Mojo: Python-Like MLIR-Based GPU Portable Science Kernels

Tatiana Melnichenko[1,2] (tdehoff@vols.utk.edu)
William F. Godoy[2] (advisor), Pedro Valera-Lara[2] (advisor)
[1]Innovative Computing Laboratory, University of Tennessee, Knoxville | [2]Oak Ridge National Laboratory

Motivation

- **Mojo:** Python-like and first MLIR-based language promising high productivity, memory safety, performance, and **out-of-box GPU portability** since June '25
- **Why another new language?** Vendor-neutral computing, reduce the two-language tax (C++ and Python), adds performance to fragmented AI workflows (PyTorch + X)
- **Our goal:** Assess Mojo's out-of-the-box GPU performance portability and tooling on NVIDIA H100 and AMD MI300A for a range of scientific workloads

Mojo vs Other Languages

Feature/ Language	Mojo	C++	Python	Julia	Rust
Compile Model	AOT, JIT	AOT	JIT Interpreted	JIT, AOT (Exp)	AOT
Memory Safety	Planned (static + dynamic checks)	Manual or RAII	Garbage collected	Garbage collected	Strong (ownership, borrowing)
GPU Support	Native, portable (MLIR)	CUDA, HIP, OpenMP, OpenACC, SYCL, OpenCL, RAJA, Kokkos	pycuda, CUDA Python	CUDA.jl, AMDGPU.jl KernelAbstractions.jl JACC.jl	Experimental
Packaging	pixi	Third-party	Third-party	Julia Pkg.jl	cargo

Approach

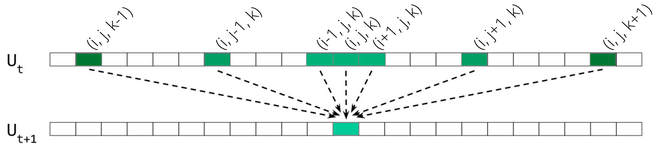
- We ported the workloads matching the existing optimized C++ CUDA and HIP codes without vendor-specific knobs
- Performance metrics: - memory bandwidth for memory-bound
  - FLOP/s for miniBUDE, wall clock time for Hartree-Fock

GPU - Memory	Theoretical Peaks		
	Bandwidth *(GB/s)*	FP32 *(TFLOPS/s)*	FP64 *(TFLOPS/s)*
NVIDIA H100 NVL - 94GB	3,900	60.0	30.0
AMD MI300A - 128GB HBM3	5,300	122.6	61.3

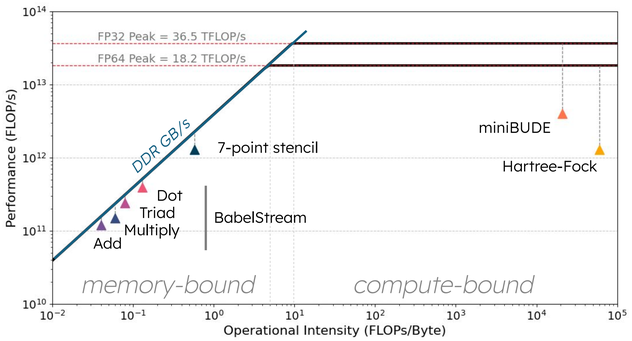
https://github.com/tdehoff/Mojo-workloads

Scientific Workloads

- **BabelStream**: Memory-bound Copy, Multiply, Triad, Add, Dot routines; commonly used for benchmarking (U of Bristol)
- **Seven-point stencil:** Memory-bound; used for modeling diffusion phenomena (AMD lab notes)



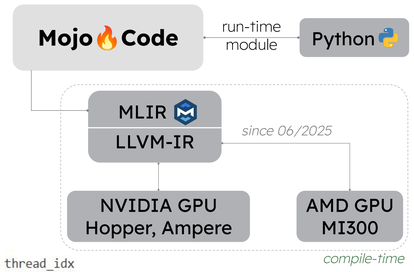
- **miniBUDE:** Compute-bound, models ligand-protein docking (U of Bristol)
- **Hartree-Fock**: Compute-bound with atomic operations; approximates the electron behavior in quantum systems (Argonne Nat Lab)



