Towards Kinetic Whole Device Modelling of Low-Temperature Plasma Devices

Algorithms and Computer Science

Andrew (Tasman) Powis

Talk Overview

Particle-in-Cell Method

(Another) introduction to PIC and a discussion on why we use it

Leveraging Modern Supercomputers

Some important considerations for programming on CPU+GPU computers

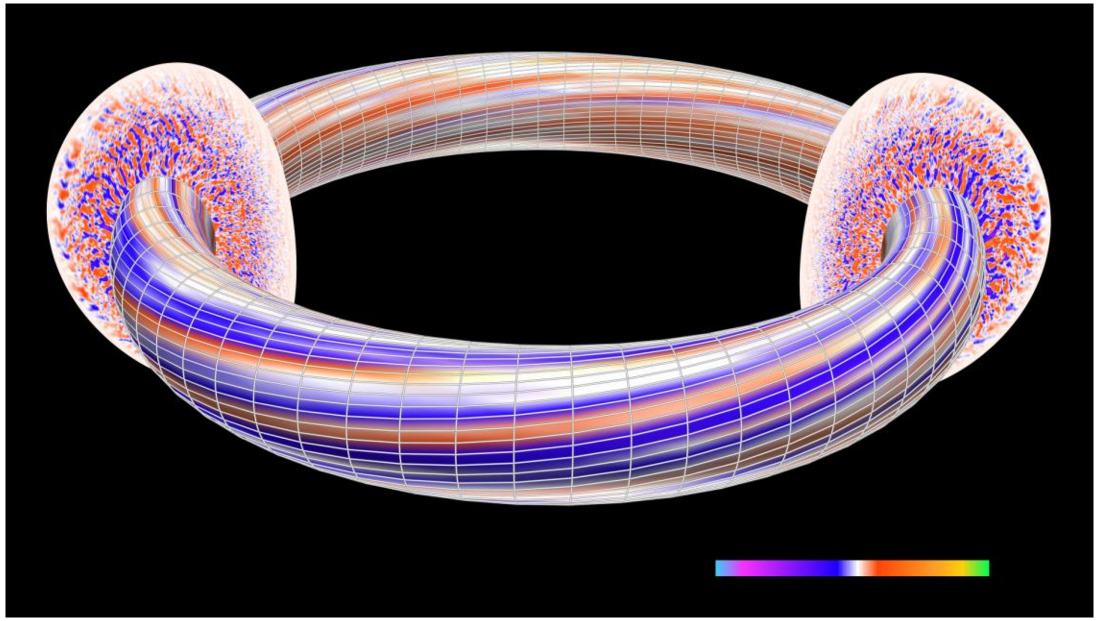
GPUs are Not Enough

Improving hardware performance is not sufficient to achieve kinetic whole-device modeling, we also need to consider new or alternative algorithms

Why Simulate?

The goal of numerical simulations is broadly:

1. To improve our understanding of fundamental physics by resolving features which are not measurable or by reaching conditions which are not achievable in experiments

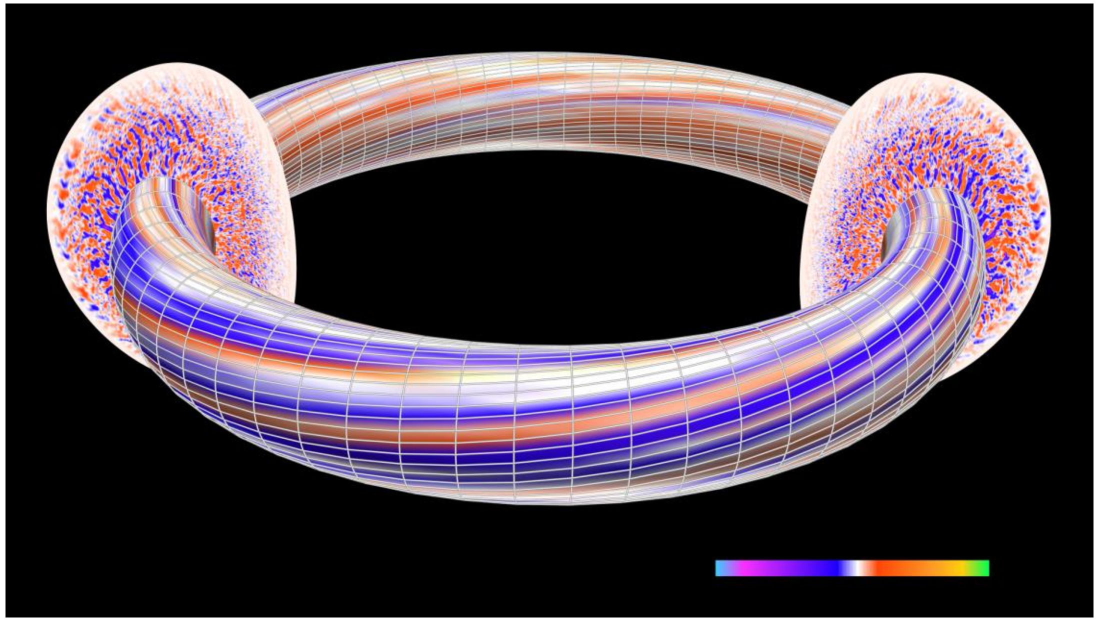


3D gyro-kinetic simulations of ion-temperature gradient instabilities within a Tokamak. Run using the GTS code [Ma *et al.*]

Why Simulate?

The goal of numerical simulations is broadly:

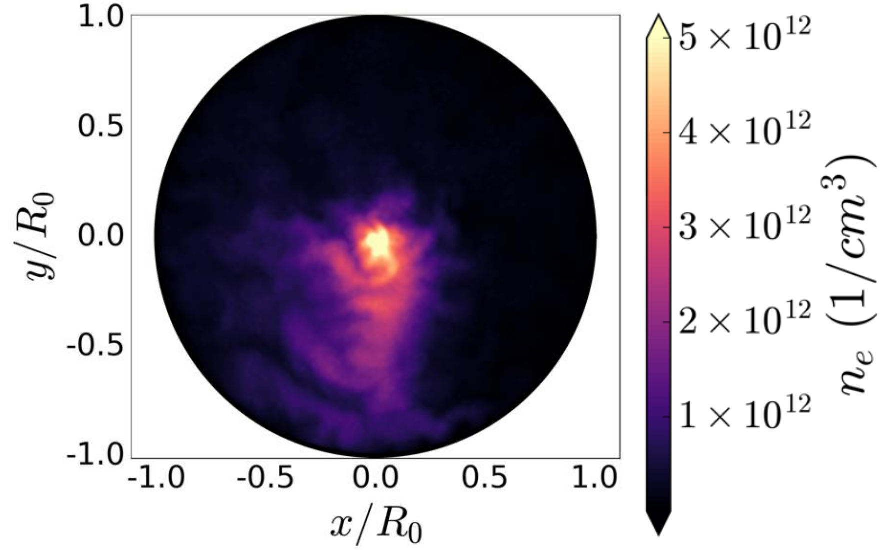
1. To improve our understanding of fundamental physics by resolving features which are not measurable or by reaching conditions which are not achievable in experiments
2. To reduce the cost of engineering design or prototyping through whole-device modelling (WDM)


3D gyro-kinetic simulations of ion-temperature gradient instabilities within a Tokamak. Run using the GTS code [Ma *et al.*]

Why Kinetic Simulations?

The global properties of *many* low-temperature plasma devices (particularly at low pressure) are dominated by kinetic processes:

- Interaction between the plasma and the wall through the sheath
- Velocity space instabilities (i.e. bump on tail)
- Collisional processes (i.e. ionization)
- Non-classical or turbulence induced transport



Electron density plots demonstrating formation of a rotating spoke within 2D Penning discharge simulations [Powis *et al.*]

Particle-in-Cell Method

Particle-in-Cell

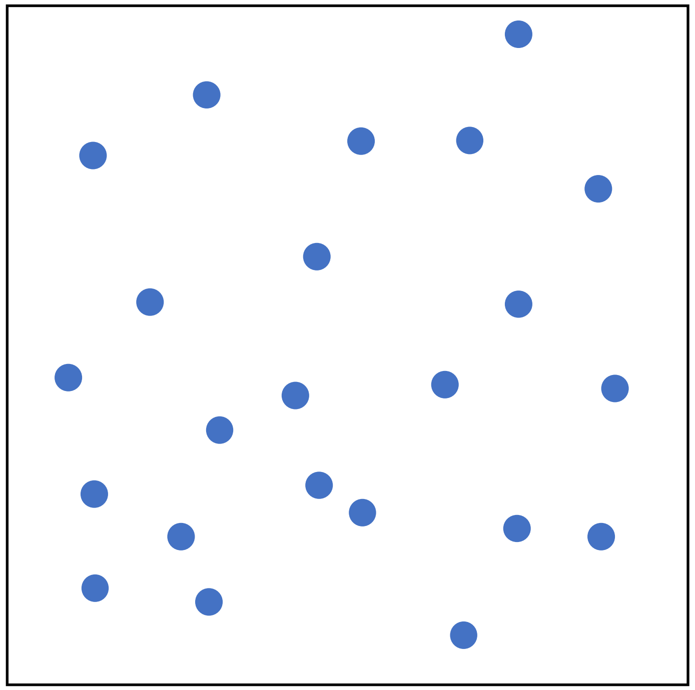
Begin with the **Klimontovich representation** of a plasma (single species for clarity). Here we will consider the **electrostatic approximation**, suitable for many low-temperature plasma systems

$$N(\boldsymbol{q},t) = \sum_p \delta(\boldsymbol{q} - \boldsymbol{Q}_p)$$

$$\boldsymbol{Q}_p = (\boldsymbol{X}_p, \boldsymbol{V}_p) = (X_p, Y_p, Z_p, VX_p, VY_p, VZ_p)$$

$$\frac{d\boldsymbol{Q}_p}{dt} = (\dot{\boldsymbol{X}}_p, \dot{\boldsymbol{V}}_p) = \left(\boldsymbol{V}_p, -\frac{q}{m}\nabla\phi\right)$$

$$\nabla^2\phi = -\frac{q}{\varepsilon_0}\sum_p \delta(\boldsymbol{x} - \boldsymbol{X}_p)$$



Particle-in-Cell

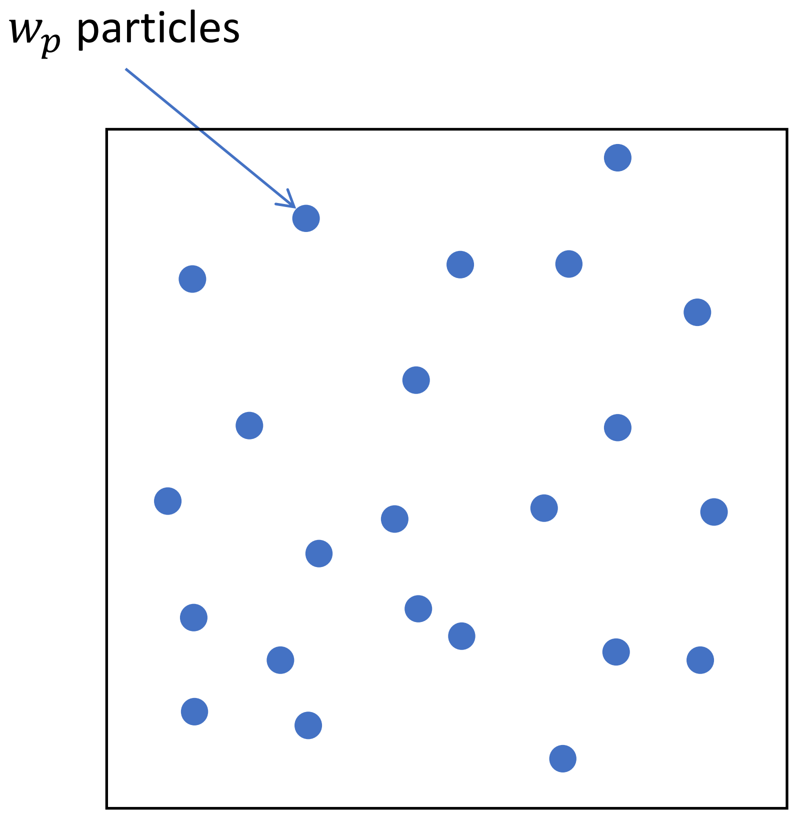
Simulating the actual number of particles in almost all plasma systems is prohibitive, therefore we “clump” them into **macro-particles** with weight w_p

$$N(\boldsymbol{q},t) = \sum_p w_p \delta(\boldsymbol{q} - \boldsymbol{Q}_p)$$

$$\boldsymbol{Q}_p = (\boldsymbol{X}_p, \boldsymbol{V}_p) = (X_p, Y_p, Z_p, VX_p, VY_p, VZ_p)$$

$$\frac{d\boldsymbol{Q}_p}{dt} = (\dot{\boldsymbol{X}}_p, \dot{\boldsymbol{V}}_p) = \left(\boldsymbol{V}_p, -\frac{q}{m}\nabla\phi\right)$$

$$\nabla^2\phi = -\frac{q}{\varepsilon_0}\sum_p \delta(\boldsymbol{x} - \boldsymbol{X}_p)$$



Particle-in-Cell

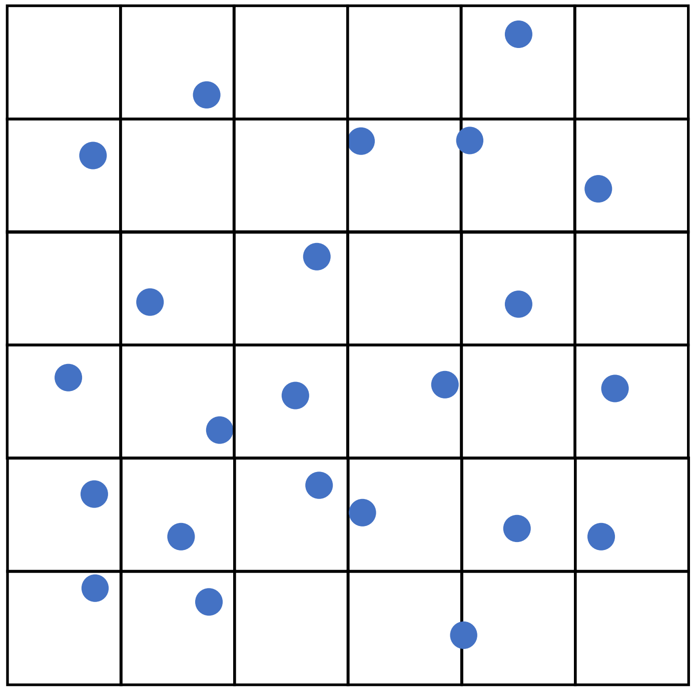
We compute the electrostatic potential on a grid. This smooths out the small time scale inter-particles forces and reduces the cost of solving Poisson's equation

$$N(\boldsymbol{q}, t) = \sum_p w_p \delta(\boldsymbol{q} - \boldsymbol{Q}_p)$$

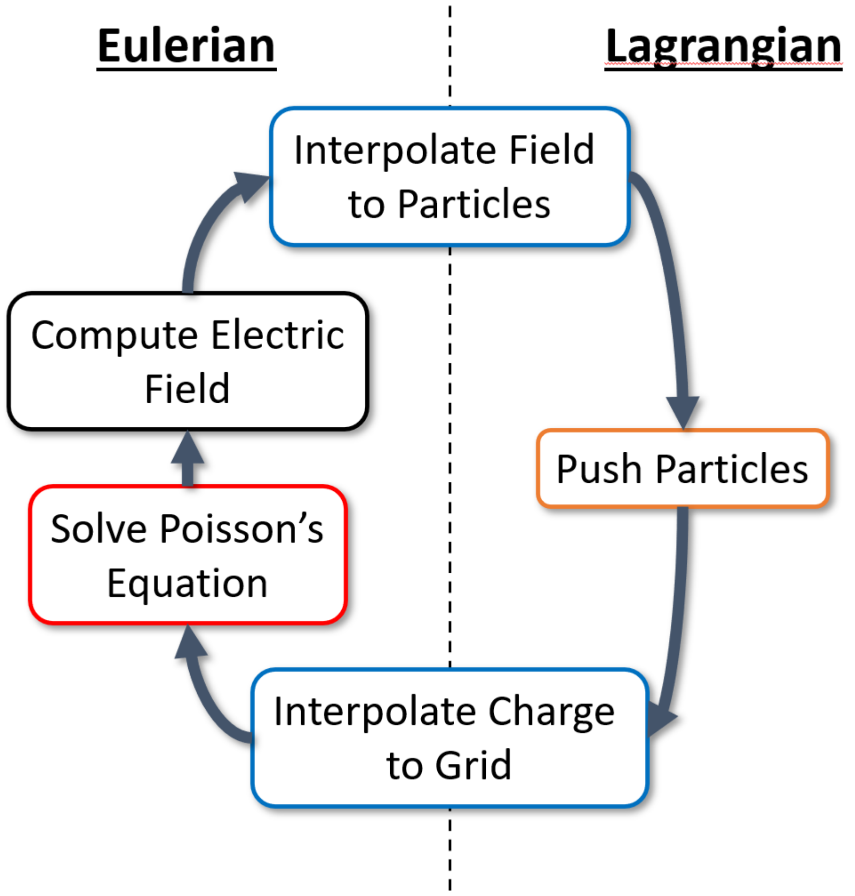
$$\boldsymbol{Q}_p = (\boldsymbol{X}_p, \boldsymbol{V}_p) = (X_p, Y_p, Z_p, VX_p, VY_p, VZ_p)$$

$$\frac{d\boldsymbol{Q}_p}{dt} = (\dot{\boldsymbol{X}}_p, \dot{\boldsymbol{V}}_p) = \left(\boldsymbol{V}_p, -\frac{q}{m} \sum_{i,j,k} S(\boldsymbol{x}_{i,j,k} - \boldsymbol{X}_p)\, [\nabla\phi]_{i,j,k}\right)$$

$$\nabla^2 \phi_{i,j,k} = -\frac{q}{\varepsilon_0} \sum_p S(\boldsymbol{x}_{i,j,k} - \boldsymbol{X}_p)$$

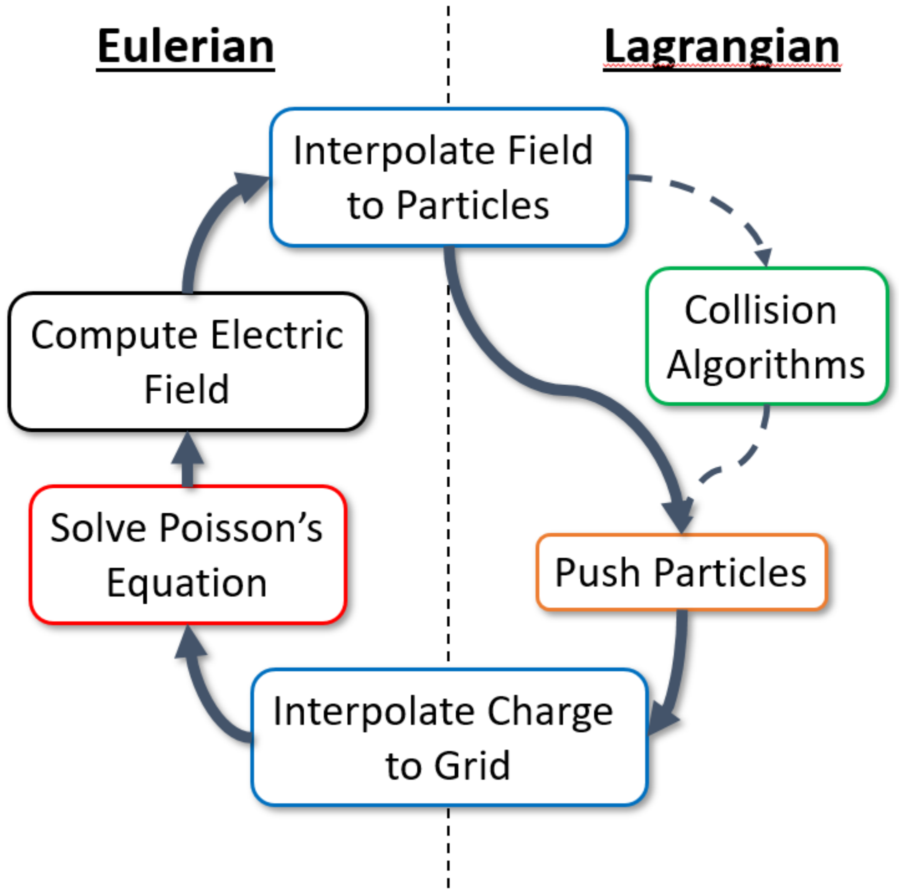


Particle-in-Cell



PIC is therefore a mixed Eulerian/Lagrangian framework for kinetic plasma simulations

Particle-in-Cell

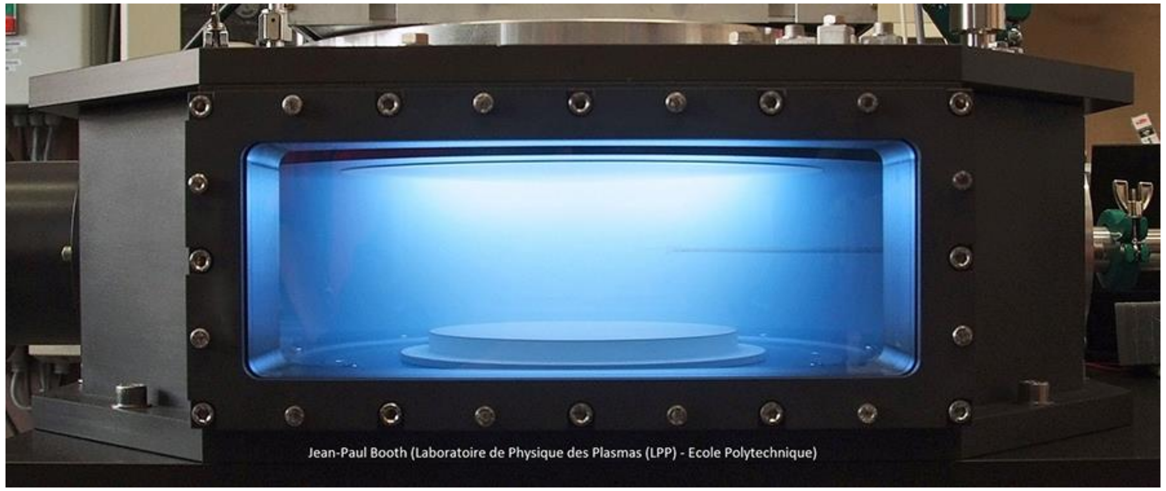


PIC is therefore a mixed Eulerian/Lagrangian framework for kinetic plasma simulations

Fine-scale physics can be added back in via Monte-Carlo collision algorithms

What do we want to model?

