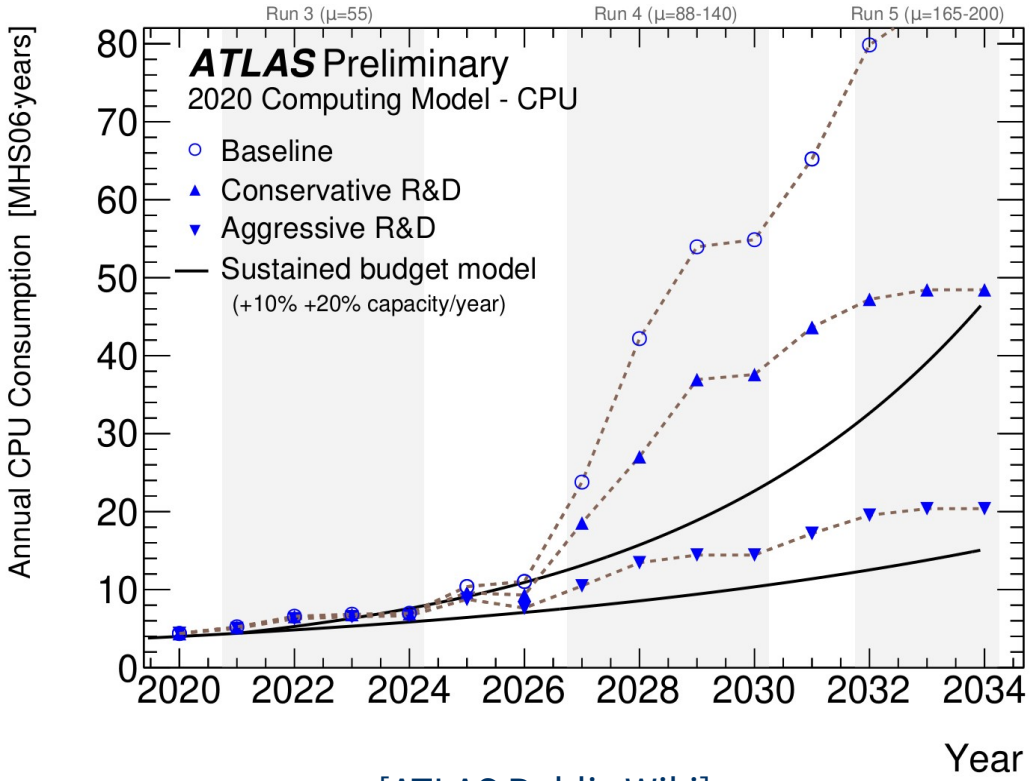
Exploring Heterogeneous Architectures in Track Reconstruction Software

CERN School of Computing, June 2021

Georgiana Mania
georgiana.mania@desy.de

Data Science in Hamburg
HELMHOLTZ Graduate School
for the Structure of Matter

The Concern



[ATLAS Public Wiki]

- Major increase in computational costs in future runs of the LHC
- Aggressive R&D strategies rely on heterogeneous architectures to deliver
  - increased performance (e.g. FLOPS)
  - descreased enery consumption (e.g. FLOPS per Watt)

  ⇨ **Code efficiency** is essential!
- World fastest supercomputers provide upgraded infrastructure to support scientific operations

Supercomputers

Accelerator Types

- Co-processors (e.g. Intel Xeon Phi)
- Graphics processing unit (GPU)

} Most widely used!

- Field-programmable gate arrays (FPGA)
- Application-specific integrated circuits (ASIC)
- System-on-Chip (SoC) (e.g. Raspberry Pi)

High-end GPU Models & Vendors

- NVIDIA V100/A100 - Summit Supercomputer
- Intel Xe HPC „Ponte Vecchio “ - Aurora Supercomputer (2022)
- AMD MI100 INSTINCT - El Capitan Supercomputer (2023)

! Code writen for a specific platform or vendor does not always work and most likely will not share the same level of performance on the others.

⇨ **Code portability** is essential!

IBM Summit, US#1 (NVIDIA V100 & A100)

JUWELS Juelich, EU#1 (NVIDIA V100 & A100)

[Top500 list]

Portable Solutions

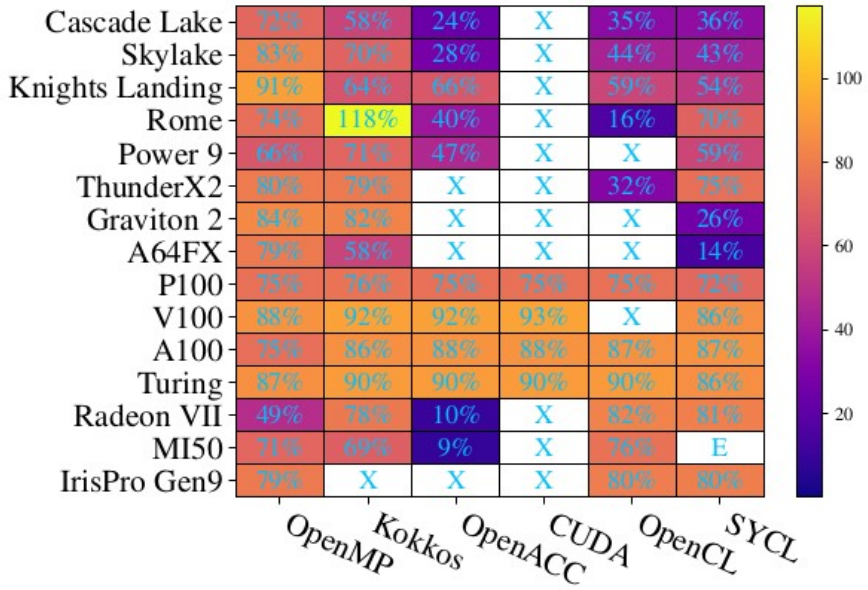
Standards & APIs

- OpenMP – vendor-agnostic CPU/GPU
- OpenACC – vendor-agnostic CPU/GPU
- SYCL with several implementations
  - Intel DPC++ - Intel hardware CPU/GPU
  - ComputeCPP – Intel CPU/GPU
  - HipSYCL – vendor agnostic CPU/GPU
  - … & others
- CUDA – CPU & NVIDIA GPU

