Performance Analysis and Optimization of a Parallel GPU-accelerated Low-Temperature 2D Particle-in-Cell Plasma Simulation Code

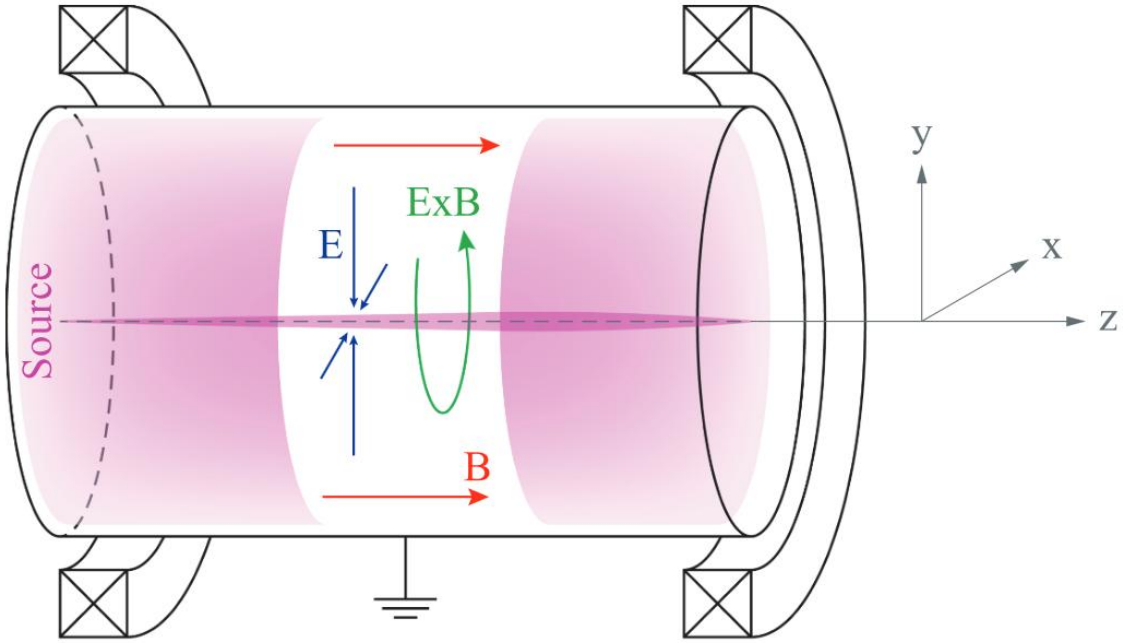
Zoltan Juhasz[1], Balint Toth[1], Peter Hartmann[2] and Zoltan Donko[2]

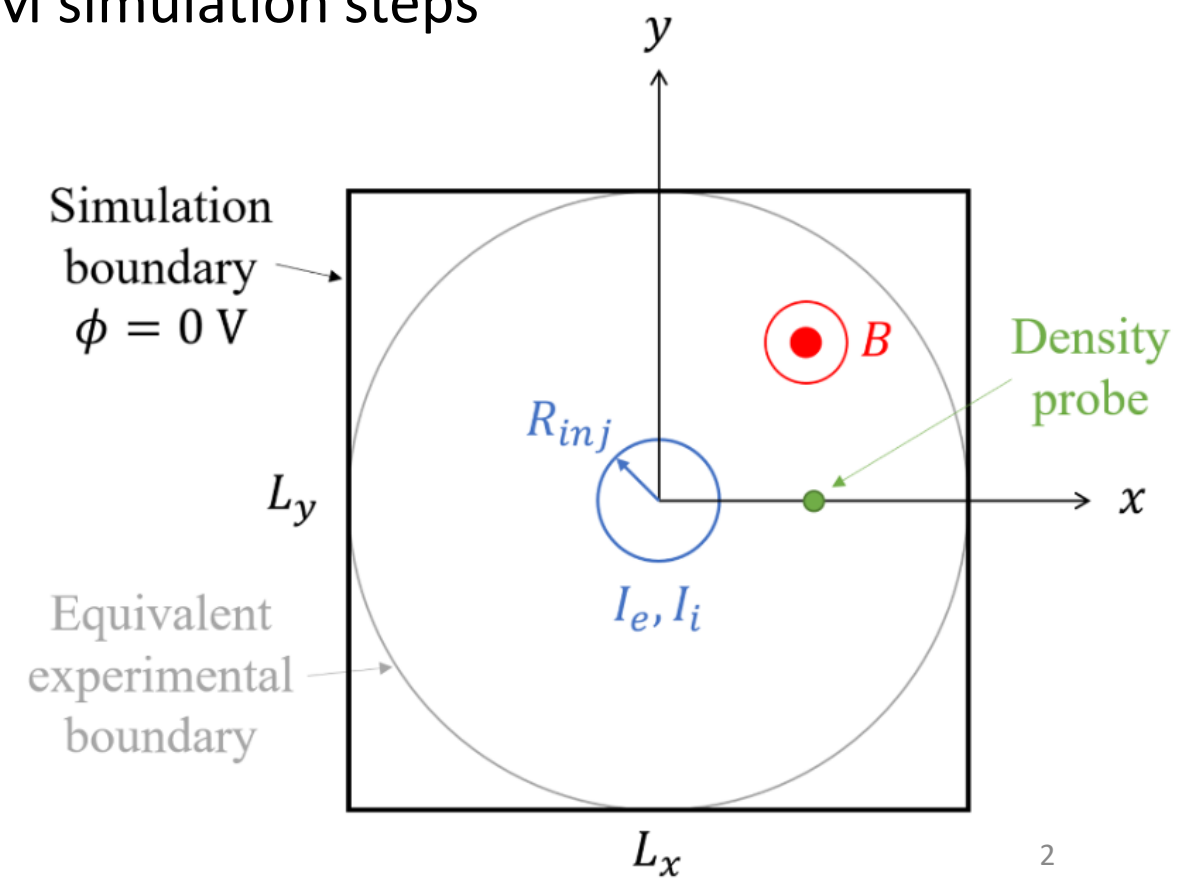
[1]Dept. of Electrical Engineering and Information Systems,
University of Pannonia, Veszprem, Hungary

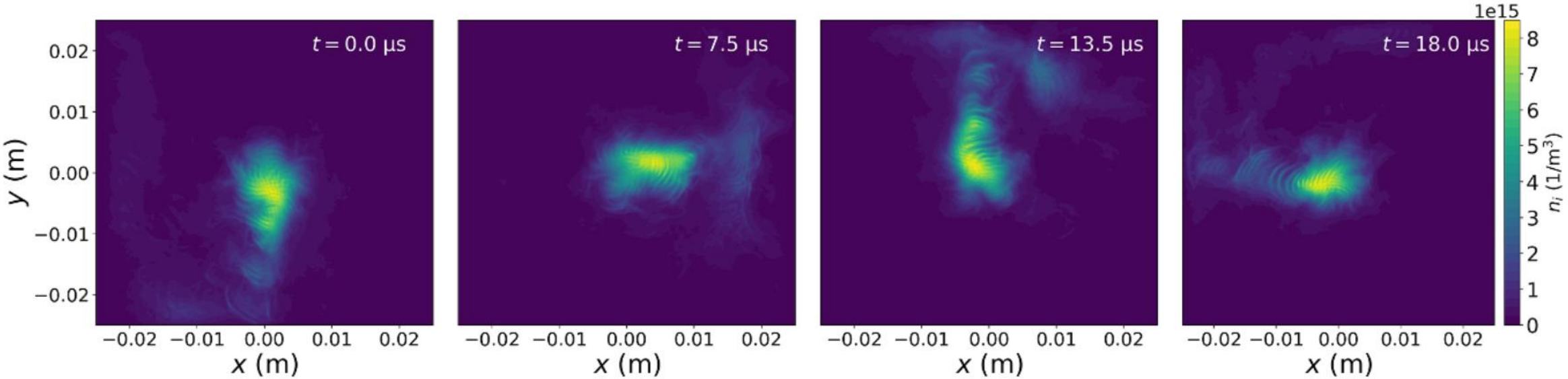
[2]Dept. of Complex Fluids, Institute for Solid State Physics and Optics,
Wigner Research Centre for Physics, Budapest, Hungary

