Task 1.7:
Common Software Developments for Heterogeneous Architectures

Jolly Chen on behalf of Task 1.7

NGT Workshop 25.11.2024

CERN | NexTGen Next Generation Triggers

Overview

Goal **Common developments and improvements** to make efficient use of **accelerator** (*GPU* and *FPGA*) devices in software for High-Luminosity LHC. Existing implementations should be **harmonized** between the experiments, and **optimization efforts** should be **shared**.

Parties ALICE, ATLAS, CMS, LHCb, EP-SFT, IT-FTI-PSE

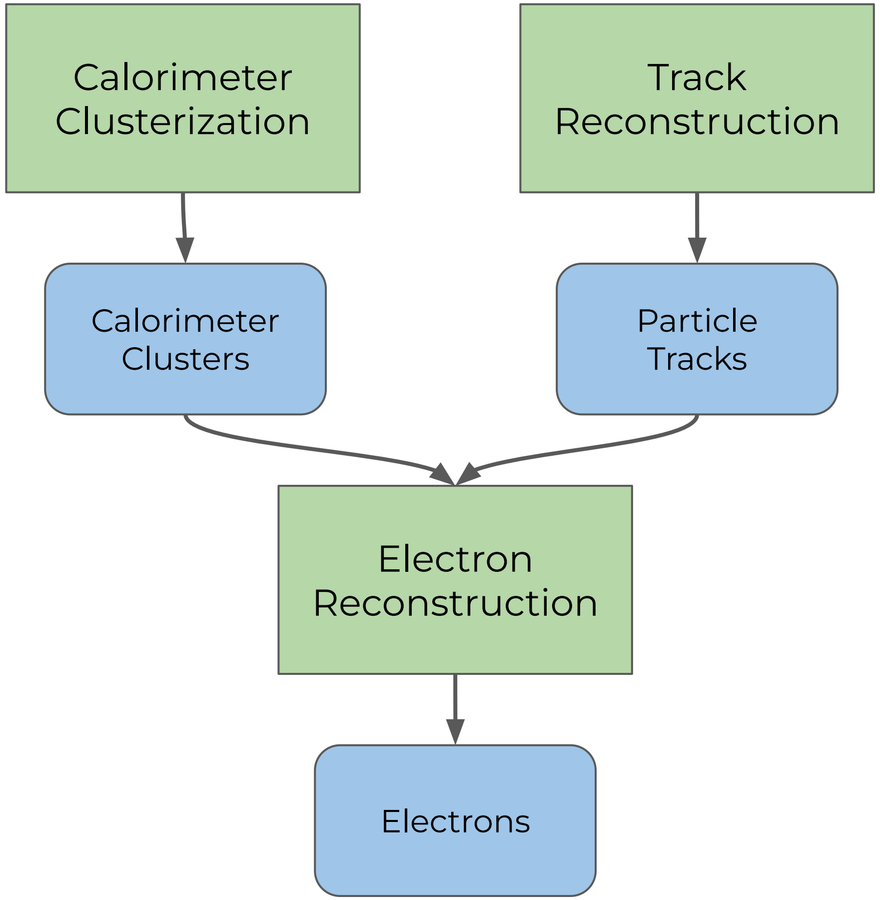
Multiple R&D Themes

- Efficient heterogeneous scheduling
- Efficient portable data structures
- Efficient accelerator interfaces to ML inference
- Common accelerated libraries
- Alternative programming languages

