HiPACE++: presentation of the code

Maxence Thévenet – DESY

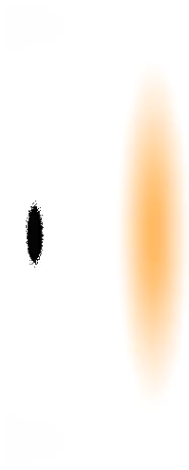
MPA – plasma acceleration

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

DESY.

Plasma acceleration is described by kinetic plasma dynamics

Plasma acceleration is described by kinetic plasma dynamics



Plasma acceleration is described by kinetic plasma dynamics

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

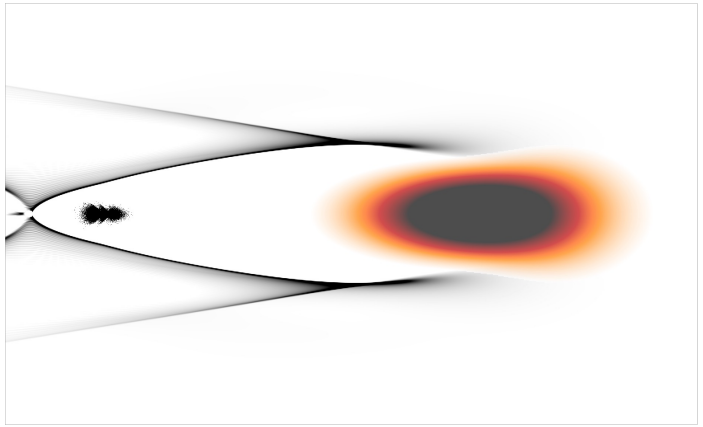
$$\frac{d\mathbf{x}}{dt} = \mathbf{v},$$
$$\frac{d(\gamma \mathbf{v})}{dt} = \frac{q}{m}(\mathbf{E} + \mathbf{v} \times \mathbf{B}),$$

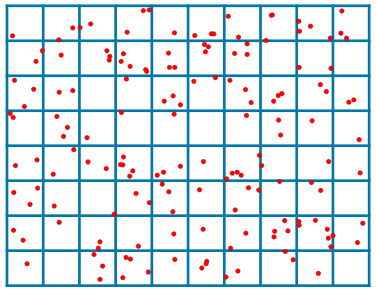
Plasma acceleration is described by kinetic plasma dynamics

$$\frac{d\mathbf{x}}{dt} = \mathbf{v},$$
$$\frac{d(\gamma\mathbf{v})}{dt} = \frac{q}{m}(\mathbf{E} + \mathbf{v} \times \mathbf{B}),$$

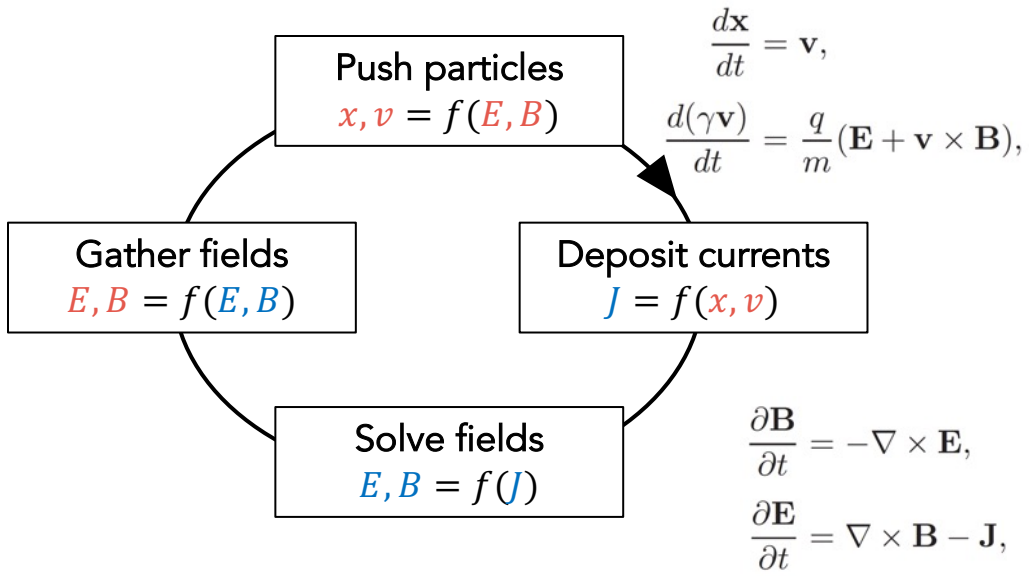
$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E},$$
$$\frac{\partial \mathbf{E}}{\partial t} = \nabla \times \mathbf{B} - \mathbf{J},$$

Plasma acceleration is described by kinetic plasma dynamics





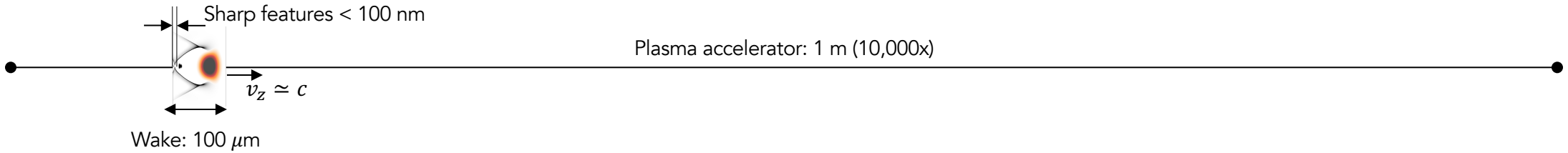
Regular mesh
Macroparticles



- *Lagrangian* description of plasma, *Eulerian* description of fields
- Physics can be added
- Time step limited by CFL condition: $\Delta t \simeq \frac{\Delta z}{c}$

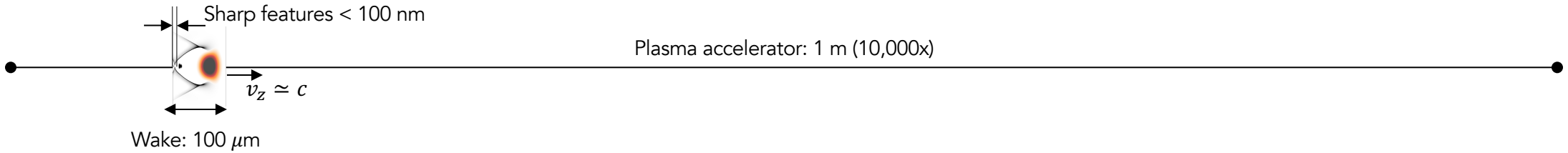
→ 3D (EM) PIC simulations of plasma acceleration are very expensive

Fast plasma acceleration simulations with quasistatic approximation



Problem: the CFL condition limits the time step to $\Delta t < c\Delta z$

Fast plasma acceleration simulations with quasistatic approximation

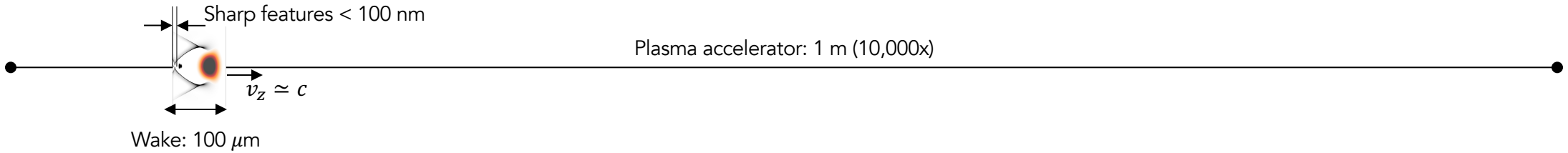


Problem: the CFL condition limits the time step to $\Delta t < c\Delta z$

Two main methods for larger time steps

- **Boosted frame method** [J.-L. Vay PRL 98, 130405 (2007)]
  - Prone to Numerical Cherenkov Instability (NCI)
  - Mitigation methods exist [R. Lehe et al., PRE 94 (2016), M. Kirchen et al., Phys. Plasmas 23 (2016), A. Pukhov *JCP* 418 (2020)]

Fast plasma acceleration simulations with quasistatic approximation



Problem: the CFL condition limits the time step to $\Delta t < c\Delta z$

Two main methods for larger time steps

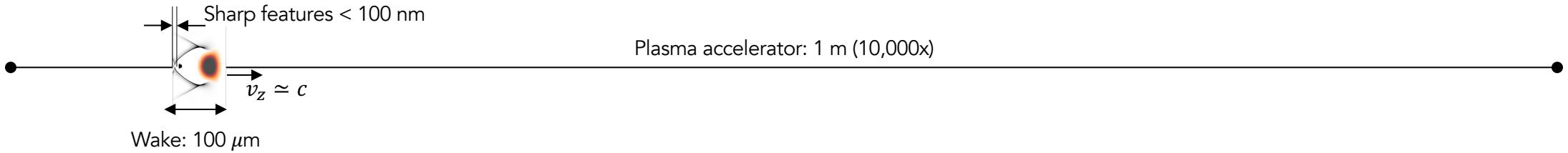
- **Boosted frame method** [J.-L. Vay PRL 98, 130405 (2007)]
  - Prone to Numerical Cherenkov Instability (NCI)
  - Mitigation methods exist [R. Lehe et al., PRE 94 (2016), M. Kirchen et al., Phys. Plasmas 23 (2016), A. Pukhov *JCP* 418 (2020)]

- **Quasi-static particle-in-cell**
  - Beam & wake: $\boldsymbol{v} \sim c\boldsymbol{e_z}$
  - Quasi-static approximation

  → No CFL condition, large time step for the beam

  → Cannot capture injection

Fast plasma acceleration simulations with quasistatic approximation

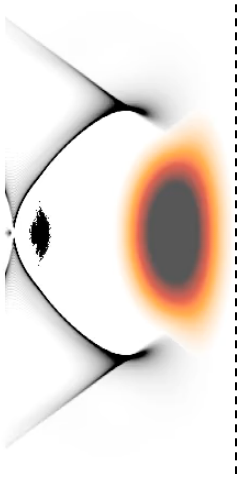


Problem: the CFL condition limits the time step to $\Delta t < c\Delta z$

Two main methods for larger time steps

- **Boosted frame method** [J.-L. Vay PRL 98, 130405 (2007)]
  - Prone to Numerical Cherenkov Instability (NCI)
  - Mitigation methods exist [R. Lehe et al., PRE 94 (2016), M. Kirchen et al., Phys. Plasmas 23 (2016), A. Pukhov *JCP* 418 (2020)]

- **Quasi-static particle-in-cell**
  - Beam & wake: $\boldsymbol{v} \sim c\boldsymbol{e_z}$
  - Quasi-static approximation
  - → No CFL condition, large time step for the beam
  - → Cannot capture injection

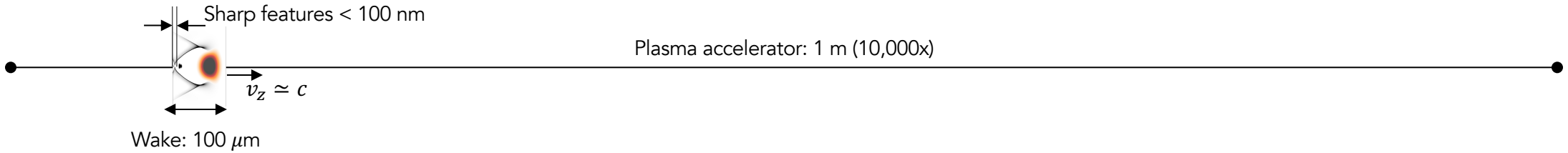


"the fields $\boldsymbol{a}$ and $\boldsymbol{\phi}$ which drive the plasma are expected to change little during a transit time of the plasma through the laser pulse"

"Under the quasistatic approximation, the $\partial/\partial z$ derivatives may be neglected in the electron (fluid) equations"

[P. Sprangle et al., PRL 64, 17 (1990)]

Fast plasma acceleration simulations with quasistatic approximation



Problem: the CFL condition limits the time step to $\Delta t < c\Delta z$

Two main methods for larger time steps

- **Boosted frame method** [J.-L. Vay PRL 98, 130405 (2007)]
  - Prone to Numerical Cherenkov Instability (NCI)
  - Mitigation methods exist [R. Lehe et al., PRE 94 (2016), M. Kirchen et al., Phys. Plasmas 23 (2016), A. Pukhov *JCP* 418 (2020)]

- **Quasi-static particle-in-cell**
  - Beam & wake: $\boldsymbol{v} \sim c\boldsymbol{e_z}$
  - Quasi-static approximation
  - → No CFL condition, large time step for the beam
  - → Cannot capture injection

$$\nabla_\perp^2 \psi = -\frac{1}{\epsilon_0}\left(\rho - \frac{1}{c} j_z\right)$$

$$E_x - c\, B_y = -\partial_x \psi$$

$$E_y + c\, B_x = -\partial_y \psi$$

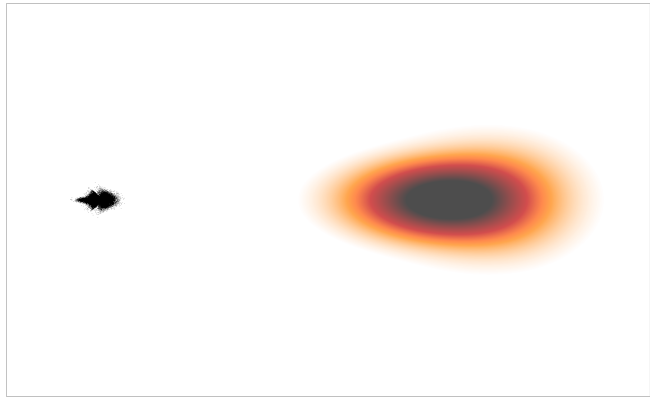
$$\nabla_\perp^2 E_z = c\mu_0 \left(\partial_x j_x + \partial_y j_y\right)$$

$$\nabla_\perp^2 B_x = \mu_0 \left(-\partial_y j_z + \partial_\zeta j_y\right)$$

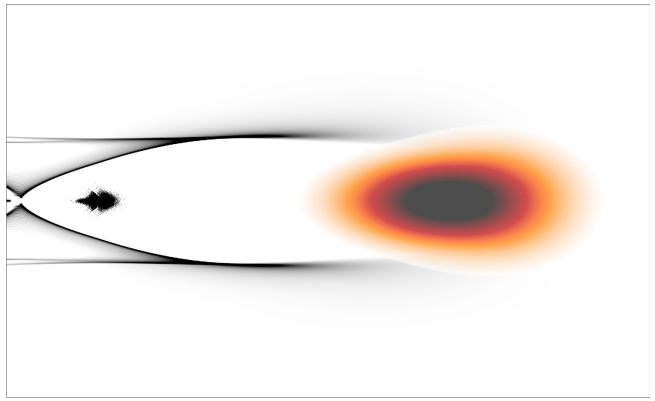
$$\nabla_\perp^2 B_y = \mu_0 \left(\partial_x j_z - \partial_\zeta j_x\right)$$