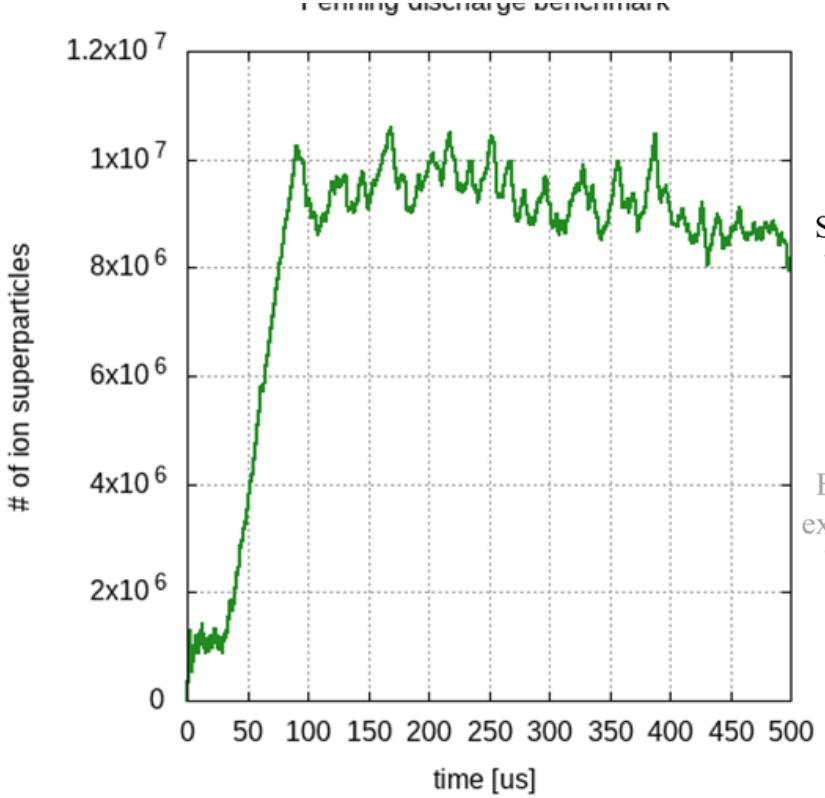
Magnetised (Penning discharge) Plasma Study

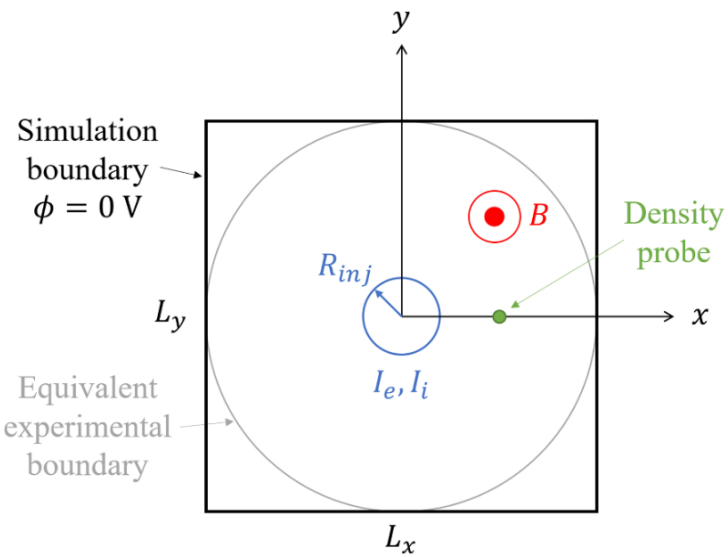
- 2D Particle-in-Cell simulation for investigating low frequency instabilities
- ~10 M superparticles, no collisions, 12.5 M simulation steps
- particle injection at the centre

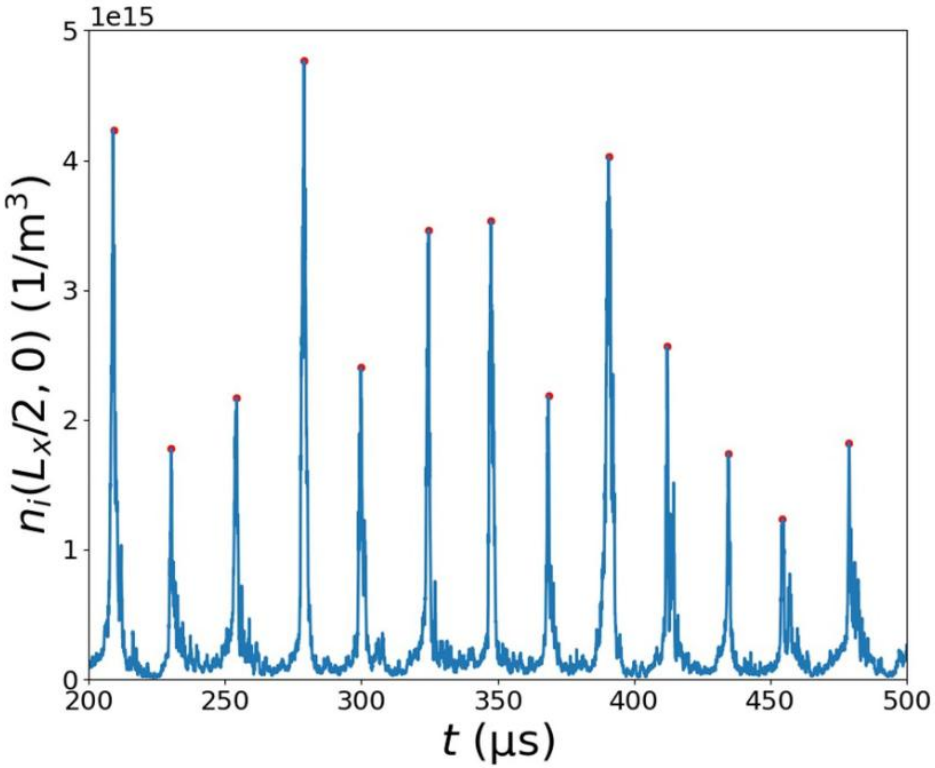





t = 0.0 μs
t = 7.5 μs
t = 13.5 μs
t = 18.0 μs
y (m)
x (m)
1e15
n_i (1/m^3)


Penning discharge benchmark
# of ion superparticles
time [us]


Simulation boundary
$\phi = 0$ V
Equivalent experimental boundary
B
R_{inj}
I_e, I_i
Density probe
L_y
L_x
x
y

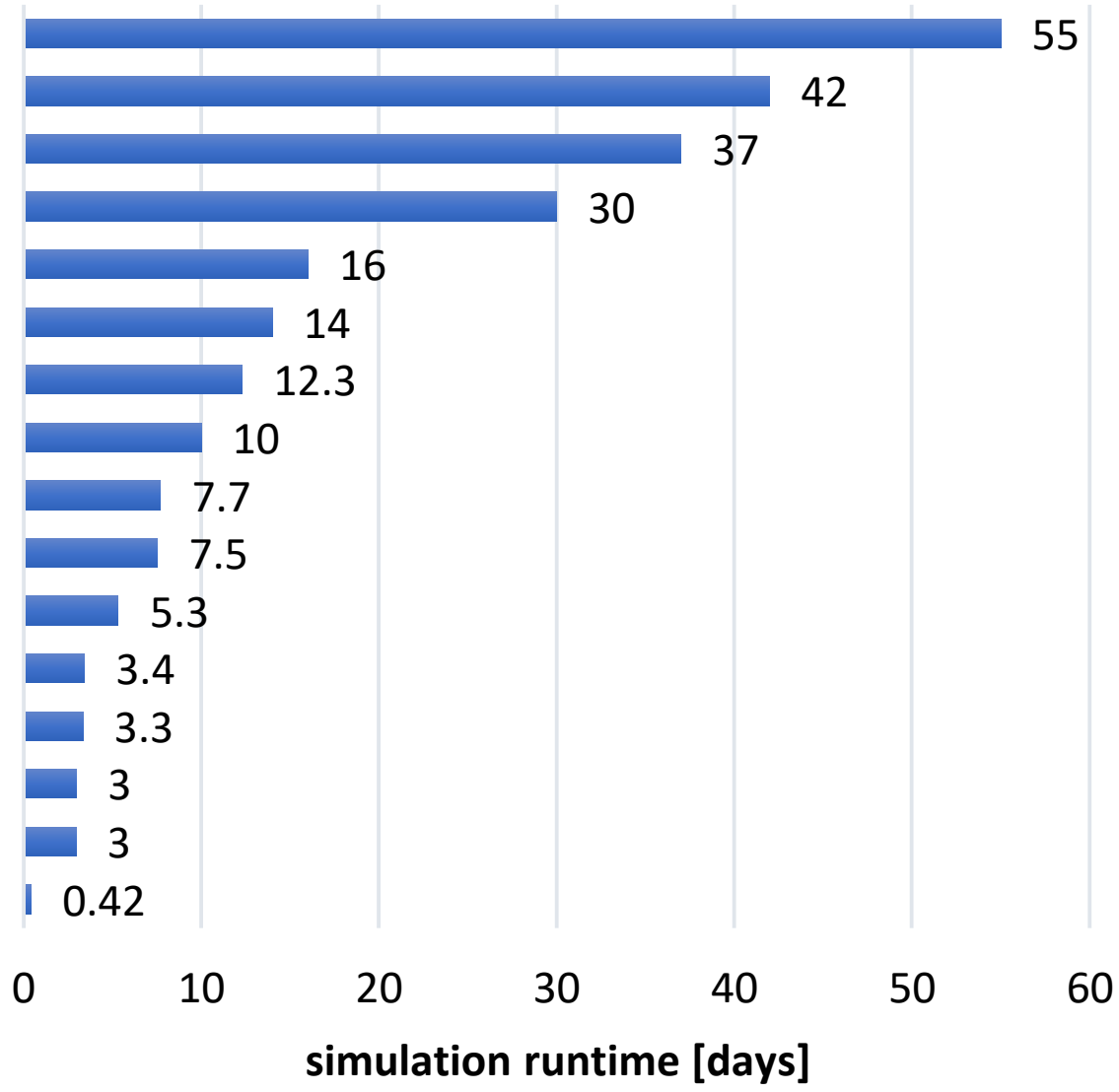

1e15
$n_i(L_x/2, 0)$ (1/m^3)
t (μs)

