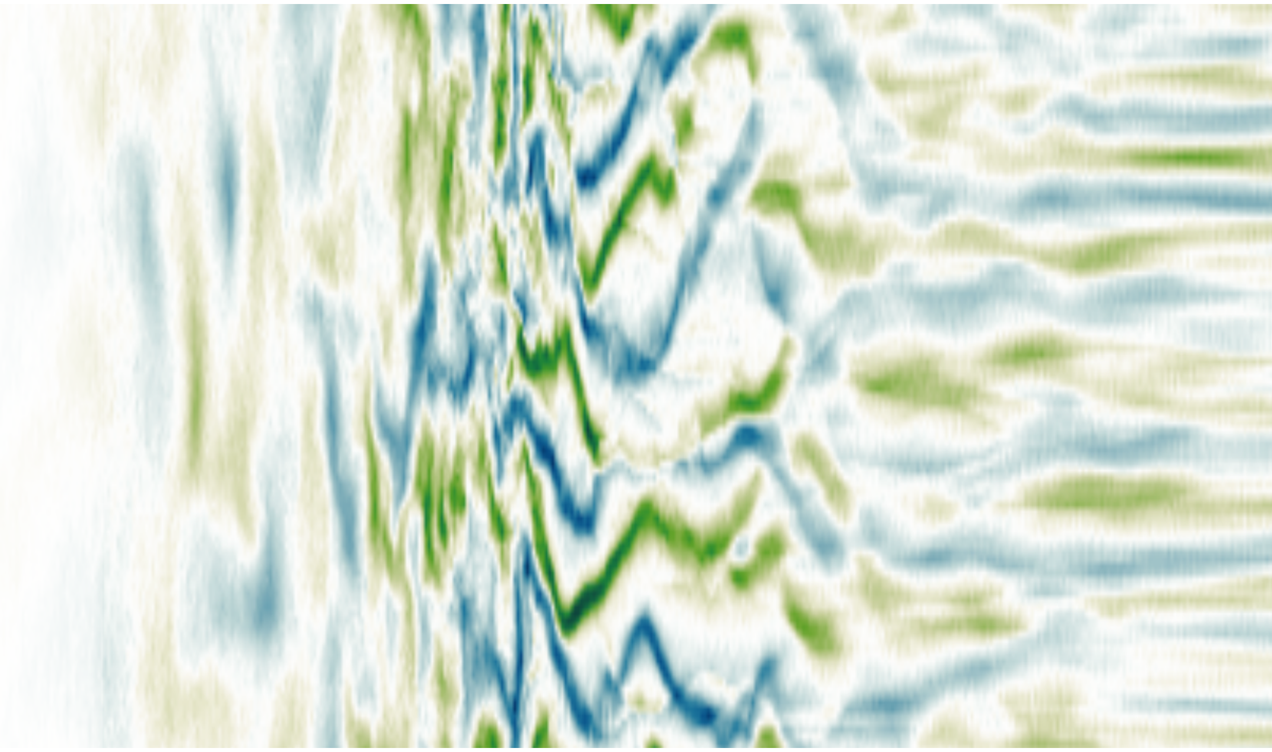
The Particle-In-Cell (PIC) simulation of plasmas

Mickael Grech, LULI, CNRS
mickael.grech@gmail.com

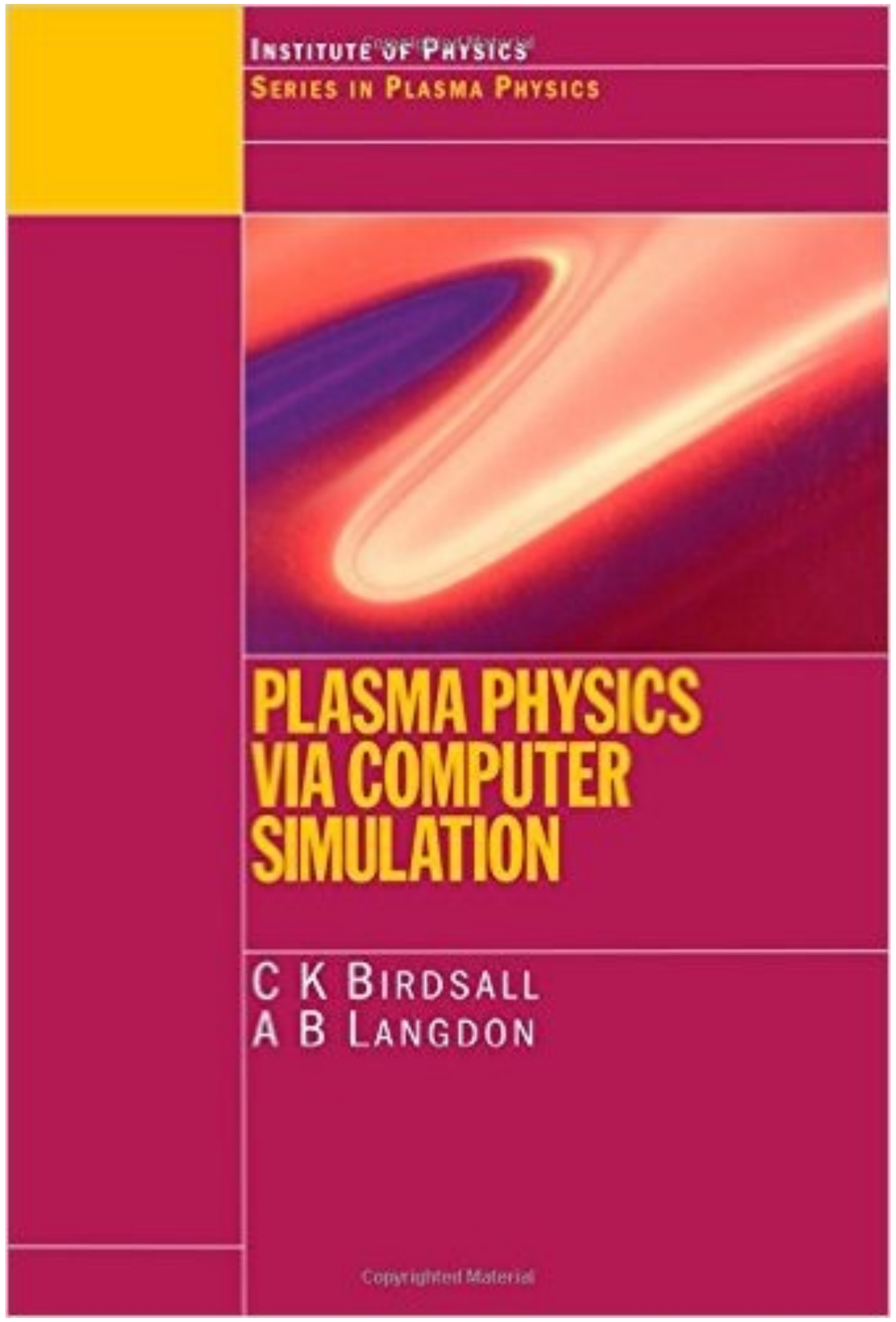
Les Houches, Mai 2019



References

References

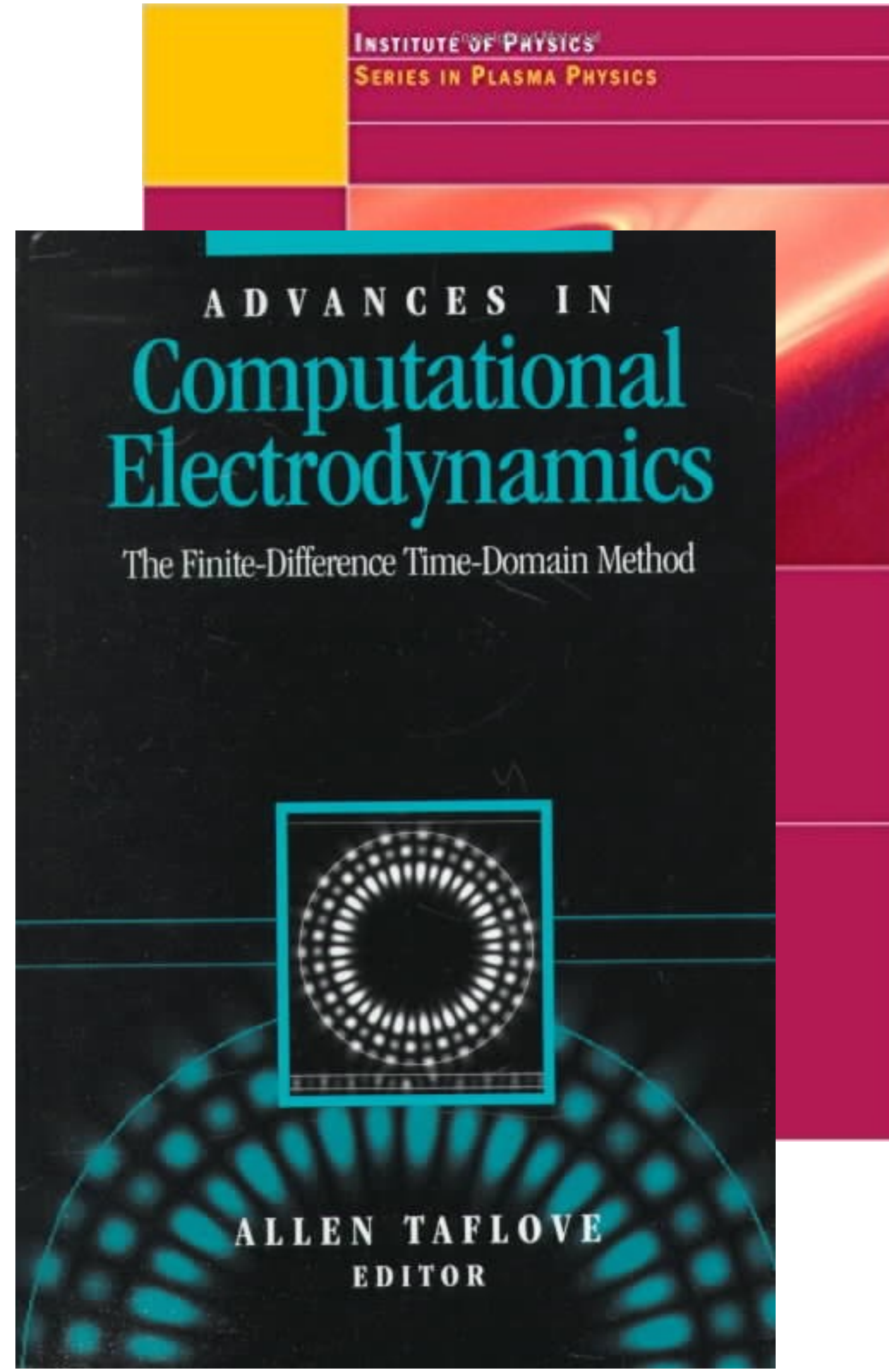
Plasma Physics via Computer Simulation
C. K. Birdsall & A. B. Langdon



References

Plasma Physics via Computer Simulation
C. K. Birdsall & A. B. Langdon

Computational Electrodynamics
A. Taflove

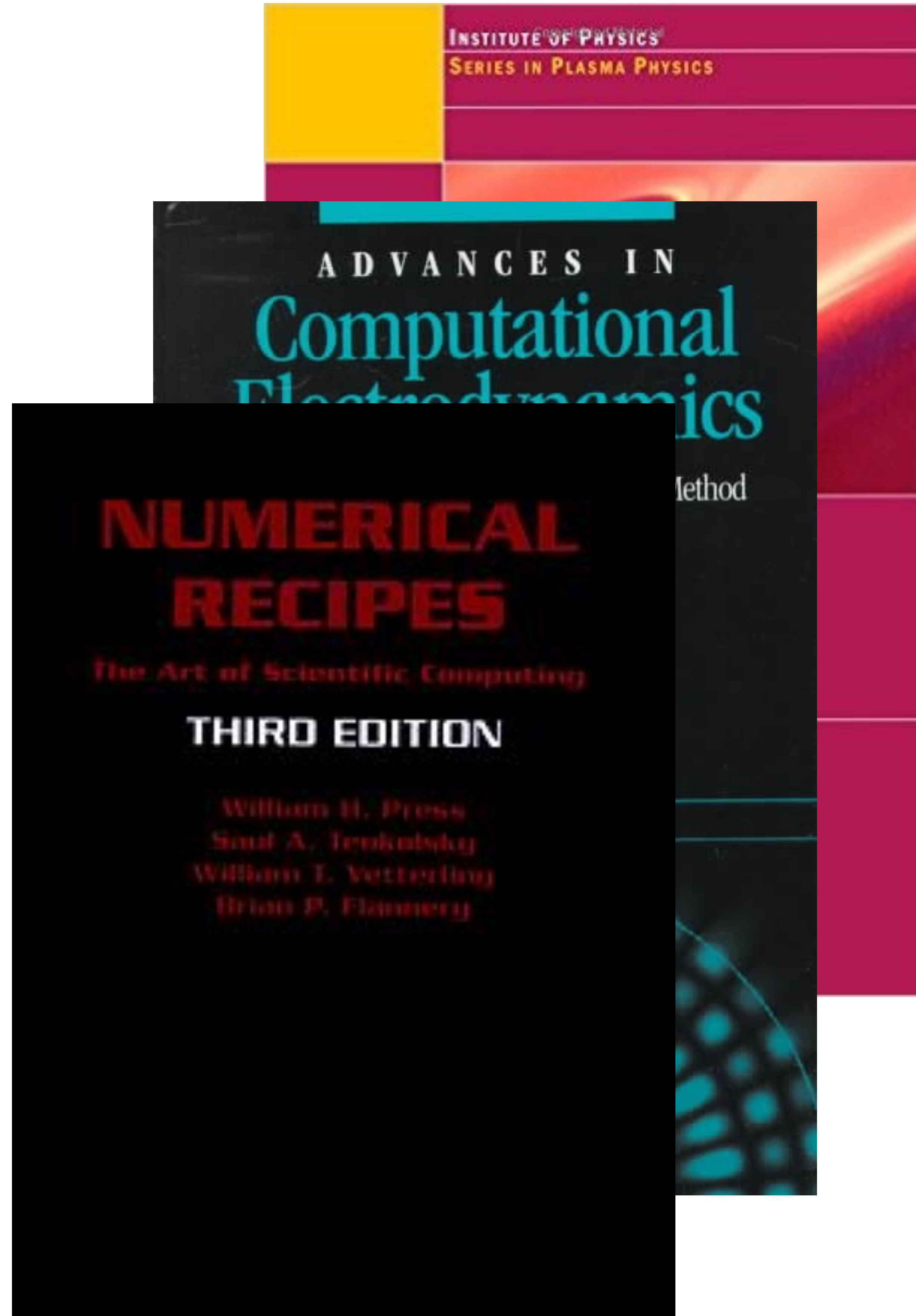


References

Plasma Physics via Computer Simulation
C. K. Birdsall & A. B. Langdon

Computational Electrodynamics
A. Taflove

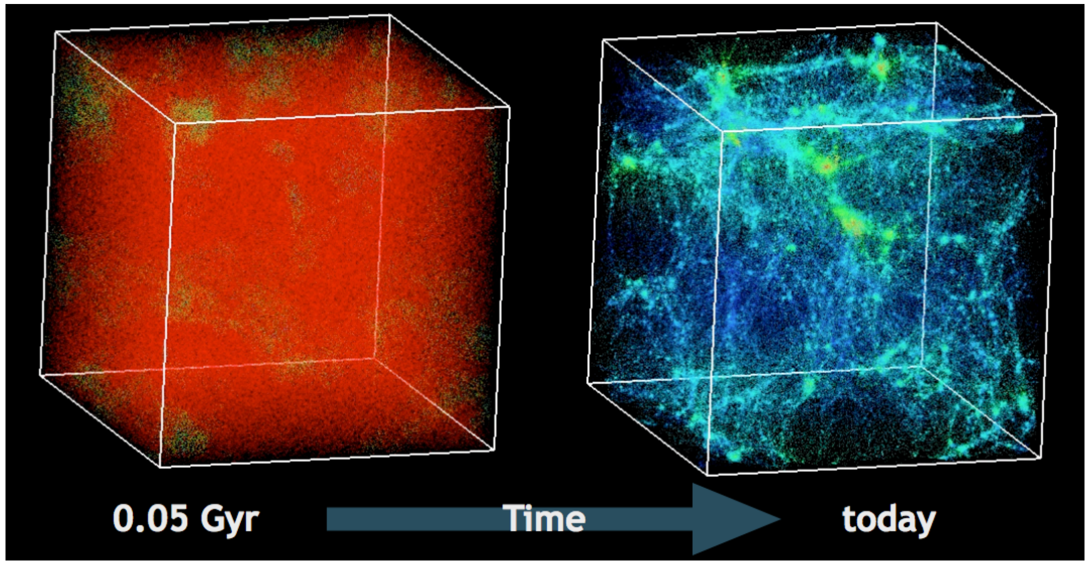
Numerical Recipies
W. H. Press *et al.*



The Particle-In-Cell (PIC) method is a central tool for simulation over a wide range of physics studies

The Particle-In-Cell (PIC) method is a central tool for simulation over a wide range of physics studies

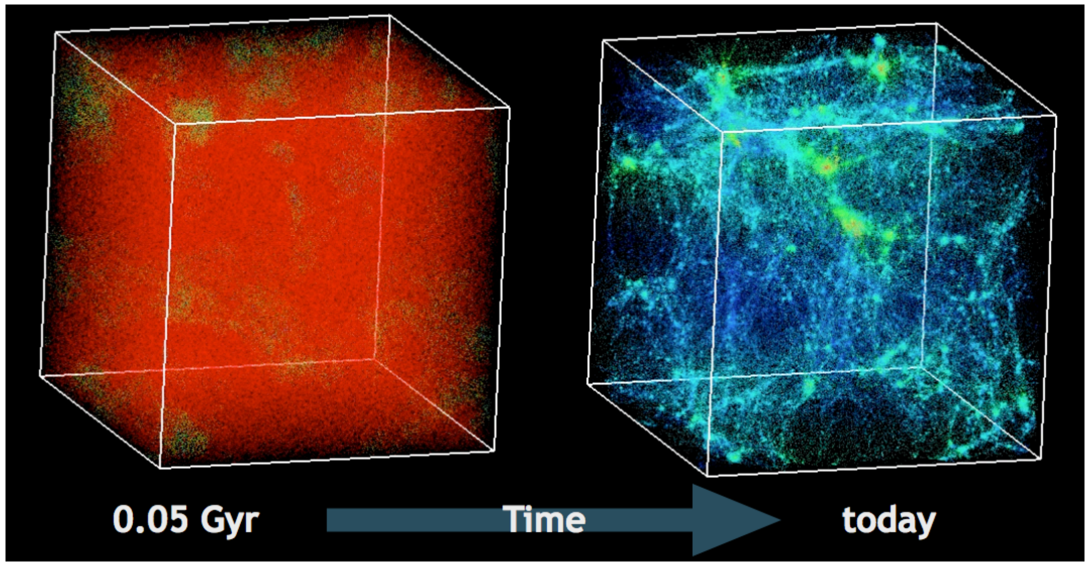
Cosmology



source: K. Heitmann, Argonne National Lab

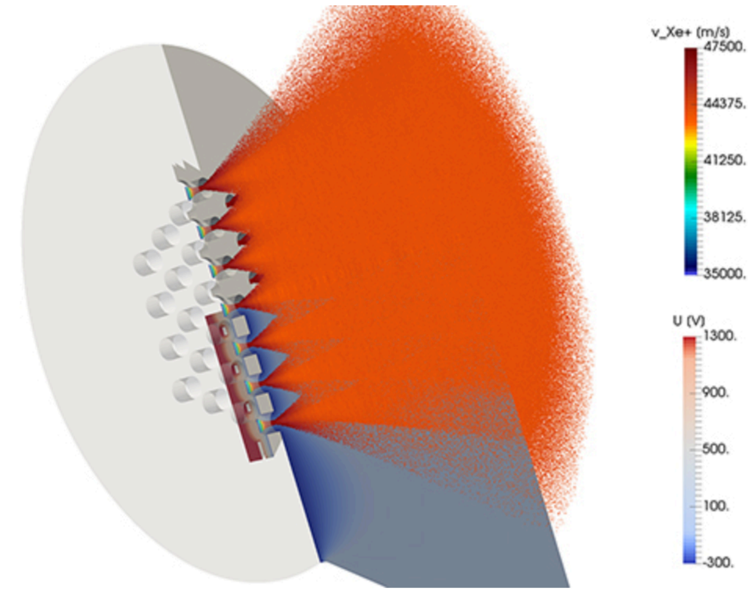
The Particle-In-Cell (PIC) method is a central tool for simulation over a wide range of physics studies

Cosmology



source: K. Heitmann, Argonne National Lab

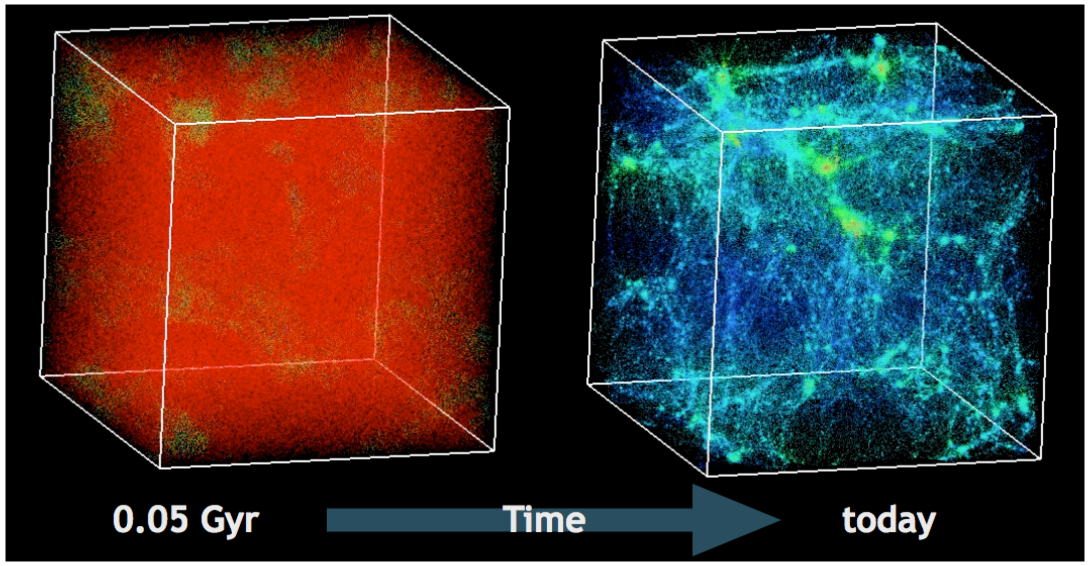
Space propulsion (Plasma thruster)



source: Gauss Center for Supercomputing

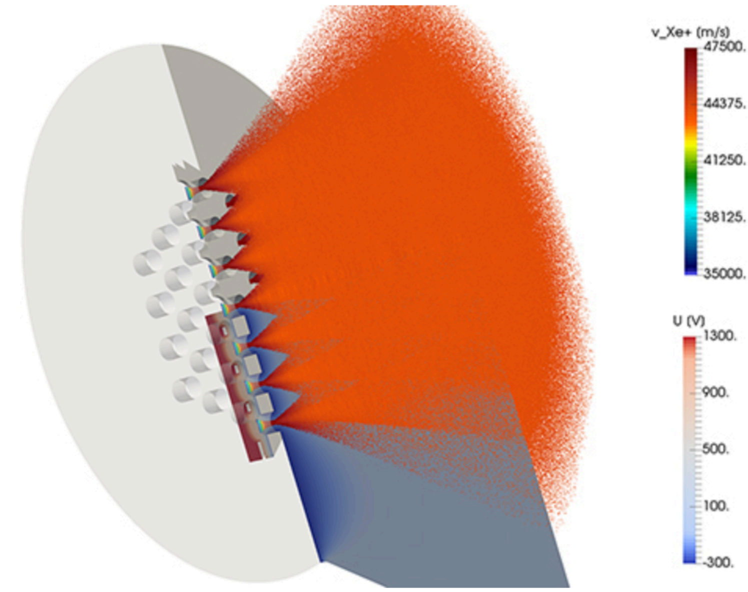
The Particle-In-Cell (PIC) method is a central tool for simulation over a wide range of physics studies

Cosmology



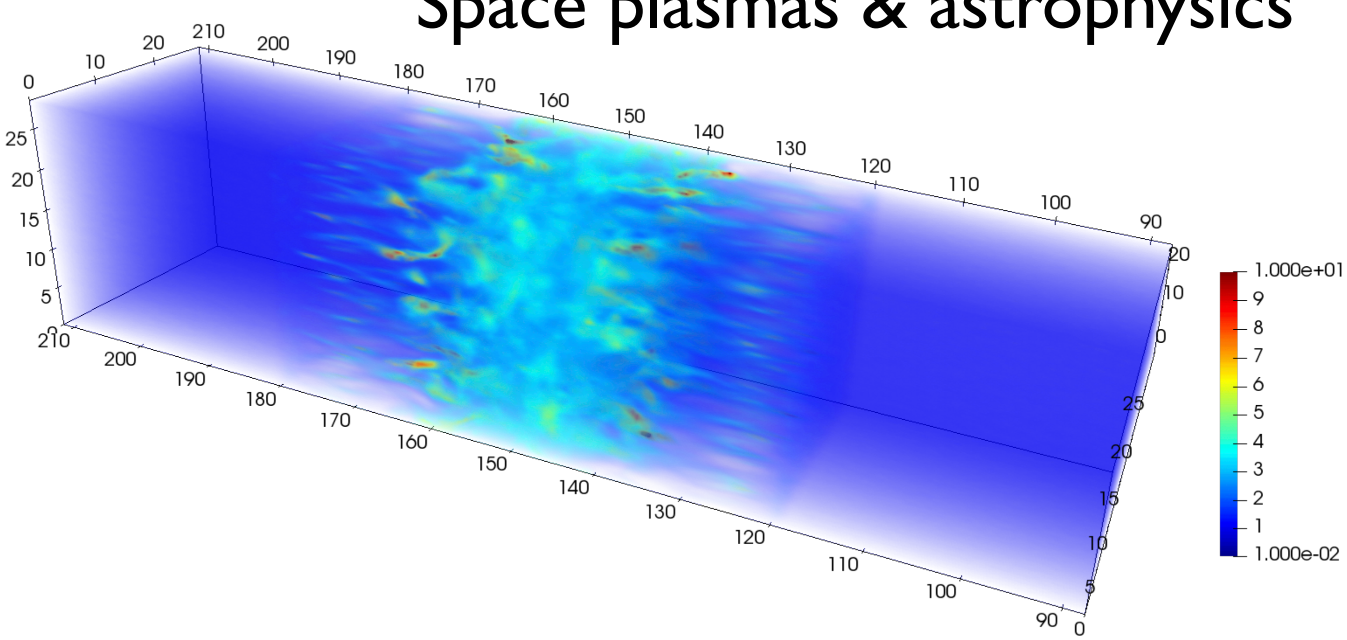
source: K. Heitmann, Argonne National Lab

Space propulsion (Plasma thruster)



source: Gauss Center for Supercomputing

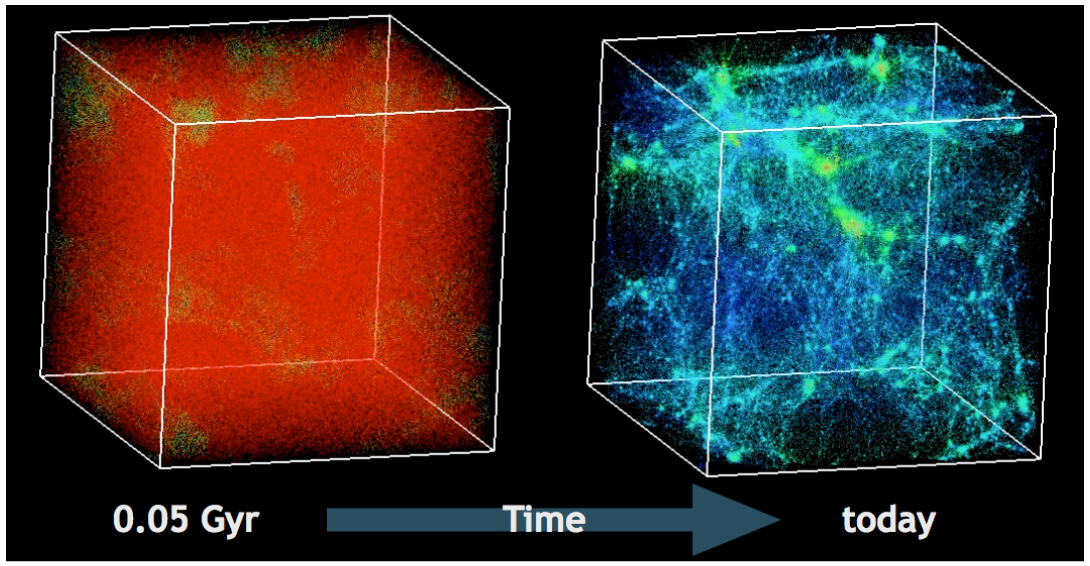
Space plasmas & astrophysics



source: SMILEI dev-team

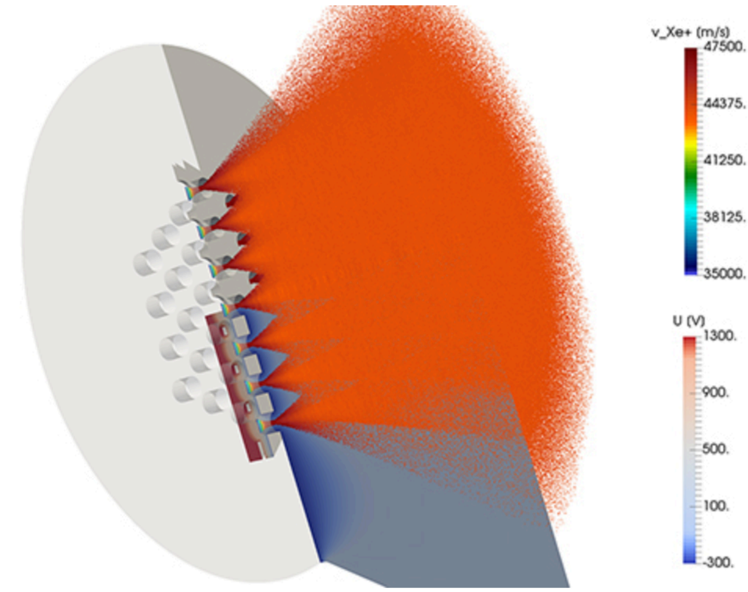
The Particle-In-Cell (PIC) method is a central tool for simulation over a wide range of physics studies

Cosmology



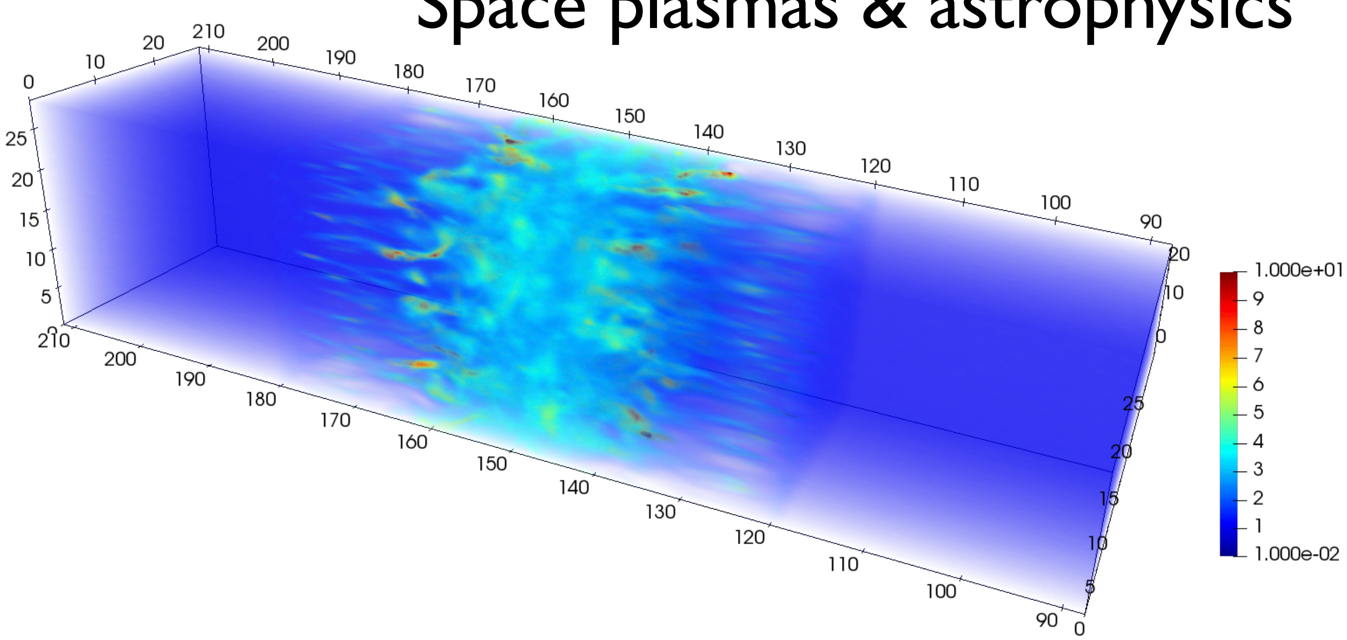
source: K. Heitmann, Argonne National Lab

Space propulsion (Plasma thruster)



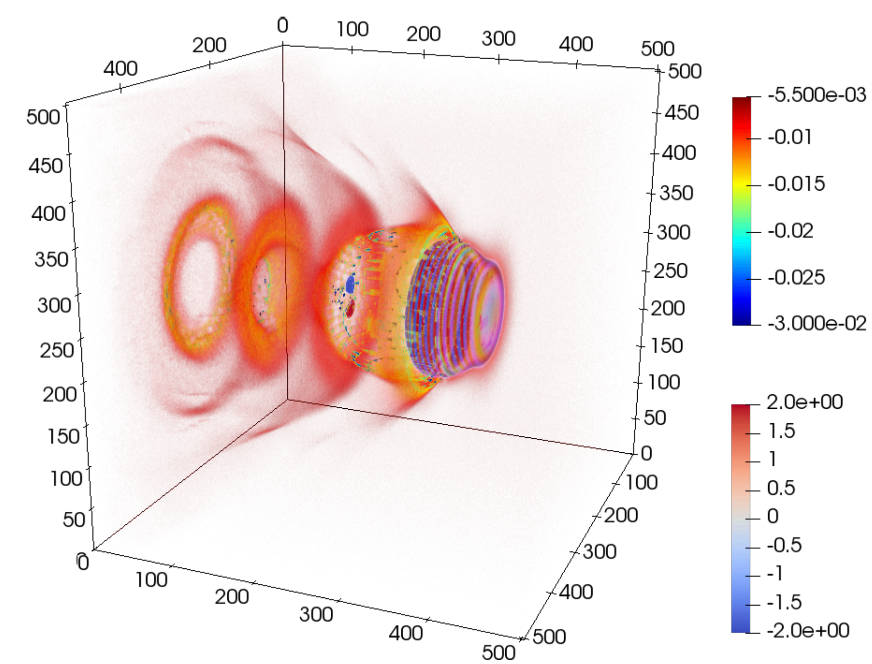
source: Gauss Center for Supercomputing

Space plasmas & astrophysics



source: SMILEI dev-team

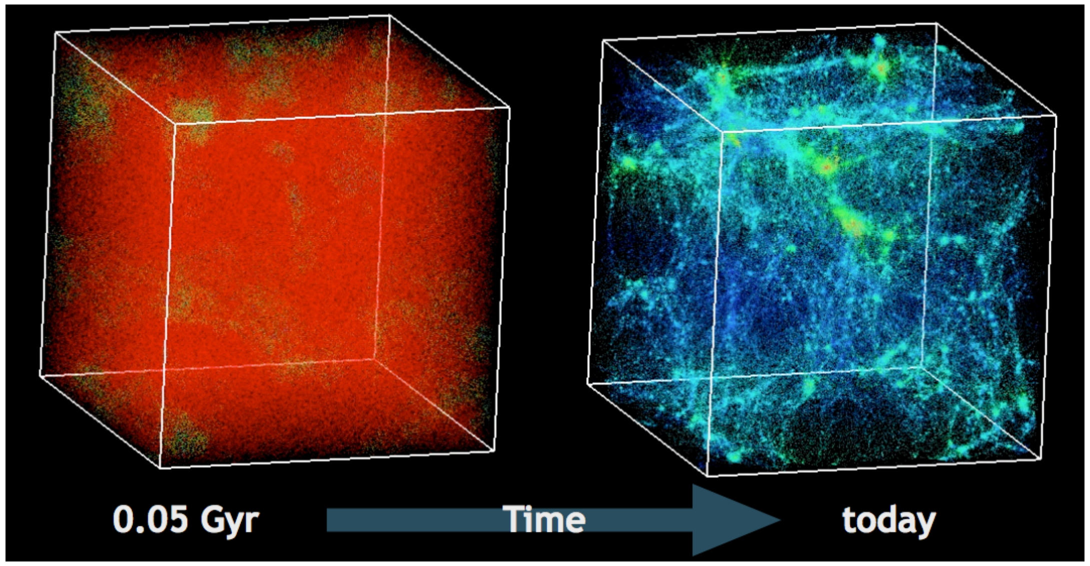
Laser plasma interaction



source: SMILEI dev-team

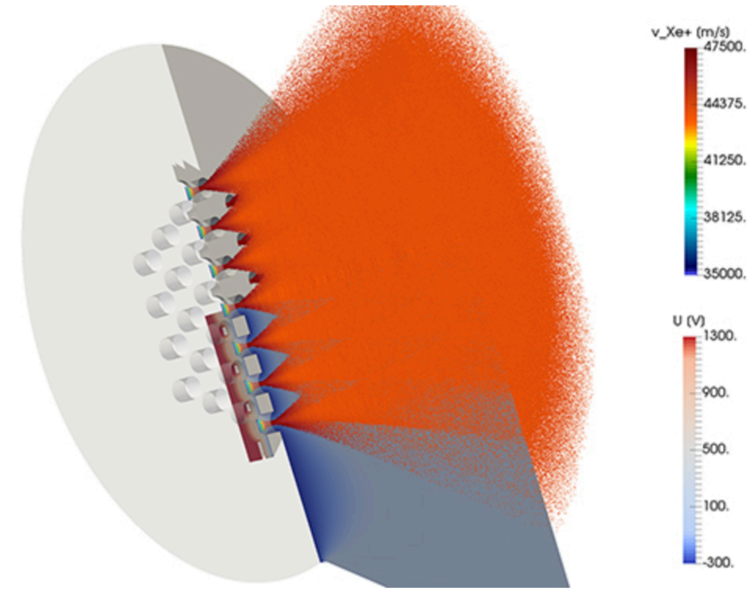
The Particle-In-Cell (PIC) method is a central tool for simulation over a wide range of physics studies

Cosmology



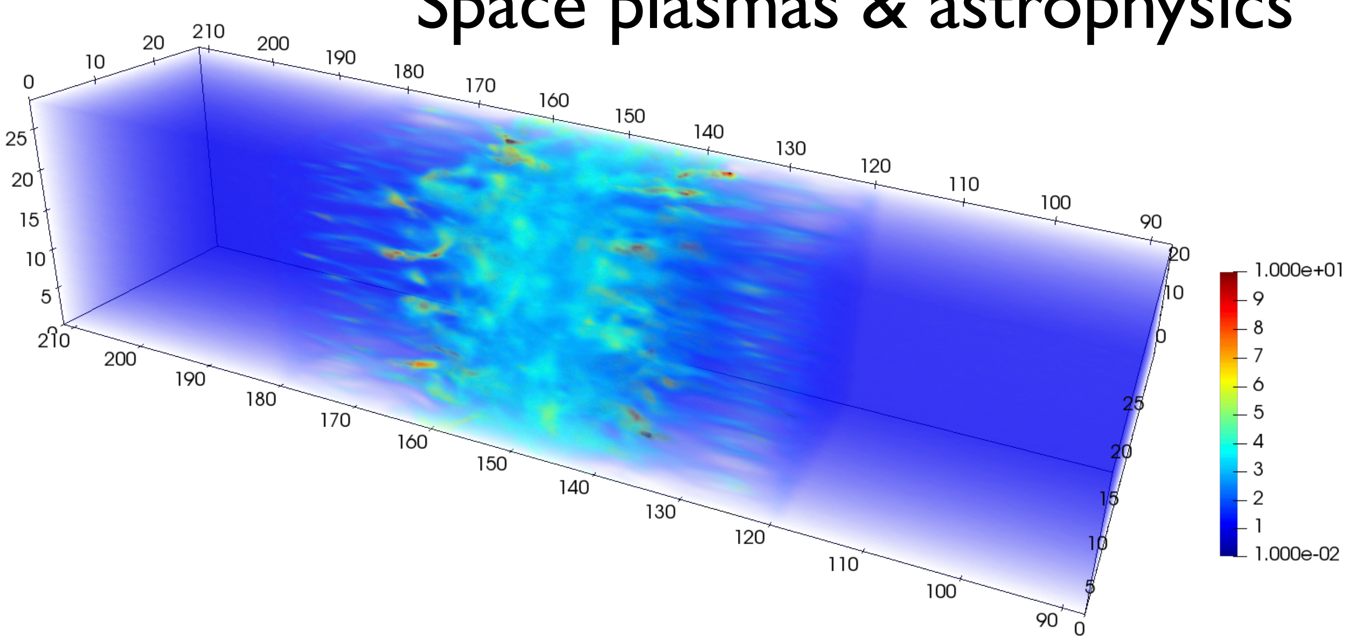
source: K. Heitmann, Argonne National Lab

Space propulsion (Plasma thruster)



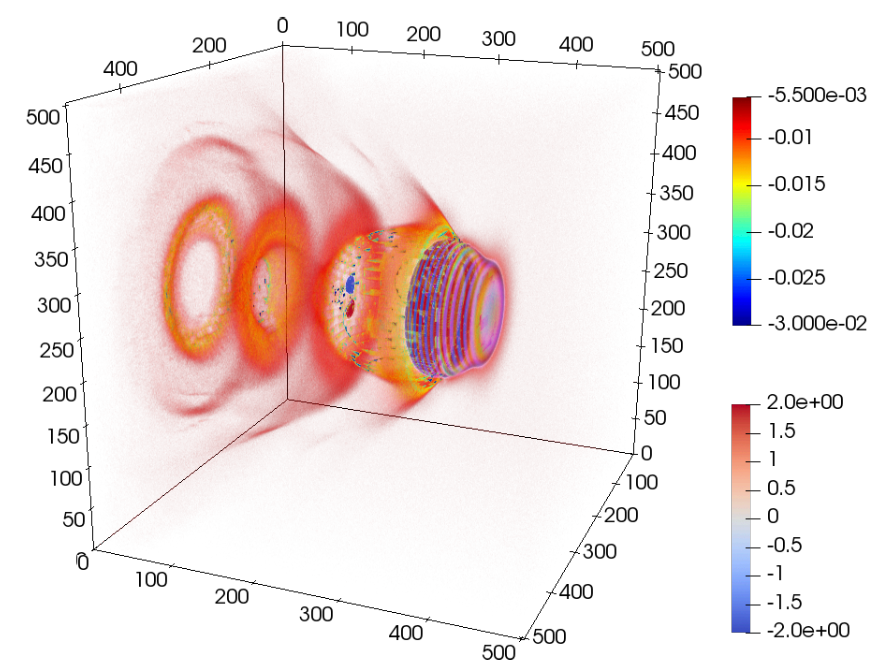
source: Gauss Center for Supercomputing

Space plasmas & astrophysics



source: SMILEI dev-team

Laser plasma interaction



source: SMILEI dev-team

- Conceptually simple
- Efficiently implemented on (massively) parallel super-computers

Our starting point is the Vlasov-Maxwell description for a *collisionless* plasma

Our starting point is the Vlasov-Maxwell description for a *collisionless* plasma

Plasma

$$\partial_t f_s + \frac{\mathbf{p}}{m_s \gamma} \cdot \nabla f_s + \mathbf{F}_L \cdot \nabla_{\mathbf{p}} f_s = 0$$

Our starting point is the Vlasov-Maxwell description for a *collisionless* plasma

Plasma

$$\partial_t f_s + \frac{\mathbf{p}}{m_s \gamma} \cdot \nabla f_s + \mathbf{F}_L \cdot \nabla_{\mathbf{p}} f_s = 0$$

Electromagnetic Field

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0} \qquad \partial_t \mathbf{E} = -\frac{1}{\epsilon_0} \mathbf{J} + c^2 \nabla \times \mathbf{B}$$

$$\nabla \cdot \mathbf{B} = 0 \qquad \partial_t \mathbf{B} = -\nabla \times \mathbf{E}$$

Our starting point is the Vlasov-Maxwell description for a *collisionless* plasma

Plasma

$$\partial_t f_s + \frac{\mathbf{p}}{m_s \gamma} \cdot \nabla f_s + \mathbf{F}_L \cdot \nabla_{\mathbf{p}} f_s = 0$$

$$\mathbf{F}_L = q_s \left(\mathbf{E} + \frac{\mathbf{p}}{m_s \gamma} \times \mathbf{B} \right)$$

$$\rho(t, \mathbf{x}) = \int d\mathbf{p} f_s(t, \mathbf{x}, \mathbf{p})$$

$$J(t, \mathbf{x}) = q_s \int d\mathbf{p} \frac{\mathbf{p}}{m_s \gamma} f_s(t, \mathbf{x}, \mathbf{p})$$

Electromagnetic Field

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0} \qquad \partial_t \mathbf{E} = -\frac{1}{\epsilon_0} \mathbf{J} + c^2 \nabla \times \mathbf{B}$$

$$\nabla \cdot \mathbf{B} = 0 \qquad \partial_t \mathbf{B} = -\nabla \times \mathbf{E}$$

1st Remark

Normalization: the Vlasov-Maxwell (relativistic) description provides us with a set of natural units

Plasma

$$\partial_t f_s + \frac{\mathbf{p}}{m_s \gamma} \cdot \nabla f_s + \mathbf{F}_L \cdot \nabla_{\mathbf{p}} f_s = 0$$

Electromagnetic Field

$$\nabla \cdot \mathbf{E} = \rho \qquad \partial_t \mathbf{E} = -\mathbf{J} + \nabla \times \mathbf{B}$$

$$\nabla \cdot \mathbf{B} = 0 \qquad \partial_t \mathbf{B} = -\nabla \times \mathbf{E}$$

Velocity	c
Charge	e
Mass	m_e
Momentum	$m_e\, c$
Energy, Temperature	$m_e c^2$
Time	ω_r^{-1}
Length	c/ω_r
Number density	$n_r = \epsilon_0\, m_e\, \omega_r^2 / e^2$
Current density	$e\, c\, n_r$
Pressure	$m_e\, c^2\, n_r$
Electric field	$m_e\, c\, \omega_r / e$
Magnetic field	$m_e\, \omega_r / e$
Poynting flux	$m_e\, c^3\, n_r / 2$

1st Remark

Normalization: the Vlasov-Maxwell (relativistic) description provides us with a set of natural units

Plasma

$$\partial_t f_s + \frac{\mathbf{p}}{m_s \gamma} \cdot \nabla f_s + \mathbf{F}_L \cdot \nabla_{\mathbf{p}} f_s = 0$$

Electromagnetic Field

$$\nabla \cdot \mathbf{E} = \rho \qquad \partial_t \mathbf{E} = -\mathbf{J} + \nabla \times \mathbf{B}$$

$$\nabla \cdot \mathbf{B} = 0 \qquad \partial_t \mathbf{B} = -\nabla \times \mathbf{E}$$

Velocity	c
Charge	e
Mass	m_e
Momentum	$m_e\, c$
Energy, Temperature	$m_e c^2$
Time	ω_r^{-1}
Length	c/ω_r
Number density	$n_r = \epsilon_0\, m_e\, \omega_r^2/e^2$
Current density	$e\, c\, n_r$
Pressure	$m_e\, c^2\, n_r$
Electric field	$m_e\, c\, \omega_r/e$
Magnetic field	$m_e\, \omega_r/e$
Poynting flux	$m_e\, c^3\, n_r/2$

The value of ω_r is not defined *a priori*, and acts as a *scaling factor*.

2nd Remark

The Particle-In-Cell method integrates Vlasov Equation along the trajectories of so-called *quasi-particles*

Vlasov Eq. is a **partial differential equation** (PDE) in *Ns*+*Nv* phase-space:

$$\partial_t f_s + \frac{\mathbf{p}}{m_s \gamma} \cdot \nabla f_s + \mathbf{F}_L \cdot \nabla_{\mathbf{p}} f_s = 0$$

2nd Remark

The Particle-In-Cell method integrates Vlasov Equation along the trajectories of so-called *quasi-particles*

Vlasov Eq. is a **partial differential equation** (PDE) in *Ns*+*Nv* phase-space:

$$\partial_t f_s + \frac{\mathbf{p}}{m_s \gamma} \cdot \nabla f_s + \mathbf{F}_L \cdot \nabla_{\mathbf{p}} f_s = 0$$

Direct integration (*Vlasov codes*) has tremendous computational cost!

2nd Remark

The Particle-In-Cell method integrates Vlasov Equation along the trajectories of so-called *quasi-particles*

Vlasov Eq. is a **partial differential equation** (PDE) in *Ns*+*Nv* phase-space:

$$\partial_t f_s + \frac{\mathbf{p}}{m_s \gamma} \cdot \nabla f_s + \mathbf{F}_L \cdot \nabla_{\mathbf{p}} f_s = 0$$

Direct integration (*Vlasov codes*) has tremendous computational cost!

The **PIC ansatz** consists in decomposing the distribution fct:

$$f_s(t, \mathbf{x}, \mathbf{p}) = \sum_{p=1}^{N} w_p \, S(\mathbf{x} - \mathbf{x}_p(t)) \, \delta(\mathbf{p} - \mathbf{p}_p(t))$$

Shape-function Dirac-distribution

2nd Remark

The Particle-In-Cell method integrates Vlasov Equation along the trajectories of so-called *quasi-particles*

Injecting this *ansatz* in Vlasov Eq., multiplying by $\mathbf{p}$ and integrating over all momenta $\mathbf{p}$

$$\sum_{p=1}^{N_s} w_p \, \frac{\mathbf{p}_p}{m_s \gamma_p} \, \mathbf{p}_p \cdot \left[\partial_{\mathbf{x}_p} S(\mathbf{x}-\mathbf{x}_p) + \partial_{\mathbf{x}} S(\mathbf{x}-\mathbf{x}_p)\right]$$

$$+ \sum_{p=1}^{N_s} w_p \, S(\mathbf{x}-\mathbf{x}_p) \, \left[\partial_t \mathbf{p}_p - q_s \left(\mathbf{E} + \mathbf{v}_p \times \mathbf{B}\right)\right] = 0$$

Let us now integrate in space:

$$\sum_{p=1}^{N_s} w_p \, \frac{\mathbf{p}_p}{m_s \gamma_p} \, \mathbf{p}_p \cdot \cancel{\int d\mathbf{x} \left[\partial_{\mathbf{x}_p} S(\mathbf{x}-\mathbf{x}_p) + \partial_{\mathbf{x}} S(\mathbf{x}-\mathbf{x}_p)\right]}$$

$$+ \sum_{p=1}^{N_s} w_p \int d\mathbf{x} S(\mathbf{x}-\mathbf{x}_p) \, \left[\partial_t \mathbf{p}_p - q_s \left(\mathbf{E} + \mathbf{v}_p \times \mathbf{B}\right)\right] = 0$$

Finally leading to solving for all $\mathbf{p}$:

$$\partial_t \mathbf{p}_p = q_s \left(\mathbf{E}_p + \mathbf{v} \times \mathbf{B}_p\right) \quad \text{with} \quad (\mathbf{E},\mathbf{B})_p \equiv \int d\mathbf{x} \, (\mathbf{E},\mathbf{B})(\mathbf{x}) \, S(\mathbf{x}-\mathbf{x}_p)$$

3rd Remark

If one does things in a *smart way*, only Maxwell-Ampère & Maxwell-Faraday Eqs. need to be solved

3rd Remark

If one does things in a *smart way*, only Maxwell-Ampère & Maxwell-Faraday Eqs. need to be solved

Take the divergence of Maxwell-Ampère's Eq. :

$$\nabla \cdot (\partial_t \mathbf{E} + \mathbf{J} = \nabla \times \mathbf{B})$$
$$\Leftrightarrow$$
$$\partial_t \nabla \cdot \mathbf{E} + \nabla \cdot \mathbf{J} = 0$$

3rd Remark

If one does things in a *smart way*, only Maxwell-Ampère & Maxwell-Faraday Eqs. need to be solved

Take the divergence of Maxwell-Ampère's Eq. :

$$\nabla \cdot (\partial_t \mathbf{E} + \mathbf{J} = \nabla \times \mathbf{B})$$
$$\Leftrightarrow$$
$$\partial_t \nabla \cdot \mathbf{E} + \nabla \cdot \mathbf{J} = 0$$

Assume charge is conserved, i.e., $\partial_t \rho + \nabla \cdot \mathbf{J} = 0$

3rd Remark

If one does things in a *smart way*, only Maxwell-Ampère & Maxwell-Faraday Eqs. need to be solved

Take the divergence of Maxwell-Ampère's Eq. :

$$\nabla \cdot (\partial_t \mathbf{E} + \mathbf{J} = \nabla \times \mathbf{B})$$
$$\Leftrightarrow$$
$$\partial_t \nabla \cdot \mathbf{E} + \nabla \cdot \mathbf{J} = 0$$

Assume charge is conserved, i.e., $\partial_t \rho + \nabla \cdot \mathbf{J} = 0$

One gets: $\partial_t (\nabla \cdot \mathbf{E} - \rho) = 0$

3rd Remark

If one does things in a *smart way*, only Maxwell-Ampère & Maxwell-Faraday Eqs. need to be solved

Take the divergence of Maxwell-Ampère's Eq. :

$$\nabla \cdot (\partial_t \mathbf{E} + \mathbf{J} = \nabla \times \mathbf{B})$$
$$\Leftrightarrow$$
$$\partial_t \nabla \cdot \mathbf{E} + \nabla \cdot \mathbf{J} = 0$$

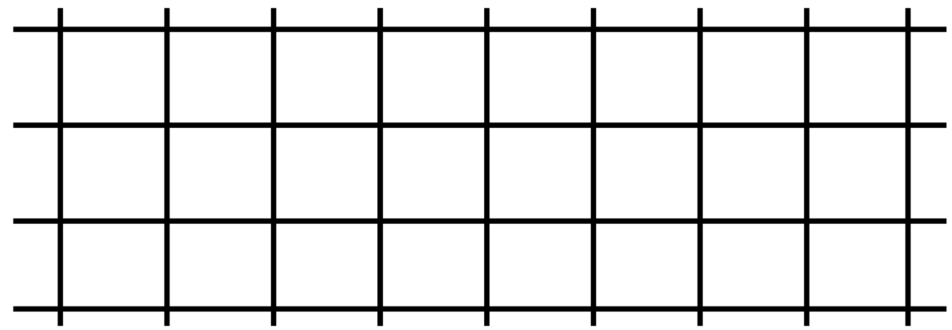
Assume charge is conserved, i.e., $\partial_t \rho + \nabla \cdot \mathbf{J} = 0$

One gets: $\partial_t (\nabla \cdot \mathbf{E} - \rho) = 0$

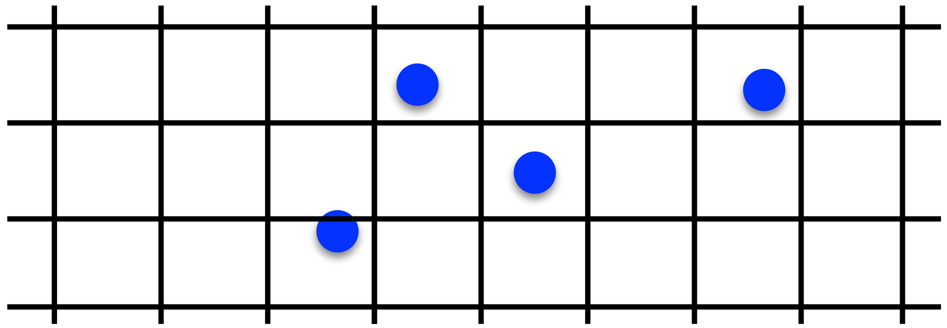
If at time t=0, Poisson & Gauss Eqs. are satisfied,
and if current deposition is made in a way that conserve charge,

then solving only Maxwell-Ampère & Maxwell-Faraday ensures that both Eqs. remain satisfied at later time.