$$\nabla_\perp^2 B_z = \mu_0 \left(\partial_y j_x - \partial_x j_y\right)$$

In QS PIC, plasma and laser/particle beams are treated differently

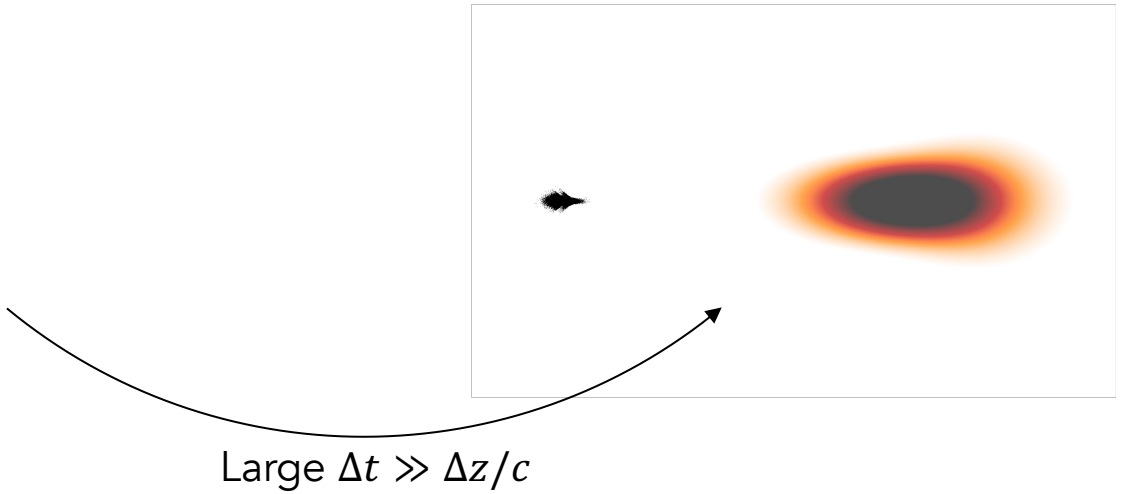


In QS PIC, plasma and laser/particle beams are treated differently



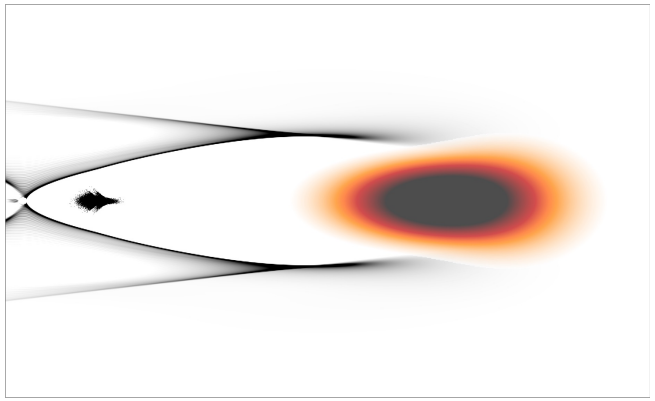
- Compute plasma response (expensive)

In QS PIC, plasma and laser/particle beams are treated differently



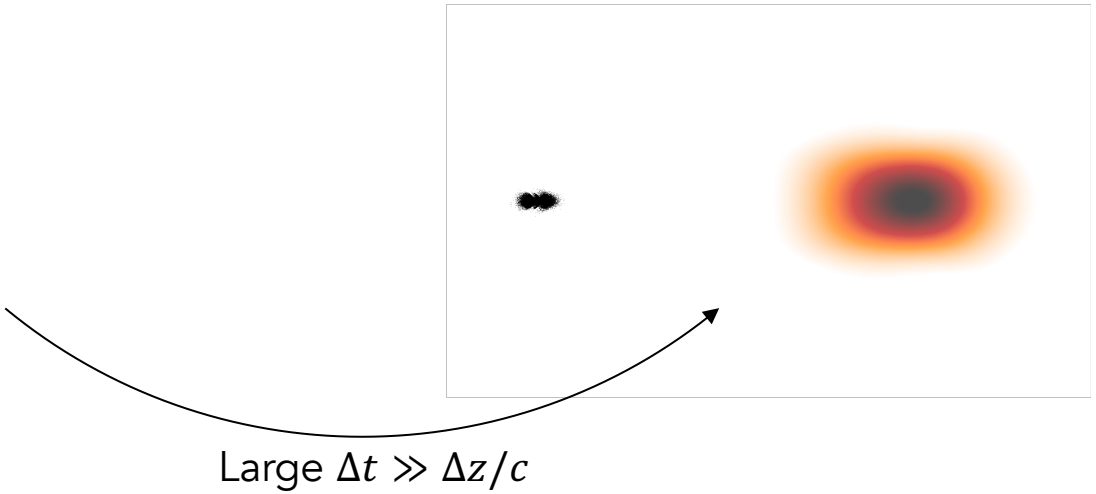
- Compute plasma response (expensive)
- Advance laser and beams with large Δt (cheap)

In QS PIC, plasma and laser/particle beams are treated differently



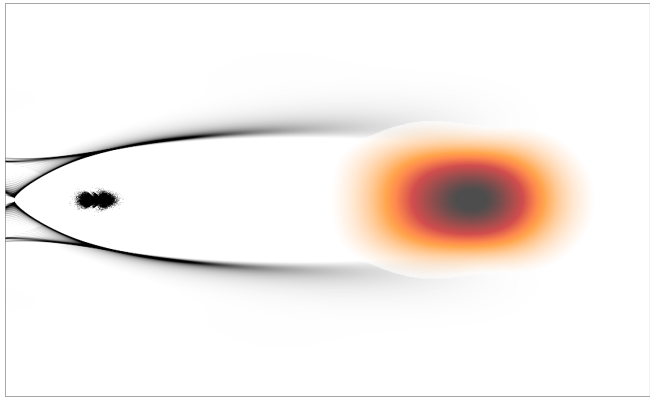
- Compute plasma response (expensive)
- Advance laser and beams with large Δt (cheap)

In QS PIC, plasma and laser/particle beams are treated differently



- Compute plasma response (expensive)
- Advance laser and beams with large Δt (cheap)

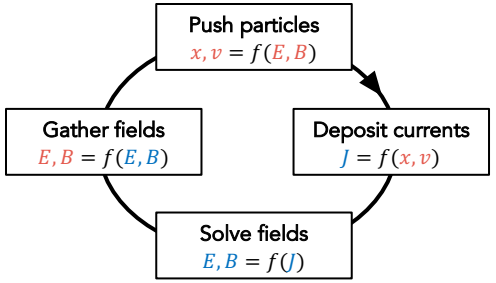
In QS PIC, plasma and laser/particle beams are treated differently



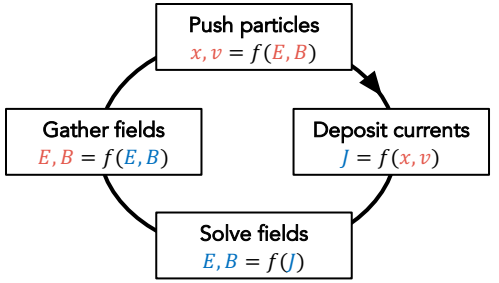
- Compute plasma response (expensive)
- Advance laser and beams with large Δt (cheap)

The plasma response is computed with a swipe from head to tile

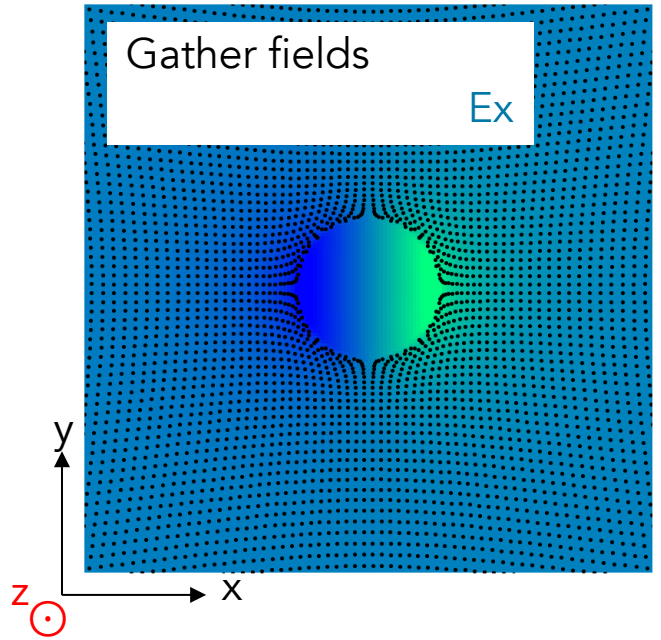
The plasma response is computed with a swipe from head to tile



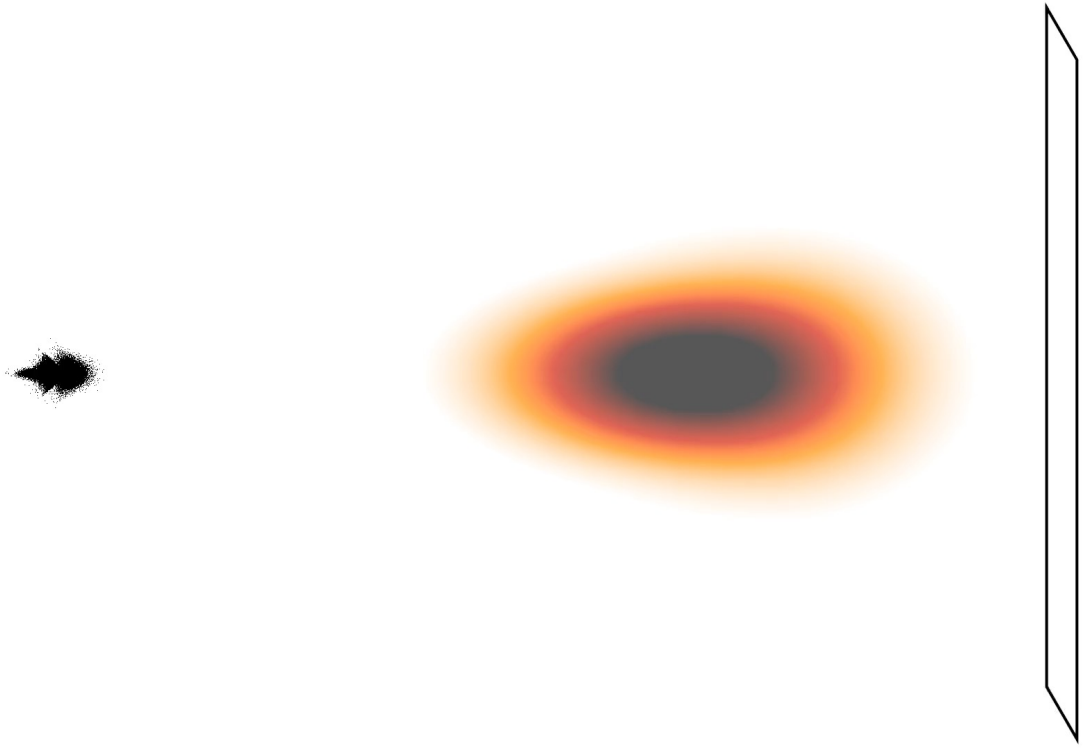
The plasma response is computed with a swipe from head to tile