Efficient Heterogeneous Scheduling

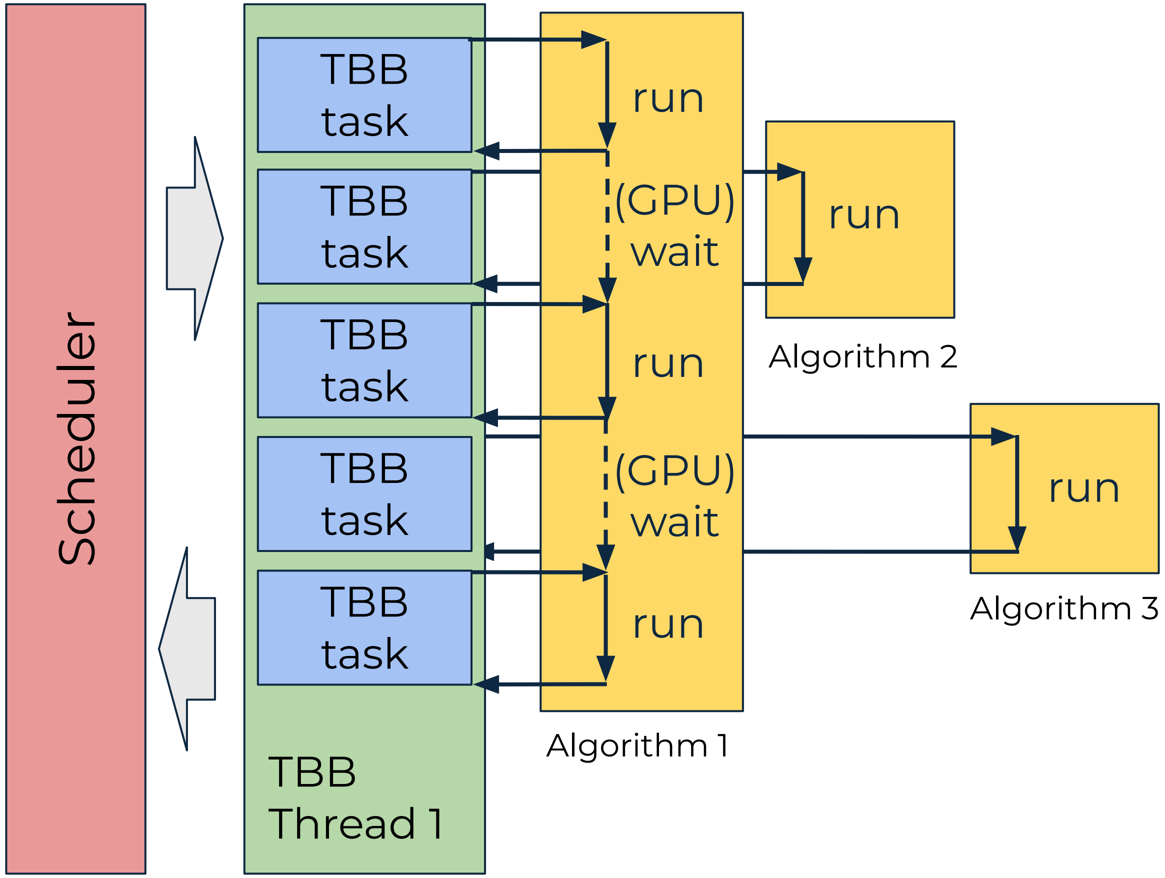
Problem Scattered approaches to efficiently schedule **asynchronous tasks** that *depend* on other tasks, across *CPUs* and *accelerators* (e.g., "multiple GPUs") with **minimal idle time**

- ATLAS, CMS, and LHCb developed similar multi-threaded task scheduling techniques in their offline/trigger software, using oneTBB.
- ALICE uses Message Passing multi-process architecture
- Started with the task scheduling method used by Gaudi (the framework of ATLAS and LHCb) to investigate a scheduling technique using **coroutines**



Using C++20 Coroutines with TBB

- Coroutines can **suspend** their execution, return control to their caller, and **resume later**
- Algorithms are executed by multiple TBB tasks:
  - When an algorithm suspends itself, the task executing it finishes
  - The scheduler adds a new task to the pool once the algorithm is ready to resume its operation (its GPU operation has finished)
- Initial, highly experimental code available: https://github.com/cern-nextgen/wp1.7-scheduler-tests



Arkadijs Slobodkins

Efficient Heterogeneous Scheduling

Planned for 2025

- Set up **representative jobs** that would reconstruct charged particle tracks with ATLAS's traccc and CMS's patatrack code for compute **performance evaluations**.
- Add a scheduler mimicking CMSSW's task scheduler for comparison.
- Investigate upcoming standard C++ parallelism (`std::execution`) for scheduling reconstruction algorithms / modules in a multi-threaded application.

