Case Study for Performance Portability of GPU Programming Frameworks for Hemodynamic Simulations

Aristotle Martin[1], and Amanda Randles (Advisor)[1]

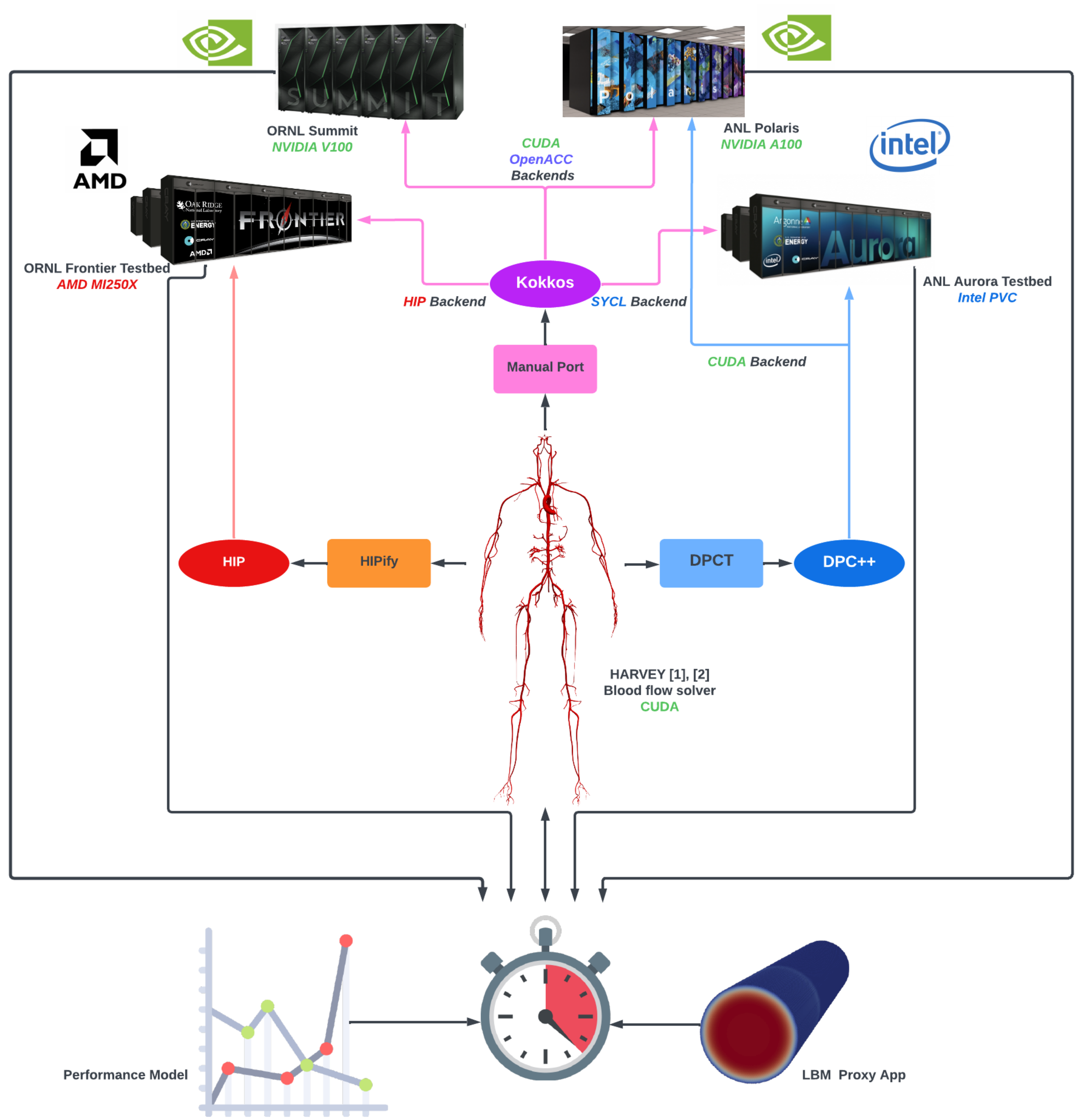
[1]*Department of Biomedical Engineering, Duke University*

Introduction

- With the advent of GPU-dense node architectures in exascale platforms, achieving vendor-agnostic performance has become critical
- Porting legacy codes to run on current systems can be non-intuitive given the large number of heterogenous programming models available
- Proxy applications and performance models can facilitate rapid prototyping on new systems and help gauge performance bounds of full applications

