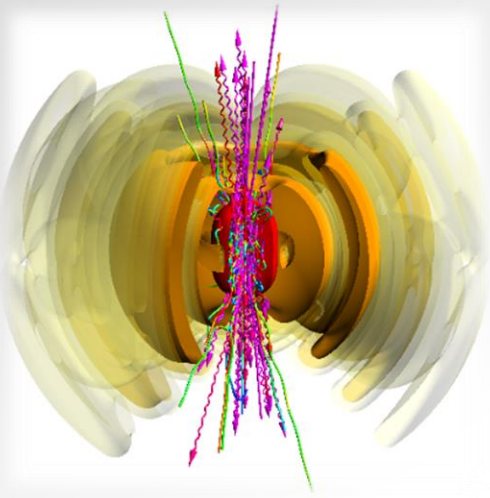
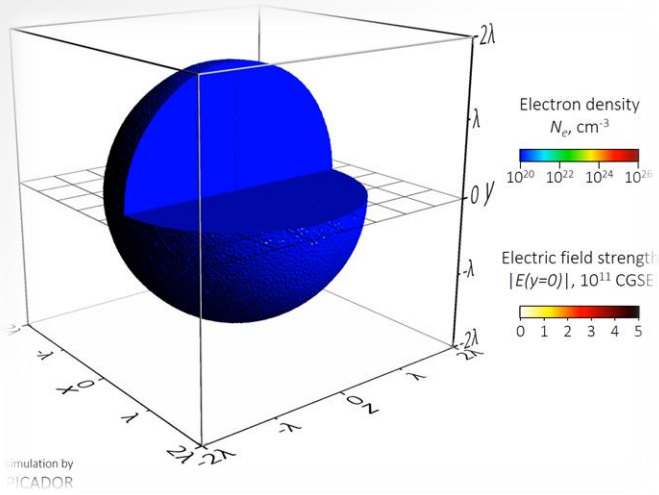
LOBACHEVSKY UNIVERSITY
INSTITUTE OF INFORMATION TECHNOLOGIES, MATHEMATICS AND MECHANICS

High performance particle-in-cell laser-plasma simulation in PICADOR. Parallelization techniques

Iosif Meyerov, meerov@vmk.unn.ru



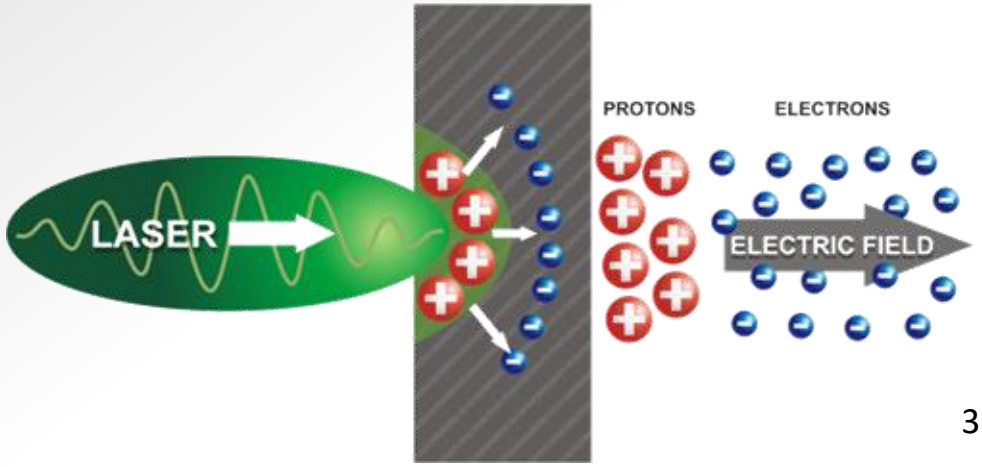


Seminar talk, 2021

BACKGROUND

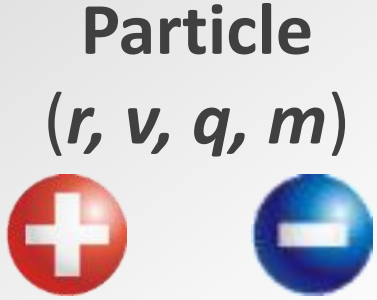
Laser Plasma Simulation. Research Area

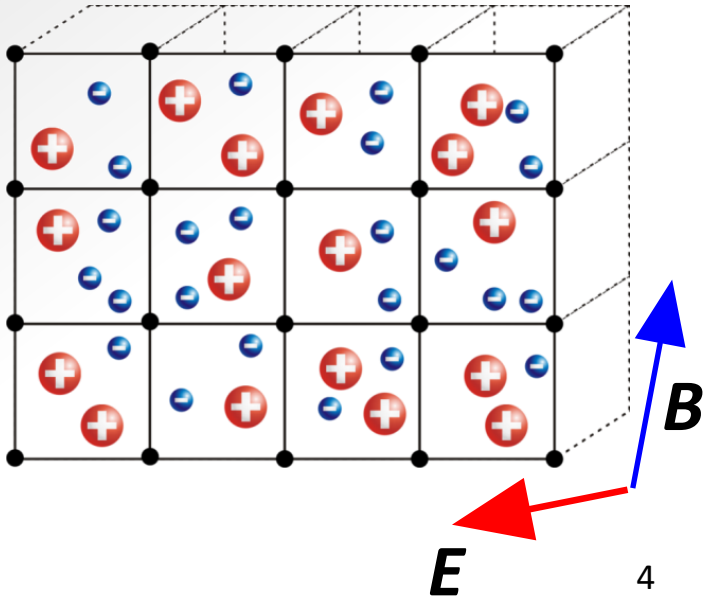
- Simulation of irradiation of various targets with high-intensity laser pulses
  - Tool for theoretical research
  - Compact particle sources for medicine
  - Generation of radiation with unique properties
- Widely used particle-in-cell method (PIC)
- 3D PIC simulation requires parallel computing:
  - $10^8 – 10^{10}$ cells
  - $10^8 – 10^{10}$ particles
  - $10^3 – 10^6$ cores



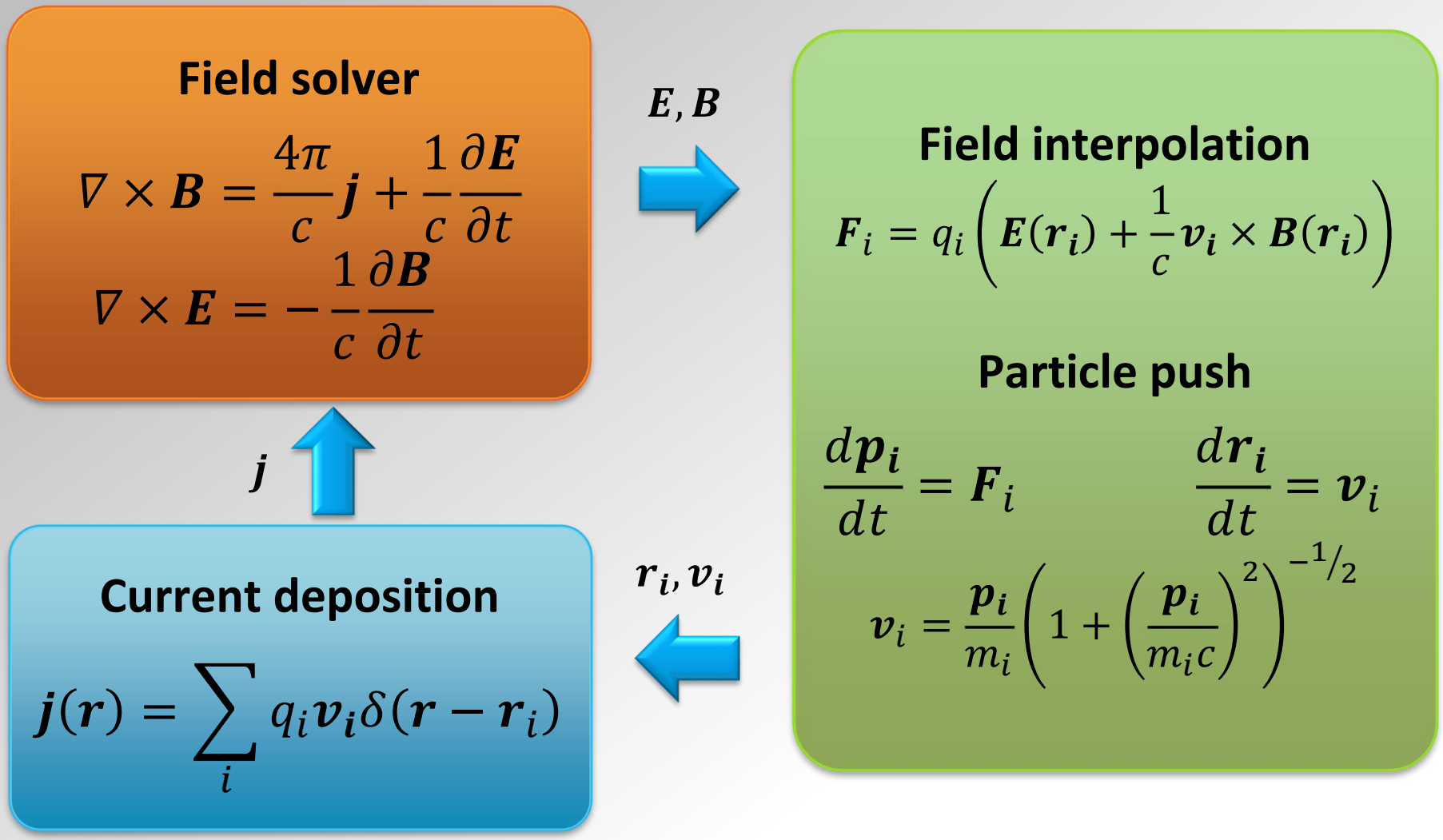
Particle-in-Cell Method

- **Two main sets of data**:
  - Particle ensemble (electrons, ions)
  - Electromagnetic field, defined on a regular grid
- No direct Particle-Particle interaction
- Spatially local particle-grid interactions
- Each simulated particle (macro particle) corresponds to a cloud of real particles
- Charge distribution in a cloud is defined by a particle form factor





Basic Computational Loop



Software

- EPOCH (University of Warwick)
- OSIRIS (IST, UCLA)
- PIConGPU (HZDR)
- SMILEI (CNRS)
- VLPL (HHU)
- WarpX (LBNL)
- …

In Russia:

- Codes developed by Keldysh Institute of Applied Mathematics, ICM&MG SB RAS, FRC ICT, UNN & IAP RAS…

OUR TEAM AND SOFTWARE

Our Team ("How it all began")

Arkady Gonoskov, PhD, Univ. of Gothenburg, UNN

Evgeny Efimenko, Research Scientist, Inst. of Appl. Phys.

Iosif Meyerov, PhD., vice-head of the dep., UNN

Code developers, users

Sergey Bastrakov, PhD, HZDR (Germany) (lead developer of PICADOR, 2010-2018)
Alexey Bashinov, researcher, IAP RAS
Tom Blackburn, PhD, researcher, Univ. of Gothenburg
Artem Korzhimanov, PhD, researcher, IAP RAS
Alexander Muraviev, researcher, IAP RAS
Valentin Volokitin, PhD student, UNN (*lead developer*)
Elena Panova, master student, UNN
Alexander Panov, master student, UNN
Yury Rodimkov, master student, UNN
Igor Surmin, engineer, UNN
Kirill Tarakanov, master student, UNN
Anastasiia Arisova, student, UNN

Collaborators

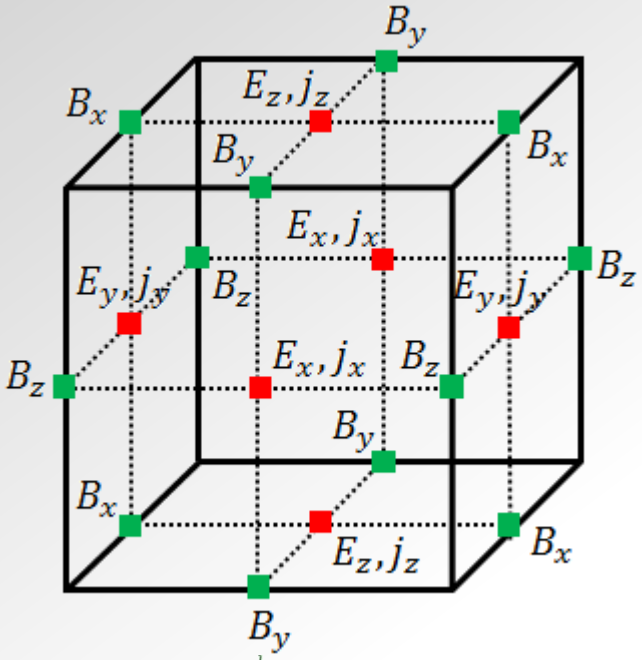
Arkady Kim, PhD., Prof., IAP RAS (Russia)
Mattias Marklund, PhD, Prof., Univ. of Gothenburg (Sweden)
Felix Mackenroth, PhD, researcher, Max Planck Institute for the Physics of Complex Systems (Germany)
László Veisz, PhD., Prof. Umea Univ., Sweden

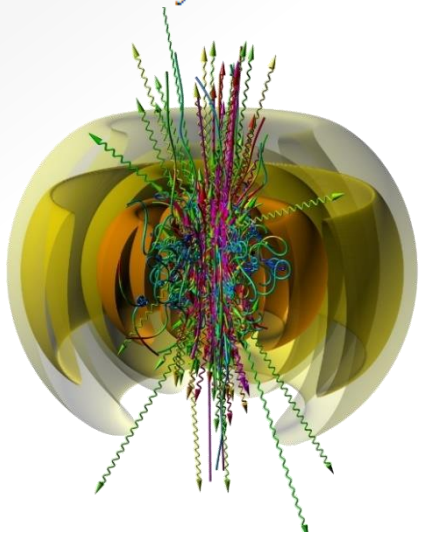
PICADOR Code

- Code for 3D plasma simulation based on the particle-in-cell method
- Developed in UNN and IAP RAS since 2010
- Multi-level infrastructure
  - Optimized computational core
  - Extendable with modules
  - Visualization tools
- Wide set of numerical schemes and extensions for additional physical effects
- Support for Intel Xeon Phi and modern manycore CPUs

