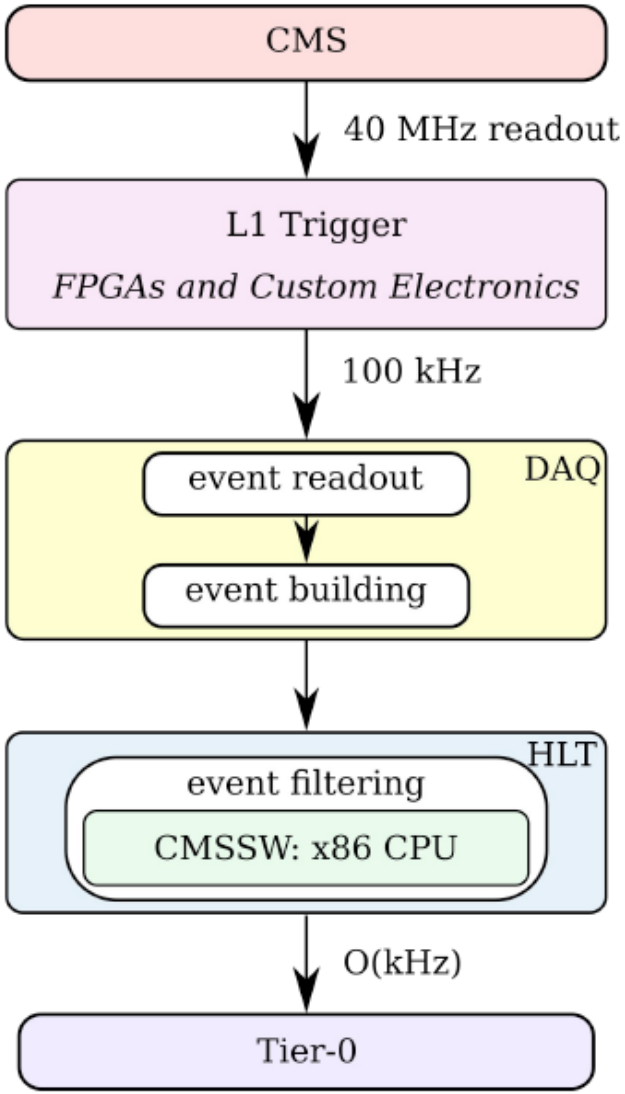
Calcolo su GPU nell'esperimento CMS, soluzioni attuali e prospettive future

Adriano Di Florio
CMS Experiment
[1]Politecnico &INFN Bari

Workshop CCR 2022 – 23 – 27 Maggio 2022

Where we did we stand in Run2?



CMS and LHC scenario at the end of Run-2

- peak average instantaneous luminosity of $2\times10^{34}\,cm^{-2}s^{-1}$
- about 50 proton-proton collisions per bunch crossing
- 100 kHz input rate (from the Level 1 Trigger rate)

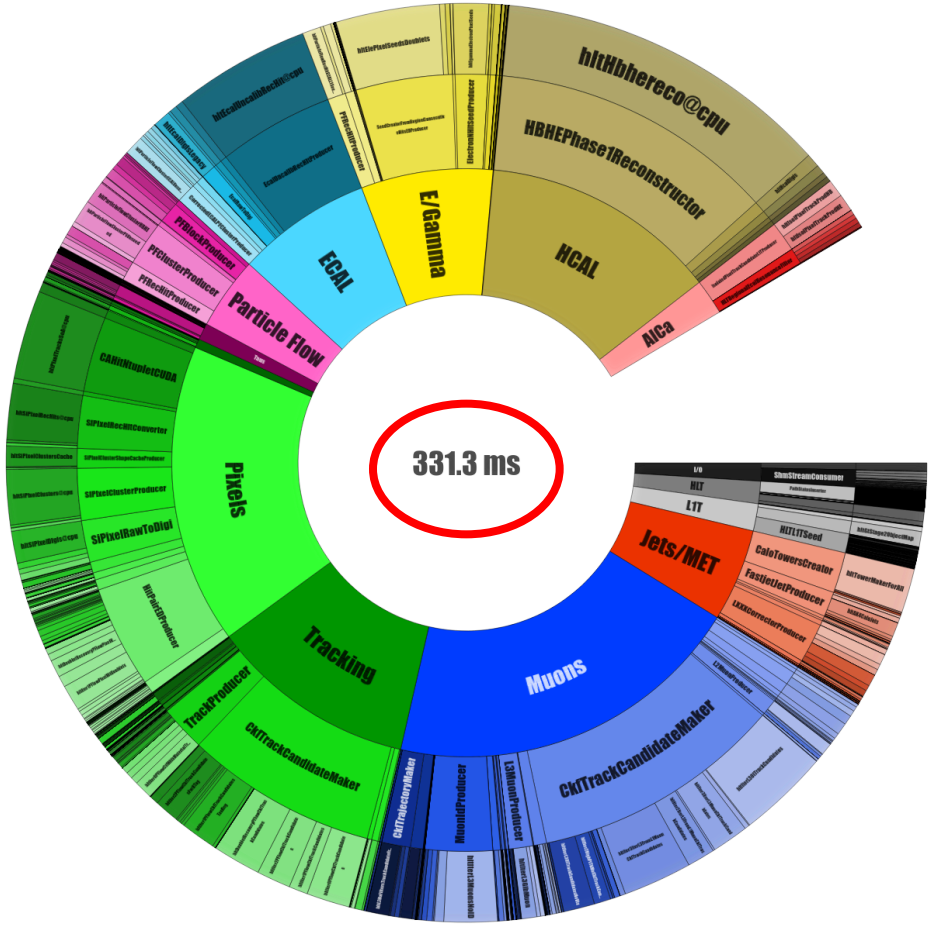
A traditional CPU farm

Over 1000 machines for 716 kHS06

30k physical CPU cores / 60k logical cores

- HLT running with multithreading
- 15k jobs with 4 threads

Where we did we stand in Run2?



CMS and LHC scenario at the end of Run-2

- peak average instantaneous luminosity of $2 \times 10^{34}\ cm^{-2}s^{-1}$
- about 50 proton-proton collisions per bunch crossing
- 100 kHz input rate (from the Level 1 Trigger rate)

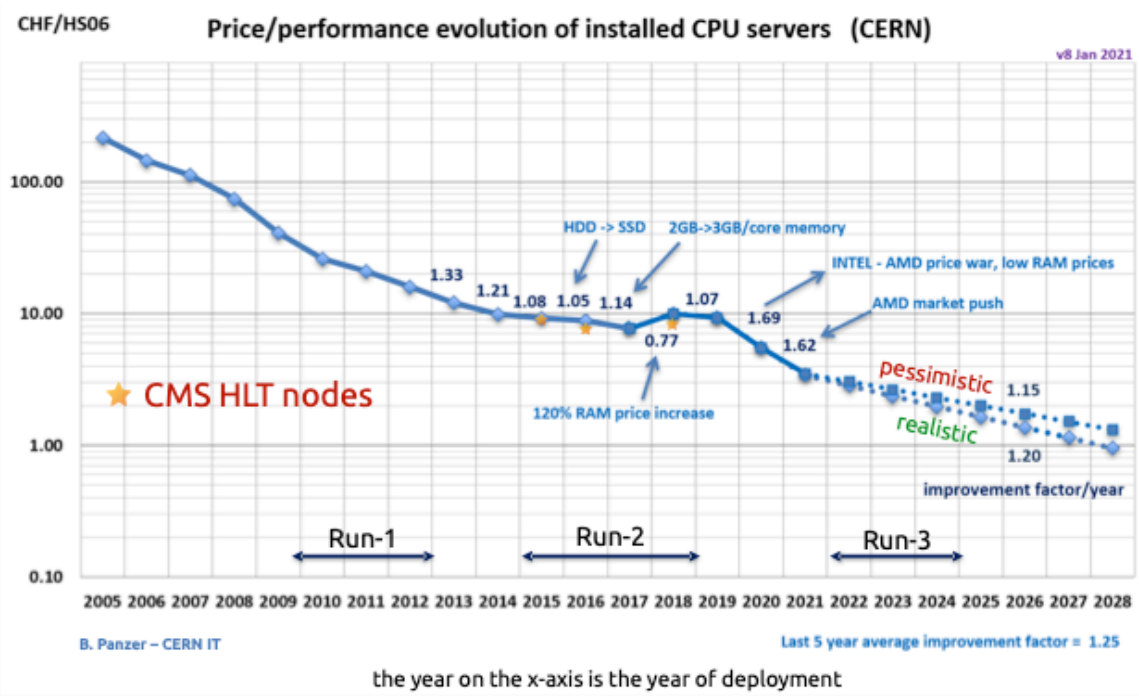
A traditional CPU farm

Over 1000 machines for 716 kHS06

30k physical CPU cores / 60k logical cores

- HLT running with multithreading
- 15k jobs with 4 threads

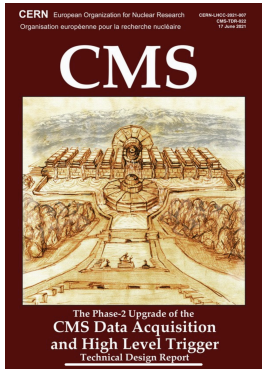
Let's crunch some numbers



Extrapolation by CERN IT of server price/performance
- based on the servers installed in 2013-2021
- servers for CMS HLT follow the same trend
- pessimistic: +15% /y
- realistic: +20% /y

