# *In situ* Visualization and Analysis of Ion Accelerator Simulations using Warp and VisIt

Oliver Rübel, Burlen Loring, Jean-Luc Vay, David P. Grote, Remi Lehe, Stepan Bulanov, Henri Vincenti, and E .Wes Bethel

June 9, 2016

## Acknowledgment

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, in part through the grant "Scalable Analysis Methods and In Situ Infrastructure for Extreme Scale Knowledge Discovery," program manager Dr. Lucy Nowell. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

**Abstract**

The generation of short pulses of ion beams through the interaction of an intense laser with a plasma sheath offers the possibility of compact and cheaper ion sources for many applications; from fast ignition and radiography of dense targets to hadron therapy and injection into conventional accelerators. To enable the efficient analysis of large-scale, high-fidelity particle accelerator simulations using the Warp simulation suite, we introduce the **W**arp ***I****n situ* **V**isualization Toolkit (WarpIV). WarpIV integrates state-of-the-art *in situ* visualization and analysis using VisIt with Warp, supports management and control of complex *in situ* visualization and analysis workflows, and implements integrated analytics to facilitate query and feature-based data analytics and efficient large-scale data analysis. WarpIV enables for the first time distributed parallel, *in situ* visualization of the full simulation data using high-performance compute resources as the data is being generated by Warp. We describe the application of WarpIV to study and compare large 2D and 3D ion accelerator simulations, demonstrating significant differences in the acceleration process in 2D and 3D simulations. WarpIV is available to the public via `https://bitbucket.org/berkeleylab/warpiv` .

# *In situ* Visualization and Analysis of Ion Accelerator Simulations using Warp and VisIt

Oliver Rübel*, Burlen Loring*, Jean-Luc Vay, David P. Grote, Remi Lehe, Stepan Bulanov, Henri Vincenti, and E .Wes Bethel

**Abstract**—The generation of short pulses of ion beams through the interaction of an intense laser with a plasma sheath offers the possibility of compact and cheaper ion sources for many applications; from fast ignition and radiography of dense targets to hadron therapy and injection into conventional accelerators. To enable the efficient analysis of large-scale, high-fidelity particle accelerator simulations using the Warp simulation suite, we introduce the **W**arp ***I**n situ* **V**isualization Toolkit (WarpIV). WarpIV integrates state-of-the-art *in situ* visualization and analysis using VisIt with Warp, supports management and control of complex *in situ* visualization and analysis workflows, and implements integrated analytics to facilitate query and feature-based data analytics and efficient large-scale data analysis. WarpIV enables for the first time distributed parallel, *in situ* visualization of the full simulation data using high-performance compute resources as the data is being generated by Warp. We describe the application of WarpIV to study and compare large 2D and 3D ion accelerator simulations, demonstrating significant differences in the acceleration process in 2D and 3D simulations. WarpIV is available to the public via https://bitbucket.org/berkeleylab/warpiv .

**Index Terms**—Data visualization, *in situ*, particle rendering, data analysis, nuclear and plasma sciences, particle beams, ion beams.

## 1 Introduction

The rapid development of laser technologies has made the interaction of high intensity laser pulses with matter a major focus of theoretical and experimental research. This interaction has attracted a lot of attention since it can potentially revolutionize a number of applications by making available compact sources of high energy beams of electrons and ions, and high frequency radiation. In particular, the beams of ions accelerated to the energies from several MeV to hundreds of MeV or even GeV are expected to be available from laser plasma interaction [1], [2]. Several lasers able to achieve 100 MeV levels of proton energy are already in operation and many more facilities are being built or planned. These ion beams have a wide range of applications, such as, radiography, deflectometry, cancer therapy, injection into conventional accelerators, fast ignition, isochoric heating of matter, positron emission tomography, nuclear physics and others (See [1], [2] and Refs. therein).

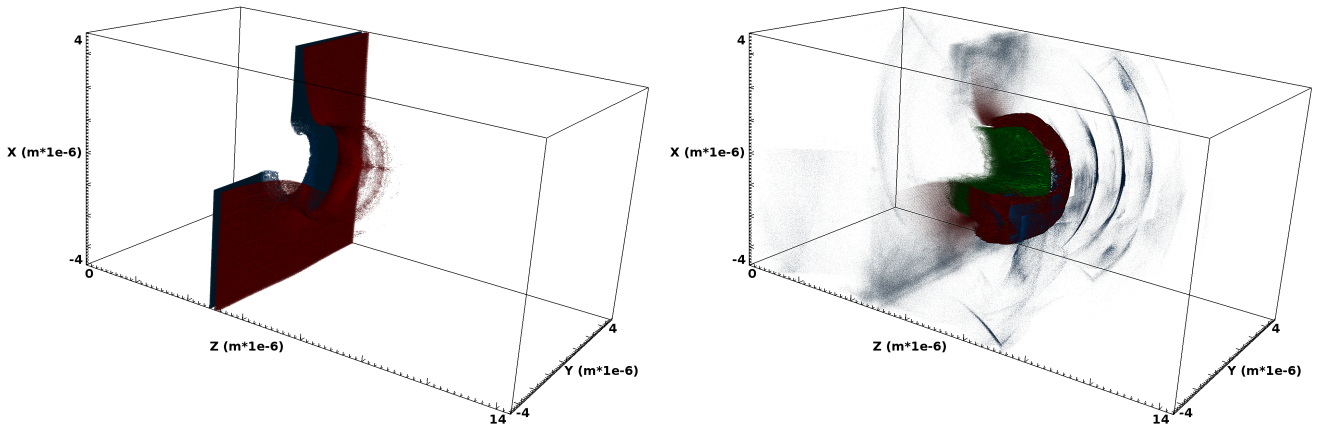
Accelerator scientists at the Berkeley Lab Laser Accelerator (BELLA) center are using Warp (Sec. 2.1)—an advanced particle-in-cell (PIC) simulation framework—for computational modeling, design, and study of laser-based ion and electron acceleration (Sec. 2.2) to improve understanding of the complex acceleration processes and to optimize and develop new accelerators. PIC [3] is a technique that uses a combination of particles and meshes to solve Vlasov-type equations and is widely used in computational studies where the modeling of kinetic effects is required.

Accurate three-dimensional modeling of laser-based particle accelerators using PIC requires: **i)** high-resolution meshes with billions of cells to resolve the high plasma frequencies, **ii)** hundreds of millions and in many cases billions to trillions of particles to model the plasma, and **iii)** $10^4$ to $10^6$ timesteps to accurately resolve the laser and particle motions and interactions. At the same time, often only a small fraction of the particles form particle features of interest, such as, a beam. Understanding of the complex acceleration processes requires visualization and analysis at high temporal and spatial resolutions and the ability to study the relationships and interactions between multiple particle types and fields. The large data sizes and need for high temporal and spatial resolution are—in conjunction with recent lag in I/O bandwidth and storage capacity relative to growing computational capacity—making it increasingly prohibitive to collect all the data required for *post hoc* analysis in persistent storage.