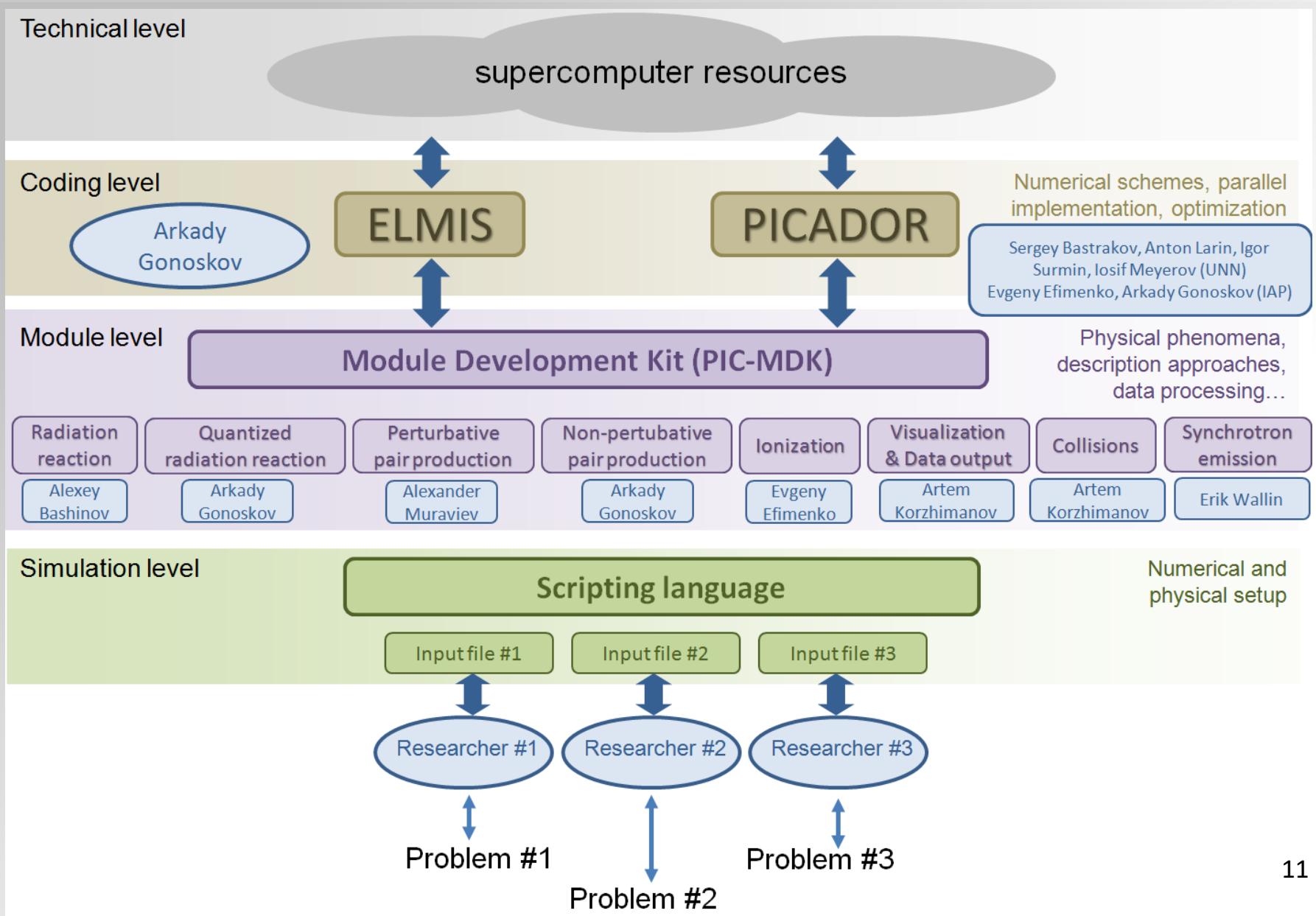
Numerical Schemes and Extensions

- CIC, TSC particle form factors
- Boris and Vay particle pushers
- Yee grid
- FDTD and NDF field solvers
- Current deposition schemes with and without charge conservation
- Absorbing boundary conditions
- Boundary pulse generators
- Moving frame
- Ionization
- QED-PIC, resampling



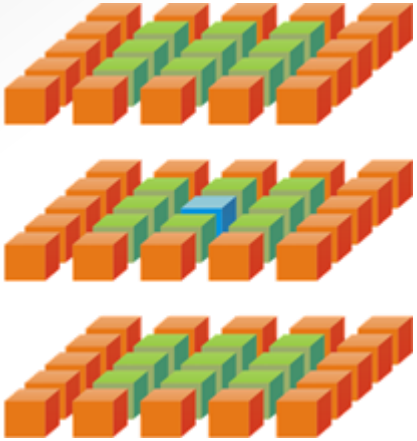


PIC-MDK Interface



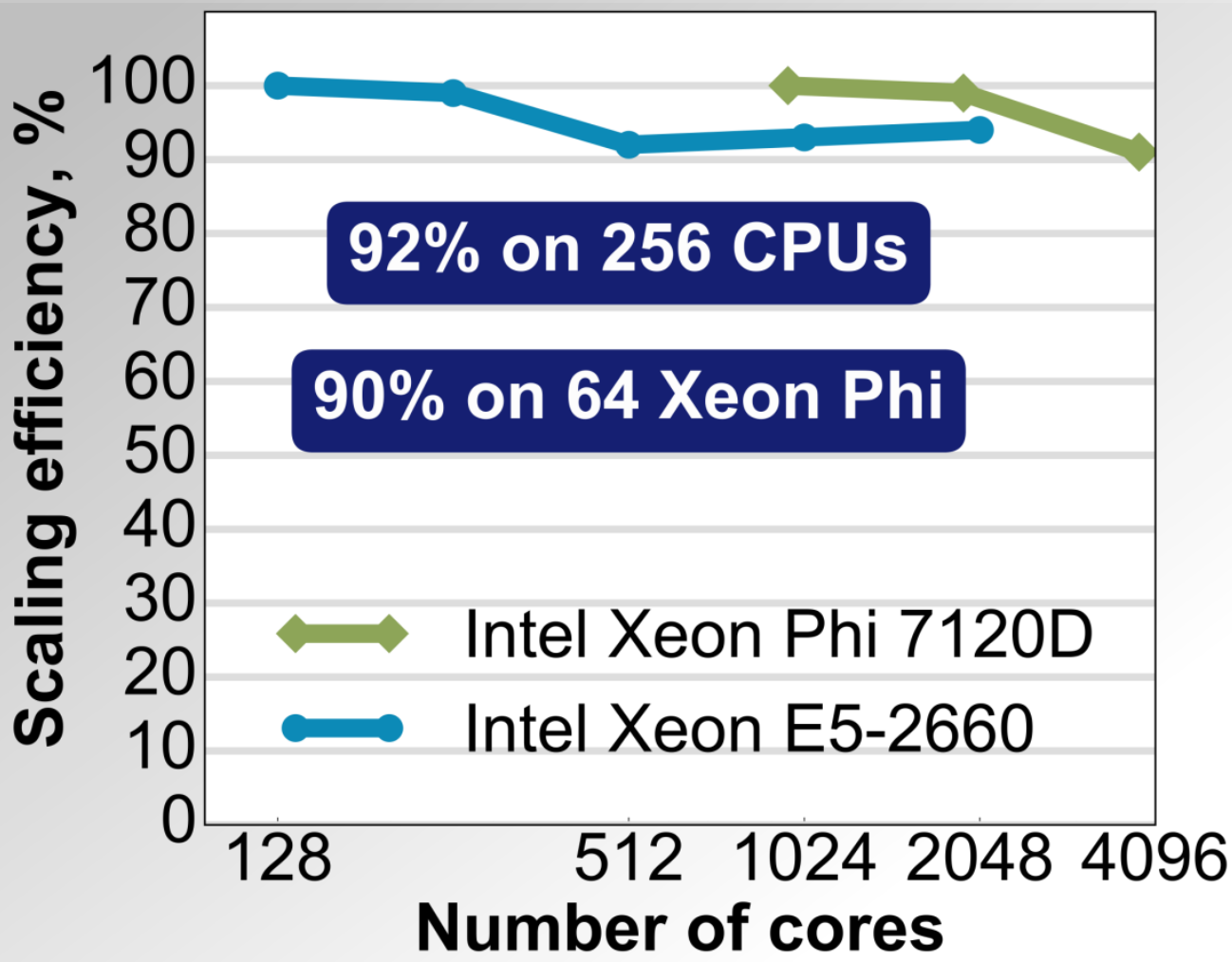
Parallelism

- **Levels of parallelism** in the Particle-in-Cell method:
  - **Distributed memory**: spatial domain decomposition, load balancing (rectilinear load balancing), MPI
  - **Shared memory**: OpenMP + custom load balancing
  - **SIMD:** loop vectorization, intrinsic functions
- Typical parameters for PICADOR:
  - 1 – 256 nodes
  - 16 – 96 CPU cores, 68 – 72 for Xeon Phi
  - 256- or 512-bit vector width
  - Overall 16 – 4096 cores



