Challenges in adapting Particle-In-Cell codes to GPUs and many-core platforms

L. Villard, T.M. Tran, F. Hariri*, E. Lanti, N. Ohana, S. Brunner

Swiss Plasma Center, EPFL, Lausanne

A. Jocksch, C. Gheller

CSCS, Lugano

A. Bleuler, R. Teyssier

University of Zurich

** Present address: CERN, Geneva*

Acknowledgements: P. Messmer, J. Progsch, *NVIDIA*

Lyon, 6 April 2016

Outline

- Introduction and motivation:
  - Gyrokinetic turbulence
  - PIC scheme
- PIC-engine: a test bed for PIC codes on many-core heterogeneous architectures
- Drift-Kinetic-engine: a test bed for PIC simulations of magnetized plasmas
- Towards full applications
- Conclusions

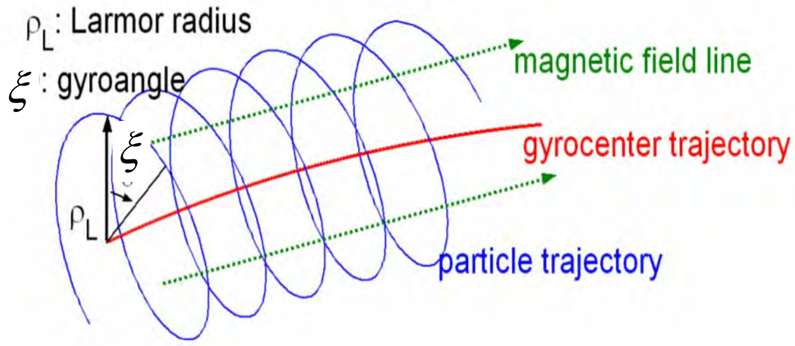
Introduction: motivation

- Particle-In-Cell (PIC) codes are used for many applications, in particular plasma physics: GTC [Z. Lin], ORB5 [T.M.Tran], GT3D [Y. Idomura], and astrophysics: RAMSES [R. Teyssier]
- The aim is to investigate how PIC codes can make efficient use of new and emerging HPC architectures, in particular many-core, hybrid. [Decyck2011, Madduri2011, Tang2014]
- Another important issue is how to deal with legacy codes in various domain science applications
- This has formed the basis for a PASC Co-Design project (Platform for Advanced Scientific Computing), funded at the Swiss Confederation level and led by the CSCS, the Swiss national Supercomputing Centre
- On top of the generic PIC approach, two specific physics applications are targeted :
  - (a) gyrokinetic simulations of magnetized plasmas (ORB5)
  - (b) gravitational problems, e.g. dark matter (RAMSES)

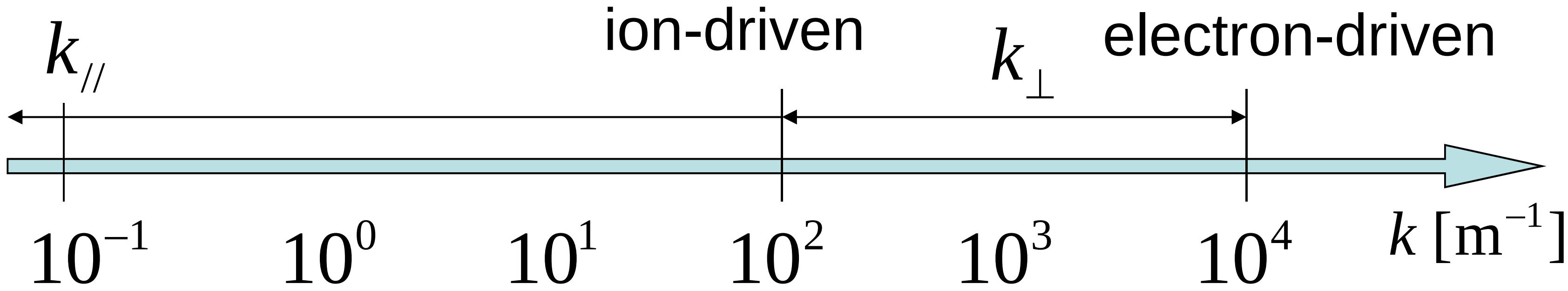
Gyrokinetic model

- Assume $\omega_{\text{turbulence}} \ll \omega_{\text{ion cyclotron}}$
- Average out the fast motion of the particle around the guiding center
- Fast parallel motion, slow perpendicular motion (drifts)

- Strong anisotropy of turbulent perturbations (// vs perp to **B**)



phase space dimension reduction ***6D*** *--->* ***5D***



Gyrokinetic equations

$f_s(\vec{R}, v_{//}, \mu)$ distribution function of species *s* in **5D** phase space

$$\frac{\partial f_s}{\partial t} + \frac{d\vec{R}}{dt} \cdot \frac{\partial f_s}{\partial \vec{R}} + \frac{dv_{//}}{dt} \frac{\partial f_s}{\partial \vec{R}} = C(f_s, f_{s'})$$

advection-diffusion
PDE, 5D

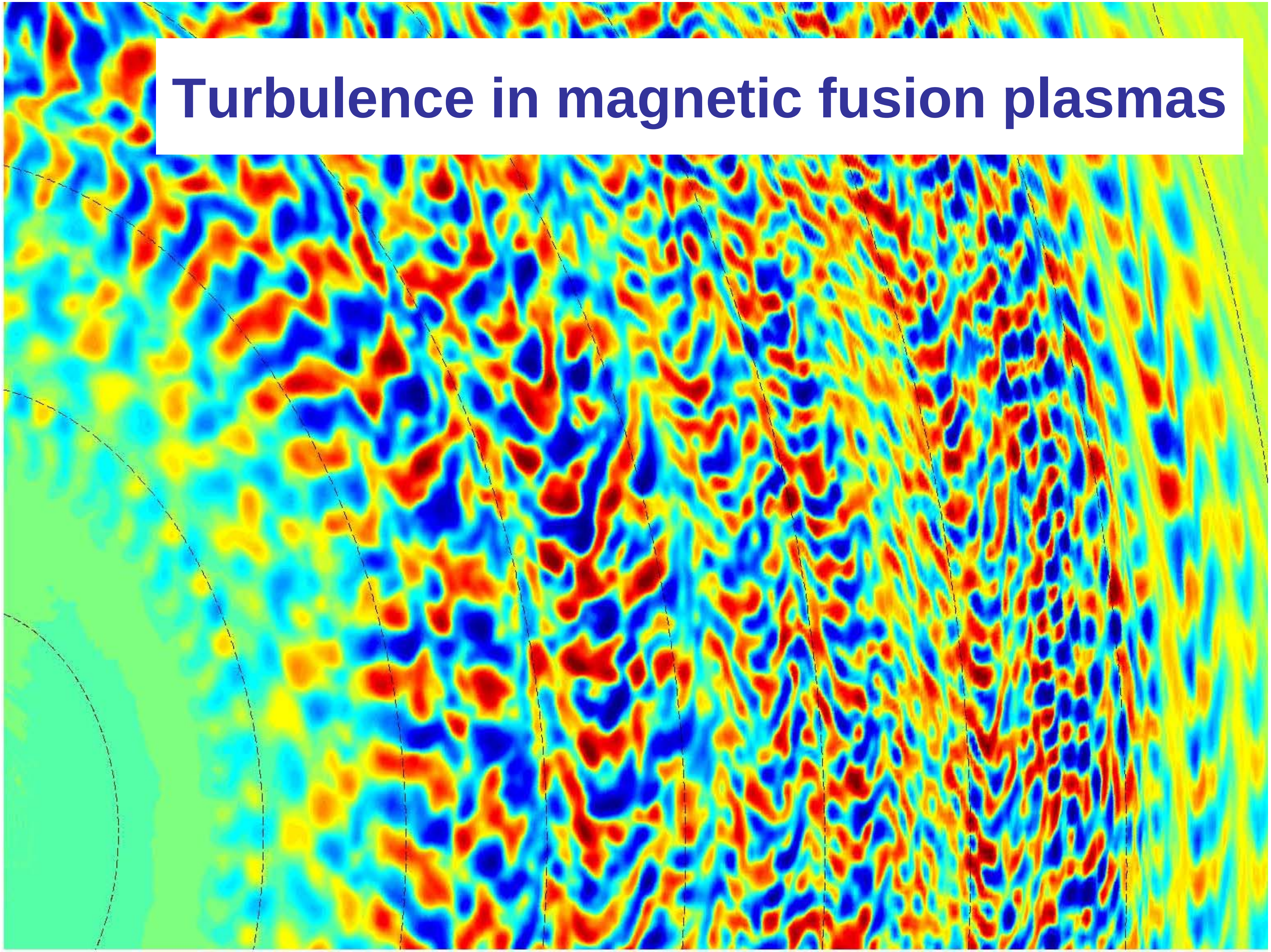
$$\frac{d\vec{R}}{dt} = \ldots \mathrm{fct}(\phi, \vec{A}), \quad \frac{dv_{//}}{dt} = \ldots \mathrm{fct}(\phi, \vec{A})$$

