Performance-portable GPU acceleration of the hybrid PIC code `dHybridR`

Bricker Ostler[1] Miha Cernetic[2,3] Damiano Caprioli[2,4]

[1]Department of Physics, University of Chicago [2]Department of Astronomy and Astrophysics, University of Chicago [3]Cerebras Systems, Sunnyvale, CA [4]Enrico Fermi Institute, University of Chicago

Paper: arXiv:2609.28422

1 codebase
for Intel, AMD, and NVIDIA GPUs

86–97%
weak scaling efficiency at 49,152 accelerators each on Aurora and Frontier

14–32×
full-node GPU vs. vectorized CPU speedup (256 ppc)

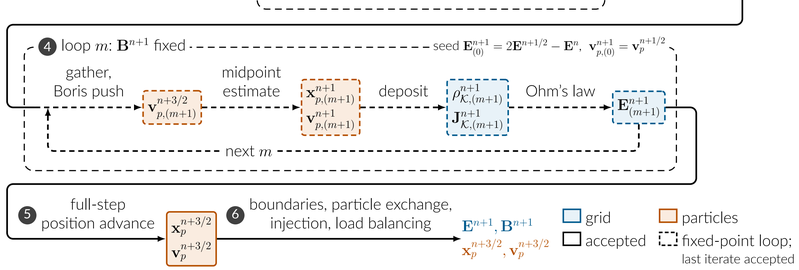
90%
less energy per particle-update (Aurora, Frontier; 256 ppc)

Why is hybrid PIC on GPUs important?

- **3D captures physics that 2D doesn't.** 2D runs suppress cross-field diffusion [1] and do not capture ion acceleration to non-thermal energies at quasi-perpendicular shocks [2].
- **3D is costly to run.** Production runs are 2D or compromised 3D. Uncompromised 3D runs require $\gtrsim 10\times$ more compute, which most leadership machines concentrate in their GPUs.
- **There is a gap in the hybrid PIC landscape.** Published GPU hybrid PIC codes [3–6] are mostly single-vendor and report scaling to at most 64 NVIDIA GPUs. **Our contribution:** A performance-portable hybrid PIC code with demonstrated scaling at leadership facility scale.

Hybrid PIC in `dHybridR` [7, 8]

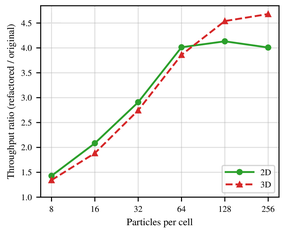
Ions ($\mathcal{K}$) are kinetic macroparticles and **electrons** an inertialess, charge-neutralizing fluid. A timestep advances $\mathbf{E}, \mathbf{B}$ from t^n to t^{n+1} and $\mathbf{x}_p, \mathbf{v}_p$ from $t^{n+1/2}$ to $t^{n+3/2}$ in steps ❶ to ❻:



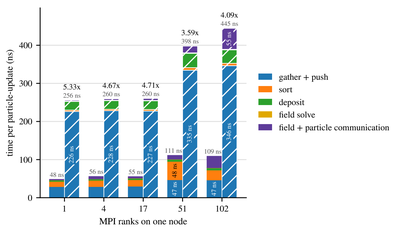
Initial CPU code refactor

Particle properties moved from `p(S,N)` to per-property arrays `x(N)`, `y(N)`, ..., so loops run unit stride and auto-vectorize. Similar refactor for grid arrays. Sorting particles by cell every step keeps field reads and moment writes from scattering across main memory as particles move, making use of the cache hierarchy.

Refactored / original throughput ratio, full Aurora node



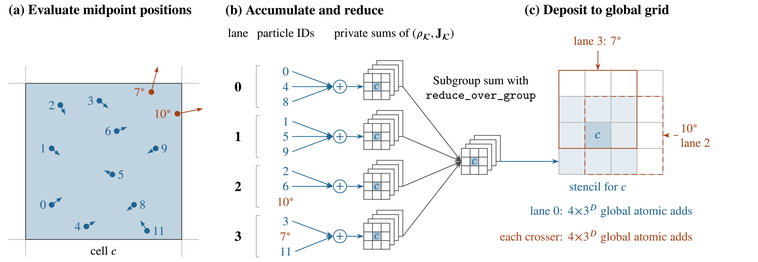
Refactored (solid), original (hatched); 2D, 256 ppc



Porting strategy

- **Goals:** Single CPU-GPU codebase; no dependence on any single GPU vendor; ability to write low-level kernels for optimization.
- **What we chose: OpenMP target offload** with select **SYCL** kernels where profiling showed subgroup and local-memory control could be beneficial.
- **Data stay on the GPU:** GPU-aware MPI and parallel on-device particle exchange avoid round trips to the host.
- **Vendor-agnostic:** Requires only OpenMP offload + SYCL interop, providing a path for existing MPI Fortran plasma codes.

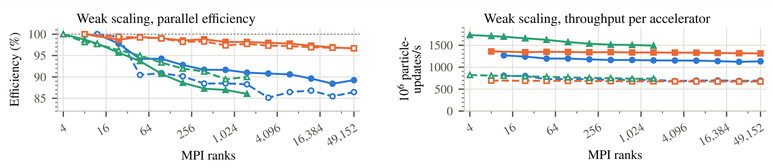
SYCL kernel: charge and current deposition



The GPU charge and current deposition kernel, illustrated in 2D for 12 macroparticles of a single species and a subgroup of 4 lanes, one subgroup per cell. (a) Arrows connect the initial position of each macroparticle to its estimated midpoint location. (b) Each lane accumulates contributions to charge and current for the non-crossing particles into four private 3^2 stencils. A subgroup reduction combines stencils element-wise across lanes. (c) Lane 0 atomically deposits the reduced stencils to the large moment arrays residing in global memory, shown in blue. Crossing particles atomically deposit their individual shifted stencils as well (orange).

