Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

A Performance Analysis of Modern Parallel Programming Models Using a Compute-Bound Application

Duy Dung Nguyen

25.01.2022

Seminar Supercomputer: Forschung und Innovation

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

Table of contents

Introduction

Modern Parallel Programming Models

- A set of program abstractions for fitting parallel activities from the application to the underlying parallel hardware.
- Shared memory and threads.
- The C++ language.
- OpenMP (for CPU and offload), OpenCL, CUDA, OpenACC, Kokkos and SYCL.

Performance portability

- What is Performance portability?

→ "Performance portability means the same source code will run productively on a variety of different architectures" (Larkin)

- Portability is a common concern for developers—and users—of modern programming models.

miniBUDE as a benchmark

cuda	Undo accidental debug changes
data	Add raw mol2/bhff dataset and bm2_long
kokkos	kokkos: add support for replacing -isystem on demand
makedeck	Add raw mol2/bhff dataset and bm2_long
miniBUDE.jl	julia: update dependencies
openacc	Rename to miniBUDE and add preferred citation
opencl	Rename to miniBUDE and add preferred citation
openmp-target	Rename to miniBUDE and add preferred citation
openmp	Rename to miniBUDE and add preferred citation
sycl	Rename to miniBUDE and add preferred citation
.gitignore	Add raw mol2/bhff dataset and bm2_long
ACKNOWLEDGEMENTS.txt	Add acknowledgement
CONTRIBUTING.md	Initial commit: OpenMP and OpenCL implementations
LICENSE.txt	Initial commit: OpenMP and OpenCL implementations
README.md	julia: initial Julia implementation

- A mini-app created from the core computation kernel for BUDE (Bristol University Docking Engine).
- Each subdirectory contains a separate C/C++ implementation.

miniBUDE is a compute-bound application

Compute-bound (CPU bound)

- means the rate at which process progresses is limited by the speed of the CPU.
- Example: multiplying small matrices.

Memory-bound

- means the rate at which a process progresses is limited by the amount memory available.
- Example: multiplying large matrices.

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Performance Analysis

Hardware platforms and compilers used

Platform	Abbrev.	Type	Cores	Clock speed	Peak SP performance
Intel Skylake 8176	SKL	CPU	2 × 28	2.1 GHz	5,734 GFLOP/s
Intel Cascade Lake 6230	CXL	CPU	2 × 20	2.1 GHz	4,096 GFLOP/s
AMD Rome 7742	Rome	CPU	2 × 64	2.25 GHz	9,216 GFLOP/s
Marvell ThunderX2	TX2	CPU	2 × 32	2.5 GHz	2,560 GFLOP/s
Fujitsu A64FX	A64FX	CPU	48	1.8 GHz	5,530 GFLOP/s
NVIDIA V100	—	GPU	80	1.13 GHz	15,700 GFLOP/s
AMD Radeon VII	—	GPU	60	1.4 GHz	13,800 GFLOP/s
Intel Iris Pro 580	—	GPU	72	0.95 GHz	1,094 GFLOP/s

- Aggressive compiler optimization flags to the level of ***–march=native -Ofast***.

Compiler	CPUs	GPUs	Frameworks
AOCC 2.3	X		m k s
AOMP 11.0		M	m
Arm Compiler 21.0	R		m k s
ComputeCpp 2.1.1	X	I	m k s
Cray Compiler 10.0	R X	N	a[1] m k s
Fujitsu Compiler 4.3	R		m k s
GCC 10.3	R X	M N	a l m k s
Intel ICX 2019	X		m k[2] s
Intel DPC++ 2021.1	X	N	m k s
LLVM 11.0	R X	N	m k s
NVCC 10.2		N	c
PGI 19.10		N	a

CPUs: **A**R**M**, **X**86; GPUs: A**M**D, **N**VIDIA, **I**NTEL
Frameworks: **c**uda, open**a**cc, open**c**l, open**m**p, **k**okkos, **s**ycl

[1] Version 9.0 only; [2] With the experimental `INTEL_GEN` backend.

Performance Analysis

- On CPU platforms, hardware counters were accessed through the built-in Linux **perf** tool.
- On GPUs, NVIDIA CUDA profiler and the OpenCL intercept layer were used.

