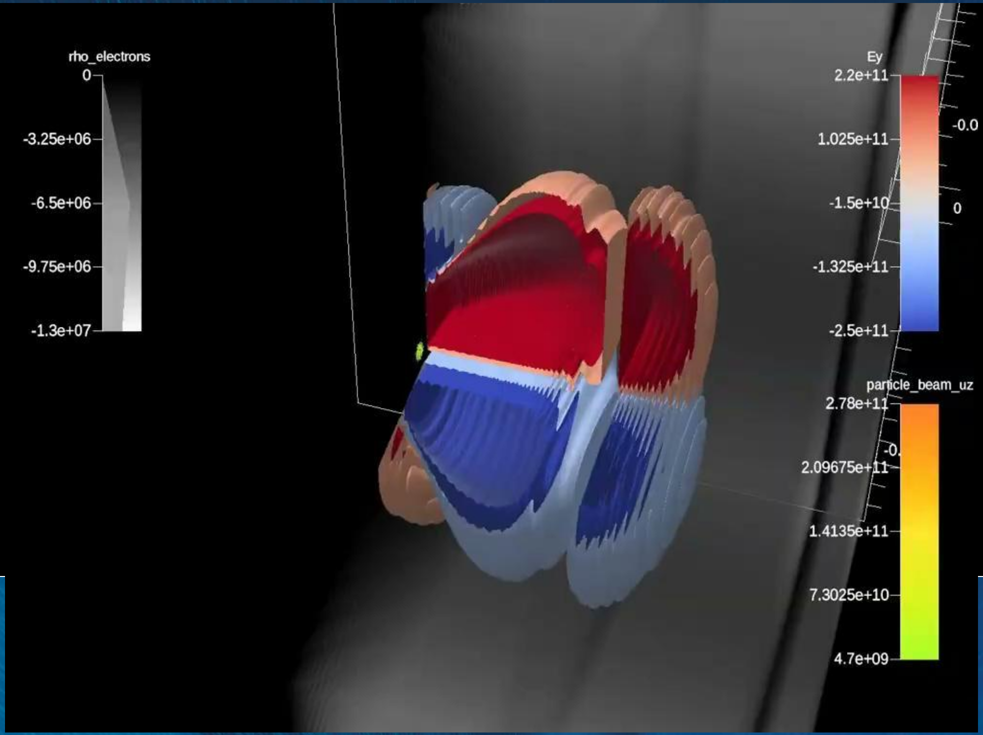
Modelization of Plasma Accelerators in the Exascale Era

Axel Huebl for the BLAST team and collaborators at
LBNL, LLNL, CEA-LIDYL, SLAC, DESY, CERN, CASUS/HZDR



multi-stage LPA simulation in a
boosted frame with WarpX
transversely focusing fields & beam

Plenary presentation
7th European Advanced
Accelerator Conference (EAAC)

Thursday, Sep 25th, 2025
Isola d'Elba, Italy

BERKELEY LAB

Advanced Modeling Program
ACCELERATOR TECHNOLOGY & APPLIED PHYSICS DIVISION ATAP

U.S. DEPARTMENT OF ENERGY | Office of Science

Abstract

Plasma accelerators have demonstrated significant milestones, from producing 10 GeV electron beams in wakefield acceleration, high-gain free-electron laser operation, energy boosting of electrons, to reaching stable (ultra-short, nC-class) proton acceleration that enable studies of ultrahigh dose-rate radiobiology. Now, the community is setting sight on **integrating plasma acceleration** deep into **future particle colliders and applications**, such as a potential 10 TeV center-of-mass collider, Higgs factory, injection into rings for next-generation light sources, stable high-repetition rate operations, among others, which continue to set demanding research challenges on particle beam quality, repetition rate and reliability.

This presentation will discuss the **current capabilities and latest trends** in modeling **plasma accelerators** and integrated modeling of **beamlines with plasma elements**. With a need for detailed kinetic modeling from design to operations, a comprehensive and **coordinated approach** is needed to cover and optimize anything from the source to the end of the beam's lifetime. An important enabler are new **technologies from Exascale Computing**, providing (GPU) accelerated computing for accelerator and plasma physicists from laptops to supercomputers. Advances in **open source modeling ecosystems** and coupling to **AI/ML with standardized data exchange** now enable **user-friendly model-building** for integrated accelerators, combining theory, kinetic modeling and fast surrogate models.

U.S. DEPARTMENT OF ENERGY | Office of Science

Modelization of Plasma Accelerators in the Exascale Era

- **Community Modeling with BLAST**
  - The Beam, Plasma & Accelerator Simulation Toolkit (BLAST)
  - Engines for accelerator start-to-end modeling
  - Building a community ecosystem
  - Standardization & Interoperability

- **Exascale Technologies for Particle Accelerator Modeling**
  - Industry trends and opportunities
  - Accelerating day-to-day modeling: from laptops to supercomputers
  - Exascale Modeling examples in plasma acceleration

- **Connecting Scales & Data with Machine-Learning Surrogates**
  - Building models from wakefield simulation *data*
  - Connecting experiments & simulations
  - Combining with differentiable modeling to solve hard, inverse problems

Community Modeling with BLAST

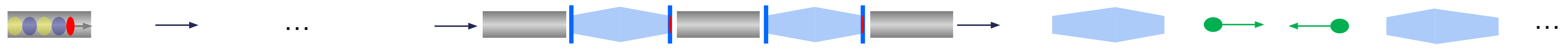
WarpX open governance

Developed by an international, multidisciplinary team

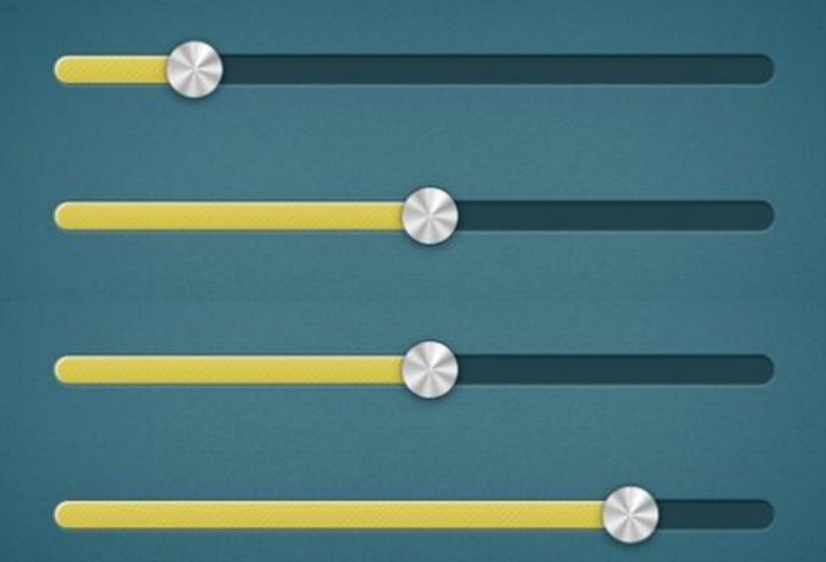
*former members

***over 110* contributors, incl. from the private sector**

There Are Many Choices to Plasma Accelerator Modelization



Speed			Fidelity	
Fast & as accurate as possible	Reduced physics		Full physics	**Accurate** & as fast as possible
	Reduced models		First principles	
	1D-1V		3D-3V	
	Low resolution		High resolution	



e.g., initial designs, optimization & operations

e.g., stability proofs, exploration, ML training data

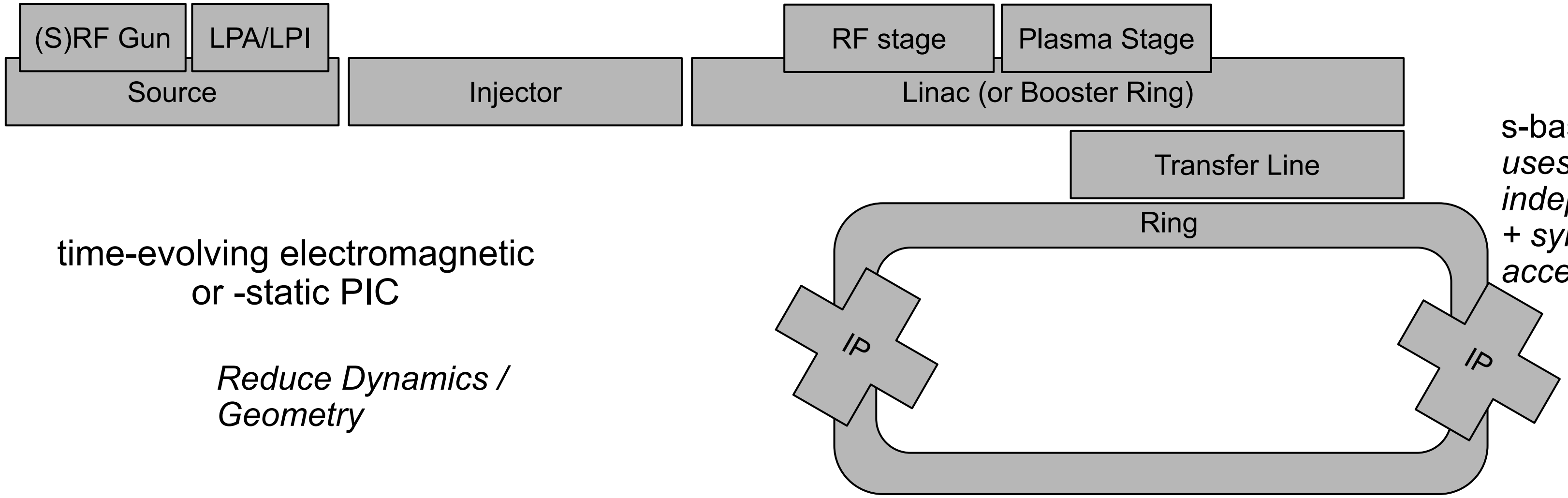
e.g., RZ geometry, quasi- and electro-static approximation, fluid background, ML data surrogate

This requires an **ecosystem of models**
⇒ share models & data between codes
⇒ works best when standardized

BLAST is a Comprehensive Simulation Toolkit for Accelerator Physics

BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT

Imagine a future, ***hybrid*** **particle accelerator**, e.g., with RF and plasma elements.



s-based PIC
uses s instead of t as independent variable + symplectic maps for accelerator elements

time-evolving electromagnetic or -static PIC

Reduce Dynamics / Geometry

Quasistatic PIC
separates timescale: plasma wake & beam evol.

buildup of electron clouds, secondary electron yield

modeling of radiative & space-charge effects

A Friedman et al., Part. Accel. (1992)
DP Grote et al., NIMA (1998)
J Qiang et al., PRSTAB (2006)
J-L Vay et al. CSD (2013)
A Huebl et al. (2015)
R Lehe et al., CPC (2016)
J-L Vay et al., NIMA (2018)
A Ferran Pousa et al., JPConf. (2019)
S Diederichs et al., CPC (2022)
A Huebl et al., NAPAC22 and AAC22 (2022)
A Ferran Pousa et al., PRAB (2023)
M Thévenet et al., EAAC23 (2023)
O Shapoval et al. PRE (2024)
Sandberg et al. PASC24 (2025)
R Lehe et al. PASC25 (2025)
J-L Vay et al. PRE (2025)

Goal
Start-to-end modeling in an open software ecosystem.

BLAST is a Comprehensive Simulation Toolkit for Accelerator Physics

BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT

Imagine a future, ***hybrid*** **particle accelerator**, e.g., with RF and plasma elements.

single codes

2014

(S)RF Gun | LPA/LPI

Source

Injector

RF stage | Plasma Stage

Linac (or Booster Ring)

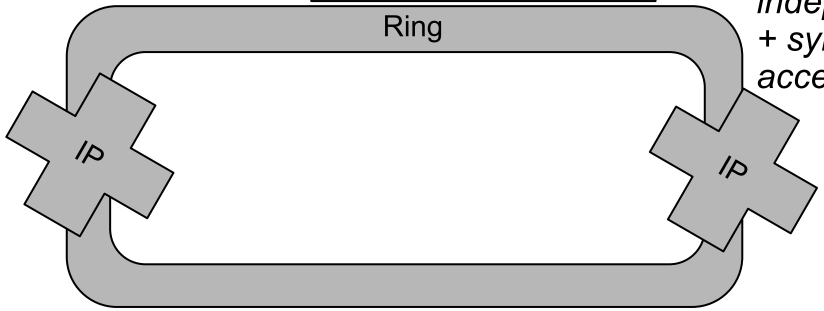
Transfer Line

Warp

IMPACT-T

time-evolving electromagnetic or -static PIC

Beam Beam 3D



IMPACT-Z

s-based PIC
uses s instead of t as independent variable + symplectic maps for accelerator elements

Codes

BLAST
CPU-only

Reduce Dynamics / Geometry

Quasistatic PIC
separates timescale: plasma wake & beam evol.

buildup of electron clouds, secondary electron yield

modeling of radiative & space-charge effects

A Friedman et al., Part. Accel. (1992)
DP Grote et al., NIMA (1998)
J Qiang et al., PRSTAB (2006)
J-L Vay et al. CSD (2013)
A Huebl et al. (2015)
R Lehe et al., CPC (2016)
J-L Vay et al., NIMA (2018)
A Ferran Pousa et al., JPConf. (2019)
S Diederichs et al., CPC (2022)
A Huebl et al., NAPAC22 and AAC22 (2022)
A Ferran Pousa et al., PRAB (2023)
M Thévenet et al., EAAC23 (2023)
O Shapoval et al. PRE (2024)
Sandberg et al. PASC24 (2025)
R Lehe et al. PASC25 (2025)
J-L Vay et al. PRE (2025)

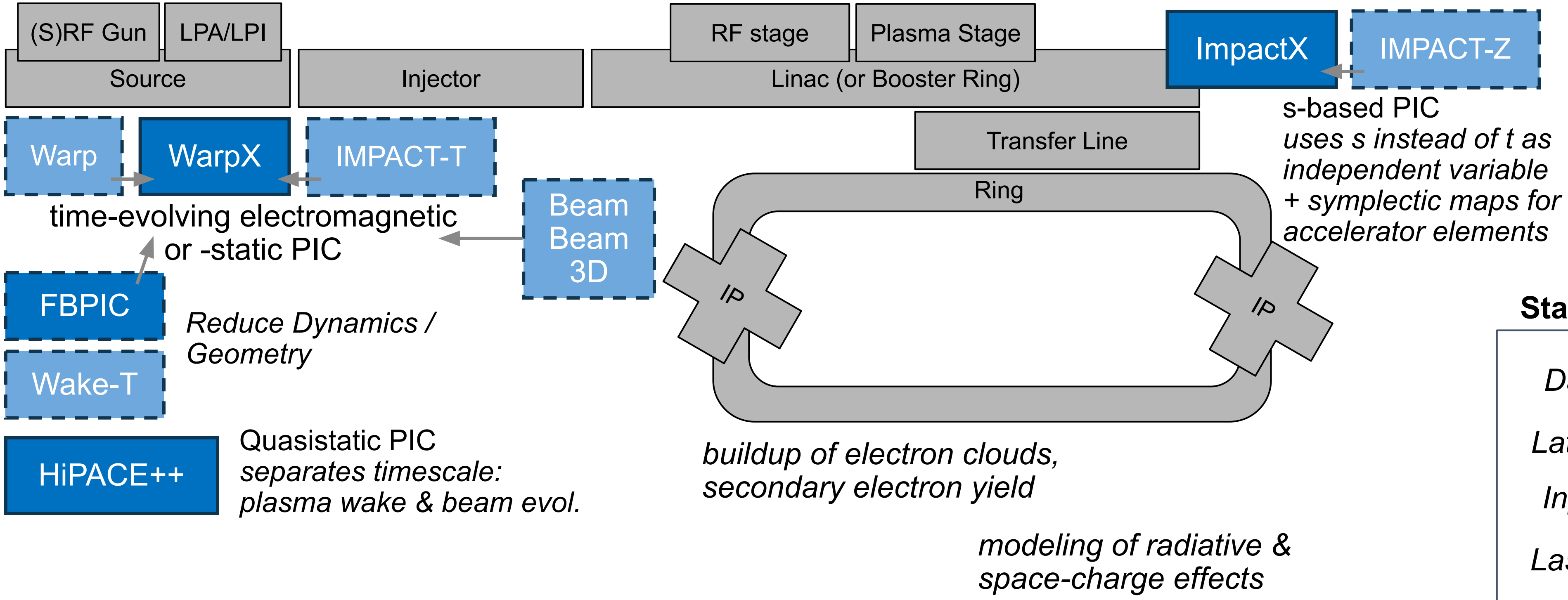
Goal
Start-to-end modeling in an open software ecosystem.

BLAST is a Comprehensive Simulation Toolkit for Accelerator Physics

BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT

Imagine a future, ***hybrid* particle accelerator**, e.g., with RF and plasma elements.

single codes
2014
2025
integrated ecosystem



Codes

BLAST CPU-only

BLAST CPU & GPU

Integrated through Standards & Workflows

Data	openPMD
Lattice	**PALS**
Input	PICMI Standard
Lasers	LASY
Optimize	optimas
AI/ML	

A Friedman et al., Part. Accel. (1992)
DP Grote et al., NIMA (1998)
J Qiang et al., PRSTAB (2006)
J-L Vay et al. CSD (2013)
A Huebl et al. (2015)
R Lehe et al., CPC (2016)
J-L Vay et al., NIMA (2018)
A Ferran Pousa et al., JPConf. (2019)
S Diederichs et al., CPC (2022)
A Huebl et al., NAPAC22 and AAC22 (2022)
A Ferran Pousa et al., PRAB (2023)
M Thévenet et al., EAAC23 (2023)
O Shapoval et al. PRE (2024)
Sandberg et al. PASC24 (2025)
R Lehe et al. PASC25 (2025)
J-L Vay et al. PRE (2025)

Goal
Start-to-end modeling in an open software ecosystem.

Standardization & Interoperability Can Provide Productivity, Reproducibility and are Enablers for ML

Particle-In-Cell Modeling Interface

PICMI Standard

→

Code A

Code B

...

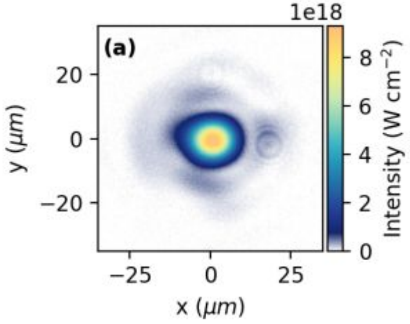
→

open Particle Mesh Data standard

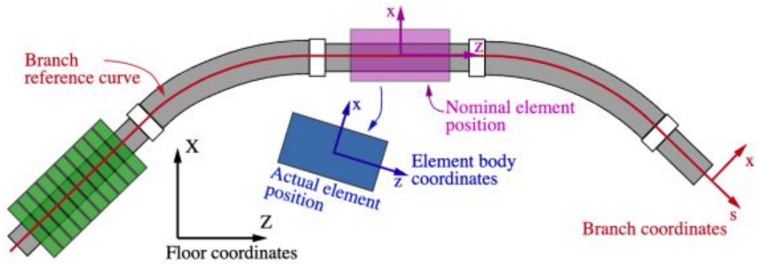
openPMD

HZDR Helmholtz Zentrum Dresden Rossendorf · Berkeley Lab · Technische Universität Dresden · EUCALL · Lawrence Livermore National Laboratory · SLAC National Accelerator Laboratory · DESY · UH · European XFEL · UCLA · IOQ Jena Friedrich-Schiller-Universität · CBG Max Planck Institute of Molecular Cell Biology and Genetics · Fermilab · Oak Ridge National Laboratory · plasmapy · Maison de la Simulation

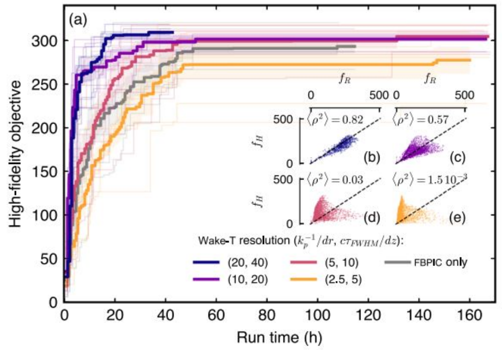
LASY



Particle Accelerator Lattice Standard (PALS)



optimas



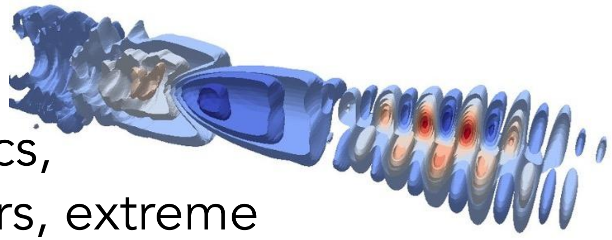
Fermilab · SLAC National Accelerator Laboratory · Berkeley Lab · CAMPA · UCLA · Argonne National Laboratory · Oak Ridge National Laboratory

A Huebl et al., DOI:10.5281/zenodo.591699 (2015); DP Grote et al., *Particle-In-Cell Modeling Interface (PICMI)* (2021); LD Amorim et al., *GPos* (2021)
M Thévenet et al., EAAC23, arXiv:2403.12191 (2023); A Ferran Pousa et al., DOI:10.5281/zenodo.7989119 (2023); RT Sandberg et al., IPAC23, DOI:10.18429/JACoW-IPAC-23-WEPA101 (2023); C Mitchell et al., *A Community Effort Toward a Particle Accelerator Lattice Standard (PALS)*, TUP004 in NAPAC25 (2025)

WarpX: a Community PIC Code on Exascale Technology

BLAST — BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT

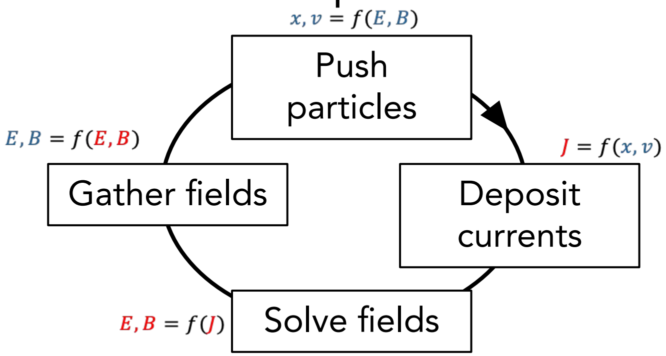
Applications

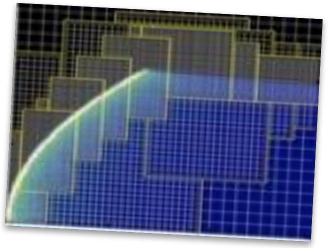


laser-plasma physics,
particle accelerators, extreme
light sources, fusion devices & plasmas, …

