LA-UR-21-22147
Approved for public release; distribution is unlimited.

Title: VPIC: Performance Portability on Modern Hardware Architectures

Author(s): Luedtke, Scott Vernon
Bird, Robert Francis (Bob)
Tan, Nigel Phillip
Harrell, Stephen Lien
Taufer, Michela
Albright, Brian James

Intended for: SIAM Conference on Computational Science and Engineering (CSE21), 2021-03-01/2021-03-05 (Online, New Mexico, United States)

Issued: 2021-03-03

Los Alamos NATIONAL LABORATORY

VPIC: Performance Portability on Modern Hardware Architectures

Scott V. Luedtke[1], Robert Bird[1], Nigel Tan[1,2], Stephen Lien Harrell[3], Michela Taufer[2], & Brian Albright[1]

[1]Los Alamos National Laboratory, Los Alamos, NM
[2]University of Tennessee at Knoxville, Knoxville, TN
[3]Texas Advanced Computing Center, University of Texas at Austin, Austin, TX

March 5, 2021

The Particle-in-Cell method and VPIC

Adapting VPIC to the Kokkos framework

VPIC performance tests

Future plans for VPIC

Conclusions

The particle-in-cell method is used for a broad range of plasmas

- The PIC method is used to simulate plasmas ranging from micron-scale laser-produced plasmas to plasmas in the global magnetosphere
- Solves full 7D plasma distribution function $f(\mathbf{x}, \mathbf{v}, t)$, with evolution governed by Vlasov's equation

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f + \frac{q}{\gamma m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_{\mathbf{p}} f = 0 \tag{1}$$

- Simulation volume is discretized, with EM fields specified on gridpoints only
- Quasi-particles move continuously throughout volume
- Fully reproduces collisionless, relativistic, kinetic plasmas
- Widely used in plasma physics where fluid approximations fail

Field solver

- Maxwell's equations solved with finite-difference time-domain (FDTD) method on Yee staggered grid
- First, update field to half time-step:

$$\mathbf{E}^{n+\frac{1}{2}} = \mathbf{E}^{n} + \frac{\Delta t}{2}\left(c^2 \boldsymbol{\nabla} \times \mathbf{B}^{n} - \frac{\mathbf{J}^{n}}{\epsilon_0}\right), \tag{2}$$

$$\mathbf{B}^{n+\frac{1}{2}} = \mathbf{B}^{n} - \frac{\Delta t}{2}\left(\boldsymbol{\nabla} \times \mathbf{E}^{n+\frac{1}{2}}\right). \tag{3}$$

- Next, run particle pusher and update currents
- Last, update fields to full time-step:

$$\mathbf{B}^{n+1} = \mathbf{B}^{n+\frac{1}{2}} - \frac{\Delta t}{2}\left(\boldsymbol{\nabla} \times \mathbf{E}^{n+\frac{1}{2}}\right), \tag{4}$$

$$\mathbf{E}^{n+1} = \mathbf{E}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\left(c^2 \boldsymbol{\nabla} \times \mathbf{B}^{n+1} - \frac{\mathbf{J}^{n+1}}{\epsilon_0}\right). \tag{5}$$

Particle pusher

Particle pusher solves the relativistic equations of motion

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}, \tag{6}$$

and

$$\frac{d\mathbf{v}}{dt} = \frac{q}{\gamma m}(\mathbf{E} + \mathbf{v} \times \mathbf{B}), \tag{7}$$

- Particles do not interact with each other, unless collisions are added
- Second-order accurate Boris pusher is most common
- Higher-order pushers possible

VPIC 1.0 achieved high performance with platform specific-performance optimization

VPIC 1.0 was built to minimize data movement and maximize performance, with platform-specific optimizations requiring significant work for every new architecture.

- VPIC is a Gordon Bell finalist code
- Used in one of the first petascale simulations on Roadrunner
- Demonstrated performance on up to 2 million MPI ranks on Trinity
- Vector intrinsics ensure good performance on CPU-like platforms

1. Bowers, K. J., B. J. Albright, B. Bergen, L. Yin, K. J. Barker and D. J. Kerbyson, "0.374 Pflop/s Trillion-Particle Kinetic Modeling of Laser Plasma Interaction on Road-runner," Proc. 2008 ACM/IEEE Conf. Supercomputing (Gordon Bell Prize Finalist Paper). http://dl.acm.org/citation.cfm?id=1413435
2. K.J. Bowers, B.J. Albright, B. Bergen and T.J.T. Kwan, Ultrahigh performance three-dimensional electromagnetic relativistic kinetic plasma simulation, Phys. Plasmas 15, 055703 (2008); http://dx.doi.org/10.1063/1.2840133
3. K.J. Bowers, B.J. Albright, L. Yin, W. Daughton, V. Roytershteyn, B. Bergen and T.J.T Kwan, Advances in petascale kinetic simulations with VPIC and Roadrunner, Journal of Physics: Conference Series 180, 012055, 2009

Kokkos framework enables portable performance

Kokkos [4] is a suite of libraries and abstractions for developing portable high-performance C++ applications. Kokkos achieves high performance by mapping its own programming model to various backends, e.g., CUDA.