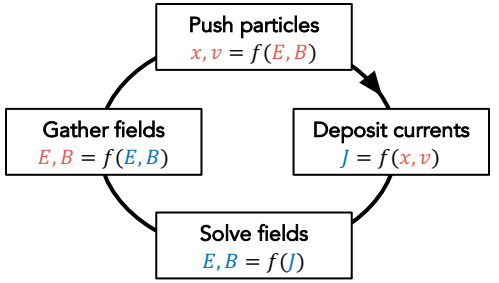


2D slice
Field & particles

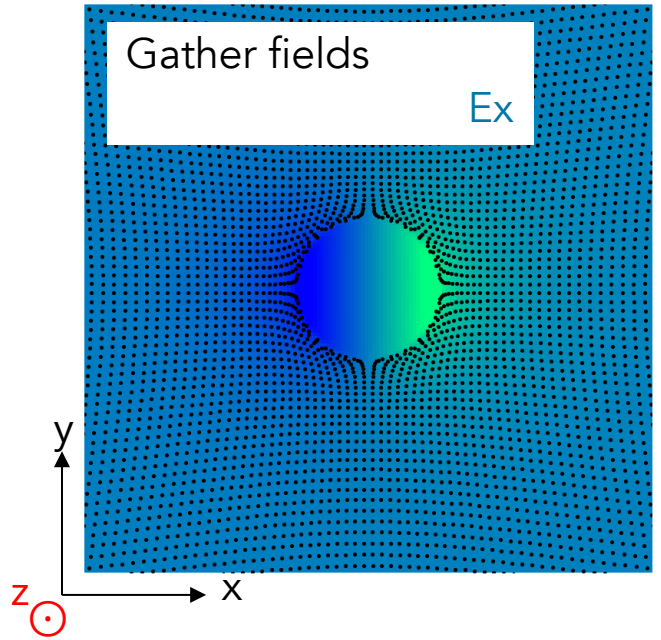


The plasma response is computed with a swipe from head to tile

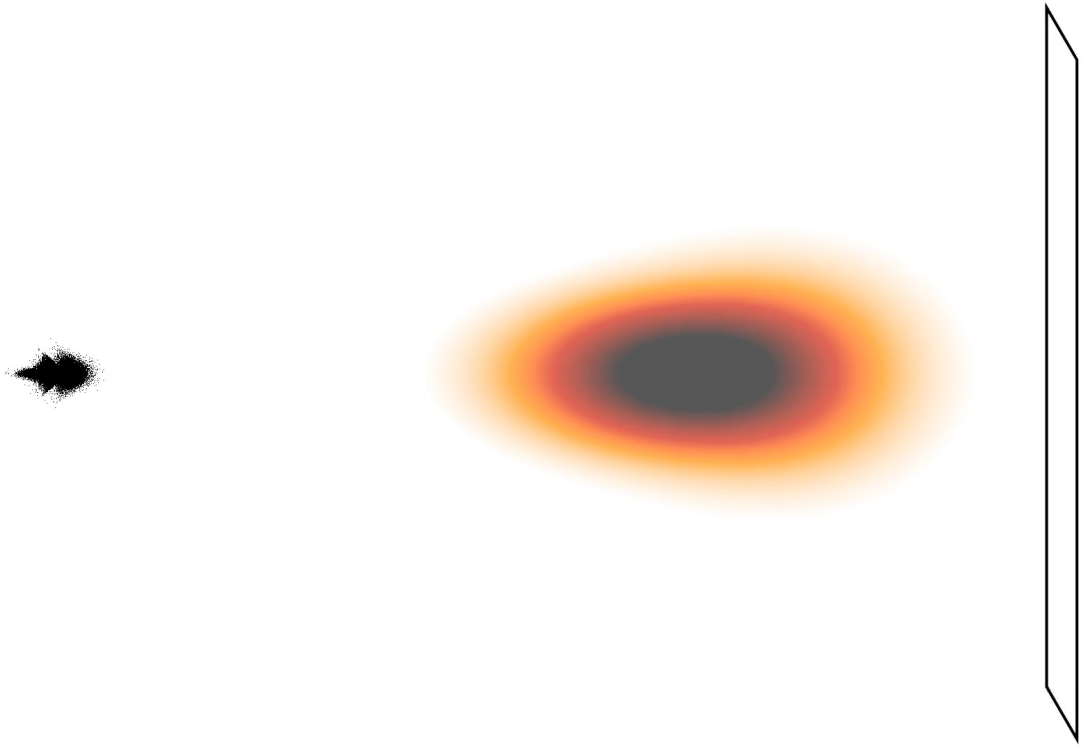


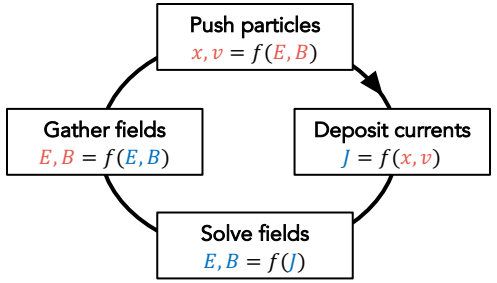


2D slice
Field & particles

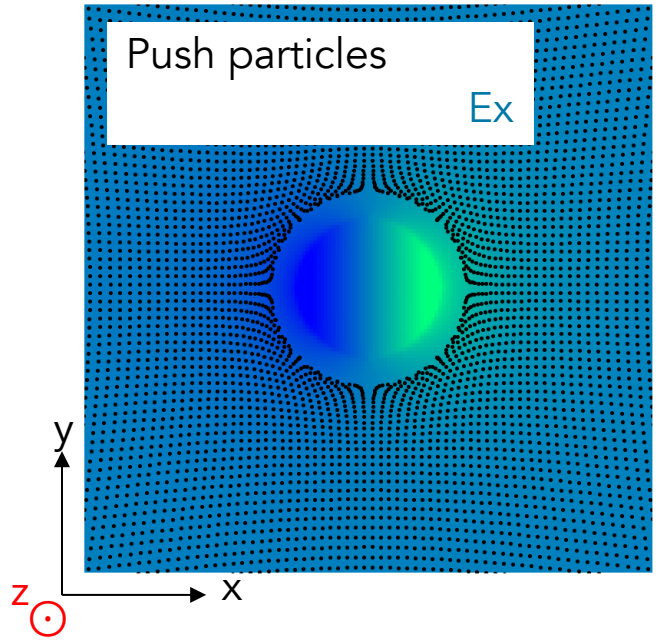


The plasma response is computed with a swipe from head to tile

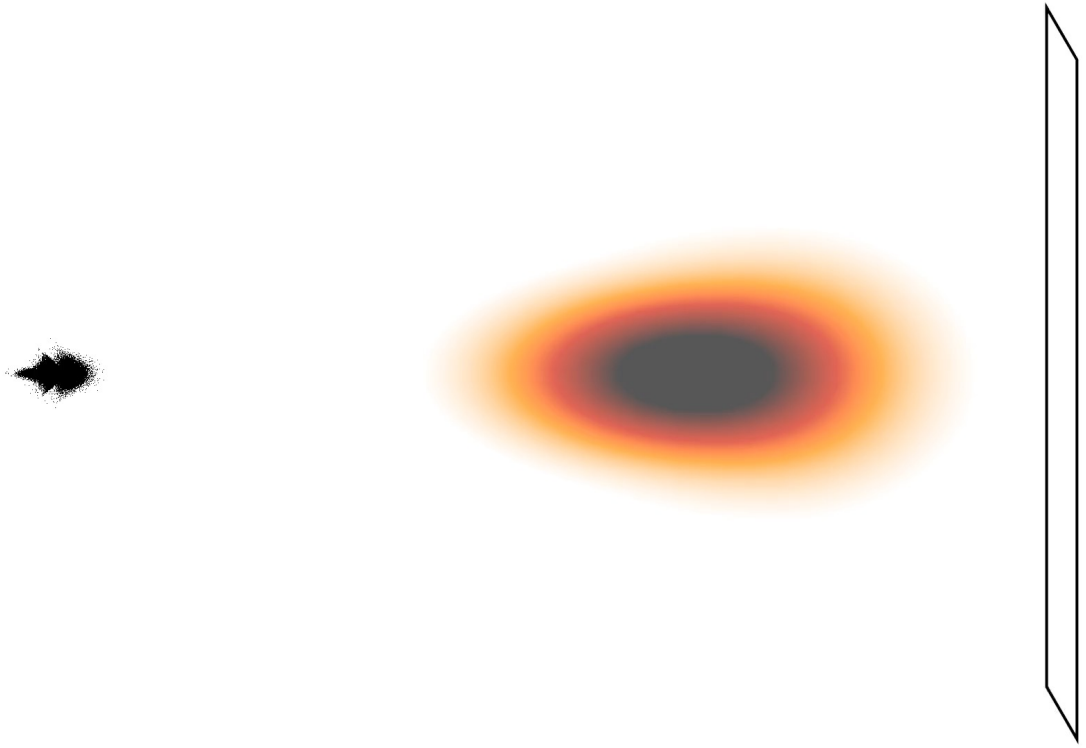


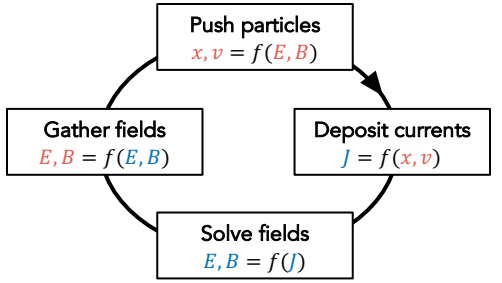


2D slice
Field & particles

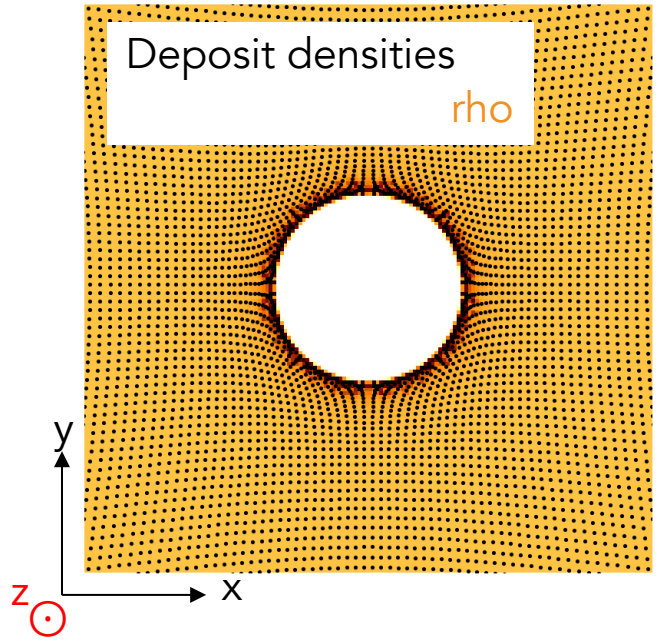


The plasma response is computed with a swipe from head to tile

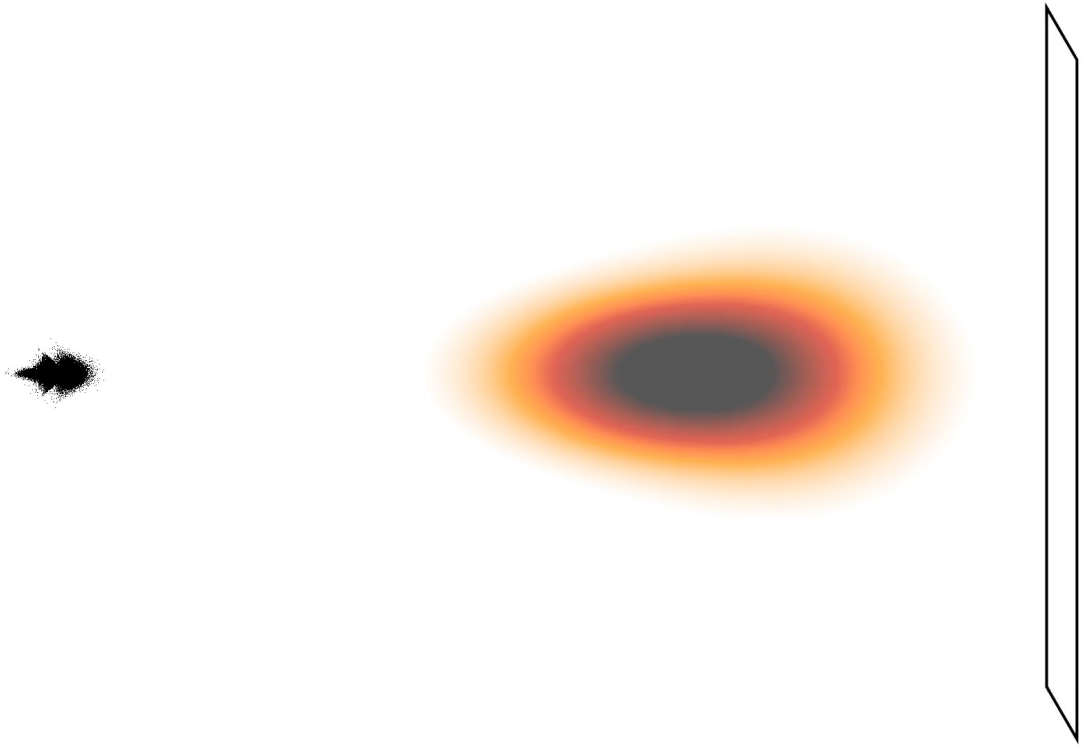


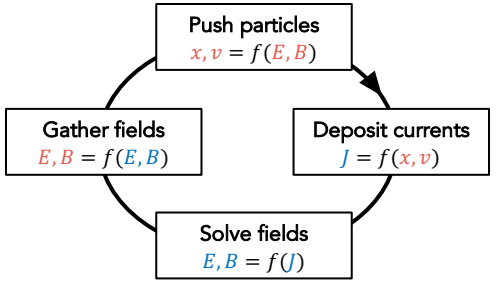


2D slice
Field & particles

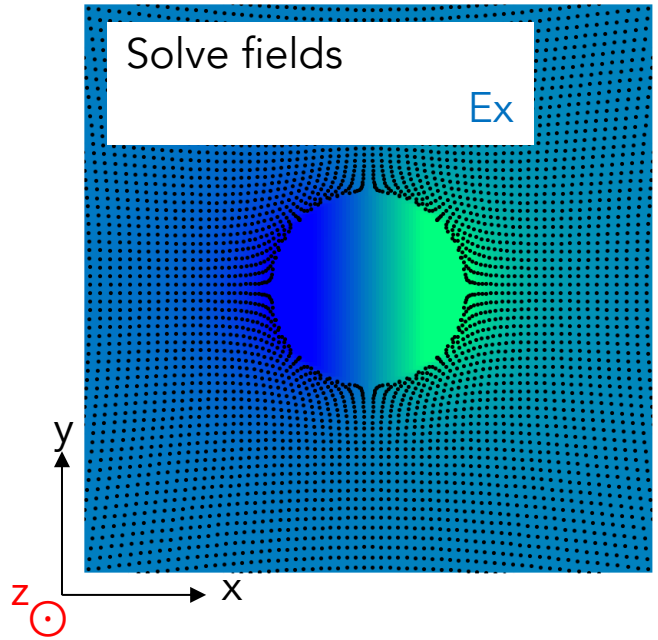


The plasma response is computed with a swipe from head to tile

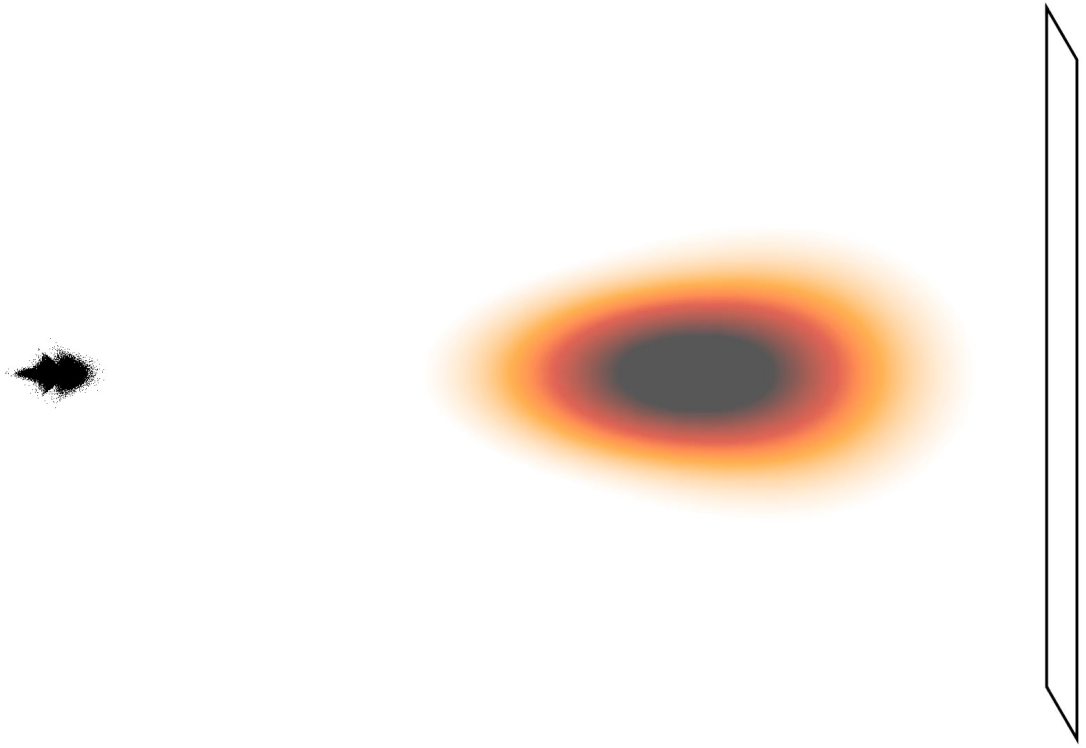


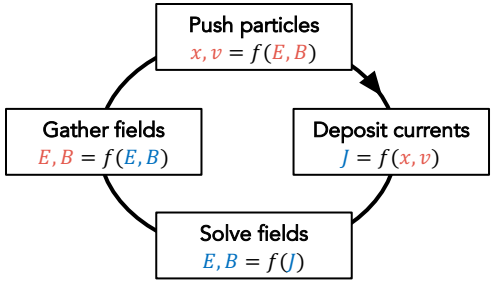


2D slice
Field & particles

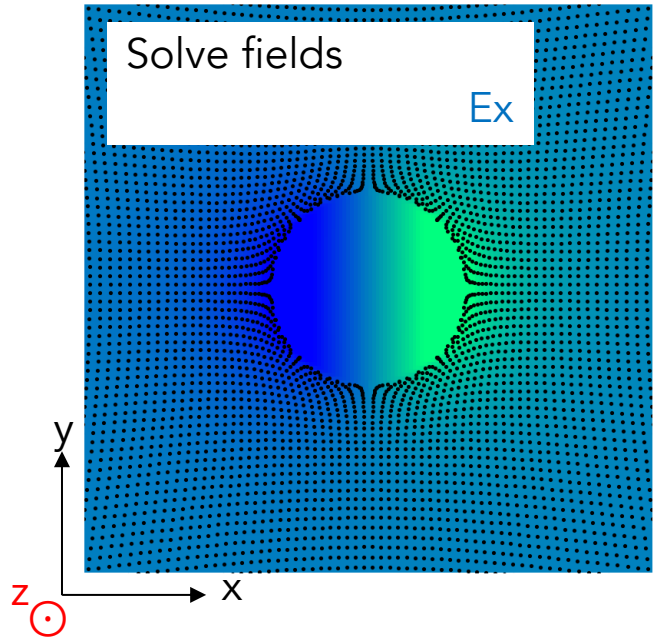


The plasma response is computed with a swipe from head to tile





2D slice
Field & particles



- All PIC operations occur on the 2D transverse domain
- Plasma particles are advanced in ζ (z), not in time
- (specific to HiPACE++: beams and lasers also advanced in the swipe)
- A simulation does $\boldsymbol{n_t \times n_z}$ PIC iterations on domain $\boldsymbol{n_x \times n_y}$
- (EM PIC: does $\boldsymbol{n_t}$ PIC iterations on domain $\boldsymbol{n_x \times n_y \times n_z}$)