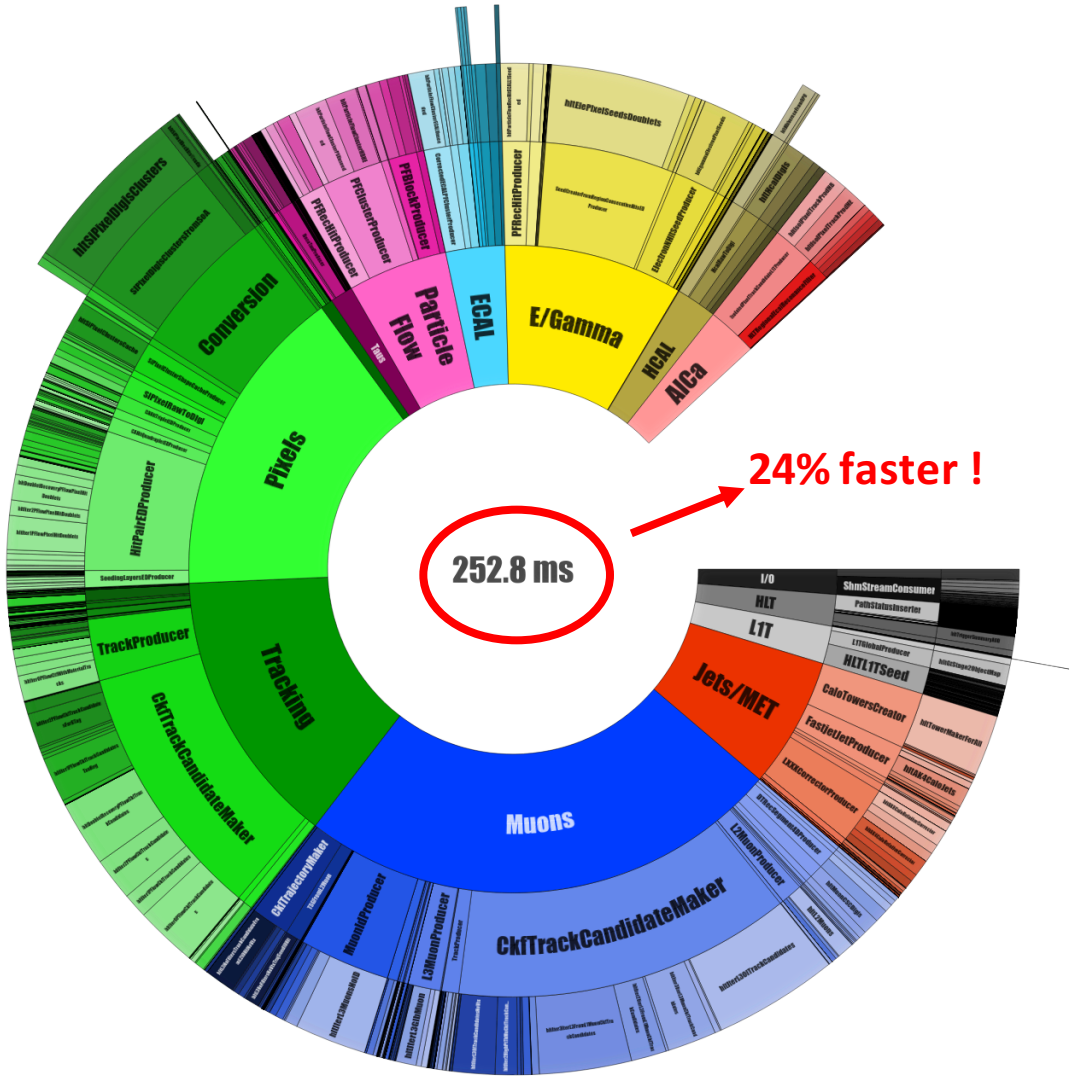
Assume the same trend for GPU price/performance
- same technology and fabrication process
- compete for the same market
- observed: +30% /y
- for A10 (2021) vs T4 (2018)

- So could accelerators help? From The Phase-2 Upgrade of the CMS Data Acquisition and High Level Trigger TDR:



- CPU-only
  - 1.55 CHF/HS06 in 2028
- 50% code ported
  - 0.70 CHF/HS06 in 2028
- 80% code ported
  - 0.22 CHF/HS06 in 2031

	Run-2	Run-3	Run-4	Run-5
peak luminosity	2×10^{34} cm^{-2}s^{-1}	2×10^{34} cm^{-2}s^{-1}	5×10^{34} cm^{-2}s^{-1}	7.5×10^{34} cm^{-2}s^{-1}
pileup	50	50	140	200
HLT input rate	100 kHz	100 kHz	500 kHz	750 kHz
HLT output rate	1 kHz	< 2 kHz	5 kHz	7.5 kHz
HLT farm size	0.7 MHS06	0.8 MHS06	16 MHS06	37 MHS06



The effort in CMS to use GPUs for reconstruction started 5-6 years ago:

2016: first attempts (EuroHack 2016)
2018 – 2019: Patatrack "demonstrator" for using GPUs at HLT
2020 – 2021: integration in the experiment's software

Intially targetted Phase2 **but** things evolve rapidly and Run3 is an ideal benchmark:

- no external pressure from LHC conditions
- gain experience
- take advantage of the extra computing capacity (e.g. scouting)

CMS HLT will offload four main components to GPUs:

- pixel tracker local reconstruction
- pixel-only track and vertex reconstruction
- electromagnetic and hadronic calorimeter local reconstruction

Chains of modules/algorithms have been ported to run on GPUs

- A single module processes one event at a time

Asynchronous execution (`acquire()`)

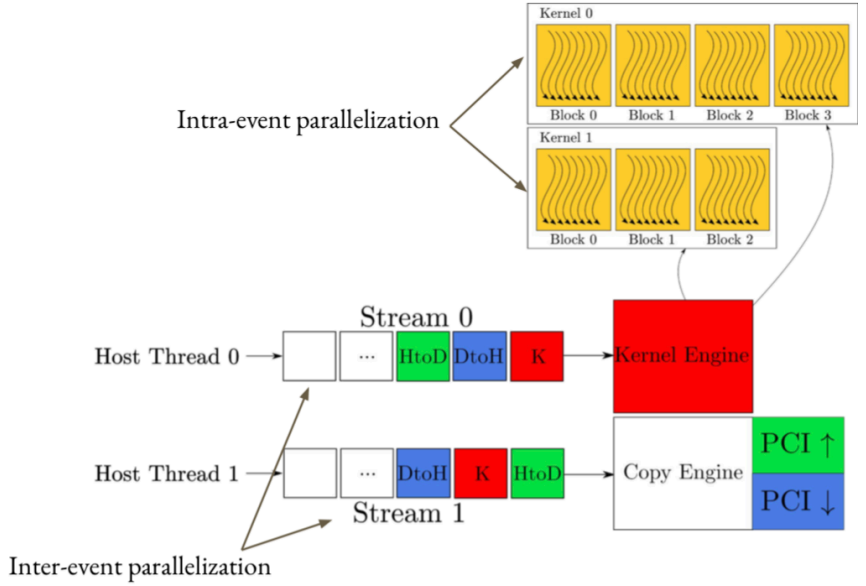
- CPU thread can execute another task, if all its data dependencies are satisfied
- When asynchronous instructions in `acquire()` have completed, callback to the framework that can execute the `produce()`

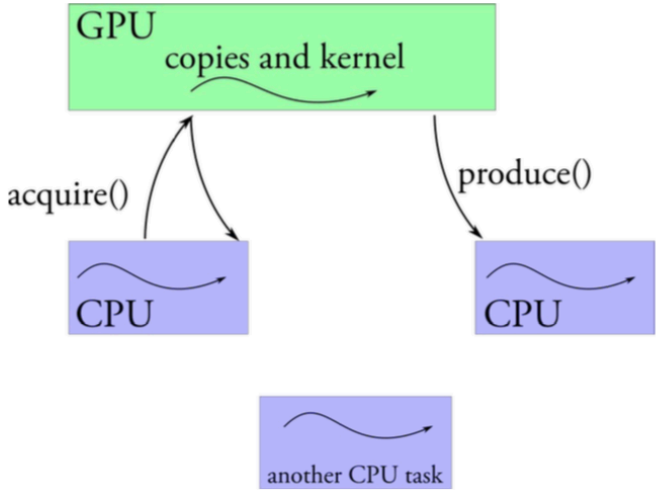
On demand copy back of intermediate results

On demand conversion to legacy data format (**n.b.** data formats rewritten as SoA):

- GPU memory bandwidth best exploited with coalesced memory access

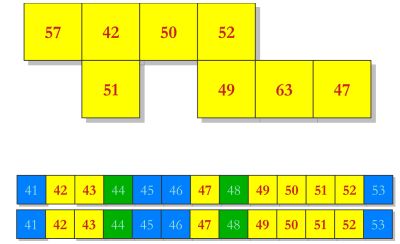
Possibility to run the same workflow on GPUs and CPUs producing the same result (**fundamental for validation**)