equations of motion
(orbits)
ODE, 5D

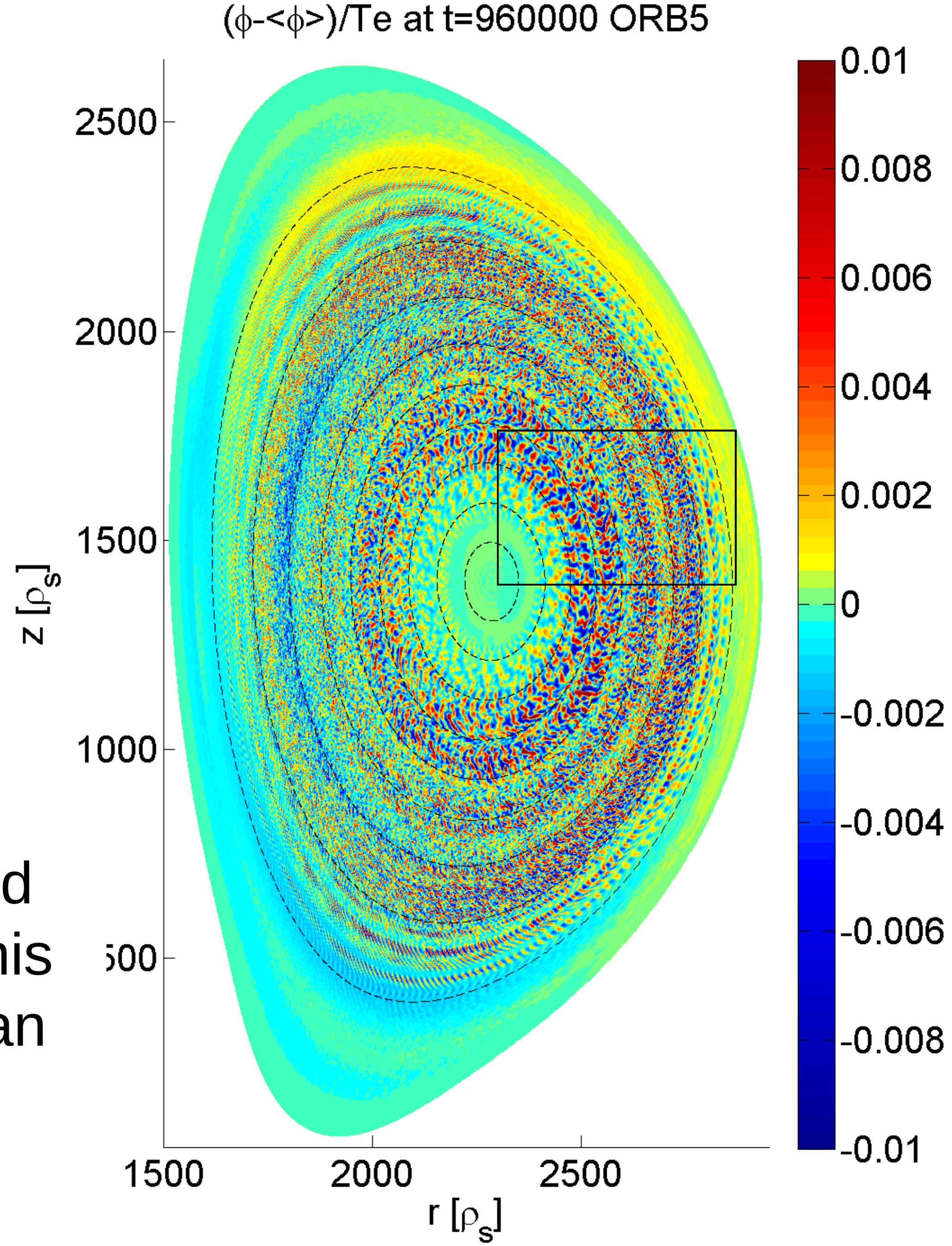
$(\phi, \vec{A})$ solution of Maxwell's equations,
with ρ, $\boldsymbol{j}$ obtained as moments of f_s
PDE, 3D

GK codes: GTC, GT3D, ORB5, GYGLES, ELMFIRE, PG3EQ, GTS…

Turbulence in magnetic fusion plasmas



ITG Turbulence in an ITER plasma



2D slice Snapshot

Contours of density perturbations

ITER size $(a/\rho_s)\sim1000$

10^9 grid (3D)(*)

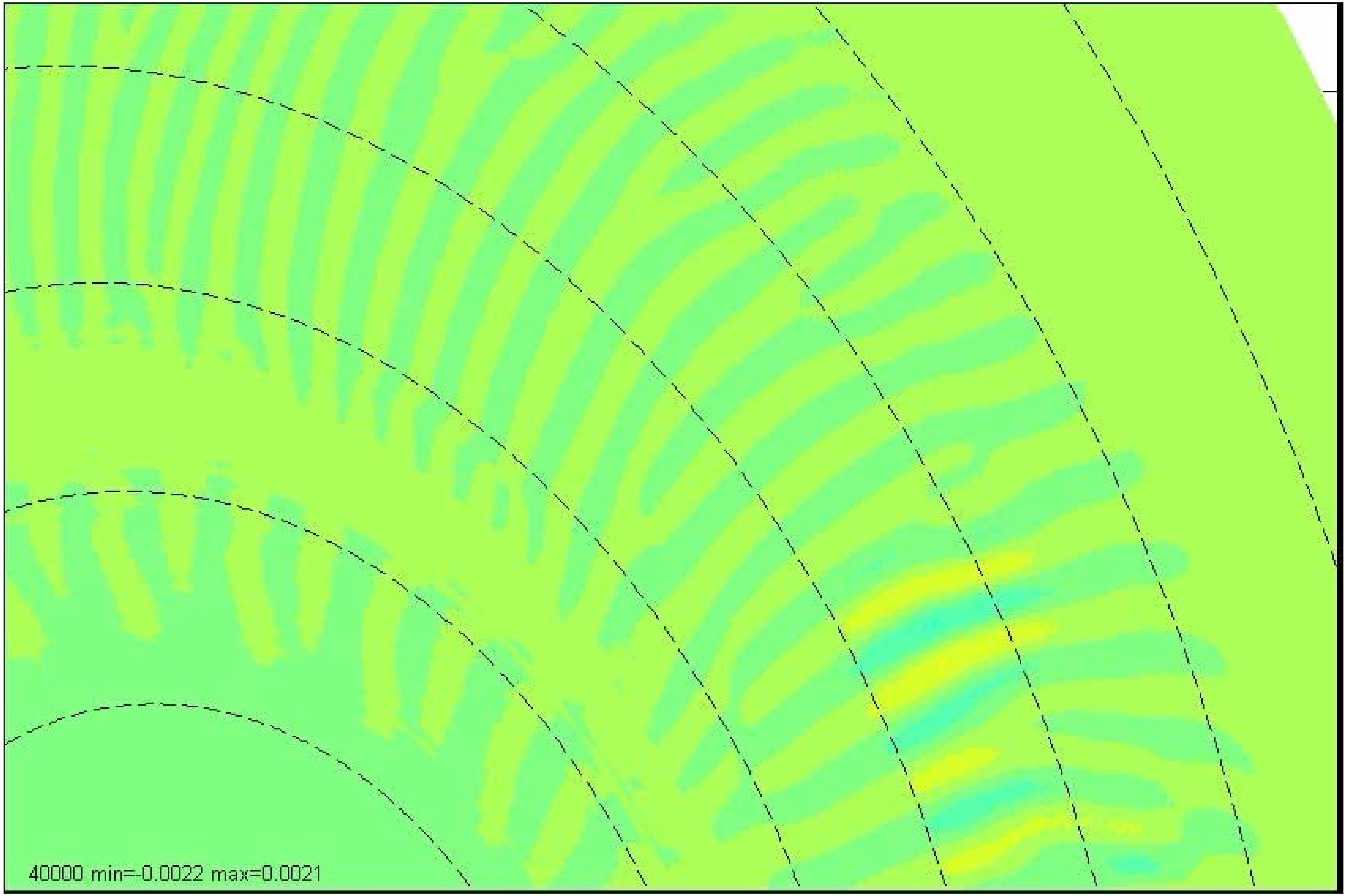
2G particles (5D)

