HEP Software Foundation

HEP Software - Status and Future

Graeme A Stewart, 2023-11-29

ICFA Seminar, DESY 2023

HEP Software Today

Tabula Orbis Terrarum Software

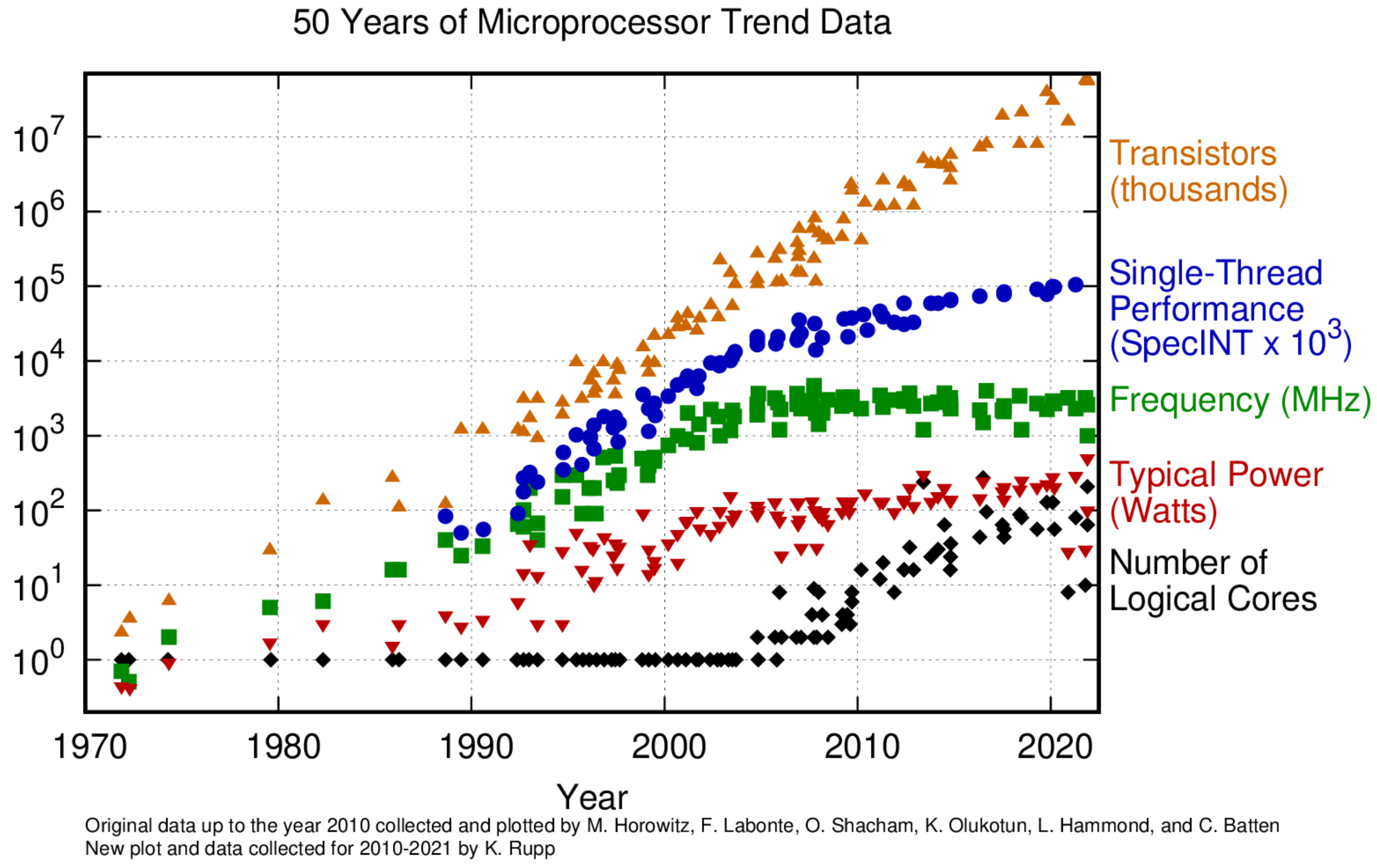
- There is a whole world of software
- Generators, simulation, reconstruction, analysis… support software
- O(50M) lines of code

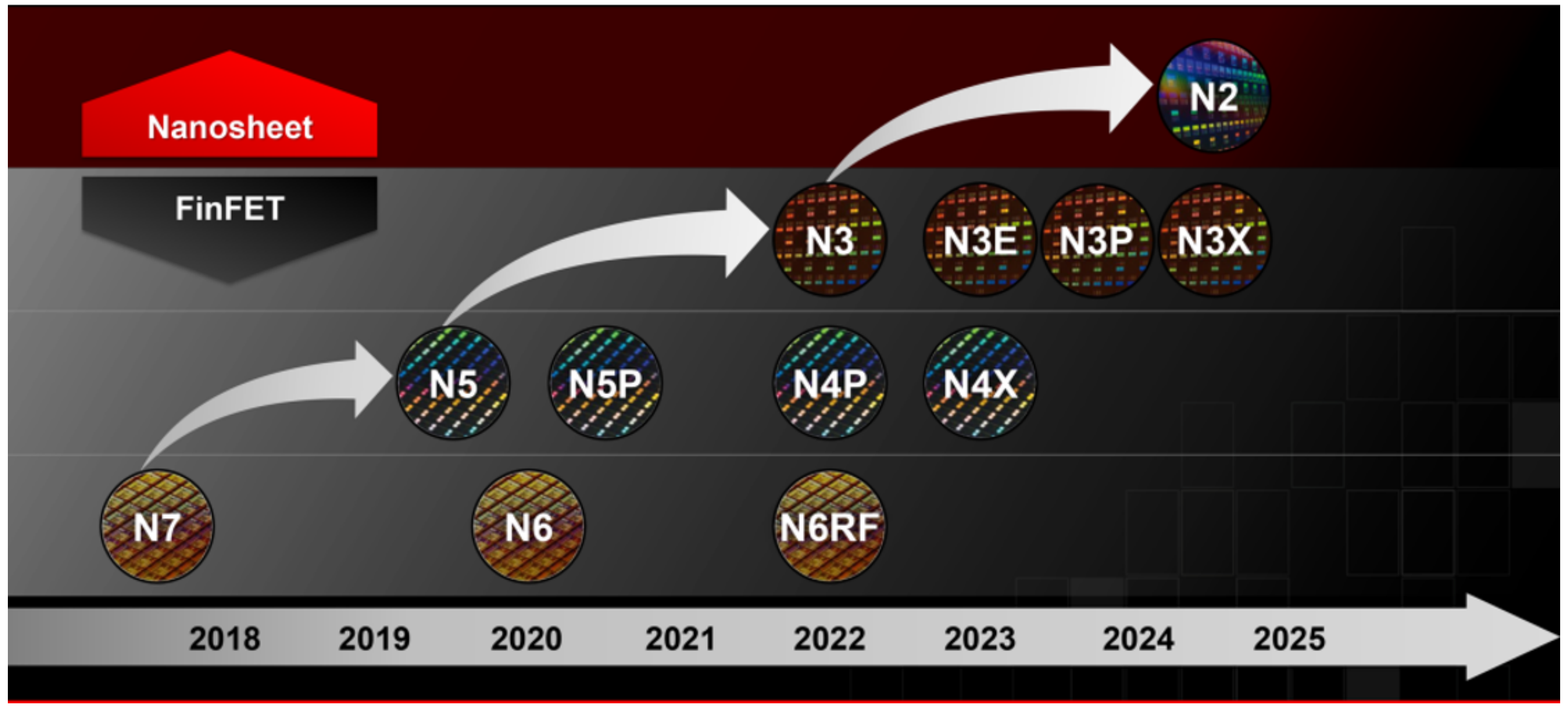
Ligua Franca

- Core software is in **C++**
    - Low level language that sits pretty close to the metal
    - Higher level abstractions are reduced at compile time
    - Difficult language
        - Extensions to handle more exotic devices: GPUs and FPGAs
- Steering, control and user interfaces dominated by **Python**
    - Higher level interpreted language
    - Increased human productivity and comfortable interfaces
    - Usually backed by low level C/C++ codes when performance is important (numpy, numba, PyTorch and friends)
- Some use or interest in other languages, e.g., Fortran, Java, Javascript, Rust, Julia
    - Mixture of legacy applications, niche areas and interesting future directions