UNIFORM DISTRIBUTION OF PARTICLES. PERFORMANCE AND SCALING EFFICIENCY

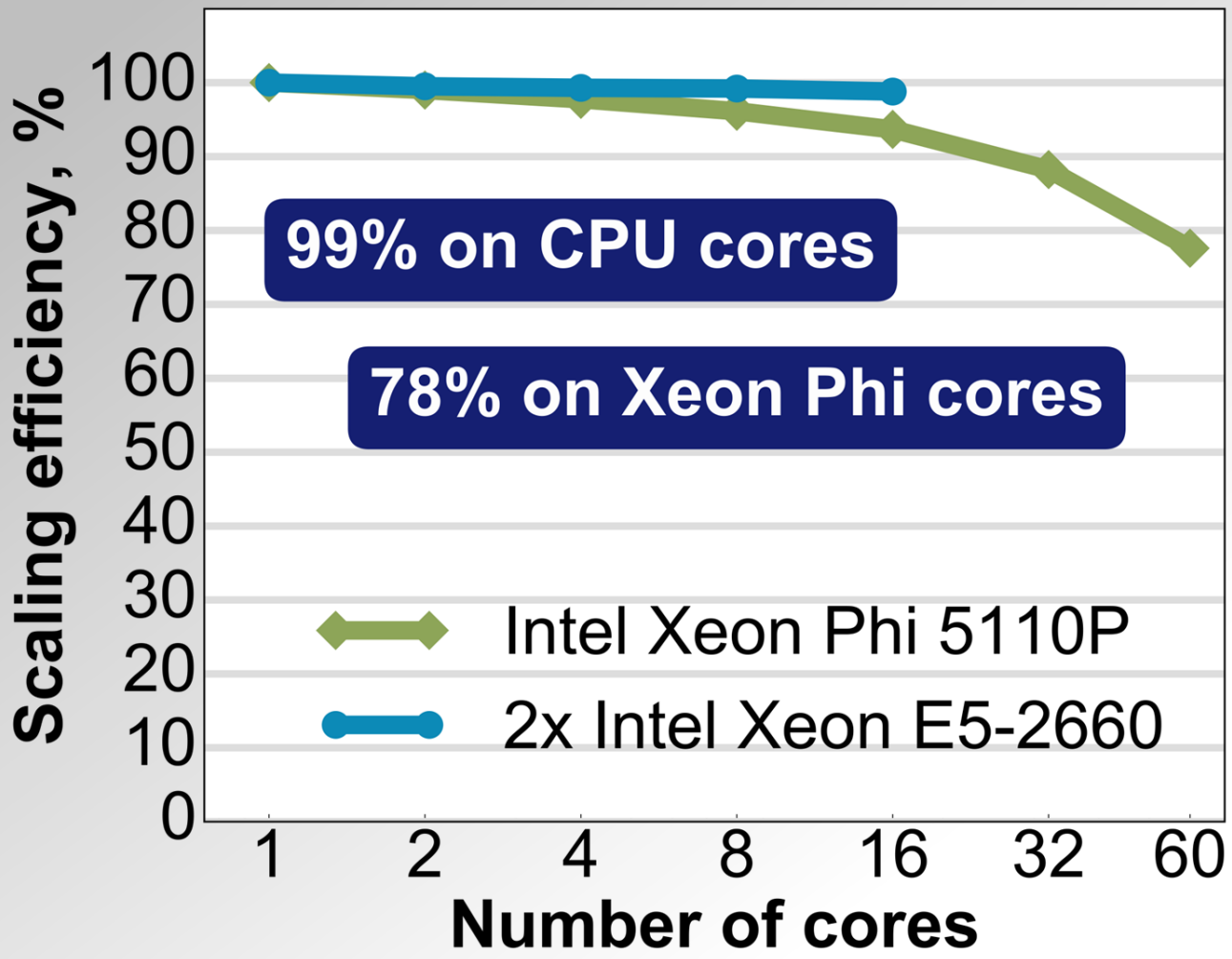
Scaling on Distributed Memory



Problem: laser wakefield acceleration

Parameters: 512×512×512 grid, 1015 mln. particles, TSC form factor

Scaling on Shared Memory

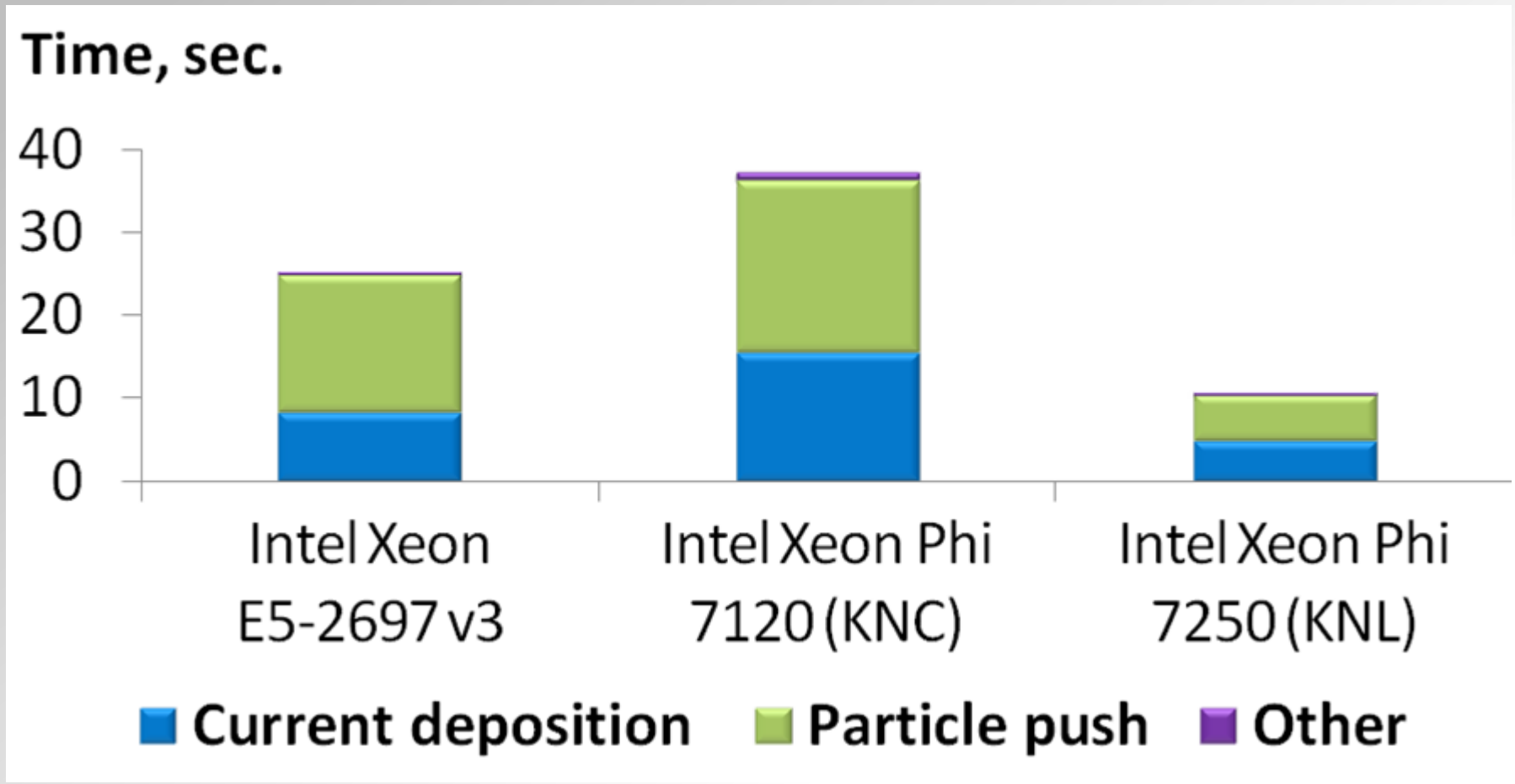


Simulation: frozen plasma benchmark

Parameters: 40×40×40 grid, 3.2 mln. particles, CIC form factor

Performance on Xeon Phi

KNL outperforms CPU by 2.35 x and KNC by 3.47 x



Scaling Efficiency and Performance

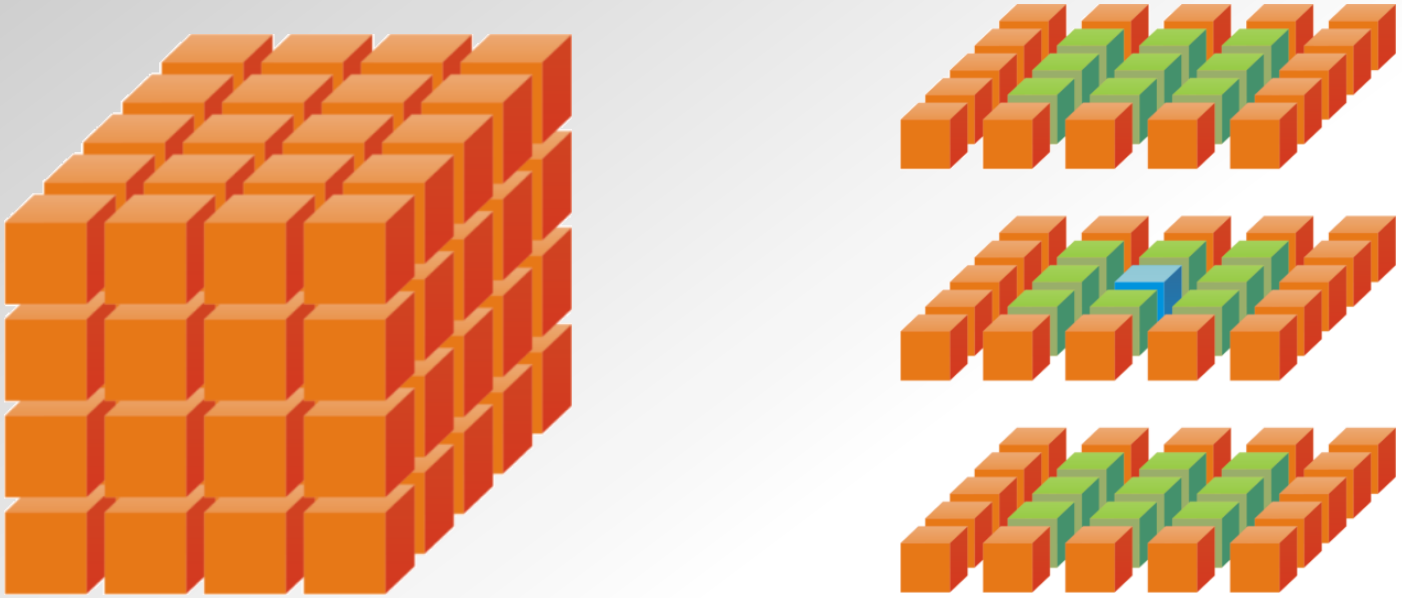
- **The distribution of particles** in a problem domain highly affects load imbalance, performance and scaling efficiency
- If the distribution of particles is **uniform**, the code scales reasonably up to thousands of cores
- If the distribution of particles is **non-uniform**, the code scales reasonably due to special load-balancing schemes (rectilinear partitioning…*).
- But if the distribution of particles is **non-uniform** and **dynamically varying**, we need custom schemes to overcome load imbalance.

* Surmin I. et al. Dynamic load balancing based on rectilinear partitioning in particle-in-cell plasma simulation. Int. Conf. on Par. Computing Technologies. Springer, Cham, 2015. P. 107-119.

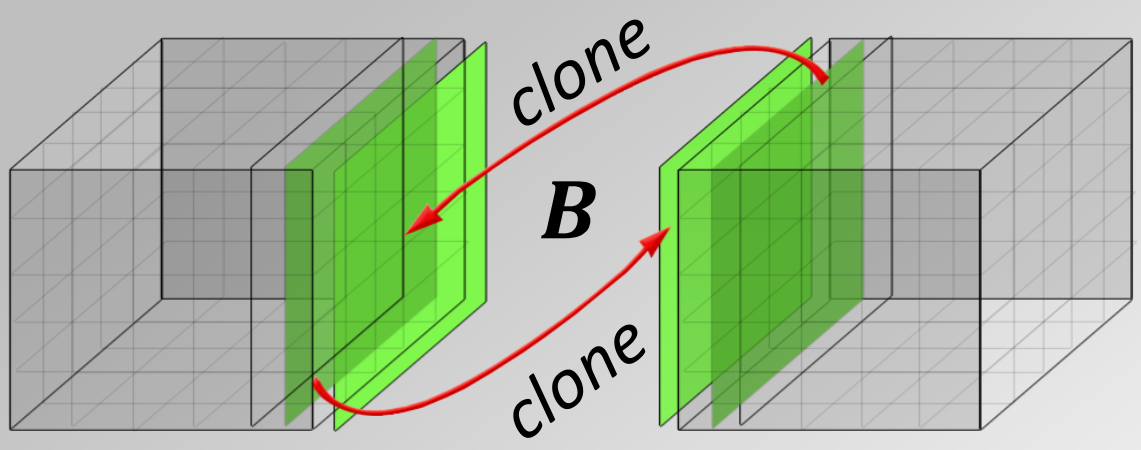
NON-UNIFORM DISTRIBUTION OF PARTICLES. LOAD BALANCING ON DISTRIBUTED MEMORY

Spatially Uniform Domain Decomposition

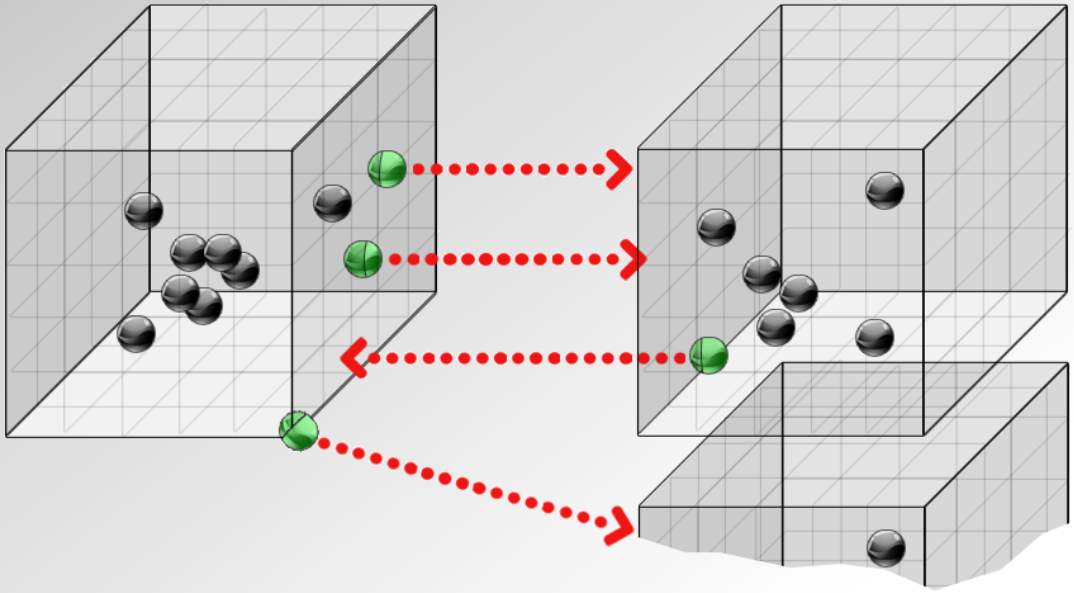
- Simulation area is axis-aligned box
- Spatial 3D domain decomposition into smaller boxes
- Each MPI process stores a set of particles and grid values in the corresponding subdomain
- Each process communicates only with neighbours



MPI Communication Pattern



Each process communicates with 6 neighbours



Each process communicates with 26 neighbours

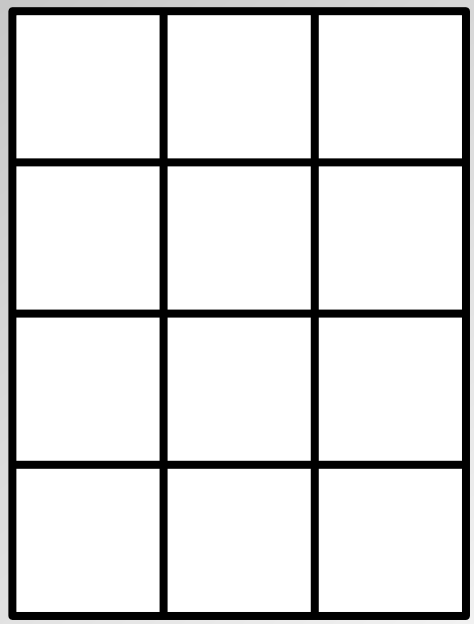
Load Balancing Overview

- Complexity of each time step: $\theta(nParticles + nCells)$
- Particle distribution can be significantly non-uniform and changing during the simulation
- Approaches to load balancing in Particle-in-Cell:
  - Recursive subdivision (octree, orthogonal bisection): good balancing, complex communication pattern
  - Floating boundaries (Quicksilver, OhHelp): ideal balancing, intensive exchanges of grid values
- **Our goal: simple communication pattern and low overhead at the cost of allowing little imbalance**

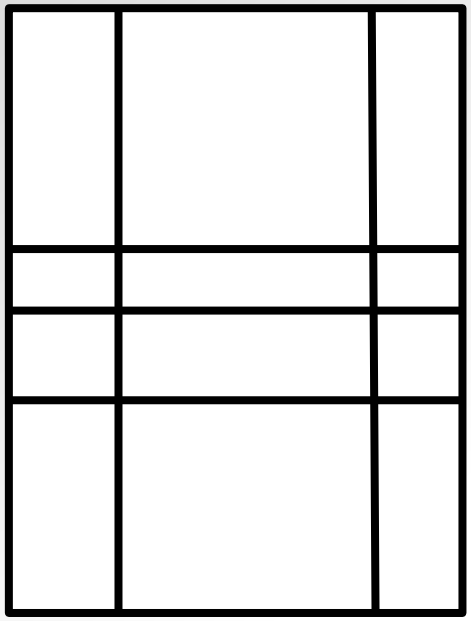
Rectilinear Partitioning

- Topologically equivalent to spatially uniform:
  - Each subdomain is axis-aligned box
  - Each subdomain has 26 neighbours

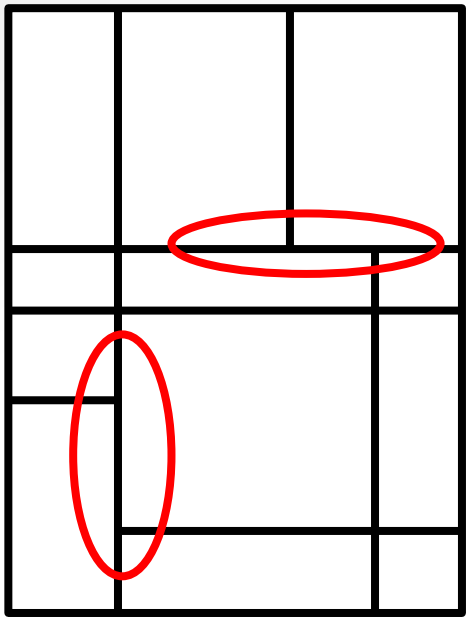
Spatially uniform



Rectilinear



Not rectilinear



Load Balancing Using Rectilinear Partitioning

- Cell workload

$$W_{i,j,k} = nParticles_{i,j,k} + 1$$

- Subdomain workload is a sum of workloads of its cells

$$W(SD) = \sum_{(i,j,k) \in SD} W_{i,j,k} = nParticles(SD) + nCells(SD)$$

- Optimal partitioning

$$P^* = \underset{P \in Partitionings}{\arg\min} \max_{SD \in Subdomains(P)} W(SD)$$

- Imbalance: max SD workload / average SD workload

Load Balancing Using Rectilinear Partitioning

- Finding optimal 3D rectilinear partitioning is NP-complete
- But tractable in 1D
- The number of subdomains along each axis is fixed
- Heuristic algorithm [Nicol, 1994]:
  - Fix decomposition along 2 axes
  - Find optimal 1D decomposition
  - Iteratively repeat for another axis
  - Stop when trying for all 3 axes does not improve

Implementation Overview

- Important to quickly compute imbalance for any trial decomposition on distributed memory
- Use parallel 3D prefix sums
- Static and dynamic load balancing
- Adjustable parameters for dynamic load balancing:
  - Frequency of imbalance estimation
  - Imbalance threshold to perform rebalancing

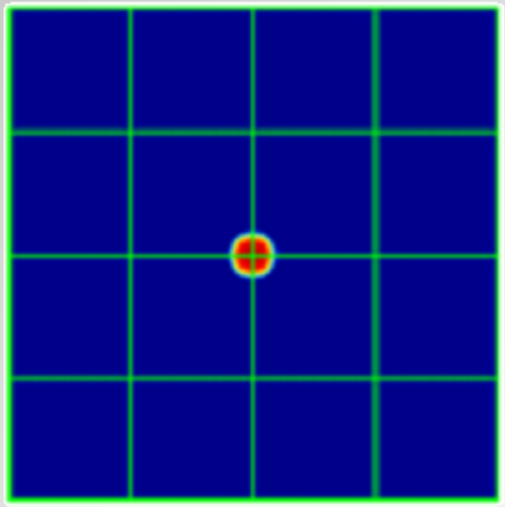
Computational Experiments

- Test problem:
  - Test plasma heating simulation
- Infrastructure:
  - MVS-100K of JSCC RAS
  - Intel Xeon E5450, 8 GB RAM, Infiniband DDR

