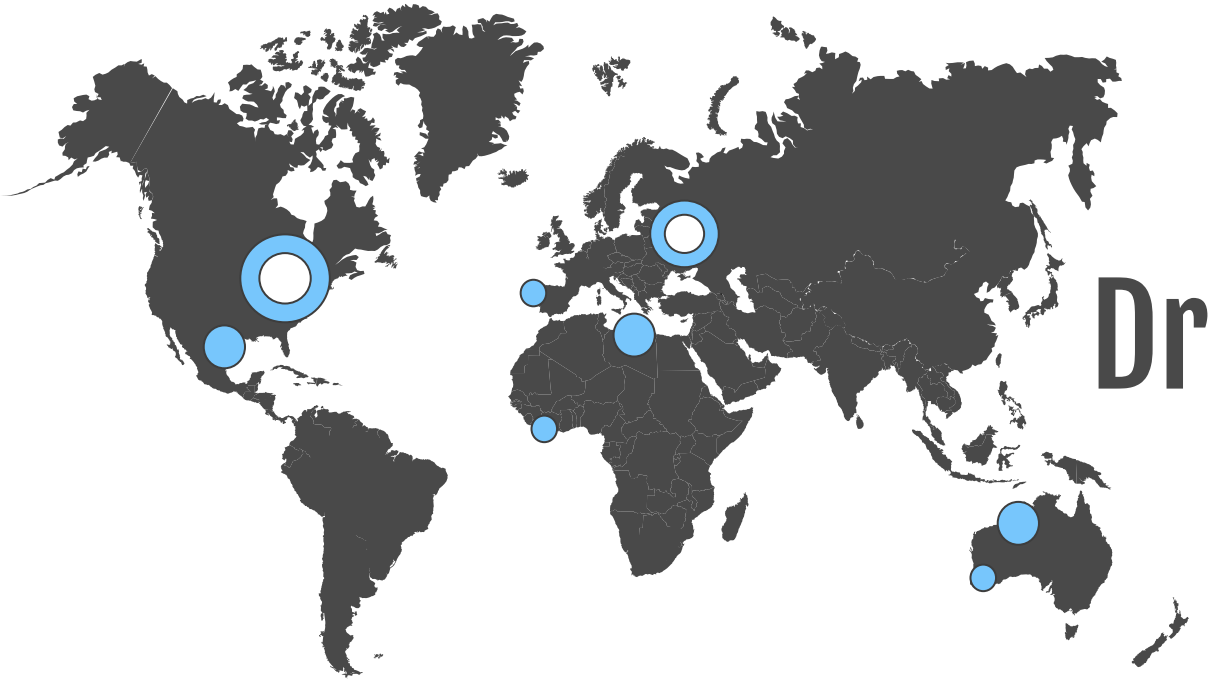


Drastic Event Simulator

Mirco Concu
Edoardo Pantè
Antonino Nigro

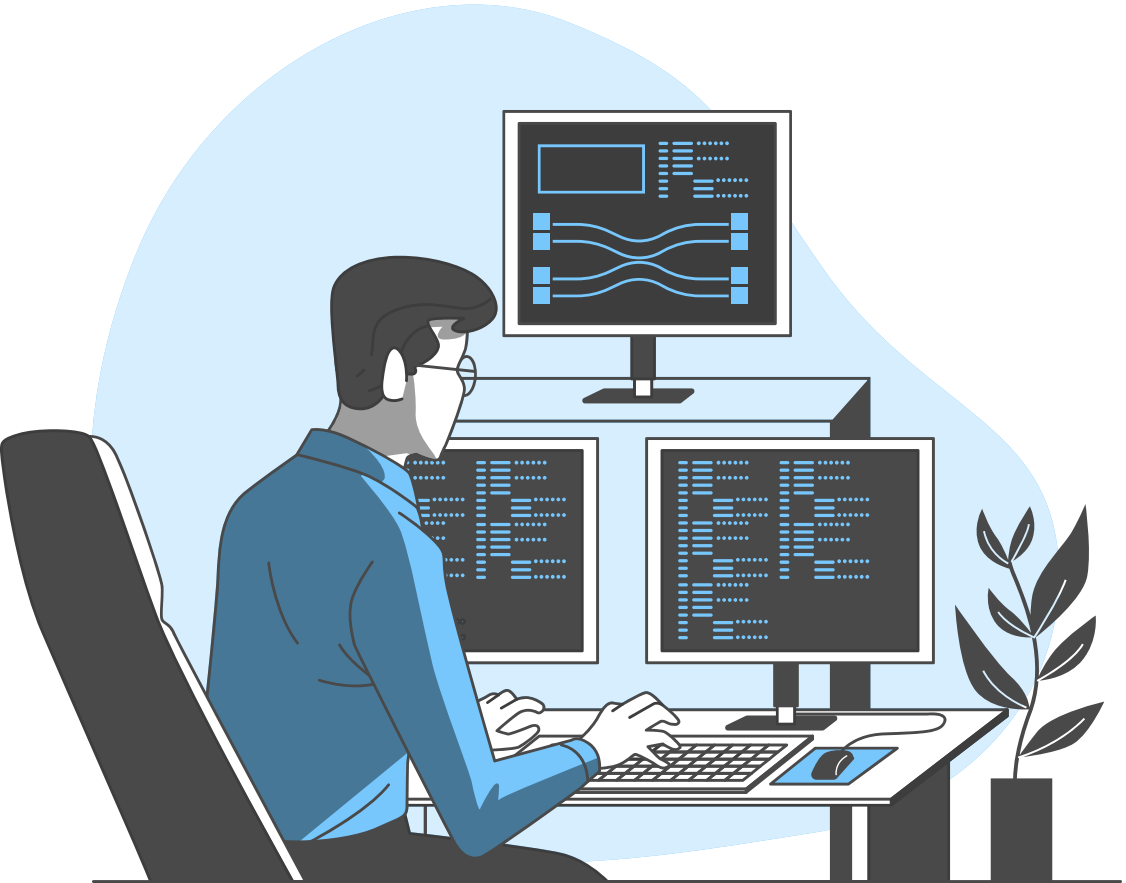
Input & Output Data

1	9	9	6	2	9	5	8	6	8	3	9	9	4	8
9	5	4	8	1	3	3	9	6	8	2	1	1	7	4
4	7	3	3	2	7	3	8	4	8	8	8	9	7	1
6	7	5	1	6	5	6	9	7	1	7	9	4	9	6
7	3	4	2	7	7	8	9	7	1	8	4	1	6	9
1	3	8	8	5	6	4	9	5	9	3	4	1	8	2
8	4	6	1	7	2	9	6	2	6	9	9	9	9	6
9	9	8	9	8	3	7	-	4	3	3	8	6	4	5
9	3	1	5	6	7	9	5	3	2	2	3	1	2	1
6	1	1	6	2	9	8	8	9	3	3	2	1	9	7
7	9	9	7	6	7	6	6	1	9	7	2	2	9	3
2	7	5	4	2	9	2	1	8	3	3	9	7	5	9
5	2	9	6	9	7	2	5	2	2	4	1	5	7	9