Initialization of a PIC simulation

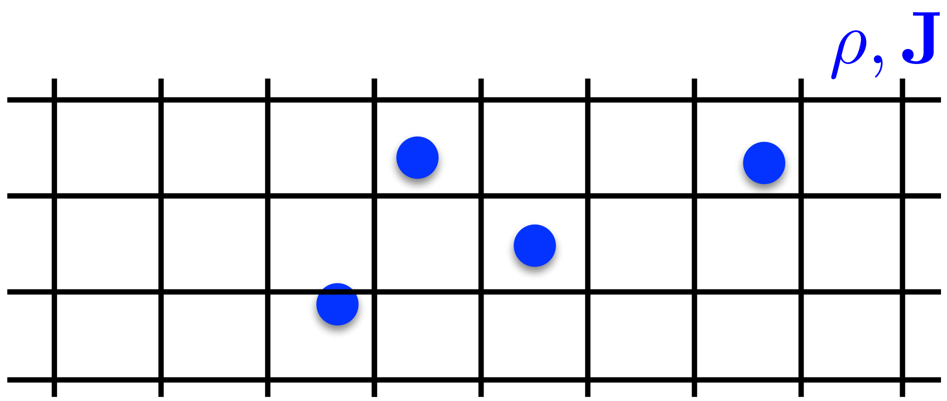


Initialization of a PIC simulation



1) **for each species of your plasma, create your quasi-particles**
   e.g. defining the species density, velocity and temperature profiles

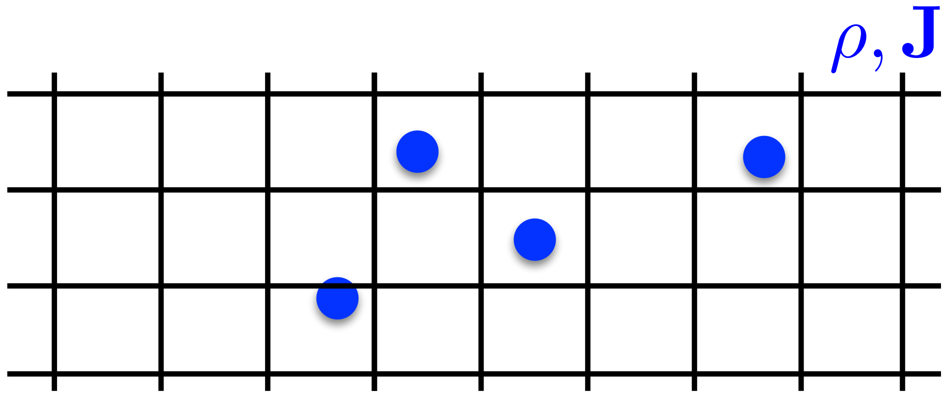
Initialization of a PIC simulation



1) for each species of your plasma, create your quasi-particles
   e.g. defining the species density, velocity and temperature profiles

2) loop over all particles and project charge and current density onto the grid

Initialization of a PIC simulation

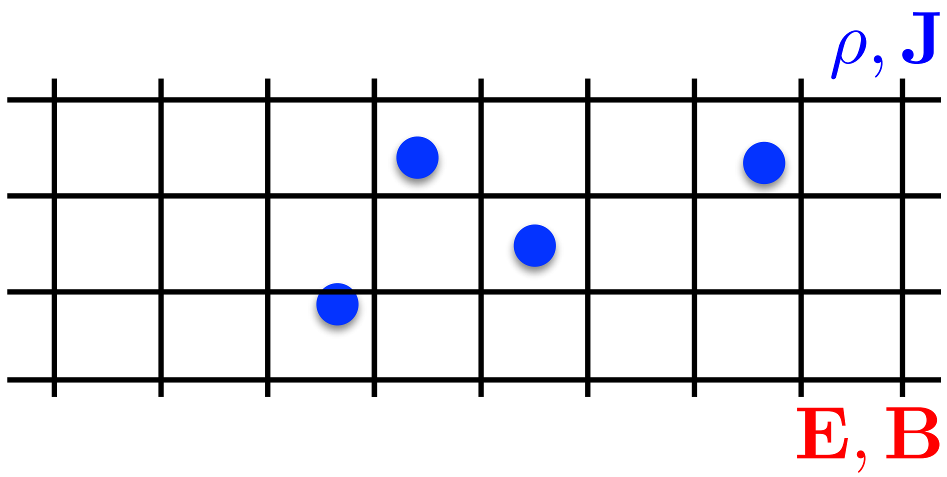


1) for each species of your plasma, create your quasi-particles
   e.g. defining the species density, velocity and temperature profiles

2) loop over all particles and project charge and current density onto the grid

3) knowing the charge density solve Poisson's Eq. to get the electrostatic field

Initialization of a PIC simulation



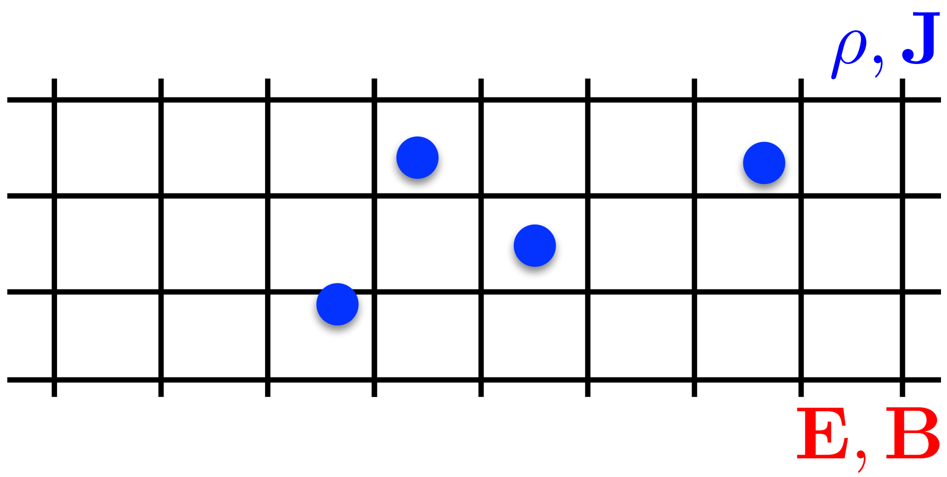
1) for each species of your plasma, create your quasi-particles
   e.g. defining the species density, velocity and temperature profiles

2) loop over all particles and project charge and current density onto the grid

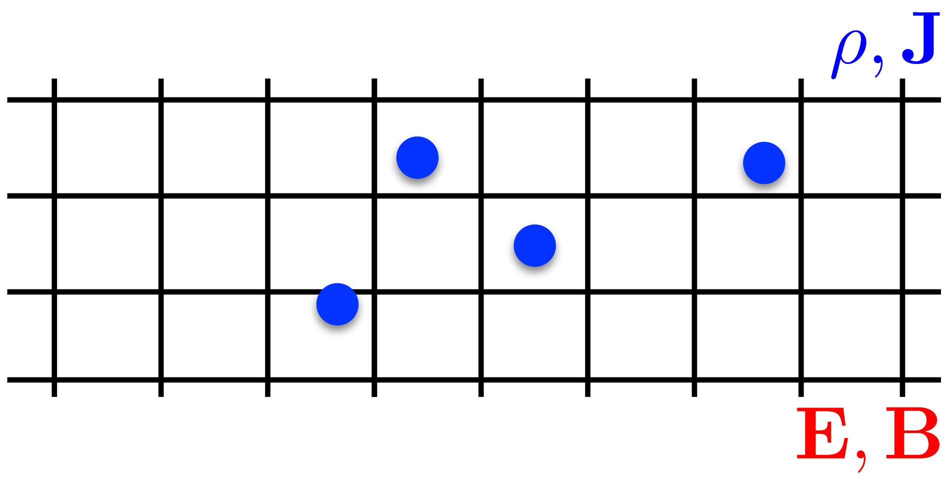
3) knowing the charge density solve Poisson's Eq. to get the electrostatic field

4) add any (user defined) external fields provided they are divergence-free

The Particle-In-Cell loop



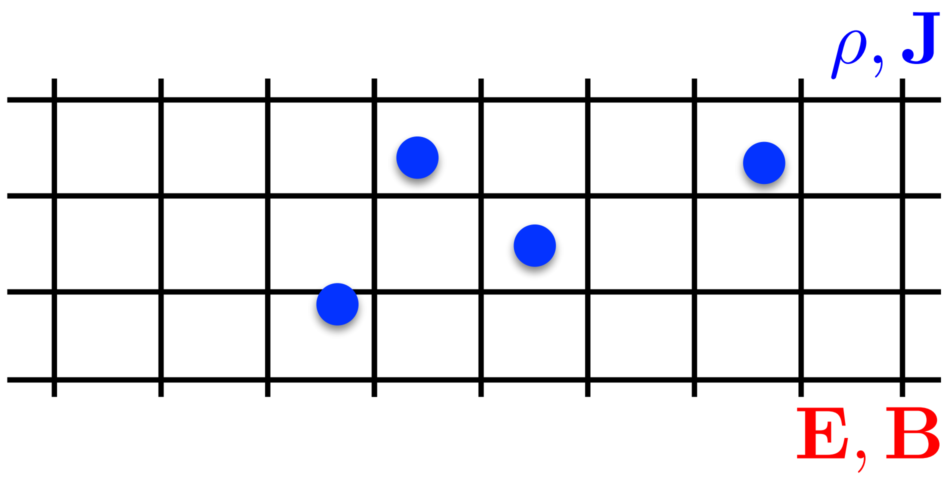
The Particle-In-Cell loop



Gather fields at particle position

$$[\mathbf{E}, \mathbf{B}] \rightarrow [\mathbf{E}_p, \mathbf{B}_p]$$

The Particle-In-Cell loop



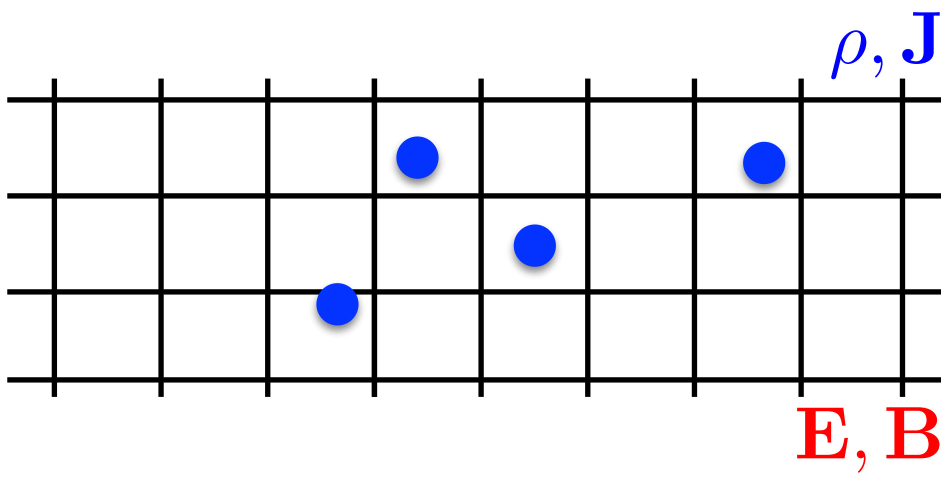
Gather fields at particle position

$$[\mathbf{E}, \mathbf{B}] \rightarrow [\mathbf{E}_p, \mathbf{B}_p]$$

Push all particles

$$\forall p \quad \begin{aligned} d_t \mathbf{u}_p &= \frac{q_s}{m_s} \mathbf{F}_L \\ d_t \mathbf{x}_p &= \mathbf{u}_p / \gamma_p \end{aligned}$$

The Particle-In-Cell loop



Gather fields at particle position

$$[\mathbf{E}, \mathbf{B}] \rightarrow [\mathbf{E}_p, \mathbf{B}_p]$$

Push all particles

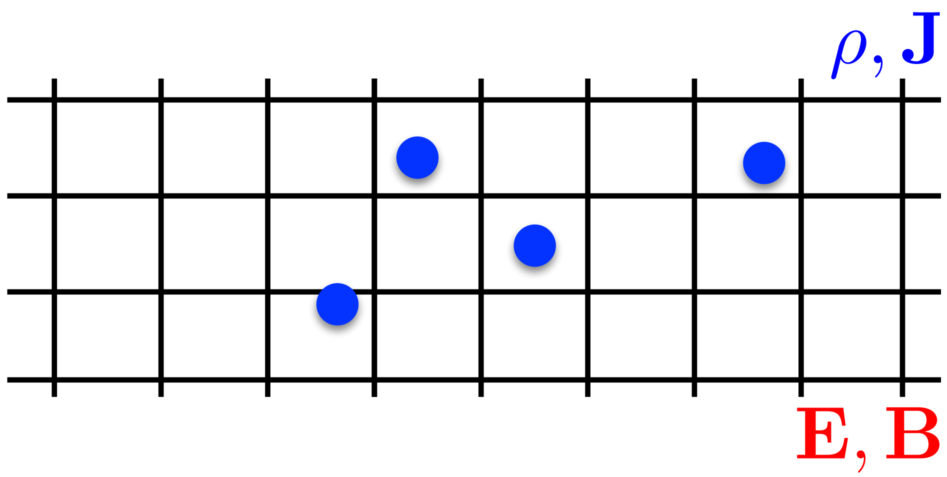
$$\forall p \quad \begin{aligned} d_t \mathbf{u}_p &= \frac{q_s}{m_s} \mathbf{F}_L \\ d_t \mathbf{x}_p &= \mathbf{u}_p / \gamma_p \end{aligned}$$

Project current densities on grid*

$$[\mathbf{x}_p, \mathbf{p}_p] \rightarrow [\rho, \mathbf{J}]$$

* using a charge conserving scheme

The Particle-In-Cell loop



Gather fields at particle position

$$[\mathbf{E}, \mathbf{B}] \rightarrow [\mathbf{E}_p, \mathbf{B}_p]$$

Solve Maxwell's Eqs.

$$\partial_t \mathbf{E} = -\mathbf{J} + \nabla \times \mathbf{B}$$

$$\partial_t \mathbf{B} = -\nabla \times \mathbf{E}$$

Push all particles

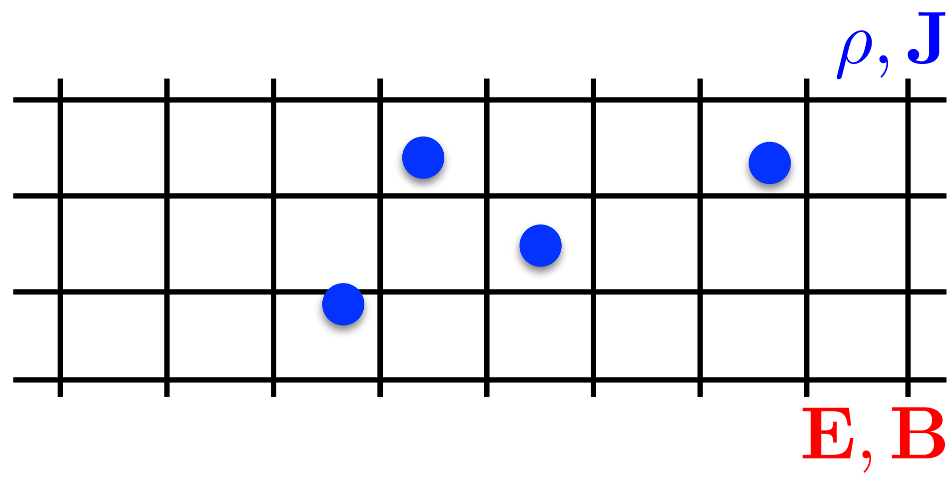
$$\forall p \quad \begin{aligned} d_t \mathbf{u}_p &= \frac{q_s}{m_s} \mathbf{F}_L \\ d_t \mathbf{x}_p &= \mathbf{u}_p / \gamma_p \end{aligned}$$

Project current densities on grid*

$$[\mathbf{x}_p, \mathbf{p}_p] \rightarrow [\rho, \mathbf{J}]$$

* using a charge conserving scheme

The Particle-In-Cell loop



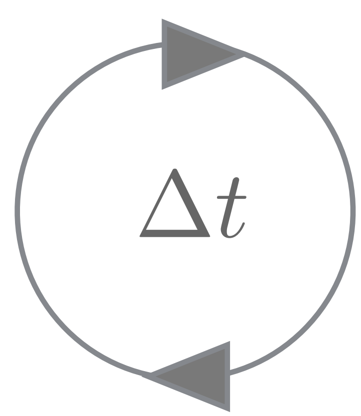
Gather fields at particle position

$$[\mathbf{E}, \mathbf{B}] \rightarrow [\mathbf{E}_p, \mathbf{B}_p]$$

Solve Maxwell's Eqs.

$$\partial_t \mathbf{E} = -\mathbf{J} + \nabla \times \mathbf{B}$$

$$\partial_t \mathbf{B} = -\nabla \times \mathbf{E}$$



Push all particles

$$\forall p \quad d_t \mathbf{u}_p = \frac{q_s}{m_s} \mathbf{F}_L$$

$$d_t \mathbf{x}_p = \mathbf{u}_p / \gamma_p$$

Project current densities on grid*

$$[\mathbf{x}_p, \mathbf{p}_p] \rightarrow [\rho, \mathbf{J}]$$

* using a charge conserving scheme

Outlines

- Numerical approach: how to build up your PIC code
- Parallelization: getting ready for the super-computers
- Additional modules: beyond the *collisionless* plasma
- Some physics highlights: what you can do with a PIC code

Outlines

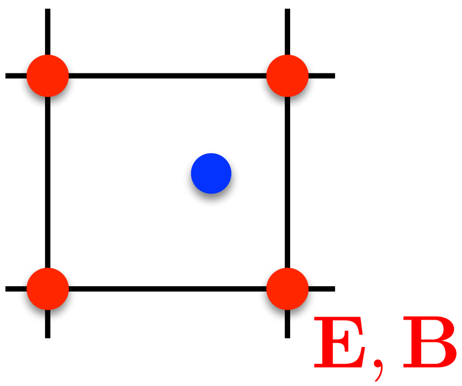
- Numerical approach: how to build up your PIC code
- Parallelization: getting ready for the super-computers
- Additional modules: beyond the *collisionless* plasma
- Some physics highlights: what you can do with a PIC code

Step 1

Field gathering: interpolation at particle position

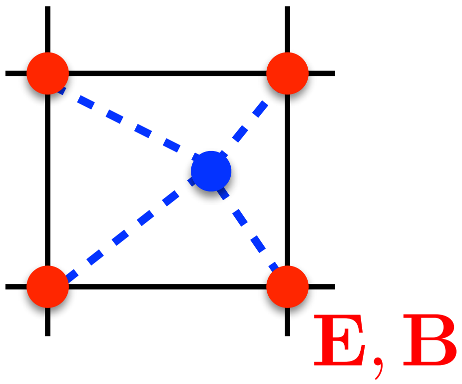
Step 1

Field gathering: interpolation at particle position



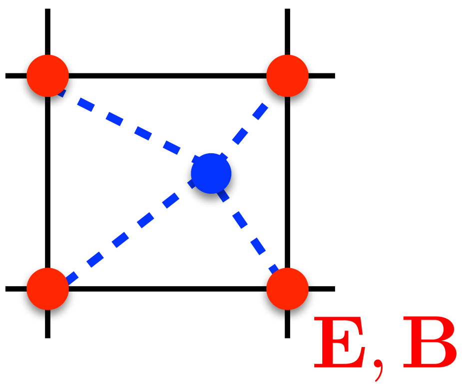
Step 1

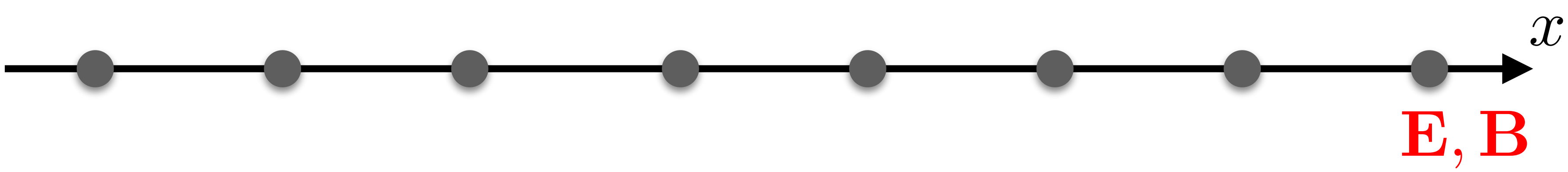
Field gathering: interpolation at particle position



Step 1

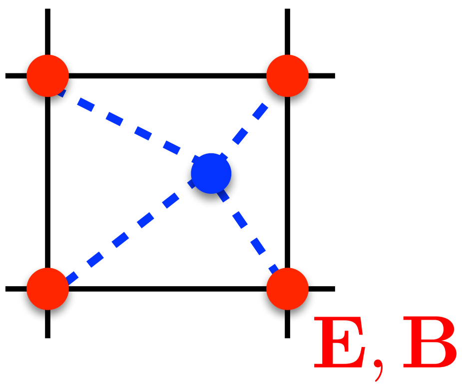
Field gathering: interpolation at particle position

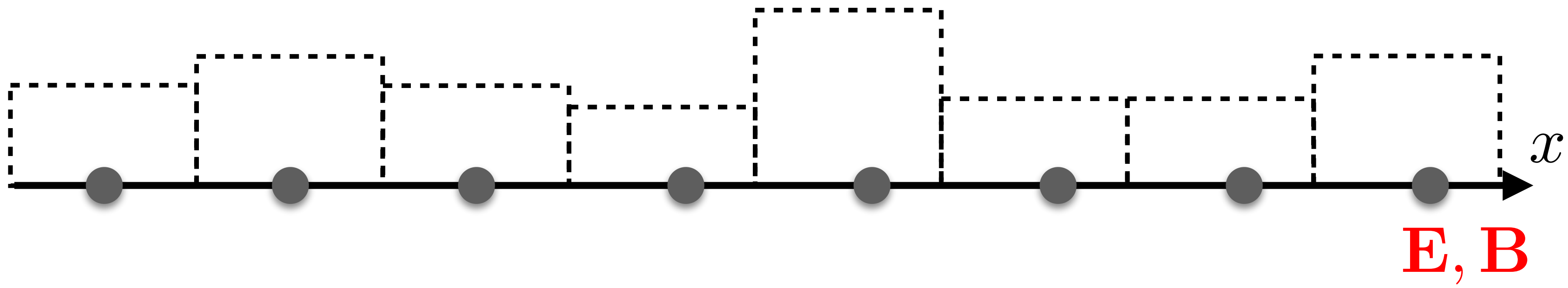




Step 1

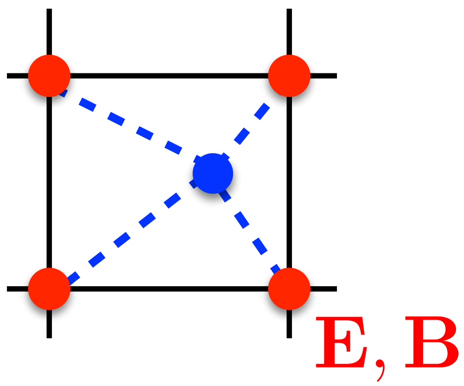
Field gathering: interpolation at particle position

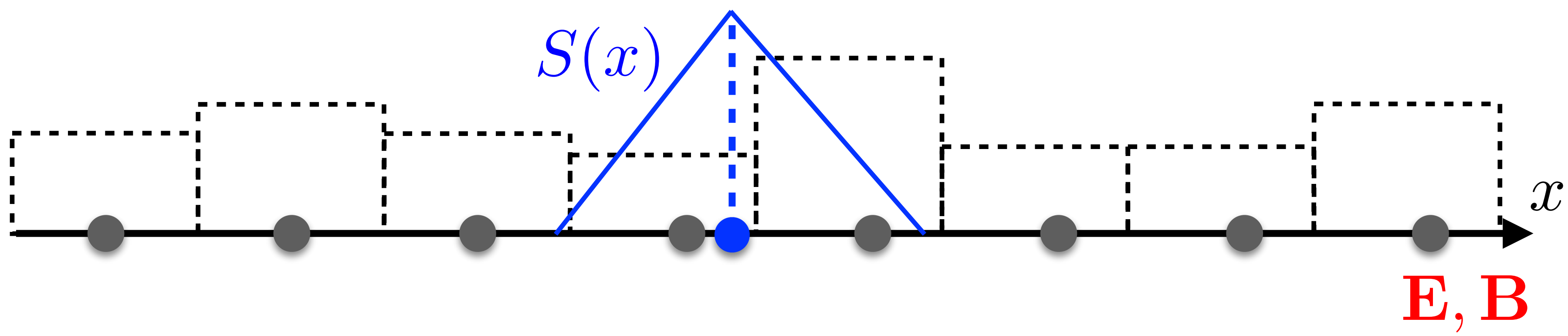




Step 1

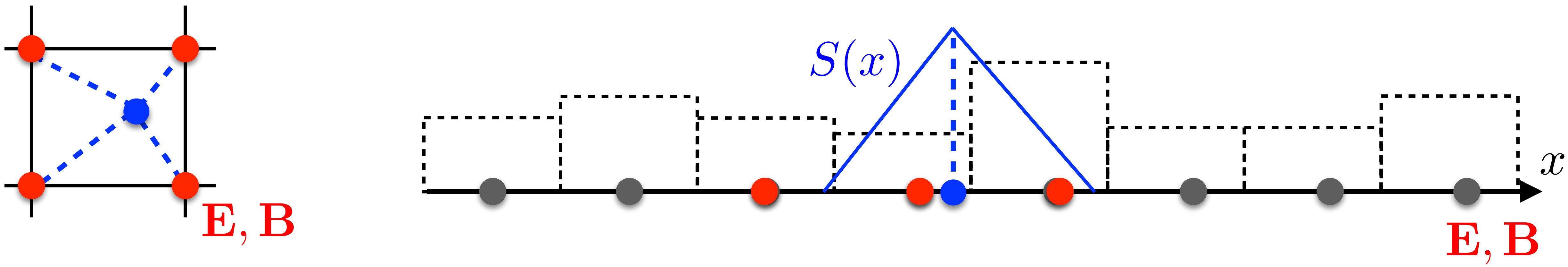
Field gathering: interpolation at particle position





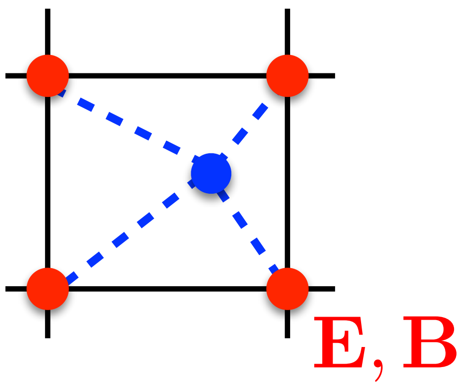
Step 1

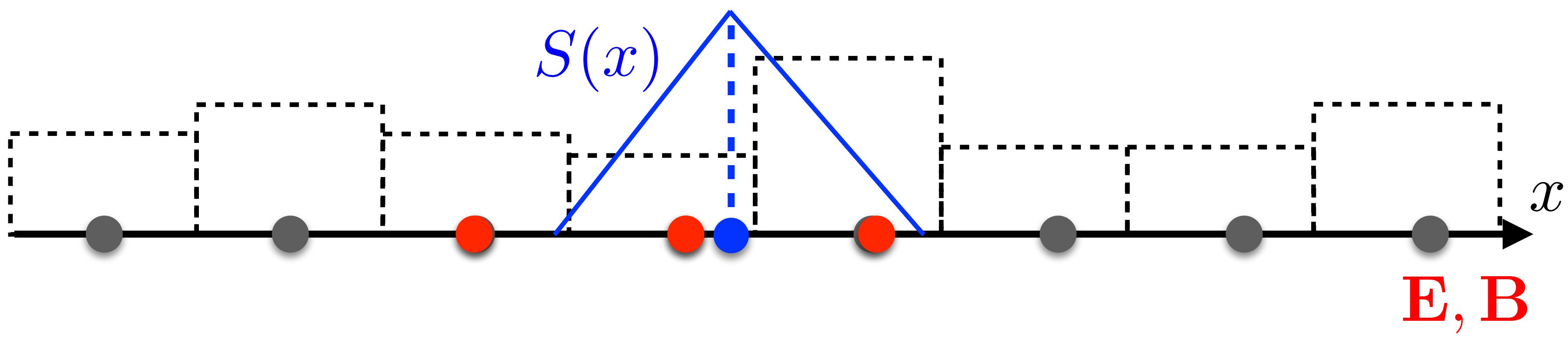
Field gathering: interpolation at particle position



Step 1

Field gathering: interpolation at particle position





$$(\mathbf{E},\mathbf{B})_p \equiv \int d\mathbf{x}\,(\mathbf{E},\mathbf{B})(\mathbf{x})\,S(\mathbf{x}-\mathbf{x}_p)$$

$$\hat{s}^{(0)}(x) = \Delta x\,\delta(x),$$

$$\hat{s}^{(1)}(x) = \begin{cases} 1 & \text{if } |x| \le \frac{1}{2}\Delta x, \\ 0 & \text{otherwise}, \end{cases}$$

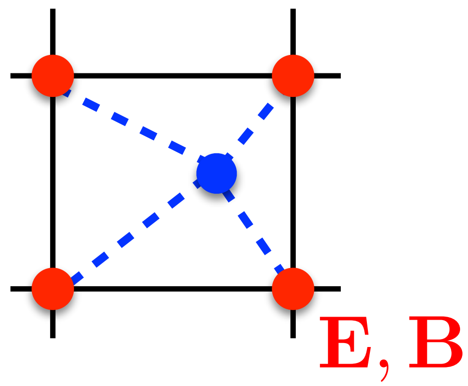
$$\hat{s}^{(2)}(x) = \begin{cases} \left(1 - \left|\frac{x}{\Delta x}\right|\right) & \text{if } |x| \le \Delta x, \\ 0 & \text{otherwise}, \end{cases}$$

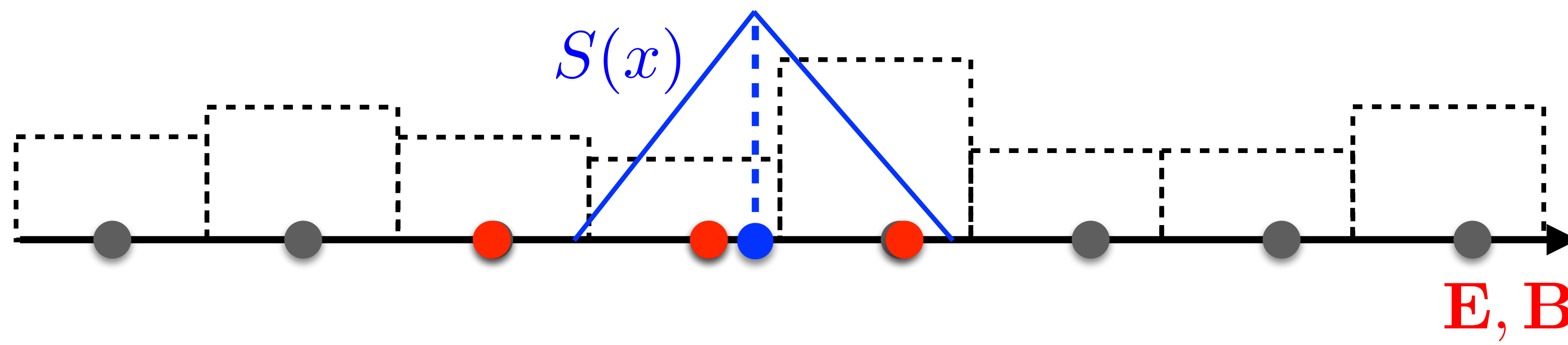
$$\hat{s}^{(3)}(x) = \begin{cases} \frac{3}{4}\left[1 - \frac{4}{3}\left(\frac{x}{\Delta x}\right)^2\right] & \text{if } |x| \le \frac{1}{2}\Delta x, \\ \frac{9}{8}\left(1 - \frac{2}{3}\left|\frac{x}{\Delta x}\right|\right)^2 & \text{if } \frac{1}{2}\Delta x < |x| \le \frac{3}{2}\Delta x, \\ 0 & \text{otherwise}, \end{cases}$$

$$\hat{s}^{(4)}(x) = \begin{cases} \frac{2}{3}\left[1 - \frac{3}{2}\left(\frac{x}{\Delta x}\right)^2 + \frac{3}{4}\left|\frac{x}{\Delta x}\right|^3\right] & \text{if } |x| \le \Delta x, \\ \frac{4}{3}\left(1 - \frac{1}{2}\left|\frac{x}{\Delta x}\right|\right)^3 & \text{if } \Delta x < |x| \le 2\,\Delta x, \\ 0 & \text{otherwise}. \end{cases}$$

Step 1

Field gathering: interpolation at particle position





$$(\mathbf{E},\mathbf{B})_p \equiv \int d\mathbf{x}\, (\mathbf{E},\mathbf{B})(\mathbf{x})\, S(\mathbf{x}-\mathbf{x}_p)$$

$$\hat{s}^{(0)}(x) = \Delta x\, \delta(x),$$

$$\hat{s}^{(1)}(x) = \begin{cases} 1 & \text{if } |x| \leq \frac{1}{2}\Delta x, \\ 0 & \text{otherwise,} \end{cases}$$

$$\boxed{\hat{s}^{(2)}(x) = \begin{cases} \left(1 - \left|\frac{x}{\Delta x}\right|\right) & \text{if } |x| \leq \Delta x, \\ 0 & \text{otherwise,} \end{cases}}$$

$$\hat{s}^{(3)}(x) = \begin{cases} \frac{3}{4}\left[1 - \frac{4}{3}\left(\frac{x}{\Delta x}\right)^2\right] & \text{if } |x| \leq \frac{1}{2}\Delta x, \\ \frac{9}{8}\left(1 - \frac{2}{3}\left|\frac{x}{\Delta x}\right|\right)^2 & \text{if } \frac{1}{2}\Delta x < |x| \leq \frac{3}{2}\Delta x, \\ 0 & \text{otherwise,} \end{cases}$$

$$\hat{s}^{(4)}(x) = \begin{cases} \frac{2}{3}\left[1 - \frac{3}{2}\left(\frac{x}{\Delta x}\right)^2 + \frac{3}{4}\left|\frac{x}{\Delta x}\right|^3\right] & \text{if } |x| \leq \Delta x, \\ \frac{4}{3}\left(1 - \frac{1}{2}\left|\frac{x}{\Delta x}\right|\right)^3 & \text{if } \Delta x < |x| \leq 2\,\Delta x, \\ 0 & \text{otherwise.} \end{cases}$$

Step 2

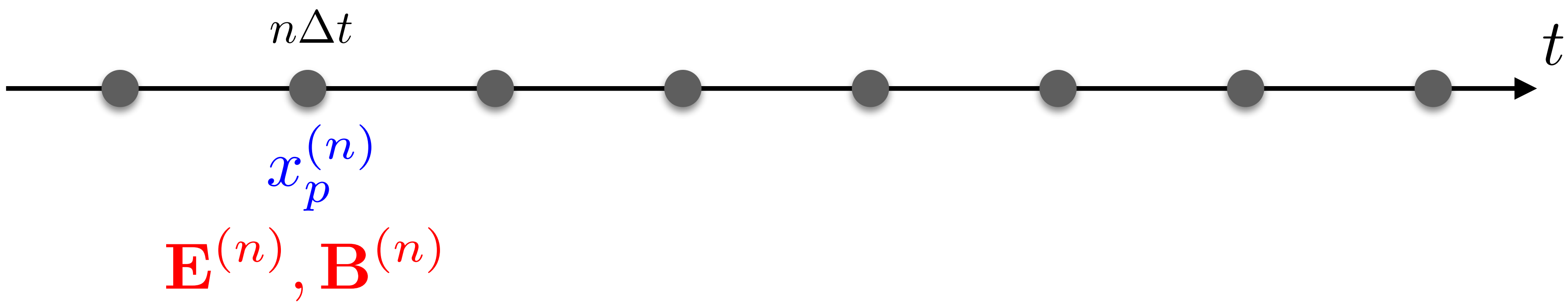
The Boris leap-frog *pusher* is a very popular method to advance particles

Step 2

The Boris leap-frog *pusher* is a very popular method to advance particles

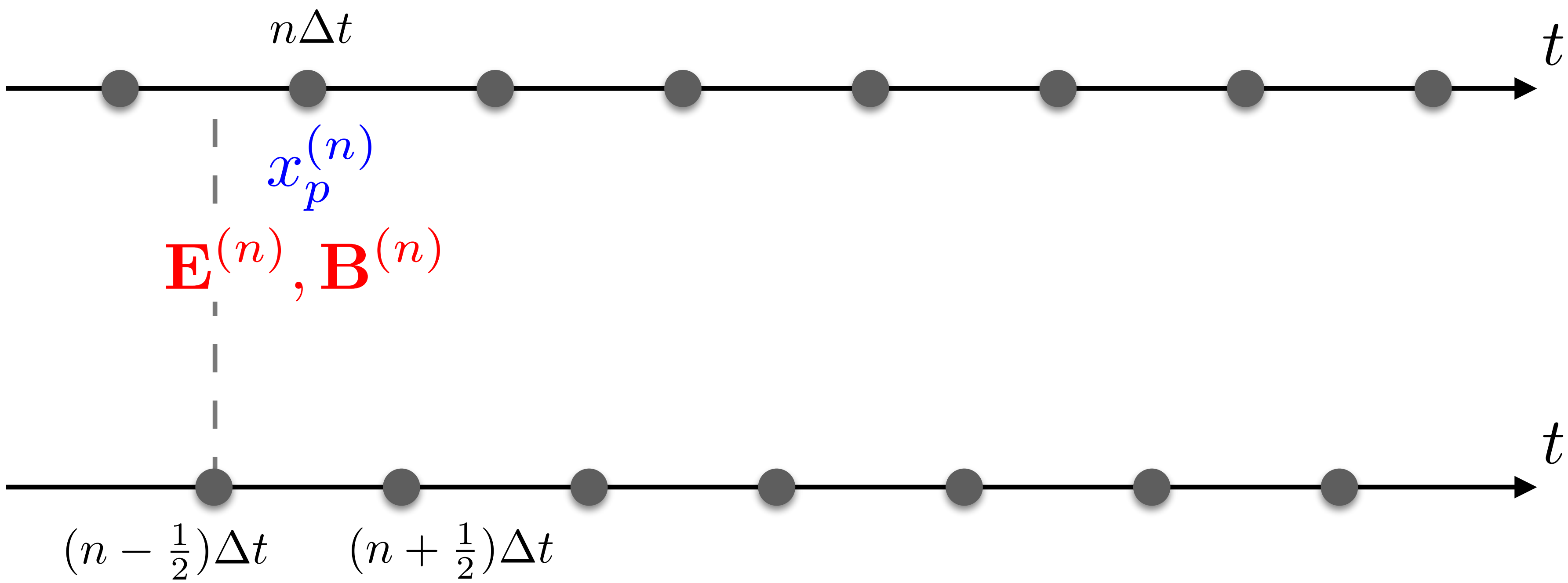
Step 2

The Boris leap-frog *pusher* is a very popular method to advance particles



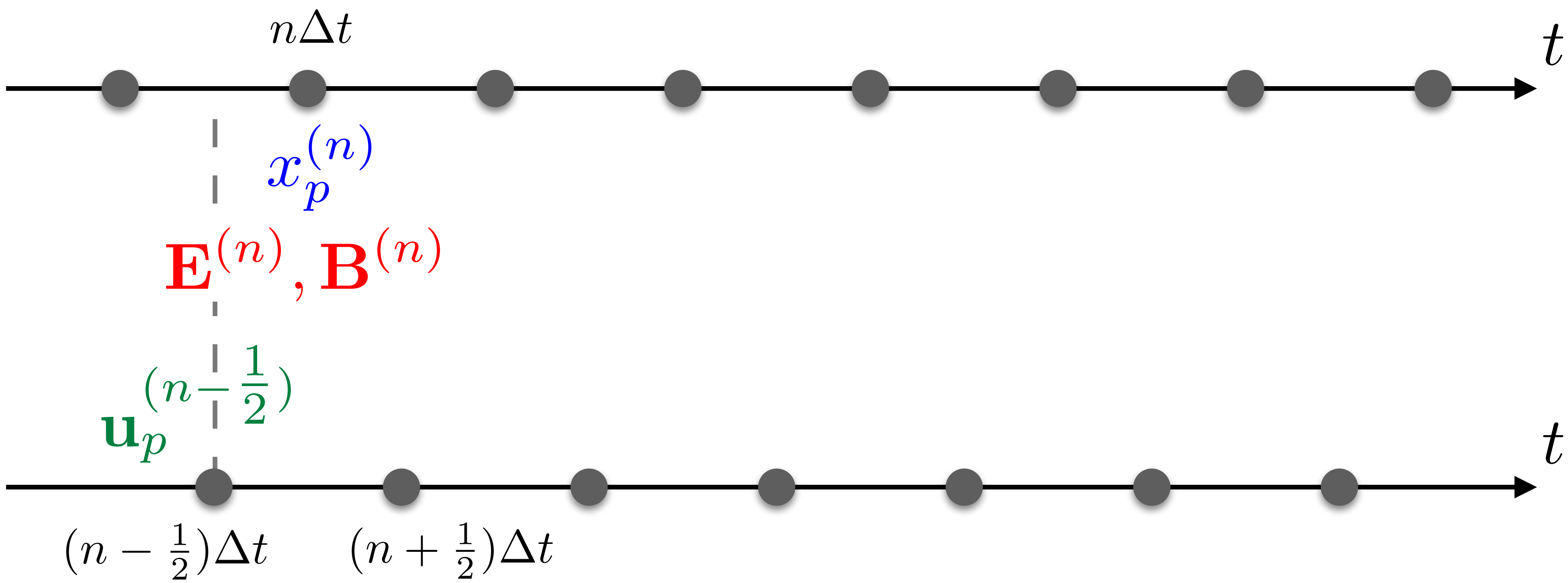
Step 2

The Boris leap-frog *pusher* is a very popular method to advance particles



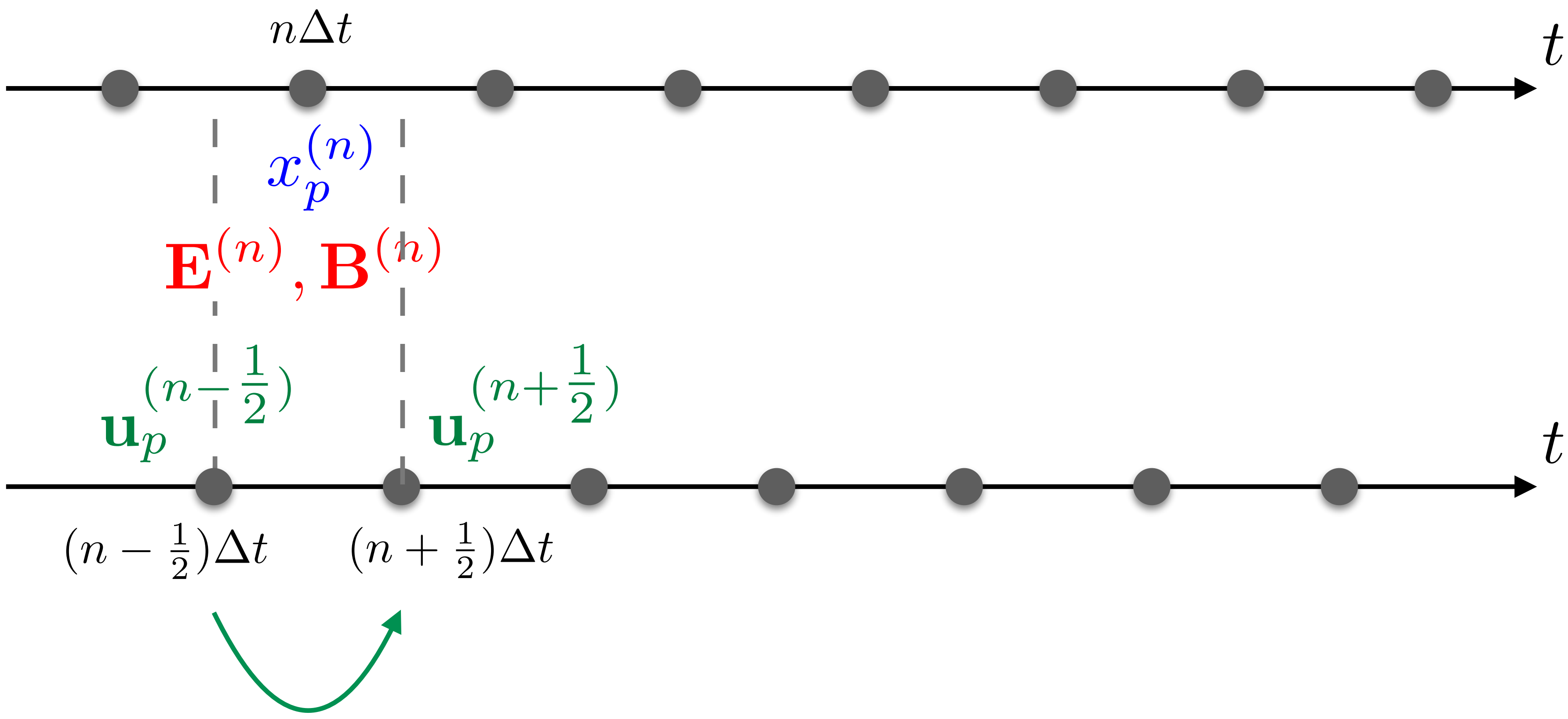
Step 2

The Boris leap-frog *pusher* is a very popular method to advance particles



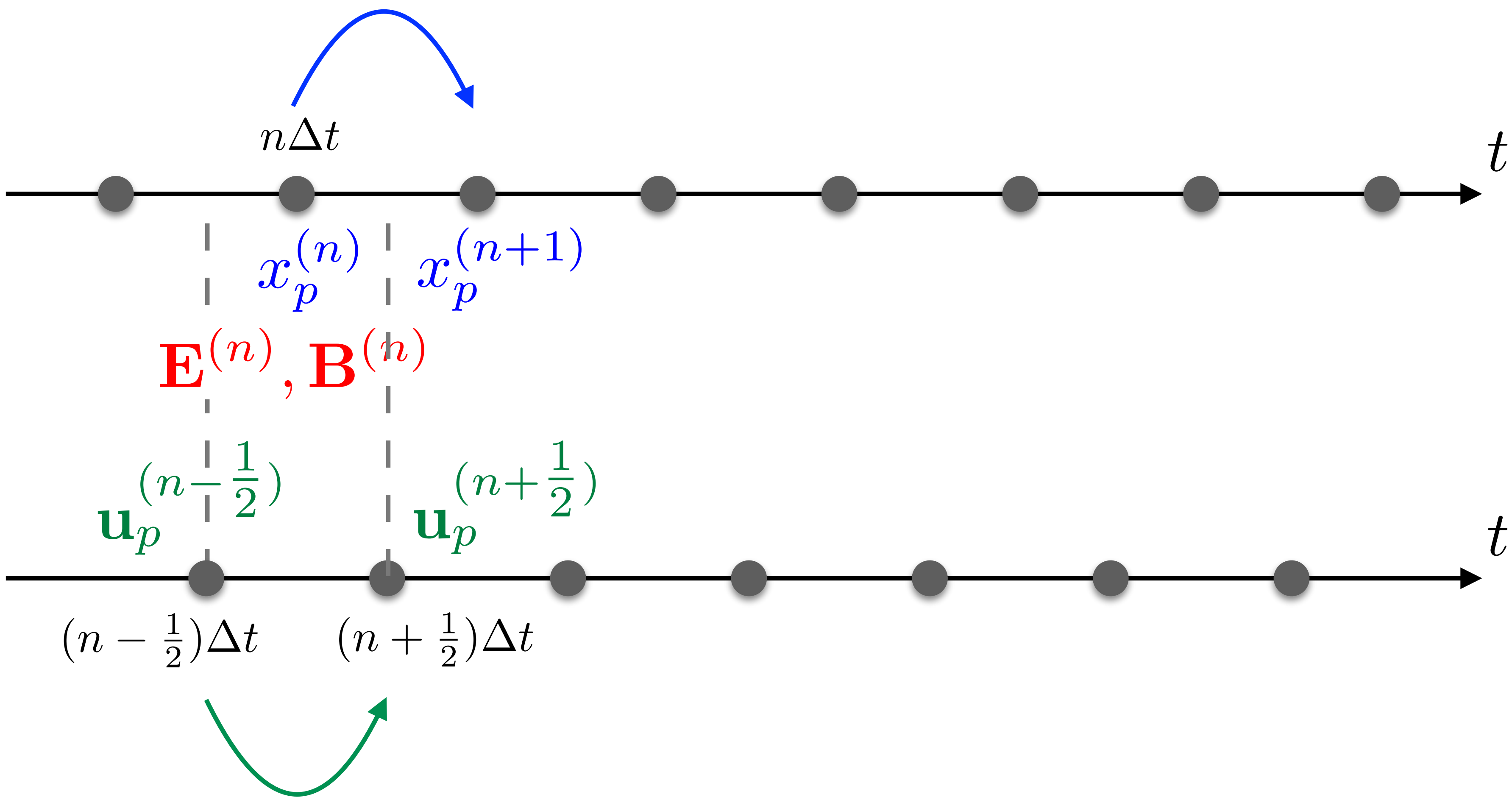
[Step 2]

The Boris leap-frog *pusher* is a very popular method to advance particles



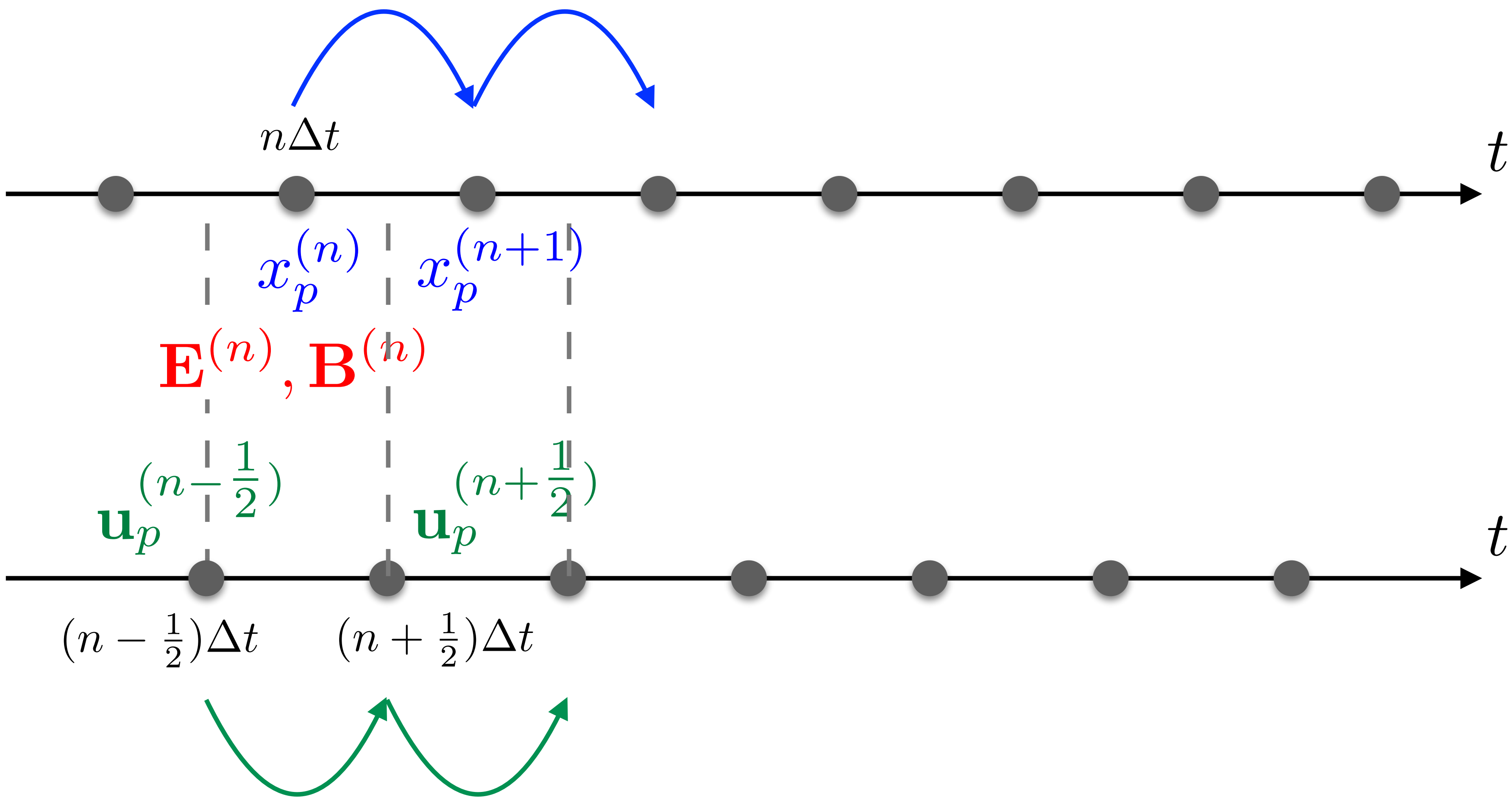
Step 2

The Boris leap-frog *pusher* is a very popular method to advance particles



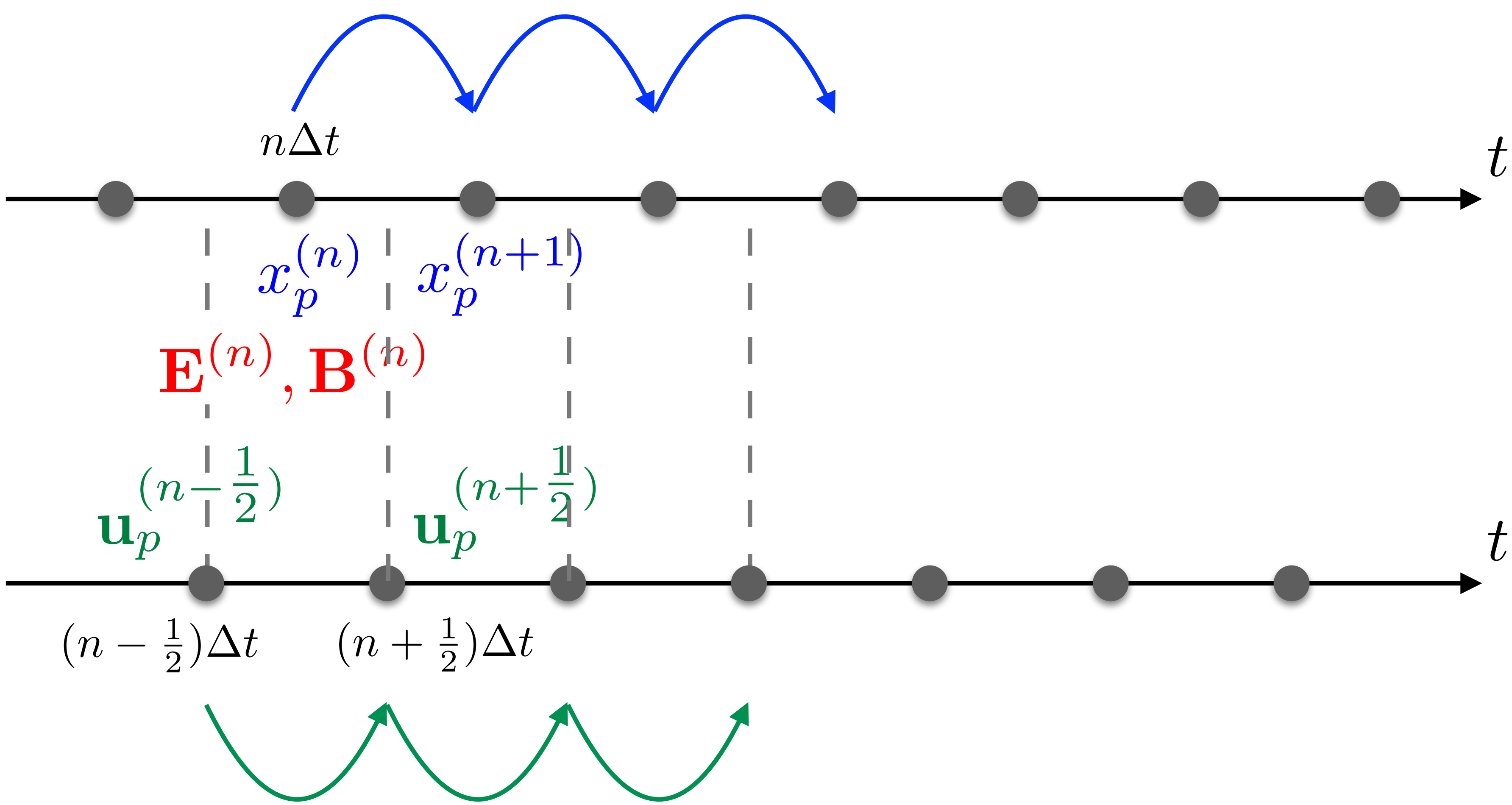
Step 2

The Boris leap-frog *pusher* is a very popular method to advance particles



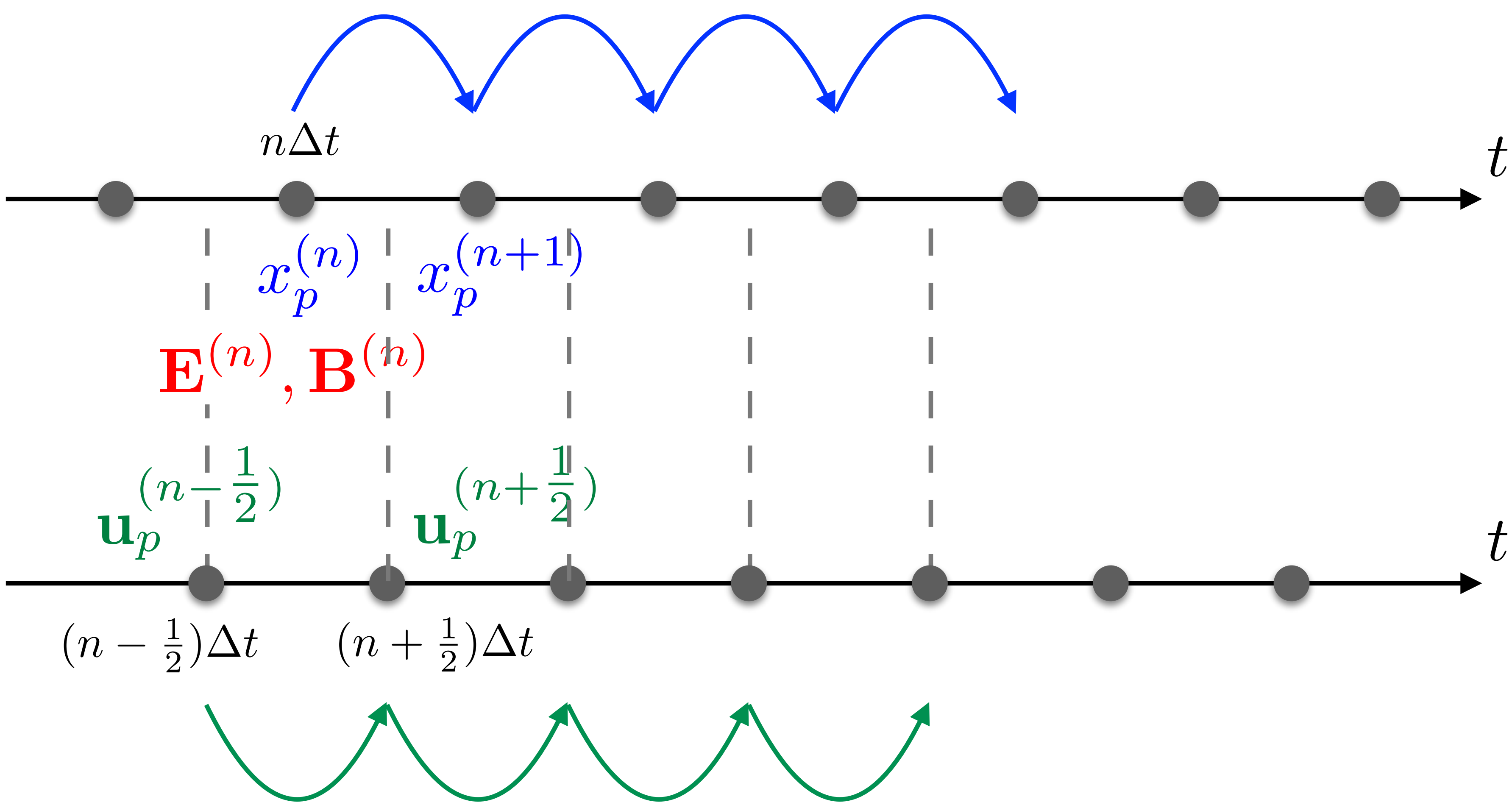
Step 2

The Boris leap-frog *pusher* is a very popular method to advance particles



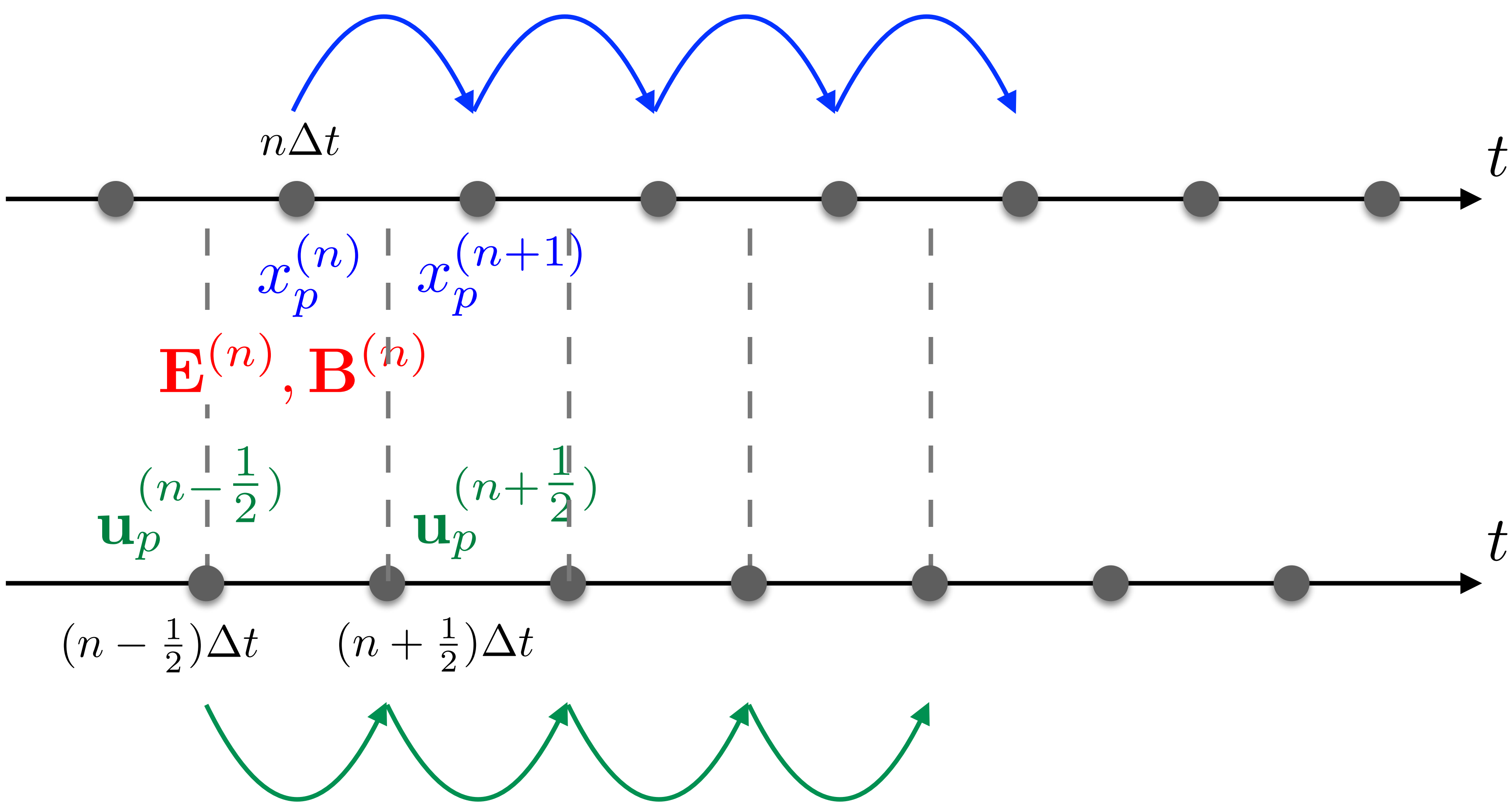
Step 2

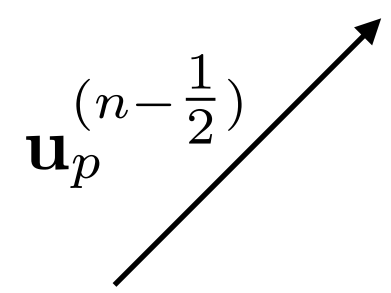
The Boris leap-frog *pusher* is a very popular method to advance particles



Step 2

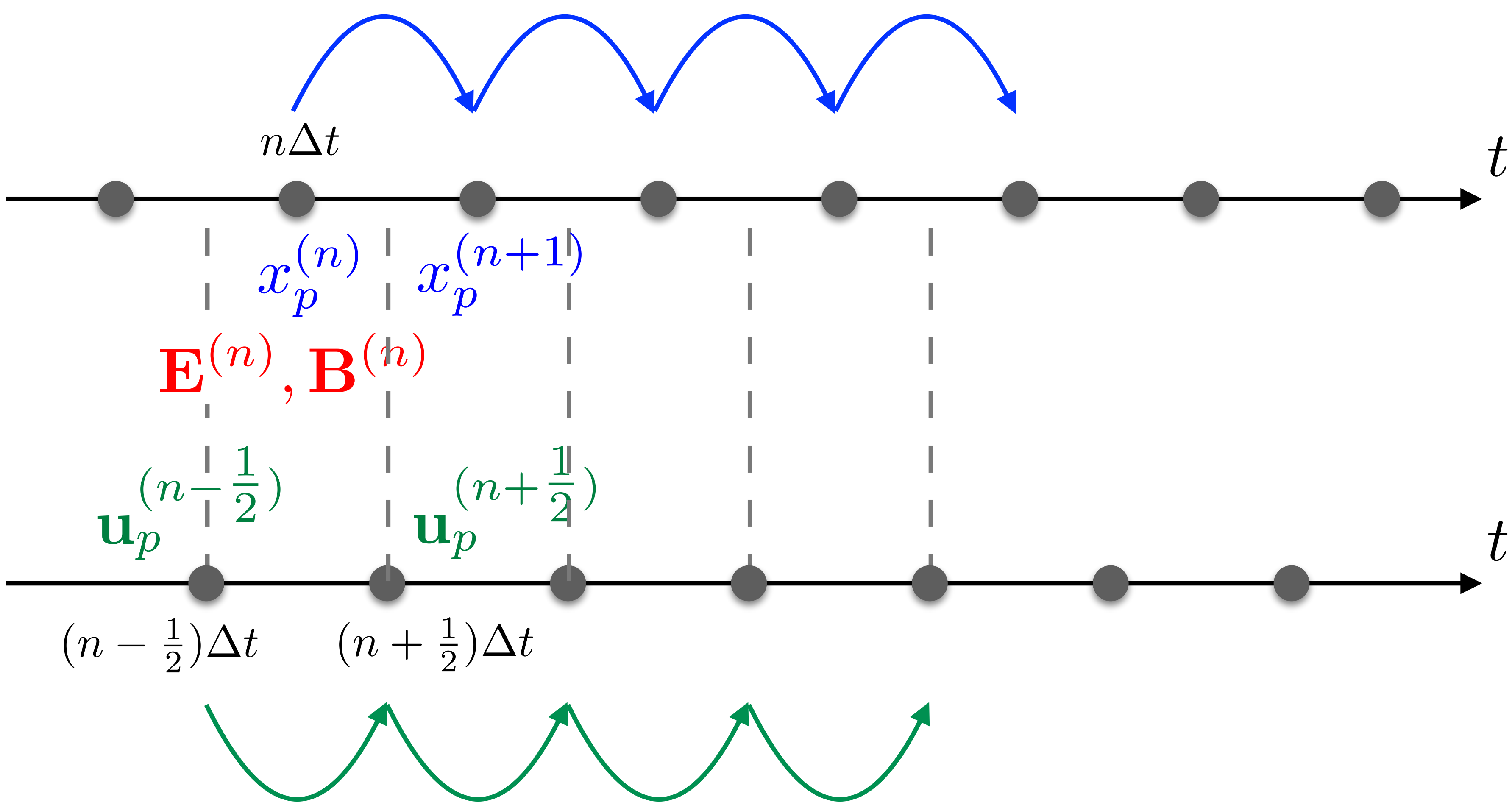
The Boris leap-frog *pusher* is a very popular method to advance particles



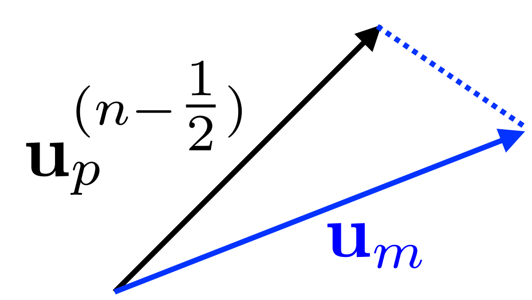


Step 2

The Boris leap-frog *pusher* is a very popular method to advance particles

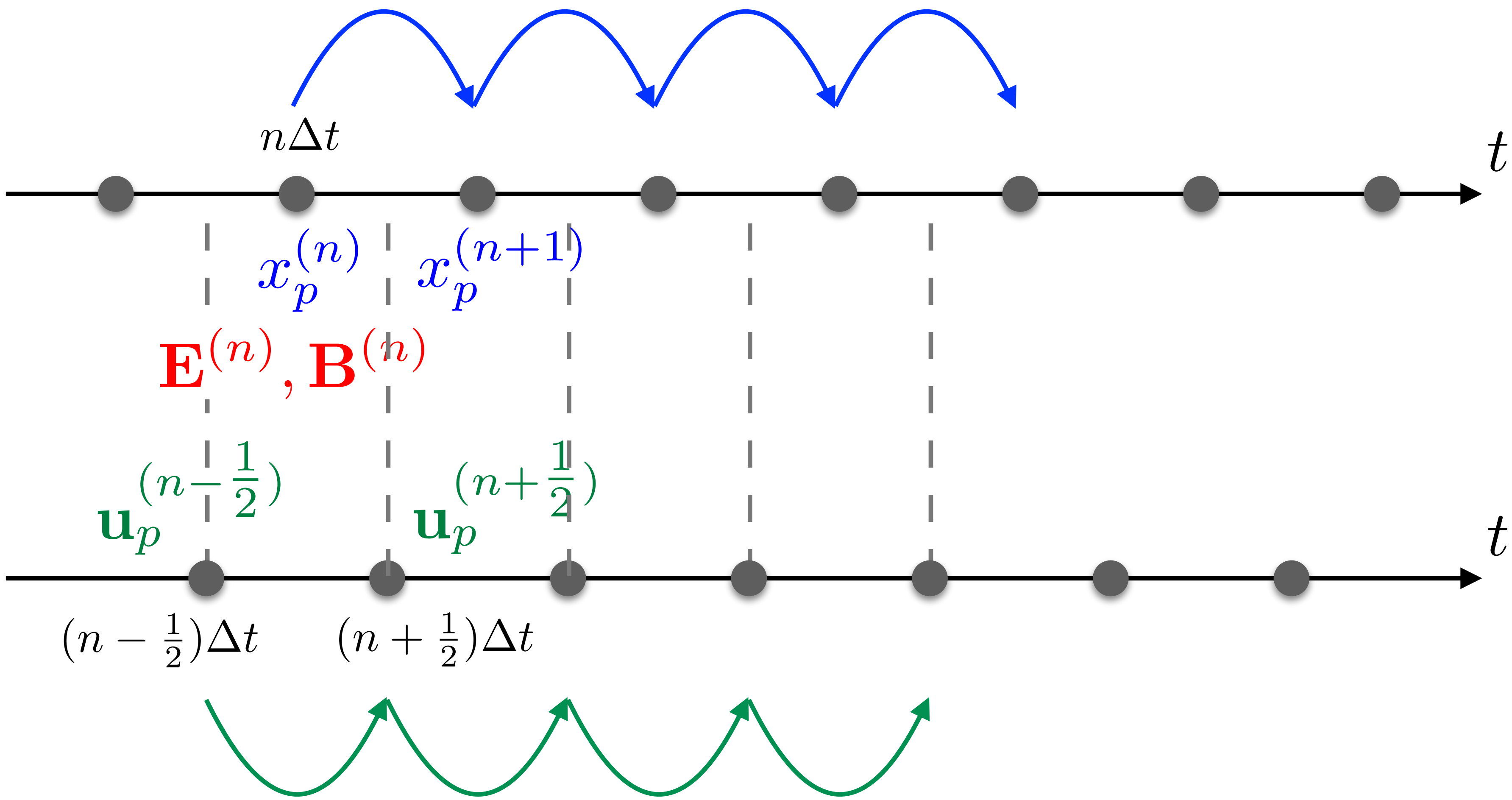


$$\mathbf{u}_m = \mathbf{u}_p^{(n-\frac{1}{2})} + \frac{q_s}{m_s}\frac{\Delta t}{2}\mathbf{E}_p$$



