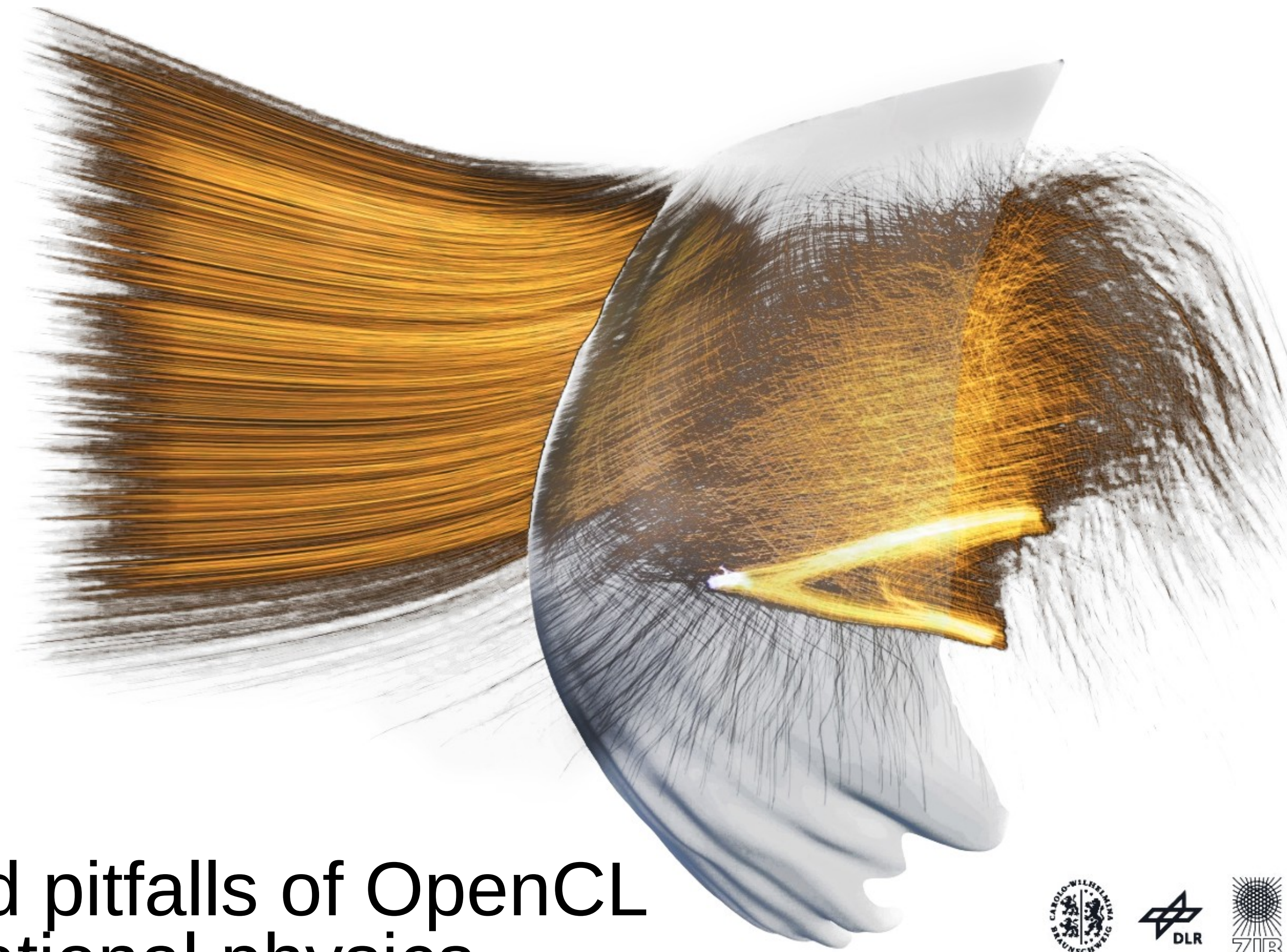
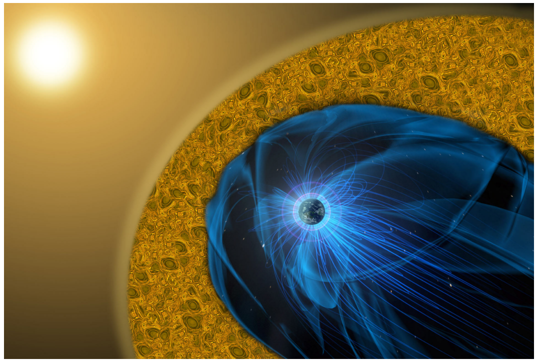