HELIOS 3×10^5 core-h

(*) more grid points on this 2D slice than pixels

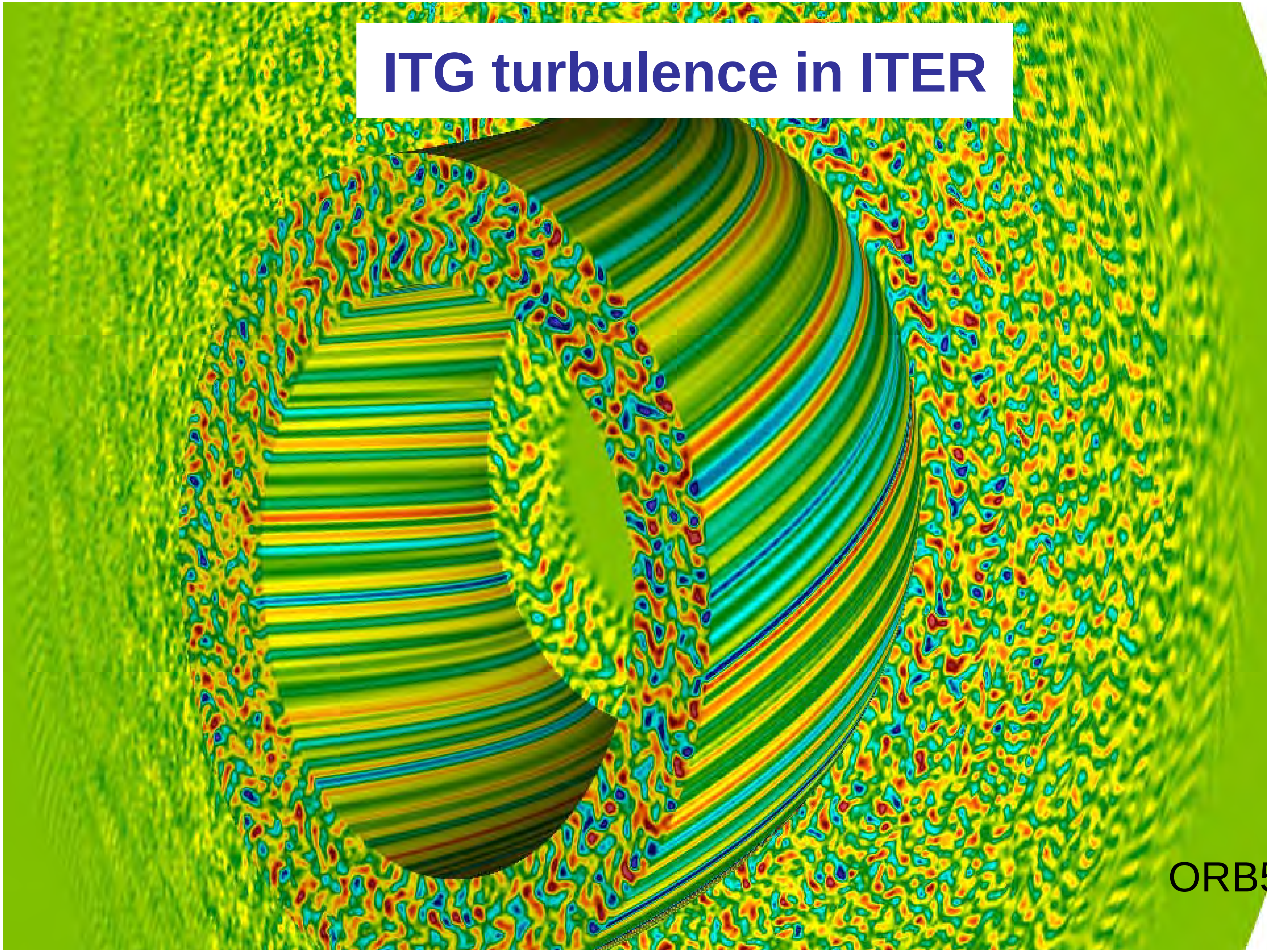
SWISS PLASMA CENTER

Turbulence and Zonal Flows



ORB5 ITER shape hybrid e-

ITG turbulence in ITER



PASC CoDesign “Particles & Fields” Project

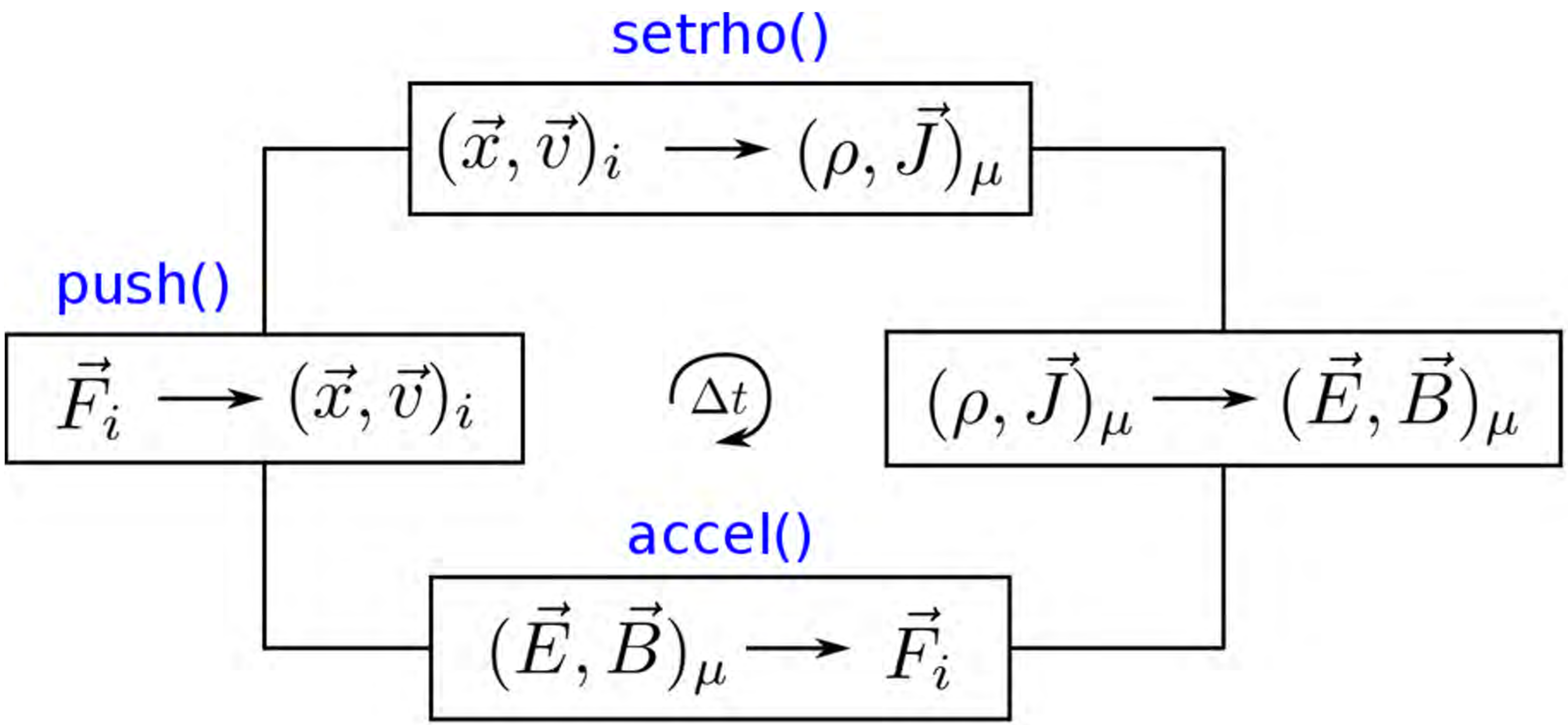
Legacy application codes are heavy / complex → cumbersome to work directly on these codes for adaptation to hybrid architectures

1. Extract fundamental algorithmic motives common to PIC codes → “**PIC-ENGINE**”
   - Test-bed for choices of fundamental algorithms and parallel programming models, on various architectures
   - MPI+OpenMP (CPU+MIC); MPI+OpenACC (CPU+GPU)

2. Develop PIC-engine features specifically relevant for gyrokinetic turbulence simulations
   - Strong background magnetic field
     - ***drift***-kinetic, ***gyro-***kinetic
       - complex geometry, anisotropy

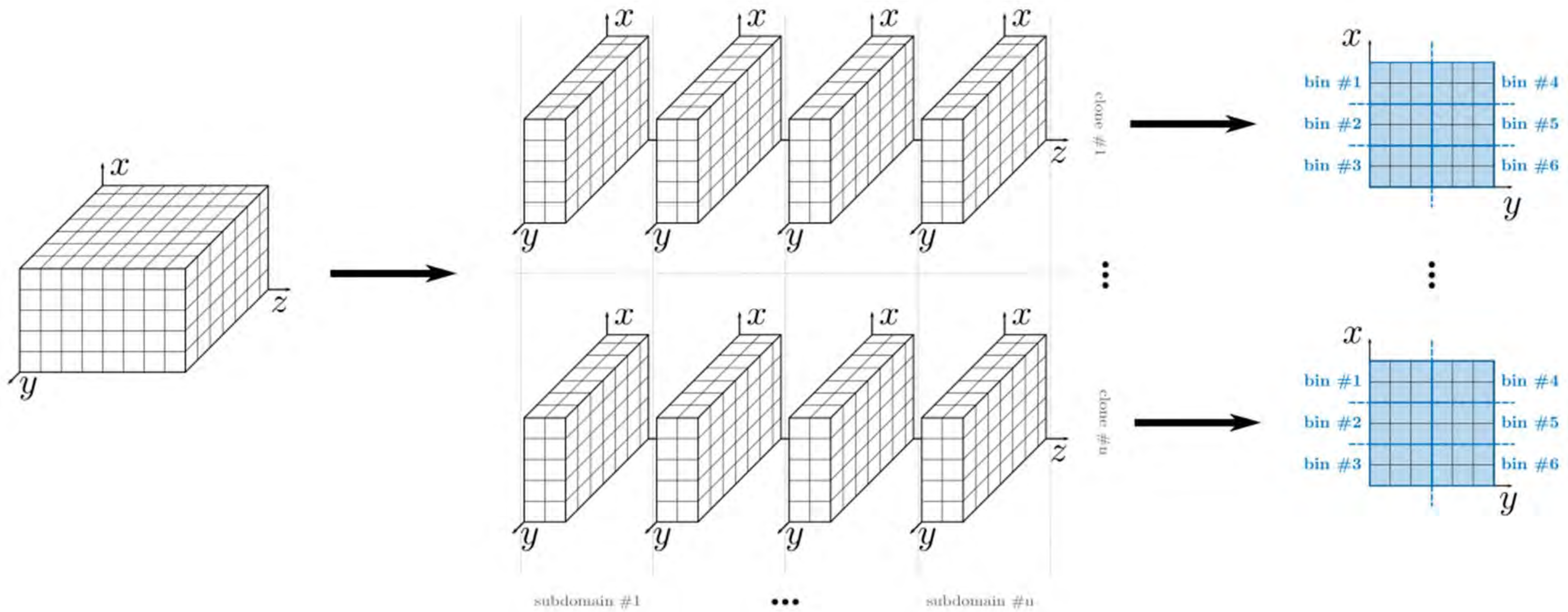
3. Adapt / Refactor legacy gyrokinetic code ORB5 to implement the algorithms and parallel programming models of the PIC-engine