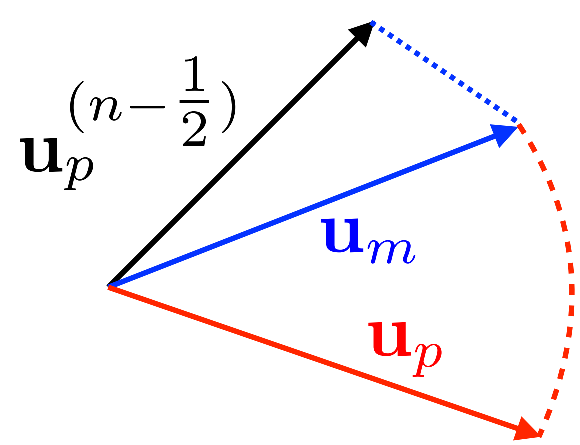
Step 2

The Boris leap-frog *pusher* is a very popular method to advance particles



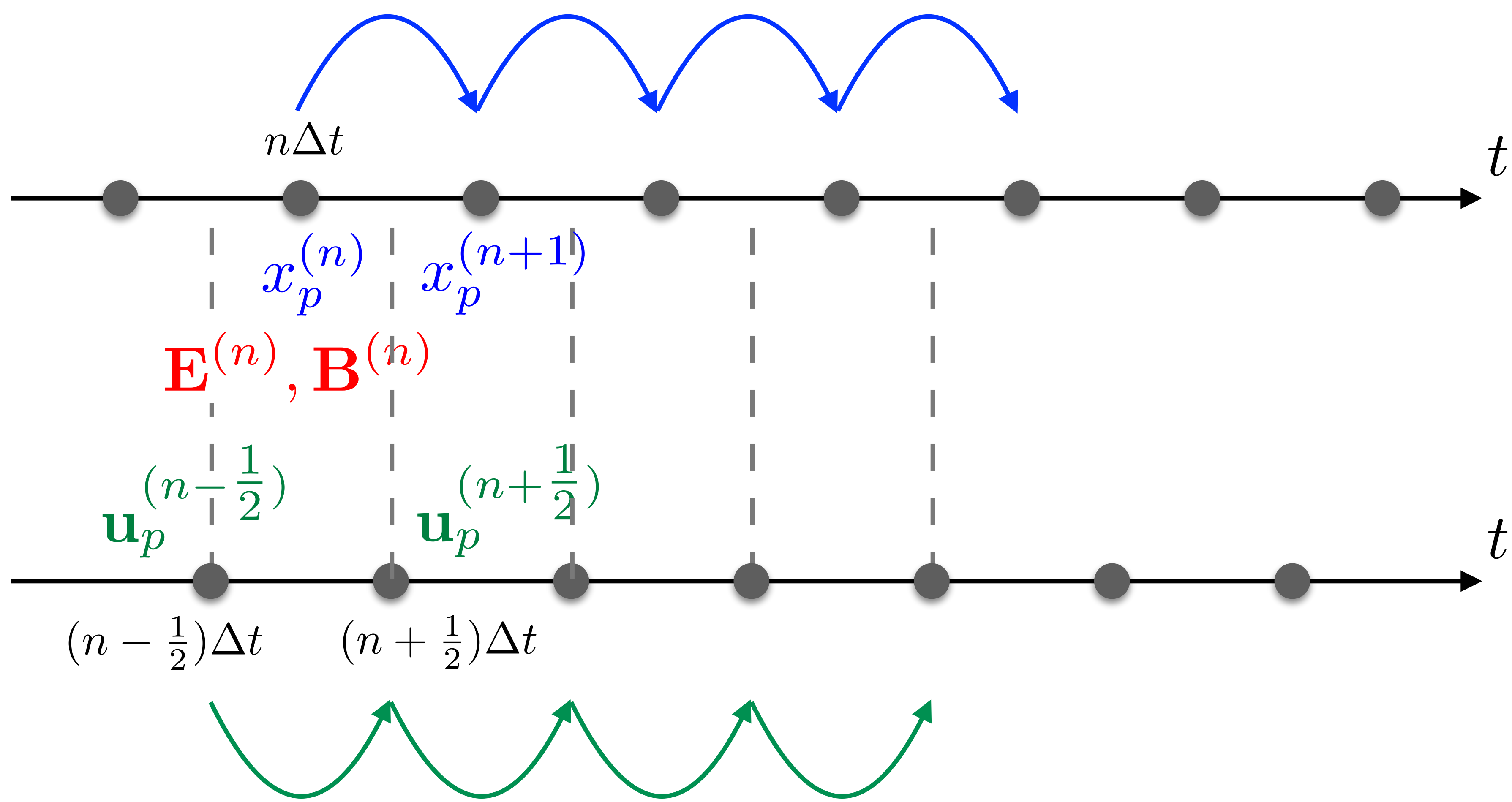
$$\mathbf{u}_m = \mathbf{u}_p^{(n-\frac{1}{2})} + \frac{q_s}{m_s}\frac{\Delta t}{2}\mathbf{E}_p$$

$$\mathbf{u}_p = \mathbf{u}_p^{(n-\frac{1}{2})} + \frac{q_s}{m_s}\Delta t\,\mathcal{M}(\mathbf{B}_p)\,\mathbf{u}_m$$



Step 2

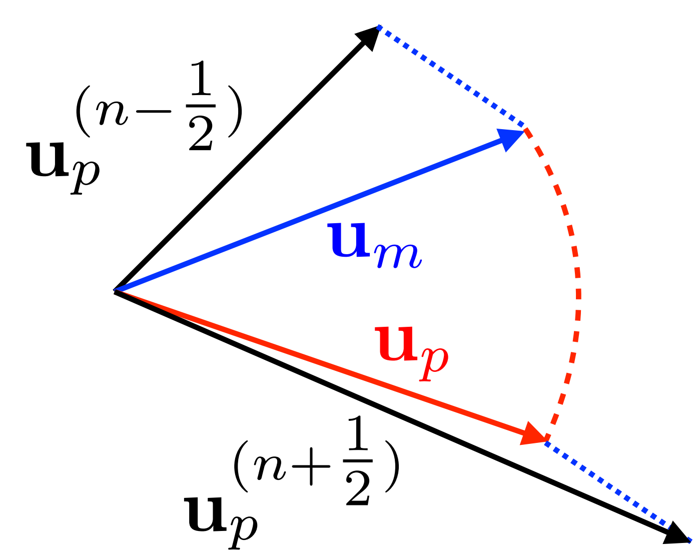
The Boris leap-frog *pusher* is a very popular method to advance particles



$$\mathbf{u}_m = \mathbf{u}_p^{(n-\frac{1}{2})} + \frac{q_s}{m_s}\frac{\Delta t}{2}\mathbf{E}_p$$

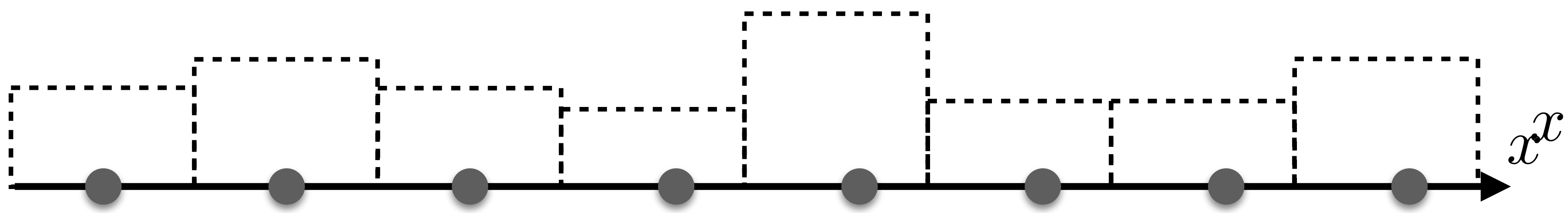
$$\mathbf{u}_p = \mathbf{u}_p^{(n-\frac{1}{2})} + \frac{q_s}{m_s}\Delta t\,\mathcal{M}(\mathbf{B}_p)\,\mathbf{u}_m$$

$$\mathbf{u}_p^{(n+\frac{1}{2})} = \mathbf{u}_p + \frac{q_s}{m_s}\frac{\Delta t}{2}\mathbf{E}_p$$



Step 3

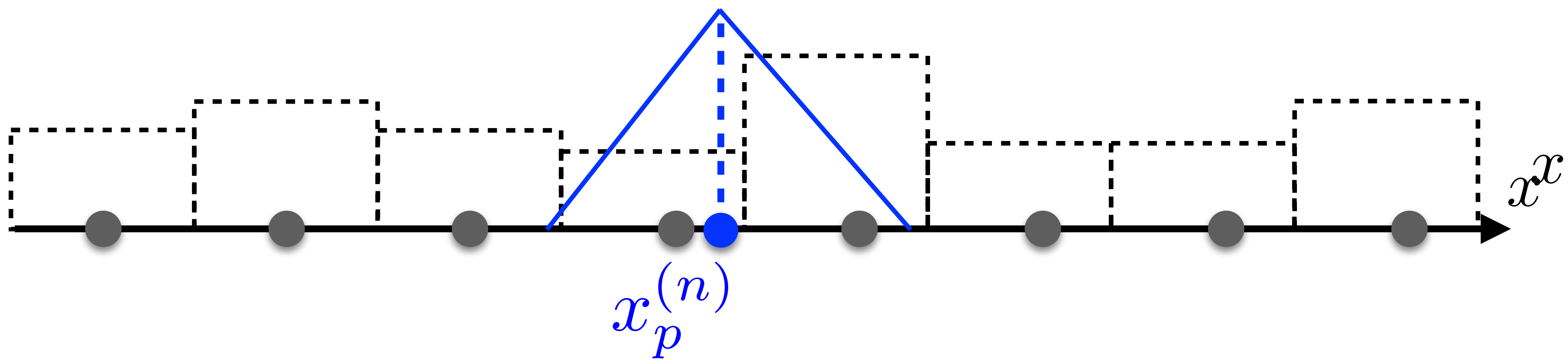
Charge-conserving current deposition scheme are available among which Esirkepov's is 'most' popular



Esirkepov, Comp. Phys. Comm. **135**, 144 (2001)

Step 3

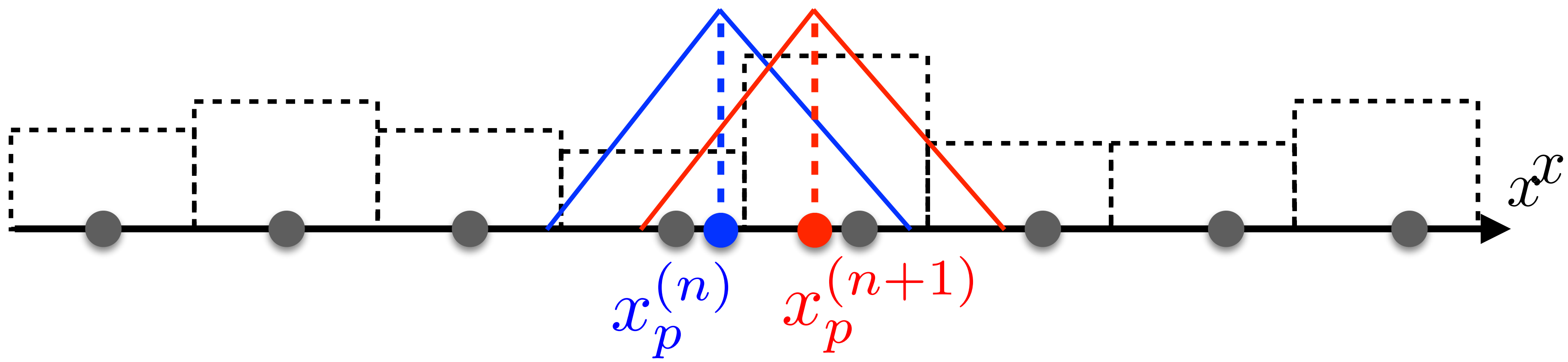
Charge-conserving current deposition scheme are available among which Esirkepov's is 'most' popular



Esirkepov, Comp. Phys. Comm. **135**, 144 (2001)

Step 3

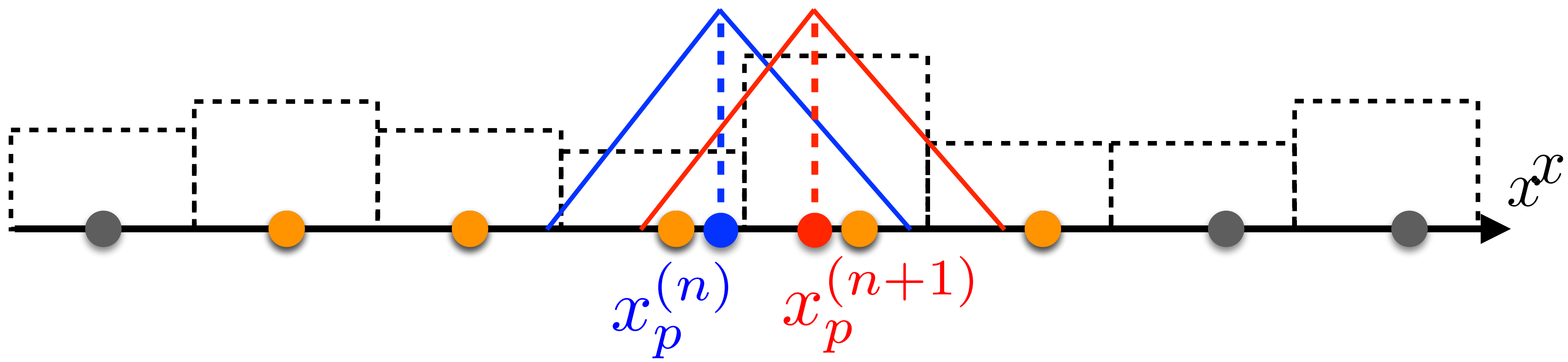
Charge-conserving current deposition scheme are available among which Esirkepov's is 'most' popular



Esirkepov, Comp. Phys. Comm. **135**, 144 (2001)

Step 3

Charge-conserving current deposition scheme are available among which Esirkepov's is 'most' popular



In 1D, current deposition is easily done directly from charge conservation:

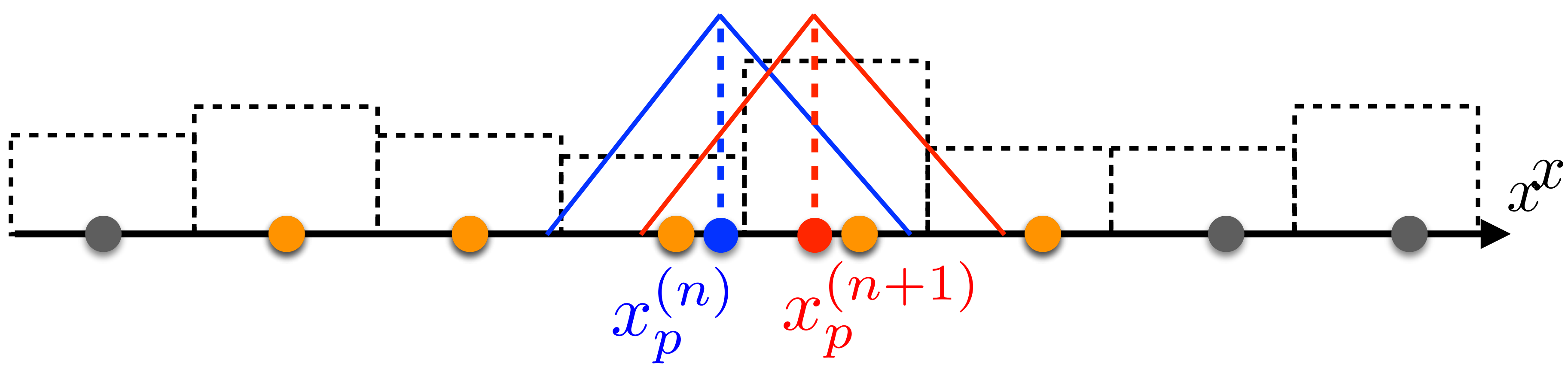
$$\partial_x J_x = -\partial_t \rho$$

while other component are 'directly' projected onto the grid (see interpolation)

Esirkepov, Comp. Phys. Comm. **135**, 144 (2001)

Step 3

Charge-conserving current deposition scheme are available among which Esirkepov's is 'most' popular

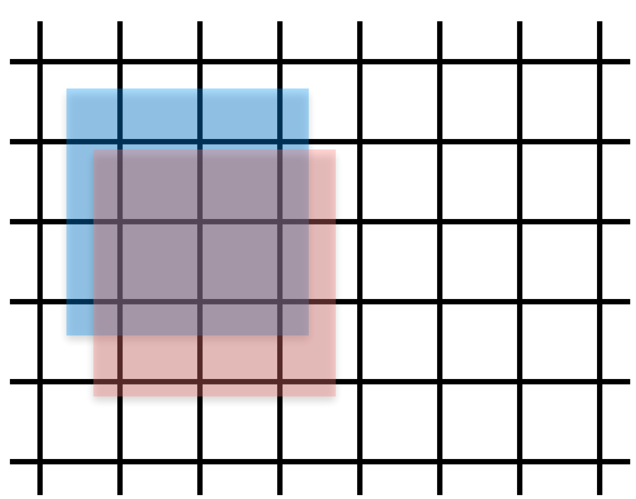


In 1D, current deposition is easily done directly from charge conservation:

$$\partial_x J_x = -\partial_t \rho$$

while other component are 'directly' projected onto the grid (see interpolation)

In 2D & 3D, Esirkepov's method allows to conserve charge (within machine presicion)



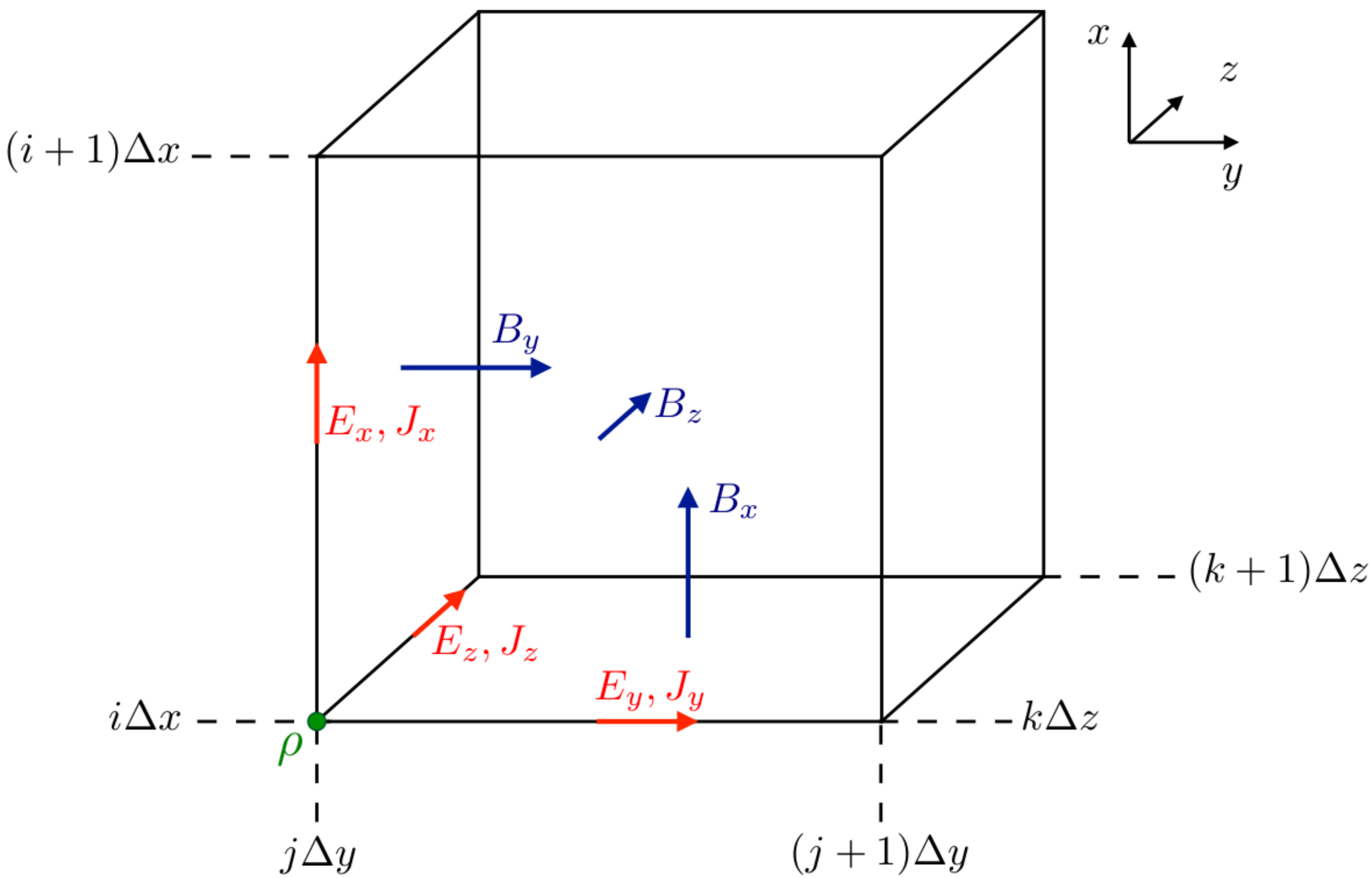
$$(J_{x,p})_{i+\frac{1}{2},j}^{(n+\frac{1}{2})} = (J_{x,p})_{i-\frac{1}{2},j}^{(n+\frac{1}{2})} + q_s\, w_p\, \frac{\Delta x}{\Delta t}\, (W_x)_{i+\frac{1}{2},j}^{(n+\frac{1}{2})}$$

$$(J_{y,p})_{i,j+\frac{1}{2}}^{(n+\frac{1}{2})} = (J_{y,p})_{i,j-\frac{1}{2}}^{(n+\frac{1}{2})} + q_s\, w_p\, \frac{\Delta y}{\Delta t}\, (W_y)_{j,i+\frac{1}{2}}^{(n+\frac{1}{2})}$$

Esirkepov, Comp. Phys. Comm. **135**, 144 (2001)

Step 4

The Finite-Difference Time-Domain (FDTD) method is a popular method for solving Maxwell's Equations



A. Taflove, *Computation electrodynamics: The finite-difference time-domain method*, 3rd Ed. (2005)

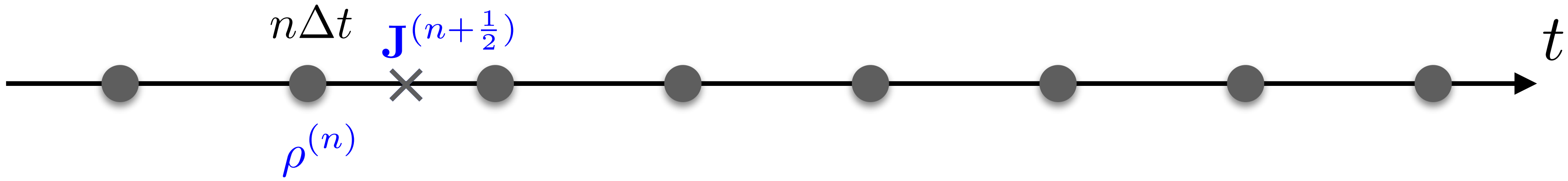
Step 4

The Finite-Difference Time-Domain (FDTD) method is a popular method for solving Maxwell's Equations

A. Taflove, *Computation electrodynamics: The finite-difference time-domain method*, 3rd Ed. (2005)

Step 4

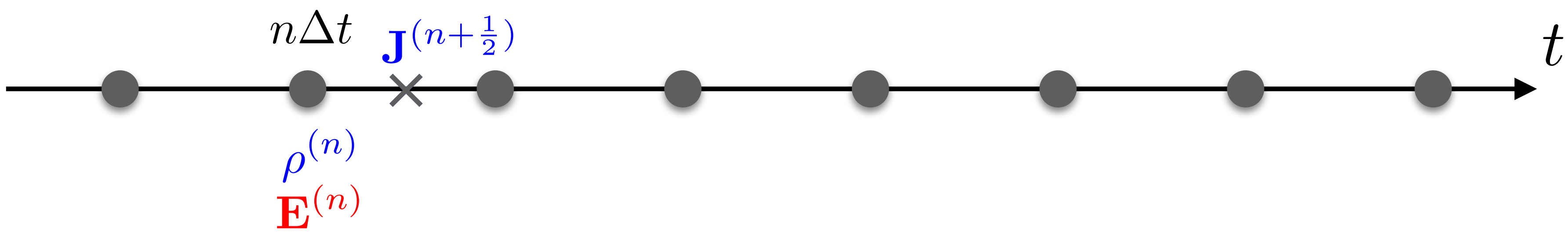
The Finite-Difference Time-Domain (FDTD) method is a popular method for solving Maxwell's Equations



A. Taflove, *Computation electrodynamics: The finite-difference time-domain method*, 3rd Ed. (2005)

Step 4

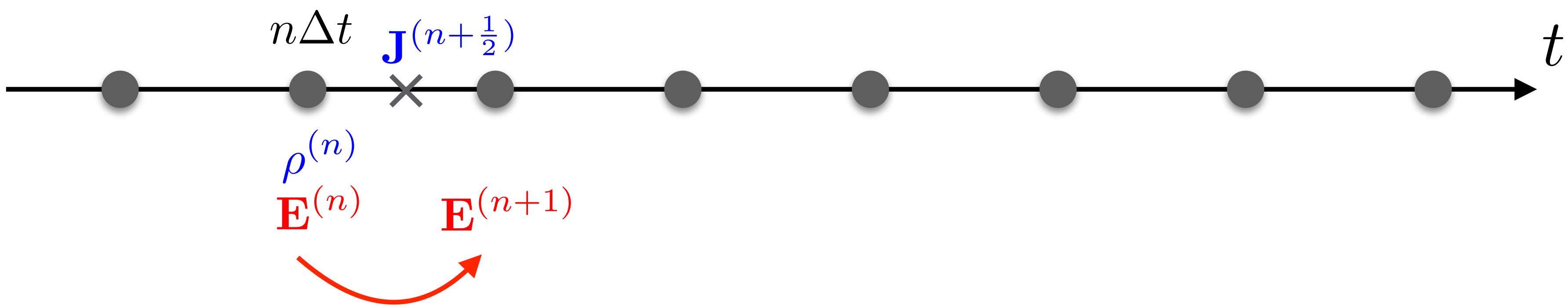
The Finite-Difference Time-Domain (FDTD) method is a popular method for solving Maxwell's Equations



A. Taflove, *Computation electrodynamics: The finite-difference time-domain method*, 3rd Ed. (2005)

Step 4

The Finite-Difference Time-Domain (FDTD) method is a popular method for solving Maxwell's Equations

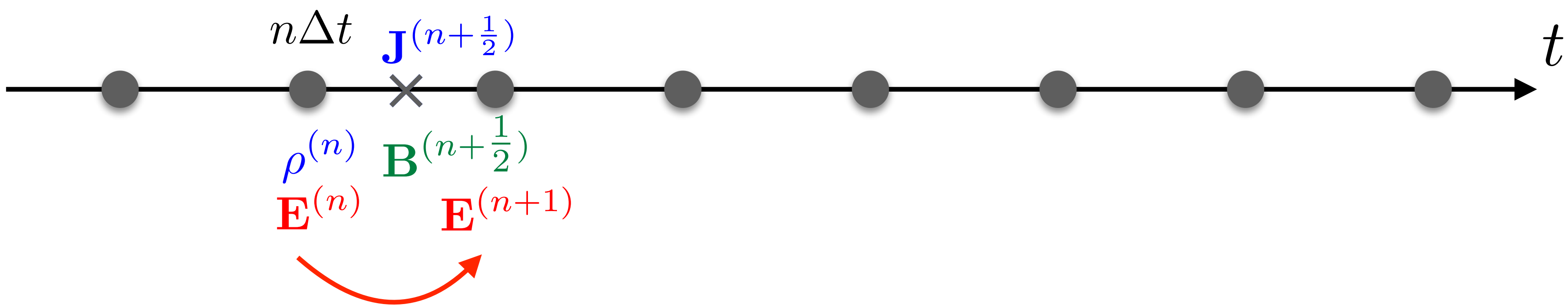


Solving Ampère's equation: $\partial_t E_y = -J_y - \partial_x B_z$

A. Taflove, *Computation electrodynamics: The finite-difference time-domain method*, 3rd Ed. (2005)

Step 4

The Finite-Difference Time-Domain (FDTD) method is a popular method for solving Maxwell's Equations



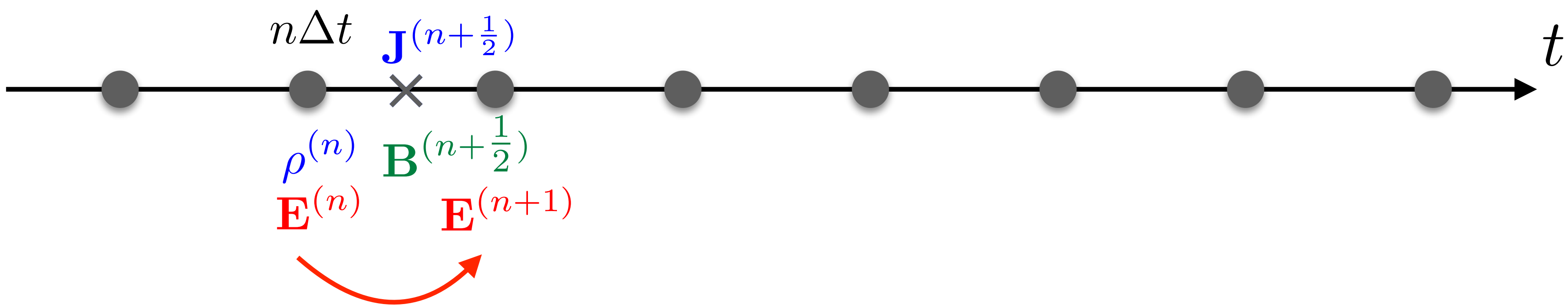
Solving Ampère's equation: $\partial_t E_y = -J_y - \partial_x B_z$

time-centering
$$\frac{(E_y)^{(n+1)} - (E_y)^{(n)}}{\Delta t} = -J_y^{(n+\frac{1}{2})} - (\partial_x B_z)^{(n+\frac{1}{2})}$$

A. Taflove, *Computation electrodynamics: The finite-difference time-domain method*, 3rd Ed. (2005)

Step 4

The Finite-Difference Time-Domain (FDTD) method is a popular method for solving Maxwell's Equations



Solving Ampère's equation: $\partial_t E_y = -J_y - \partial_x B_z$

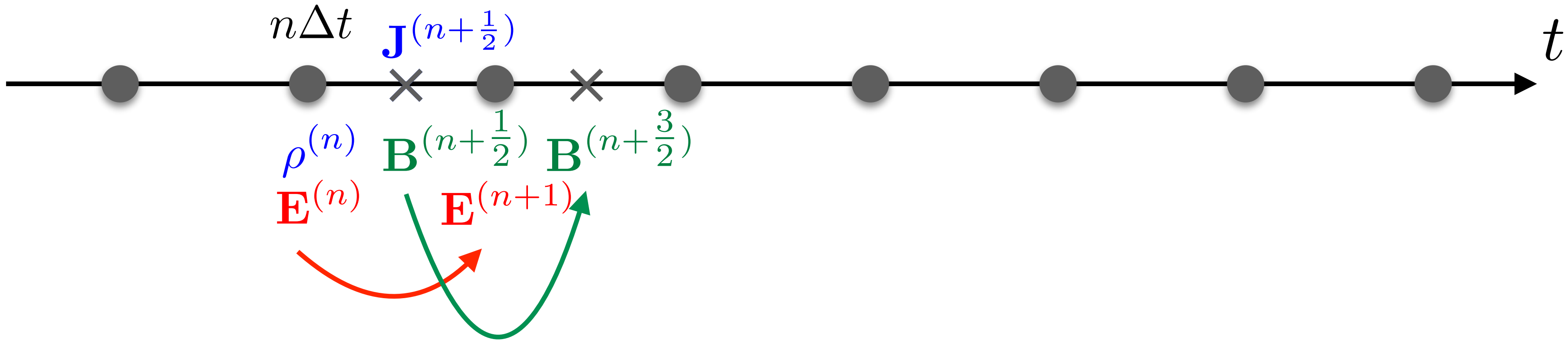
time-centering
$$\frac{(E_y)^{(n+1)} - (E_y)^{(n)}}{\Delta t} = -J_y^{(n+\frac{1}{2})} - (\partial_x B_z)^{(n+\frac{1}{2})}$$

space-centering
$$\frac{(E_y)_i^{(n+1)} - (E_y)_i^{(n)}}{\Delta t} = -(J_y)_i^{(n+\frac{1}{2})} - \frac{(B_z)_{i+\frac{1}{2}}^{(n+\frac{1}{2})} - (B_z)_{i+\frac{1}{2}}^{(n-\frac{1}{2})}}{\Delta x}$$

A. Taflove, *Computation electrodynamics: The finite-difference time-domain method*, 3rd Ed. (2005)

Step 4

The Finite-Difference Time-Domain (FDTD) method is a popular method for solving Maxwell's Equations



Solving Ampère's equation: $\partial_t E_y = -J_y - \partial_x B_z$

time-centering $$\frac{(E_y)^{(n+1)} - (E_y)^{(n)}}{\Delta t} = -J_y^{(n+\frac{1}{2})} - (\partial_x B_z)^{(n+\frac{1}{2})}$$

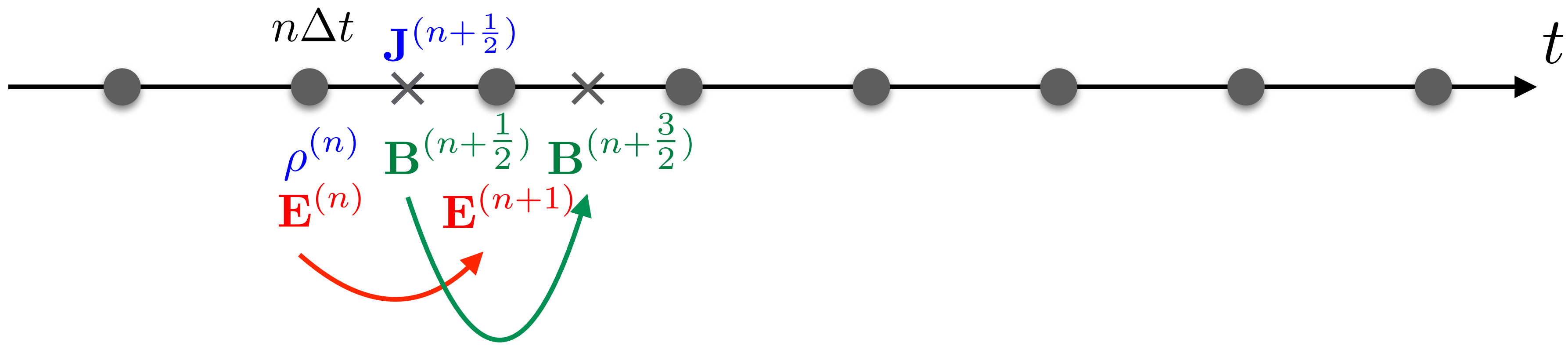
space-centering $$\frac{(E_y)_i^{(n+1)} - (E_y)_i^{(n)}}{\Delta t} = -(J_y)_i^{(n+\frac{1}{2})} - \frac{(B_z)_{i+\frac{1}{2}}^{(n+\frac{1}{2})} - (B_z)_{i+\frac{1}{2}}^{(n-\frac{1}{2})}}{\Delta x}$$

Solving Faraday's equation: $\partial_t B_z = \partial_x E_y$

A. Taflove, *Computation electrodynamics: The finite-difference time-domain method*, 3rd Ed. (2005)

Step 4

The Finite-Difference Time-Domain (FDTD) method is a popular method for solving Maxwell's Equations



Solving Ampère's equation: $\partial_t E_y = -J_y - \partial_x B_z$

time-centering
$$\frac{(E_y)^{(n+1)} - (E_y)^{(n)}}{\Delta t} = -J_y^{(n+\frac{1}{2})} - (\partial_x B_z)^{(n+\frac{1}{2})}$$

space-centering
$$\frac{(E_y)_i^{(n+1)} - (E_y)_i^{(n)}}{\Delta t} = -(J_y)_i^{(n+\frac{1}{2})} - \frac{(B_z)_{i+\frac{1}{2}}^{(n+\frac{1}{2})} - (B_z)_{i+\frac{1}{2}}^{(n-\frac{1}{2})}}{\Delta x}$$

Solving Faraday's equation: $\partial_t B_z = \partial_x E_y$

space/time-centering
$$\frac{(B_z)_{i+\frac{1}{2}}^{(n+\frac{3}{2})} - (B_z)_{i+\frac{1}{2}}^{(n+\frac{1}{2})}}{\Delta t} = \frac{(E_y)_{i+1}^{(n+1)} - (E_y)_i^{(n+1)}}{\Delta x}$$

A. Taflove, *Computation electrodynamics: The finite-difference time-domain method*, 3rd Ed. (2005)

Step 4

Numerical analysis of the FDTD solvers gives you access to the numerical dispersion relation & CFL condition

Nuter et al., Eur. Phys. J. D (2014); All papers by B. Godfrey, from the 70's up to now !!!

Step 4

Numerical analysis of the FDTD solvers gives you access to the numerical dispersion relation & CFL condition

The *numerical* electromagnetic wave equation in a vacuum

$$\partial_t^N \mathbf{E} = +\nabla^N \times \mathbf{B}$$

$$\partial_t^N \mathbf{B} = -\nabla^N \times \mathbf{E}$$

Nuter et al., Eur. Phys. J. D (2014); All papers by B. Godfrey, from the 70's up to now !!!

Step 4

Numerical analysis of the FDTD solvers gives you access to the numerical dispersion relation & CFL condition

The *numerical* electromagnetic wave equation in a vacuum

$$\partial_t^N \mathbf{E} = +\nabla^N \times \mathbf{B}$$

$$\partial_t^N \mathbf{B} = -\nabla^N \times \mathbf{E}$$

with:

$$\partial_t^N F = \Delta t^{-1} \left[F^{(n+\frac{1}{2})} - F^{(n-\frac{1}{2})} \right]$$

$$\partial_\mu^N F = \Delta \mu^{-1} \left[F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}} \right]$$

Nuter et al., Eur. Phys. J. D (2014); All papers by B. Godfrey, from the 70's up to now !!!

Step 4

Numerical analysis of the FDTD solvers gives you access to the numerical dispersion relation & CFL condition

The *numerical* electromagnetic wave equation in a vacuum

$$\partial_t^N \mathbf{E} = +\nabla^N \times \mathbf{B} \qquad \qquad \partial_t^N F = \Delta t^{-1}\left[F^{(n+\frac{1}{2})} - F^{(n-\frac{1}{2})}\right]$$

with:

$$\partial_t^N \mathbf{B} = -\nabla^N \times \mathbf{E} \qquad \qquad \partial_\mu^N F = \Delta \mu^{-1}\left[F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}\right]$$

Using the standard technique to derive the wave equation leads to:

$$\partial_{tt}^N \mathbf{E} + \sum_\mu \partial_{\mu\mu}^N \mathbf{E} = 0$$

Nuter et al., Eur. Phys. J. D (2014); All papers by B. Godfrey, from the 70's up to now !!!

Step 4

Numerical analysis of the FDTD solvers gives you access to the numerical dispersion relation & CFL condition

The *numerical* electromagnetic wave equation in a vacuum

$$\partial_t^N \mathbf{E} = +\nabla^N \times \mathbf{B}$$
$$\partial_t^N \mathbf{B} = -\nabla^N \times \mathbf{E}$$

with:

$$\partial_t^N F = \Delta t^{-1}\left[F^{(n+\frac{1}{2})} - F^{(n-\frac{1}{2})}\right]$$
$$\partial_\mu^N F = \Delta \mu^{-1}\left[F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}\right]$$

Using the standard technique to derive the wave equation leads to:

$$\partial_{tt}^N \mathbf{E} + \sum_\mu \partial_{\mu\mu}^N \mathbf{E} = 0$$

Looking for *numerical* solution in the form:

$$(E_y)^{(n)}_{i,j+\frac{1}{2},k} = E_{y0}\,\exp\{\imath\left[ik_x\Delta x + (j+\tfrac{1}{2})k_y\Delta y \right.$$
$$\left. + kk_z\Delta z - n\omega\Delta t\right]\}$$

Nuter et al., Eur. Phys. J. D (2014); All papers by B. Godfrey, from the 70's up to now !!!

Step 4

Numerical analysis of the FDTD solvers gives you access to the numerical dispersion relation & CFL condition

Nuter et al., Eur. Phys. J. D (2014); All papers by B. Godfrey, from the 70's up to now !!!

Step 4

Numerical analysis of the FDTD solvers gives you access to the numerical dispersion relation & CFL condition

After some algebra, one finds the *numerical dispersion relation*:

$$\frac{\sin^2\left(\omega\Delta t/2\right)}{\Delta t^2} = \sum_{\mu} \frac{\sin^2\left(k_\mu \Delta\mu/2\right)}{\Delta\mu^2}$$

Nuter et al., Eur. Phys. J. D (2014); All papers by B. Godfrey, from the 70's up to now !!!

Step 4

Numerical analysis of the FDTD solvers gives you access to the numerical dispersion relation & CFL condition

After some algebra, one finds the *numerical dispersion relation*:

$$\frac{\sin^2\left(\omega\Delta t/2\right)}{\Delta t^2} = \sum_\mu \frac{\sin^2\left(k_\mu \Delta\mu/2\right)}{\Delta\mu^2}$$

There exists a stability condition: Courant-Friedrich-Lewy (CFL)

$$\Delta t < \sum_\mu \left(\Delta\mu^{-2}\right)^{-1/2}$$

Nuter et al., Eur. Phys. J. D (2014);   All papers by B. Godfrey, from the 70's up to now !!!

Step 4

Numerical analysis of the FDTD solvers gives you access to the numerical dispersion relation & CFL condition

After some algebra, one finds the *numerical dispersion relation*:

$$\frac{\sin^2\left(\omega\Delta t/2\right)}{\Delta t^2} = \sum_\mu \frac{\sin^2\left(k_\mu \Delta\mu/2\right)}{\Delta\mu^2}$$

There exists a stability condition: Courant-Friedrich-Lewy (CFL)

$$\Delta t < \sum_\mu \left(\Delta\mu^{-2}\right)^{-1/2} \xrightarrow[\Delta x = \Delta y]{\text{2D}} \Delta t < \Delta x/\sqrt{2}$$

Nuter et al., Eur. Phys. J. D (2014); All papers by B. Godfrey, from the 70's up to now !!!

Step 4

Numerical analysis of the FDTD solvers gives you access to the numerical dispersion relation & CFL condition

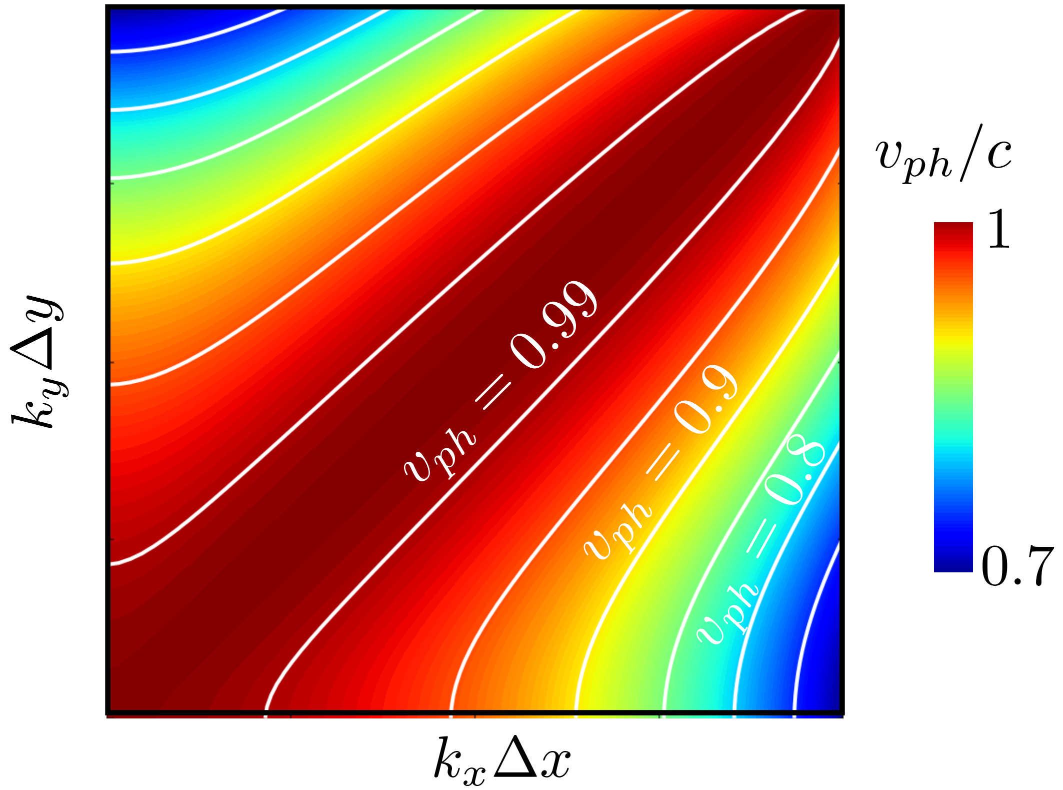
After some algebra, one finds the *numerical dispersion relation*:

$$\frac{\sin^2(\omega\Delta t/2)}{\Delta t^2} = \sum_\mu \frac{\sin^2(k_\mu \Delta\mu/2)}{\Delta\mu^2}$$

There exists a stability condition: Courant-Friedrich-Lewy (CFL)

$$\Delta t < \sum_\mu \left(\Delta\mu^{-2}\right)^{-1/2} \xrightarrow[\Delta x = \Delta y]{\text{2D}} \Delta t < \Delta x/\sqrt{2}$$

The FDTD solver is subject to *numerical dispersion* as the numerical light wave velocity is found to depend on its wavenumber and orientation.



Nuter et al., Eur. Phys. J. D (2014); All papers by B. Godfrey, from the 70's up to now !!!

A quick summary

The PIC approach in a nutshell

Initialization	time step $n = 0$, time $t = 0$

Particle loading $\quad \forall p$, define $(\mathbf{x}_p)^{n=0}$, $(\mathbf{u}_p)^{n=-\frac{1}{2}}$

Charge projection on grid $\quad \left[\forall p, (\mathbf{x}_p)^{n=0}\right] \rightarrow \rho^{(n=0)}(\mathbf{x})$

Compute initial fields

- solve Poisson on grid: $\left[\rho^{(n=0)}(\mathbf{x})\right] \rightarrow \mathbf{E}^{(n=0)}_{\text{stat}}(\mathbf{x})$
- add external fields: $\mathbf{E}^{(n=0)}(\mathbf{x}) = \mathbf{E}^{(n=0)}_{\text{stat}}(\mathbf{x}) + \mathbf{E}^{(n=0)}_{\text{ext}}(\mathbf{x})$

$$\mathbf{B}^{(n=\frac{1}{2})}(\mathbf{x}) = \mathbf{B}^{(n=\frac{1}{2})}_{\text{ext}}(\mathbf{x})$$

PIC loop: from time step n to $n+1$, time $t = (n+1)\,\Delta t$

Restart charge & current densities
Save magnetic fields value (used to center magnetic fields)

Interpolate fields at particle positions $\quad \forall p, \left[\mathbf{x}_p, \mathbf{E}^{(n)}(\mathbf{x}), \mathbf{B}^{(n)}(\mathbf{x})\right] \rightarrow \mathbf{E}^{(n)}_p, \mathbf{B}^{(n)}_p$

Push particles

- compute new velocity $\forall p$, $\mathbf{p}_p^{(n-\frac{1}{2})} \left[\mathbf{E}^{(n)}_p, \mathbf{B}^{(n)}_p\right] \mathbf{p}_p^{(n+\frac{1}{2})}$
- compute new position $\forall p$, $\mathbf{x}_p^{(n)} \left[\mathbf{p}_p^{(n+\frac{1}{2})}\right] \mathbf{x}_p^{(n+1)}$

Project current onto the grid using a charge-conserving scheme

$$\left[\forall p \;\; \mathbf{x}_p^{(n)}, \mathbf{x}_p^{(n+1)}, \mathbf{p}_p^{(n+\frac{1}{2})}\right] \rightarrow \mathbf{J}^{(n+\frac{1}{2})}(\mathbf{x})$$

Solve Maxwell's equations

- solve Maxwell-Faraday: $\mathbf{E}^{(n)}(\mathbf{x}) \left[\mathbf{J}^{(n+\frac{1}{2})(\mathbf{x})}\right] \mathbf{E}^{(n+1)}(\mathbf{x})$
- solve Maxwell-Ampère: $\mathbf{B}^{(n+\frac{1}{2})}(\mathbf{x}) \left[\mathbf{E}^{(n+1)}(\mathbf{x})\right] \mathbf{B}^{(n+\frac{3}{2})}(\mathbf{x})$
- center magnetic fields: $\mathbf{B}^{(n+1)}(\mathbf{x}) = \frac{1}{2}\left(\mathbf{B}^{(n+\frac{1}{2})}(\mathbf{x}) + \mathbf{B}^{(n+\frac{3}{2})}(\mathbf{x})\right)$

There are still a few things to know before running your first PIC simulation

There are still a few things to know before running your first PIC simulation

- **noise is inherent to PIC code**
  electromagnetic fluctuations inherent to a thermal plasma
  the level of noise is however much exaggerated in PIC codes

There are still a few things to know before running your first PIC simulation

- **noise is inherent to PIC code**
  electromagnetic fluctuations inherent to a thermal plasma
  the level of noise is however much exaggerated in PIC codes

- **some numerical instabilities have to be taken care off carefully**
  - numerical heating usually requires $\Delta x \lesssim \lambda_{De}$
  - numerical-Cherenkov can also plague simulation with relativistically drifting particles

There are still a few things to know before running your first PIC simulation

- **noise is inherent to PIC code**
  electromagnetic fluctuations inherent to a thermal plasma
  the level of noise is however much exaggerated in PIC codes

- **some numerical instabilities have to be taken care off carefully**
  - numerical heating usually requires $\Delta x \lesssim \lambda_{De}$
  - numerical-Cherenkov can also plague simulation with relativistically drifting particles

- **PIC codes are usually very robust, beware of your results!**
  A PIC code will most likely not crash, even if your simulation is complete non-sense!

There are still a few things to know before running your first PIC simulation

- **noise is inherent to PIC code**
  electromagnetic fluctuations inherent to a thermal plasma
  the level of noise is however much exaggerated in PIC codes

- **some numerical instabilities have to be taken care off carefully**
  - numerical heating usually requires $\Delta x \lesssim \lambda_{De}$
  - numerical-Cherenkov can also plague simulation with relativistically drifting particles

- **PIC codes are usually very robust, beware of your results!**
  A PIC code will most likely not crash, even if your simulation is complete non-sense!

- **I did not discuss boundary conditions nor ghost-cells**

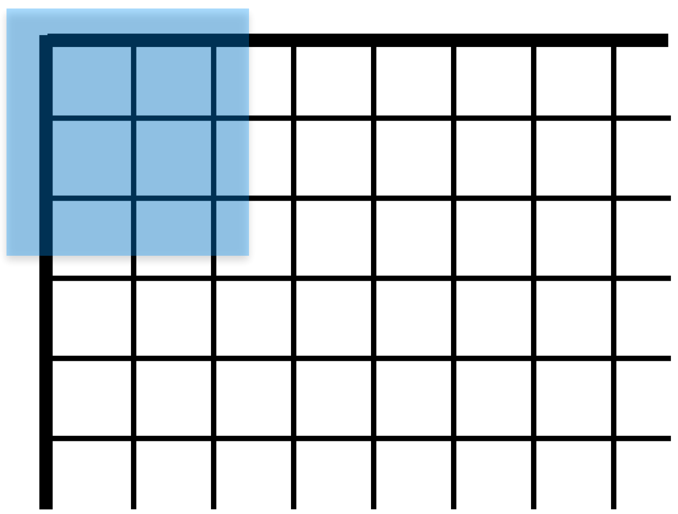
There are still a few things to know before running your first PIC simulation

- **noise is inherent to PIC code**
  electromagnetic fluctuations inherent to a thermal plasma
  the level of noise is however much exaggerated in PIC codes

- **some numerical instabilities have to be taken care off carefully**
  - numerical heating usually requires $\Delta x \lesssim \lambda_{De}$
  - numerical-Cherenkov can also plague simulation with relativistically drifting particles

- **PIC codes are usually very robust, beware of your results!**
  A PIC code will most likely not crash, even if your simulation is complete non-sense!

- **I did not discuss boundary conditions nor ghost-cells**



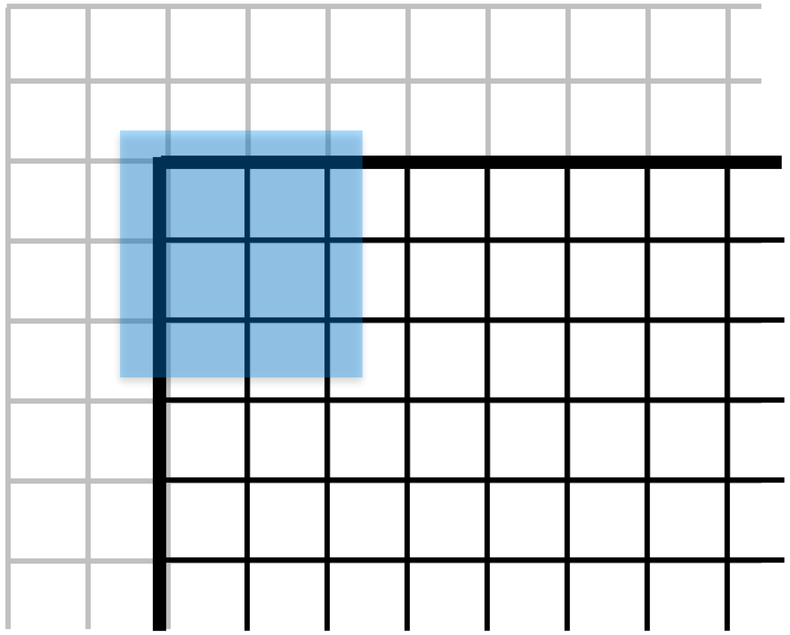
There are still a few things to know before running your first PIC simulation

- **noise is inherent to PIC code**
  electromagnetic fluctuations inherent to a thermal plasma
  the level of noise is however much exaggerated in PIC codes

- **some numerical instabilities have to be taken care off carefully**
  - numerical heating usually requires $\Delta x \lesssim \lambda_{De}$
  - numerical-Cherenkov can also plague simulation with relativistically drifting particles

- **PIC codes are usually very robust, beware of your results!**
  A PIC code will most likely not crash, even if your simulation is complete non-sense!

- **I did not discuss boundary conditions nor ghost-cells**



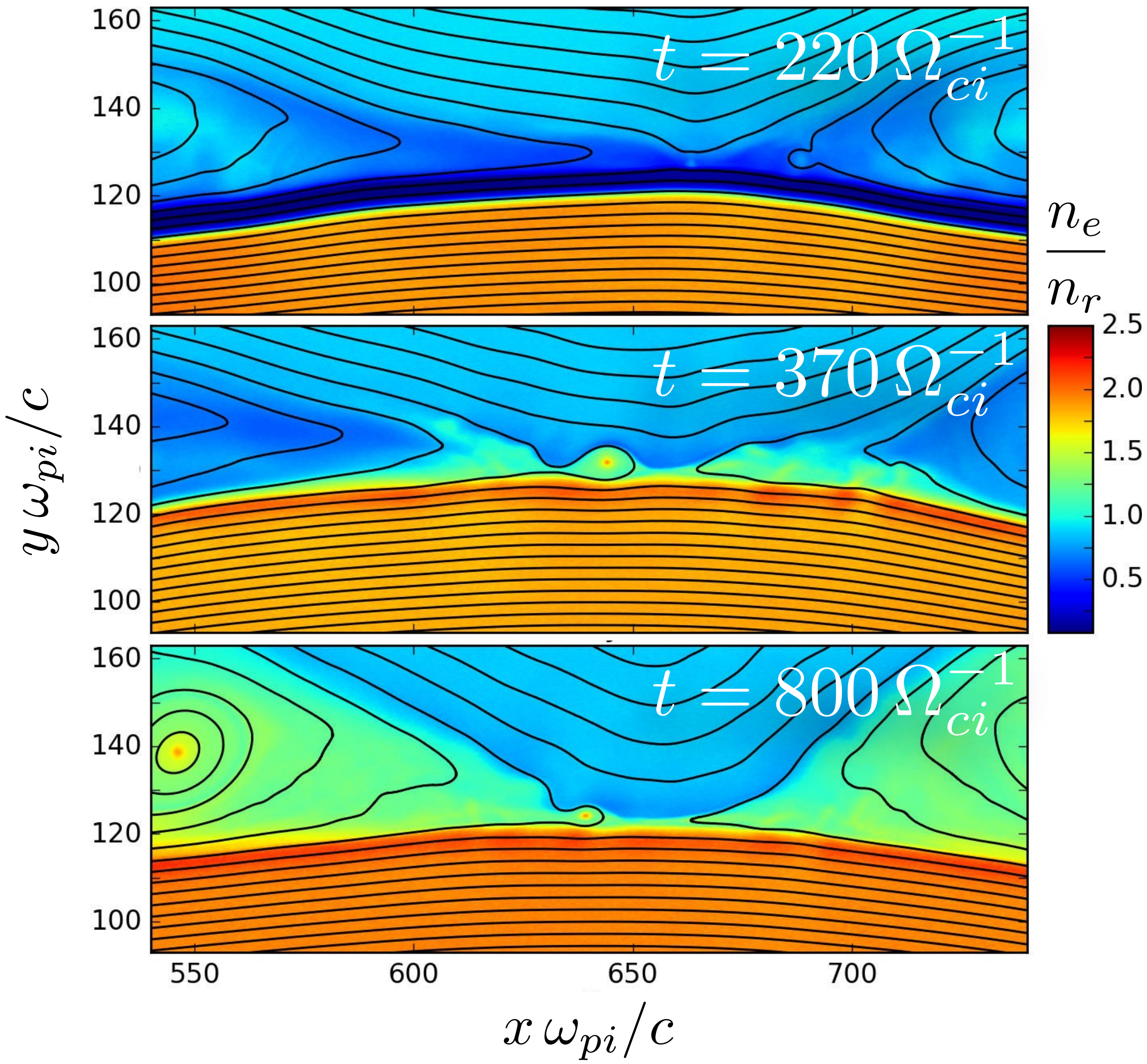
Outlines

- Numerical approach: how to build your PIC code
- High-performance computing:
  getting ready for the super-computers
- Additional modules: beyond the *collisionless* plasma
- Some physics highlights: what you can do with a PIC code

Parallelization is mandatory for large-scale PIC simulation

Parallelization is mandatory for large-scale PIC simulation

Large scale PIC simulation of
magnetic reconnection at the earth magnetopause



Simulation box: $1280\,\frac{c}{\omega_{pi}} \times 256\,\frac{c}{\omega_{pi}}$

25600×10240 PIC cells

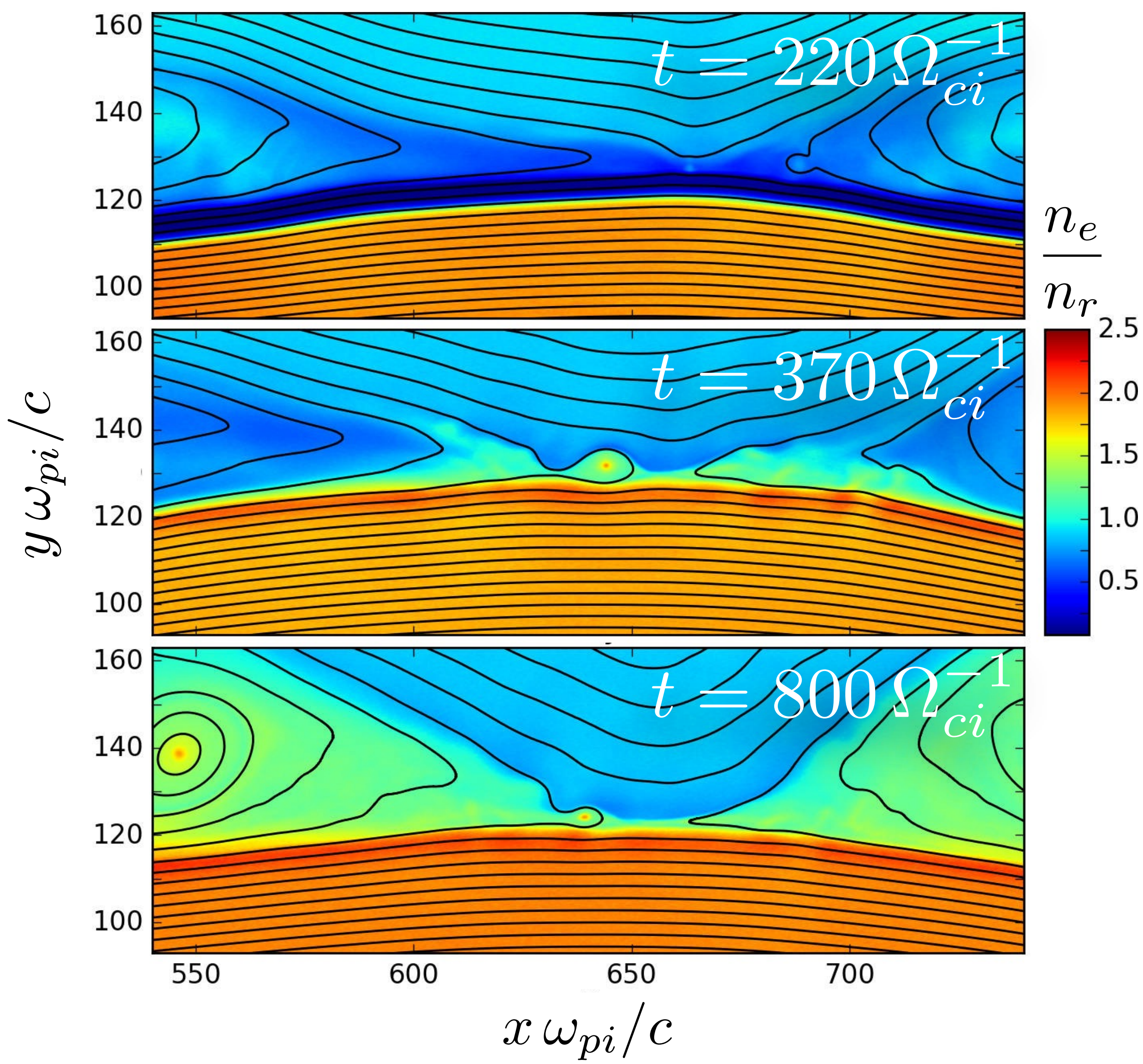
run up to $t = 800\,\Omega_{ci}^{-1}$

$N_t \sim 9.5 \times 10^5$ timesteps

for a total of 22×10^9 quasi-particles.

Parallelization is mandatory for large-scale PIC simulation

Large scale PIC simulation of
magnetic reconnection at the earth magnetopause



Simulation box: $1280\,\frac{c}{\omega_{pi}} \times 256\,\frac{c}{\omega_{pi}}$

25600×10240 PIC cells

run up to $t = 800\,\Omega_{ci}^{-1}$

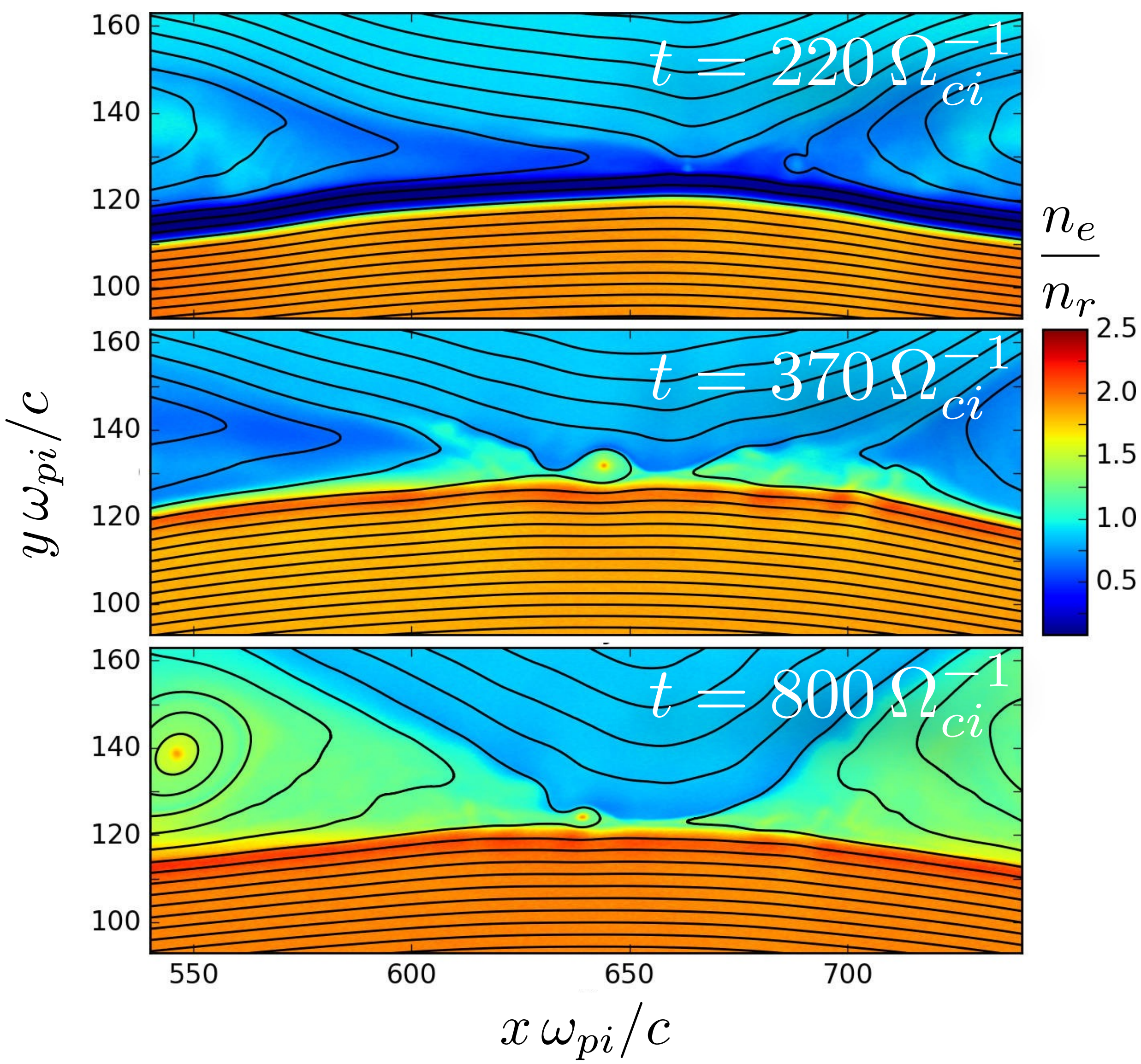
$N_t \sim 9.5 \times 10^5$ timesteps

for a total of 22×10^9 quasi-particles.