- Materials processing plasma reactor
- Assume a $10^{17}\ 1/m^3$ density Argon plasma with $10\ eV$ electrons
- Length scale $\sim 1\ m$
- Time scale $\sim 1\ ms$
- Full 3D kinetic simulation


Jean-Paul Booth (Laboratoire de Physique des Plasmas (LPP) - Ecole Polytechnique)

Materials processing plasma reactor [Booth, Ecole Polytechnique]

Challenges for Partice-in-Cell

Particle-in-cell (and other kinetic methods) suffer from severe stability constraints:

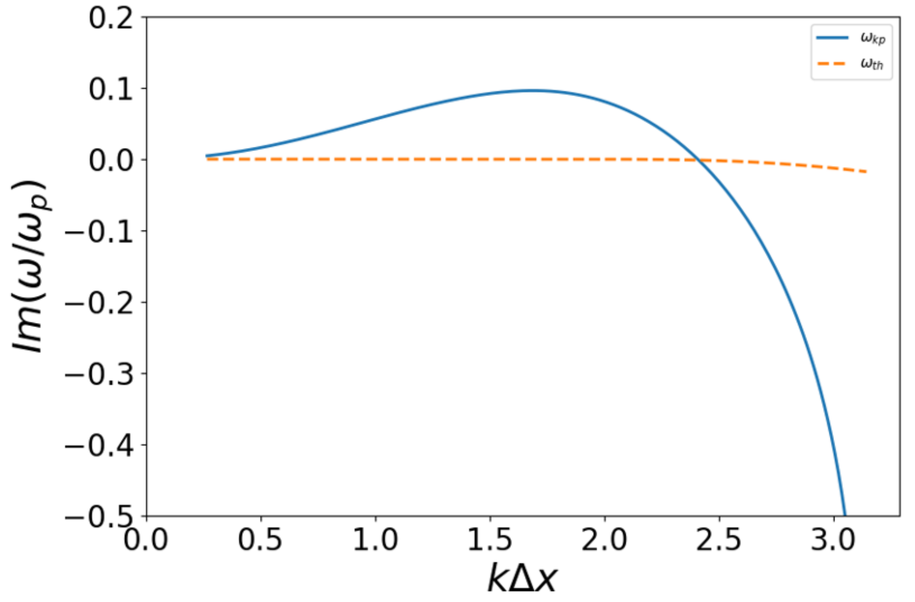
The time step must suitably resolve the plasma frequency:

$$\Delta t \omega_p \approx 0.2$$

The cell size should resolve the Debye length, to avoid the *finite-grid-instability*

$$\Delta x \approx \lambda_D$$

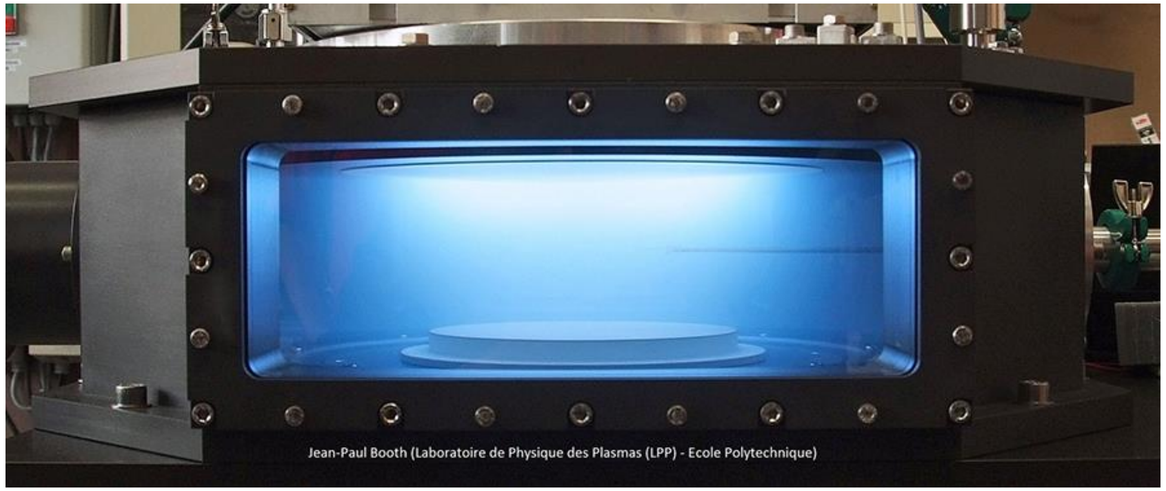
For plasma problems which have vast time and length scales this results in high computational cost



Plasma wave dispersion relationship for $\Delta x = 10\lambda_D$ and 0th order interpolation
(orange) theoretical response
(blue) PIC response

Requirements for Whole Device Modelling

- Materials processing plasma reactor
- Assume a $10^{17}\ 1/m^3$ density Argon plasma with $10\ eV$ electrons
- Length scale $\sim 1\ m$
- Time scale $\sim 1\ ms$
- Based on PIC requirements we need:
  - 10^8 time steps
  - 10^{12} cells
  - 10^{15} particles


Jean-Paul Booth (Laboratoire de Physique des Plasmas (LPP) - Ecole Polytechnique)

Materials processing plasma reactor [Booth, Ecole Polytechnique]

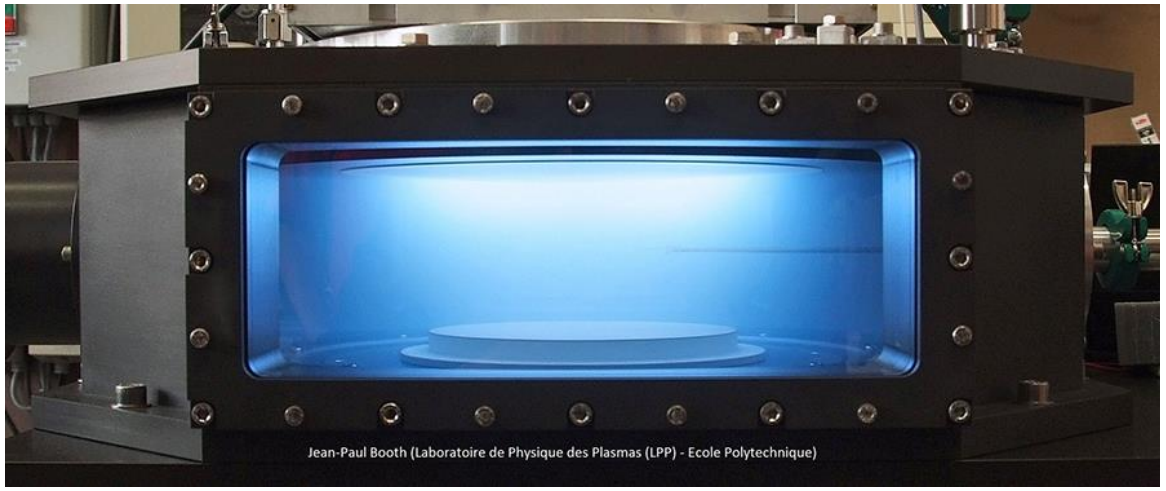
Time Step Requirements for Practical Simulations

We want to perform simulations in a reasonable time-frame to be useful to industry

Target: Simulation time less than 10 days

- For $1\ ms$ we need $\sim 10^8$ time steps, this gives us a performance goal of $t_{step} \approx 10\ ms$

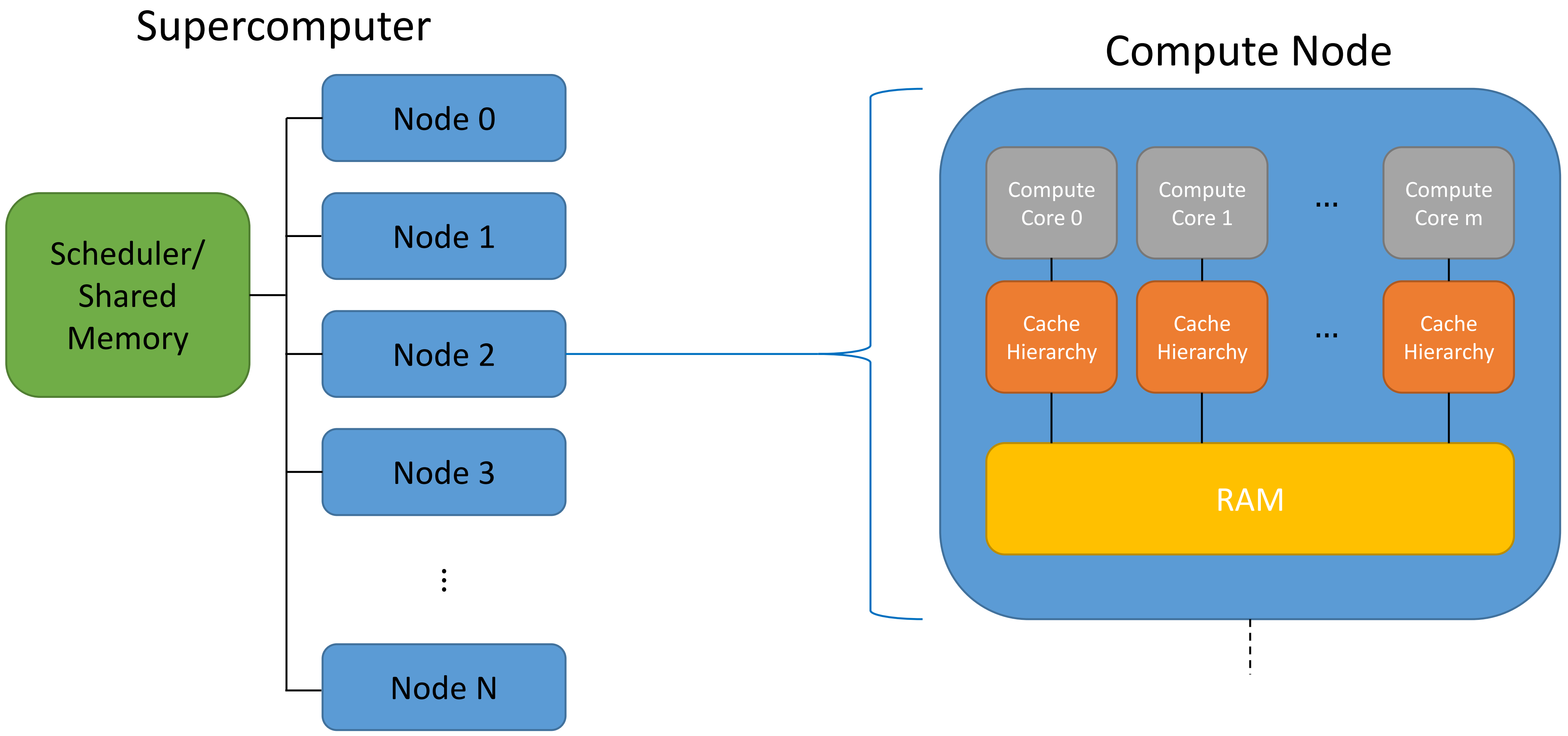
- We need high performance super computers to meet these goals


Jean-Paul Booth (Laboratoire de Physique des Plasmas (LPP) - Ecole Polytechnique)

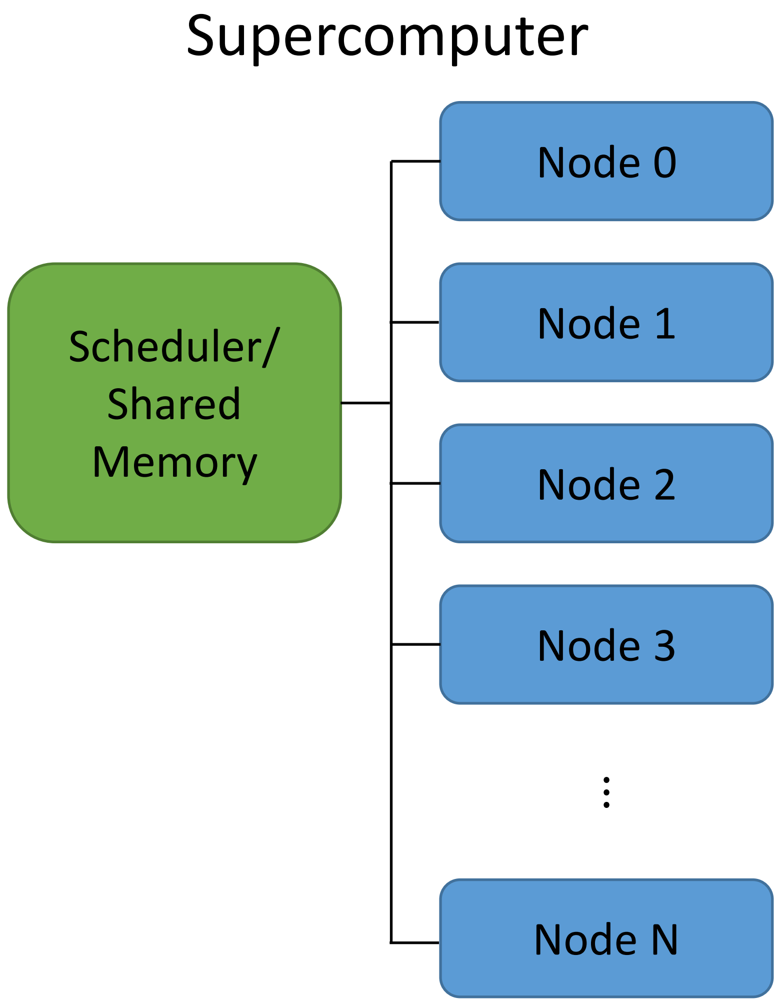
Materials processing plasma reactor [Booth, Ecole Polytechnique]

Leveraging Modern Supercomputers

Anatomy of a Supercomputer



Coding for Supercomputers

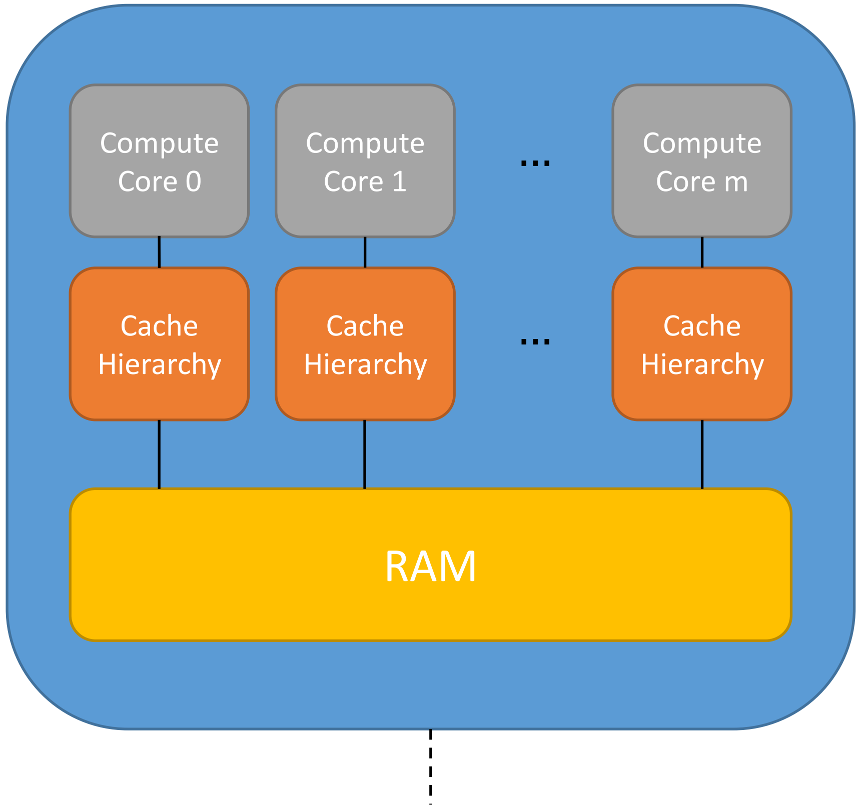


- Data sharing between nodes is the slowest form of data transfer on the system
- Communication of data between nodes should be minimized as much as possible
- Data communication is generally handled by a *multiple-data multiple-instruction* (MIMD) paradigm
    - OpenMPI
    - MPICH
    - Intel MPI

Coding for Supercomputers

Compute nodes have shared memory

Compute Node

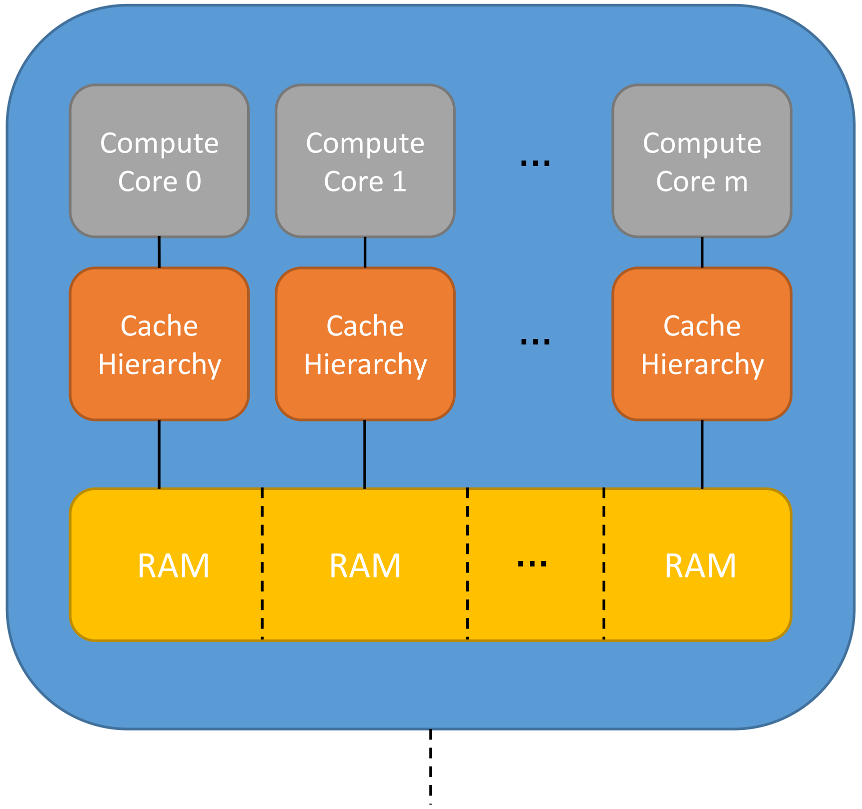


Coding for Supercomputers

Compute nodes have shared memory

We could allocate different portions of this memory to different compute cores using MIMD instructions