9	9	7	4	4	8	2	5	9	9	9	5	1	6	9
9	9	2	1	6	2	8	7	8	1	1	4	1	7	9

Terrain before the event

1	0	0	1	1	0	1	1	1	1	1	0	0	1	0
0	1	1	1	2	2	2	0	1	1	2	2	2	1	1
1	1	1	2	2	1	2	1	2	1	1	1	0	1	1
1	1	1	2	1	2	2	1	2	3	1	1	1	0	1
1	2	2	2	2	2	2	2	2	4	1	2	2	1	0
2	2	1	1	3	4	5	2	5	2	4	2	2	1	1
1	1	1	3	2	6	5	9	9	4	2	1	1	1	1
0	1	1	1	2	6	8	0	9	6	4	1	1	1	1
0	2	2	2	3	4	6	9	9	6	4	3	2	2	2
1	2	2	2	4	3	3	3	2	6	4	3	2	0	1
1	1	1	1	2	2	3	3	4	1	2	2	2	1	1
1	1	1	2	2	1	3	3	1	3	2	1	1	1	0
1	1	0	1	1	1	2	2	2	2	2	2	1	1	0
0	0	1	1	1	1	2	1	0	1	0	1	2	1	0
0	0	1	1	1	1	1	1	1	2	2	1	1	0	0