Required simulation time:
14 000 000 hours ~ 1600 years!!!

Parallelization is mandatory for large-scale PIC simulation

Large scale PIC simulation of
magnetic reconnection at the earth magnetopause



Simulation box: $1280\,\frac{c}{\omega_{pi}} \times 256\,\frac{c}{\omega_{pi}}$

25600×10240 PIC cells

run up to $t = 800\,\Omega_{ci}^{-1}$

$N_t \sim 9.5 \times 10^5$ timesteps

for a total of 22×10^9 quasi-particles.

Required simulation time:
14 000 000 hours ~ 1600 years!!!

Solution:
share the work on 16384 CPUs !!!

High-performance computing new paradigms & challenges

High-performance computing new paradigms & challenges

Tianhe-2 34 PF:

17 MW

High-performance computing new paradigms & challenges

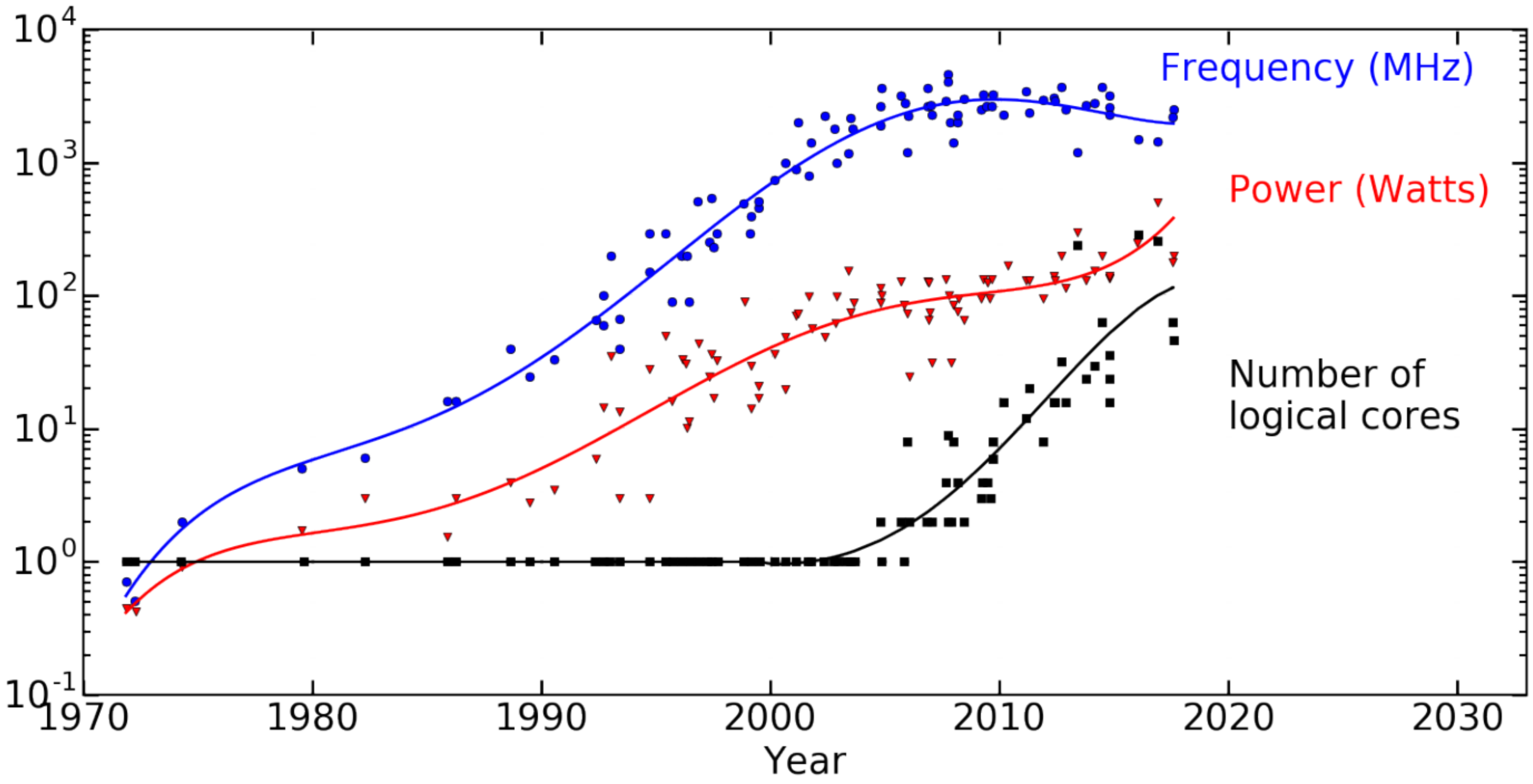
Tianhe-2 **34 PF**: → Exascale **1000 PF**:

17 MW → 500 MW

High-performance computing new paradigms & challenges

Tianhe-2 **34 PF**: → Exascale **1000 PF**:

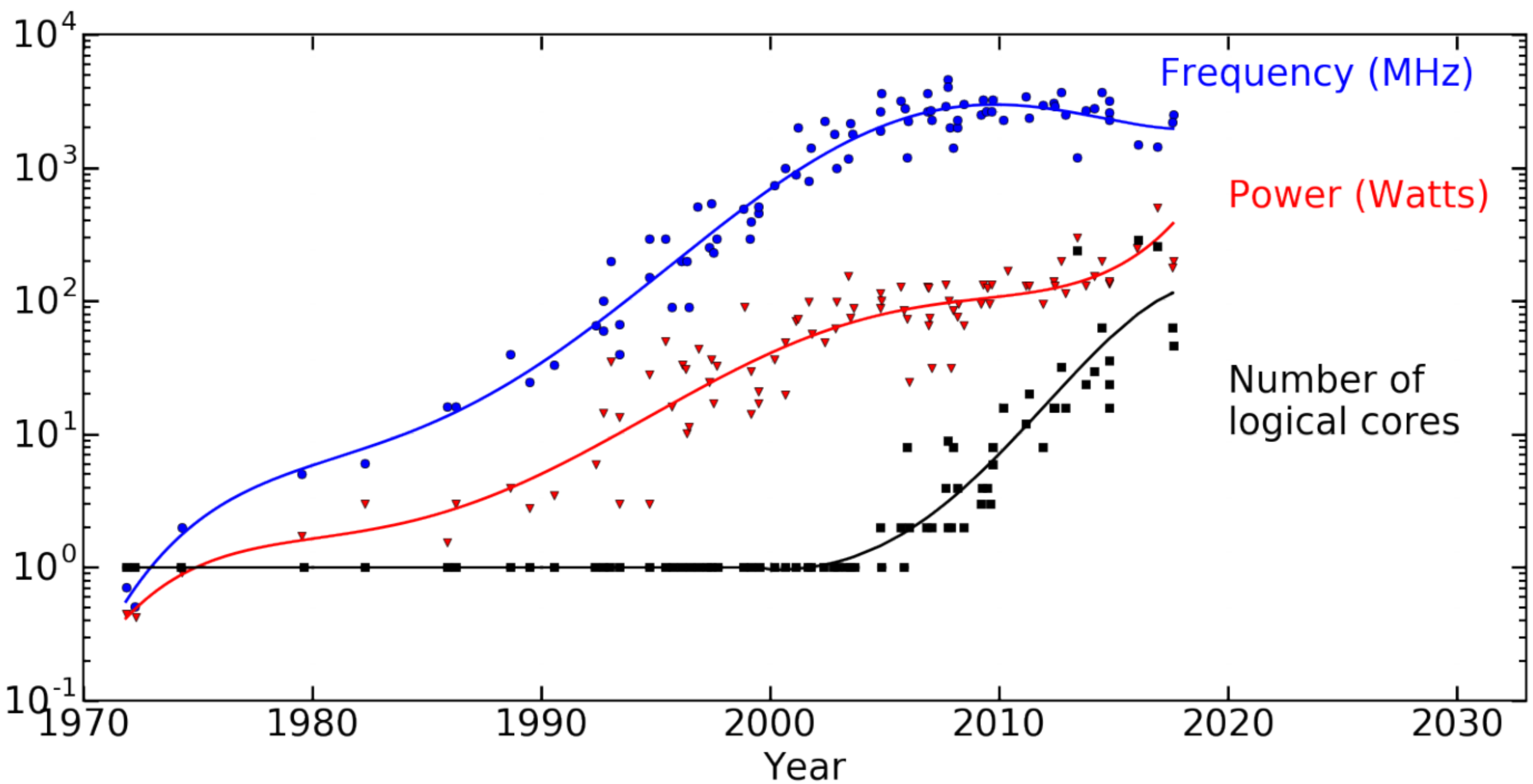
17 MW → 500 MW



High-performance computing new paradigms & challenges

Tianhe-2 **34 PF**: → Exascale **1000 PF**:

17 MW → 500 MW



Parallelism*
- massive
- hybrid MPI-OpenMP
- dynamic (load balance)

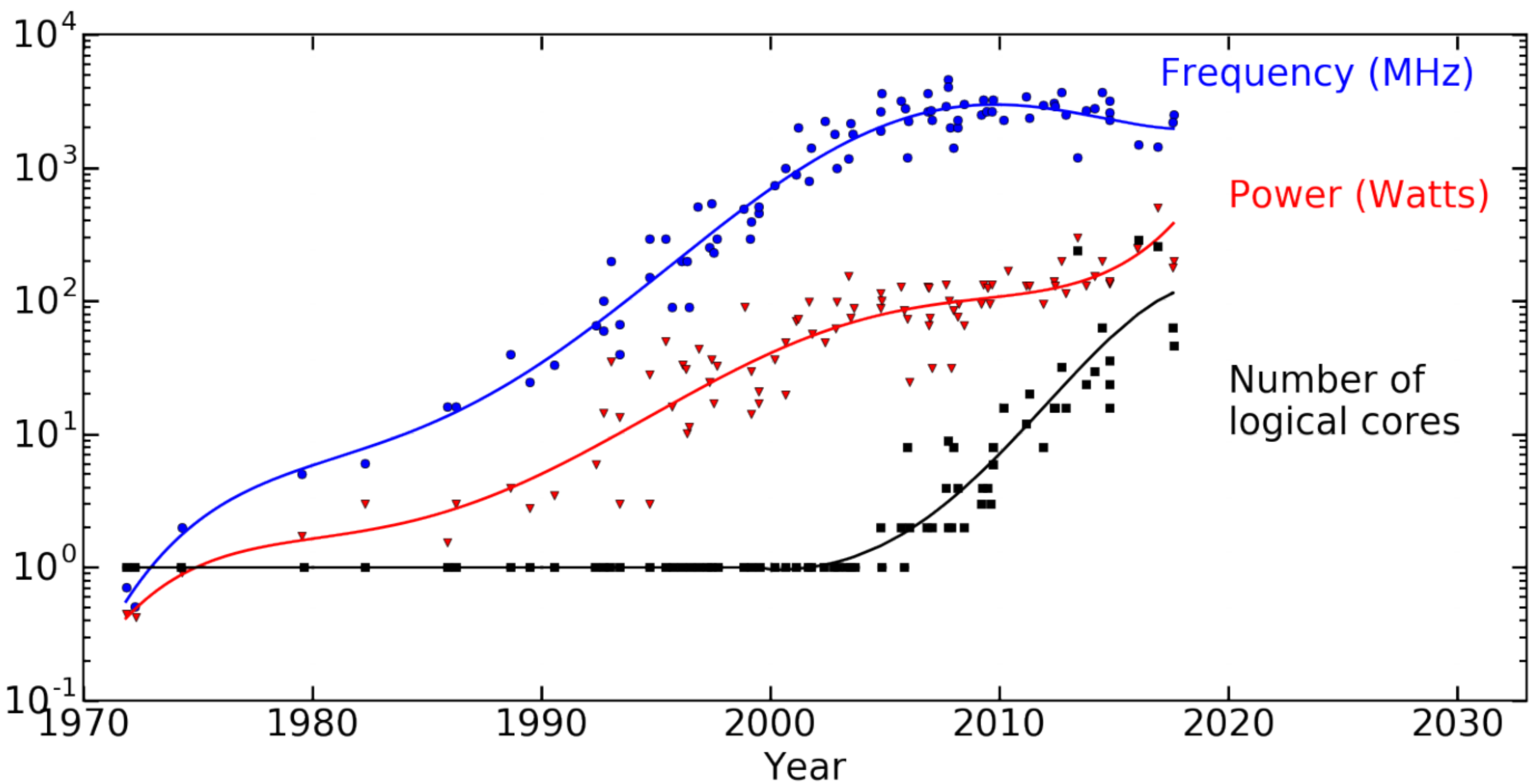
Memory
- shared vs. distributed
- cache use

Vectorization**
- SIMD

Parallel I/O
- hdf5, openPMD

High-performance computing new paradigms & challenges

Tianhe-2 **34 PF**: → Exascale **1000 PF**:
17 MW → 500 MW



Parallelism*
- massive
- hybrid MPI-OpenMP
- dynamic (load balance)

Memory
- shared vs. distributed
- cache use

Vectorization**
- SIMD

Parallel I/O
- hdf5, openPMD

*Derouillat *et al.*, Comp. Phys. Comm. **222**, 351 (2018)
**Beck *et al.*, arXiv:1810.03949

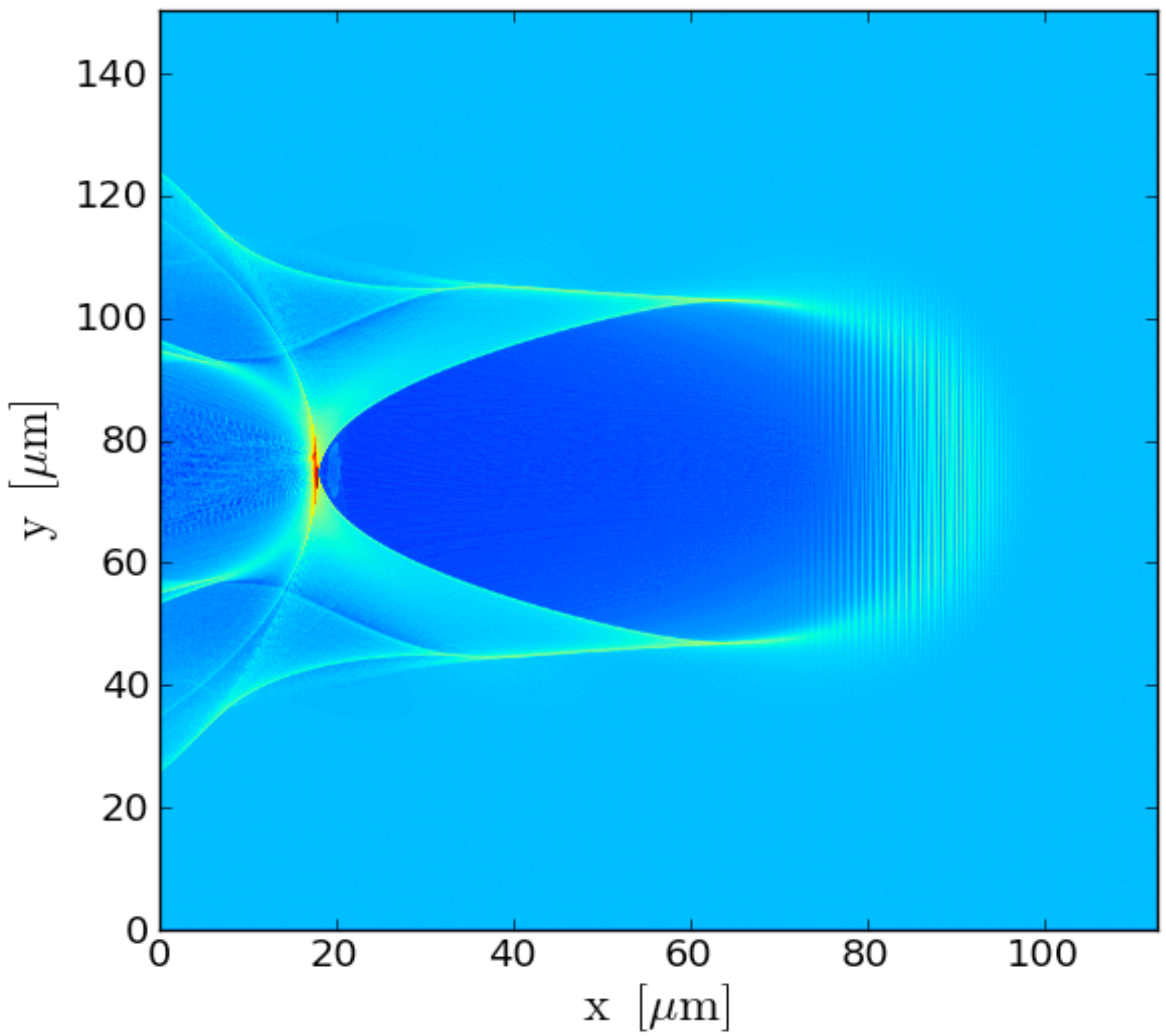
Step 1: Parallelization

PIC codes are well adapted to massive parallelism

Step 1: Parallelization

PIC codes are well adapted to massive parallelism

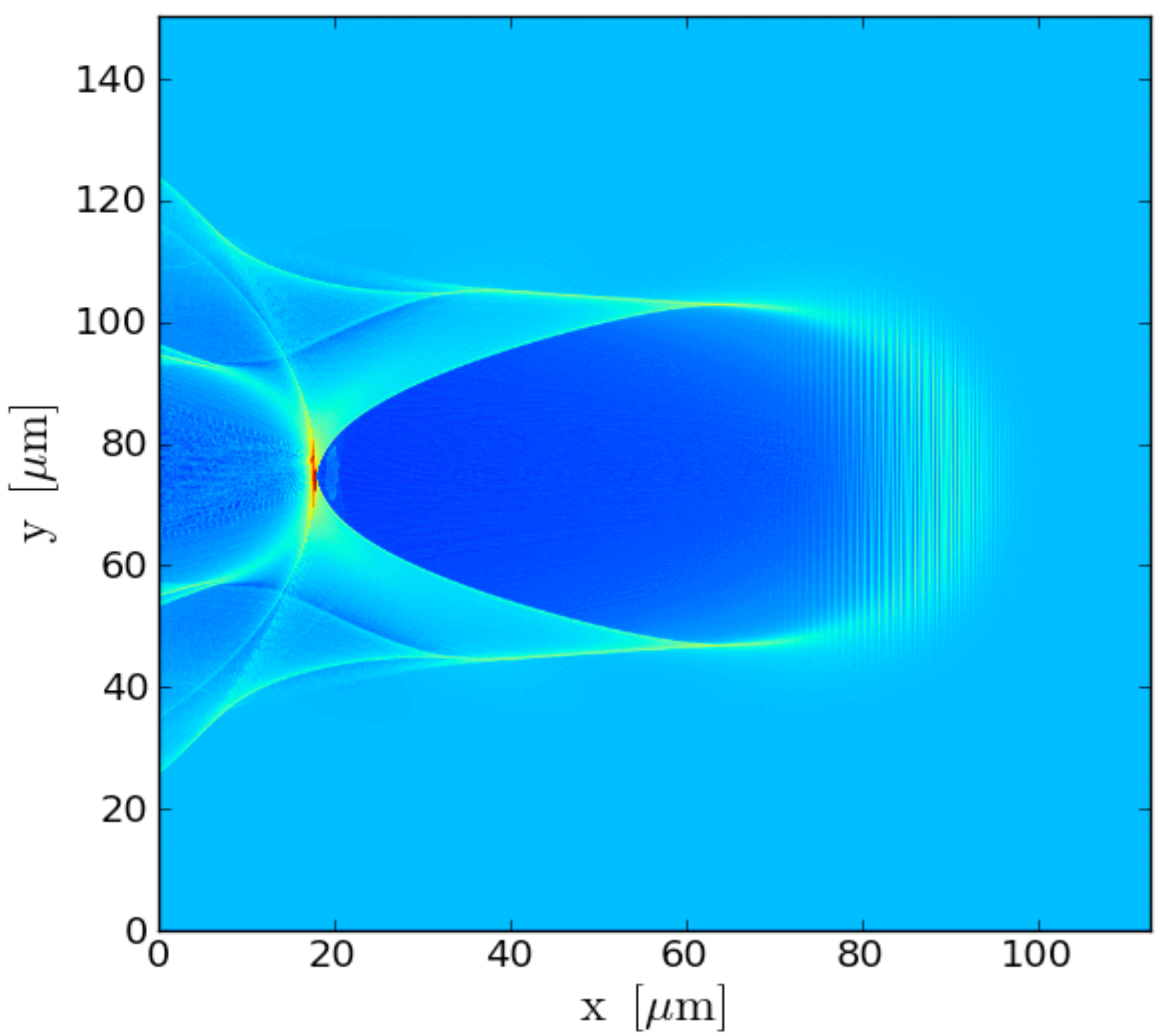
My Simulation (LWFA)

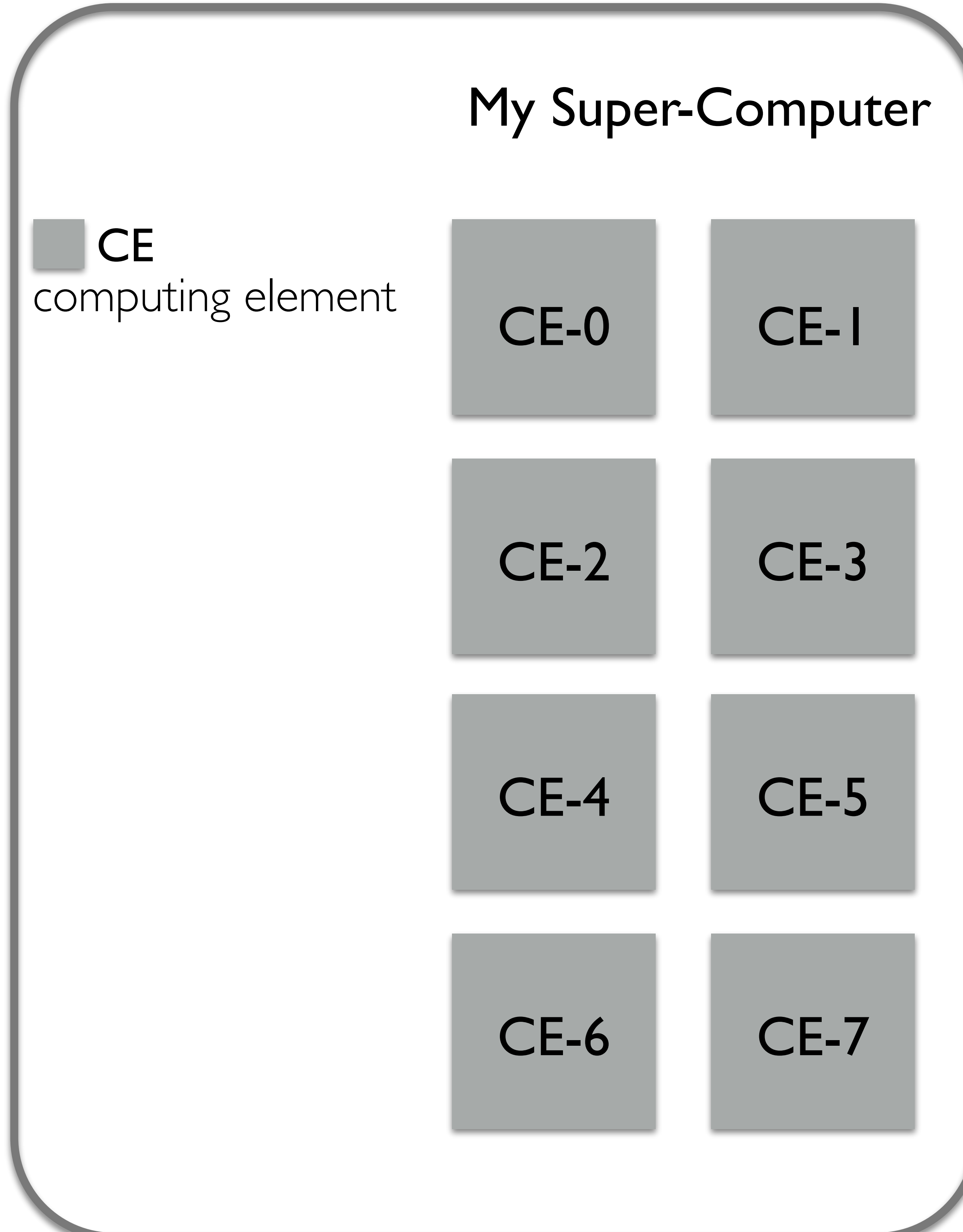


Step 1: Parallelization

PIC codes are well adapted to massive parallelism

My Simulation (LWFA)

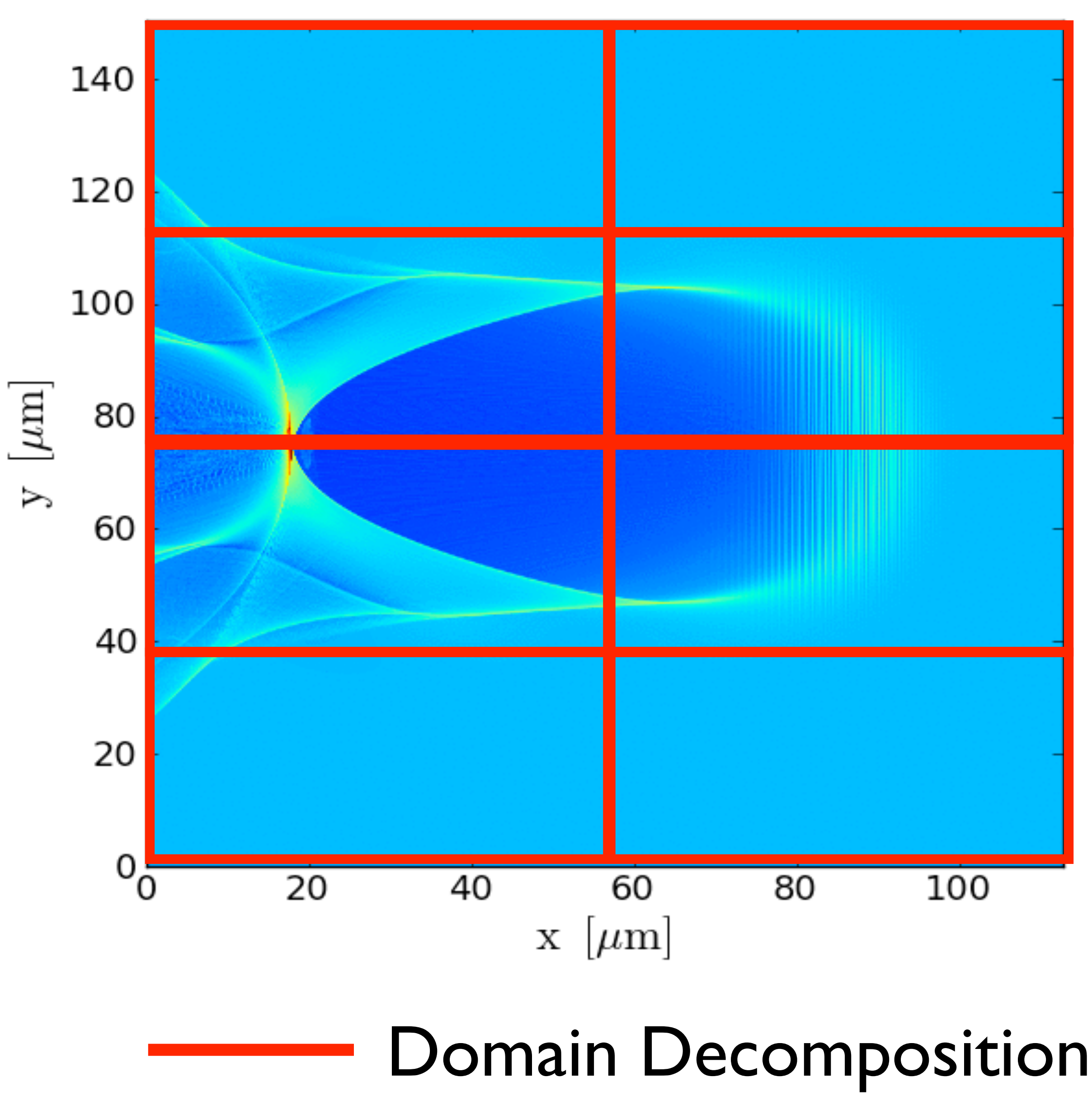


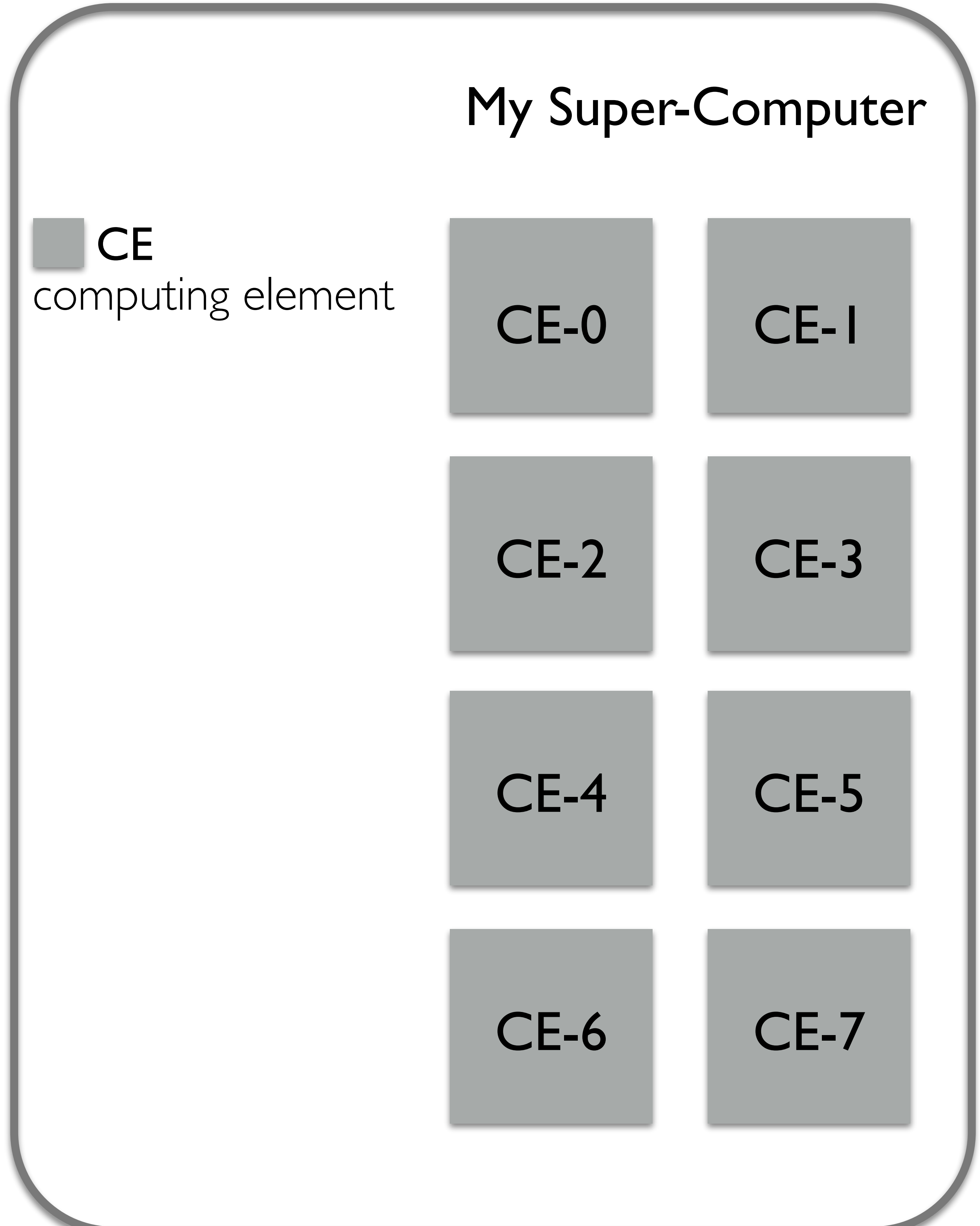


Step 1: Parallelization

PIC codes are well adapted to massive parallelism

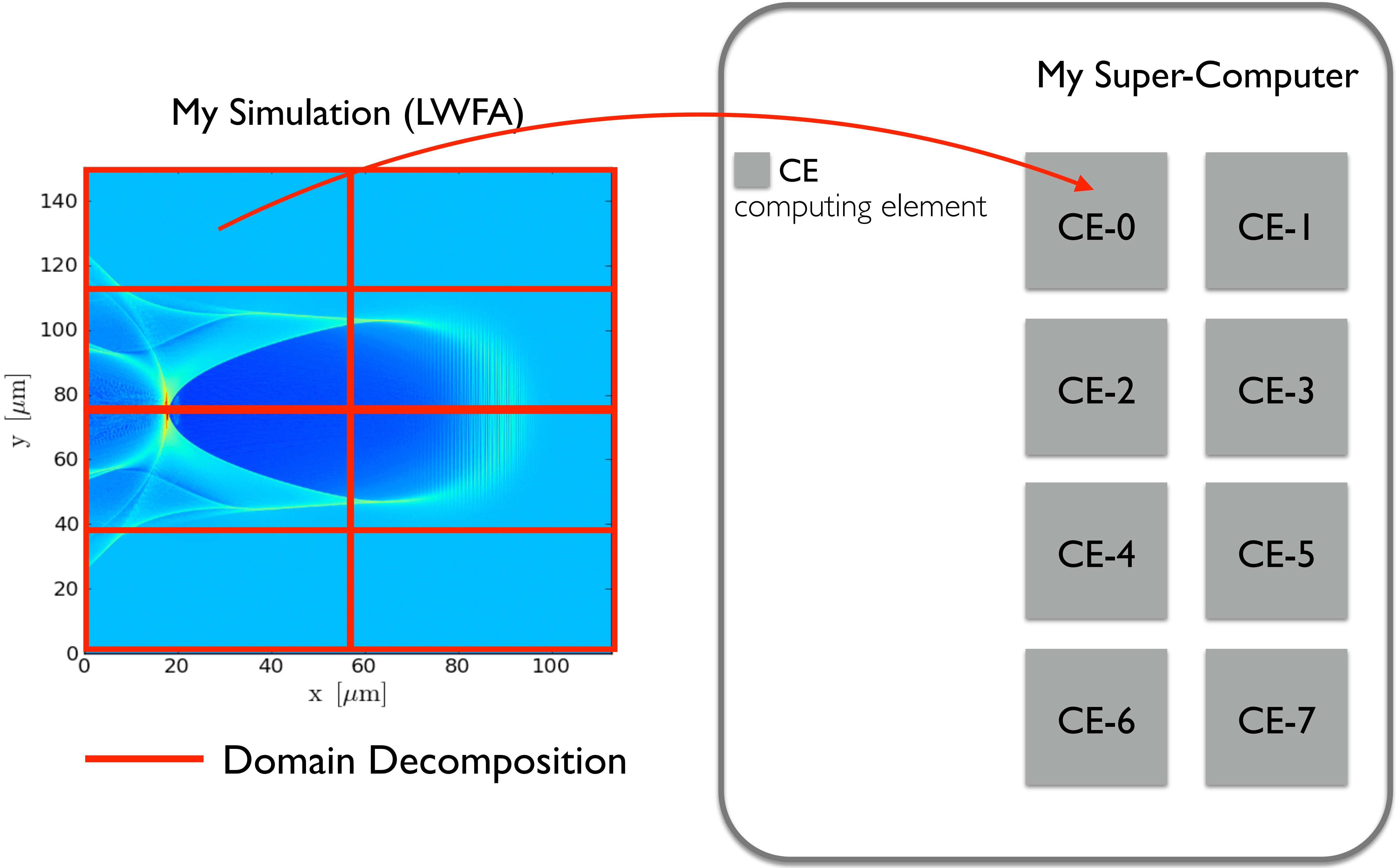
My Simulation (LWFA)





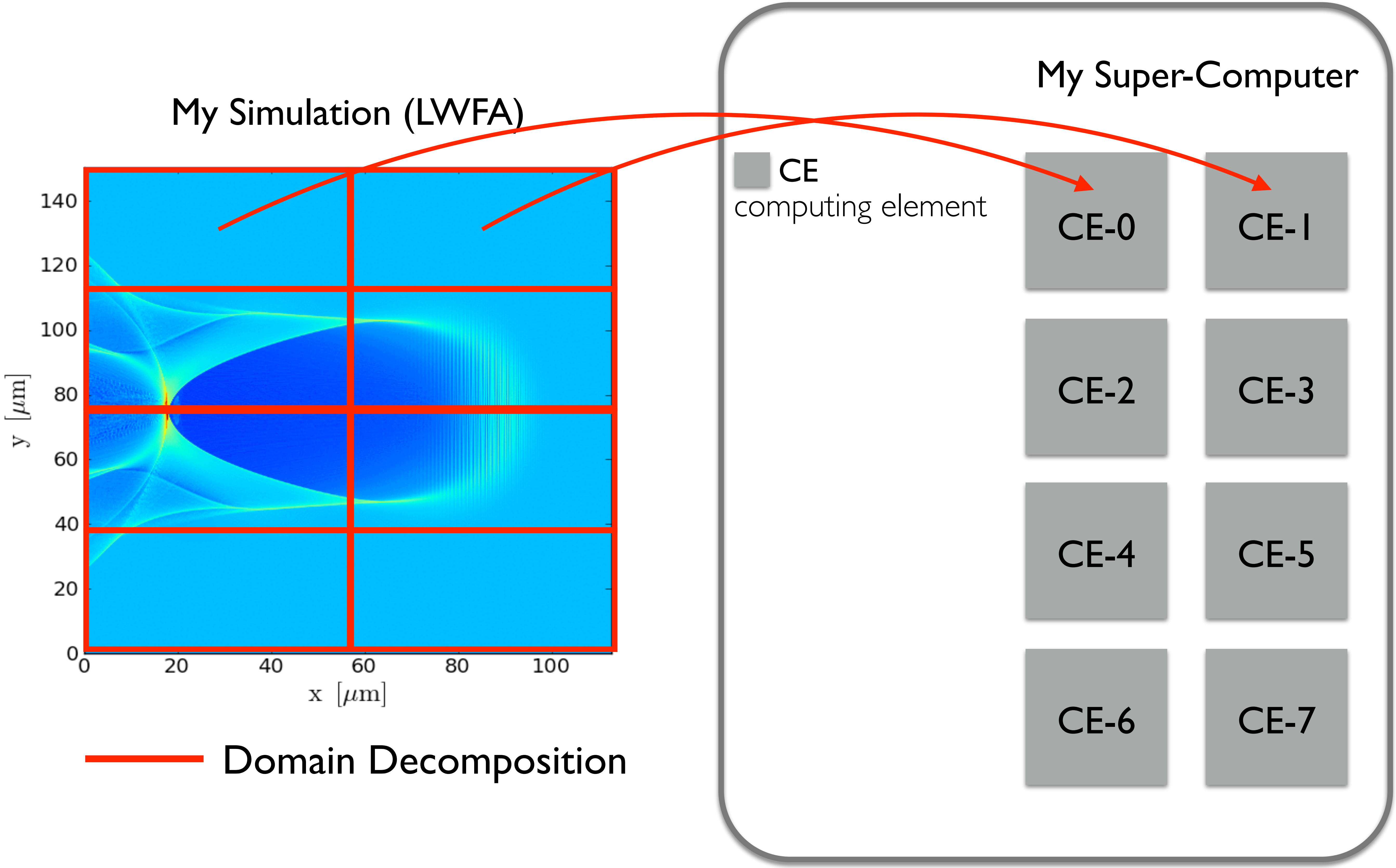
Step 1: Parallelization

PIC codes are well adapted to massive parallelism



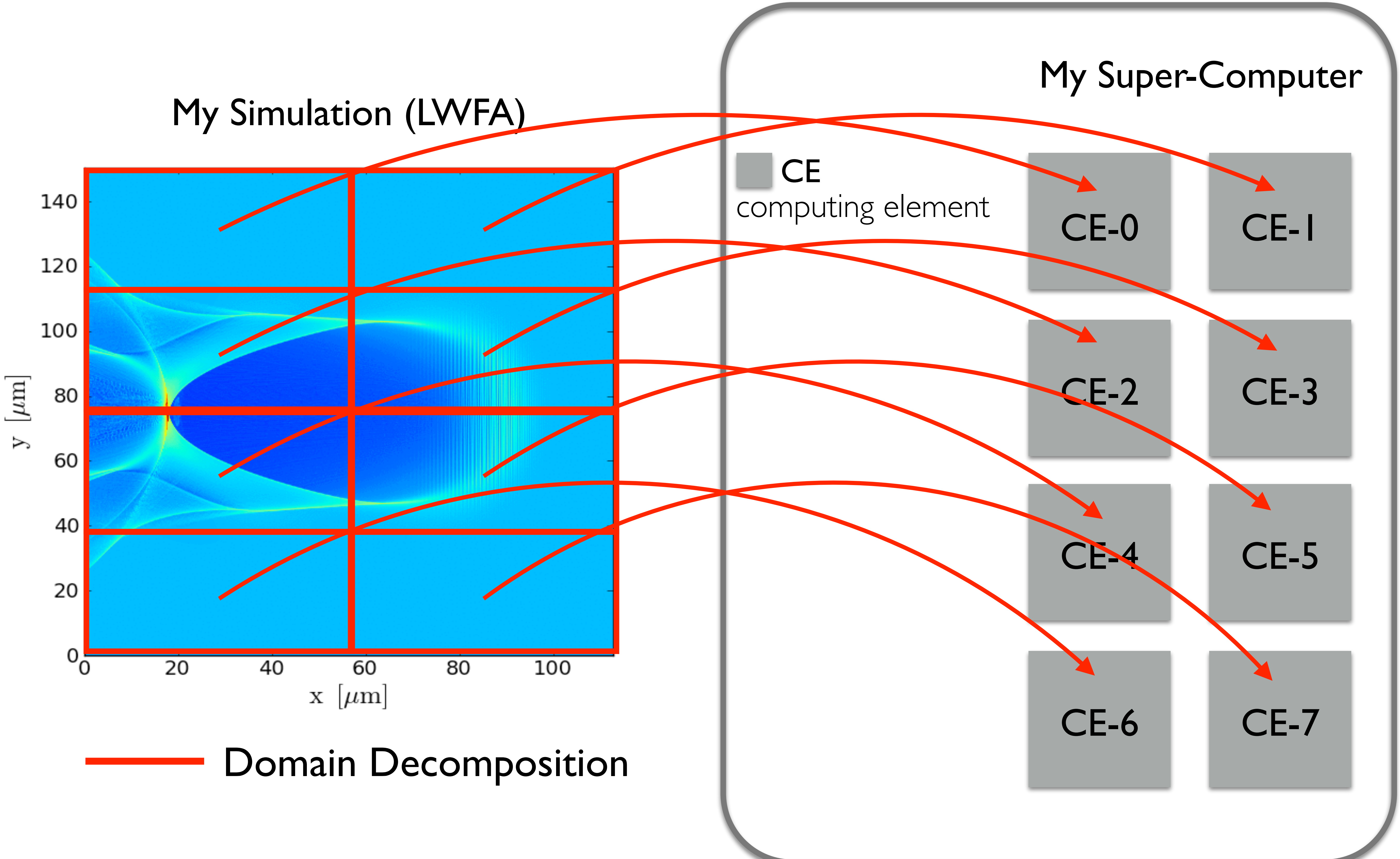
Step 1: Parallelization

PIC codes are well adapted to massive parallelism



Step 1: Parallelization

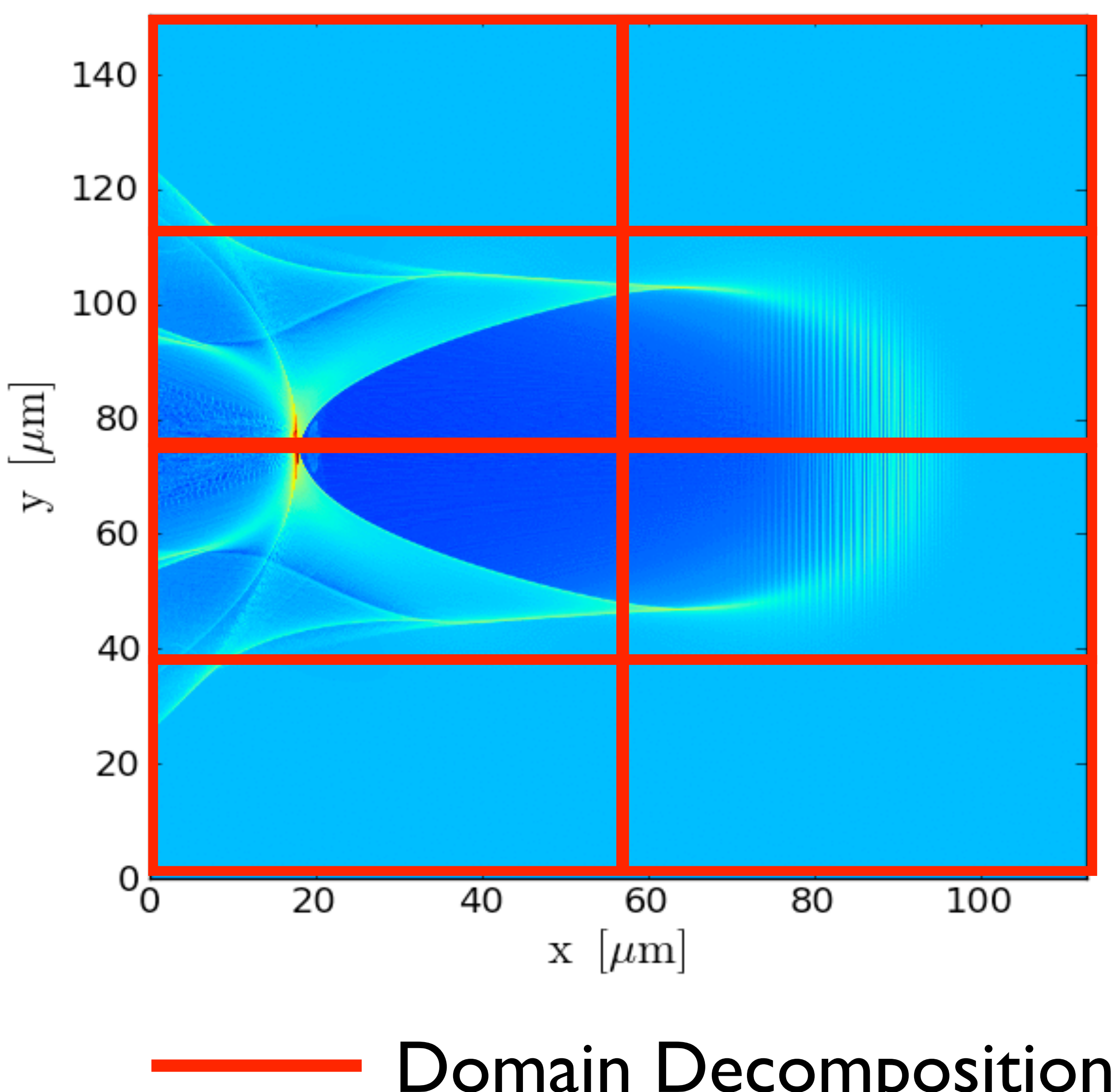
PIC codes are well adapted to massive parallelism

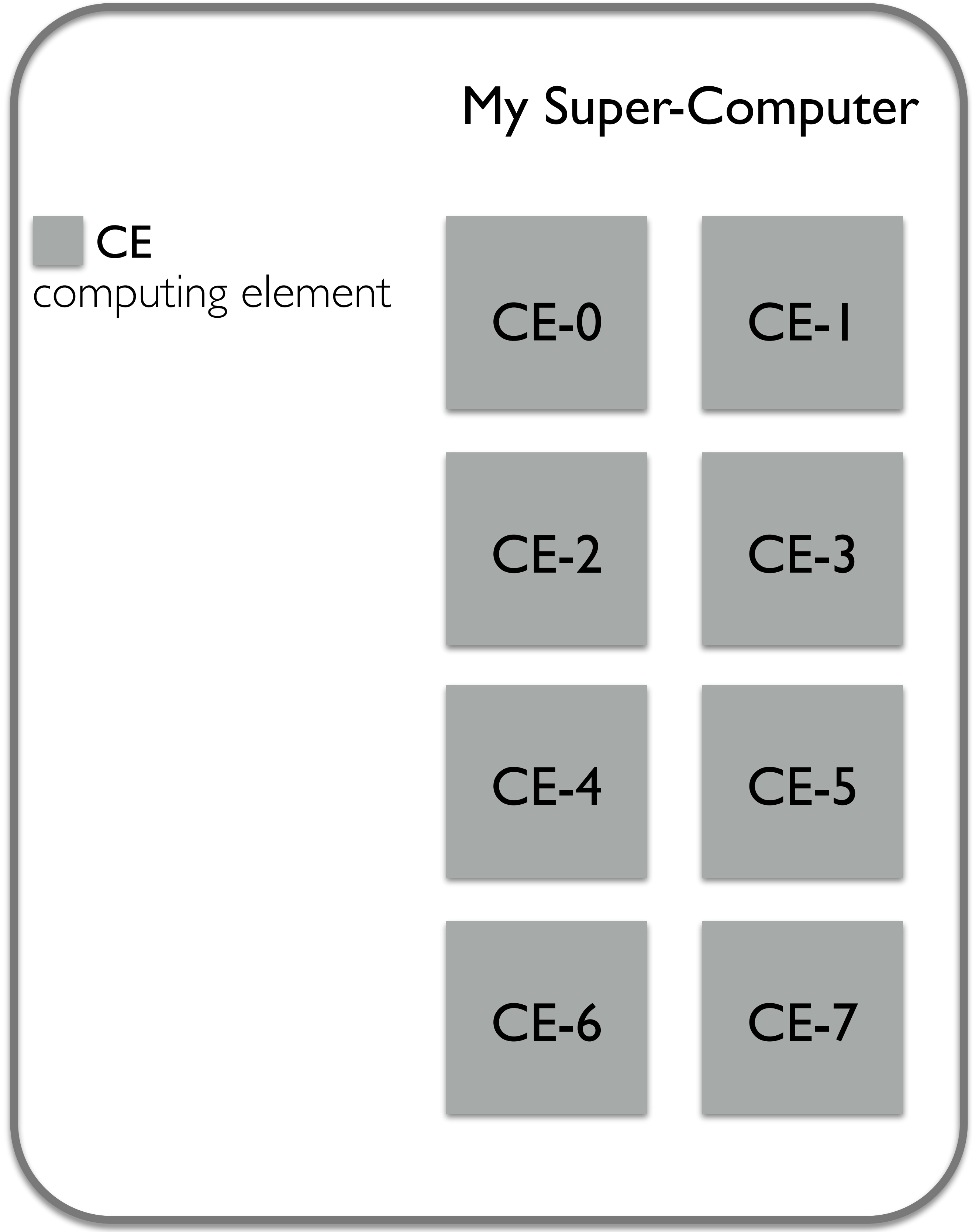


Step 1: Parallelization

PIC codes are well adapted to massive parallelism

My Simulation (LWFA)

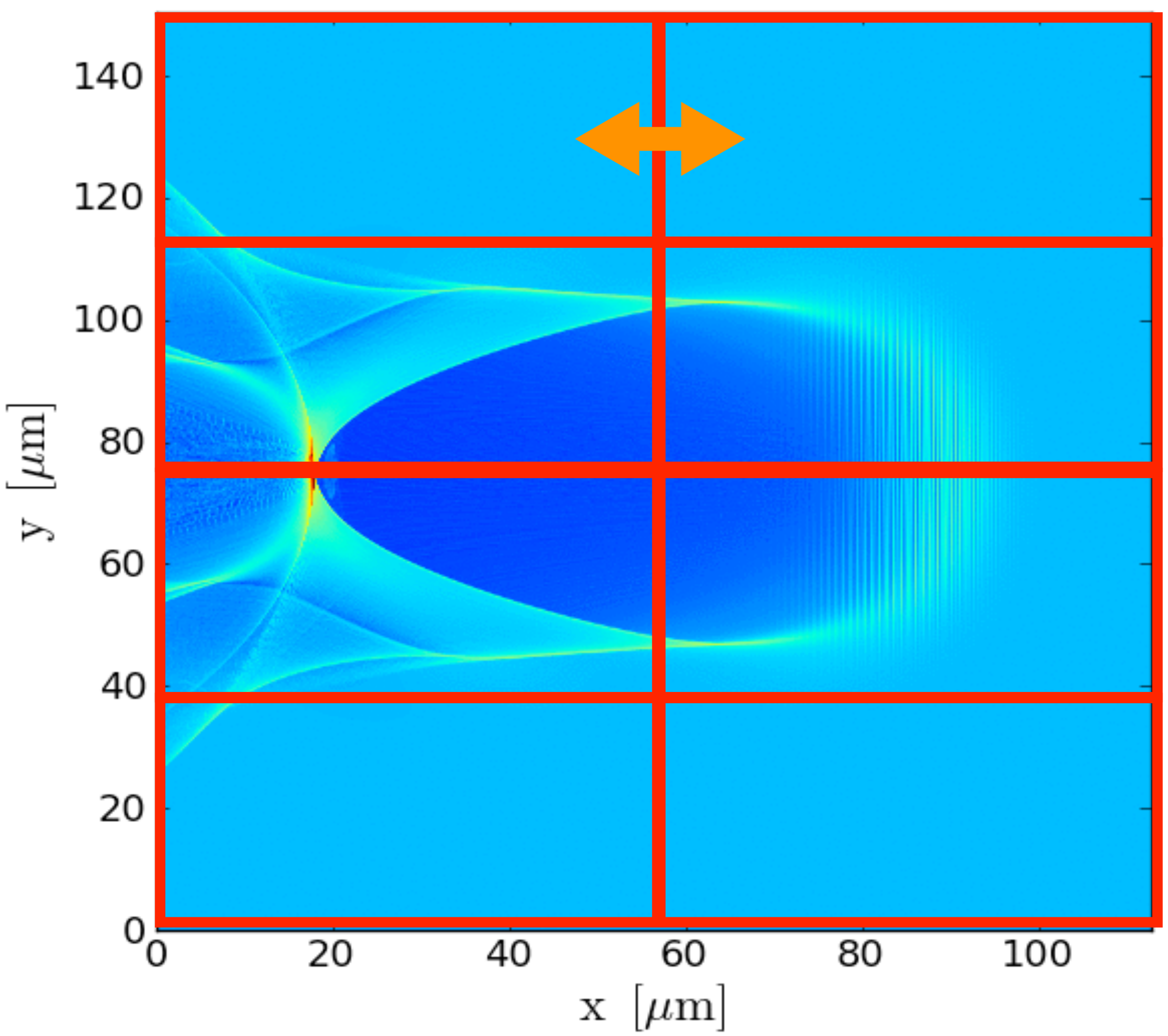




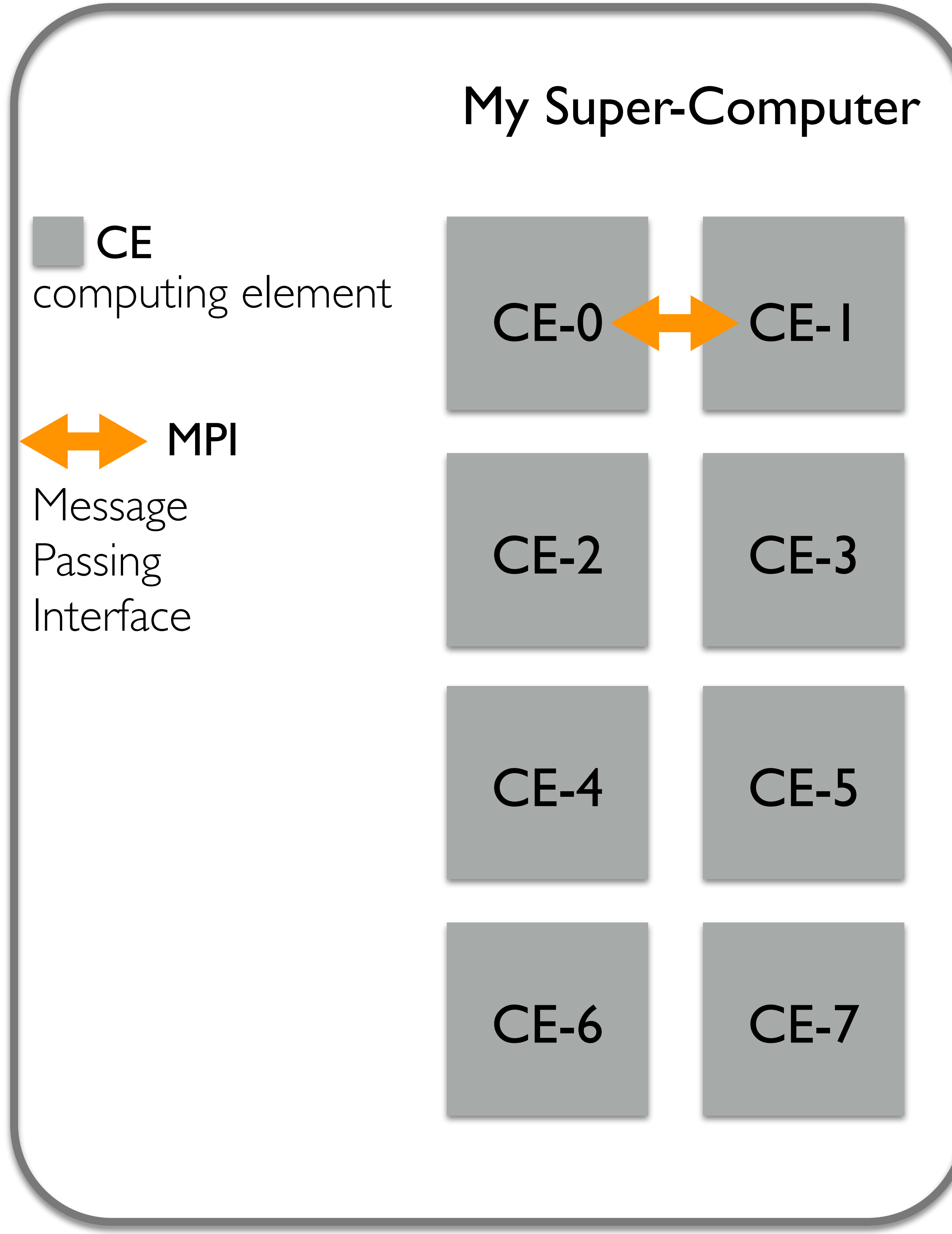
Step 1: Parallelization

PIC codes are well adapted to massive parallelism

My Simulation (LWFA)

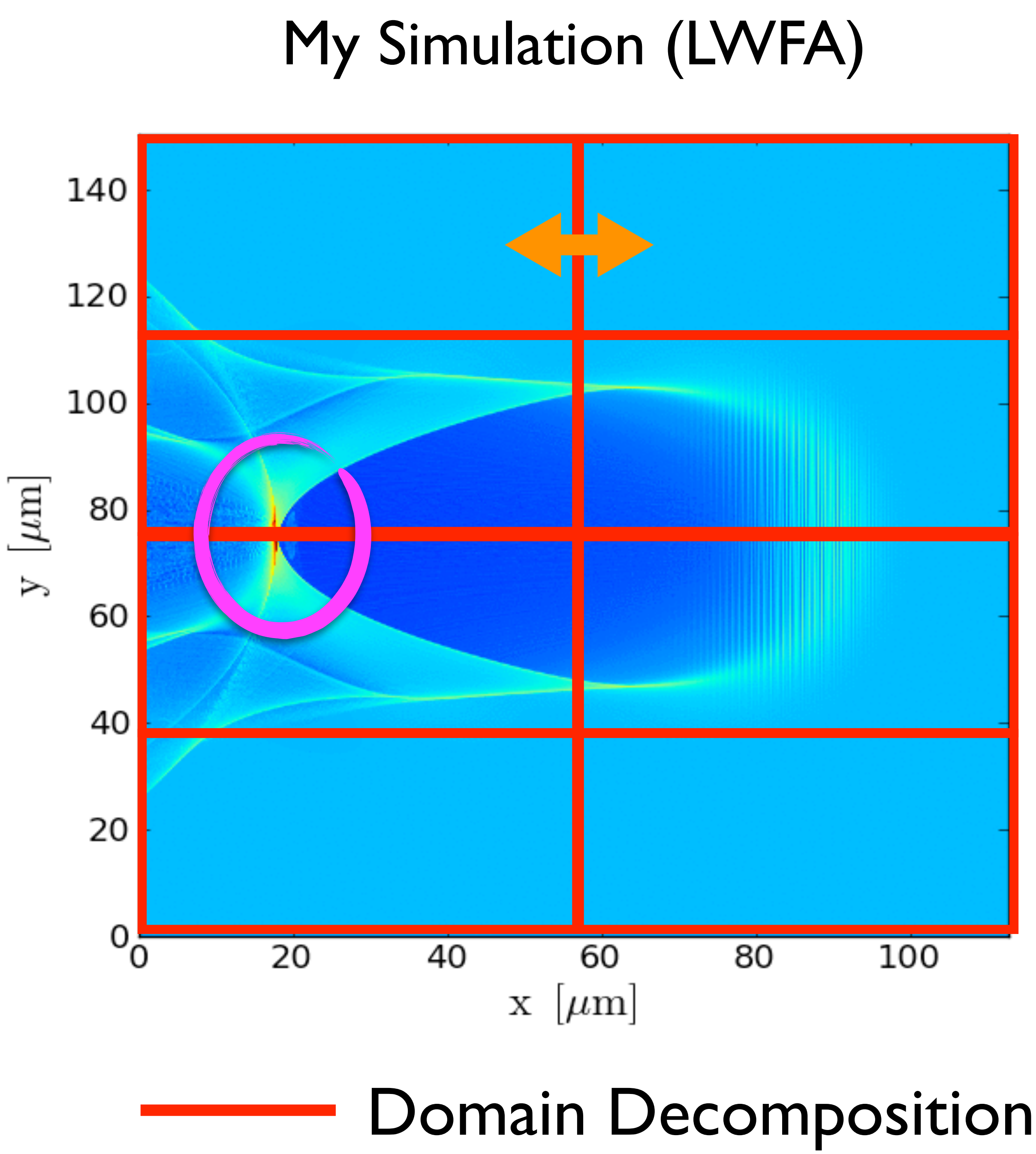


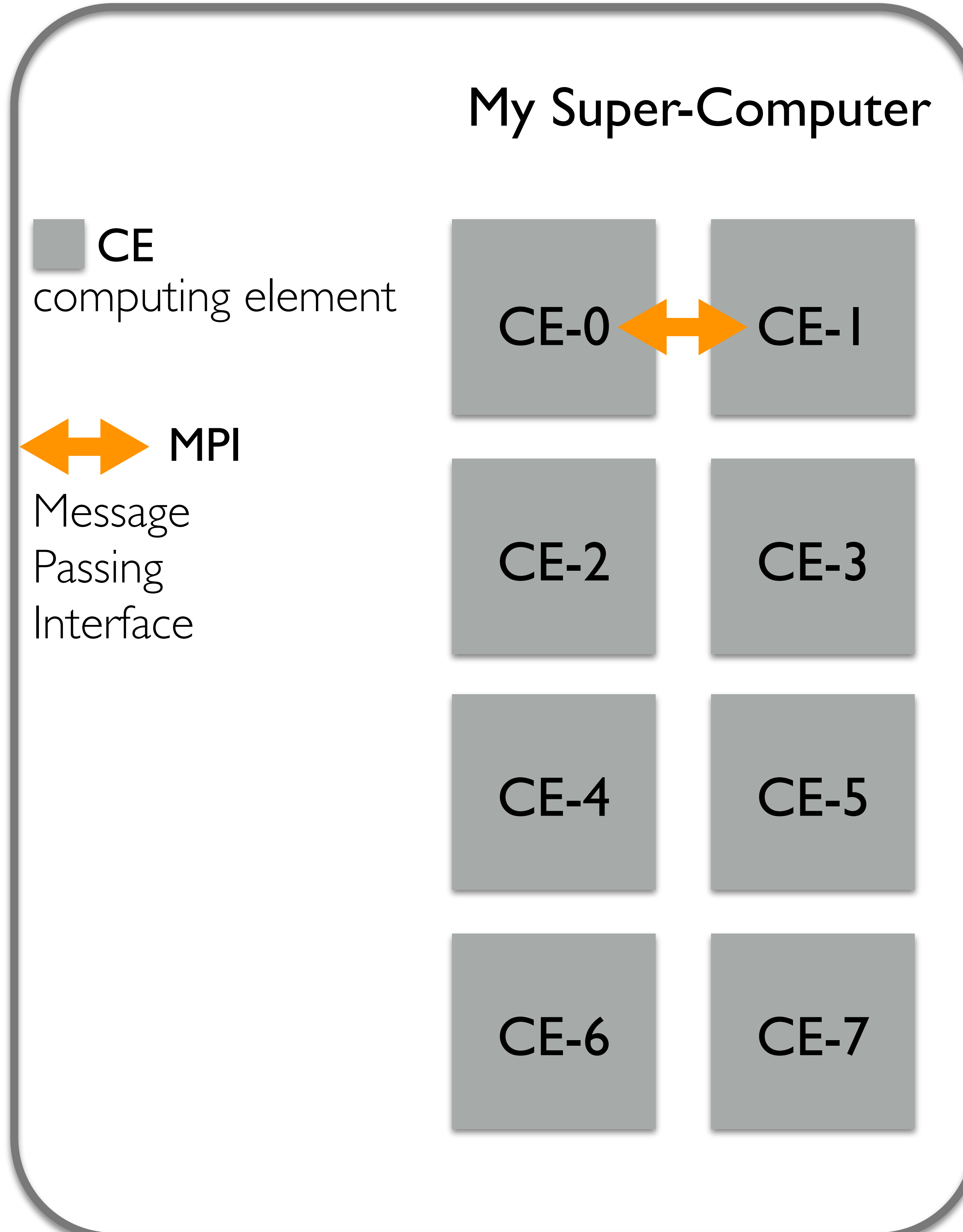
Domain Decomposition



Step 1: Parallelization

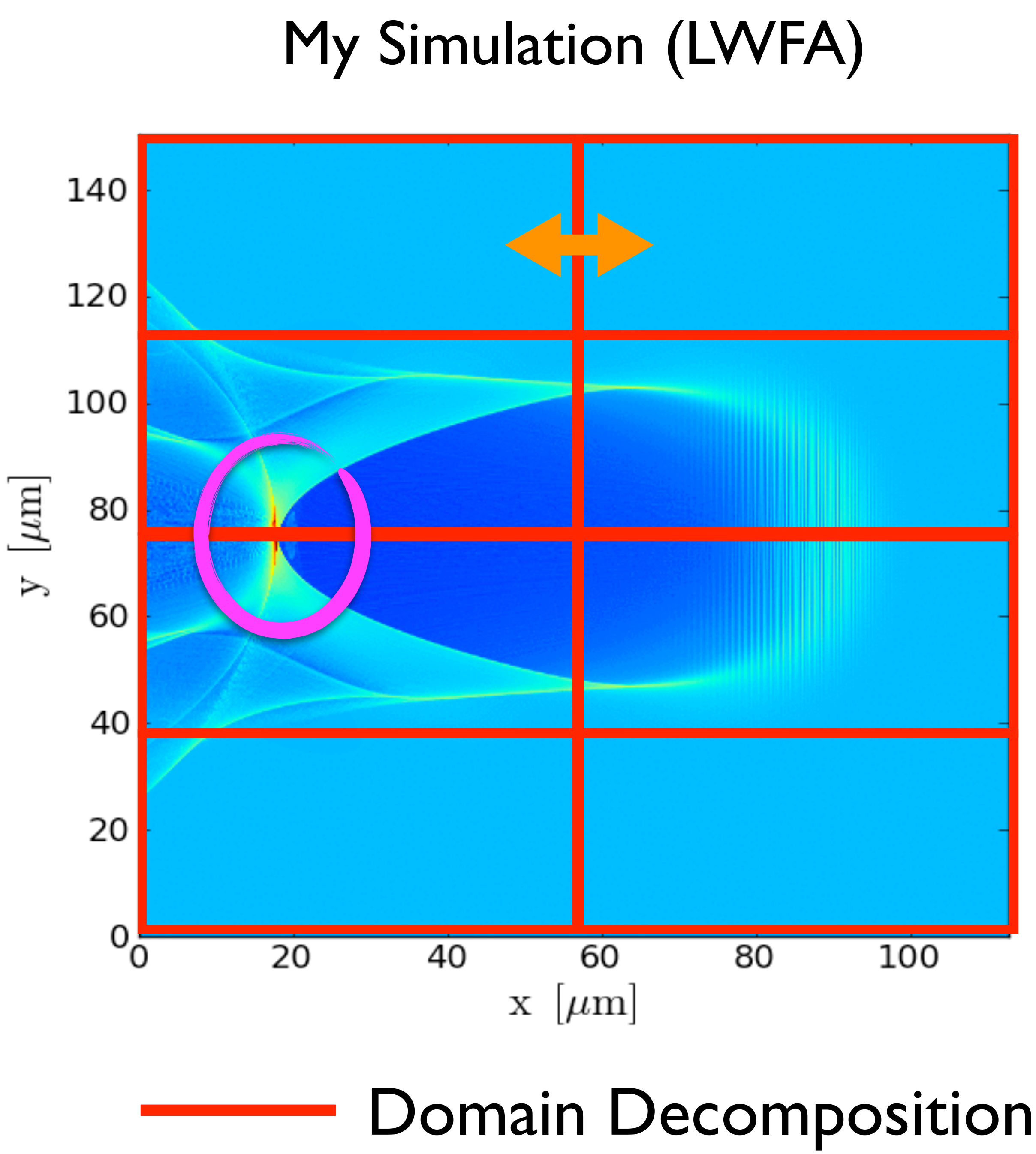
PIC codes are well adapted to massive parallelism