- Kokkos kernels libraries provide common algorithms, e.g., BLAS
- Data structures are abstracted and can change memory space, layout, and traits
- Parallel executions have space, pattern, and policy attributes

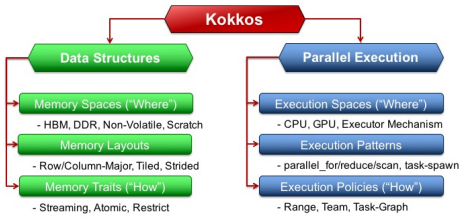

Figure: Core Kokkos abstractions for ensuring performance portability.

4. C. R. Trott, S. J. Plimpton, and A. P. Thompson, "Solving the performance portability issue with kokkos." Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2017.

Kokkos implementation is mostly straightforward

Broadly speaking, implementing Kokkos means changing parallel statements to the Kokkos syntax and changing data structures to be Kokkos views.

- `Kokkos::parallel_for` is nearly a drop in replacement
- Particle and field arrays become 2D views
- Kokkos decides if particles and fields are arrays of structs (CPU) or structs of arrays (GPU)
- Need to be careful that host and device views are up to date; use `deep_copy`
- Adding atomics boosts performance on GPUs
- Some user decks need to be made Kokkos-aware

Performance test emulates a large SBS simulation

Our test deck is modified from a stimulated Brillouin scattering simulation for inertial confinement fusion applications [5].

- Modified it to have 2 lasers, which ramp faster and stronger, incident on a uniform underdense plasma
- 80 ions and electrons per cell and 100^3 cells per GPU or socket, nearly filling VRAM on V100
- “Full size” is 13,500 GPUs
- Higher densities and/or more realistic lower initial temperature would require significantly more resolution, so scaling this even larger is a realistic use case

5. B. Albright, L. Yin, K. Bowers, and B. Bergen, “Multi-dimensional dynamics of stimulated brillouin scattering in a laser speckle: Ion acoustic wave bowing, breakup, and laser-seeded two-ion-wave decay,” Physics of Plasmas, vol. 23, no. 3, p. 032703, 2016.

More modern GPUs provide great performance

- V100 and A100 benefit from increased atomics support
- Single precision nature of VPIC means good performance on workstation cards
- The Kokkos AMD backend is relatively immature
- On A100, this is $\sim$ 2.3 pushes per ns per GPU, including all communication and overhead

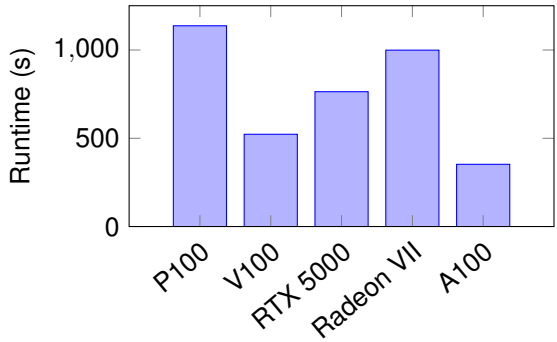

Figure: Total simulation time for small scale GPU performance (8 cards).

CPU performance is mixed

- CPUs are run with one MPI rank per core, which makes communication time significant compared to GPUs
- Only AMD Zen 2 is competitive with GPU performance
- The Kokkos CPU backends currently do not vectorize well
- The CPU backends are not a development priority

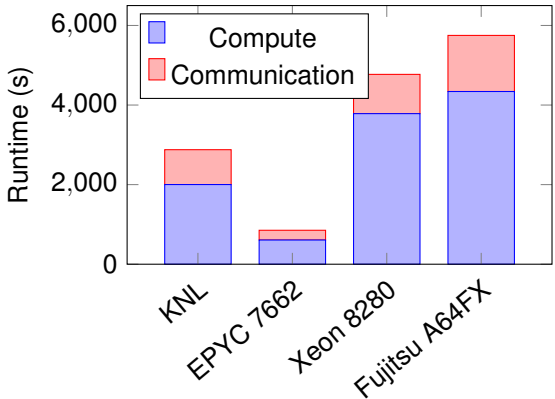

Figure: Compute and communication time for small scale CPU performance (8 sockets).