Advantages and pitfalls of OpenCL in computational physics

K. Ostaszewski, P. Heinisch, H. Ranocha

Computational Physics

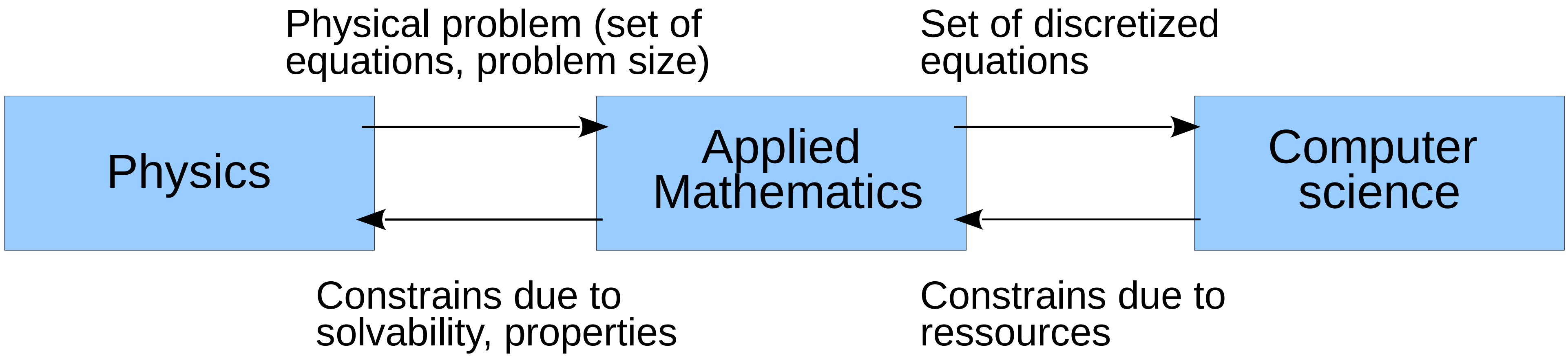
- Very few physical problems can be solved analytically due to
  - Complexity,
  - Lack of algebraic solvability.
- Numerical approximations are required.



- Fields of application:
  - Plasmaphysics (Fusion, Astrophysics, industrial plasmas),
  - Weather prediction,
  - Solid state physics, etc.

Computational Physics

- Overlap of physics, applied mathematics and computer science.



- Different fields constrain eather other.

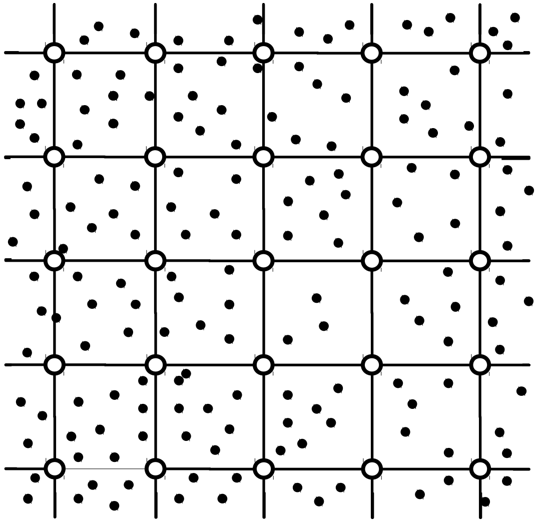
Motivation

- Legacy Plasmacode (A.I.K.E.F) used for different NASA/ESA missions.

- Parallelized with MPI (CPU based).
- Limits in scalability and ressources reached.