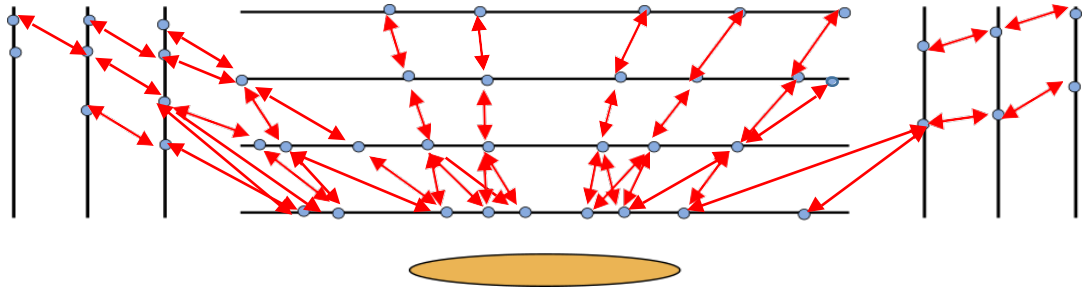
Parallelisation: where and how? An example

raw data unpacking and decoding: **parallelised** across all input pixel hits
clustering of the pixel hits: parallelised across the pixel detectors and across the input pixel hits
conversion to global coordinates parallelised across each cluster



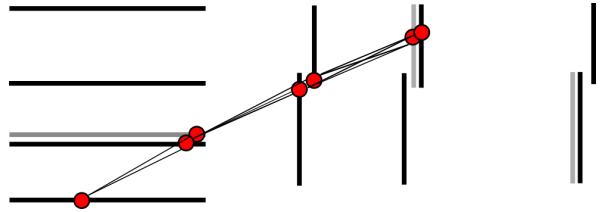
building doublets: **parallelised** on the hits of each layer
building ntuplets:

- 2D **parallelisation** on the inner and outer layers
- Cellular Automaton algorithm with depth-first search



ntuplets cleaning

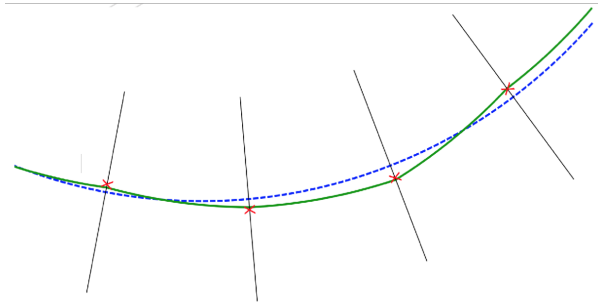
- Fishbone algorithm merges overlapping ntuplets
- 2D **parallelisation** over ntuplets and possible duplicates



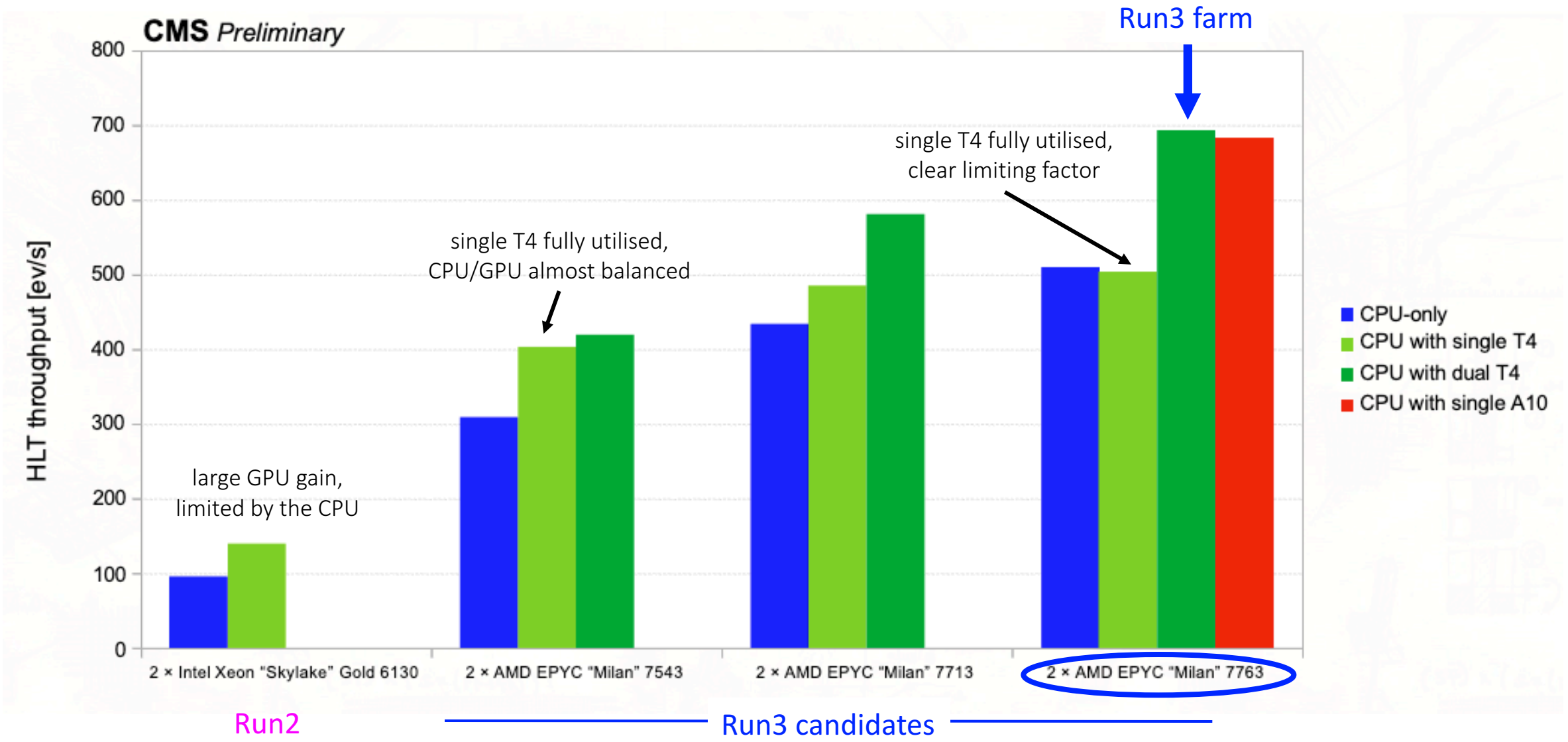
track fitting: implemented using Eigen, **parallelised** over the ntuplets

vertex reconstruction

- along z cluster tracks: **parallelised** across all input tracks
- split low quality vertices: **parallelised** across the vertices