Recent advances improved accuracy of the field solver

$$\nabla_\perp^2 \psi = -\frac{1}{\epsilon_0}\left(\rho - \frac{1}{c} j_z\right)$$

$$E_x - c\, B_y = -\partial_x \psi$$

$$E_y + c\, B_x = -\partial_y \psi$$

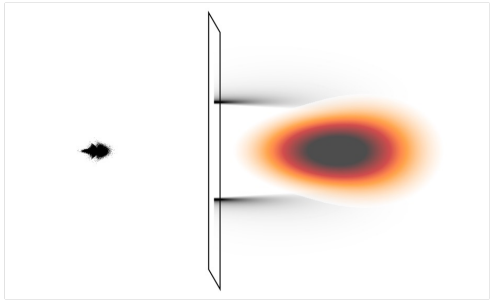
$$\nabla_\perp^2 E_z = c\mu_0 \left(\partial_x j_x + \partial_y j_y\right)$$

$$\nabla_\perp^2 B_x = \mu_0 \left(-\partial_y j_z + \partial_\zeta j_y\right)$$

$$\nabla_\perp^2 B_y = \mu_0 \left(\partial_x j_z - \partial_\zeta j_x\right)$$

$$\nabla_\perp^2 B_z = \mu_0 \left(\partial_y j_x - \partial_x j_y\right)$$

Recent advances improved accuracy of the field solver



$$\nabla_\perp^2 \psi = -\frac{1}{\epsilon_0}\left(\rho - \frac{1}{c} j_z\right)$$

$$E_x - c\, B_y = -\partial_x \psi$$

$$E_y + c\, B_x = -\partial_y \psi$$

$$\nabla_\perp^2 E_z = c\mu_0 \left(\partial_x j_x + \partial_y j_y\right)$$

$$\nabla_\perp^2 B_x = \mu_0 \left(-\partial_y j_z + \partial_\zeta j_y\right)$$

$$\nabla_\perp^2 B_y = \mu_0 \left(\partial_x j_z - \partial_\zeta j_x\right)$$

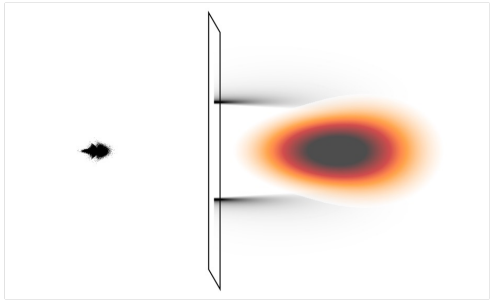
$$\nabla_\perp^2 B_z = \mu_0 \left(\partial_y j_x - \partial_x j_y\right)$$

Source terms $\partial_\zeta j_{x/y}$ are difficult to obtain

- predictor-corrector solver: *the old one*
  - [Mora & T. Antonsen, Phys. Plasmas (1997), W. An et al., JCP (2013)]
  - Not very stable
- explicit solver: *the new one*
  - [T. Wang et al., Phys. Plasmas (2017), P. Baxevanis & G. Suakov, PRAB (2018), T. Wang, et al. PRAB 25.10 (2022)]
  - Analytic integration of the source term
  - Gives a screened Poisson equation, solved with multigrid solver

$$\nabla_\perp^2 B_\perp - \frac{n^*}{1+\psi} B_\perp = -[e_z \times S]$$

Recent advances improved accuracy of the field solver



$$\nabla_\perp^2 \psi = -\frac{1}{\epsilon_0}\left(\rho - \frac{1}{c} j_z\right)$$

$$E_x - c\,B_y = -\partial_x \psi$$

$$E_y + c\,B_x = -\partial_y \psi$$

$$\nabla_\perp^2 E_z = c\mu_0\left(\partial_x j_x + \partial_y j_y\right)$$

$$\nabla_\perp^2 B_x = \mu_0\left(-\partial_y j_z + \partial_\zeta j_y\right)$$

$$\nabla_\perp^2 B_y = \mu_0\left(\partial_x j_z - \partial_\zeta j_x\right)$$

$$\nabla_\perp^2 B_z = \mu_0\left(\partial_y j_x - \partial_x j_y\right)$$

Source terms $\partial_\zeta j_{x/y}$ are difficult to obtain

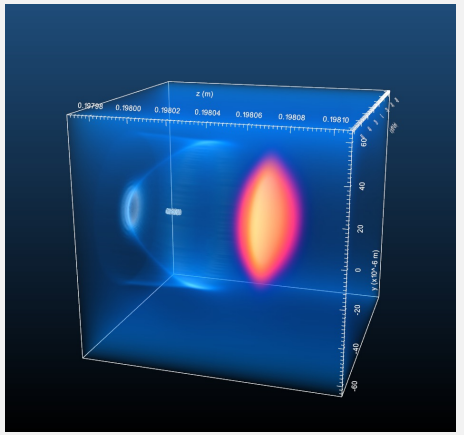
- **predictor-corrector solver**: *the old one*
  - [Mora & T. Antonsen, Phys. Plasmas (1997), W. An et al., JCP (2013)]
  - Not very stable
- **explicit solver**: *the new one*
  - [T. Wang et al., Phys. Plasmas (2017), P. Baxevanis & G. Suakov, PRAB (2018), T. Wang, et al. PRAB 25.10 (2022)]
  - Analytic integration of the source term
  - Gives a screened Poisson equation, solved with multigrid solver

$$\nabla_\perp^2 B_\perp - \frac{n^*}{1+\psi} B_\perp = -[e_z \times S]$$

WakeT

Wakefield particle Tracker [1]

- 2D (RZ) axisymmetric
- Particle beam or laser pulse drivers [2]
- Gridless model based on explicit solver [3]
- Python, open-source, openPMD



→ **Realistic multi-stage simulations within second/minutes on a laptop**

https://github.com/AngelFP/Wake-T
https://wake-t.readthedocs.io

[1] A. Ferran Pousa et al., J. Phys.: Conf. Ser. (2019)
[2] C. Benedetti et al., PPCF 60 014002 (2018)
[3] P. Baxevanis and G. Stupakov, PRAB 21 (2018)

→ Ask Ángel

Accelerated computing dominates supercomputing

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, Atos EuroHPC/CINECA Italy	1,824,768	238.70	304.47	7,404
5	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096

https://www.top500.org/lists/top500/2023/06/

Accelerated computing dominates supercomputing

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, Atos EuroHPC/CINECA Italy	1,824,768	238.70	304.47	7,404
5	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096

https://www.top500.org/lists/top500/2023/06/

Equipped with GPUs:

Top500 33/50

Green500 46/50

Accelerated computing dominates supercomputing

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, Atos EuroHPC/CINECA Italy	1,824,768	238.70	304.47	7,404
5	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096

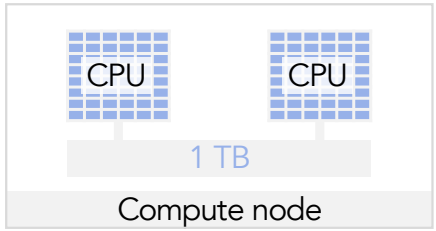
https://www.top500.org/lists/top500/2023/06/

Equipped with GPUs:

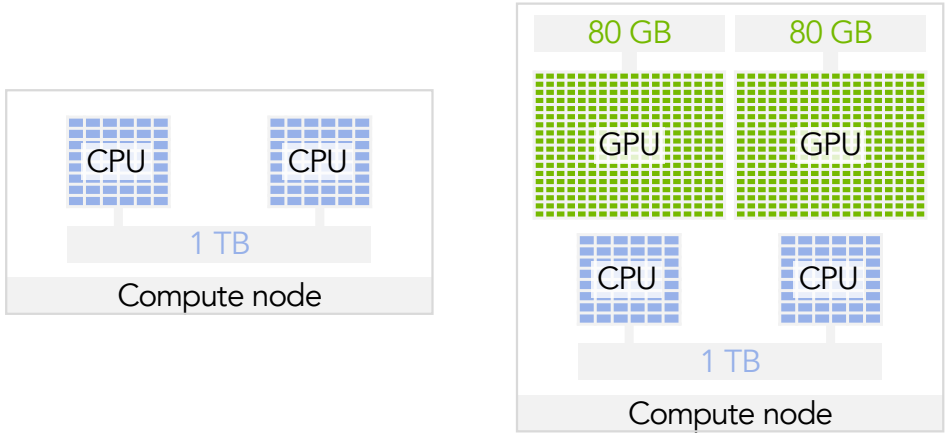
Top500	33/50
Green500	46/50

- GPUs meant for rendering, adopted by HPC
- Strong trend in HPC & scientific computing towards GPU (fast and energy-efficient)
- Trend driven by AI, here to stay

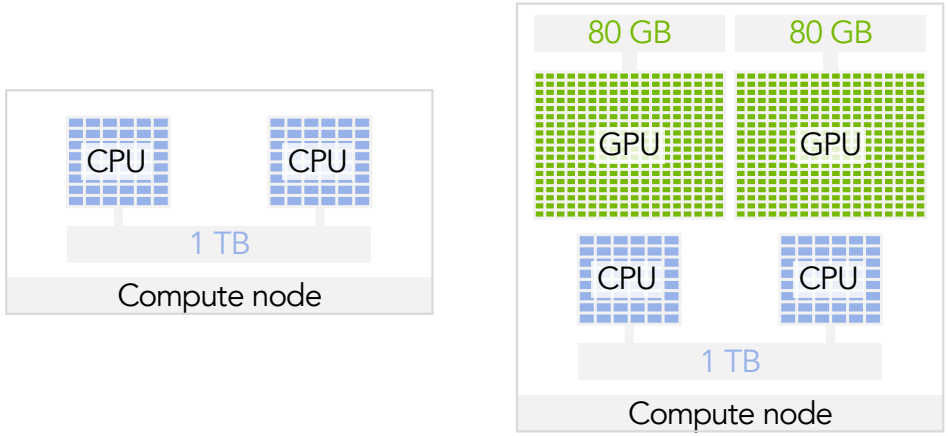
Accelerated computing dominates supercomputing



Accelerated computing dominates supercomputing



Accelerated computing dominates supercomputing

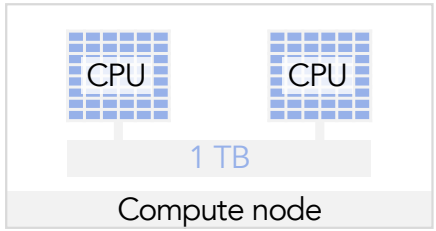


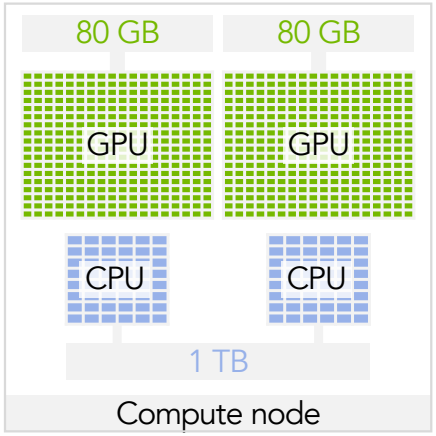
- What changes?
  - Many slow cores → expose parallelism
  - Modest memory available

- Performance-portability
  - In particular GPU computing
  - Portability layer (Kokkos, Alpaka, RAJA) C++

- Open Source & Open Repository
  - Software can be freely used, modified and shared
  - Encourages flexible, modular code
  - Favor good dependency graph rather than duplication

Accelerated computing dominates supercomputing





CPU

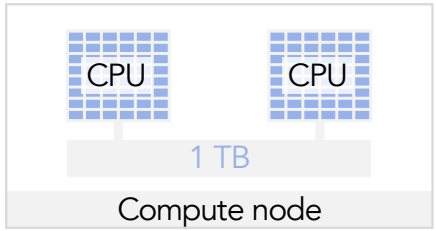
```
for(int i=0; i<N; i++){
    xp[i] += 1.;
}
```

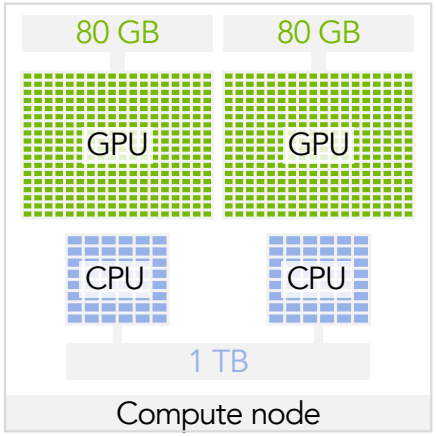
GPU (CUDA)

```
kernel(int* xp) {
    int i = blockIdx.x *
            blockDim.x
            + threadIdx.x;
    if (i<N) xp[i] += 1.; }
kernel<<<N, 256>>>(xp);
```

- What changes?
  - Many slow cores → expose parallelism
  - Modest memory available
- Performance-portability
  - In particular GPU computing
  - Portability layer (Kokkos, Alpaka, RAJA) C++
- Open Source & Open Repository
  - Software can be freely used, modified and shared
  - Encourages flexible, modular code
  - Favor good dependency graph rather than duplication

Accelerated computing dominates supercomputing





CPU

```
for(int i=0; i<N; i++){
    xp[i] += 1.;
}
```

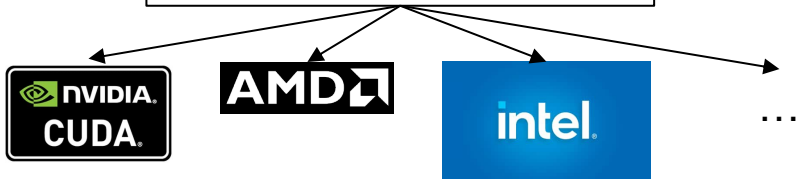
GPU (CUDA)

```
kernel(int* xp) {
    int i = blockIdx.x *
            blockDim.x
            + threadIdx.x;
    if (i<N) xp[i] += 1.; }
kernel<<<N, 256>>>(xp);
```

- What changes?
  - Many slow cores → expose parallelism
  - Modest memory available
- Performance-portability
  - In particular GPU computing
  - Portability layer (Kokkos, Alpaka, RAJA) C++
- Open Source & Open Repository
  - Software can be freely used, modified and shared
  - Encourages flexible, modular code
  - Favor good dependency graph rather than duplication

Any platform

```
ParallelFor( N,
    [=] (int i) {
        xp[i] += 1.;
    }
);
```



HiPACE++ – The Team

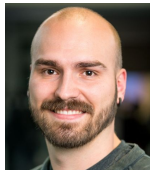
Advanced algorithms and high-performance computing for fast and energy-efficient 3D simulations of plasma acceleration – for everyone


Maxence Thévenet (lead)


Severin Diederichs


Alexander Sinn

DESY – MPA


Axel Huebl


Rémi Lehe


Jean-Luc Vay

LBNL – AMP

ECP *project WarpX*


Andrew Myers


Weiqun Zhang

LBNL – AMCR

Developers of AMReX


Carlo Benedetti

LBNL – BELLA

- Started mid-2020
- International project, open-source
- New contributors most welcome!

