Sandia National Laboratories

Exceptional service in the national interest

Performance Portability of an SpMV Kernel Across Scientific Computing and Data Science Applications

Stephen Olivier, Nathan Ellingwood, Jonathan Berry, Danny Dunlavy

Originally presented and published at

IEEE HPEC 2021

SAND2021-11467C

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

U.S. DEPARTMENT OF ENERGY

NNSA

Microapp Design, Coding, and Data

Computation and baseline code

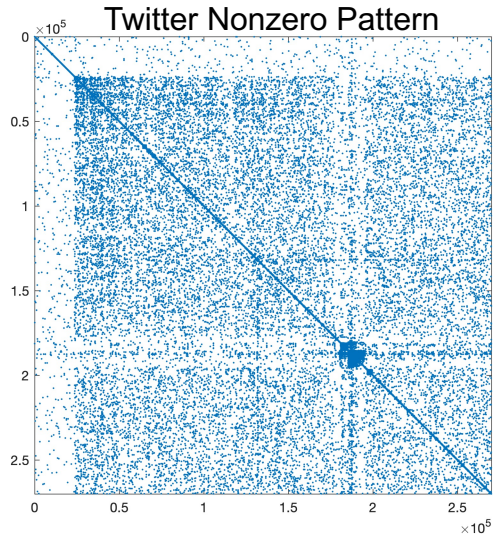
- Sparse matrix-vector product, repeated 100x
- Nested loops (outer loop over rows, inner loop over nonzeros)
- Supports variable number of nonzeros per row
- OpenMP `parallel for` enables CPU multithreading

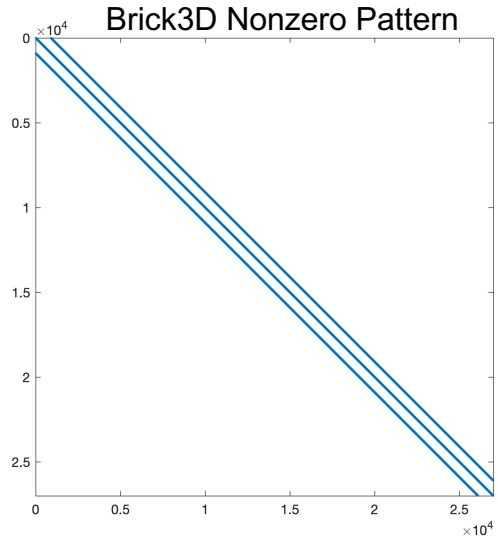
Performance portable code

- OpenMP GPU offload: ~5 lines added or changed
- Kokkos `range`: ~20 lines added or changed
- Kokkos `team`: ~50 lines added or changed
- Includes special data structures used by Kokkos