Fundamental PIC engine



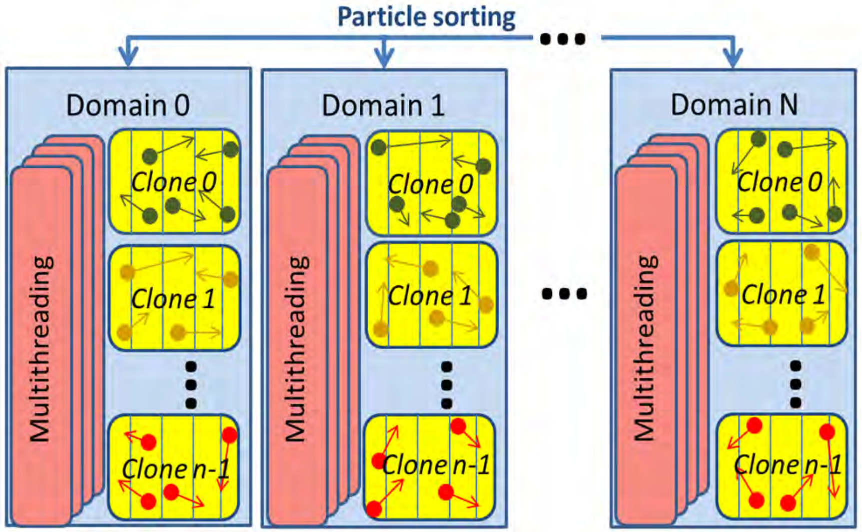
Particle data positions are “random” wrt grid positions
Critical are particle → grid operations in setrho() and
grid → particle operations in accel()

PIC-engine: parallelization



- Multiple-level parallelism:
  - Domain decomposition in the z-direction.
  - Domain cloning: grid data replication on each z-domain.
  - 2D bucket sorting of particle data within domains/clones.

PIC-engine: parallelization

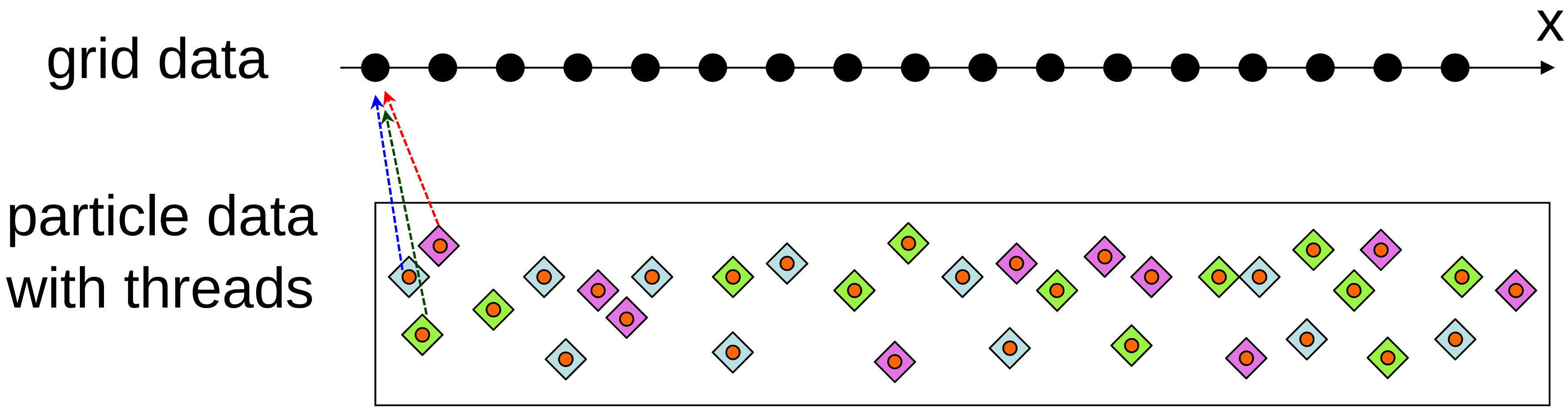


- Multiple-level parallelism:
  - Domain decomposition in the z-direction → MPI
  - Domain cloning: grid data replication on each z-domain → MPI
  - 2D bucket sorting of particle data within domains/clones → Multithreading OpenMP / OpenACC . On GPU: 3 levels of multithreading: thread blocks, warps, threads
- "Architecture-aware" parallelism: domain → compute node; clone → socket; massive multithreading → MIC, GPU

PIC-engine: summary

- 6D Vlasov-Poisson ; 3D real space grid, cartesian
- MPI+OpenACC/OpenMP hybrid parallel programming models
- Simplified: linear interpolations for particle-grid operations; electrostatic; frozen E field (no field solver); Euler explicit; equidistant normalized grid $\Delta x=\Delta y=\Delta z=1$; no background B field
- ***Several options*** for particle data structures: AOS or SOA; binned or contiguous
- Particle ***sorting*** in buckets (=partition of real space; 1 bucket contains 1 or more grid cells). Aim is to increase ***data locality***. ***Several algorithms,*** including a new one performing well for cases where < 30-50% of particles have to be moved to a different bucket. Allow for particle motion to any bucket (not only nearest neighbours) at every time step.
- ***Several options*** for charge assignment (setrho) multithreading
- Implemented and tested on GPUs, CPUs and MICs

Multithreading - Charge Assignment
(1) Collision-resolving : Threads on particles

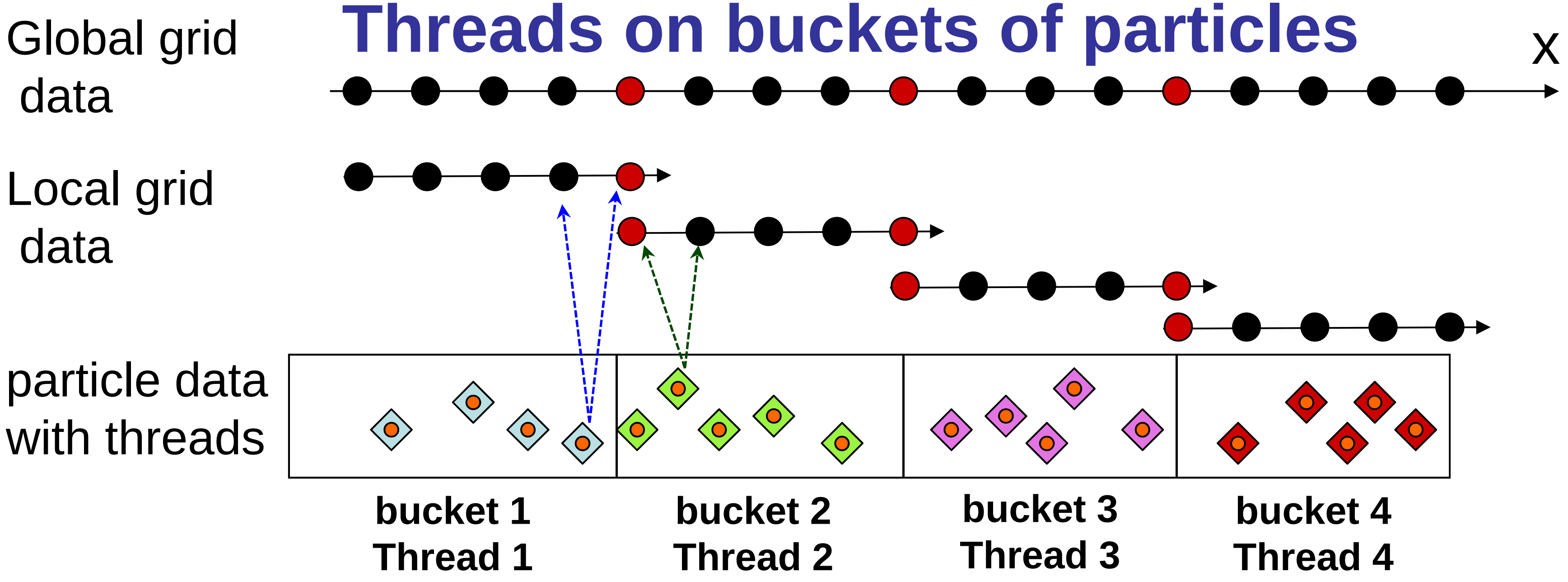


- Threads (represented by different colors) are associated with particle data
- Race condition: different threads update the same grid data ("collision")