Efficient Portable Data Structures

Problem Need `vector<Track>` as array-of-structs (**AoS**), `vector<pt>, vector<phi>,...` as struct-of-arrays (**SoA**), array-of-struct-of-array (**AoSoA**) etc.

Currently automatic AoS->SoA conversion done by each experiment, separately, e.g. with C++ preprocessor macros (CMS) and template metaprogramming (ACTS R&D)

Microbenchmark of benefits of SoA over AoS

AoS

```
struct S { double x1, x2, x3, … };
using AoS = std::vector<S>;
```

SoA

```
struct SoA { std::vector<double> x1, x2, x3, … }
```

Operations

MulAdd $(s2.x1 - s1.x1)^2 + (s2.x2 - s1.x2)^2$

SqrtLog $(\sqrt{s2.x1} - log(s1.x1))^2 + (log(s2.x2) - \sqrt{s1.x2})^2$

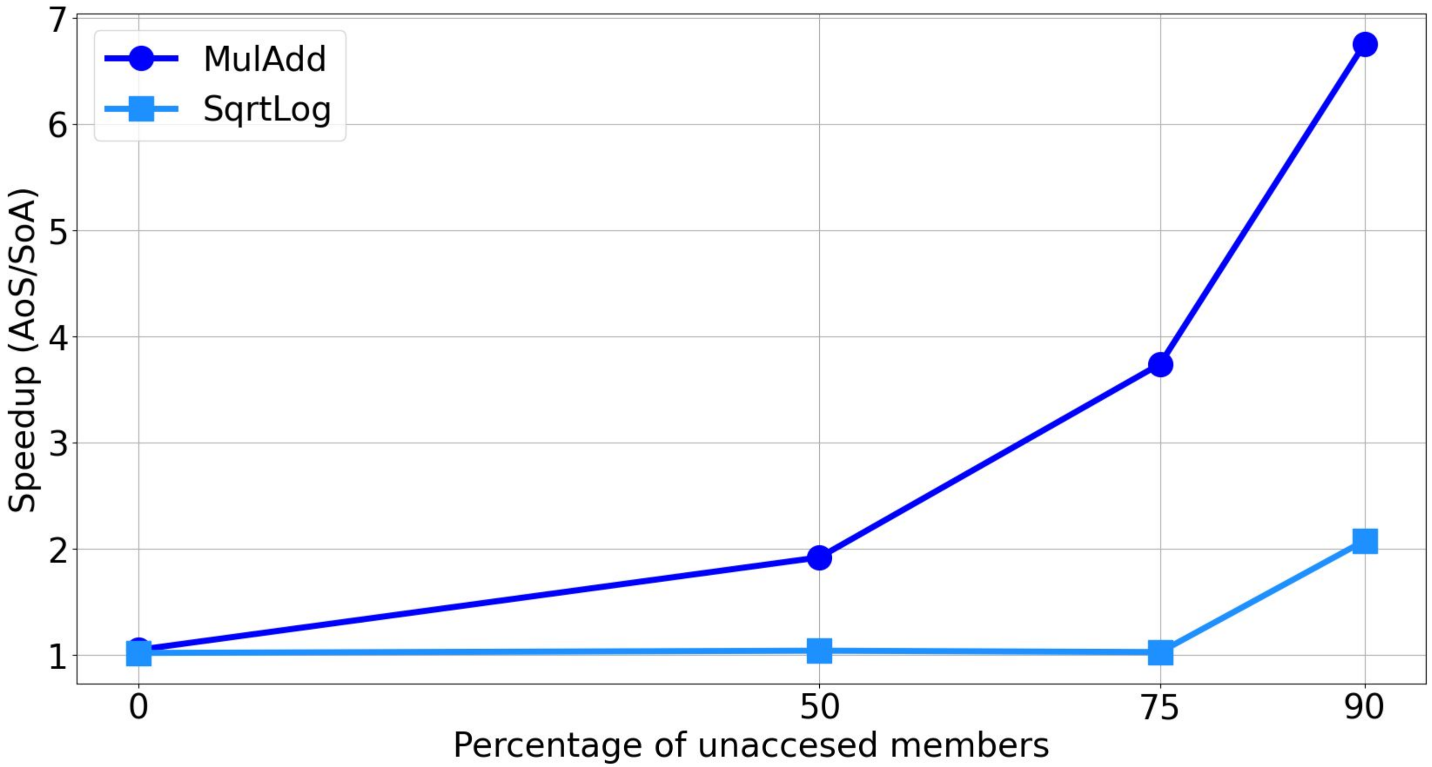
Jolly Chen

Benefits of SoA over AoS