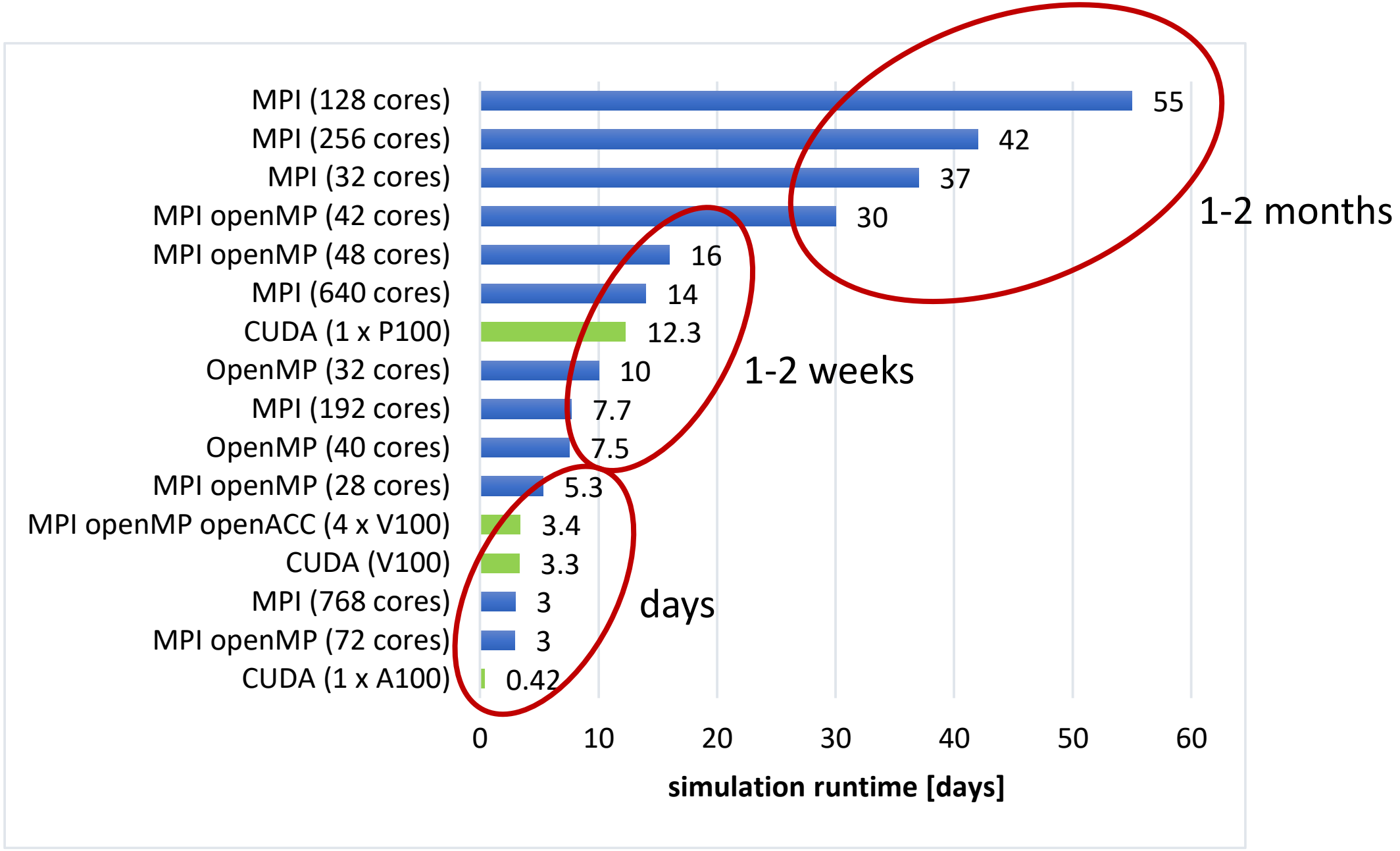
Landmark Benchmarking Effort

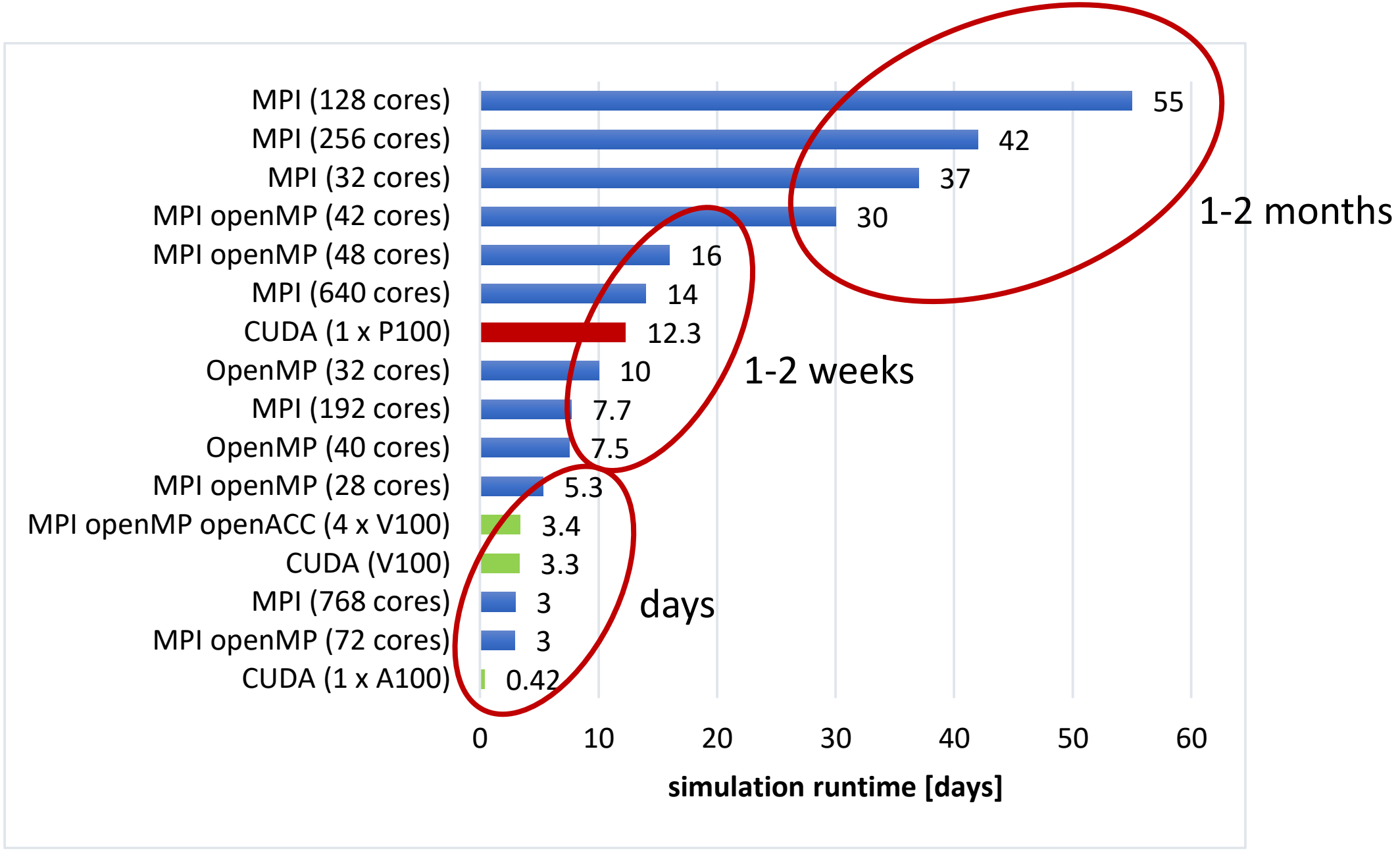
- **Princeton Plasma Physics Laboratory, Princeton, USA**
- Universidad Carlos III de Madrid, Spain
- Laboratoire de Physique des Plasmas (LPP), CNRS, Sorbonne Université, ´Ecole Polytechnique, Institut Polytechnique de Paris,
- Sandia National Laboratories, USA
- Université de Toulouse, Toulouse INP, CNRS, LAPLACE, France
- CERFACS, Toulouse, France
- Nuclear (NUC) Department, ENEA C.R. Frascati, Frascati, Rome, Italy
- Stanford University, USA
- **HUN-REN Wigner Research Centre for Physics, Hungary**
- DPHY, ONERA, Université Paris-Saclay, France
- Ruhr University Bochum, Germany
- Imperial College London, United Kingdom
- Aeronautics and Aerospace Department, von Karman Institute for Fluid Dynamics, Belgium
- KU Leuven, Belgium
- Université Libre de Bruxelles, Belgium
- Institute for Plasma Science and Technology (ISTP), CNR, Bari, Italy
- University of Saskatchewan, Canada
- University of Alberta, Canada
- Dublin City University, Ireland