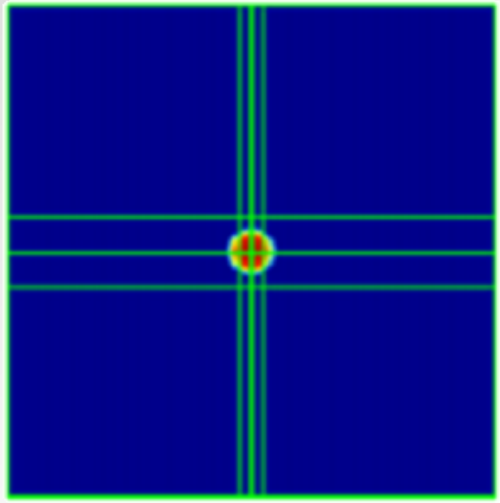
Test Plasma Heating Problem

- Initially small ball of plasma in the center
- Particles drift from the center in random directions
- 42 M particles, 128 × 128 × 128 grid, 256 MPI processes
- Dynamic balancing: check each 50 steps, threshold 1.2

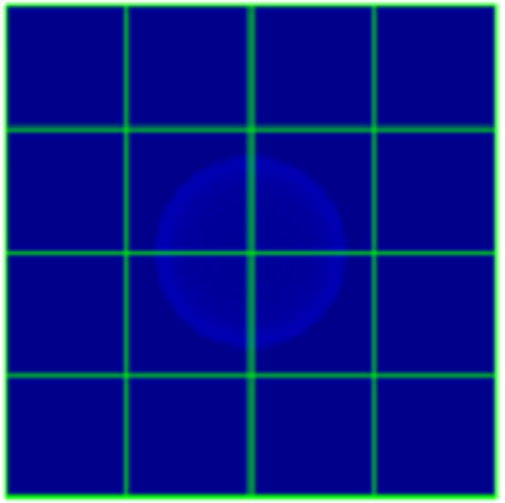
Initial uniform



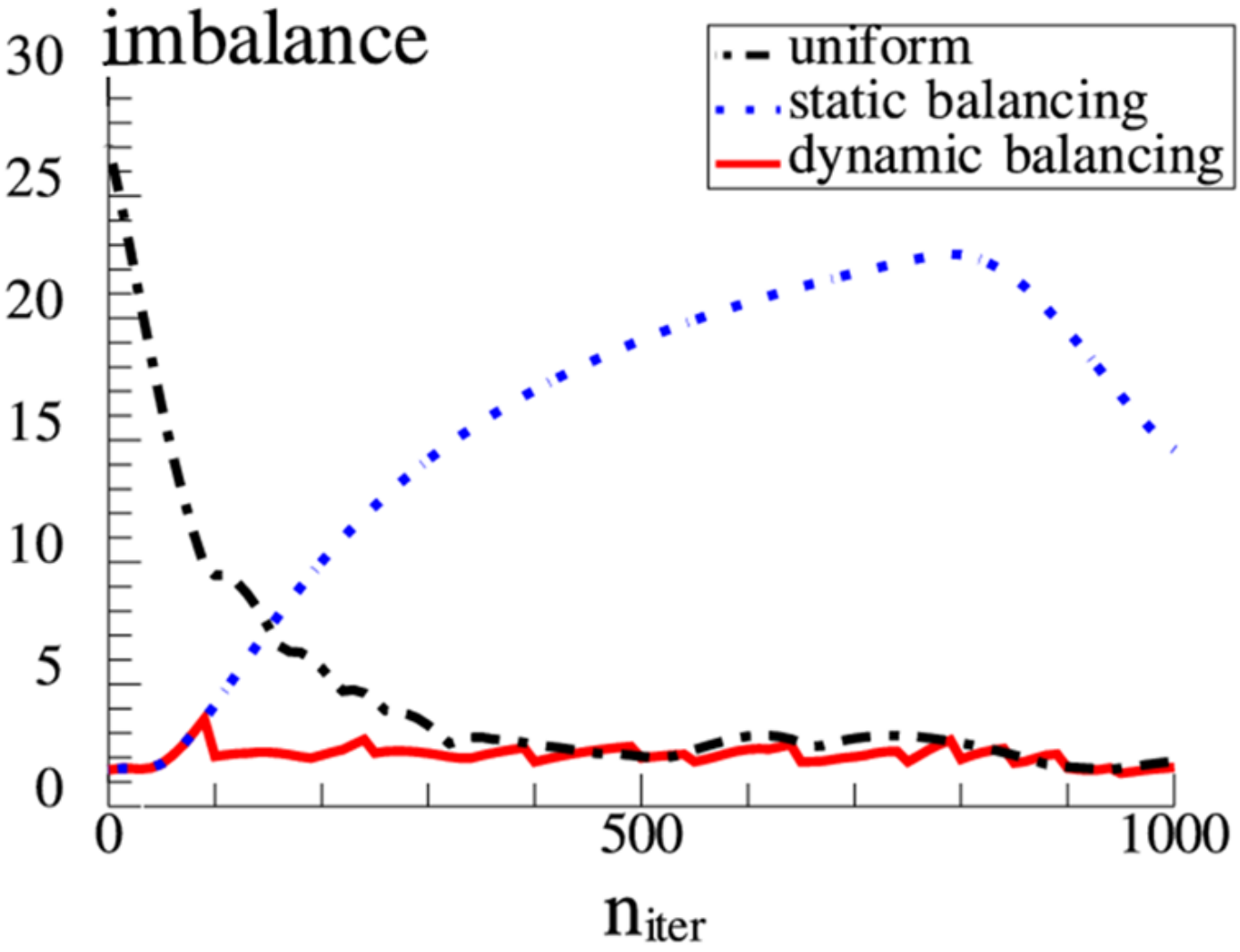
Initial rectilinear



Final uniform



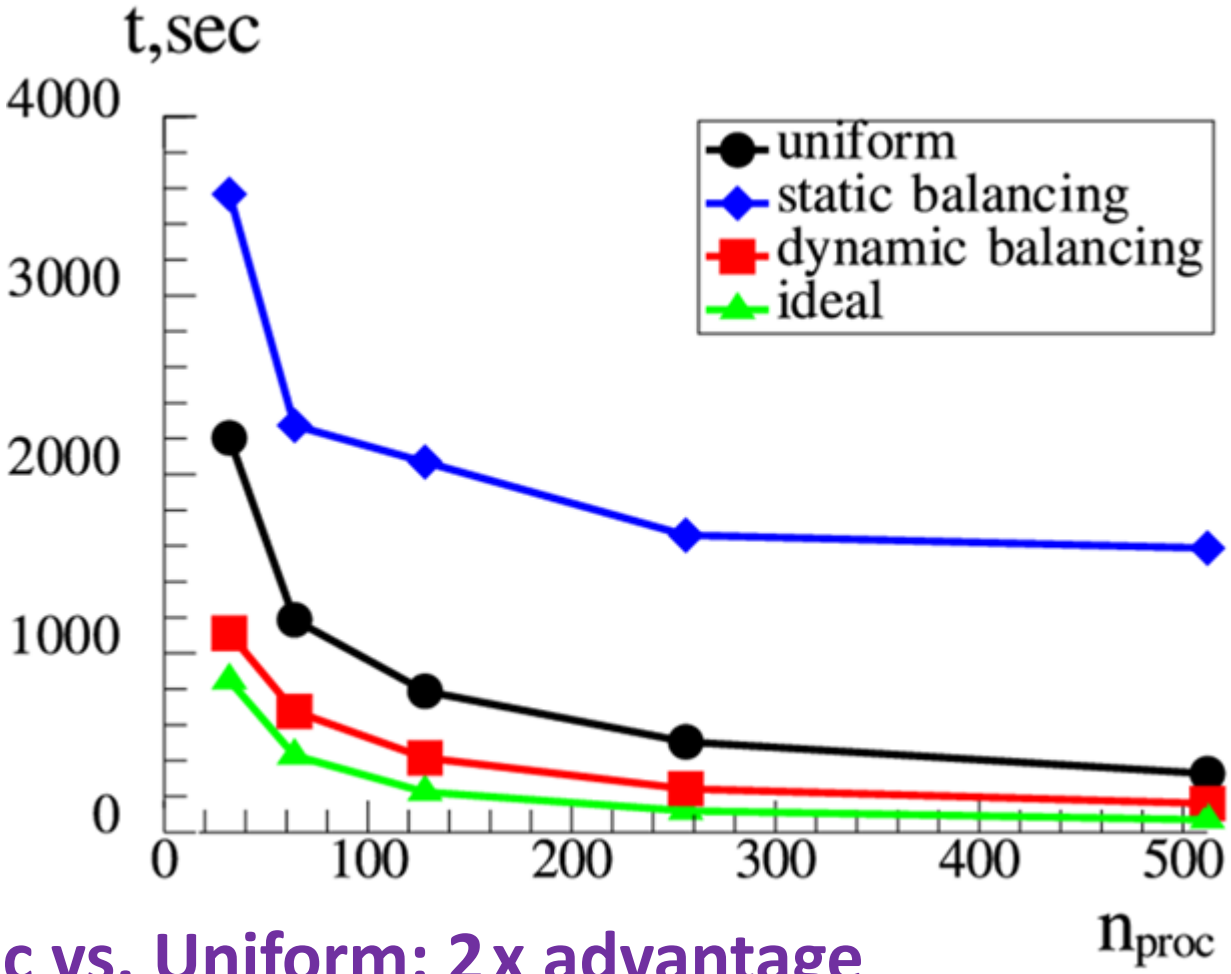
Test Problem: Imbalance



Dynamic load balancing overhead ≈ 1% of run time

Test Problem: Scaling Efficiency

Ideal – simulation of the same size with ideal balancing



Dynamic vs. Uniform: 2x advantage

Dynamic vs. Ideal: 1.5x disadvantage

NON-UNIFORM DISTRIBUTION OF PARTICLES. LOAD BALANCING ON SHARED MEMORY

Quantum Electrodynamics Effects

- In the case of **extremely strong electromagnetic fields**, the **QED processes** come into play.

- Charged particles accelerated in extreme laser field emit high-energy photons, which in turn can decay into a pair of electron and positron.

- These processes may lead to an avalanche like pair density growth leading to the development of the **QED cascades**

QED-PIC Method

- Before starting the QED-cascade, estimate the sub-step value
- Propagate the particle by sub-step to grid step in time with the Boris pusher
- At each time step, take into account a probabilistic emission of a photon by a particle or the generation of an electron-positron pair by a photon
- In certain simulations of interest, accounting for QED effects takes up to ***95%*** of the total QED-PIC run time.

Performance and Scalability Limiting Factors

- Due to the development of QED cascades the number of particles may **rapidly increase**
  - **in local regions** of a problem domain
  - **by many orders of magnitude**
- **Problem #1**: There is a need of preserving the use of **a reasonable number of particles** (reweighting particles using particle thinout or particle merging procedures*).
- **Problem #2**: We need to overcome **huge load imbalance** in many state-of-the-art simulations
  - highly affects scaling efficiency in **substantially non-uniform** and **dynamically varying distribution** of macroparticles in a computational area in QED simulations
  - requires the development of custom load balancing schemes

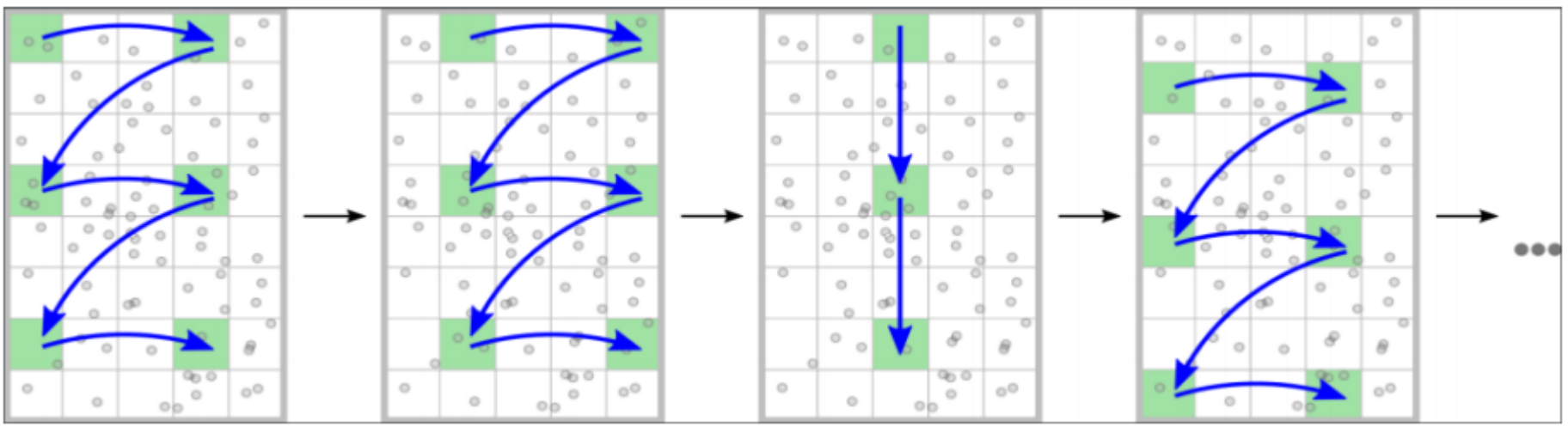
* Muraviev A. et al. Strategies for particle resampling in PIC simulations //arXiv preprint arXiv:2006.08593. – 2020.