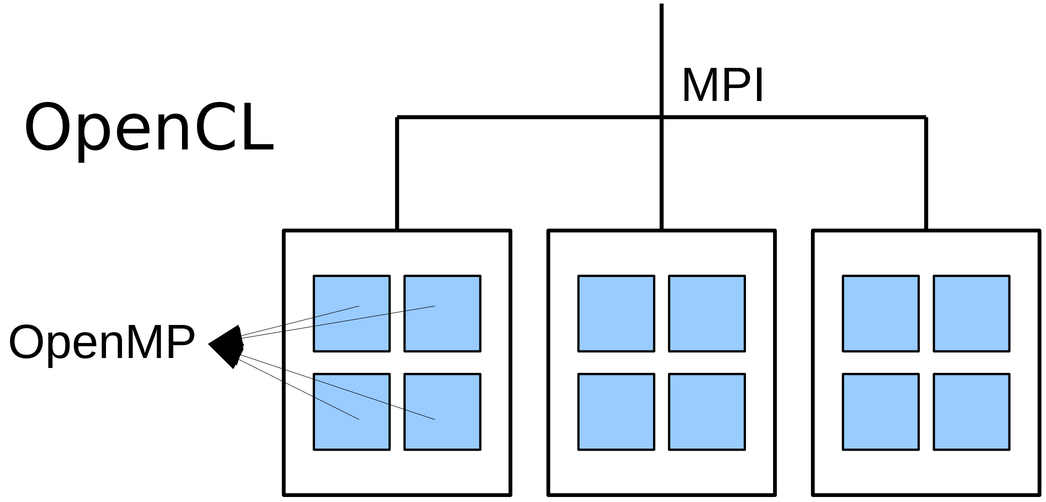
Plasmaphysics

- Different models to describe a plasma: Fluid-Model, Hybrid-Model, Particle-in-Cell (PiC).
- Need to describe particles and electromagnetic fields.
- Hybrid-Model includes the following mathematical problems:
  - Systems of linear equations,
  - Ordinary/partial differential equations,
  - N-Body interactions.



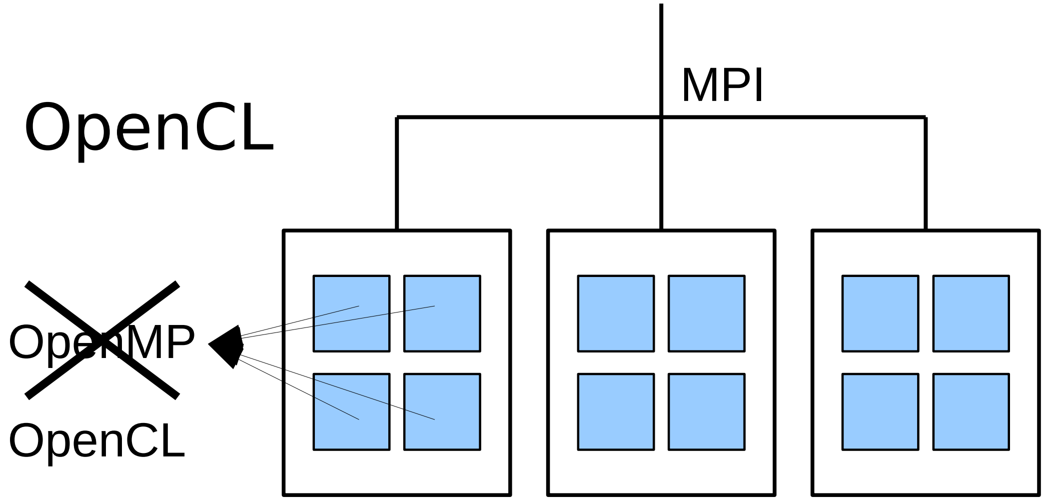
Simulation Approach

- Usual approach is to combine a global inter-node based MPI parallelization with a local OpenMP parallelization.
- Developed for deployment on clusters (HLRN,...)
- Substitute OpenMP with OpenCL



Simulation Approach

- Usual approach is to combine a global inter-node based MPI parallelization with a local OpenMP parallelization.
- Developed for deployment on clusters (HLRN,...)
- Substitute OpenMP with OpenCL



Advantages and pitfalls

Advantages of OpenCL

- Deployment on heterogenous systems (portability),
- Runtime advantage by using GPUs,
- Easy testing/changing of numerical submodules (Python, Oclgrind, MatCL),

