Journal of Physics:
Conference Series

PAPER • **OPEN ACCESS**

Toward the modeling of chains of plasma accelerator stages with WarpX

To cite this article: J.-L. Vay *et al* 2020 *J. Phys.: Conf. Ser.* **1596** 012059

Journal of Physics: Conference Series **1596** (2020) 012059 doi:10.1088/1742-6596/1596/1/012059

Toward the modeling of chains of plasma accelerator stages with WarpX

J.-L. Vay[1], A. Almgren[1], L. D. Amorim[1], J. Bell[1], L. Ge[3], K. Gott[1], D. P. Grote[2], M. Hogan[3], A. Huebl[1], R. Jambunathan[1], R. Lehe[1], A. Myers[1], C. Ng[3], J. Park[1], M. Rowan[1], O. Shapoval[1], M. Thévenet[1], W. Zhang[1], Y. Zhao[1], E. Zoni[1]

[1]Lawrence Berkeley National Laboratory, Berkeley, CA, USA
[2]Lawrence Livermore National Laboratory, Livermore, CA, USA
[3]SLAC National Accelerator Laboratory Menlo Park,, CA, USA

E-mail: jlvay@lbl.gov

Abstract. The Particle-In-Cell code WarpX is being developed by a team of the U.S. DOE Exascale Computing Project to enable the modeling of chains of tens of plasma accelerators on exascale supercomputers, for future collider designs. The code is combining the latest algorithmic advances (e.g., boosted frame, pseudo-spectral Maxwell solvers) with mesh refinement and runs on the latest CPU and GPU architectures. An example of the application to the modeling of up to three successive muti-GeV stages is presented. The latest implementation on GPU architectures is also reported.

1. Introduction

One of the most challenging application of plasma accelerators is the development of a plasma-based collider for high-energy physics studies [1]. Fast and accurate simulation tools are essential to study the physics toward configurations that enable the production and acceleration of very small beams with low energy spread and emittance preservation over long distances, as required for a collider. The Particle-In-Cell code WarpX [2] is being developed by a team of the U.S. DOE Exascale Computing Project [3] (with additional non-U.S. collaborators on part of the code) to enable the modeling of chains of tens of plasma accelerators on exascale supercomputers, for future collider designs. The code is combining the latest algorithmic advances (e.g., boosted frame, pseudo-spectral Maxwell solvers) with mesh refinement and runs on the latest Computer Processing Unit (CPU) and Graphical Processing Unit (GPU) architectures. The first version of the code, together with its first application to the modeling of plasma acceleration with mesh refinement, were reported in [4]. The addition and effect of subcycling on simulations with mesh refinement were reported in [5], together with a discussion on load balancing. In this paper, we present an application of WarpX (version 20.01) to the modeling of three successive muti-GeV stages. The latest implementation on GPU architectures is also reported.

2. Convergence study of a three-stage laser-driven plasma accelerator

A laser-driven plasma accelerator configuration with three identical consecutive stages was simulated. The simulations were done using a Lorentz boosted frame of reference, with