Methodology

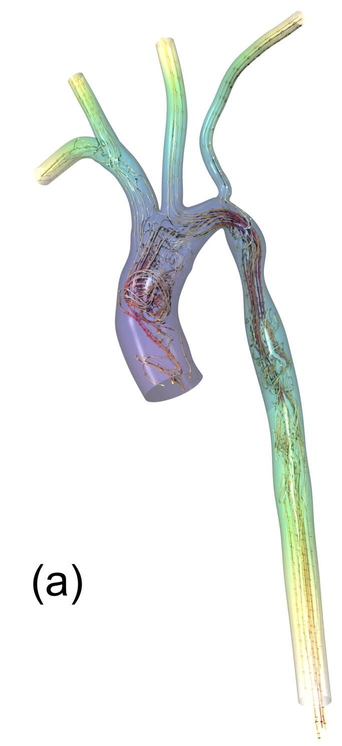
- Ported a massively parallel fluid dynamics application, HARVEY [1], as well as a proxy app, from CUDA to SYCL/DPC++, HIP, Kokkos + backends using manual handtuning and automated assist tools
- Runs conducted on Summit (ORNL/NVIDIA V100), Polaris (ALCF/NVIDIA A100), Crusher (ORNL/AMD MI250X) and Sunspot (ALCF/Intel PVC)
- Compared performance (millions of fluid lattice updates per second) of HARVEY against LBM proxy app and GPU performance model



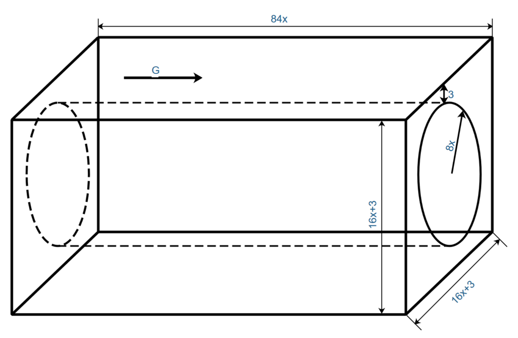
Applications Overview

- HARVEY is an LBM-based, computational fluid dynamics code capable of simulating blood flow in image-derived vasculature at cellular resolution, like the aorta shown in (a)
- We developed an open-source proxy app based on the LBM that can solve fluid flows in simple geometries as shown in (b)

(a)



(b)



Lattice Boltzmann Method

- The lattice Boltzmann method is used to model fluid flow
- Nearest-neighbor communication pattern lends LBM to parallelization

$$f_i(x + c_i, t+1) = \left(1 - \frac{1}{\tau}\right) f_i(x,t) + \frac{1}{\tau} f_i^{eq}(x,t) + F_i(x,t)$$

GPU Performance Model

- We extend a forecast model we previously developed for CPUs [2] to predict scaling performance on GPU nodes
- Time is estimated from memory bandwidth measured with BabelStream [4] and communication times collected from custom pingpong benchmark:

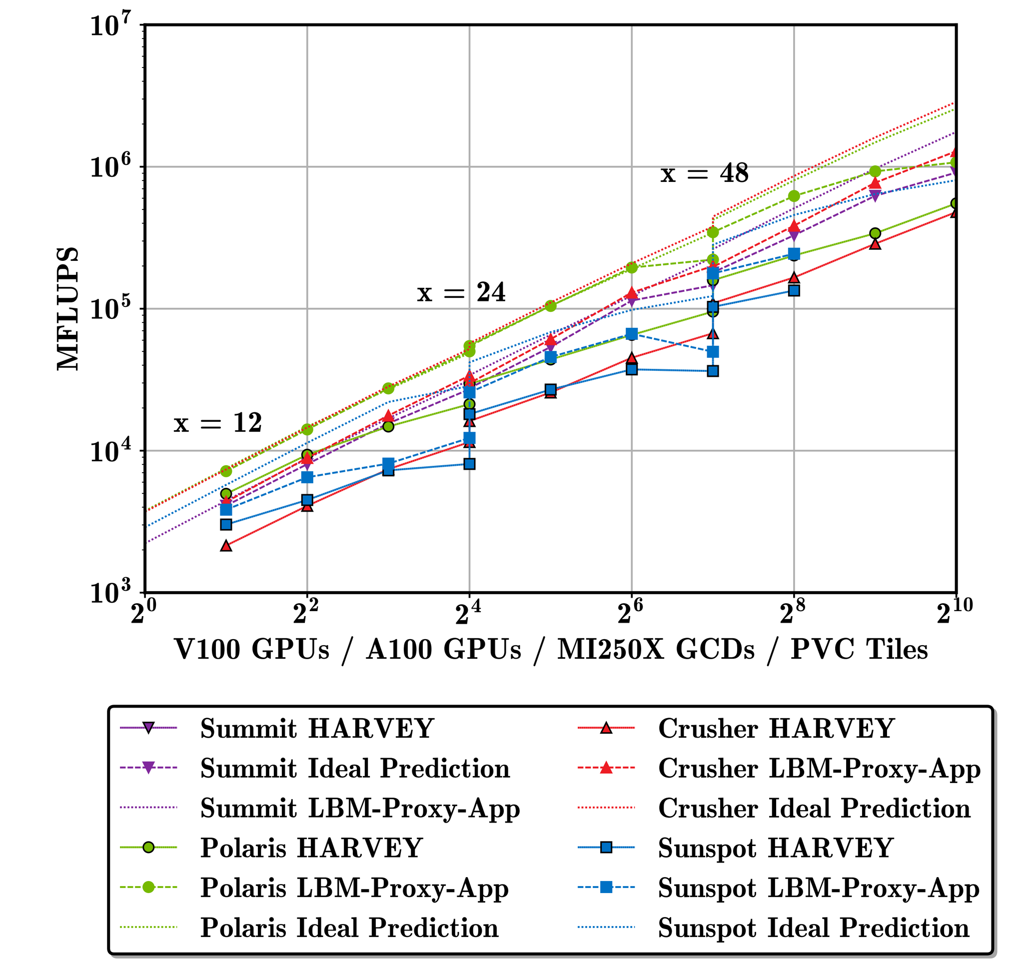
$$t = t_{streamcollide} + \sum_{j}^{n_{events}} t_{comm_j}$$

