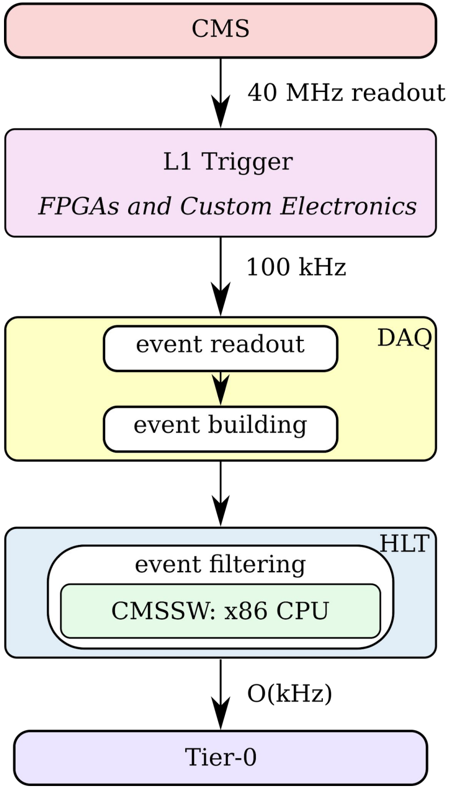
The CMS heterogeneous reconstruction

Andrea Bocci, **Felice Pantaleo** (CERN)

For the CMS Collaboration

CMS Trigger and reconstruction


CMS

40 MHz readout

L1 Trigger

FPGAs and Custom Electronics

100 kHz

DAQ

event readout

event building

HLT

event filtering

CMSSW: x86 CPU

O(kHz)

Tier-0

Since Run-3 (2022+), CMS operates an heterogeneous HLT farm:

- CMS software, running on 200 nodes:
  - ~26k CPU cores AMD Milan 7633
  - 400 NVIDIA T4 GPUs
- Event size ~MB

CMS Phase-2 upgrade:

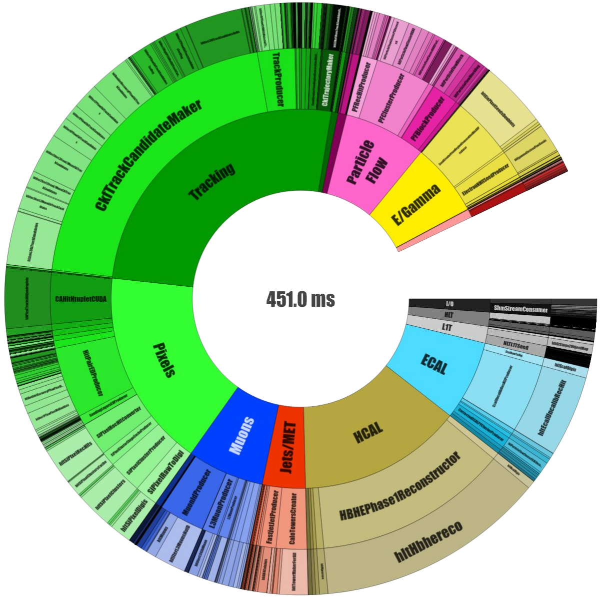
- L1T 100kHz -> 750kHz (7.5x)
- Pile-up 60 -> 200 (~3x)
- More complex detectors

Almost an order of magnitude higher performance required: CPU evolution is not able to cope with the increasing demand of performance

Use of GPUs

- Potential advantages of heterogeneous software:
  - faster computational performance
  - more complex algorithms
  - run globally more often
  - better physics performance
- Heterogeneous computing allows experiments to increase their physics potential
  - Economically sustainable online/offline computing systems
  - Benefit from future industry developments
  - Fully exploit HPC centres
- Heterogeneous software development result in valuable training for young scientists approaching complex event reconstruction

Where it all started



*Online reconstruction timing measured on pileup 50 events from Run2018D on a full **Run-2** HLT node (2x Intel Skylake Gold 6130) with HT enabled, running 16 jobs in parallel, with 4 threads each.*

While looking for opportunities for GPU offloading:

- No single hotspot
- Many instances of the same algorithm with different configurations (regional, global)
- Chains of algorithms containing enough parallelism
- Early in the reconstruction, no dependencies and then move downstream
- Enough expertise in the particular reconstruction algorithm and parallelism