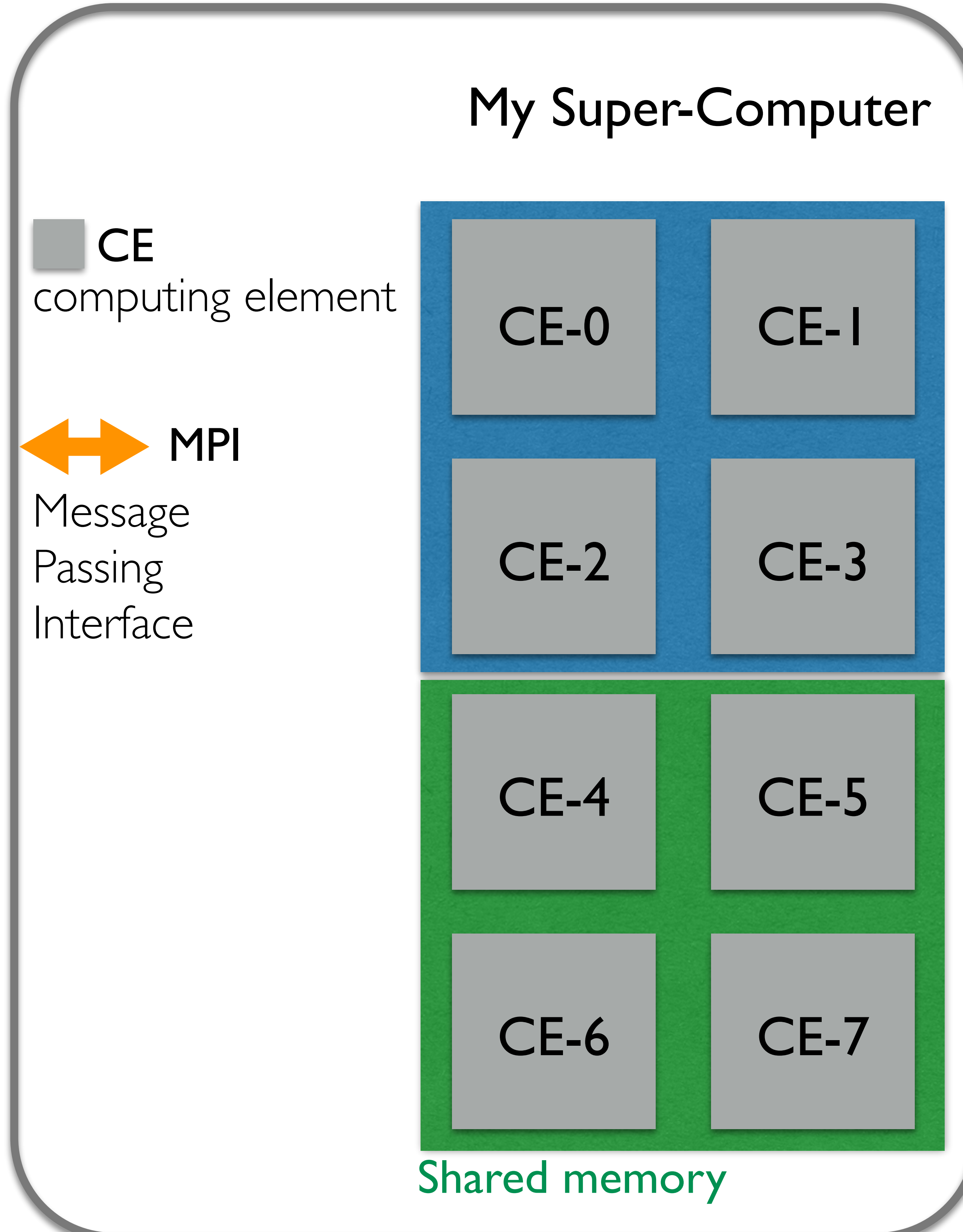




Step 1: Parallelization

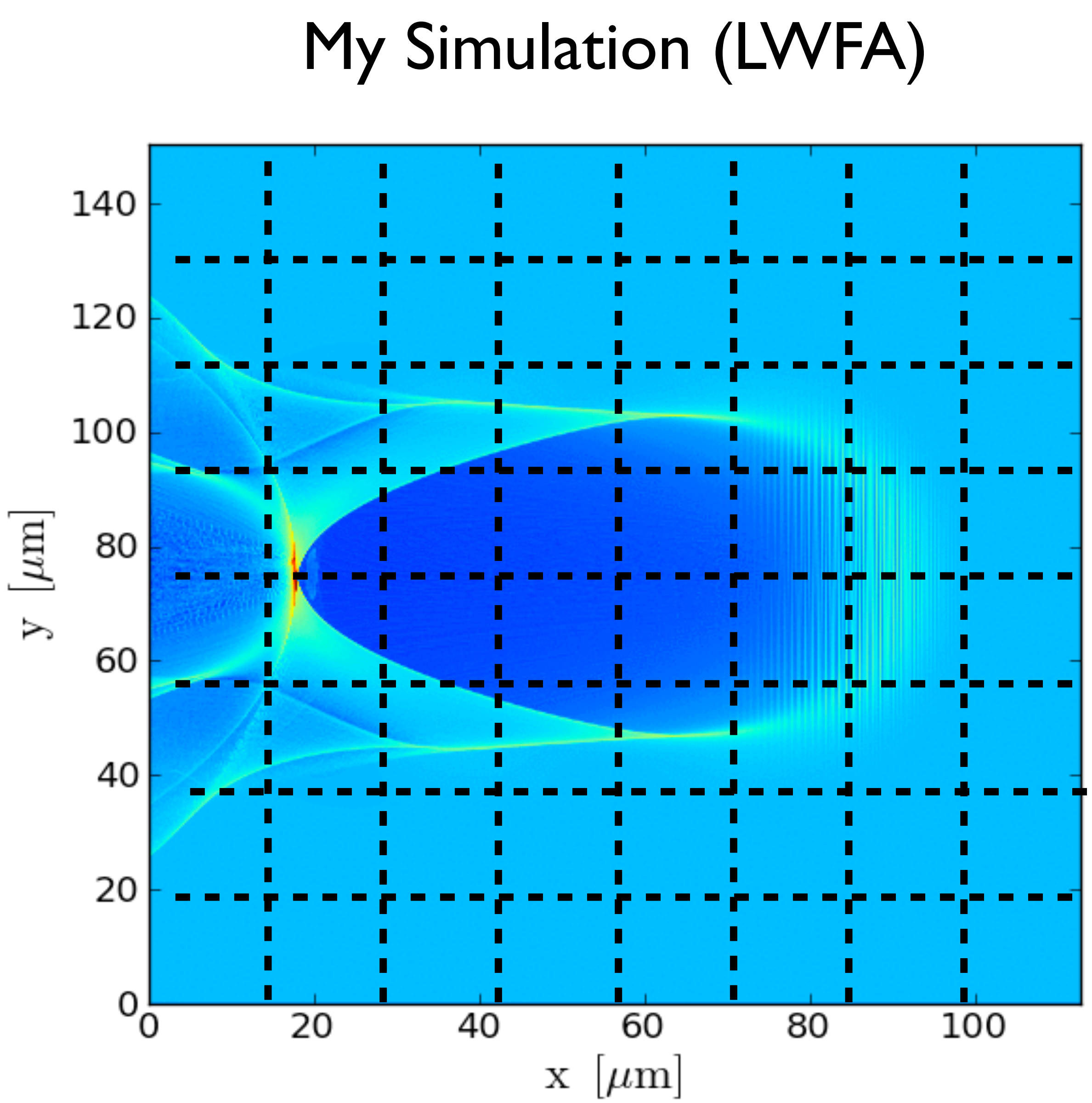
PIC codes are well adapted to massive parallelism

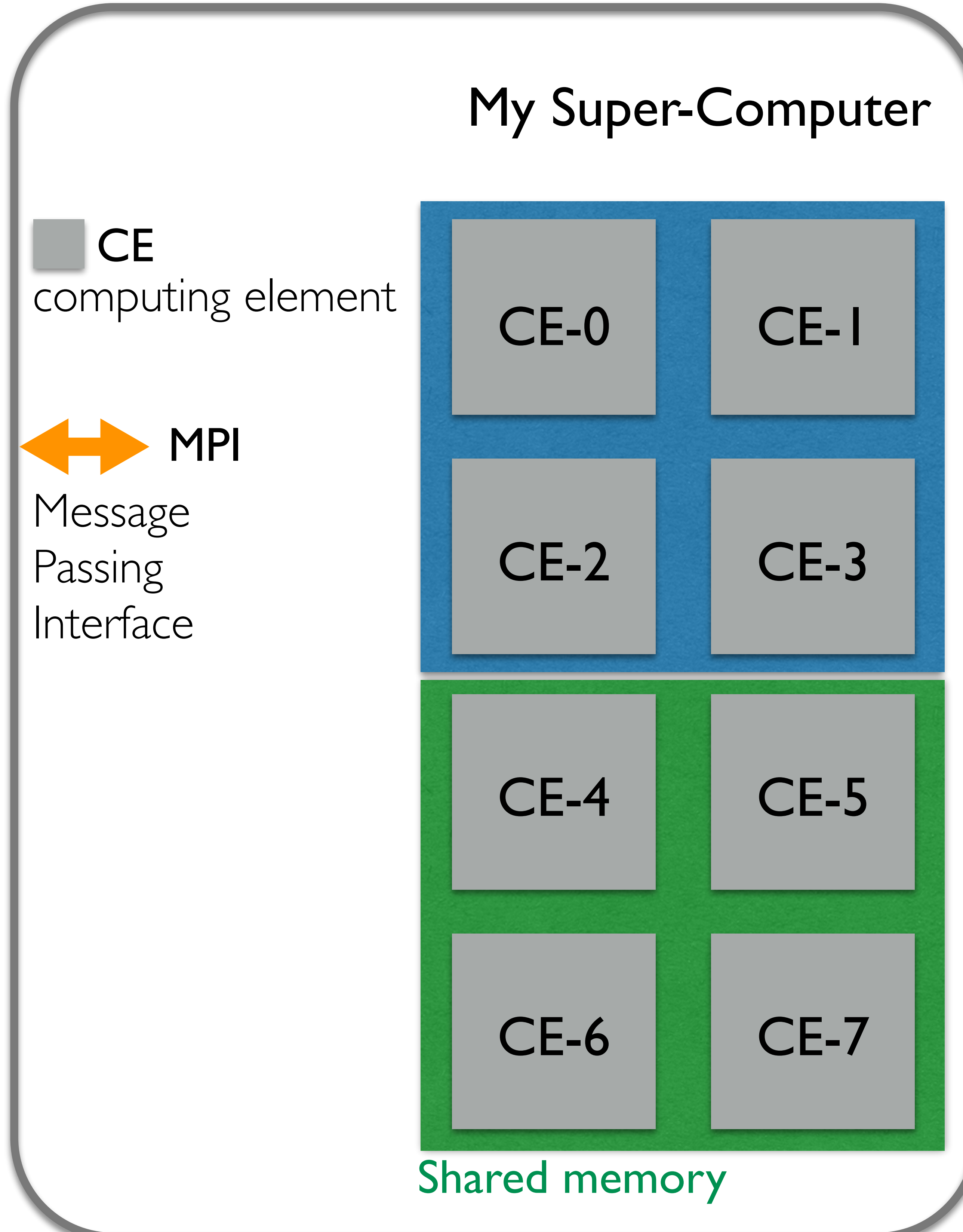




Step 1: Parallelization

PIC codes are well adapted to massive parallelism



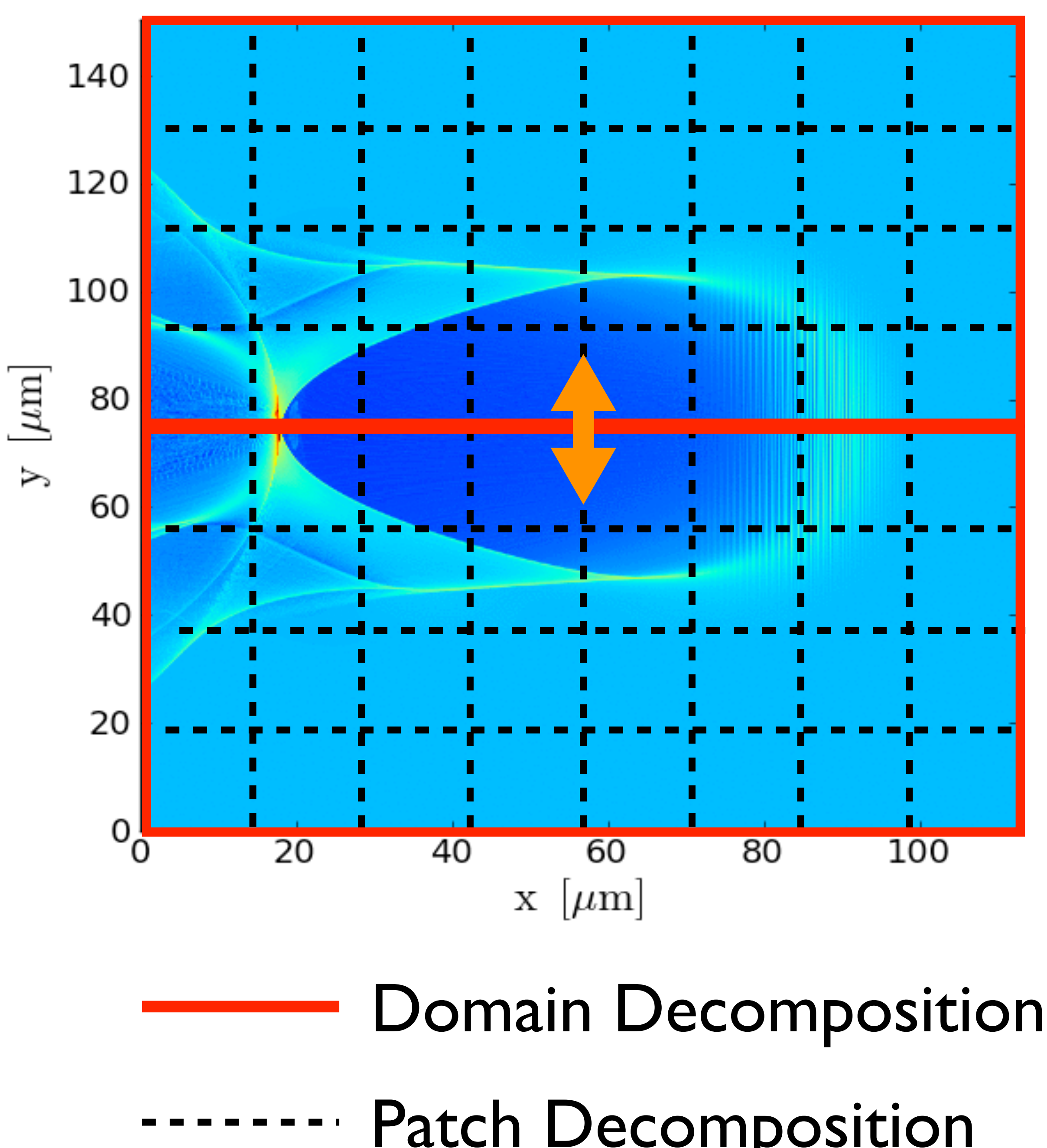


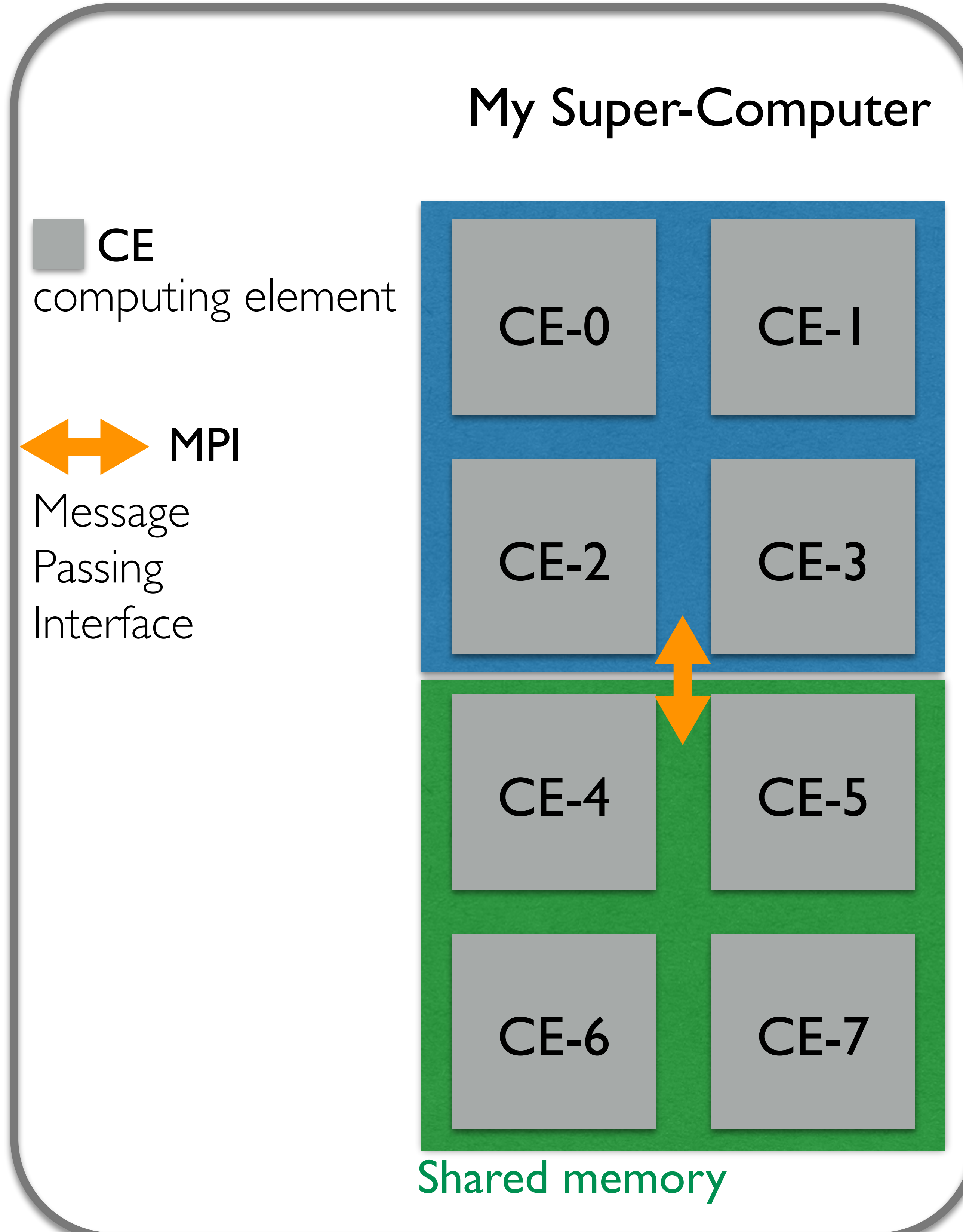
------- Patch Decomposition

Step 1: Parallelization

PIC codes are well adapted to massive parallelism

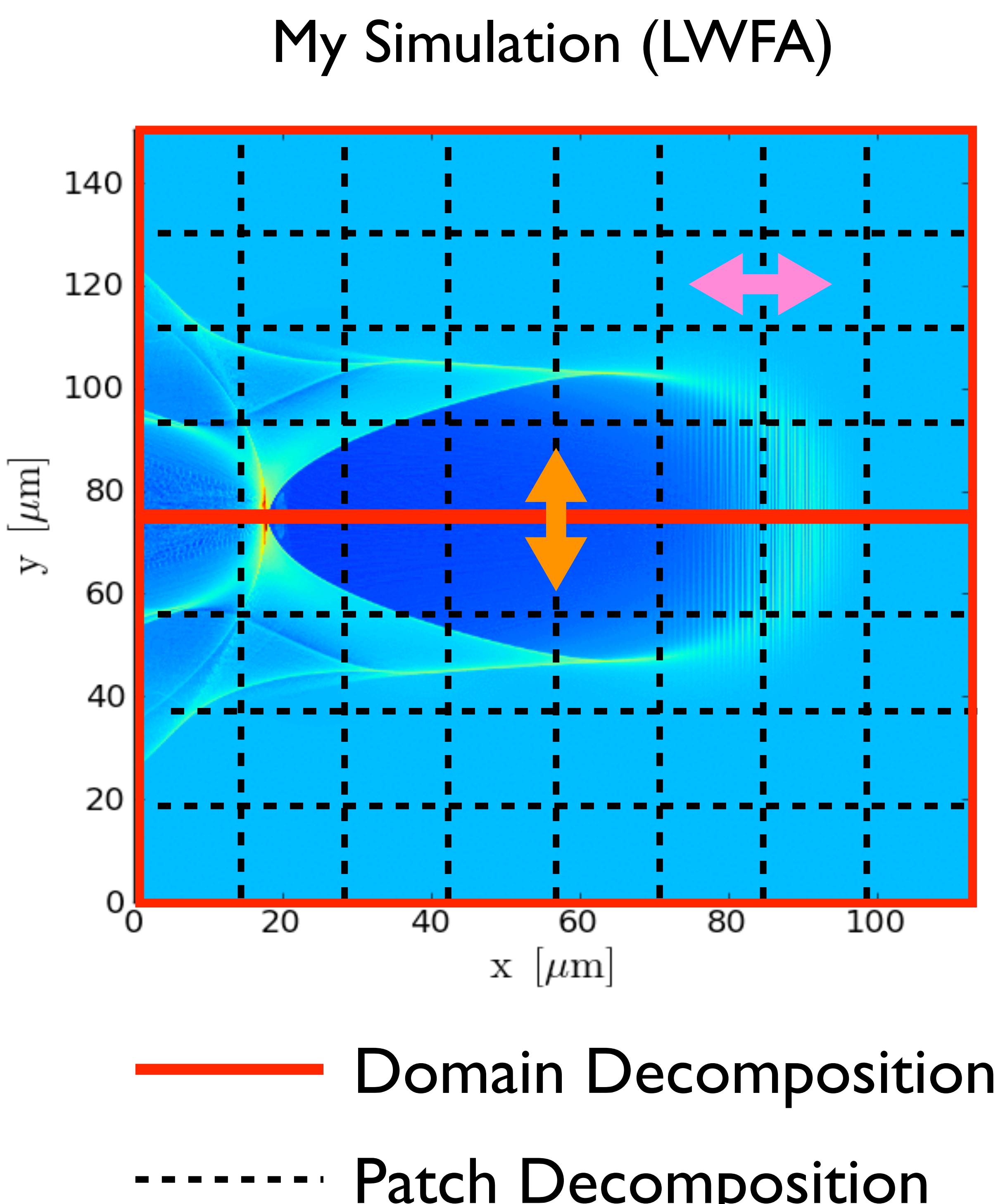
My Simulation (LWFA)

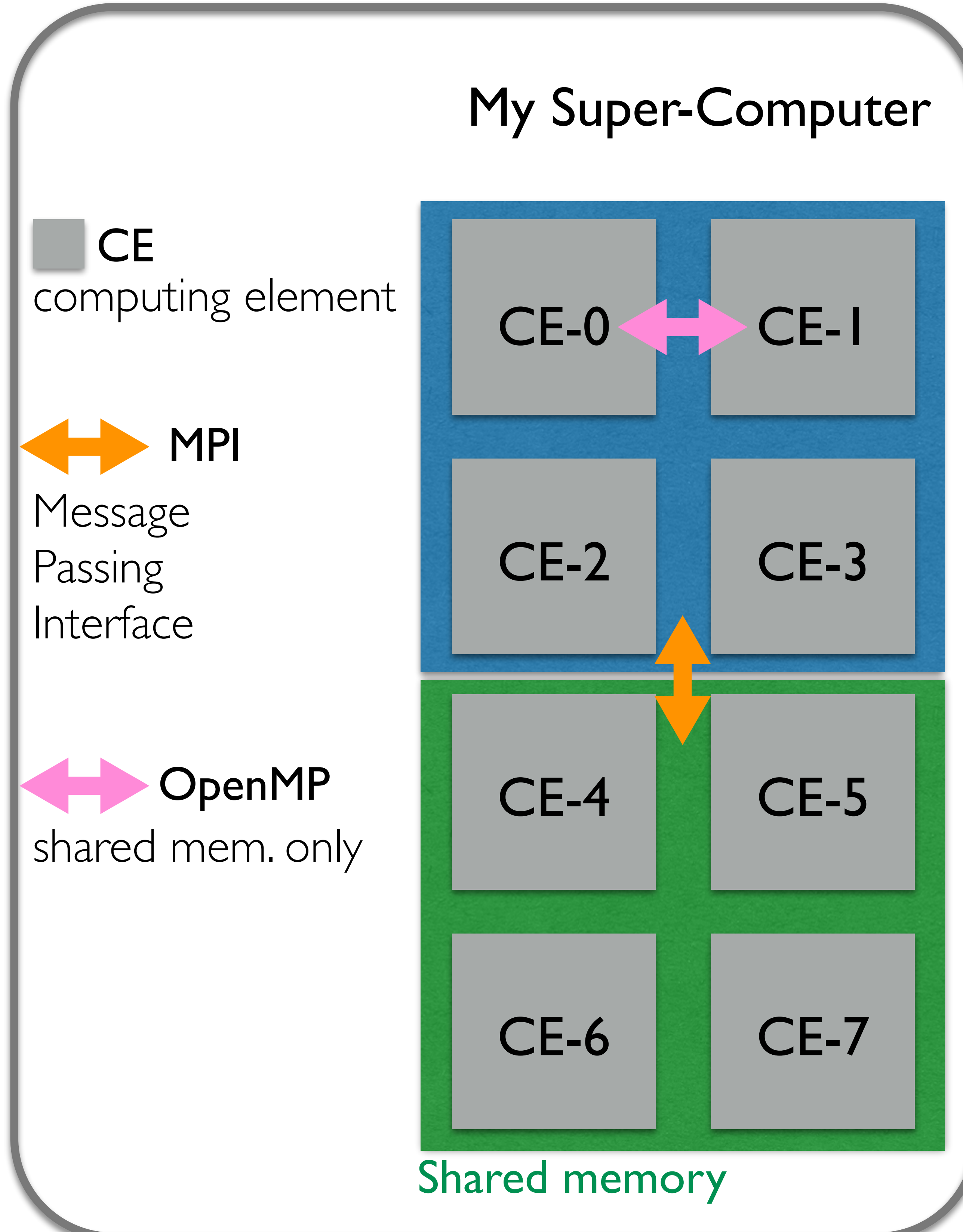




Step 1: Parallelization

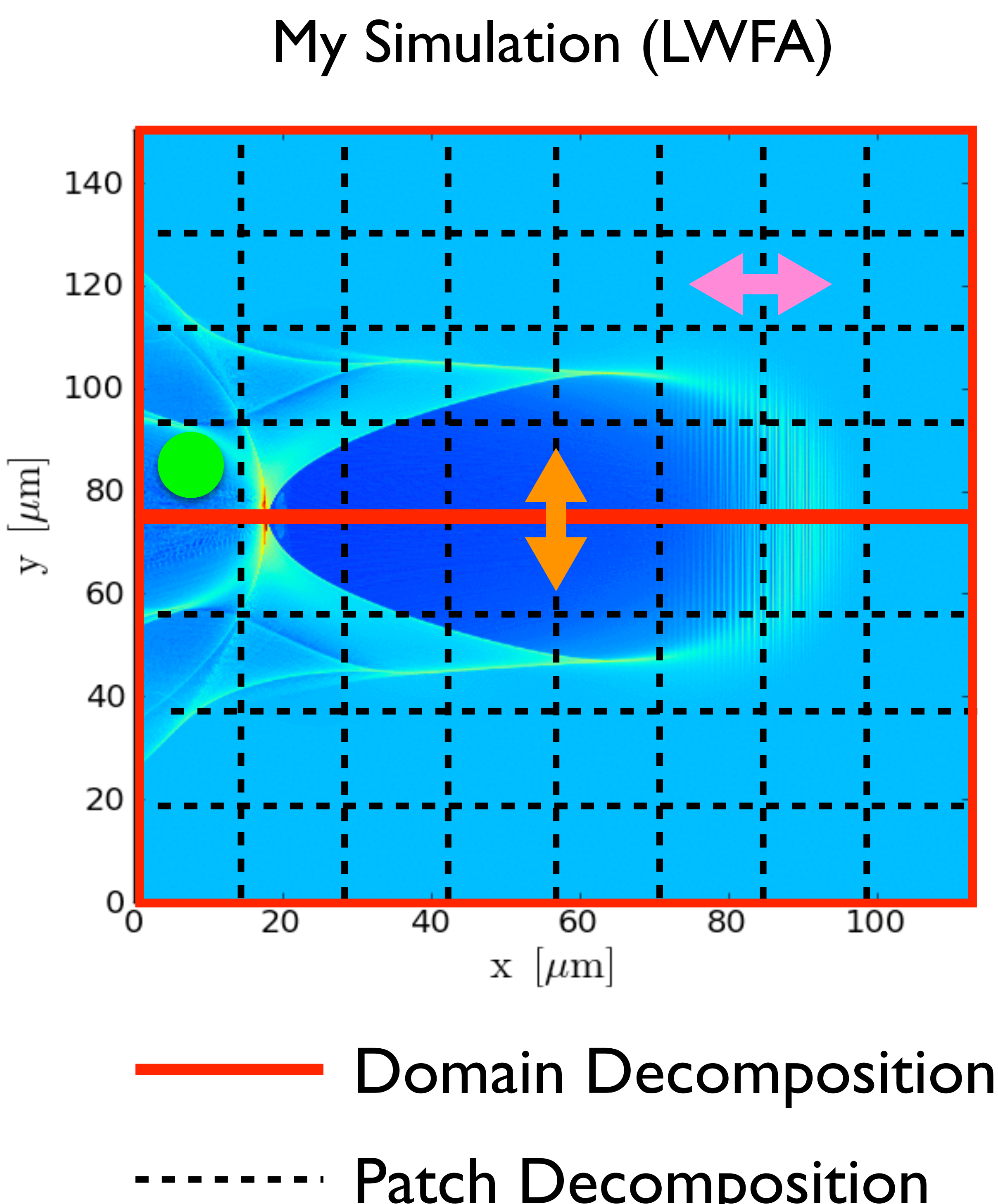
PIC codes are well adapted to massive parallelism

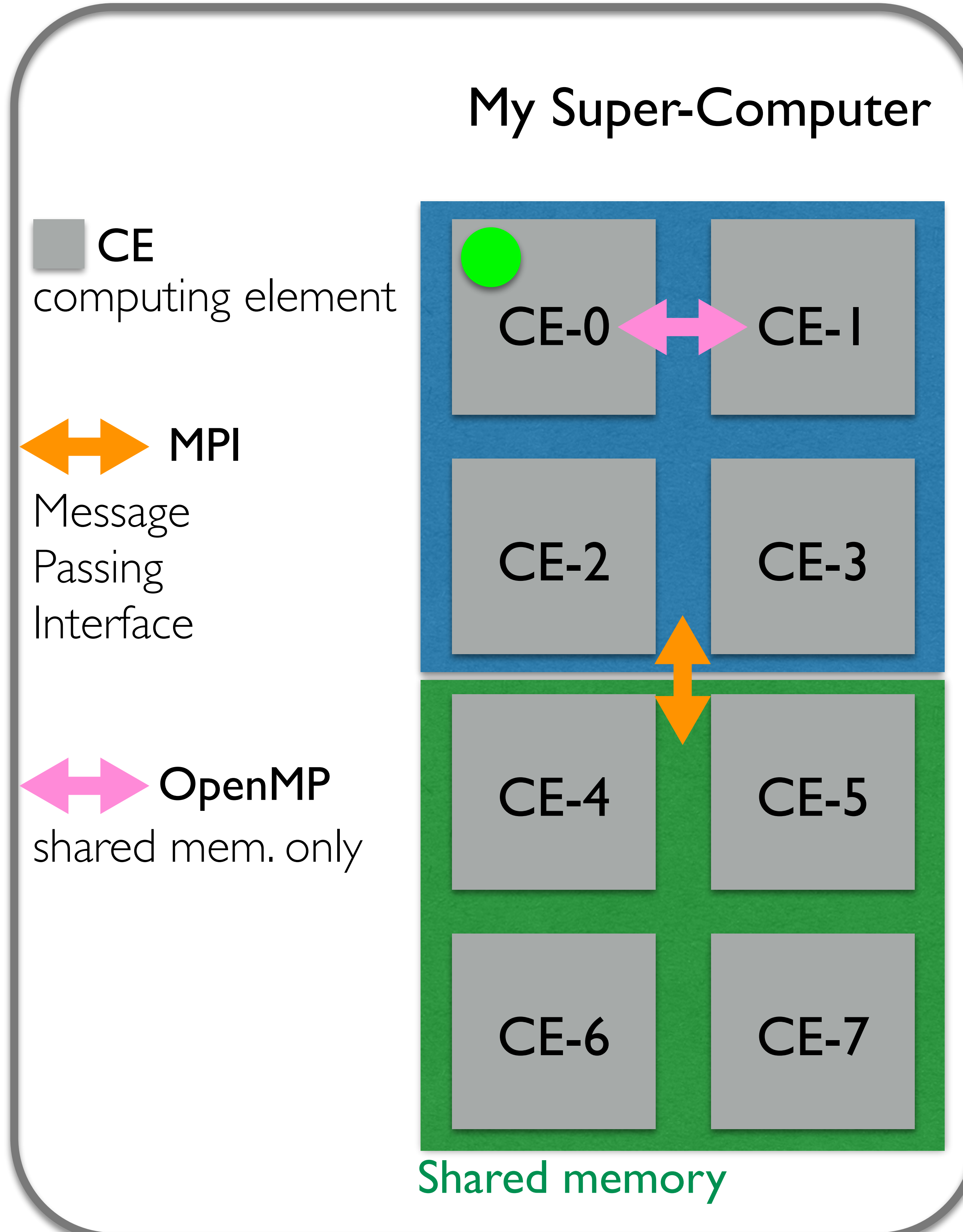




Step 1: Parallelization

PIC codes are well adapted to massive parallelism

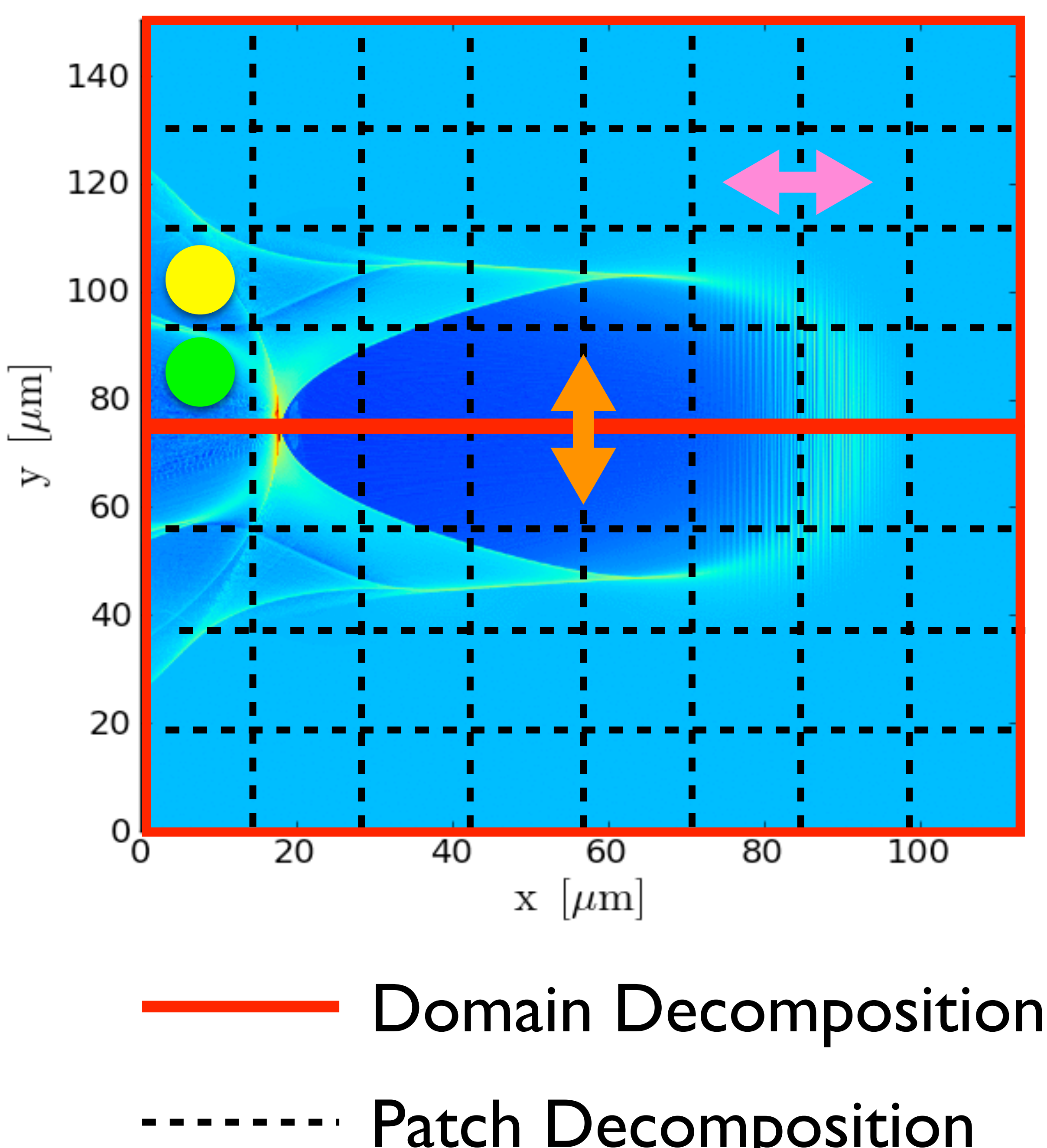


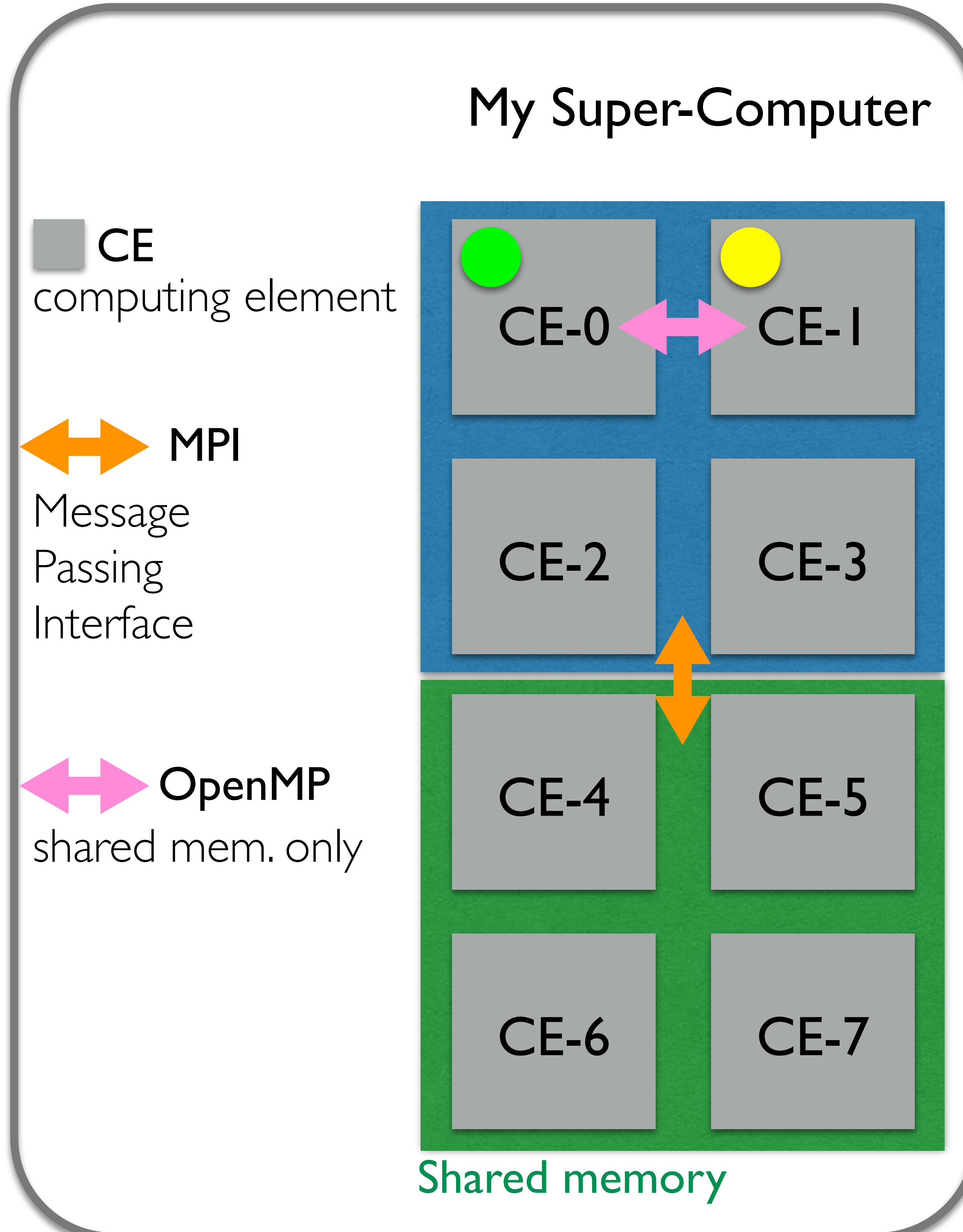


Step 1: Parallelization

PIC codes are well adapted to massive parallelism

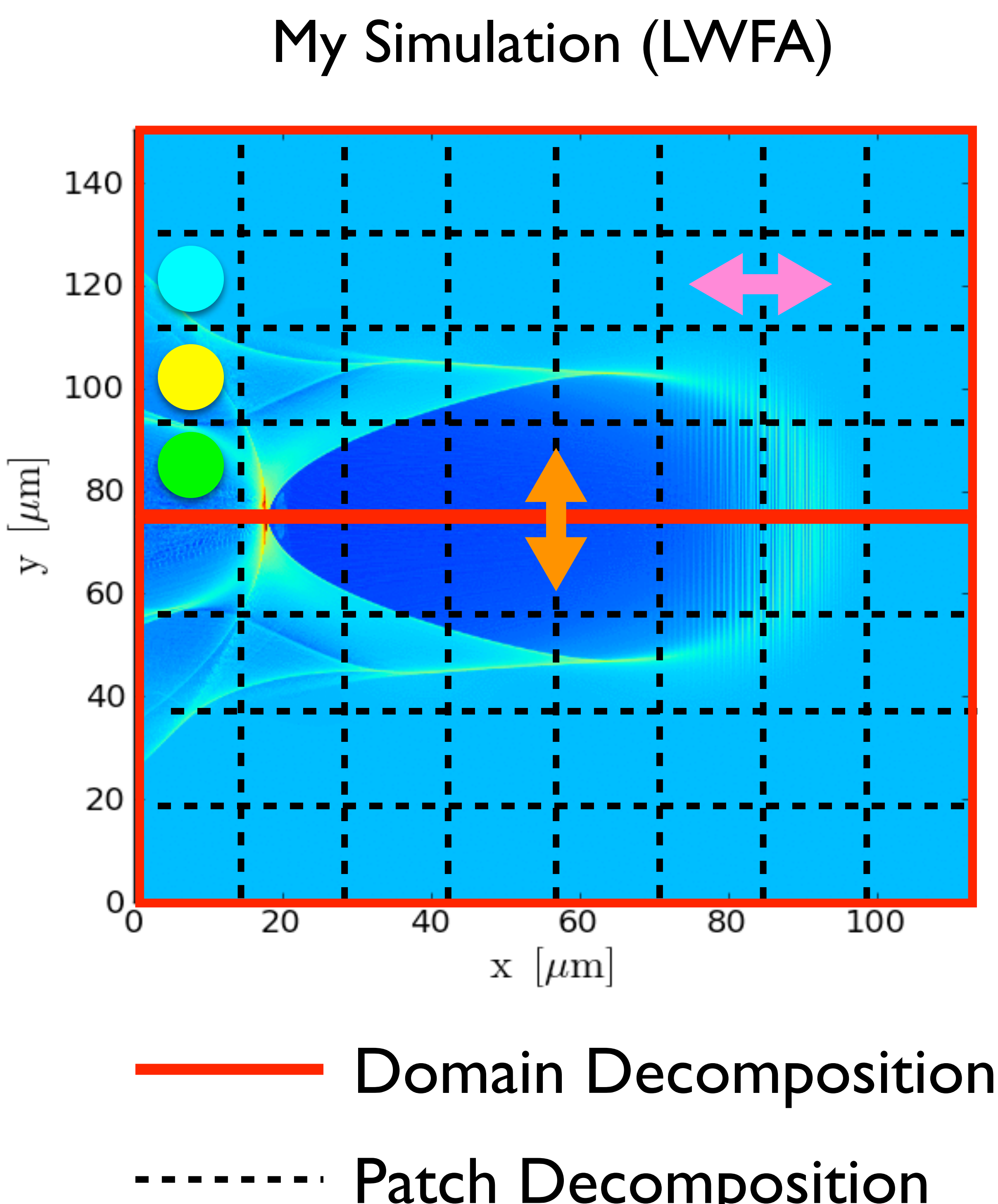
My Simulation (LWFA)

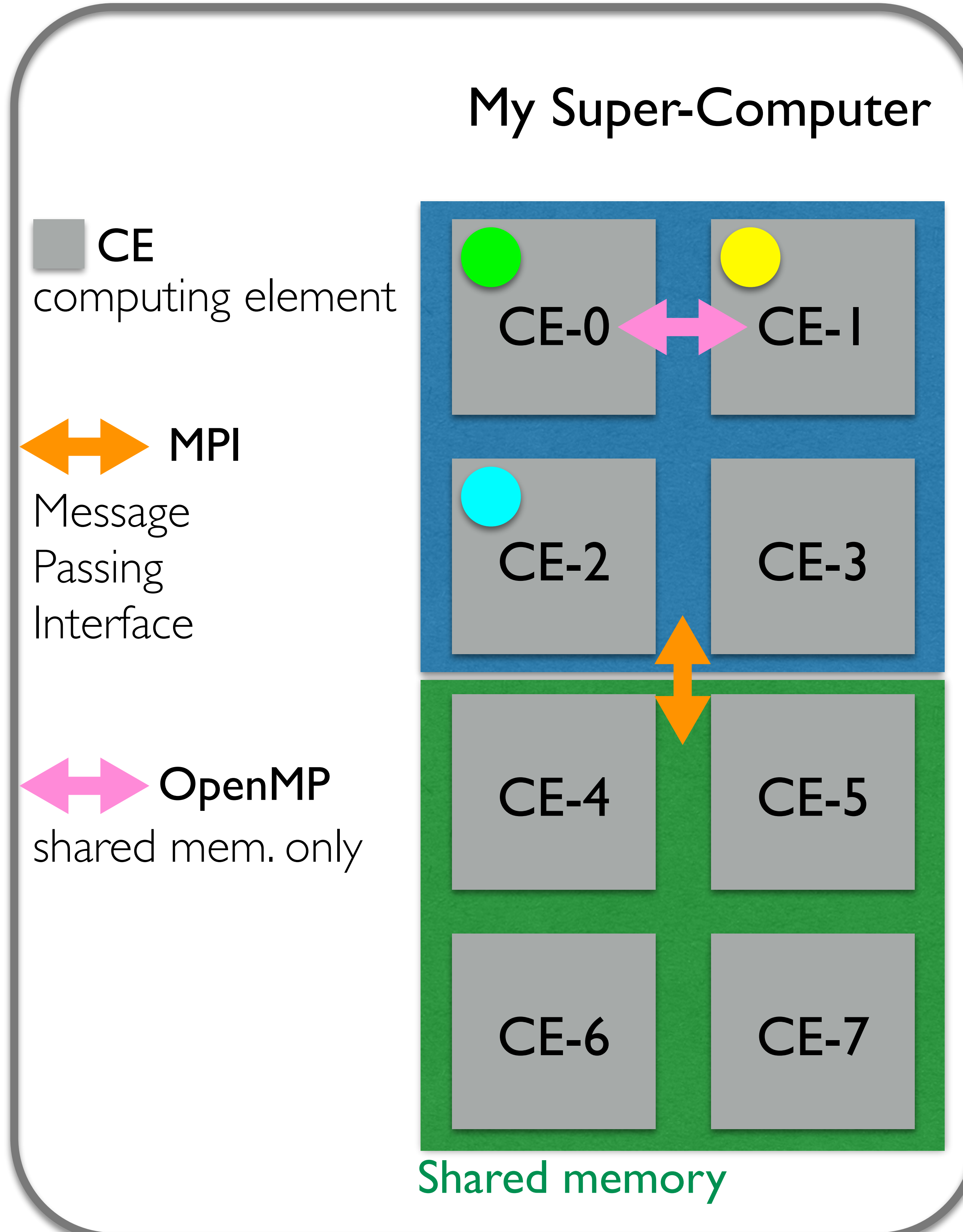




Step 1: Parallelization

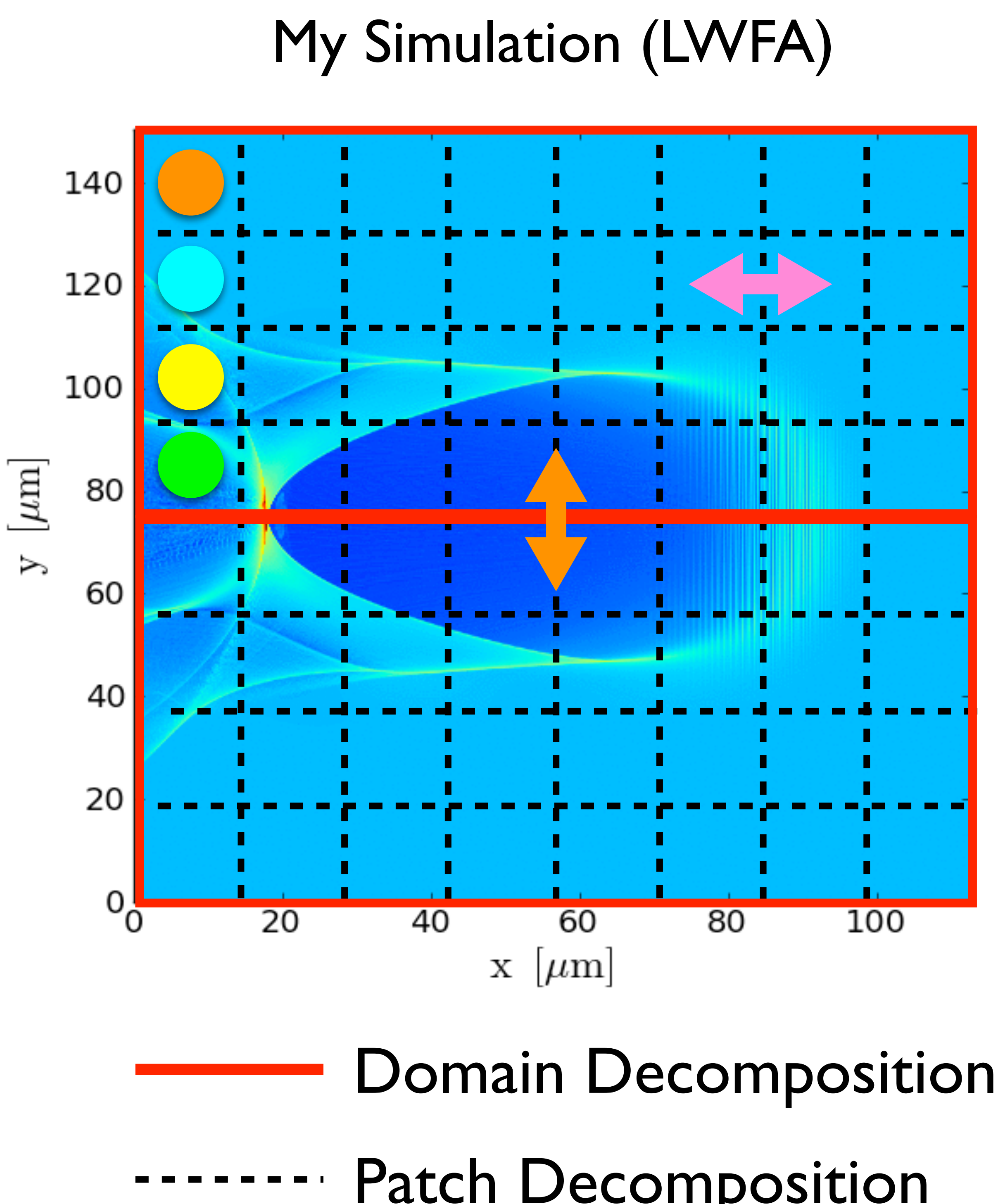
PIC codes are well adapted to massive parallelism

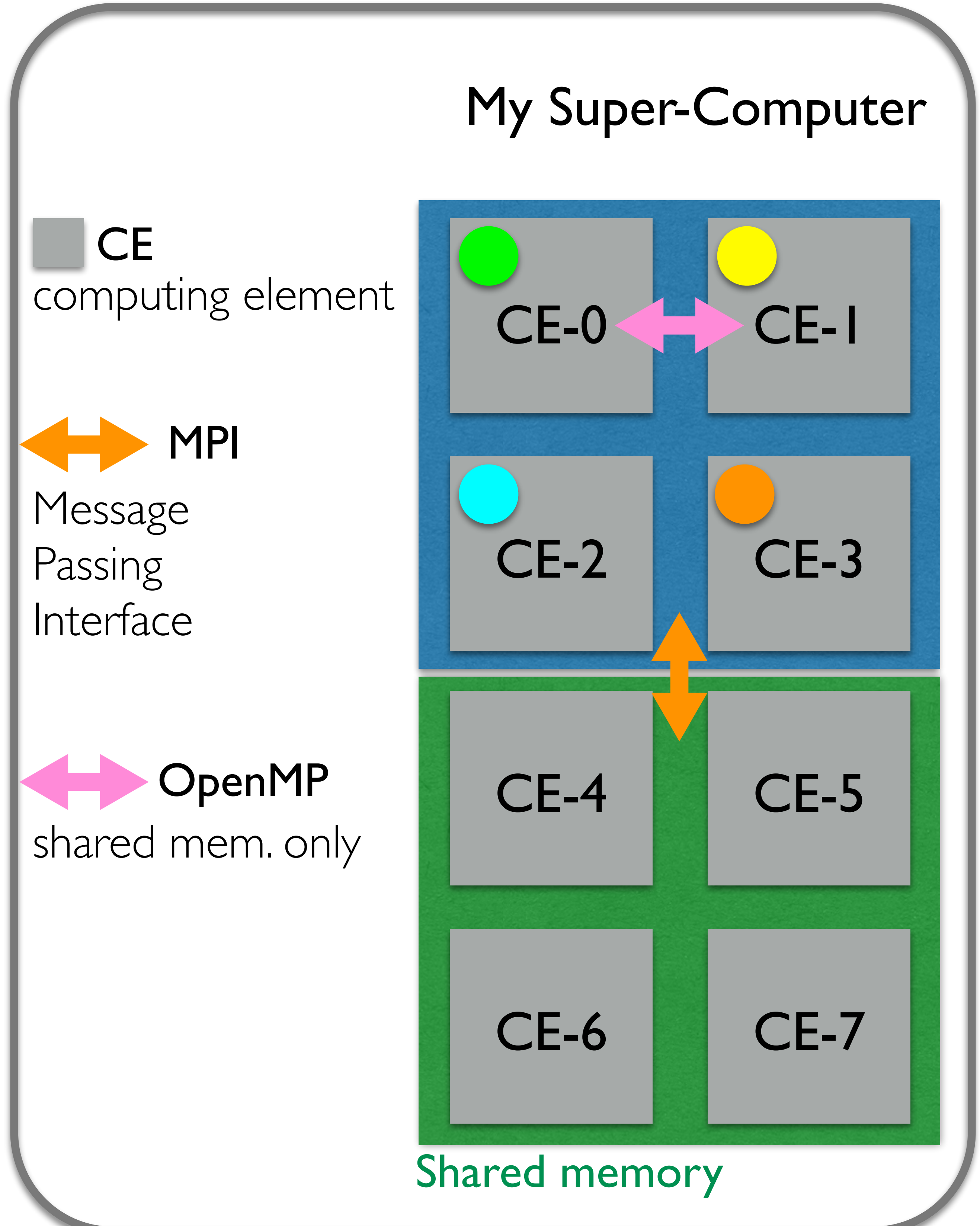




Step 1: Parallelization

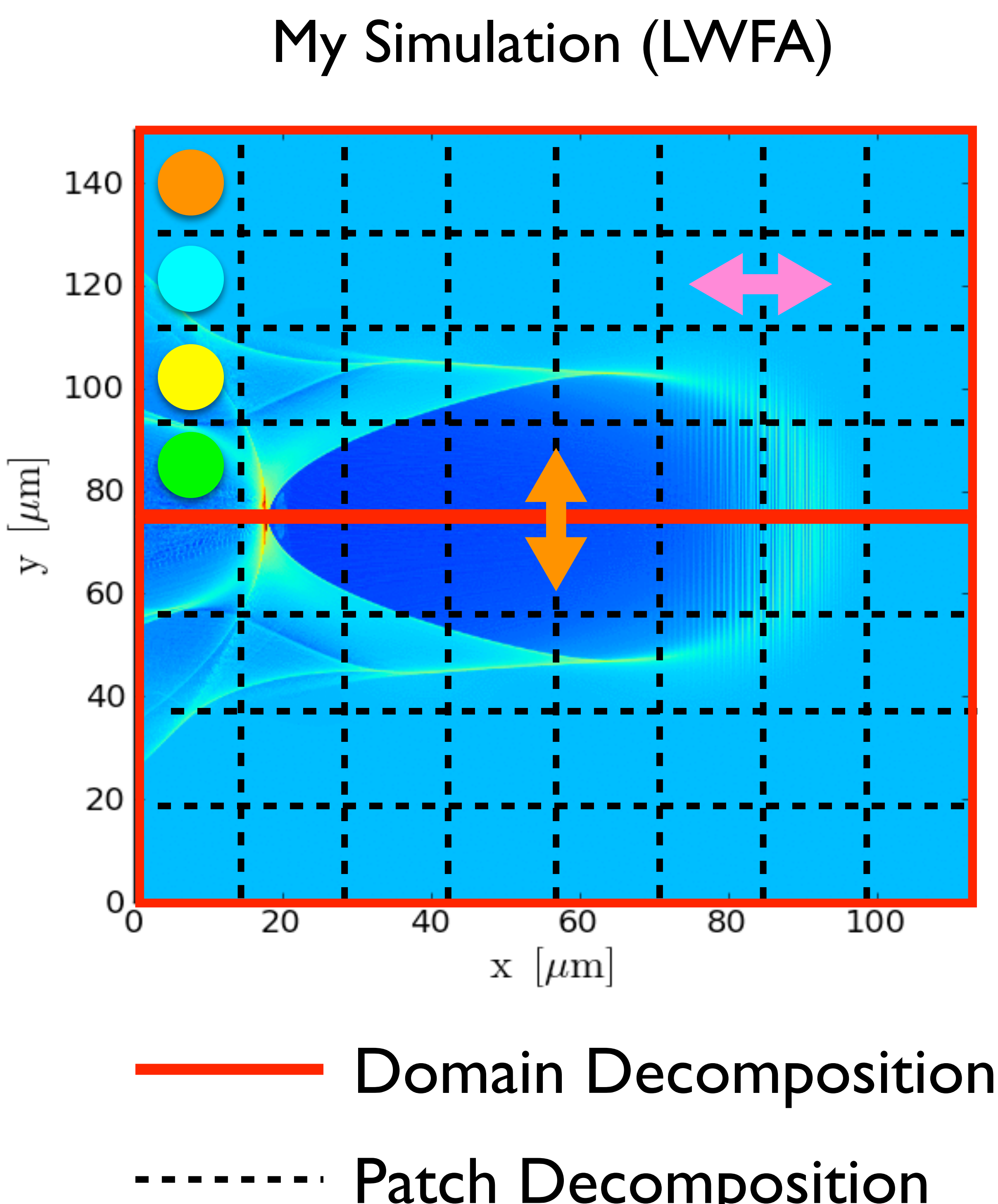
PIC codes are well adapted to massive parallelism