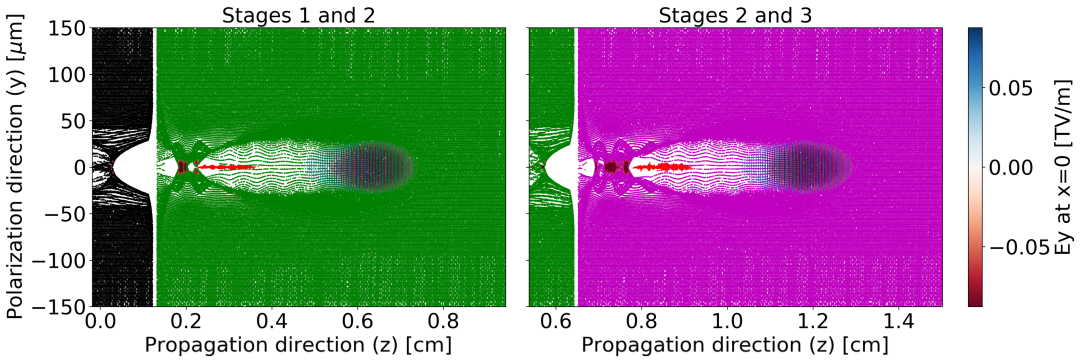

Figure 1. 2D colormap of the fields in the polarization direction (y), in the z-y plane (z being the propagation direction). The accelerated electron beam (red dots) and plasma electron macro-particles within a slice of 4 μm thickness along the x axis are plotted. Left and right plots show the transition from the first (black dots) to the second (green) and from the second to the third (pink) plasma stages, in the boosted frame (note that in this frame, the laser and electron beam are propagating from left to right, while the plasma stages are drifting from right to left).

relativistic factor $\gamma_{\text{boost}} = 30$, so as to minimize the overall number of time steps. A moving window traveling at the speed of light in vacuum, c, was also used to follow the accelerated beam and the wakefield (both traveling near the speed of light in the chosen boosted frame) as they progress in the longer plasma column.

Each plasma column consisted of a neutral mix of electrons and protons, with a longitudinal density profile consisting of a 20 mm linear up-ramp, followed by a plateau of 297 mm at a density of 1.7×10^{23} m^{-3} on axis, and ending with a 3 mm linear down-ramp. Transversely, the plasma stages have a parabolic profile with a matched radius of 40 μm and a maximum radius of 150 μm. A 0.24 PW laser pulse was injected at the entrance of each stage, with a wavelength of 0.8 μm, a duration of 73.4 fs, and focused to a spot size $W_0 = 50$ μm located at 8.8 mm after each stage entrance. The lasers were initialized by virtual antennas placed at the entrance of each stage and polarized in the y direction, for light propagation along z.

The first plasma stage started at $z = 0$ and consecutive stages were separated by 3 mm gaps. In future plasma-based colliders, it is envisioned that these gaps will be long enough to contain elements to deflect driver beams exiting a stage and inject a new driver beam into the next stage, as well as elements to focus the accelerated electron (or positron) beams transversely for matching between stages. We note, however, that the length of the physical gaps is to be minimized, as to maximize the average acceleration gradient along the accelerator, and it is thus expected that their length is to be much shorter than - or at most comparable to - the plasma stages. In the case where the gap may be of comparable length to the plasma stages, most of it is expected to be drift sections where the electron beam propagates in vacuum, at a very small computational cost. Since the goal of these initial studies is to demonstrate the ability to perform multistage simulations, the numerical cost of which is to be dominated by the plasma sections, the complexity associated with modeling the physics in the gaps is neglected for now, using very short gaps between stages. Since the actual modeling of the gap physics is essential to the final science goal, the capabilities to do so are being implemented in the code. Incorporation of this physics will occur progressively — and be reported on — in future reports.

An electron beam with 0.15 nC of total electrical charge was injected at a carefully chosen phase behind the laser powering the first stage, with an initial energy of 1 GeV, 2% energy spread, and an initial transverse emittance (a statistical measure of beam quality and focusability) of

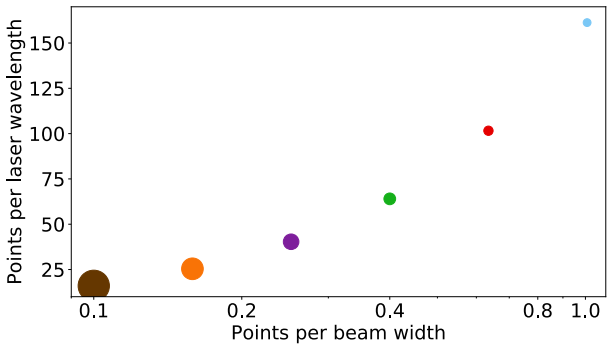

Figure 2. Each circle represents a simulation from the scan for a given transverse resolution (number of points per initial electron beam RMS width $\sigma_x = \sigma_y$) and a given longitudinal resolution (number of grid points per laser wavelength). The color and size of each circle are used as a reference for this figure and the following figures in this section.