Particle-in-Cell

- electromagnetic or electro/magnetostatic
- 1-3D, RZ+, spherical
- time integration: explicit, implicit





International Contributors incl. private sector

BERKELEY LAB, cea PARIS-SACLAY, DESY., loa, CERN, MODERN ELECTRON, tae TECHNOLOGIES, UR LLE, SLAC, AVALANCHE

Award–Winning Code & Science

Detailed Physical Models

- Full documentation, benchmarks, examples
- Easy-to-use boosted frame
- collisional, atomic & fusion processes
- PIC-fluid hybrid, *and much more*

Portable, Multi-Level Parallelization

- GPUs & CPUs
- Desktop to supercomputer

Scalable & Standardized

- Python APIs, openPMD data
- In situ processing
- Open community ecosystem

openPMD, LASY, PICMI Standard

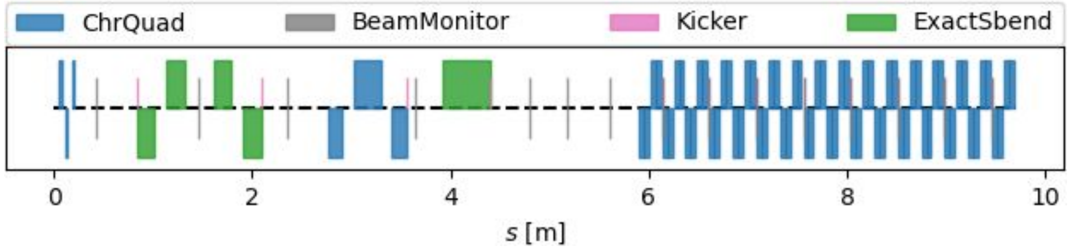
J-L Vay et al., NIMA 909.12 (2018)
L Fedeli, A Huebl et al., SC22, DOI:10.1109/SC41404.2022.00008 (2022)

ImpactX Leverages WarpX to Model *Whole Beamlines*

BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT

Applications

Beam-dynamics in transport lines, Linacs, Rings, Colliders, Final Focus (BDS), *e.g.,*



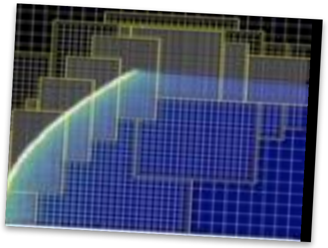
LBNL BELLA Hundred-Terawatt Undulator (HTU)

Electrostatic Particle-in-Cell

evolve beam relative to a *reference particle*

- particle advance: symplectic maps
- collective effects: **space charge**, CSR, ISR
- also: rapid envelope tracking

efficient modeling of large scales (e.g. km) for full beamlines



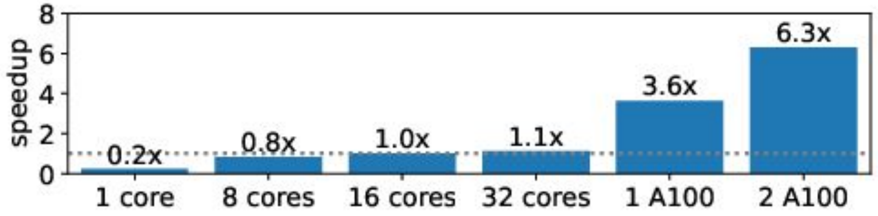
A Huebl, C Mitchell et al., NAPAC22 and AAC22 (2022) and NAPAC25 (2025)
C Mitchell et al., HB2023, THBP44 and TUA2I2 (2023)
J Qiang et al., PRSTAB (2006); RD Ryne et al., ICAP2006 ICAP2006 (2006)

Selected, Recent Features

- exchange beams w/ wakefield sims (openPMD)
- *new:* ML surrogate models
- *new:* static plasma lenses (tapered)

Portable, Multi-Level Parallelization

- GPUs & GPUs
- Desktop to supercomputer

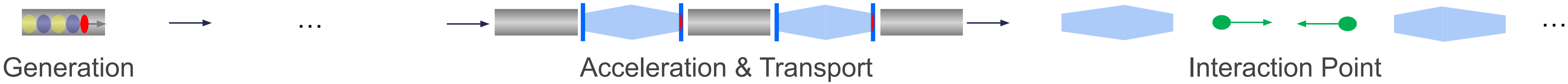


User-friendly

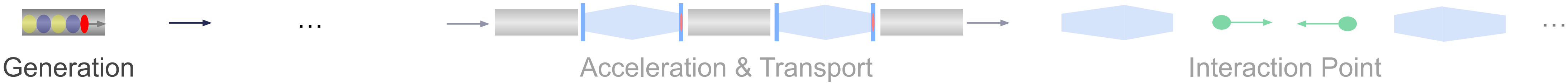
- Python API, openPMD data
- In situ processing
- Open community ecosystem

preview: lattices from **PALS**

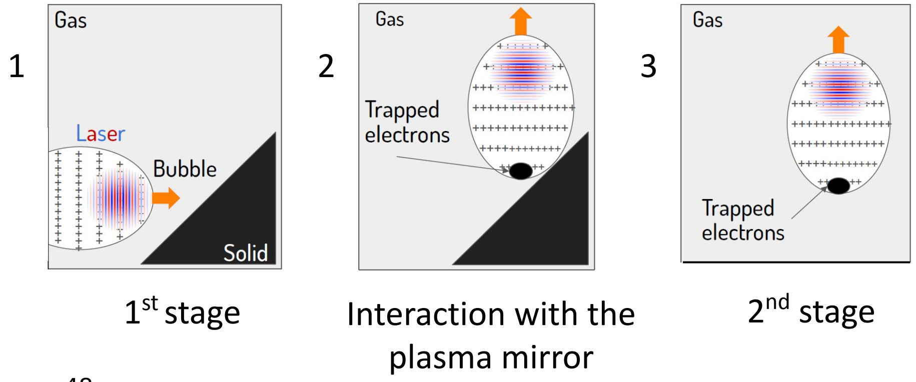
BLAST Codes Cover Wakefield Collider Modeling from Source to Interaction Point

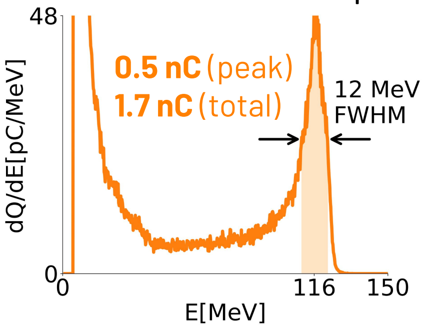


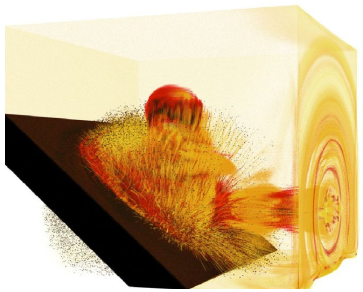
Detailed Modeling of Injection Physics



Two-stage injection+acceleration with a plasma mirror



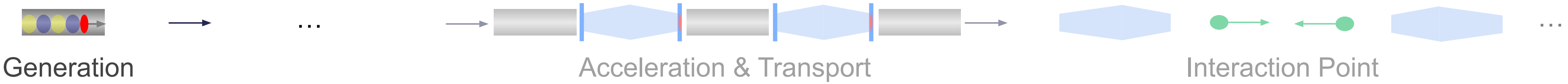




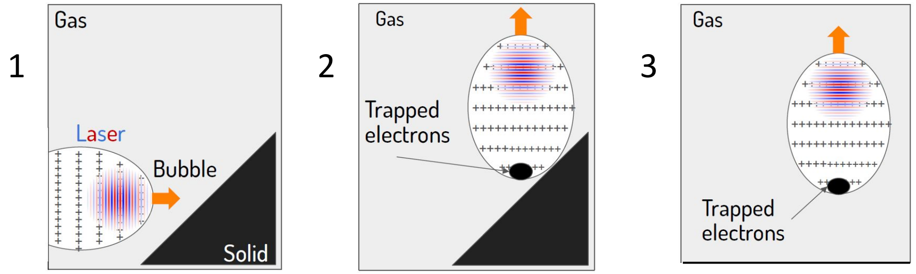
L Fedeli, A Huebl et al., SC22, **ACM Gordon Bell Prize for WarpX** (2022)

M. Thévenet et al., Nat. Phys., 12.4 (2016)

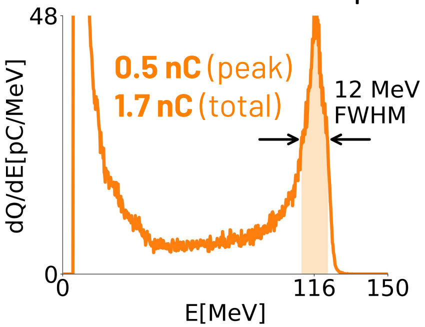
Detailed Modeling of Injection Physics

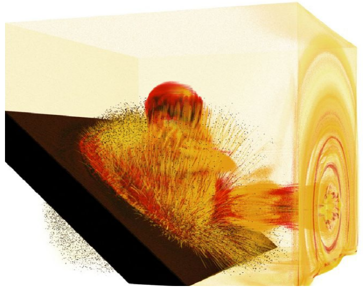


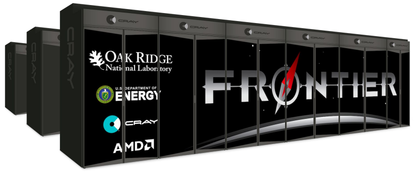
Two-stage injection+acceleration with a plasma mirror



1st stage — Interaction with the plasma mirror — 2nd stage

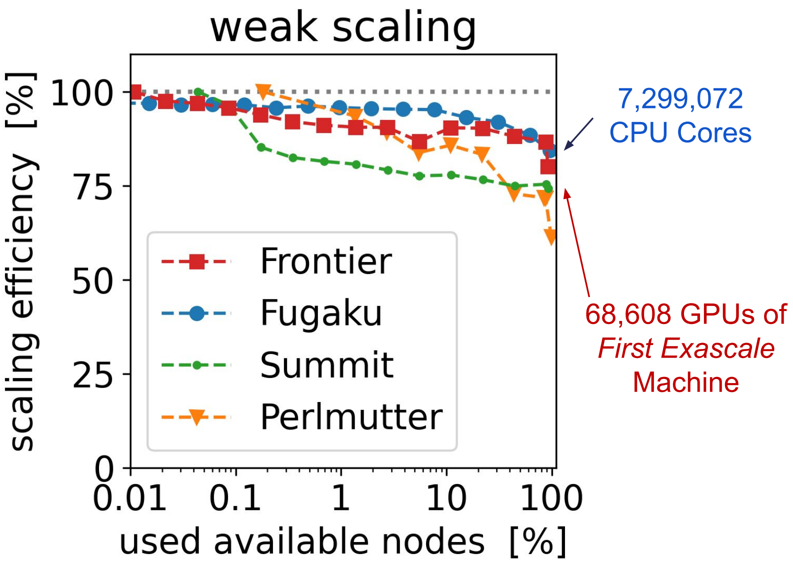






Computers:

- 69K GPUs on Frontier (OLCF)
- 7.3M CPU cores on Fugaku (RIKEN)



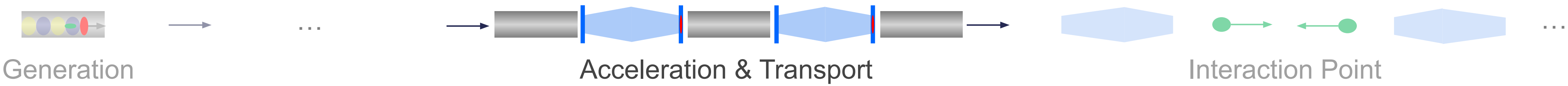
cea PARIS-SACLAY · BERKELEY LAB · cnrs · GENCI · RIKEN · arm · AtoS

A success story of a multidisciplinary, multi-institutional team!

L Fedeli, A Huebl et al., SC22, **ACM Gordon Bell Prize for WarpX** (2022)

M. Thévenet et al., Nat. Phys., 12.4 (2016)

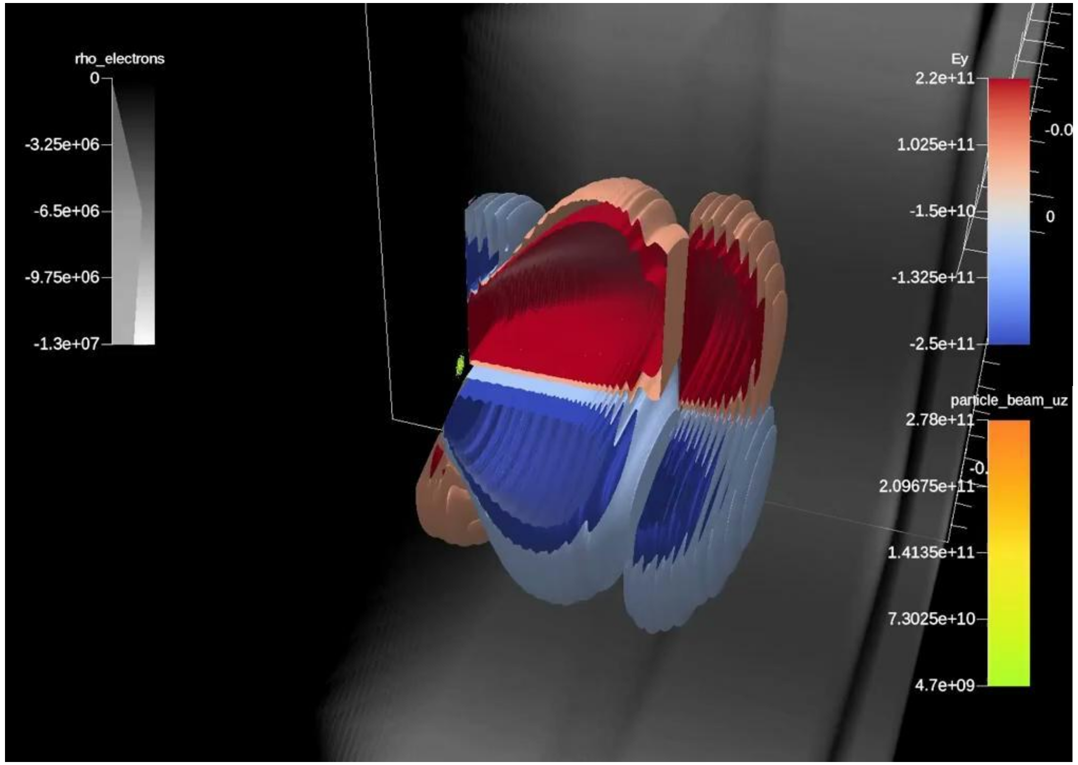
Optimization and 3D Verification of Staging



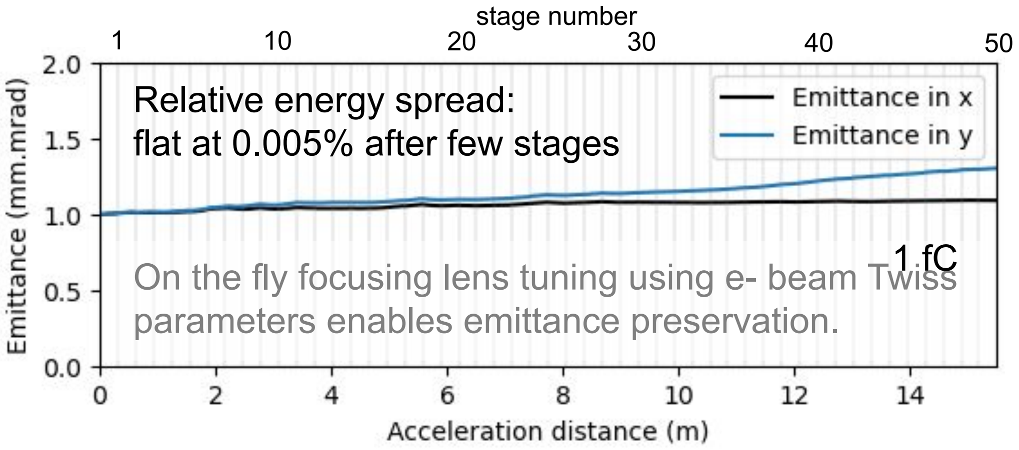
50 Multi-GeV LPA Stages in 3D

Work by our team at LBNL

In Situ Visualization of the first 15 stages:



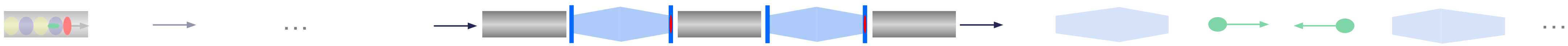
Computer: 256 GPUs for 8h on Perlmutter (NERSC)



- Plasma channels: 28cm, 3cm gaps
- linear thick lens (3 mm)
- negligible beam charge

J-L Vay et al., PoP 28.2, 023105 (2021)
WarpX ECP MS FY23.1 & FY23.2 (2023); T Barklow et al., JINST (2023)
A Ferran Pousa et al., IPAC23, *TUPA093 & PRAB* (2023); CB Schroeder et al., JINST (2023)

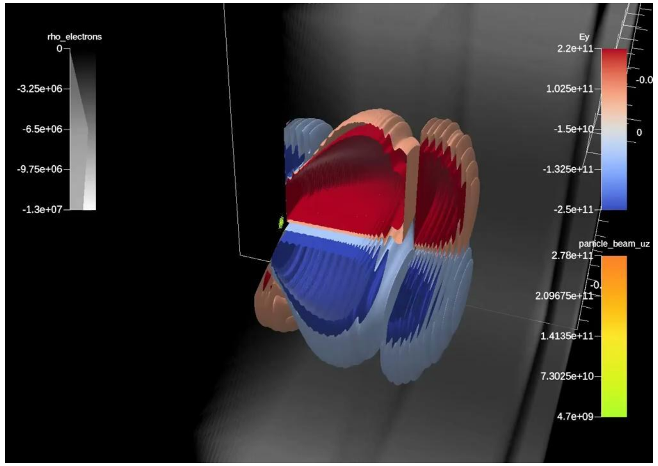
Optimization and 3D Verification of Staging

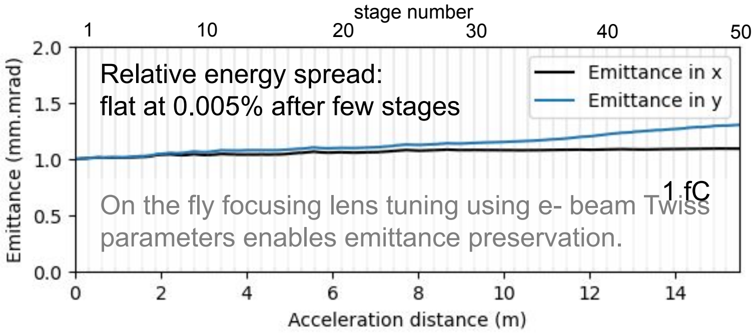


Generation | Acceleration & Transport | Interaction Point

50 Multi-GeV LPA Stages in 3D

In Situ Visualization of the first 15 stages:





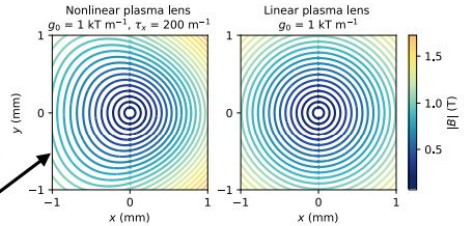
Novel Chromatic Staging Optics

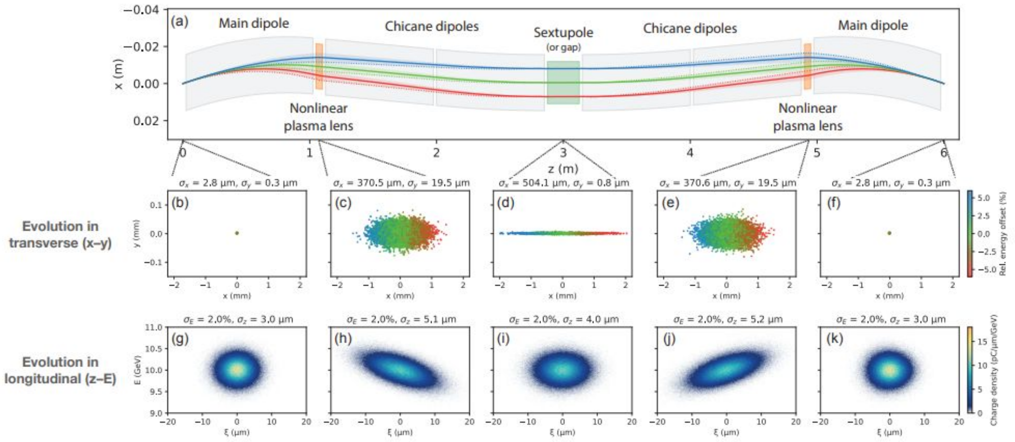
Work by Carl Lindstrøm et al.