CMOS Transistors

- Moore's Law continues to deliver increases in transistor density - at least for now!
  - Increasingly challenging technical issues, but there is a roadmap to 2nm by 2025
  - Transistors now consist of only O(10s) of atoms, so we are in the endgame
- Clock speed scaling failed many years ago
  - No longer possible to ramp the clock speed as process size shrinks
  - Leak currents become important source of power consumption
- So we are basically stuck at ~3GHz clocks from the underlying Wm^{-2} limit
  - This is the Power Wall
  - Limits the capabilities of serial processing
- Memory access times are ~100s of clock cycles

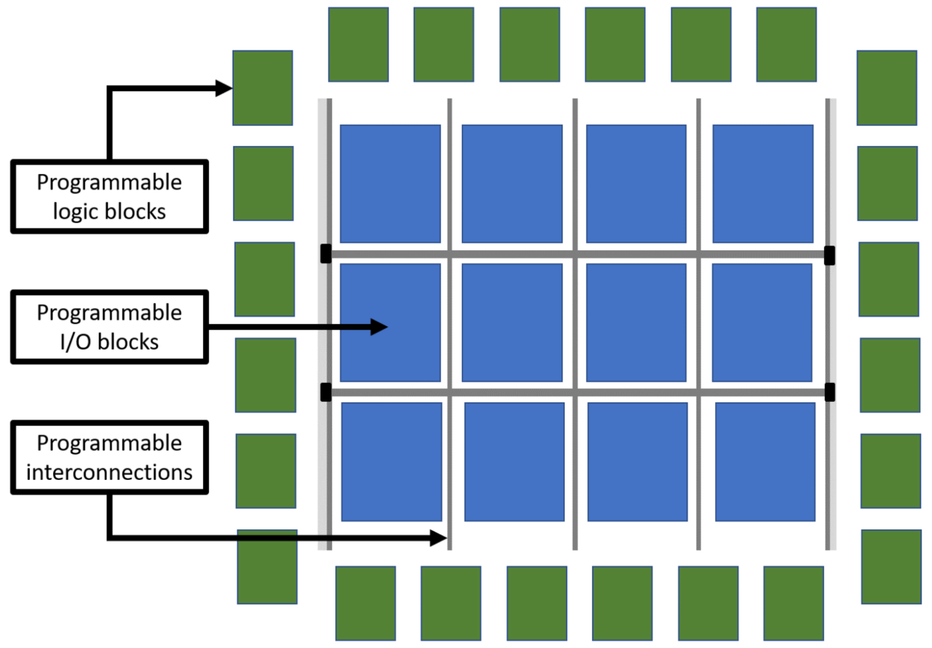




TSMC technology roadmap

Decreasing Returns and Diversity

- Diversity of new architectures will only grow
    - Chiplets technique enables “Lego” style custom chips
- AMD currently ruling in the CPU domain
    - New ARM data centre chips are competitive - particularly good in power efficiency
- NVidia dominate the GPU market
    - New datacenter architectures address the memory bandwidth issue
- Intel are a bit missing in action…
- Long term strategic investments in RISC-V architecture (Europe)
    - EU and US push to onshore chip making capacity



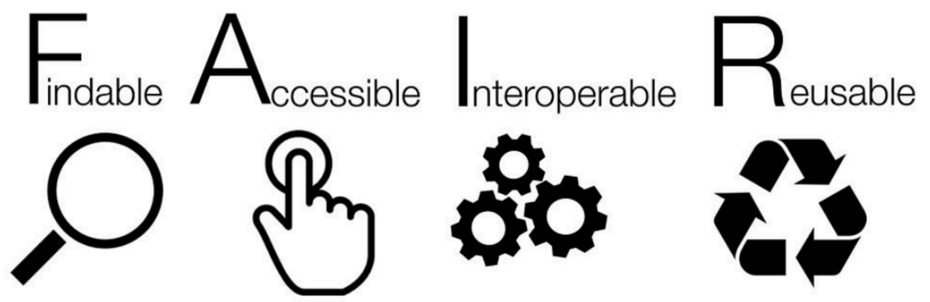
FPGAs implement data and logic flow directly on their hardware

GPUs dedicate far more transistor area to arithmetic calculations

Challenges!

- Maintenance of current software
  - Need to keep bit rot at bay and develop capabilities of existing codes
  - However, eventually we need to also deprecate software!
- Much reduced processing budget per event (money and energy!)
  - 10x more events in the same budget means we need to improve by a factor of 10
  - This is a challenge that can be framed in GWh/fb^{-1}
- Shifting hardware landscape
  - Move from pure CPU processing to a heterogeneous landscape
- Shifting skills base
  - Diminished skill levels in C++ in particular
- FAIR for software is both a challenge and an opportunity