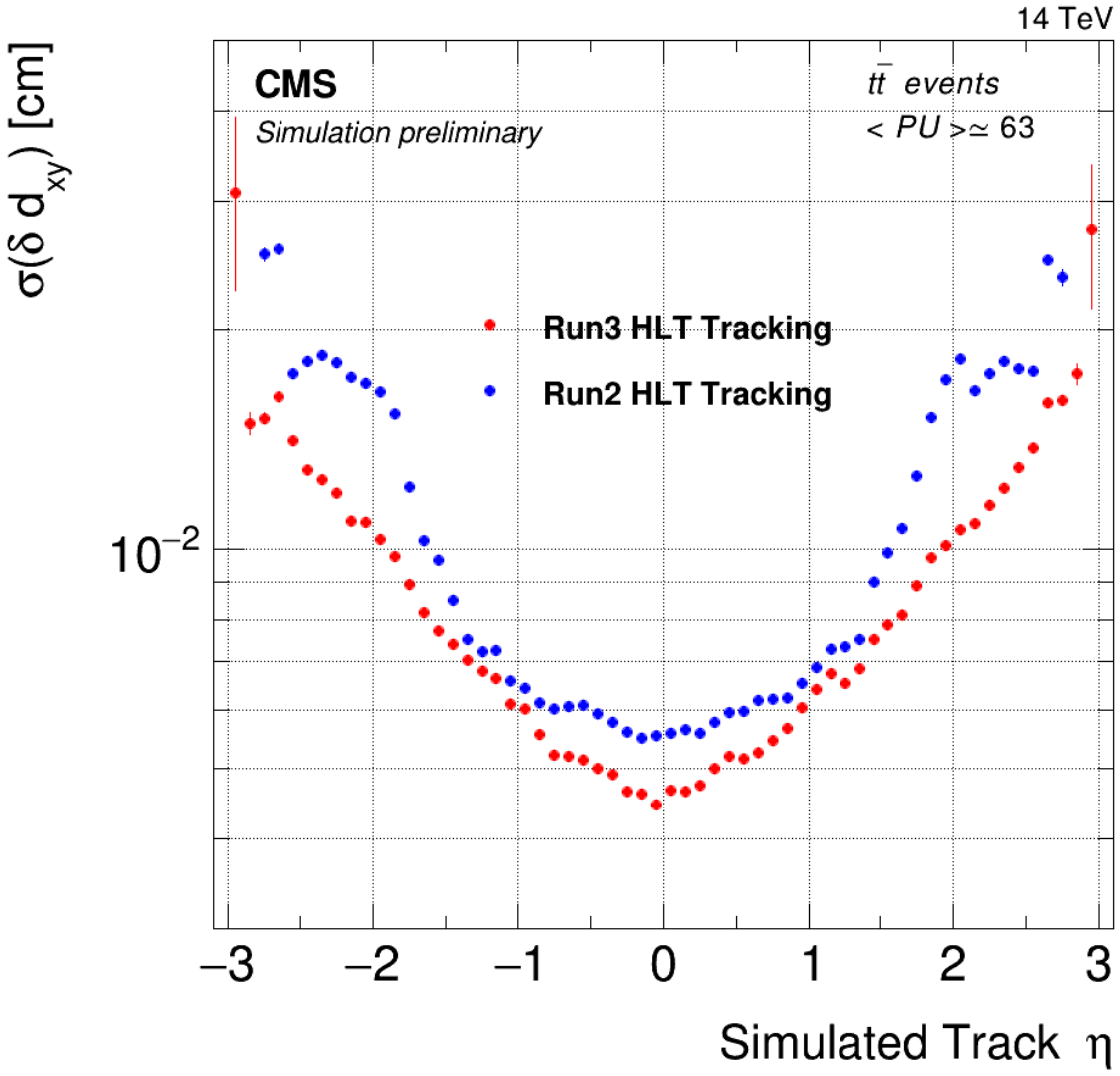
- HLT througput

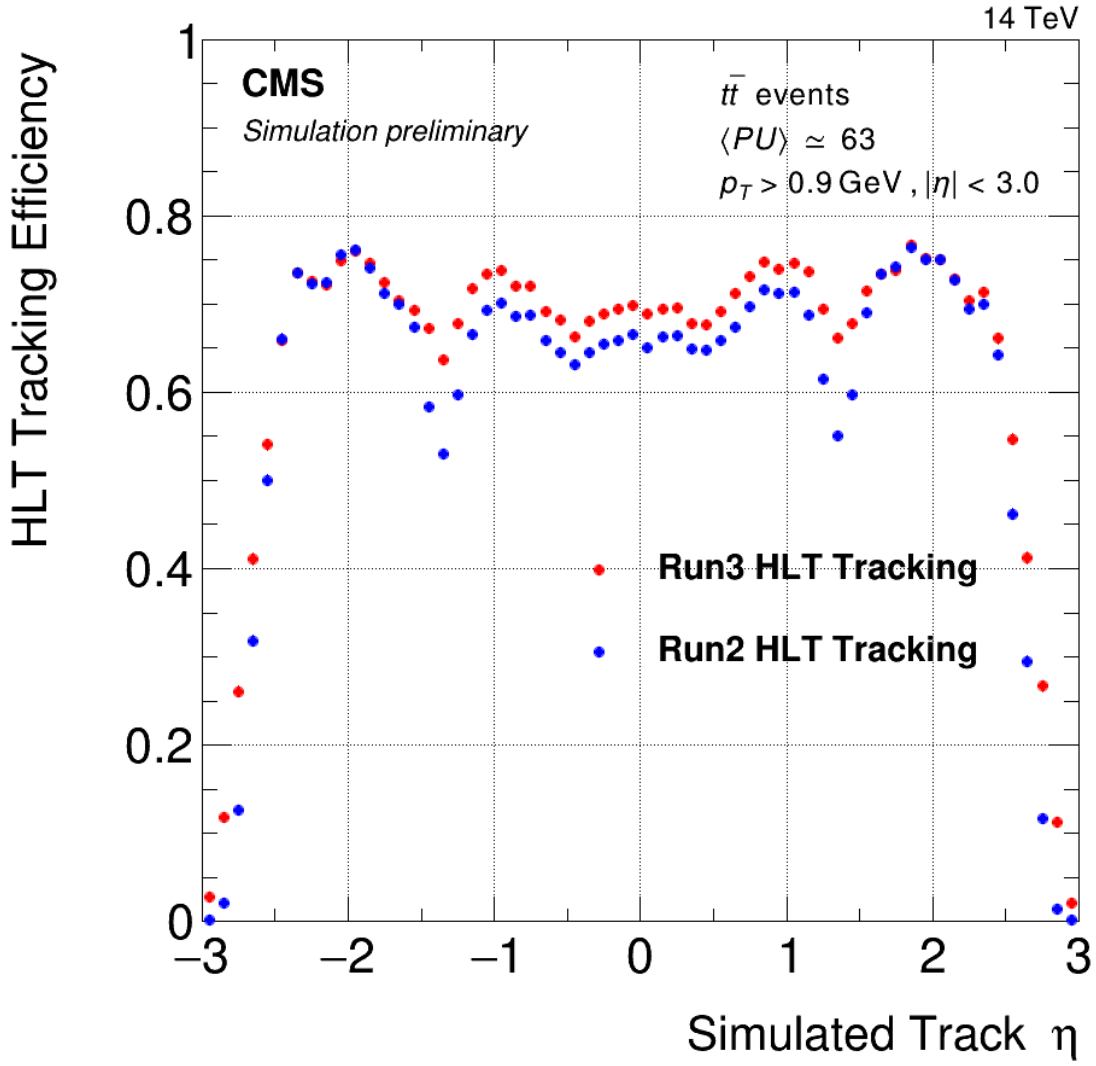


Coupling different CPUs and GPUs, either of the component can be the limiting factor.

And physics performance?

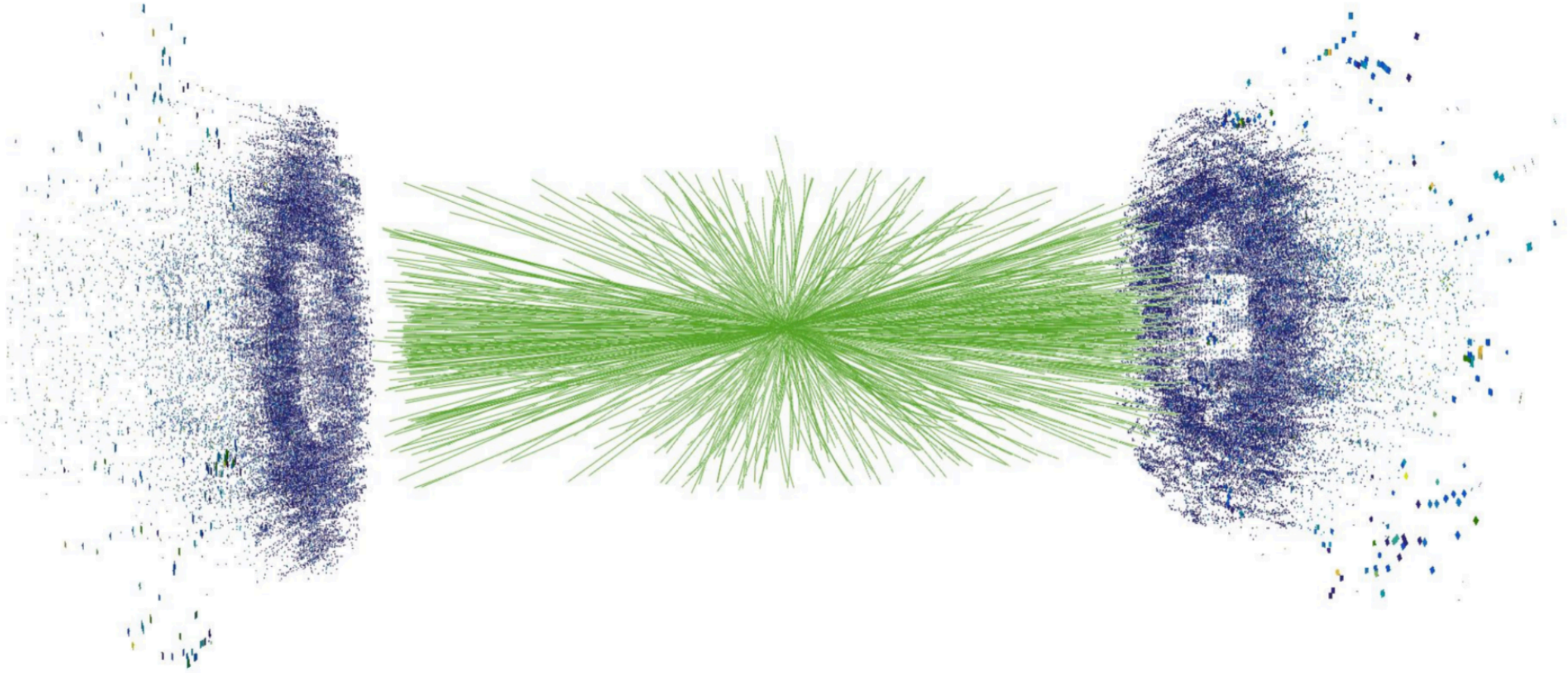
- better tracking performance





(further performance plots here)