where

$$t_{streamcollide} = \frac{n_{bytes}}{B_{mem}}$$

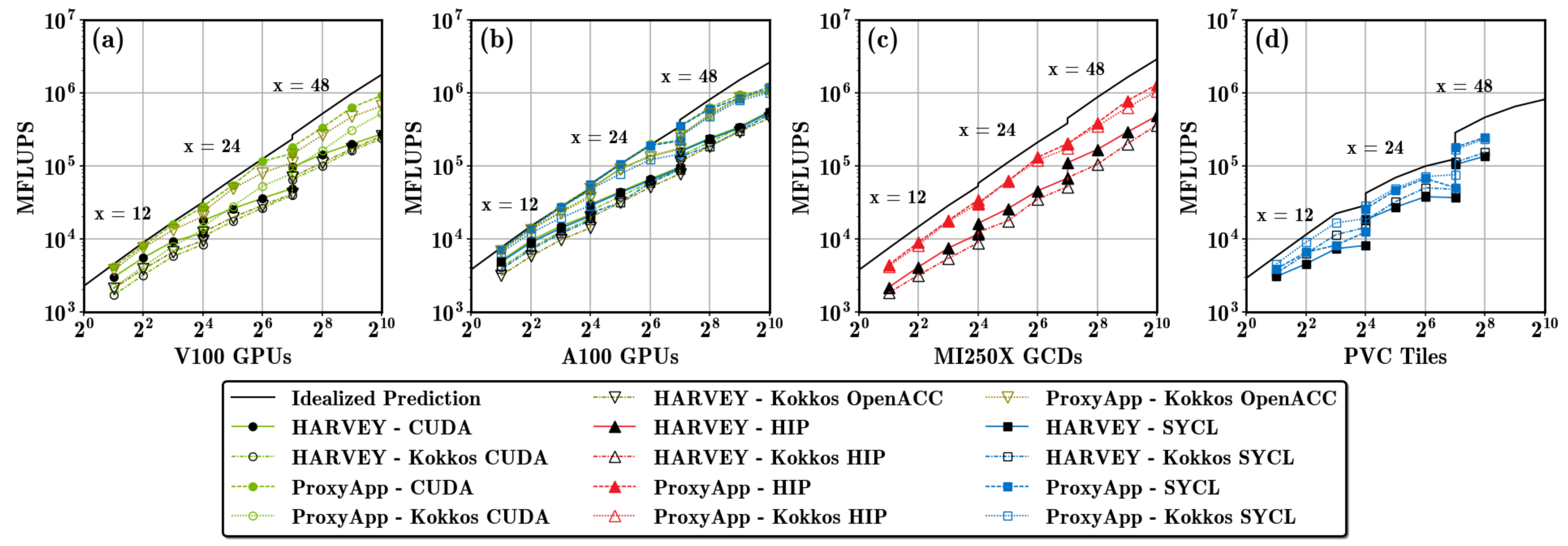
Results: Hardware Comparison

- Compared native programming models on each system for HARVEY, LBM proxy app, and performance model through strong-weak scaling



Results: Backend Comparison

- Compared programming model backends against native language on Summit (a), Polaris (b), Crusher (c), and Sunspot (d)



Lessons Learned

- With backends for CUDA, SYCL, HIP, and OpenACC, the Kokkos version of the HARVEY application was most portable but required the most porting effort
- The HIP codes required the least porting effort but were the most limited in portability
- Out-of-the-box machine-generated SYCL and HIP ports were competitive as informed by performance predictions and proxy application
- Native programming models generally outperformed off-brand models
- Performance predictions and proxy applications proved invaluable tools for navigating porting process and facilitating manual tuning efforts

References:

[1] Randles, Amanda Peters, et al. "Performance analysis of the lattice Boltzmann model beyond Navier-Stokes." *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*. IEEE, 2013.
[2] Ladd, William, et al. "Optimizing Cloud Computing Resource Usage for Hemodynamic Simulation." *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2023.

Acknowledgements:

Computing support for this work came from the Argonne National Laboratory (ANL) Aurora Early Science program. An award of compute time was provided by the INCITE Program. This research also used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

SC23 DENVER NOV 12-17