DESY. BERKELEY LAB

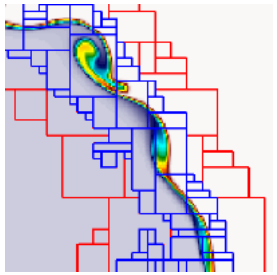
HiPACE++ – The Code

Advanced algorithms and high-performance computing for fast and energy-efficient 3D simulations of plasma acceleration – for everyone

- **Modern HPC standard**
  - Modern C++, Python for pre- and post-processing
  - Cmake build systems, supports various compilers (Clang, GCC, Intel)
  - Open-source, documented, versioning system (1 per month), CI

- **Built on strong libraries**
  - Build on AMReX – Adaptive Mesh Refinement at eXascale for data structures & portability
  - Build on openPMD-api for I/O

- **Meant for inter-operability**
  - OpenPMD standard
  - Shares the ecosystem with WarpX and others: BLAST
  - Beams and laser readers and writers



amrex

https://github.com/Hi-PACE/hipace
https://hipace.readthedocs.io
S. Diederichs et al., *Comput. Phys. Comm.* 278: 108421 (2022)

https://amrex-codes.github.io
https://github.com/openPMD/openPMD-api

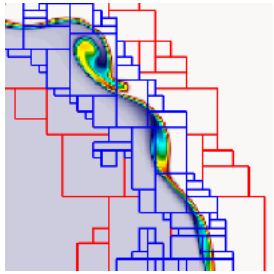
HiPACE++ – The Code

Advanced algorithms and high-performance computing for fast and energy-efficient 3D simulations of plasma acceleration – for everyone

➢ **Modern HPC standard**
- Modern C++, Python for pre- and post-processing
- Cmake build systems, supports various compilers (Clang, GCC, Intel)
- Open-source, documented, versioning system (1 per month), CI

➢ **Built on strong libraries**
- Build on AMReX – Adaptive Mesh Refinement at eXascale for data structures & portability
- Build on openPMD-api for I/O

➢ **Meant for inter-operability**
- OpenPMD standard
- Shares the ecosystem with WarpX and others: BLAST
- Beams and laser readers and writers



amrex

https://github.com/Hi-PACE/hipace
https://hipace.readthedocs.io
S. Diederichs et al., *Comput. Phys. Comm.* 278: 108421 (2022)

https://amrex-codes.github.io
https://github.com/openPMD/openPMD-api

HiPACE++ – The Code

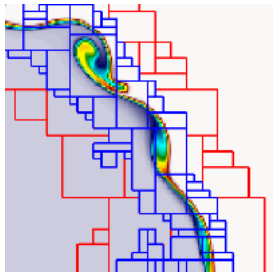
Advanced algorithms and high-performance computing for fast and energy-efficient 3D simulations of plasma acceleration – for everyone

- **Modern HPC standard**
  - Modern C++, Python for pre- and post-processing
  - Cmake build systems, supports various compilers (Clang, GCC, Intel)
  - Open-source, documented, versioning system (1 per month), CI

- **Built on strong libraries**
  - Build on AMReX – Adaptive Mesh Refinement at eXascale for data structures & portability
  - Build on openPMD-api for I/O

See presentation by Rémi Lehe (#3)

- **Meant for inter-operability**
  - OpenPMD standard
  - Shares the ecosystem with WarpX and others: BLAST
  - Beams and laser readers and writers



amrex

https://amrex-codes.github.io

https://github.com/openPMD/openPMD-api

https://github.com/Hi-PACE/hipace

https://hipace.readthedocs.io

S. Diederichs et al., *Comput. Phys. Comm.* 278: 108421 (2022)

HiPACE++ – The Capabilities

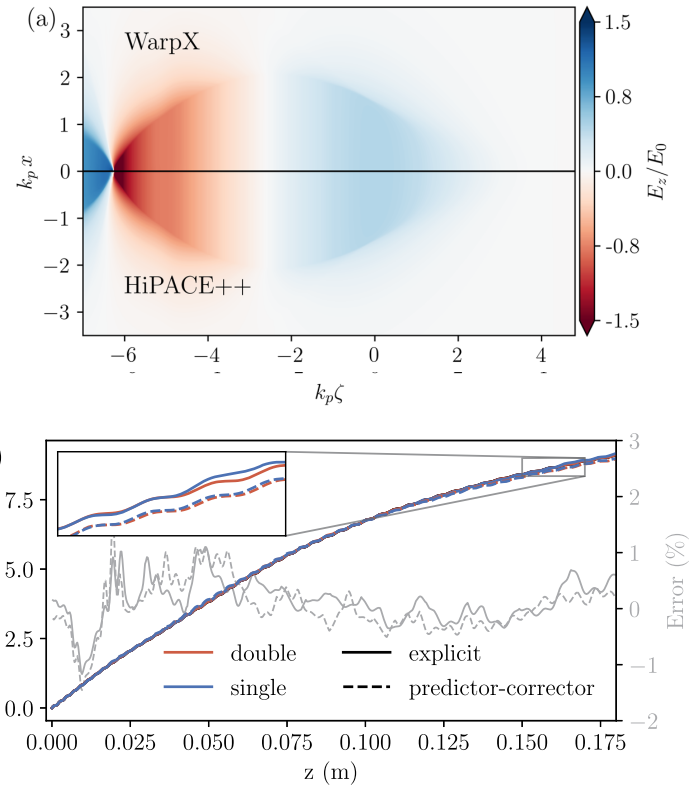
Advanced algorithms and high-performance computing for fast and energy-efficient 3D simulations of plasma acceleration – for everyone

➢ Physics

- Laser and particle beam drivers
- Two unit systems (*SI* and *normalized*)
- Multi-physics (binary collisions, field ionization, radiation reaction)
- SALAME algorithm for automatic beam loading

➢ Numerics

- Single or double-precision
- Explicit solver (and predictor-corrector loop)
- Runs on various architectures (NVIDIA GPUs, AMD GPUs, most CPUs)
  Used on laptops, gaming GPUs, LUMI, Perlmutter, JUWELS Booster, etc.
- Considerable speedup over CPU-only
- Builds on Linux, Windows, MacOS
- Adaptive time step
- In-situ diagnostics



	4 GPUs	1024 CPU cores
Runtime (seconds)	6	556
Cost (node-hours)	6	11900

Used in (a few) scientific publications

R. D'Arcy et al., **Nature** 603.7899 (2022)
S. Diederichs et al., **PRAB** 25.9 (2022)
S. Diederichs et al., **Phys. Plasmas** 29.4 (2022)
F. Peña et al., **arXiv**:2305.09581 (2023)

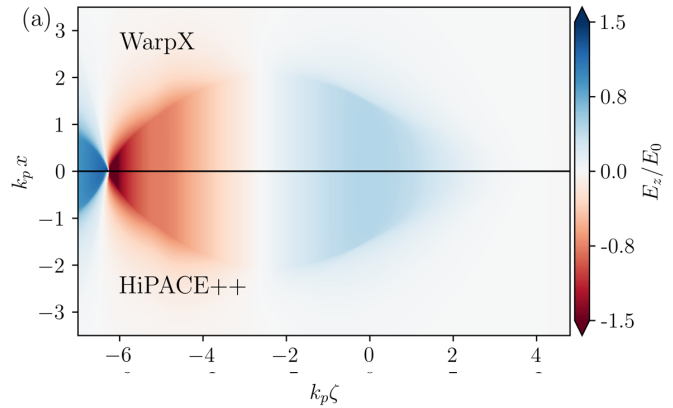
HiPACE++ – The Capabilities

Advanced algorithms and high-performance computing for fast and energy-efficient 3D simulations of plasma acceleration – for everyone

➢ Physics

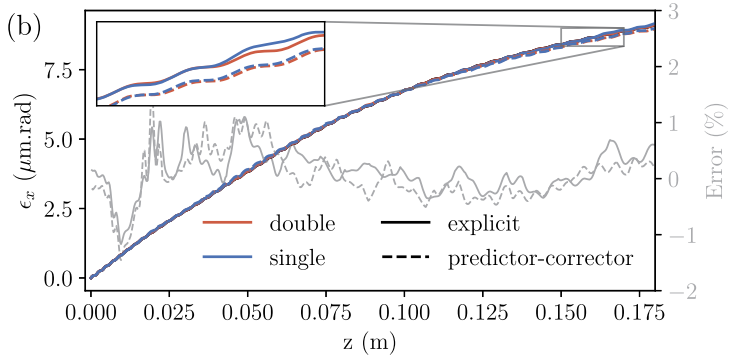
- Laser and particle beam drivers
- Two unit systems (*SI* and *normalized*)
- Multi-physics (binary collisions, field ionization, radiation reaction)
- SALAME algorithm for automatic beam loading

See presentation by Severin Diederichs (#2)



➢ Numerics

- Single or double-precision
- Explicit solver (and predictor-corrector loop)
- Runs on various architectures (NVIDIA GPUs, AMD GPUs, most CPUs)
  Used on laptops, gaming GPUs, LUMI, Perlmutter, JUWELS Booster, etc.
- Considerable speedup over CPU-only
- Builds on Linux, Windows, MacOS
- Adaptive time step
- In-situ diagnostics

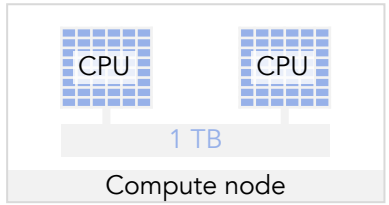


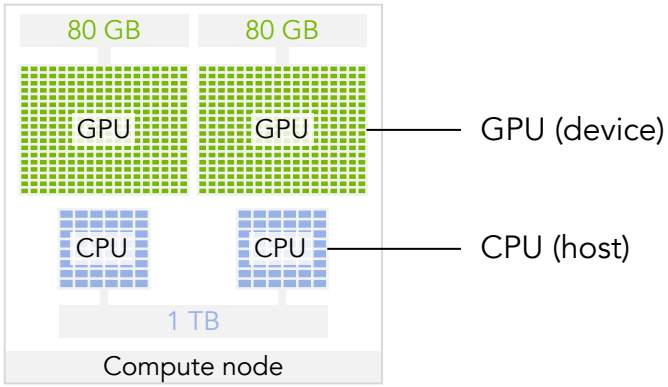
	4 GPUs	1024 CPU cores
Runtime (seconds)	6	556
Cost (node-hours)	6	11900

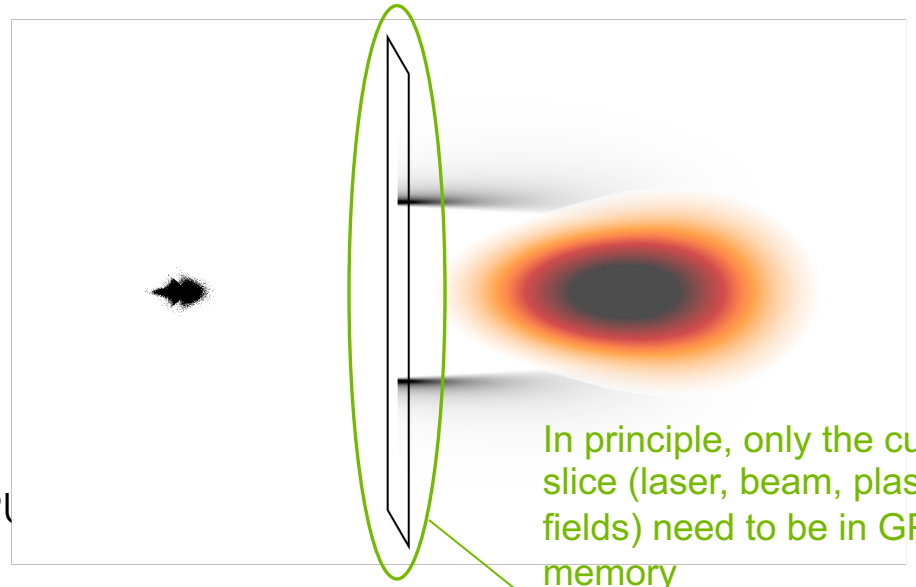
Used in (a few) scientific publications

R. D'Arcy et al., **Nature** 603.7899 (2022)
S. Diederichs et al., **PRAB** 25.9 (2022)
S. Diederichs et al., **Phys. Plasmas** 29.4 (2022)
F. Peña et al., **arXiv**:2305.09581 (2023)