Identified three targets:

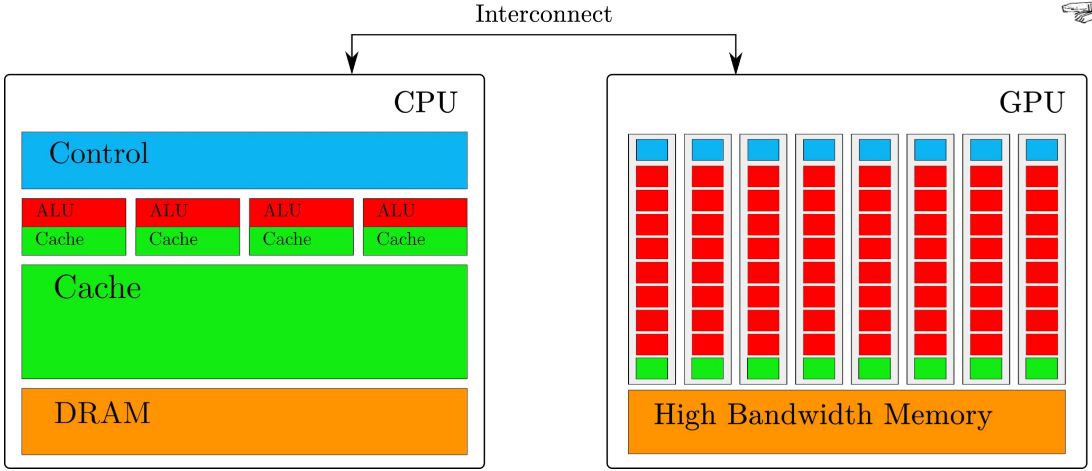
- Global pixel reconstruction chain, from RAW data down to pixel-only tracks and vertices
- ECAL local reconstruction
- HCAL local reconstruction

Make CPU and GPU codes to run together efficiently in the CMSSW framework

A problem with many faces

- How do I make the code run faster?
- Should I change the data structures and memory management?
- What if the algorithm is not exposing enough parallelism?
- What if there is no single algorithm or detector that is a hotspot taking most of the reconstruction time?
- How do I integrate the code into the framework?
- How do I make the code portable?
- How do I make sure that the CPU threads are not idle waiting for the GPU?
- How do I balance the number and type of GPUs with the fraction of GPU code in the software?
- How do I match the GPU requirements of the job with the machines available on the grid?
- How do I engage the community to develop heterogeneous code?

Heterogeneous architectures



- Avg bandwidth between CPU and host memory
- Low core count/Powerful ALU
- Complex control unit
- Large caches

Latency

- High bandwidth between GPU cores and GPU memory
- High core count
- No complex control unit
- Small caches

Throughput

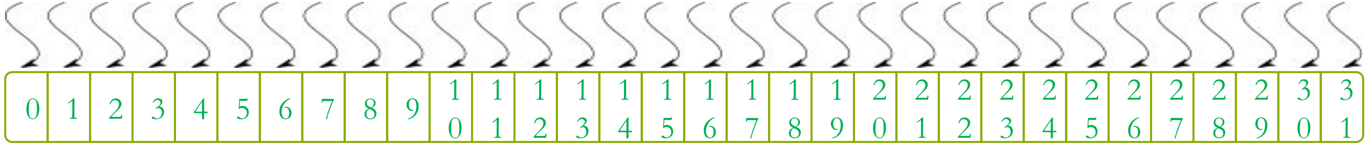
GPU Programming model

GPU code is executed by many “threads” in parallel

- Parallel functions, aka “kernels” spawn threads organized in a “grid” of blocks