Compute Node



Coding for Supercomputers

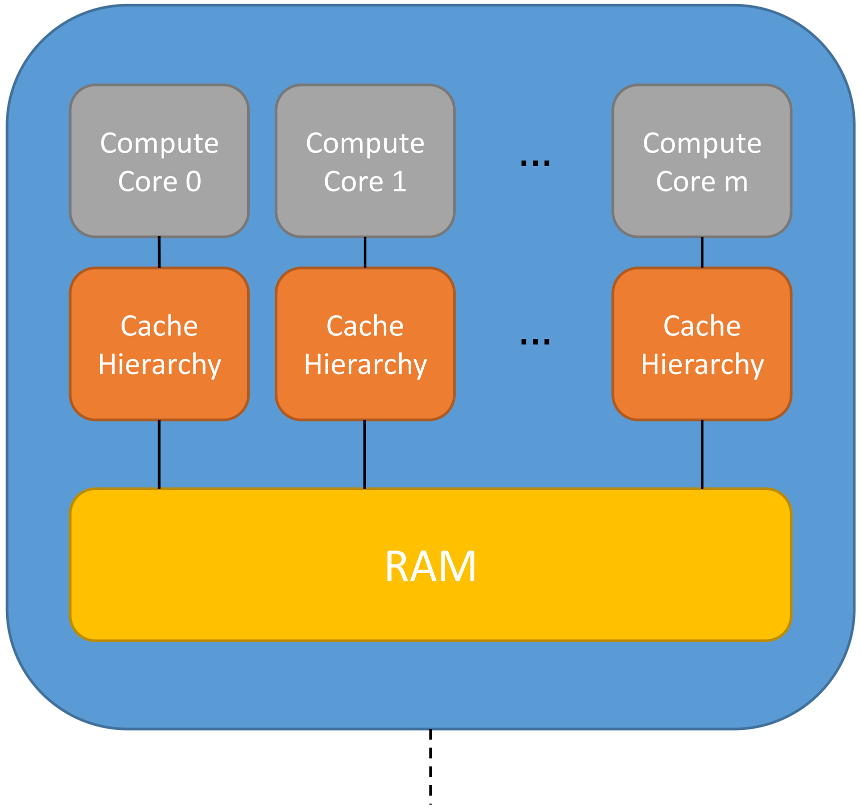
Compute nodes have shared memory

We could allocate different portions of this memory to different compute cores using MIMD instructions

But the modern trend is to use what are called *single-instruction multiple-data* (SIMD) instruction sets

The standard for this is known as OpenMP and comes with all modern compilers

Compute Node

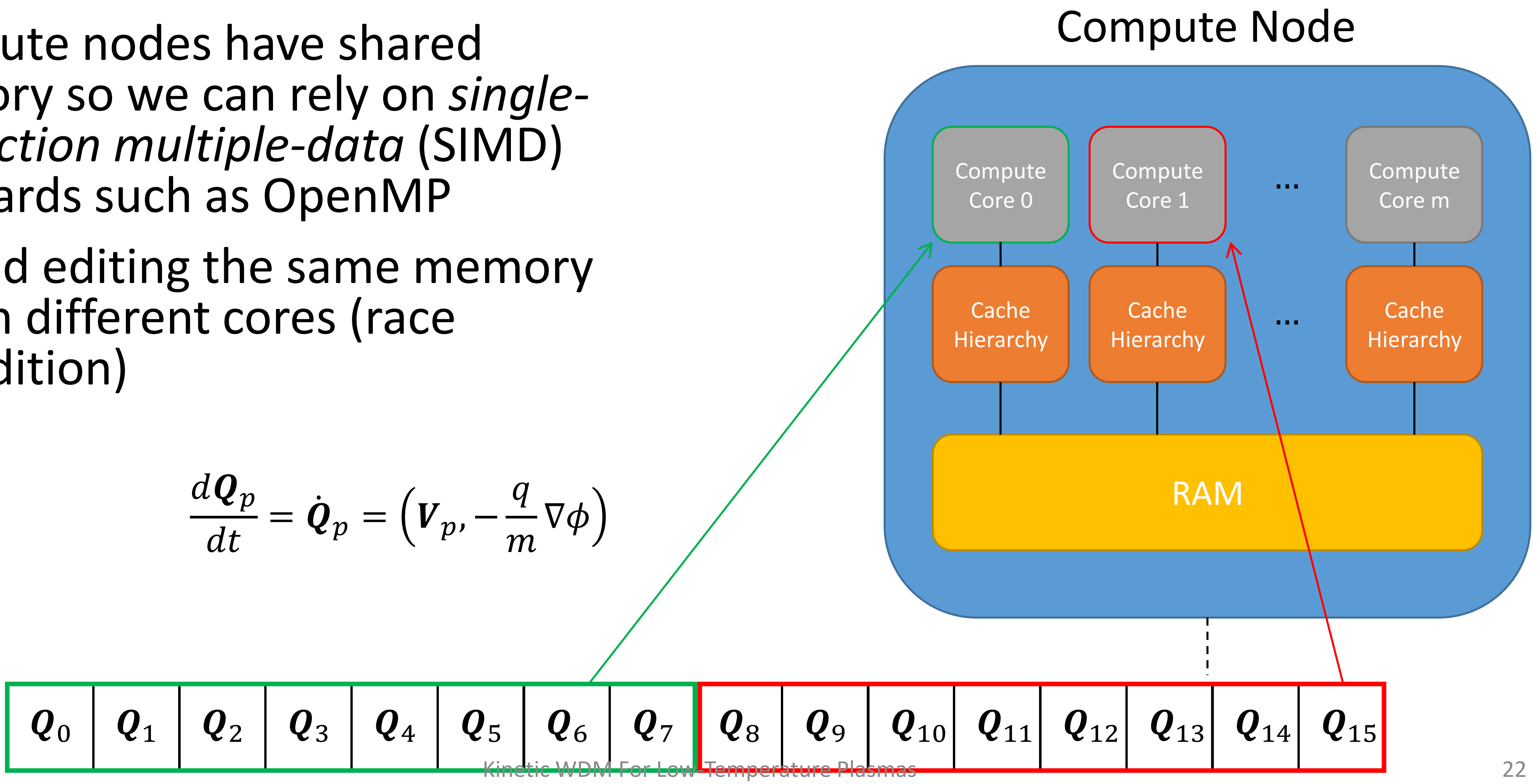


Coding for Supercomputers

Compute nodes have shared memory so we can rely on *single-instruction multiple-data* (SIMD) standards such as OpenMP

- Avoid editing the same memory from different cores (race condition)

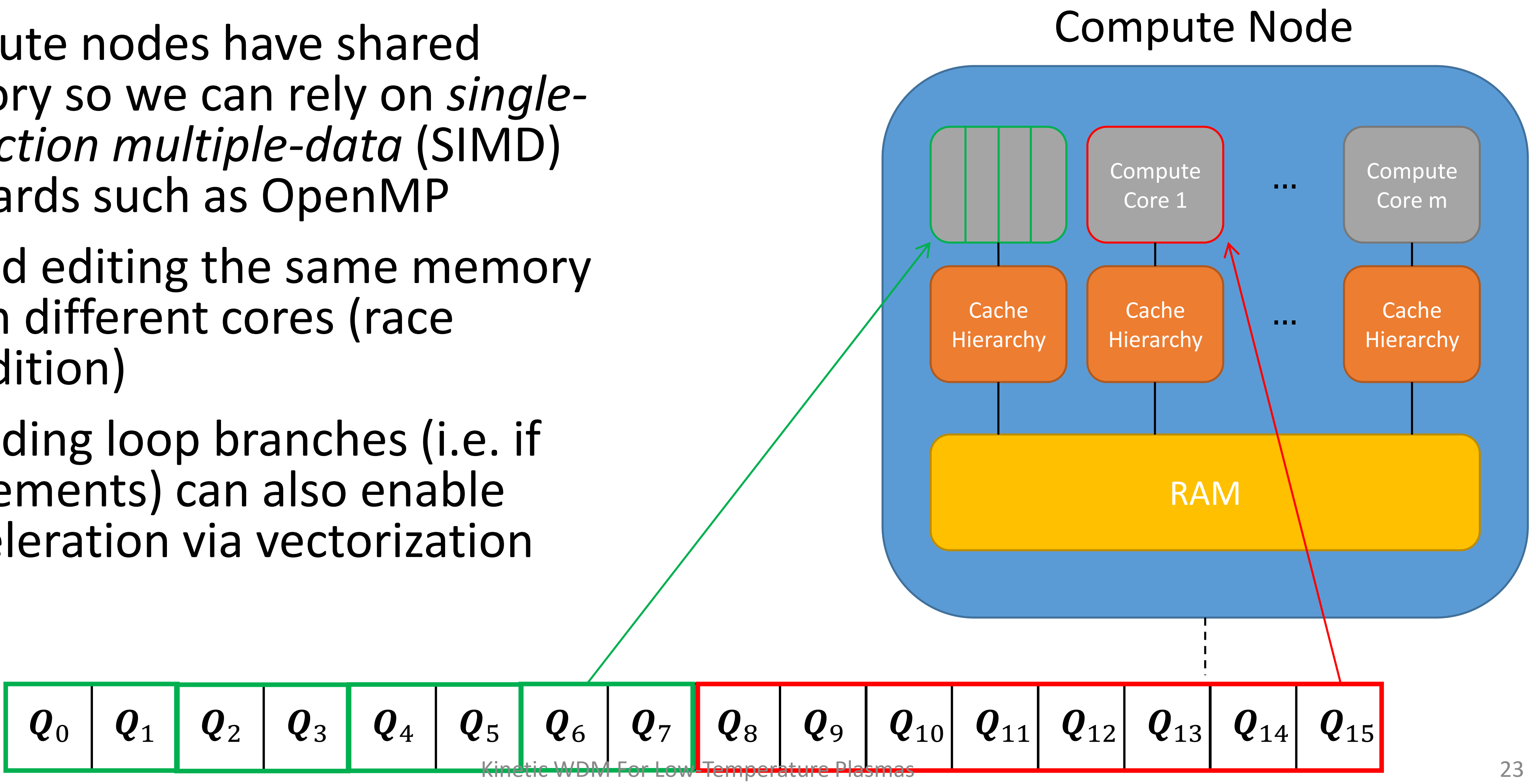
$$\frac{d\boldsymbol{Q}_p}{dt} = \dot{\boldsymbol{Q}}_p = \left(\boldsymbol{V}_p, -\frac{q}{m}\nabla\phi\right)$$


Compute Node
Compute Core 0
Compute Core 1
…
Compute Core m
Cache Hierarchy
Cache Hierarchy
…
Cache Hierarchy
RAM
Q_0 Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7
Q_8 Q_9 Q_{10} Q_{11} Q_{12} Q_{13} Q_{14} Q_{15}

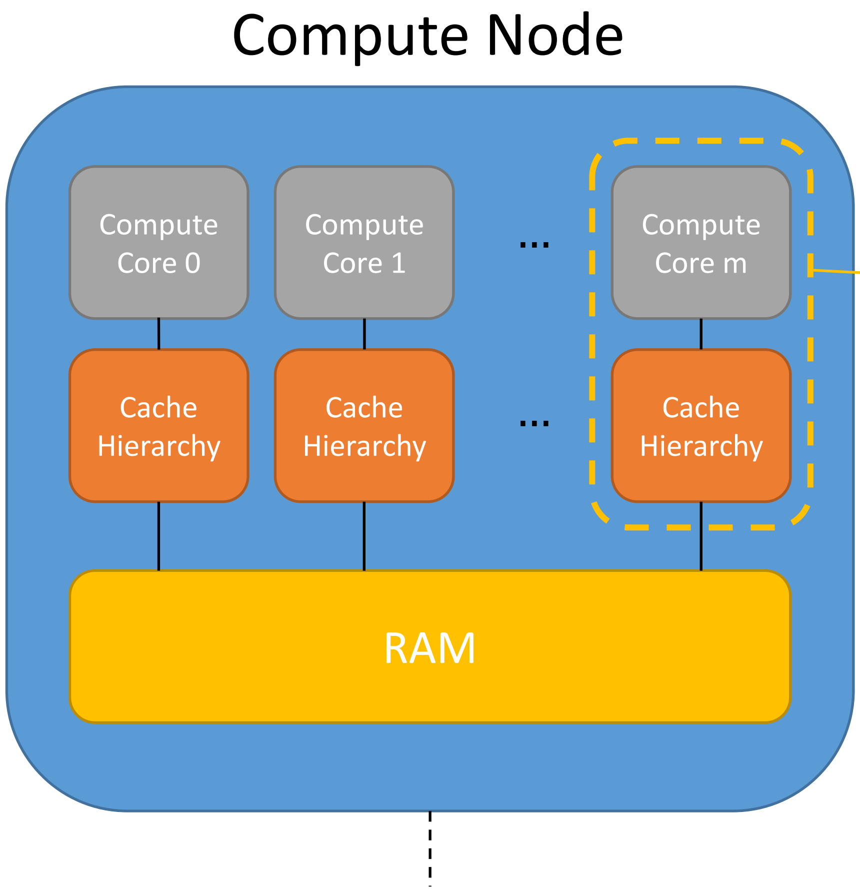
Coding for Supercomputers

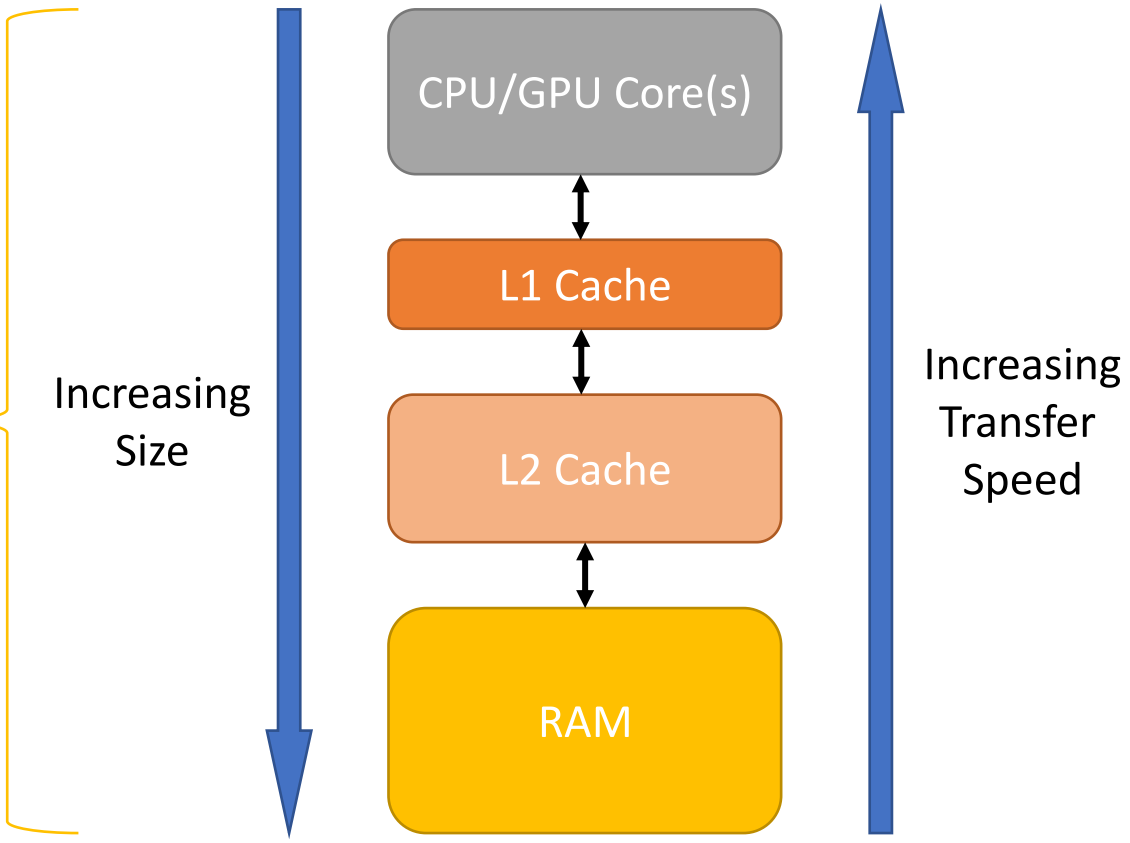
Compute nodes have shared memory so we can rely on *single-instruction multiple-data* (SIMD) standards such as OpenMP

- Avoid editing the same memory from different cores (race condition)
- Avoiding loop branches (i.e. if statements) can also enable acceleration via vectorization


Compute Node
Compute Core 1
...
Compute Core m
Cache Hierarchy
Cache Hierarchy
...
Cache Hierarchy
RAM
Q_0 Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7 Q_8 Q_9 Q_{10} Q_{11} Q_{12} Q_{13} Q_{14} Q_{15}