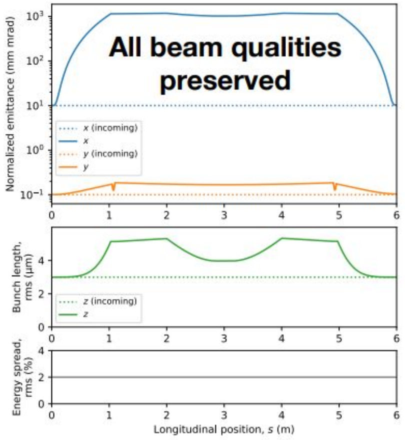
HiPACE++ | ImpactX

Local chromaticity correction and a new plasma lens

- Inspiration: chromaticity correction in collider final focusing
  - *Disperse, apply stronger focusing for higher energies (+ vice versa)*
- Made compact and simple using a **nonlinear plasma lens**



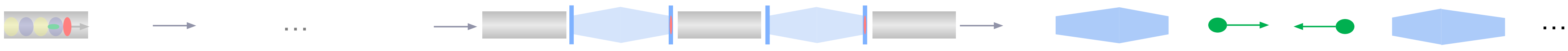




J-L Vay et al., PoP 28.2, 023105 (2021)
WarpX ECP MS FY23.1 & FY23.2 (2023); T Barklow et al., JINST (2023)
A Ferran Pousa et al., IPAC23, *TUPA093 & PRAB* (2023); CB Schroeder et al., JINST (2023)

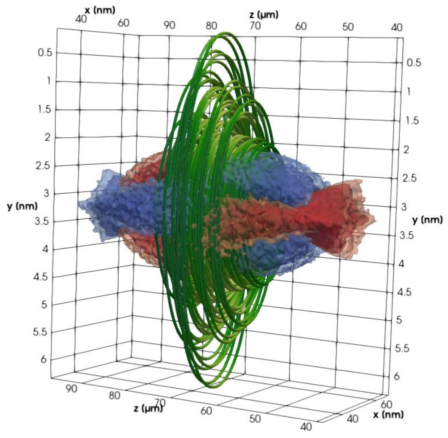
C. A. Lindstrøm et al., Chromatic optics for staging of plasma accelerators using nonlinear plasma lenses (manuscript in prep., EAAC25 talk on Mon)
B. Chen et al., ABEL: A Start-to-End Simulation and Optimisation Framework for Plasma-Based Accelerators and Colliders (EAAC25 Talk on Tue)

Beam-Beam Modeling at the Interaction Point



Work by Arianna Formenti et al.

10 TeV COM e^-e^+



WarpX can now simulate flat, spherical, round and asymmetric beams in **linear colliders**:
ILC, C³, wakefield, HALHF, …

and is exercised for & advanced towards **circular colliders:**
FCC-ee, Muons

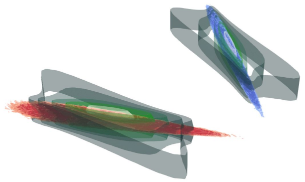
Many beam-beam effects

💔 disruption (beam-beam parameter)
🔦 photon emission
👯 e^+e^- pair creation
🎱 scattering
💪hadron photoproduction

💥 what are the actual **luminosities**?
☁️ what are the actual **backgrounds**?

New Capabilities Added

- spectral integrated Green function (IGF) solvers
- **luminosity diagnostics**: 1D as a function of E_{COM} and 2D as a function of Ene_1 & Ene_2
- **binary collisions** (linear Compton scattering, linear Breit Wheeler) and virtual photons
  - simulate *incoherent pair production* via Bethe-Heitler and Landau-Lifshitz processes
- **linear compton scattering** is used to simulate gamma-gamma colliders: electron-laser scattering

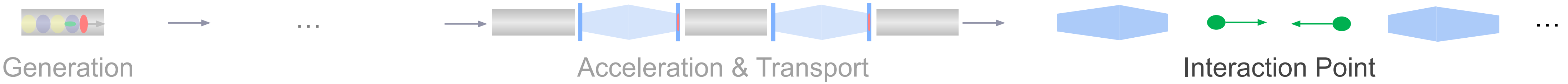


during collision: disrupted beams

Future Circular Collider ○

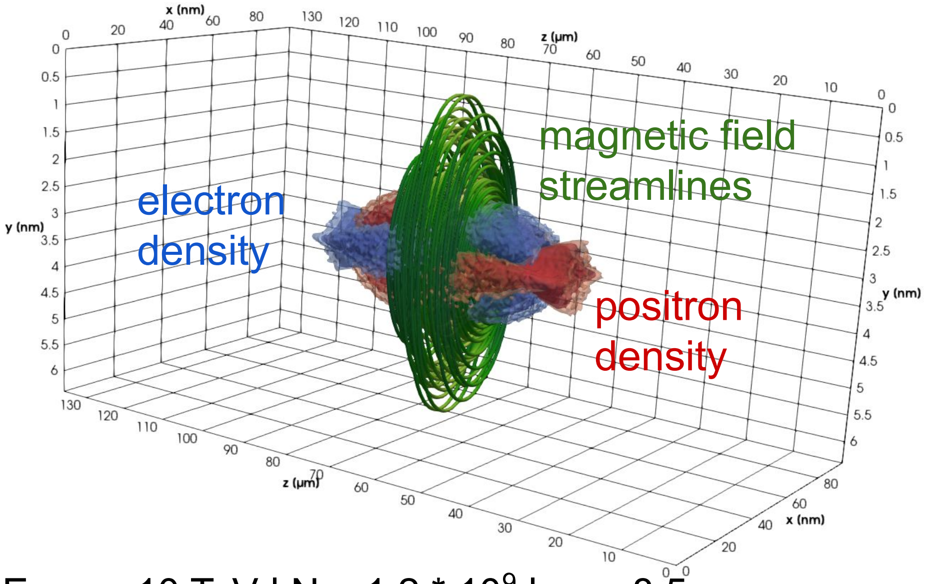
International Linear Collider /

Beam-Beam Modeling at the Interaction Point



Work by Arianna Formenti et al.

Preliminary simulations with wakefield lepton beams at 10TeV



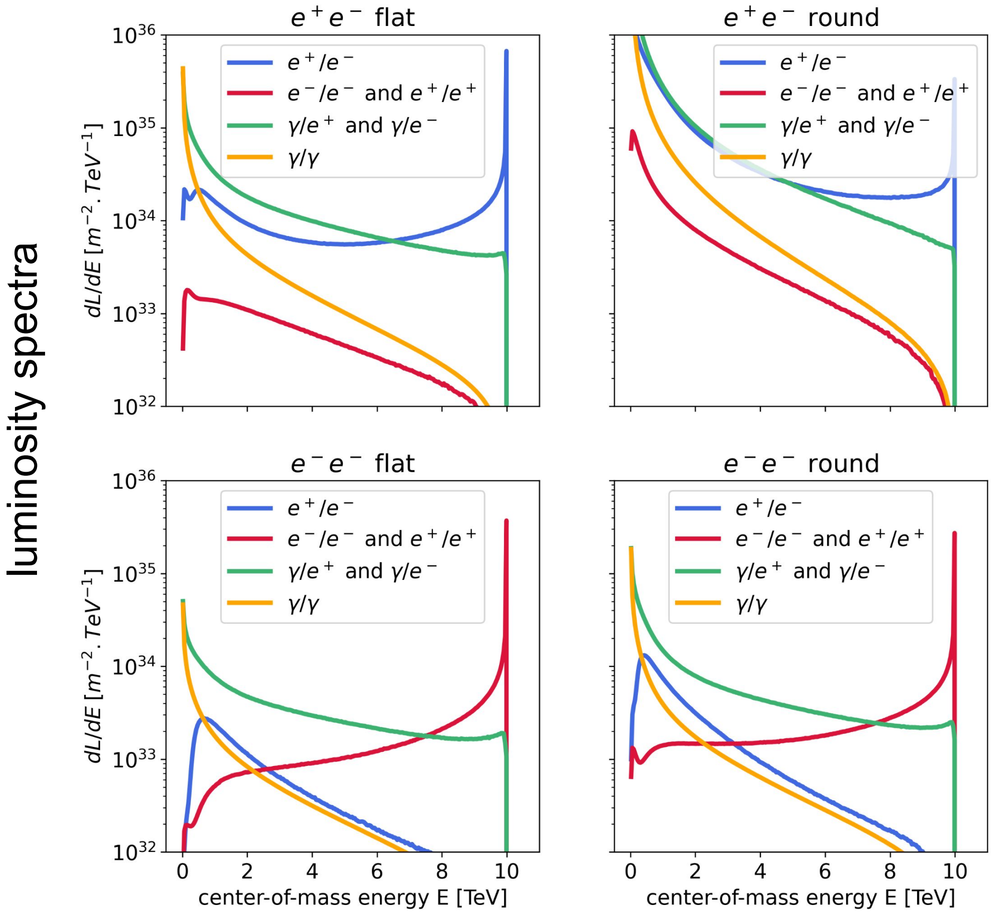
E_{COM} = 10 TeV | N = 1.2 * 10^9 | σ_z = 8.5 um

e^+e^- vs. e^-e^-

round: σ* = 1.55 nm | D = 1.22 | χ = 970

flat: σ^*_x = 6 nm | σ^*_y = 0.4 nm | D_x= 0.15 | D_y= 2.3 | χ = 470

→ **results used by particle and detector physicists**



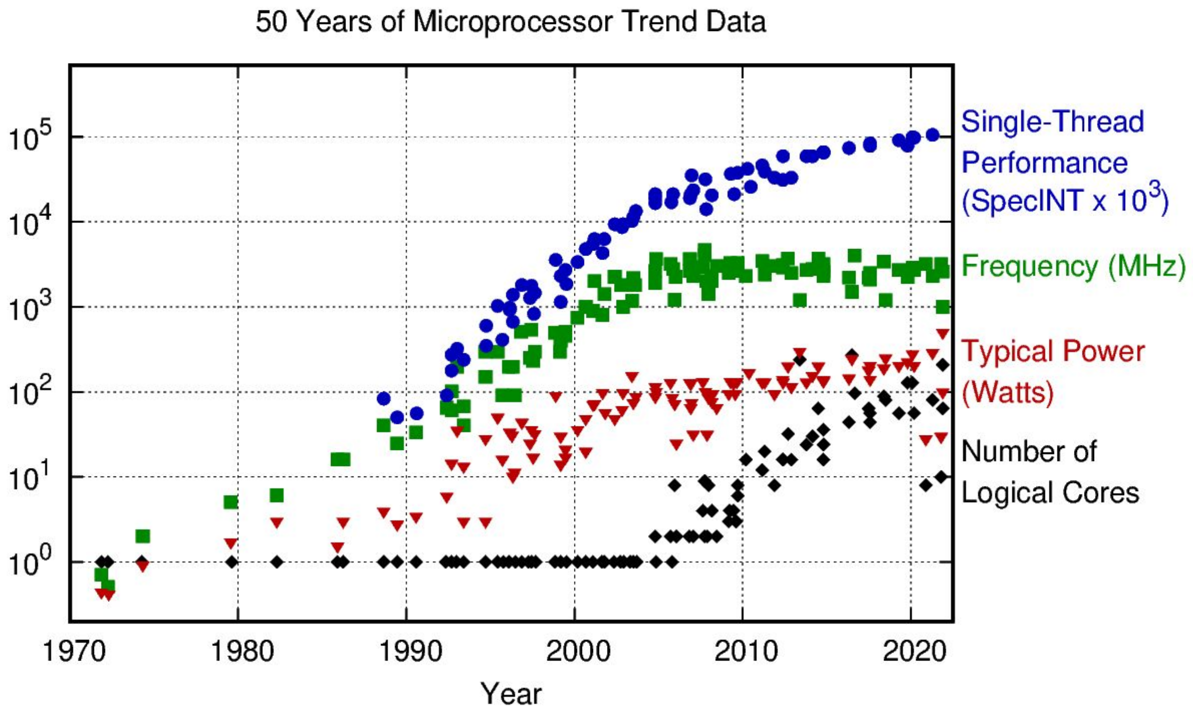
CB Schroeder et al., JINST (2023)

see backup slides for detailed benchmark studies

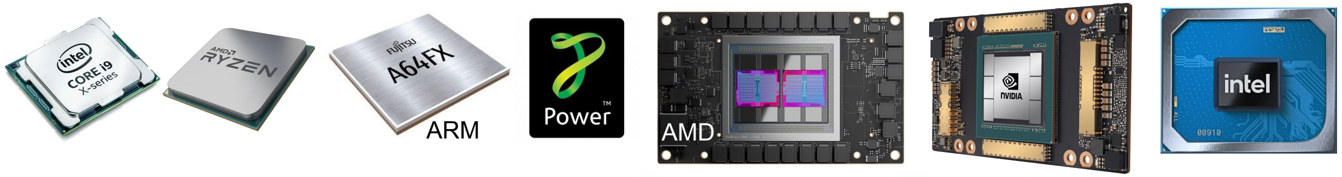
Exascale Technologies for Particle Accelerator Modeling

Power-Limits Seeded a Cambrian Explosion of Compute Architectures

Personal Computers



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

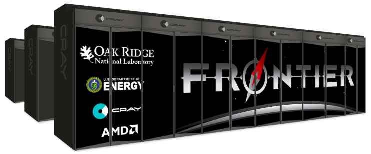


CPUs

GPUs

Supercomputers

El Capitan (USA): 1.7 EFlops
- AMD GPUs

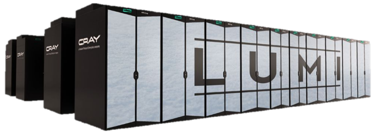


Frontier (USA): 1.3 EFlops
- AMD GPUs

Aurora (USA): 1.0 EFlops
- Intel GPUs

Jupiter Booster: 0.8 EFlops
- Nvidia GPUs (Germany)

Fugaku (Japan): 0.44 EFlops
- Fujitsu ARM CPUs

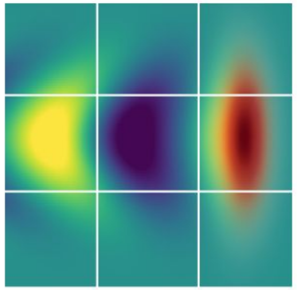


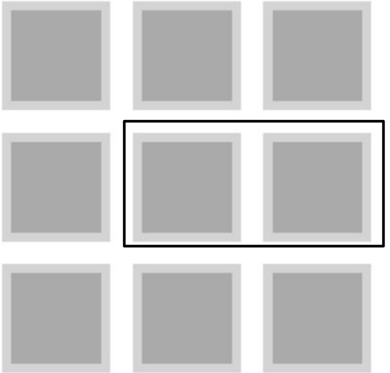
Lumi (Finland): 0.38 EFlops
- AMD GPUs

Leonardo (Italy): 0.24 EFlops
- Nvidia GPUs

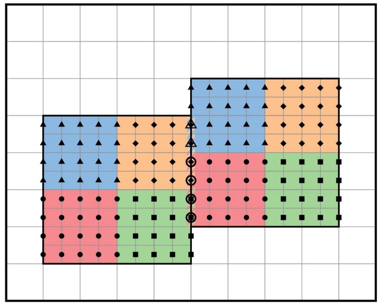
Power-Limits Seed a *Cambrian Explosion* of Compute Architectures

distribute *one* simulation over **10,000s of computers** each **often 100s of cores**

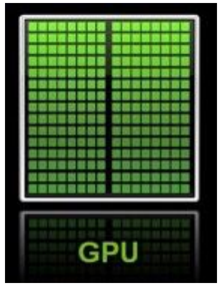


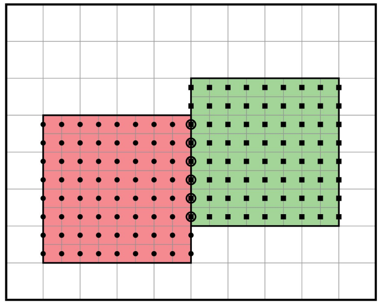


optional: for detailed simulations



with tiling





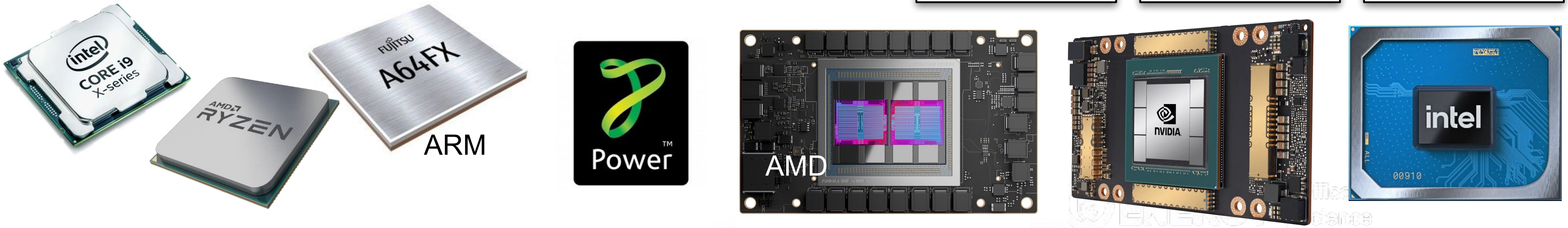
without tiling

potential future

Field-Programmable Gate Array (FPGA)

Application-Specific Integrated Circuit (ASIC)

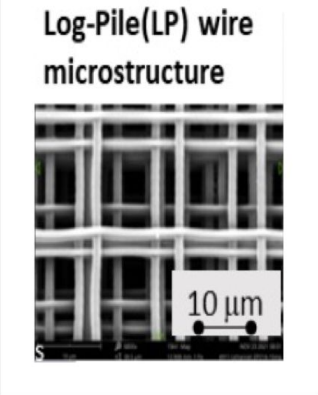
Quantum-Circuit ?

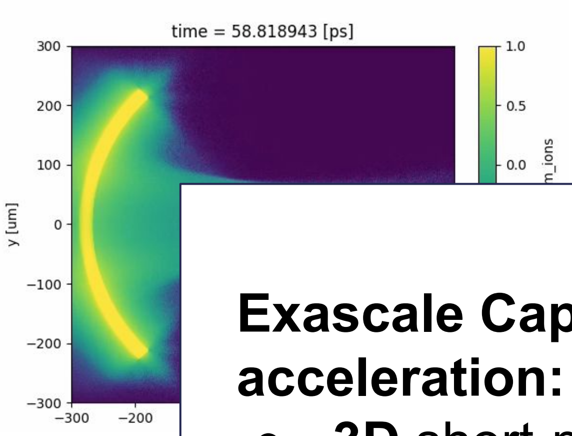


Laser-Plasma Acceleration of Ions: Many 3D Laser-Solid Simulations will even need Post-Exascale

Laser-Matter Interaction with complex targets

Work with Andreas Kemp (LLNL)





- Cost and feasibility of fast energetic ions depends di efficiency
- Complex target geometries require modeling at scale – enabled by GPU based explicit particle-in-cell

Lawrence Livermore National Laboratory

A Kemp et al., Phys. Plasmas 32, 093102 (2025)

Laser-Ion Acceleration from solids

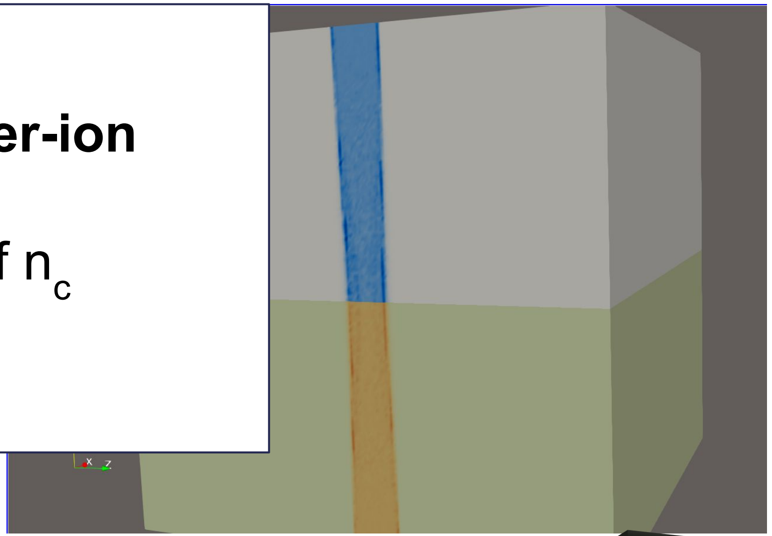
Work with Davide Terzani (LBNL)

BERKELEY LAB
Lawrence Berkeley National Laboratory

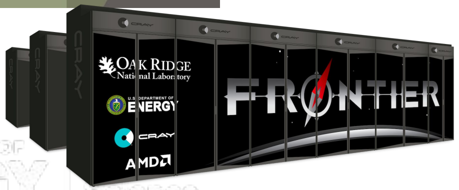
- investigating energy scaling for laser-ion acceleration experiments with future laser systems (more on this soon)

Exascale Capabilities for laser-ion acceleration:

- **3D** short-pulse up to 10s of n_c
- **2D** for 10s of ps, >>100n_c



30-50% of OLCF's

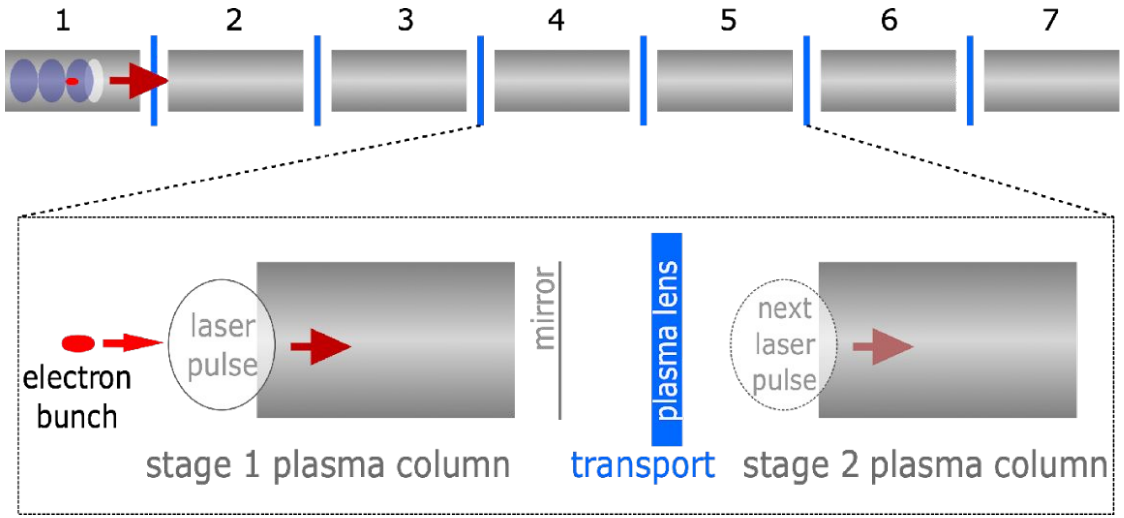


Connecting Scales & Models with Machine-Learning

Building *Ultra-Fast* Plasma Stage Models from WarpX Data

Central BLAST Code Interoperability: Combine Plasma & RF Accelerator Elements for start-to-end modeling

example: high-quality, first-principle ***WarpX data*** (1fC witness beam) used for ***ImpactX*** ML surrogate training



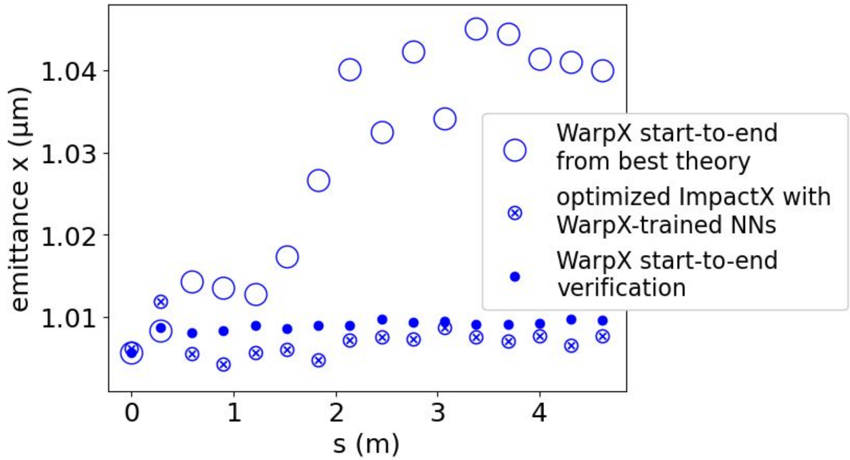
tightly-coupled LPA-*neural networks* inside **ImpactX**

WarpX start-to-end simulation
256 GPUs
1 simulation / 5.1 hours

⇩

ImpactX with WarpX-trained NNs
1 GPU
2-4 simulations / sec

LPA + Transport Optimization
with 1000s of evaluations



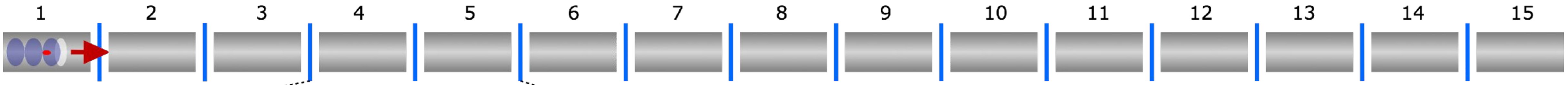
≈750x estimated cost savings with in-the-loop ML optimization workflow

RT Sandberg et al., IPAC23, DOI:10.18429/JACoW-IPAC2023-WEPA101 (2023)
RT Sandberg et al., ***PASC24 Best Paper*** (2024)

We Exploit our High-Quality HPC Data for ML-Boosted Collider Design

Central BLAST Code Interoperability: Combine Plasma & RF Accelerator Elements for start-to-end modeling

example: high-quality, first-principle ***WarpX data*** (1fC witness beam) used for ***ImpactX*** ML surrogate training



Advances BLAST capabilities towards:

- rapid start-to-end designs
- digital twins & "real-time" feedback

Also works for *non-LPA segments:*

e.g., IOTA nonlinear lens [IPAC23]

What's next?