Global atomic adds per cell (for spatial dimension D):
$4 \times 3^D N_{\mathrm{ppc}} \longrightarrow 4 \times 3^D (1 + N_{\mathrm{cross}}),$ typically $N_{\mathrm{cross}} \ll N_{\mathrm{ppc}}$

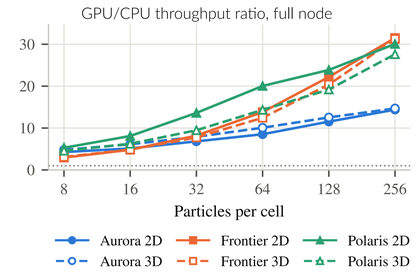
Weak scaling across three GPU vendors

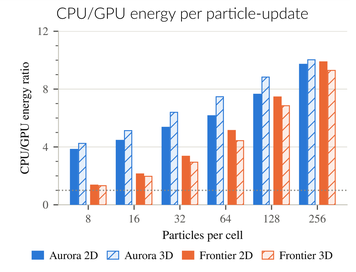


Machines	2D	3D	GPUs/node	MPI ranks/node
Aurora	●	○	6× Intel Max 1550 (2 tiles)	12
Frontier	■	□	4× AMD MI250X (2 GCDs)	8
Polaris	▲	△	4× NVIDIA A100	4

Uniform thermal plasma benchmark. One rank per accelerator (tile, GCD, or A100); 10^6 cells per rank, 256 particles per cell (ppc).

GPU vs. CPU on the same node

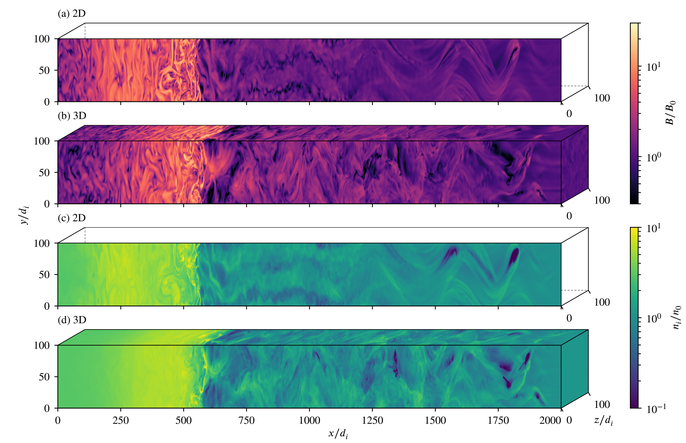




- **At 256 ppc:** 14–32× faster than refactored CPU code (still rising with ppc); ≃ 90% less energy per particle-update on Aurora and Frontier (Polaris has no whole-node energy counter).
- **CPU runs:** One rank per physical core. CPU-side HBM partly explains Aurora's lower speedup.

Application: 2D vs. 3D collisionless shock

Twin runs of a benchmark $\mathcal{M}_A \simeq 25$ parallel shock [9] with 64 ppc, 5 cells per d_i, and a $2000 \times 100\,(\times\,100)\,d_i$ box, shown below at $t\Omega_i \simeq 104$ (shock at $x \approx 560\,d_i$, downstream to its left).



- **2D vs. 3D:** In 2D, B_z holds ≃ 50% of the magnetic energy; B, n_i stay near peak for hundreds of d_i downstream. In 3D, B_x, B_y, B_z are near energy equipartition and B, n_i taper off downstream.
- **Compute cost:** For the 3D run, 15.6 h on 512 Frontier nodes. The same run on the refactored CPU code would take ∼ 10× the node-hours; on the original code, 40–50× (up to ∼ 100× for 256-ppc runs).

Animations with Ascent [10]

[1] J. R. Jokipii et al. *Geophysical Research Letters* 20 (1993), 1759.
[2] L. Orusa and D. Caprioli. *Physical Review Letters* 131 (2023), 095201.
[3] S. Fatemi et al. *Journal of Physics: Conference Series* 837 (2017), 012017.
[4] E. Behar et al. *Annales Geophysicae* 40 (2022), 281.
[5] S. Kim et al. *Computer Physics Communications* 315 (2025), 109726.
[6] R. E. Groenewald et al. *Physics of Plasmas* 30 (2023), 122508.
[7] C. C. Haggerty and D. Caprioli. *The Astrophysical Journal* 887 (2019), 165.
[8] L. Gargaté et al. *Computer Physics Communications* 176 (2007), 419.
[9] D. Caprioli and A. Spitkovsky. *The Astrophysical Journal* 783 (2014), 91.
[10] M. Larsen et al. *In Situ Visualization for Computational Science* (2022), 255.

This research used resources of the Argonne Leadership Computing Facility, which is a U.S. Department of Energy Office of Science User Facility operated under contract No. DE-AC02-06CH11357, and resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Advanced Scientific Computing Research programs in the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. D.C. was partially supported by NASA grant 80NSSC18K1726 and NSF grant AST-2510951.

bostler@uchicago.edu