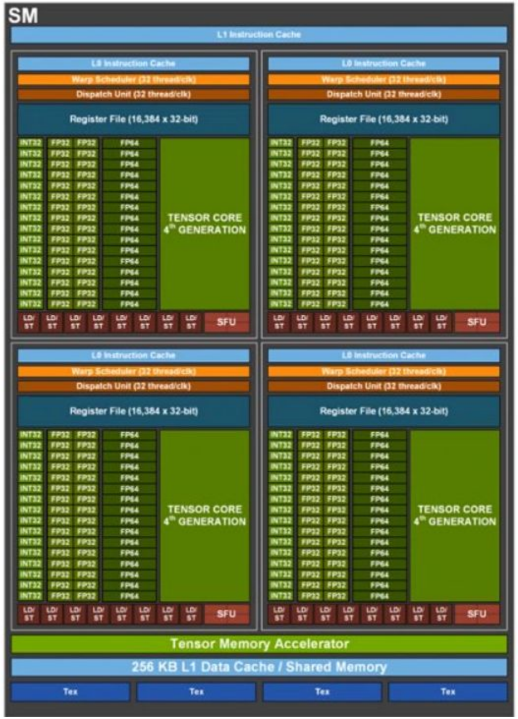
Threads in the same block can:

- Communicate via fast on-chip shared memory
- Synchronize



Threads in the same block work in steplock in groups (aka warps) of 16, 32, 64 threads depending on the GPU brand.

- Preferred access to main device memory is coalesced
- Lose an order of magnitude in performance if cached access pattern used on GPU


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
TENSOR CORE 4th GENERATION
LD/ST
SFU
Tensor Memory Accelerator
256 KB L1 Data Cache / Shared Memory
Tex

One of the 132 Streaming Multiprocessors of a NVIDIA H100

Generic SoA, Portable collections

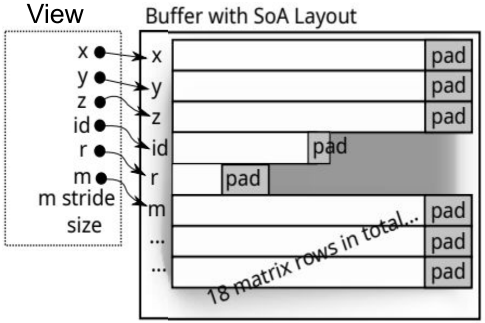
SoA use widespread in CMSSW with many ad-hoc implementations.

A Generic SoA [1] has been developed to automate definition and implementation of runtime sized SoAs:

- **Layout** divides a buffer into runtime sized columns
- **View** is the interface to the data. Lightweight, this is the structure passed to kernels
    - Just pointers to columns, size
- **Buffers** host memory, pinned host memory or device memory, allocated from the framework (CUDA or Alpaka)

PortableCollection wraps the Layout and View

- Manages the buffer allocation
- Provides an interface for memory transfers
- Manages serialization to ROOT files



---

[1] *Implementation of generic SoA data structures in the CMS software*, E. Cano, A. Bocci

Asynchronous heterogeneous computing

CMSSW Core framework plays a huge role in the scalability of the performance

A GPU kernel execution can be asynchronous wrt to:

- Data transfers between host and device memory
- CPU code execution
- Execution of other kernels

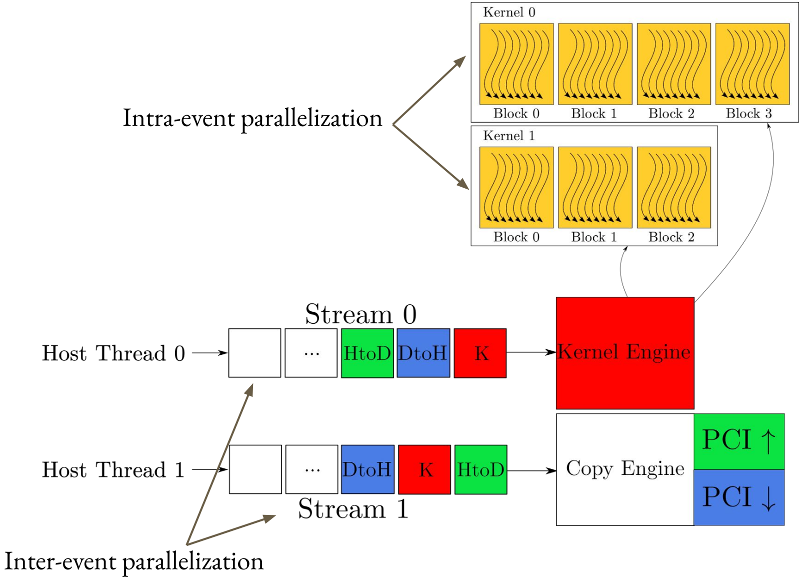
Plenty of opportunities for concurrency.