Damage on the terain



The CPU

The Hardware

01

CPU

M1 Pro – **8** Core – **8** Thread

- **6** *performance* @ **3.22** GHz
- **2** *efficiency* @ **2.06** GHz

02

Performance Core Cache

- L1*i* - **192** KB
- L1*d* - **128** KB
- L2 *Shared* - **28** MB
- L3 - **16** MB

03

Efficiency Core Cache

- L1*i* - **128** KB
- L1*d* - **64** KB
- L2 *Shared* - **4** MB
- L3 - **16** MB

What We Obtain

01 High Scalability

Performance increase **proportionaly** with the number of cores

02 Max Performance

Using **Sequential** approach no margin of improvement

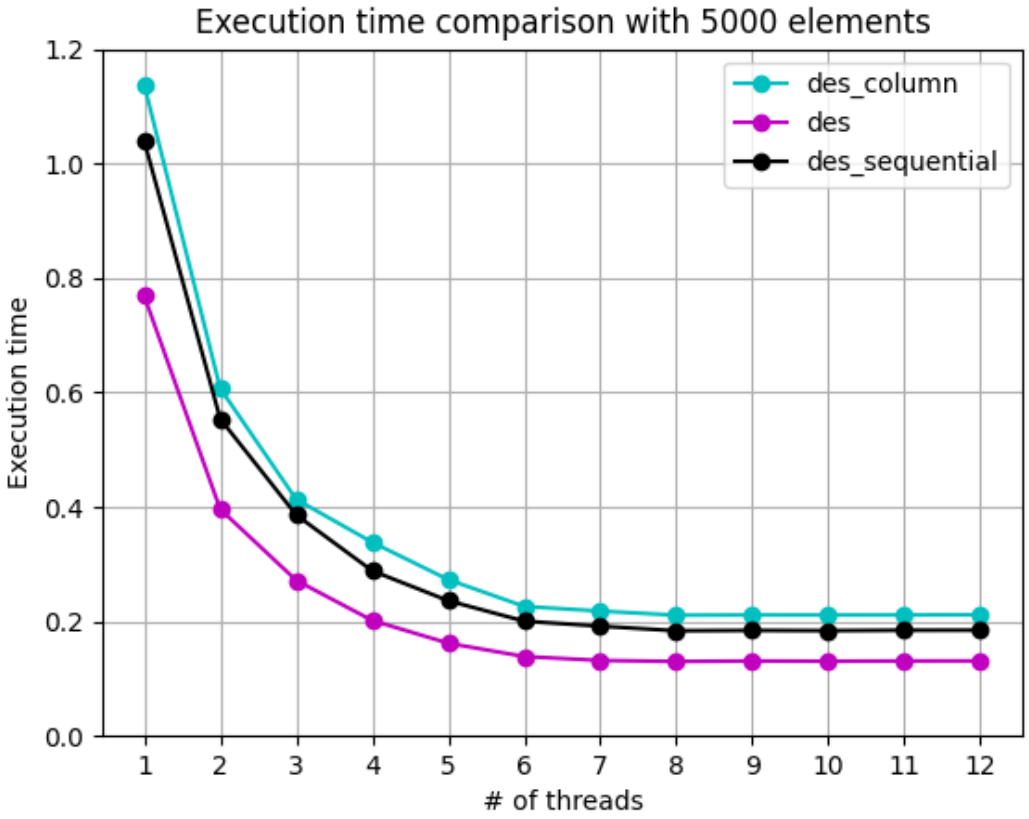
03 Power Consumption

Good energetic efficiency

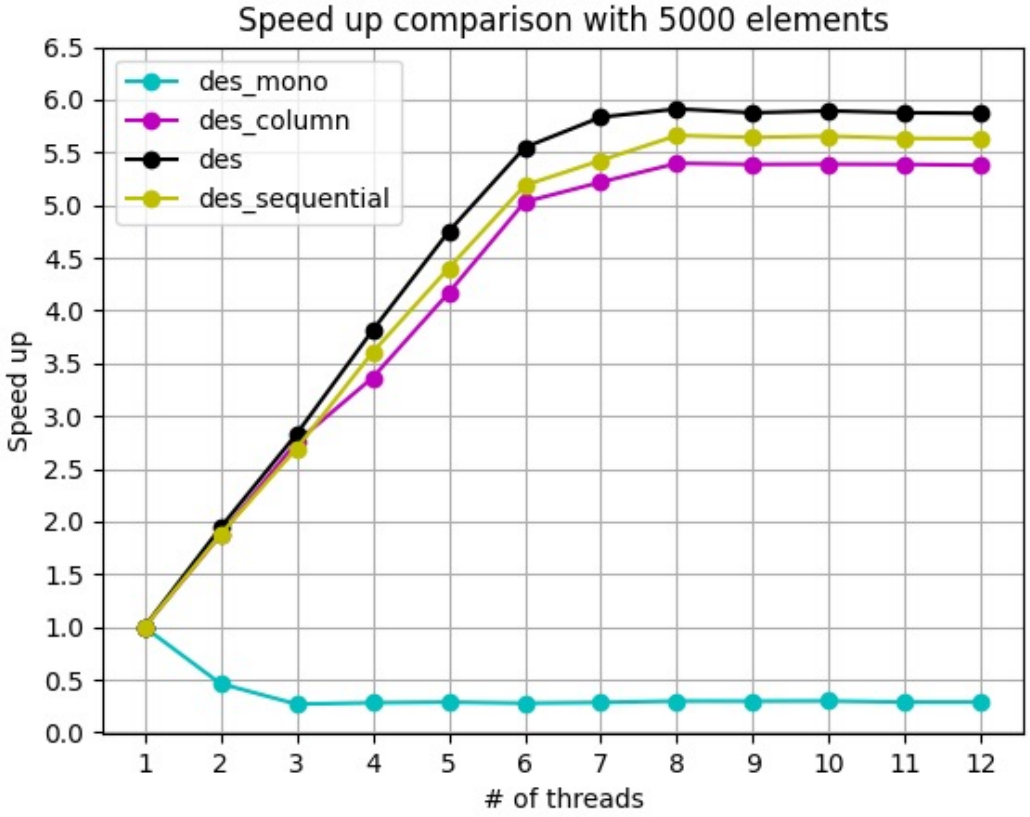
About Time Measurement

```
steady_clock::time_point begin = steady_clock::now();

for (int i = 0; i < activated_threads;i++)
{
    threads[i] = thread(simulate,&sim);
}
for(auto& t : threads){
    t.join();
}

steady_clock::time_point end = steady_clock::now();
double interval = duration_cast<microseconds>(end - begin).count();
```