Quasi-static PIC is well-suited for GPU computing





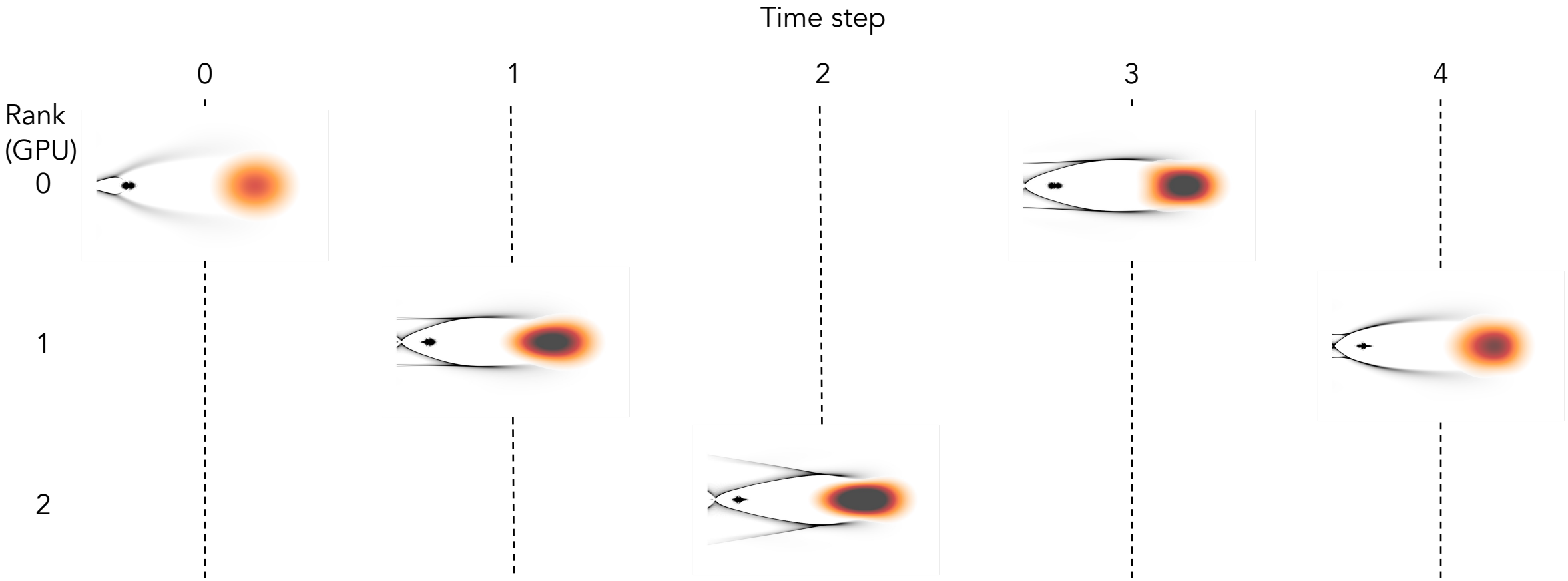


- Massive parallelism: Particle-in-cell can be ported efficiently to GPU
  - PIConGPU (HZDR, Germany), WarpX (LBNL, USA)
  - > millions of cells and particles
  - Main challenge: particle-grid exchanges
- Modest high-bandwidth memory: Only 1 slice in device memory
  - In reality, currently full particle driver
  - High transverse resolution (8000^2) on a single GPU → no comms
  - Parallelization in longitudinal direction

Parallelization: the new (old?) pipeline algorithm

Transverse parallelization:
- None on GPU
- OpenMP on CPU

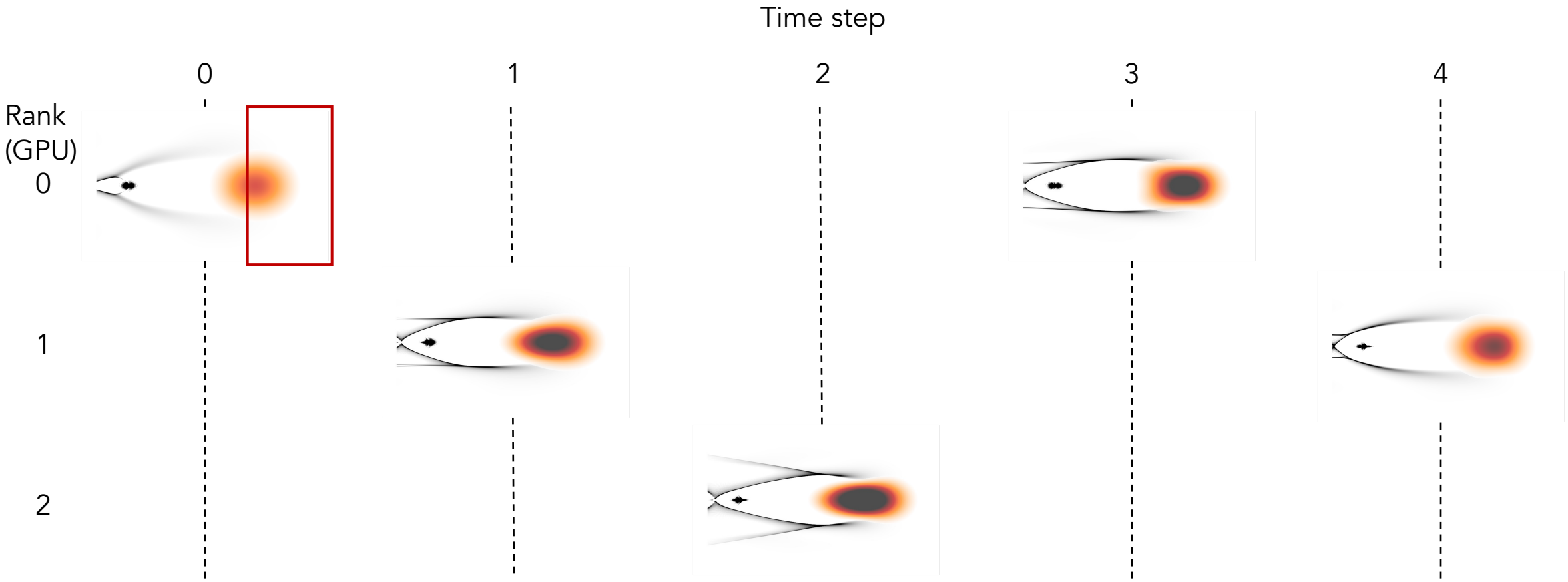
Longitudinal parallelization: MPI (Message Passing Interface)
- $n_{ranks} < n_{slices}$ ($4 \times n_{slices}$ currently in HiPACE++)
- $n_{ranks} \leq n_{steps}$



Parallelization: the new (old?) pipeline algorithm

Transverse parallelization:
- None on GPU
- OpenMP on CPU

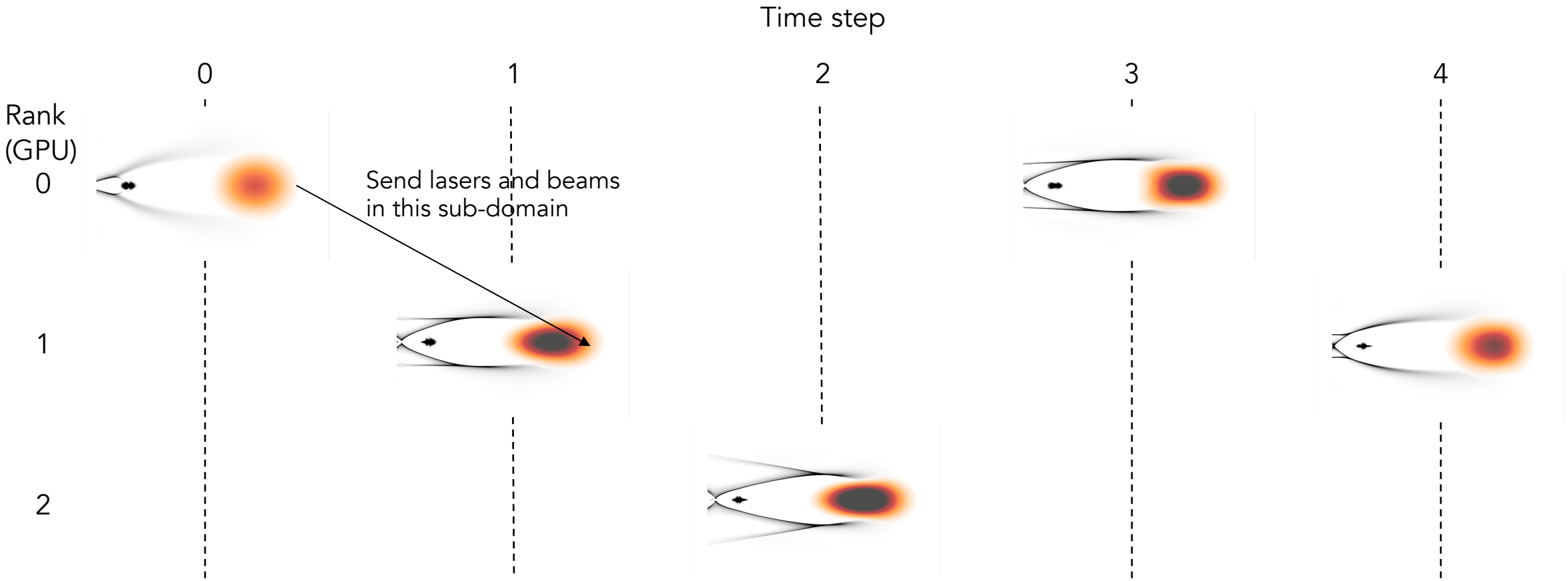
Longitudinal parallelization: MPI (Message Passing Interface)
- $n_{ranks} < n_{slices}$ ($4 \times n_{slices}$ currently in HiPACE++)
- $n_{ranks} \leq n_{steps}$



Parallelization: the new (old?) pipeline algorithm

Transverse parallelization:
- None on GPU
- OpenMP on CPU

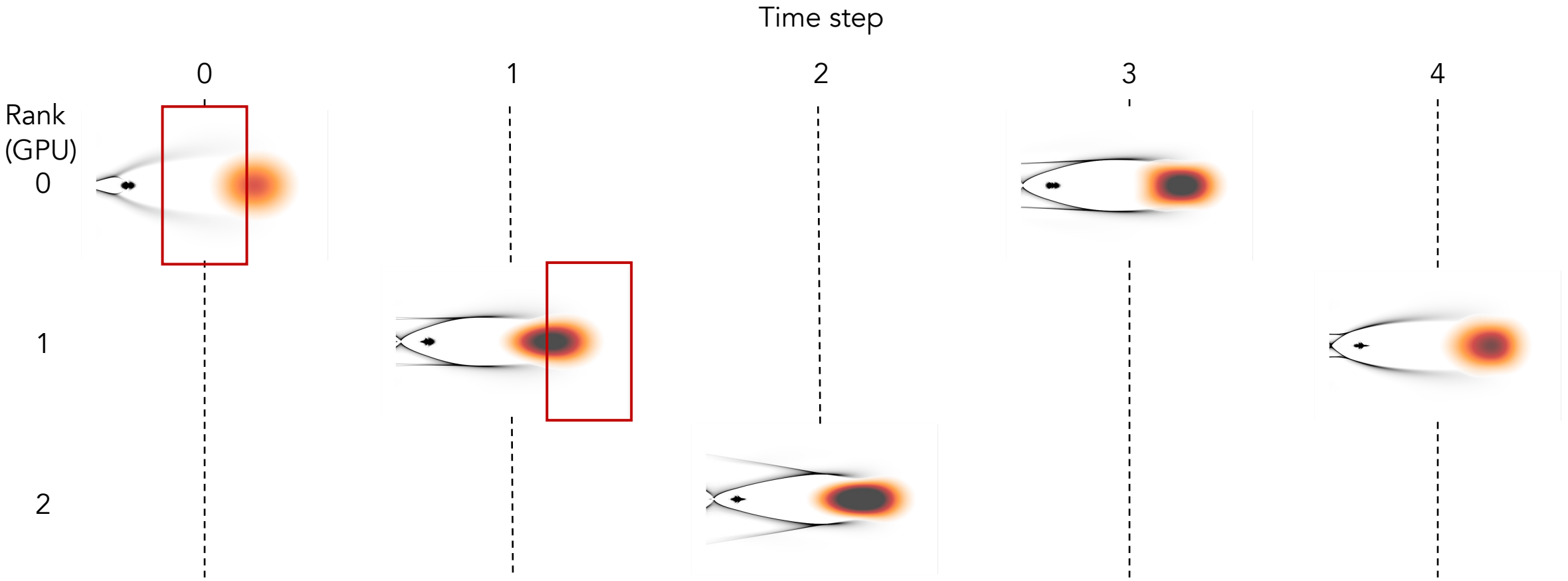
Longitudinal parallelization: MPI (Message Passing Interface)
- $n_{ranks} < n_{slices}$ ($4\times n_{slices}$ currently in HiPACE++)
- $n_{ranks} \leq n_{steps}$



Parallelization: the new (old?) pipeline algorithm

Transverse parallelization:
- None on GPU
- OpenMP on CPU

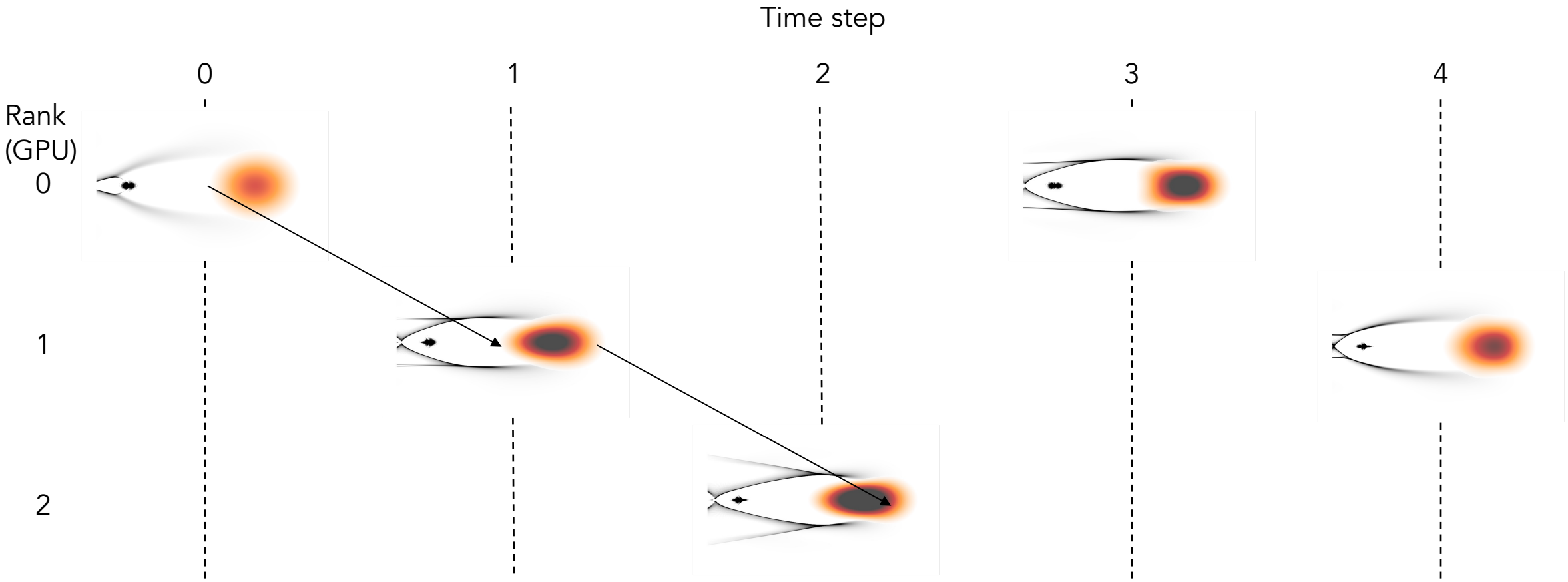
Longitudinal parallelization: MPI (Message Passing Interface)
- $n_{ranks} < n_{slices}$ ($4 \times n_{slices}$ currently in HiPACE++)
- $n_{ranks} \leq n_{steps}$