Implementation Overview

- Particles are stored and processed **separately for each cell**
- During particle processing only local grid data is used
  - Preload all grid values needed for field interpolation
  - Accumulate currents in a local array
- Dependencies between particles which are close enough
  - Particle migration, at most to neighbor cell
  - Reduction for accumulated values of current density
- **Parallel processing scheme:**
  - Subdivide cells into several groups so that cells in a group can be processed independently
  - Process groups sequentially
  - Parallelize loops over cells inside a group using OpenMP

Baseline Parallel Processing Scheme

- A simplified example of cells split into four **walks**
- Particles are represented with the grey dots
- Cells inside each walk are processed **independently in parallel**
- Walks are performed sequentially with a barrier between walks



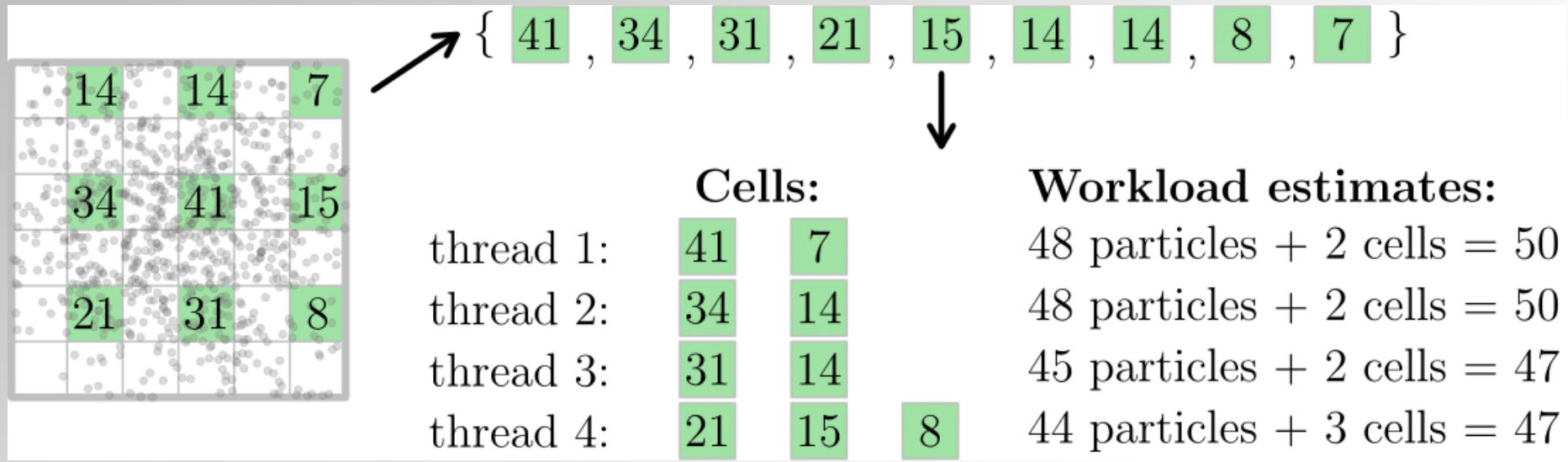
Workload Distribution*

- Inside each group of cells we just have a loop with independent iterations to parallelize
- ***OpenMP static***: standard static OpenMP schedule
  - Generally extremely good with chunk = 1
  - Inefficient for highly non-uniform particle distributions
- ***OpenMP dynamic***: standard dynamic OpenMP schedule
  - Potentially better balancing
  - Potentially large overhead
- ***Sorted dynamic***: sort by number of particles in cell, process in descending order, dynamic OpenMP schedule:
  - Definitely better balancing
  - Potentially large overhead

Larin, A., et al.: Load Balancing for Particle-in-Cell Plasma Simulation on Multicore Systems. In: Wyrzykowski R., Dongarra J., Deelman E., Karczewski K. (eds) PPAM 2017. LNCS, vol 10777. Springer, Cham (2018).

Workload Distribution*

- ***Manual distribution***:
  - Workload estimate for a cell: #particles + 1
  - Greedily distribute cells between threads



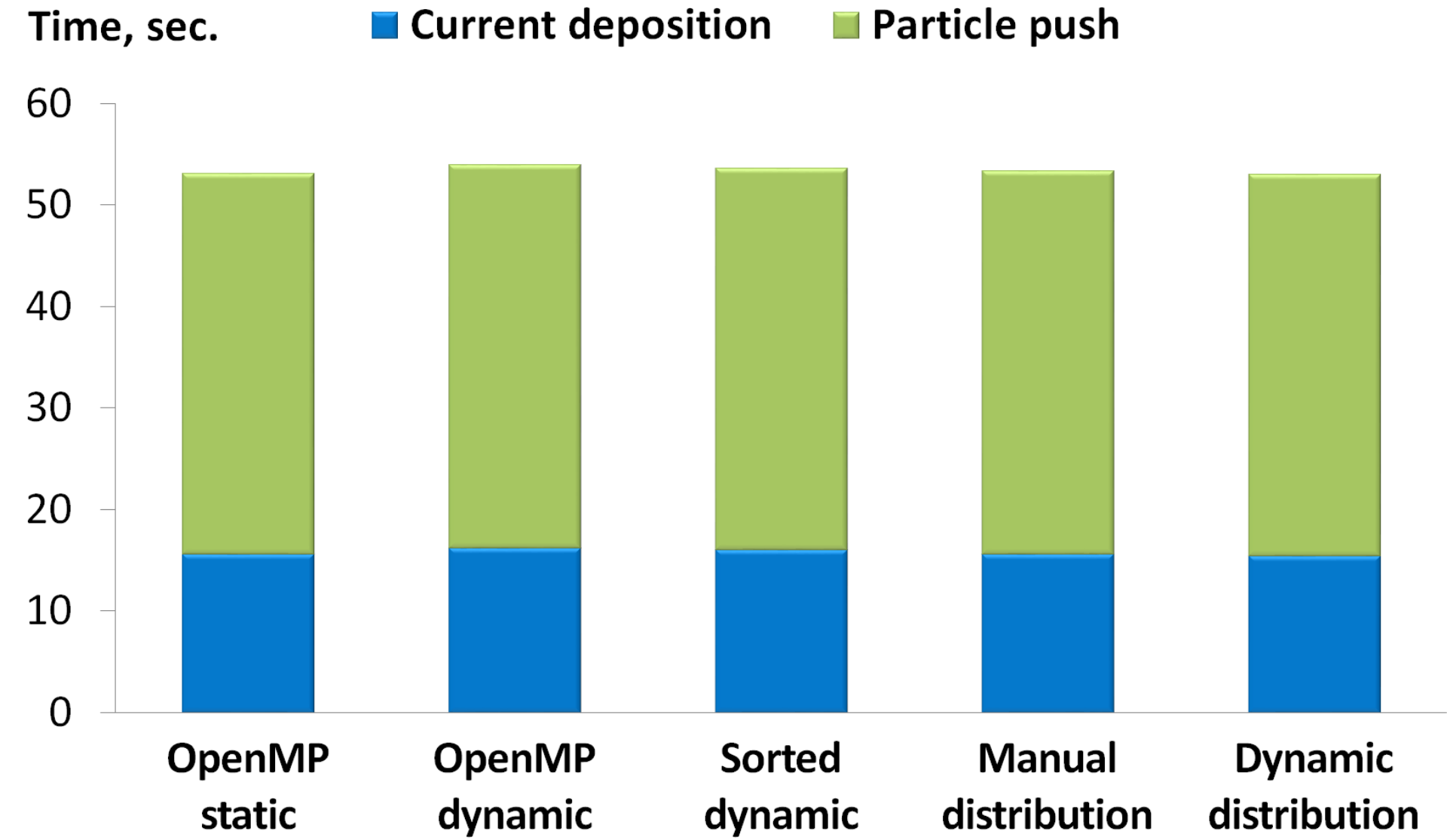
- ***Dynamic distribution***:
  - Perform Sorted dynamic scheme and save distribution
  - Use the same distribution for next K iterations (e.g. 100)

Larin, A., et al.: Load Balancing for Particle-in-Cell Plasma Simulation on Multicore Systems. In: Wyrzykowski R., Dongarra J., Deelman E., Karczewski K. (eds) PPAM 2017. LNCS, vol 10777. Springer, Cham (2018).

Test Problem

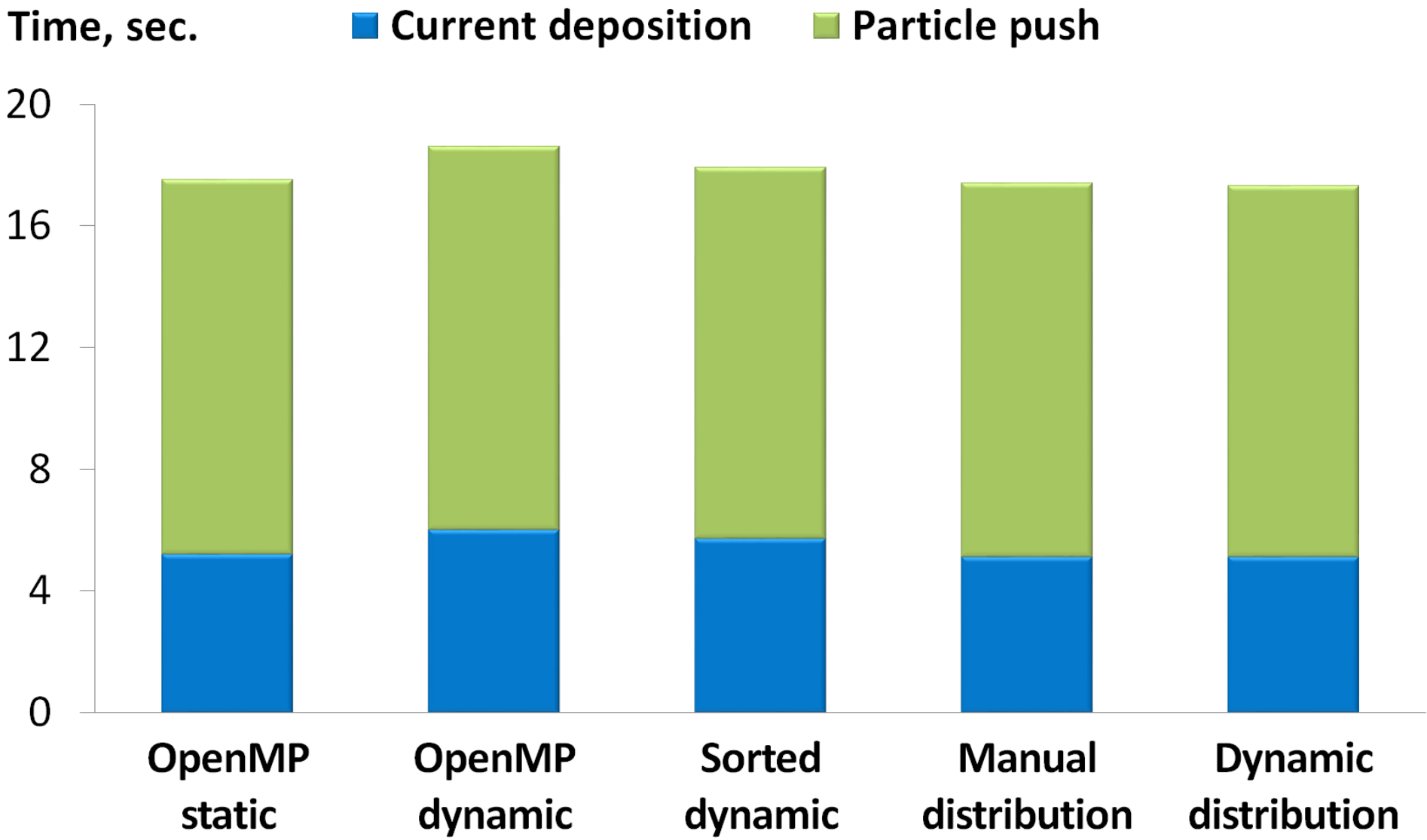
- 160 × 160 grid, average 100 particles per cell
- Normal distribution of particles
  - Mean at center of simulation area
  - Diagonal covariance matrix, same variance for x, y
  - Variance: $\sigma_1^2 = 25\Delta x/8$, $\sigma_2^2 = 2\sigma_1^2$, $\sigma_3^2 = 3\sigma_1^2$
  - All 3 values result in severely non-uniform distribution
- First-order form factor, direct current deposition
- 1000 time steps, distribution of particles does not change
- 24-core Intel Xeon E7-8890 v4 CPU at Intel Endeavor
- Intel C++ Compiler 17.0

Performance Evaluation: $\sigma_1^2 = 25\Delta x/8$, 24 cores



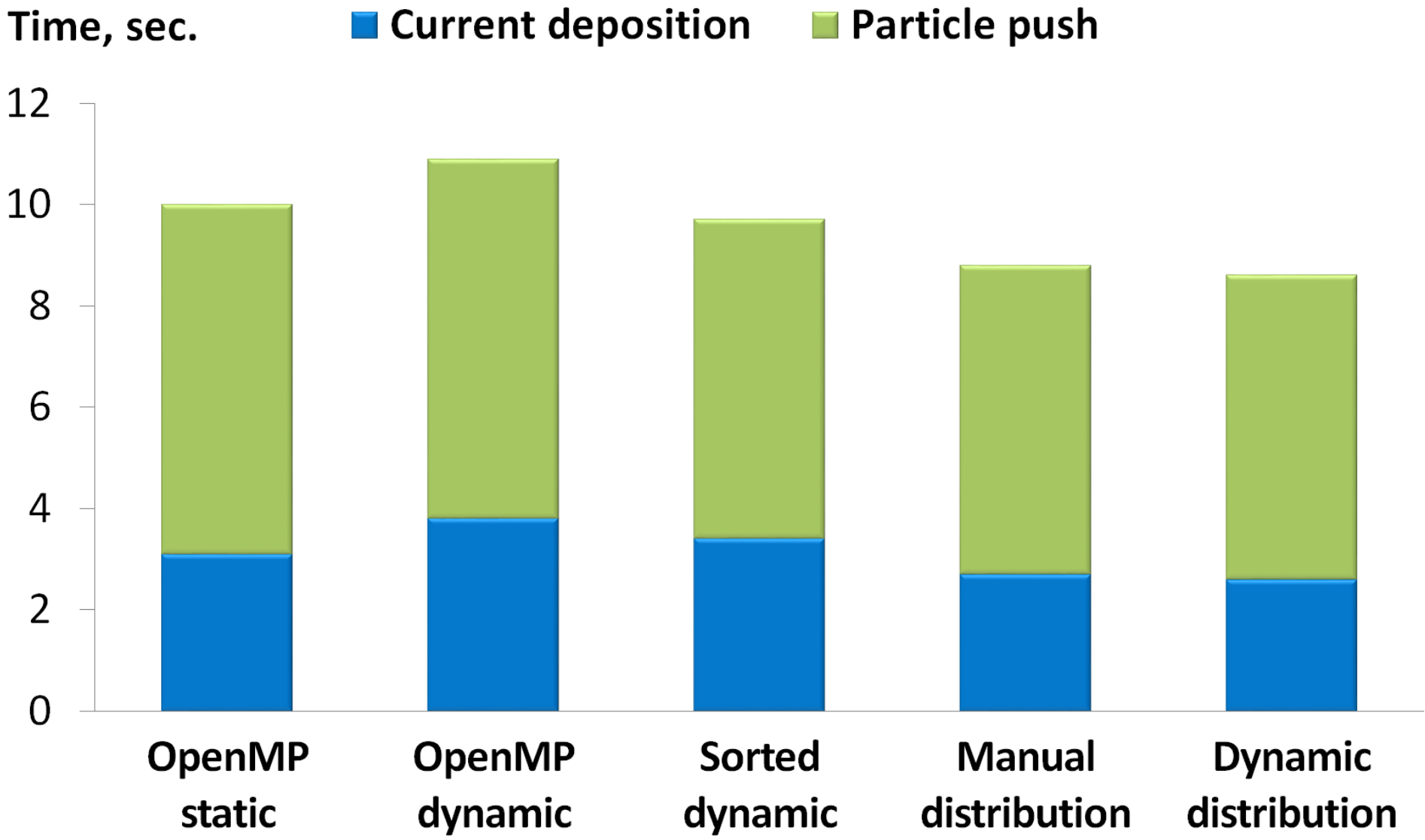
Larin, A., et al.: Load Balancing for Particle-in-Cell Plasma Simulation on Multicore Systems. In: Wyrzykowski R., Dongarra J., Deelman E., Karczewski K. (eds) PPAM 2017. LNCS, vol 10777. Springer, Cham (2018).

