Exploring the Performance Portability of Modern Programming Languages

Brett Eiffert · Mohammad Alaul Haque Monil · Narasinga Rao Miniskar · William Godoy · Keita Teranishi · Jeffrey S. Vetter

Oak Ridge National Laboratory (eiffertbc@ornl.gov)

Goals

- Demonstrate and inform the scientific computing community of the state of performance portability of high-productivity programming languages.
- Evaluate a balance of relevant compute- and memory-bound scientific benchmarks.
- Target modern NVIDIA and AMD accelerators across multiple levels of precision.
- Observe the benefits and costs of built-in abstractions and compiler features for each programming model.

Languages and Frameworks

KernelAbstractions.jl	Triton	Native	CubeCL
Version 0.9.42	Version 3.7.1	Version 1.0.0	Version 0.10.0

MI300A · ROCm 7.14

H100 NVL · nvhpc 26.1

Methods

- Selected and ported four established scientific benchmarks to span the compute- and memory-bound limits of each target:
  - **Seven-point stencil** · memory-bound — updates grid points using neighboring points.
  - **BabelStream** · memory-bound — measures attainable memory bandwidth.
  - **Hartree-Fock** · compute-bound — approximates molecular electronic structure using quantum chemistry calculations.
  - **miniBUDE** · compute-bound — computationally scores molecular docking poses relevant to drug discovery.
- Validated every implementation against defined numerical error bounds.
- Idiomatic approach: languages were not forced to be what they are not.
- Preserved comparable parameters and configurations throughout.
- Trusted each compiler to make best performance decisions per benchmark.

Follow along on GitHub as this research matures.

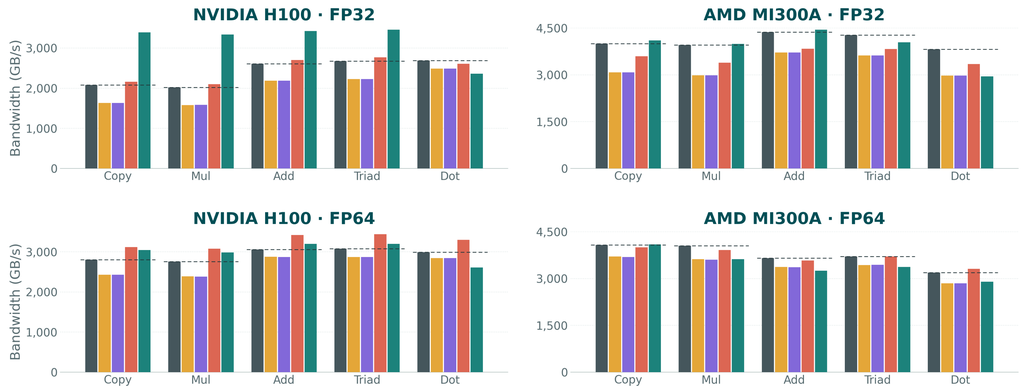
Results

Native · CubeCL · KernelAbstractions.jl · Mojo · Triton

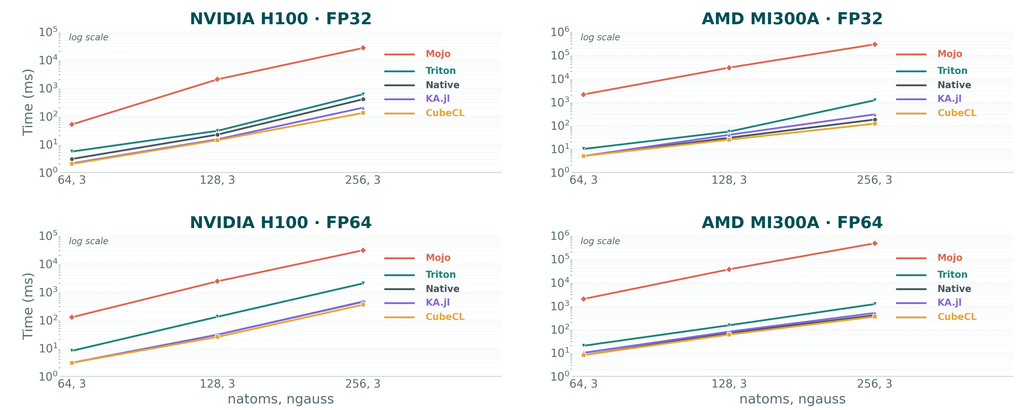
Seven-point stencil · memory-bound — higher is better