Parallelization: the new (old?) pipeline algorithm

Transverse parallelization:
- None on GPU
- OpenMP on CPU

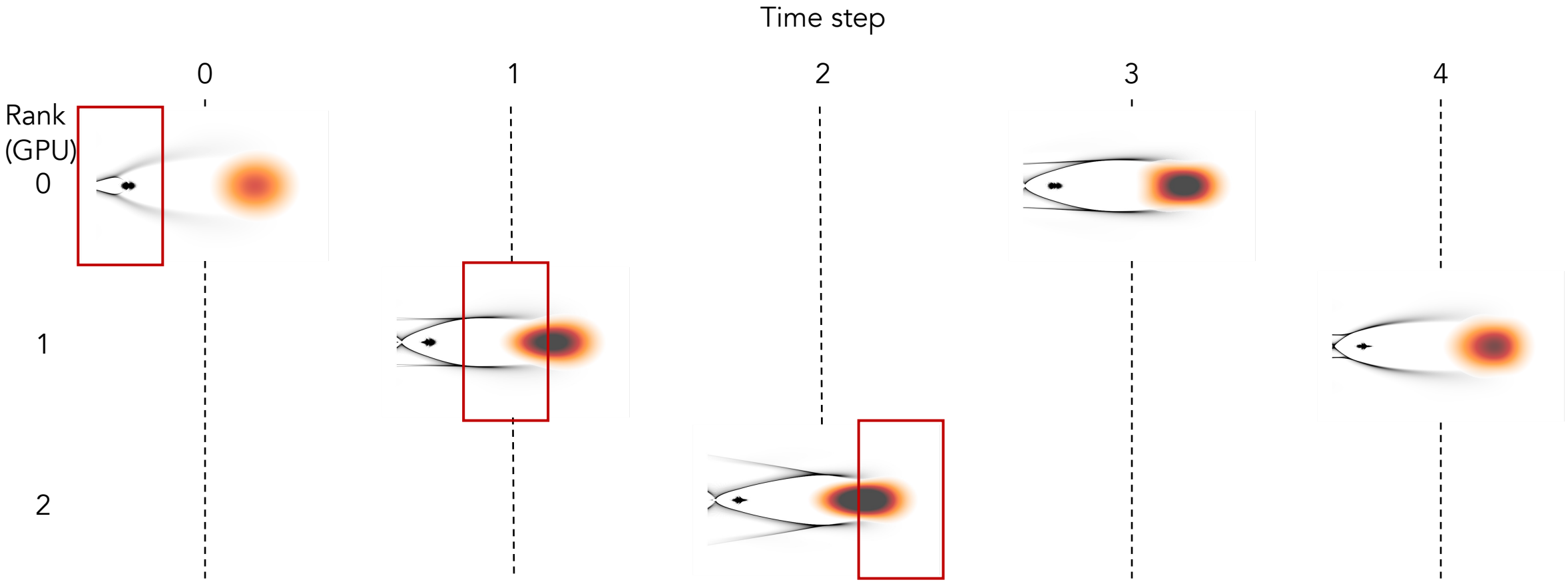
Longitudinal parallelization: MPI (Message Passing Interface)
- $n_{ranks} < n_{slices}$ ($4\times n_{slices}$ currently in HiPACE++)
- $n_{ranks} \leq n_{steps}$



Parallelization: the new (old?) pipeline algorithm

Transverse parallelization:
- None on GPU
- OpenMP on CPU

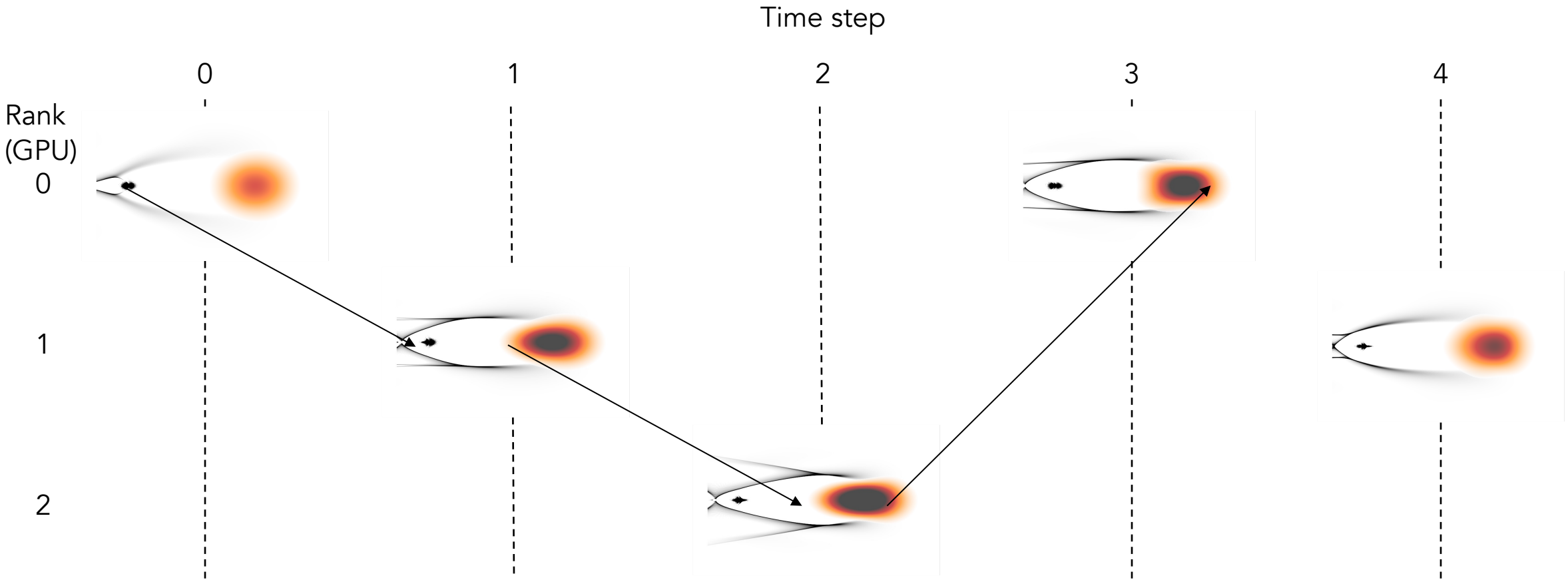
Longitudinal parallelization: MPI (Message Passing Interface)
- $n_{ranks} < n_{slices}$ ($4 \times n_{slices}$ currently in HiPACE++)
- $n_{ranks} \leq n_{steps}$



Parallelization: the new (old?) pipeline algorithm

Transverse parallelization:
- None on GPU
- OpenMP on CPU

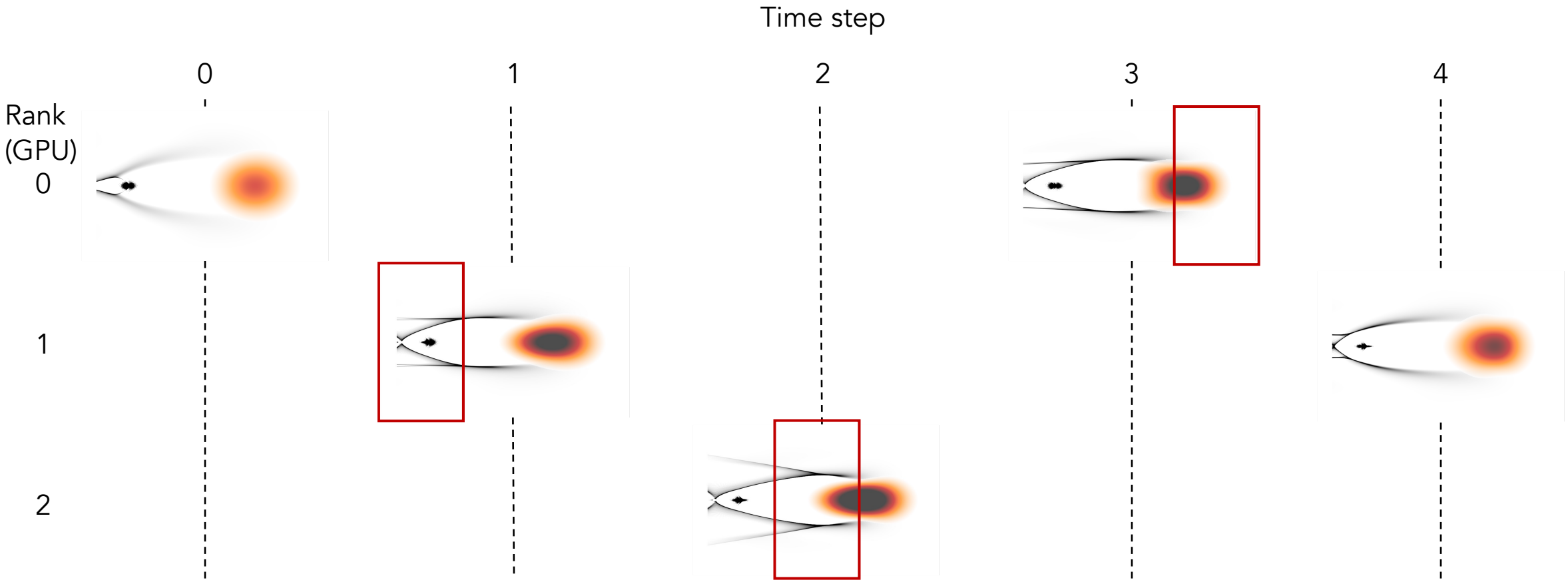
Longitudinal parallelization: MPI (Message Passing Interface)
- $n_{ranks} < n_{slices}$ ($4\times n_{slices}$ currently in HiPACE++)
- $n_{ranks} \leq n_{steps}$



Parallelization: the new (old?) pipeline algorithm

Transverse parallelization:
- None on GPU
- OpenMP on CPU

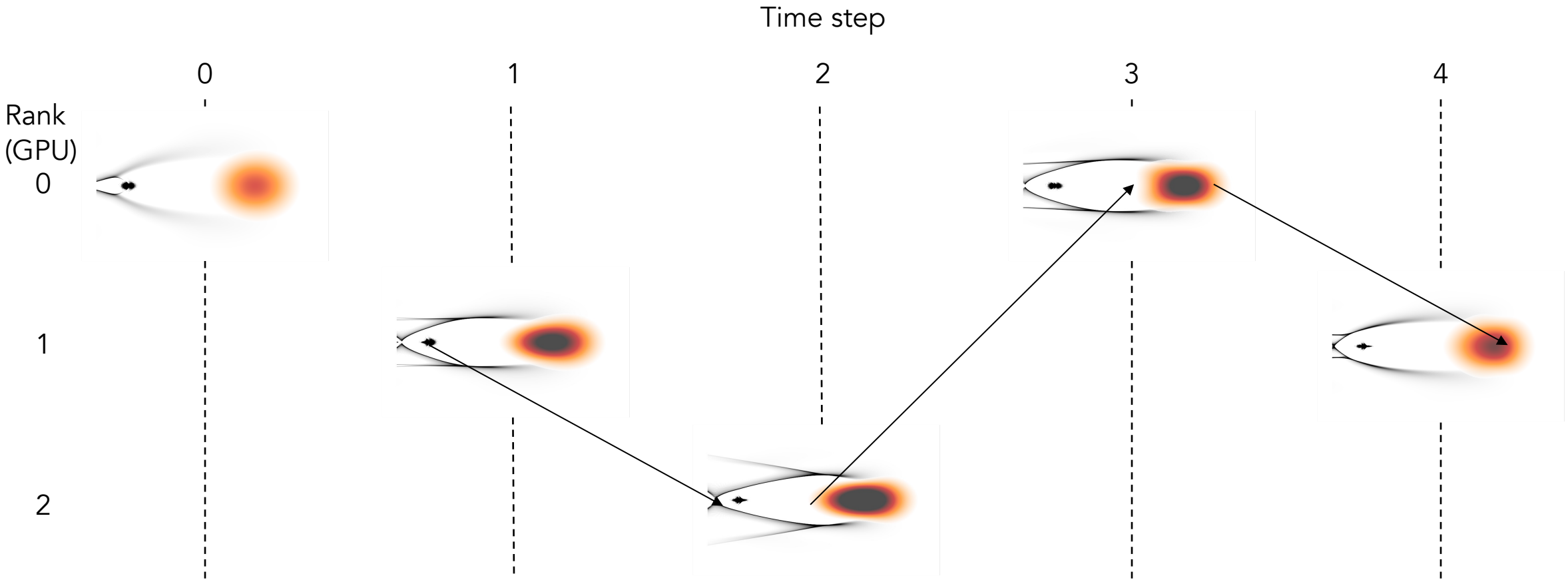
Longitudinal parallelization: MPI (Message Passing Interface)
- $n_{ranks} < n_{slices}$ ($4 \times n_{slices}$ currently in HiPACE++)
- $n_{ranks} \leq n_{steps}$



Parallelization: the new (old?) pipeline algorithm

Transverse parallelization:
- None on GPU
- OpenMP on CPU

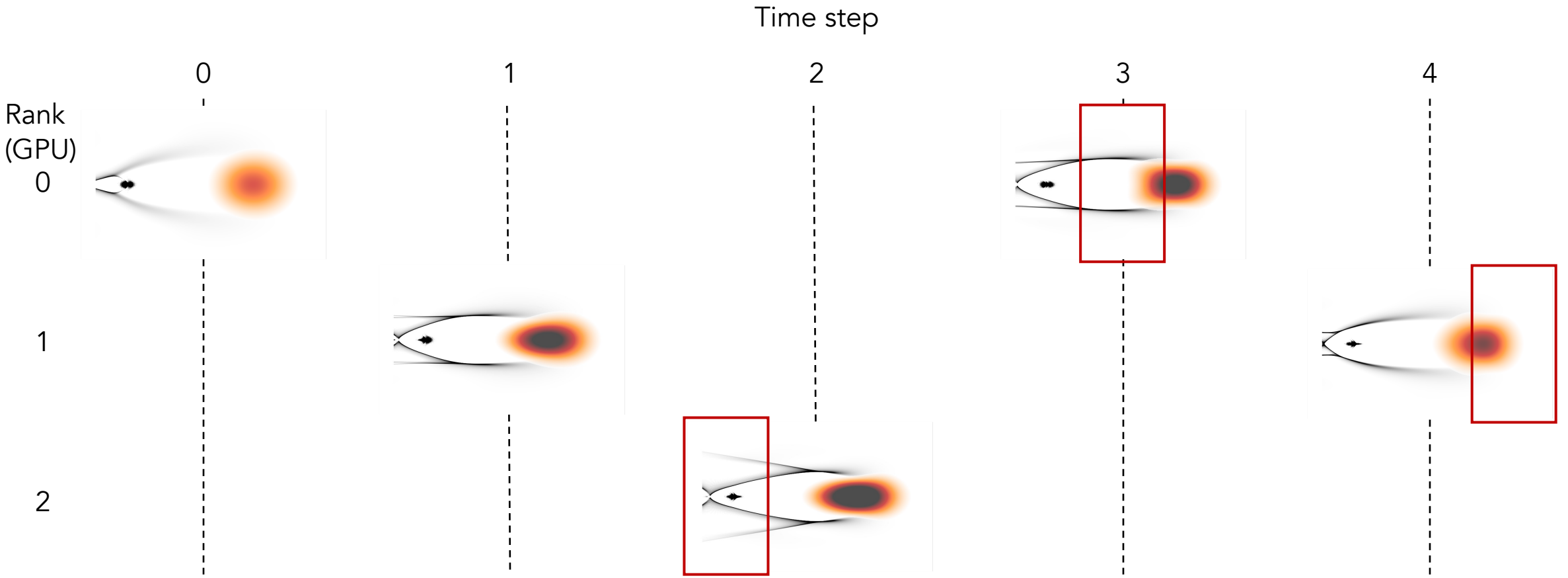
Longitudinal parallelization: MPI (Message Passing Interface)
- $n_{ranks} < n_{slices}$ ($4 \times n_{slices}$ currently in HiPACE++)
- $n_{ranks} \leq n_{steps}$