Results of the performance analysis

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

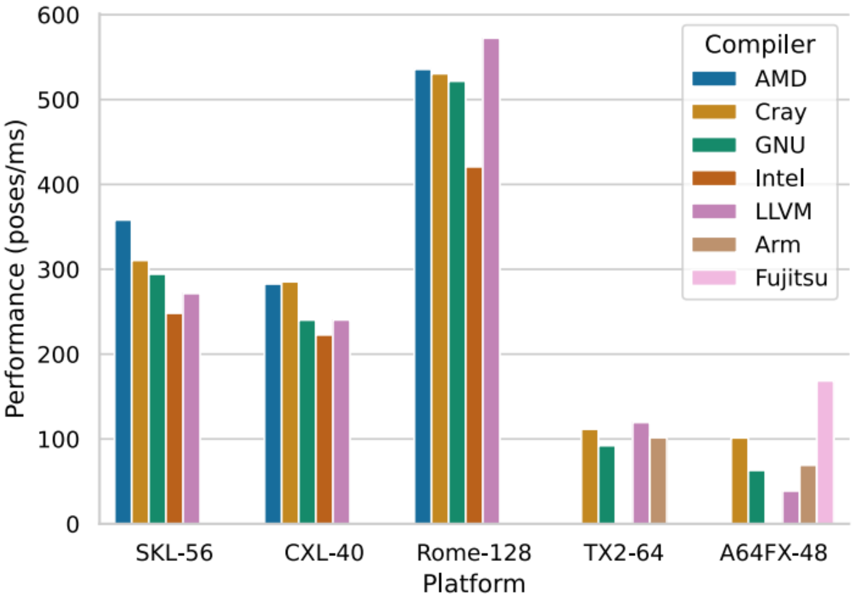
OpenMP, Kokkos und SYCL

CPUs

OpenMP - Introduction

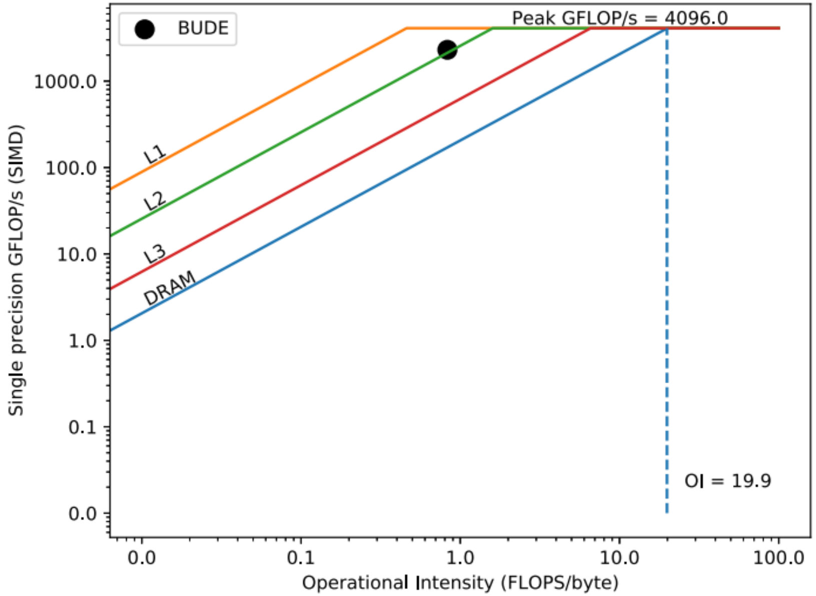
- An Application Program Interface (API)
- Gives parallel programmers a simple and flexible interface for developing portable parallel applications.
- The API is specified for C/C++ and Fortran
- API components:
  - Compiler Directives
  - Runtime Library Routines
  - Environment variables

OpenMP - Performance on CPUs



- There is no OpenMP implementation that works optimal on every single platform.
- AMD seems to be a good choice.

OpenMP - Performance on CPUs

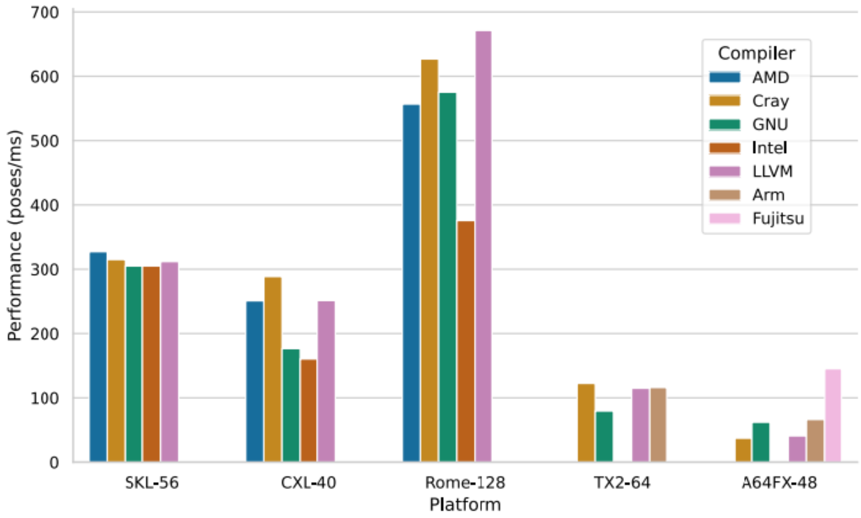


- Cache-aware roofline for the Cascade Lake platform showing the achieved performance for miniBUDE.