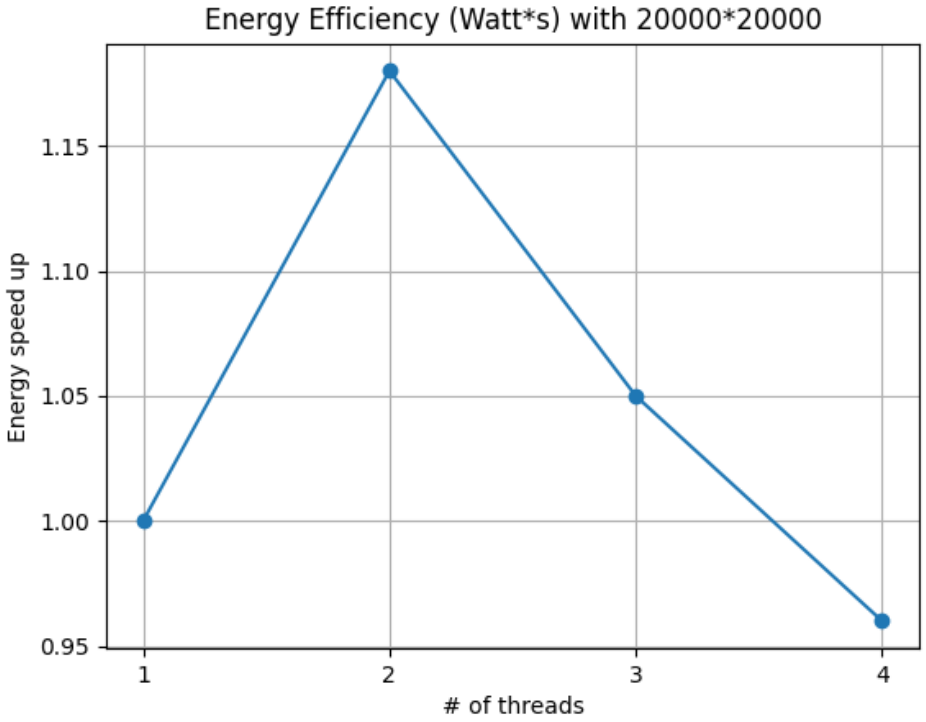
Execution Time



Speed Up

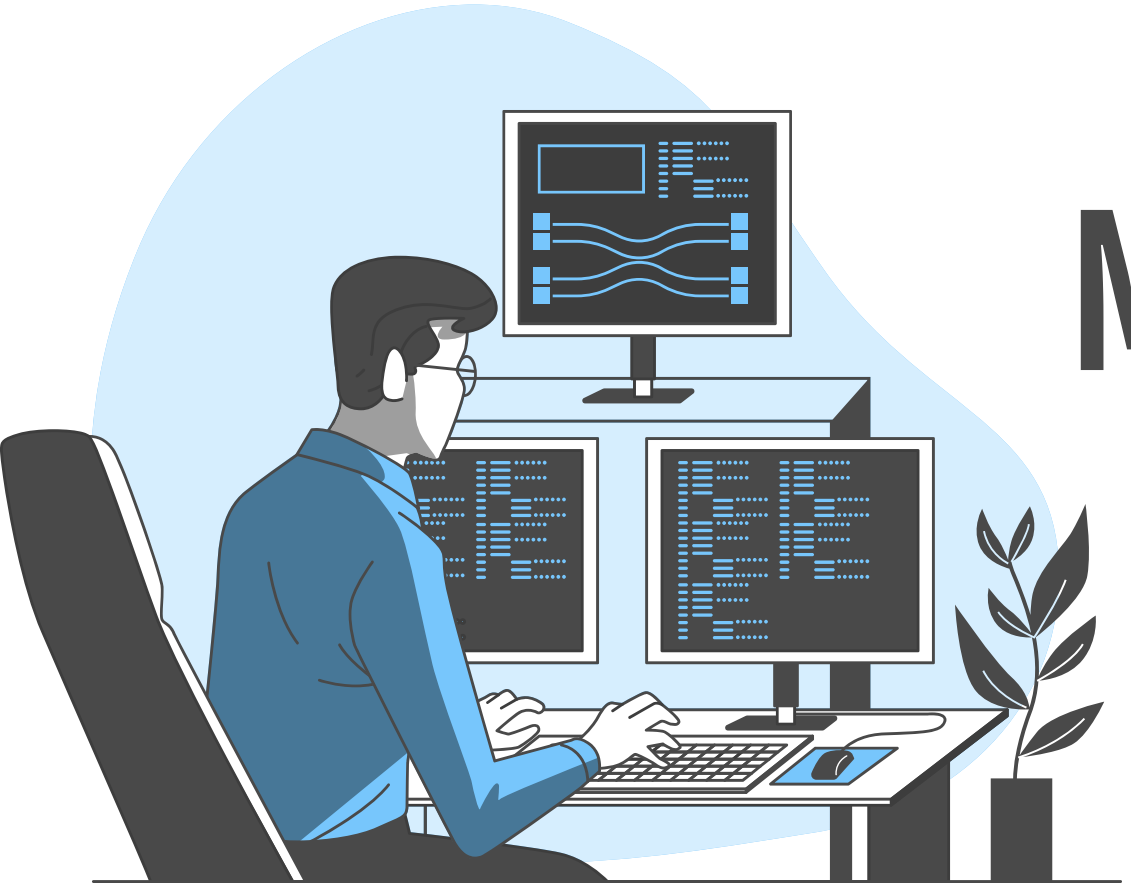

Speed up comparison with 5000 elements
des_mono
des_column
des
des_sequential
Speed up
# of threads
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
1
2
3
4
5
6
7
8
9
10
11
12

Power Consumption



Hardware:
i5-4790K – 4 Core

Software:
- CoreTemp 1.18
- HWMonitor



Moving to the GPU

The Hardware

```
Device 0: "Tesla T4"
  CUDA Driver Version / Runtime Version          12.1 / 12.1
  CUDA Capability Major/Minor version number:    7.5
  Total amount of global memory:                 15984 MBytes (16760700928 bytes)
  (040) Multiprocessors, (064) CUDA Cores/MP:    2560 CUDA Cores
  GPU Max Clock rate:                            1590 MHz (1.59 GHz)
  Memory Clock rate:                             5001 Mhz
  Memory Bus Width:                              256-bit