- Intel® Core™ i9-9900K CPU @ 3.60GHz
- Calculations performed on **double-precision** floating point values stored in **std::vector** containers
- Container size: 10 million
- 10 repetitions
- **MulAdd***

  AoS: ~10-100ms
  SoA: ~10ms
- **SqrtLog***

  AoS: ~150-300ms
  SoA: ~150ms
- Code on [GitHub]



*Similar results with AVX2 auto-vectorization enabled

Efficient Portable Data Structures

We design data structures that can **adapt to different memory layouts** and hence to different hardware (CPU/GPU) and algorithms with two approaches:

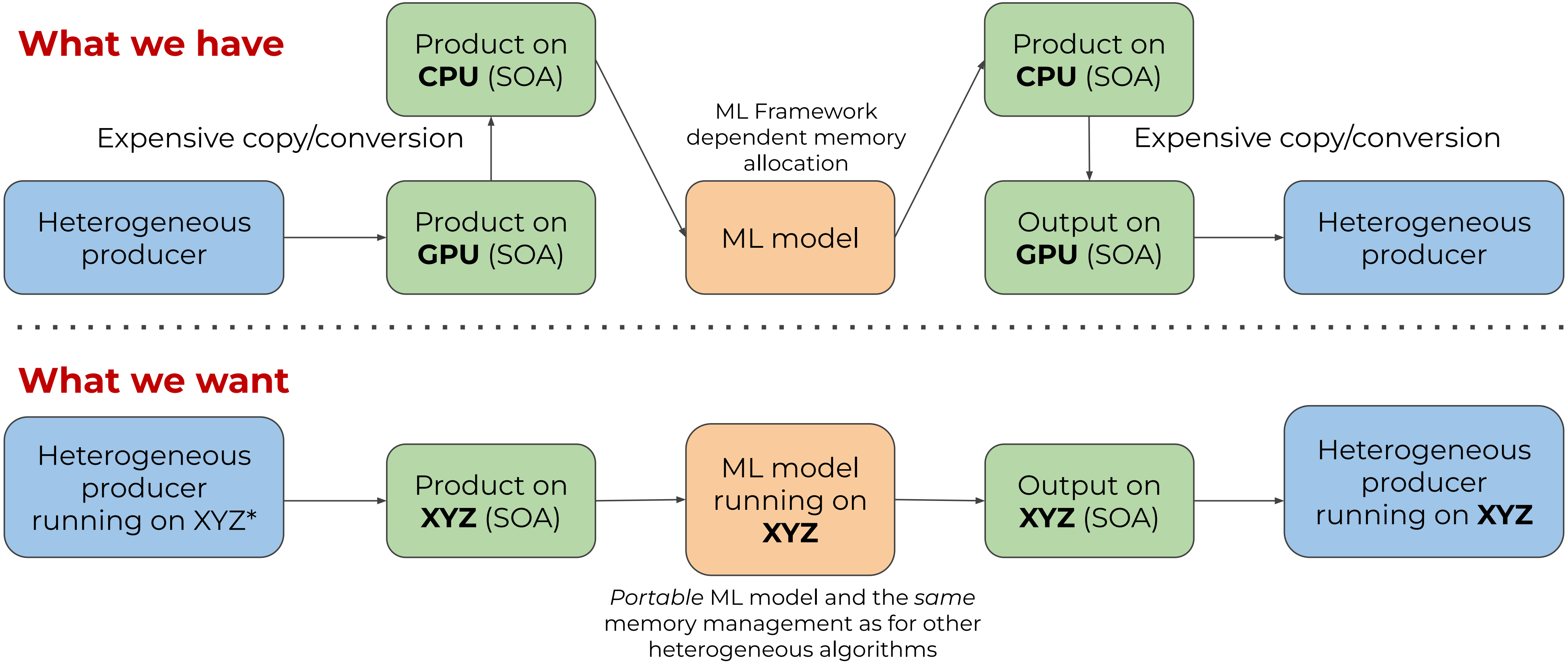
- Using C++ reflection[a], very forward-looking;
- Template-based[b] approach, using only features supported in C++23. A refinement of a current implementation used in ACTS R&D.