0.25 μm·rad. The beam had an initial transverse root-mean-square (RMS) width of $\sigma_{x,y} = 0.6$ μm and $\sigma_z = 3$ μm RMS length. The beam species was modeled with 50,000 macro-particles. The 3D computational box size was $192 \times 192 \times 161$ μm^3.

Figure 1 shows the transition of the electron beam between the first (black), second (green) and third (purple) stages in the boosted frame. As expected, the wake structure (lower density region) is identical in each stage and captures the full length of the beam.

A longitudinal and transverse resolution convergence scan was run on the Cori supercomputer at NERSC [6] (using the intel KNL nodes), with the simulation configurations summarized in Fig. 2.

For the analysis, beam electrons were selected within a region of $2\sigma_z$ of the beam center along the propagation direction, to exclude the very few electrons that were decelerated and then lost. The deceleration and loss of electrons resulted from the tail of the beam being close to the end of the plasma bucket (as can be seen in the left plot of Fig. 1). Total losses of electrons along the three stages were in the range of 1 per 10,000.

The results from the scan shown in Fig. 3 confirm that the predicted electron beam quantities are converging nicely with increasing transverse and longitudinal resolutions. As expected, a first order moment of the electron beam, such as its mean energy, is converging faster (and more monotonically), than more sensitive second order moments, such as the relative energy spread and emittances (projected or slice).

3. GPU portability strategy and progress

WarpX relies on the ECP AMReX framework [7] to achieve performance portability on CPU and GPU, with a **single-source code base**. More specifically, AMReX provides a `C++` function `ParallelFor` as well as similar primitives for reduction, which are converted to GPU kernel launches when the code is compiled for GPU (and is converted to a simple `For` loop when the code is compiled for CPU). Note that this solution is similar to that used in Kokkos [8] or Raja [9]. WarpX has been extensively refactored, in order to use this performance-portable framework throughout the code.

In addition, AMReX provides containers for particles and mesh data. When the code is compiled for GPU, these containers are allocated in GPU managed memory. This means that host-device memory transfers are performed automatically whenever the data is used on the host or the device, and thus that the code does not need to contain explicit calls to, e.g., `cudaMemcpy`.

During a typical WarpX GPU simulation, **all** the compute operations now run on the GPU.