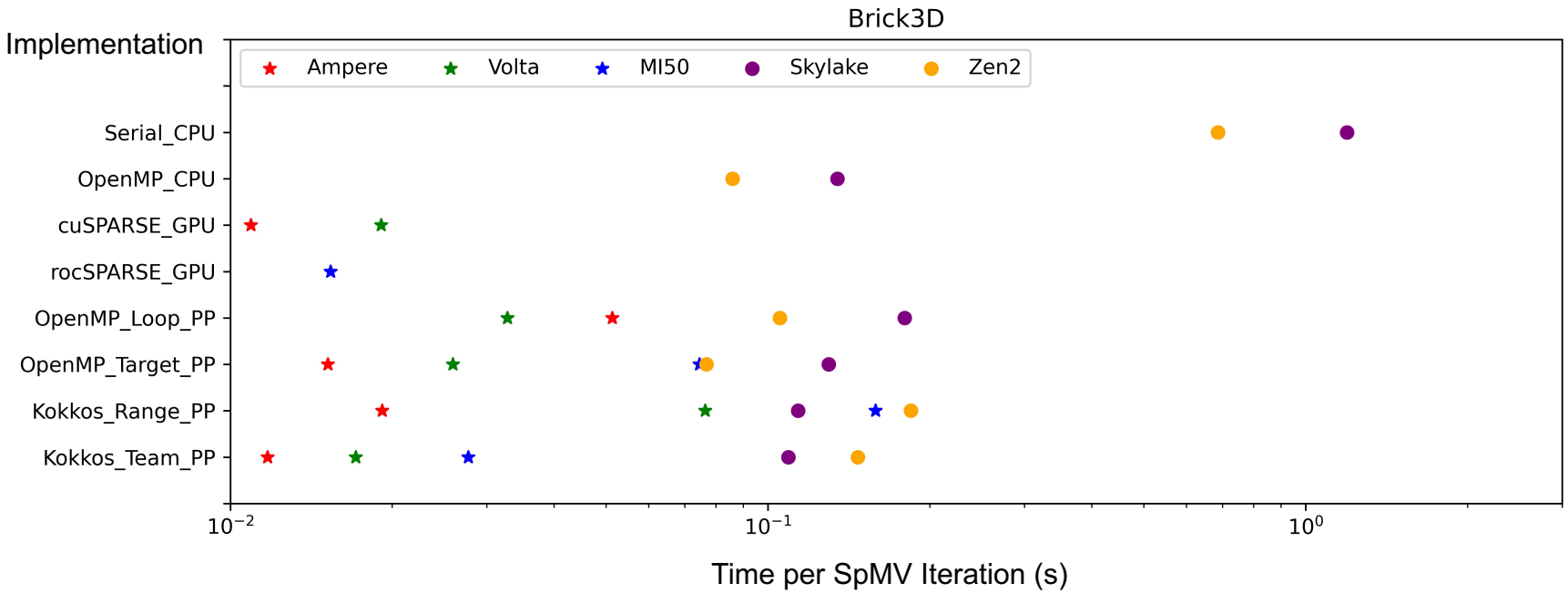
Input Data

- **Twitter:** Twitter-2010 followers network (~265M edges) *[top plot]*
- **Brick3D:** 27-point stencil for the Laplace operator on 3D hex mesh (320^3 mesh points) *[bottom plot]*





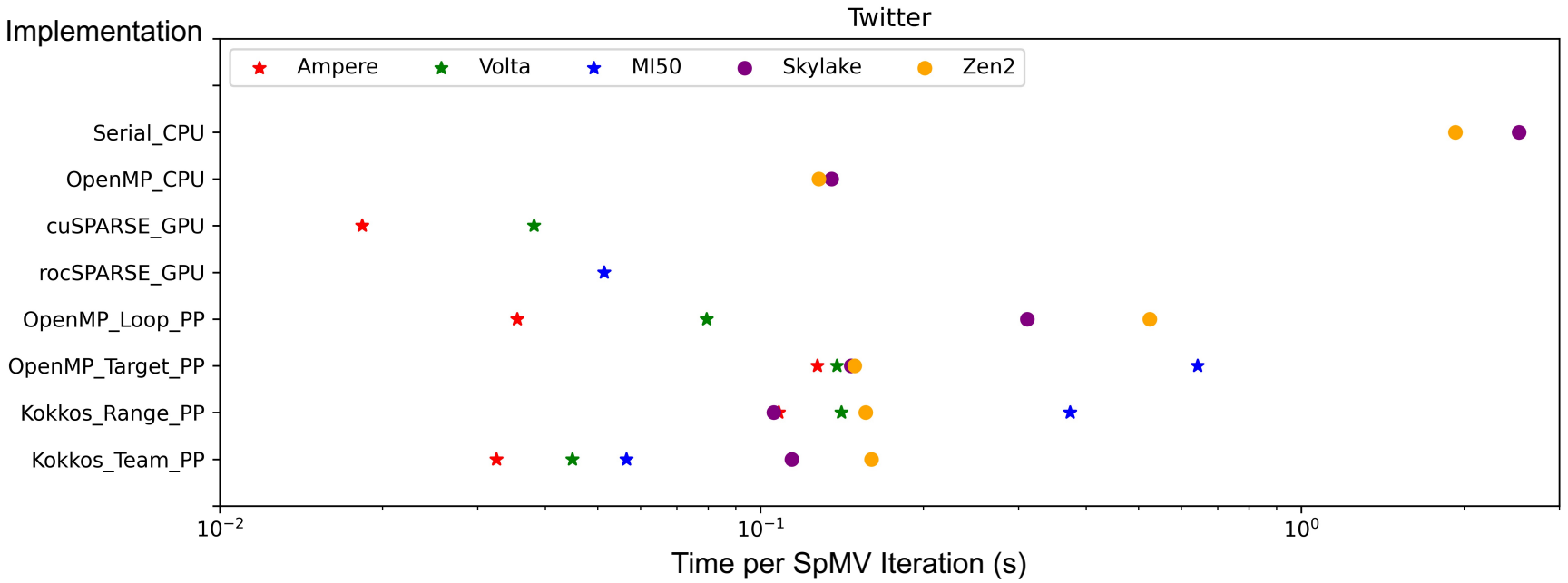
Results: 3D Hex Mesh Stencil



Some success for performance portable [PP] methods

- None significantly underperform CPU-only “classic” OpenMP
- Several perform at or near the level of vendor math libraries on GPU