GPUs weak scale very well

- Boundary conditions mean there is less communication per GPU below 64 GPUs (the normalization point)
- Sierra sees only a 3.45% slowdown on 7,200 vs. 64 GPUs
- Selene and Fronterra scale about half as well as Sierra, but are newer and have less-mature MPI implementations
- Runs planned for all of Sierra (17,280 GPUs).

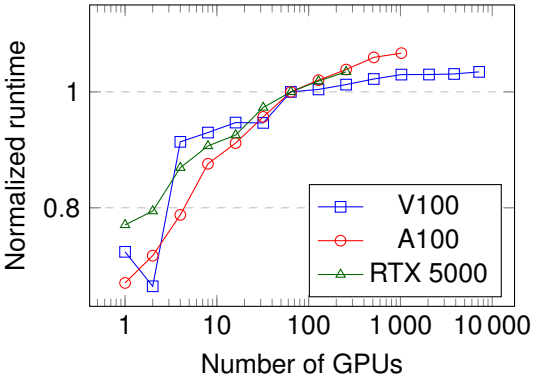

Figure: Weak Scaling on three GPU-based platforms (Sierra with Power 9 CPUs and V100 GPUs, Selene with EPYC 7742 CPUs and A100 GPUs, and the Frontera with Xeon E5-2620 v4 CPUs and RTX 5000 GPUs).

CPU weak scaling is good, but platform dependent

- Normalized to 64 CPU runs to match GPU results, though per rank communication is constant for 4 or more CPUs
- Significantly slower than GPUs in absolute terms, but still add value from a portability perspective
- Better performance on AMD Zen 2 than Cascade Lake and more MPI ranks per socket/GPU may expose the MPI implementation more

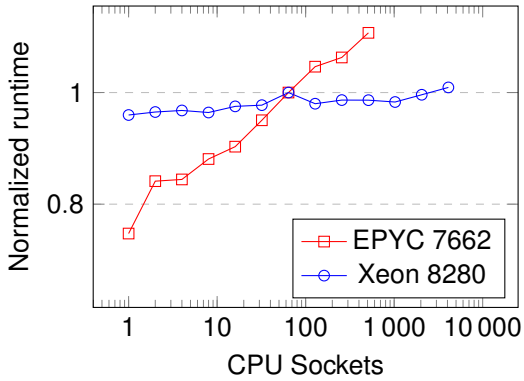

Figure: Weak scaling on two CPU-based platforms (i.e., Bell cluster at Purdue University using EPYC 7662 CPUs and the Frontera cluster with Xeon 8280 CPUs only).

Development continues on VPIC 2.0 towards a public release

VPIC development does not stop with some scaling plots! We hope to have a public open-source release later this year. Compared to VPIC 1.0 look forward to:

- Better documentation
- More features built-in, as opposed to existing in user's decks
- Automatic performance improvements as the Kokkos backends are improved
- Easier I/O and data visualization

Conclusions

- VPIC 2.0, with its Kokkos implementation, is performant on several GPU architectures, and portable to CPUs
- VPIC 2.0 demonstrates exceptional weak scaling up to 7,200 GPUs on Sierra
- The Kokkos framework means that VPIC 2.0 should perform well on future architectures with minimal programming effort

More details, including strong scaling and interesting cache effects, are in our preprint on arXiv: `http://arxiv.org/abs/2102.13133`

Acknowledgments

Work performed under the auspices of the U.S. DOE by Triad National Security, LLC, and Los Alamos National Laboratory(LANL). This work was supported by the LANL ASC and Experimental Sciences programs. Approved for public release: LA-UR-21-XXXXX. The UTK authors acknowledge the support of LANL under contract #578735 and IBM through a Shared University Research Award.
The authors thank John Cazes and Tommy Minyard at the Texas Advanced Computing Center at the University of Texas at Austin, Preston Smith and Xiao Zhu from the Research Computing department at Purdue University, the Innovative Computing Laboratory at UTK, and Max Katz and the entire team at the NVIDIA Corporation for allocating time for and assisting with the experiments in this paper.
Additionally we thank the Stony Brook Research Computing and Cyberinfrastructure, and the Institute for Advanced Computational Science at Stony Brook University for access to the innovative high-performance Ookami computing system, which was made possible by a $5M National Science Foundation grant (#1927880).