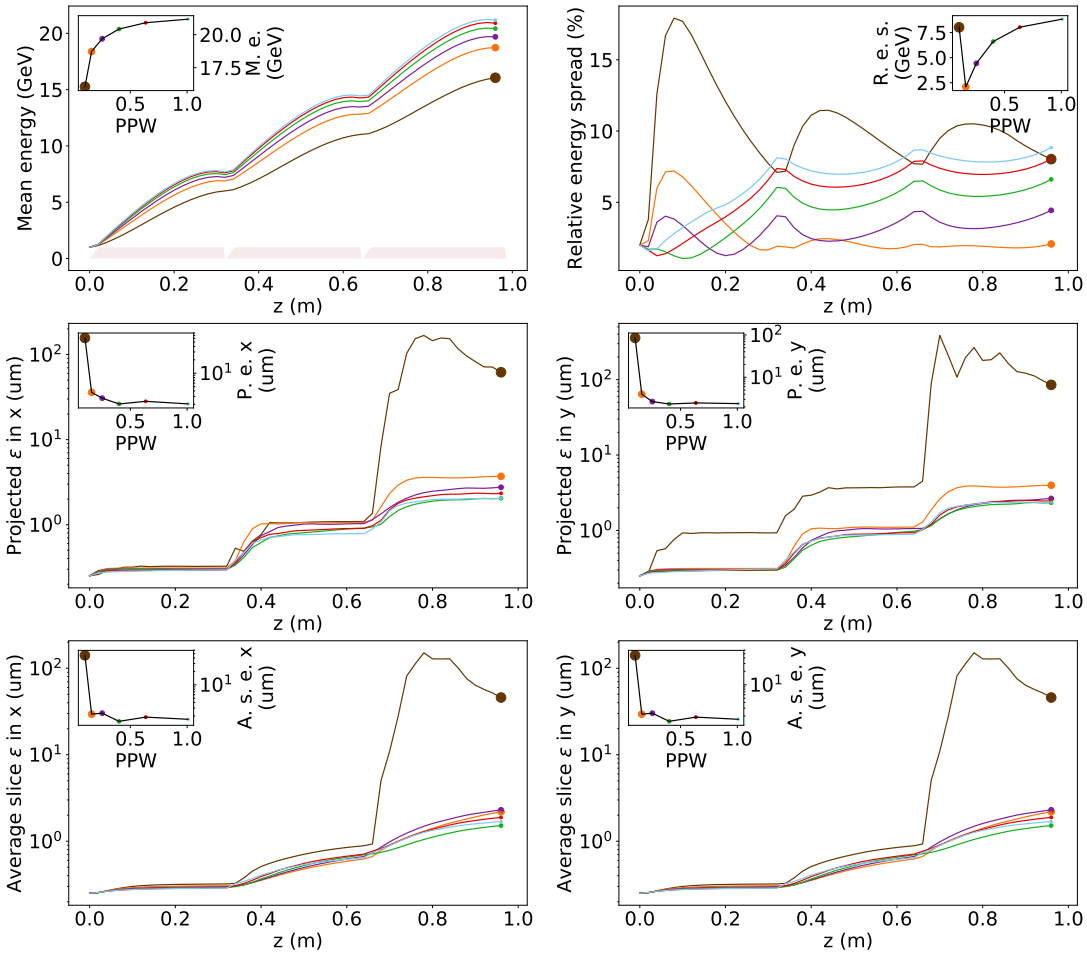

Figure 3. Evolution of the values of beam mean energy (top left), relative energy spread (top right), projected emittance ϵ in x and y (center left and right) and charge weighted average of the slice emittance in x and y (bottom left and right), along the propagation direction z. The mean energy plot (top left) also shows the plasma density longitudinal profile (shaded beige color). Insets show the final values of each quantity as a function of resolution (expressed as a number of points per beam width - PPW - in the transverse direction). Results follow the color and circle sizes of Fig. 2, with increasing longitudinal and transverse resolutions, with constant transverse-to-longitudinal resolution ratio $\Delta_{x,y}/\Delta_z$.

As a consequence, all the simulation data is automatically transferred to the device **on the first timestep** (via managed memory transfer), and then stays on the device for the rest of the simulation. As a consequence, host-device transfers are only needed when writing output data to disk.

Prior to the port on GPUs, WarpX was running efficiently on CPUs and KNL architectures (NERSC Cori) via low-level Fortran routines of the PIC kernel (PICSAR), which were called from a high-level `C++` code (WarpX code base) [4]. When transitioning to GPU, we initially experimented with OpenACC [10], in order to keep the same Fortran code base. However, we found this approach to be hindered by the limited availability and capabilities of compilers that support OpenACC in Fortran for GPU.