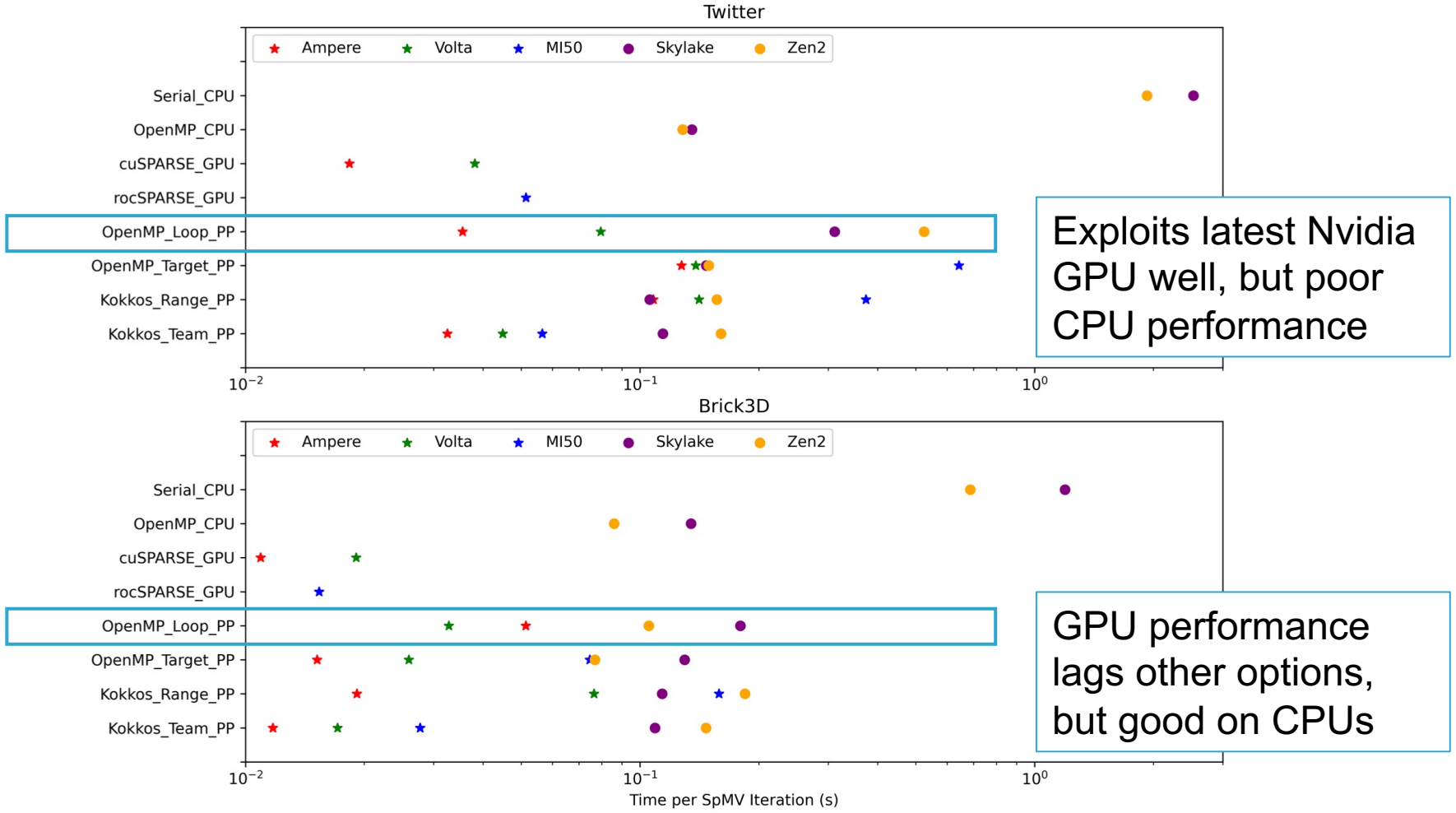
Results: Twitter Followers Network



Less success for performance portable [PP] methods

- Several significantly underperform CPU-only “classic” OpenMP
- Only the most complex (Kokkos `team`) is near the performance of both vendor math libraries on GPU and “classic” OpenMP on CPU