To address this critical challenge, we introduce WarpIV—an advanced *in situ* visualization and analysis toolkit for Warp—enabling the efficient, parallel visualization and analysis of simulation data while it is being generated. The goal of WarpIV is to enable particle-in-cell (PIC) simulations using Warp to: **i)** more effectively utilize high-performance computing resources, **ii)** perform analysis at high temporal resolution, and to **iii)** enable knowledge discovery from large-scale simulations. To achieve this goal, we are implementing a three-fold strategy in WarpIV. First, we couple general-purpose, state-of-the-art *in situ* visualization technology using VisIt [4], [5] with Warp to make new advanced analysis capabilities accessible to Warp and to enable *in situ* processing of the complete data in parallel (Sec. 3.1). Second,

- *B. Loring*, O. Rübel*, and E. Wes Bethel are with the Data Visualization and Analytics Group of the Computational Research Division, Lawrence Berkeley National Laboratory (LBNL), 1 Cyclotron Road, Berkeley, CA, 94720. E-mail: bloring@lbl.gov*
- *J. L. Vay, D. Grote, R. Lehe, S. Bulanov, and H. Vincenti are with the Accelerator Technology & Applied Physics Division, LBNL.*
- *D. Grote is with the Fusion Energy Sciences Program, Lawrence Livermore National Laboratory.*

*Manuscript received September 1, 2015; revised February 2, 2016*
**These authors contributed equally to this work.*




(a) Particle density (b) Kinetic Energy

Fig. 1. Visualization of **(a)** the particle density and **(b)** the kinetic energy of all particles in a 3D simulation of a laser ion accelerator model generated *in situ* using WarpIV. Both visualizations show the protons (red), carbon (green), and electrons (blue) at an intermediate timestep $it = 4105$, $time = 5.53716e-14s$ after the laser has hit the target foil. When comparing **(a)** and **(b)** we observe that the bubble has low density but high kinetic energy compared to the foil.

we support a broad range of *in situ* operational modes and workflows, enabling scientific discovery via: **i)** batch, production *in situ* runs, **ii)** on-demand *in situ* monitoring of simulations, **iii)** interactive *in situ* data exploration, and **iv)** *in situ* debugging and development via the interactive simulation prompt (Sec. 3.2). Third, WarpIV implements a series of integrated analytics to **i)** enable *in situ* query and feature detection, **ii)** to optimize the analysis of spatial distributions of particle quantities, and **iii)** to facilitate the efficient visualization of staggered "Yee" grids (Sec. 3.3). WarpIV has been designed with extensibility in mind to facilitate the fast integration of new Warp simulation models and *in situ* visualization, analysis and I/O methods with WarpIV (Sec. 3.4). We demonstrate the application of WarpIV to study advanced ion accelerator models in 2D and 3D (Sec. 4).

## 2 Background

Before discussing our methods in detail, we provide in the following a brief introduction to Warp (Sec. 2.1), the fundamental concept of laser ion acceleration (Sec. 2.2), and an overview of methods for *in situ* visualization (Sec. 2.3).

### 2.1 Warp

Warp [6], [7] is an advanced particle-in-cell (PIC) simulation framework that supports a broad range of electrostatic and electromagnetic field solvers; a variety of geometries (3-D x,y,z; 2-D r,z; 2-D x,y); the use of "warped" coordinates for bent beam lines (hence the name Warp); Adaptive Mesh Refinement (AMR); models for particle interactions with gas and walls; Lorentz-boosted frame solver, and many other advanced features. Warp has a broad range of applications, including modeling of laser-based electron and ion acceleration, non-neutral plasmas in traps, or stray "electron clouds" in accelerators.

Warp's basic architecture combines efficient, compiled Fortran routines for large-scale numerical operations with a modern, object-oriented Python upper layer and user interface. The Fortran routines are wrapped using the Forthon [8] library, making all major code quantities accessible to both Fortran and Python code and enabling compiled subroutines to be called from Python. Warp may be thought of as a set of "physics extensions to Python." In fact, input files to Warp are Python programs that flexibly combine Warps efficient solvers, models, and diagnostics to define advanced kinetic simulations of particle beams and plasmas. This basic design empowers users, facilitates scripting and control of runs, and facilitates the flexible extension of Warp.

Basic diagnostics and plots are typically implemented in Warp on the Python level to ease reuse and customization. Warp primarily uses Gist [9] for two-dimensional graphics and OpenDX [10] for three-dimensional visualization. While this approach has been successful, the method is inherently serial. This means that all data needed for plotting must be gathered to a single compute core resulting in **i)** increasingly large communication and computation overheads as simulation sizes grow and **ii)** dramatically limiting the amount of data that can be visualized, in particular as the amount of available memory relative to compute continues to decrease. Also, the legacy OpenDX software is no longer supported.

### 2.2 Ion Accelerator Modeling

There are several ion acceleration mechanisms that are being discussed in the literature from both theoretical and experimental points of view. The basic ones are **i)** Target Normal Sheath Acceleration (TNSA), **ii)** Coulomb Explosion (CE), **iii)** Radiation Pressure Acceleration (RPA), and **iv)** Magnetic Vortex Acceleration (MVA). Also several mechanism that are the combinations of the basic ones are being discussed: Break-out-Afterburner (BOA), Directed Coulomb explosion (DCE). Another mechanism of laser ion acceleration being discussed in the literature is proton acceleration by a laser generated shock wave in a near critical density plasma. For the most part, up to now the experiments performed used conditions in the TNSA regime. The highest energy ions generated experimentally were generated by the LANL laser in the BOA regime. There are several results that give an experimental indication of the RPA and MVA regimes.

However, most of the experiments demonstrate exponentially decaying spectra, while most of the applications require monoenergetic ion beams with highly controllable maximum energy and energy spread. That is why a mechanism is required that can produce a spectrum with a peak at high energies in a controllable way, as well as allow for the adjustment of the maximum ion energy.

Here we consider the laser ion acceleration from ultra-thin foil targets in the Directed Coulomb Explosion regime [11], [12]. The most efficient realization of this regime is achieved with ultra-thin solid density two-layer (high Z/low Z, carbon/hydrogen) foils, which allows the production of quasi monoenergetic proton beams applicable to a variety of applications. In this case, the foil is at first accelerated by the radiation pressure of the laser pulse, then, as the electrons are expelled from the irradiated spot (see Fig. 1, blue), the remaining ion core begins to explode due to the excess of positive charge. Since the ion core is moving, the Coulomb Explosion in the lab frame produces a cloud of carbon ions (see Fig. 1, green) expanding predominantly in the direction of the laser pulse propagation. The protons (see Fig. 1, red) are accelerated at the front of this expanding cloud by the charge separation field. The effectiveness of acceleration depends on the laser pulse properties as well as on the target properties. For increasingly thin foils, both the Coulomb Explosion and the Radiation Pressure should become weaker. It is due to the fact that the Coulomb Explosion field depends on the total charge in the irradiated spot, and the increasing transparency of the foil reduces the effectiveness of the RPA. Correspondingly, thicker foils produce high Coulomb Explosion fields but are harder to accelerate by the radiation pressure as well as to remove a sizable amount of electrons from the irradiated spot, which ensures the Coulomb Explosion to occur at all.

### 2.3 *In situ* Visualization and Analytics

Data analytics and visualization are processes that enable scientific knowledge discovery. Data analytics describes the transformation of data into an information-rich form via algorithms to promote better understanding. Visualization is the transformation of data into images to facilitate visual understanding.