Powis, Andrew T., et al. "Benchmark for two-dimensional large scale coherent structures in partially magnetized E × B plasmas—community collaboration & lessons learned." *Plasma Sources Science and Technology* **35**.2 (2026): 025002.

Execution times for 500 µs simulation



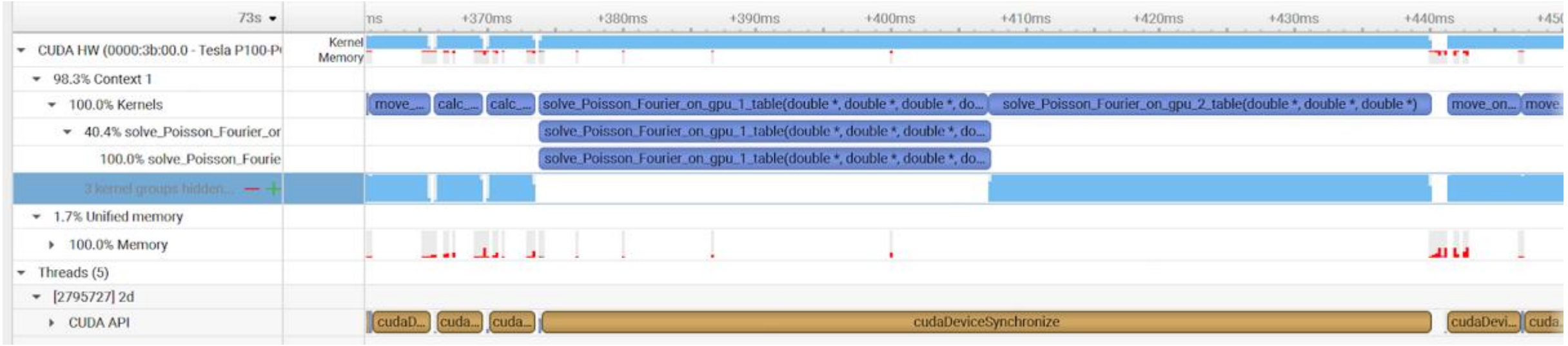

MPI (128 cores)
MPI (256 cores)
MPI (32 cores)
MPI openMP (42 cores)
MPI openMP (48 cores)
MPI (640 cores)
CUDA (1 x P100)
OpenMP (32 cores)
MPI (192 cores)
OpenMP (40 cores)
MPI openMP (28 cores)
MPI openMP openACC (4 x V100)
CUDA (V100)
MPI (768 cores)
MPI openMP (72 cores)
CUDA (1 x A100)
55
42
37
30
16
14
12.3
10
7.7
7.5
5.3
3.4
3.3
3
3
0.42
1-2 months
1-2 weeks
days
0
10
20
30
40
50
60
simulation runtime [days]


MPI (128 cores)
MPI (256 cores)
MPI (32 cores)
MPI openMP (42 cores)
MPI openMP (48 cores)
MPI (640 cores)
CUDA (1 x P100)
OpenMP (32 cores)
MPI (192 cores)
OpenMP (40 cores)
MPI openMP (28 cores)
MPI openMP openACC (4 x V100)
CUDA (V100)
MPI (768 cores)
MPI openMP (72 cores)
CUDA (1 x A100)
55
42
37
30
16
14
12.3
10
7.7
7.5
5.3
3.4
3.3
3
3
0.42
1-2 months
1-2 weeks
days
0
10
20
30
40
50
60
simulation runtime [days]

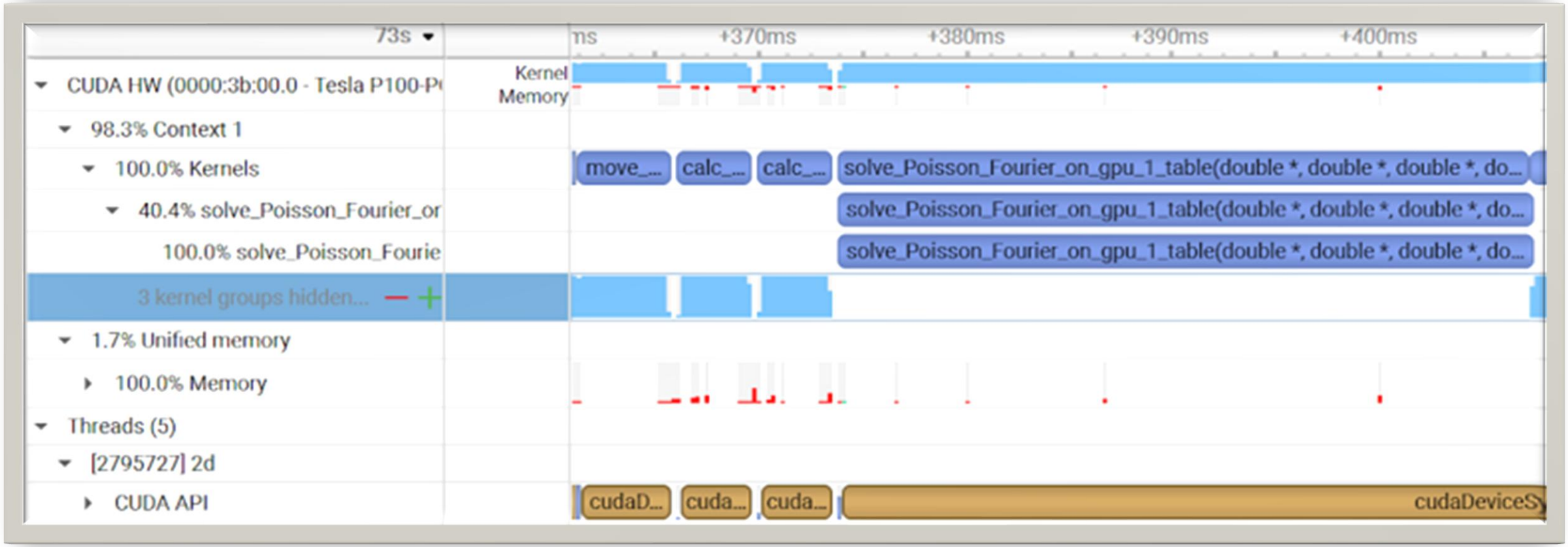
Wigner MZ/X Penning code

- Hybrid CPU-GPU implementation
  - particle move, density calculation, direct spectral Poisson solver on GPU
  - particle injection, diagnostics, particle management on CPU
- Using managed memory for simplified programming style
- 100 µs simulation time section is used for profiling
  - 2.5 million simulation step