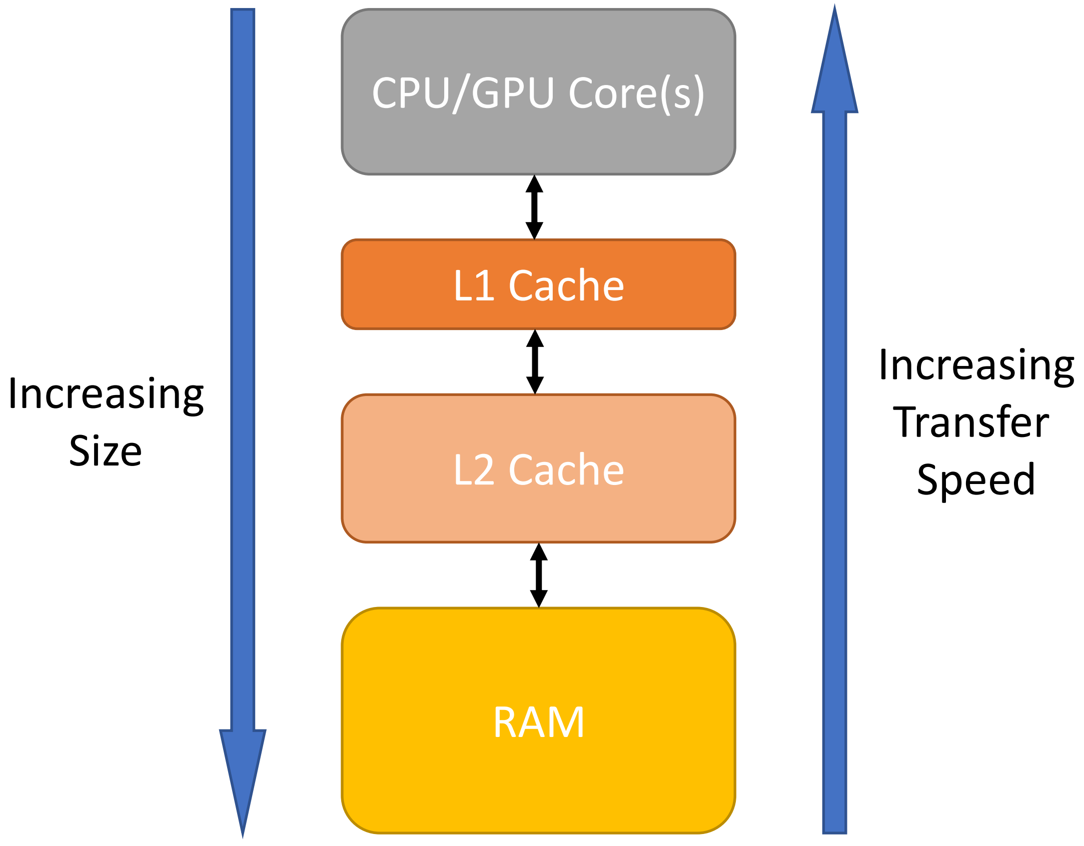
The Memory Hierarchy

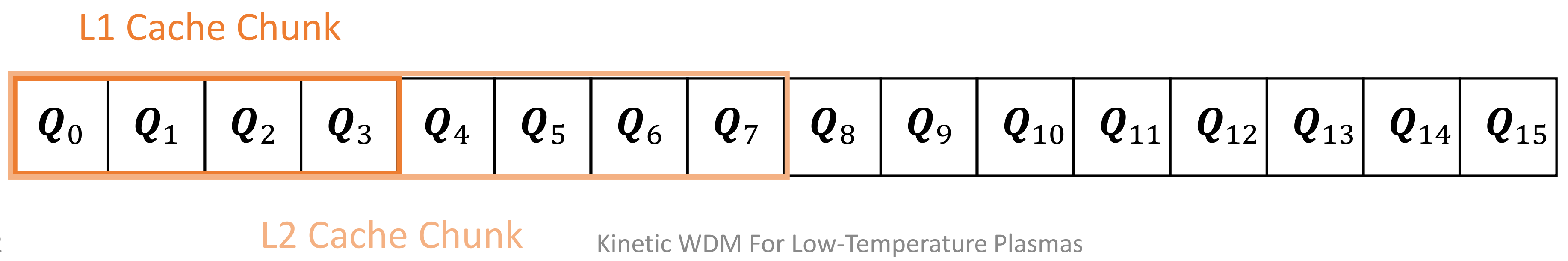




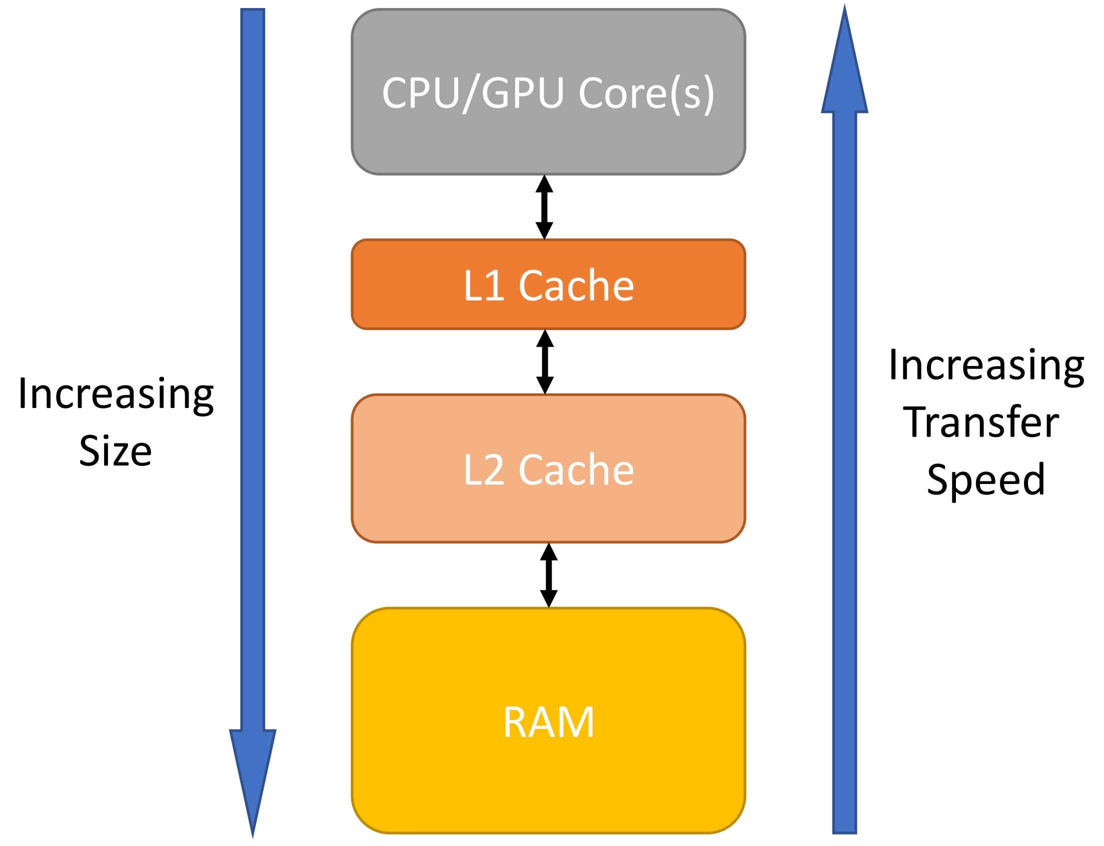
The Memory Hierarchy

Many algorithms access memory contiguously (i.e. all in order). Knowing this, CPUs will load *chunks* of memory all at once




L1 Cache Chunk
Q_0 Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7 Q_8 Q_9 Q_{10} Q_{11} Q_{12} Q_{13} Q_{14} Q_{15}
L2 Cache Chunk

The Memory Hierarchy



<u>Why PIC is good</u>

- When updating particles we access memory contiguously and using the same instructions. This leads to optimal memory use efficiency

$$\frac{d\boldsymbol{Q}_p}{dt} = \dot{\boldsymbol{Q}}_p = \left(\boldsymbol{V}_p, -\frac{q}{m}\nabla\phi\right)$$

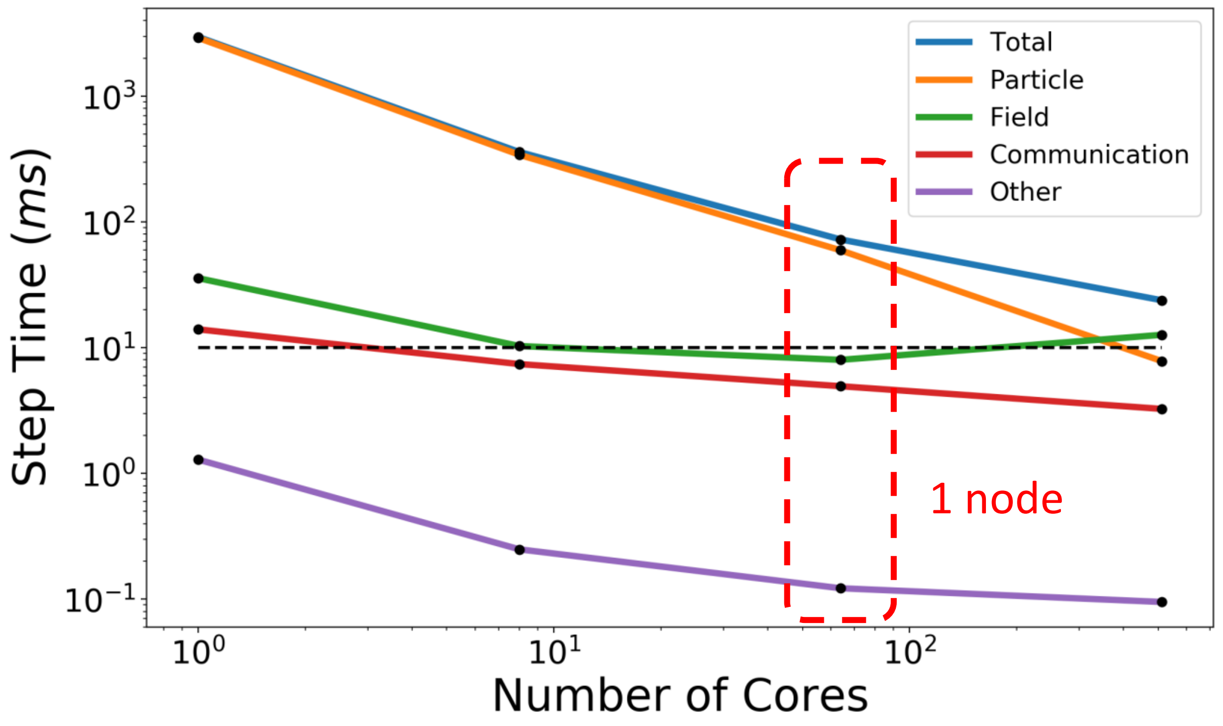
$\boldsymbol{Q}_0$	$\boldsymbol{Q}_1$	$\boldsymbol{Q}_2$	$\boldsymbol{Q}_3$	$\boldsymbol{Q}_4$	$\boldsymbol{Q}_5$	$\boldsymbol{Q}_6$	$\boldsymbol{Q}_7$	$\boldsymbol{Q}_8$	$\boldsymbol{Q}_9$	$\boldsymbol{Q}_{10}$	$\boldsymbol{Q}_{11}$	$\boldsymbol{Q}_{12}$	$\boldsymbol{Q}_{13}$	$\boldsymbol{Q}_{14}$	$\boldsymbol{Q}_{15}$

Strong Scaling in 3D

Modelling a thermal cube of plasma with periodic boundary conditions

$32 \times 32 \times 32$ cells with 1,000 particles-per-cell

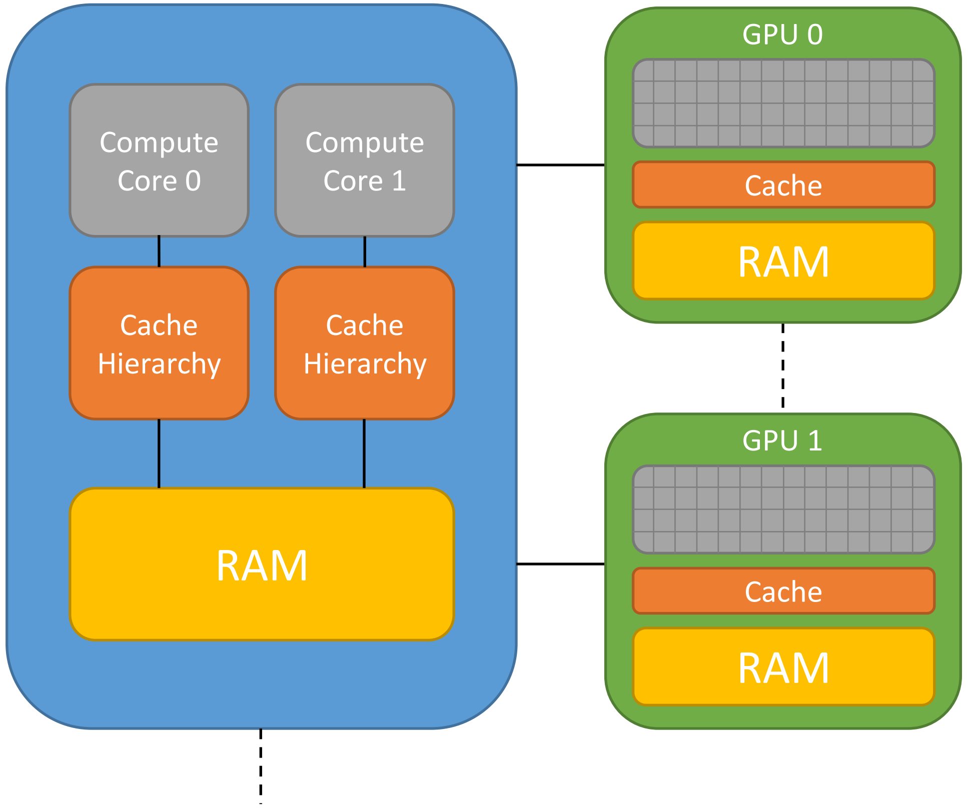
$$\Delta x = \lambda_D, \;\; \Delta t \omega_p = 0.2$$



Acceleration

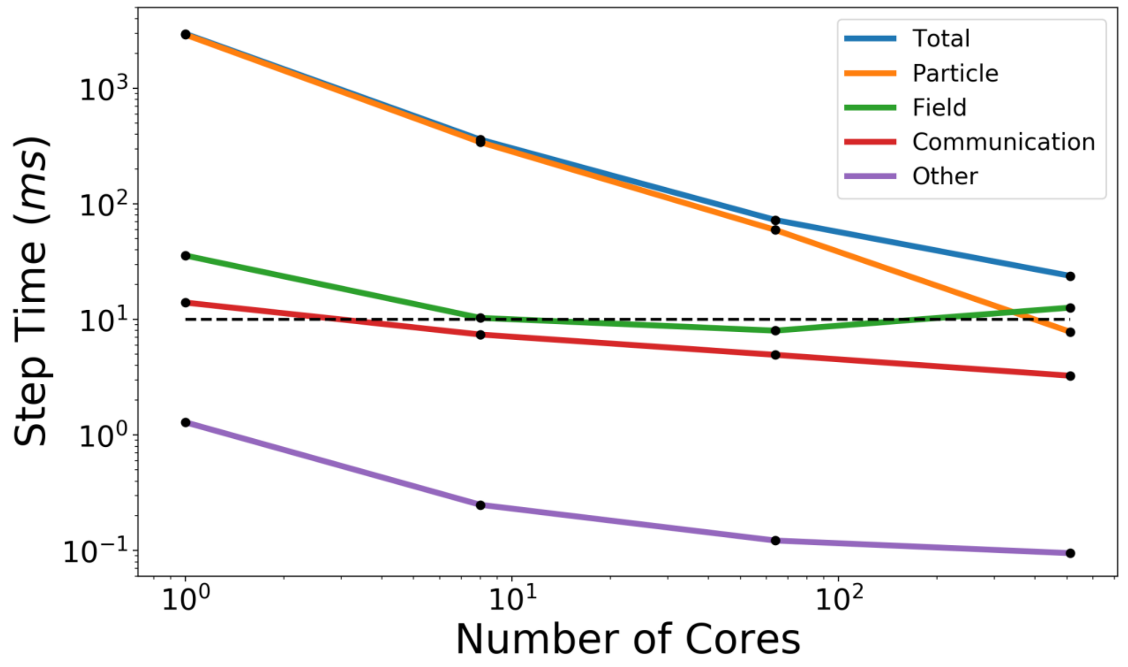
- GPUs are like a compute node but with thousands of low performance cores
- The perform well on algorithms which repeat the same thing on huge amounts of data
- This makes PIC particle routines (update, collisions, interpolation) ideal for acceleration!
- Data must be transferred from the node to CPU to the GPU which can be time consuming

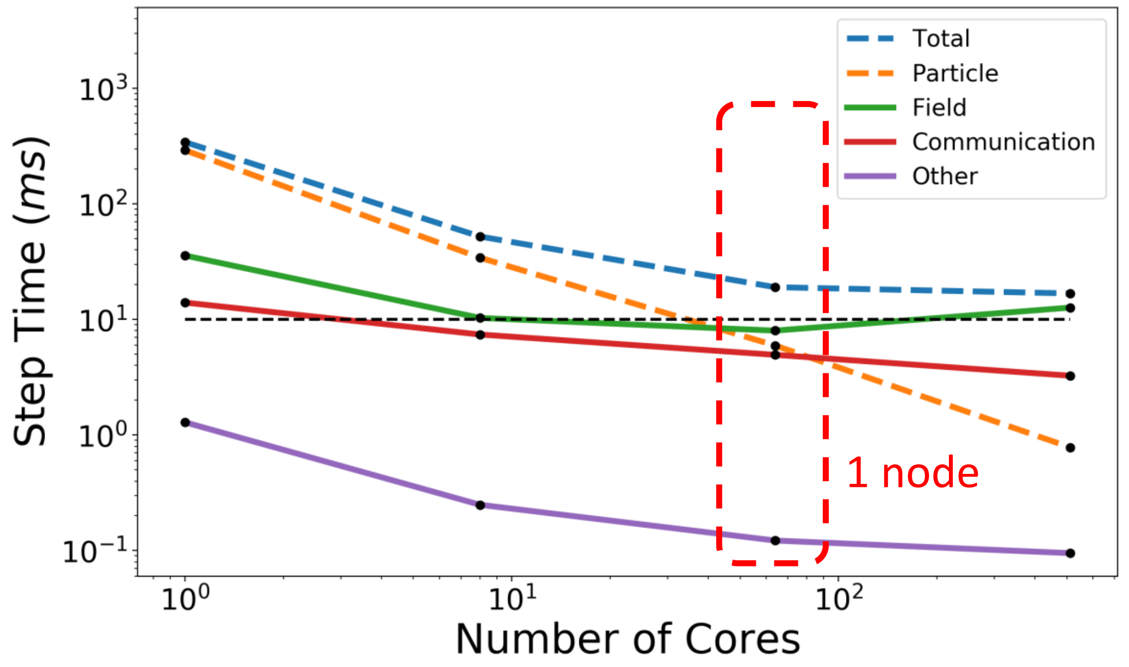
- Highly recommend 2021 paper by *Juhasz, Durian, Derzsi, Matejcik, Donko* and *Hartmann* for far more details!



Improved Strong Scaling in 3D

Brings us closer to the goal of 10ms time step for 3D simulations

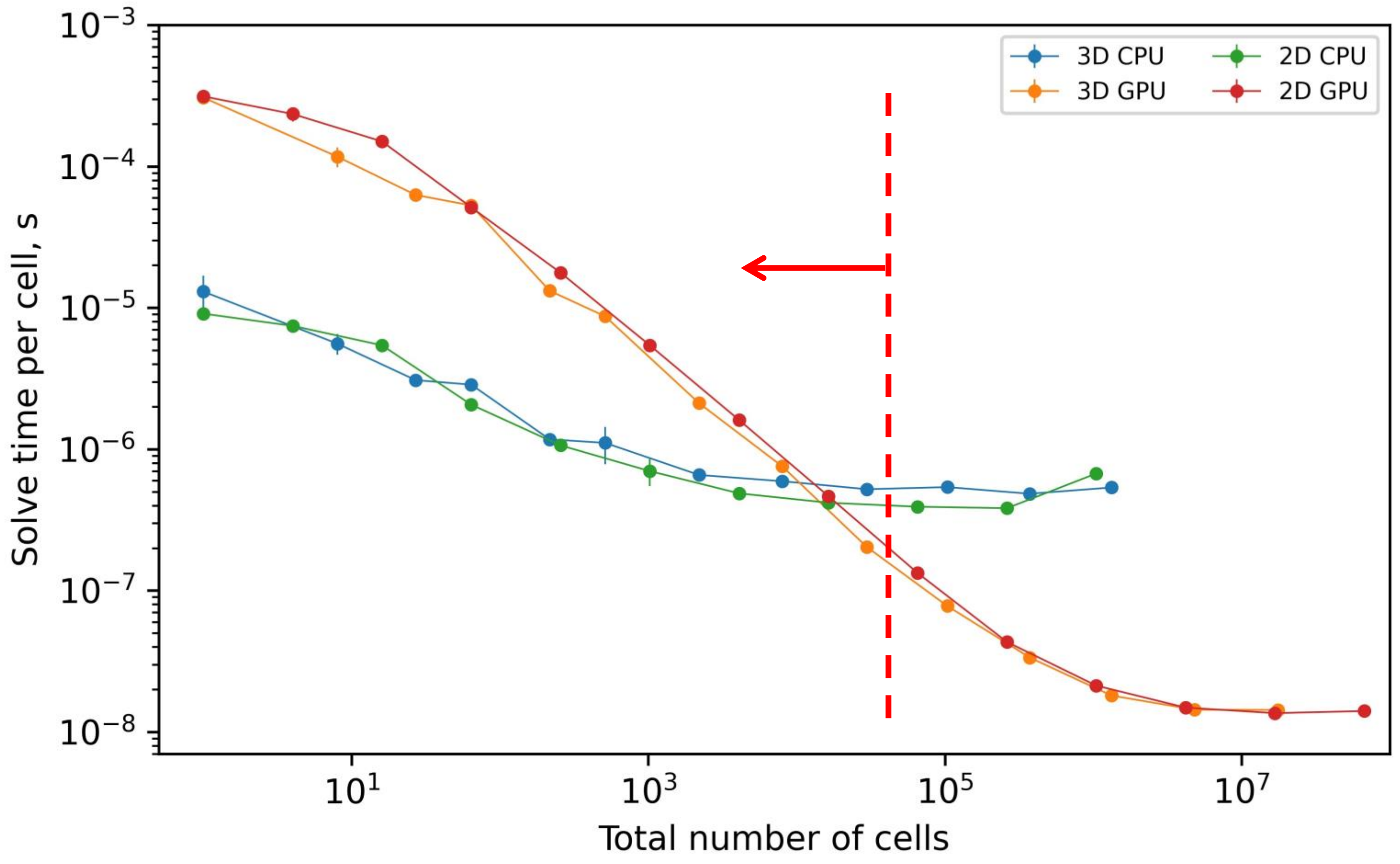




What about the Field Solver?