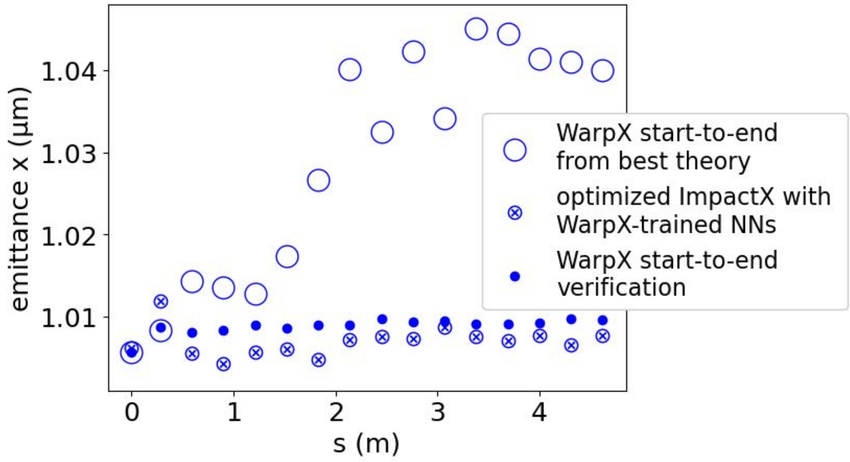
- *Collective effects:* space charge, wakes, feedback, etc. – coming soon!
- Use as *plasma model* in *system codes*?

WarpX start-to-end simulation
256 GPUs
1 simulation / 5.1 hours

⇩

ImpactX with WarpX-trained NNs
1 GPU
2-4 simulations / sec

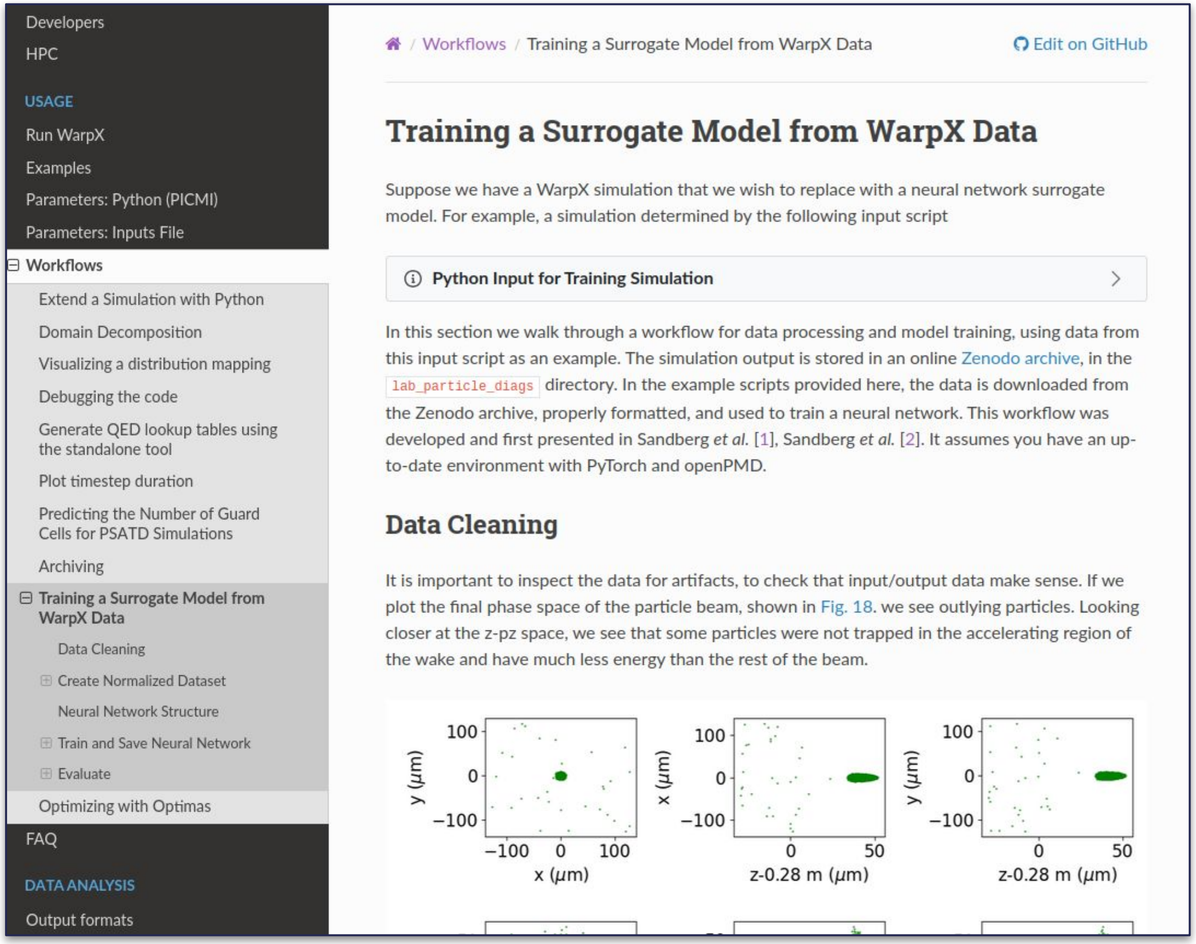
LPA + Transport Optimization
with 1000s of evaluations



≈750x estimated cost savings with in-the-loop ML optimization workflow

RT Sandberg et al., IPAC23, DOI:10.18429/JACoW-IPAC2023-WEPA101 (2023)
RT Sandberg et al., ***PASC24 Best Paper*** (2024) A Dhamrait, R Lehe et al., in preparation

Build Your Own In-the-loop Machine Learning Surrogates Beyond Single-Particle Tracking Maps



Developers
HPC
USAGE
Run WarpX
Examples
Parameters: Python (PICMI)
Parameters: Inputs File
Workflows
Extend a Simulation with Python
Domain Decomposition
Visualizing a distribution mapping
Debugging the code
Generate QED lookup tables using the standalone tool
Plot timestep duration
Predicting the Number of Guard Cells for PSATD Simulations
Archiving
Training a Surrogate Model from WarpX Data
Data Cleaning
Create Normalized Dataset
Neural Network Structure
Train and Save Neural Network
Evaluate
Optimizing with Optimas
FAQ
DATA ANALYSIS
Output formats

Workflows / Training a Surrogate Model from WarpX Data — Edit on GitHub

Training a Surrogate Model from WarpX Data

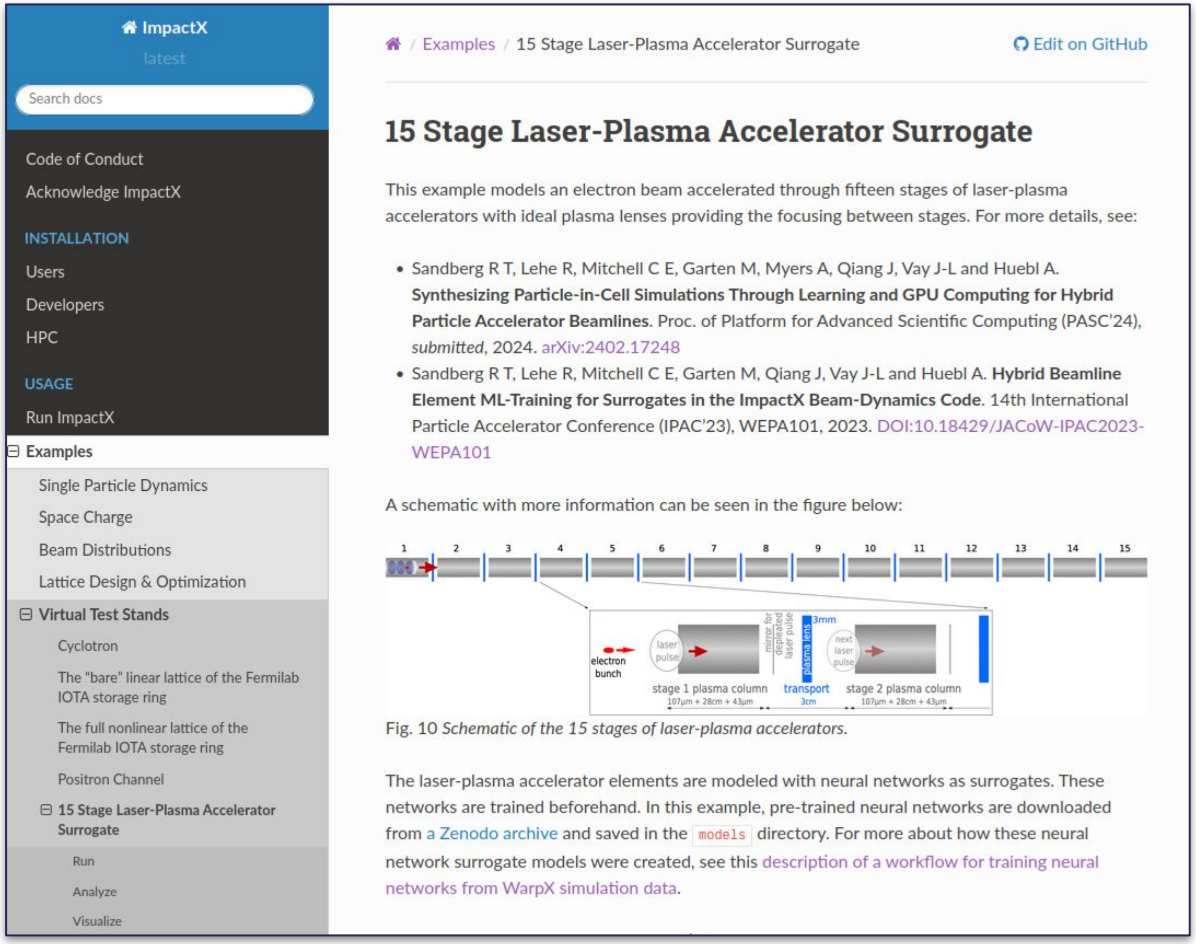
Suppose we have a WarpX simulation that we wish to replace with a neural network surrogate model. For example, a simulation determined by the following input script

Python Input for Training Simulation

In this section we walk through a workflow for data processing and model training, using data from this input script as an example. The simulation output is stored in an online Zenodo archive, in the `lab_particle_diags` directory. In the example scripts provided here, the data is downloaded from the Zenodo archive, properly formatted, and used to train a neural network. This workflow was developed and first presented in Sandberg *et al.* [1], Sandberg *et al.* [2]. It assumes you have an up-to-date environment with PyTorch and openPMD.

Data Cleaning

It is important to inspect the data for artifacts, to check that input/output data make sense. If we plot the final phase space of the particle beam, shown in Fig. 18. we see outlying particles. Looking closer at the z-pz space, we see that some particles were not trapped in the accelerating region of the wake and have much less energy than the rest of the beam.



ImpactX
latest
Search docs
Code of Conduct
Acknowledge ImpactX
INSTALLATION
Users
Developers
HPC
USAGE
Run ImpactX
Examples
Single Particle Dynamics
Space Charge
Beam Distributions
Lattice Design & Optimization
Virtual Test Stands
Cyclotron
The "bare" linear lattice of the Fermilab IOTA storage ring
The full nonlinear lattice of the Fermilab IOTA storage ring
Positron Channel
15 Stage Laser-Plasma Accelerator Surrogate
Run
Analyze
Visualize

Examples / 15 Stage Laser-Plasma Accelerator Surrogate — Edit on GitHub

15 Stage Laser-Plasma Accelerator Surrogate

This example models an electron beam accelerated through fifteen stages of laser-plasma accelerators with ideal plasma lenses providing the focusing between stages. For more details, see:

- Sandberg R T, Lehe R, Mitchell C E, Garten M, Myers A, Qiang J, Vay J-L and Huebl A. **Synthesizing Particle-in-Cell Simulations Through Learning and GPU Computing for Hybrid Particle Accelerator Beamlines.** Proc. of Platform for Advanced Scientific Computing (PASC'24), *submitted*, 2024. arXiv:2402.17248
- Sandberg R T, Lehe R, Mitchell C E, Garten M, Qiang J, Vay J-L and Huebl A. **Hybrid Beamline Element ML-Training for Surrogates in the ImpactX Beam-Dynamics Code.** 14th International Particle Accelerator Conference (IPAC'23), WEPA101, 2023. DOI:10.18429/JACoW-IPAC2023-WEPA101

A schematic with more information can be seen in the figure below:

Fig. 10 *Schematic of the 15 stages of laser-plasma accelerators.*

The laser-plasma accelerator elements are modeled with neural networks as surrogates. These networks are trained beforehand. In this example, pre-trained neural networks are downloaded from a Zenodo archive and saved in the `models` directory. For more about how these neural network surrogate models were created, see this description of a workflow for training neural networks from WarpX simulation data.

These and your own ML ideas can now easily be implemented (Python) & studied in BLAST codes WarpX/ImpactX - see our documentation and detailed examples on how to get started 🚀

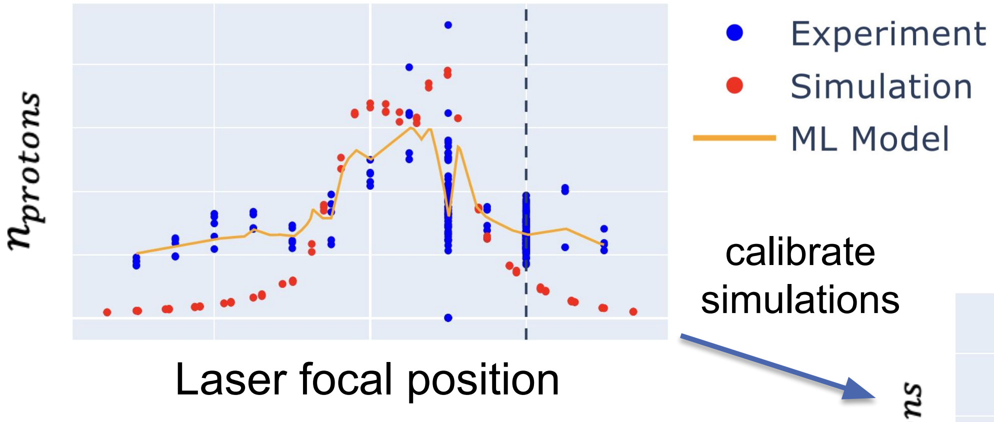
Disagreement between experiments and simulation can be overcome by learning an empirical calibration

Simulations generally reproduce the **correct trends**, but are not always in **quantitative agreement** with experimental observations.

Many potential reasons:

- Simplifying physics assumptions in simulations
- Imperfect knowledge of experimental conditions
- Uncalibrated experimental diagnostics

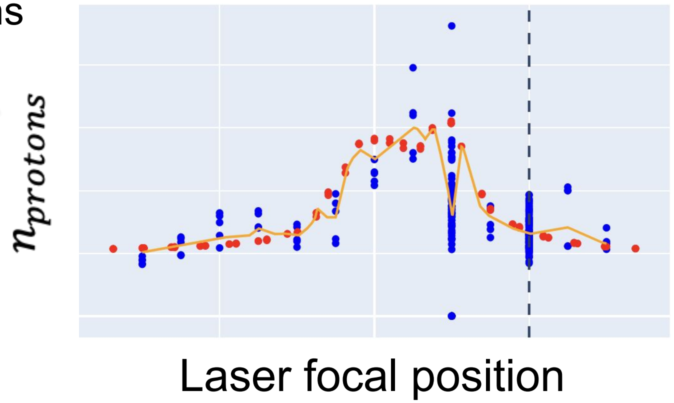
Need addressing, to train a *predictive* ML model on combined data.



calibrate simulations

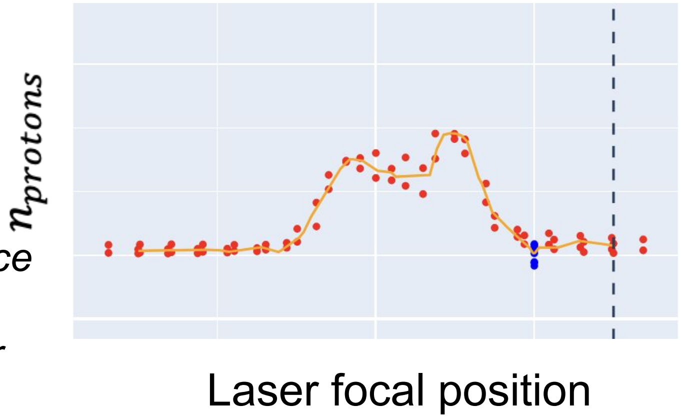
$$n_{protons,exp.} = \alpha \, n_{protons,sim.} + \beta$$

Learned by gradient descent, while training the ML model.



predict where exp. data is sparse

e.g., performance for a different temporal laser profile



T. Boltz et al., arXiv:2403.03225 (2024) R Lehe et al., *manuscript in preparation*

Surrogate Models are Connecting Experimental & Simulation Data

We will soon publish a framework for ML integration between experiments & simulations.

NERSC Spin
Platform for continuously-running science services

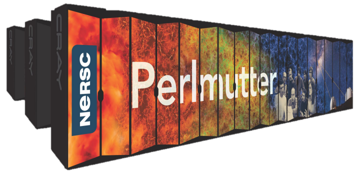
Automatically retrain model

ML model

Database of experiments and simulations

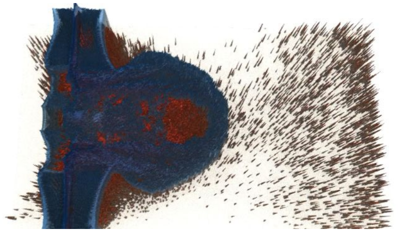
Backend software

NERSC Perlmutter
GPU supercomputer



Database connection

Send data at the end of simulations

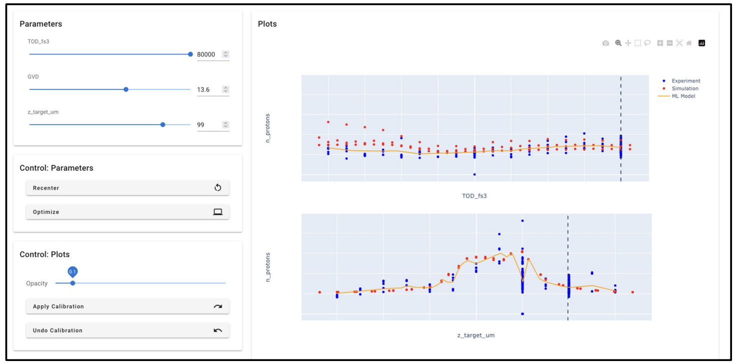


Automatically launch new simulations

BELLA Control room

HTTPS connection

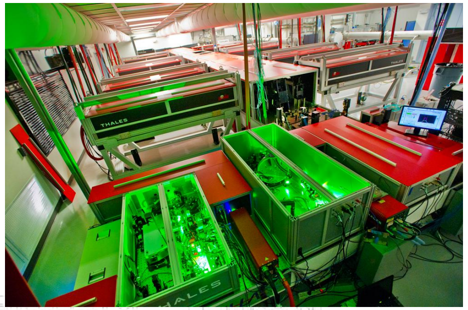
Dashboard



Database connection

Detect new shots and extracts relevant data

BELLA experiments



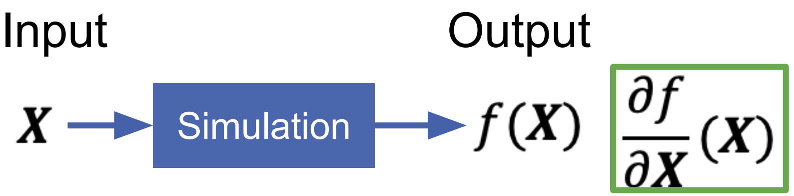
R Lehe et al., *manuscript in preparation*

Embedding NNs in Simulations can Solve Hard, Inverse Problems

Why Differentiable Modeling?