Baseline Performance Profiling



- Poisson solver dominates execution time
- frequent host-device data transfers

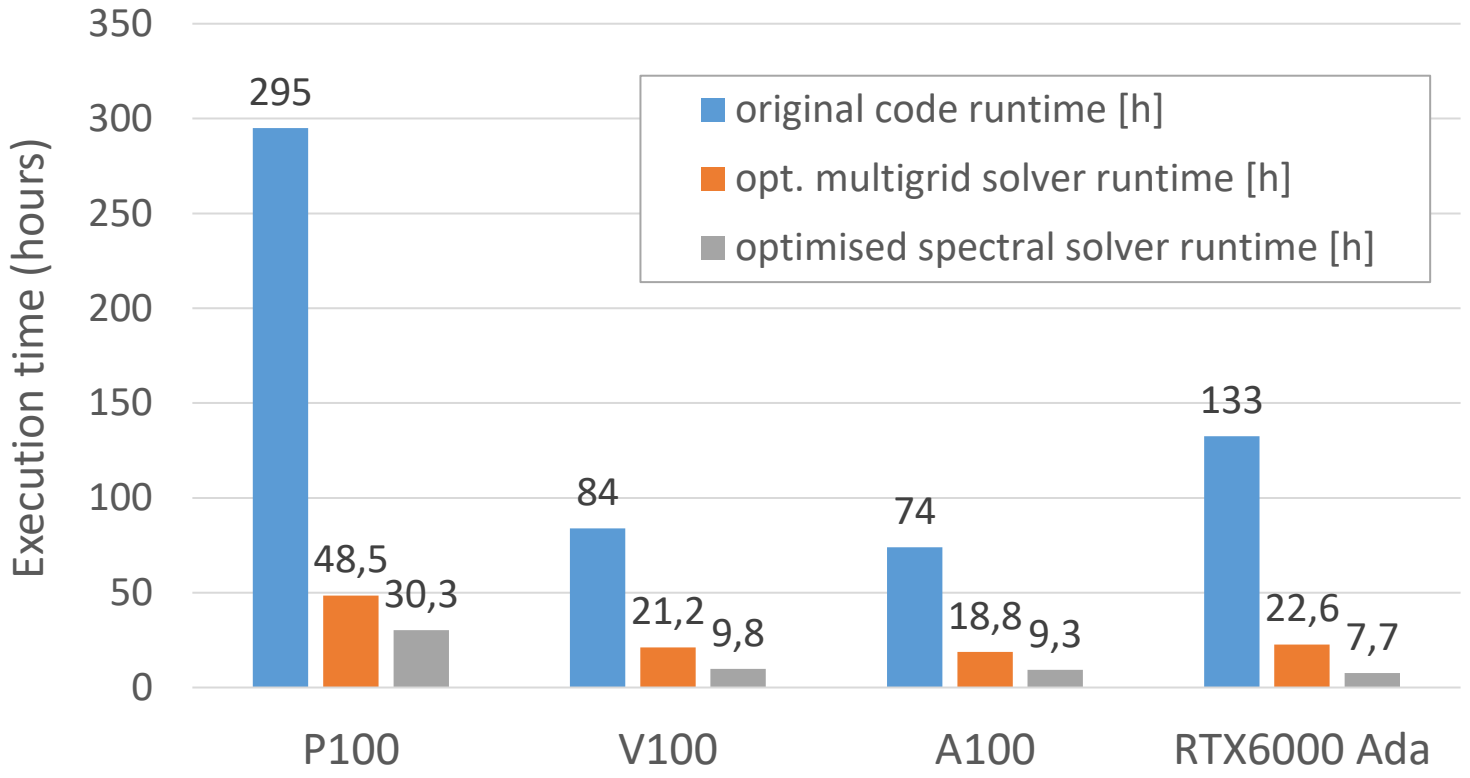

73s
ns
+370ms
+380ms
+390ms
+400ms
CUDA HW (0000:3b:00.0 - Tesla P100-P
Kernel
Memory
98.3% Context 1
100.0% Kernels
move_...
calc_...
calc_...
solve_Poisson_Fourier_on_gpu_1_table(double *, double *, double *, do...
40.4% solve_Poisson_Fourier_or
solve_Poisson_Fourier_on_gpu_1_table(double *, double *, double *, do...
100.0% solve_Poisson_Fourie
solve_Poisson_Fourier_on_gpu_1_table(double *, double *, double *, do...
3 kernel groups hidden...
1.7% Unified memory
100.0% Memory
Threads (5)
[2795727] 2d
CUDA API
cudaD...
cuda...
cuda...
cudaDeviceS

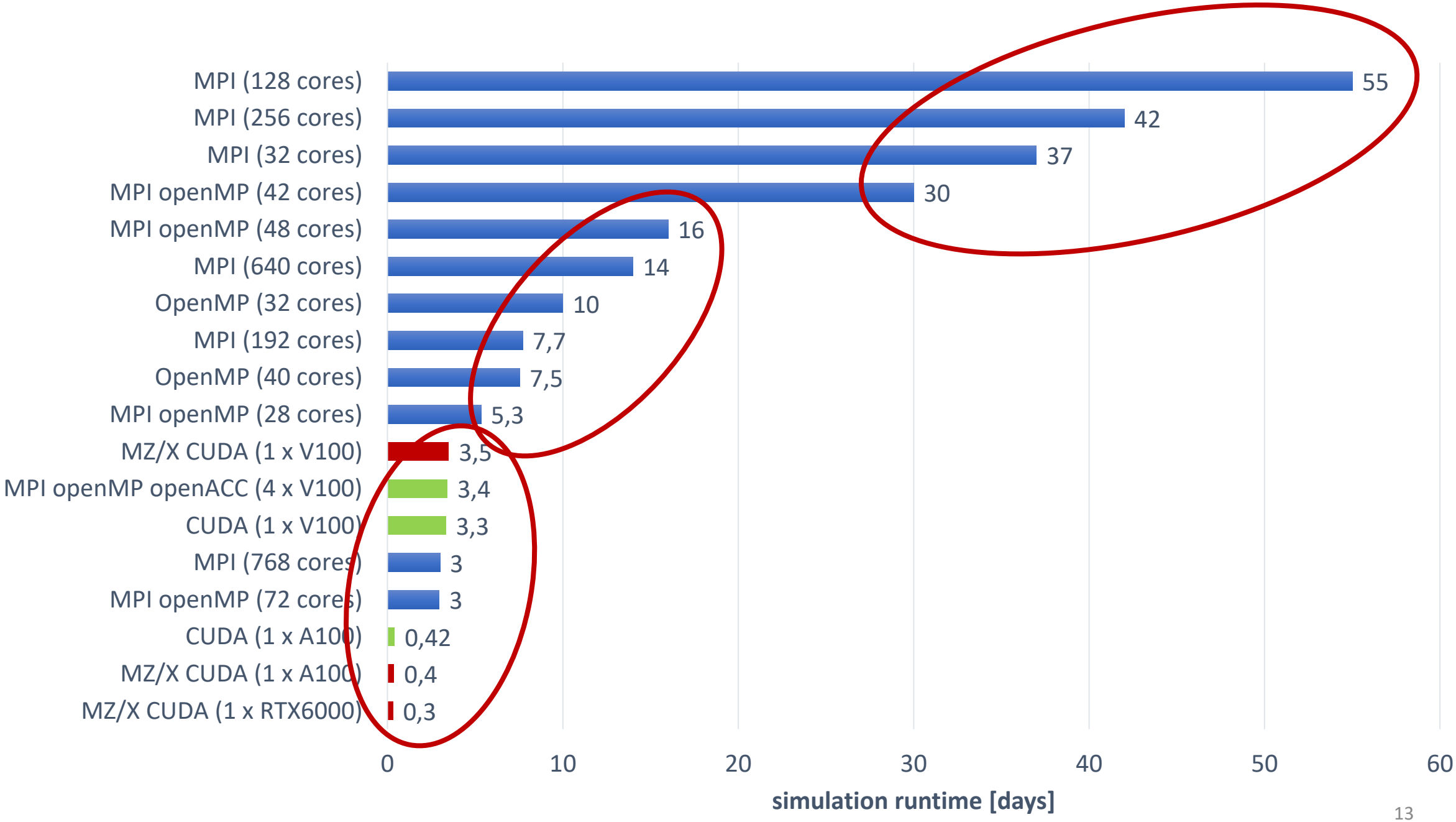
Optimisations steps

1. Explicit memory management, removing managed memory
2. Moving all simulation steps to GPU
3. Implementing new Poisson solvers
    1. Multigrid
    2. Spectral solver
4. Optimising diagnostics computation

Execution time results

- Four GPU architectures: P100, V100, A100, RTX 6000 Ada
- Execution time for 100 µs simulation
- Results for
  - original,
  - multigrid and
  - spectral solver versions




MPI (128 cores)
MPI (256 cores)
MPI (32 cores)
MPI openMP (42 cores)
MPI openMP (48 cores)
MPI (640 cores)
OpenMP (32 cores)
MPI (192 cores)
OpenMP (40 cores)
MPI openMP (28 cores)
MZ/X CUDA (1 x V100)
MPI openMP openACC (4 x V100)
CUDA (1 x V100)
MPI (768 cores)
MPI openMP (72 cores)
CUDA (1 x A100)
MZ/X CUDA (1 x A100)
MZ/X CUDA (1 x RTX6000)
55
42
37
30
16
14
10
7,7
7,5
5,3
3,5
3,4
3,3
3
3
0,42
0,4
0,3
0
10
20
30
40
50
60
simulation runtime [days]

Why stop here?