Kokkos - Introduction

- Parallelism expressed via the idiomatic Kokkos:: ***parallel_for*** function.
  - ParallelFunctor functor;
    Kokkos::parallel_for(numberOfIterations, functor);
- A C++ compiler is the only requirement.
- Kokkos lets you write algorithms once and run on many architectures, including both CPUs and GPUs.

Kokkos – Performance on CPUs



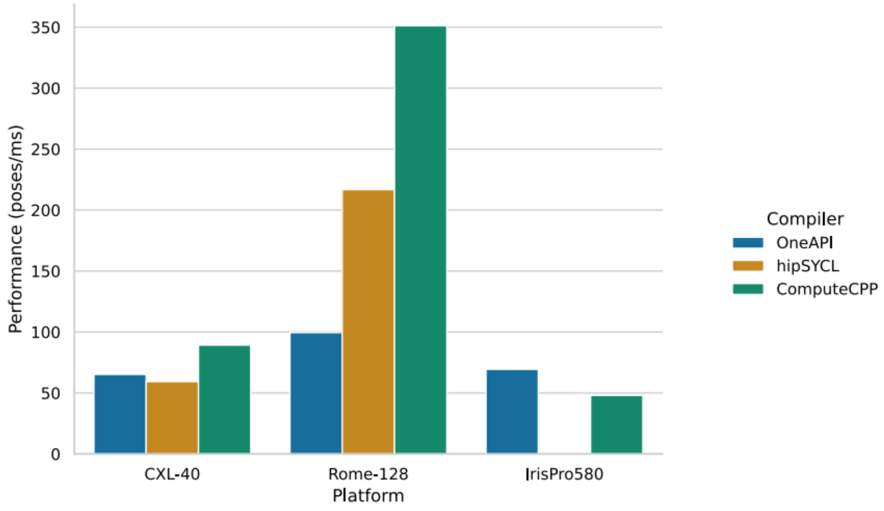
- The results shows a strong correlation compared to the OpenMP implementation results.

SYCL - Introduction

- Enables code to be written in a “single-source” style using completely standard C++.
- The kernel is a direct port of the OpenCL version.
- For comparison, a separate kernel that is closer to the OpenMP was implemented.

UHH Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

SYCL - Performance on CPUs



- Only results on platforms where at least two implementations were supported.
- The Skylake platform is missing from these results.

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Open CL, CUDA, OpenMP Offload, Open ACC, Kokkos und SYCL

GPUs

Low-level: OpenCL and CUDA

OpenCL

- A framework for writing programs that execute across heterogeneous platforms consisting of CPUs, GPUs and other processors.

CUDA

- Is a parallel computing platform and programming model developed by Nvidia for general computing on its own GPUs.

Low-level: OpenCL and CUDA

- On the NVIDIA V100: the CUDA implementation was 18% faster than the OpenCL code.
- Both versions also ran on the AMD Radeon VII, but OpenCL was 1.6× faster on this platform.
- CUDA and HIP cannot be used on the Intel GPU.

Directives-Based: OpenMP Offload and OpenACC

OpenMP Offload

- In OpenMP API 4.0, the specification provides a set of directives to instruct the compiler to offload a block of code to the device.

OpenACC

- A programming standard for parallel computing on accelerators (mostly on NIVDIA GPU).

Directives-Based: OpenMP Offload and OpenACC

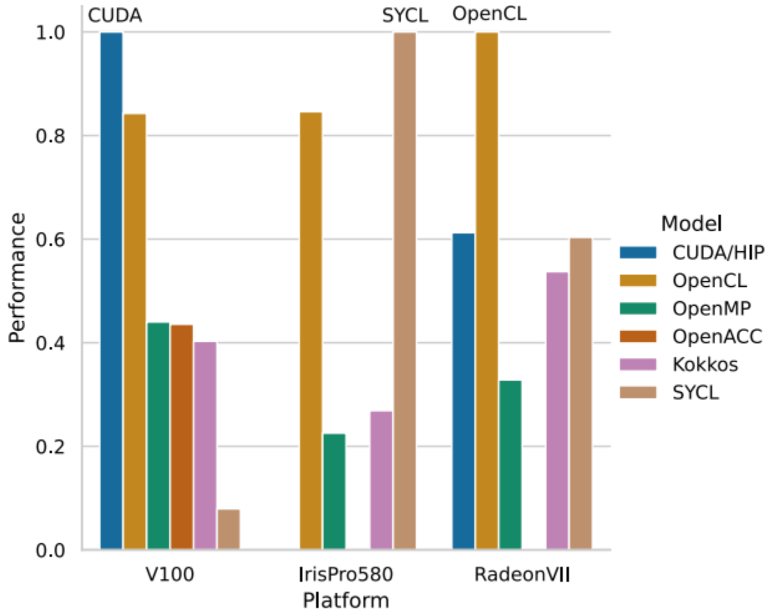
- Virtually identical performance on the V100.
- The directives-based approach showed about 0.4× the performance of the optimized CUDA code.
- On the Radeon, OpenACC was two orders of magnitude slower than OpenMP, which in turn only reached 0.3× the performance of the fastest model, OpenCL.
- On the Intel GPU, OpenMP ***target*** reached only 0.2–0.3× the performance of the fastest model (SYCL).