Advantages and pitfalls

Advantages of OpenCL

- Deployment on heterogenous systems (portability),
- Ru
- Ea (M

```
convec.x = d_field_b_old[idx].x*(d_field_u[increment_x(idx,1)].x - d_field_u[increment_x(idx,-1)].x)/((2.0f*(REAL)DX))
           +d_field_b_old[idx].y*(d_field_u[increment_y(idx,1)].x - d_field_u[increment_y(idx,-1)].x)/((2.0f*(REAL)DY))
           -0.5*d_field_b_old[idx].x*((d_field_u[increment_x(idx,1)].x - d_field_u[increment_x(idx,-1)].x)/(2.0f*(REAL)DX)
           +(d_field_u[increment_y(idx,1)].y - d_field_u[increment_y(idx,-1)].y)/((2.0f*(REAL)DY)))
           -0.5*(d_field_u[idx].x*(d_field_b_old[increment_x(idx,1)].x - d_field_b_old[increment_x(idx,-1)].x)/(2.0f*(REAL)DX)
           +d_field_u[idx].y*(d_field_b_old[increment_y(idx,1)].x - d_field_b_old[increment_y(idx,-1)].x)/(2.0f*(REAL)DY))
           -0.5*((d_field_u[increment_x(idx,1)].x*d_field_b_old[increment_x(idx,1)].x
           -d_field_u[increment_x(idx,-1)].x*d_field_b_old[increment_x(idx,-1)].x)/(2.0f*(REAL)DX)
           +(d_field_u[increment_y(idx,1)].y*d_field_b_old[increment_y(idx,1)].x
           -d_field_u[increment_y(idx,-1)].y*d_field_b_old[increment_y(idx,-1)].x)/(2.0f*(REAL)DY));

convec.y = d_field_b_old[idx].x*(d_field_u[increment_x(idx,1)].y - d_field_u[increment_x(idx,-1)].y)/((2.0f*(REAL)DX))
          +d_field_b_old[idx].y*(d_field_u[increment_y(idx,1)].y - d_field_u[increment_y(idx,-1)].y)/((2.0f*(REAL)DY))
          -0.5*d_field_b_old[idx].y*((d_field_u[increment_x(idx,1)].x - d_field_u[increment_x(idx,-1)].x)/(2.0f*(REAL)DX)
          +(d_field_u[increment_y(idx,1)].y - d_field_u[increment_y(idx,-1)].y)/((2.0f*(REAL)DY)))
          -0.5*(d_field_u[idx].x*(d_field_b_old[increment_x(idx,1)].y - d_field_b_old[increment_x(idx,-1)].y)/(2.0f*(REAL)DX)
          +d_field_u[idx].y*(d_field_b_old[increment_y(idx,1)].y - d_field_b_old[increment_y(idx,-1)].y)/(2.0f*(REAL)DY))
          -0.5*((d_field_u[increment_x(idx,1)].x*d_field_b_old[increment_x(idx,1)].y
          -d_field_u[increment_x(idx,-1)].x*d_field_b_old[increment_x(idx,-1)].y)/(2.0f*(REAL)DX)
          +(d_field_u[increment_y(idx,1)].y*d_field_b_old[increment_y(idx,1)].y
          -d_field_u[increment_y(idx,-1)].y*d_field_b_old[increment_y(idx,-1)].y)/(2.0f*(REAL)DY));
```

Pitfalls of OpenCL

- Copy overhead serious bottleneck,
- Lack of debugging capability (e.g. no buffer overflow check),
- Documentation/examples lacking.

Partial Differential Equations

- Most simulation codes involve solving of differential equations on a discretized grid:

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \qquad \nabla \times \vec{B} = \mu_0 \vec{j} + \frac{1}{c^2}\frac{\partial \vec{E}}{\partial t}$$

- Discretization schema influences physical properties of solution.
  - Trade of between accuracy and computational cost!