Multi-GPU implementation

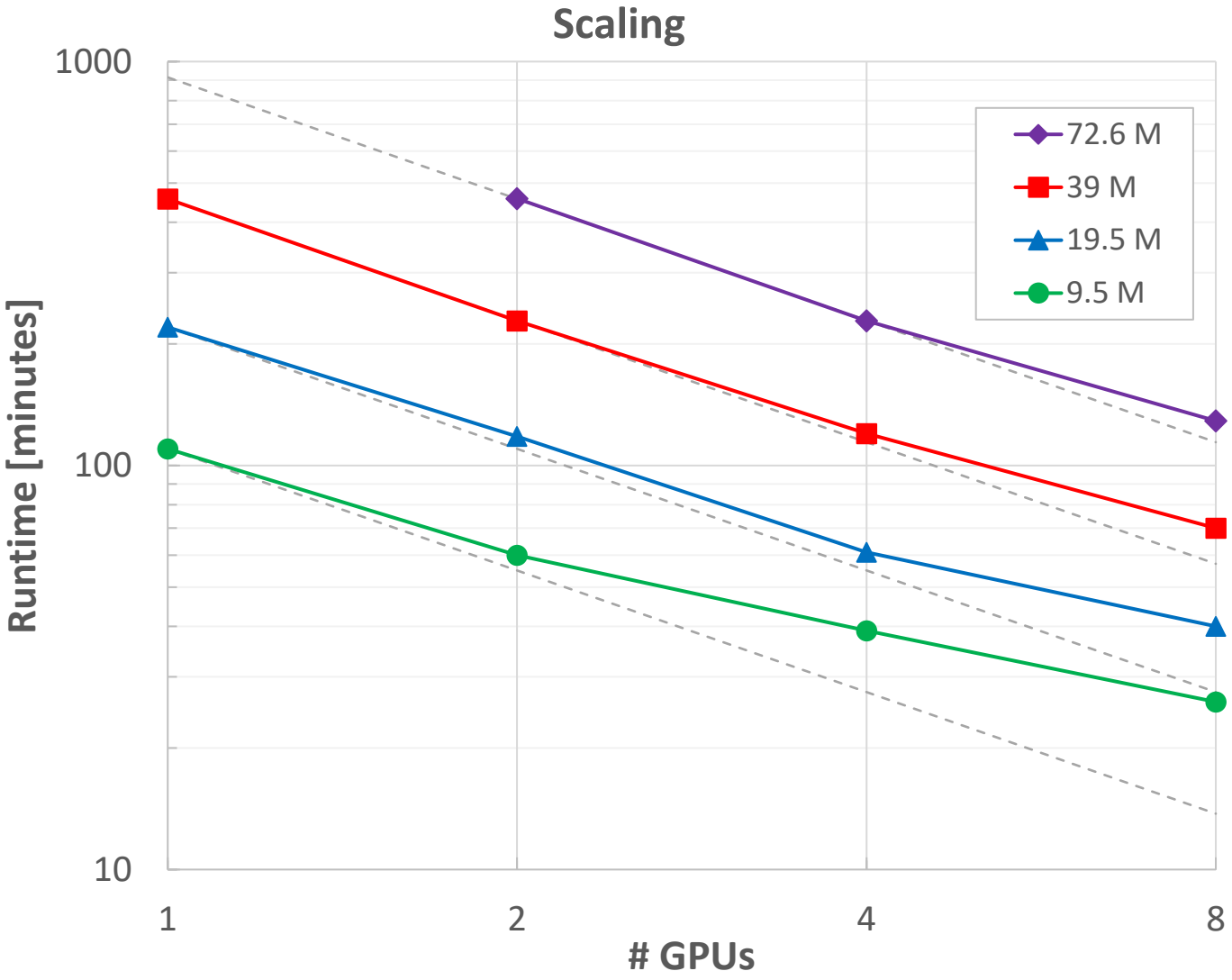
- Target systems
  - Wigner: 2 RTX 6000 Ada GPUs
  - Komondor: 1-16? A100 GPUs
- Particle partitioning
- Relying on superposition principle
- MPI and NCCL collective communications
  - automatic particle partitioning based on GPU configuration
  - AllReduce before Poisson solver
- Diagnostics running overlapped in host thread

Multi-GPU advantage:

- introduce **further speedup** for a system of given size (strong scaling)
- **increase particle count** without increasing execution time (weak scaling)

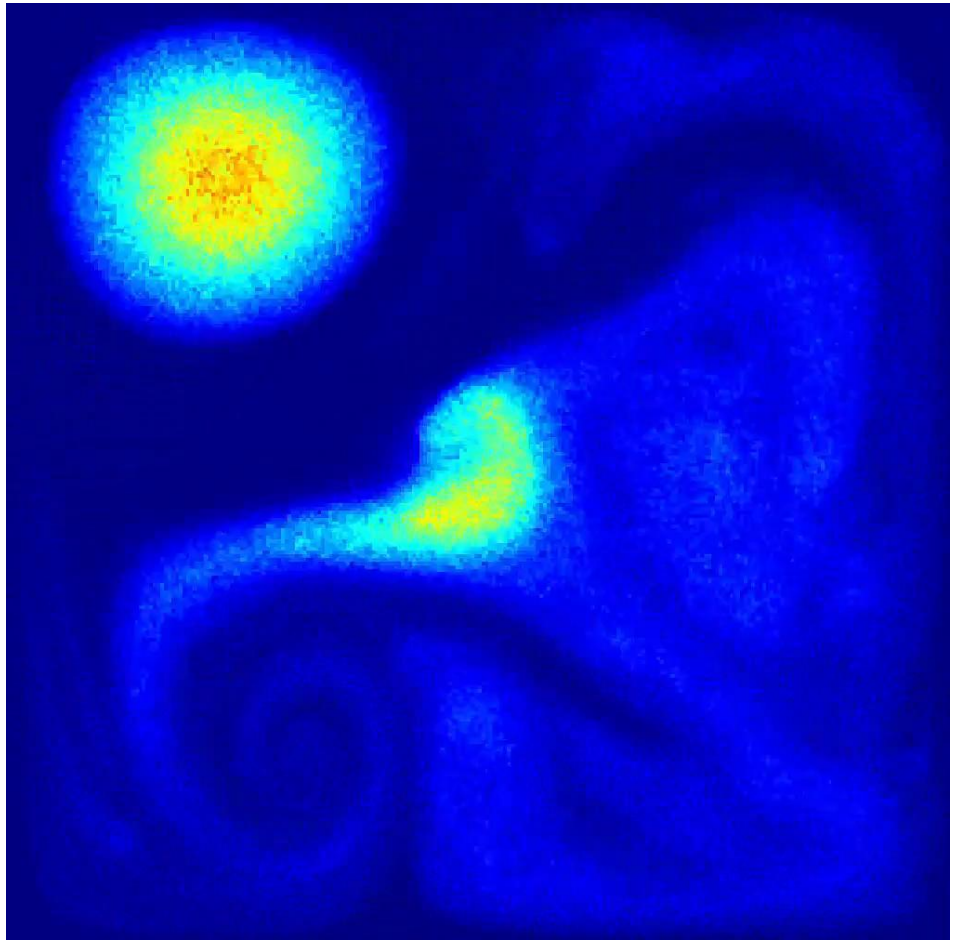
Scalability

- 100 µs simulation
- strong scaling
  - 1 to 8 GPUs
- weak scaling
  - 9.5 to 72.6 M particles
- Best time on A100
  - 26 mins (100 µs)
  - ~ 2 hours (full simulation)



Conclusions

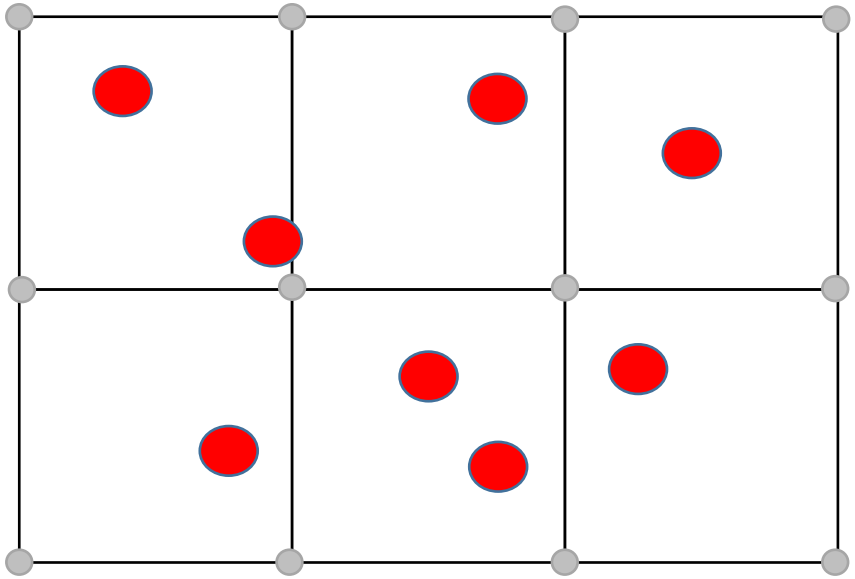
- Algorithmic and implementation optimisations paid off
  - fastest Penning simulation within the Benchmark Team
- Opened new opportunities
  - higher spatial resolution
  - effect of injecting current amplitude
  - full 3D simulation
- Next steps
  - Large-scale scalability benchmarking is planned on the **MareNostrum 5** (H100 GPUs, Barcelona) and **Leonardo** (A100 GPUs, Bologna) supercomputers; larger grids, up to 1 billion particles
  - 3D Penning code development
  - Collisional 2D and 3D simulation