Planned for 2025

- Implement struct-of-array for **nested data structures** for both approaches.
- Analyze the **requirements** of the stakeholders and implementing them for our data structures.
- Benchmarking
  - **Create test cases** for both approaches, including agreeing on a prototype EDM
  - Benchmarking these data structures on the **TPC reconstruction** code of ALICE.

[a] Jolly Chen, [b] Oliver Rietmann

Efficient Accelerator Interfaces to ML Inference



* **XYZ** = Backend among CPU, AMD GPU, Intel GPU, NVIDIA GPU, future backends...

Efficient Accelerator Interfaces to ML Inference

Planned for 2025

- Prototypes with **XLA/AOT**, **PyTorch, SOFIE** direct inference to eliminate intermediate data transformations
  - Study **portability** of these solutions on different GPUs.
  - Determine relevance / criteria / requirements of memory layout for input and output data for different inference engines
- Develop a **continuous benchmarking suite** for inference of different model kinds (CNN, GNN, …) and testing the options of different inference engines.
- Investigating aspects used by Services for Optimized Network Inference on Coprocessors (SONIC)

TECH starting March 2025

Common Accelerated Libraries

Problem Most experiments have their own collection of kernels for common operations in HEP. Having a **central repository** with optimal and portable versions ensures that our software can efficiently run on various hardware architectures without significant rework.

Planned for 2025

- Do a **market survey** of existing **small matrix** (10x10..50x50) kernels. Benchmark w.r.t existing general (i.e. not optimized for small) matrix solutions for GPUs. Decide whether an in-house development is worthwhile.
- Implement **continuous performance monitoring** for an accelerated **lorentz vector library**, with CPU and CUDA implementations.
- Preliminary **study on caching strategies** of detector geometry, navigation, propagation computations for accelerator devices
- **Prototype** parallel and portable algorithms like prefix scans, sorting, clustering

LD expected to start early Feb 2025, DOCT starting Jan 2025

Alternative Programming Languages

Problem Compiled languages C++ / CUDA / SYCL /... are difficult, Python is slow; what else is suitable?