  L2 Cache Size:                                 4194304 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total shared memory per multiprocessor:        65536 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  1024
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 3 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Managed Memory:                Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 0 / 5
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 12.1, CUDA Runtime Version = 12.1, NumDevs = 1
Result = PASS
```

Time measurements

```
cudaEventRecord(start_simulation);
DES(copy_map,num_thread,num_blocchi);
cudaEventRecord(stop_simulation);
```

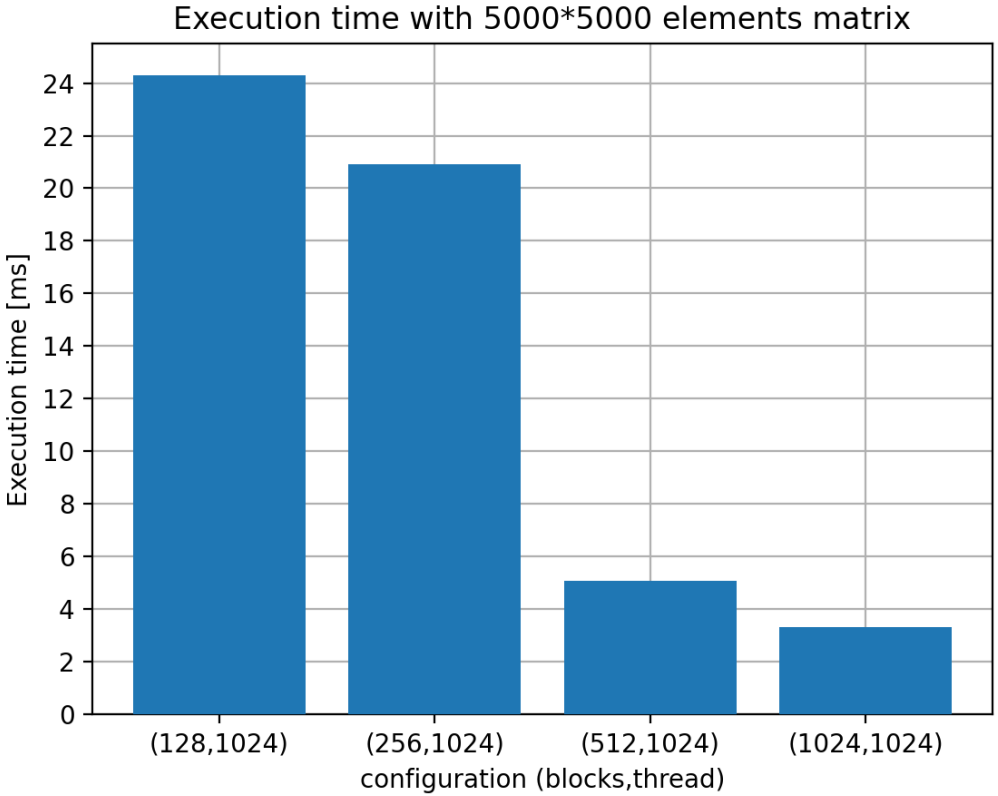
```
steady_clock::time_point begin = steady_clock::now();
cudaMemcpy(copy_map, sim.map, size_map, cudaMemcpyHostToDevice);
steady_clock::time_point end = steady_clock::now();
```