In preparation of the code for Oak Ridge National Laboratory's GPU-based supercomputer

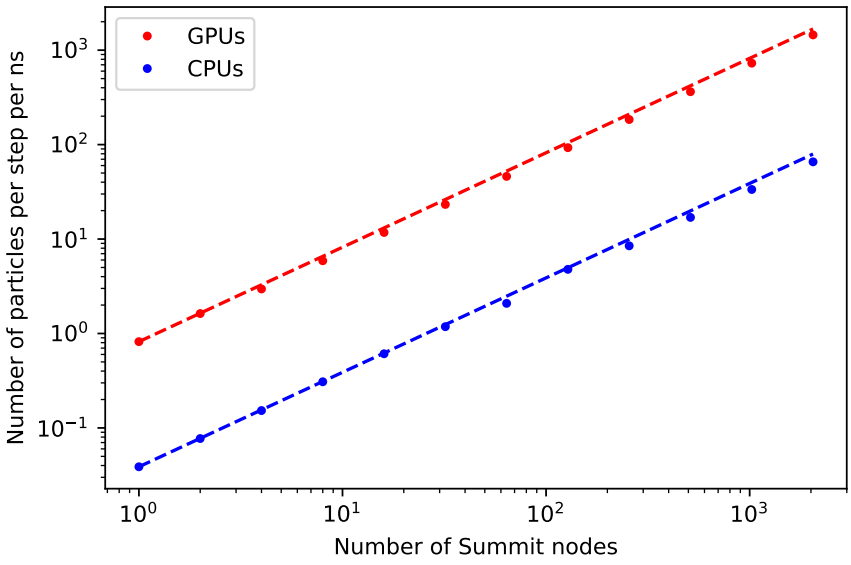

Figure 4. Results of a weak scaling study on a uniform plasma benchmark with WarpX, comparing GPU and CPU performance on Summit. The colored circles indicate simulation results, while the dotted line illustrates perfect weak scaling. Both cases used 8 particles per cell, Esirkepov current deposition, and 3rd-order interpolation. The GPU-accelerated runs used 6 GPUs per node, with 1 MPI task per GPU. The CPU-only runs used all 42 available POWER9 cores on a Summit node, using 6 MPI tasks and 7 OpenMP threads per node. Both runs exhibit excellent weak scaling. On 2048 Summit nodes, the GPU-accelerated run was approximately 22 times faster that the CPU-only run.

Summit [11], we therefore transitioned to a `C++`-only strategy, based on the above-mentioned portability layer from AMReX. WarpX was extensively refactored, and all particle-mesh routines were converted to `C++`. The transition to a `C++`-only core enabled us to use a wider range of compilers for various architectural targets, and single-source programming for both OpenMP, CUDA and upcoming HIP programming models, improving code maintainability.

Benchmarks on Summit revealed excellent weak scaling, and a considerable speed-up from the use of GPUs, as shown in Fig. 4.

We also performed comparisons between Cori's Intel KNL architecture and Summit's NVIDIA V100 GPU architecture. In general, we noticed that the core compute routines (e.g. particle push, field update on the mesh, etc.) were typically much faster on V100, but that the routines associated with MPI communications (e.g. packing/unpacking of MPI buffers before/after MPI communications) did not benefit from the same speed-up. Upon further investigation, we noticed that these routines involved many small kernel launches, and that the kernel launch overhead (when running on V100) became a bottleneck in this case. These communication routines were therefore refactored, so as to have as few kernel launches as possible. For instance, in the case of guard cell exchanges, this was achieved by having a *single kernel* that loops over the 3D boundary of a subdomain and packs the guard cell data into different MPI buffers (as opposed to having one kernel launch per subdomain face/edge). The resulting speed-up is shown in Fig. 5, which compares the time taken by these routines before this refactoring (June 2019) and after this refactoring (October 2019).