Descriptive OpenMP: `loop` Construct

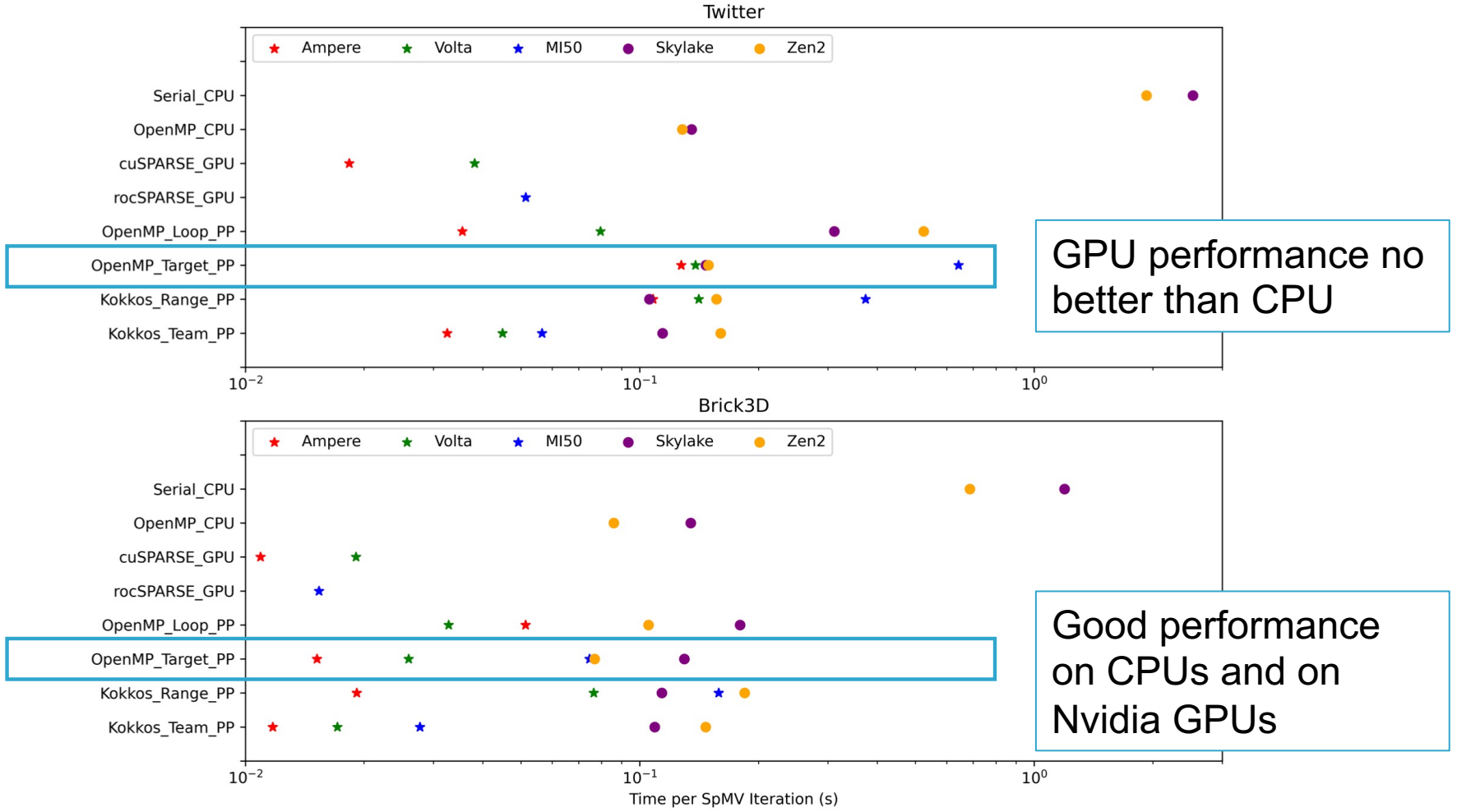


Exploits latest Nvidia GPU well, but poor CPU performance

GPU performance lags other options, but good on CPUs

Not maximizing performance portability