Roofline representation of the workloads using NVIDIA NSight on an H100 GPU

Mojo Programming

- MLIR-based Just-In-Time (JIT) or Ahead-of-Time (AOT) compiled language
- **Portable high-performance GPU programming**
- Run-time interoperability with Python and its ecosystem
- Memory-safe via variable lifetime and ownership
- Open-source by 2026



GPU-powered DOE HPC in the Top500


```
1  from gpu.host import DeviceContext
2  from gpu.id import block_dim, block_idx, thread_idx
3  from layout import Layout, LayoutTensor
4  ...
5  from python import Python
6  alias dtype = DType.float32
7  alias Nx = 1024
8  alias layout = Layout.row_major(Nx)
9  alias block_size = 256
10 alias num_blocks = ceildiv(Nx, block_size)
11 fn fill_one(tensor: LayoutTensor[mut=True, dtype,layout]):
12     var tid = block_idx.x * block_dim.x + thread_idx.x
13     if tid < Nx:
14         tensor[tid] = 1
15 fn main()
16     ctx = DeviceContext()
17     d_u = ctx.enqueue_create_buffer[dtype](nx)
18     u_tensor = LayoutTensor[dtype,layout](d_u)
19     ctx.enqueue_function[fill_one](u_tensor,
20         grid_dim=num_blocks,
21         block_dim=block_size
22     )
23     ctx.synchronize()
24     np = Python.import_module("numpy")
25     array = np.array(Python.list(1, 2, 3))
26     print(array)
```

Compile-time GPU programming Requires tensor type, size, and layout

GPU kernel launching

GPU memory model

GPU kernel execution

Python interoperability uses a separate runtime approach

Just-in-Time
> pixi run mojo prog.mojo

Ahead-on-Time
> pixi shell
> mojo build prog.mojo
> ./prog

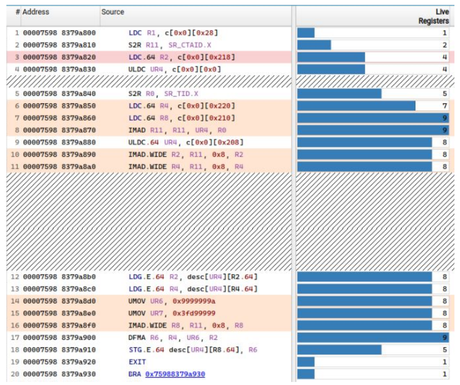
https://docs.modular.com/mojo/faq/

Performance: Mojo vs CUDA and HIP

Mojo can be profiled with NVIDIA Nsight Compute CLI (and AMD rocprof) with live registers

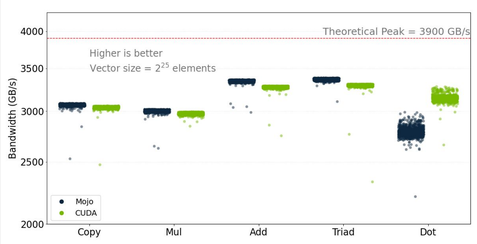
Mojo BabelStream Triad (from MLIR)