Green Computing

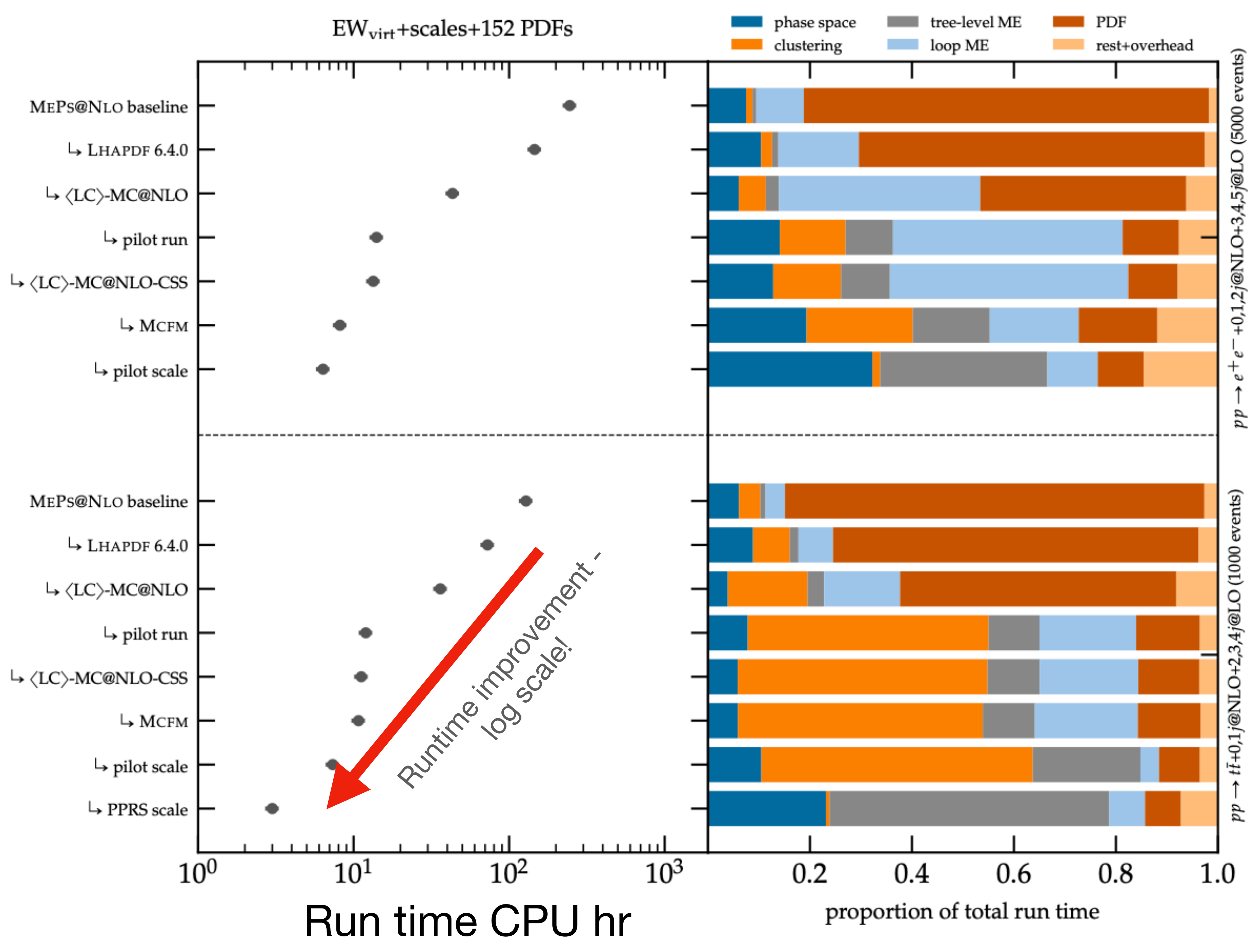


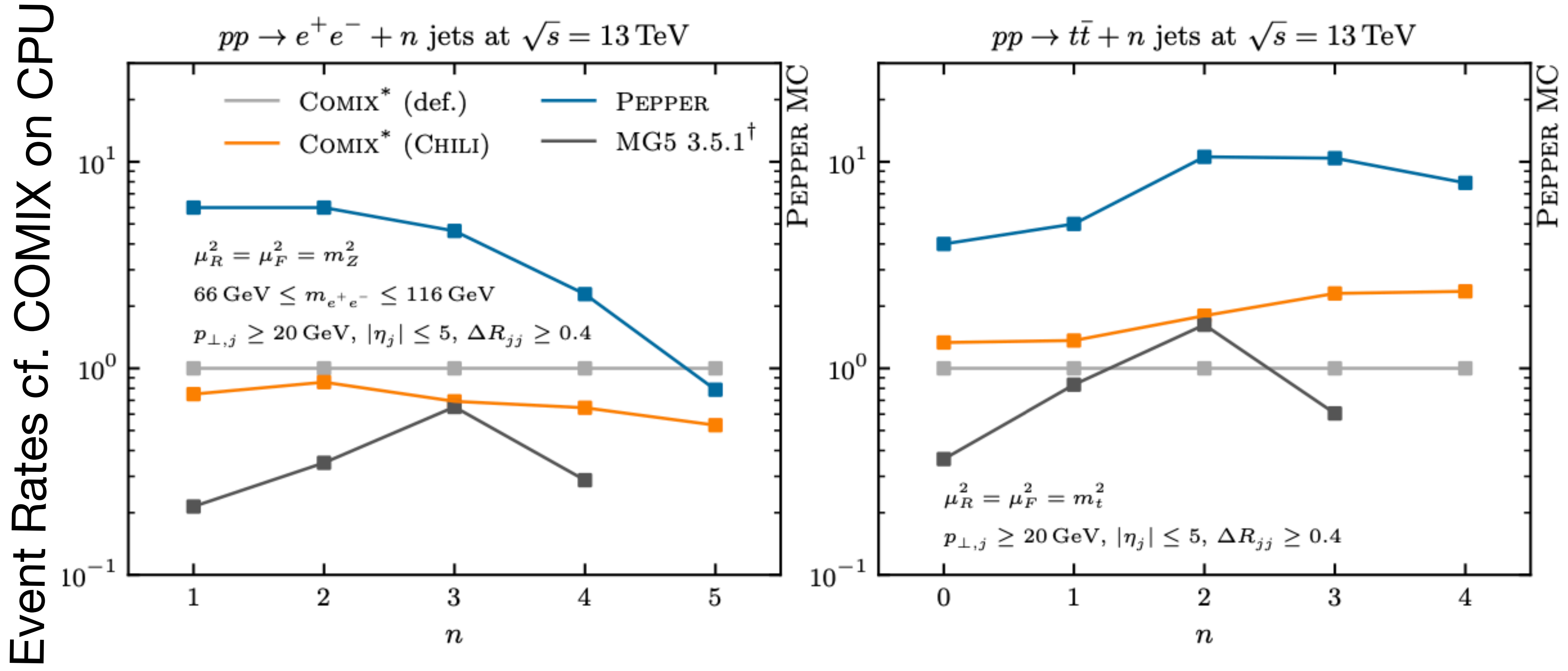
Trends towards the future in HEP Software

See the next talks from Lukas and Donatella for **AI and Machine Learning** and **Quantum Computing**

Generators go forth!

- We can find *significant gains in existing software*
  - Optimisation of Sherpa and LHAPDF, plus improved techniques in sampling
  - Lead to x40 improvement in ATLAS's V+jets sample
    - Gives room for NNLO, N3LO…
- And *write software for new architectures* (pioneered by MadGraphGPU)
  - Newly developed PEPPER + CHILI parton level calculations aimed at high-multiplicity unweighted events
  - Significant speed-ups on CPU, plus now runs on all GPU architectures
    - HPC ready code!