Performance Evaluation: $\sigma_2^2 = 2\sigma_1^2$, 24 cores



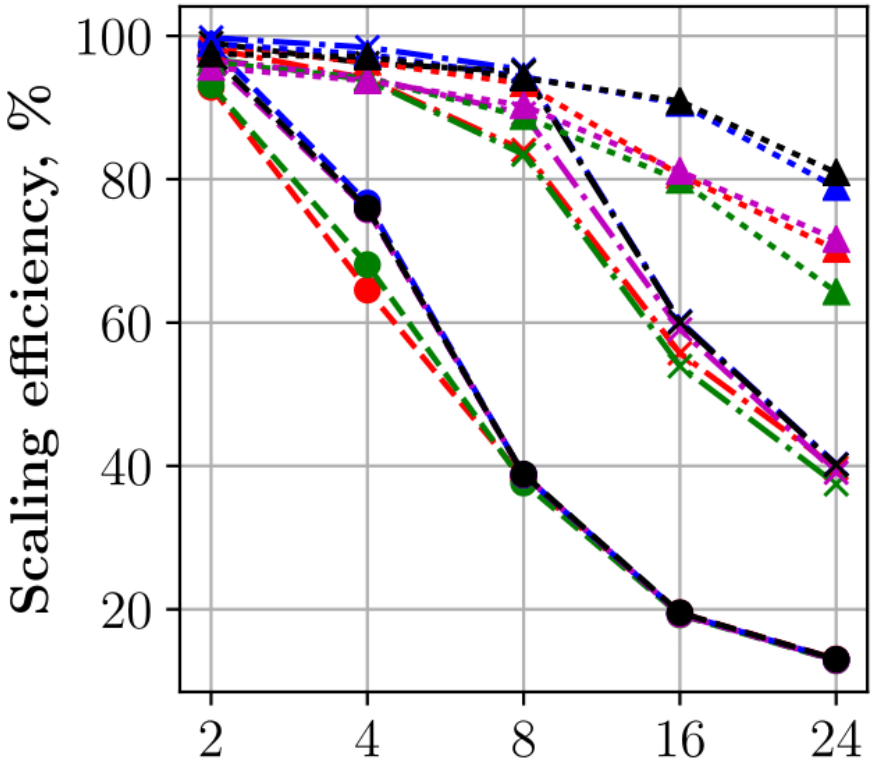
Larin, A., et al.: Load Balancing for Particle-in-Cell Plasma Simulation on Multicore Systems. In: Wyrzykowski R., Dongarra J., Deelman E., Karczewski K. (eds) PPAM 2017. LNCS, vol 10777. Springer, Cham (2018).

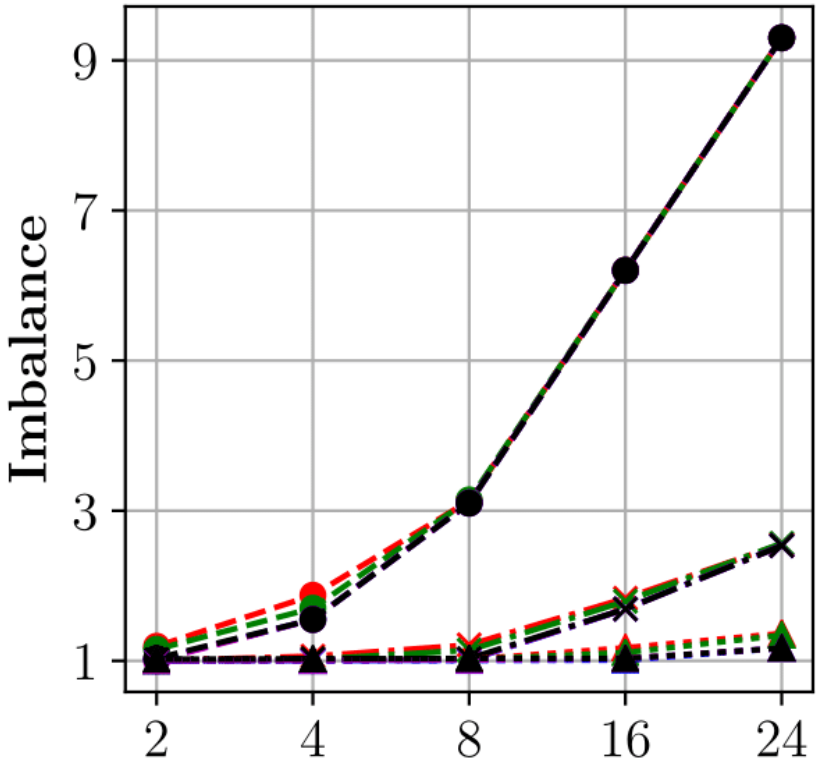
Performance Evaluation: $\sigma_3^2 = 3\sigma_1^2$, 24 cores



Larin, A., et al.: Load Balancing for Particle-in-Cell Plasma Simulation on Multicore Systems. In: Wyrzykowski R., Dongarra J., Deelman E., Karczewski K. (eds) PPAM 2017. LNCS, vol 10777. Springer, Cham (2018).

Scaling Efficiency





Variance:

σ_1^2 σ_2^2 σ_3^2

OpenMP static
OpenMP dynamic
Sorted dynamic
Manual distribution
Dynamic distribution

Larin, A., et al.: Load Balancing for Particle-in-Cell Plasma Simulation on Multicore Systems. In: Wyrzykowski R., Dongarra J., Deelman E., Karczewski K. (eds) PPAM 2017. LNCS, vol 10777. Springer, Cham (2018).

Real Problem…

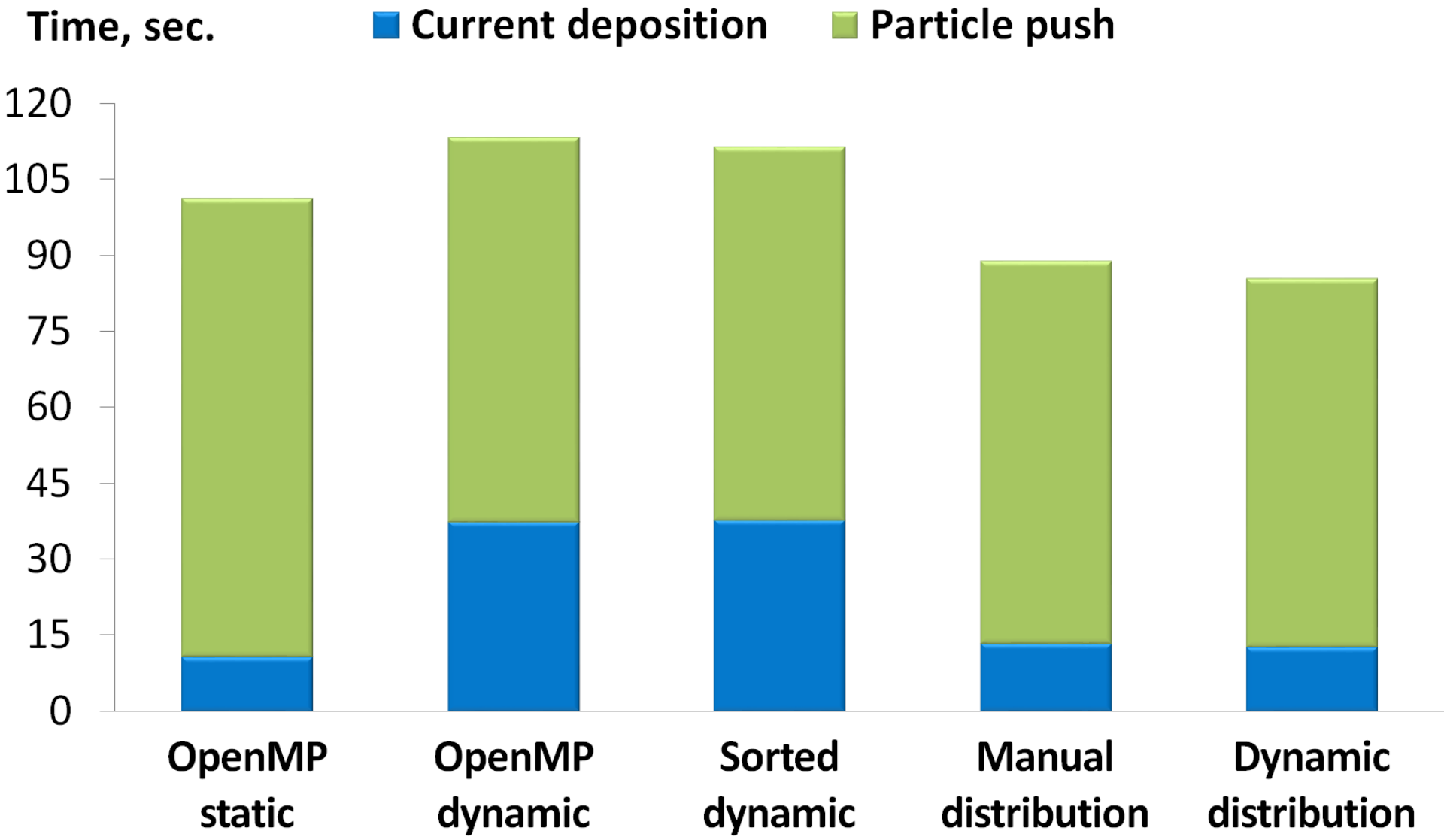
- Interaction of dense electron-positron plasma target with a cylindrical wave
  - 256 × 256 grid, initially average 18 particles per cell
  - Second-order form factor, direct current deposition
- Accounting for QED effects: photon generation and decay into electron-positron pairs, QED cascades
- Rapid increase in the number of particles in small areas could result in intricate particle distributions
- 4x 24-core Intel Xeon E7-8890 v4 CPUs at Intel Endeavor
- Intel C++ Compiler 17.0

Larin, A., et al.: Load Balancing for Particle-in-Cell Plasma Simulation on Multicore Systems. In: Wyrzykowski R., Dongarra J., Deelman E., Karczewski K. (eds) PPAM 2017. LNCS, vol 10777. Springer, Cham (2018).

Real Problem

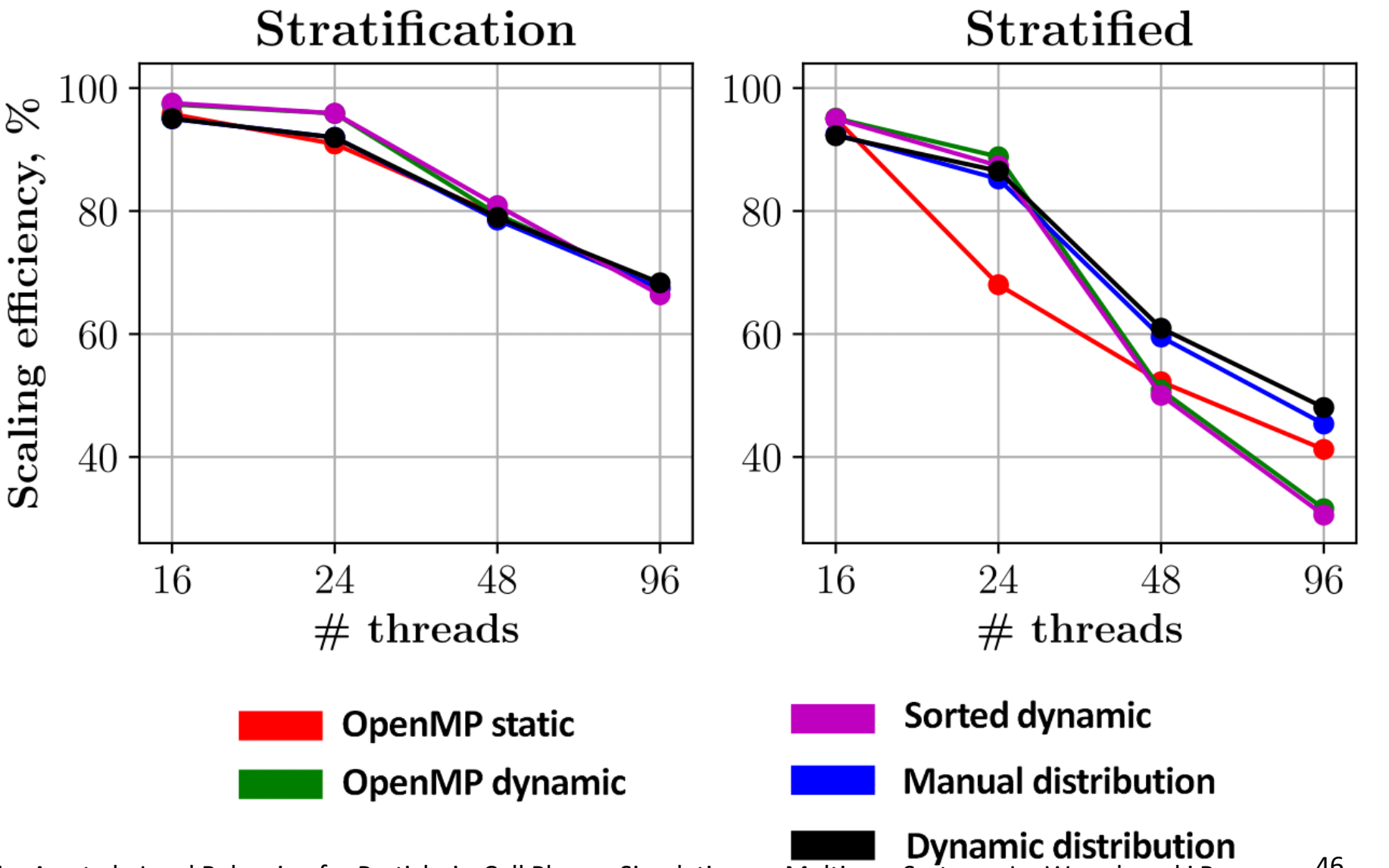
- Preparation phase: creating incoming cylindrical wave, particles are not injected yet
- Stratification phase: laser pulse penetrates target, compresses plasma and starts to create separated sheets
  - Particle distribution is not uniform and changes
  - However, number of particles in neighbor cells does not vary drastically
- Stratified phase: several separated current sheets, particles actively drift
  - Particle distribution is intricate and highly non-uniform
  - Could be significant difference between neighbor cells
  - More challenging for load balancing

Performance: Overall, 96 cores



Larin, A., et al.: Load Balancing for Particle-in-Cell Plasma Simulation on Multicore Systems. In: Wyrzykowski R., Dongarra J., Deelman E., Karczewski K. (eds) PPAM 2017. LNCS, vol 10777. Springer, Cham (2018).