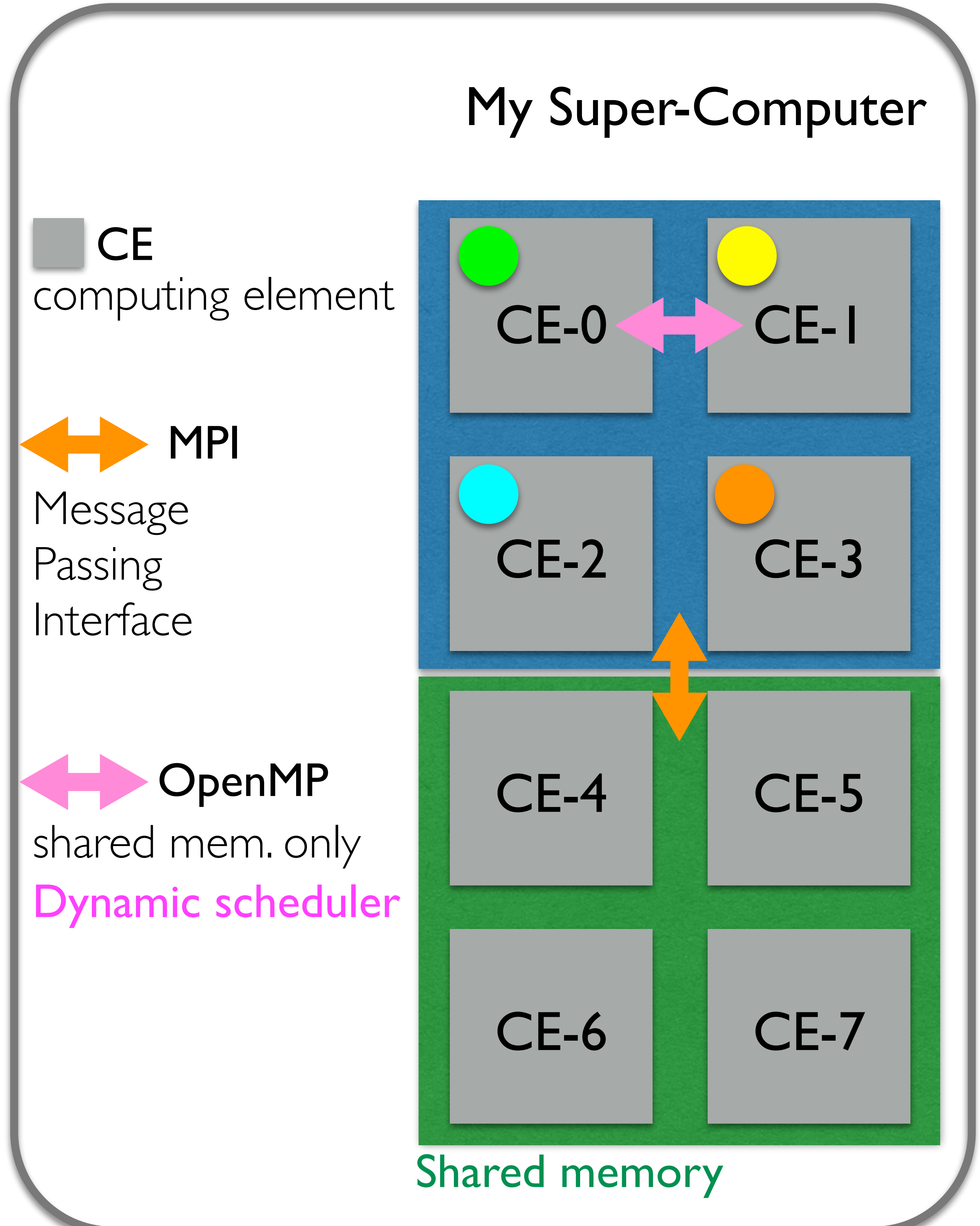




Step 1: Parallelization

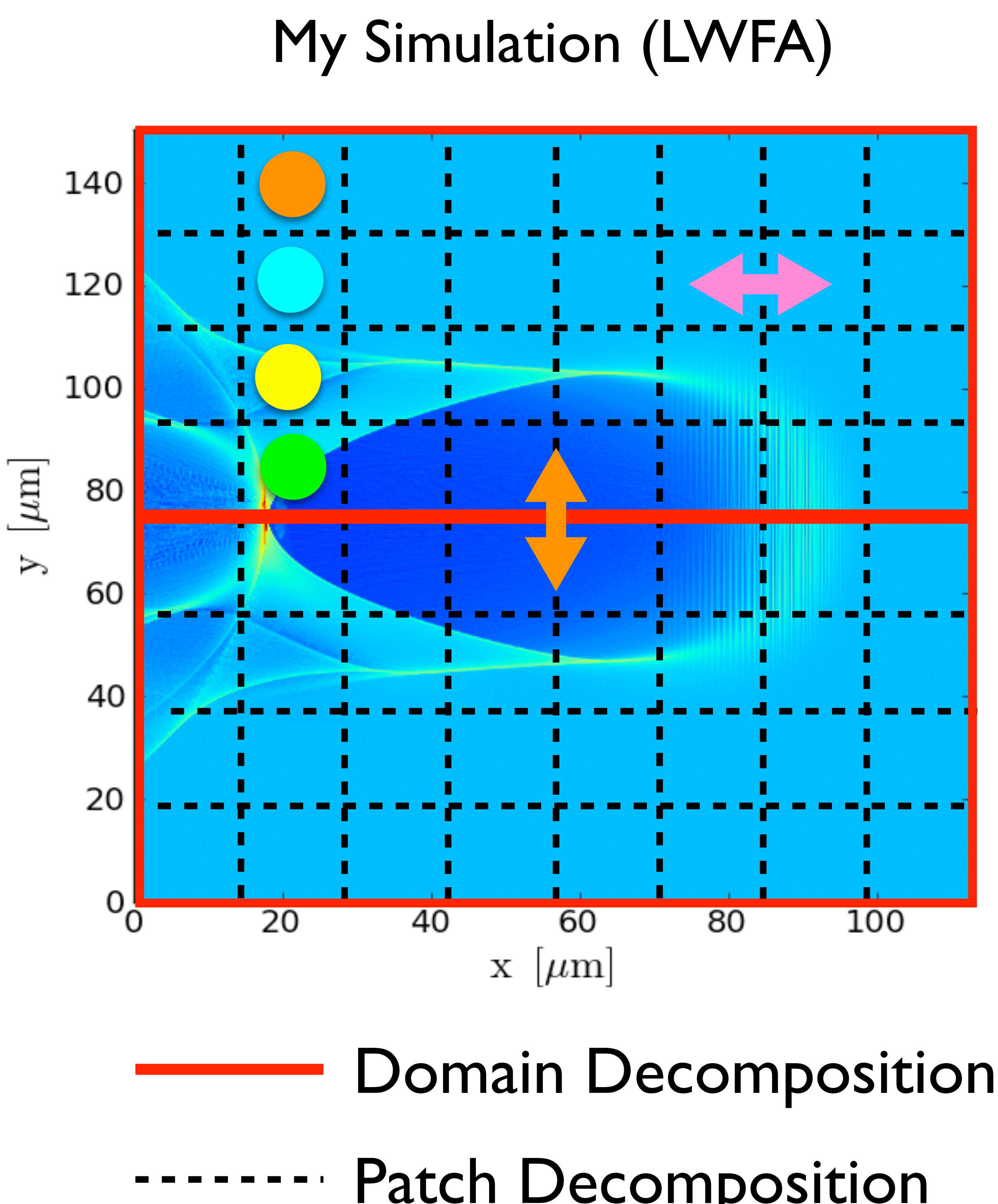
PIC codes are well adapted to massive parallelism

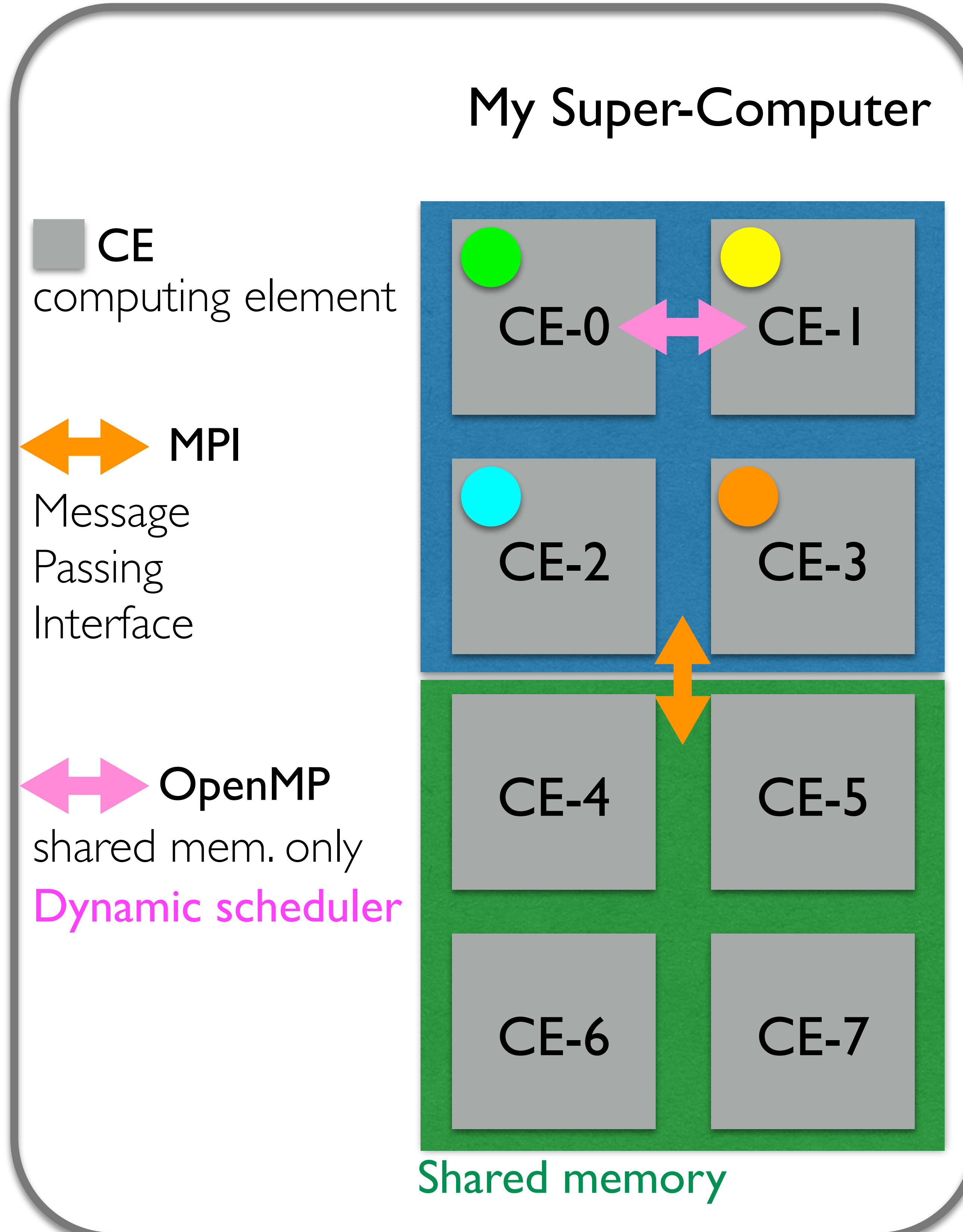




Step 1: Parallelization

PIC codes are well adapted to massive parallelism

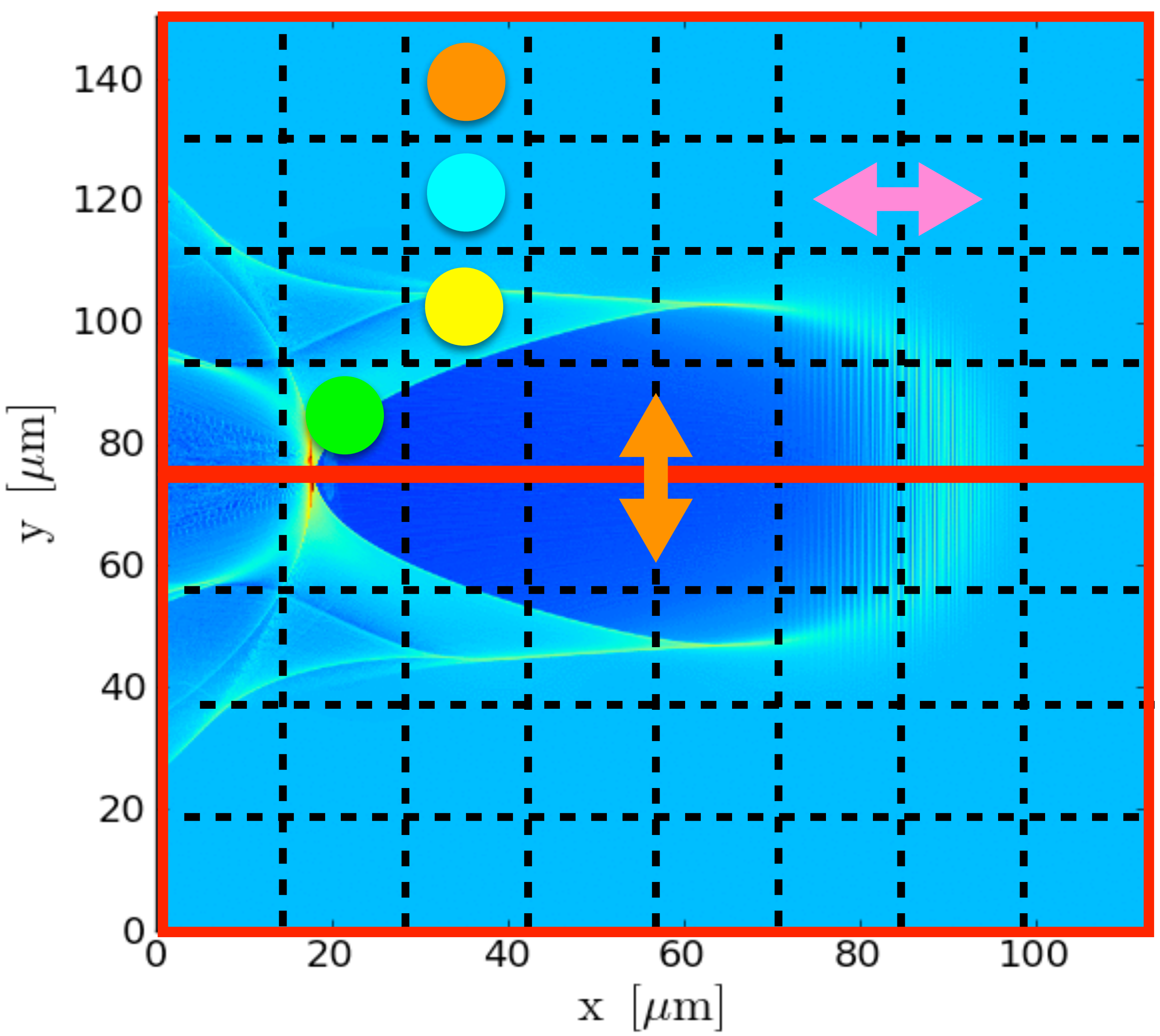




Step 1: Parallelization

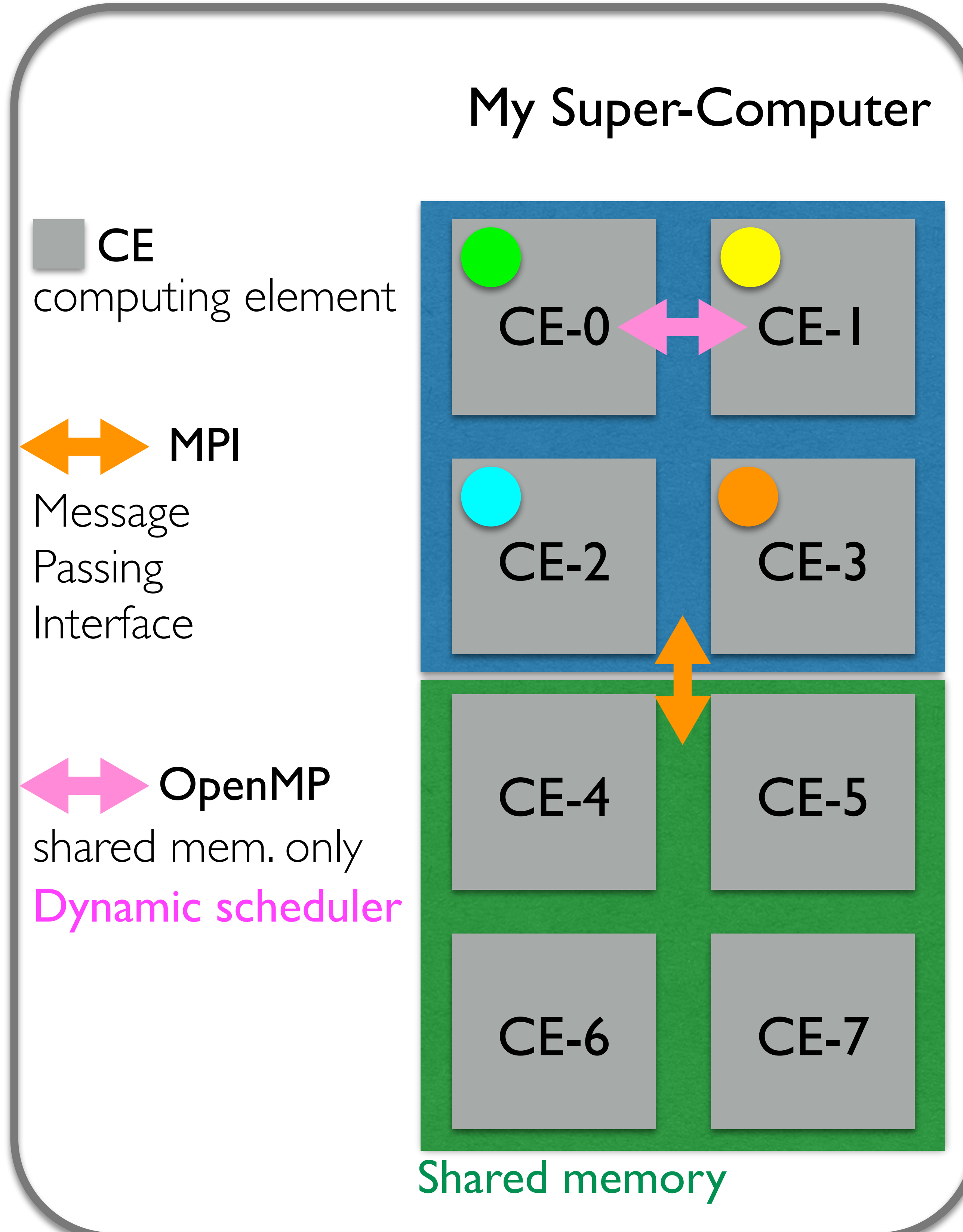
PIC codes are well adapted to massive parallelism

My Simulation (LWFA)



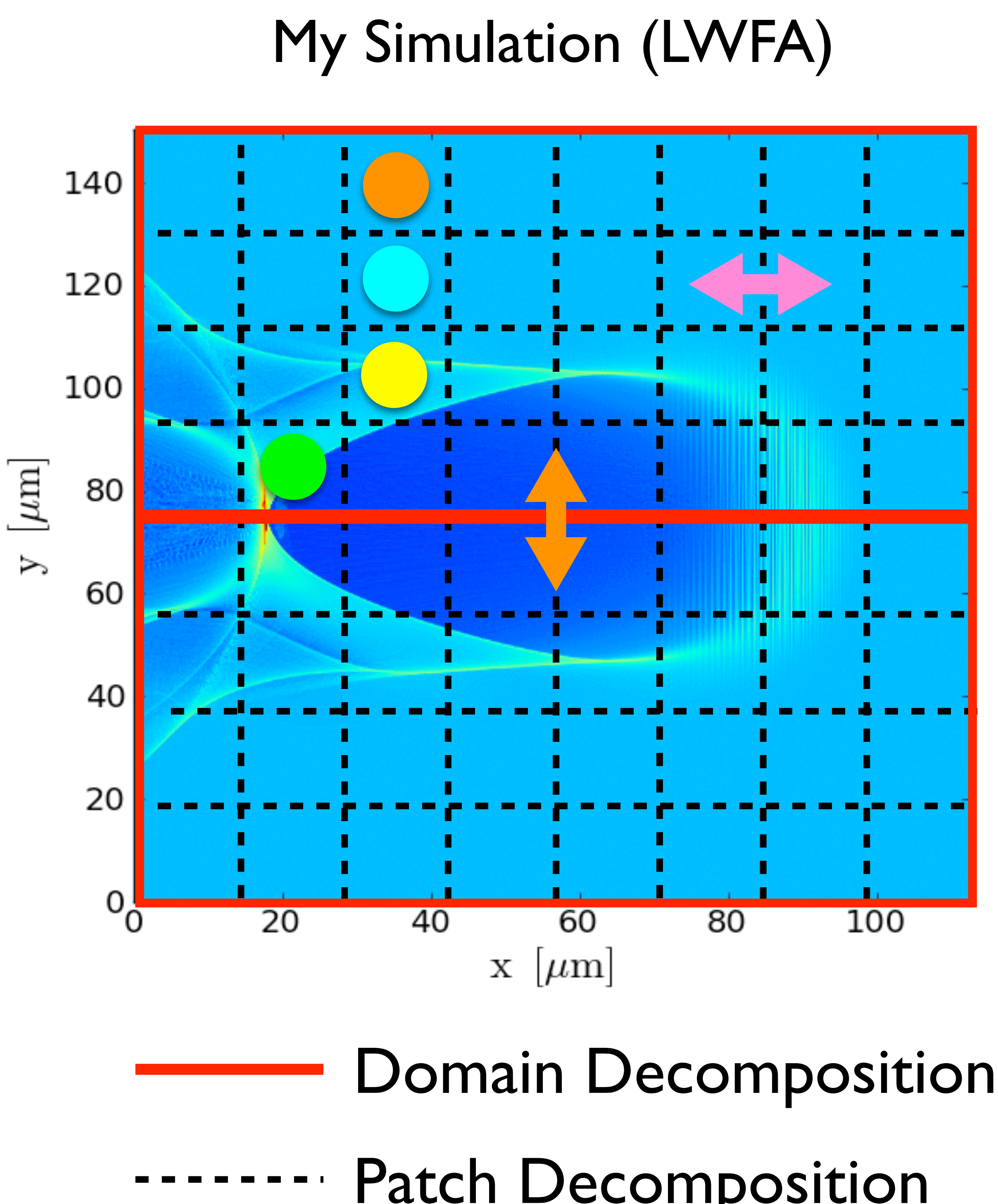
Domain Decomposition

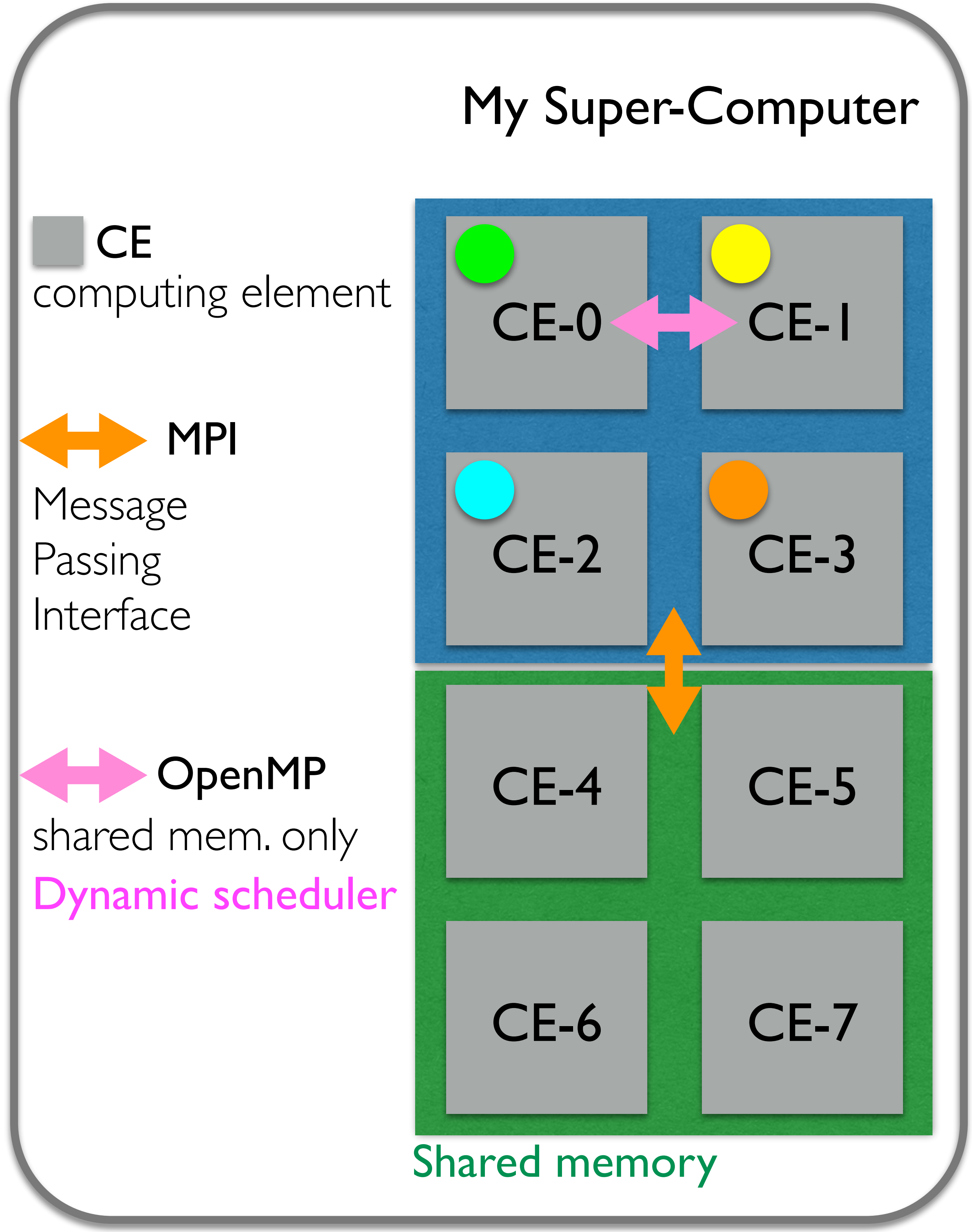
Patch Decomposition



Step 1: Parallelization

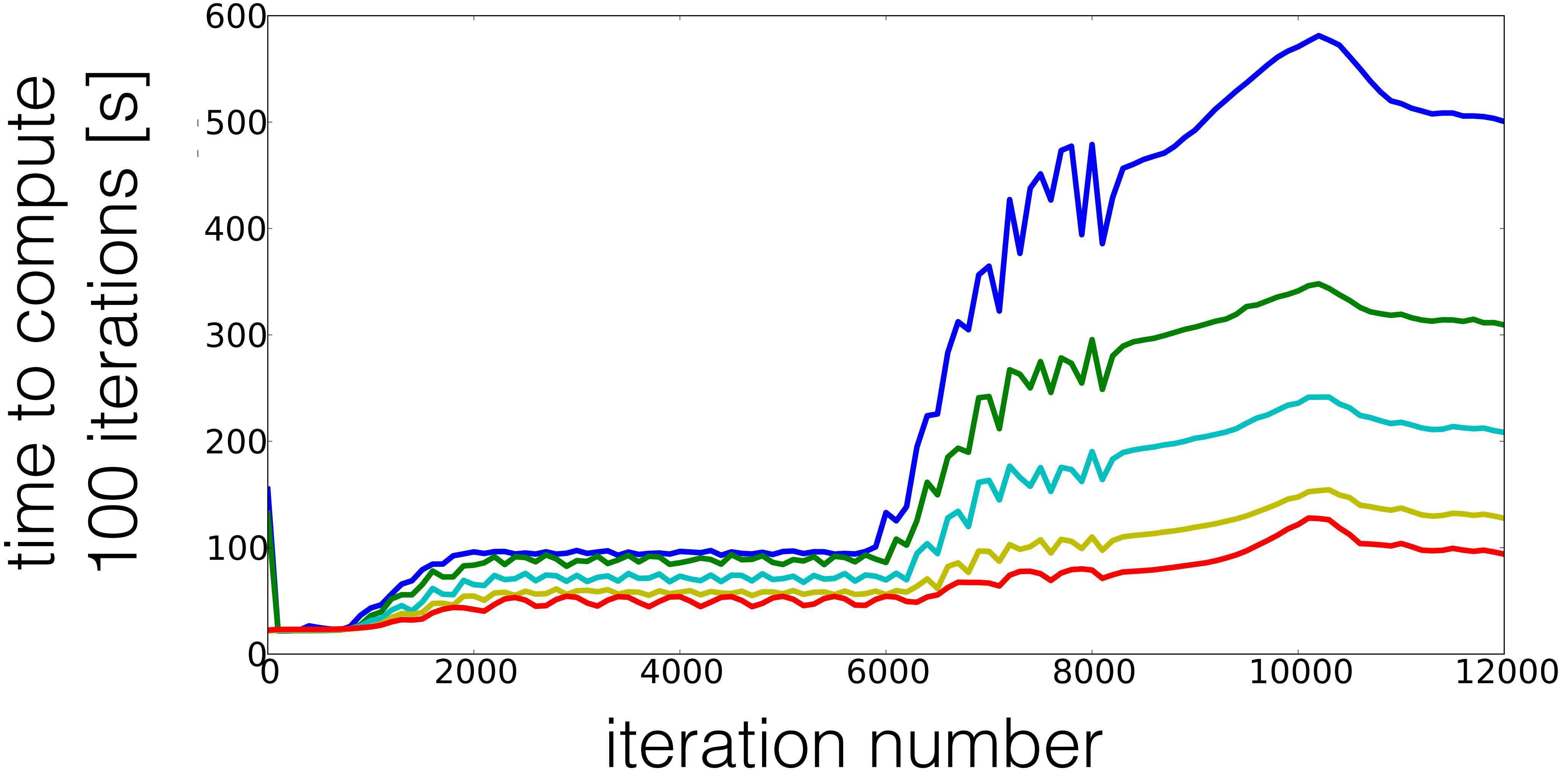
PIC codes are well adapted to massive parallelism





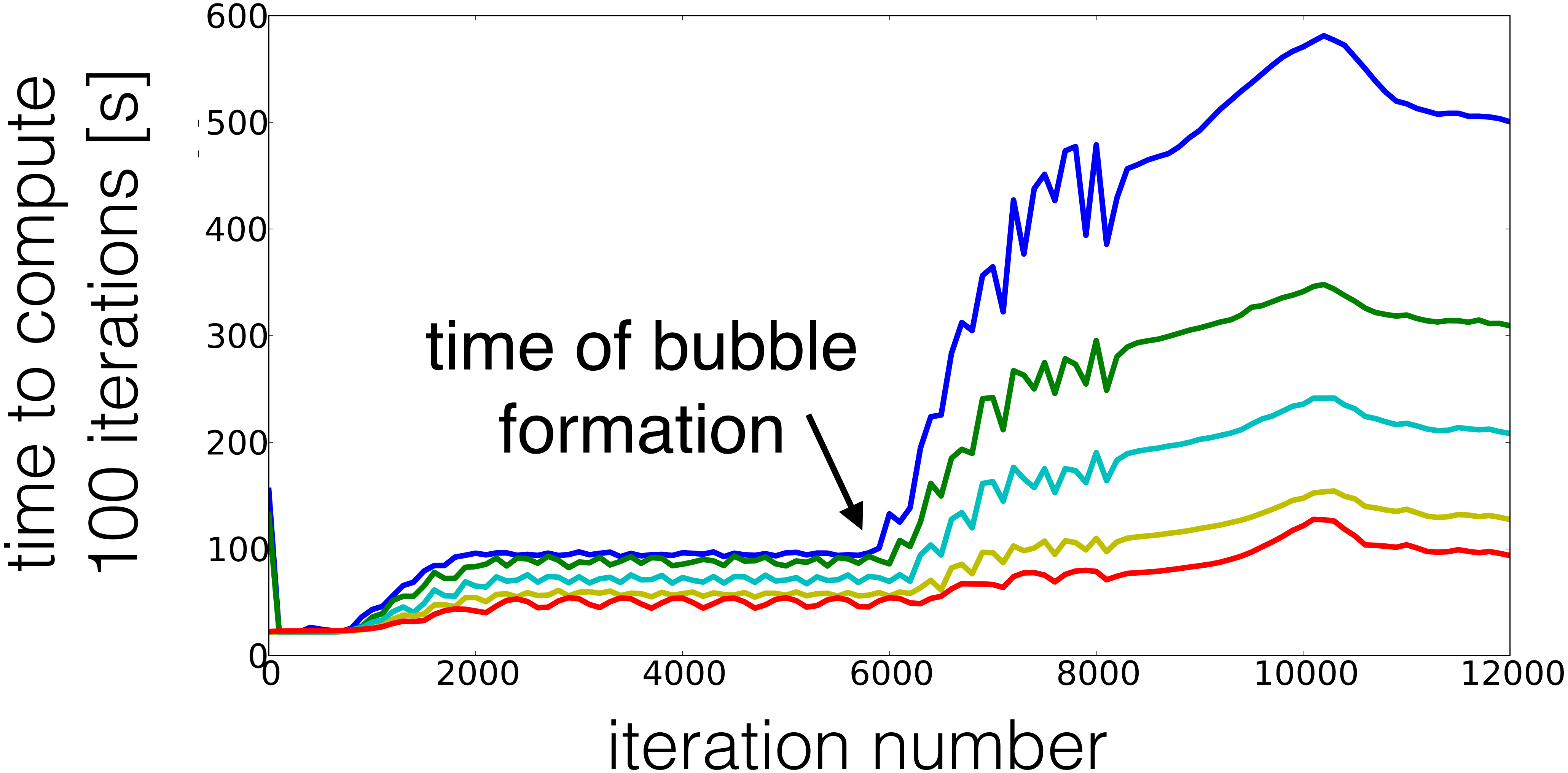
Step 1: Parallelization

Hybrid parallelization significantly improves performance



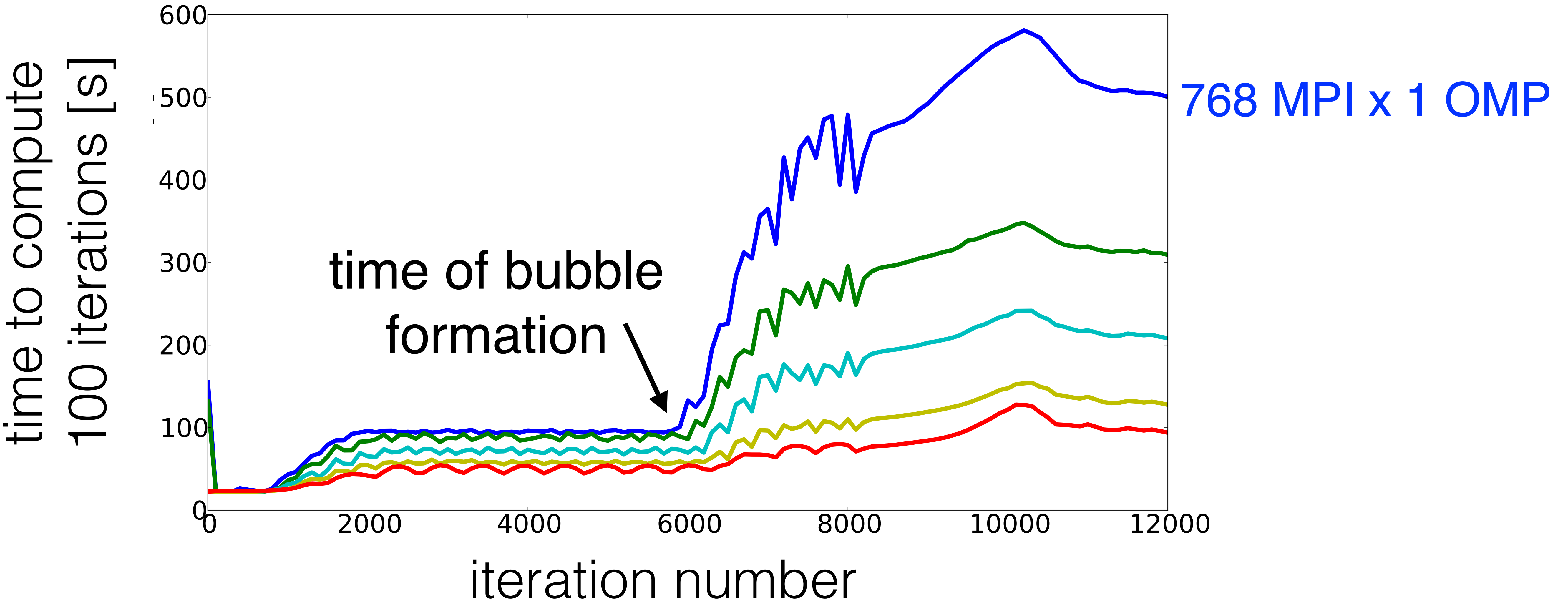
Step 1: Parallelization

Hybrid parallelization significantly improves performance



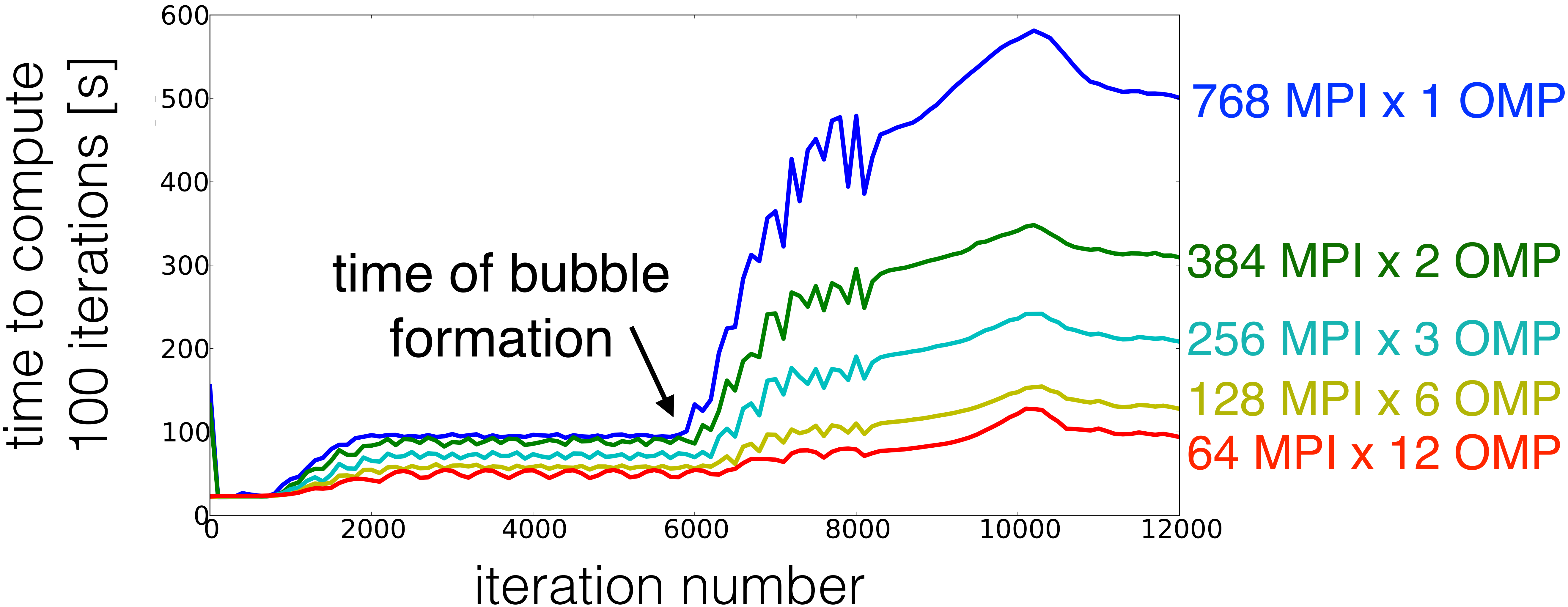
Step 1: Parallelization

Hybrid parallelization significantly improves performance



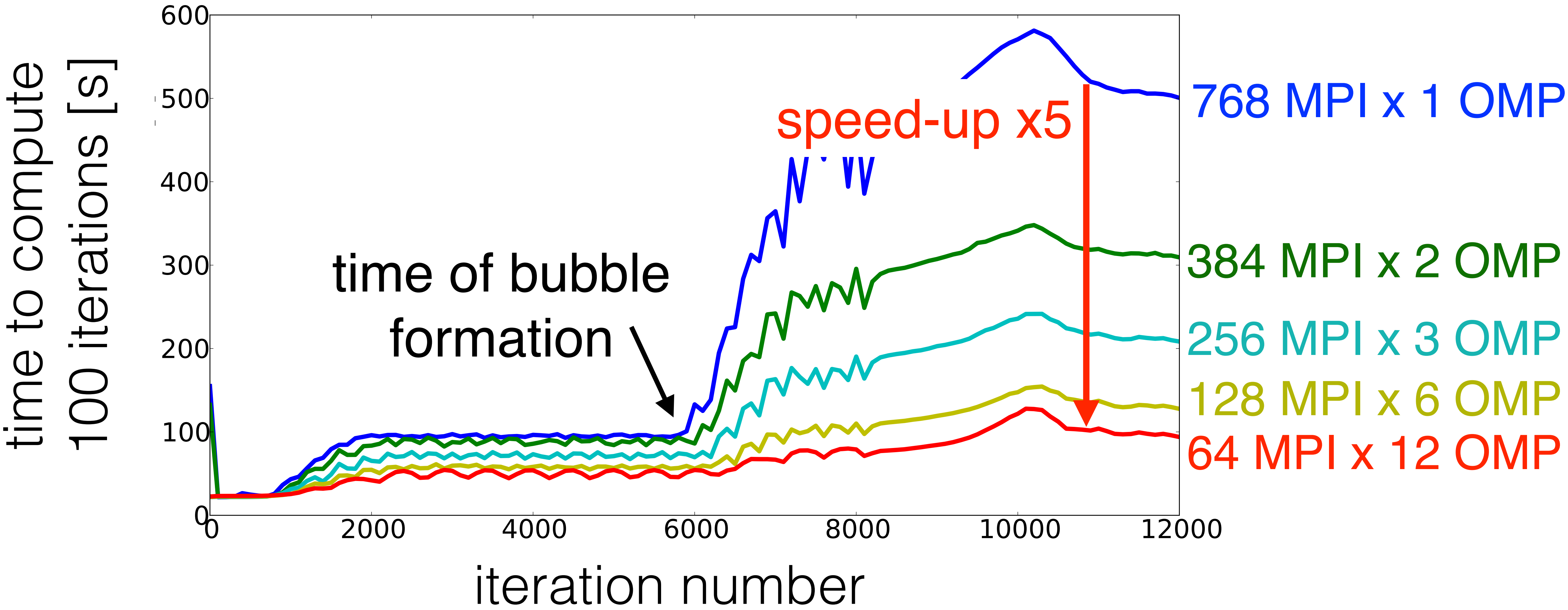
Step 1: Parallelization

Hybrid parallelization significantly improves performance



Step 1: Parallelization

Hybrid parallelization significantly improves performance

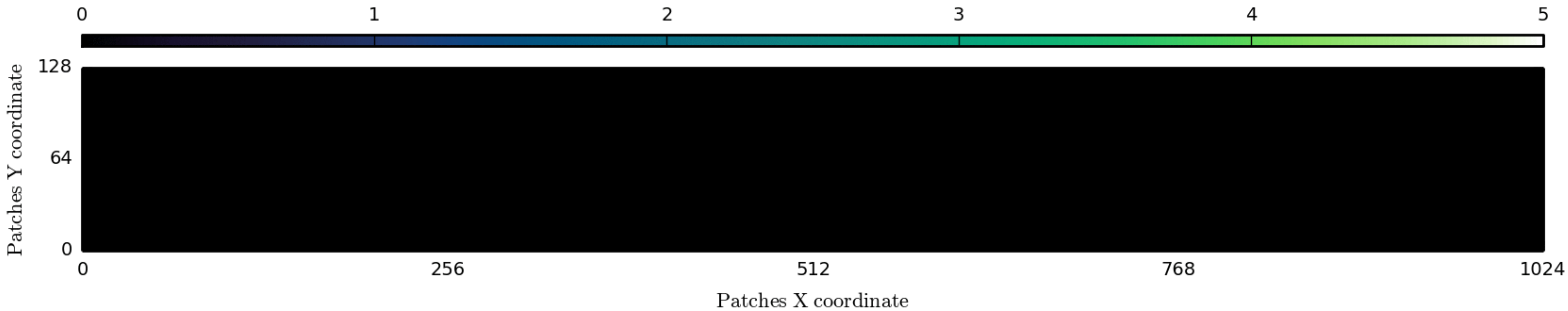


Step 1: Parallelization

Dynamic load balancing further improve performances

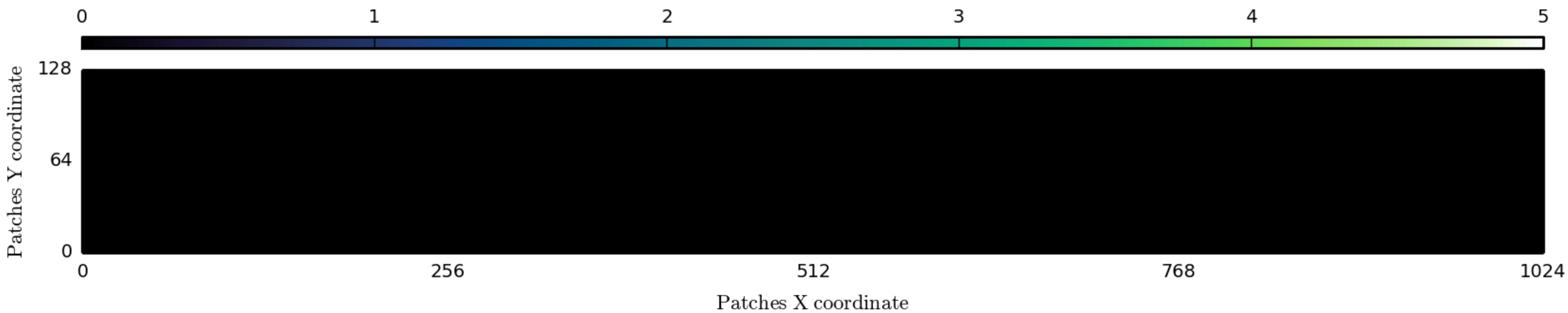
Step 1: Parallelization

Dynamic load balancing further improve performances



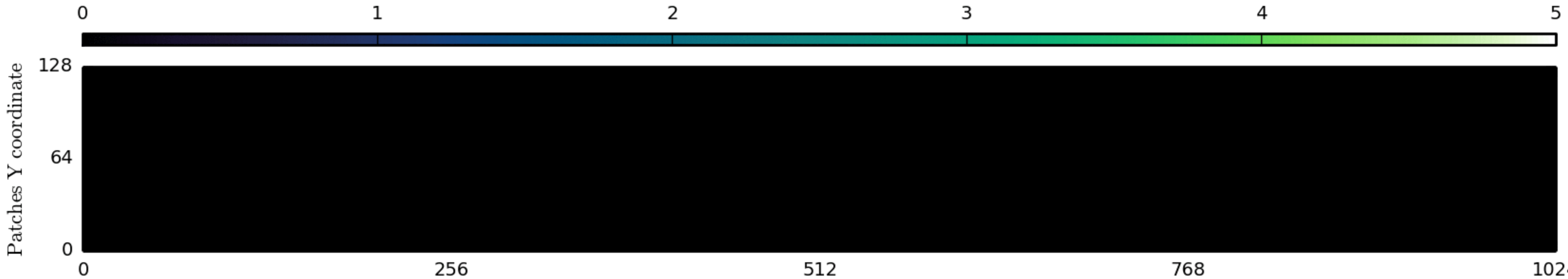
Step 1: Parallelization

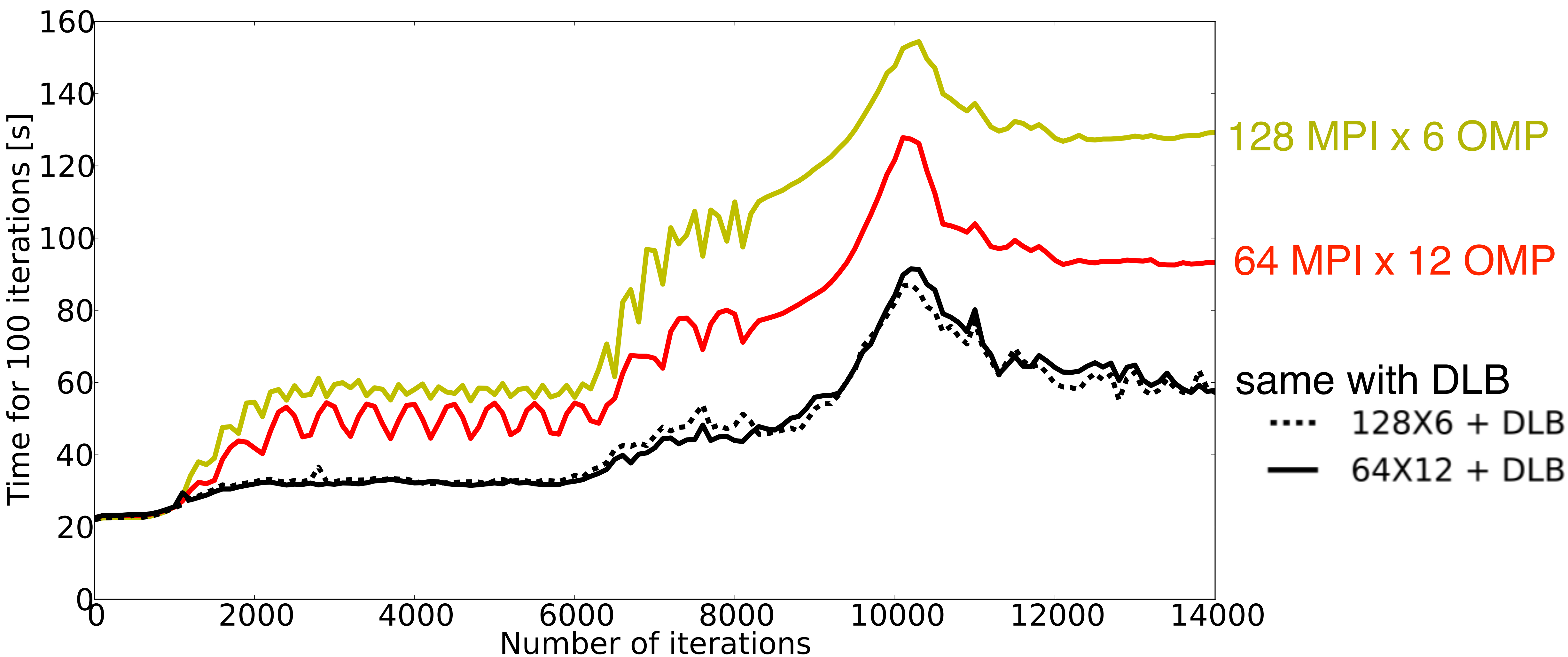
Dynamic load balancing further improve performances



Step 1: Parallelization

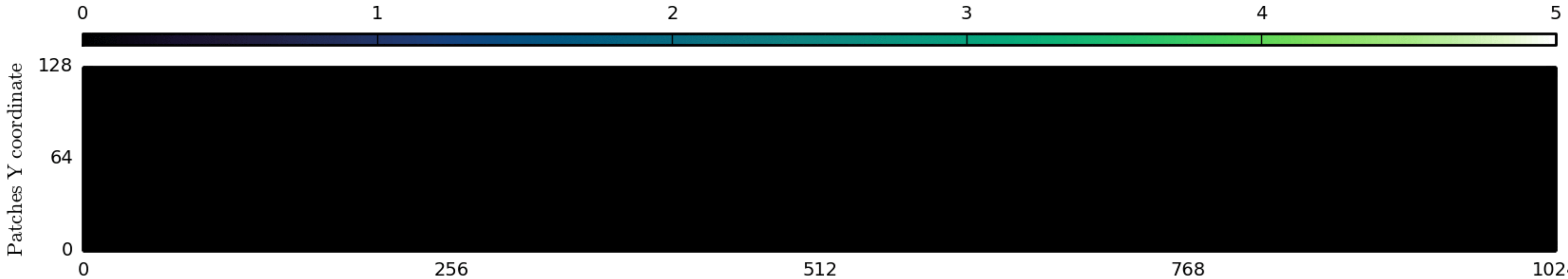
Dynamic load balancing further improve performances

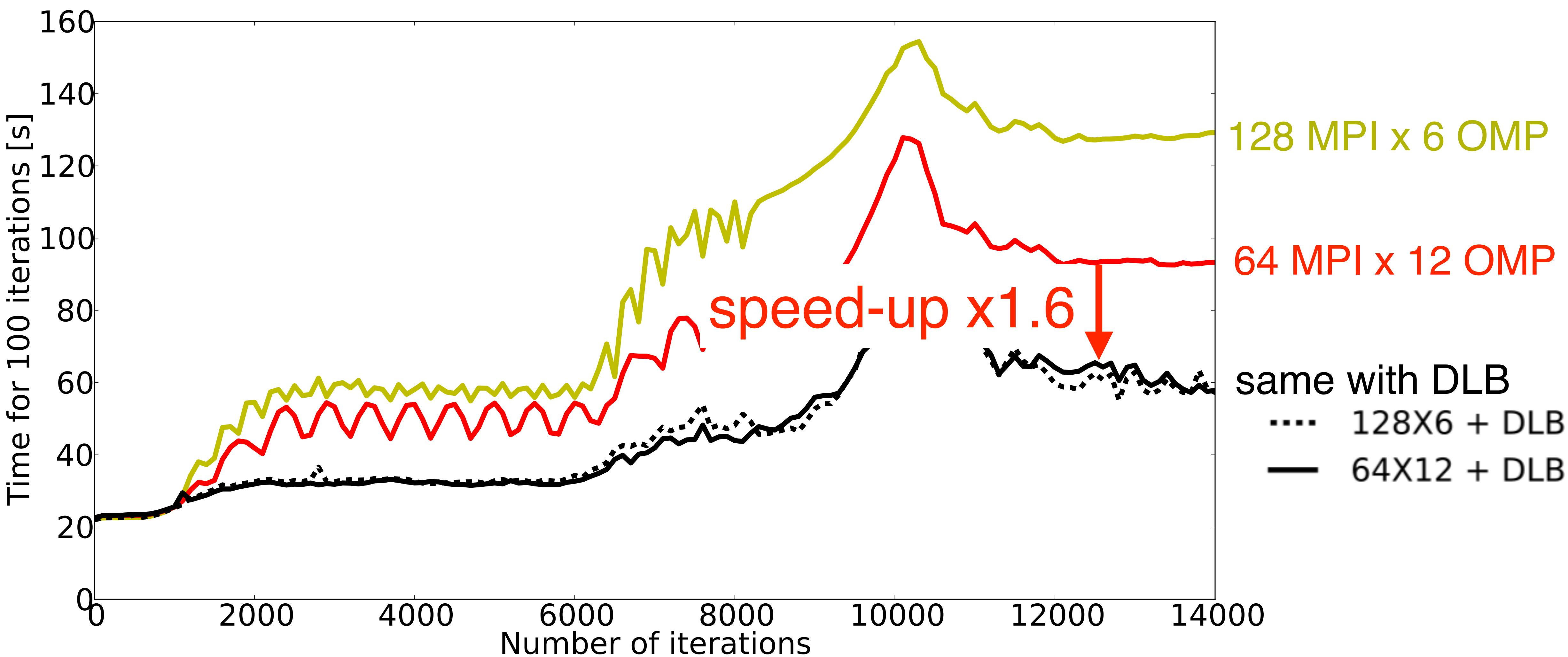




Step 1: Parallelization

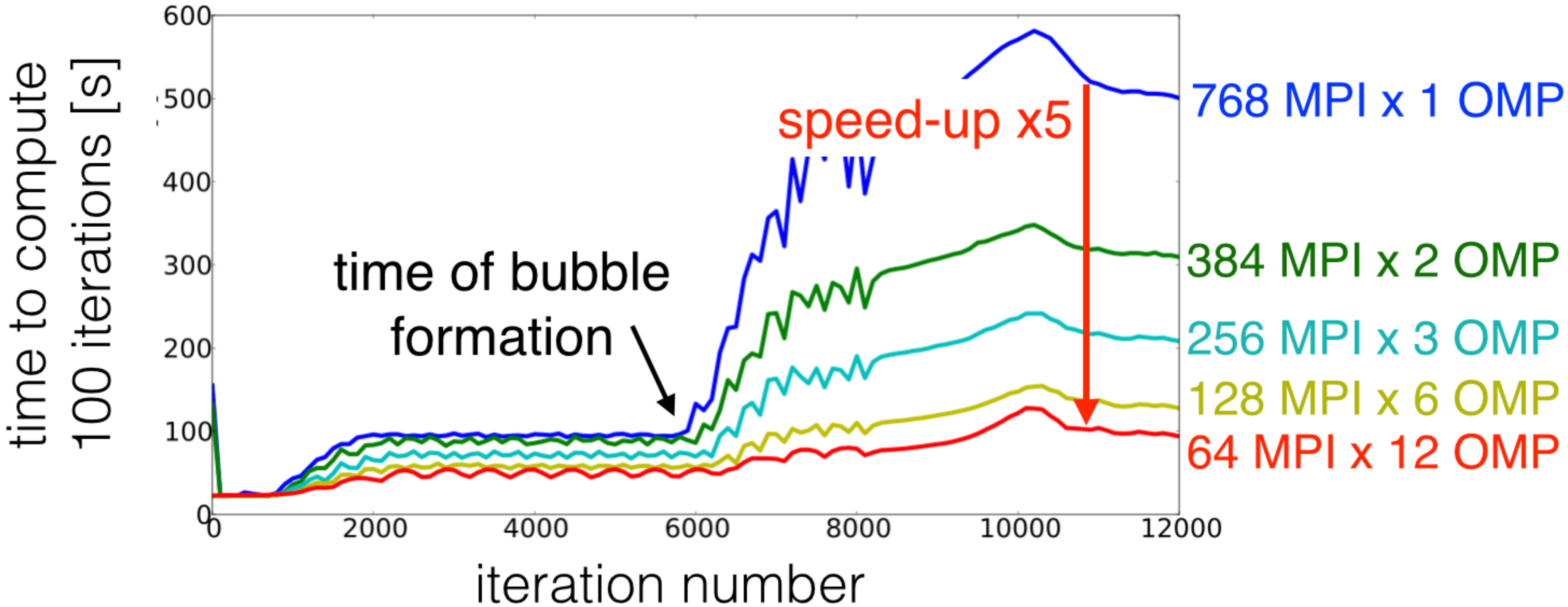
Dynamic load balancing further improve performances