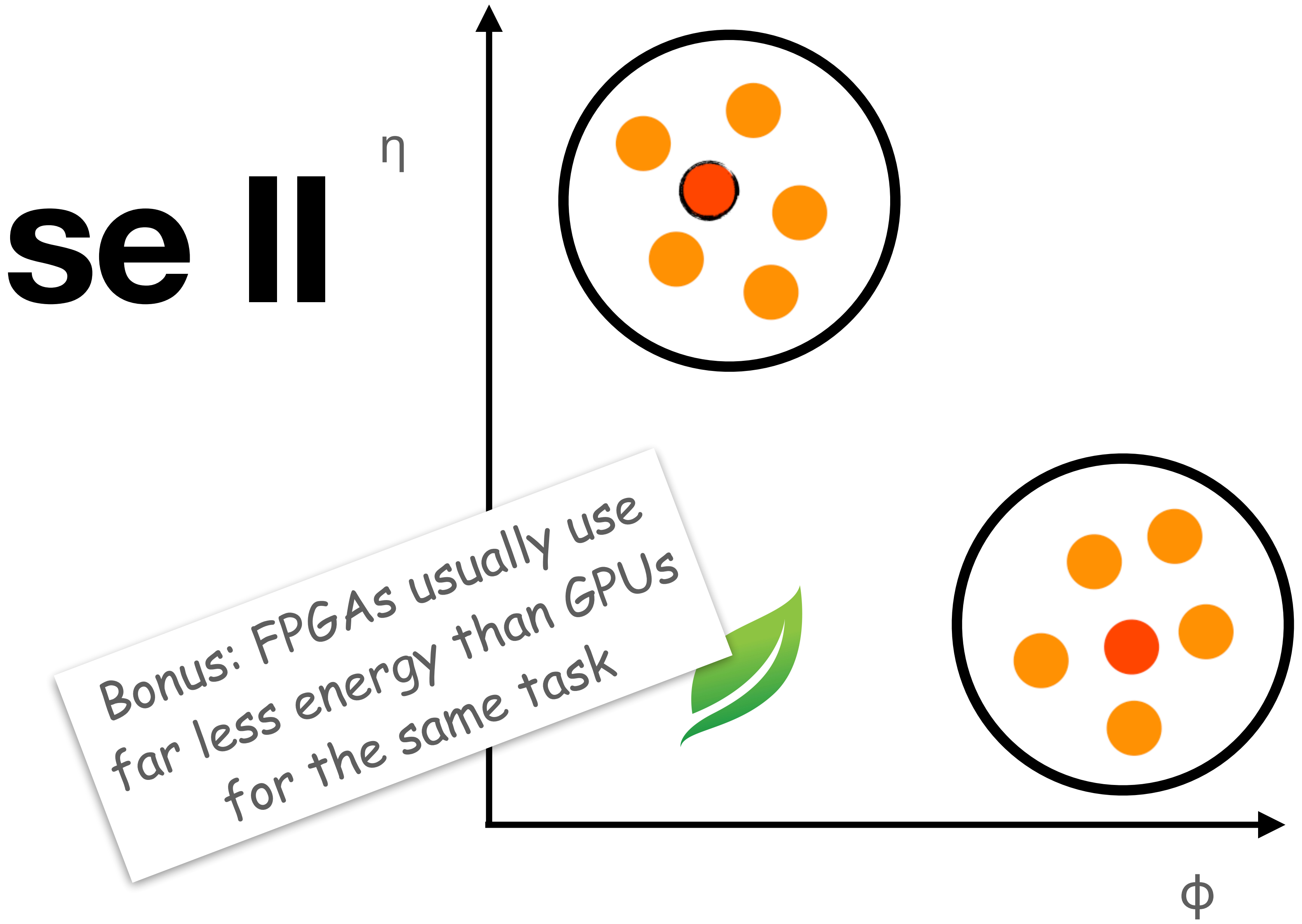


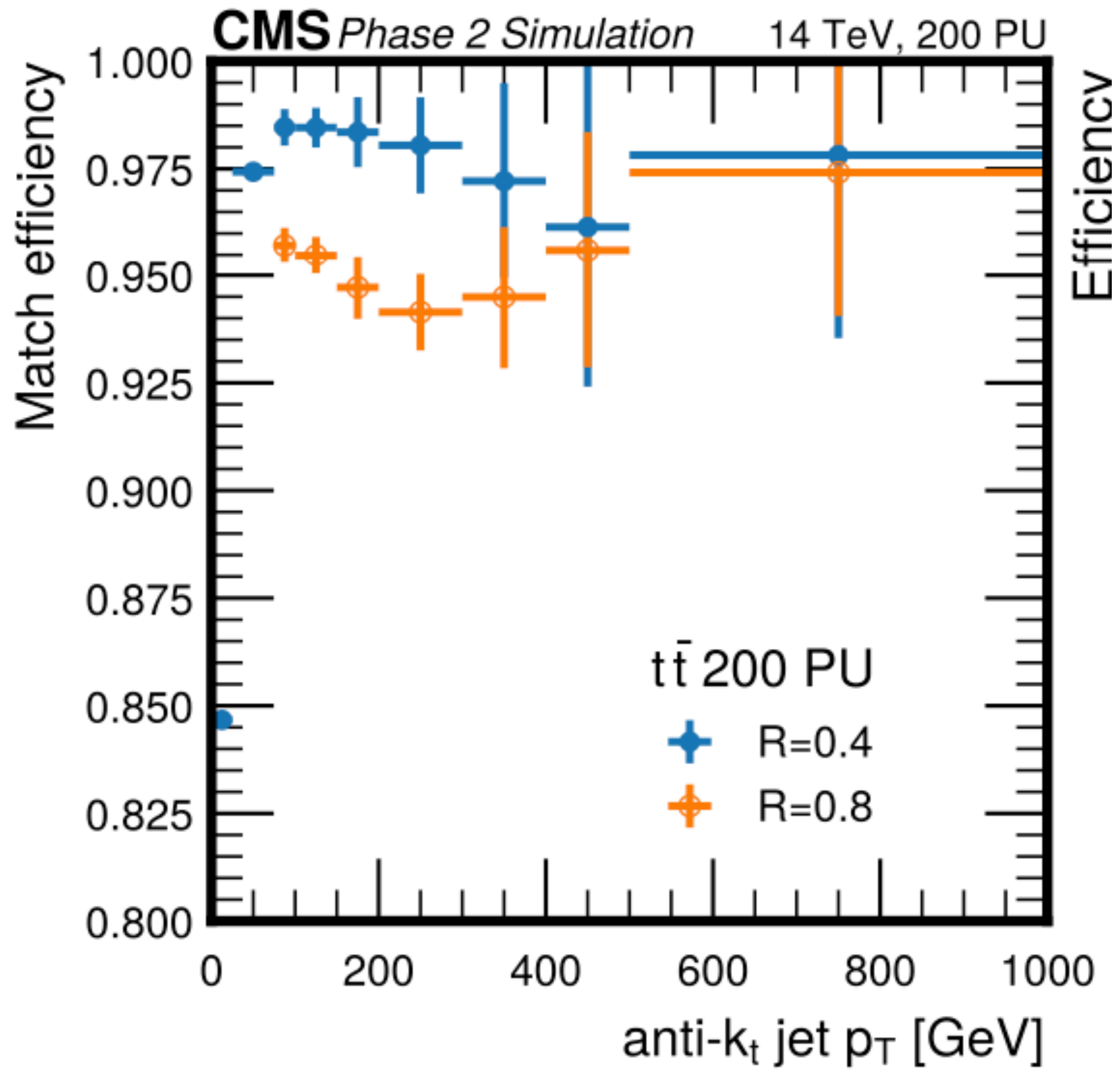


See recent workshop on N(n)LO generators

Trigger Jet Finding CMS Phase II

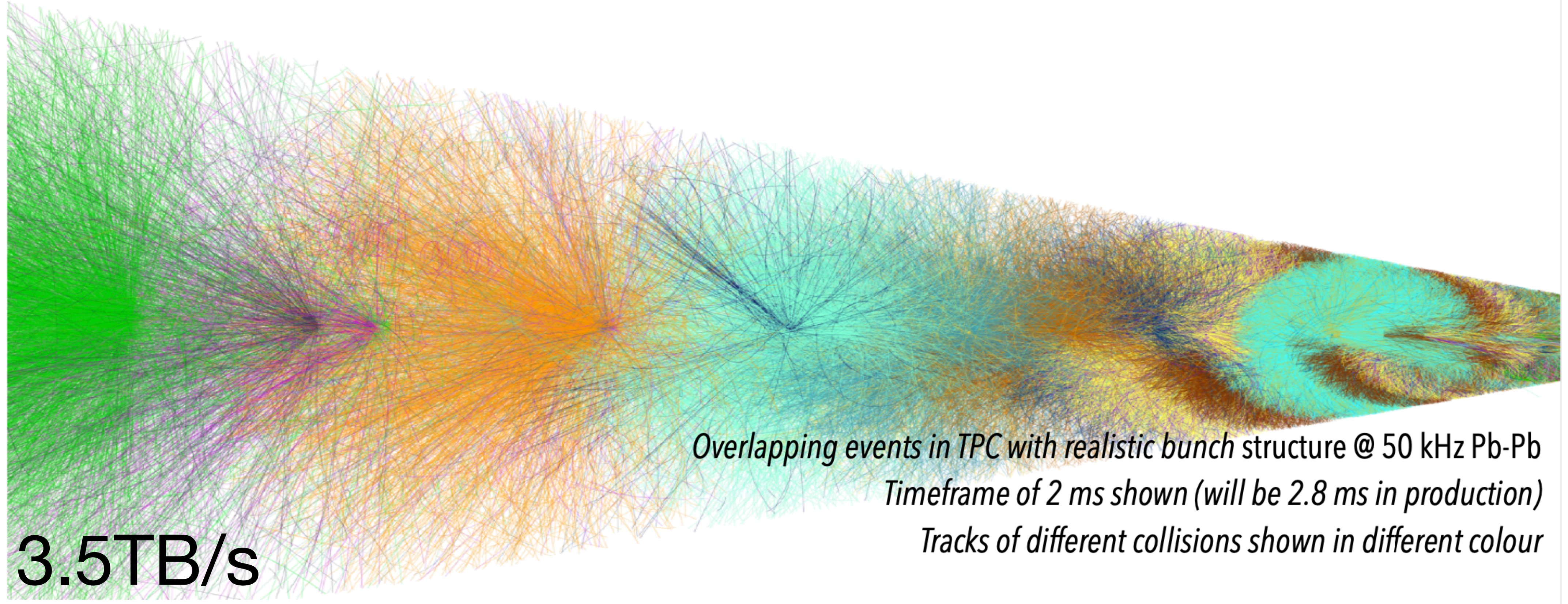
- Push software logic as close as possible to the data
- Jet finding is a clusterisation problem, reassembling the decay components of a higher energy primary particle
  - Data shaping is critical - assemble a flattened array of particle hits from disparate regional data
- Run a seeded cone algorithm, from the most energetic particle find the neighbours and merge into a jet, then repeat
- Very good matching with a more sophisticated offline algorithm (anti-k_T)
- Event processing in 744ns, pipeline processes one event per 150ns
  - 100 million jets per second!
- Use **High Level Synthesis C++** to ease programming an maintainability *this is critical to sustainable code*