→ Synchronization is needed
→ Use of **atomics**
→ Performance can be increased if particle data is **sorted** (→ data locality)

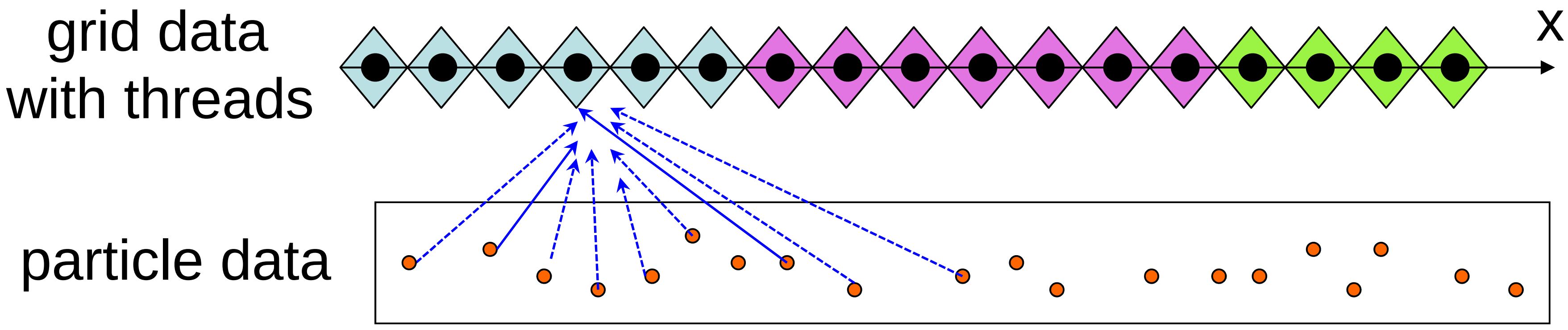
NB: illustration is 1D grid, but PIC-engine is 3D grid

(2) Collision-avoiding, data replication: Threads on buckets of particles



- Grid data is replicated from *Global* to *Local* data
- Particle data is sorted in buckets (according to their position on the grid)
- Threads are associated with buckets of particle data
- Each thread does the charge assignment on its *Local* Grid data
  ➔ NO race condition
- Ghost-cell data are added separately to the global grid data
- **NEED PARTICLE SORTING at every time step**

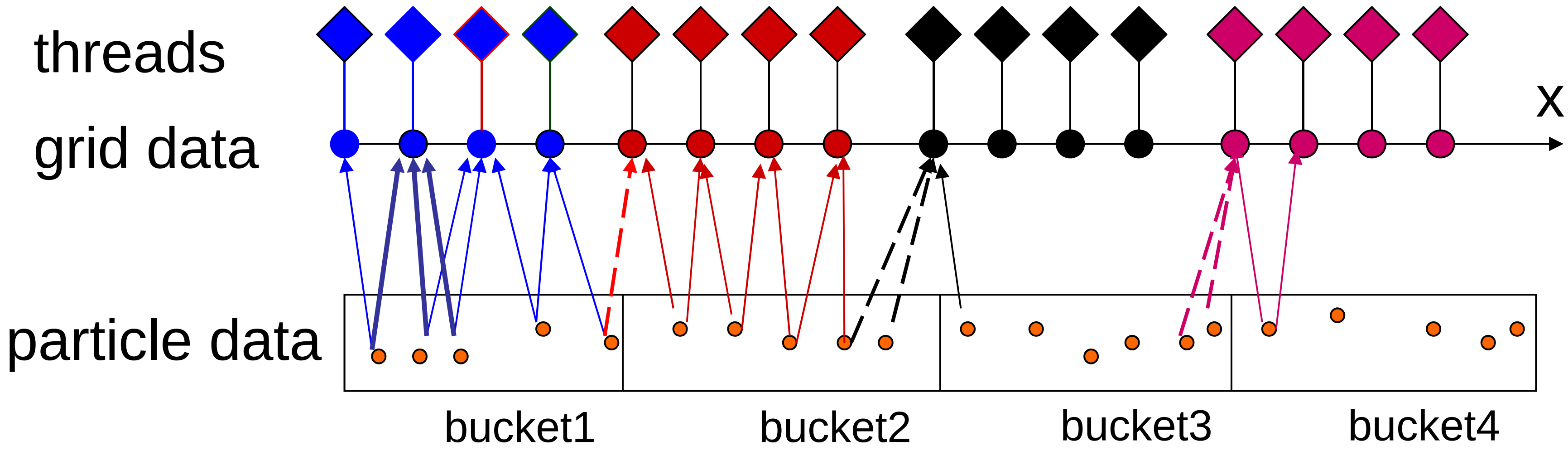
(3) Collision-avoiding : Threads on grid



- Threads (represented by different colours) are associated with grid data
- Different Threads may ***read*** the same particle data
  (Do not need to update particle data)
- NO Race condition
- NO Synchronization needed

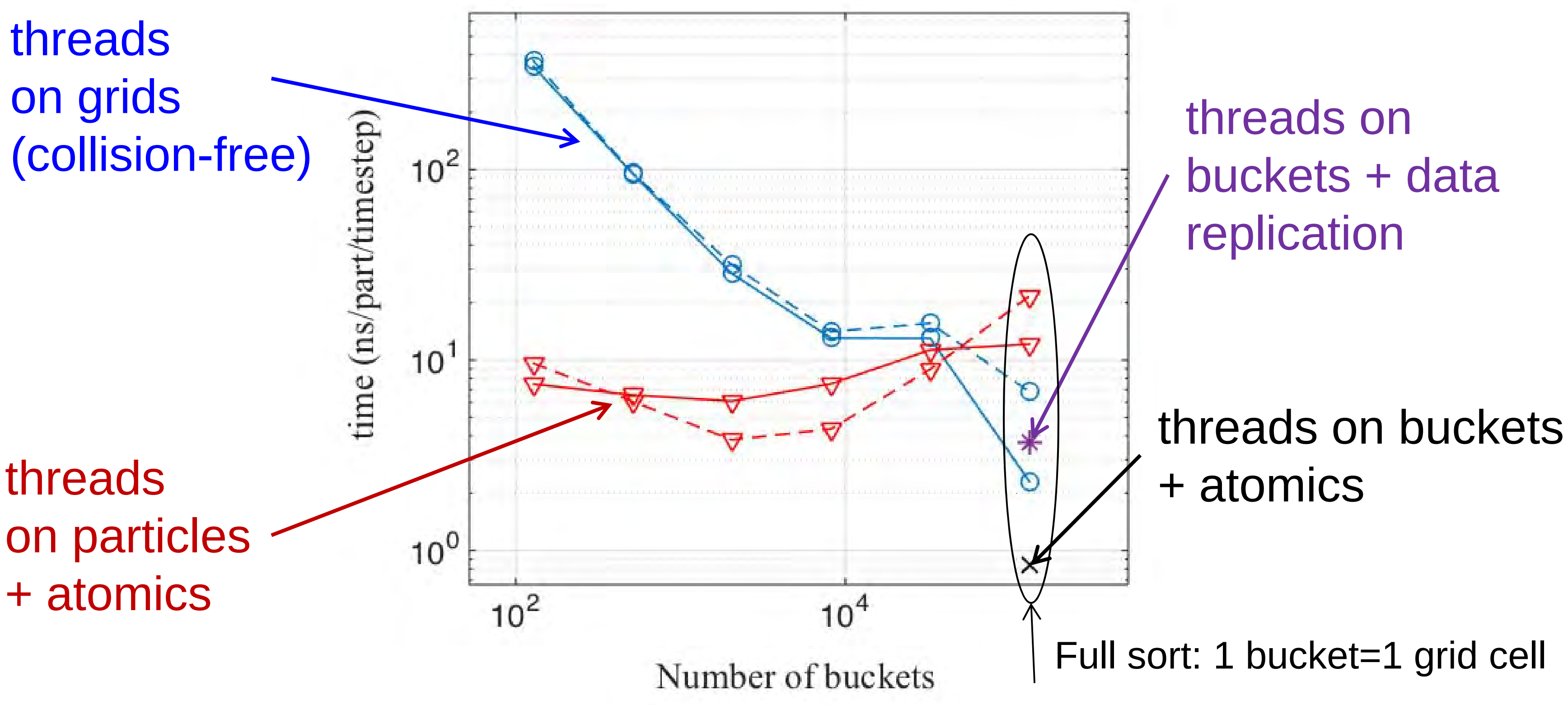
BUT Each Thread must loop over all the particles to read data: ***COSTLY***