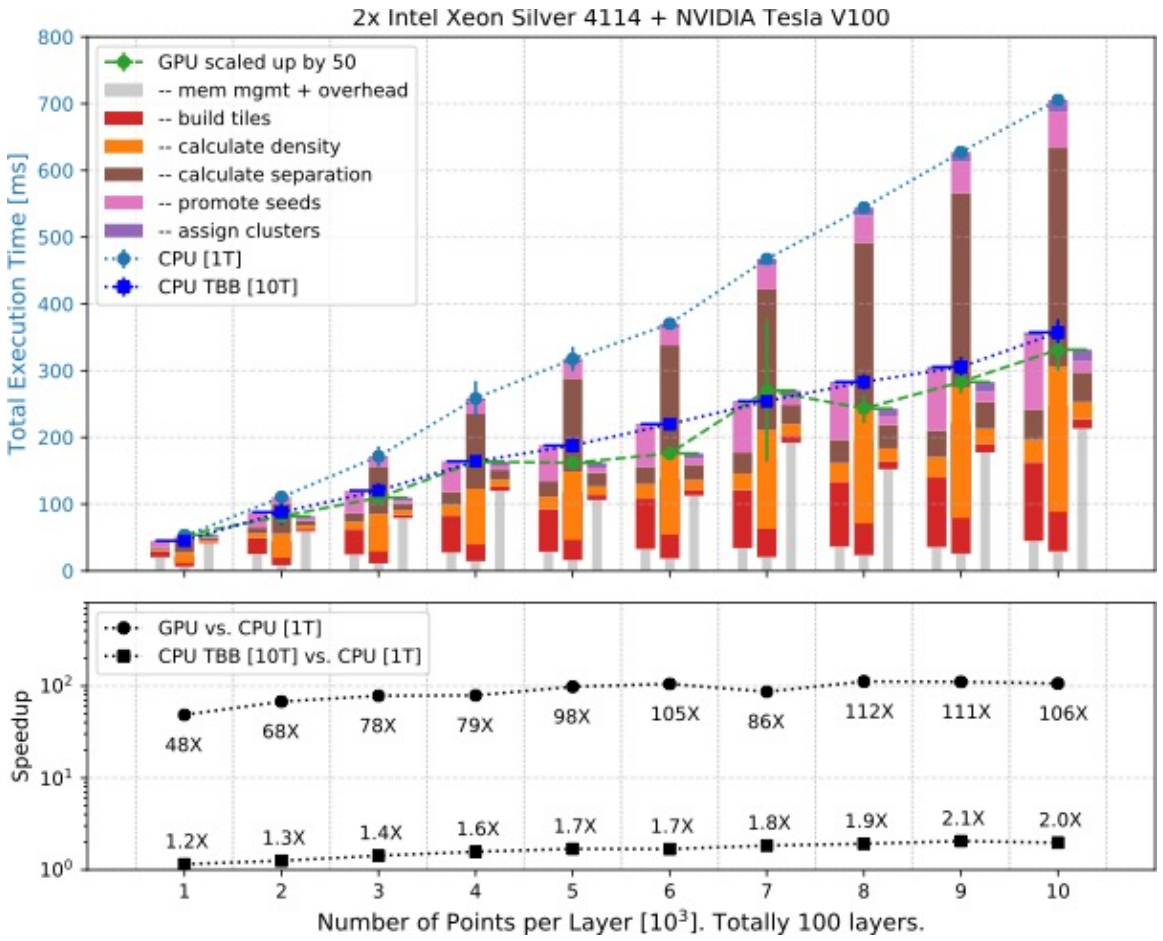
And Phase2?

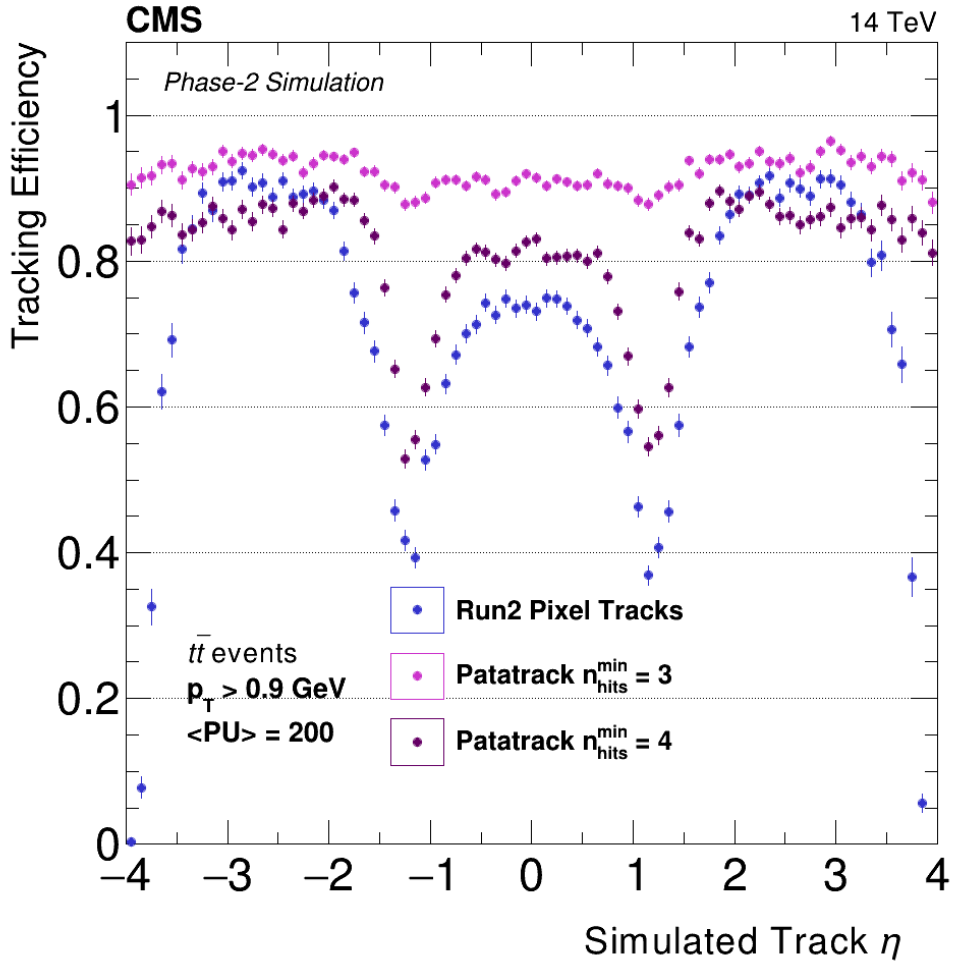


single electron in PU200

- Two main actors at the moment with milestone already reached for the HLT TDR:

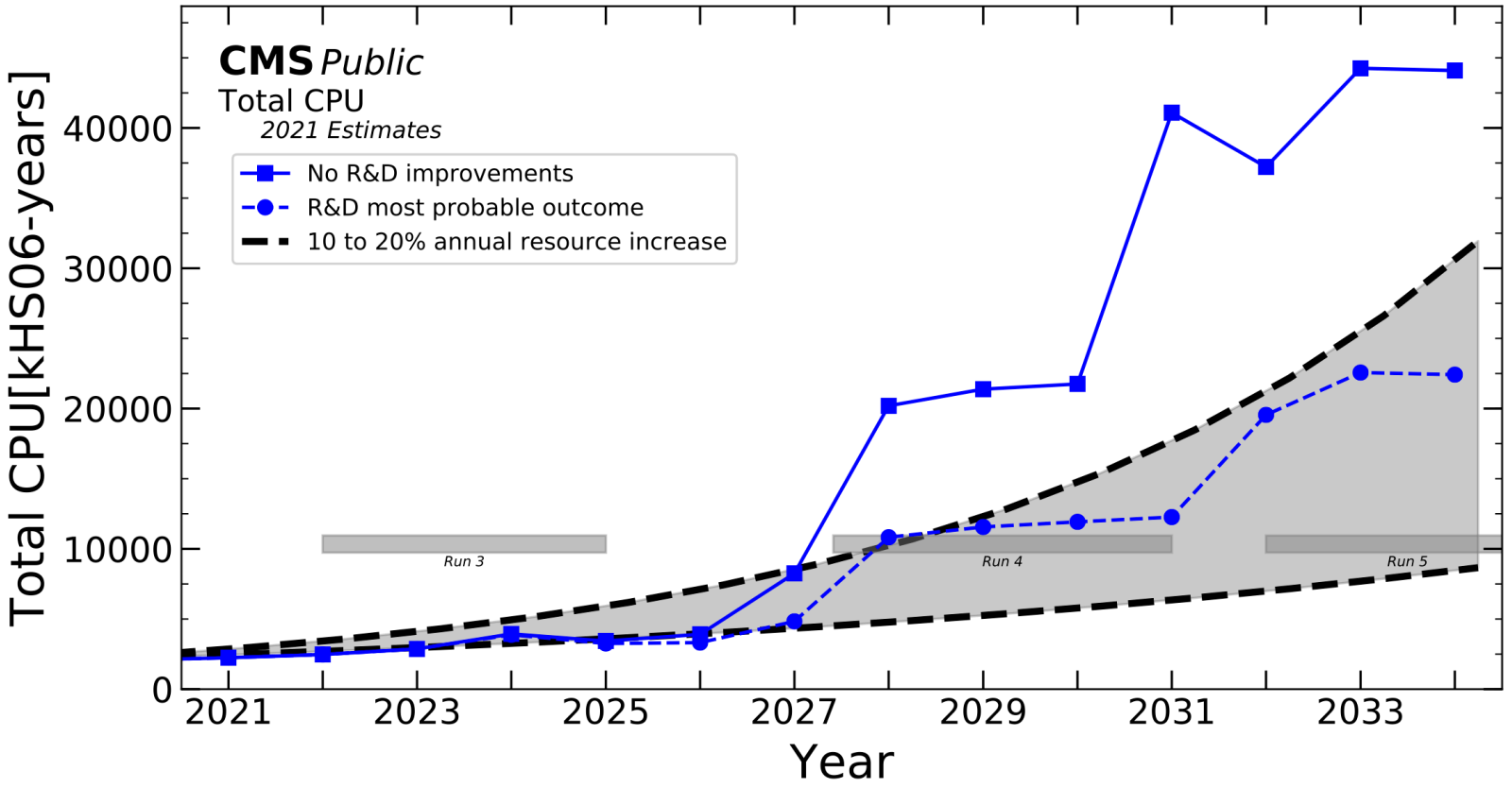
1. HGCal: designed from the beginning to be parallelizable and portable to GPU, from layer clustering (**CLUE**) to shower reconstruction. Modular and iterative framework: **TICL.**
2. Tracker: inherit and adapt what has been developed for Run3. Additional efforts for the Outer Tracker.