Prescriptive OpenMP: `target teams distribute parallel for`

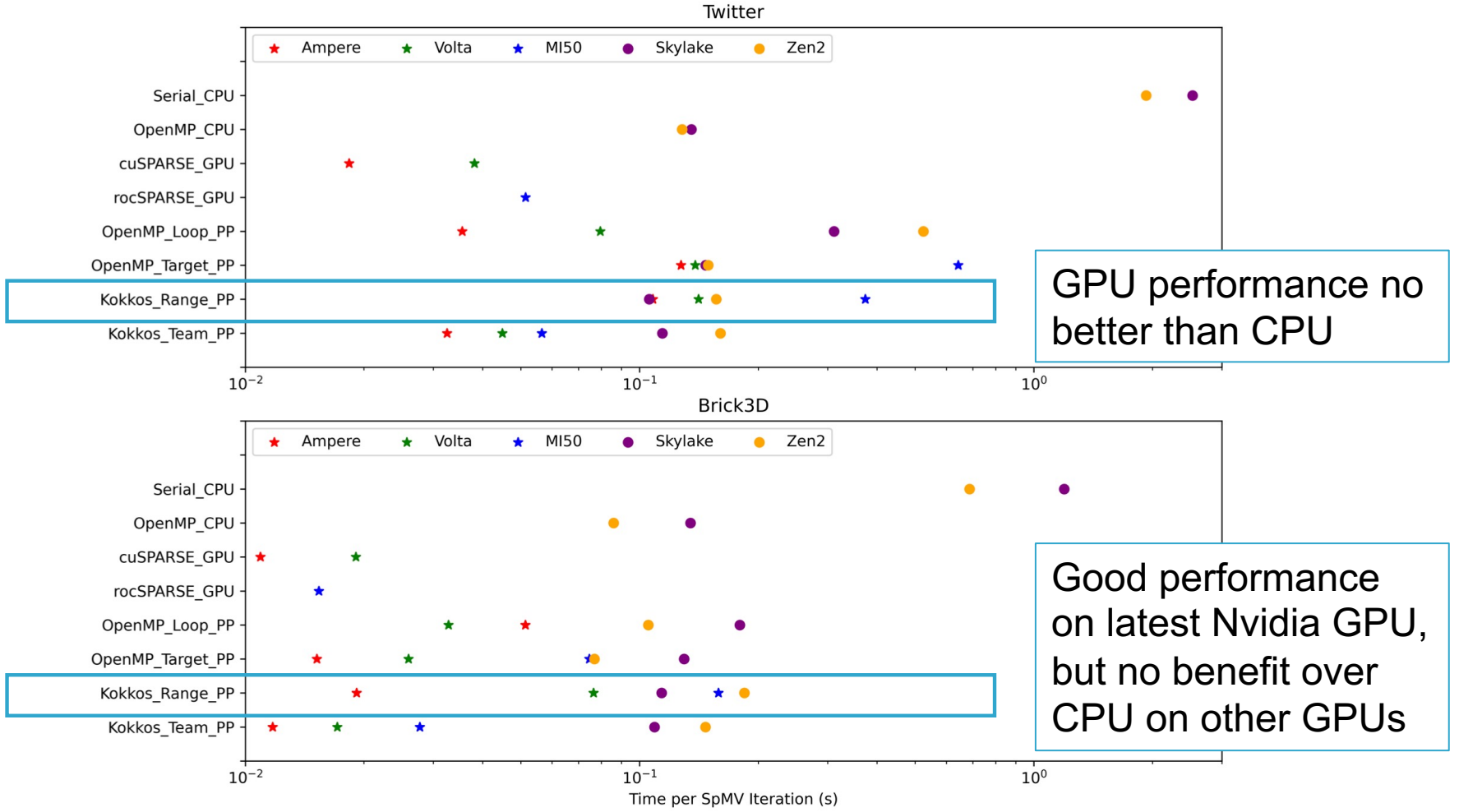


GPU performance no better than CPU

Good performance on CPUs and on Nvidia GPUs

Not maximizing performance portability

Basic Kokkos: `range` Policy (Single-Level Parallelization)