Can we achieve a speedup with the field solver on GPUs?

Unfortunately in the region where GPUs could help, the time step is too slow for whole device modelling

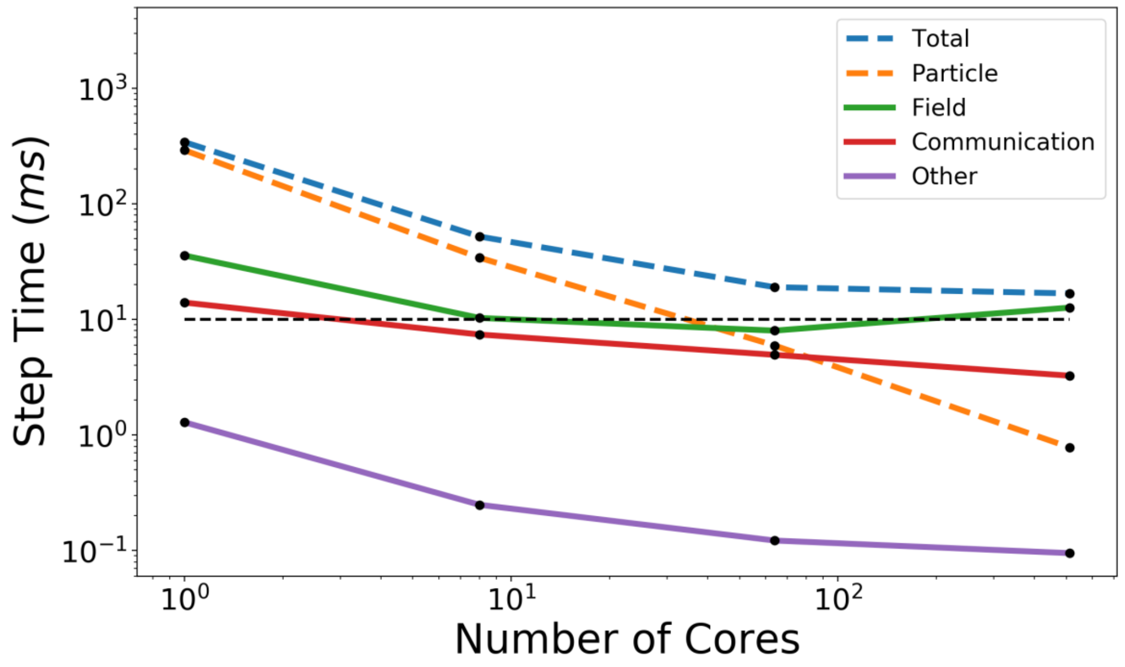


GPUs are Not Enough

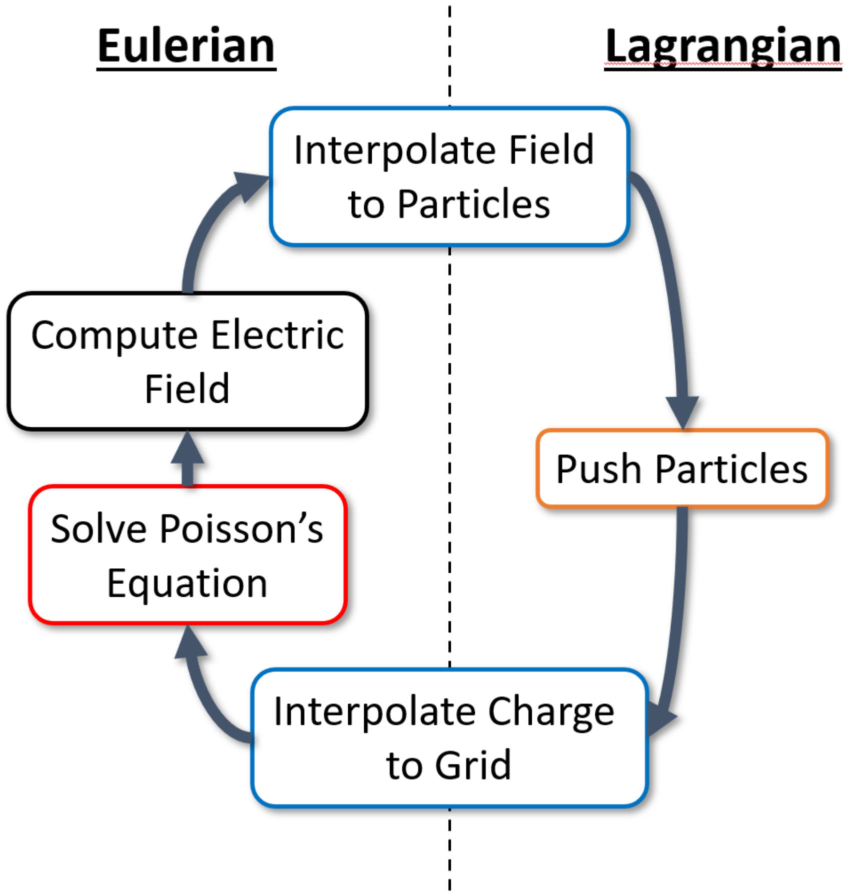
Development of new algorithms for kinetic WDM

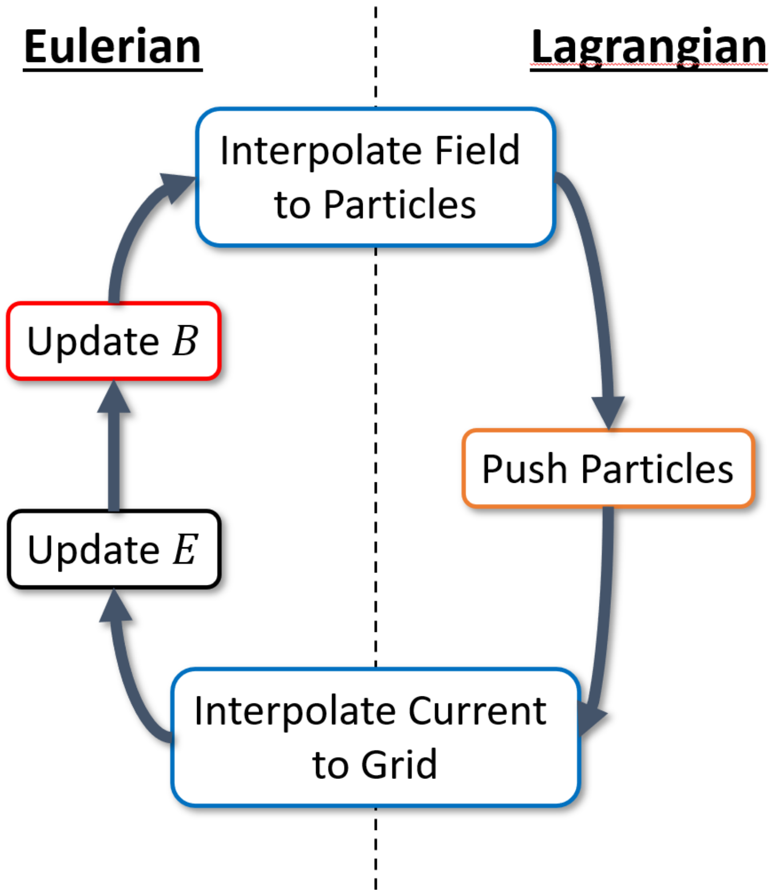
Re-considering the Electrostatic Approximation

The Poisson equation is now the bottleneck to performance



Poisson vs Maxwell Field Solver

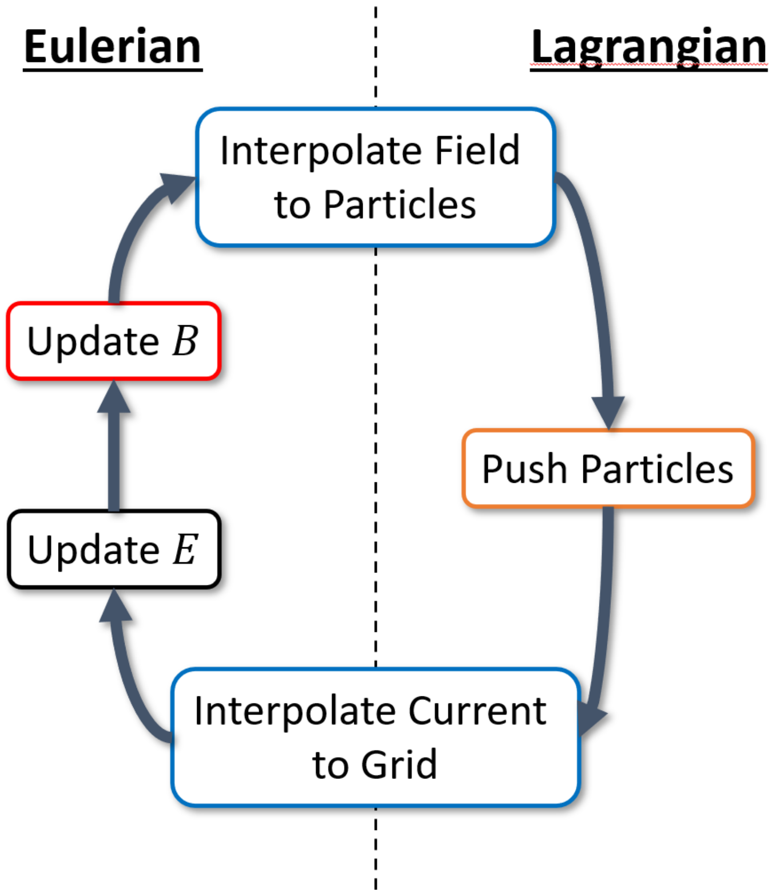




Poisson vs Maxwell Field Solver

For our particles, we need

$$\Delta t_{part} < 0.2/\omega_p$$



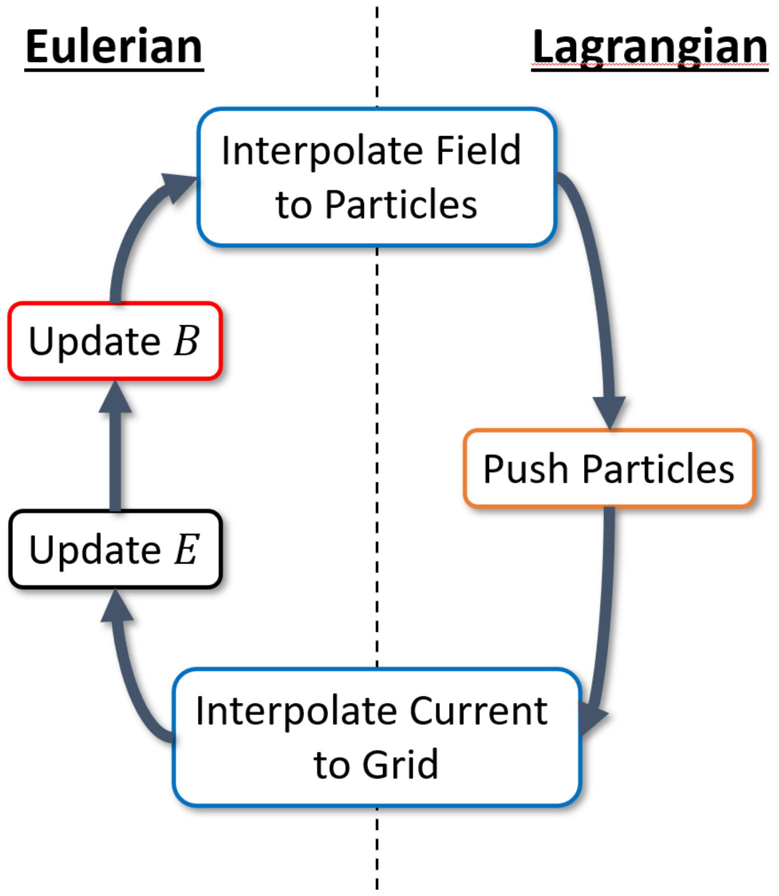
Poisson vs Maxwell Field Solver

For our particles, we

$$\Delta t_{part} < 0.2/\omega_p$$

But, for an explicit electromagnetic field solver we require:

$$\Delta t_{field} < \Delta x/c$$



Poisson vs Maxwell Field Solver

For our particles, we need

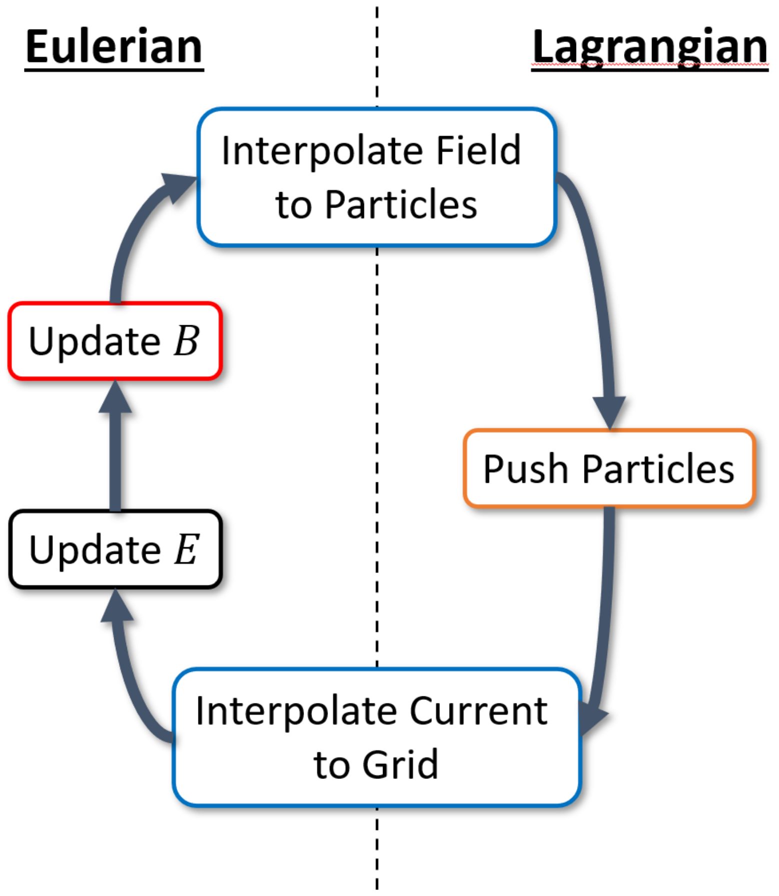
$$\Delta t_{part} < 0.2/\omega_p$$

But, for an explicit electromagnetic field solver we require:

$$\Delta t_{field} < \Delta x/c$$

For a 10 eV plasma with $\Delta x = \lambda_D$ we have

$$n_c = \frac{\Delta t_{part}}{\Delta t_{field}} = \frac{0.2c}{v_{th}} \approx 50$$



Sub-cycled Maxwell Solver

For our particles, we need

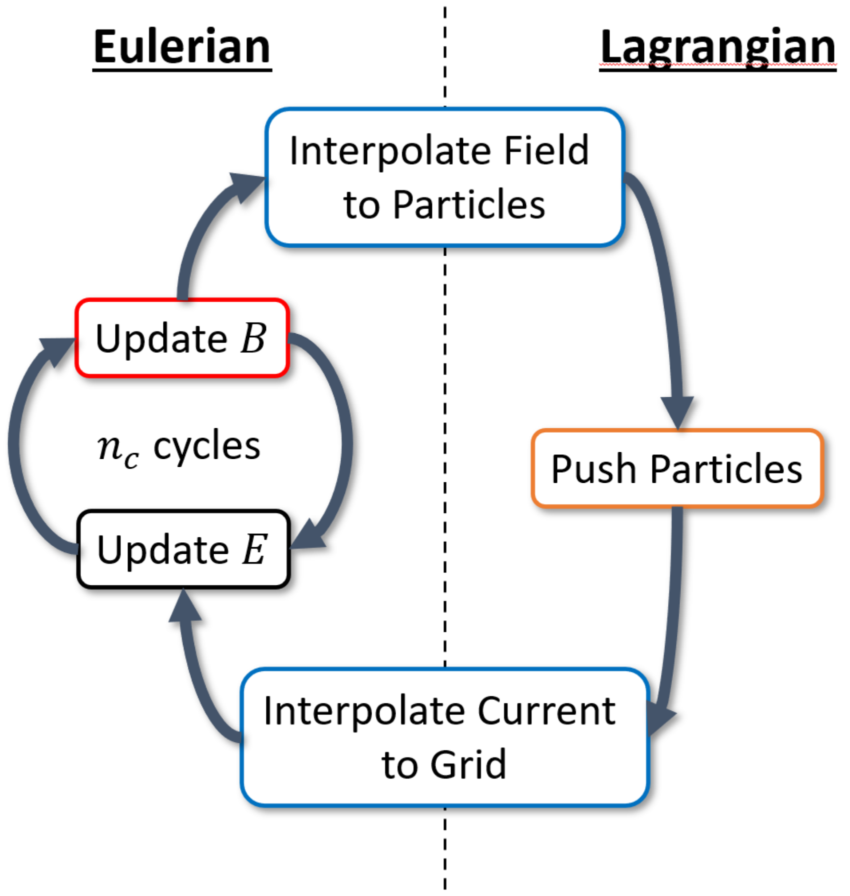
$$\Delta t_{part} < 0.2/\omega_p$$

But, for an explicit electromagnetic field solver we require:

$$\Delta t_{field} < \Delta x/c$$

For a 10 eV plasma with $\Delta x = \lambda_D$ we have

$$n_c = \frac{\Delta t_{part}}{\Delta t_{field}} = \frac{0.2c}{v_{th}} \approx 50$$