Acknowledgements: NVIDIA Academic Grant Program, DKF Kft for access to the Komondor supercomputer

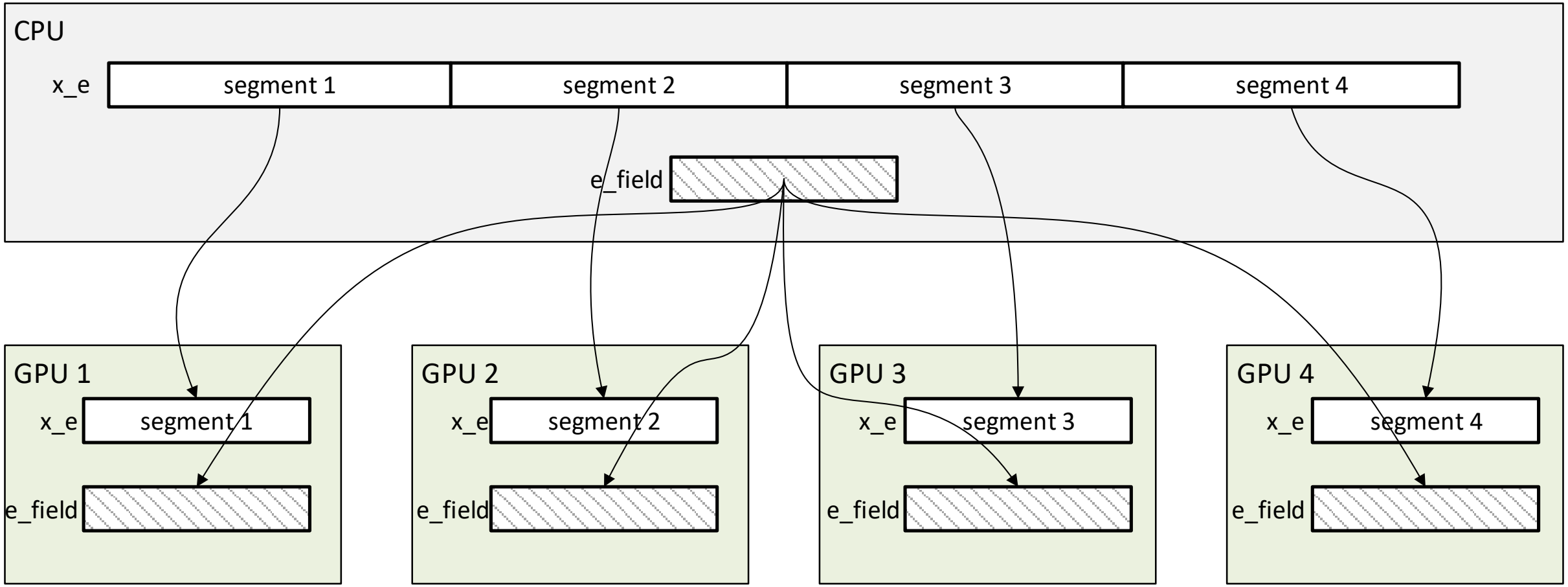
Approach: Particle-in-Cell (PIC) simulation

- N-body problem
  - no direct particle interaction, particles interact with the field
  - place particle charges to grid
  - solve grid for field – Poisson equation
  - move particles based on field forces
- Particle count: from 100k to 10m particles
- Complication: collision!
- CPU execution is long: ranging from days to months
  - parallel solutions: OpenMP and/or MPI code
  - irregular memory accesses make code difficult to parallelise efficiently



Data distribution

- each GPU moves particles locally and computes new charge densities
- Poisson solver on CPU or on each GPU (requires collective communication)



Problems

- Requires collective communication operations involving CPU and GPU memory
  - traditional MPI is not suitable – requires extra copy operations from GPU to CPU
  - CUDA-aware MPI is OK, can use GPU pointers
- Communication initiation is on CPU
  - difficult to overlap comms. with GPU kernel execution
  - communication can eventually become a performance bottleneck
  - scalability limit

Alternative No 1 – NCCL

- NVIDIA Collective Communication Library
- Provides uniform API for all-gather, all-reduce, broadcast, reduce, reduce-scatter, point-to-point send and receive comm. routines
- Optimized to achieve high bandwidth and low latency over PCIe, NVLink and other high-speed interconnects
- Automatic topology detection for high bandwidth paths on AMD, ARM, PCI Gen4 and InfiniBand
- Supports multi-threaded and multi-process applications
- InfiniBand, RoCE and IP Socket internode communication
- NCCL 2.4 introduced tree-based all-reduce instead of rings

Mismatch of communication libraries and the CUDA model

- MPI and NCCL provide CPU-initiated communication routines
- Assumes separate computation and communication steps
- Fine for synchronous execution where all processes execute in parallel
- CUDA model relies on large number of threads to hide memory and instruction latency: threads >> cores
- Internal scheduling of kernels (thread blocks) makes it difficult to hide communication
- Need a distributed model more aligned with the CUDA programming model

Summary

- Still in progress, single node and rudimentary MPI versions are complete, NVSHMEM version coming soon
- There are many alternatives for structuring and implementing pre-exascale multi-GPU programs
- Selecting the best strategy and creating efficient code might not be easy
- Following implementation patterns developed for much smaller CPU-based systems might not be a good long-term choice
- Do not be afraid to re-design, re-structure your program
- Programs with hundreds of millions or billions of threads can be executed efficiently on state-of-the-art systems

Abstract

In this talk, we plan to report the results of our performance optimisation effort aimed at speeding up a GPU-accelerated 2D Particle-in-Cell plasma simulation code, following an international plasma simulation benchmarking effort of 19 leading plasma research groups. We studied the effects of memory management, data movements, the choice and implementation of the Poisson solver, the use of mixed precision computation and various kernel-level optimisation techniques. Finally, the optimised version was improved to support multi-GPU execution as well. The key steps of the optimisation, the performance critical factors and our decisions will be presented in detail in the talk. Compared to the original baseline version that executed the simulation in 10 days, the runtime was reduced to less than 10 hours on an NVIDIA A100 GPU. Taking into account the performance increase of each new generation of GPU cards (from P100 to A100) as well, the overall speedup achieved with the optimised version exceeds 30×. The multi-GPU implementation will further reduce the execution time; using 4 A100 GPUs, we expect the final runtime to be less than 3 hours (note: final steps of the implementation and testing are currently underway, exact timings will be presented at the conference), that will make our implementation the fastest Penning discharge simulation code available. This level of performance not only means that we can execute simulations within a few hours instead of several days or weeks on GPUs, but it also allows for tackling large and complex simulation problems that were previously thought to be impossible to execute.

draft

- Penning benchmark effort: simulation, participant groups
- some outputs, execution time histogram
- architecture and parallelism analysis
- GPU code v0 → fast development, correct operation, no pre-mature optimisation
- profiling: Poisson solver, managed memory
- optimisation process on single GPU → steps, speedup, live simulation video
- multi-GPU work current results → scalability, times, etc.
- Next steps: 3D Penning, 3D collisional PIC/MCC