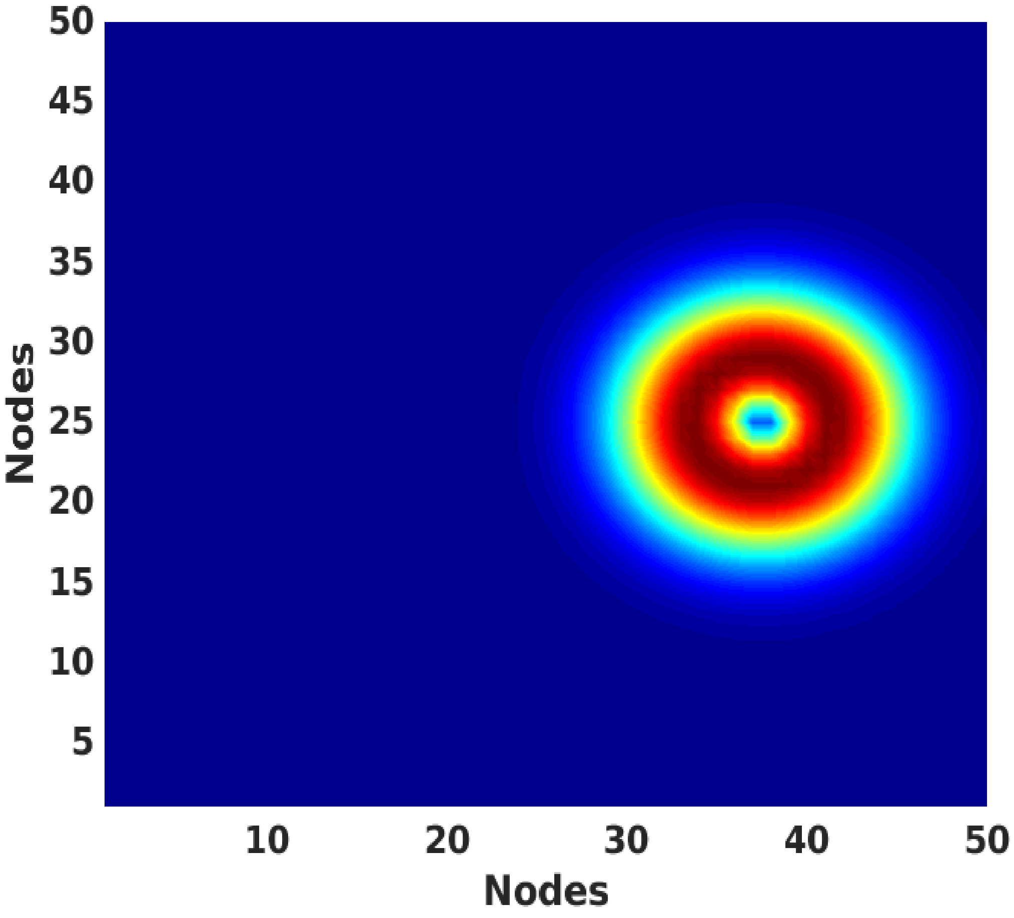
Example: Partial Differential Equations

- Example: Frozen-in-Theorem

$$\frac{\partial \vec{B}}{\partial t} = \nabla \times (\vec{u} \times \vec{B})$$

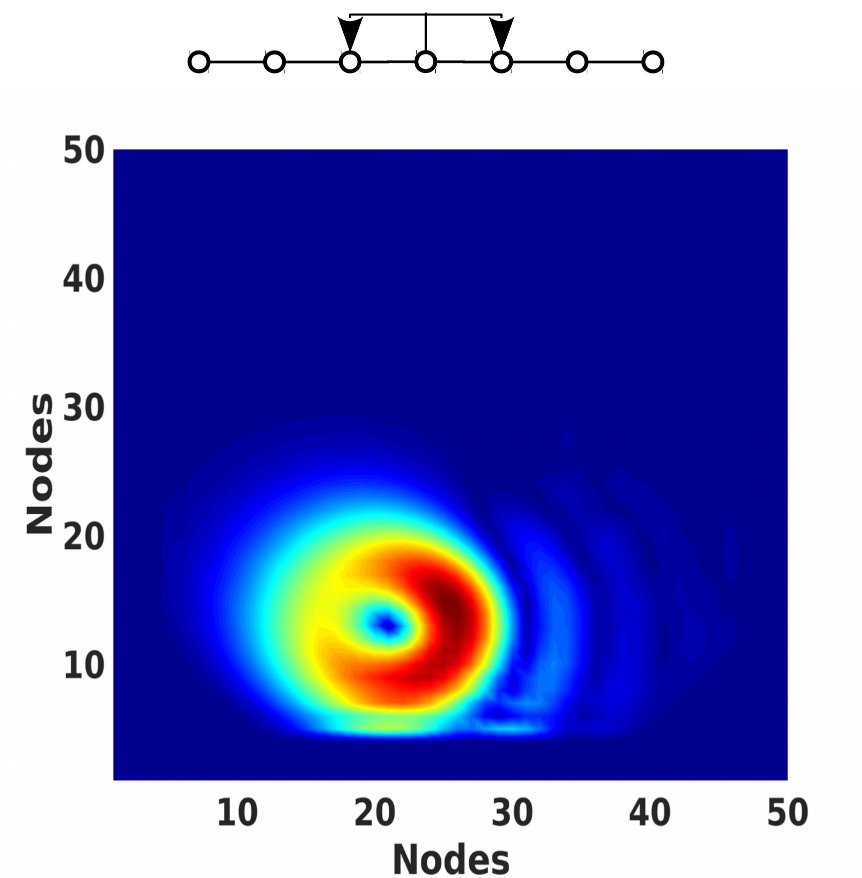
B: Magnetic field
u: Fluid velocity

- Solution will rotate around middle of the box.