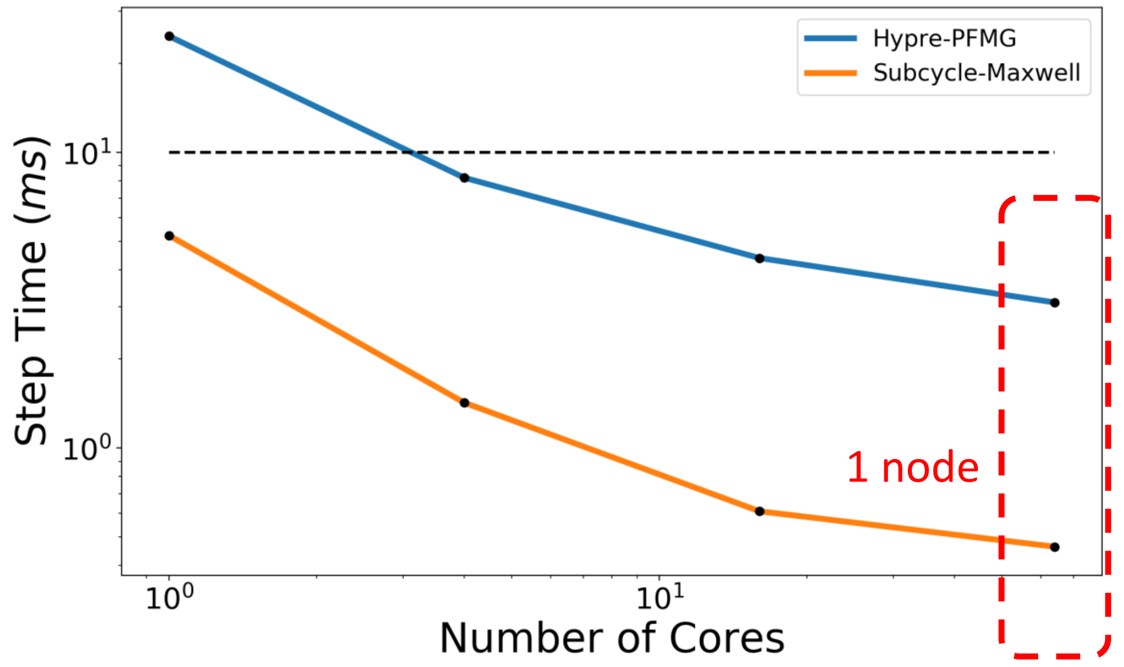


Sub-cycling a Maxwell Field Solver

Strong scaling tests of the Hypre-PFMG solver vs the sub-cycled Maxwell solver

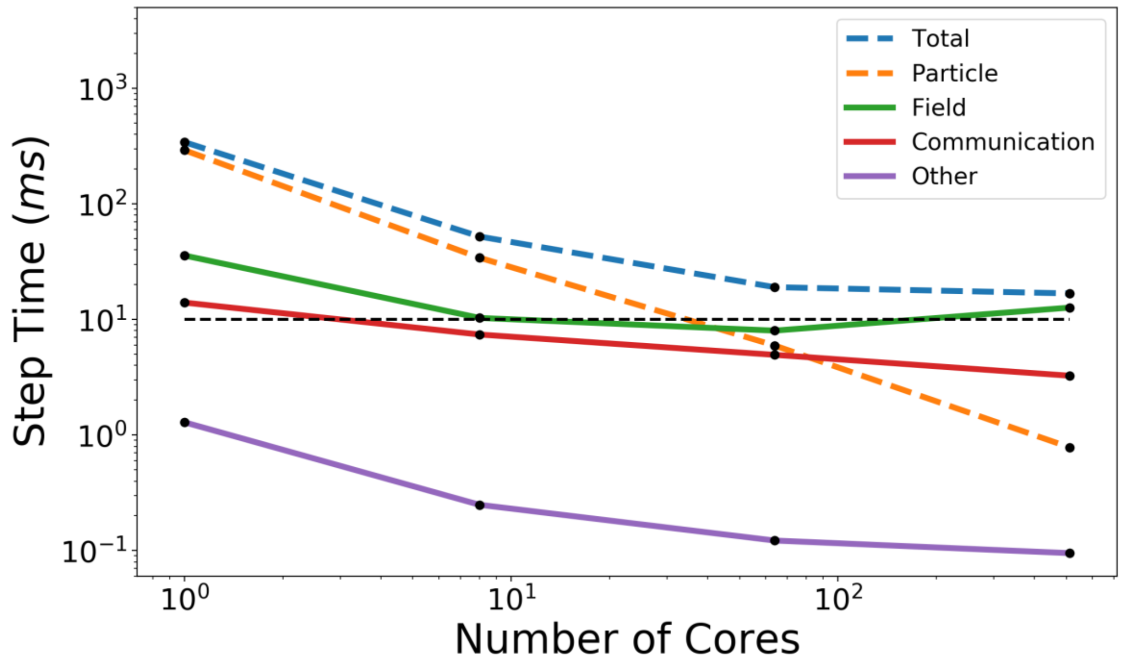
2D Simulation: 256×256 cells

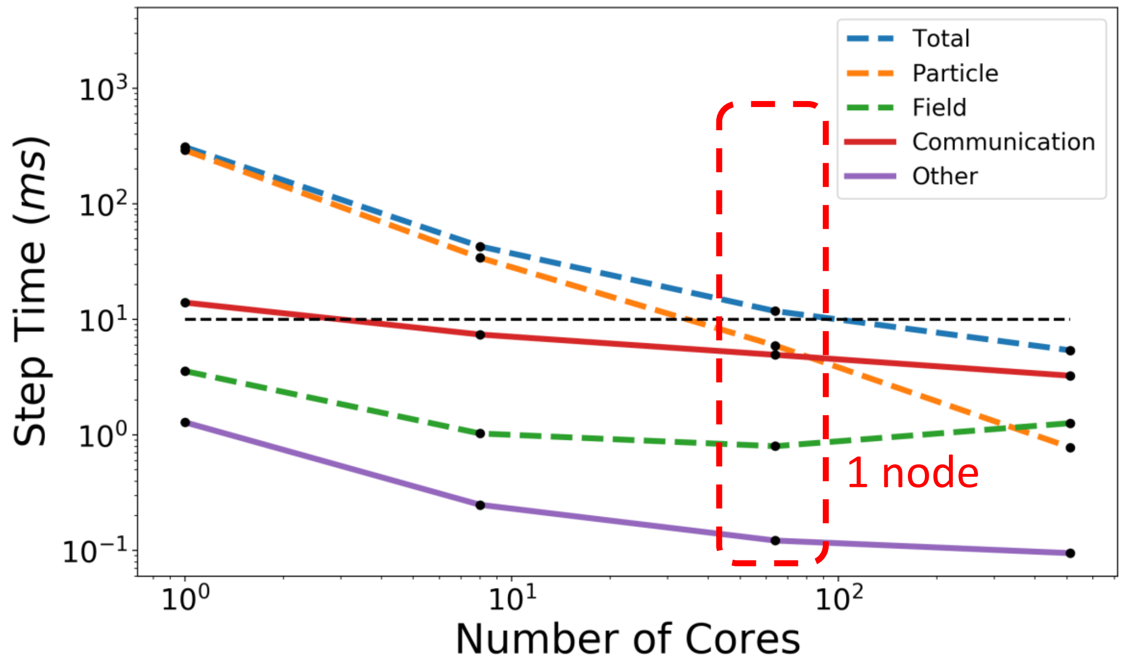
We see a $7 - 12 \times$ speedup with the sub-cycled Maxwell solver



Improved Strong Scaling in 3D

Allows us to achieve the goal of 10ms time step size for 3D simulations





Sub-cycling a Maxwell Field Solver

Advantages:

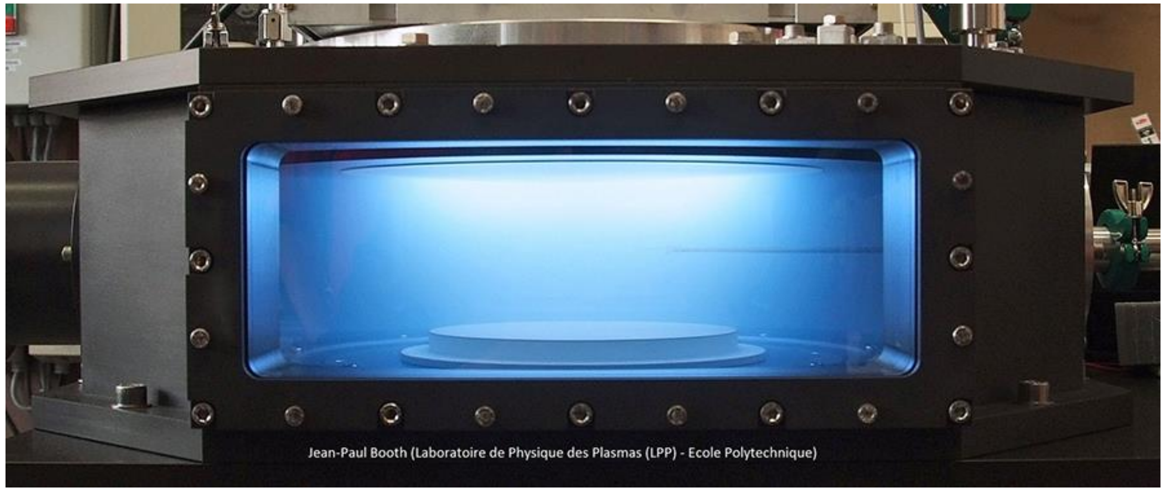
- Faster!
- Perfectly scalable, communication time equivalent to domain boundary size
- Less dependent on external software
- We can model EM effects in low-temperature devices!

Disadvantages:

- Stability issues
- Might be noisy?

Requirements for Whole Device Modelling

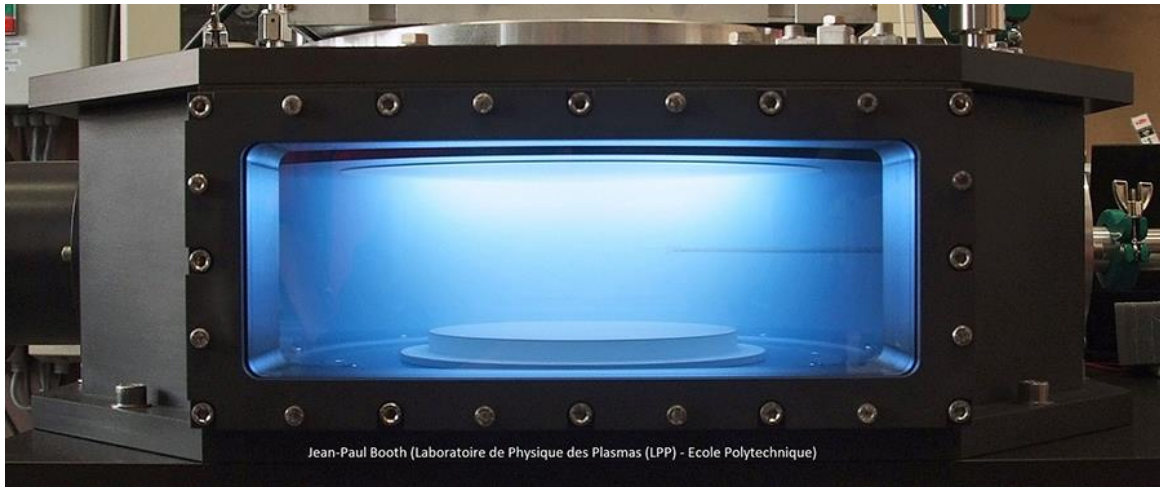
- Materials processing plasma reactor
- Assume a $10^{17}\ 1/m^3$ density Argon plasma with $10\ eV$ electrons
- Length scale $\sim 1\ m$
- Time scale $\sim 1\ ms$
- Based on PIC requirements we need:
  - 10^8 time steps
  - 10^{12} cells
  - 10^{15} particles
- If we give each node 32^3 cells with $1{,}000$ particle-per-cell we need 30 million compute nodes!
- This is not even possible, let alone practical for industrial design


Jean-Paul Booth (Laboratoire de Physique des Plasmas (LPP) - Ecole Polytechnique)

Materials processing plasma reactor [Booth, Ecole Polytechnique]

Requirements for Whole Device Modelling

- Materials processing plasma reactor
- Assume a $10^{17}\ 1/m^3$ density Argon plasma with $10\ eV$ electrons
- Length scale $\sim 1\ m$
- Time scale $\sim 1\ ms$
- Based on PIC requirements we need:
  - 10^8 time steps
  - 10^{12} cells
  - 10^{15} particles
- If we give each node 32^3 cells with $1{,}000$ particle-per-cell we need 30 million compute nodes!
- We *must* relax the constraint on the cell size and allow for $\Delta x \gg \lambda_D$


Jean-Paul Booth (Laboratoire de Physique des Plasmas (LPP) - Ecole Polytechnique)

Materials processing plasma reactor [Booth, Ecole Polytechnique]

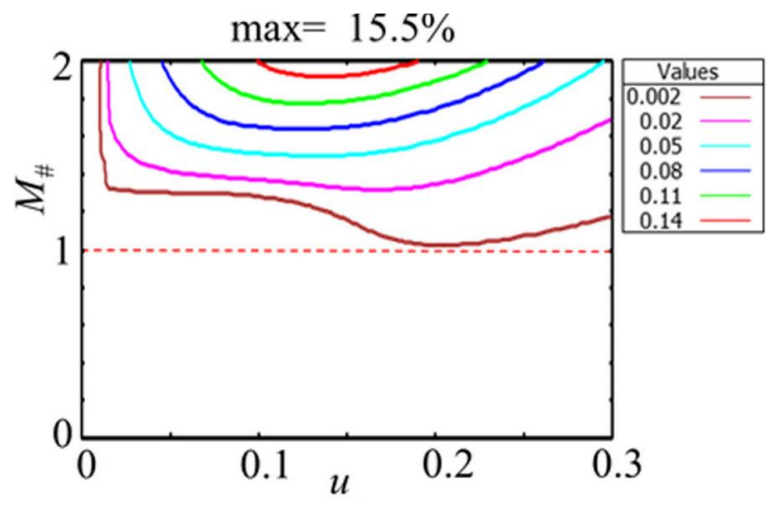
Energy Conserving PIC

This can be achieved via energy-conserving PIC methods [Lewis 1970, Eremin 2022]

Seems to have been rejected by the community due to non-physical cold beam plasma instabilities

Recent work by Barnes & Chacon (2021) has demonstrated that the energy-conserving PIC method is appropriate for modelling low-temperature plasma devices dominated by ambipolar electric fields

Importantly, does not suffer from the *finite-grid-instability* allowing $\Delta x \gg \lambda_D$

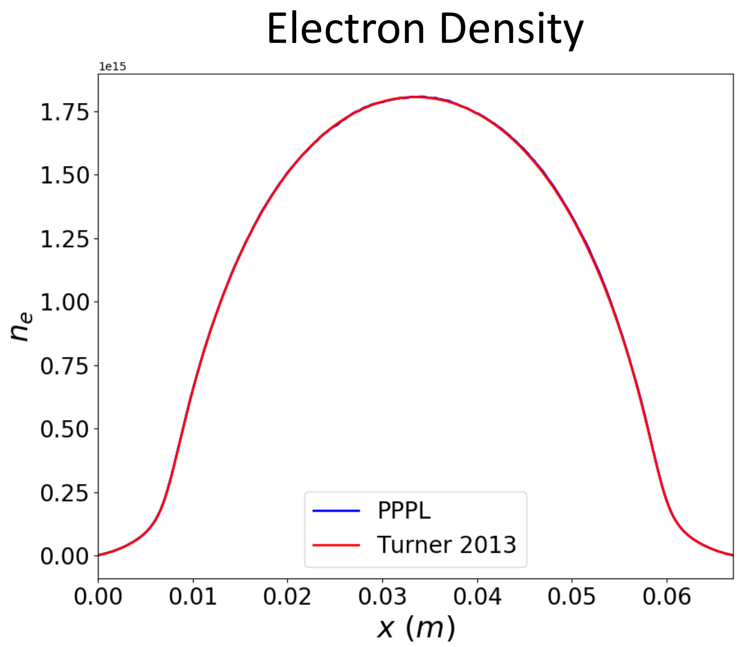


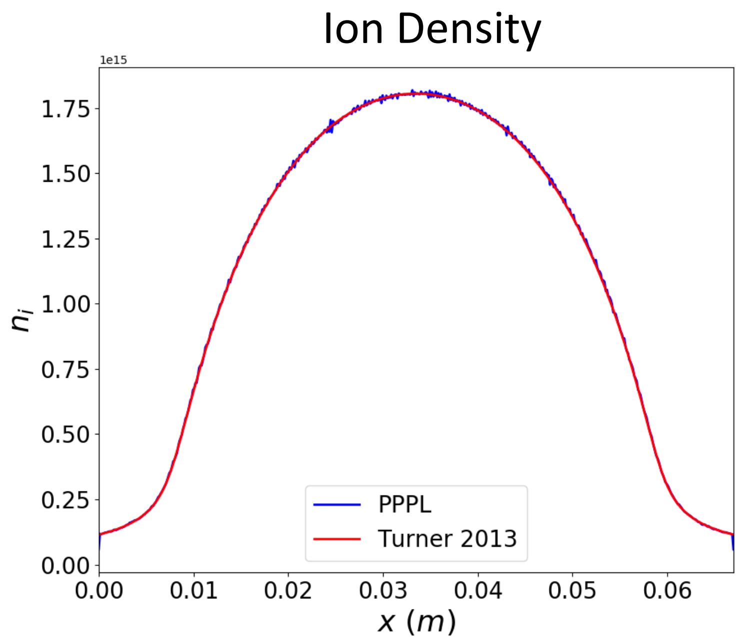
Stability regions for a warm beam moving through a background plasma with the energy-conserving PIC scheme [Barnes & Chacon 2021]

$u = U_{beam}/\omega_p \Delta x$

$M_\# = U_{beam}/v_{th,e}$

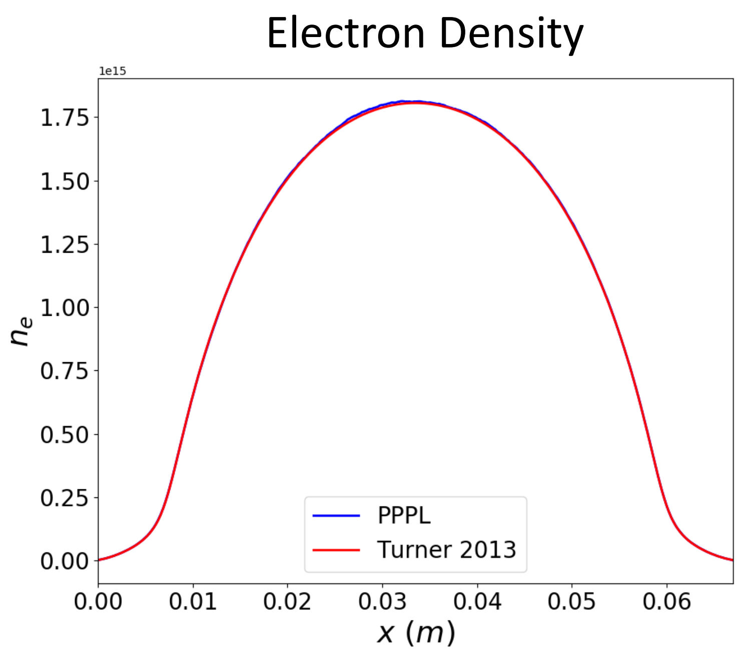
Benchmark Results with Standard PIC

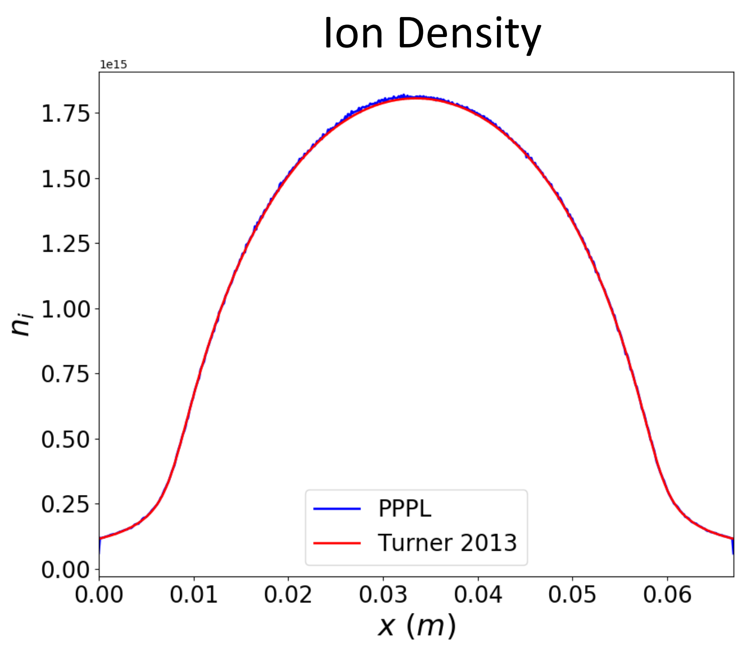




Consider the RF discharge benchmarking case from Turner *et al.* 2013. We begin by studying Case 2 from this paper with the standard electrostatic momentum conserving PIC algorithm

