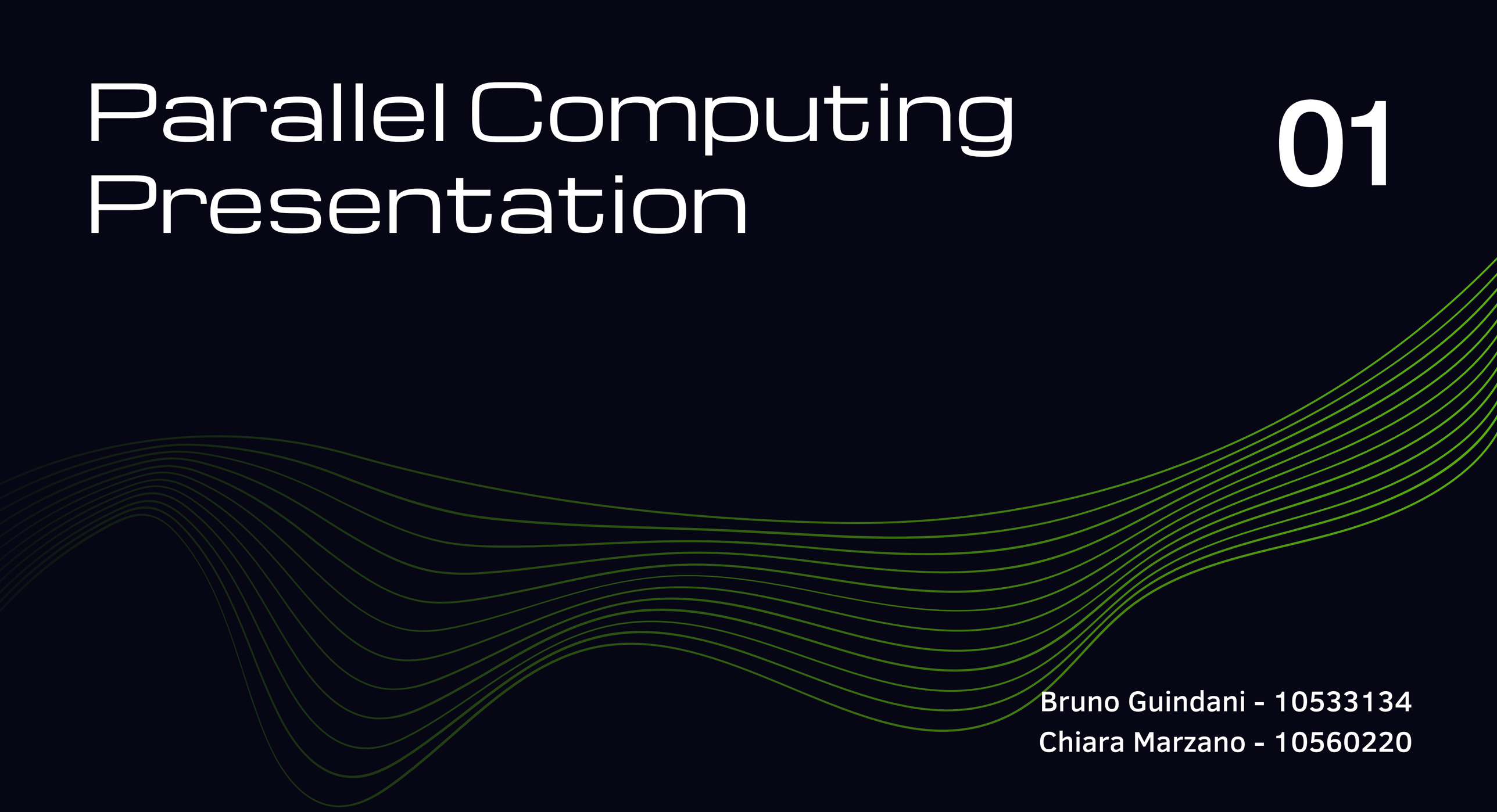
Parallel Computing Presentation

01



Bruno Guindani - 10533134
Chiara Marzano - 10560220

Agenda

02

ORIGINAL PROBLEM

Assessment of the original problem, profiling and measurements

TECHNOLOGY CHOICES

CUDA for GPGPU

IMPLEMENTATION

Where and how we implemented parallelism

PERFORMANCE

Evaluation of our solution

NUMERICAL RESULTS

Comparison with the original

ORIGINAL PROBLEM

03

Original Problem

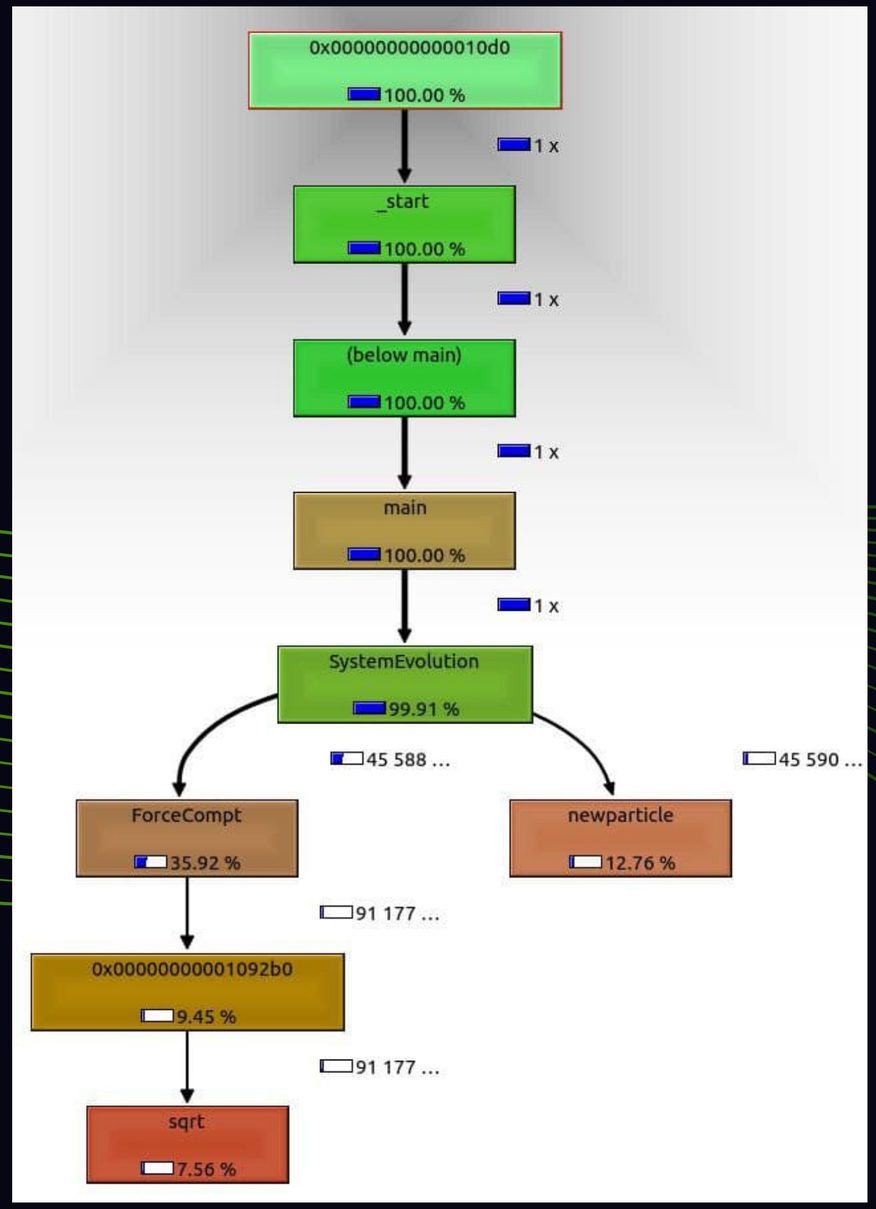
Interaction of particles in space

GENERATING FIELD

PARTICLE GENERATION

SYSTEM EVOLUTION

04



Performance Assessment

using KCacheGrind

OUR SOLUTION

Technology choice

CUDA for GPGPU

FINE-GRAINED CONTROL

EDUCATIONAL CHALLENGE

HIGHLY SUITABLE
APPLICATION DOMAIN

07

Implementation

Selecting functions to parallelize

Our first focus was deciding what to execute GPU-side and CPU-side. The decision was mainly based on:

- bottlenecks
- highly-parallelizable and concurrency-free operations
- educational impact