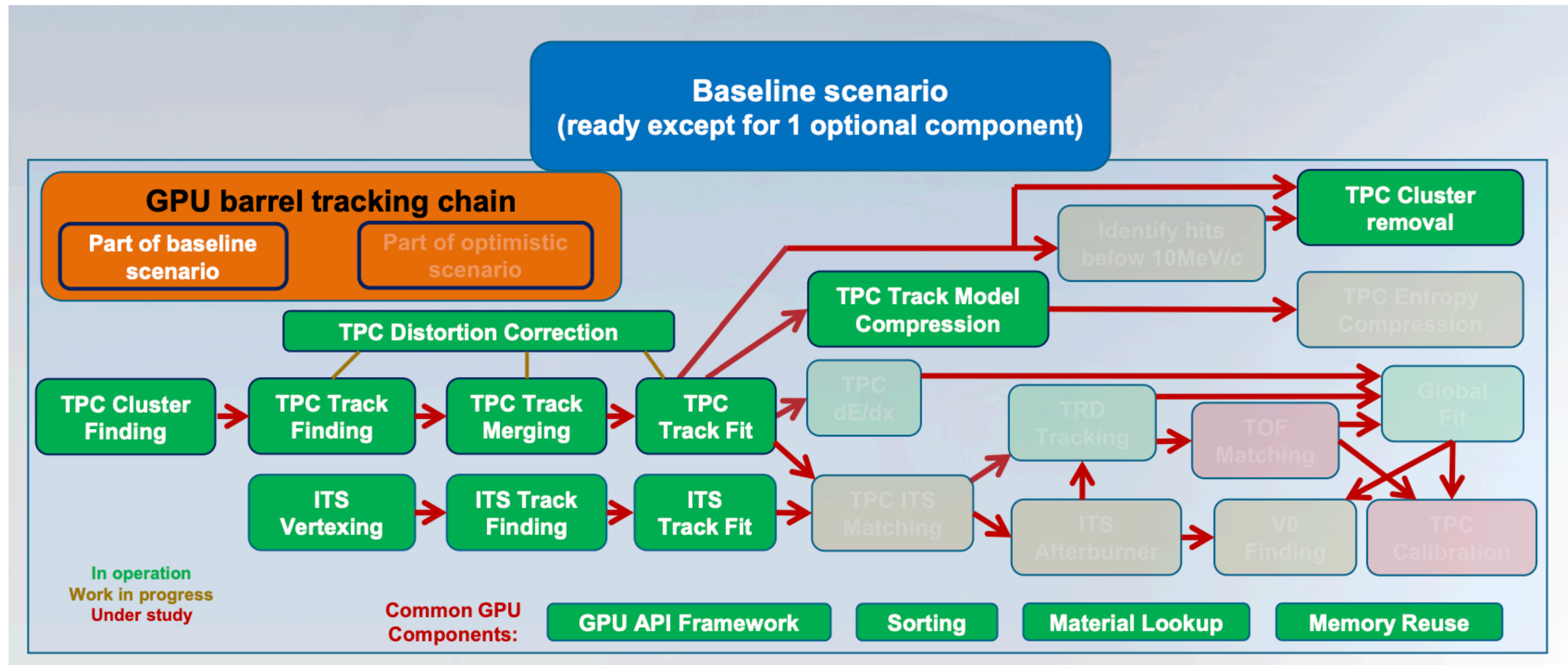
Ref: https://indico.jlab.org/event/459/contributions/11386

ALICE GPU Processing



Overlapping events in TPC with realistic bunch structure @ 50 kHz Pb-Pb
Timeframe of 2 ms shown (will be 2.8 ms in production)
Tracks of different collisions shown in different colour

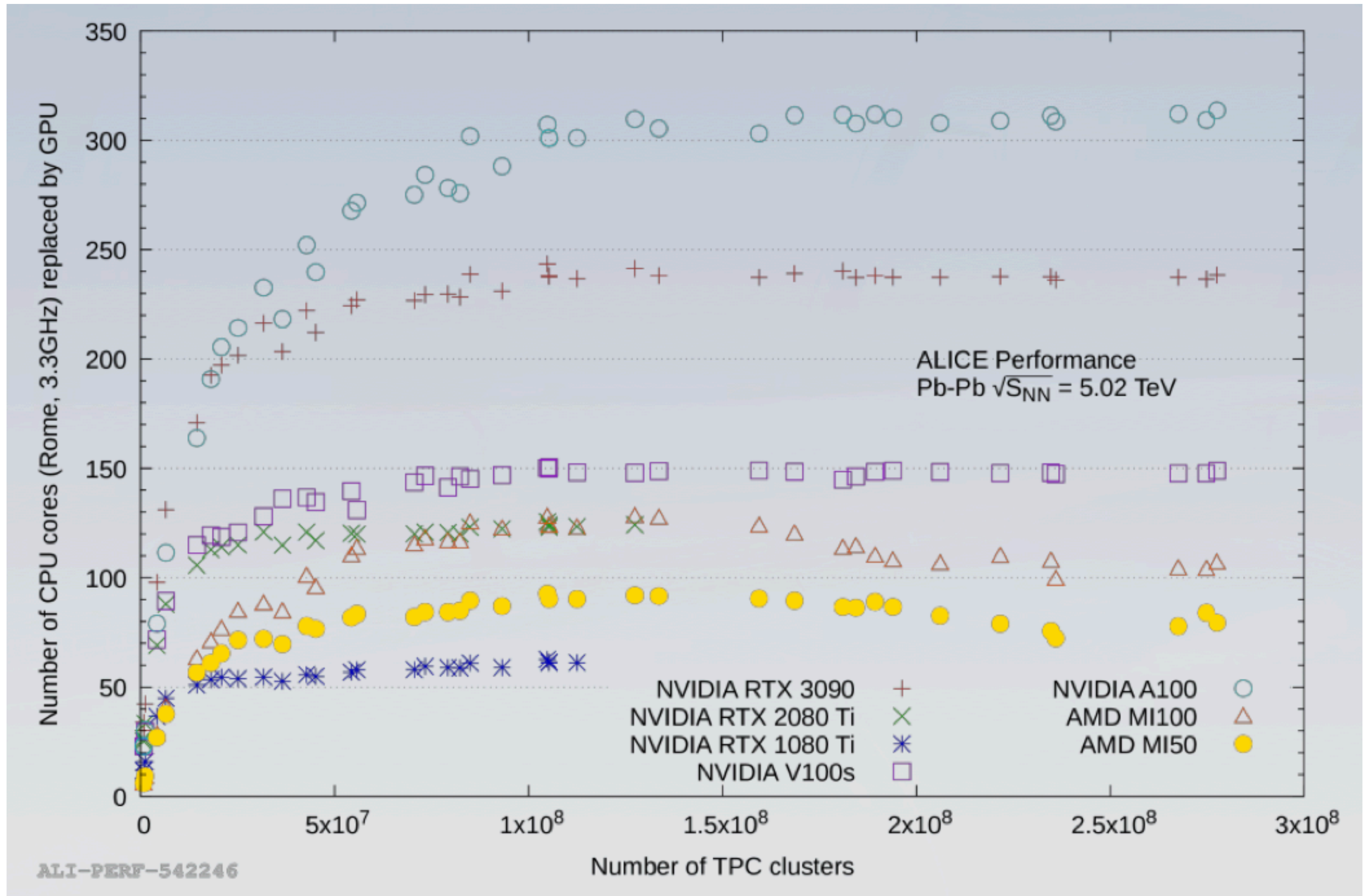
- ALICE reads out PbPb collision data continuously at 50kHz - 3.5TB/s
  - TPC reconstruction dominates first pass of data taking - ideal candidate for GPU
- Modular GPU code
  - Run each part independently, with minimal host interaction
    - *Host to device latency can kill throughput*
  - *Vendor independence*: core code is common C++, wrapper to adapt between Nvidia and AMD cards
- Memory management uses arenas - make a large initial allocation and then sub-divide
- *Asynchronous data transfer* pipelines - process while transferring data
- CPU and GPU should give the same results
  - Small differences due to concurrency or non-associative arithmetic have to be tolerated