In practice, visualization and analysis are performed in conjunction with the generation of data (e.g., via simulation or experimentation) in three main modes: i) *post hoc*, ii) *in transit*, and iii) *in situ*. *Post hoc* refers to the processing of data after it has been generated and stored in persistent storage. Using this approach, the visualization is decoupled from the data generation, providing great flexibility but at potentially large cost for data I/O and storage. Most traditional scientific visualization systems support and have been optimized for *post hoc* data processing, e.g., VisIt, ParaView, Matlab, R, IDL, among many others.

*In transit* then refers to the processing of data as part of the I/O transport of the simulation, while the visualization runs in a separate process in a "staging" area. GLEAN, ADIOS, and NESSIE are just three examples for frameworks designed for *in transit* data processing [13]. The "loose," *in transit* coupling of the visualization with the simulation can improve performance and reduce I/O and storage cost by enabling processing of data without having to write the data to disk and hiding latency of the persistent data store when writing data to disk.

Finally, *in situ* (latin for "in position" or "on site") refers to the processing of data in place while it is being generated and the simulation and visualization are running concurrently. This "tight" coupling of the visualization with the simulation enables the visualization to access the data directly in-memory, without the need for costly I/O or network transport. ParaView Catalyst [14] and VisIt libsim [5] are two examples for libraries designed for *in situ* coupling of state-of-the-art visualization systems with simulations. *In situ* and *in transit* visualization and analysis enable the processing of data at higher frequency and resolution than possible using traditional, *post-hoc* approaches and are increasingly becoming an essential tool for scientific discovery.

In practice, *in situ*, *in transit*, and *post hoc* are complementary approaches that together enable more advanced and efficient data analysis and scientific discovery. In fact, the product of *in situ* analysis is in many cases not "just" an image, but often the goal is to generate a reduced, derived data product—e.g., distribution, compressed dataset, descriptions of data features etc.—that can be efficiently processed *in transit* or *post-hoc*. Hybrid analysis workflows combine *in situ*, *in transit*, and *post hoc* analysis with the goal to balance processing, I/O, and storage cost factors and enable more advanced analysis than possible using a single data processing strategy.

## 3 Methods

In the following we describe WarpIV, a novel application that enables high-performance *in situ* visualization and analysis for Warp.

### 3.1 High-performance *In Situ* Visualization for Warp

To enable distributed parallel, *in situ* visualization and analysis of large-scale simulations using Warp, WarpIV couples general-purpose, state-of-the-art *in situ* visualization technology using VisIt [4], [5] with Warp. WarpIV is a Python application and integrates with Warp and VisIt directly using the respective Python APIs. This strategy allowed us to couple Warp and VisIt, without the need to modify Warp itself, while at the same time enabling WarpIV to directly access all simulation data and functions. WarpIV enables Warp to perform scalable visualization and analysis of the full simulation data in place and in parallel, without the need to reduce and collect the data to a single process. WarpIV supports introspection—i.e., we automatically determine, based on the solvers used, which variables are valid and relevant to the visualization—and also exposes to VisIt a series of control commands to enable steering of the execution of the simulation (e.g, via step, stop, and run).

When analyzing data *in situ*, because the simulation and visualization share resources, performance is critical and the visualization needs to be cognizant of the needs of the simulation and limit its impact on the simulation as much as possible. A central cost factor for *in situ* visualization is memory usage. To reduce memory usage and enable long production runs, we added NumPy zero-copy

support to VisIts libsim simulation interface, avoiding the need for additional data copies to convert Warp data arrays to Python lists. We apply the same zero-copy techniques to the integrated data conversions and analyses that take place in the WarpIV layer. Also, *in situ* visualization and analyses workflows, typically run for longer periods of time and perform many more iterations than is common during traditional *post-hoc* analysis. We have profiled and optimized the memory usage of VisIt and WarpIV to avoid memory leaks over time and ensure reliability. We present details of this work in the supplemental material.

Another main cost factor, in addition to memory, is compute time. We have optimized a number of analyses using modern C++ to leverage compiler optimizations such as inlining and auto-vectorization (see Sec. 3.3). In addition, to optimize rendering performance of complex visualizations of transparent scenes commonly used for the study of the three-dimensional structures of particles and fields, we have extended VisIt to support alpha-blend sort-last compositing and implemented an ordered compositing strategy that allows in-place rendering of translucent block-disjoint decomposed datasets. Our strategy and performance analyses are detailed in [15]. Our ordered compositing optimization resulted in a 4× speed up of workstation rendering performance and an 8× speed up of server rendering performance at 512 cores on NERSC's Cray Edison.

### 3.2 Managing *In Situ* Visualization Workflows

A primary function of WarpIV consists in the management and control of the end-to-end, integrated simulation and *in situ* visualization and analysis workflow. WarpIV initializes and coordinates all required processes and components, including Warp; VisIt's compute engine, viewer and, command line interface(CLI); and depending on the mode of operation the simulation prompt and/or VisIt's graphical user interface (GUI). To coordinate all aspects of the workflow, WarpIV defines the main control loop while providing the user with fine-grained control of when which tasks are executed, e.g., advancing the simulation, executing specific visualization and analysis tasks, performing I/O, responding to user controls *et cetera*.

#### 3.2.1 Modes of Operation

WarpIV uses the strategy design pattern [16] to support four main modes of operation—batch, monitoring, interactive, and prompt mode—each of which supports a different approach towards, *in situ* scientific discovery. Fig. 2 provides an overview of the high-level control flow between the main processes in the different operational modes.

In **Batch Mode** WarpIV executes the simulation and *in situ* analysis automatically, without intervention by the user. Batch mode enables production runs in which sets of predefined analytics are executed in conjunction with the simulation based on user-defined conditions. For example, at specific time intervals or based on the detection of specific events, such as, a spike in energy.

In **Interactive Mode** the user controls the simulation directly from VisIt's GUI. Here the user can interactively explore the simulation data and define visualizations. The user can also directly control from VisIt when the simulation should step, run, pause, or terminate. Being able to interactively explore simulation data as it is being generated is a critical tool for scientific discovery and provides users incredible flexibility to explore data as events of interest occur, test hypotheses, and debug, validate and refine new simulations and models.

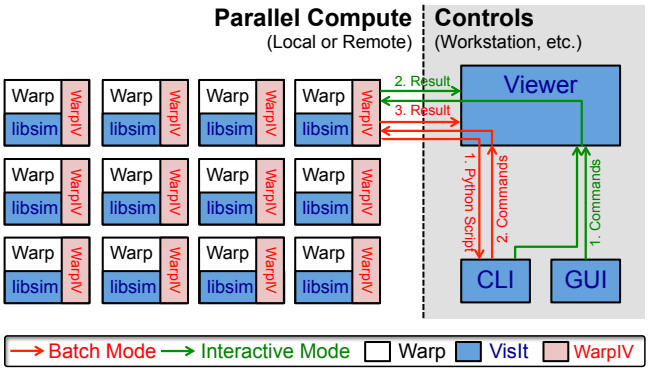



Fig. 2. Illustration showing the main processes (boxes) and high-level control flow (arrows) for batch and interactive mode using WarpIV. Monitoring mode then provides the user with the flexibility to switch between batch and interactive mode and vice versa. Finally, prompt mode adds the ability to control the simulation and visualization directly from the simulation prompt (not shown).

**Monitoring Mode** is a hybrid of the batch and interactive modes enabling users to perform batch style runs while still being able to flexibly connect to the simulation to interactively inspect and explore the simulation data and afterwards resume the simulation run in batch mode. Being able to monitor large-scale simulations is critical, for example, to detect and investigate errors and to enable scientists to make informed decisions about whether to continue, terminate, or modify a simulation run.