08

Implementation

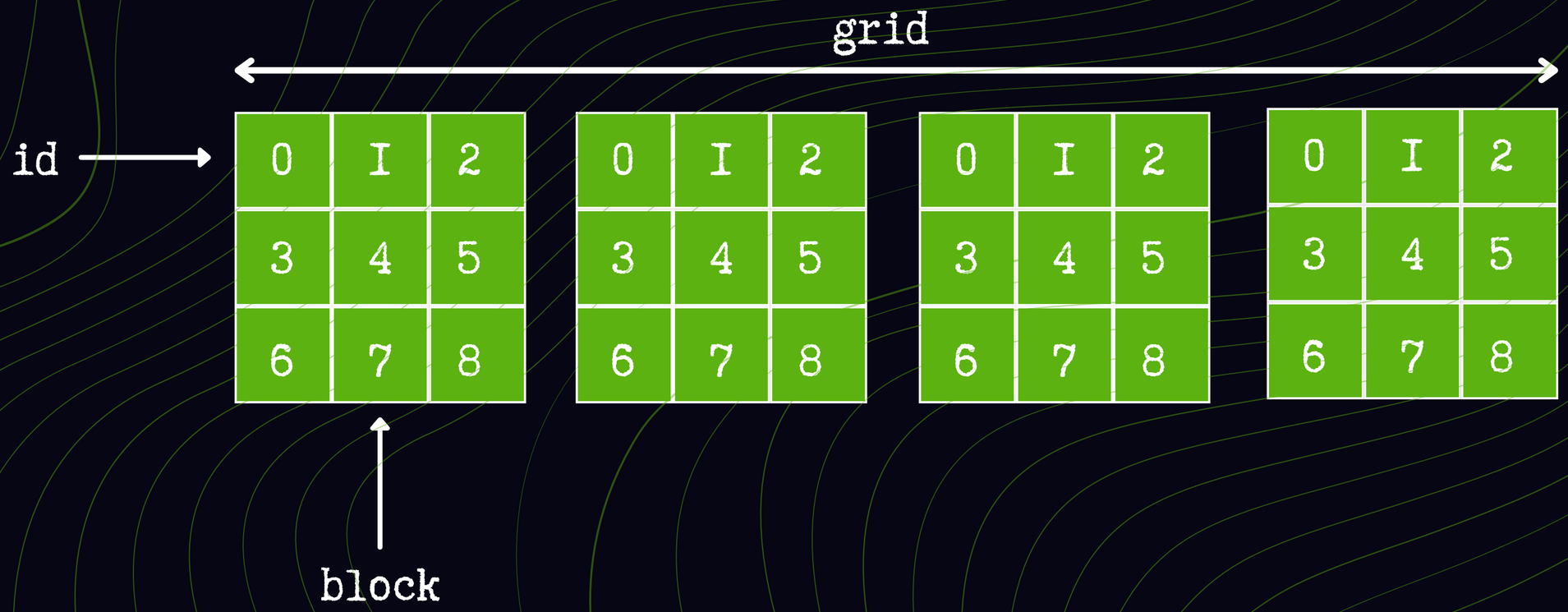
Techniques

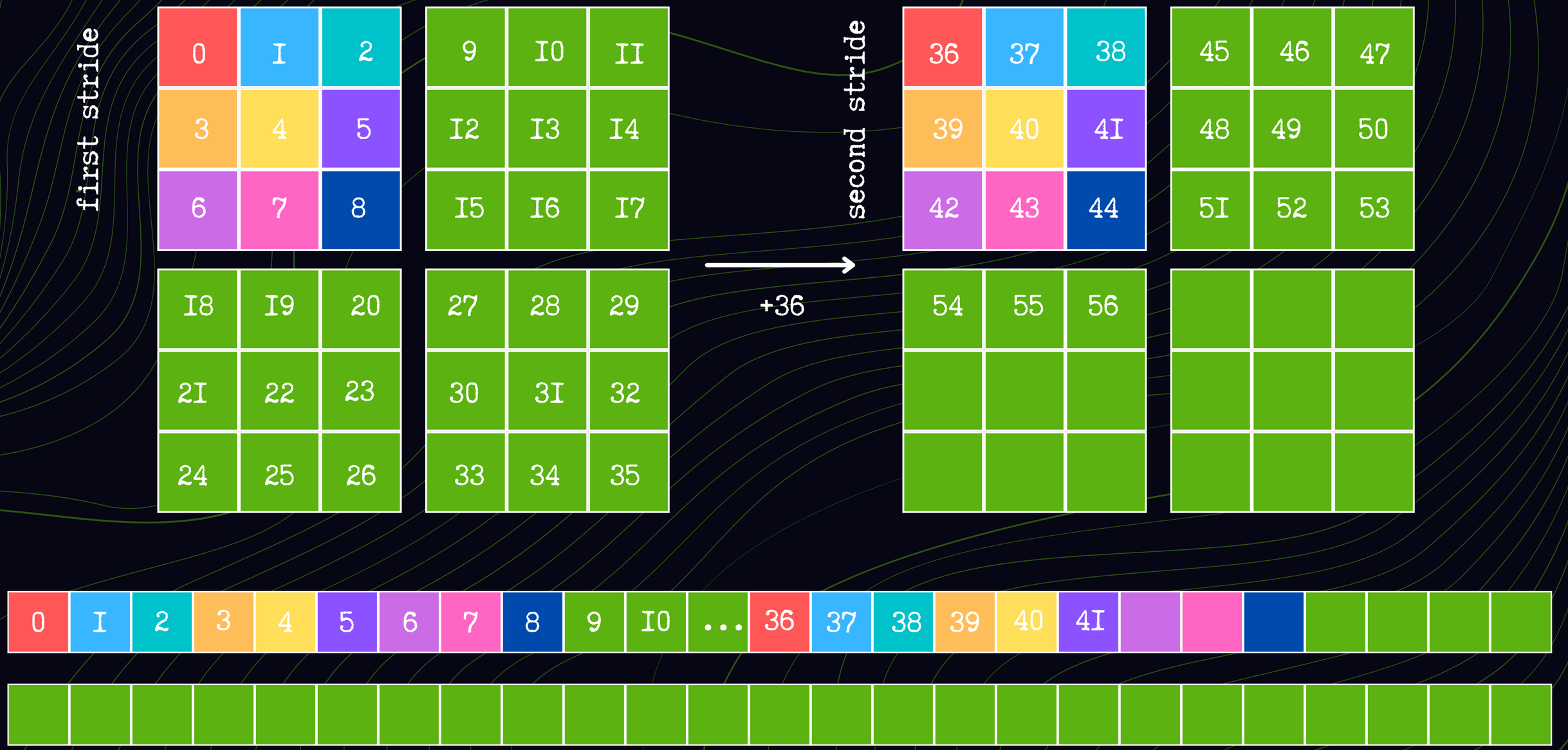
- mono and bidimensional stride
- shared memory

09

Implementation: stride technique

```
idx = blockIdx.x * blockDim.x + threadIdx.x;
stridex = gridDim.x * blockDim.x;
```




first stride
0 I 2
3 4 5
6 7 8
9 I0 II
I2 I3 I4
I5 I6 I7
I8 I9 20
2I 22 23
24 25 26
27 28 29
30 3I 32
33 34 35
second stride
36 37 38
39 40 4I
42 43 44
45 46 47
48 49 50
5I 52 53
54 55 56
+36
0 I 2 3 4 5 6 7 8 9 I0 ... 36 37 38 39 40 4I