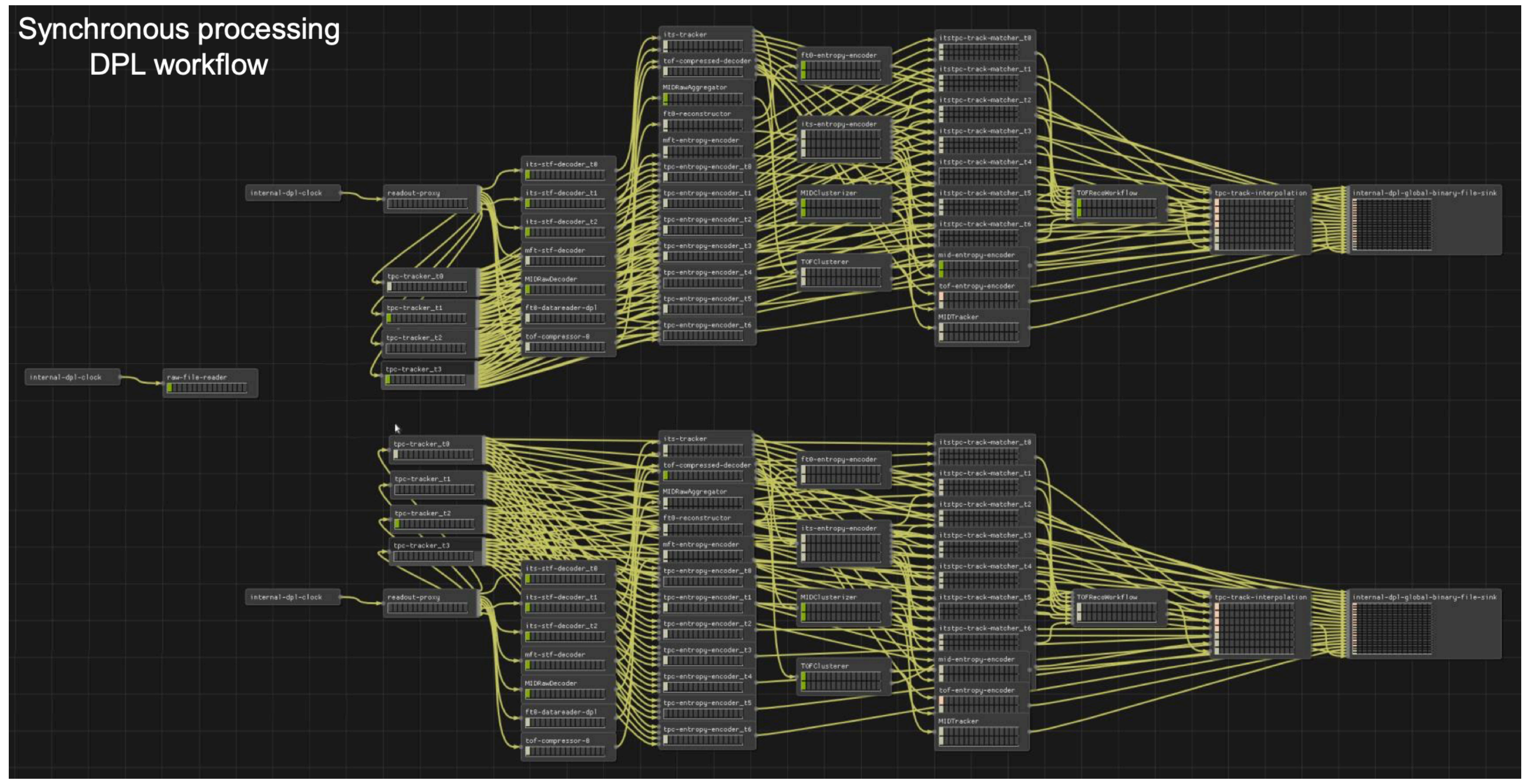
Ref: https://indico.jlab.org/event/459/contributions/12432/

ALICE Results

- GPUs capable of replacing 100s of CPU cores
  - Target the hot spots in the code
- Scheduling to balance the use of the CPU and the GPUs is *delicate*
  - Smooth time frame publication rate to maximise resource usage
  - Optimise time frame size to avoid over subscribing memory
  - Use NUMA domains to avoid memory bottlenecks and maximise throughput
- New Data Processing Layer allows complex workflows to be managed - O^2 system shared with FAIR
- Many people can contribute to algorithms, but expert GPU knowledge needed



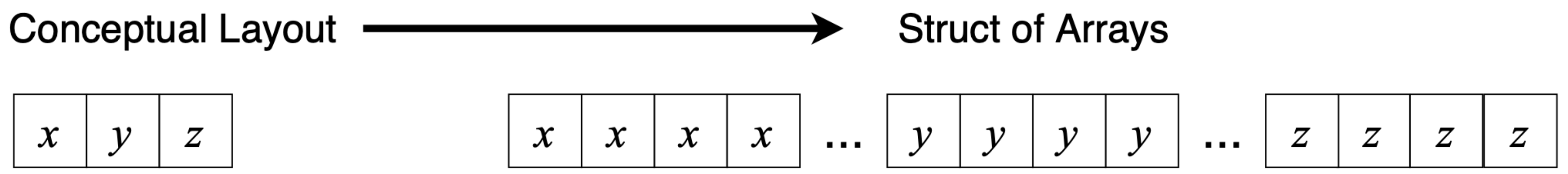
Configuration (2022 pp, 650 kHz)	Time per TF (1 instance)	Time per TF (full server)
CPU 8 core	76.91s	4.81s
CPU 16 core	34.18s	4.27s
1 GPU + 16 CPU cores	14.60s	1.83s
1 NUMA domain (4 GPUs + 64 cores)	3.5s	1.70s