(4) Collision-avoiding, Threads on grid + particles sorted in buckets



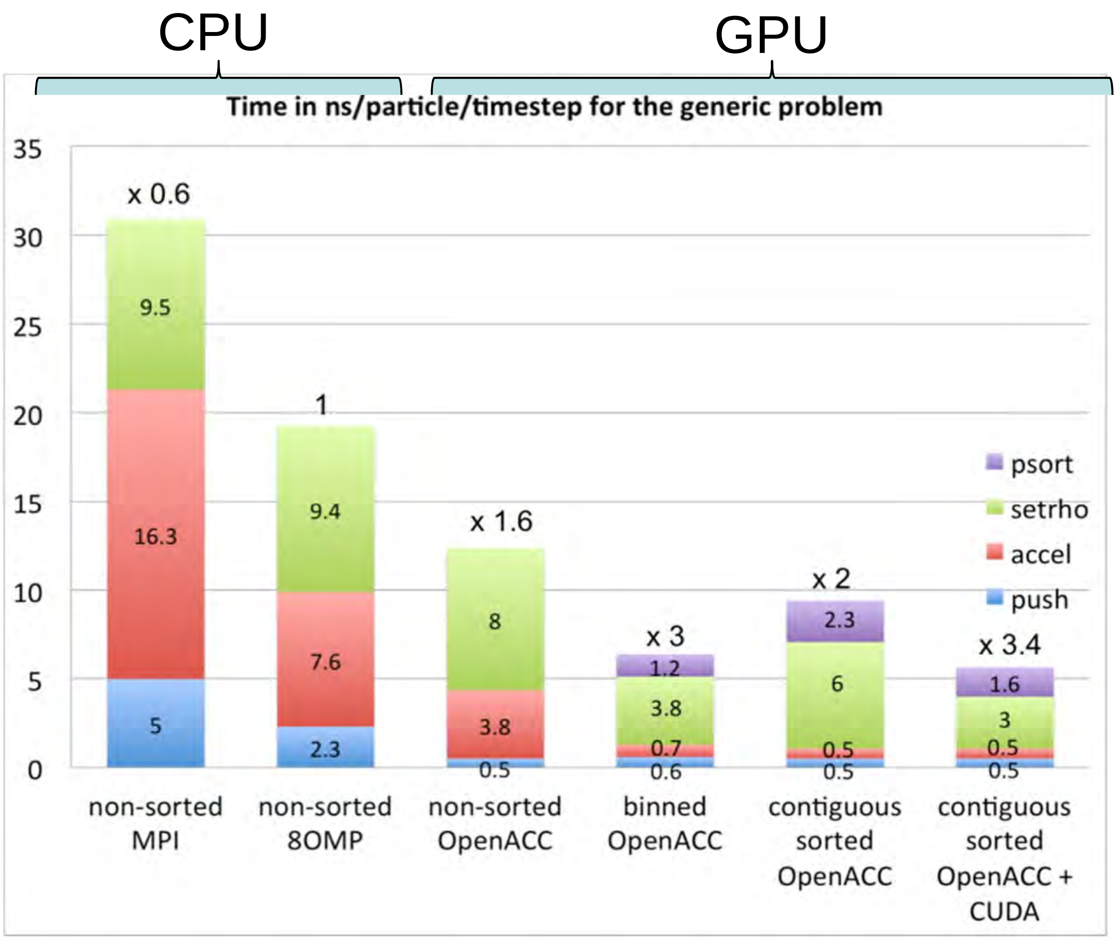
- Threads are associated with grid data
- Particle data is sorted in buckets (according to their position on the grid)
- Each grid (thread) must look into particles of its own bucket and nearest neighbour
- No data replication
- Collision-free: no synchronization required
- **NEED PARTICLE SORTING at every time step**

PIC engine on GPU – setrho



- Solid lines for contiguous data structure, dashed lines for binned data structure. Piz Daint 1-node, NVIDIA Tesla K20X.

PIC-engine on CPU vs GPU

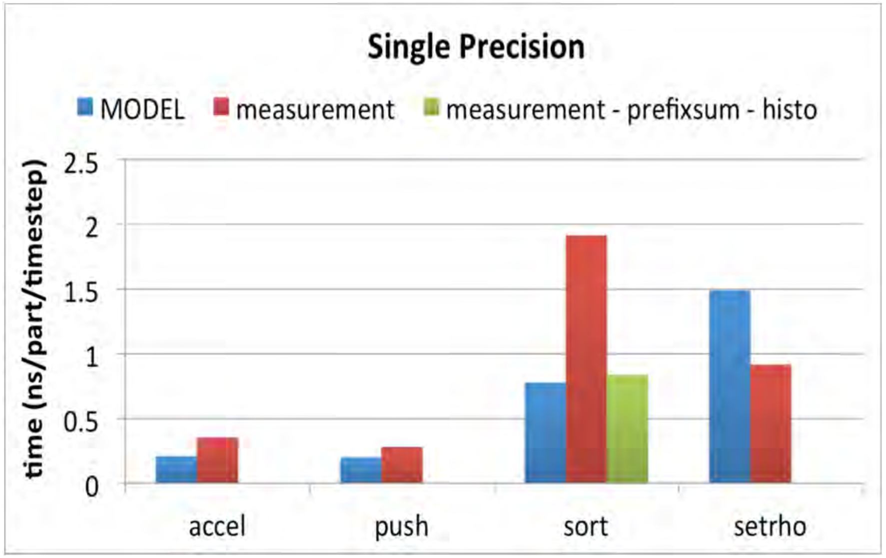


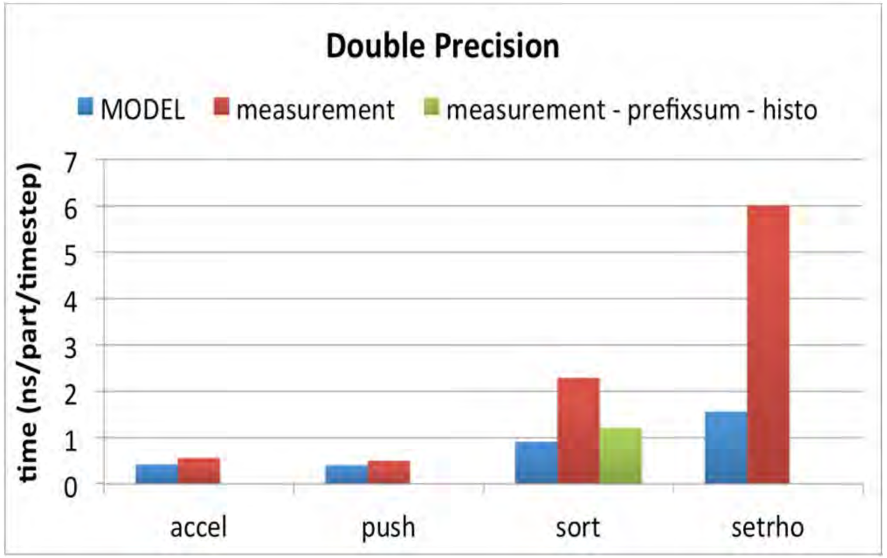
- Timings for various algorithmic options. 1 Piz Daint node (1x8 Intel SandyBridge, 1x Nvidia Tesla K20X). The best GPU version is up to **3.4 times** faster than the best CPU version

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE | SWISS PLASMA CENTER

PIC-engine: performance model

- Acknowledgement: Peter Messmer, Jakob Progsch (NVIDIA)
- Assumes memory-bound, several idealized simplifications