Scaling Efficiency



Larin, A., et al.: Load Balancing for Particle-in-Cell Plasma Simulation on Multicore Systems. In: Wyrzykowski R., Dongarra J., Deelman E., Karczewski K. (eds) PPAM 2017. LNCS, vol 10777. Springer, Cham (2018).

Five Custom Schemes. Is it enough?

- For many PIC simulation scenarios particle distribution changes rather slowly relative to the cell size
- In such simulations one of the considered schemes provide **excellent load balancing**
- For **QED PIC** some cells can have **significantly more particles than others**
- Distribution of particles can **vary significantly over time**
- In such a case a cell is **too coarse** of a workload unit

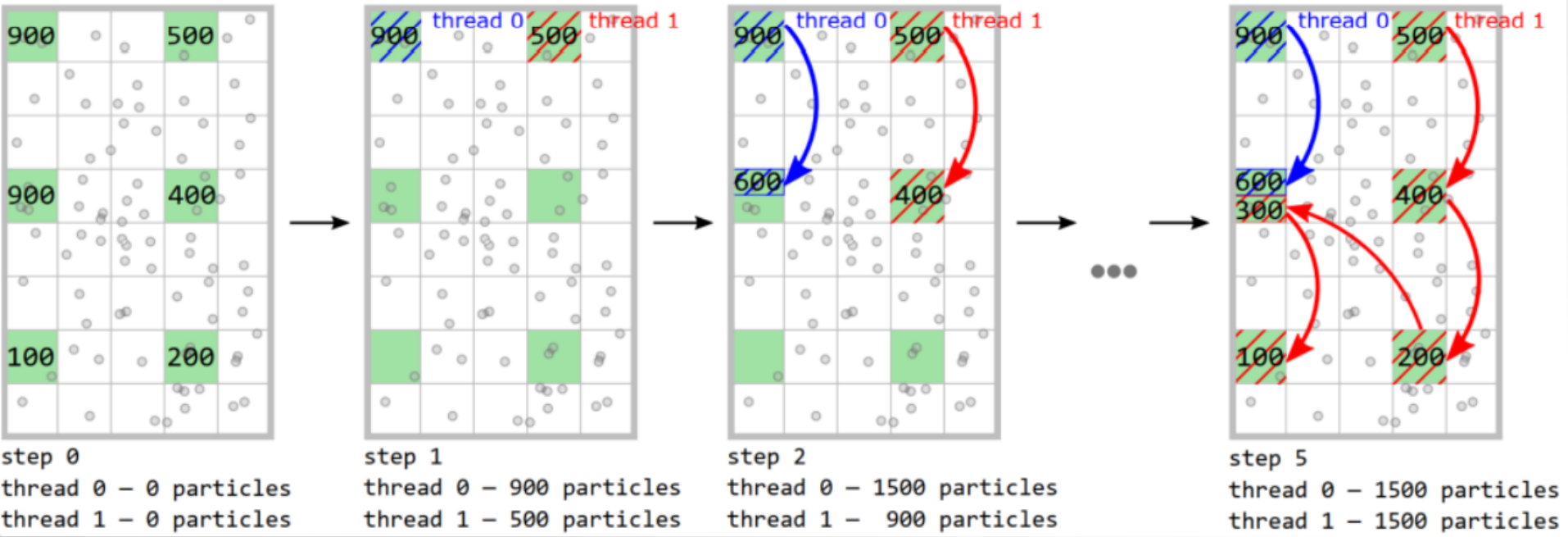
We developed a new load balancing scheme employing cell subdivision

New Dynamic Load Balancing Scheme

- **The main idea is to treat subsets of particles in a cell as separate pieces of work**
- This allows balancing the workload so that each thread processes almost the same number of particles
- The walks play the same role, but processing a cell consists of several tasks, each handling a subset of particles
- Tasks of the same cell are dependent, but tasks of different cells are not. Each thread has a queue of tasks in which no more than one task corresponds to a subdivided cell
- Thus, **a relatively small number of cells** not exceeding the number of threads **can be subdivided**
- Rebalancing is performed every **K** iterations. If the imbalance is too high, **K** decreases

Meyerov I. et al. Exploiting Parallelism on Shared Memory in the QED PIC Code PICADOR with Greedy Load Balancing . Int. Conf. on Par. Proc. and Appl. Math. Springer, 2019. P. 335-347.

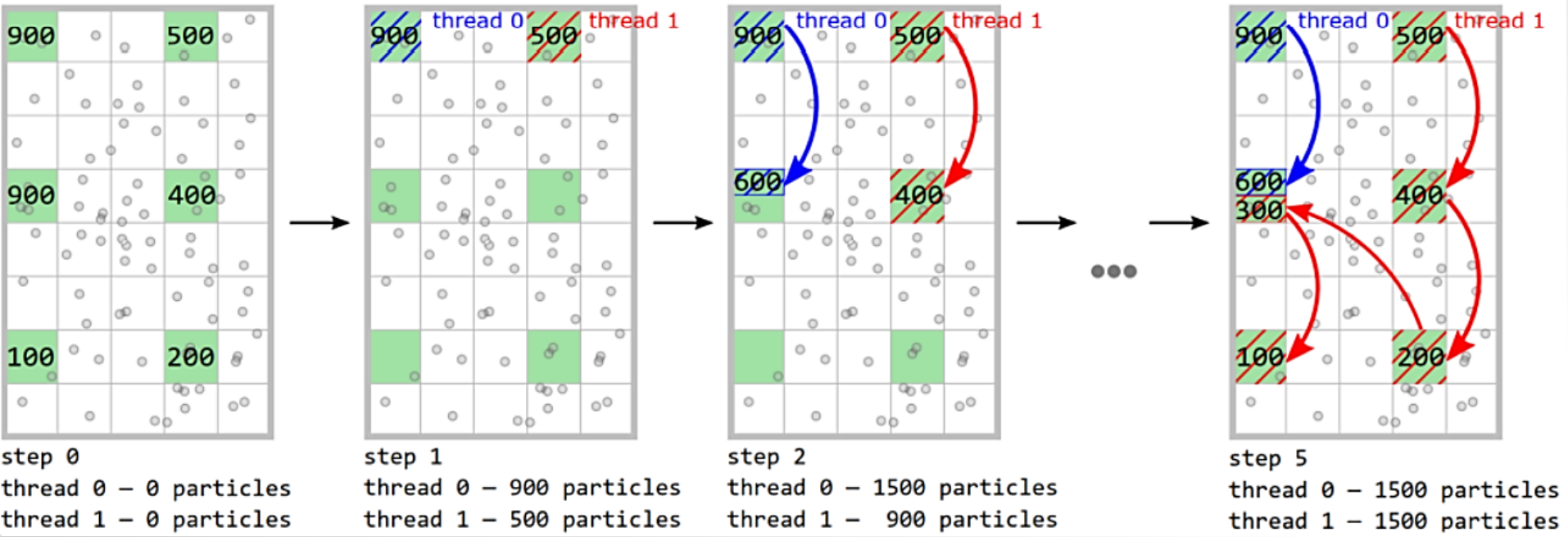
Example of the New Scheme Applied to a Single Walk



- The numbers represent the amounts of particles in cells
- The blue and red arrows illustrate two threads working in parallel
- One of the cells is subdivided into two tasks

Meyerov I. et al. Exploiting Parallelism on Shared Memory in the QED PIC Code PICADOR with Greedy Load Balancing . Int. Conf. on Par. Proc. and Appl. Math. Springer, 2019. P. 335-347.

Example of the New Scheme Applied to a Single Walk

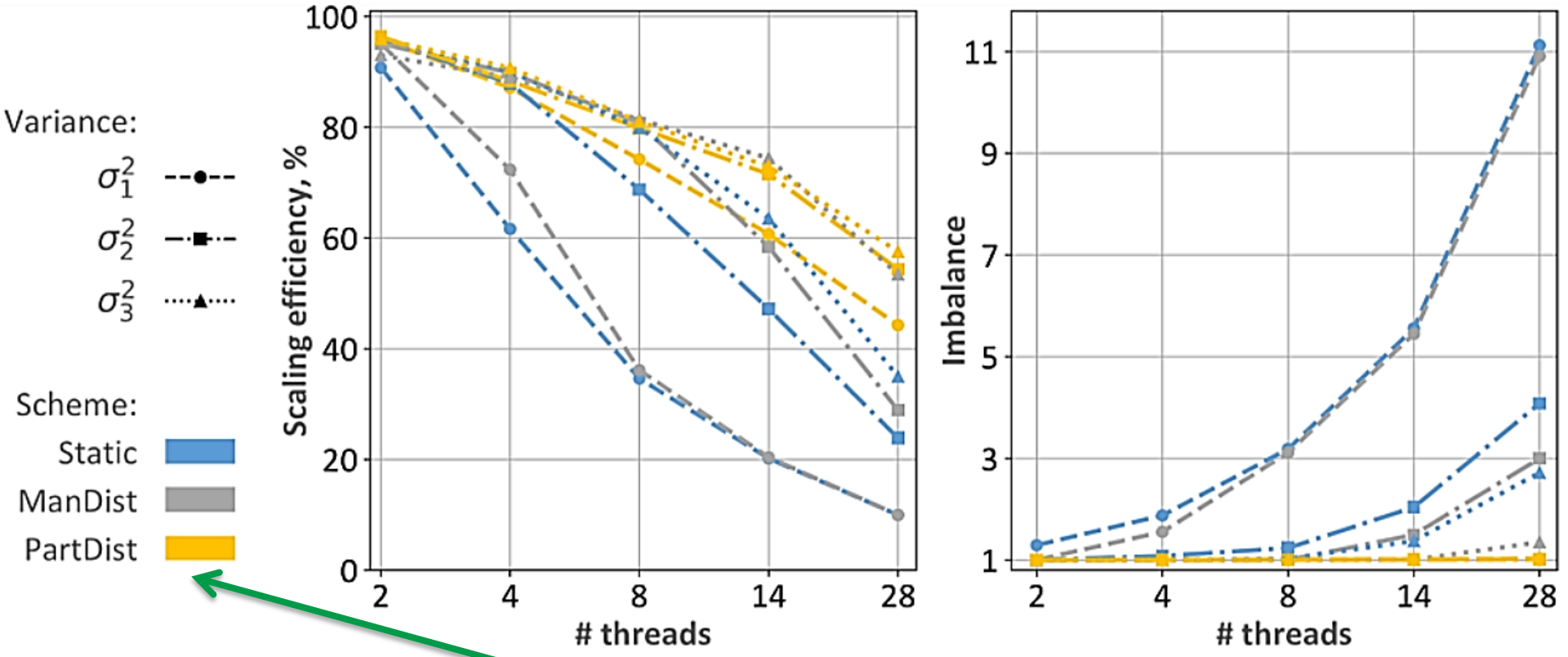


- Greedy algorithm (linear-time in terms of # cells)
- The cell with particles is added to the tasks queue of the thread **if the total size of the tasks in the queue does not violate the ideal balance by more than M times**
- Otherwise, the cell is divided into two parts to provide the ideal balance.

Test Problem

- 2D 160 × 160 grid, average 100 particles per cell
- Normal distribution of particles
  - Mean at the center of the simulation area
  - Diagonal covariance matrix, same variance for x, y
  - Variance: $\sigma_1^2 = 25\Delta x/8$, $\sigma_2^2 = 2\sigma_1^2$, $\sigma_3^2 = 3\sigma_1^2$
  - All 3 values result in severely non-uniform distribution
- First-order form factor, direct current deposition
- 1000 time steps, distribution of particles does not change
- 2 × Intel Xeon Gold 6132 (28 cores overall), 192 GB RAM at Intel Endeavor

Scaling Efficiency



For the first (most unbalanced) problem, **the new scheme (PartDist)** outperforms the best of the others by factor of 4.4

Meyerov I. et al. Exploiting Parallelism on Shared Memory in the QED PIC Code PICADOR with Greedy Load Balancing . Int. Conf. on Par. Proc. and Appl. Math. Springer, 2019. P. 335-347.

QED Simulations

- **Highly unbalanced problem of the QED cascade development in extreme laser fields***
  - The maximum intensity of each of counter-propagating pulses is $I_0 = 10^{25}\ W/cm^2$
  - The wavelength is $0.8\mu m$
  - Half infinite pulses with a 1 wave period front edge.
  - An electron-positron plasma slab with width of one wavelength and density $1cm^{-3}$ serves as a seed and is located at the center of the simulation area.
  - Incident laser pulses compress seed plasma. Laser pulses overlap, standing wave is formed and a QED cascade starts to develop.
  - Plasma is highly localized in the vicinity of the antinode.