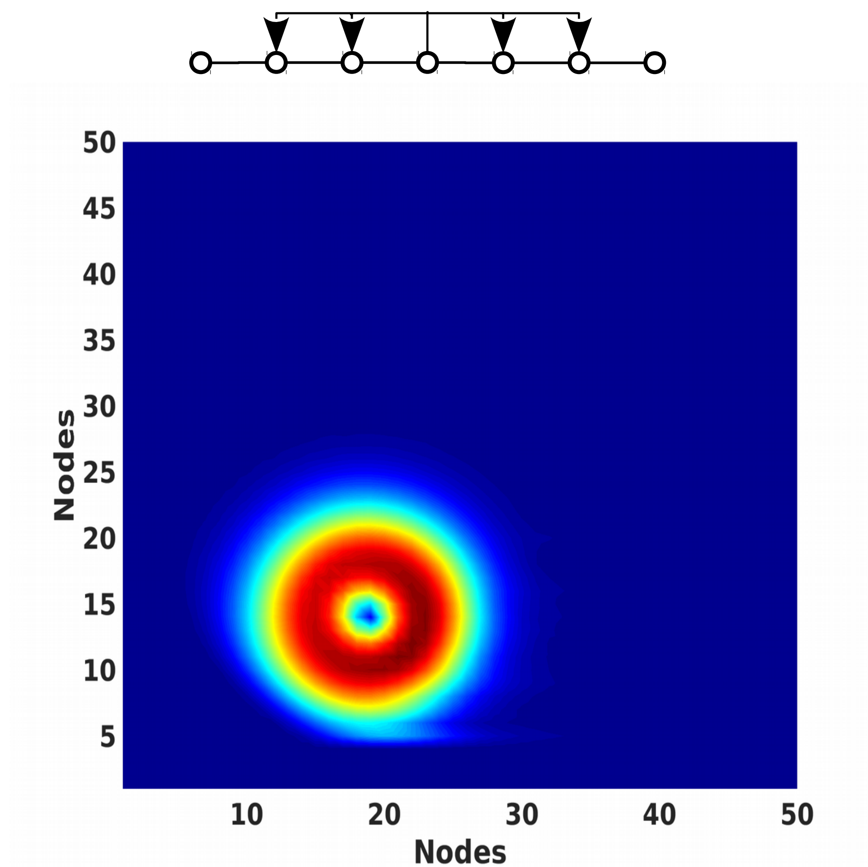


Example: Partial Differential Equations

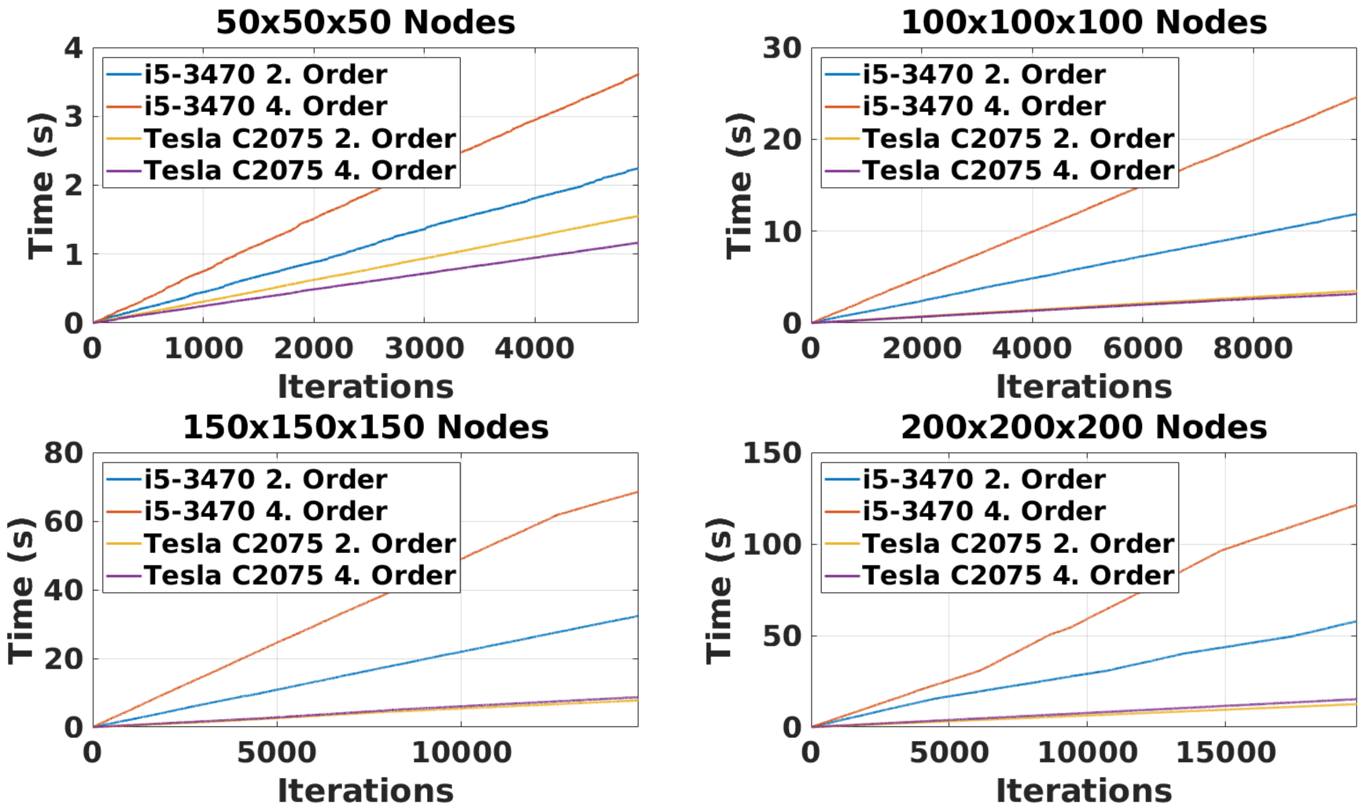
- Solver 2nd Order



- Solver 4th Order



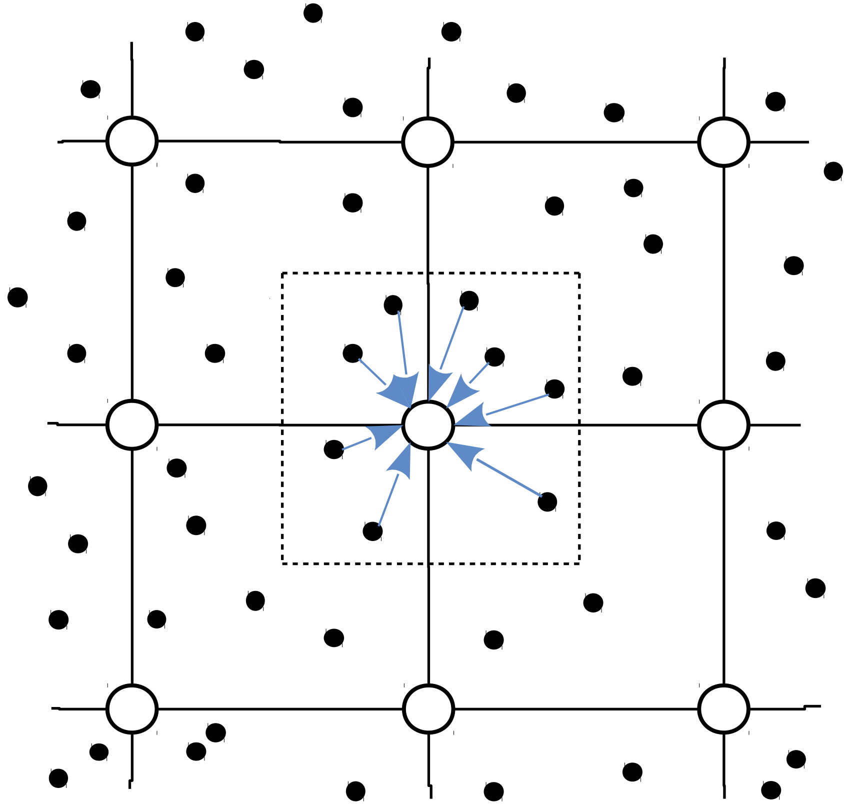
Example: Partial Differential Equations



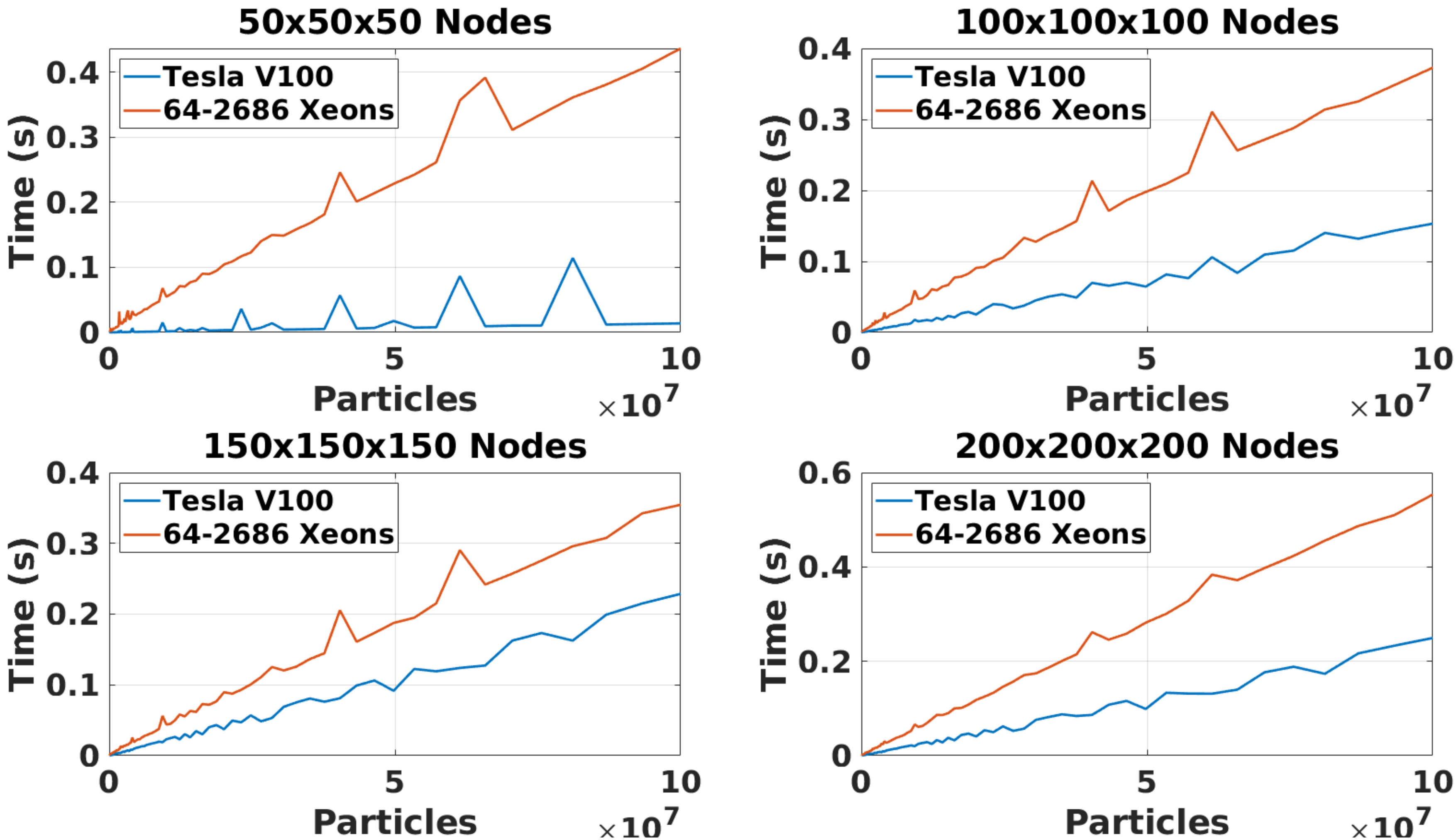
Number of Nodes	Performance penalty GPU | CPU
50x50x50	0.78 | 1.63
100x100x100	0.95 | 2.04
150x150x150	1.11 | 2.23
200x200x200	1.21 | 1.89

Particle to Grid Reduction

- Sum up all particle on next node.