Type of measurements

- Total execution time
- Copy time
- Simulation time

```
float milliseconds = 0;
cudaEventElapsedTime(&milliseconds, start, stop);
cout << endl << setprecision(7) << "total simulation time " << milliseconds/pow(10,3) << endl << endl;


cudaEventElapsedTime(&milliseconds, start_simulation, stop_simulation);
cout << endl << setprecision(7) << "simulation time " << milliseconds/pow(10,3) << endl << endl;

f <<  milliseconds/pow(10,3)<<endl;

cout << endl << " copia: " << duration_cast<microseconds>(end - begin).count() / pow(10, 6) << endl << endl;
```

Execution time

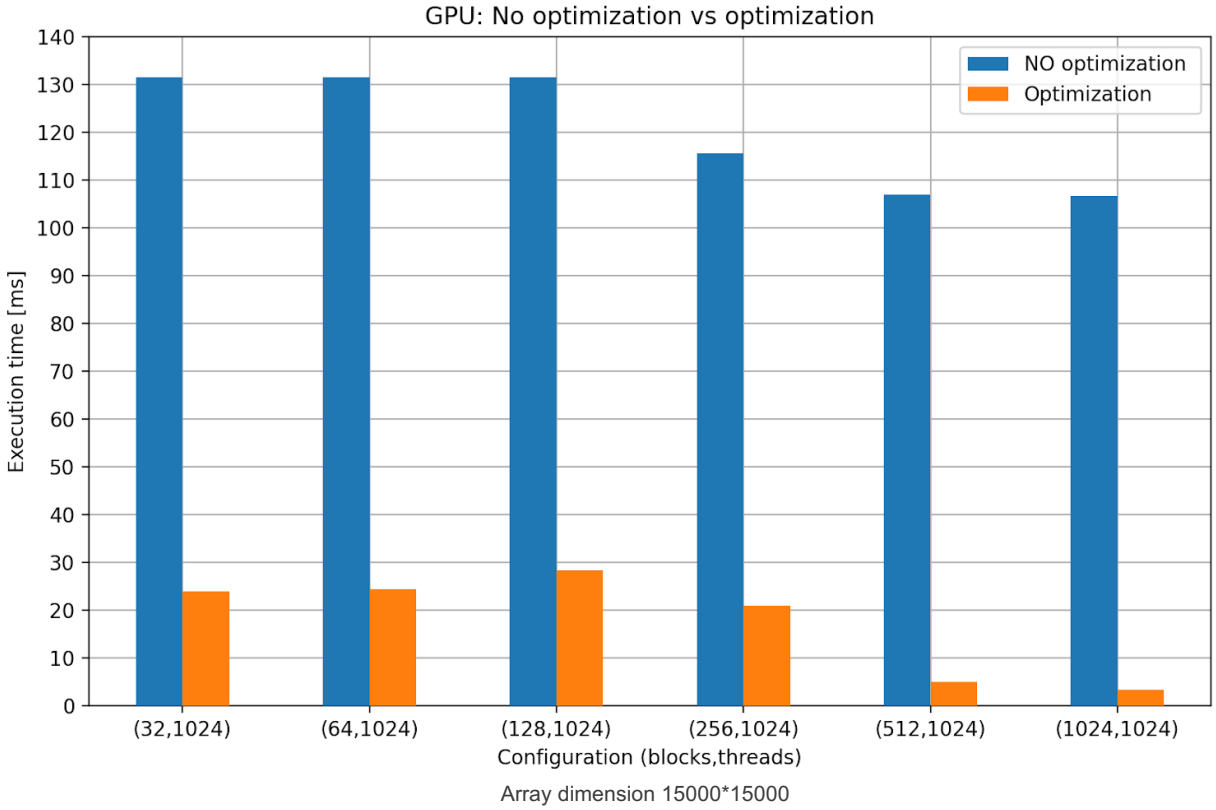


Optimization