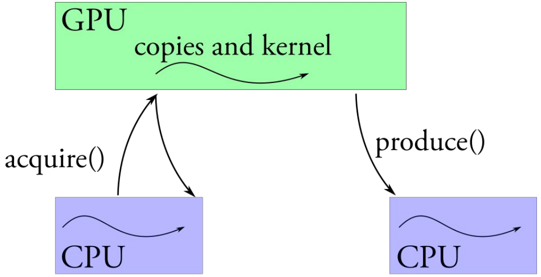
A single module processes one event at a time

Asynchronous execution (`acquire()`):

- CPU thread can execute another task, if all its data dependencies are satisfied
- When asynchronous instructions in `acquire()` have completed, callback to the framework that can execute the `produce()`

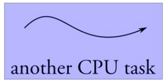
Porting consecutive chains of modules to avoid synchronizations and data transfers between host and device





[1] [*Bringing heterogeneity to the CMS software framework*](), A. Bocci, D. Dagenhart, V. Innocente, C. Jones, M. Kortelainen, F. Pantaleo, M. Rovere

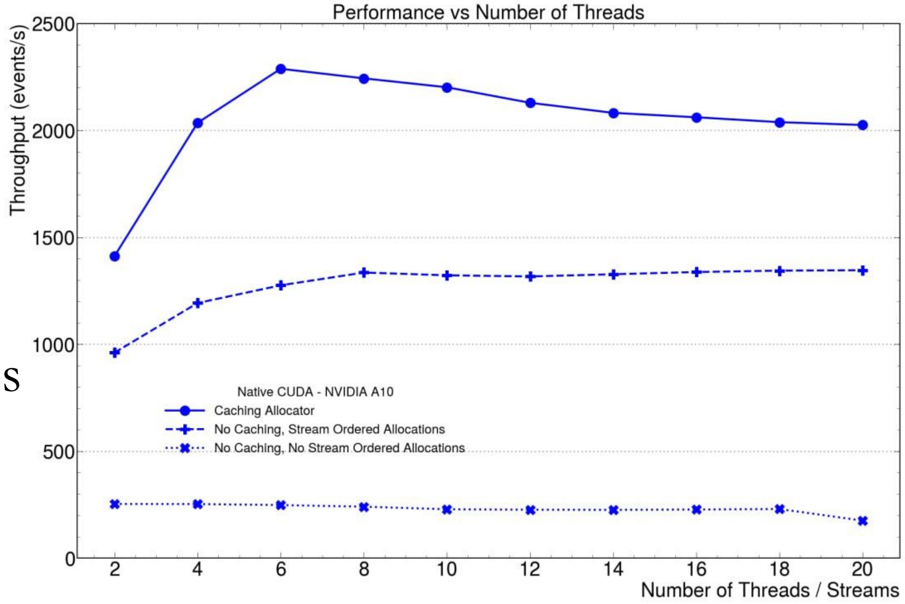
[2] See talk by M. Kortelainen, [Performance of Heterogeneous Algorithm Scheduling in CMSSW]()



Caching allocator

In absence of single hotspots, with many algorithms running with many events in flight, allocations on the GPU can become a bottleneck

- Asynchronous stream-ordered allocations have been introduced recently in CUDA
- Custom Caching allocator was developed [1]
  - reusing memory can bring big leaps in performance
  - Supports alpaka, CUDA, SYCL



[1] *Performance portability for the CMS Reconstruction with Alpaka*, A. Bocci, A. Czirkos, A. Di Pilato, F. Pantaleo, G. Hugo, M. Kortelainen , W. Redjeb