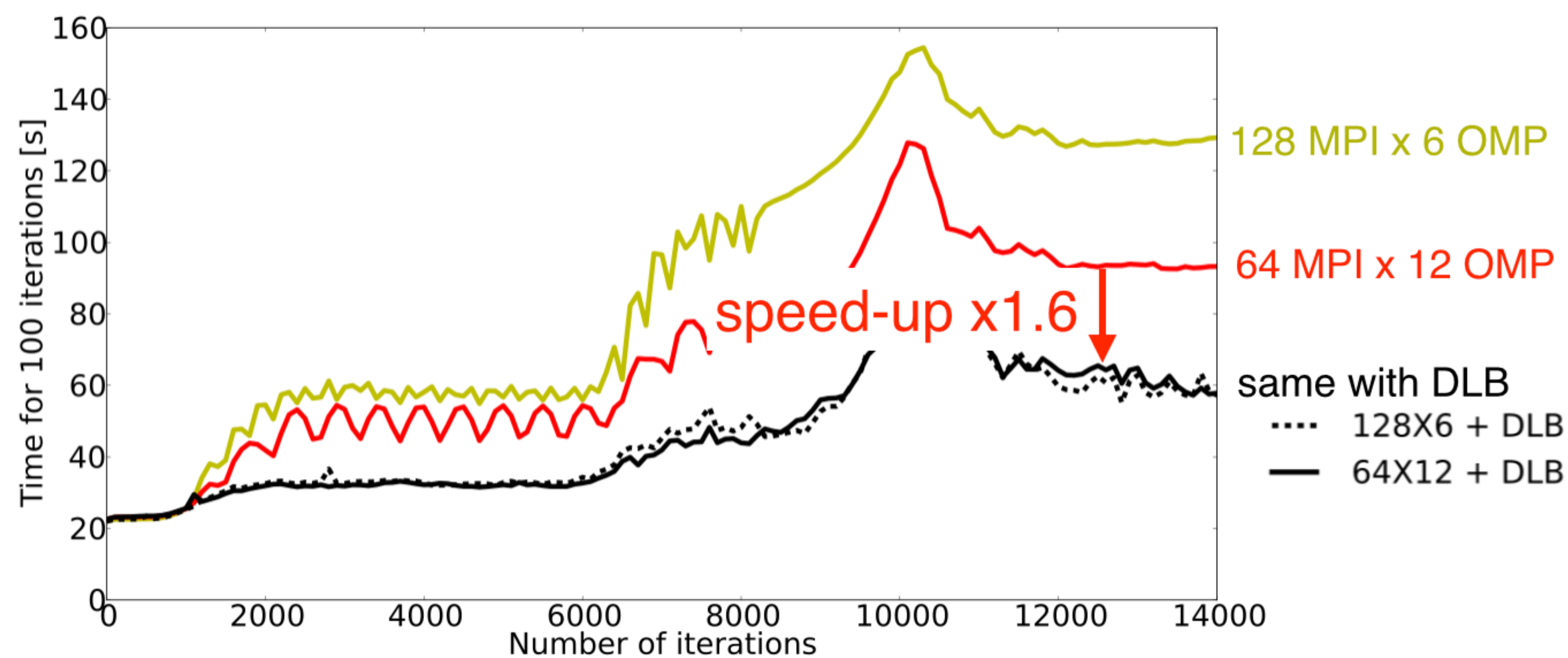




Step 1: Parallelization

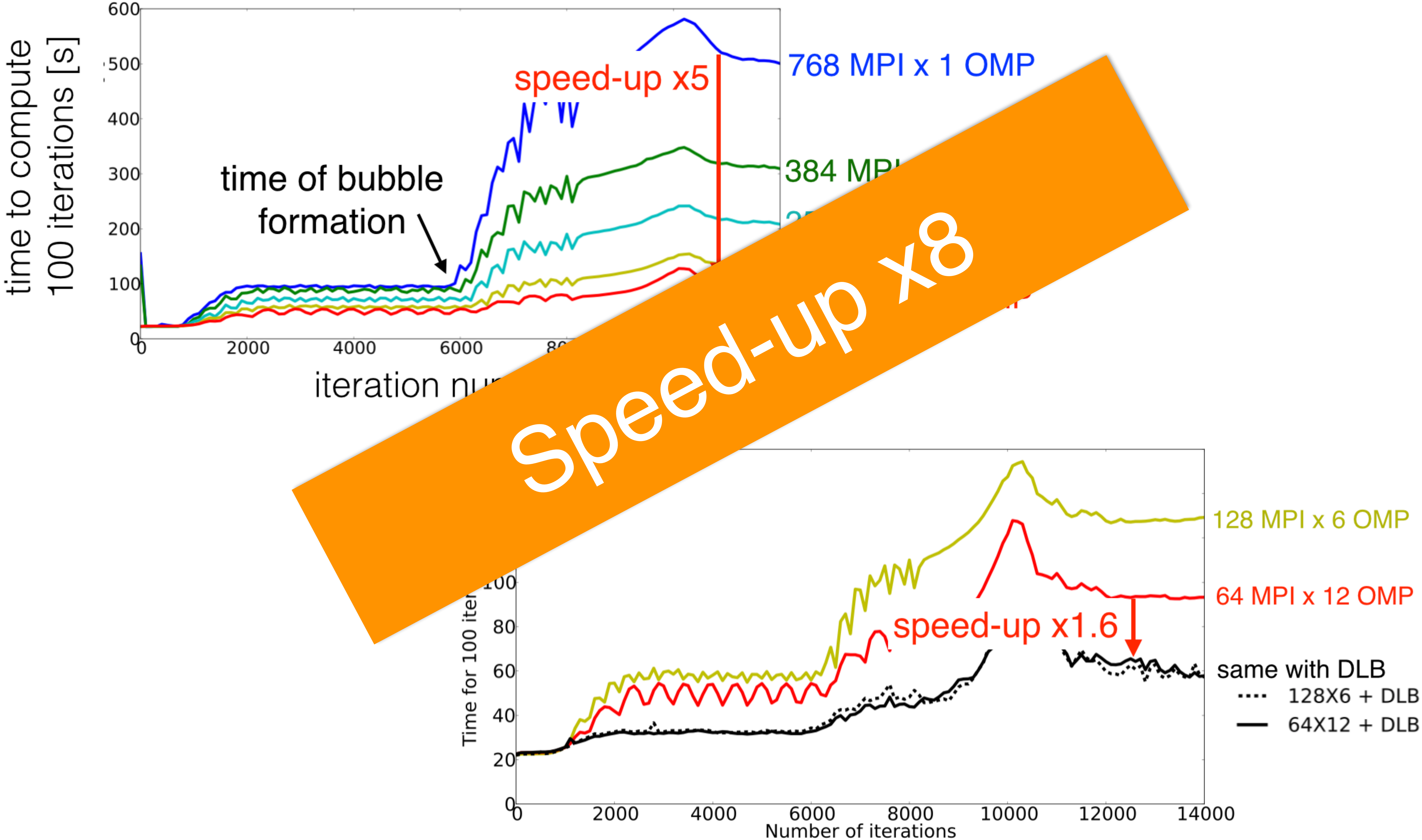
Hybrid + Dynamic Load Balancing





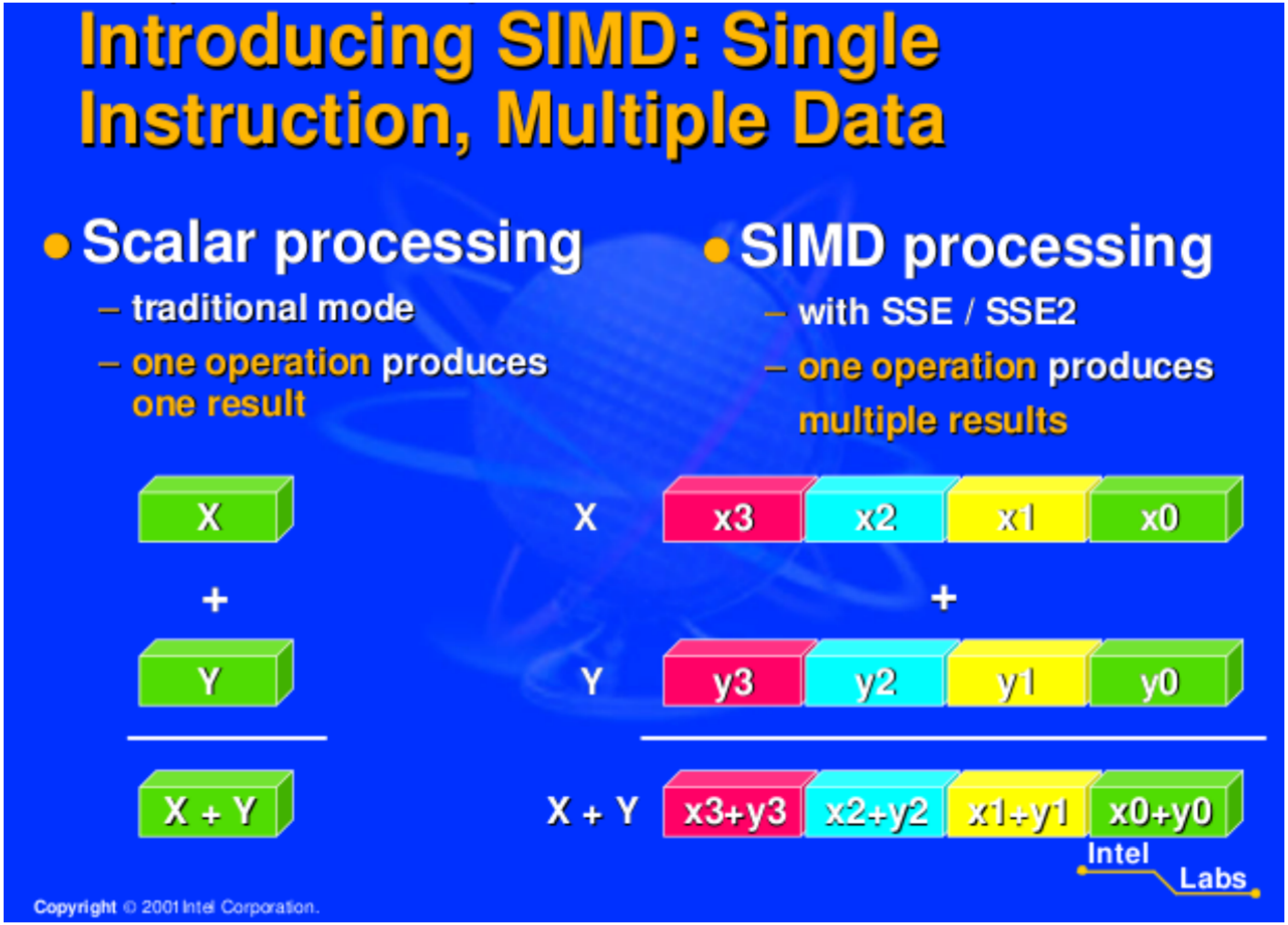
Step 1: Parallelization

Hybrid + Dynamic Load Balancing

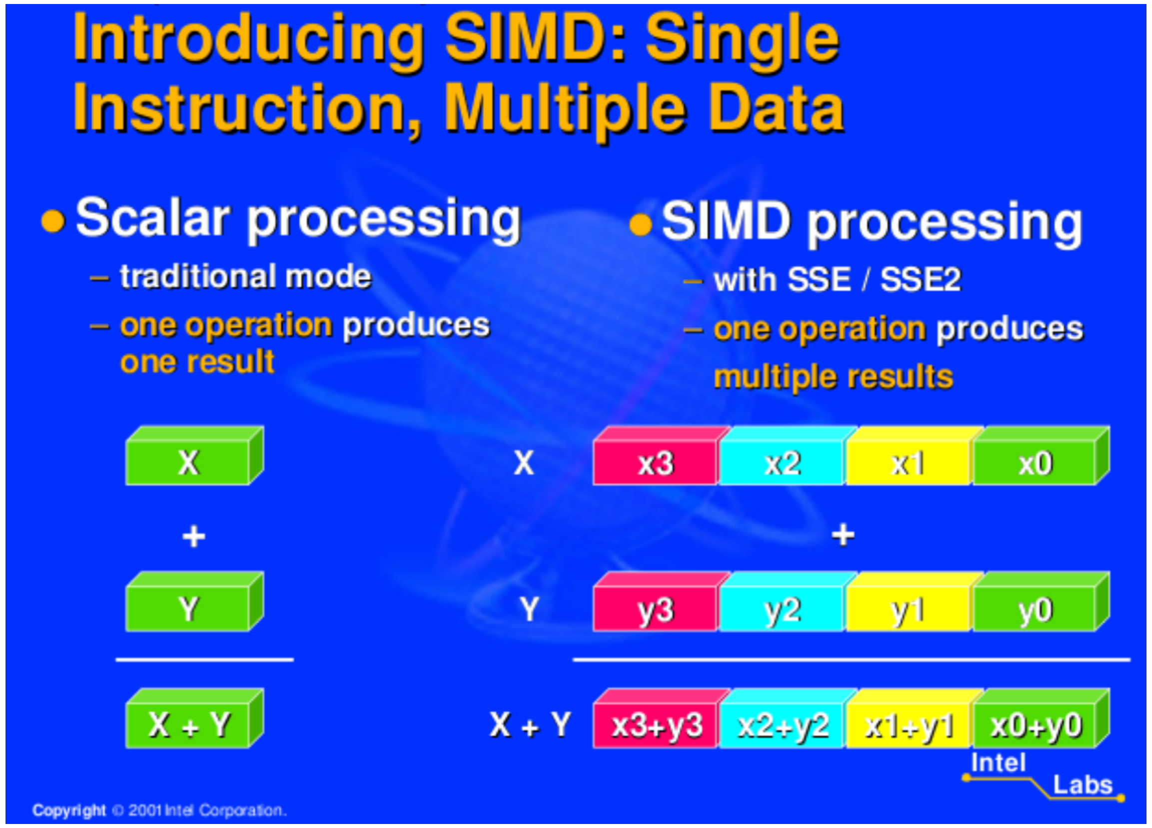


Step 2: Vectorization

Vectorization in a nutshell



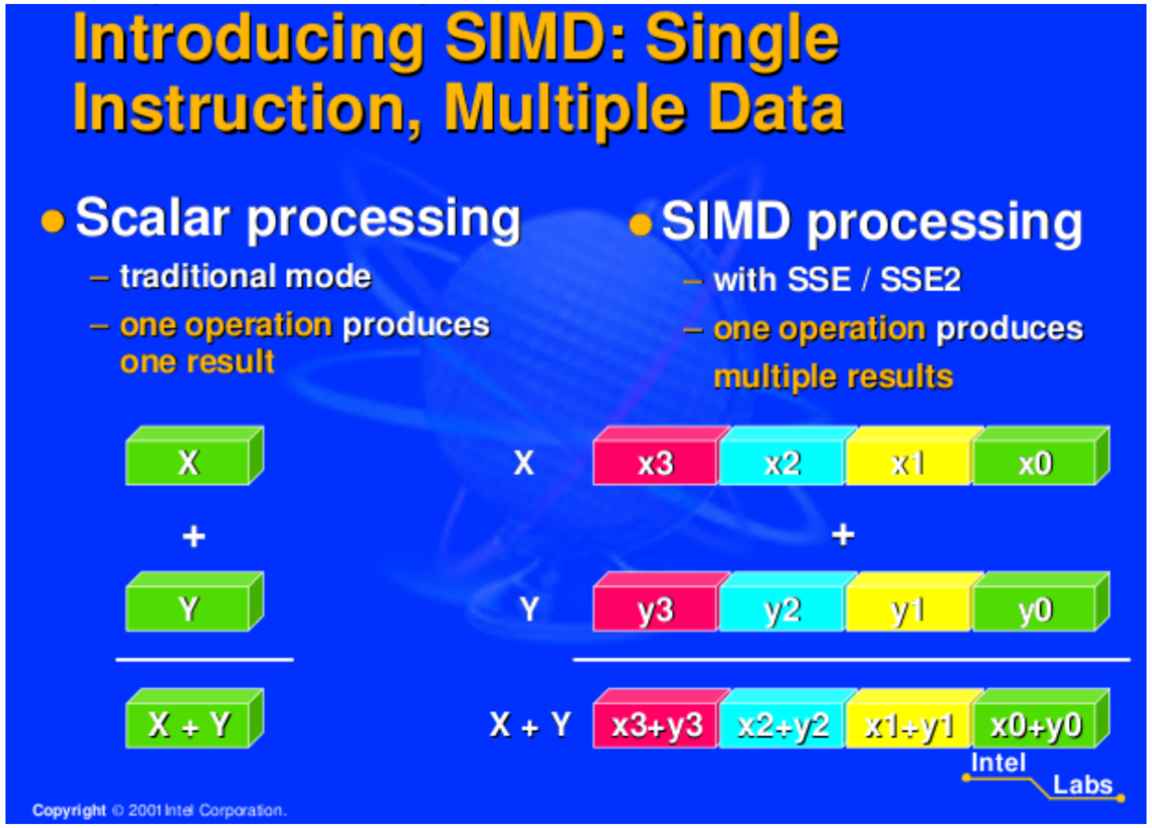
Vectorization in a nutshell



Beck *et al.*, arXiv:1810.03949

Step 2: Vectorization

Vectorization in a nutshell



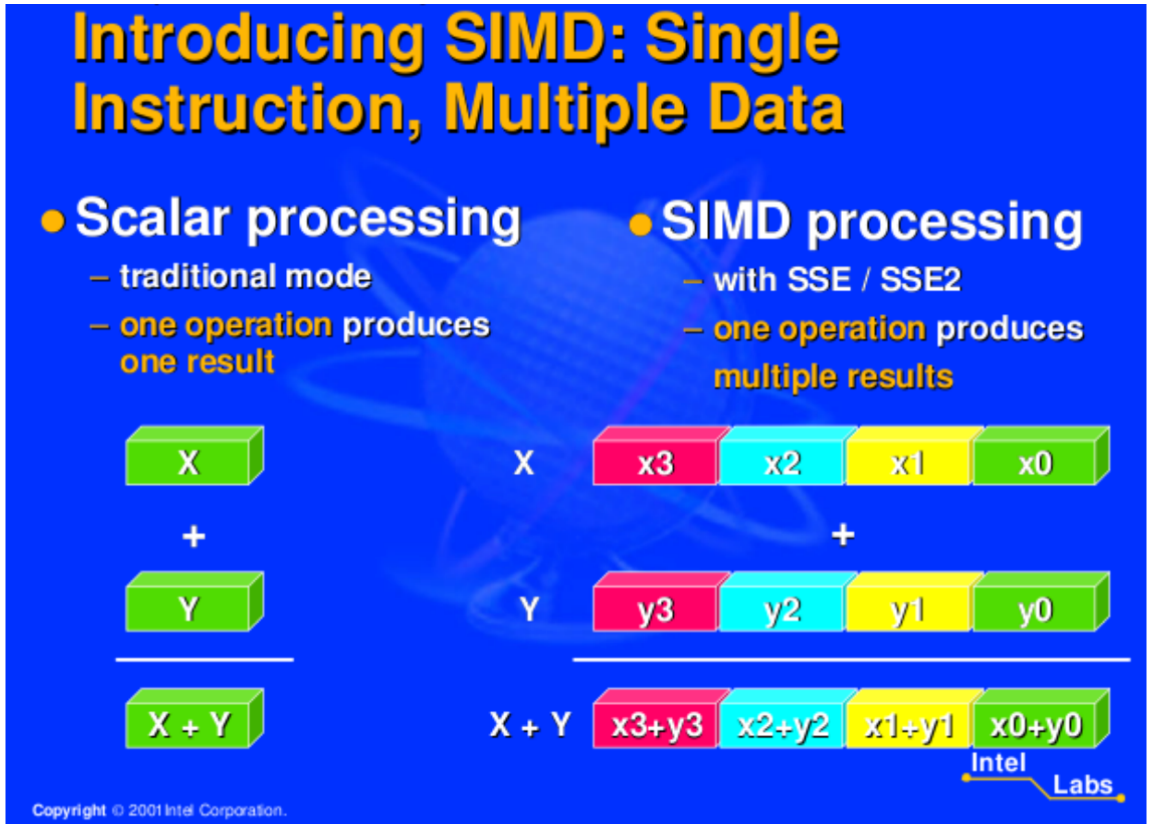
Smart (particles) operators:

- interpolator, pusher, projector

Beck *et al.*, arXiv:1810.03949

Step 2: Vectorization

Vectorization in a nutshell



Smart (particles) operators:

- interpolator, pusher, projector

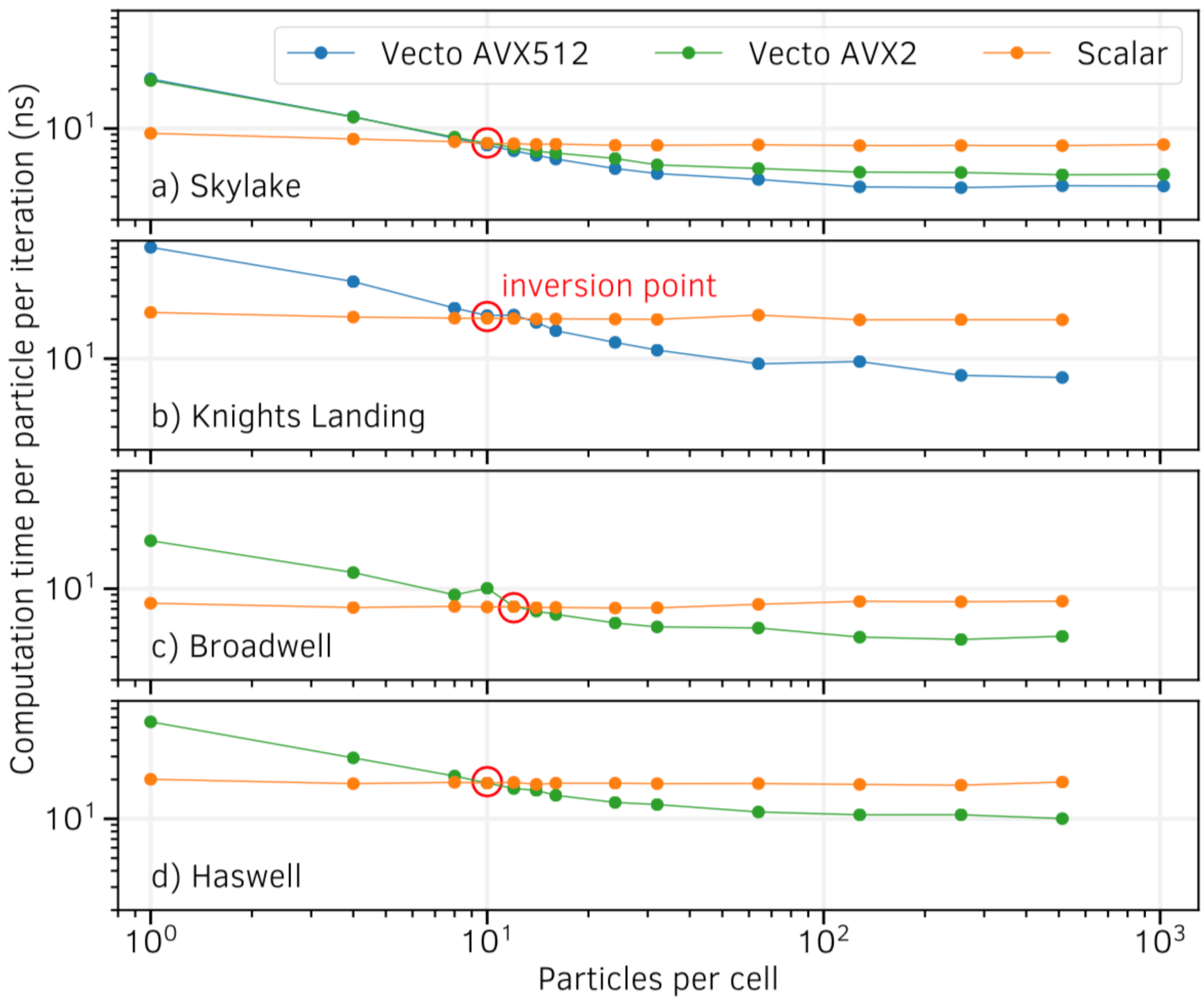
Smart (particles) data structures:

- beware random mem. access
- contiguous memory
- sort at all times!

Beck *et al*., arXiv:1810.03949

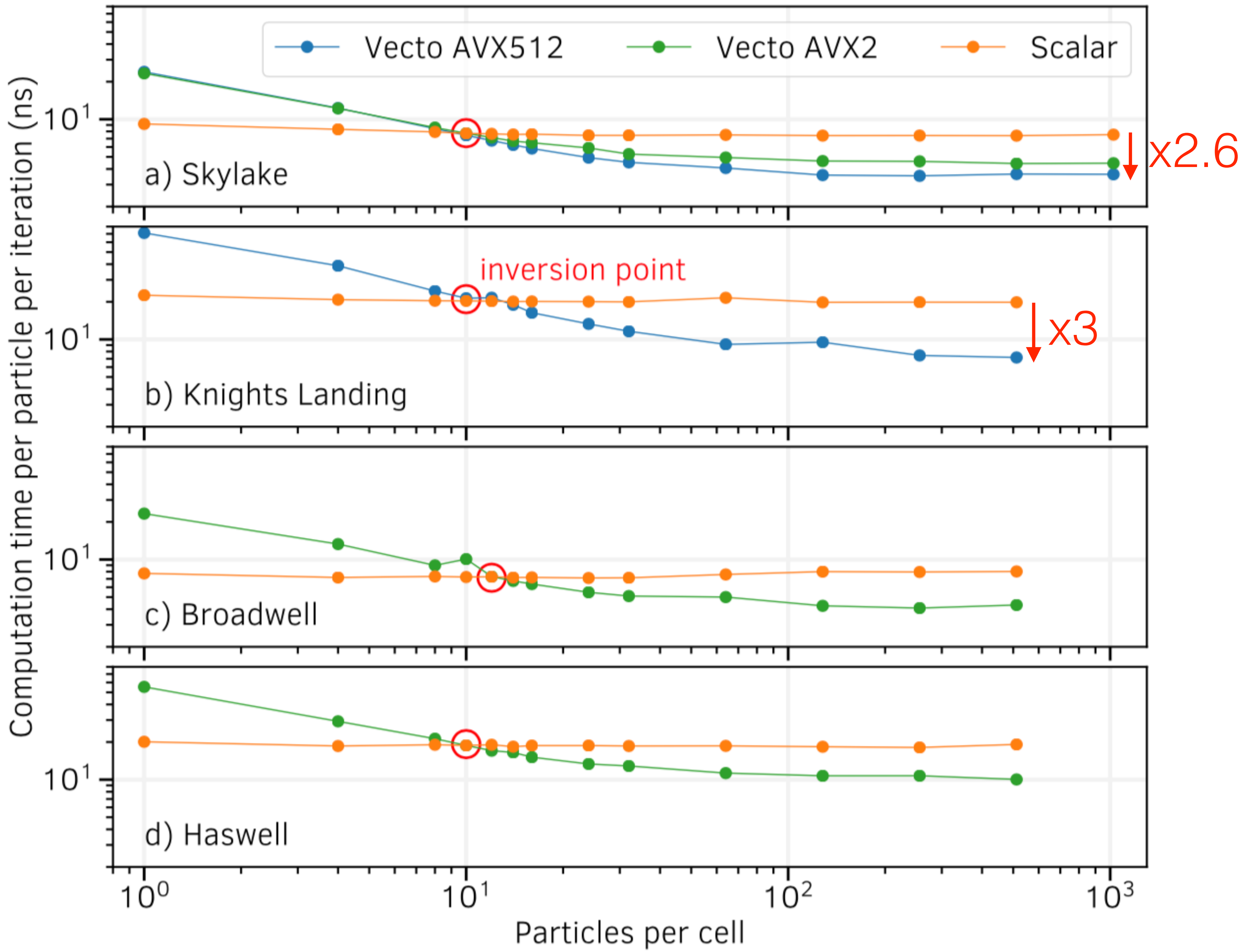
Step 2: Vectorization

SMILEI uses an adaptive vectorization approach



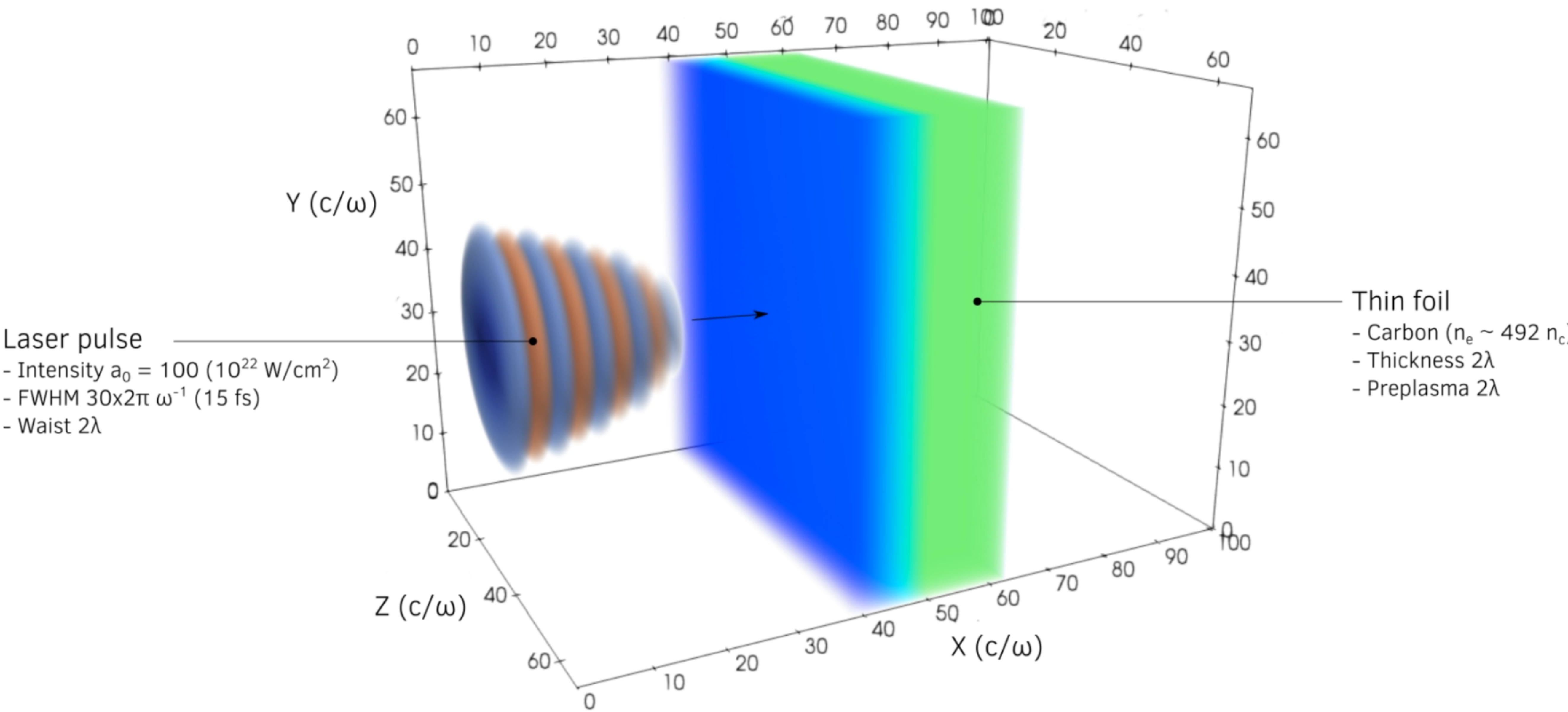
Step 2: Vectorization

SMILEI uses an adaptive vectorization approach



Step 2: Vectorization

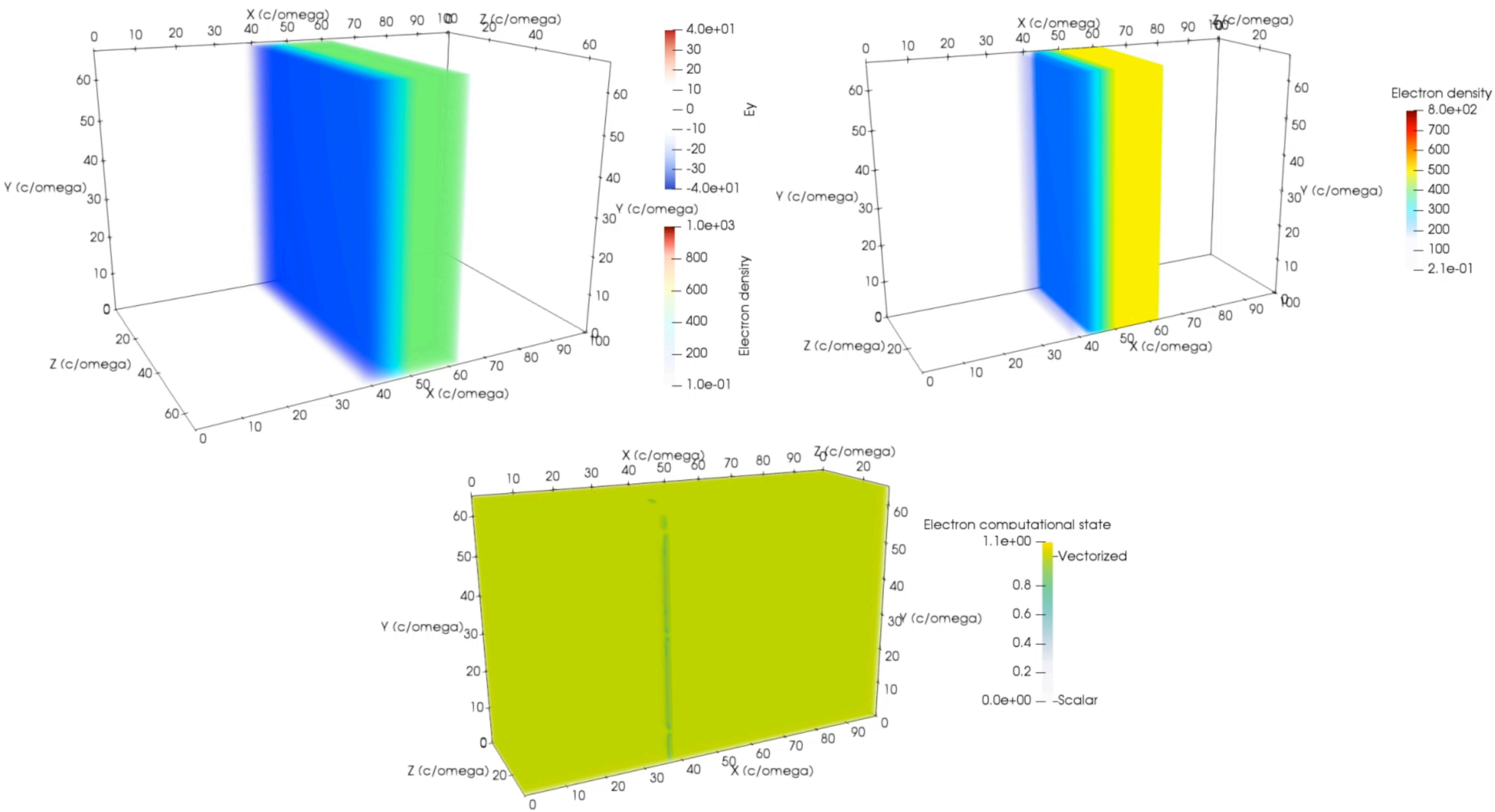
Laser-driven hole-boring



@ 32 PPC : speed-up x 1.5

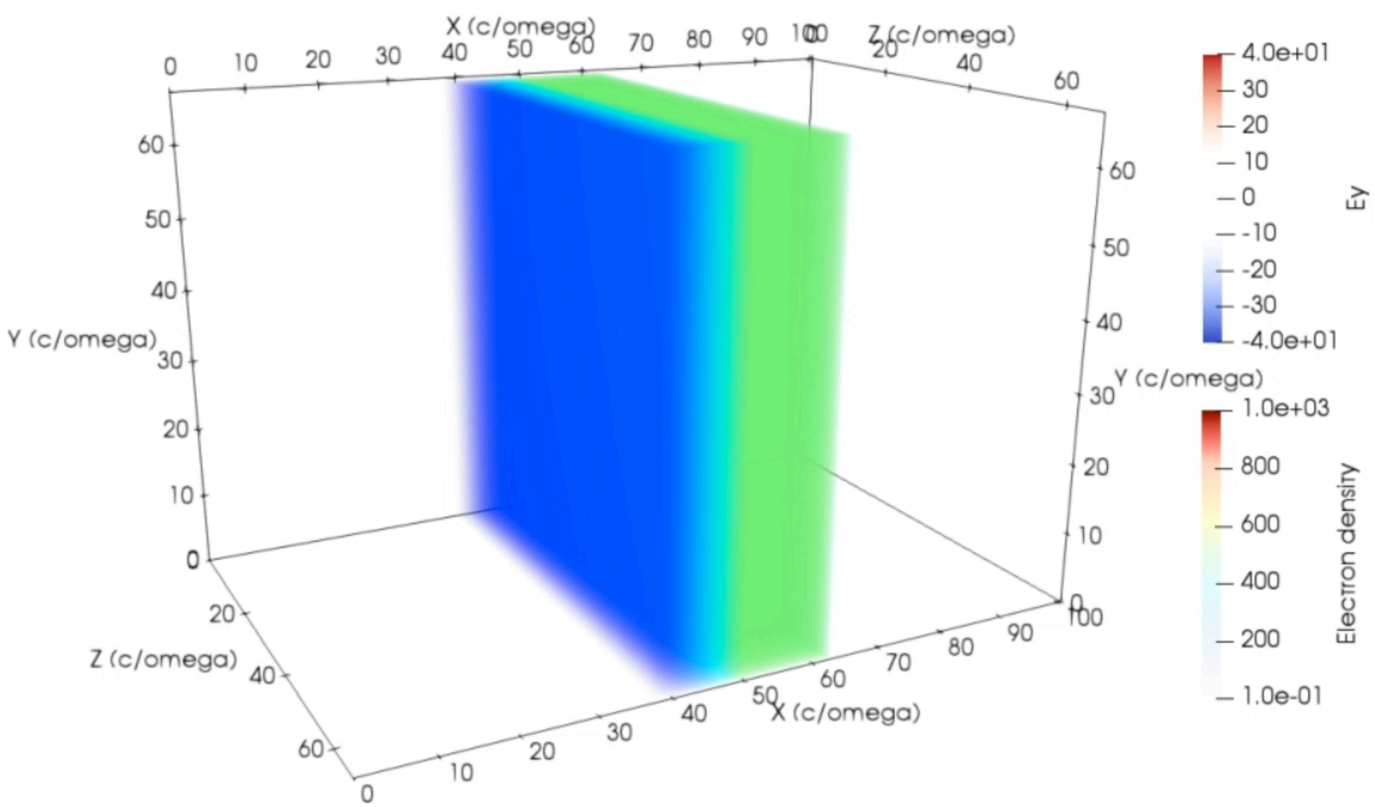
Step 2: Vectorization

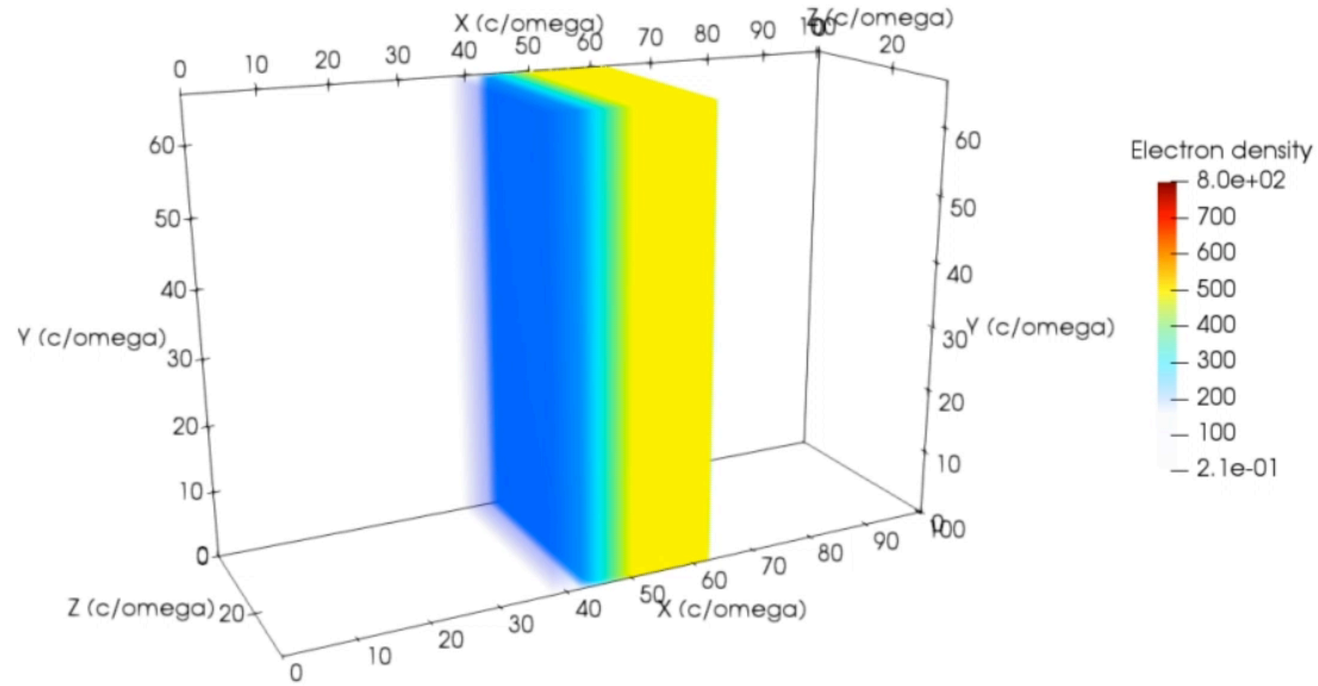
Laser-driven hole-boring

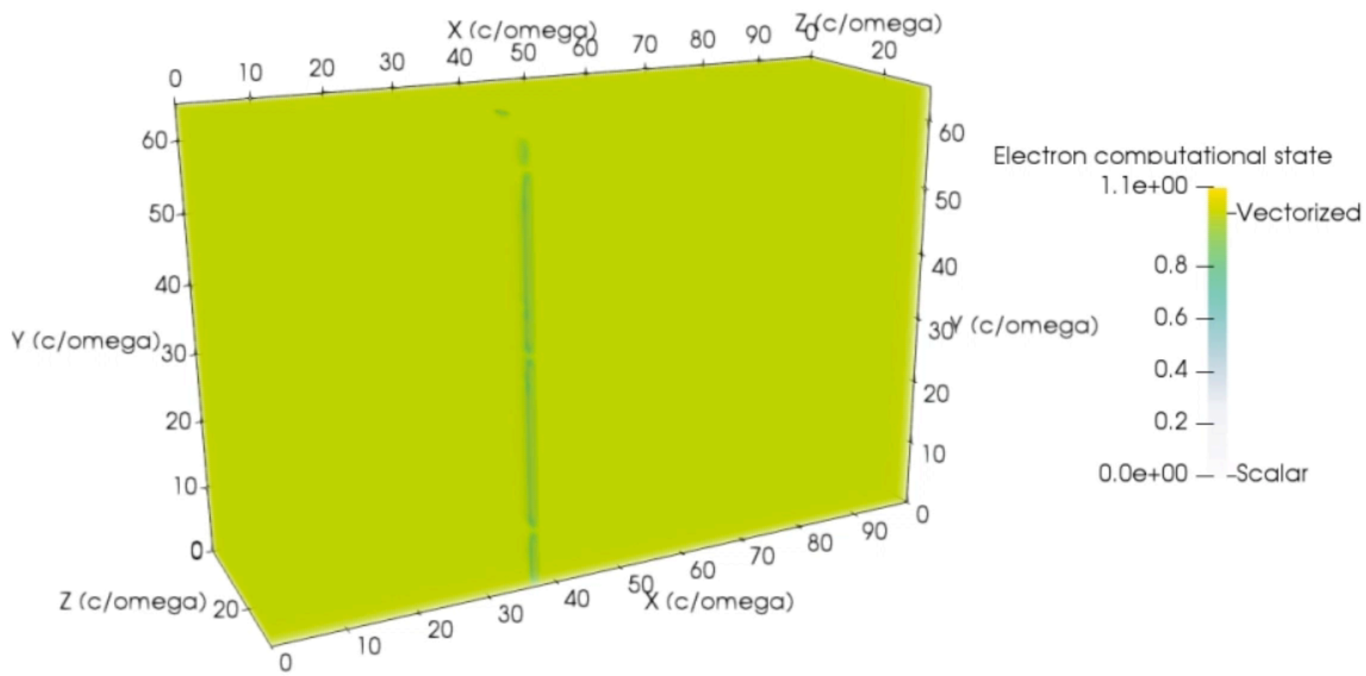


Step 2: Vectorization

Laser-driven hole-boring

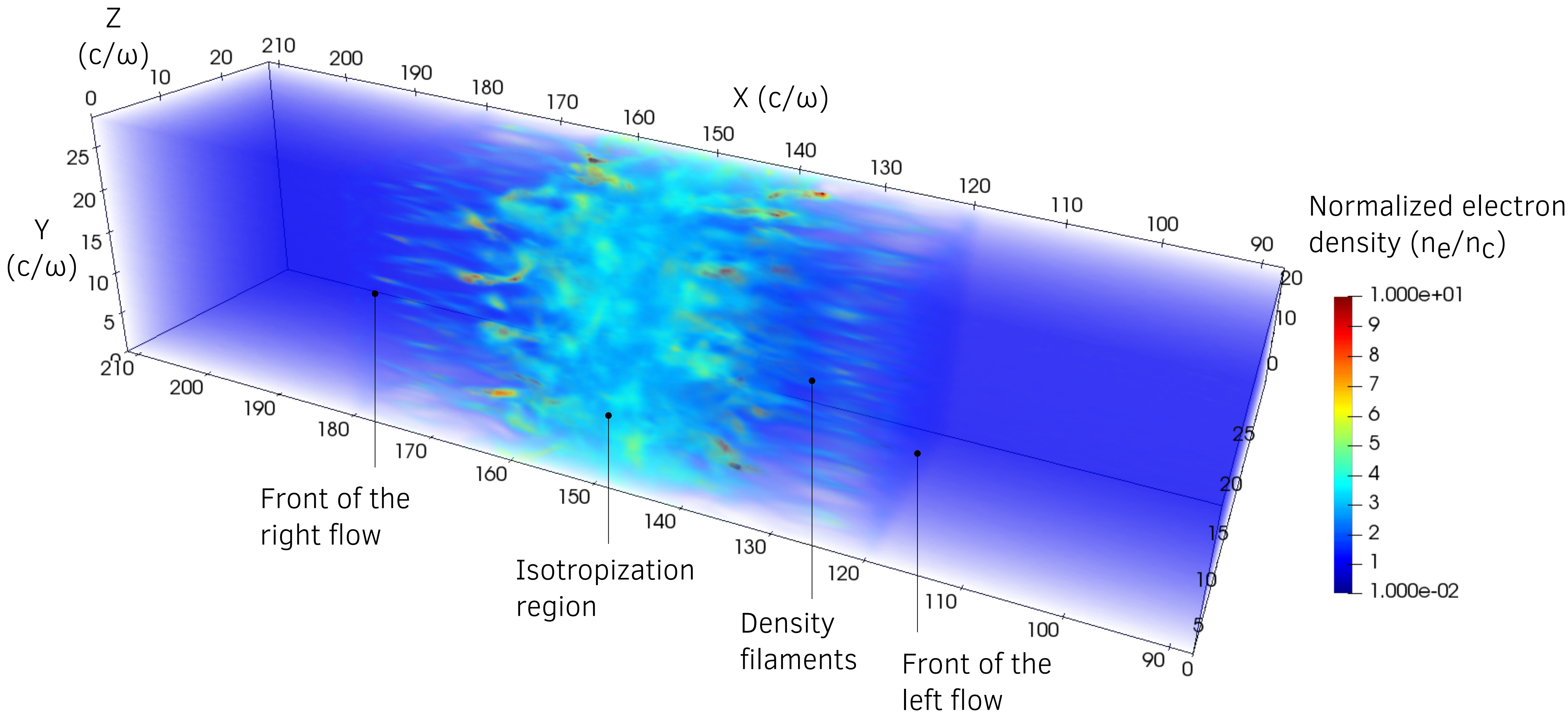






Step 2: Vectorization

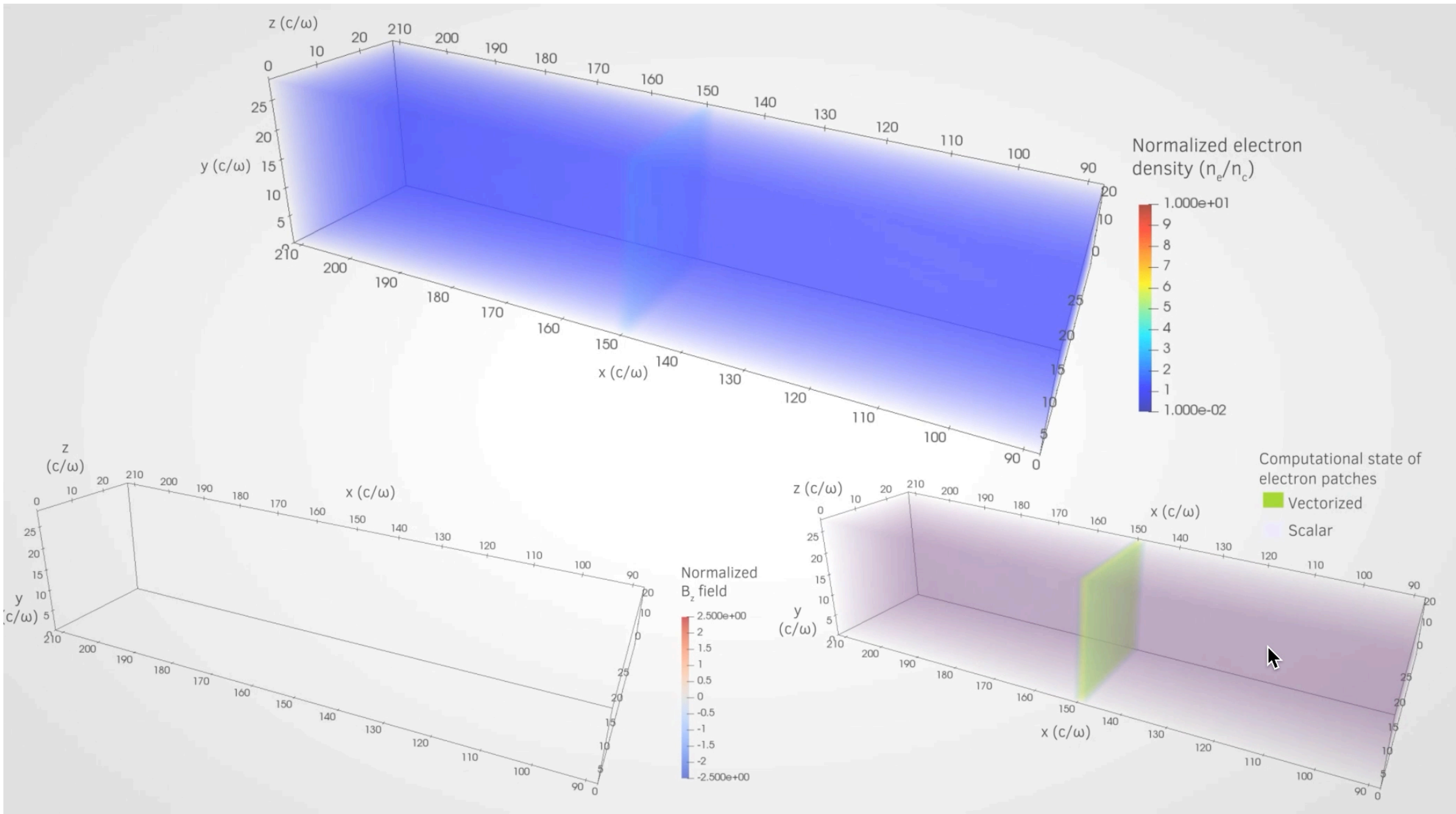
Weibel-mediated collisionless shocks



@ 32 PPC : speed-up x 1.5

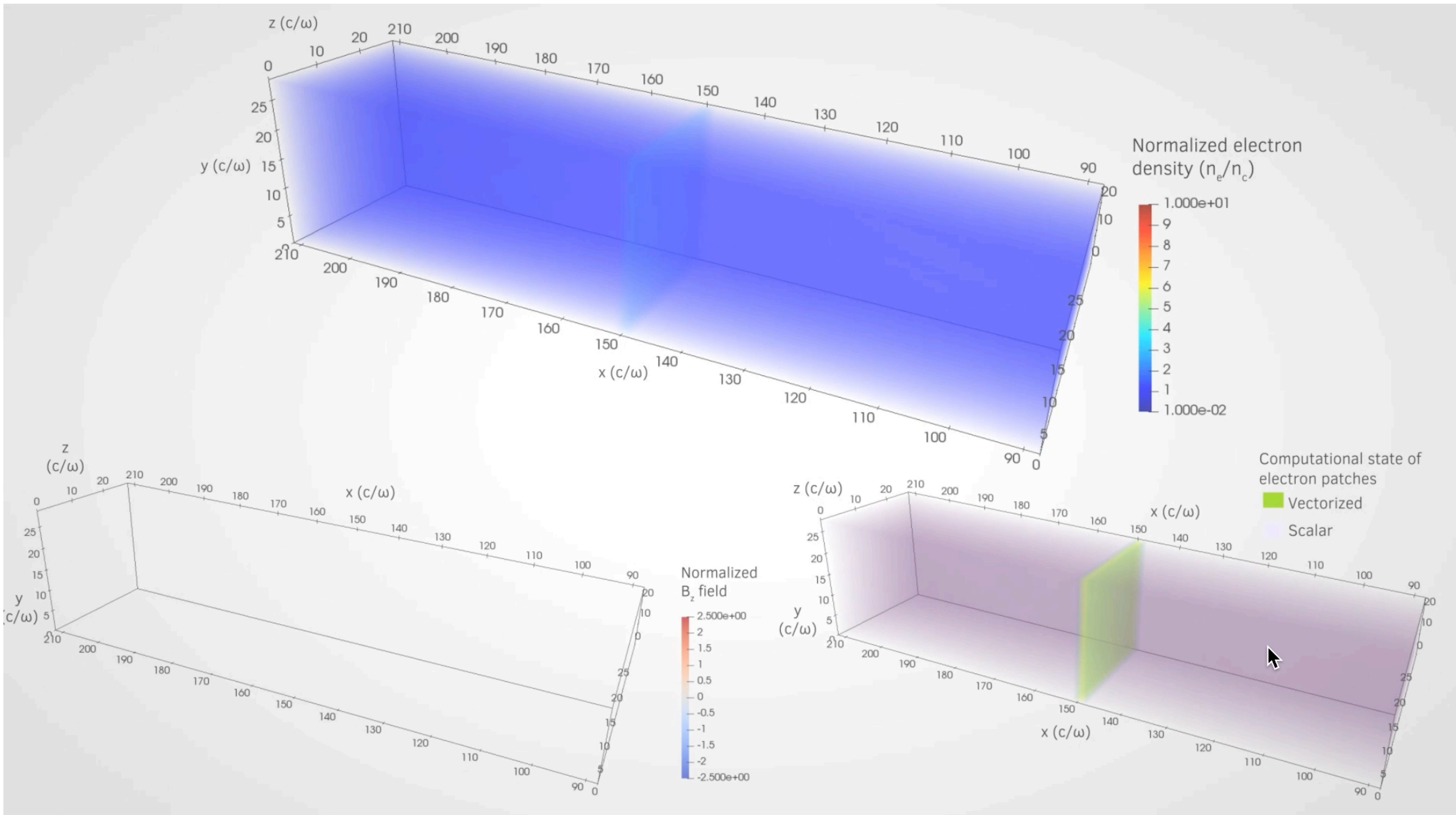
Step 2: Vectorization

Weibel-mediated collisionless shocks



Step 2: Vectorization

Weibel-mediated collisionless shocks



Outlines

- Numerical approach: how to build your PIC code
- Parallelization: getting ready for the super-computers
- Additional modules: beyond the *collisionless* plasma
- Some physics highlights: what you can do with a PIC code

Collisions can be introduced using an *ad-hoc* Monte-Carlo module

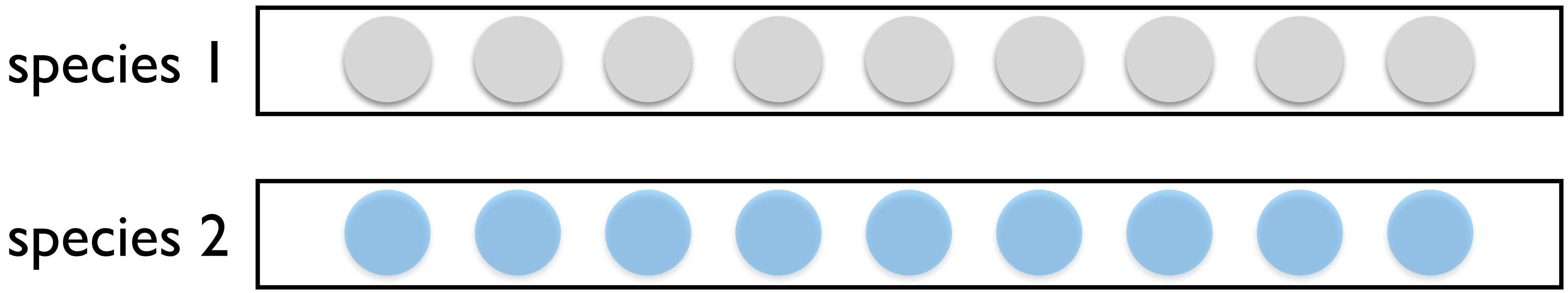
Collisions can be introduced using an *ad-hoc* Monte-Carlo module

Collisions are computed inside the cell

Collisions can be introduced using an *ad-hoc* Monte-Carlo module

Collisions are computed inside the cell

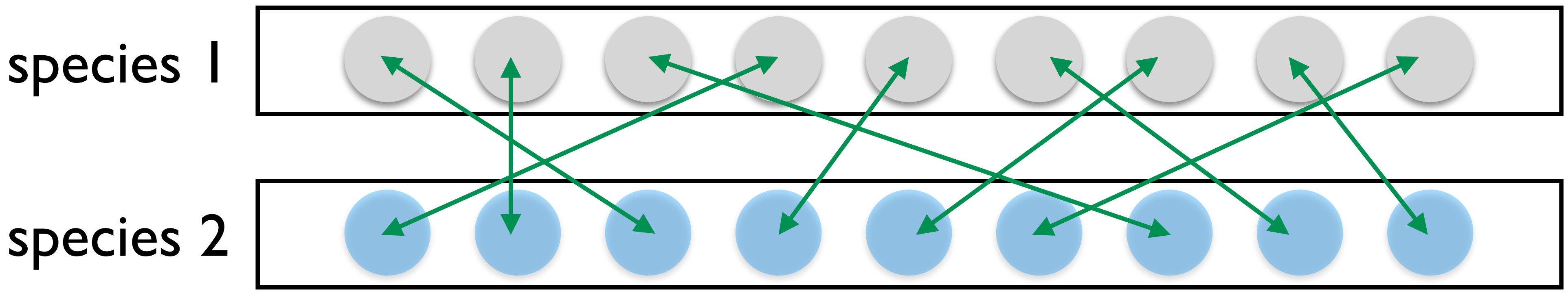
To avoid the N-body problem, quasi-particles in the cell are randomly "paired"



Collisions can be introduced using an *ad-hoc* Monte-Carlo module

Collisions are computed inside the cell

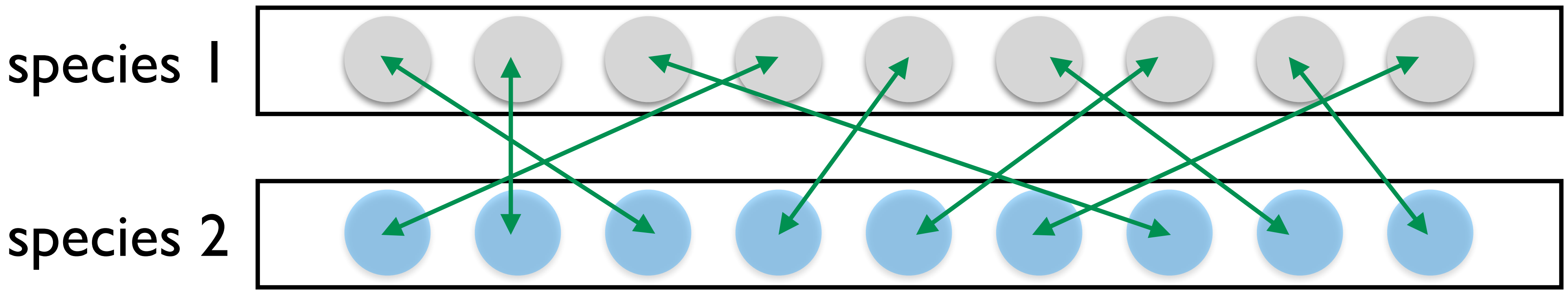
To avoid the N-body problem, quasi-particles in the cell are randomly "paired"



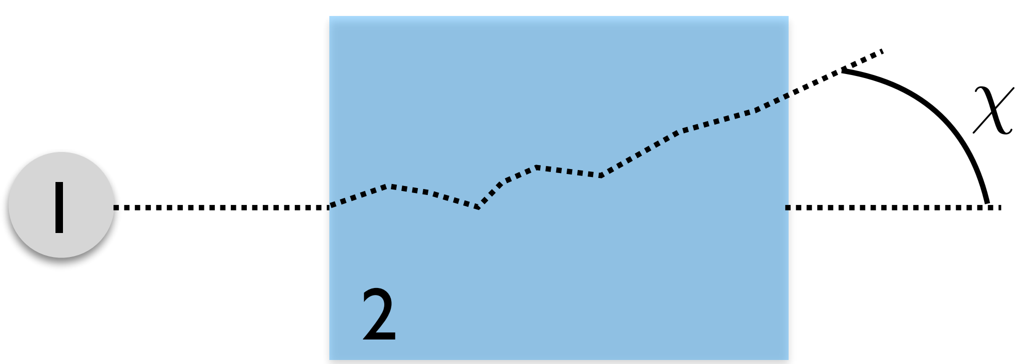
Collisions can be introduced using an *ad-hoc* Monte-Carlo module

Collisions are computed inside the cell

To avoid the N-body problem, quasi-particles in the cell are randomly "paired"



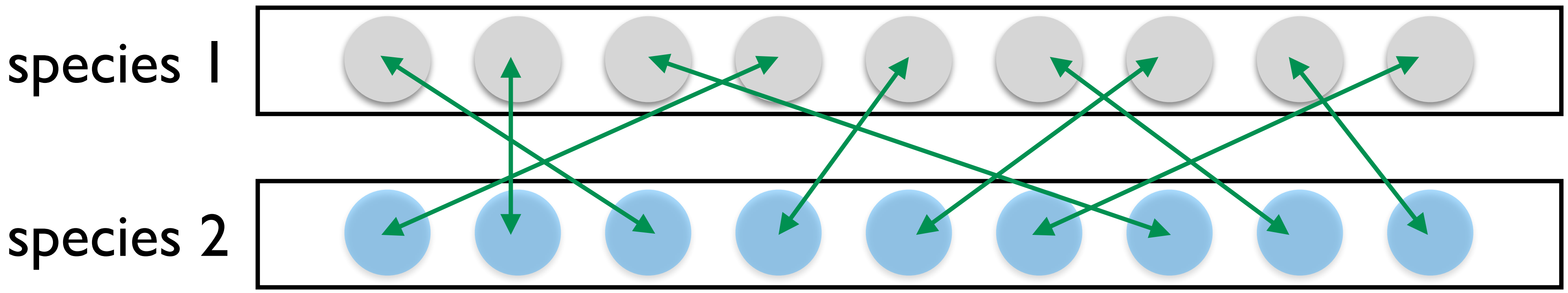
A single particle goes through many $(N \gg 1)$ collisions at small angle θ which translates in a total deflection angle χ (not necessarily small)



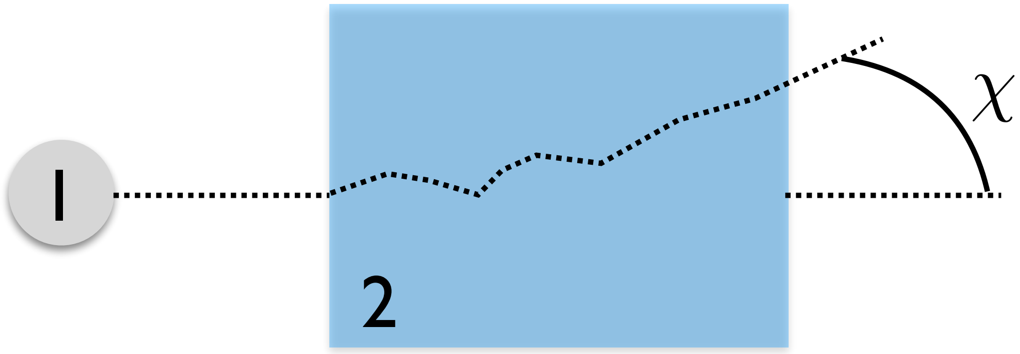
Collisions can be introduced using an *ad-hoc* Monte-Carlo module

Collisions are computed inside the cell

To avoid the N-body problem, quasi-particles in the cell are randomly "paired"



A single particle goes through many ($N \gg 1$) collisions at small angle θ which translates in a total deflection angle χ (not necessarily small)



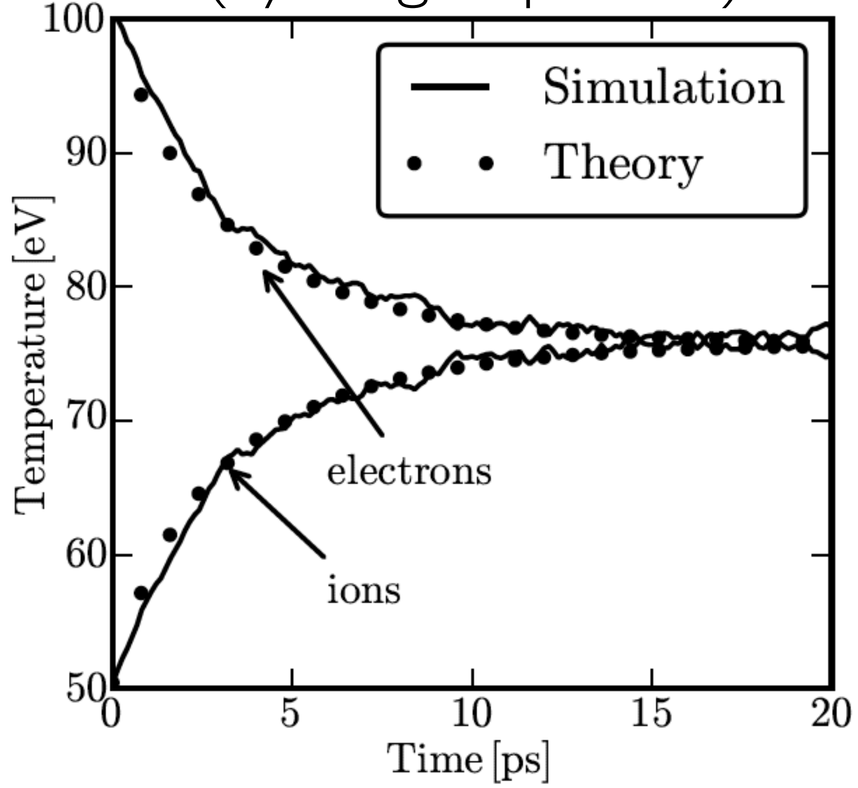
for each pair (Monte-Carlo)

- compute the collision rate
- compute the deflection angle
- deflect one or both particles

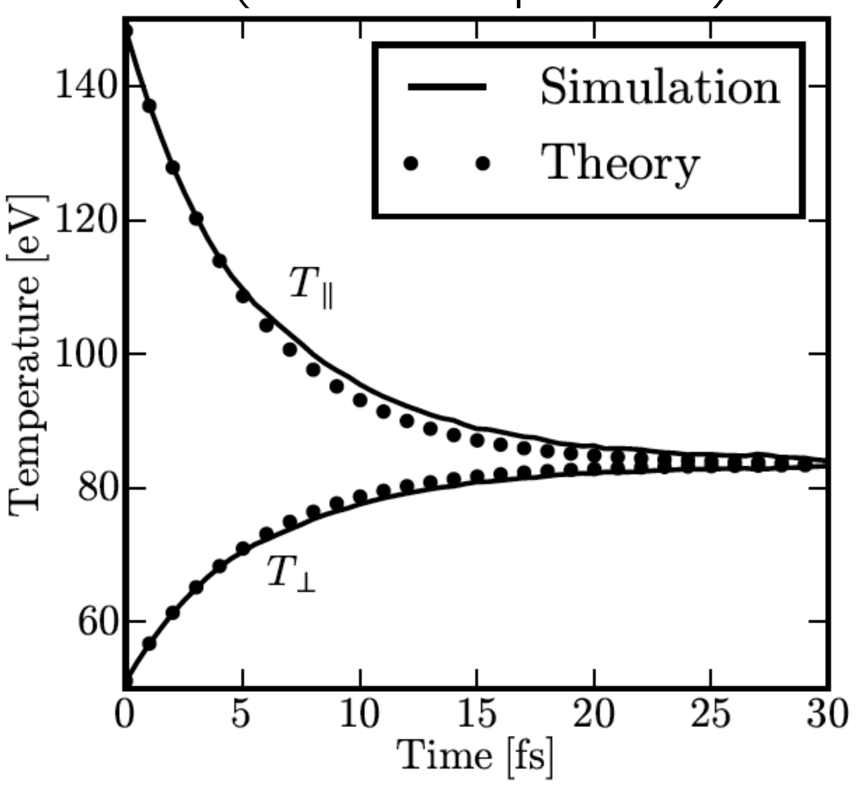
Nanbu, Phys. Rev. E **55**, 4642 (1997); J. Comp. Phys. **145**, 639 (1998)
F. Pérez et al., Phys. Plasmas **19**, 083104 (2012)

PIC codes are then able to treat purely collisional processes

Thermalization
(hydrogen plasma)



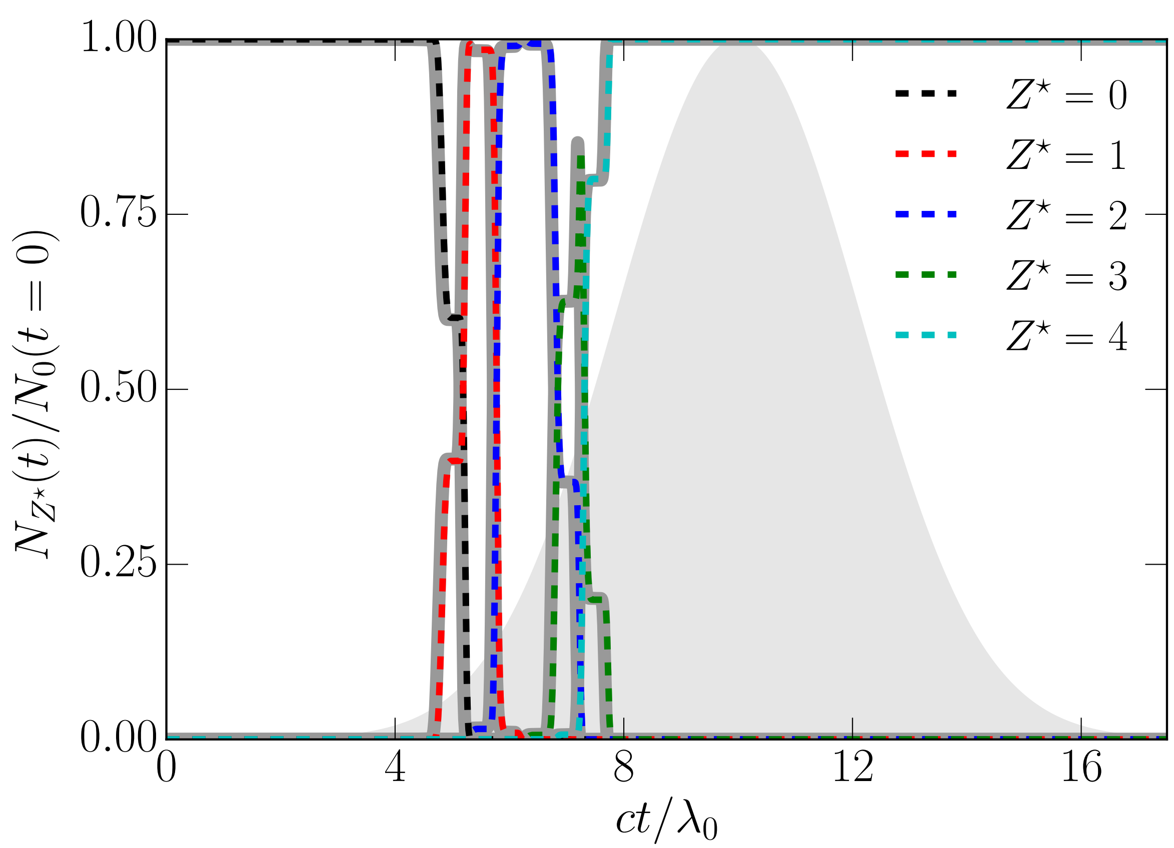
Isotropization
(electron plasma)