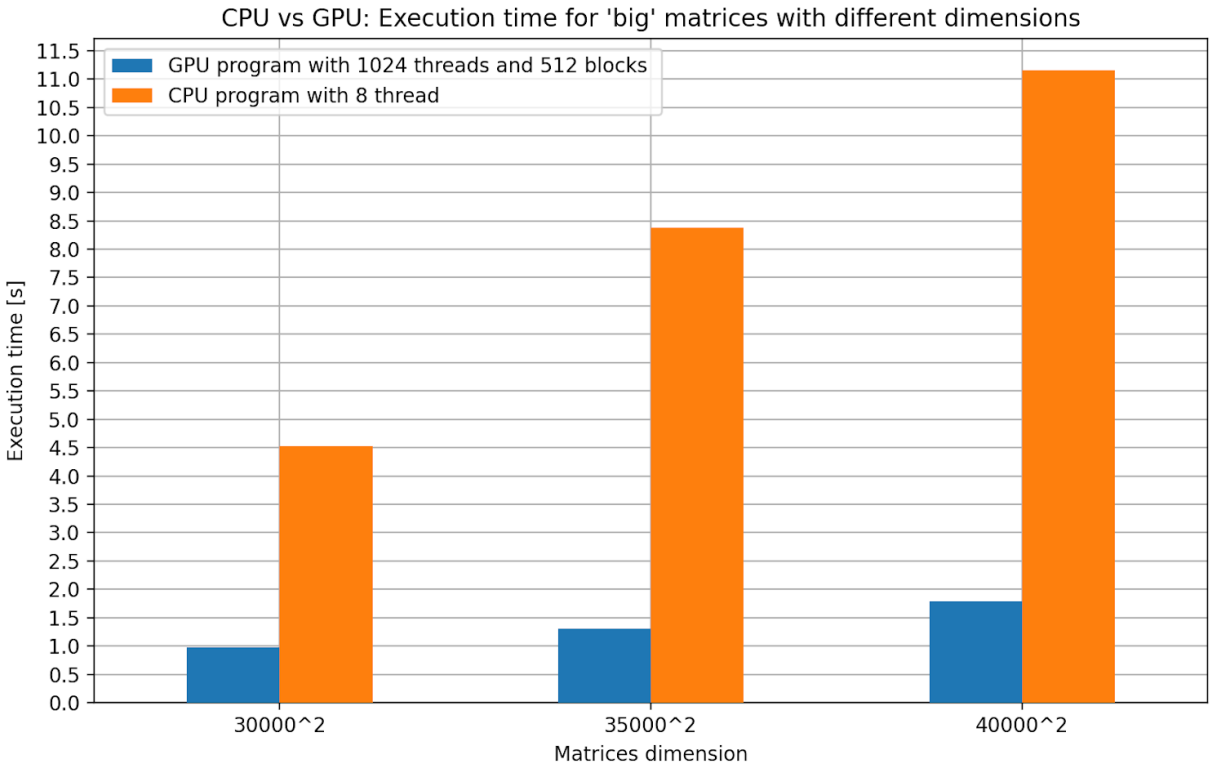
What the profiler showed:

- High throughput (85%)
- Low IPC (1 istruction / 36.7 cycles)
- Low use of fused instruction

Optimized vs non optimized



CPU vs GPU


CPU vs GPU: Execution time for 'big' matrices with different dimensions
GPU program with 1024 threads and 512 blocks
CPU program with 8 thread
Execution time [s]
11.5
11.0
10.5
10.0
9.5
9.0
8.5
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
30000^2
35000^2
40000^2
Matrices dimension

In conclusion

We can assert:

- With big matrix is convinient to move to a GPU implementation.
- With small matrix the margin of improvement between the CPU and the GPU is negligible.