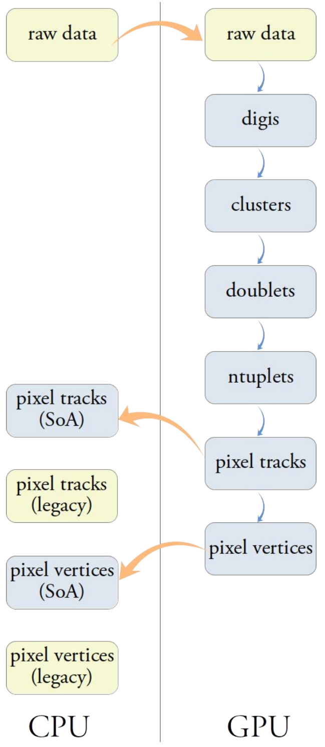
The Patatrack Pixel Reconstruction

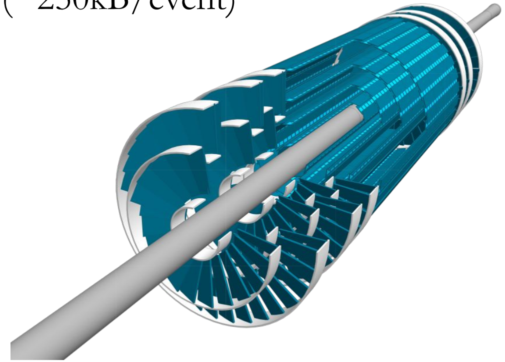
The Patatrack Pixel Tracks and Vertices reconstruction [1] from RAW data is one of the main contribution from the project with new algorithms developed and implemented

- Raw data decoding
- Clustering of nearby active pixels (Hits)
- Linking of doublets of hits on different layers
- Pattern recognition linking doublets segments (Cellular Automaton)
- Fitting of the found n-tuplets (Pixel Tracks)
- Clustering of Pixel Tracks (Vertices)

Pixel RAW data for each event is transferred to the GPU initially (~250kB/event)
Integer results are identical between CPU and GPU

Small differences in the results of floating point
can be explained by differences in re-association





[1] *Heterogeneous Reconstruction of Tracks and Primary Vertices With the CMS Pixel Tracker*, A. Bocci, V. Innocente, M. Kortelainen, F. Pantaleo, M. Rovere

Calorimeters local reconstruction and PF Clustering

Electromagnetic Calorimeter (ECAL):
  61,200 crystals in Barrel (EB)
  14,648 crystals in Endcap (EE) of PbWO4
Hadron Calorimeter (HCAL):
  9,072 channels in Barrel (HB)
  6,768 channels in Endcap (HE)

The signals from crystals/scintillator tiles are readout by APD/VPT (ECAL), siPM (HCAL)

Charge sampling over 25ns

Out-of-time pile-up reconstructed by fitting the in-time pulse together with multiple other pulse templates shifted by k*25ns at the same time [1][2]

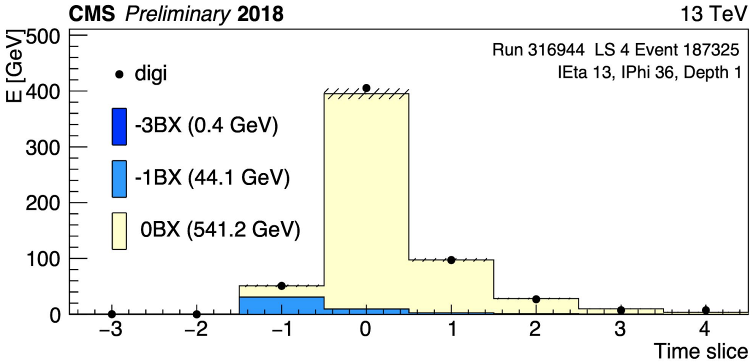
Iterative algorithm based on Cholesky decomposition

Implemented using Eigen library

- independent fits sequential on different CUDA threads

Reconstructed signals form Hits

- Ongoing ParticleFlow clustering development with alpaka [3]