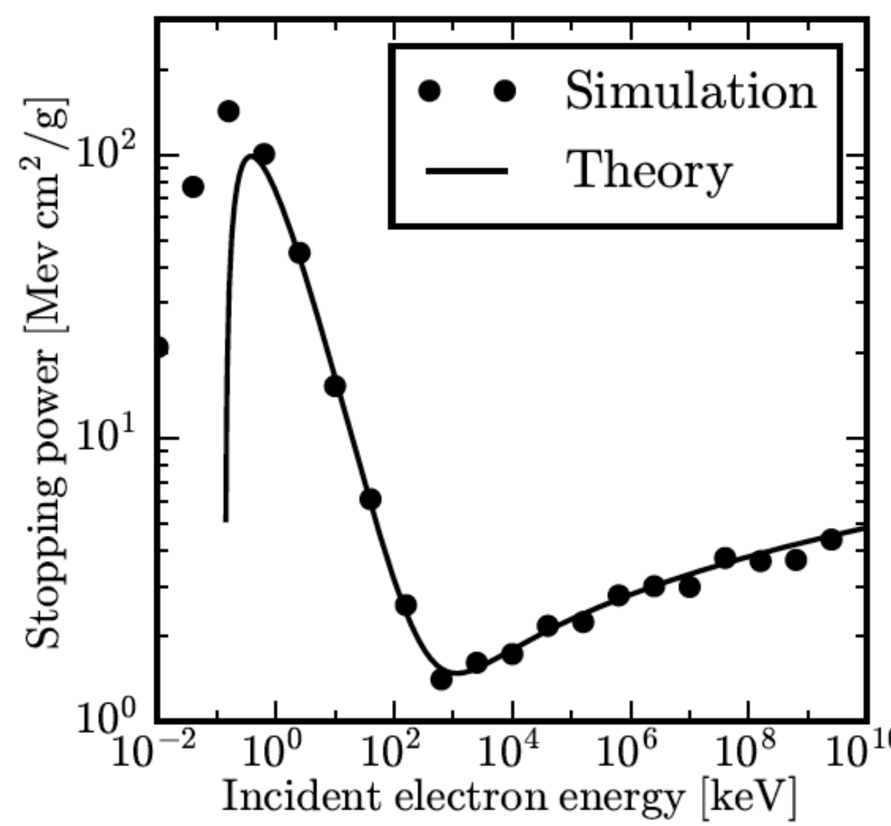
J. Derouillat et al., *SMILEI: a collaborative, open-source, multi-purpose PIC code for plasma simulation,* to be submitted (available upon request)

Similarly field and collisional ionization can be treated using a Monte-Carlo approach

Field ionization of Carbon by a 5x10^{16} W/cm^2 20 fs laser pulse



Stopping power of a cold aluminium plasma of density 10^{21} cm^{-3}



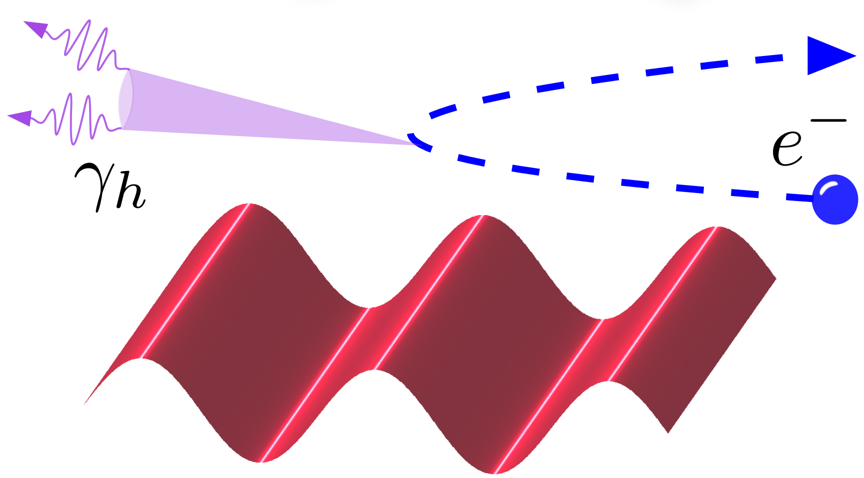
R. Nuter *et al.*, Phys. Plasmas **18**, 033107 (2011); F. Pérez et al., Phys. Plasmas **19**, 083104 (2012)
J. Derouillat et al., *SMILEI: a collaborative, open-source, multi-purpose PIC code for plasma simulation*, to be submitted (available upon request)

Adding Quantum Electrodynamics (QED) effect is also very interesting for forthcoming multi-petawatt facilities

Adding Quantum Electrodynamics (QED) effect is also very interesting for forthcoming multi-petawatt facilities

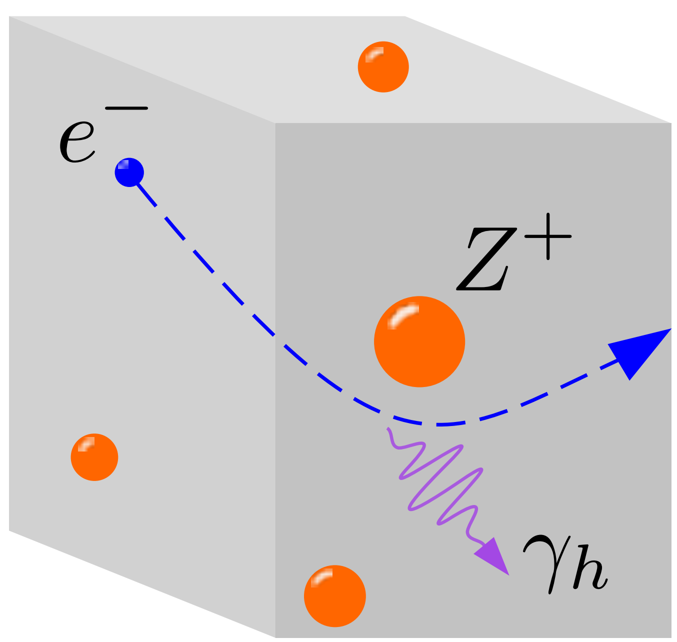
High-Energy Photon Production

Nonlinear Thomson and Compton scattering

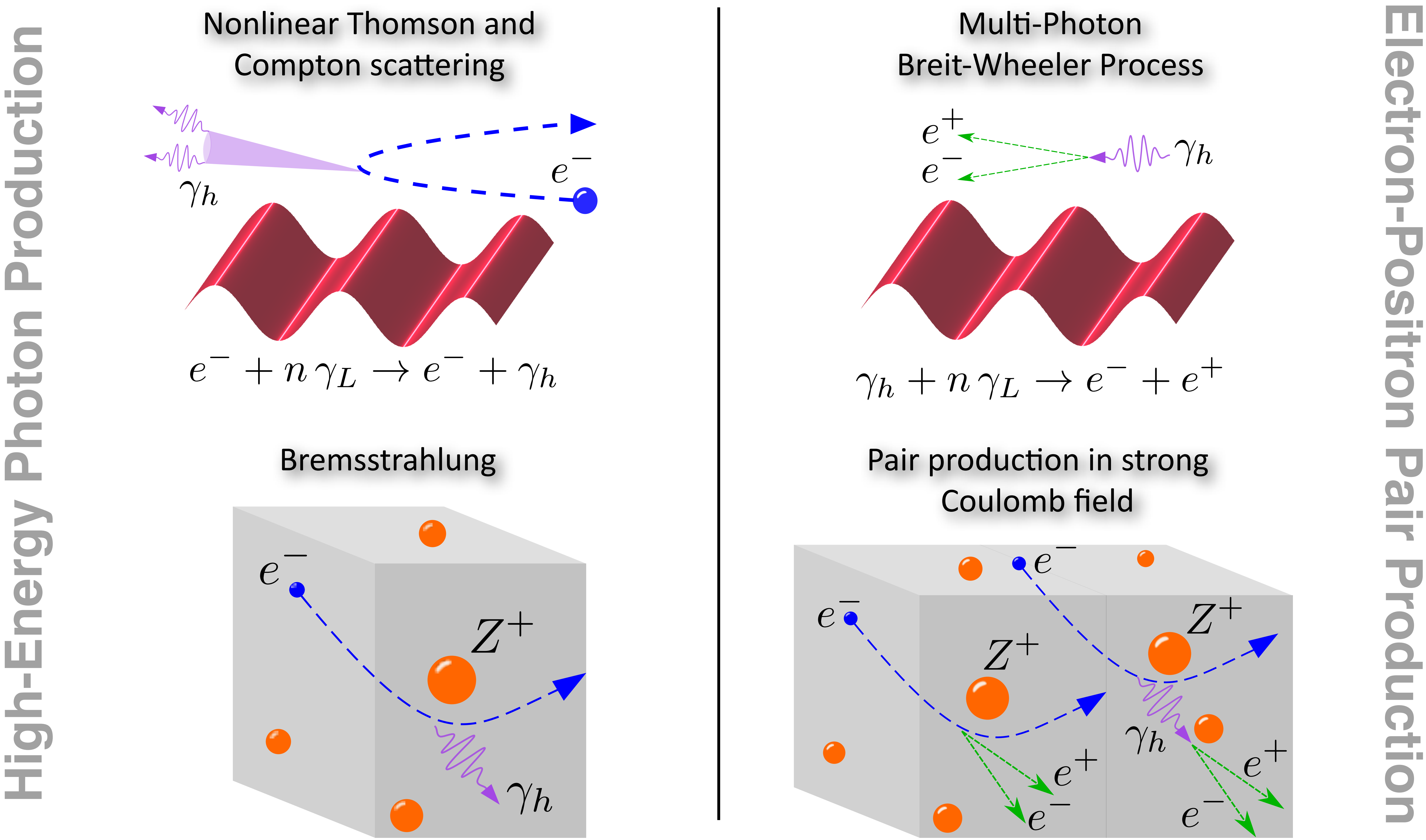


$$e^- + n\,\gamma_L \rightarrow e^- + \gamma_h$$

Bremsstrahlung



Adding Quantum Electrodynamics (QED) effect is also very interesting for forthcoming multi-petawatt facilities



Outlines

- Numerical approach: how to build your PIC code
- Parallelization: getting ready for the super-computers
- Additional modules: beyond the *collisionless* plasma
- Some physics highlights: what you can do with a PIC code

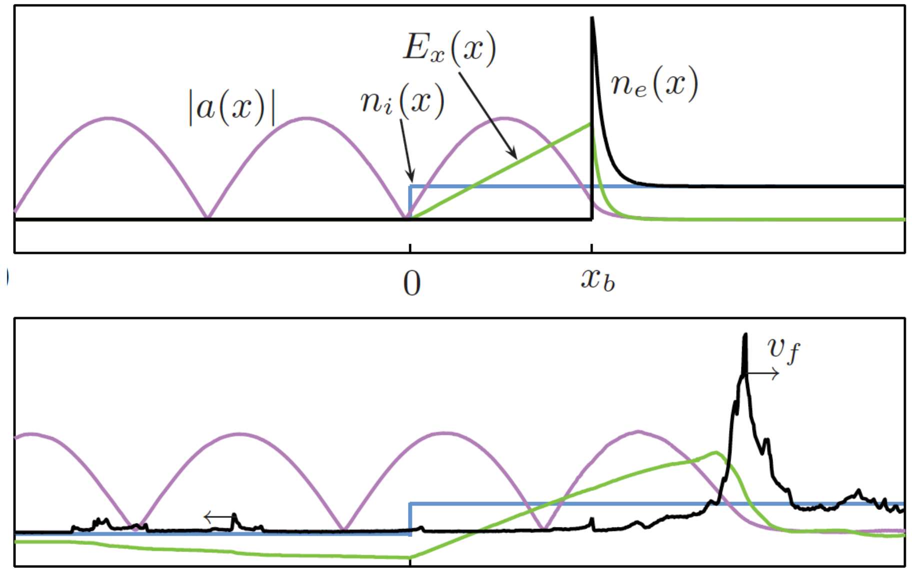
PIC codes are an excellent tool to support theoretical modelling

Even 1D simulation can bring a deep insight into the physics at play

PIC codes are an excellent tool to support theoretical modelling

Even 1D simulation can bring a deep insight into the physics at play

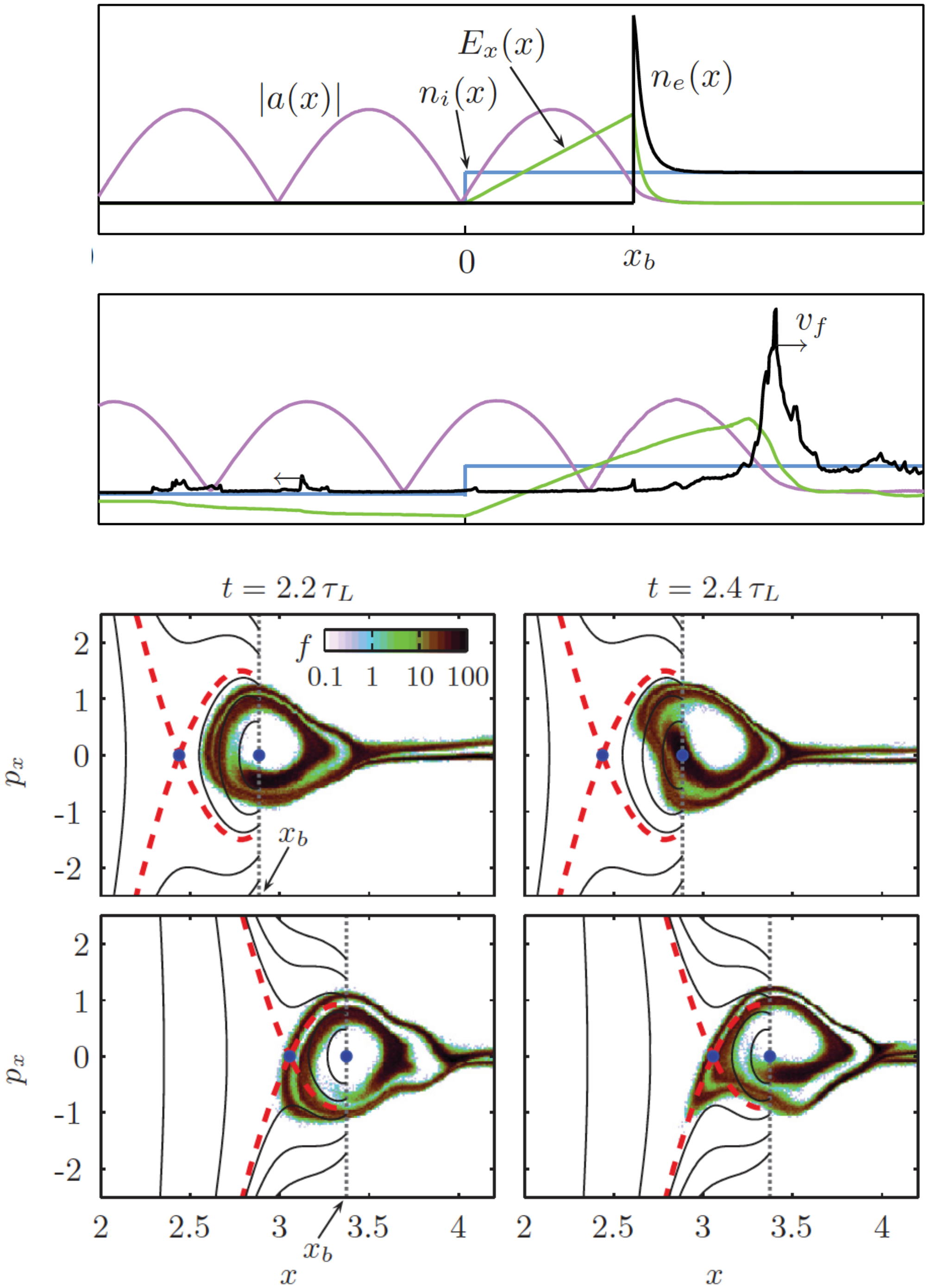
Relativistically-Induced Transparency



PIC codes are an excellent tool to support theoretical modelling

Even 1D simulation can bring a deep insight into the physics at play

Relativistically-Induced Transparency

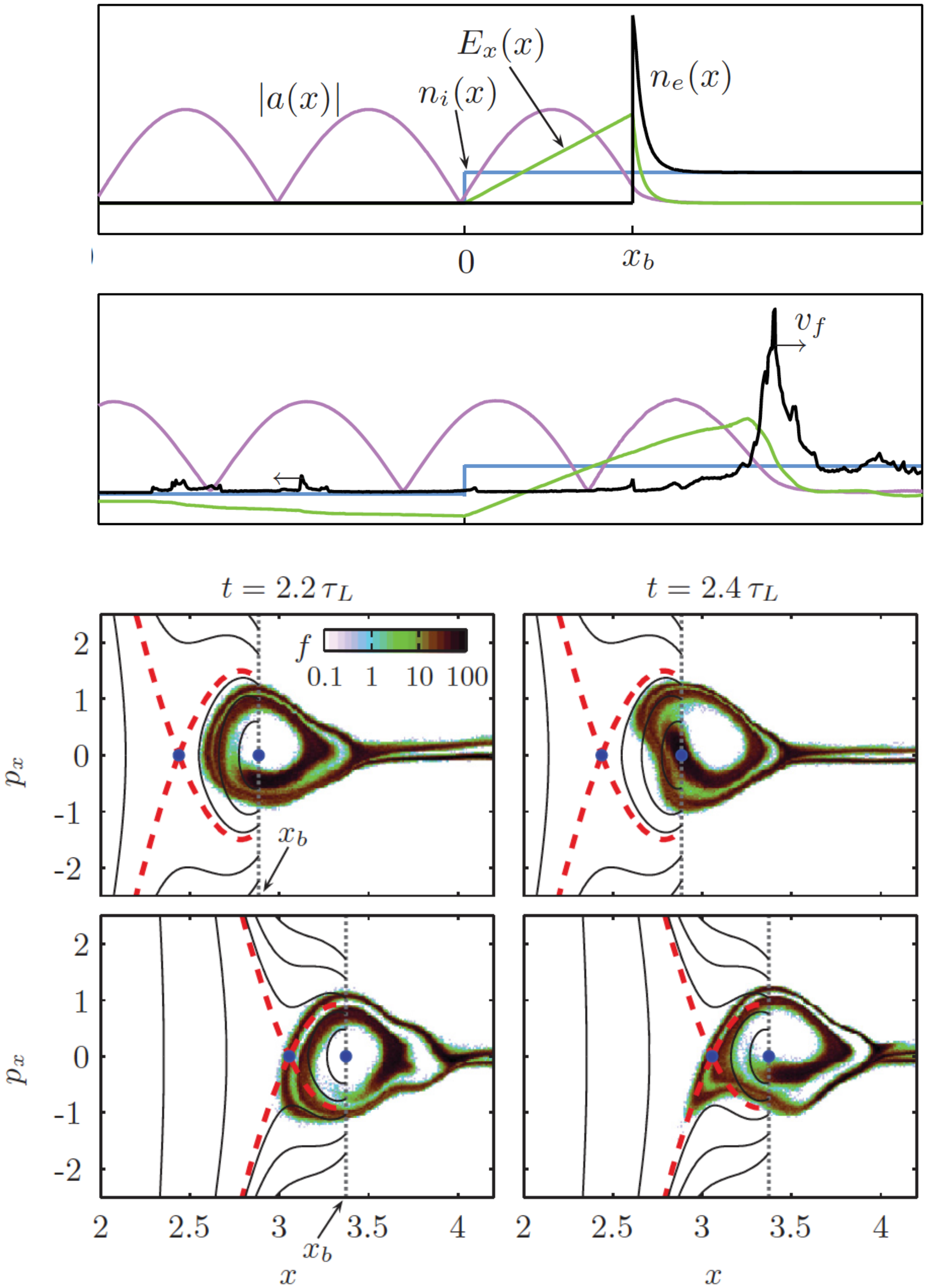


E. Siminos *et al.*, Phys. Rev. E **86**, 056404 (2012)

PIC codes are an excellent tool to support theoretical modelling

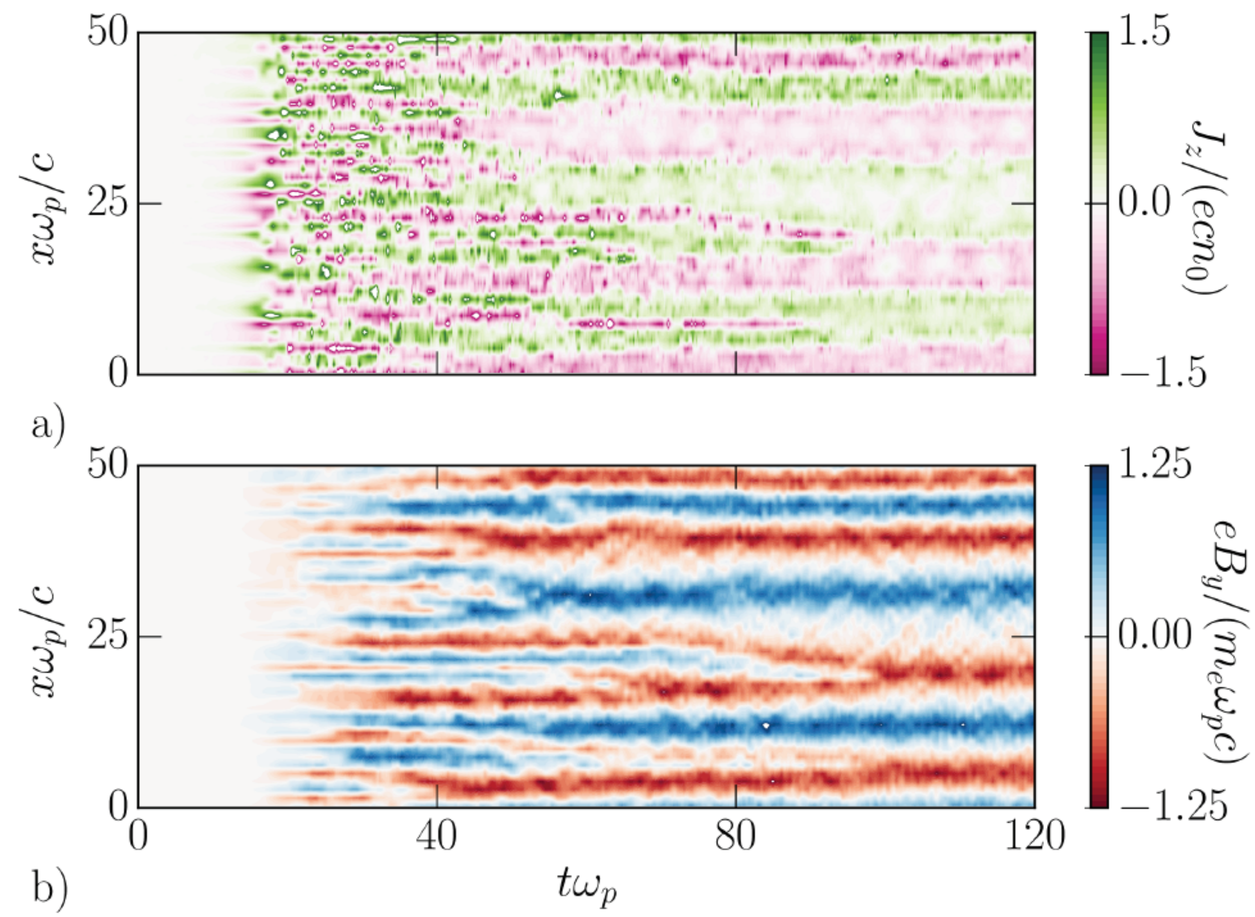
Even 1D simulation can bring a deep insight into the physics at play

Relativistically-Induced Transparency



E. Siminos *et al.*, Phys. Rev. E **86**, 056404 (2012)

Weibel instability in the presence of an external magnetic field



A. Grassi *et al.*, Phys. Rev. E (in press)

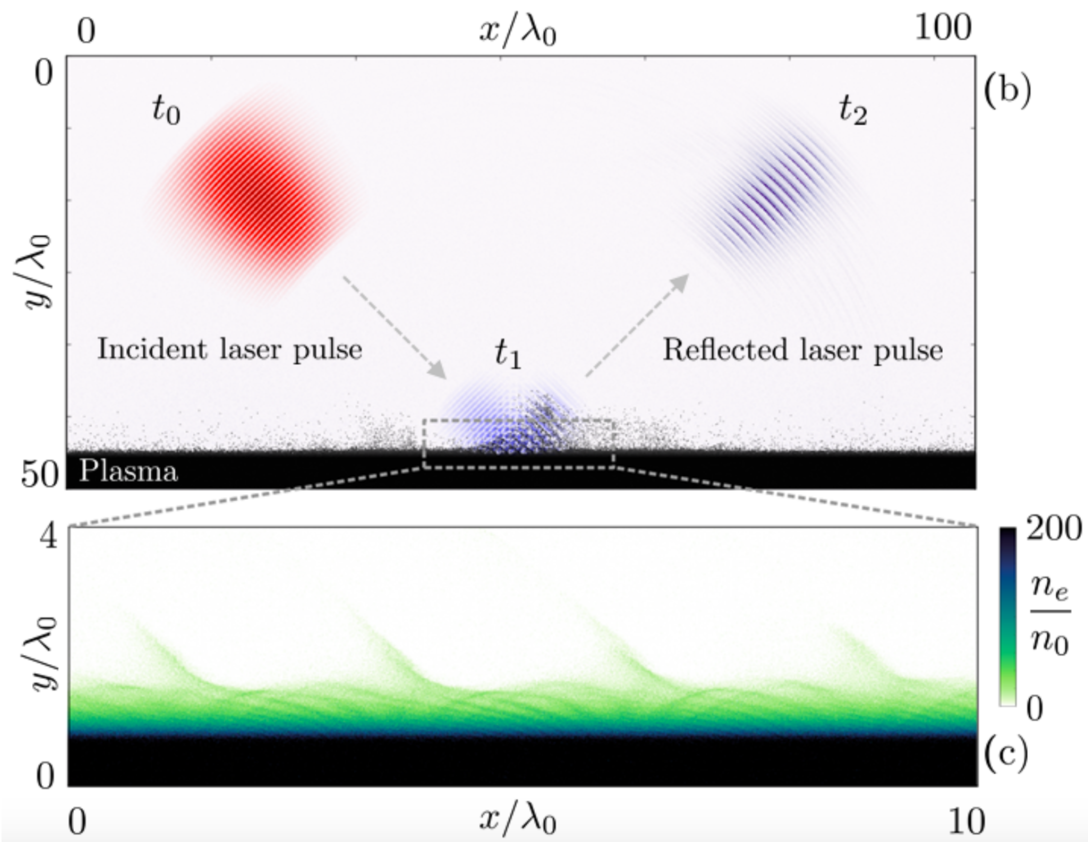
PIC codes can help design & interpret experimental campaigns

2D and 3D simulations on super-computers will be necessary here

PIC codes can help design & interpret experimental campaigns

2D and 3D simulations on super-computers will be necessary here

High-harmonic generation & electron acceleration from laser-solid interaction

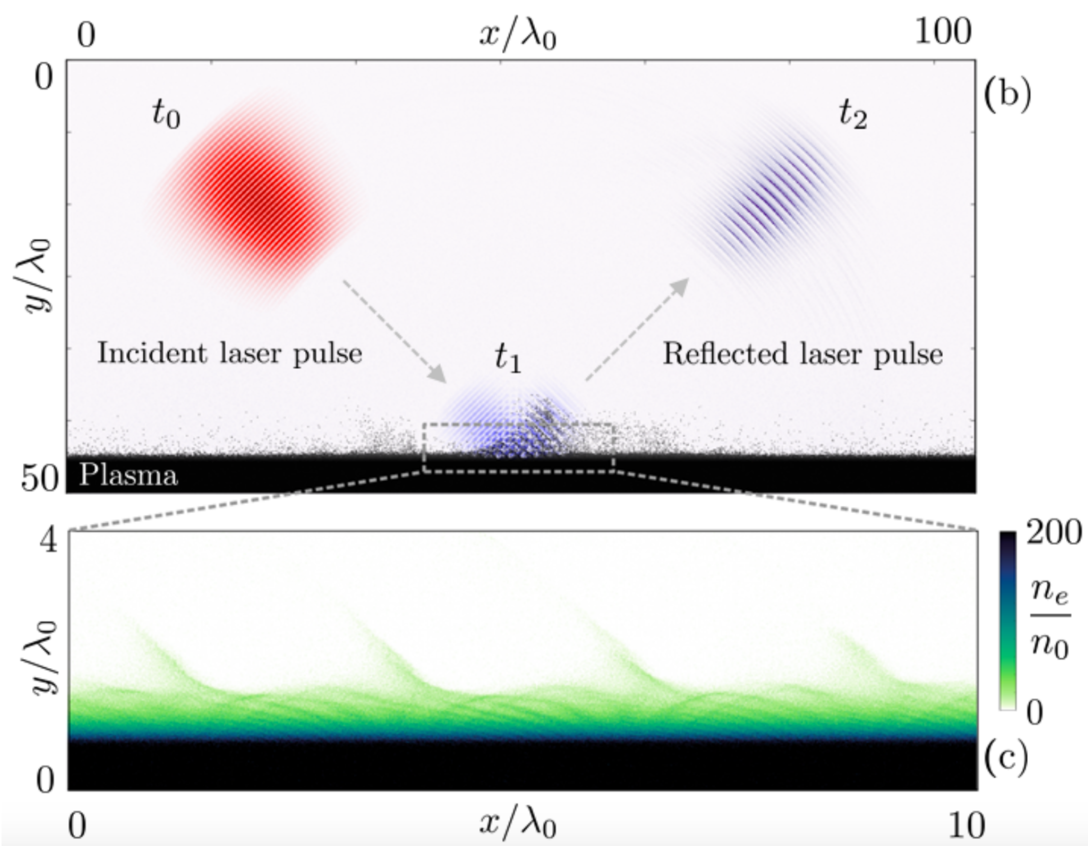


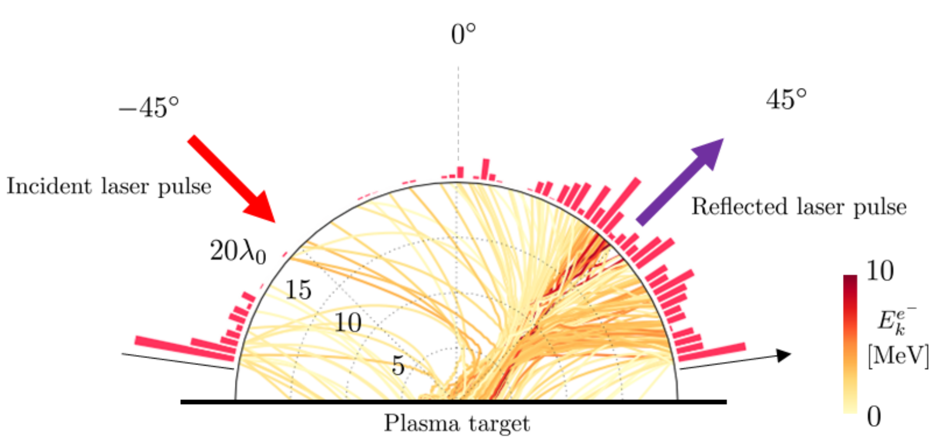
G. Bouchard, F. Quéré, CEA/IRAMIS

PIC codes can help design & interpret experimental campaigns

2D and 3D simulations on super-computers will be necessary here

High-harmonic generation & electron acceleration from laser-solid interaction



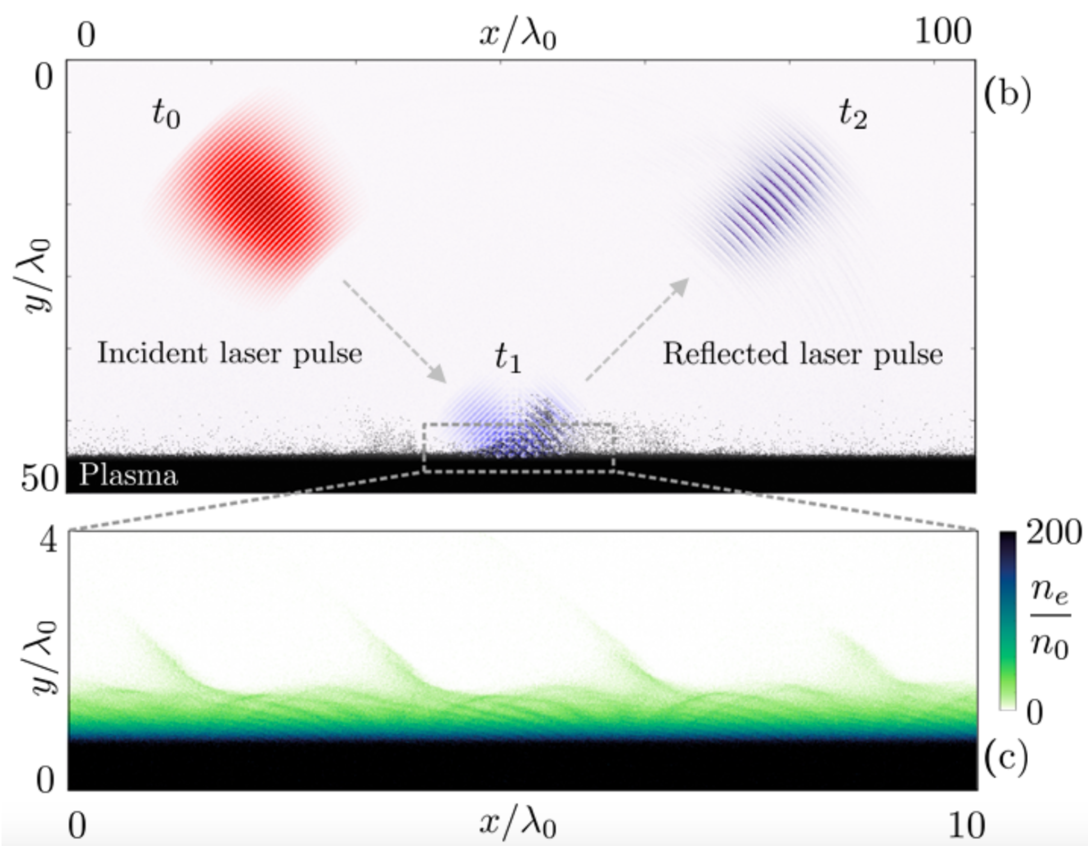


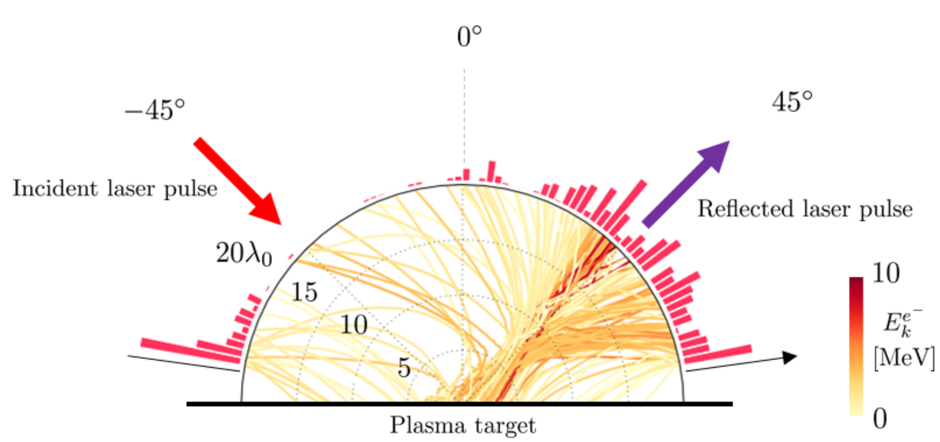
G. Bouchard, F. Quéré, CEA/IRAMIS

PIC codes can help design & interpret experimental campaigns

2D and 3D simulations on super-computers will be necessary here

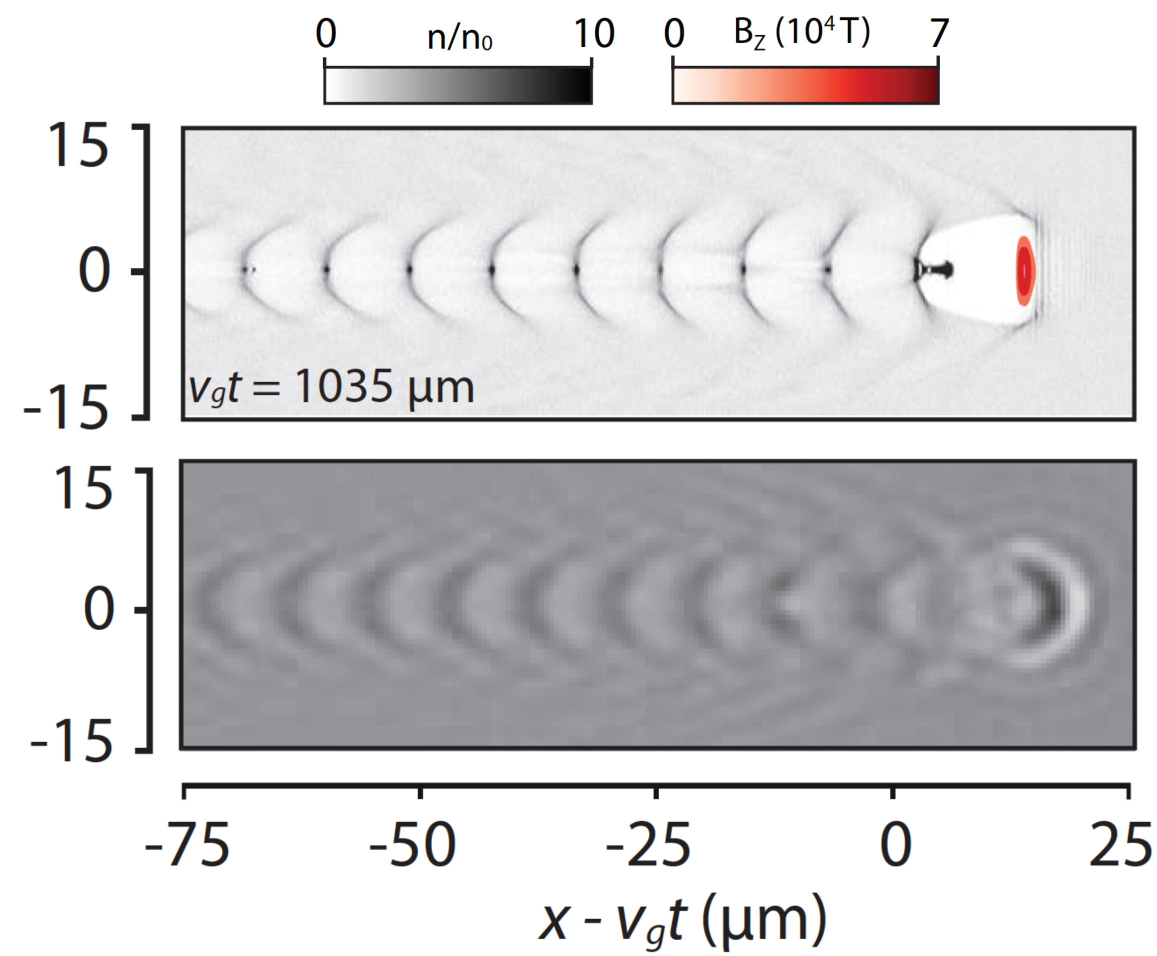
High-harmonic generation & electron acceleration from laser-solid interaction





G. Bouchard, F. Quéré, CEA/IRAMIS

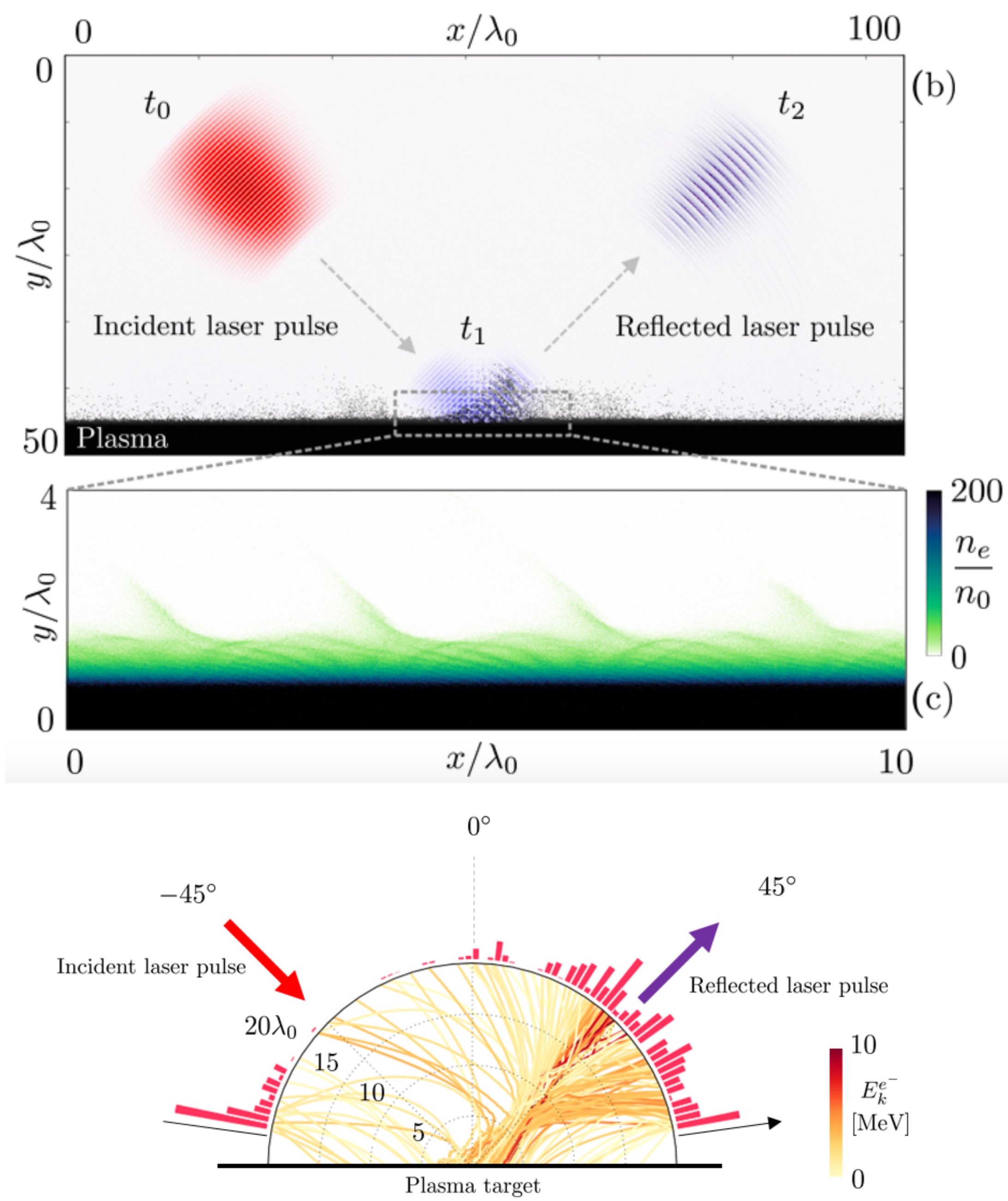
Laser wakefield acceleration of electrons



PIC codes can help design & interpret experimental campaigns

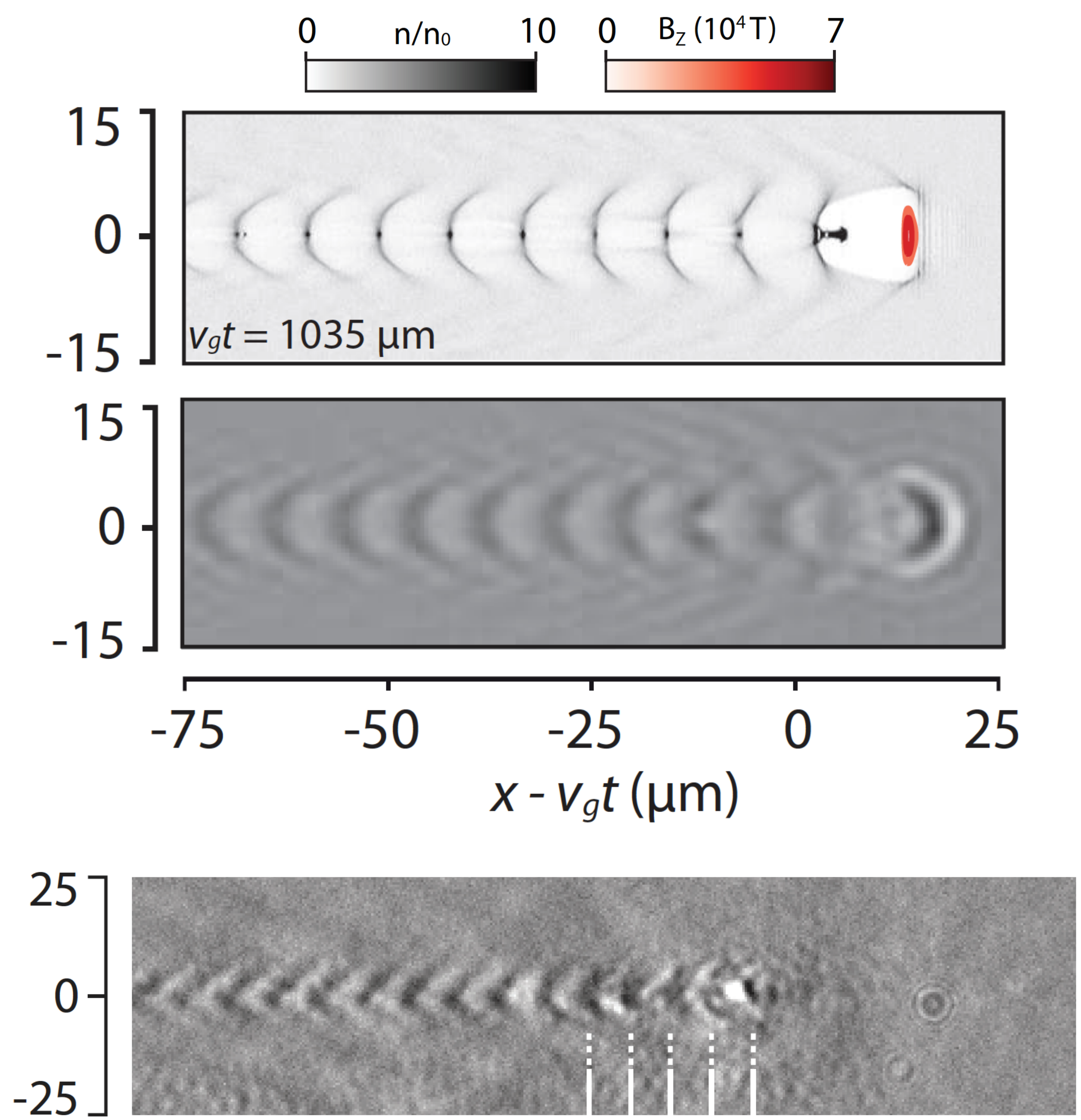
2D and 3D simulations on super-computers will be necessary here

High-harmonic generation & electron acceleration from laser-solid interaction



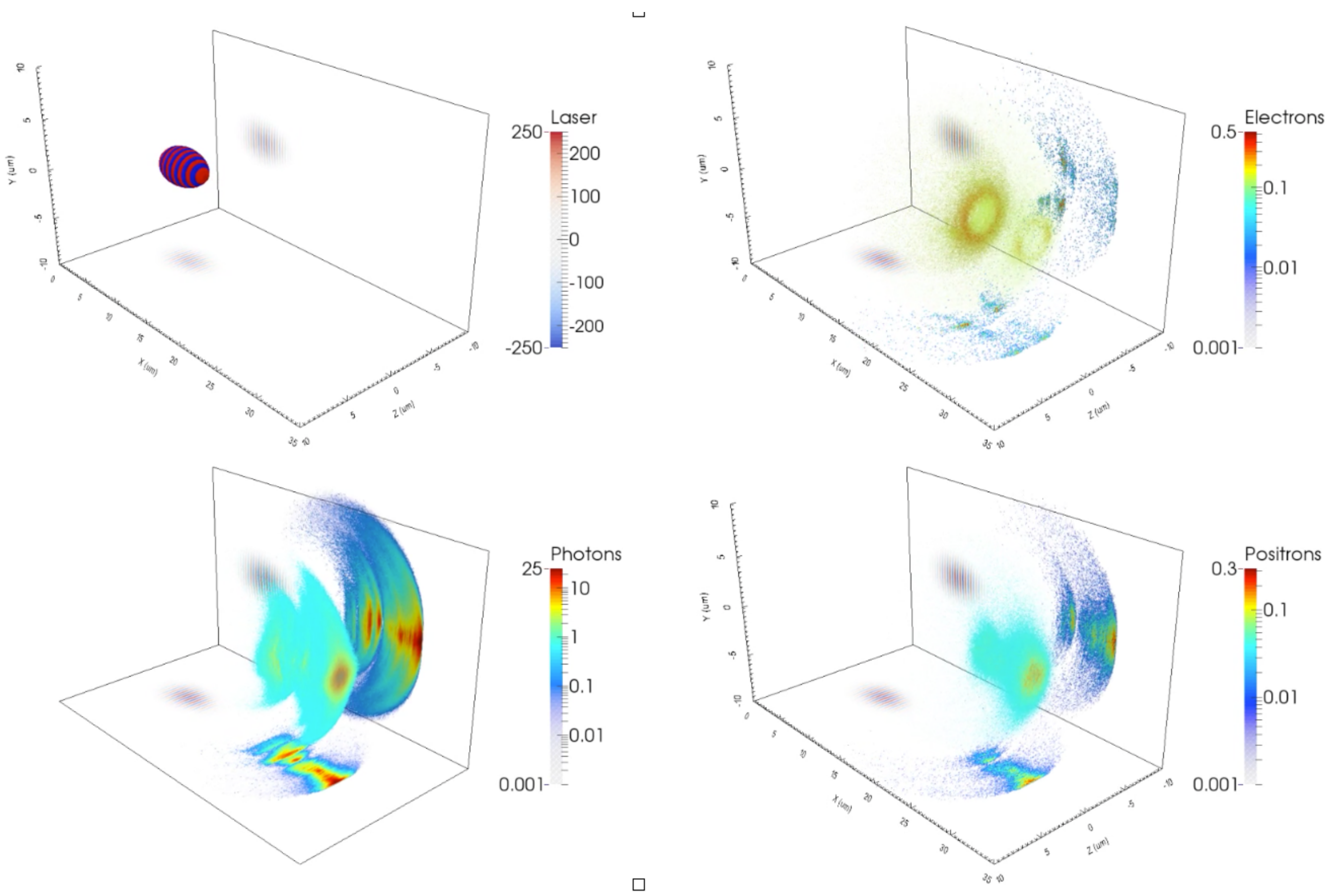
G. Bouchard, F. Quéré, CEA/IRAMIS

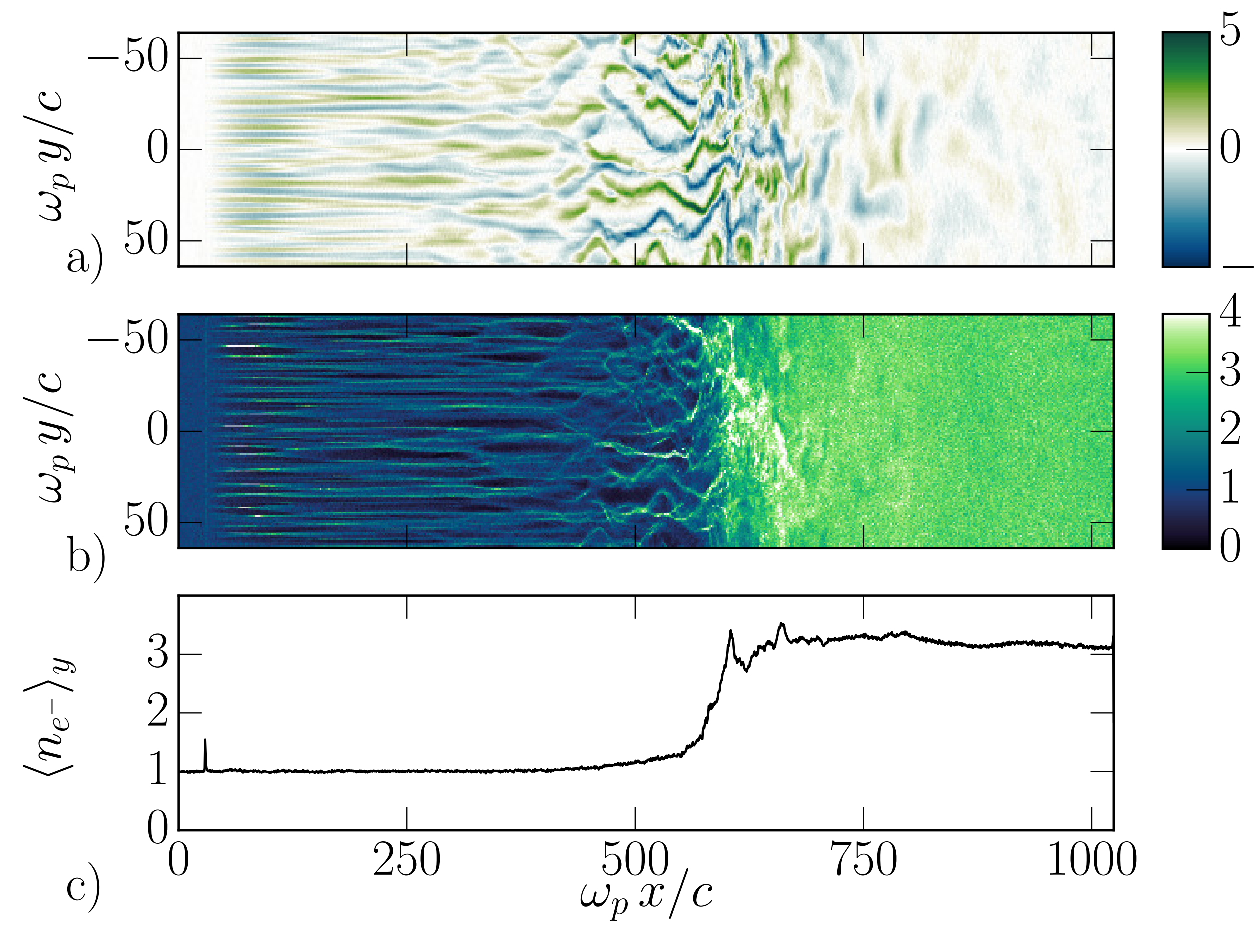
Laser wakefield acceleration of electrons



A. Sävert *et al.*, Phys. Rev. Lett. **115**, 055002 (2015)

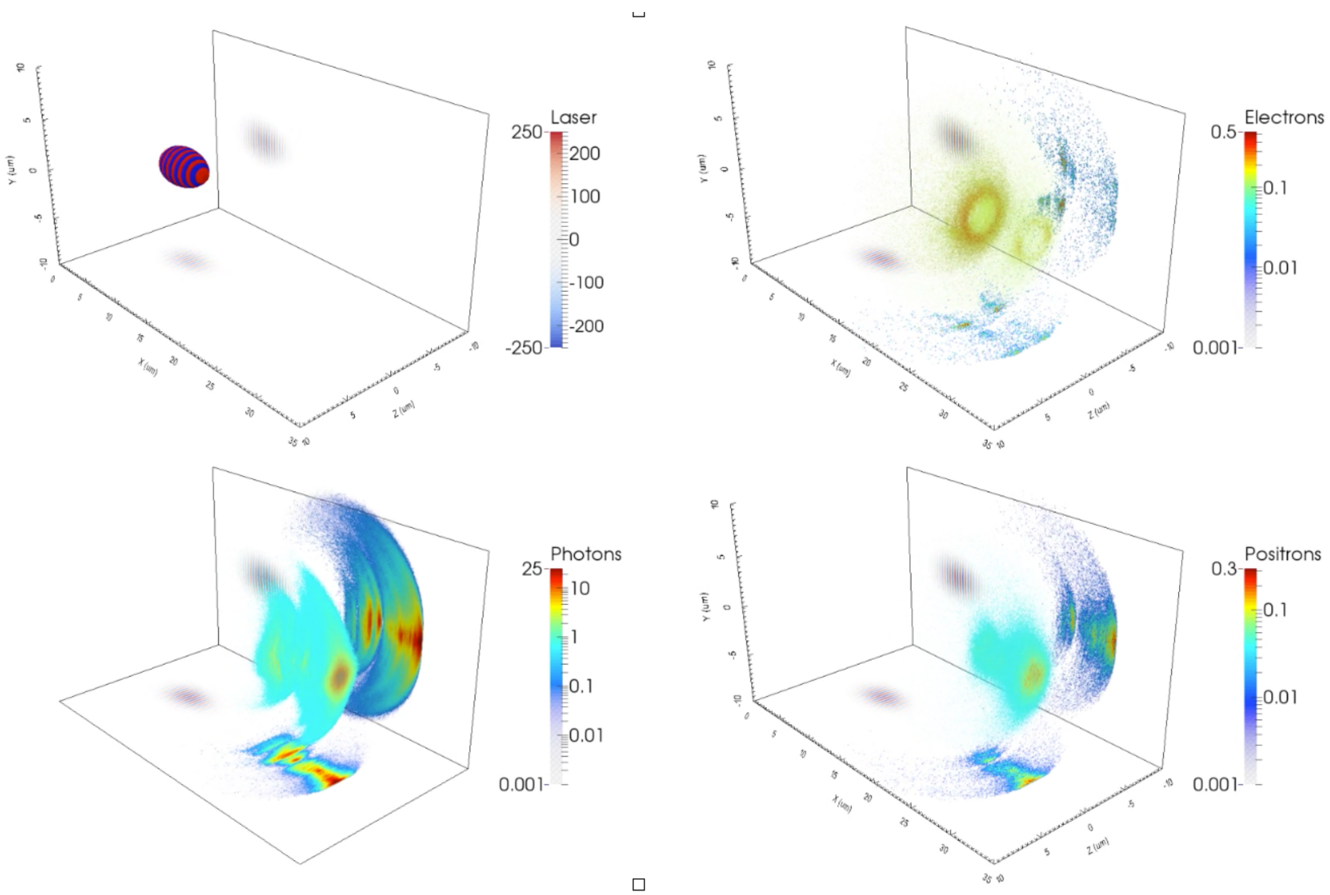
PIC codes are very versatile: they can be applied to a wide range of physical scenarii, from laser-plasma interaction to astrophysics





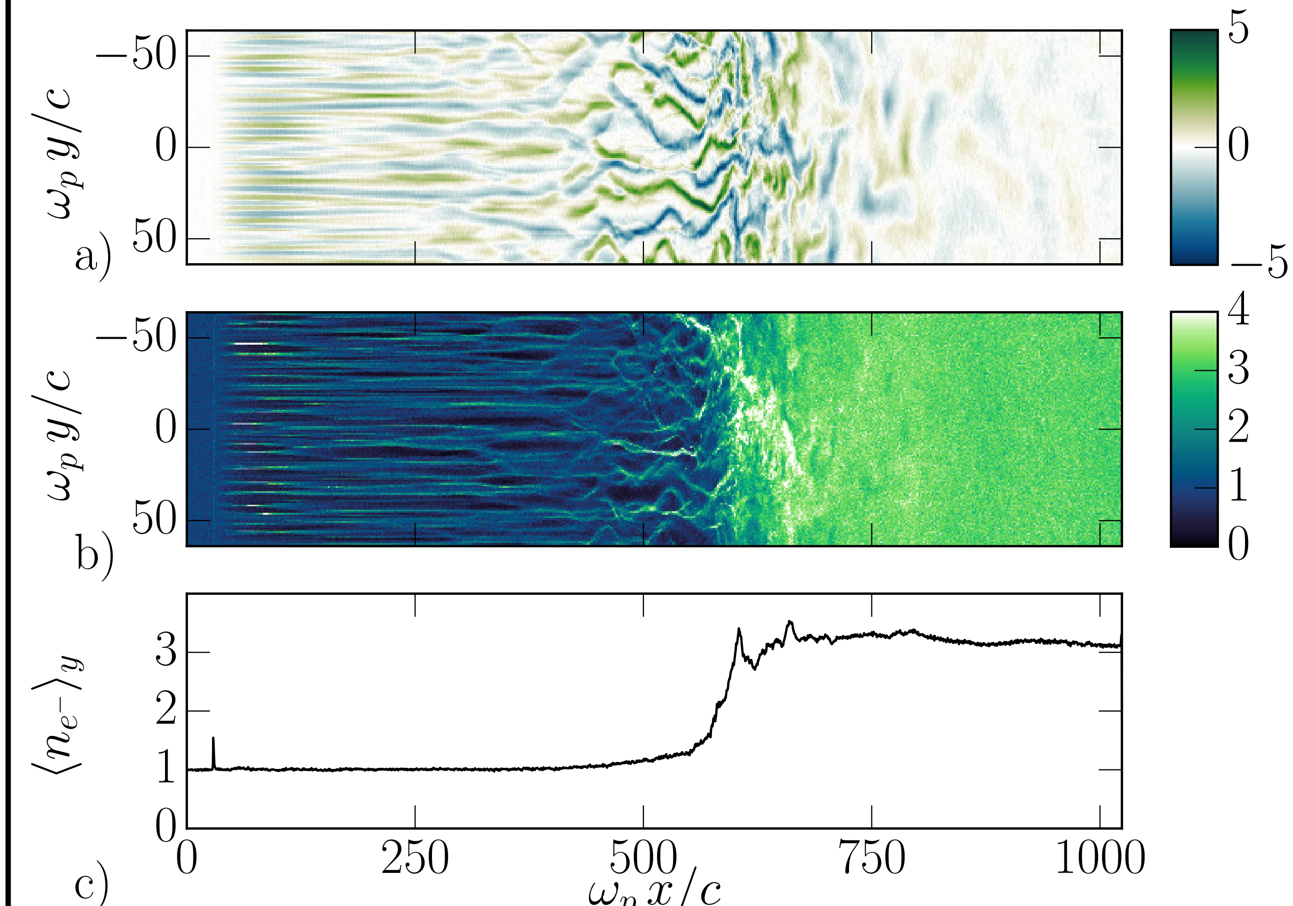
PIC codes are very versatile: they can be applied to a wide range of physical scenarii, from laser-plasma interaction to astrophysics

Pair production on multi-petawatt laser facilities



M. Lobet *et al.*, arXiv:1510.02301v2 (2015)

Relativistic shocks in electron-positron plasmas



Plotnikov, Grassi & Grech, MNRAS

Conclusions

Conclusions

- PIC codes are very popular, versatile & efficient tools for plasma simulation

Conclusions

- PIC codes are very popular, versatile & efficient tools for plasma simulation
- A large variety of physical problems can be addressed using PIC codes

Conclusions

- PIC codes are very popular, versatile & efficient tools for plasma simulation
- A large variety of physical problems can be addressed using PIC codes
- Additional physics modules can be implemented in PIC codes, but the physics cannot be scaled anymore (ω_r needs to be defined!)

Conclusions

- PIC codes are very popular, versatile & efficient tools for plasma simulation
- A large variety of physical problems can be addressed using PIC codes
- Additional physics modules can be implemented in PIC codes, but the physics cannot be scaled anymore (ω_r needs to be defined!)
- The PIC method is conceptually simple & can be efficiently implemented in a (massively) parallel framework

Conclusions

- PIC codes are very popular, versatile & efficient tools for plasma simulation
- A large variety of physical problems can be addressed using PIC codes
- Additional physics modules can be implemented in PIC codes, but the physics cannot be scaled anymore (ω_r needs to be defined!)
- The PIC method is conceptually simple & can be efficiently implemented in a (massively) parallel framework
- Implementation on new & future architectures requires a strong input (co-development) from HPC specialists

Checkout SMILEI !!!

Code, Diagnostics/visualization tools and tutorials
available online!

Smilei) | Overview | Understand | Use | More

Smilei)

Smilei is a Particle-In-Cell code for plasma simulation. Open-source, collaborative, user-friendly and designed for high performances on super-computers, it is applied to a wide range of physics studies: from relativistic laser-plasma interaction to astrophysics.

Download GitHub Partners Publications Tutorials

http://www.maisondelasimulation.fr/smilei

Checkout SMILEI !!!

Code, Diagnostics/visualization tools and tutorials
available online!

http://www.maisondelasimulation.fr/smilei