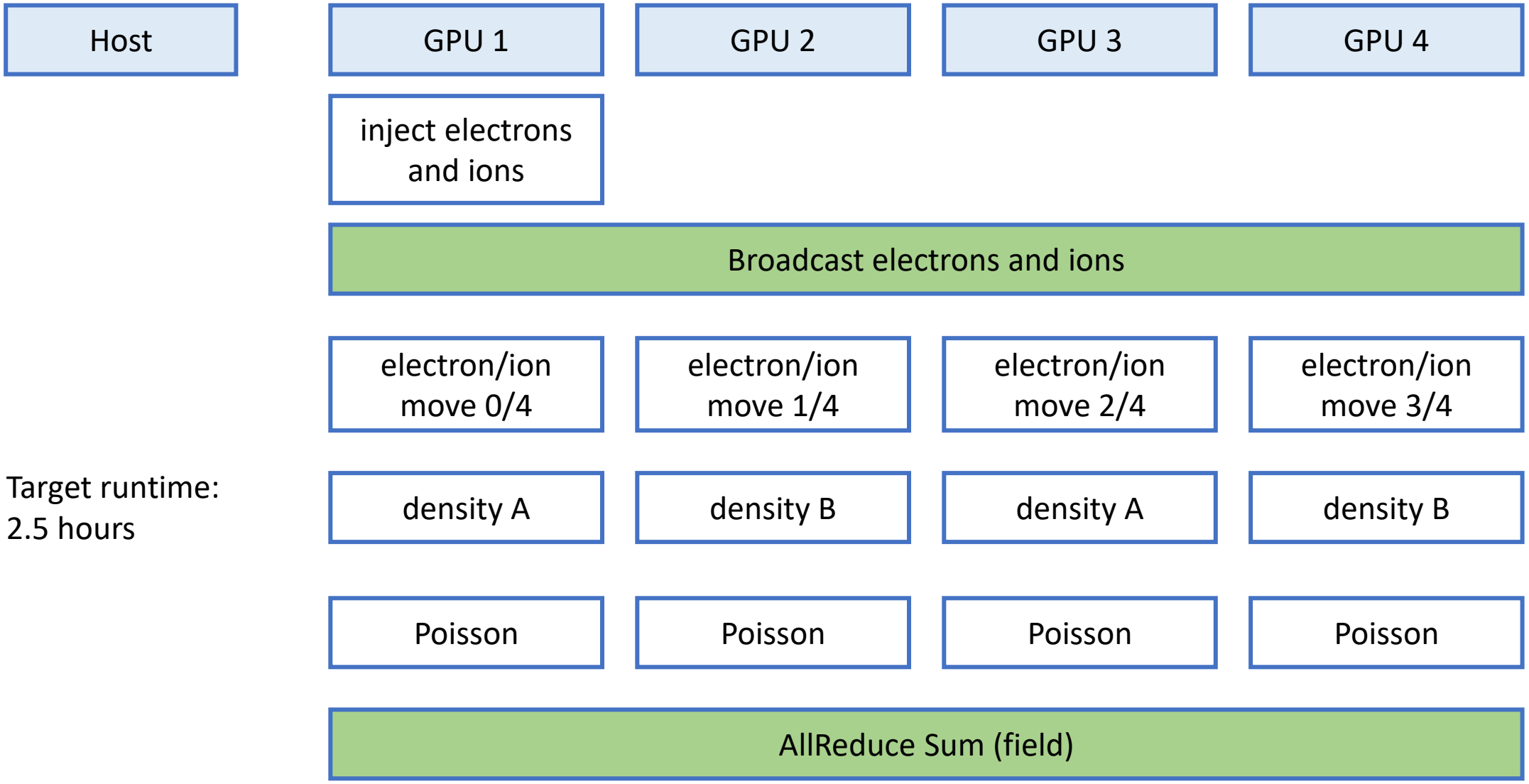
Single node, single thread multi-GPU program

```
// distributing the workload across multiple devices
for (int i = 0; i < ngpus; i++) {
    cudaSetDevice(i);
    cudaMemcpyAsync(d_A[i], h_A[i], iBytes,
                    cudaMemcpyHostToDevice, stream[i]);
    cudaMemcpyAsync(d_B[i], h_B[i], iBytes,
                    cudaMemcpyHostToDevice, stream[i]);
    kernel<<<grid, block, 0, stream[i]>>> (d_A[i], d_B[i], d_C[i], iSize);
    cudaMemcpyAsync(h_C[i], d_C[i], iBytes,
                    cudaMemcpyDeviceToHost, stream[i]);
}
cudaDeviceSynchronize();
```

communication

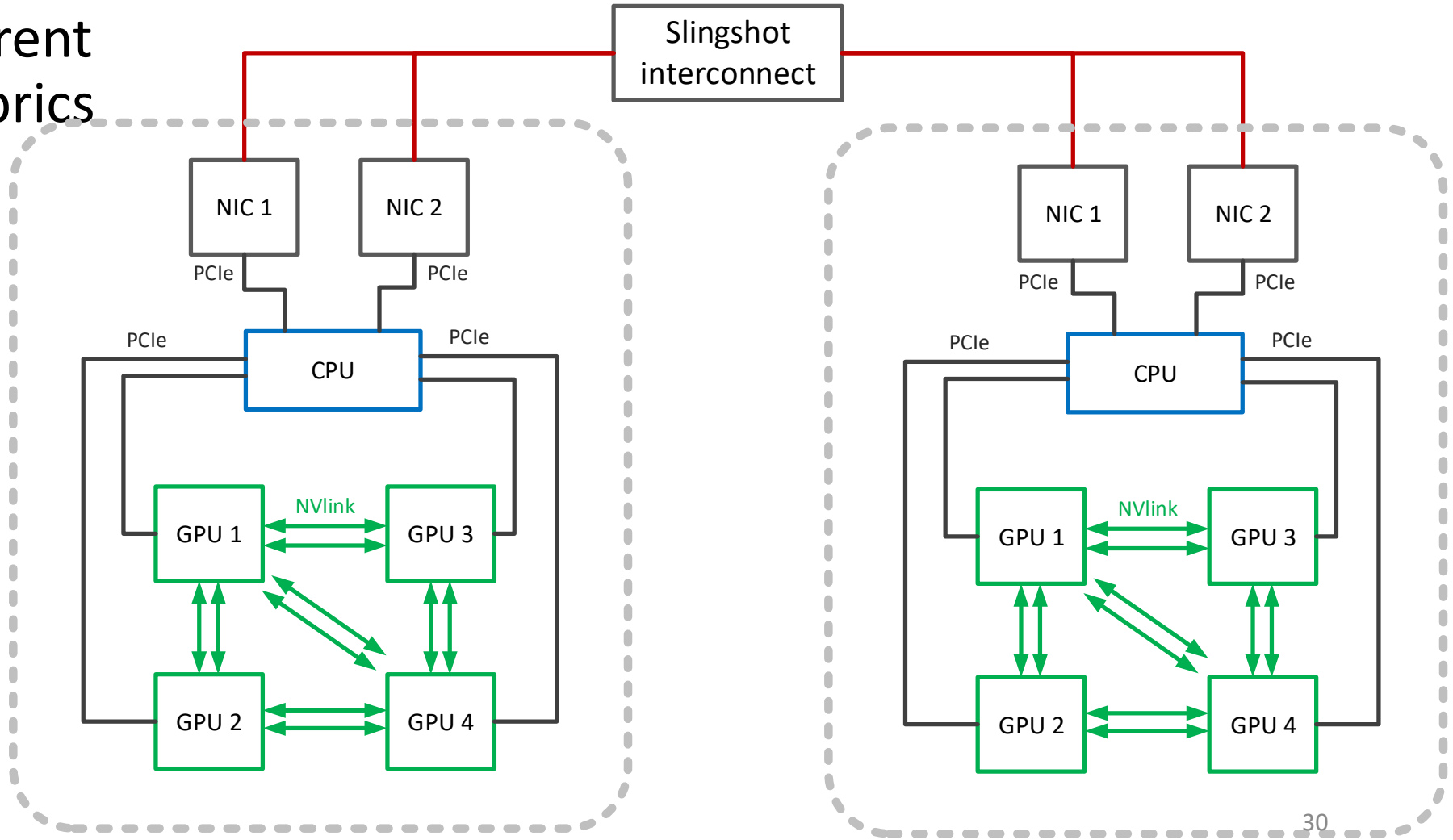
- data distribution, AllReduce
- MPI vs NCCL
- NCCL issue

Single node, 4 GPUs



Pre-exascale multi-GPU architecture

- At least three different communication fabrics
  - PCI-e (CPU-GPU)
  - Nvlink (GPU-GPU)
  - Slingshot or Infiniband (node-node)
- HPE Cray EX
- MARCONI 100
- LEONARDO



2D geometry

- 1D case
  - particle count up to 1-10 millions: 1k cells, 1-10k particles per cell
- 2D case
  - 500 x 500 or 1k x 1k grid
  - 100 to 1k particles per cell
  - 25 M – 1 B particles
- e_field should be in shared memory on each GPU
  - 1M instead of 1K elements
  - Ampere persistent shared memory could help, otherwise must be moved into global memory

- single node 1,2 ,4 GPUs
- multi-node
- strong scaling
- weak scaling