High-level: Kokkos and SYCL

- Kokkos and SYCL both run on all the GPUs studied, but only one implementation, hipSYCL, runs on AMD and NVIDIA.
- Kokkos: the code run on the GPU platforms was unchanged from the version run on CPUs.

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Performance of the GPU implementations

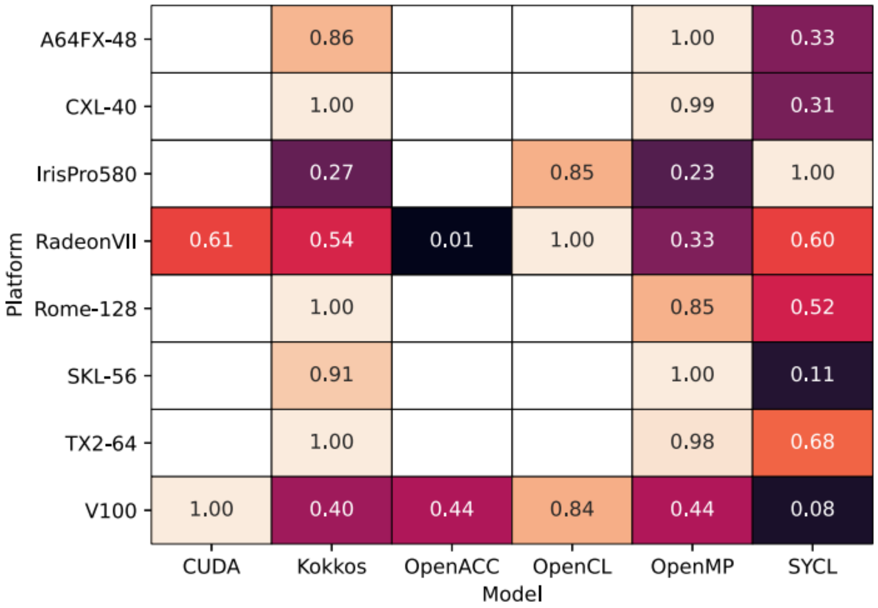


- A direct performance comparison between these platforms is not useful.
- CUDA (V100) = 2× OpenCL (RadeonVII) = 14× SYCL (Iris Pro580)

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Towards achieving performance portability - Summary

Achieved performance across all programming models



- No programming model can currently achieve optimal performance on all platforms.
- Kokkos was the only framework that was able to support all CPU and GPU platforms in one package.

Towards achieving performance portability

- Performance was lower with hipSYCL compared to Kokkos or plain OpenMP.
- Portability between CPUs and GPUs remains a concern.
- When running several iterations of a benchmark, the first run was usually up to 2× slower than subsequent runs.

Summary

- True performance portability is still out of reach.
- On GPUs, low-level APIs continue to provide the highest possible performance.
- Kokkos emerged as a reliable choice, and OpenMP remains in a strong position.

References

1. Andrei Poenaru , Wei-Chen Lin and Simon McIntosh-Smith, et al.: A Performance Analysis of Modern Parallel Programming Models Using a Compute-Bound Application. https://research-information.bris.ac.uk/en/publications/a-performance-analysis-of-modern-parallel-programming-models-usin.

2. Copyright 2018 OpenMP Architecture Review Board: OpenMP 5.0 API Syntax Reference Guide. https://www.openmp.org/wp-content/uploads/OpenMPRef-5.0-111802-web.pdf.

3. Aleksandar Vitorović, Veljko M. Milutinović, et al. in Advances in Computers, 2014: Manual Parallelization Versus State-of-the-Art Parallelization Techniques. https://www.sciencedirect.com/topics/computer-science/parallel-programming-model.

4. Prof. Dr. Thomas Ludwig, et al.: Hochleistungsrechnen – Vorlesung im Wintersemester 2021/2022. https://wr.informatik.uni-hamburg.de/teaching/wintersemester_2021_2022/hochleistungsrechnen.