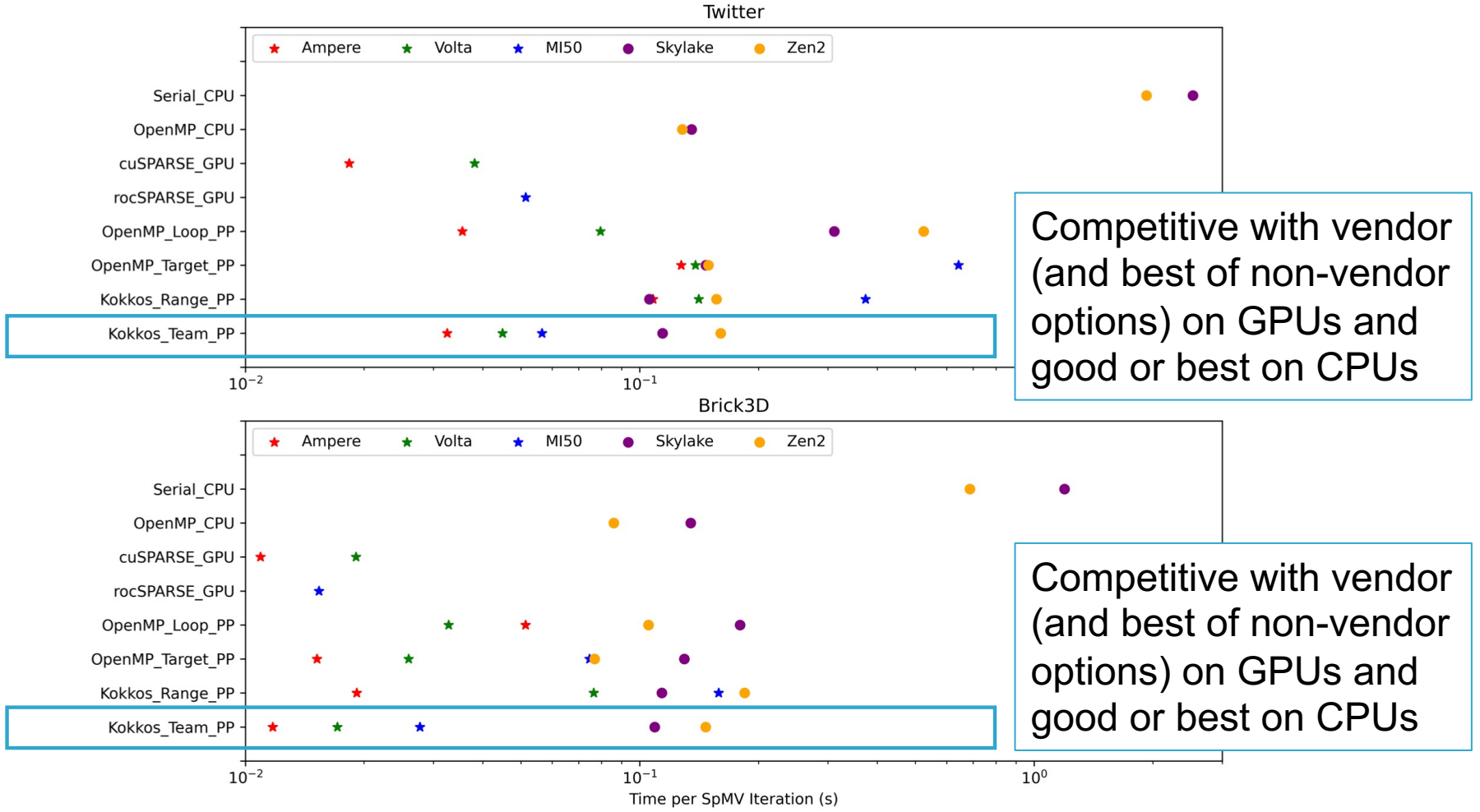
GPU performance no better than CPU

Good performance on latest Nvidia GPU, but no benefit over CPU on other GPUs

Not maximizing performance portability

More Complex* Kokkos: `team` Policy

* Explicitly parallelizing both levels of loops in code AND tuning runtime parameters



Competitive with vendor (and best of non-vendor options) on GPUs and good or best on CPUs

Competitive with vendor (and best of non-vendor options) on GPUs and good or best on CPUs

Maximizes performance portability on this kernel

Takeaways and Future Work

Performance portability is challenging

- Some programming models that were originally designed primarily for scientific computing struggle with our graph analytics input.
- Low-effort code often performs no better on GPU than multicore CPU, especially for the graph analytics input.
- Only more complex code with execution time tuning of key parameters demonstrated performance consistently at or approaching the level of nonportable GPU vendor libraries.

Key considerations for programmers

- Is my specific kernel available in a vendor library?
- Am I willing to work for performance portability?

Many possibilities for future work

- Try newer AMD hardware (MI100) and software (continually improving ROCm stack).
- Test other OpenMP compilers for GPU offload (GCC, Cray).
- Implement new versions in other programming models, e.g., OpenACC.