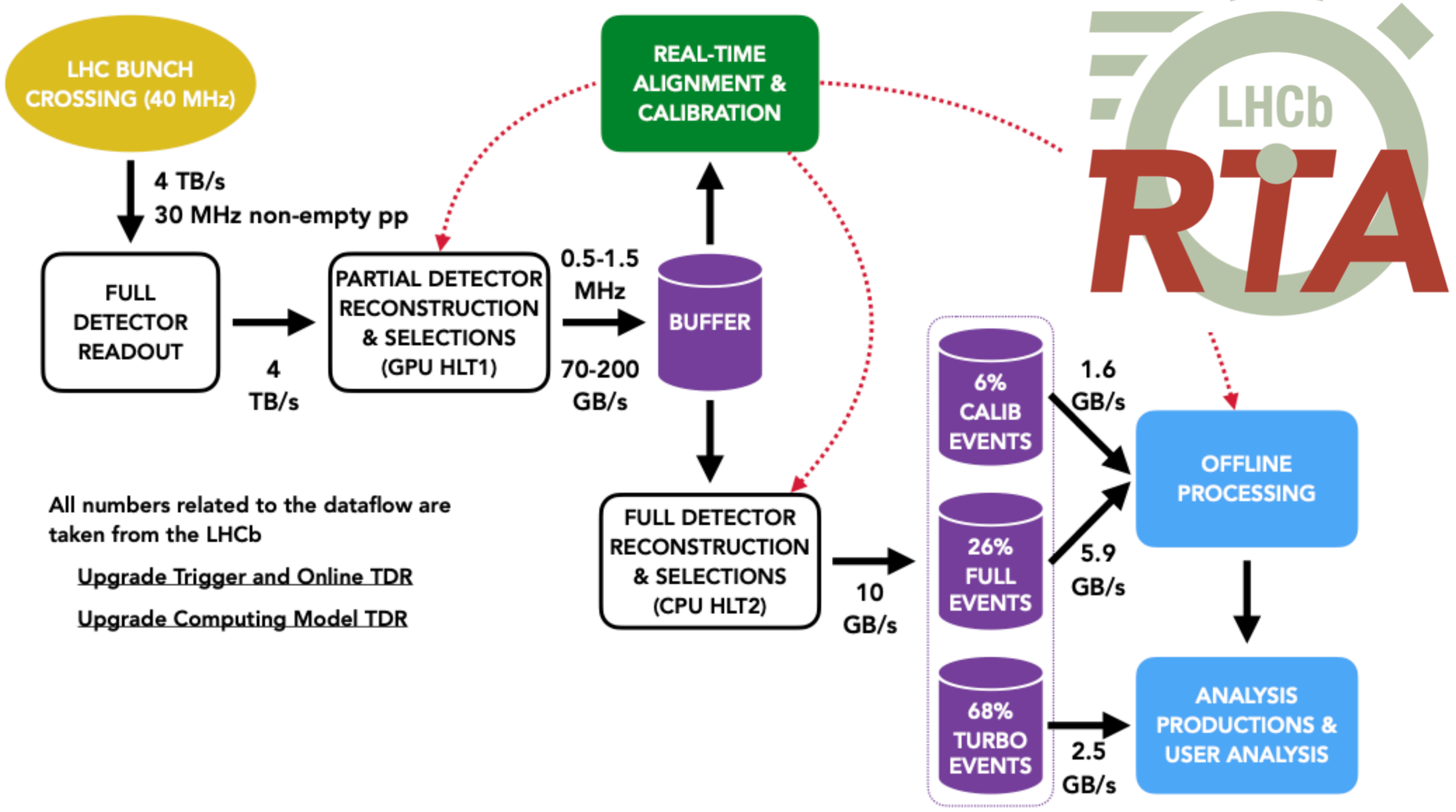


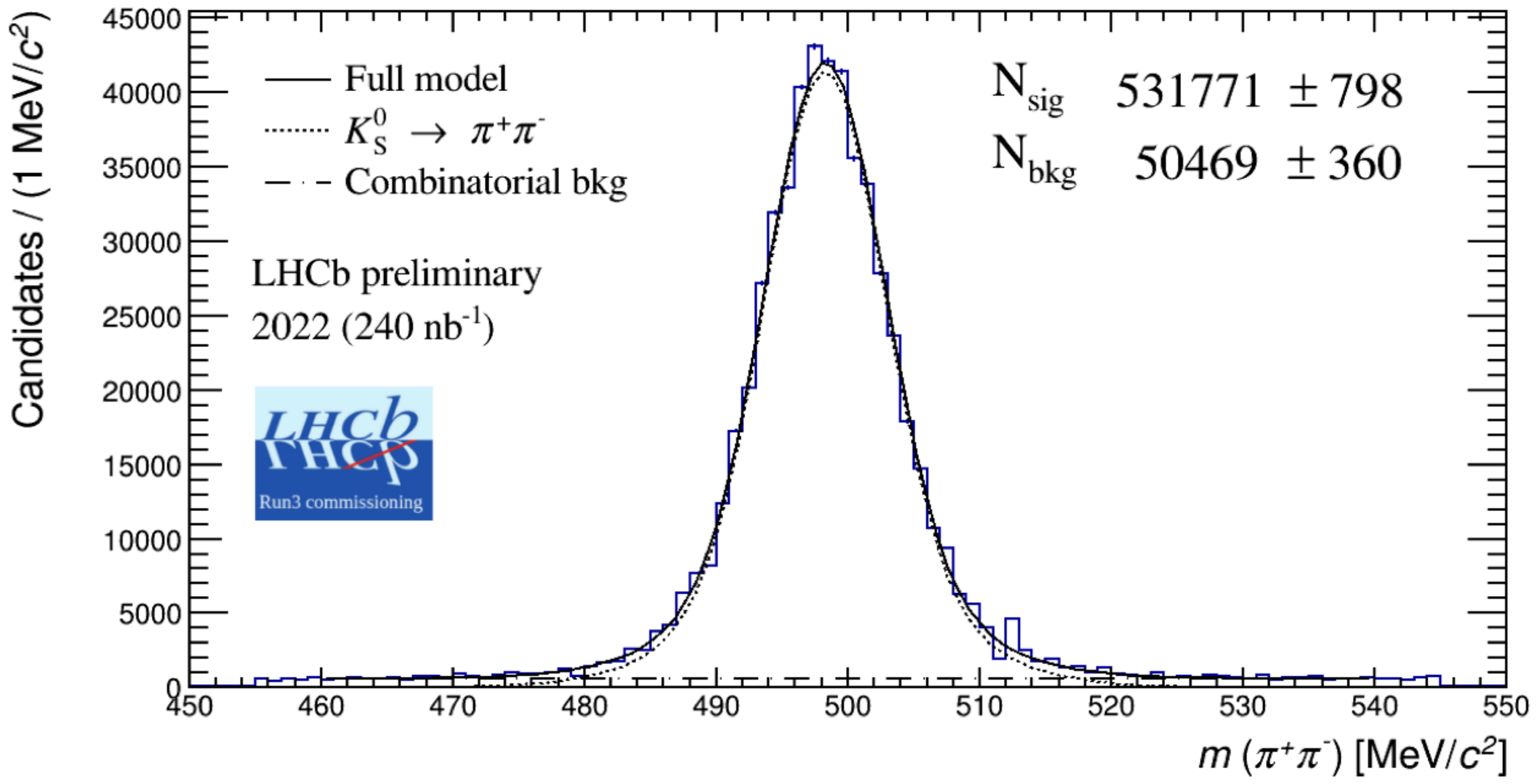
Real Time Analysis

- LHCb processing HLT data on GPUs at 30MHz
    - Cannot store full events at this rate
- Reduce events *in the HLT* to analysis level outputs
    - Only keep what you need!
- Requires fast calibration loops to ensure full offline quality in the HLT
    - No RAW data to go back to
- **Optimise data layout** for processing using Structure of Arrays
    - Profits from CPU SIMD instructions
    - Hide this from the end user!

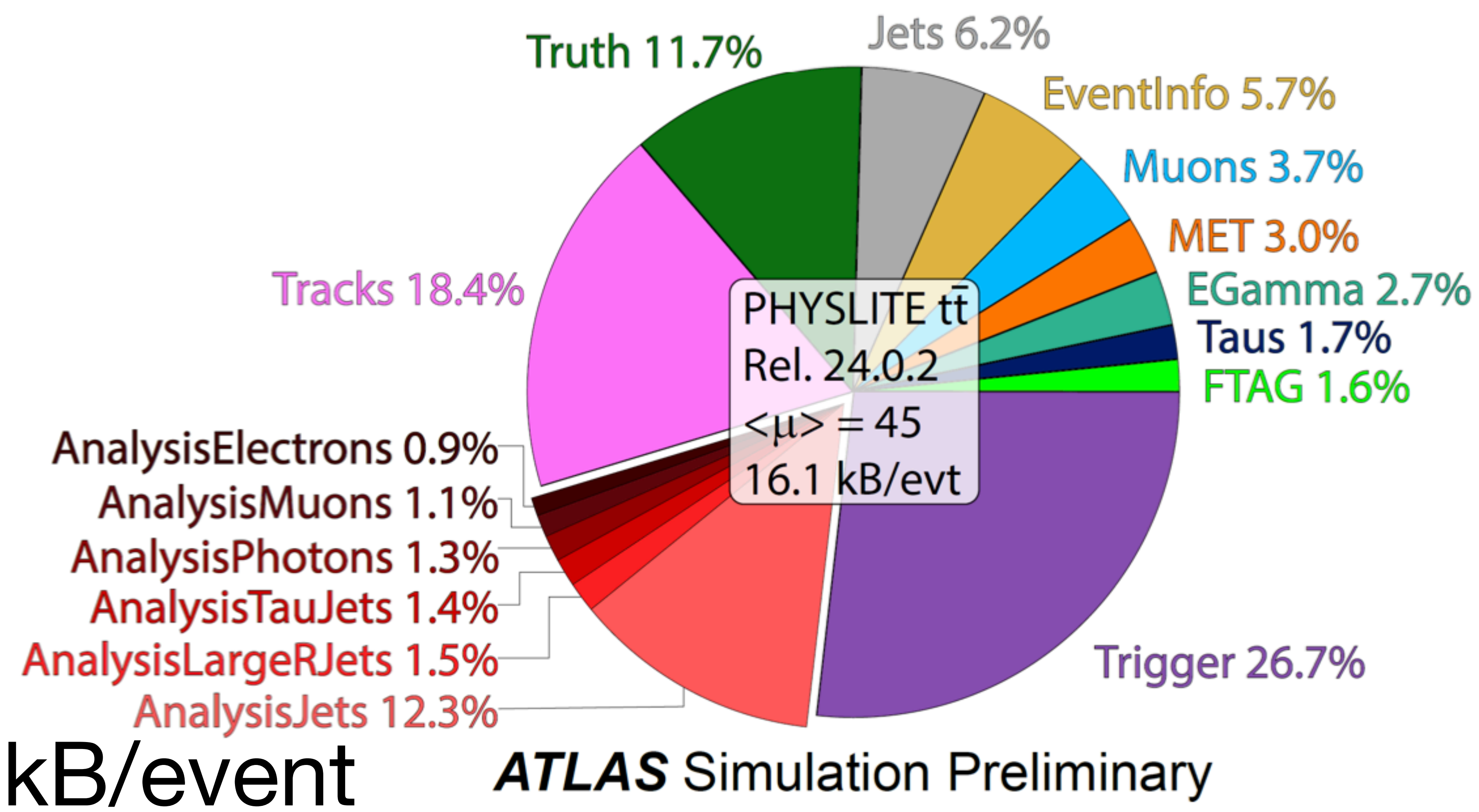
SOA : Struct of Arrays - well suited for SIMD approach