Finally, **Prompt Mode** runs inside an interactive Python shell. The simulation data structures can be accessed directly, visualization scripts executed, and WarpIV simulation commands issued. This enables the scientist to programmatically interact with the simulation and visualization and is intended primarily for development and debugging of simulation models.

#### 3.2.2 Automated Analyses via Scripts

A unique feature of WarpIV is its strategy for controlling automated *in situ* visualizations via scripts. This is built around an often overlooked feature in libsim, namely the ability to execute VisIt CLI Python scripts stored in a string via the libsim API. This feature allows us to use Python scripts for configuration and execution of visualization and analysis tasks. In contrast to the rather sparse functionality exposed explicitly via the libsim API, CLI Python scripts expose VisIt's full range of functionality for *in situ* use. In addition, CLI scripts can be generated automatically, simply by recording a user's actions in VisIt's GUI.

During each update WarpIV collects the user-defined visualization scripts to be executed and sends them to the VisIt CLI via the libsim API. The CLI in turn interprets the scripts, sending commands that control visualizations to VisIt's engine or sending simulation commands that control the execution of analyses tasks back to WarpIV (Fig. 2,

red arrows). VisIt's CLI executes incoming scripts asynchronously, each in a separate thread. However, VisIt's CLI Python API is not thread-safe leading to a host of issues if multiple scripts execute simultaneously. Therefore, at each update the set of active scripts are concatenated into a single script prior to execution. This strategy also simplifies synchronization of the visualization and simulation and allows us to instrument the scripts to gather coarse grained run time performance data and to gracefully handle errors in the user-provided code. Because CLI scripting is a VisIt feature, this approach could be used in Fortran or C/C++ based codes as well.

One particularly tricky aspect in the design of WarpIV's script-based visualization controls has been the synchronization of the simulation and VisIt. When using zero-copy data transfers, the simulation must not modify its data structures during visualization operations. However, VisIt CLI scripts are executed asynchronously in a separate process where Python code is translated into VisIt's internal remote procedure calls which trigger actions on the VisIt engine, running in the process shared with the simulation.

To achieve the required synchronization we leverage CLI Python bindings to the GUI's "simulation command" mechanism. These are typically used for interactive control of the simulation. WarpIV implements a "synchronous mode" during which incoming simulation commands are queued. This temporarily prevents the simulation from advancing. Synchronous mode is activated prior to requesting the asynchronous execution of a CLI script. During the concatenation of user provided scripts, an "end-syn" simulation command is added, which when executed gets sent back to WarpIV to indicate that the visualization is finished. In response to the end-syn command WarpIV de-queues and acts upon any pending commands before exiting synchronous mode and resuming normal operation.

In addition to visualization scripts, WarpIV also supports "simulation scripts". Simulation scripts are user provided Python analysis scripts that are executed in parallel on the simulation side. Similar to visualization scripts, simulation scripts are executed by WarpIV at user defined intervals and/or in response to specific conditions or events in the simulation. Simulation scripts enable users to easily incorporate custom visualization, analysis, and I/O methods that do not rely on VisIt.

### 3.3 Integrated Analytics

WarpIV supports a number of integrated data analytics to facilitate *in situ* analysis of advanced accelerator simulations, in particular derived particle species (Sec. 3.3.1), data binning (Sec. 3.3.2), and re-centering of Yee grids (Sec. 3.3.3).

#### *3.3.1 Derived Particle Species*

A central challenge in the analysis of complex particle simulations arises from the fact that while $10^7$ to $10^9$ particles are required for accurate simulation, often only a small fraction of the particles form features of interest, e.g., a particle beam. WarpIV addresses this challenge via the concept of filtered species. Filtered species define custom, derived particle species while exposing to the analysis the same interface as Warp's regular particle species. This concept enables: **i)** flexible *in situ* analysis of particle features, **ii)** analysis and collection of select data subsets of interest at higher temporal frequency, and **iii)** reduces memory and compute cost for subsequent visualization and I/O. Similar to filtered species, WarpIV also supports merged species. Merged species allow multiple particle species to be combined, e.g., to facilitate analysis of the joint distributions of all electrons that may arise from different particle types, such as, electrons from Carbon and Hydrogen in the foil. WarpIV provides a set of customizable, reusable particle filters—including threshold, particle Id, accumulative query, cone, and halo filters—to facilitate common use cases while at the same time allowing users to easily add their own custom filters. Multiple filtered and merged species may also be combined to define more complex species via a sequence of filters.

#### *3.3.2 Data Binning*

Rather than visualizing individual particles, it is in practice often useful to study the distribution of particle quantities in space, such as, the density of particles or average kinetic energy. We call this re-meshing operation "data binning" as particle quantities can be mapped to other axes, such as velocity or momentum, in addition to the spatial dimensions. VisIt supports data binning. However, in VisIt's data binning implementation the output is constructed on a single node. While this works well for small output resolutions, for higher resolutions, and in particular 3D output, this strategy can run out of memory, and encounters performance issues as operations on the results take place in serial.

We address these issues by implementing a truly data-parallel data binning operator in WarpIV. A key feature of our implementation is that the resulting mesh remains distributed allowing for much high resolution 3D output. By implementing data binning in WarpIV we can also take advantage of Warp's internal data structures to enable the fast, parallel computation of high-resolution, spatial distributions of derived particles quantities. In order to deliver the highest possible performance, our algorithm is written in C++ and made accessible in WarpIV via Python bindings. We make these derived grids and quantities directly accessible in VisIt, enabling efficient, parallel visualization of high-resolution, spatial distributions of particle quantities.

#### *3.3.3 Yee Grid Re-centering*

Many of Warp's electromagnetic field solvers use a staggered "Yee" discretization [17] where the electric field components are located on the edge centers and magnetic field components are located on the face centers. To enable *in situ* visualization of these meshes, we developed a C++ extension to WarpIV for fast conversion of Yee-style meshes to node-centered meshes. Our C++ implementation has shown to be one order of magnitude faster than using NumPy broadcasting, and three orders of magnitude faster than the base Python implementation. See supplemental material for details.

### 3.4 Integrating New Simulation Models and Analytics

WarpIV uses a factory pattern design [16] to define simulation models, enabling scientists to create new simulation and *in situ* analysis models in a self-contained fashion,

simply by defining a derived simulation class type. The effort to create a new simulation in WarpIV is low and is comparable to the effort needed to define models for Warp itself. In addition to the simulation model, a user also needs to define and add the *in situ* visualization and analysis scripts to be performed in conjunction with the simulation. As mentioned in Sec. 3.2, WarpIV supports two types of *in situ* scripts: **i)** visualization scripts are shipped to and interpreted by the VisIt CLI (Fig. 2) to create advanced *in situ* visualizations and analytics, whereas **ii)** simulation scripts are executed in parallel directly on the simulation side (Fig. 2, left) and are used to perform I/O and custom python-based visualization and analytics. Users may also customize other behaviors, such as, the advance, finalization, or termination of runs and define policies and triggers for the execution of visualization and simulation scripts by overriding the corresponding control functions. WarpIV provides a convenient command-line interface to assist users with the execution of simulations and scripts.