** Bashinov, A.V., et al.: Particle dynamics and spatial e−e+ density structures at QED cascading in circularly polarized standing waves, Phys. Rev. A 95, 042127 (2017)*

Electric field

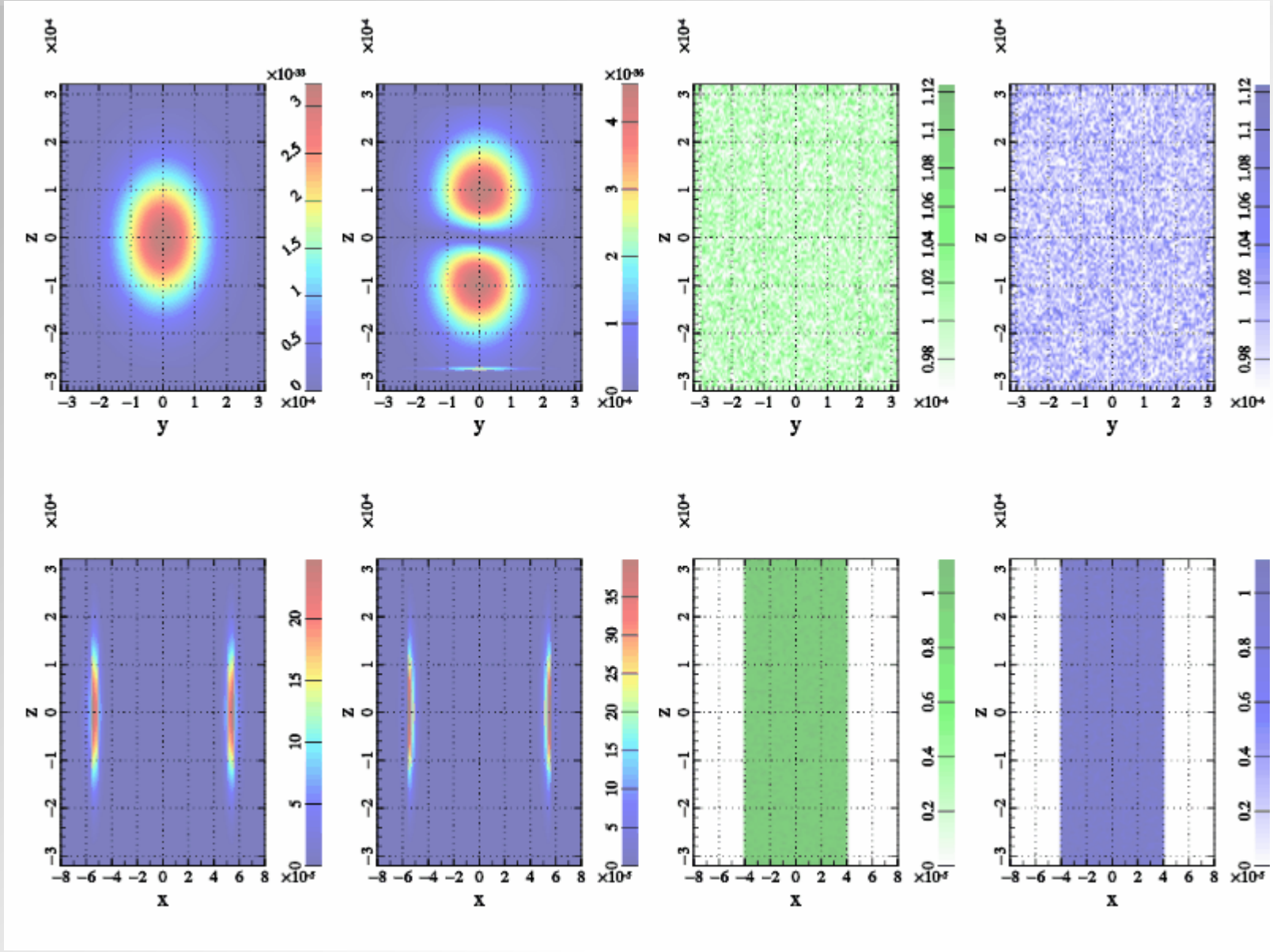
Magnetic field

Electron density

Positron density

yz plane

xz plane

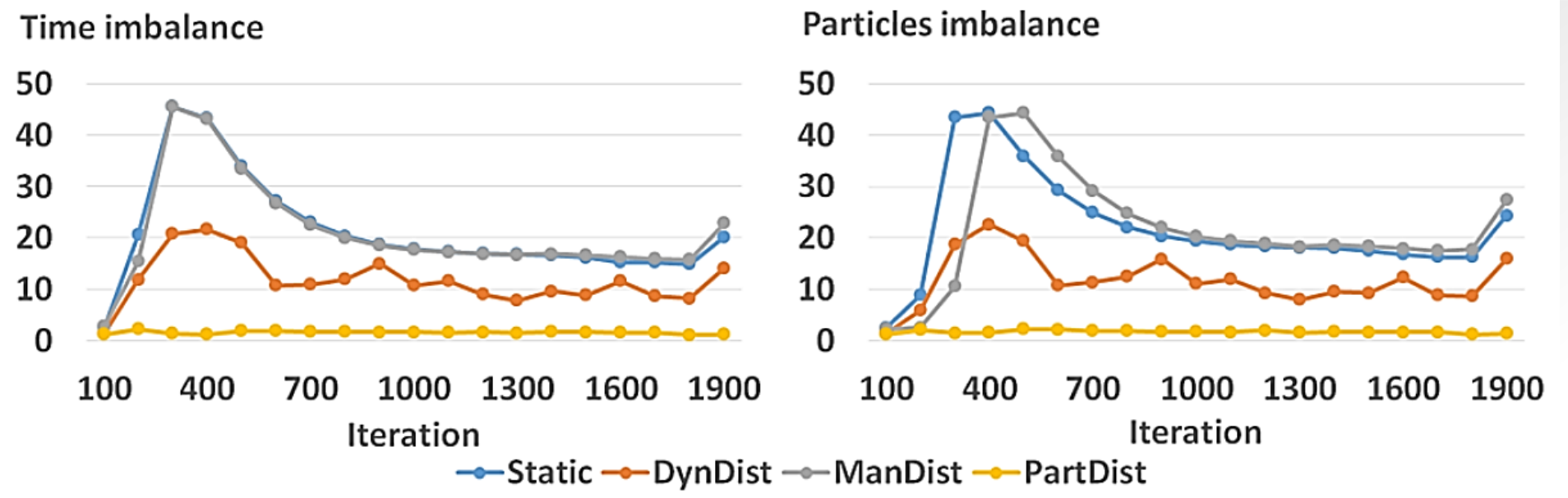


Computational Infrastructure/Test Problem

- **Intel Endeavor supercomputer with high-end CPUs of the Cascade Lake generation**
  - Cluster node: 2 × Intel Xeon Platinum 8260L CPU (48 cores overall), 192 GB of RAM.
  - 1 MPI process per socket, 2 OpenMP threads per core.
  - The code was built using the Intel Parallel Studio XE software package.
- **QED Simulations**

Problem	# Cells	Simulation area	Initial # particles	Thinning threshold	# cores
1D	128	2μm	10^6	2 x 10^6	48 (1 node)
2D	64 x 112	2μm x 8μm	5 x 10^6	5 x 10^6	96 (2 nodes)
3D	64 x 112 x 112	2μm x 8μm x 8μm	2.5 x 10^6	2.5 x 10^6	96 (2 nodes)

‘Time Imbalance’ vs. ‘Particles Imbalance’

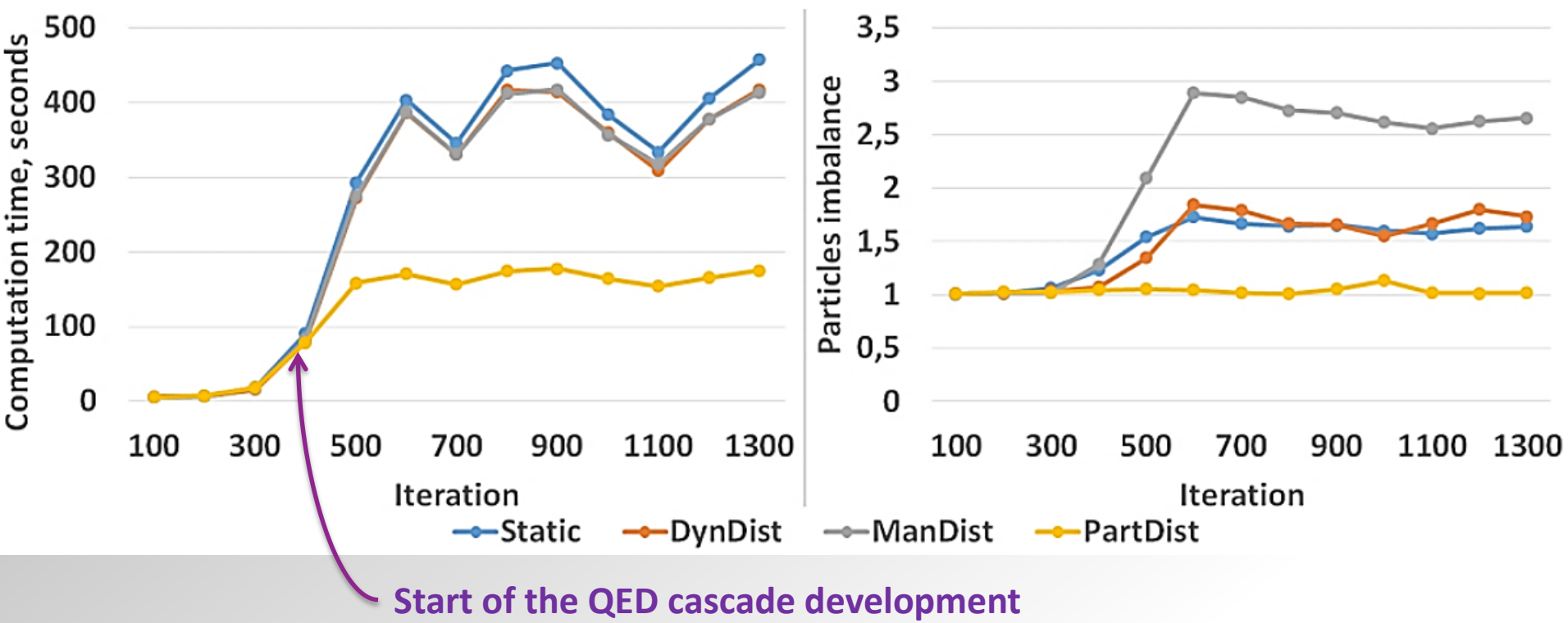


- **Time imbalance is measured by means of Intel Amplifier**

- **Particles imbalance is estimated as** $I = \max_w \left(\underset{i \in \{0,\dots,N-1\}}{\text{mean}} \left(\frac{\max_t P_{wti}}{\text{mean}_t P_{wti}} \right) \right)$
  - P_{wti} is a number of particles processed
    - by the thread t
    - within walk w
    - on i-th out of total N iterations

Meyerov I. et al. Int. Conf. on Par. Proc. and Appl. Math. Springer, 2019. P. 335-347.

Performance Results (3D)



- When calculating 1300 iterations, **the new scheme** speeds up the simulation
  - by **10 times** in **the 1D problem**
  - by **2.5 times** in **the 2D problem**
  - by **2.1 times** in **the 3D problem**

Meyerov I. et al.. Int. Conf. on Par. Proc. and Appl. Math. Springer, 2019. P. 335-347.

ONGOING RESEARCH

Ongoing Research…

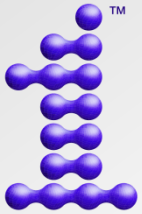
- The project **High-Intensity Collisions and Interactions (hi-Chi)** – an open-source collection of Python-controlled tools for performing simulations and data analysis in the research area of strong-field particle and plasma physics.
  - Open Source
  - Python + C++ (flexibility + HPC)
  - Easily extendable
- We plan to port hi-Chi to the oneAPI model

https://github.com/hi-chi

Ongoing Research…

- Low-precision and mixed precision computations
- Heterogeneous computing (oneAPI Center of Excellence)

https://github.com/hi-chi

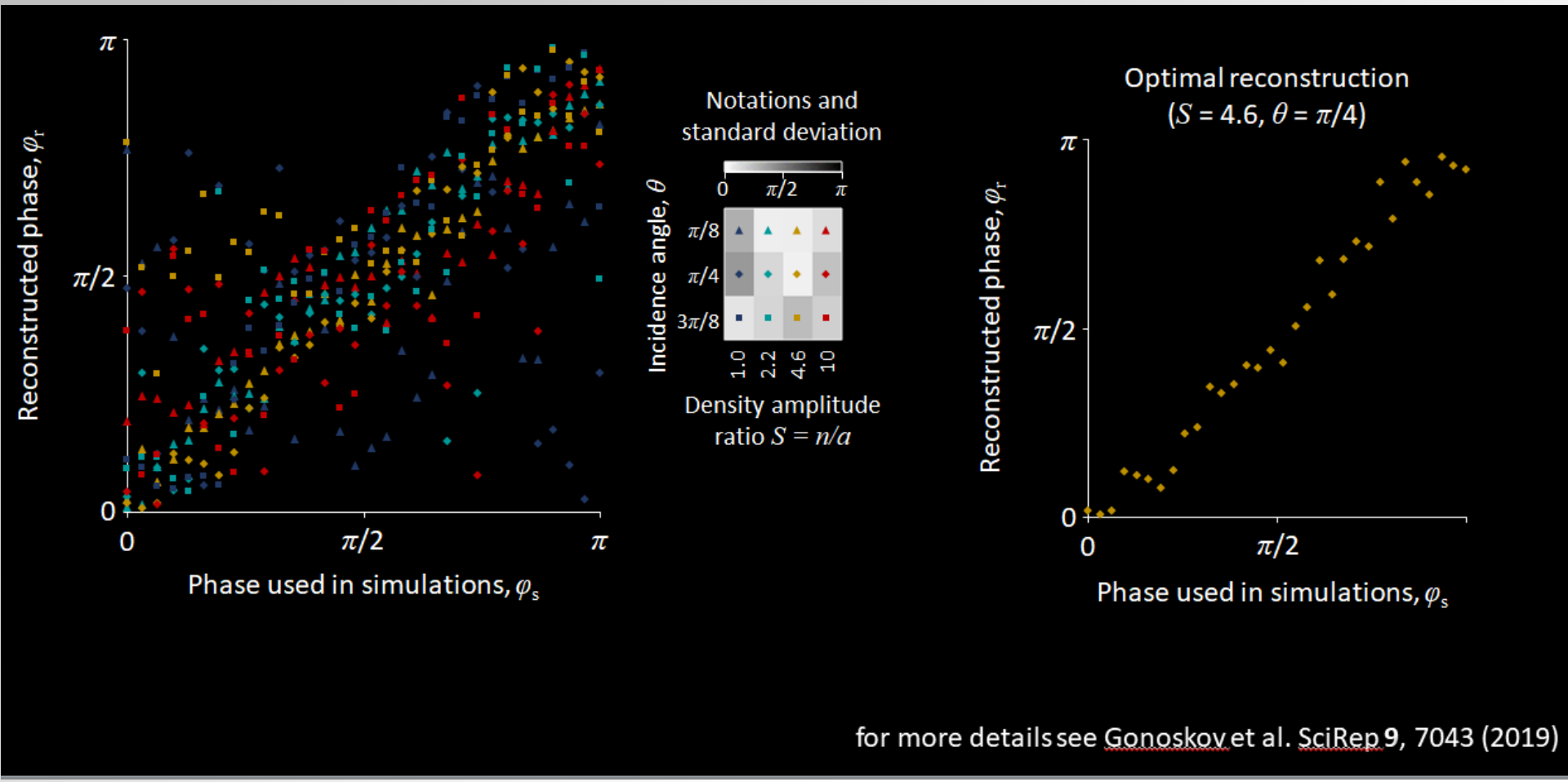


oneAPI

LOBACHEVSKY UNIVERSITY

Ongoing Research

- **Reconstructing experimental conditions with ML**



for more details see Gonoskov et al. SciRep **9**, 7043 (2019)

Summary

- We considered the computational loop of the PIC method and the problem of efficient utilization of modern CPUs in PIC simulations
- We discussed parallelization techniques and performance limiting factors
- We addressed a problem of improving load balancing in QED PIC simulations
- To overcome the load imbalance, we developed and implemented a special scheme in the PICADOR code that allows subdividing cells with a large number of particles
- This approach substantially increased the potential for parallelization

Contact

- Iosif Meyerov, PhD, associate professor, vice-chair of the Mathematical Software and Supercomputing Technologies department, Lobachevsky University
  meerov@vmk.unn.ru

- Project page:

  http://hpc-education.unn.ru/en/research/overview/laser-plasma

  https://github.com/hi-chi