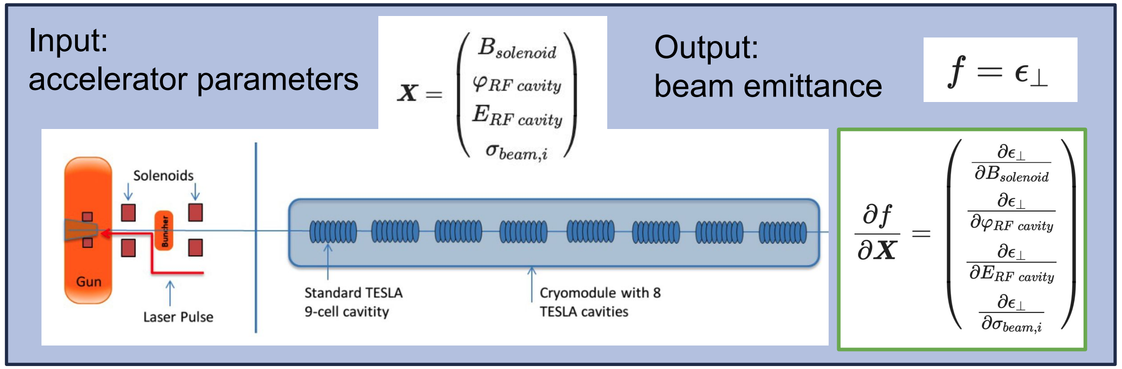
Differentiability is **essential** for many AI/ML techniques, e.g., in **rapid optimization** and **neural network training** (backpropagation).

Regular Simulation →**Differentiable**

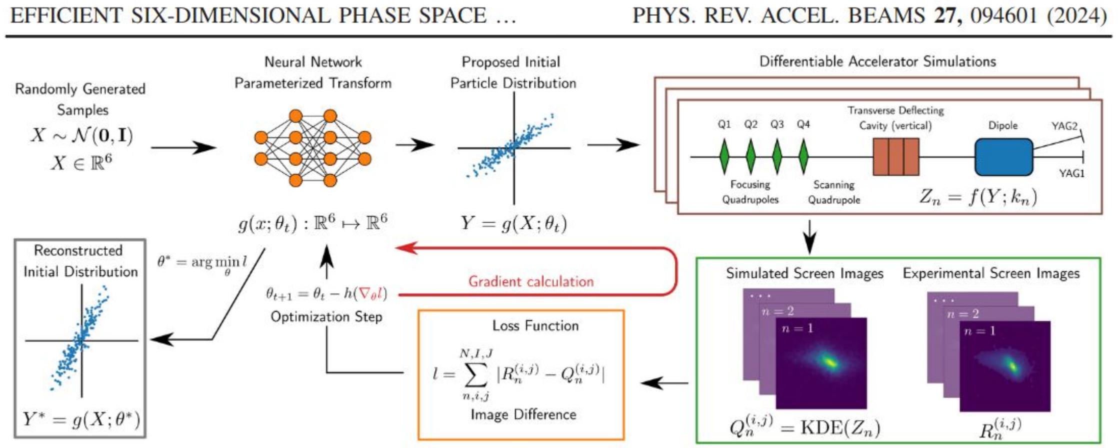


Contributed space charge to recent work (Cheetah), studied scaling laws, and started to implement **differentiable models** in **BLAST**.

"Hard-to-Scan": Multi-Dimensional Optimization Example



"Hard-to-Measure": Reconstruction Example

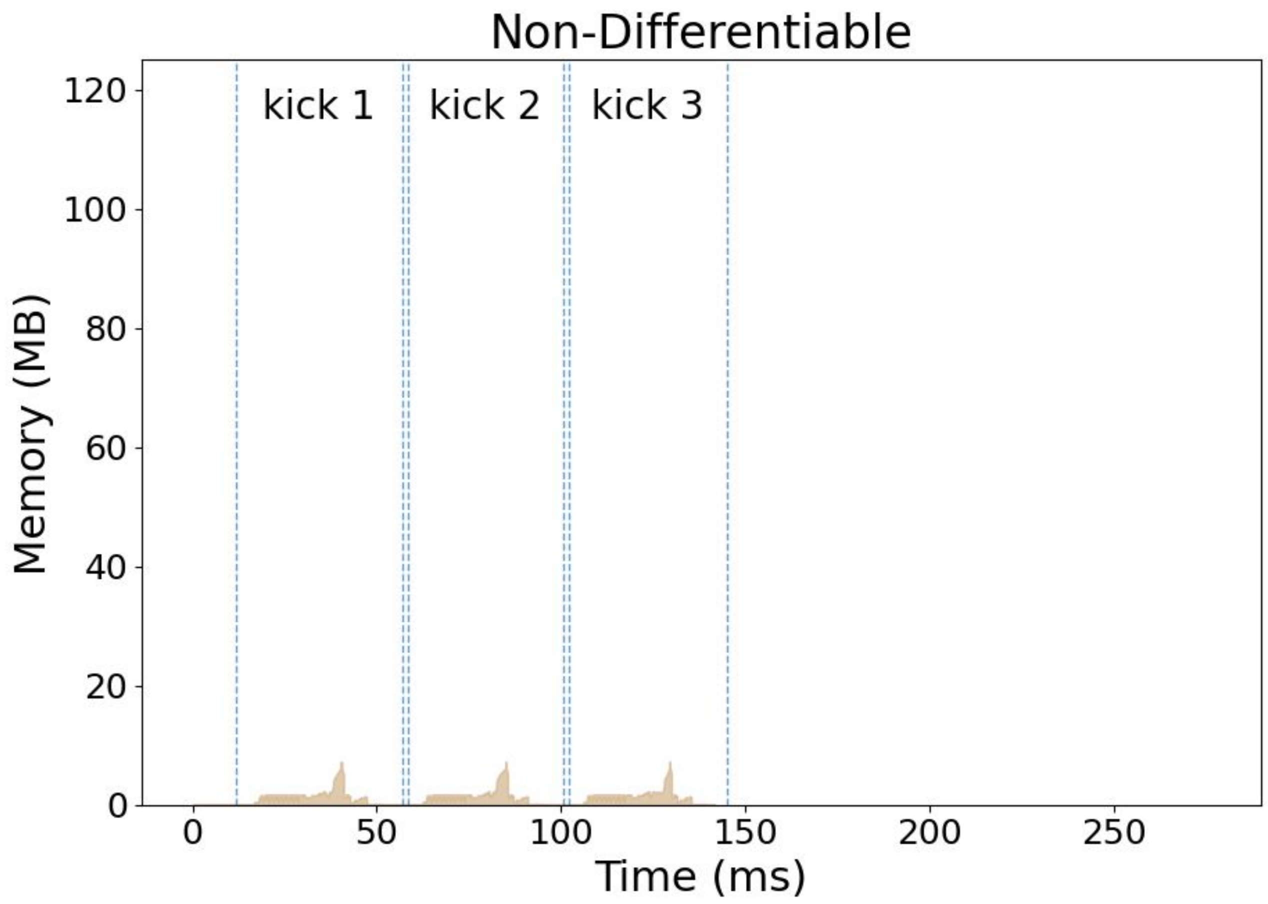


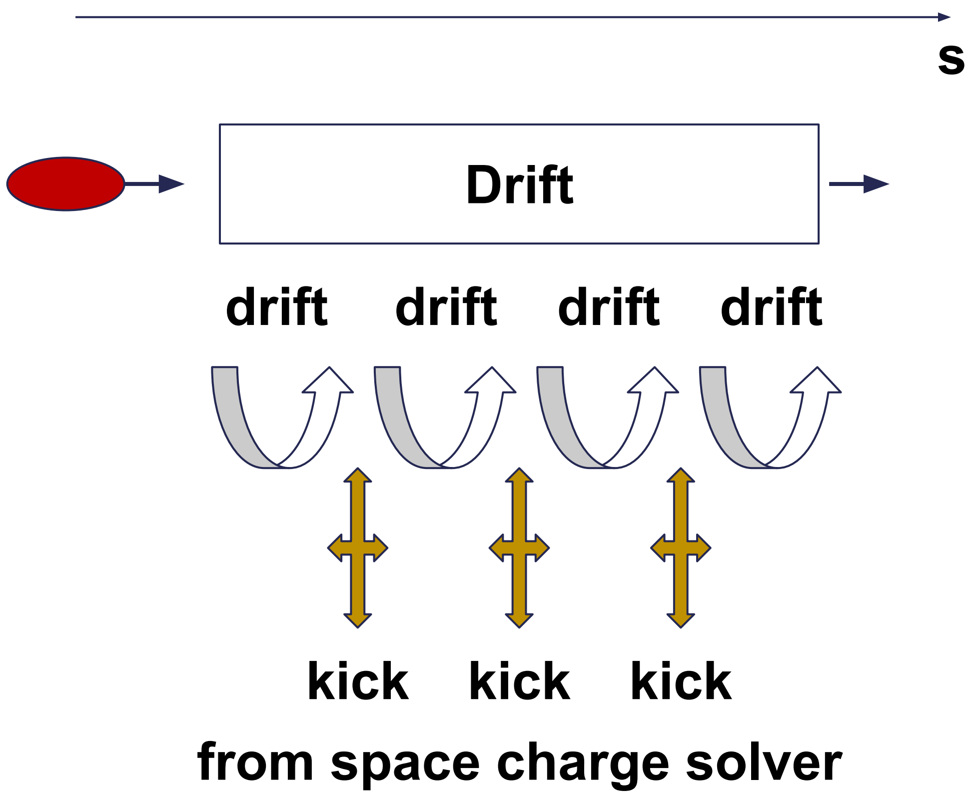
Further applications: self-calibrating beamlines, uncertainty quantification, surrogate-training, digital twin training, …

J.-P. Gonzalez-Aguilera et al., WEPA065 at IPAC2023 (2023) J. Kaiser et al., PRAB 27, 054601 (2024)
A Hoover et al., PRR 6, 033163 (2023) R. Roussel et al., PRL 130 (2023) and PRAB 27, 094601 (2024)
A. Huebl et al., TUP101 at NAPAC25 (2025) W.S. Moses et al., Enzyme, SC22 (2022)

Gradient-Tracking in Differentiable Simulations Quickly Requires a lot of Memory or Intermediate Data Storage

Overall memory use graphs for a full simulation with 3 space charge kicks

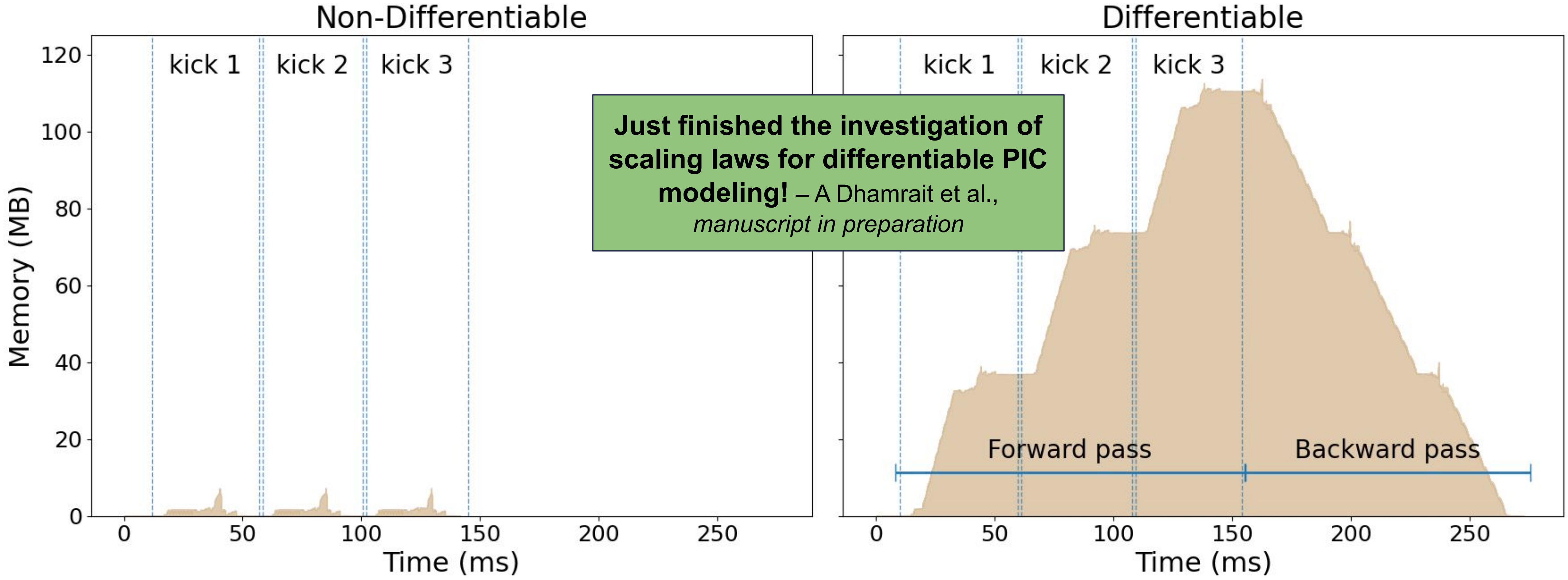




Credit: Remi Lehe & Arjun Dhamrait, Gregoire Charleux, Axel Huebl, Chad Mitchell, Edoardo Zoni   Code: Cheetah (DESY/KIT/SLAC/ANL/LBNL)

Gradient-Tracking in Differentiable Simulations Quickly Requires a lot of Memory or Intermediate Data Storage

Overall memory use graphs for a full simulation with 3 space charge kicks



Just finished the investigation of scaling laws for differentiable PIC modeling! – A Dhamrait et al., *manuscript in preparation*

Credit: Remi Lehe & Arjun Dhamrait, Gregoire Charleux, Axel Huebl, Chad Mitchell, Edoardo Zoni   Code: Cheetah (DESY/KIT/SLAC/ANL/LBNL)

Summary

Exascale Technologies

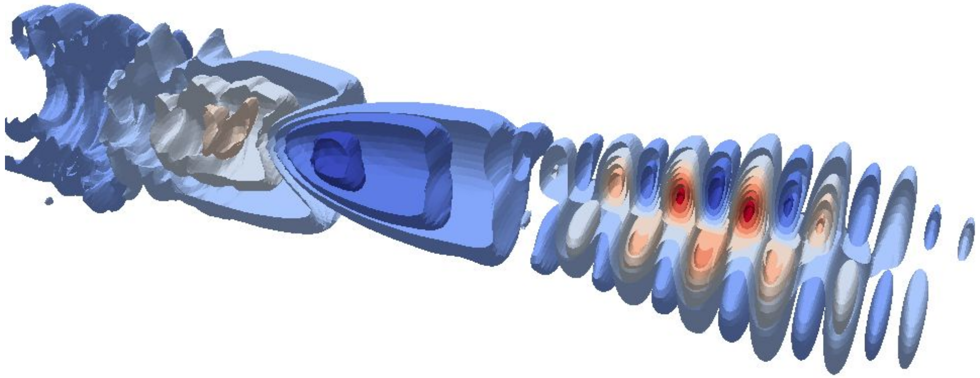
- Make an impact in day-to-day accelerator modeling: from **laptops** to supercomputers

Machine-Learning: Modelization from Data

- Fast, very detailed, *specialized* models
- Connects experiments & simulations
- Could assist to solve hard, inverse problems

Start-to-End: Community Modeling

- Beam, Plasma & Accelerator Simulation Toolkit (BLAST)
- **Comprehensive, multi-physics** tools for model building
- **Fully open, active** community on codes & standards:
  - **contribute** online and in open meetings: Q&A, benchmarks, new features, …
  - new **integrations** in optimizers, system codes, ML



github.com/**BLAST-WarpX**
github.com/**BLAST-ImpactX**
github.com/**Hi-PACE**
github.com/**AngelFP/Wake-T**
github.com/**picmi-standard**
github.com/**openPMD** **openPMD.org**
github.com/**optimas-org**
github.com/**campa-consortium/pals**
campa.lbl.gov, blast.lbl.gov

Contacts and Funding Support

Presenter & Contacts

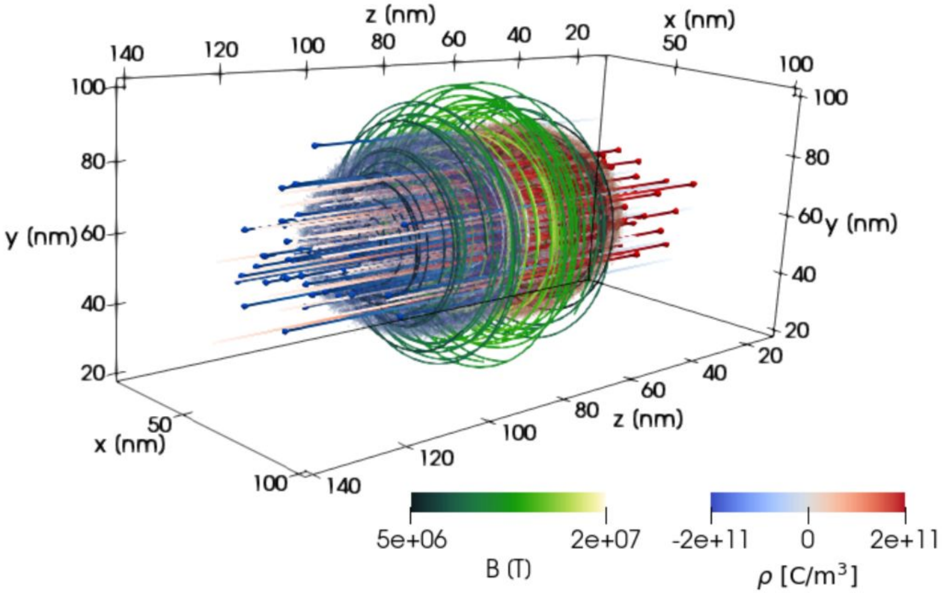
- Axel Huebl axelhuebl@lbl.gov
- Remi Lehe rlehe@lbl.gov
- Chad Mitchell ChadMitchell@lbl.gov
- Arianna Formenti ariannaformenti@lbl.gov
- Jean-Luc Vay jlvay@lbl.gov

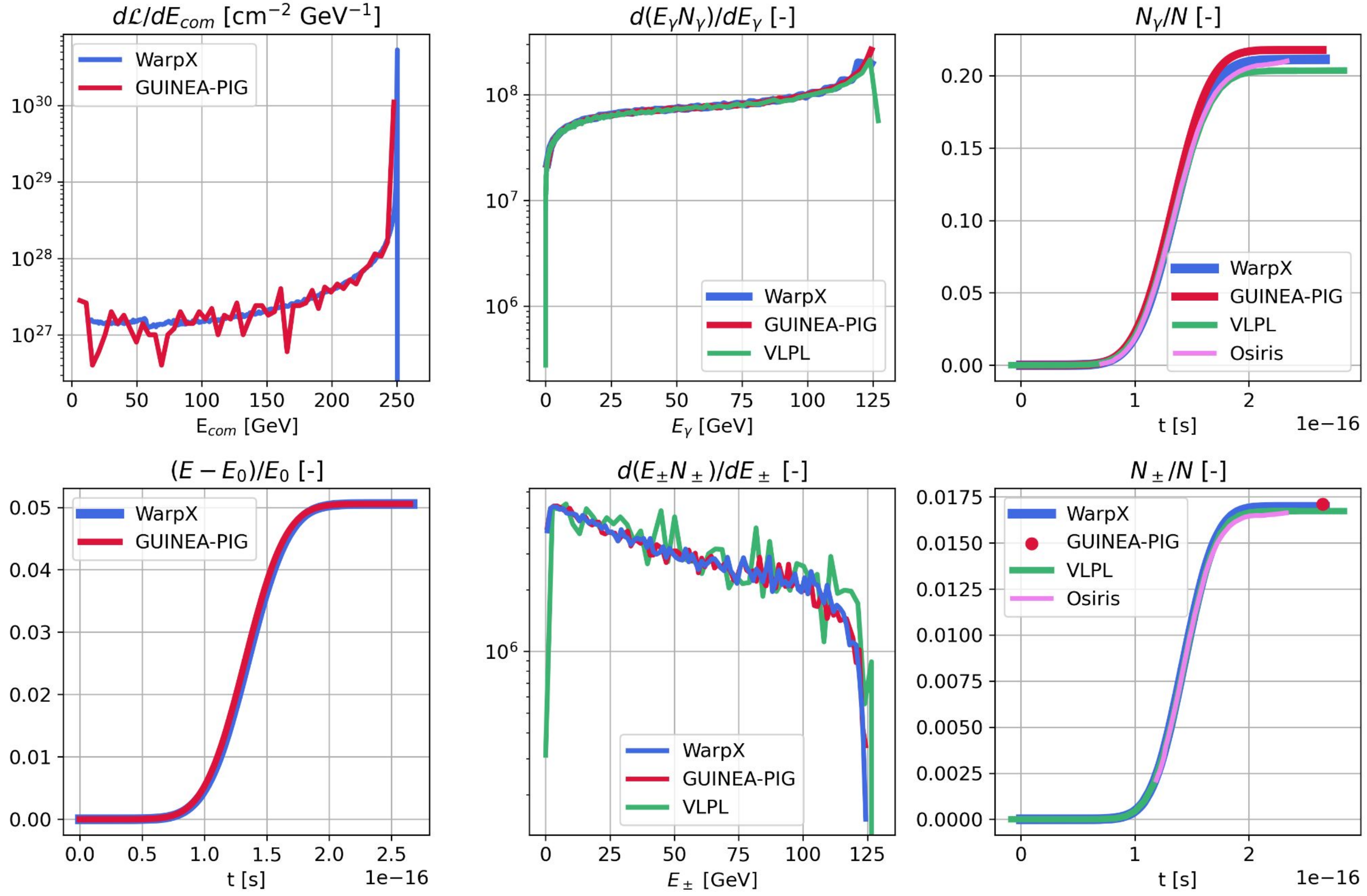
github.com/**BLAST-WarpX**
github.com/**BLAST-ImpactX**
github.com/**Hi-PACE**
github.com/**AngelFP/Wake-T**
github.com/**picmi-standard**
github.com/**openPMD** **www.openPMD.org**
github.com/**optimas-org**
github.com/**campa-consortium/pals**
campa.lbl.gov, blast.lbl.gov

Supported by the **CAMPA (and KISMET) collaborations**, a project of the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of High Energy Physics (Fusion Energy Sciences, resp.), **Scientific Discovery through Advanced Computing (SciDAC)** program. This work was also performed in part by the **Laboratory Directed Research and Development Program** of **Lawrence Berkeley National Laboratory** under U.S. Department of Energy Contract No. DE-AC02-05CH11231, **Lawrence Livermore National Laboratory** under Contract No. DE-AC52–07NA27344 and **SLAC National Accelerator Laboratory** under Contract No. AC02–76SF00515.
This research used resources of the **Oak Ridge Leadership Computing Facility**, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725, the **National Energy Research Scientific Computing Center (NERSC)**, a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231, and the supercomputer Fugaku provided by **RIKEN**.