## 4 Results

In the following we demonstrate the application of WarpIV to analyze and compare a series of ion accelerator simulations, with the goal to study the impact of using $2D$ vs. $3D$ simulation models on the scientific interpretation. We, therefore, designed a simulation campaign of the same laser ion accelerator model as shown in Table 1 in two and three dimensions. The actual simulation box size is $(8\mu m \times 8\mu m \times 14\mu m)$ and the actual duration is $6.2710e^{-14}$ seconds. The temporal resolution is $r_{time} = 6400$ steps and the spatial mesh resolution is $r_{space} = (900 \times 900 \times 1800)$ in 3D and correspondingly $r_{space} = (900 \times 1800)$ in 2D. The 2D simulation contains 251,600 electron, 179,776 carbon, and 71,824 proton particles per timestep and the 3D simulation models the motions of more than 2.1 billion particles at each timestep; 1,069,251,640 electron, 809,557,568 carbon, and 259,694,072 proton particles.

For every simulation we computed a broad range of *in situ* visualizations and analytics, including, **i)** histograms of various quantities, **ii)** basic statistics of various quantities, e.g., mean, standard error, **iii)** filtering of particles to extract and analyze beam particles, **iv)** merging of particle species to define joint particle distributions, **v)** two and three-dimensional binning of the data to compute multi-dimensional histograms and derived spatial statistics, and **vi)** 3D renderings using transparent iso-surfaces of binned quantities. For visualizations of derived data products, such as histograms and 2D data binnings, we typically save the reduced data product for *post hoc* rendering. For complex 3D visualizations, where saving the raw data is prohibitive, we store the images only.

| | |
|---|---|
| Laser wavelength | $800nm$ |
| Laser intensity | $1.3x10^{22}W.cm^{-2}$ |
| Focal spot size | 0.8 microns |
| Laser duration | $27fs$ |
| Carbon layer thickness | $75nm$ |
| Hydrogen layer thickness | $50nm$ |
| Electron density of the carbon layer | $\approx 8 \times 10^{23}cm^{-3}$ |
| Electron density of the hydrogen layer | $\approx 4 \times 10^{22}cm^{-3}$ |
| Polarization of the laser | Y-axis |

TABLE 1
Parameters of the ion accelerator model

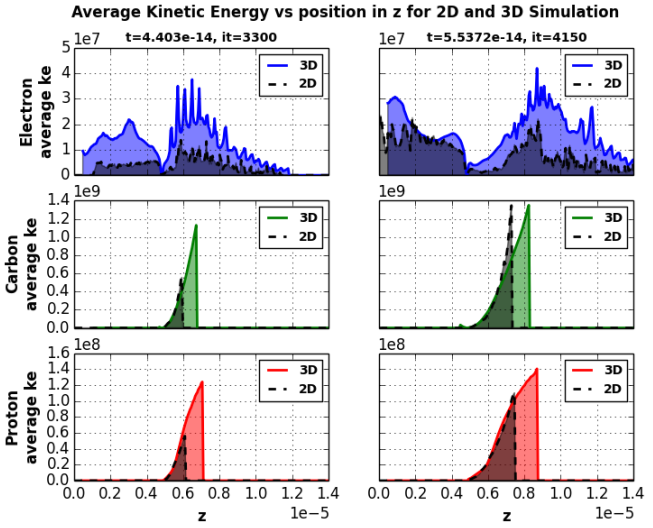



Fig. 3. Average kinetic energy over $z$ for electrons (blue), carbon (green), and protons (red) for the 2D and 3D simulation at $it = 3300$, $time = 4.403e - 14s$ (left) and $it = 4105$, $time = 5.5372e - 14s$ (right).

### 4.1 2D vs. 3D

In what follows we compare the results of 2D and 3D simulation runs. It is well known that the propagation of tightly focused laser pulses in 2D and in 3D is different. This difference should imprint itself into the distributions of energetic electrons and ions. In Fig. 3 the energy of carbon ions and protons averaged at each value of z coordinate is shown for two time steps. The 3D case shows consistently larger energy and the front of the 3D ion distribution is ahead of the 2D one. This means that the ions acquire higher energies in the 3D case. The distribution of ion density, shown in Fig. 4 (right), gives an insight into the difference between the 2D and 3D cases. The figure shows a consistently larger extension of the accelerated ion cloud in the 3D case. This is mainly due to the higher divergence of the tightly focused laser after it passes the focal plane. The higher divergence also contributes to higher carbon ion and proton energies in 3D. This is due to the fact that the more divergent laser pulse is able to evacuate electrons from a larger spot on the foil, boosting the Coulomb field, which accelerates the protons. This effect is similar to using a transversely Super Gaussian pulse instead of a Gaussian pulse, as studied in [12]. As we can see from the comparison of 2D and 3D cases, the 2D one correctly catches the qualitative nature of the interaction, i.e., the creation of the expanding carbon ion cloud and the layer of protons being accelerated by the charge separation field in front of the carbon ion cloud. However the 2D fails to correctly predict the quantitative parameters of the ions, i.e., the spectrum and angular distribution. Thus for the correct analysis of the laser driven ion acceleration as well as interpretation of experimental results, 3D simulations are of paramount importance. A key challenge when comparing simulations is that we often do not know *a priori* when important features occur and what key differences we may

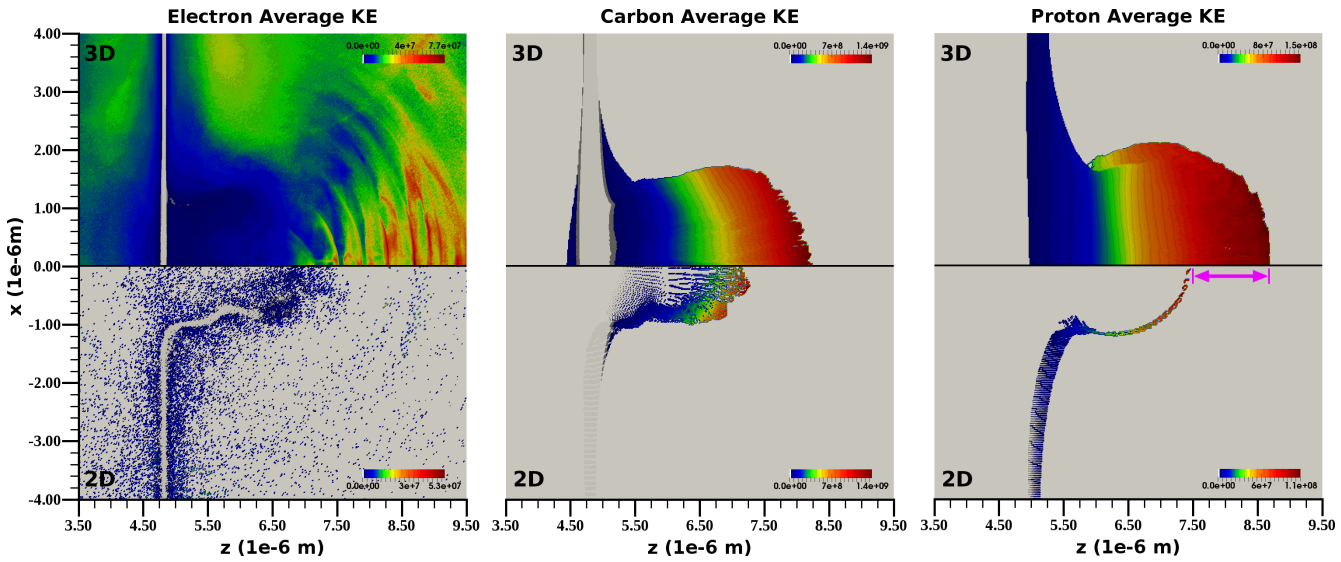



Fig. 4. Average kinetic energy in $x/z$ for the 3D ($x > 0$) and 2D ($x < 0$) simulation. There are a number of immediately apparent differences between the 2D and 3D result. We observe that the acceleration process is quantitatively different in 3D as the proton bubble (right) has propagated significantly further ($\approx 1.24\mu m$, indicated by the magenta arrow) after the same amount of time, here $time = 5.53716e-14s$. We also see a similar discrepancy in the propagation of Carbon (center).