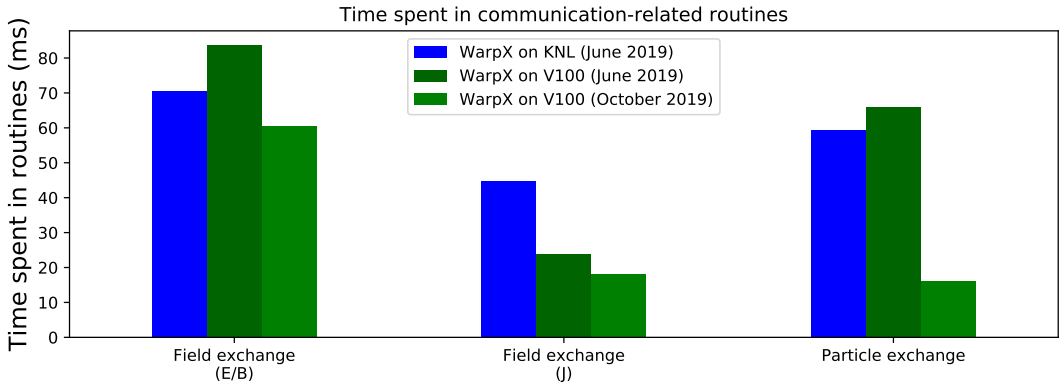

Figure 5. Time spent in communication-related routines (e.g. packing and unpacking of MPI buffers) in a benchmark involving a typical boosted-frame simulation of laser-wakefield acceleration with WarpX. The code was run on Cori (blue) and Summit (green), using 900 KNL nodes and 900 V100 GPUs on 150 Summit nodes respectively.

4. Conclusion and outlook

The WarpX code is getting ready for production-level, three-dimensional, simulations of chains of a few multi-GeV plasma accelerator stages on CPU- or GPU-based supercomputers. Further developments are underway to speedup the code further with mesh refinement, algorithms that enable larger time steps, reduced models and dimension, further optimization of the implementation on GPUs and improved dynamic load balancing. With these developments, the projects is targeting the modeling of chains of tens of stages on upcoming exascale supercomputers.

Acknowledgment

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations—the Office of Science and the National Nuclear Security Administration—responsible for the planning and preparation of a capable exascale ecosystem—including software, applications, hardware, advanced system engineering, and early testbed platforms—to support the nation's exascale computing imperative.

This work was performed in part under the auspices of the U.S. Department of Energy by Lawrence Berkeley National Laboratory under Contract DE-AC02-05CH11231, SLAC National Accelerator Laboratory under contract AC02-76SF00515, and Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

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. 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.

References

[1] ALEGRO 2019 (*Preprint* 1901.10370) URL http://arxiv.org/abs/1901.10370
[2] WarpX URL https://github.com/ECP-WarpX/WarpX
[3] Home Page - Exascale Computing Project URL https://www.exascaleproject.org/

[4] Vay J L, Almgren A, Bell J, Ge L, Grote D, Hogan M, Kononenko O, Lehe R, Myers A, Ng C, Park J, Ryne R, Shapoval O, Thévenet M and Zhang W 2018 *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* ISSN 01689002
[5] Vay J L, Almgren A, Bell J, Lehe R, Myers A, Park J, Shapoval O, Thevenet M, Zhang W, Grote D P, Hogan M, Ge L and Ng C 2019 *2018 IEEE Advanced Accelerator Concepts Workshop, ACC 2018 - Proceedings* (Institute of Electrical and Electronics Engineers Inc.) ISBN 9781538677216
[6] National Energy Research Scientific Computing Center URL `http://www.nersc.gov/`
[7] AMReX URL `https://amrex-codes.github.io/`
[8] Carter Edwards H, Trott C R and Sunderland D 2014 *Journal of Parallel and Distributed Computing* **74** 3202–3216 ISSN 07437315
[9] Beckingsale D A, Scogland T R, Burmark J, Hornung R, Jones H, Killian W, Kunen A J, Pearce O, Robinson P and Ryujin B S 2019 *2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)* (IEEE) pp 71–81 ISBN 978-1-7281-6003-0 URL `https://ieeexplore.ieee.org/document/8945721/`
[10] Homepage — OpenACC URL `https://www.openacc.org/`
[11] Summit Oak Ridge Leadership Computing Facility URL `https://www.olcf.ornl.gov/summit/`