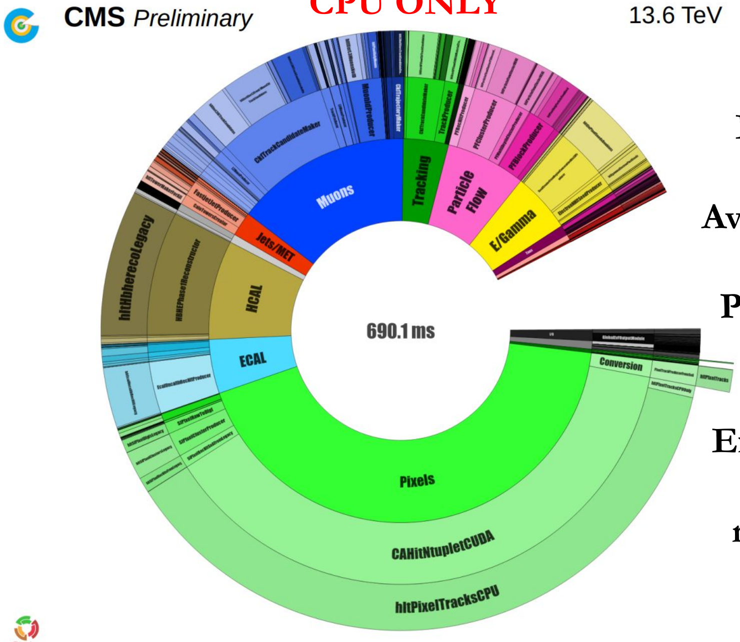
[1]: Nonnegativity constraints in numerical analysis | The Birth of Numerical Analysis

[2]: Reconstruction of signal amplitudes in the CMS electromagnetic calorimeter in the presence of overlapping proton-proton interactions

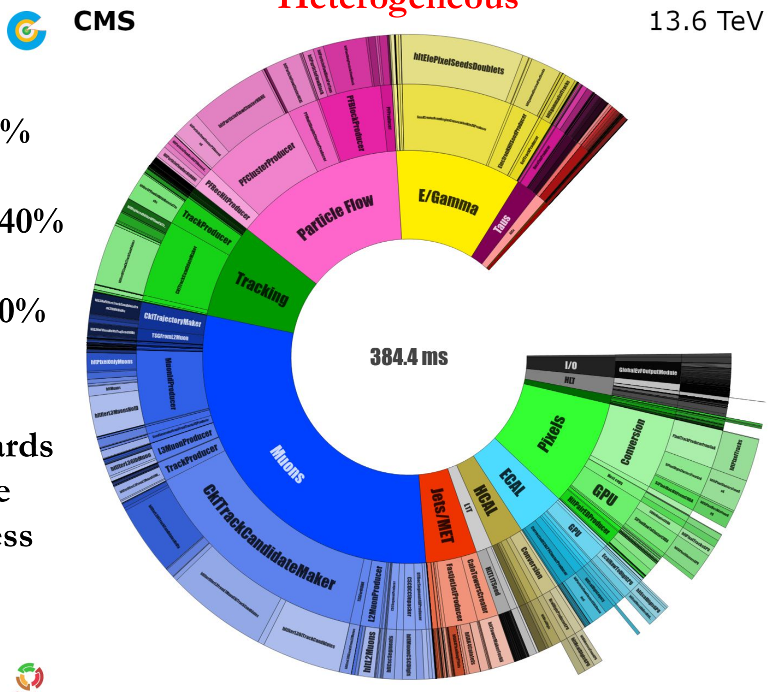
[3]: Particle Flow Reconstruction on Heterogeneous Architecture for CMS

Putting every ingredient together, the status today at the HLT

CPU ONLY



Heterogeneous



Event throughput +80%

Average time per event -40%

Power consumption: -30%

Experience gained towards heterogeneous offline reconstruction: Priceless

Online reconstruction time measured under realistic conditions, on 64000 proton-proton events with an average pileup of 56 collisions, collected on October 7th 2022 (run 359998, luminosity sections 242-243), on a full ***Run-3*** *HLT node (2x AMD Milan 7633 + 2 NVIDIA T4)*

See talk by Ganesh Parida next Tuesday: *Run-3 Commissioning of CMS Online HLT reconstruction using GPUs*

Deployment on the grid

The validation of the CMSSW release and the heterogeneous HLT were carried out on GridKa TOpAS cluster (24x A100, 24x V100S, 8x V100, expandable to KIT HPC cluster) [1] and Wisconsin (35xT4) [2]

Double matchmaking process:

1. HTCondor for hardware matching (e.g. compute capability, device memory)
2. CMS Pilot job for matching of the software requirements (e.g. min CUDA version)