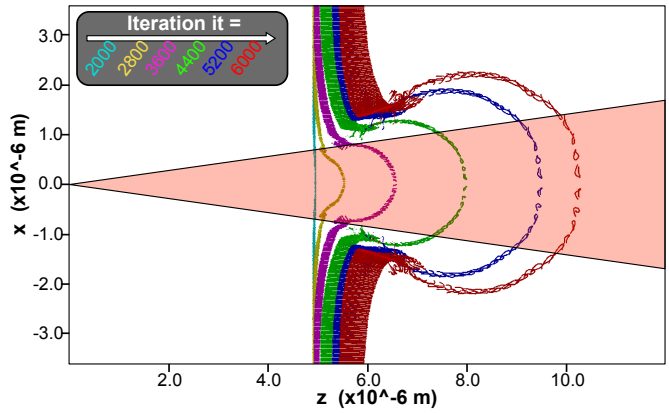



Fig. 5. Illustration of a high-resolution laser ion accelerator simulation showing the evolution of the protons over time, while the color of particles indicates time. Protons that are located within the red cone are selected as being part of the beam at the given time point.

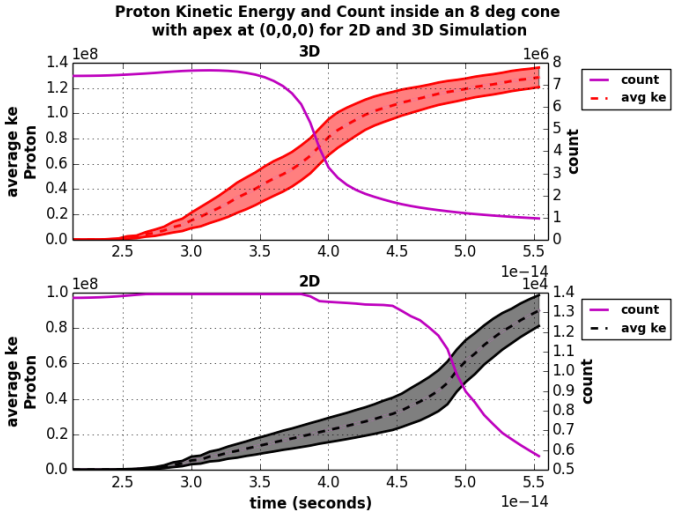



Fig. 6. Analysis of the acceleration over time of the main ion beam in 2D (bottom) and 3D (top) showing the average and standard deviation in kinetic energy as well as the number of particles selected as part of the beam.

find. As such, *in situ* analysis and visualization has been critical to this study as it allowed us to perform visualization and analysis at high temporal frequency at significantly reduced I/O cost (see Sec. 4.3).

### 4.2 Beam Analytics

Ultimately the goal is to produce a high-quality ion beam. Similar to the diagnostic proposed by Bulanov et al. [12], we use a cone-shaped filter with an $8°$ opening angle normal to the target and centered at the origin $(0, 0, 0)$ to extract the set of protons that are part of the beam at a given point in time. I.e, the beam filter selects all particles $p$ that satisfy the following condition:

$$arccos\left(\frac{|p_z|}{\sqrt{p_x^2 + p_y^2 + p_z^2}}\right) <= 8° \cdot \frac{\pi}{180°} \qquad (1)$$

In WarpIV we implement this filter via the concept of filtered species. Fig. 5 illustrates the behavior of the beam filter over the course of the 2D laser ion accelerator simulation.

Fig. 6 shows the mean kinetic energy, and standard deviation in kinetic energy (i.e, energy spread), and count of all protons selected by the beam filter over time. We observe that the beam accelerates continuously over time, while we can identify at least three main phases of acceleration. First, early on, before the laser has hit the target, the protons are largely stationary (i.e., $ke \approx 0$). Second, after collision of the laser with the target we initially observe very strong acceleration while the energy spread (i.e., standard error in $ke$) is fairly large. Third, later on in time, we can see that

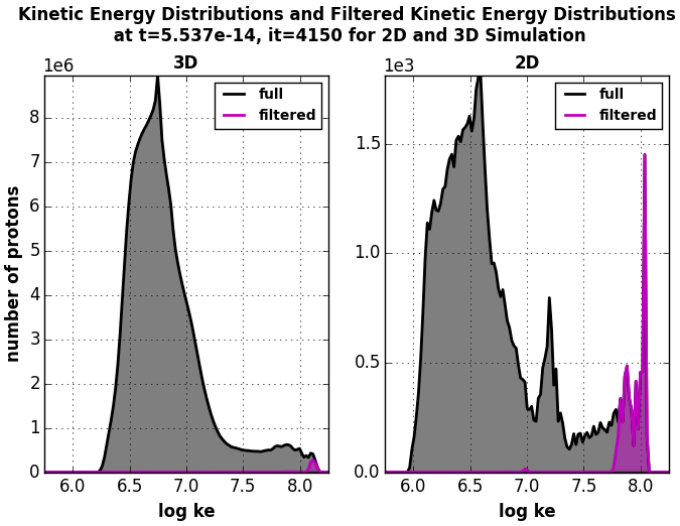



Fig. 7. Histogram of the kinetic energy for all protons (gray) and the subset of protons selected by the beam filter (magenta) at $it = 4150$, $time = 5.537e-14s$ for the 3D laser ion accelerator simulation (left) and the equivalent timestep of the corresponding 2D simulation (right).

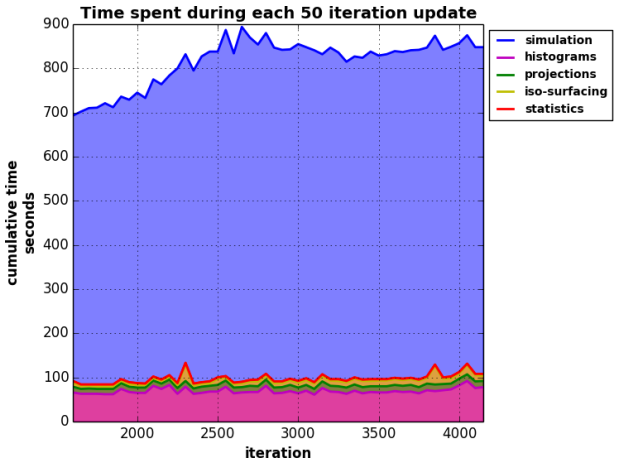



Fig. 8. Run time in seconds by category at 50 iteration updates.

the acceleration slows down while the energy spread of the beam becomes smaller. The reduction in energy spread also coincides with a continues loss in the number of particles that are part of the beam. When comparing the distribution in kinetic energy $ke$ of all protons with the corresponding distribution for the beam protons (see Fig. 7), we then observe that the beam filter, despite being defined solely in physical space, selects the protons with the highest kinetic energies. This behavior is consistent over time once the proton bubble starts to from.