And for offline?

- Again, Phase2 conditions are the driving force (but not only):



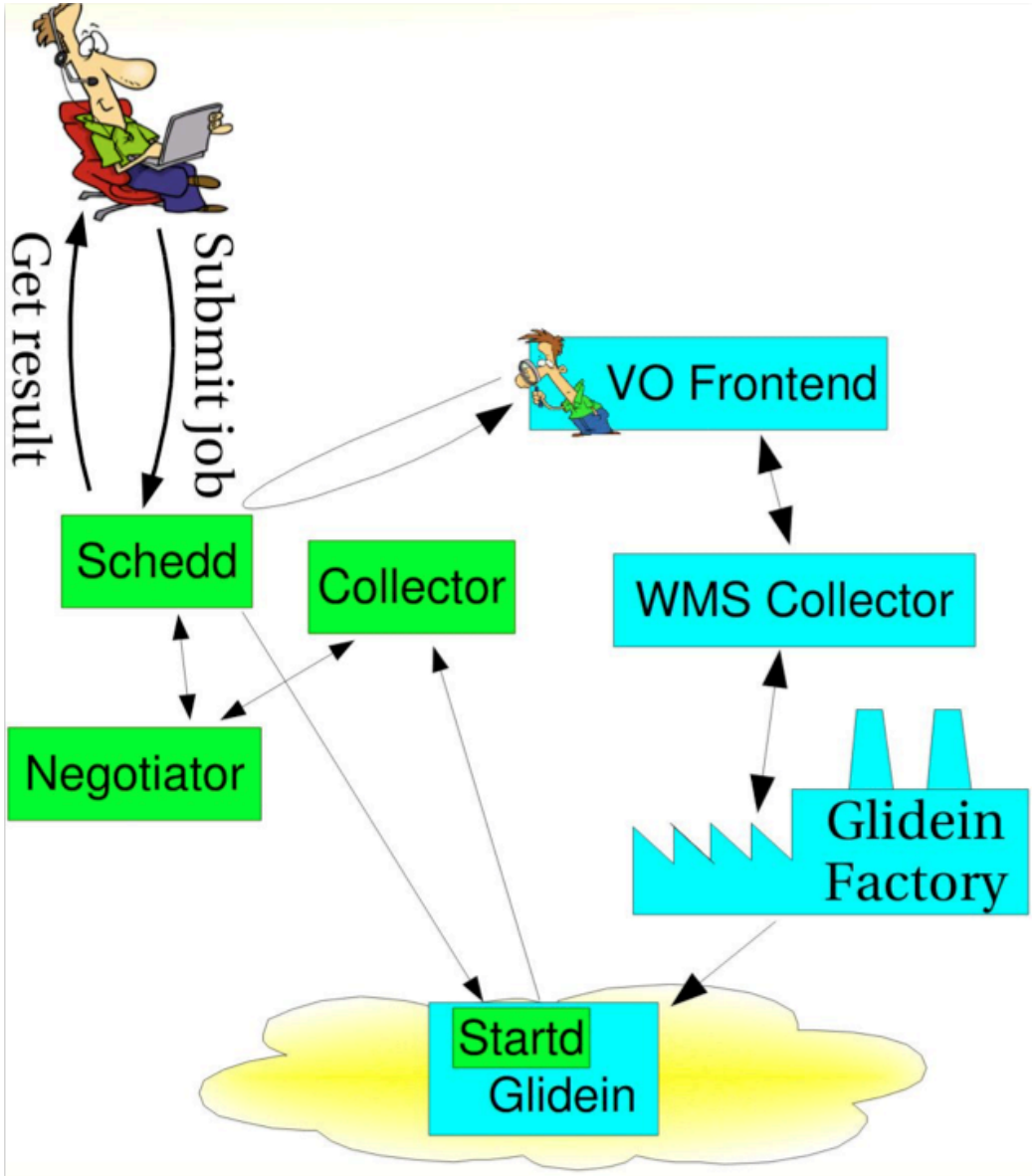
N.B. no GPU developments included in the trends

- The HLT reco has been the kickstarter for hetherogenerous coding around CMS and GPU-based algorithms are spreading quite fast among all the groups also for offline reconstruction.
- There are few "low hanging fruit" coming from HLT experience, from which CMS will profit in a short-mid term. Also adaptation of well established algos seems a way to go. Also Some developments do not directly aim to timing reduction as first issue but more to an harmonization of data structures and algos towards a full(-ish) heterogeneous reco.

From fall 2019 CMS started to investigate how to enable CMS WM and SI to make use of distributed GPU resources.

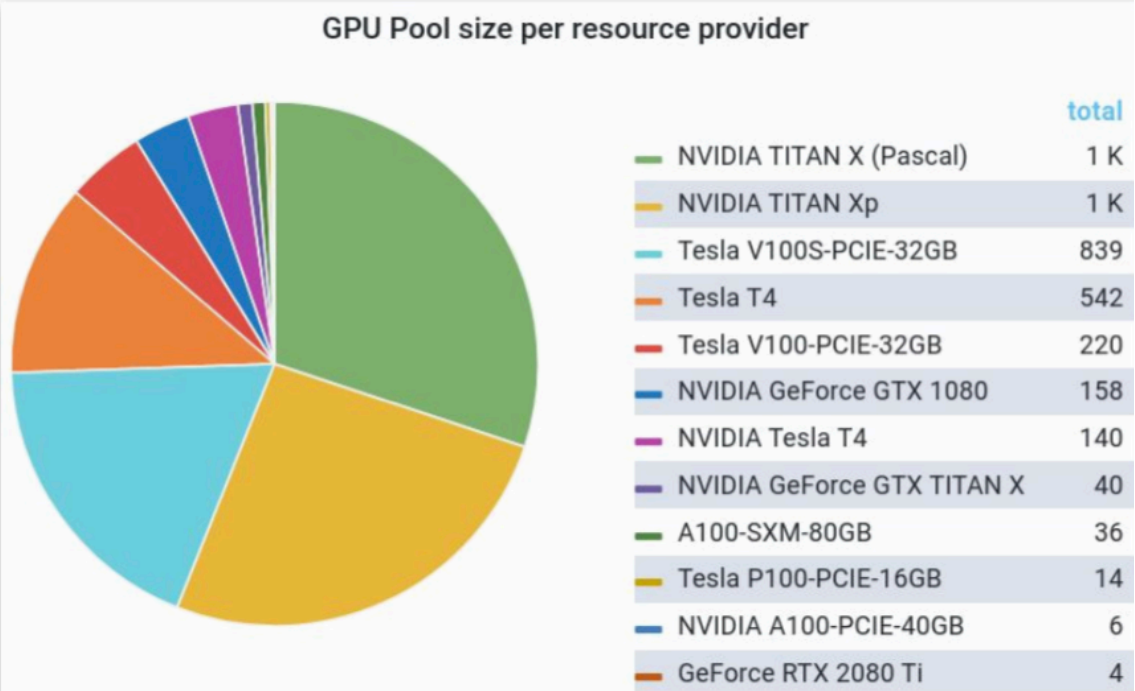
Resources are opportunistic (for now) sites voluntarily adding GPUs to CMS pool.

- **Regularly scanning the Global Pool for GPU resources** and their properties
- HTCondor **matchmaking**: attach resources to jobs.
- Already **GPUs are available** in the Global Pool from a number of sites

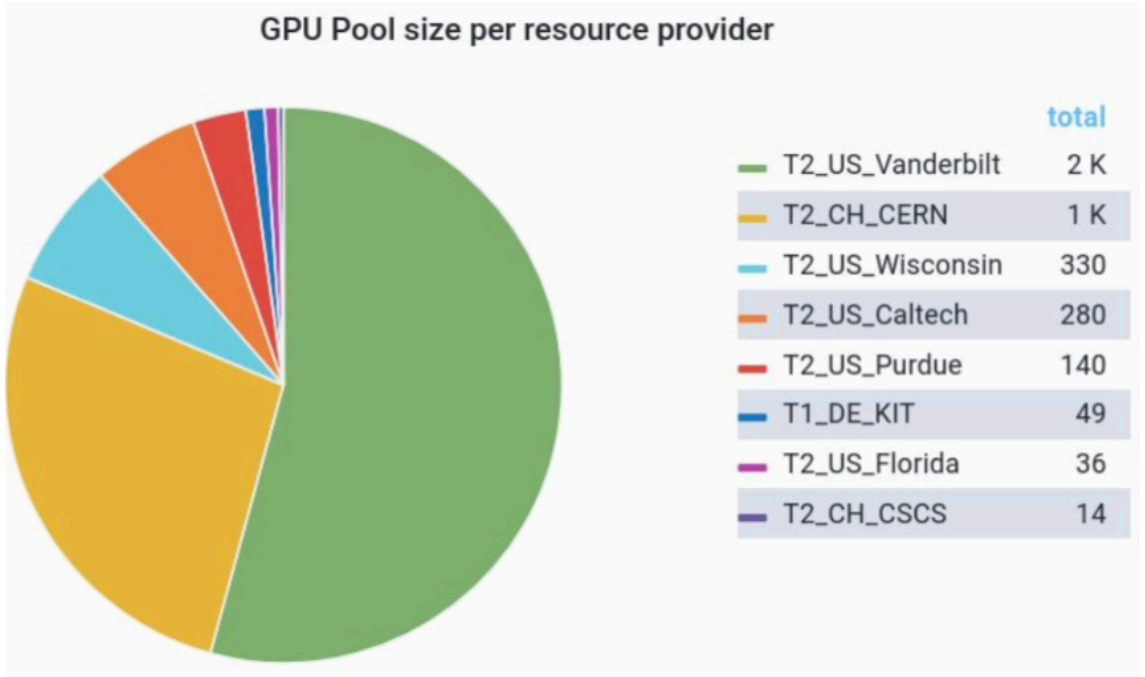


- Successfully executed tests with real workflows

Great variety of GPUs!