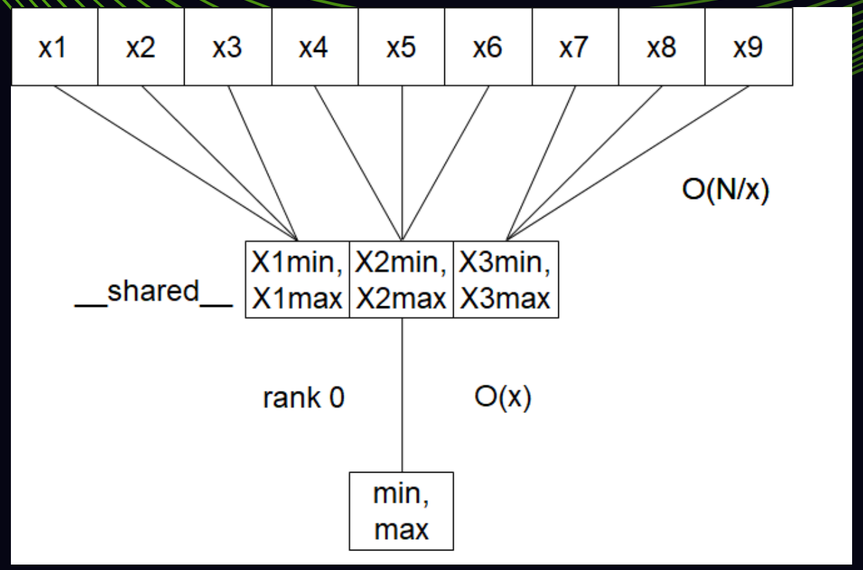
11

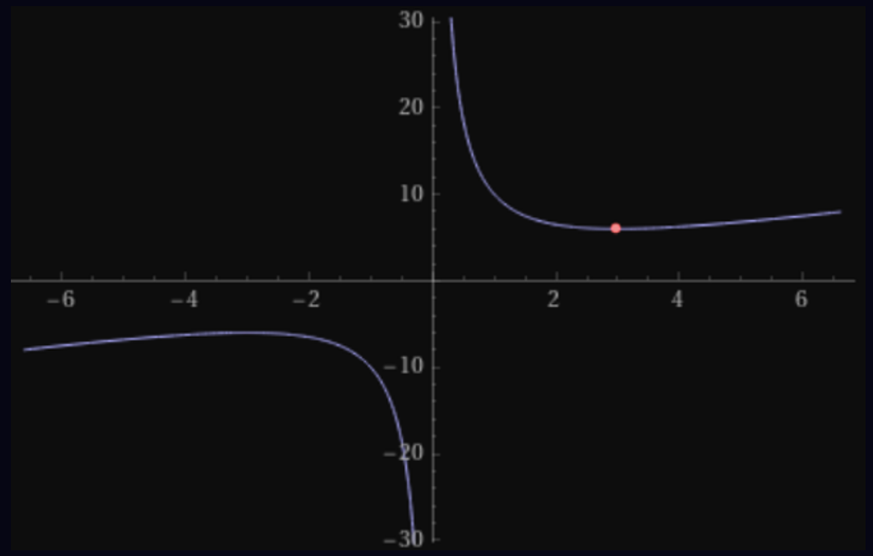
Generating Field

- Computes the generating field grid values in all points of the defined space
- Deterministic generation
- EX * EY independent iterations

- Bidimensional stride technique
- Bidimensional threads per block: 32, 32
- Bidimensional number of blocks: 32 * num_SMs, 32 * num_SMs

Minimum and maximum





- Joint MinMaxInt() and MinMaxDouble() functions
- Single block, shared memory
- Optimal number of threads: argmin f(x) = x + N/x
  → x = sqrt(N) (capped at 1024)

System Instant Evolution

- Executes a single evolution iteration
- NP independent iterations

- Monodimensional stride technique
- Monodimensional threads per block: 1024
- Monodimensional number of blocks: 32 * num_SMs

14

InitializeEmptyGrid

- Initializes empty particle grid
- EX * EY independent iterations
- Bidimensional stride technique

15

Compt Population

- Computes effects of forces on particles
- NP independent iterations
- Monodimensional stride technique

16

Performance

data collected over 10 runs, in microseconds

1200 * 1000

Sequential
- average: 234'457'848
- standard deviation: 2'478'229

Parallel
- average: 11'110'354
- standard deviation: 110'340

Speedup: 20x

2400 * 2000

Sequential
- average: 895'111'148
- standard deviation: 26'741'978

Parallel
- average: 42'099'243
- standard deviation: 195'185

Speedup: 20x

4800 * 4000

Sequential (one iteration)
- average: 373'628'321
- standard deviation: 3'001'303

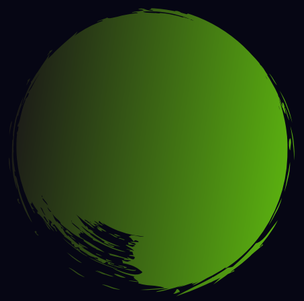
Parallel (one iteration)
- average 2'189'173
- standard deviation: 20068

Speedup: 170x

17

Numerical Results

- Sometimes, the populations generated by the sequential and parallel programs are identical except for a single particle



particle number 214
sequential weight: 1650
parallel weight: 1950

THANK YOU FOR YOUR ATTENTION

Bruno Guindani - 10533134
Chiara Marzano - 10560220

20 Compiling and running

REPOSITORY

Can be cloned at https://github.com/ChiaraMarzano/ParComProject

COMPILING

```
nvcc -arch=sm_70 -o particles.x particles_c.cu
```

RUNNING

```
./particles.x
```