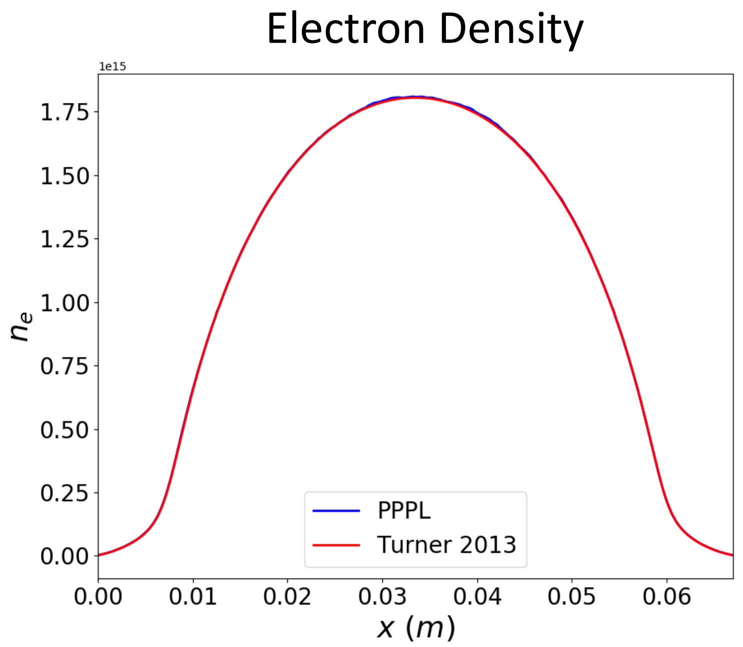
Results with Energy-Conserving PIC

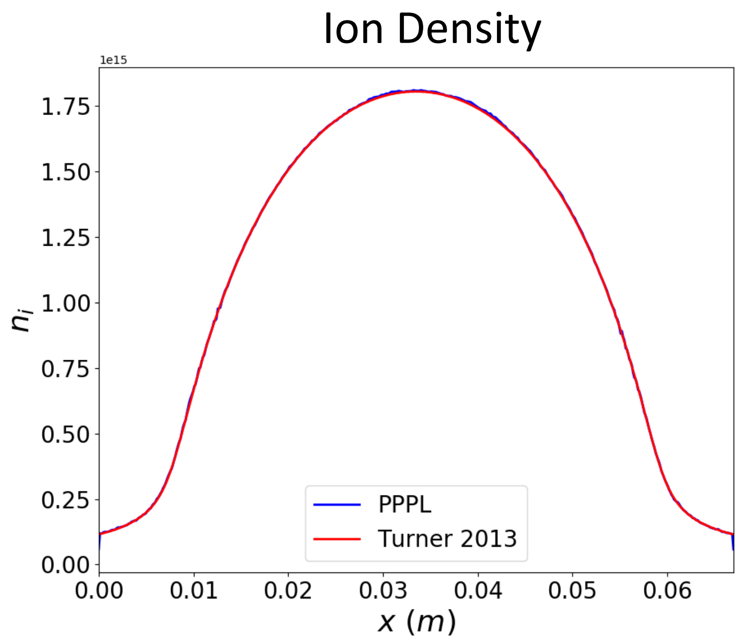




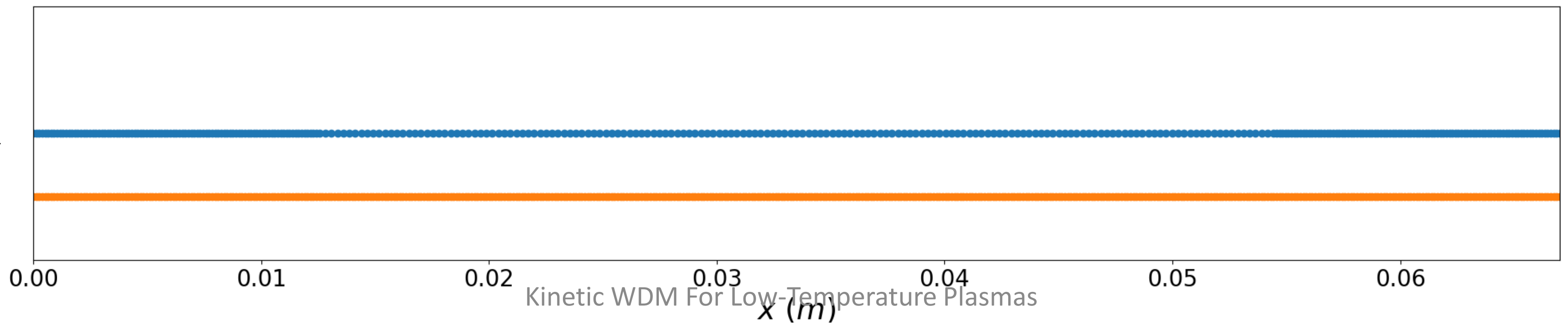
Now compare the results with energy-conserving PIC

Energy Conserving PIC & Non-Uniform Cells





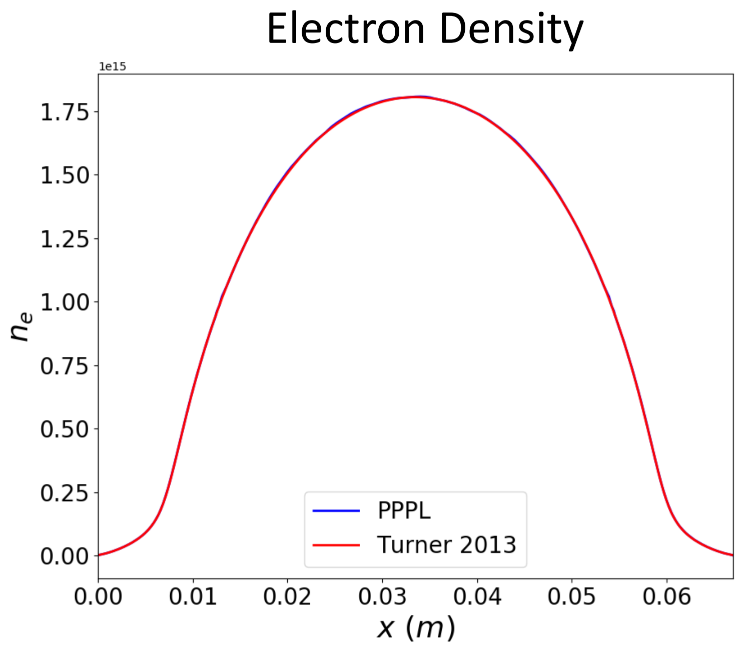
$$\Delta x_{max} = 2\Delta x_{min}$$

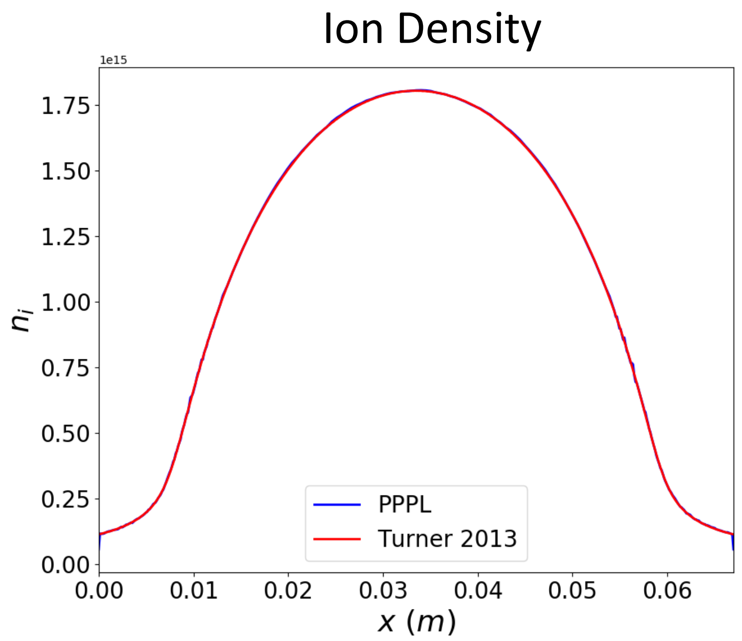


Non-uniform grid

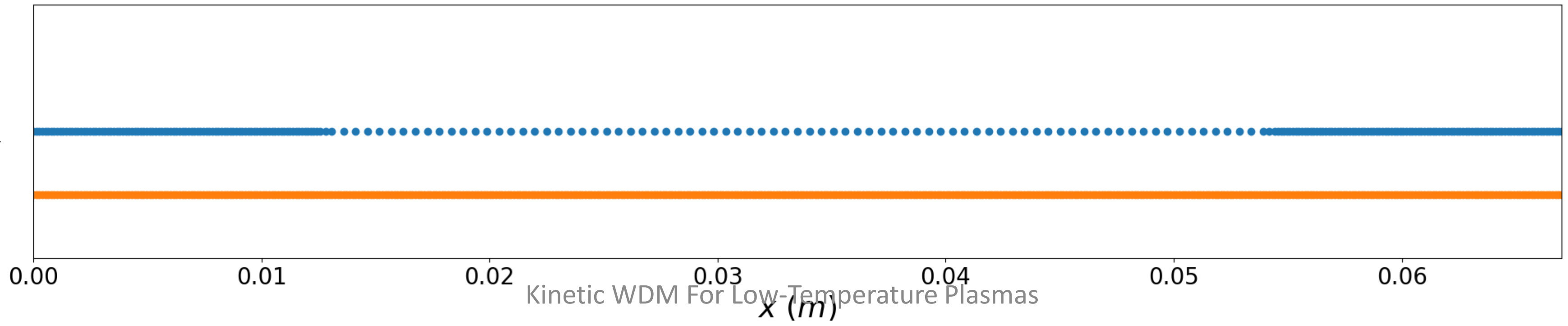
Uniform grid

Energy Conserving PIC & Non-Uniform Cells





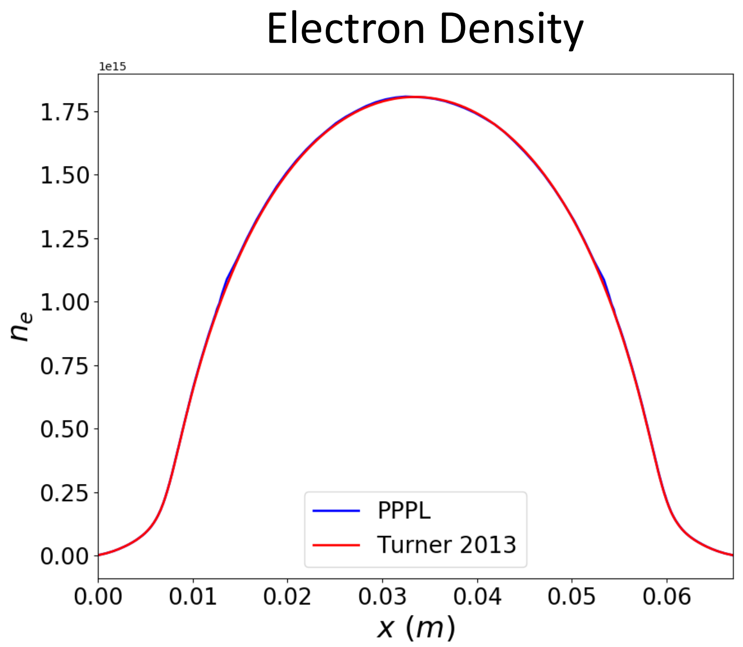
$$\Delta x_{max} = 4\Delta x_{min}$$

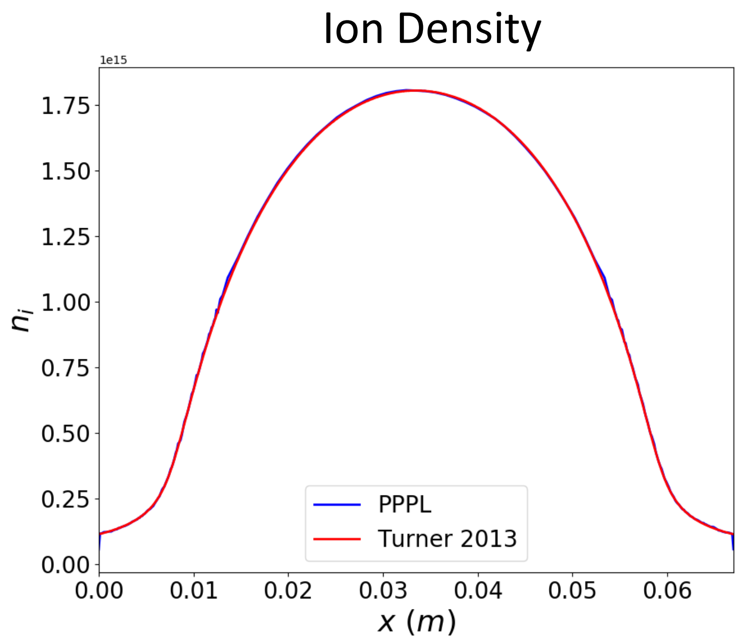


Non-uniform grid

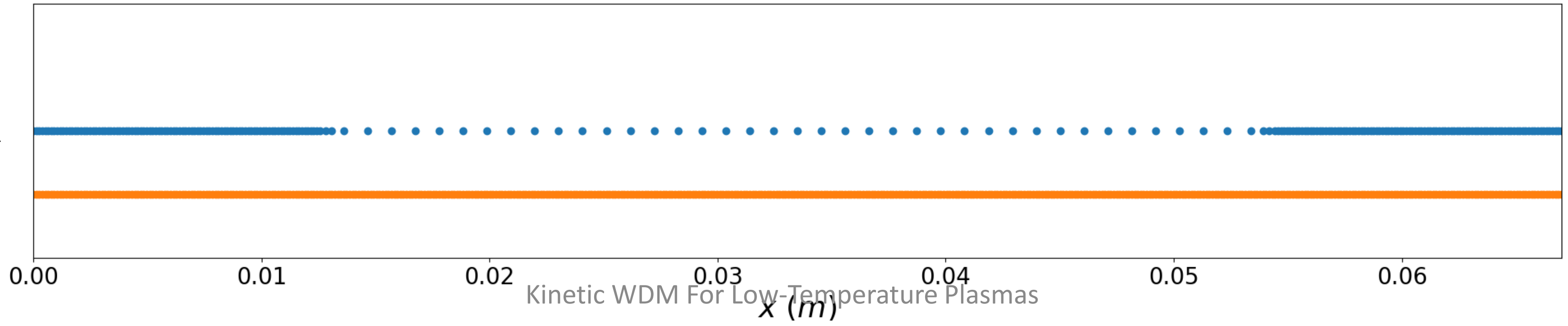
Uniform grid

Energy Conserving PIC & Non-Uniform Cells





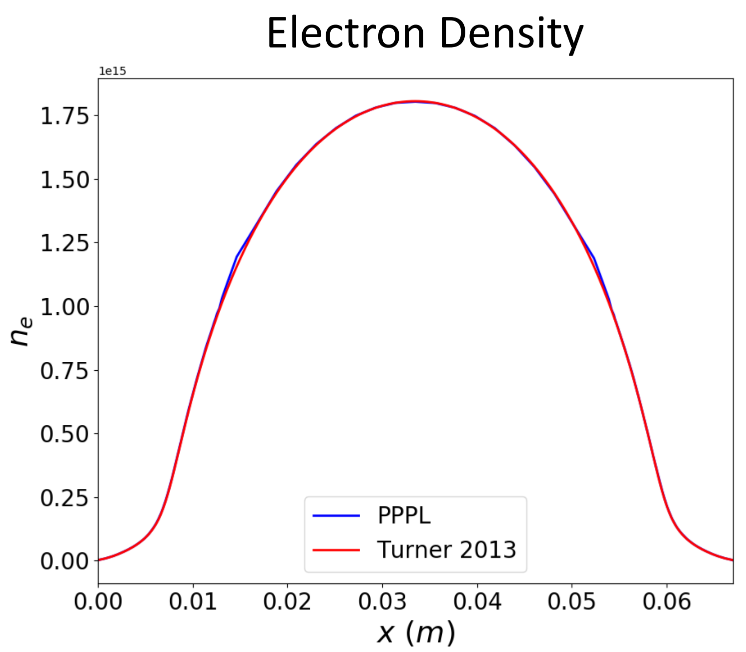
$$\Delta x_{max} = 8\Delta x_{min}$$

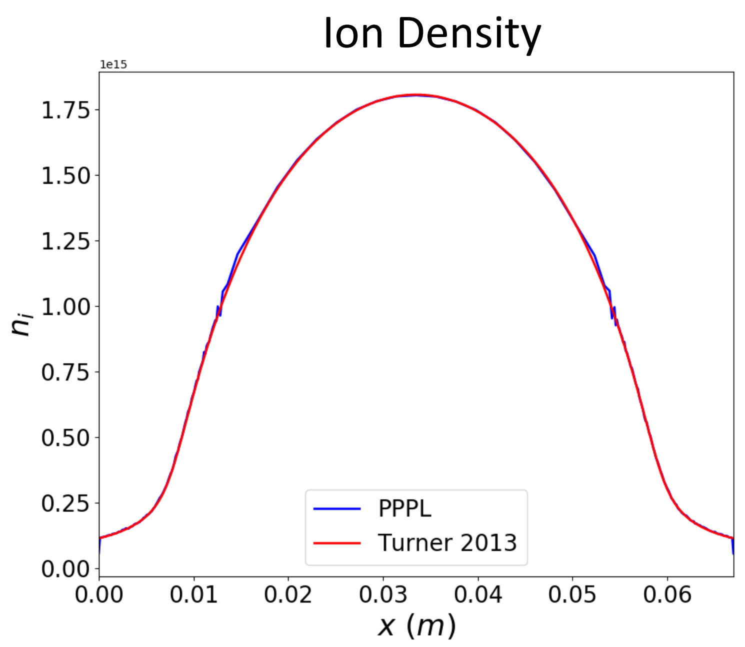


Non-uniform grid

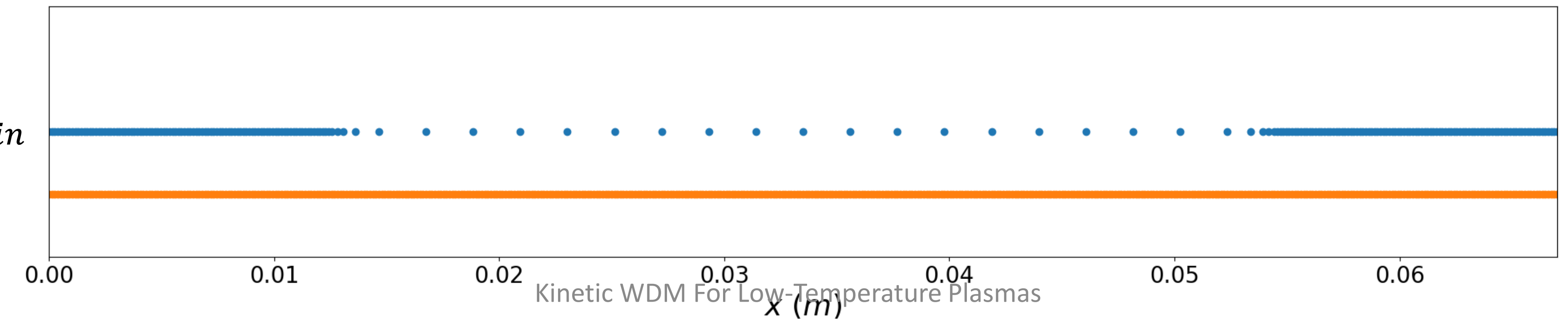
Uniform grid

Energy Conserving PIC & Non-Uniform Cells





$$\Delta x_{max} = 16\Delta x_{min}$$

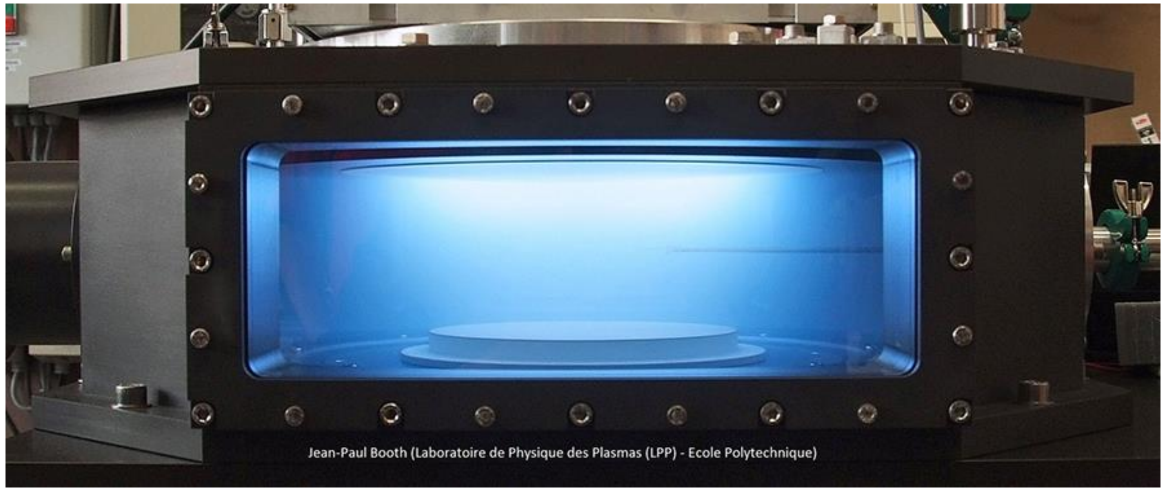


Non-uniform grid

Uniform grid

Requirements for Whole Device Modelling

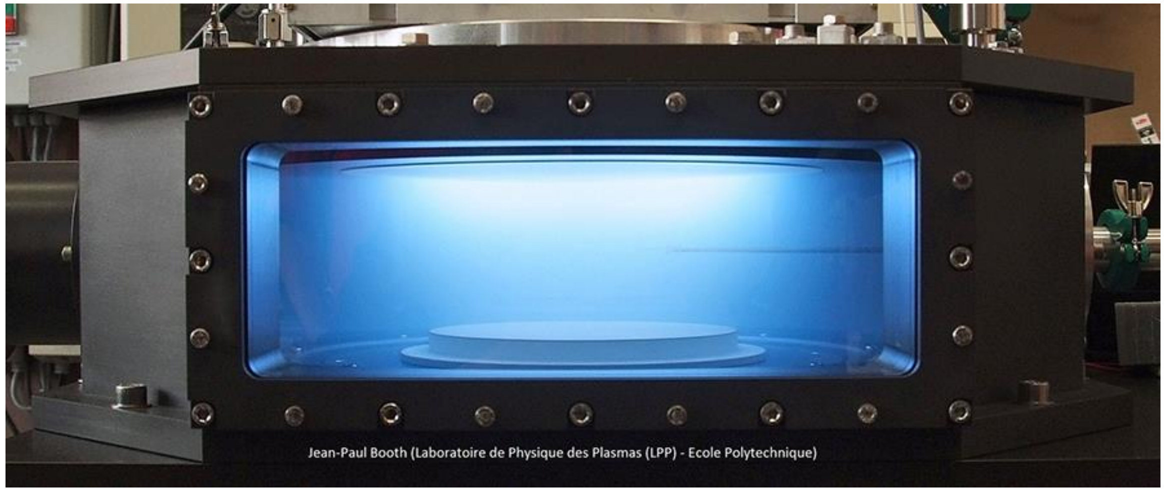
- Materials processing plasma reactor
- Assume a $10^{17}\ 1/m^3$ density Argon plasma with $10\ eV$ electrons
- Length scale $\sim 1\ m$
- Time scale $\sim 1\ ms$
- Based on PIC requirements we need:
    - 10^8 time steps
    - 10^{12} cells
    - 10^{15} particles
- Let's assume we can take $\Delta x \approx 100\lambda_D$