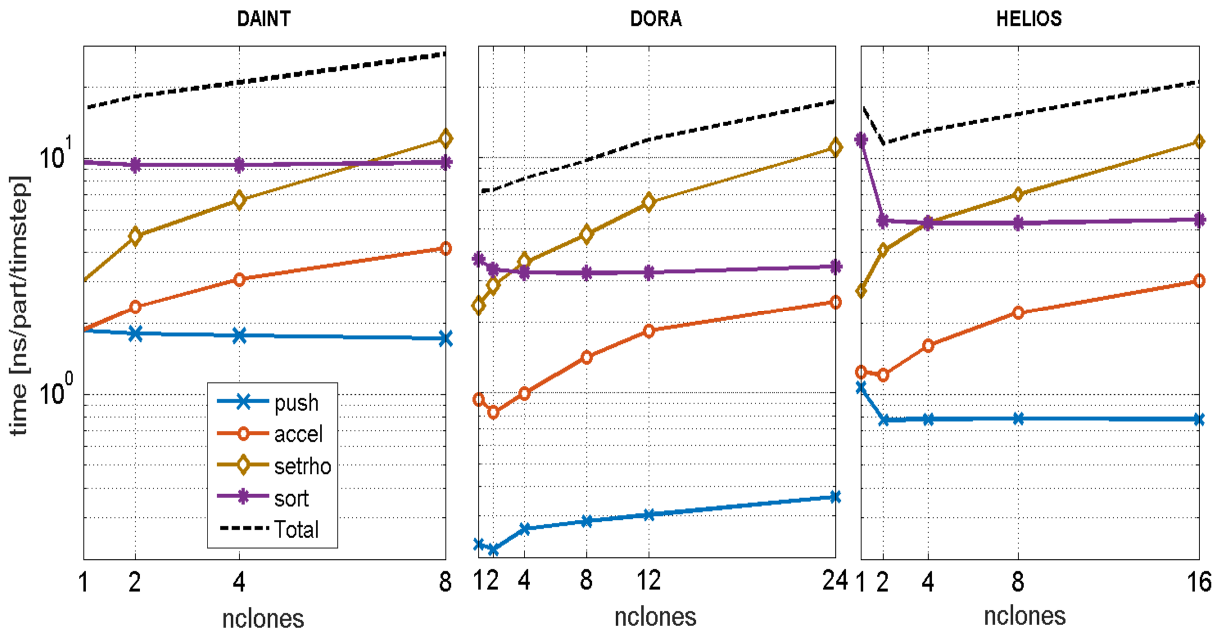


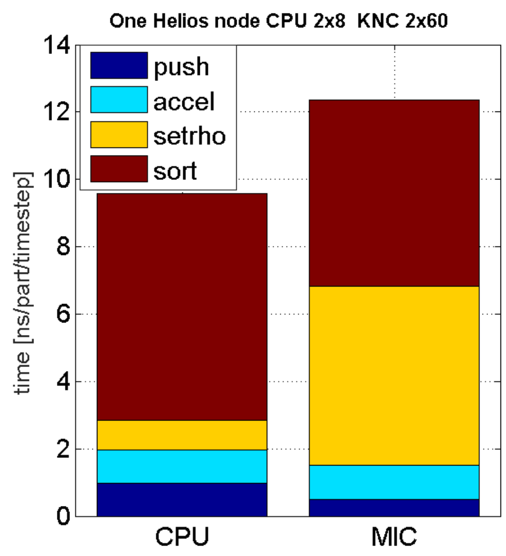


- Lack of native atomics on NVIDIA Tesla K20X for double precision is limiting performance of setrho
- Better result of setrho in single precision is due to the model not accounting for the impact of caching

PIC-engine on CPU and MIC

- Application of sorting improves timings of sethro, push and accel but the cost of sorting almost erases the gains on conventional CPU.
- Tested on various CPUs: Sandybridge 1x8 (Piz Daint), 2x8 (Helios), Haswell 2x12 (Piz Dora). Optimization of Nclones vs Nthreads, keeping Nclones x Nthreads = const → 1 clone per socket is optimal
- On Helios: 2 x Xeon Phi (KNC), similar timings than on CPUs for large number of particles/cell. Optimum: 20 clones, 24 threads

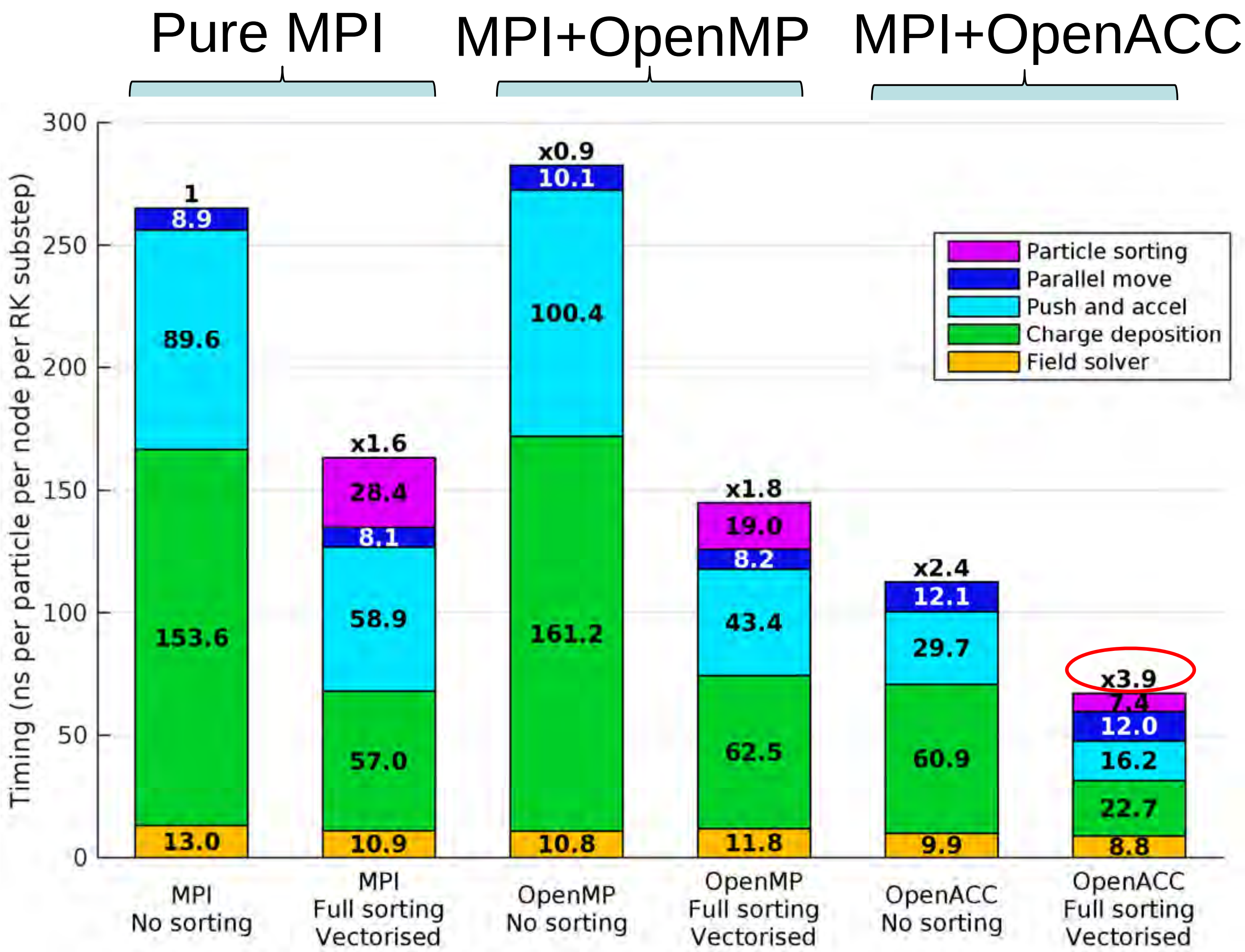




Towards Gyrokinetic Application

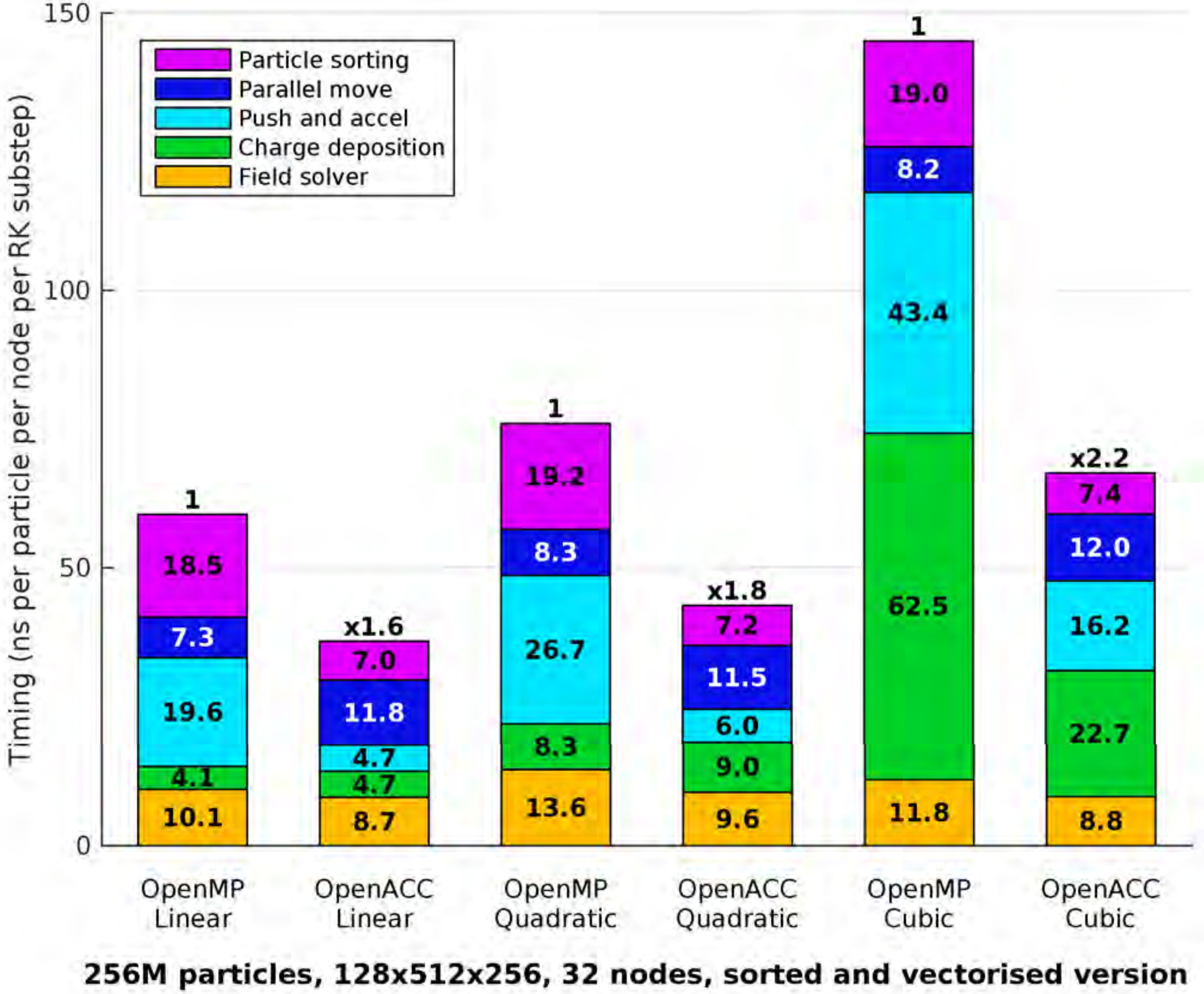
- A **Drift-Kinetic-Engine** was developed from the PIC-engine.
- MPI+OpenACC/OpenMP – single source files
- Domain decomposition (z), domain cloning and multithreading
- Turbulence in a sheared magnetized plasma slab
- W.r.t. PIC-engine, the DK-engine includes:
  - Drift Kinetic Equations in physical units
  - Strong anisotropy (background magnetic field)
  - Finite element 3D field solver (quasineutrality)
  - B-splines up to 4^{th} order
  - Control variates (delta-f) scheme
  - DFT in y and z (requiring parallel data transpose) and field-aligned Fourier filtering
  - 3D bucket sorting within z-domains & clones