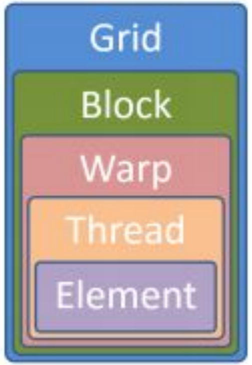
```
...
'RequestType': 'TaskChain',
'SubRequestType': 'RelVal',
'Task1': {
          ...
        'RequiresGPU': None,
        'TaskName': 'ZMM_14TeV_TuneCP5_2021_GenSim'},
'Task2': {
          ...
        'GPUParams': None,
        'RequiresGPU': None,
        'TaskName': 'Digi_2021'},
'Task3': {
          ...
        'GPUParams': {'CUDACapabilities': ['7.5'],
                      'CUDADriverVersion': '',
                      'CUDARuntime': '11.2',
                      'CUDARuntimeVersion': '',
                      'GPUMemory': '8000',
                      'GPUName': ''},
        'RequiresGPU': 'required',
        'TaskName': 'Reco_Patatrack_PixelOnlyGPU_2021'},
'TaskChain': 3,
'TimePerEvent': 10}
```

[1] M. Giffels, A. Gottmann, M. Schnepf, T. Voigtländer

[2] See talk by Charis-Kleio Koraka next Tuesday: Running GPU-enabled CMSSW workflows through the production system

See talk by Antonio Perez-Calero Yzquierdo later today: Brokering to heterogeneous resources in the CMS Computing Grid

Performance portability



Maintaining and testing two codebases is not the most sustainable solution.

- Plethora of CPUs, GPUs, FPGAs

Portability could be achieved by blindly translating CUDA threads to, e.g., CPU threads or viceversa (plus some synchronization mechanism)

- You would not need to learn how a GPU works

Unfortunately, this is a terrible idea and will almost certainly lead you to poor performance

Portability does not imply Performance Portability.

Alpaka in online reconstruction

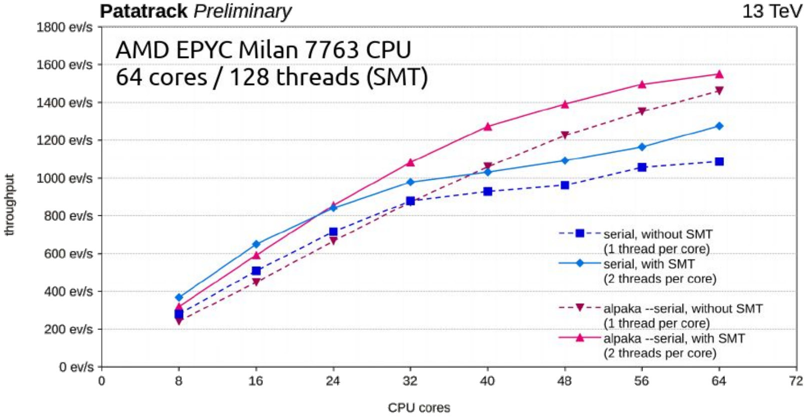
During 2023, the heterogeneous online reconstruction is being ported to alpaka [1]

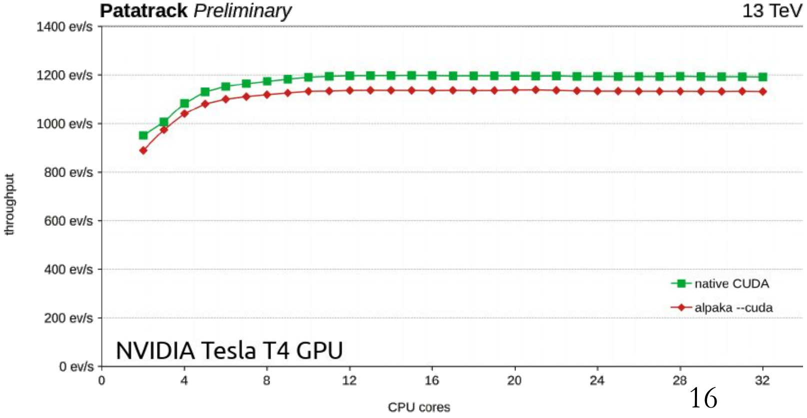
It achieves the promise of performance portability:

- Close to native performance when executed sequential
- Can run on different GPU brands with close to native performance

Unfortunately, it does not save you from learning how a GPU works.