Jean-Paul Booth (Laboratoire de Physique des Plasmas (LPP) - Ecole Polytechnique)

Materials processing plasma reactor [Booth, Ecole Polytechnique]

Requirements for Whole Device Modelling

- Materials processing plasma reactor
- Assume a $10^{17}\ 1/m^3$ density Argon plasma with $10\ eV$ electrons
- Length scale $\sim 1\ m$
- Time scale $\sim 1\ ms$
- Based on PIC requirements we need:
  - 10^8 time steps
  - 10^{12} cells -> 10^6 cells
  - 10^{15} particles -> 10^9 cells
- If we give each node 32^3 cells with $1{,}000$ particle-per-cell we need only need 30 compute nodes


Jean-Paul Booth (Laboratoire de Physique des Plasmas (LPP) - Ecole Polytechnique)

Materials processing plasma reactor [Booth, Ecole Polytechnique]

Conclusions

Practical whole-device modelling of low-temperature plasma devices is not yet achievable. To get there we need:

- To fully leverage modern computing systems
- Apply new algorithms to reduce the immense computational cost

Things I didn't get to (but am happy to discuss)

- Geometric PIC methods (combining energy conservation and sub-cycled Maxwell solvers)
- Memory intensity vs arithmetic intensity
- Comparing PIC and Vlasov solvers

Backup Slides

Geometric PIC Methods

Energy conserving PIC methods can be derived directly from the underlying differential geometric structure [Xiao *et al.* 2015, Glasser & Qin, 2020]

Formally, the Vlasov-Maxwell system can be described by Poisson bracket:

$$\{\{F,G\}\}^{\mathrm{red}}[\bar{f},\boldsymbol{B},\boldsymbol{E}] = \int \mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{v}\left[\bar{f}\left\{\frac{\delta F}{\delta \bar{f}},\frac{\delta G}{\delta \bar{f}}\right\}_{\boldsymbol{xv}} + \bar{f}\boldsymbol{B}\cdot\left(\frac{\partial}{\partial \boldsymbol{v}}\frac{\delta F}{\delta \bar{f}}\times\frac{\partial}{\partial \boldsymbol{v}}\frac{\delta G}{\delta \bar{f}}\right) + \left(\frac{\delta F}{\delta \boldsymbol{E}}\cdot\frac{\partial \bar{f}}{\partial \boldsymbol{v}}\frac{\delta G}{\delta \bar{f}} - \frac{\delta G}{\delta \boldsymbol{E}}\cdot\frac{\partial \bar{f}}{\partial \boldsymbol{v}}\frac{\delta F}{\delta \bar{f}}\right)\right] + \int \mathrm{d}\boldsymbol{x}\left(\frac{\delta F}{\delta \boldsymbol{E}}\cdot\nabla\times\frac{\delta G}{\delta \boldsymbol{B}} - \frac{\delta G}{\delta \boldsymbol{E}}\cdot\nabla\times\frac{\delta F}{\delta \boldsymbol{B}}\right).$$

With Hamiltonian:

$$H(f,\mathbf{E},\mathbf{B}) = \int \mathbf{v}\cdot\mathbf{v}\,f\,d\mathbf{x}d\mathbf{v} + \frac{1}{2}\int \mathbf{E}^2 d\mathbf{x} + \frac{1}{2}\int \mathbf{B}^2 d\mathbf{x}$$

Geometric PIC Methods

The dynamical equations are then recovered using:

$$\frac{\partial f}{\partial t} = \{f, H\} = -\mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{x}} - (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}}$$

$$\frac{\partial \mathbf{E}}{\partial t} = \{\mathbf{E}, H\} = -\int \mathbf{v} f d\mathbf{v} + \nabla \times \mathbf{B}$$

$$\frac{\partial \mathbf{B}}{\partial t} = \{\mathbf{B}, H\} = -\nabla \times \mathbf{E}$$

Geometric PIC Methods

We can discretise the Poisson bracket as follows:

- Use a Klimontovich particle distribution
- Discretise field quantities over a finite grid
- Interpolations via structure preserving Whitney forms
- Differential operators are chosen via Discrete Exterior Calculus

$$\{\{F,G\}\}^{\mathrm{red}}[\bar{f},\boldsymbol{B},\boldsymbol{E}] = \int \mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{v}\left[\bar{f}\left\{\frac{\delta F}{\delta \bar{f}},\frac{\delta G}{\delta \bar{f}}\right\}_{\boldsymbol{x}\boldsymbol{v}} + \bar{f}\boldsymbol{B}\cdot\left(\frac{\partial}{\partial \boldsymbol{v}}\frac{\delta F}{\delta \bar{f}}\times\frac{\partial}{\partial \boldsymbol{v}}\frac{\delta G}{\delta \bar{f}}\right) + \left(\frac{\delta F}{\delta \boldsymbol{E}}\cdot\frac{\partial \bar{f}}{\partial \boldsymbol{v}}\frac{\delta G}{\delta \bar{f}} - \frac{\delta G}{\delta \boldsymbol{E}}\cdot\frac{\partial \bar{f}}{\partial \boldsymbol{v}}\frac{\delta F}{\delta \bar{f}}\right)\right] + \int \mathrm{d}\boldsymbol{x}\left(\frac{\delta F}{\delta \boldsymbol{E}}\cdot\nabla\times\frac{\delta G}{\delta \boldsymbol{B}} - \frac{\delta G}{\delta \boldsymbol{E}}\cdot\nabla\times\frac{\delta F}{\delta \boldsymbol{B}}\right).$$

$$H(f,\mathbf{E},\mathbf{B}) = \int \mathbf{v}\cdot\mathbf{v}\,f\,d\mathbf{x}d\mathbf{v} + \frac{1}{2}\int \mathbf{E}^2 d\mathbf{x} + \frac{1}{2}\int \mathbf{B}^2 d\mathbf{x}$$

Geometric PIC Methods

We can discretise the Poisson bracket as follows:

- Use a Klimontovich particle distribution
- Discretise field quantities over a finite grid
- Interpolations via structure preserving Whitney forms
- Differential operators are chosen via Discrete Exterior Calculus

$$
\begin{aligned}
\{\{F, G\}\}_d^{\mathrm{red}}[\boldsymbol{X}_i, \boldsymbol{V}_i, \boldsymbol{B}_n, \boldsymbol{E}_n] \\
= \sum_{i=1}^{L} \Bigg(\frac{\partial F}{\partial \boldsymbol{X}_i} \cdot \frac{\partial G}{\partial \boldsymbol{V}_i} - \frac{\partial G}{\partial \boldsymbol{X}_i} \cdot \frac{\partial F}{\partial \boldsymbol{V}_i} + \left[\frac{\partial F}{\partial \boldsymbol{V}_i} \times \frac{\partial G}{\partial \boldsymbol{V}_i} \right] \cdot \sum_{n=1}^{N} \boldsymbol{B}_n W_{\sigma_2}(\boldsymbol{X}_i - \boldsymbol{x}_n) \Bigg) \\
+ \sum_{n=1}^{N} \Bigg(\left[\sum_{i=1}^{L} \frac{\partial F}{\partial \boldsymbol{V}_i} W_{\sigma_1}(\boldsymbol{X}_i - \boldsymbol{x}_n) - \left(\nabla_d^{-} \times \frac{\partial F}{\partial \boldsymbol{B}} \right)_n \right] \cdot \frac{\partial G}{\partial \boldsymbol{E}_n} \\
- \left[\sum_{i=1}^{L} \frac{\partial G}{\partial \boldsymbol{V}_i} W_{\sigma_1}(\boldsymbol{X}_i - \boldsymbol{x}_n) - \left(\nabla_d^{-} \times \frac{\partial G}{\partial \boldsymbol{B}} \right)_n \right] \cdot \frac{\partial F}{\partial \boldsymbol{E}_n} \Bigg).
\end{aligned}
$$

$$
H_d^{\mathrm{red}}[\boldsymbol{X}_i, \boldsymbol{V}_i, \boldsymbol{B}_n, \boldsymbol{E}_n] = \frac{1}{2} \sum_{i=1}^{L} \boldsymbol{V}_i^2 + \frac{1}{2} \sum_{n=1}^{N} (\boldsymbol{E}_n^2 + \boldsymbol{B}_n^2) = \sum_{\alpha=1}^{3} H_V^{\alpha} + H_B + H_E
$$

Geometric PIC Methods

Lie splitting of the Hamiltonian leads to the following sets of semi-discrete dynamical equations

$$H_d^{\mathrm{red}}[\boldsymbol{X}_i, \boldsymbol{V}_i, \boldsymbol{B}_n, \boldsymbol{E}_n] = \frac{1}{2}\sum_{i=1}^{L} \boldsymbol{V}_i^2 + \frac{1}{2}\sum_{n=1}^{N}(\boldsymbol{E}_n^2 + \boldsymbol{B}_n^2) = \sum_{\alpha=1}^{3} H_{\boldsymbol{V}}^{\alpha} + H_{\boldsymbol{B}} + H_{\boldsymbol{E}}$$

$$H_{\boldsymbol{V}}^{\alpha}\begin{cases} \dot{X}_i^{\beta} = \delta_{\alpha}^{\beta} V_i^{\alpha}, \\ \dot{V}_i^{\beta} = \epsilon_{\beta\alpha\gamma} V_i^{\alpha} \sum_{n=1}^{N} B_n^{\gamma} W_{\sigma_2}(\boldsymbol{X}_i - \boldsymbol{x}_n), \\ \dot{B}_n^{\beta} = 0, \\ \dot{E}_n^{\beta} = -\delta_{\alpha}^{\beta} \sum_{i=1}^{L} V_i^{\alpha} W_{\sigma_1}(\boldsymbol{X}_i - \boldsymbol{x}_n), \end{cases}$$

$$H_{\boldsymbol{B}}\begin{cases} \dot{\boldsymbol{X}}_i = 0, \\ \dot{\boldsymbol{V}}_i = 0, \\ \dot{\boldsymbol{B}}_n = 0, \\ \dot{\boldsymbol{E}}_n = (\nabla_d^{-} \times \boldsymbol{B})_n, \end{cases}$$

$$H_{\boldsymbol{E}}\begin{cases} \dot{\boldsymbol{X}}_i = 0, \\ \dot{\boldsymbol{V}}_i = \sum_{n=1}^{N} \boldsymbol{E}_n W_{\sigma_1}(\boldsymbol{X}_i - \boldsymbol{x}_n), \\ \dot{\boldsymbol{B}}_n = -(\nabla_d^{+} \times \boldsymbol{E})_n, \\ \dot{\boldsymbol{E}}_n = 0. \end{cases}$$

Geometric PIC Methods

These equations can be integrated exactly, providing phase space maps

$$\Theta_x(\Delta t) = \begin{cases} X_i^\beta(t+\Delta t) & = X_i^\beta(t) + \Delta t \delta_\alpha^\beta V_i^\alpha(t) \\ V_i^\beta(t+\Delta t) & = V_i^\beta(t) + \epsilon_{\beta\alpha\gamma} V_i^\alpha \int_0^{\Delta t} \sum_n W_{\sigma 2} B_n^\gamma(t)\left(\mathbf{x}_n - \mathbf{X}_i(s)\right) ds \\ B_n^\beta(t+\Delta t) & = B_n^\beta(t) \\ E_n^\beta(t+\Delta t) & = E_n^\beta(t) - \delta_\alpha^\beta \int_0^{\Delta t} \sum_i V_i^\alpha W_{\sigma 1}\left(\mathbf{x}_n - \mathbf{X}_i(s)\right) ds \end{cases}$$

$$\Theta_B(\Delta t) = \begin{cases} \mathbf{X}_i(t+\Delta t) & = X_i(t) \\ \mathbf{V}_i(t+\Delta t) & = V_i(t) \\ \mathbf{B}_n(t+\Delta t) & = B_n(t) \\ \mathbf{E}_n(t+\Delta t) & = E_n(t) + \left(\nabla_d^- \times \mathbf{B}\right)_n \end{cases}$$

$$\Theta_E(\Delta t) = \begin{cases} \mathbf{X}_i(t+\Delta t) & = X_i(t) \\ \mathbf{V}_i(t+\Delta t) & = V_i(t) + \sum_n \mathbf{E}_n W_{\sigma 1}\left(\mathbf{x}_n - \mathbf{X}_i(t)\right) \\ \mathbf{B}_n(t+\Delta t) & = B_n(t) - \left(\nabla_d^- \times \mathbf{E}\right)_n \\ \mathbf{E}_n(t+\Delta t) & = E_n(t) \end{cases}$$

Geometric PIC Methods

We construct numerical methods by combining these maps:

- First order scheme:

$$\Theta_1(\Delta t) = \Theta_E(\Delta t)\,\Theta_B(\Delta t)\,\Theta_z(\Delta t)\,\Theta_y(\Delta t)\,\Theta_x(\Delta t)$$

- Second order scheme:

$$\Theta_1(\Delta t) = \Theta_x(\Delta t/2)\,\Theta_y(\Delta t/2)\,\Theta_z(\Delta t/2)\,\Theta_B(\Delta t/2)\,\Theta_E(\Delta t)\,\Theta_B(\Delta t/2)\,\Theta_z(\Delta t/2)\,\Theta_y(\Delta t/2)\,\Theta_x(\Delta t/2)$$

- Plus there are simple recipes for higher order schemes

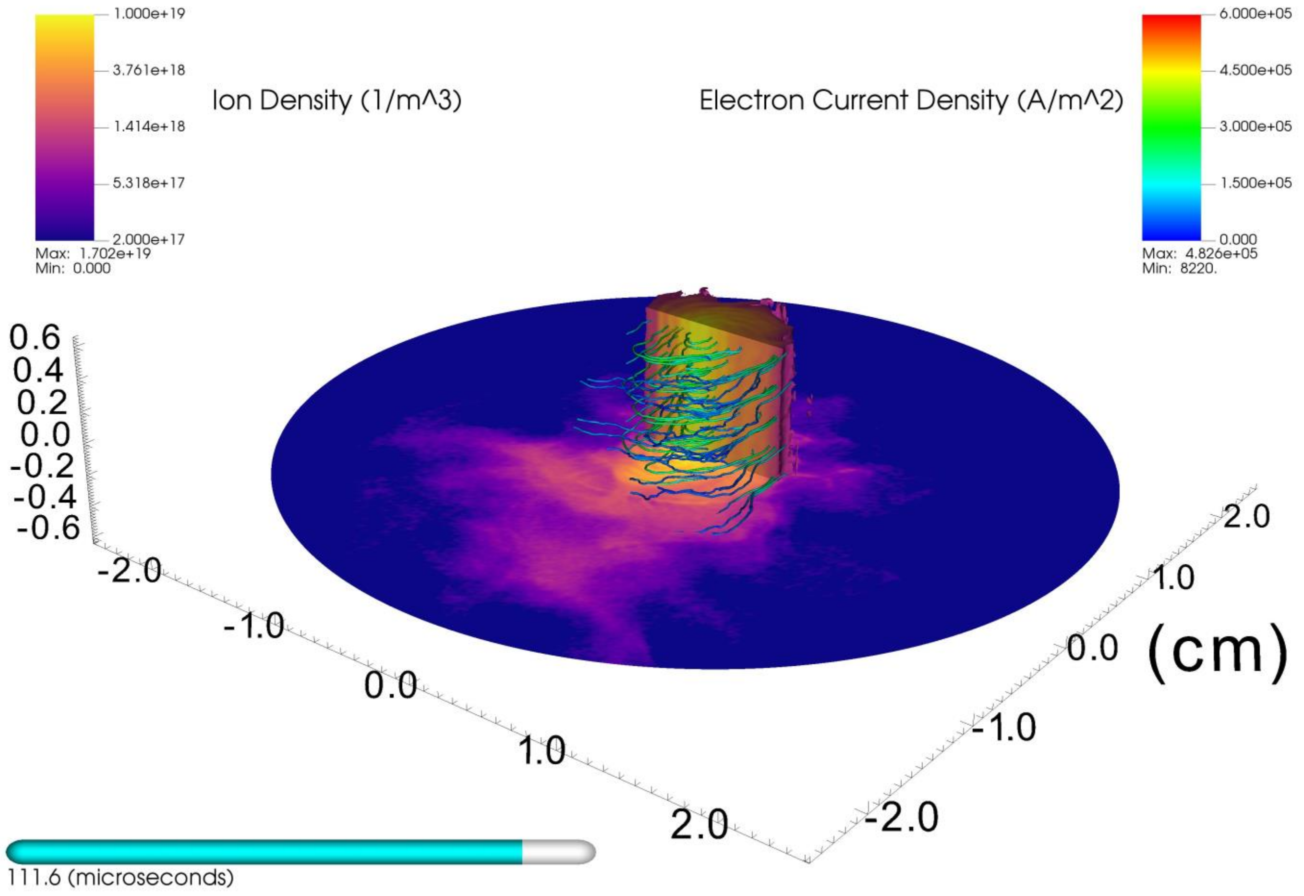
Geometric PIC Methods

Advantages of these methods:

- They conserve charge exactly
- They narrowly bound energy growth (no finite grid instability)
- There is a clear recipe for building the schemes (no ad-hoc selection of numerical methods)
- The algorithms are explicit (great for GPUs)

Disadvantages of these methods:

- Not momentum conserving (although fairly well bounded)
- Still limited by the standard explicit algorithm constraints on time steps



3D collisionless simulation of the Penning discharge with density contours and electron current streamlines