BabelStream · memory-bound — higher is better

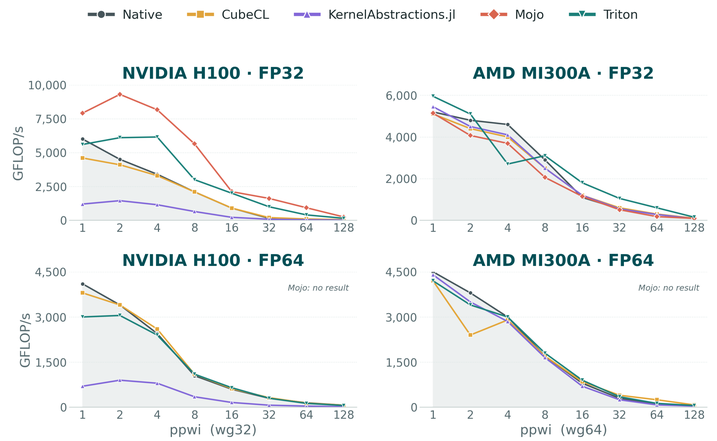


Hartree-Fock · compute-bound — lower is better



Results continued

miniBUDE · compute-bound — higher is better



Portability Summary

Performance relative to native — geometric mean over problem sizes and precisions, peak over the miniBUDE ppwi sweep. 1.00 matches native.

	Stencil		BabelStream		Hartree-Fock		miniBUDE	
	H100	MI300A	H100	MI300A	H100	MI300A	H100	MI300A
CubeCL	0.86	0.85	0.89	0.87	1.47	1.18	0.84	0.96
Kernel-Abstractions.jl	0.53	0.76	0.89	0.87	1.27	0.80	0.23	1.01
Mojo	0.93	1.02	1.07	0.93	0.02	0.00	1.55	0.99
Triton	0.77	0.84	1.15	0.95	0.40	0.38	0.87	1.02

× native (1.8 · 1.4 · 1.0 · 0.6 · 0.2)

Conclusions and Future Work

- Out-of-the-box modern language abstractions can outperform native implementations that are not heavily optimized.
- **KernelAbstractions.jl** struggled on memory-bound benchmarks but performed well on compute-bound AMD implementations.
- **Mojo** performed well across both vendors of the memory-bound operations but performed poorly with atomics and lacks support for FP64 trig functions.
- **Triton**'s tile-centric model and compiler delivered impressive, productive results in single-precision BabelStream implementations.
- **CubeCL** delivered consistent performance relative to native implementations across all benchmarks, but some programming productivity concerns exist.
- This work was the first step in exploring the performance portability of modern programming languages. More work in this area is currently underway.

Acknowledgements

This work was supported by the UT-Battelle LLC through U.S. Department of Energy (DOE) under Contract DE-AC05-00OR22725. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research Program, through "Competitive Portfolios" DE-FOA-0003264, under award number DE-SC0025645.

This research used the same benchmarks as a 2025 paper (Godoy et al.) on the Mojo programming language. (https://dl.acm.org/doi/10.1145/3731599.3767573).

This research used resources of the Experimental Computing Laboratory (https://excl.ornl.gov) at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

The authors used generative AI and AI-assisted tools, including frontier large language models, to support background research, literature discovery, text editing, data analysis, code development, and the preparation of graphs and figures. All AI-assisted outputs were reviewed, verified, edited, and integrated by the authors, who take full responsibility for the accuracy, originality, interpretation, and integrity of the final work.