Other performance portability frameworks like Kokkos, SYCL, std::par, and OpenMP are also under study [2]





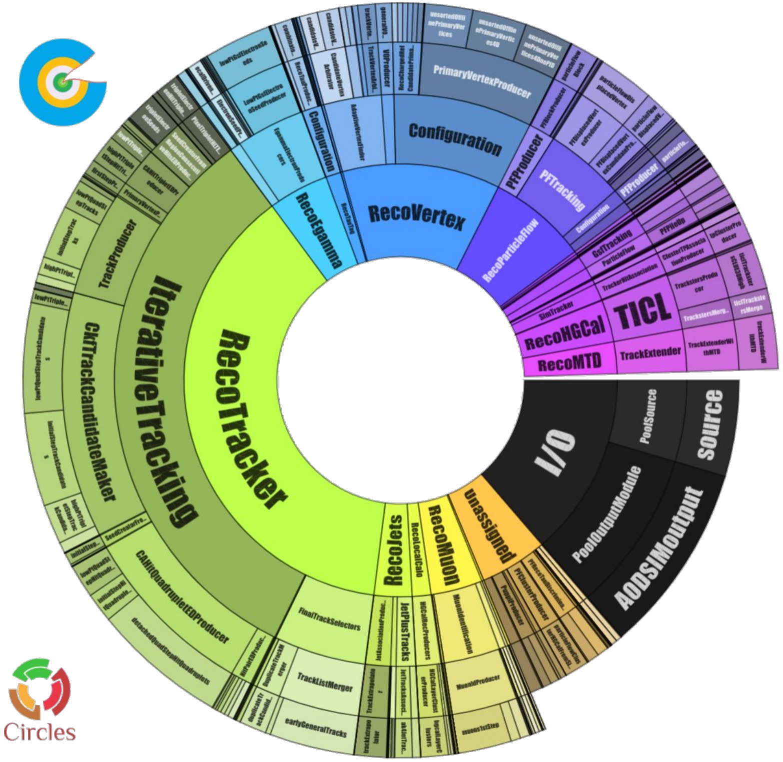
[1] See talk next Tuesday by A. Bocci: Adoption of the alpaka performance portability library in the CMS software

[2] See talk next Thursday by M. Kortelainen: Evaluating Performance Portability with the CMS Heterogeneous Pixel Reconstruction code

Offline reconstruction

Overlap between heterogeneous algorithms running online with their offline counterparts

- Pixel Tracks and vertices are not directly used offline, but they provide seeds for the Kalman Filter
    - Phase-2 pixel reconstruction will be replaced by Patatrack pixel reconstruction in 2023
- For Phase-2 HGCAL replaces the ECAL-HCAL endcap calorimeters
    - HGCAL Reconstruction by the TICL framework will be completely ported to Alpaka in 2023 [1][2]
- Developments ongoing for heterogeneous tracking in the Outer Tracker [3]
- Overall aiming towards a 10% offload of Phase-2 offline reconstruction, and common modules for Run 3 by the end of 2023



CPU time for offline Phase-2 reconstruction in ttbar events with an average of 200 overlapping proton-proton collisions as of CMSSW_13_1_0_pre3

[1] A. Di Pilato et al. Performance study of the CLUE algorithm with the alpaka library

[2] CMS collaboration, The TICL reconstruction at the CMS Phase-2 High Granularity Calorimeter Endcap

[3] See talk next Tuesday by Philip Chang, Line Segment Tracking in the High-luminosity LHC

Engaging the community and partners

The Patatrack R&D continuously engages the community helping from the proofs of concept to the integration phase

- Continuous support throughout the development
  - Thanks to experts and the support of partners like CERN openlab, Flatiron Institute, Intel, NVIDIA, E4
- Boosts of productivity during Patatrack Hackathons
  - 12 Hackathons organized so far, 2 more to come in 2023 with 30 participants
  - Foster a familiar environment: about 170kg of pasta cooked in total…

Conclusion

- The CMS experiment has established a strong R&D&I line in heterogeneous computing for the online and offline reconstruction
  - Many other R&Ds ongoing
  - Vibrant ML community power-using GPUs
- More than 40% of the CMS online reconstruction is heterogeneous
  - It will be entirely ported to alpaka during 2023
  - Targeting 80% for Phase-2
- 10% of the CMS offline reconstruction will be heterogeneous by 2023
  - Computing infrastructure is running heterogeneous production jobs