Multiple Sites In the Pool



Portability: support multiple accelerator platforms with minimal changes to code base :

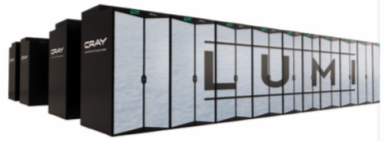
- Rewriting the same code for each architecture is not feasible
- Easier maintenance
- **Avoid vendor lock-in!**
- Going to offline **distributed** reconstruction means «heterogeneity», also: HPCs (5% for CMS in 2019-2020)!

Frontier ORNL, 2021
AMD CPU, AMD GPU, 1.5 ExaFlop

Leonardo, Cineca, 2021
Intel CPU, NVIDIA GPU, 200+PFlops

El Capitan LLNL, 2023
AMD CPU, AMD GPU, > 1.5 ExaFlop

LUMI, CSC, 2021
AMD CPU, AMD GPU, 550 PFlops



A complete C++ standard for heterogeneous computing **is way in the future. Need to rely on portability layers:**

- Kokkos, Alpaka

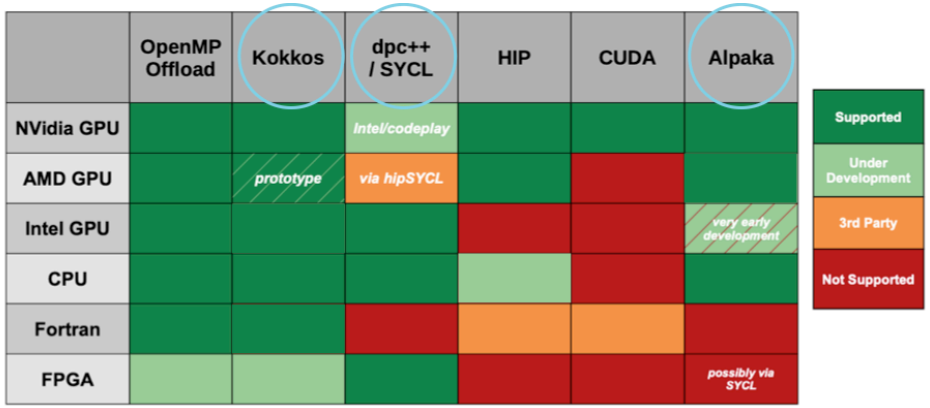
alpaka kokkos

In Run 3 timescale:

- Given the use cases, we require the portability layer to have good CPU and CUDA backend
- Migrate CUDA GPU codes to use portability layer

In Run 4 timescale:

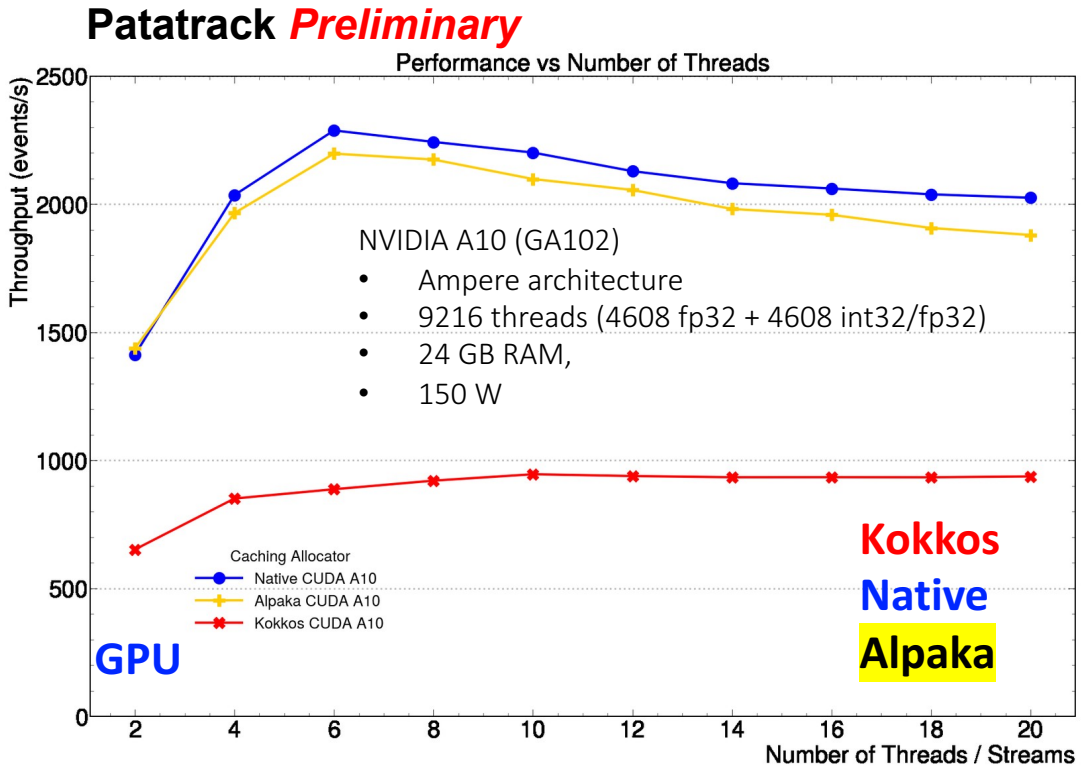
- Support as much architectures as we can
- Landscape (software & hardware) maybe very different by then: no decision casted in stone.
- May need to think beyond GPUs (FPGAs?)

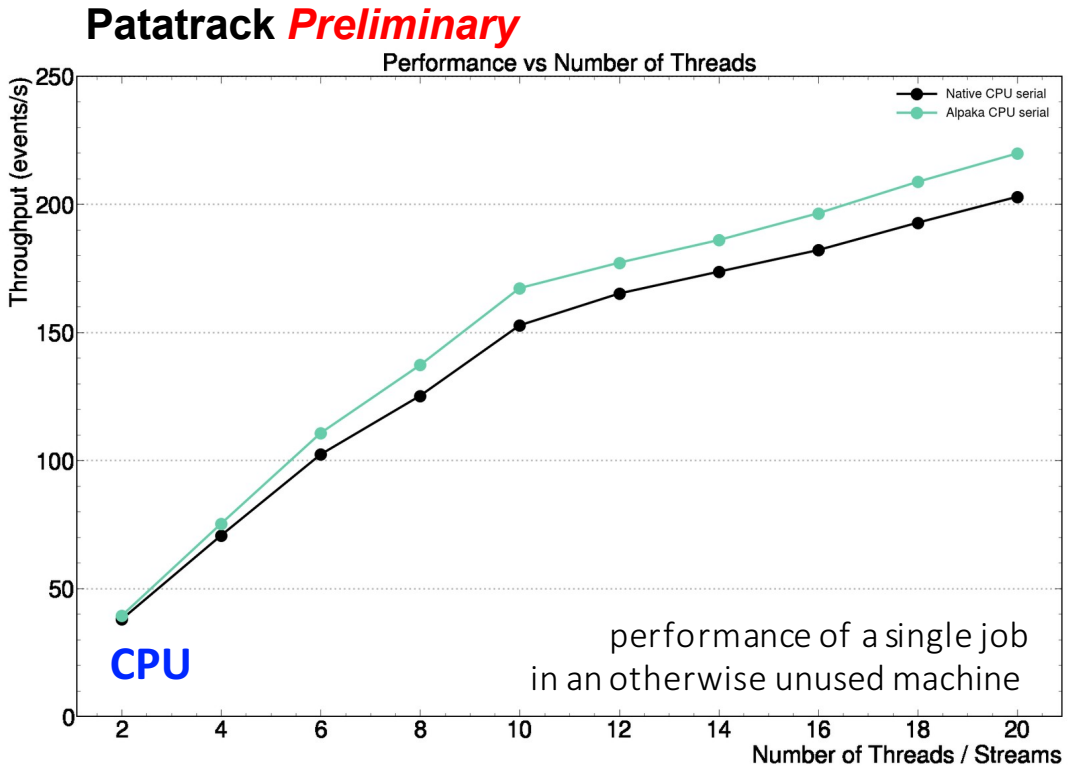


Perfromance Portability Results

- Patatrack and HEP-CCE's pixeltrack-standalone project (git)
  - prototype different data structures user friendly SoA abstractions
  - port to different backends
  - test different *performance portability* solutions Kokkos, Alpaka

- Throughput results for the patatrack-standalone prototype:





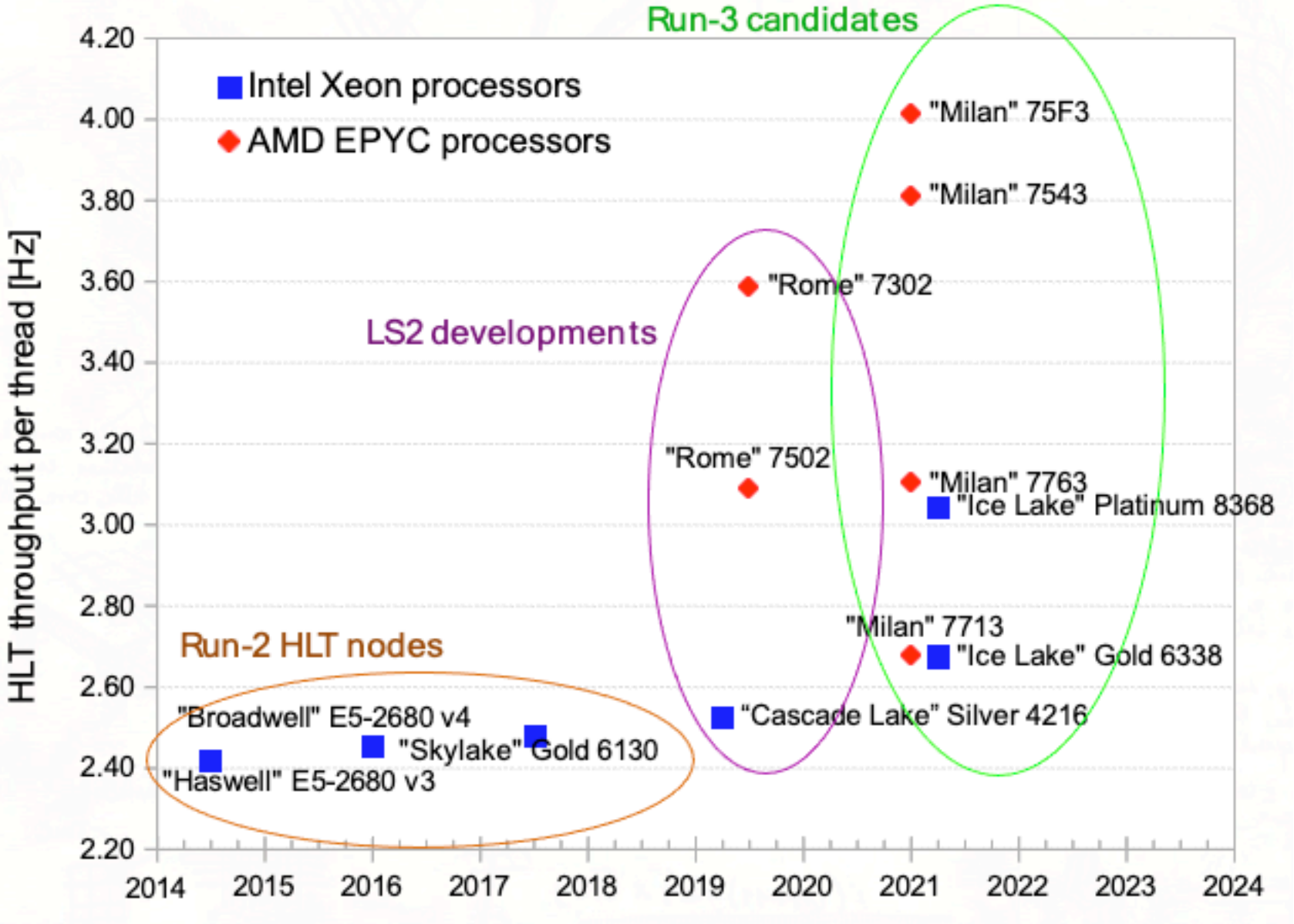
alpaka achieves ~ 95% of the native performance for the CUDA and CPU backends --> It has been chosen as Run-3 perfromance portability layer.

(under investigation )

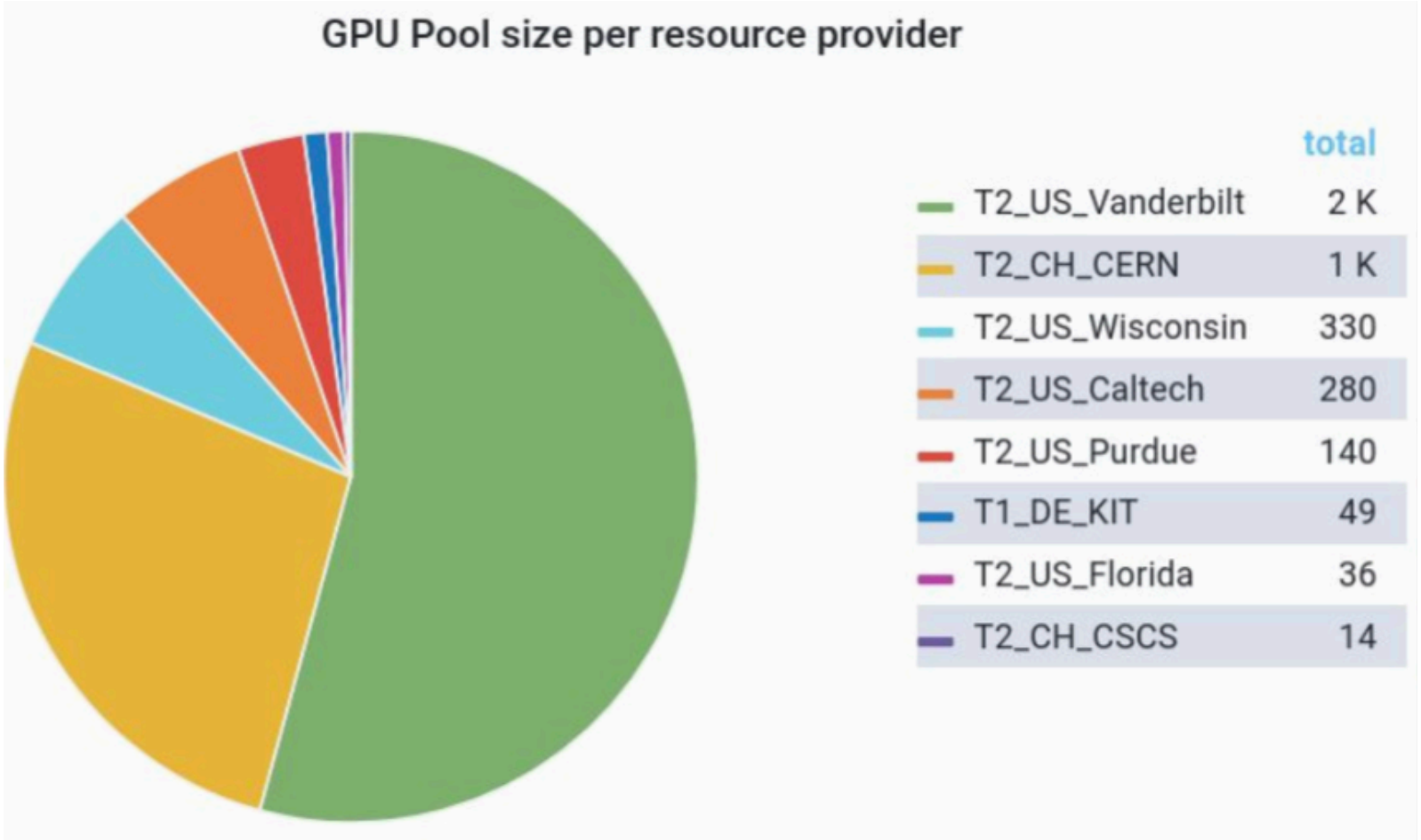
Summary & further thoughts

- After work kickstarted by the Patatrack incubator:
  - **Multiple sub-system reconstruction and algos** are being ported: Particle Flow, Muon Seeding, E/y electron seeding, primary vertex reconstruction just to cite a few.
  - Writing and running code on GPU is an a **normalization phase** in CMS.
- The **distributed infrastructure** is ready to cope with running the offline reco on GPUs and the software is catching up.
- **Code portability** is the next big thing for CMS that will (among other advantages) free us from the single-vendor and will ease code porting from the user perspective.
- Some futher topics of discussion
  - SoA abstraction;
  - unified memory managing and handling;
  - CPU vs GPU validation: with a a focus on results (ir-)reproducibility [that are here to stay]

HLT CPU Performance



GPU Pool Size



Further readings

- Performance portability for the CMS Reconstruction with Alpaka
- Heterogeneous techniques for rescaling energy deposits in the CMS Phase-2 endcap calorimeter
- Clustering in the Heterogeneous Reconstruction Chain of the CMS HGCAL Detector
- CMS High Level Trigger performance comparison on CPUs and GPUs
- Developing GPU-compliant algorithms for CMS ECAL local reconstruction during LHC Run 3 and Phase 2
- CLUE: a clustering algorithm for current and future experiments
- The Iterative Clustering framework for the CMS HGCAL Reconstruction
- https://github.com/cms-patatrack/pixeltrack-standalone
- Alpaka@ACAT2019
- Compute Accelerator Forum / HSF Reconstruction and Software Triggers - Patatrack and ACTS
- CMS Phase2 CMS TDR
- Reproducibility