DK-engine: results



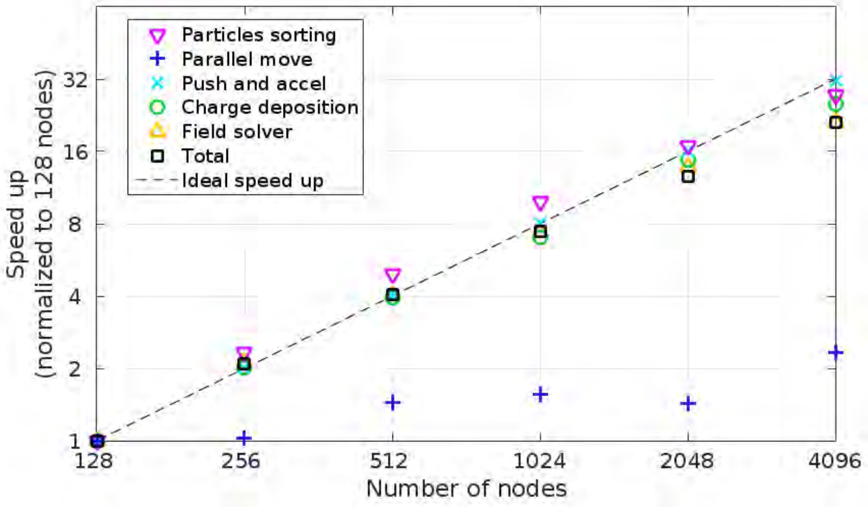
256M particles, 128x512x256, cubic splines, vmax.dt=7dz, 32 nodes, SOA

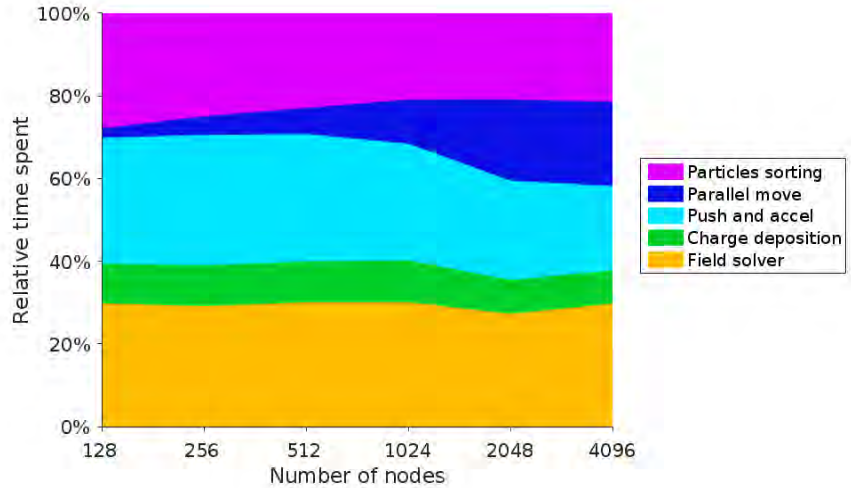
DK-engine: results



- GPU outperforms best CPU version – more so for higher order splines

DK-engine: strong scaling

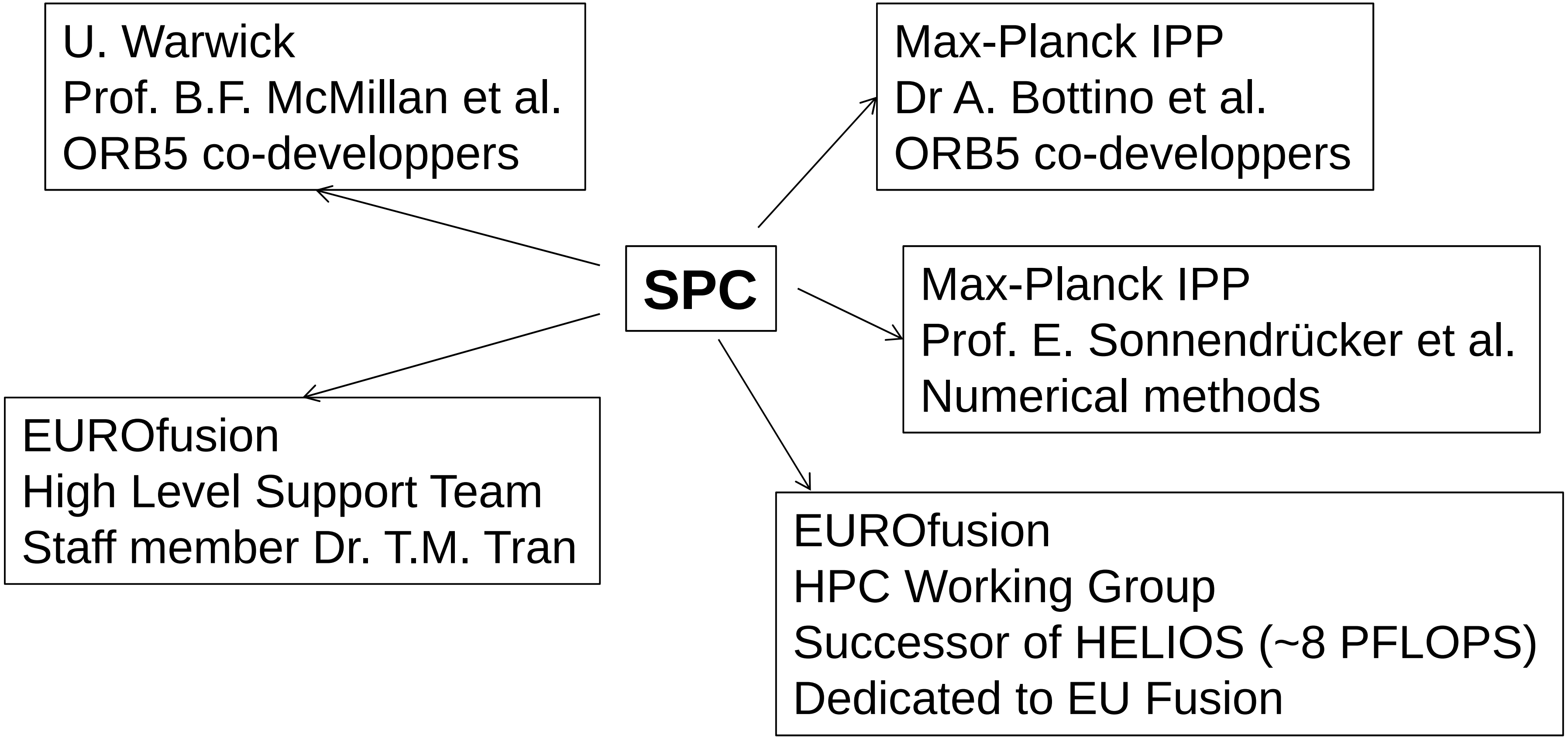




- 128 × 512 × 256 cells, 4.096x10^9 particles, (244 particles/cell), 32 clones, 8 threads, **4 to128 z-domains**, **128 to 4096 nodes**. Excellent parallel scalability up to ~full PIZ DAINT.
- Parallel move (sort in z-direction across nodes, dark blue) may become a problem for very large grids and number of nodes. Challenging case here: $v_{max}\Delta t=7\Delta z$

Community involvement

- SPC activities are embedded in the EUROfusion Consortium



Conclusions

- We have progressed on the path to make use of many-, multi-core and GPU-equipped supercomputers for applications based on the PIC scheme
  - New software: PIC-engine, DK-engine
  - Demonstrated performance and scalability on hybrid systems
  - Demonstrated capability, performance and potential of hybrid programming models MPI+OpenMP and MPI+OpenACC
  - Findings about the performance of the PIC method and its relation to particular hardware features (e.g. atomics on the GPU) → useful feedback to Cray/NVIDIA
- Future steps: developer community will be directly involved in the refactoring project of ORB5 (effort led by T.M. Tran)
  - Use of directive-based programming models → code refactoring does not require a complete rewriting
- Charge assignment (setrho) from PIC-engine is being implemented in the astrophysics code RAMSES