Performance Portability With *alpaka*

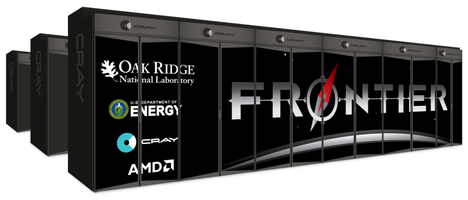
J. Stephan[1,2,4], S. Bastrakov[2], A. Di Pilato[1,2], S. Ehrig[2],
B. M. Gruber[1,2,3,4], J. Vyskočil[1,2], R. Widera[2], M. Bussmann[1,2]

[1]Center for Advanced Systems Understanding, [2]Helmholtz-Zentrum Dresden-Rossendorf, [3]CERN, [4]TU Dresden

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

Motivation

- Hardware configurations of supercomputers and workstations become increasingly heterogeneous.
- Programmers and scientists have to adapt to different processor architectures and accelerator types.
- **Goal: A stack of abstraction layers which forms a performance-portable ecosystem.**



Challenges

- How do we express the problem in an abstract and user-friendly fashion?
- How do we turn these expressions into hardware-aware algorithms and data structures?
- How do we achieve maximum performance without sacrificing portability?

$$\vec{y} \leftarrow a\vec{x} + \vec{y}$$

The *Caravan* Ecosystem

alpaka – abstraction layer for parallel kernels

LLAMA – low-level abstraction layer for memory access

vikunja – high-level parallel algorithms based on alpaka

bactria – performance analysis library

mallocMC – on-device memory allocator

RedGrapes – task-graph library

Hardware Support Library – planned

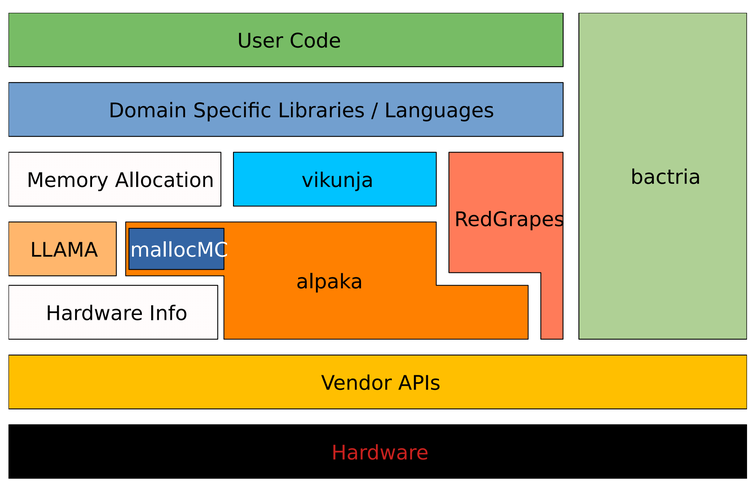
Memory Allocation Library – planned

Mapping To Hardware Models

- *alpaka* utilizes C++17 metaprogramming facilities to resolve abstractions at compile time. This leads to performance which is very close or (in many instances) equal to code programmed in the native model.
- The usage of modern C++ hides verbose implementation details from the user and instead offers a concise programming interface.
- **Modern C++ contributes to our goal of performance-portable software.**

New Features in 2022

- *alpaka* now features experimental support for Intel oneAPI and compatible SYCL implementations. This enables support for Intel hardware (CPUs, GPUs, FPGAs) as well as AMD/Xilinx FPGAs.
- *alpaka* offers full support for single- and double-precision complex numbers, including various math functions.
- *alpaka* has a new abstract Philox-based (pseudo-)random number generator that works in a uniform way across different platforms and devices, producing the exact same numbers on all of them.



Benefits

- *alpaka* makes code portable across vendors and hardware types while enabling full performance on all of them.
- *alpaka* makes code future-proof. Planning for future hardware is no longer a problem for the programmer.
- ***alpaka* reduces the amount of technical debt in HPC codes.**

SPONSORED BY THE Federal Ministry of Education and Research

SACHSEN This project is co-financed by the Saxon State government out of the State budget approved by the Saxon State Parliament