The behavior we have seen here can be explained as follows. When the laser hits the target, it preferentially gives momentum to ions in the laser beam's propagation direction. However, due to the tight focusing, ions receive transverse momentum as they are being pulled out of focus by transverse components of the laser radiation pressure. This defocusing force primarily effects less energetic ions, pulling the low-energy ions out from the main propagation direction, whereas the more energetic ions will not be deflected. Hence, as the ions move away from the target the low-energy ions escape the beam filter resulting in a reduced energy spread and explaining the selectivity of the filter for high-energy ions. Similarly, as we narrow the cone angle we expect the energy spread and charge of the beam to decrease as well. The appropriate cone angle and distance to the target, or equivalently the slit used in lab experiments to extract the beam, therefore, depends on the beam charge and energy-spread requirements of a given application. These kind of *in situ* analytics are important to simulation studies as they help understand the beam acceleration process, validate and determine beam properties, and predict optimal parameters of the laser, plasma, and slit.

When comparing the 3D simulation with the corresponding 2D simulation, we observe, consistent with our observations in Sec. 4.1, that the beam accelerates significantly faster in 3D than in 2D and reaches a higher level of energy. This behavior again illustrates the critical need for high resolution 3D simulations. This example further illustrates the critical need for visualization and analysis at high temporal resolution to enable accurate estimation of the acceleration gradient of the beam, identification of the various phases of acceleration, and definition of appropriate parameters to extract ion beams with the desired energy level and spread.

### 4.3 Performance

The melding of simulation and visualization and analysis codes for *in situ* use holds the promise to enable advanced, high-fidelity analyses and drastically reduced I/O cost. However, performance and scalability of the merged code base is of paramount importance. We instrumented WarpIV with a light weight, coarse grained, profiling API and have used it to study the performance of the $3D$ run presented above. For the performance analysis we grouped similar operations into one of five categories: **i)** "simulation", for the Warp solver computations; **ii)** "histograms", for the computation and I/O of 12 histograms (e.g. Fig 3, 7); **iii)** "projections", for the computation and I/O of 18 $2D$ projections that were used to compare $2D$ and $3D$ wave fronts (e.g. Fig. 4); **iv)** "iso-surfaces", for the $3D$ iso-surface computation, translucent rendering, and image I/O of the Ke and density fields (e.g. Fig. 1); and **v)** "statistics", for a number of simple descriptive statistics computations and I/O (e.g. Fig. 6).

The stack plot in Fig. 8 shows the time spent to complete each category at every 50 iteration simulation update. At each update the visualization, analysis, and I/O operations consumed approximately $11\%$ to $15\%$ of the total time, while the remainder of the time, i.e. $> 85\%$, was used by the simulation. This ratio of simulation vs. *in situ* data analysis is quite reasonable as the overall run time performance and scalability of the simulation has not been impacted drastically, in particular when considering the large range of analyses performed.

In addition to run time, another primary cost factor is I/O. As part of the *in situ* analyses, histograms are written in an ASCII CSV format, projections in VTK compressed binary VTR format, iso-surfaces are rendered and written

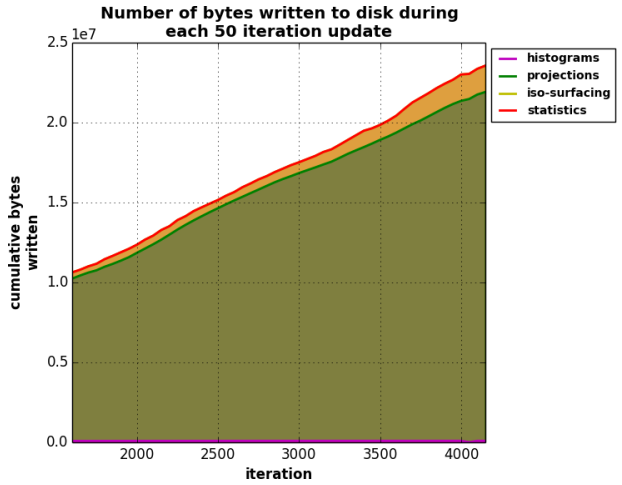



Fig. 9. I/O cost in bytes by category at 50 iteration updates.

| | Category | I/O Cost |
|---|---|---|
| *in situ* | histogram | 4.49 MB |
| | projections | 795.97 MB |
| | iso-surfaces | 40.77 MB |
| | statistics | 3484 b |
| | **total** | **841.23 MB** |
| **hypothetical post process** | particle position and velocity | **3.24 TB** |

TABLE 2
Total I/O cost and comparison with identical post-process analysis.

as PNG images, and statistics are output to standard error stream. Figure 9 shows the cumulative I/O cost in bytes at each 50 iteration update by category. The size of both iso-surface and projection categories grow in time because compression becomes less effective as the simulation evolves and particles begin to fill the entire simulation box. Table 2 shows the total I/O cost of the run by category. The total I/O cost in the run is approximately $841MB$. If one were to make the same analysis *post hoc*, one would need to save the particle positions and velocities for each species at each 50 iteration update, resulting in a total of approximately $3.2TB$ written to disk. Thus, by using *in situ* we have reduced our I/O cost by a factor of more than $4000\times$.

## 5 Conclusion

### 5.1 Lessons Learned

While we have focused on a specific application, many of the design features of WarpIV and lessons learned from this study are relevant more broadly to applications of *in situ* visualization.

**Performance and reliability**: In contrast to the *post hoc* approach, *in situ* visualization and analysis shares resources with the simulation. Also, in the *in situ* approach, many more iterations of the visualization and analysis operations are made. Careful optimization and testing of the *in situ* visualization and analysis are, therefore, critical. For example, the profiling and optimization of the memory usage of VisIt; the addition of numpy and zero-copy support; the C++ optimized Yee-grid re-meshing, data binning operations, and derived particle species; all were critical to enable WarpIV to support large-scale 3D PIC simulations.

**Tuning visualization parameters**: One challenge to making effective use of *in situ* rendering is the loss of interactive adjustment and fine tuning of parameters. Setting rendering parameters in advance is difficult, especially so with translucent rendering techniques such as volume rendering and rendering translucent geometry (Fig. 1). These techniques also exhibit a resolution dependence which confounds the use of a low resolution sample to set up the visualization in advance. For *in situ* applications it is, therefore, often preferable to generate reduced data products for subsequent rendering (or multiple or editable variants of the same visualization) to provide users with the fidelity that *in situ* provides while preserving at least some of the flexibility of *post hoc* analysis and rendering.

**Data compatibility**: The level of compatibility between simulation and visualization library data structures greatly impacts performance. Where data structures have binary compatibility and metadata is equivalent between the two codes, performance will be optimal. However, a mismatch in data structures and/or metadata on the visualization side can lead to potentially expensive data conversions and redundant or unnecessary computations. Given the size and complexity of visualization codes and their web of third-party dependencies (e.g. VisIt+VTK), the amount of effort needed to add support for a new data structure can be prohibitive.