Backup Slides

Excellent agreement between WarpX and other codes with spherical nanobeams

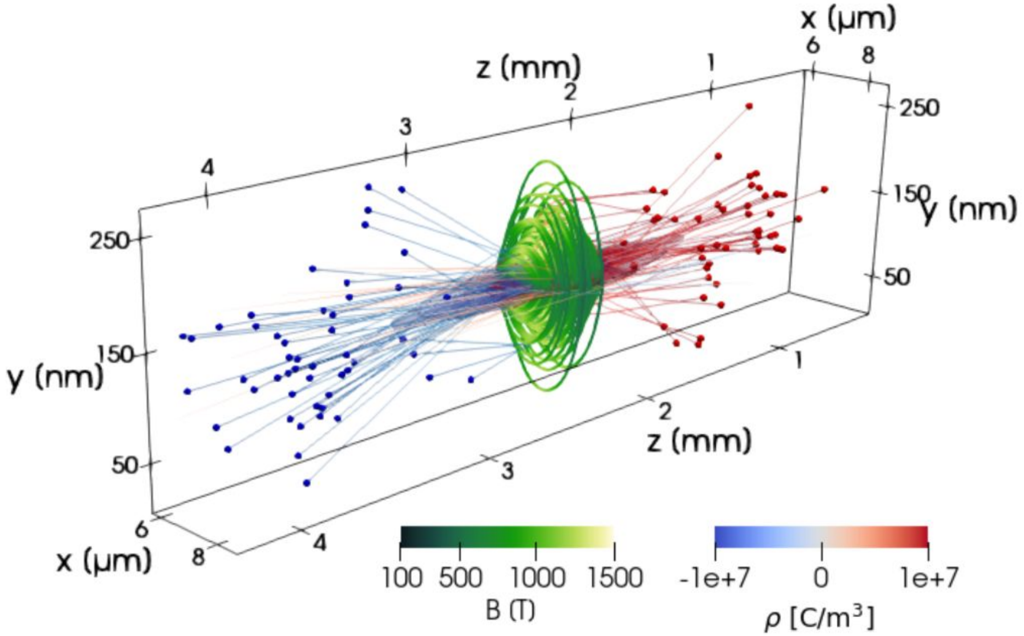




- E_{COM} = 250 GeV
- N = 8.7 * 10^8
- spherical beams: $\sigma_z = \sigma_x = \sigma_y$ = 10 nm
- zero emittance
- low disruption D = 0.001
- max quantum parameter χ = ϒ ~ 1700

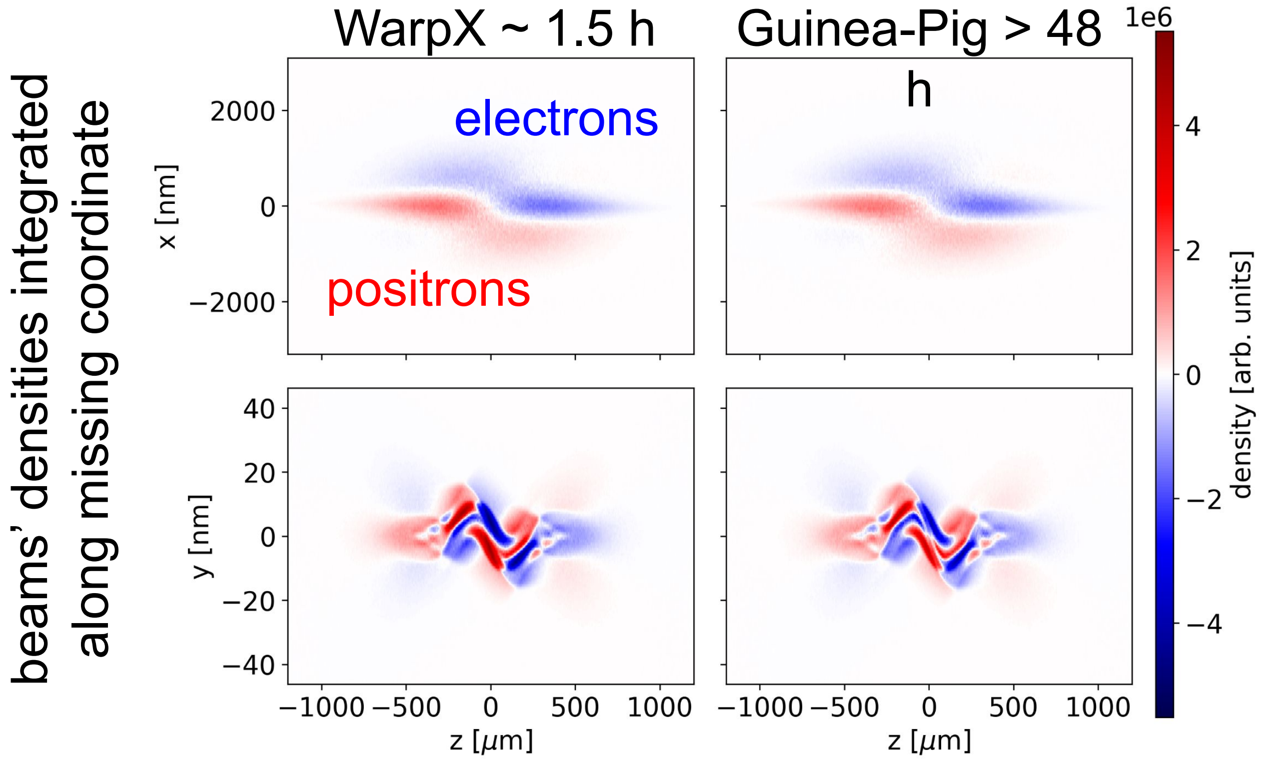
[Yakimenko et al. Phys. Rev. Lett. 122, 190404 (2019)]

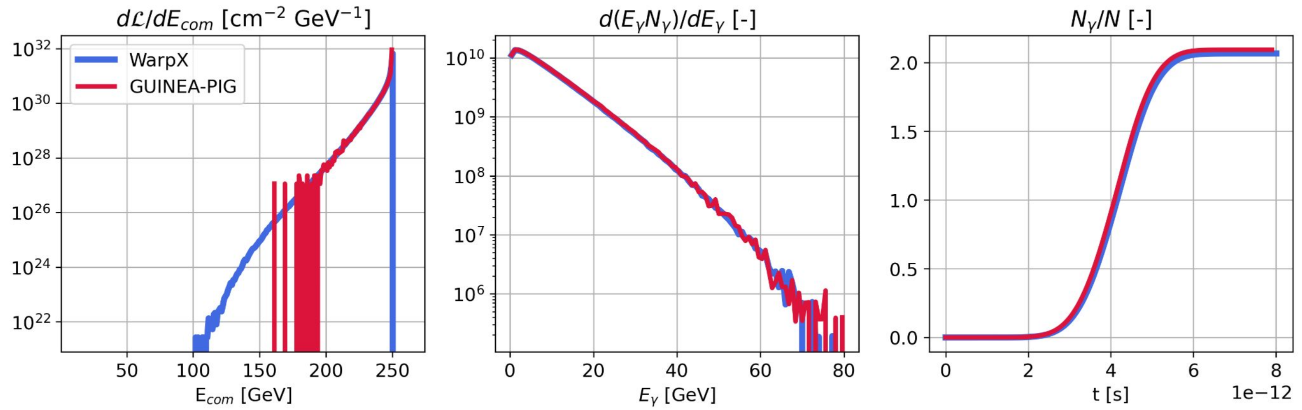
Excellent agreement between WarpX and Guinea-Pig with flat ILC beams



- E_{COM} = 250 GeV
- N = 2×10^{10}
- σ_z = 300 μm
- σ^*_x = 516 nm | σ^*_y = 7.7 nm
- ϵ_x = 5 μm | ϵ_y = 35 nm
- flat beams
- significant disruption D_x = 0.30, D_y = 24.39
- max quantum parameter $\chi = \Upsilon \sim 0.3$

[The International Linear Collider: Report to Snowmass 2021





Model Level of Realism: Benchmarking Interaction Point Physics

Source → Staging of ~800 elements → >10 TeV IP ← Staging of ~800 elements ← Source

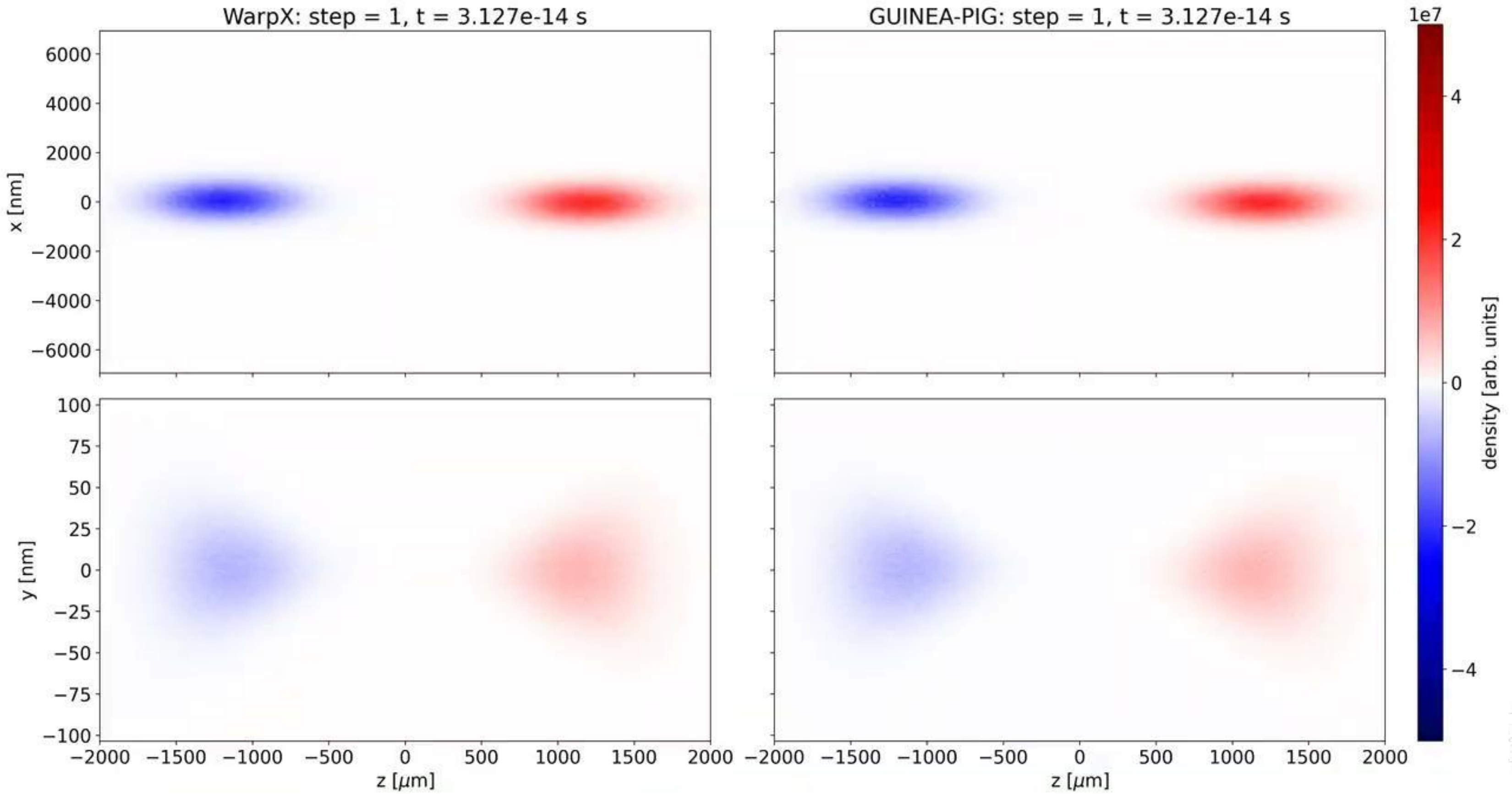
Flat ILC Beams 250 GeV COM*
- high beam disruption
- no significant pair creation

WarpX

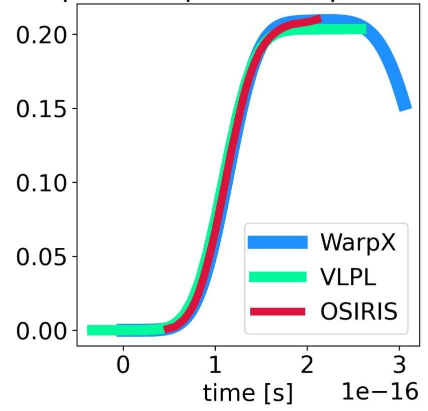
*We also tested: **spherical, round and asymmetric beams** incl. HALHF parameters

Spherical ~nm beams
- low beam disruption
- significant pair creation

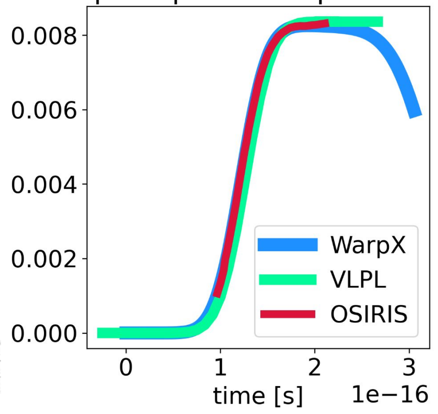
electrons positrons



photons per beam particle

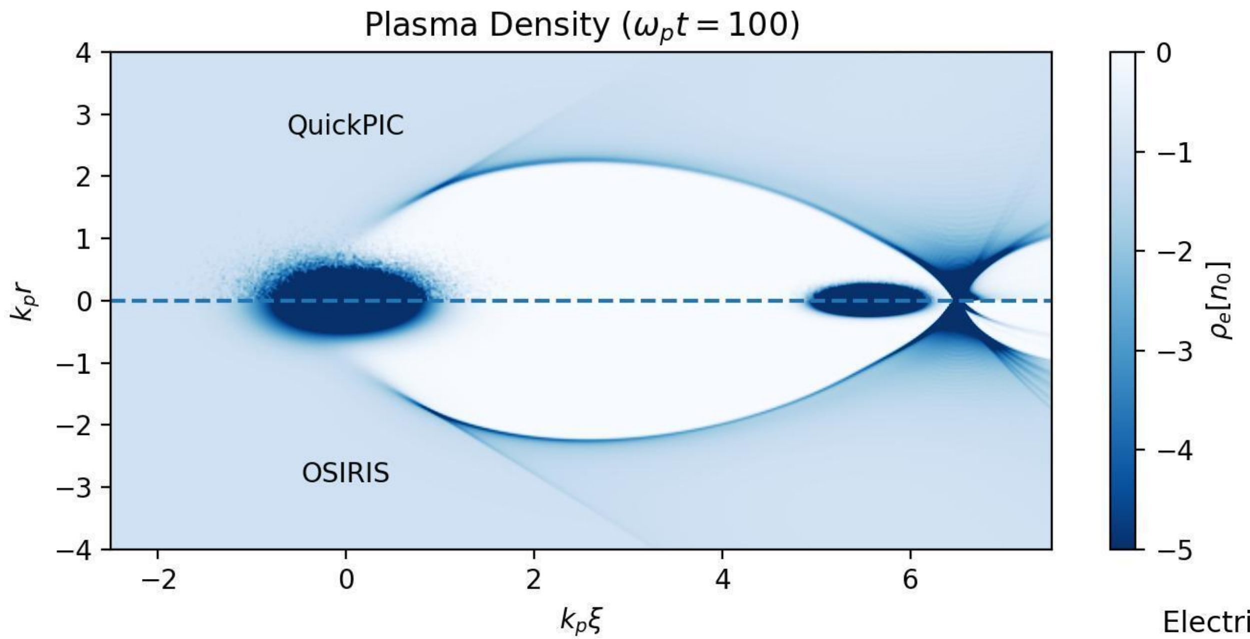


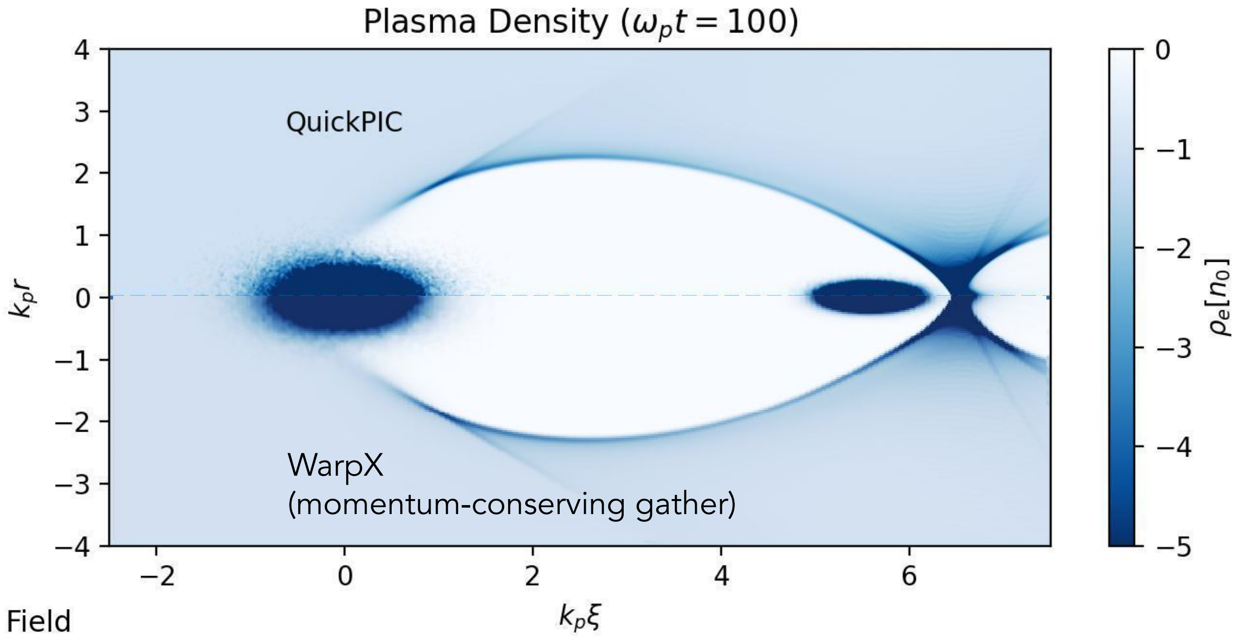
pairs per beam particle

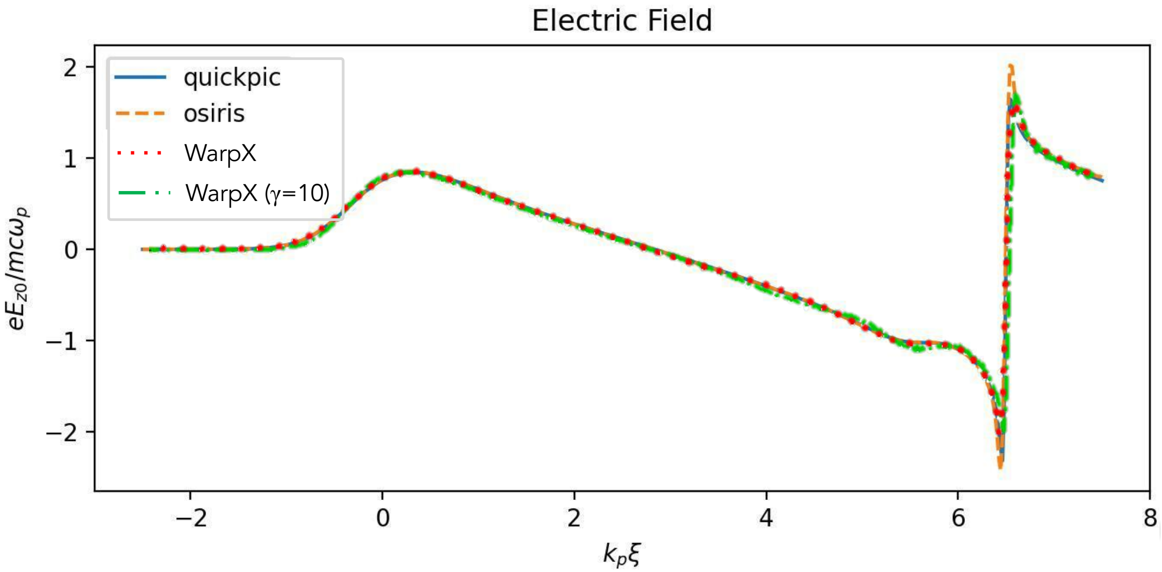


Yakimenko et al., PRL (2019)

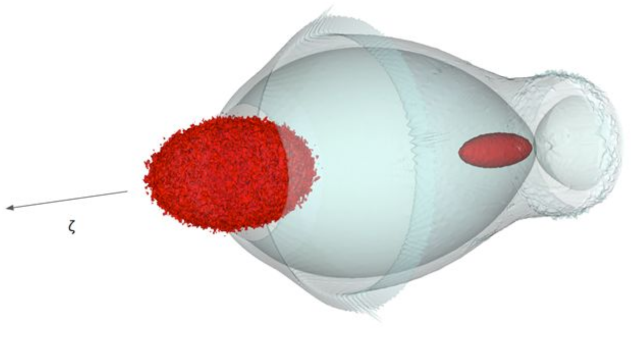
PICMI enables (90%) same input script with different codes







PWFA FACET Example



Standard

≈752x estimated cost savings with in-the-loop ML optimization workflow

Previously (Estimate)