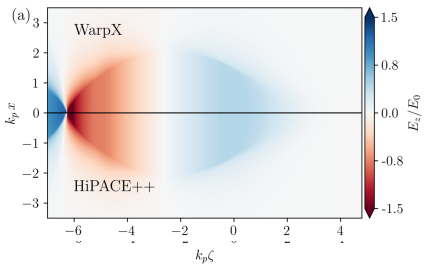
Parallelization: the new (old?) pipeline algorithm

Transverse parallelization:
- None on GPU
- OpenMP on CPU

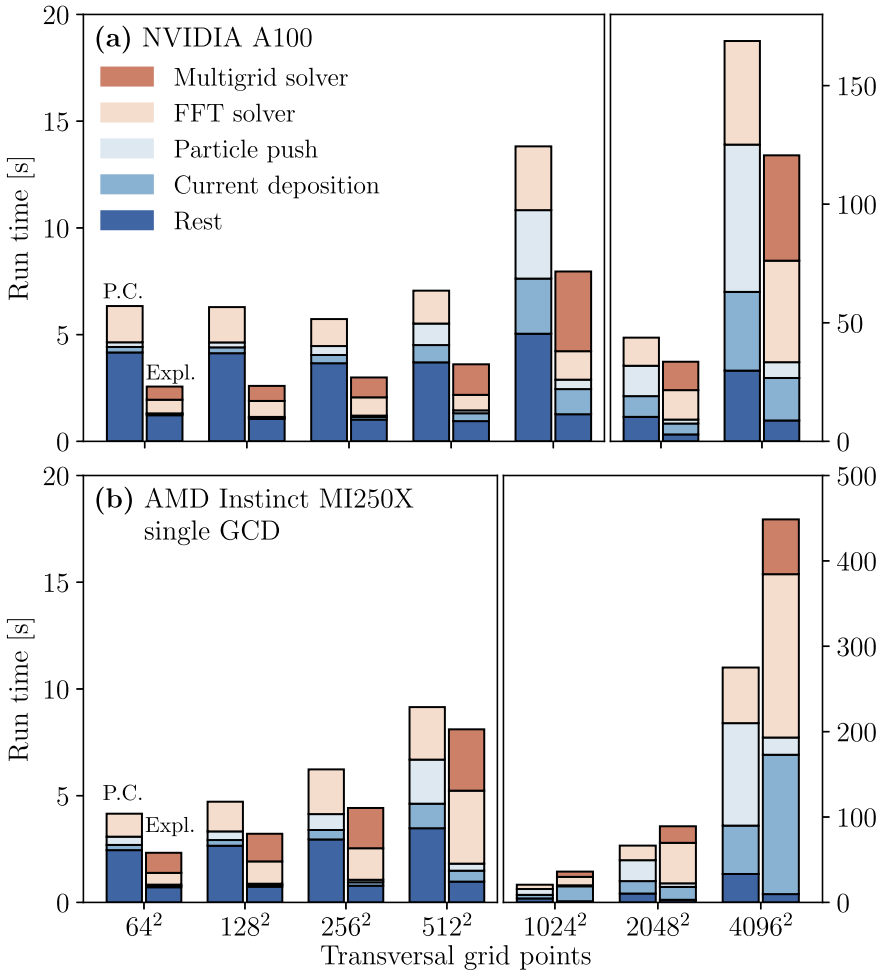
Longitudinal parallelization: MPI (Message Passing Interface)
- $n_{ranks} < n_{slices}$ ($4 \times n_{slices}$ currently in HiPACE++)
- $n_{ranks} \leq n_{steps}$

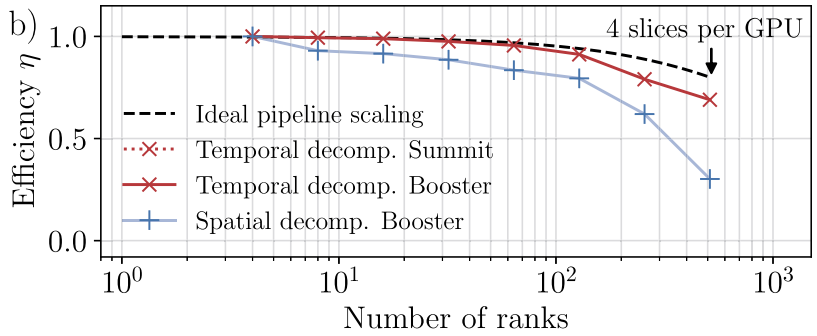


Performance of the code & parallelization



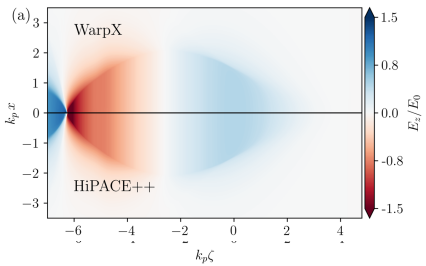
- S. Diederichs et al., *Comput. Phys. Comm.* 278: 108421 (2022)



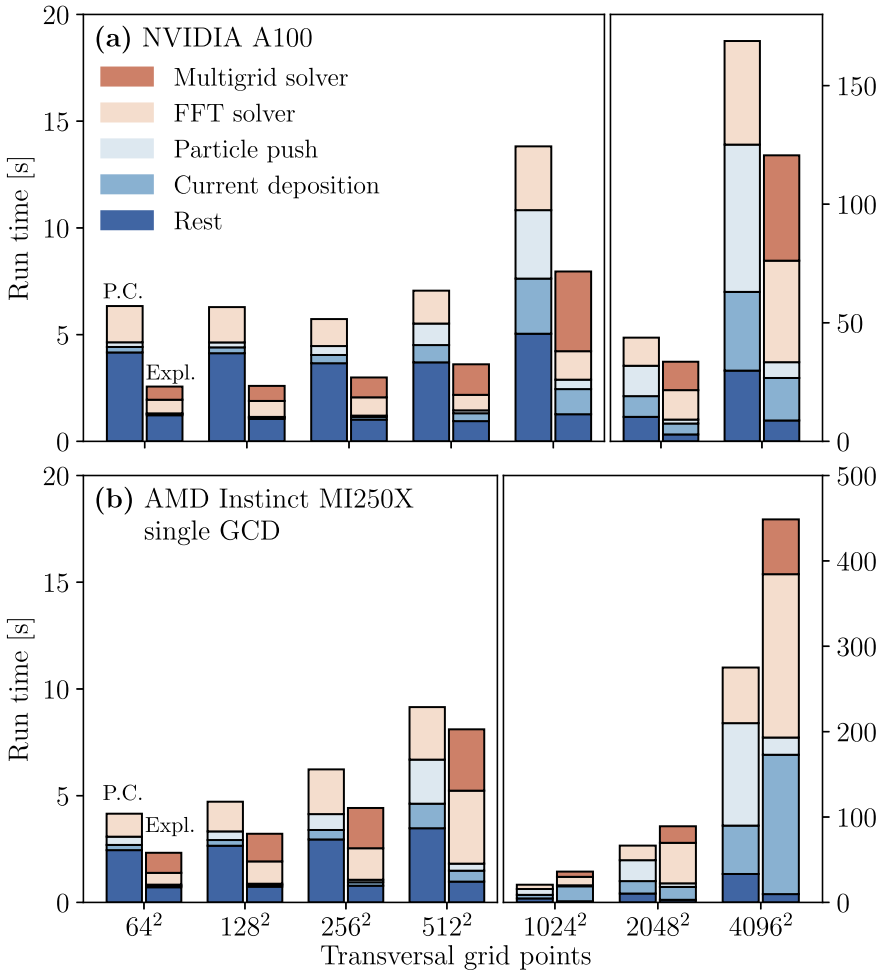


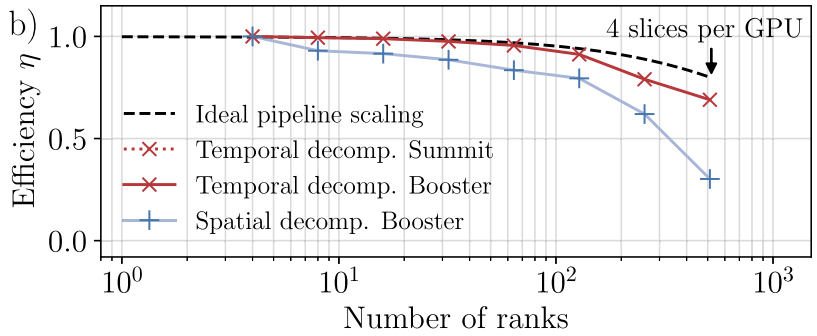
- Good scaling, good performance and AMD and NVIDIA GPUs
- Relatively old tests, significant speedup since then

Performance of the code & parallelization



- S. Diederichs et al., *Comput. Phys. Comm.* 278: 108421 (2022)





- Good scaling, good performance and AMD and NVIDIA GPUs
- Relatively old tests, significant speedup since then

See presentation by Alex Sinn (#5)

Code usage (illustration)

```
// Provided proper environment

> git clone https://github.com/Hi-PACE/hipace.git

> git checkout v23.05

> cmake -S . -B build -DHiPACE_COMPUTE=CUDA

> cmake --build build -j 16

> ./build/bin/hipace inputs

> mpirun -np 4 ./build/bin/hipace inputs
```

https://hipace.readthedocs.io/en/latest/run/get_started.html

inputs

```
max_step = 300
amr.n_cell = 1024 1024 1024
amr.max_level = 0
hipace.max_time = 0.3/clight
diagnostic.output_period = 1
hipace.dt = adaptive

geometry.is_periodic = true  true  false
geometry.prob_lo     = -250.e-6 -250.e-6 -250.e-6
geometry.prob_hi     =  250.e-6  250.e-6  110.e-6

beams.names = driver
driver.position_mean = 0. 0. 0.
driver.position_std = 2.e-6 2.e-6 30.e-6
driver.injection_type = fixed_weight
driver.num_particles = 1000000
driver.total_charge = .6e-9
driver.u_mean = 0. 0. 1000.
driver.u_std = 2. 2. 10.
driver.do_symmetrize = 1

plasmas.names = electron
electron.density(x,y,z) = 2.e22
electron.ppc = 1 1
electron.u_mean = 0.0 0.0 0.
electron.element = electron

diagnostic.diag_type = xz
```

Code usage (illustration)

```
// Provided proper environment

> git clone https://github.com/Hi-PACE/hipace.git

> git checkout v23.05

> cmake -S . -B build -DHiPACE_COMPUTE=CUDA

> cmake --build build -j 16

> ./build/bin/hipace inputs

> mpirun -np 4 ./build/bin/hipace inputs
```

https://hipace.readthedocs.io/en/latest/run/get_started.html

See presentation by Severin Diederichs (#4)

inputs

```
max_step = 300
amr.n_cell = 1024 1024 1024
amr.max_level = 0
hipace.max_time = 0.3/clight
diagnostic.output_period = 1
hipace.dt = adaptive

geometry.is_periodic = true  true  false
geometry.prob_lo     = -250.e-6 -250.e-6 -250.e-6
geometry.prob_hi     =  250.e-6  250.e-6  110.e-6

beams.names = driver
driver.position_mean = 0. 0. 0.
driver.position_std = 2.e-6 2.e-6 30.e-6
driver.injection_type = fixed_weight
driver.num_particles = 1000000
driver.total_charge = .6e-9
driver.u_mean = 0. 0. 1000.
driver.u_std = 2. 2. 10.
driver.do_symmetrize = 1

plasmas.names = electron
electron.density(x,y,z) = 2.e22
electron.ppc = 1 1
electron.u_mean = 0.0 0.0 0.
electron.element = electron

diagnostic.diag_type = xz
```

Conclusion

- HiPACE++: 3D, QS PIC for plasma acceleration
- Advanced methods (pipeline, MG)
- HPC (GPU) computing
- Open-source, documented
- New contributors are welcome!

Perspective

- More physics added
- Goals:
  - multi-stage collider-relevant parameters
  - Target new physics problems
- Advanced algorithms (MR)
- Enjoy the workshop!

Thank you for your attention

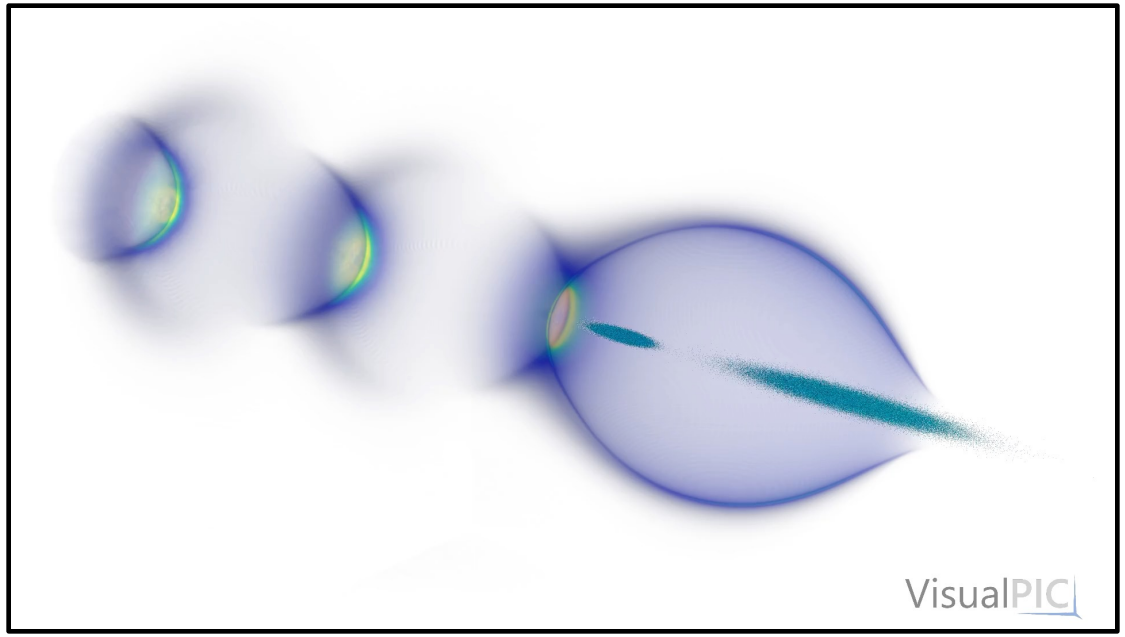
Contributions from all HiPACE++ contributors

Conclusion

- HiPACE++: 3D, QS PIC for plasma acceleration
- Advanced methods (pipeline, MG)
- HPC (GPU) computing
- Open-source, documented
- New contributors are welcome!

Perspective

- More physics added
- Goals:
  - multi-stage collider-relevant parameters
  - Target new physics problems
- Advanced algorithms (MR)
- Enjoy the workshop!



Thank you for your attention

Contributions from all HiPACE++ contributors