An alternative approach moves a portion of the visualization and analysis work that rely on the unsupported data structure into the coupling layer that sits in between the simulation and the visualization library. Working in this layer can have the additional advantage of reducing data to be processed by the visualization library. We have successfully taken this approach in WarpIV with our specialized data-binning operator that takes advantage of native simulation data layout, with our Yee-grid conversions, and with our "filtered-species".

As *in situ* analysis becomes more-and-more central to simulations, co-design of simulation codes and *in situ* visualization infrastructure becomes desirable to achieve optimal integration and performance.

**Visualization strategies**: The visualization strategy we choose, whether it is *in situ*, *in transit*, *post hoc*, or some hybrid approach, dictates which optimizations, operations, and interactions we can perform. *In situ* processing uniquely enables us to take advantage of the structure and location of the data, to early on reduce data and cost for subsequent processing, and to access data as it is being generated. On the other hand, by separating the visualization and analysis from the simulation, *post hoc* data processing enables us to optimize the visualization independent of the simulation and to perform interactive and complex analyses that may be prohibitive *in situ*. To address these challenges, we plan to expand WarpIV in the future to also support advanced *in transit*, *post hoc*, and hybrid visualization and analysis workflows.

**Flexibility**: The practical use cases for *in situ* visualization are broad, ranging from debugging, interactive exploration, monitoring, to batch runs. In WarpIV we have addressed this critical challenge by enabling users to flexibly

expand WarpIV and to easily choose between a variety of modes of operation; prompt, interactive, monitoring, and batch mode. In our experience, this flexibility has been critical to both the scientific discovery process and to achieve productivity. In particular, the design of new simulation and *in situ* workflow scenarios critically relies on the ability to easily transition between the different use cases and modes.

**Ease-of-use**: *In situ* visualization is ultimately an end-user tool. Ease-of-use is, hence, a critical requirement in the design of *in situ* applications. We have addressed this challenge in WarpIV by enabling users to create *in situ* visualizations using the same Python scripts and tools used for *post hoc* analysis and by providing simple interfaces for implementing new simulation and *in situ* analysis scenarios. To further ease adoption, we plan to also develop libraries of standard, reusable *in situ* visualization and analysis scripts.

### 5.2 Conclusion

We have introduced WarpIV, a novel application for *in situ* visualization and analysis of large-scale particle-in-cell simulations using Warp and VisIt. We have made WarpIV available to the public via https://bitbucket.org/berkeleylab/warpiv. We have described the application of WarpIV to study and compare large 2D and 3D ion accelerator simulations, demonstrating the utility of advanced *in situ* visualization and analytics. We analyzed the run time performance of the *in situ* visualization and analysis operations and the simulation itself and found that the addition of *in situ* analysis introduced a reasonable overhead and did not drastically impact the overall performance and scaling of the code. We also analyzed I/O cost of the *in situ* analysis compared to an identical hypothetical *post process* analysis and found that the use of *in situ* analysis reduced I/O cost by a factor of more than 4000. Our scientific investigations revealed significant differences between corresponding simulations in 2D and 3D. These results highlight the critical need for high-resolution, 3D simulation in conjunction with advanced *in situ* visualization and analysis to enable accurate modeling and study of the complex laser ion acceleration processes and prediction of beam properties.

## ACKNOWLEDGMENTS

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 as part of the SciDAC Institute for Scalable Data Management Analysis and Visualization (SDAV). This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

## LEGAL DISCLAIMER



## REFERENCES

[1] G. A. Mourou, T. Tajima, and S. V. Bulanov, "Optics in the relativistic regime," *Rev. Mod. Phys.*, vol. 78, pp. 309–371, Apr 2006. [Online]. Available: http://link.aps.org/doi/10.1103/RevModPhys.78.309

[2] H. Daido, M. Nishiuchi, and A. S. Pirozhkov, "Review of laser-driven ion sources and their applications," *Reports on Progress in Physics*, vol. 75, no. 5, p. 056401, 2012.

[3] C. K. Birdsall and A. B. Langdon, *Plasma physics via computer simulation*. CRC Press, 2004.

[4] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, G. H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E. W. Bethel, D. Camp, O. Rübel, M. Durant, J. Favre, and P. Navratil, "VisIt: An End-User Tool for Visualizing and Analyzing Very Large Data," in *High Performance Visualization—Enabling Extreme-Scale Scientific Insight*, ser. Chapman & Hall, CRC Computational Science, E. W. Bethel, H. Childs, and C. Hansen, Eds. Boca Raton, FL, USA: CRC Press/Francis–Taylor Group, Nov. 2012, pp. 357–372.

[5] B. Whitlock, J. M. Favre, and J. S. Meredith, "Parallel in situ coupling of simulation with a fully featured visualization system," in *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization*, ser. EGPGV '11. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2011, pp. 101–109.

[6] A. Friedman, R. Cohen, D. Grote, S. Lund, W. Sharp, J.-L. Vay, I. Haber, and R. Kishek, "Computational methods in the warp code framework for kinetic simulations of particle beams and plasmas," *Plasma Science, IEEE Transactions on*, vol. 42, no. 5, pp. 1321–1334, May 2014.

[7] Warp. [ONLINE] http://warp.lbl.gov.

[8] Forthon. [ONLINE] http://hifweb.lbl.gov/Forthon/.

[9] Gist. [ONLINE] http://hifweb.lbl.gov/public/software/gist.

[10] OpenDX. [ONLINE] http://www.opendx.org.

[11] S. S. Bulanov, A. Brantov, V. Y. Bychenkov, V. Chvykov, G. Kalinchenko, T. Matsuoka, P. Rousseau, S. Reed, V. Yanovsky, K. Krushelnick, D. W. Litzenberg, and A. Maksimchuk, "Accelerating protons to therapeutic energies with ultraintense, ultraclean, and ultrashort laser pulses," *Medical Physics*, vol. 35, no. 5, pp. 1770–1776, 2008. [Online]. Available: http://scitation.aip.org/content/aapm/journal/medphys/35/5/10.1118/1.2900112

[12] S. S. Bulanov, A. Brantov, V. Y. Bychenkov, V. Chvykov, G. Kalinchenko, T. Matsuoka, P. Rousseau, S. Reed, V. Yanovsky, D. W. Litzenberg, K. Krushelnick, and A. Maksimchuk, "Accelerating monoenergetic protons from ultrathin foils by flat-top laser pulses in the directed-coulomb-explosion regime," *Phys. Rev. E*, vol. 78, p. 026412, Aug 2008. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.78.026412

[13] K. Moreland, R. Oldfield, P. Marion, S. Jourdain, N. Podhorszki, V. Vishwanath, N. Fabian, C. Docan, M. Parashar, M. Hereld *et al.*, "Examples of in transit visualization," in *Proceedings of the 2nd international workshop on Petascal data analytics: challenges and opportunities*. ACM, 2011, pp. 1–6.

[14] A. C. Bauer, B. Geveci, and W. Schroeder, "The paraview catalyst users guide," 2013.

[15] B. Loring and O. Rübel, "Rendering and compositing infrastructure improvements to visit for insitu rendering," Lawrerence Berkeley Lab, Tech. Rep. LBNL-1004236, 2016.

[16] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[17] K. S. Yee *et al.*, "Numerical solution of initial boundary value problems involving maxwells equations in isotropic media," *IEEE Trans. Antennas Propag*, vol. 14, no. 3, pp. 302–307, 1966.