1500 GPU hours simulation
x 1000 iterations

+ 1500 GPU hours validation simulation

= 1 501 500 GPU hours

Optimization with in-the-loop ML surrogate model

450 GPU hours training simulation
+ 3 GPU hours PyTorch training
x 15 stages
+ 10 GPU seconds ImpactX+NN
x 1000 iterations
+ 1500 GPU hours validation simulation

= 1 998 GPU hours

In-the-loop Machine Learning Surrogates Beyond Single-Particle Tracking Maps

- **$R^6 \to R^6$ surrogate**: intentional choice, for the detailed study of **chromatic effects**
  - high level of detail, *arbitrary* low-charge phase spaces, conserves the *phase* of each particle
  - *drop-in* replacement for single-particle, first-principle models

Examples to **include collective effects** in ML surrogates:

- 🔨 **double down**: trajectory + collective beam parameters $R^{6+m} \to R^{6+m}$
  - how: expose additionally *m* collective beam parameters to ML model for various beam charges
  - note: very costly learning phase, unless constrained (e.g., only change 1D current profile)
- 📽️ **project**: learn & predict phase spaces
  - how: learn & predict selected 2D phase spaces for various beam charges
  - note: less detailed; resampling loses phase, e.g., for tune calculations in rings
  - e.g., Emma et al, PRAB 21, 112802 (2018); Edelen et al., TUPS72, IPAC24 (2024)
- 🌱 **simplify**: work with beam moments and simpler distributions
  - how: learn & predict *only* collective beam parameters, learn simpler distributions (e.g., KV)
  - note: little detail; resampling loses phase, e.g., for tune calculations in rings
  - e.g., Edelen et al., PRAB 23, 044601 (2020); Garcia-Cardona & Scheinker, PRAB 27, 024601 (2024)

These and your own ML ideas can now easily be implemented (Python) & studied in BLAST codes WarpX/ImpactX - see our documentation and detailed examples on how to get started 🚀

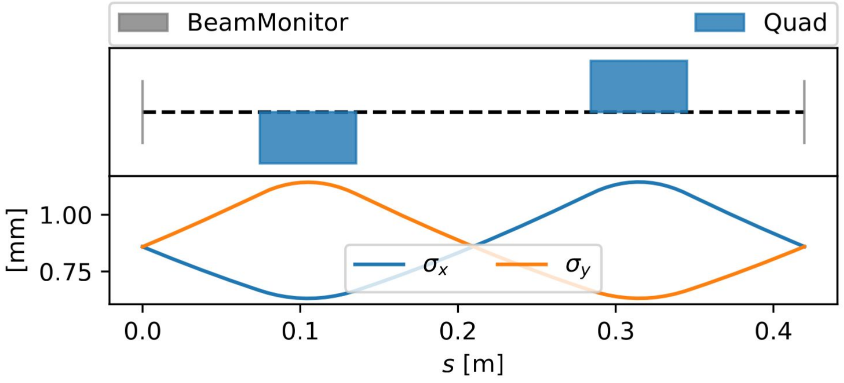
Preparing BLAST for Differentiable Modeling

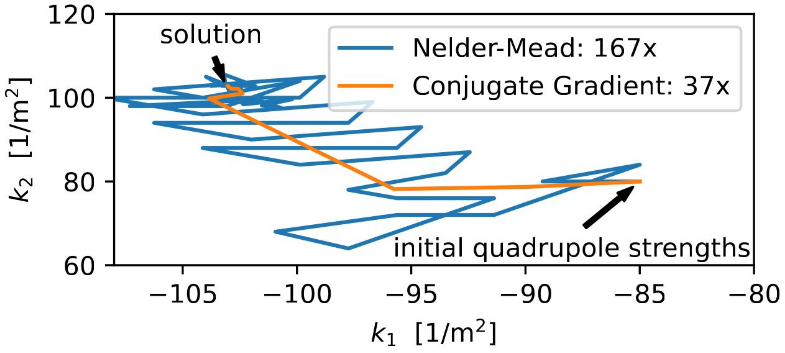
Approach

- Enabling ***automatic differentiation***: the compiler infers the code to calculate gradients **from the existing code** for *f(**X**)*
- **Leverage & enhance** the existing high-performance **BLAST** codes

By slightly restructuring the existing ImpactX code base, we developed a first prototype that supports both **forward-mode and reverse-mode** differentiation for **envelope-based modeling, including space charge effects**.

Example: Gradient-free (Nelder-Mead) and gradient-based (Conjugate Gradient) **optimization of quadrupole strengths** and necessary *number of simulations* to perform.





A. Huebl et al., NAPAC25 poster/paper TUP101 (2025); W.S. Moses et al., "Scalable Automatic Differentiation of Multiple Parallel Paradigms through Compiler Augmentation", SC22 (2022) DOI:10.1109/SC41404.2022.00065; R.D. Ryne et al., Proc. EPAC (2004), Lucerne, Switzerland: KV Beam in a FODO Channel

Surrogate models learn initial ⇨ final phase space map from data generated by a high-fidelity WarpX simulation

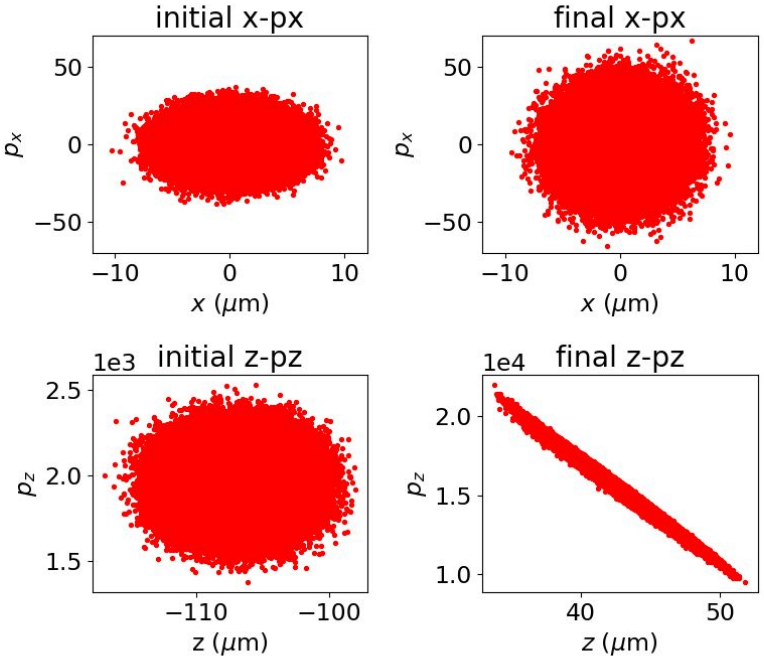
Surrogate model: Generic Transport Map

$$\begin{pmatrix} x \\ y \\ z \\ p_x \\ p_y \\ p_z \end{pmatrix}_i$$

Initial → final phase space

$$f : \mathbb{R}^6 \rightarrow \mathbb{R}^6$$

example: stage 1 training data



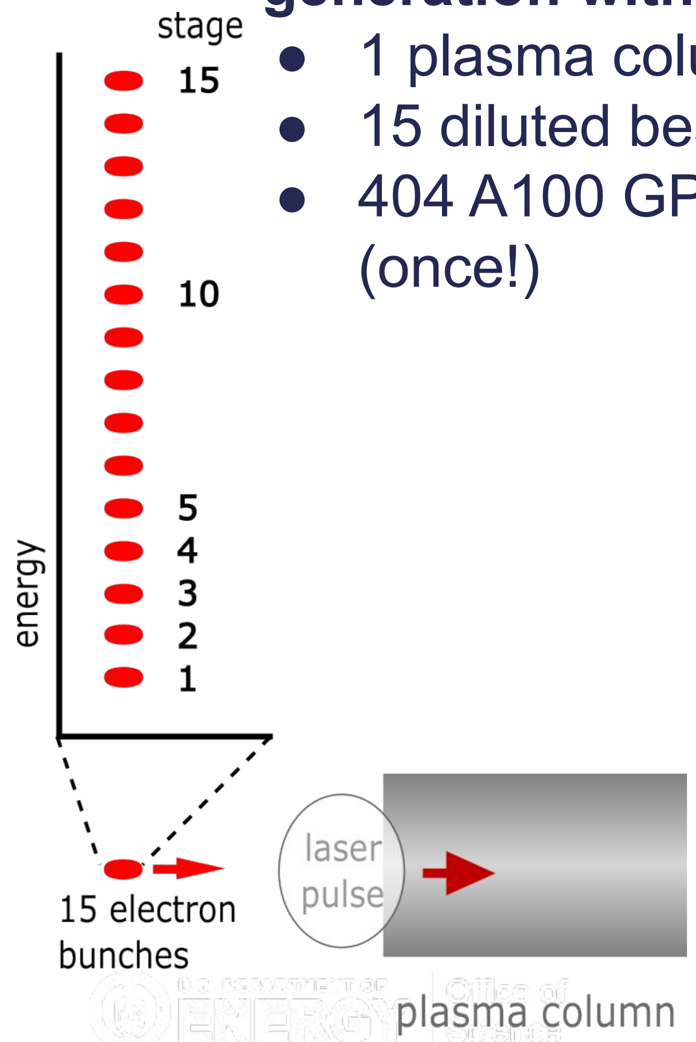
supports beams with
✔ arbitrary profiles
✔ chromatic effects
✗ collective effects

Notes:

- *intentional* choice
- very easy to modify models from Python
- *ideal ground for ML model development*

Training Data generation with WarpX

- 1 plasma column
- 15 diluted beams
- 404 A100 GPUhrs (once!)

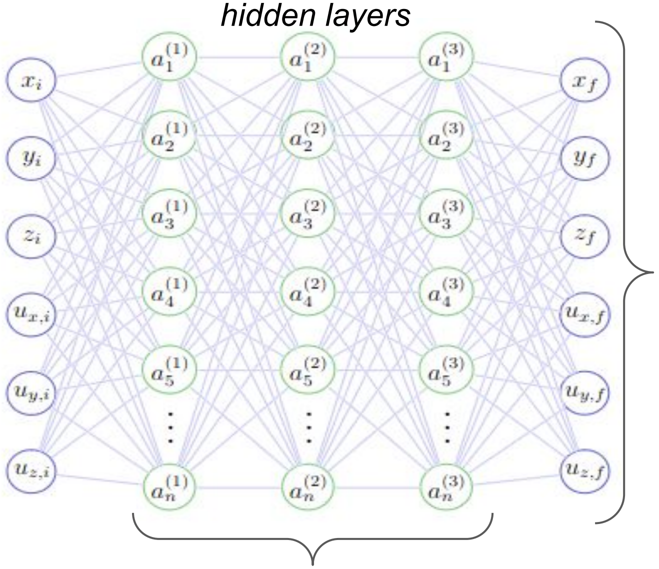


RT Sandberg et al., IPAC23 (2023)

Hyperparameter tuning indicated that relatively simple neural networks were sufficiently accurate

Model of a single stage

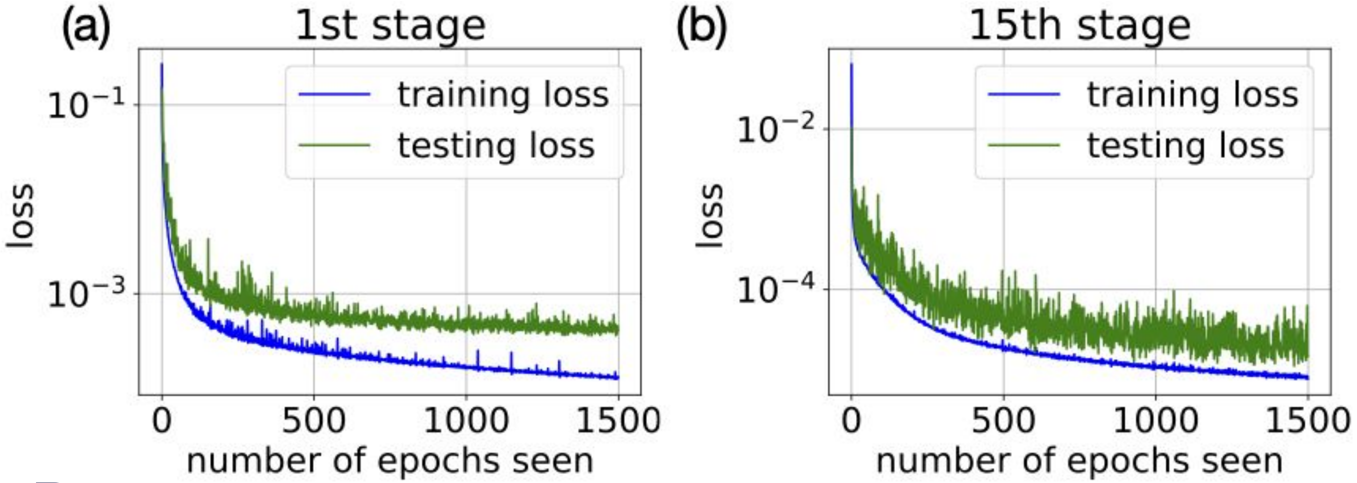
Example of neural network with three hidden layers



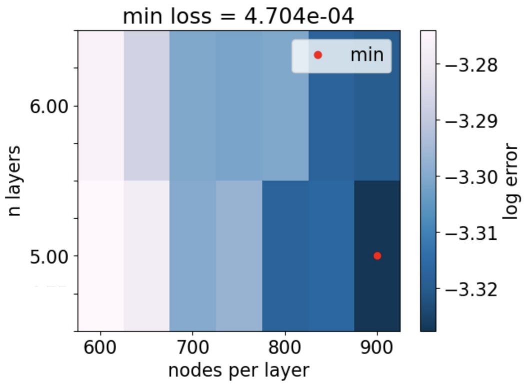
Number of hidden nodes

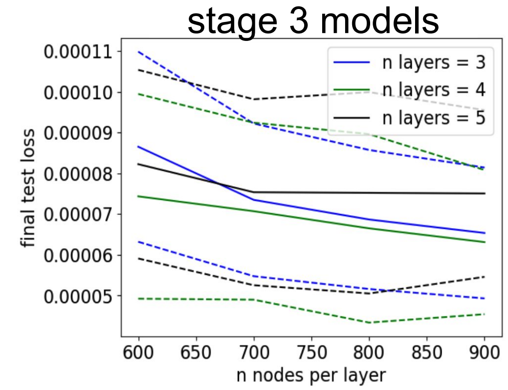
Multiple hidden layers

implemented in PyTorch
- PReLU
- MSE loss
- Adam optimizer



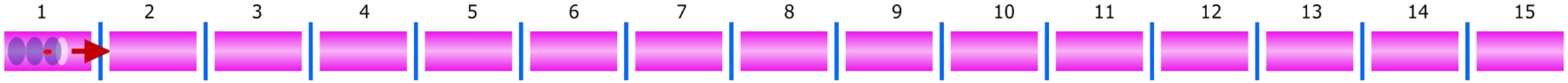
stage 1 hyperparameter tuning



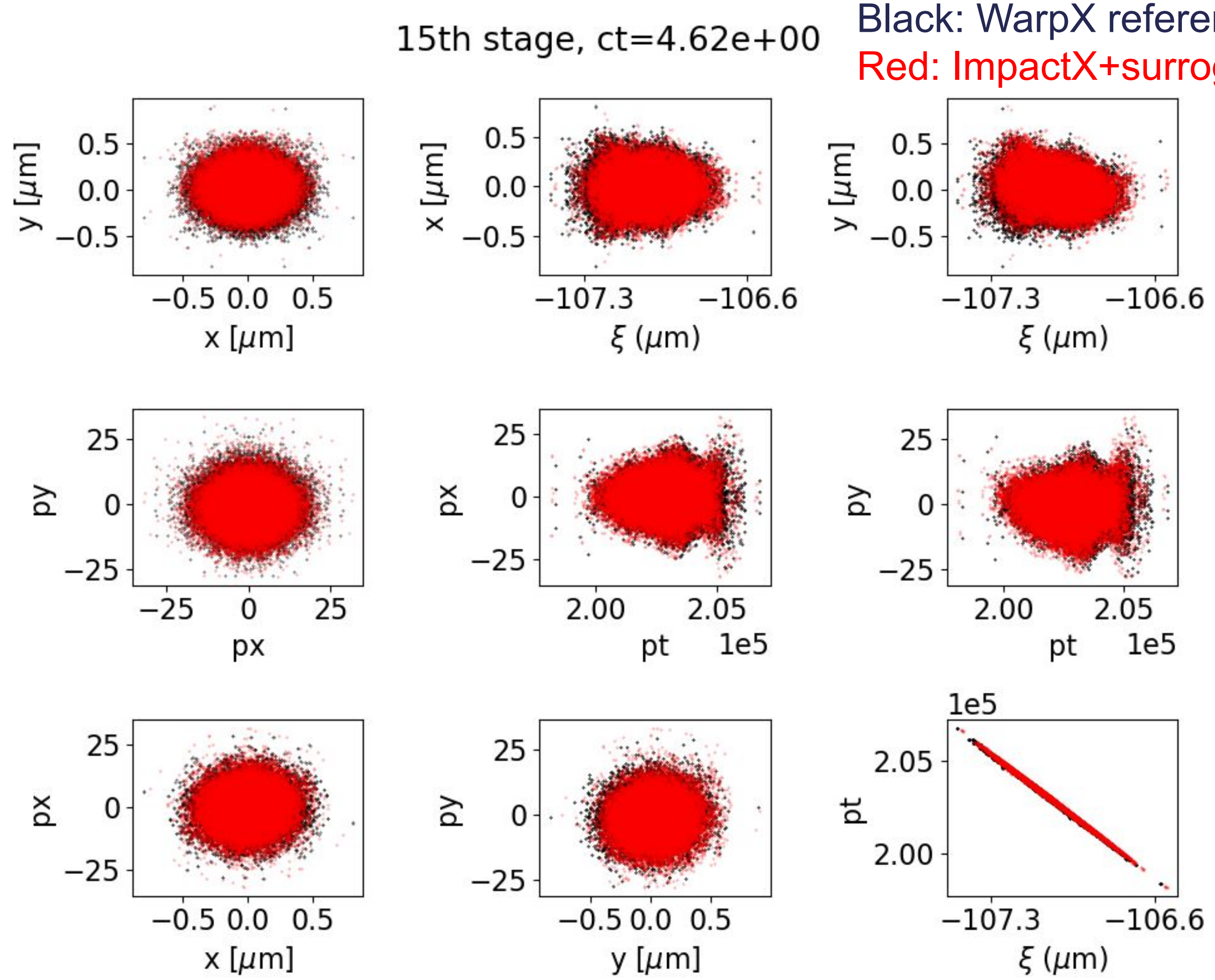


Stages 1-3:	5 hidden layers, 900 nodes per layer
Stages 4-15:	3 hidden layers, 700 nodes per layer

ImpactX+WarpX surrogate agrees with WarpX reference after 15 stages



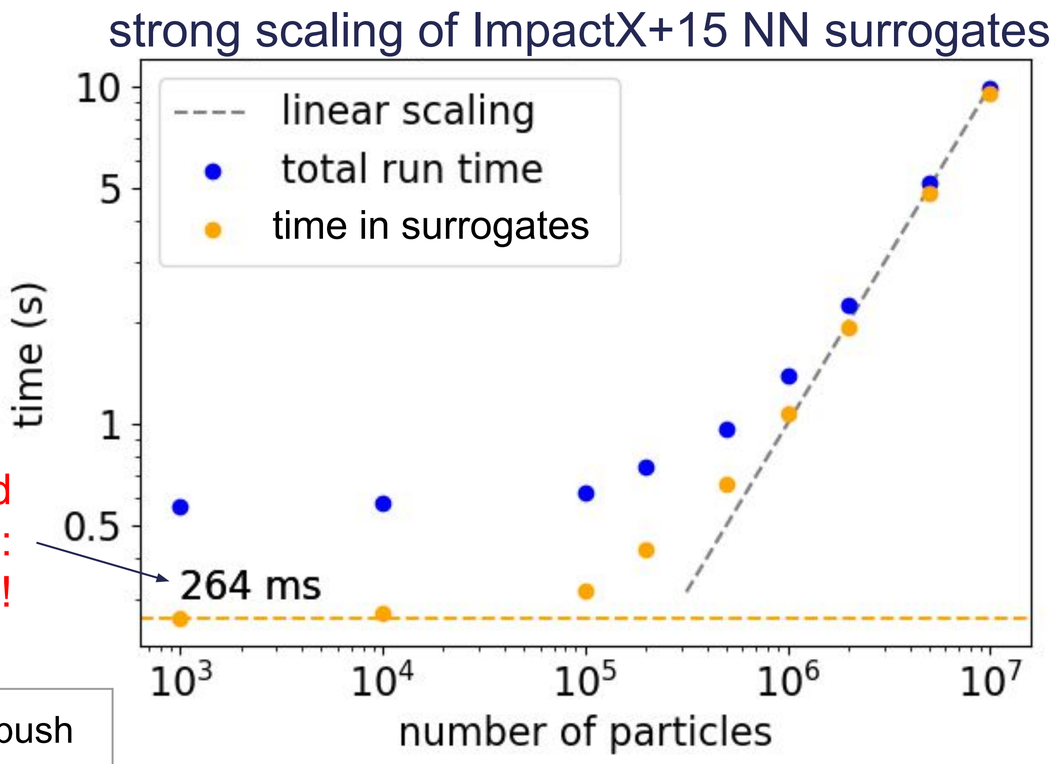
Black: WarpX reference
Red: ImpactX+surrogate



Relative errors in beam moments

	stage 1	stage 2	stage 15
σ_x	0.12%	1.8%	3.2%
σ_{px}	0.54%	2.1%	2.8%
ε_x	0.43%	0.38%	0.39%
σ_y	0.03%	1.5%	1.2%
σ_{py}	0.3%	1.9%	3.2%
ε_y	0.3%	0.44%	2.1%

Modeling + ML Inference are fully GPU accelerated, approaches linear strong scaling in number of particles



ImpactX with WarpX-trained surrogates: 10 GPU sec for 15 stages

10^7 particles	Time (ms)	% of push
Stage 15 Push	495	100
Inference	477	96.4
Data Preparation	18	3.6

ImpactX with WarpX-trained surrogates:
2-4 simulations / second!