CUDA BabelStream Triad

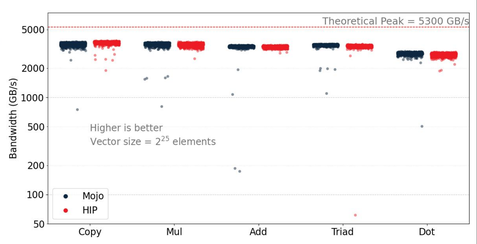


BabelStream *(memory-bound)*

- **NVIDIA H100**



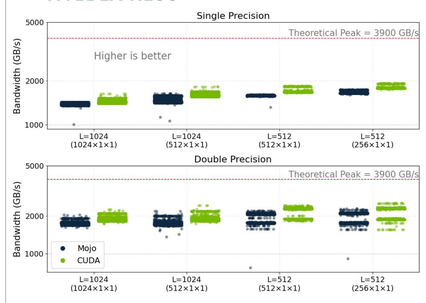
- **AMD MI300A**



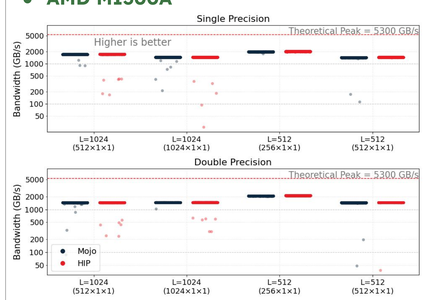
Performance (cont.)

7-point stencil *(memory-bound)*

- **NVIDIA H100**



- **AMD MI300A**



Performance Portability metric

- Adapted from Pennycook et al. (2021) and Marowka (2025)

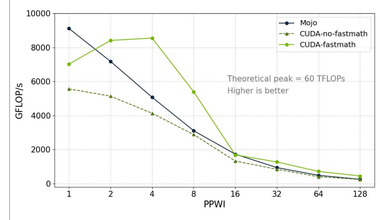
$$\overline{\Phi}_{Mojo} = \frac{\sum_{i \in T} e_i(a)}{|T|}$$

$$e_i(a) = \frac{Mojo_{perf-i}}{vendor(CUDA/HIP)_{perf-i}}$$

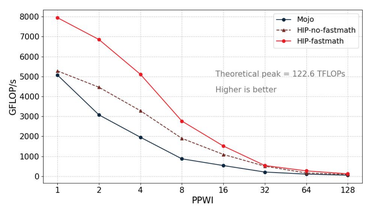
Mojo Efficiency	NVIDIA H100	AMD MI300A
7-point stencil		
FP32	0.82	1.00
FP64	0.87	1.00
$\overline{\Phi}$ = 0.92		
BabelStream		
Copy	1.01	1.00
Mul	1.02	1.00
Add	1.01	1.00
Triad	1.01	1.00
Dot	0.78	1.00
$\overline{\Phi}$ = 0.96		
miniBUDE		
PPWI=8 wg=8	0.82	0.38
PPWI=4 wg=64	0.59	0.38
$\overline{\Phi}$ = 0.54		
Hartree-Fock		
a=1024 ngauss=6	17E-3	–
a=256 ngauss=3	2.52	7E-3
a=128 ngauss=3	2.52	8E-3
a=64 ngauss=3	2.33	8E-3
$\overline{\Phi}$ = 0.92		

miniBUDE *(compute-bound)*

- **NVIDIA H100**



- **AMD MI300A**



Hartree-Fock *(compute-bound + atomic)*

Kernel execution duration *(ms)*	NVIDIA H100		AMD MI300A	
	Mojo	CUDA	Mojo	HIP
a=1024, ngauss=6	147,250	2,652	-	846
a=256, ngauss=3	187	472	25,266	178
a=128, ngauss=3	21	53	2,765	23
a=64, ngauss=3	3	7	436	4

Conclusions and Observations

1. **Performance Portability:** Mojo is on par with CUDA and HIP on memory-bound workloads; for compute-bound, Mojo lacks fast-math; Mojo's atomic operations have high overhead on AMD MI300A, but outperform CUDA + NVIDIA H100 on small cases
2. **Compile-Time Model:** Might limit adoption due to the run-time nature of HPC
3. **Python Interoperability:** Requires linking to a Python runtime, 100% outside MLIR compilation
4. **Ecosystem and Tools:** Mojo works with NVIDIA NSight and AMD rocprof; library ecosystem must grow for real adoption in HPC (BLAS, LAPACK, MPI)

This project was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Community College Internships Program (CCI). This research used resources of the Experimental Computing Laboratory (ExCL) at the 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