Porting the CMS Patatrack pixel reconstruction to Julia

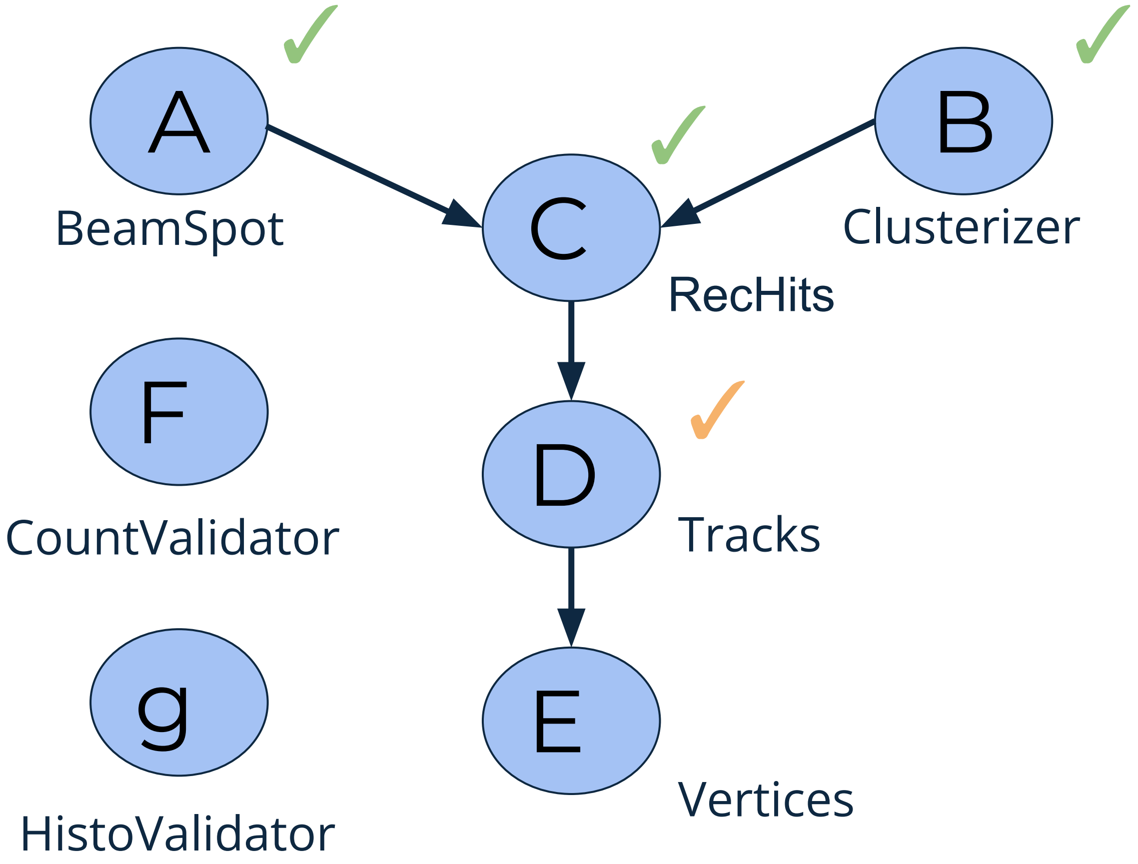
- *Pixel reconstruction* = The process of identifying and reconstructing particle trajectories by analyzing data from pixel detectors.
- Standalone application extracted from CMS software
- Tested over the years on multiple CPU and GPU technologies (Alpaka, std::par, OpenMP, CUDA, HIP, SYCL, Kokkos, etc.)
- Julia has shown potential in previous HEP evaluations[1,2]

[1] https://link.springer.com/article/10.1007/s41781-023-00104-x
[2] https://github.com/JuliaHEP/JetReconstruction.jl

Maya Ali, Mohamad Ayman Charaf, Mohamad Khaled Charaf

CMS Pixel Reconstruction - Preliminary Results

- No prior experience with Julia and CMSSW
- Ported and Validated **100%** of the local reconstruction
- Ported the Tracks Modules and currently validating.
- Achieved a running time of **35.1s** in Julia, comparable to **31.8s** in C++ for processing 1000 events with a single-threaded CPU.
- Results also presented at [JuliaHEP]



Alternative Programming Languages

Planned for 2025

- Define an **set of languages** to study. This list must contain at least Julia and Mojo.
- Define a set of **usability and feature benchmarks** and corresponding test cases, based on studies done for Julia. These need to cover the criteria of interoperability with C++; use of accelerators; ease of use. Other criteria should be investigated.

Conclusion

- Lots of progress on Heterogeneous scheduling, Portable Data Structures and Alternative Programming Languages!
  - Prototype of multi-threaded task scheduler with oneTBB and coroutines
  - Two implementations of data structures with adaptive memory layouts
  - CMS local pixel reconstruction fully ported to Julia with comparable performance to C++
- More prototyping and benchmarking planned for 2025
- Common Accelerated Libraries and ML Inference subprojects starting up in 2025

Resources & Contact

Github for R&D Code at https://github.com/cern-nextgen

Coordination at ngt-1-7-coord@cern.ch

Discussions at https://mattermost.web.cern.ch/nextgen-triggers/channels/task17

Indico category https://indico.cern.ch/category/17798/

Task Leaders

Axel.Naumann@cern.ch Andrea.Bocci@cern.ch Attila.Krasznahorkay@cern.ch

The Team

Arkadijs Slobodkins

Jolly Chen

Oliver Rietmann

Maya Ali

Mohamad Ayman Charaf

Mohamad Khaled Charaf

CERN

NexTGen

Next Generation Triggers

Backup - Layout Benchmark Runtimes