10^3 particles	Time (ms)	% of push
Stage 15 Push	2.77	100
Inference	0.77	27.8
Data Preparation	2.00	72.2

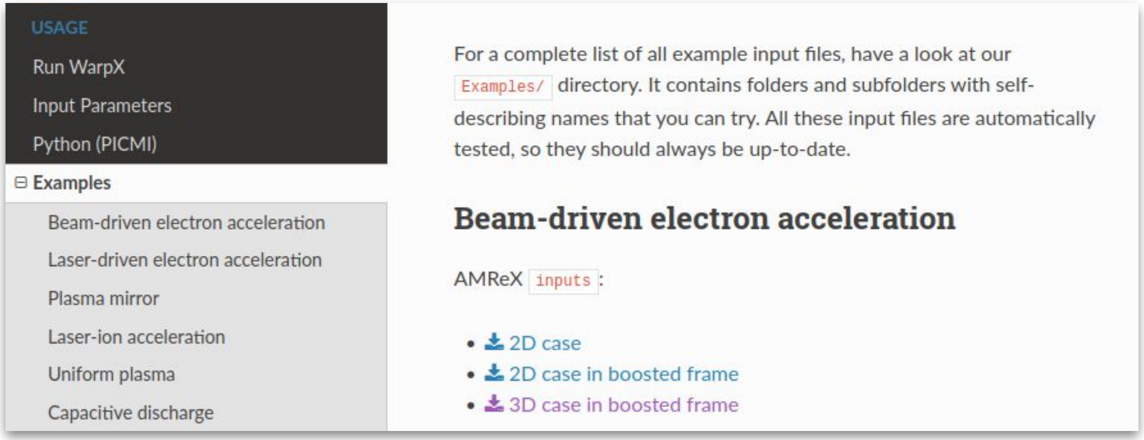
GPU inference time: 63ns / particle / stage
ImpactX tracking >1M particles

We Develop Openly with the Community

BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT

Online Documentation:

warpx|hipace|impactx.readthedocs.io


USAGE
Run WarpX
Input Parameters
Python (PICMI)
⊟ Examples
Beam-driven electron acceleration
Laser-driven electron acceleration
Plasma mirror
Laser-ion acceleration
Uniform plasma
Capacitive discharge

For a complete list of all example input files, have a look at our `Examples/` directory. It contains folders and subfolders with self-describing names that you can try. All these input files are automatically tested, so they should always be up-to-date.

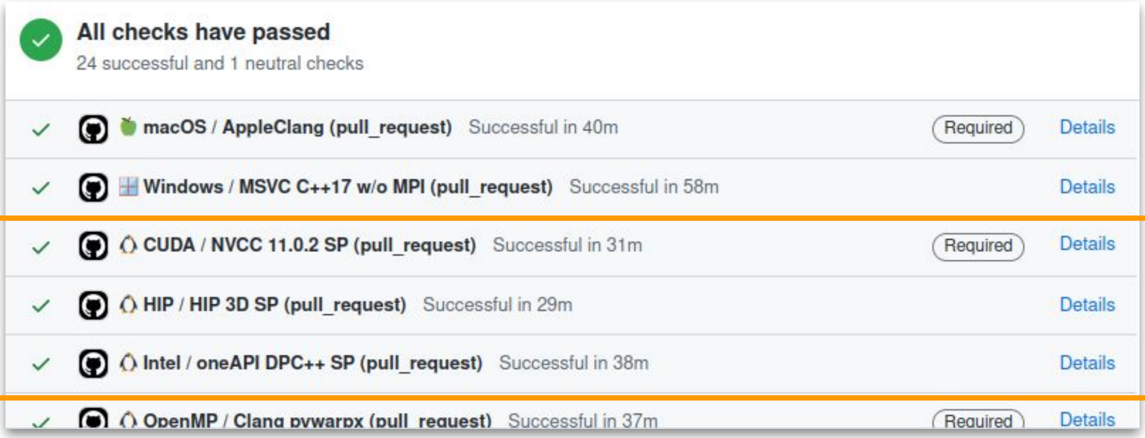
Beam-driven electron acceleration

AMReX `inputs`:

- 2D case
- 2D case in boosted frame
- 3D case in boosted frame

Open-Source Development & Benchmarks:

github.com/ECP-WarpX


All checks have passed
24 successful and 1 neutral checks

macOS / AppleClang (pull_request)	Successful in 40m	Required	Details
Windows / MSVC C++17 w/o MPI (pull_request)	Successful in 58m		Details
CUDA / NVCC 11.0.2 SP (pull_request)	Successful in 31m	Required	Details
HIP / HIP 3D SP (pull_request)	Successful in 29m		Details
Intel / oneAPI DPC++ SP (pull_request)	Successful in 38m		Details
OpenMP / Clang pywarpx (pull_request)	Successful in 37m	Required	Details

230 physics benchmarks *run on every code change* of WarpX
34 physics benchmarks for ImpactX

Rapid and easy installation on any platform:

conda install
-c conda-forge warpx

spack install warpx
spack install py-warpx

cmake -S . -B build
cmake --build build --target install

python3 -m pip install .

brew tap ecp-warpx/warpx
brew install warpx

module load warpx
module load py-warpx

BLAST Codes: Easy to Use, Extend, Tested and Documented

```python
from impactx import ImpactX, elements

sim = ImpactX()
# ...

# design the accelerator lattice)
ns = 25  # number of slices per ds in the element
fodo = [
    elements.Drift(ds=0.25, nslice=ns),
    elements.Quad(ds=1.0, k=1.0, nslice=ns),
    elements.Drift(ds=0.5, nslice=ns),
    elements.Quad(ds=1.0, k=-1.0, nslice=ns),
    elements.Drift(ds=0.25, nslice=ns),
    monitor,
]
# assign a fodo segment
sim.lattice.extend(fodo)

# run simulation
sim.evolve()
```

💡 **Same Script**
CPU/GPU & multi-node

Example: ImpactX FODO Cell Lattice

INSTALLATION
- Users
- Developers
- HPC

USAGE
- Run ImpactX
- Parameters: Python
- Parameters: Inputs File

⊟ Examples
- FODO Cell
- Chicane
- Constant Focusing Channel
- Constant Focusing Channel with Space Charge
- Expanding Beam in Free Space
- Kurth Distribution in a Periodic Focusing Channel
- Kurth Distribution in a Periodic Focusing Channel with Space Charge
- Acceleration by RF Cavities
- FODO Cell with RF
- FODO Cell, Chromatic
- Chain of thin multipoles
- A nonlinear focusing channel based on the IOTA nonlinear lens
- The "bare" linear lattice of the Fermilab IOTA storage ring

🏠 / Examples — Edit on GitHub

Examples

This section allows you to **download input files** that correspond to different physical situations or test different code features.

- FODO Cell
- Chicane
- Constant Focusing Channel
- Constant Focusing Channel with Space Charge
- Expanding Beam in Free Space
- Kurth Distribution in a Periodic Focusing Channel
- Kurth Distribution in a Periodic Focusing Channel with Space Charge
- Acceleration by RF Cavities
- FODO Cell with RF
- FODO Cell, Chromatic
- Chain of thin multipoles
- A nonlinear focusing channel based on the IOTA nonlinear lens
- The "bare" linear lattice of the Fermilab IOTA storage ring
- Solenoid channel
- Drift using a Pole-Face Rotation
- Soft-edge solenoid
- Soft-Edge Quadrupole
- Positron Channel
- Cyclotron
- Combined Function Bend
- Ballistic Compression Using a Short RF Element
- Test of a Transverse Kicker

EXASCALE COMPUTING PROJECT   BERKELEY LAB   LDRD

github.com/**ECP-WarpX/impactx**

The HEP Team

BERKELEY LAB — Accelerator modeling [PI: JL Vay]

AMReX, I/Os [coPI: A. Almgren]

Fermilab — Conventional accelerator modeling [coPI: E. Stern]

UCLA — Plasma accelerator modeling [coPI: W. Mori]

SLAC NATIONAL ACCELERATOR LABORATORY — Machine learning for accelerators [coPI: A. Edelen]

Argonne NATIONAL LABORATORY — Optimization (libEnsemble, POPAS) [coPI: J. Larson]

OAK RIDGE National Laboratory — ADIOS I/Os [coPI: N. Podhorszki]

SciDAC
Scientific Discovery through Advanced Computing

FES

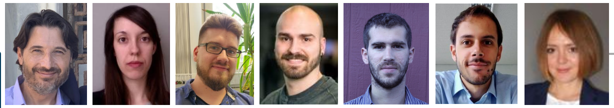
Kinetic IFE Simulations at Multiscale with Exascale Technologies

KISMET

WARPX

Team

(lead) BERKELEY LAB



PIC algorithms & WarpX Code, Plasma Modeling

FASTMATH — AMReX, Solvers

Lawrence Livermore National Laboratory

Target Surface & Hotspot Physics, Implicit Solvers

UNIVERSITY of ROCHESTER

Low Density Plasma Physics, Laser Absorption & Transport

kitware

RAPIDS — Data Visualization & Analysis

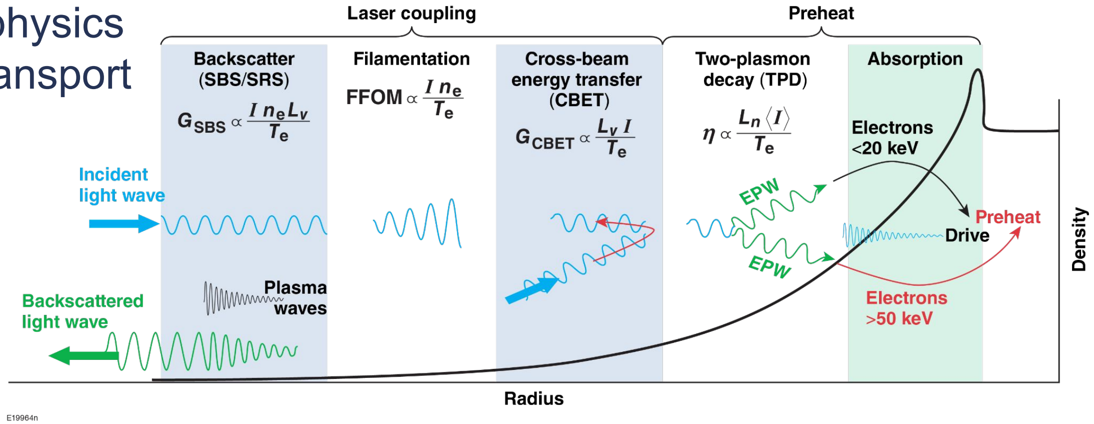
Two Computational Thrusts

a) Particle-In-Cell algorithms & WarpX
b) Scalable data visualization & analysis

Thrusts

Four Physics Thrusts (aligned with 2023 IFE BRN)

a) low-density plasma physics
b) laser absorption & transport
c) proton-driven FI
d) hotspot physics



Augmenting & GPU-accelerating PIC Simulations & ML Models

GPU Workflows are blazingly fast

- PIC simulations
- Machine learning

Can we augment & accelerate on-GPU PIC simulations with on-GPU ML models?

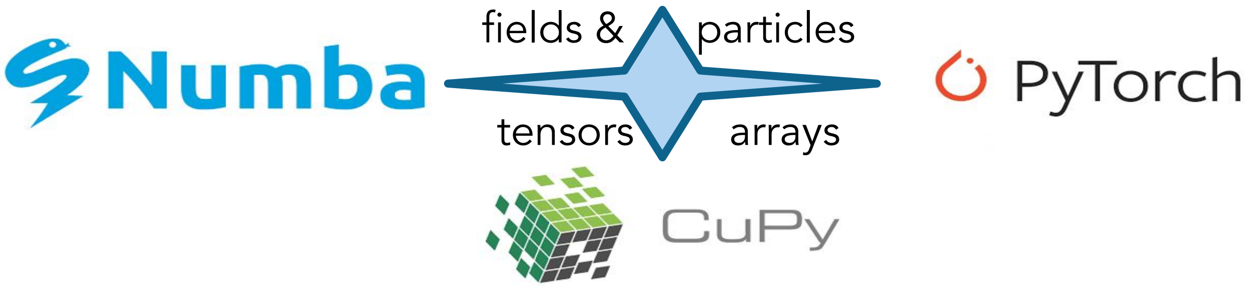
```python
from pywarpx import picmi
import torch
# ...

# iterate all density boxes
for i in rho_device:
    rho = torch.as_tensor(
        rho_device.array(i),
        device="cuda")

    # apply ML in-memory
    with torch.no_grad():
        surrogate_model(rho)
```

Compatible ecosystem between:

BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT



Persistent GPU data placement

- read+write access, no CPU transfer

Cross-Ecosystem, In Situ Coupling:
Consortium for Python Data API Standards *data-apis.org*

Modular Software Architecture

BLAST
BEAM PLASMA & ACCELERATOR SIMULATION TOOLKIT

Python: Modules, PICMI interface, Workflows

WarpX
full PIC, LPA/LPI

ImpactX
accelerator lattice design

…

HiPACE++
quasi-static, PWFA

ARTEMIS
microelectronics

ML Frameworks
PyTorch, Tensorflow, …

pyAMReX

PICSAR
QED Modules

ABLASTR: shared PIC

AMReX
Containers, Communication, Portability, Utilities

openPMD
diagnostics

Math
FFTs, lin. alg.

CUDA, OpenMP, SYCL, HIP

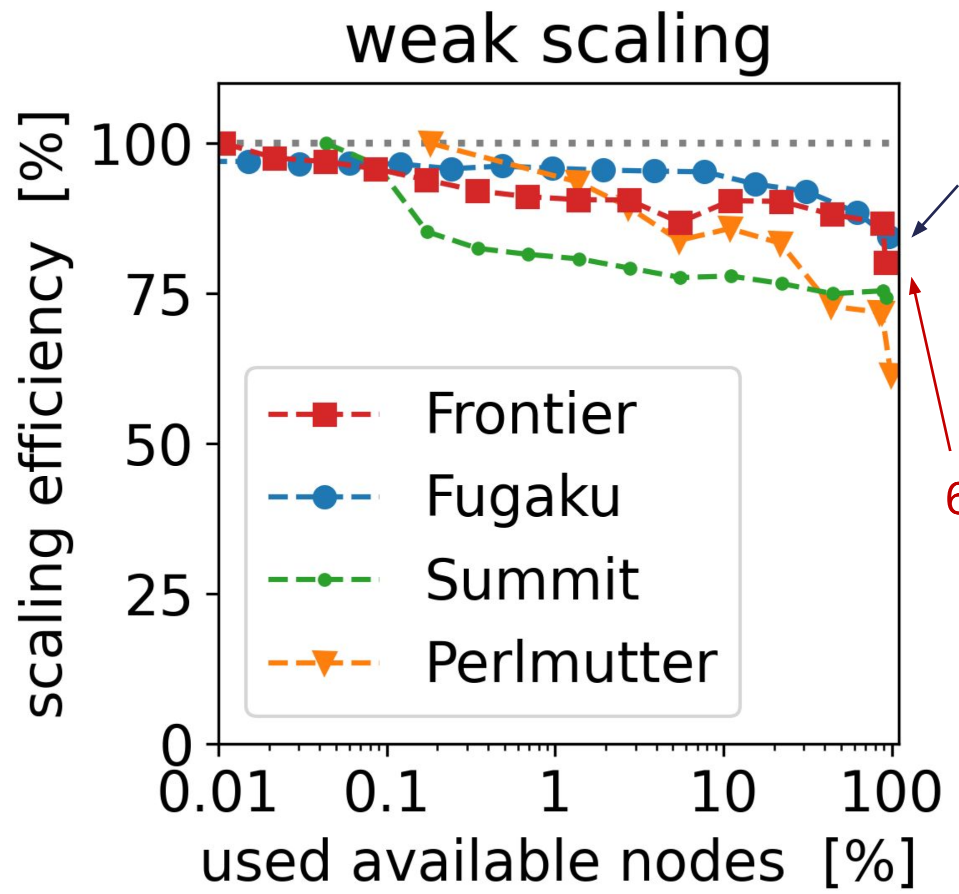
MPI

Desktop to HPC

WarpX Scales to the World's Largest HPCs

April-July 2022: WarpX on **world's largest HPCs**

L. Fedeli, A. Huebl et al., *Gordon Bell Prize Winner* at SC'22, 2022



7,299,072 CPU Cores

68,608 GPUs of *First Exascale* Machine

Note: Perlmutter & Frontier were pre-acceptance measurements!

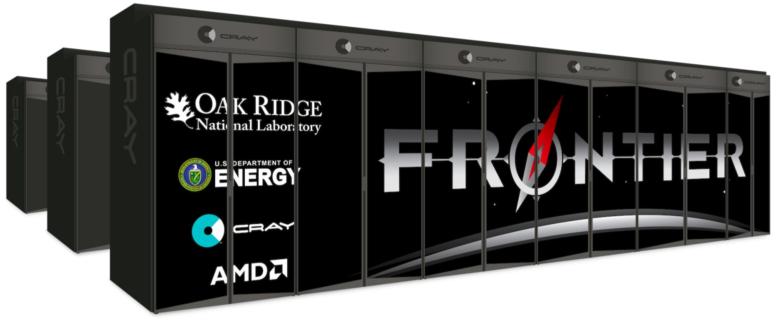


Figure-of-Merit: weighted updates / sec

Date	Code	Machine	N_c/Node	Nodes	FOM
3/19	Warp	Cori	0.4e7	6 625	2.2e10
3/19	WarpX	Cori	0.4e7	6 625	1.0e11
6/19	WarpX	Summit	2.8e7	1 000	7.8e11
9/19	WarpX	Summit	2.3e7	2 560	6.8e11
1/20	WarpX	Summit	2.3e7	2 560	1.0e12
2/20	WarpX	Summit	2.5e7	4 263	1.2e12
6/20	WarpX	Summit	2.0e7	4 263	1.4e12
7/20	WarpX	Summit	2.0e8	4 263	2.5e12
3/21	WarpX	Summit	2.0e8	4 263	2.9e12
6/21	WarpX	Summit	2.0e8	4 263	2.7e12
7/21	WarpX	Perlmutter	2.7e8	960	1.1e12
12/21	WarpX	Summit	2.0e8	4 263	3.3e12
4/22	WarpX	Perlmutter	4.0e8	928	1.0e12
4/22	WarpX	Perlmutter†	4.0e8	928	1.4e12
4/22	WarpX	Summit	2.0e8	4 263	3.4e12
4/22	WarpX	Fugaku†	3.1e6	98 304	8.1e12
6/22	WarpX	Perlmutter	4.4e8	1 088	1.0e12
7/22	WarpX	Fugaku	3.1e6	98 304	2.2e12
7/22	WarpX	Fugaku†	3.1e6	152 064	9.3e12
7/22	WarpX	Frontier	8.1e8	8 576	1.1e13

110x

500x

GPUs enable kinetic simulations of relativistic laser-matter interaction with complex targets

Scientific Achievements

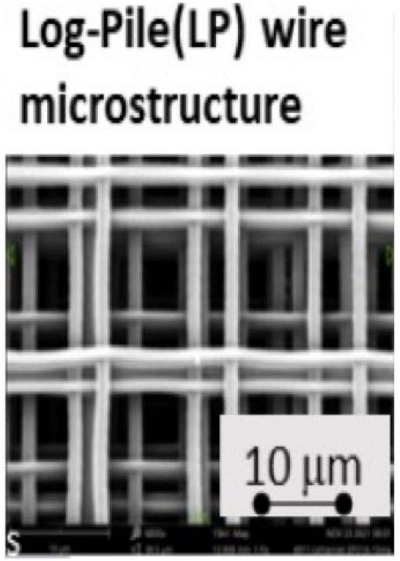
- Omega-EP experiments with log-pile targets yield unprecedented coupling efficiency and max. ion energy
- hemispherical targets promise focusing laser-driven ion beams for ion Fast Ignition IFE

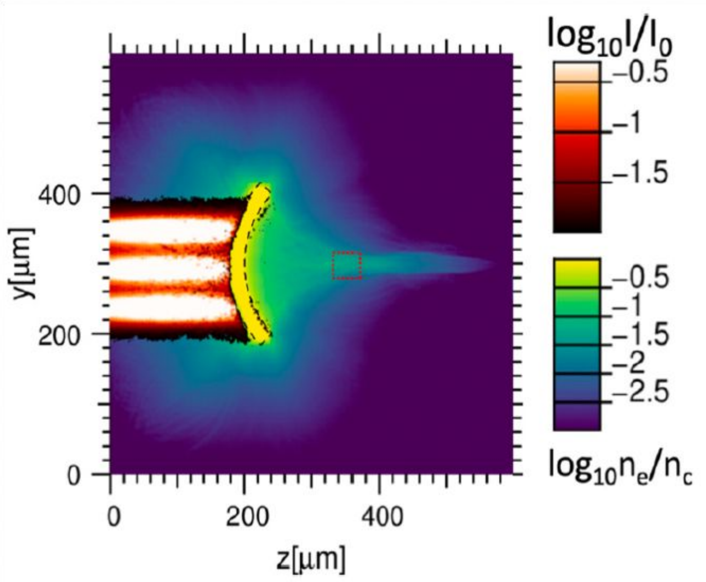
Significance and Impact

- Cost and feasibility of fast ignition of ICF targets with energetic ions depends directly on laser-to-ion coupling efficiency
- Complex target geometries require modeling at scale – enabled by GPU based explicit particle-in-cell

Technical Approach

- WarpX performance on GPUs slashes time to solution by 100x compared to CPU-based PSC
- Livermore Computing Grand Challenge on Tuolumne





Complex target geometries used in recent experiments on Omega-EP and NIF-ARC require sophisticated computer models at realistic scale; left: log-pile; right: focusing hemispherical target for relativistic laser-driven ion acceleration

PI(s)/Facility Lead(s): Jean-Luc Vay (FES), Ann Almgren (ASCR)
Collaborating Institutions: LLNL, U. Rochester (LLE), Kitware
ASCR Program: SciDAC
ASCR PM: Dr. Marco Fornari
Publication(s) for this work: R. Lehe, M. Haseeb, J. Angus, D. P. Grote, R. E. Groenwald, A. Formenti, A. Huebl, J. R. Deslippe, J.-L. Vay, "An Efficient GPU Parallelization Strategy for Binary Collisions in Particle-In-Cell Plasma Simulations", Proceedings of the 2025 Platform for Advanced Scientific Computing Conference (PASC '25).

SciDAC
Scientific Discovery through Advanced Computing