- Needed to describe the interaction between electromagnetic fields and particles.
- Naive approach: add up all particles using atomics.



Particle to Grid Reduction



Particle to Grid Reduction

- Problem also encountered in gravitational N-Body simulations.

- Solution available, but they depend on GPU architecture ⇒ loss of portability.

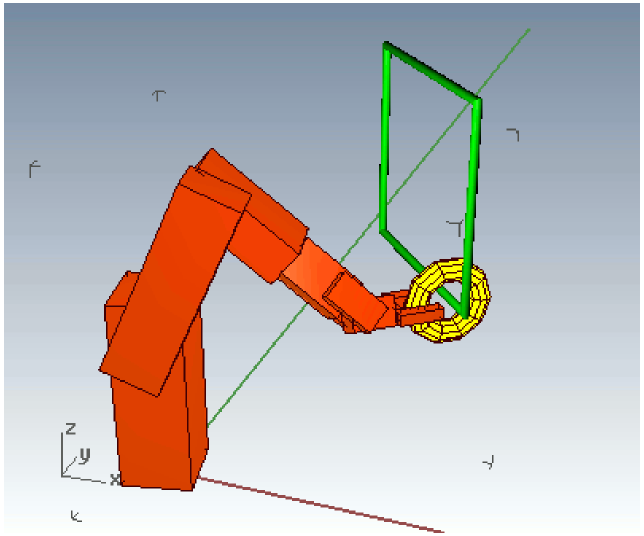
- FPGAs perfect for this kind of task ⇒ limited availability.

Conclusion

- OpenCL offers many advantages in computational science.
  - deployment on heterogenous systems,
  - seperation between "physics" and (architecture dependent) host code.
- But it is difficult getting started:
  - lack of documentation/examples,
  - limited debug possibilites.

Systems of linear Equations

- Systems of linear equations are used in:
  - solving of implicit differential equations,
  - inverse kinematics,
  - computer vision (OpenCV).



- Common methods:
  - Gauß-Seidel-Methods (SOR,...),
  - Conjugate Gradient Method (CG),
  - Cholesky-Method.

System of linear Equations