Frameworks

- OneAPI (based on DPC++) - Intel CPU/GPU
- OpenCL – vendor-agnostic CPU, GPU, FPGA
- Kokkos – CUDA, HPX, OpenMP backends
- Alpaka – supports most of CPU/GPU vendors

Architectural efficiency = the fraction of the theoretical performance on a platform that an applicationcan achieve. [arXiv:1611.07409v1]



Architectural efficiency for dot product arrays of 2^{25} FP64 elements [DOI 10.1109/P3HPC51967.2020.00006]

Track Reconstruction on Heterogeneous Hardware

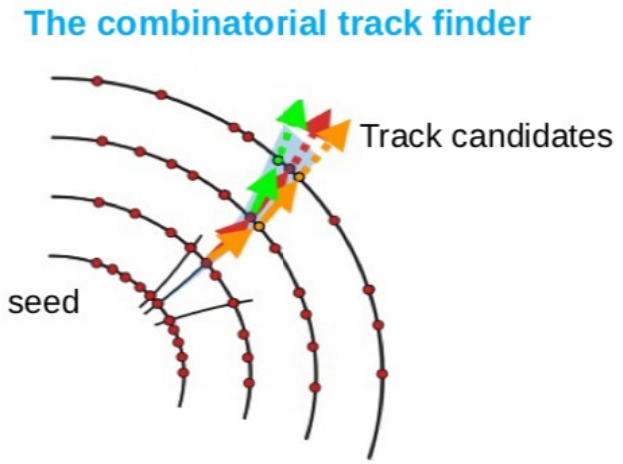
Exploiting parallelism on CPUs and/or GPUs

Pros

- the combinatorial nature of the problem
- high complexity calculations
- mostly small and predefined size data structures which can easily fit the L1/L2 caches

Cons

- large read-only data like detector geometries or magnetic field maps which does not fit the low level caches
- existing code has inefficient data structures (e.g. AoS) and data access patterns (e.g. row-major layout for GPU matrices)



Time [%] \ Run	Run 2 (μ~60)	HL-LHC (μ~200)
Seed finding time / Reconstruction time	20%	50%

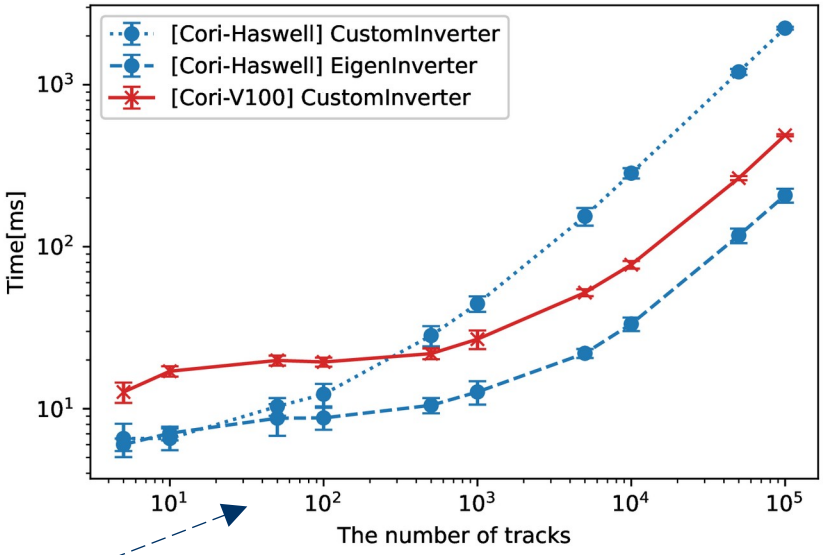
e.g. ATLAS (inhomogeneous) field map = ~300 MB

⇨ **Flexible** and **extendable code** is essential.

R&D Results

- Track reconstruction implementations in CUDA (targeting CPUs and NVIDIA GPUs)
  - ALICE [arXiv:1712.09430]
  - LHCb (Allen Framework) [arXiv:1912.09161]
  - CMS (Patatrack) [arXiv:2008.13461]
  - ATLAS [arXiv:2103.14737]

- ACTS project – an open source, experiment-independent set of track reconstruction tools
  - Github repository: https://github.com/acts-project/acts
  - On-going R&D work exploring CUDA, SYCL, KOKKOS, HIP, Intel TBB and OpenMP to efficiently target different hardware resources
  - Track Fitting [arXiv:2105.01796]



My Research

- A consensus on the best approach to ensure **code efficiency**, **portability** and **maintainability** while preserving the precision of the results has not been reached yet.

My research questions

- Is it even possible to cover all these requirement in depth?
- How to better achieve portability without having different code repositories for different hardware and different vendors?
- Could a Domain Specific Language supported by a compiler help us?
- Can heterogeneous code deliver enough performance (e.g. both time and precision wise) so to replace custom implementations (like CUDA based)?

⇨ **CERN School of Computing** is essential! (to me )