Small is Beautiful (and useful!)

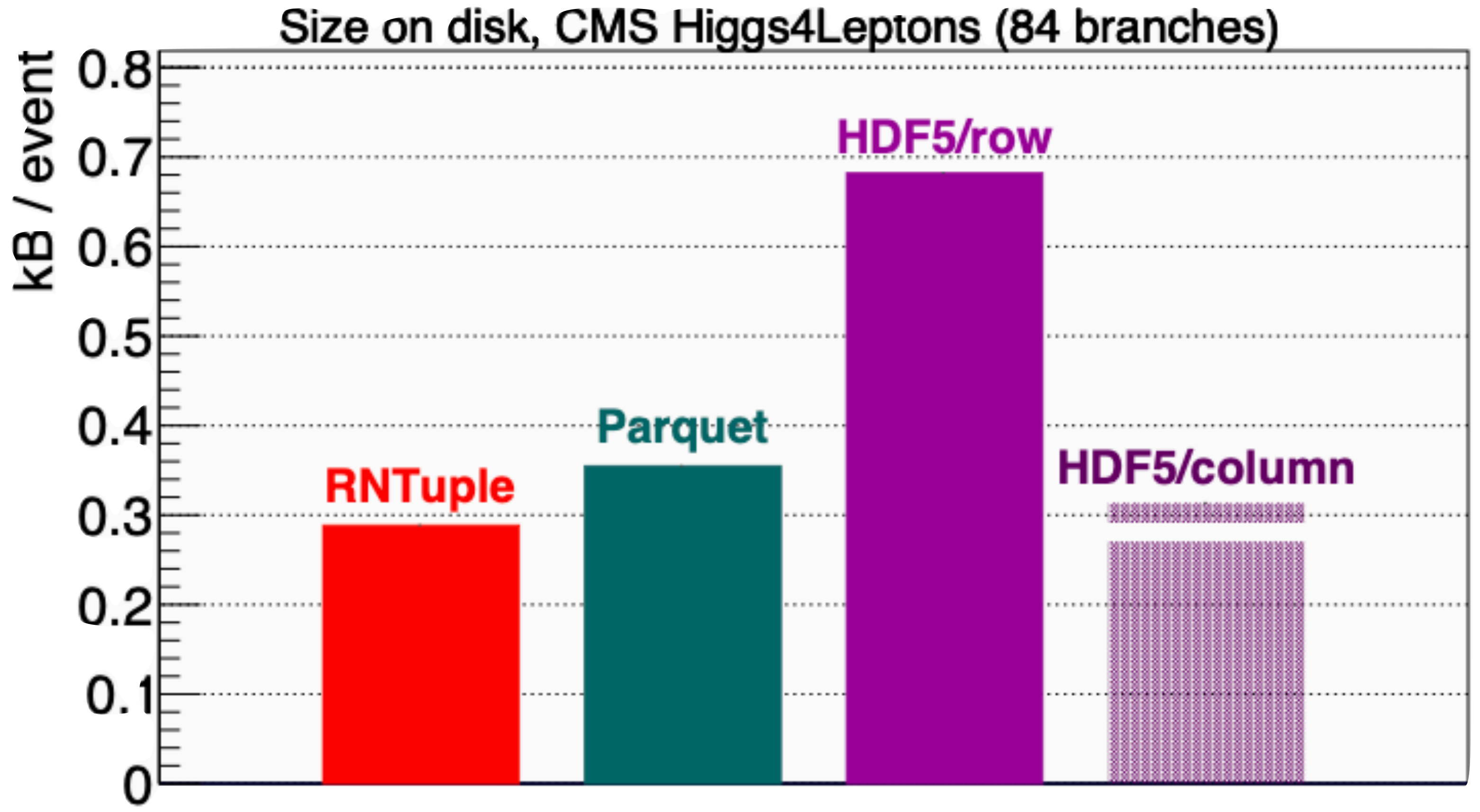


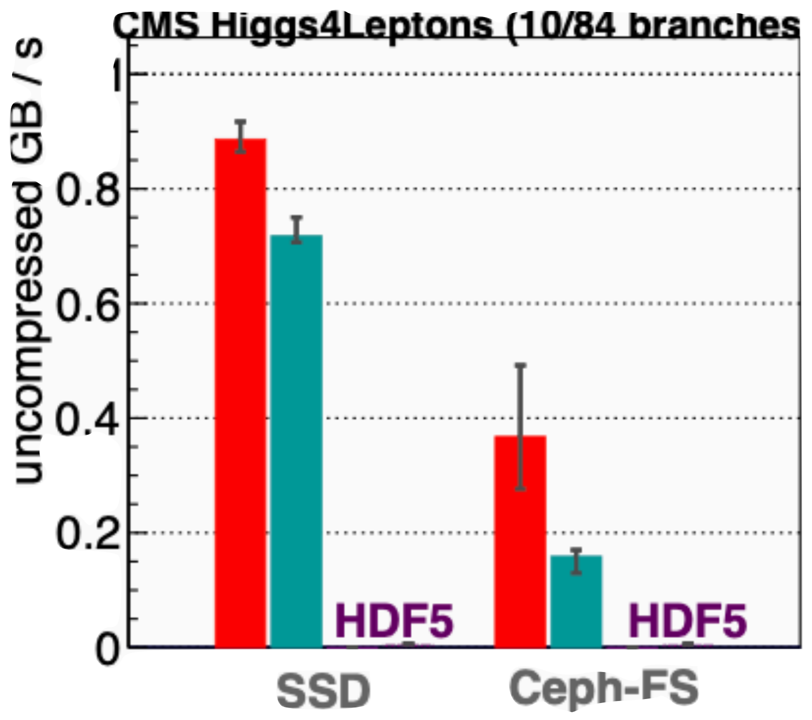
- Big detectors = big data!
  - e.g., Analysis Object Data from ATLAS/CMS is 300-500kB/event
  - At 10kHz trigger rate this can't even fit on disk anymore!
- Need to aggressively reduce data volumes to manageable levels
  - CMS pioneered the use of ultra-small tuple formats with NanoAOD
  - New ATLAS data format DAOD_PHYSLITE that is pre-calibrated and suitable for around 80% of analysis use cases
  - Target 10-12kB average per event
    - x40 reduction from initial AOD
- Smaller events mean more physics per megabyte
  - This means faster results! Incentive to adopt reduced formats

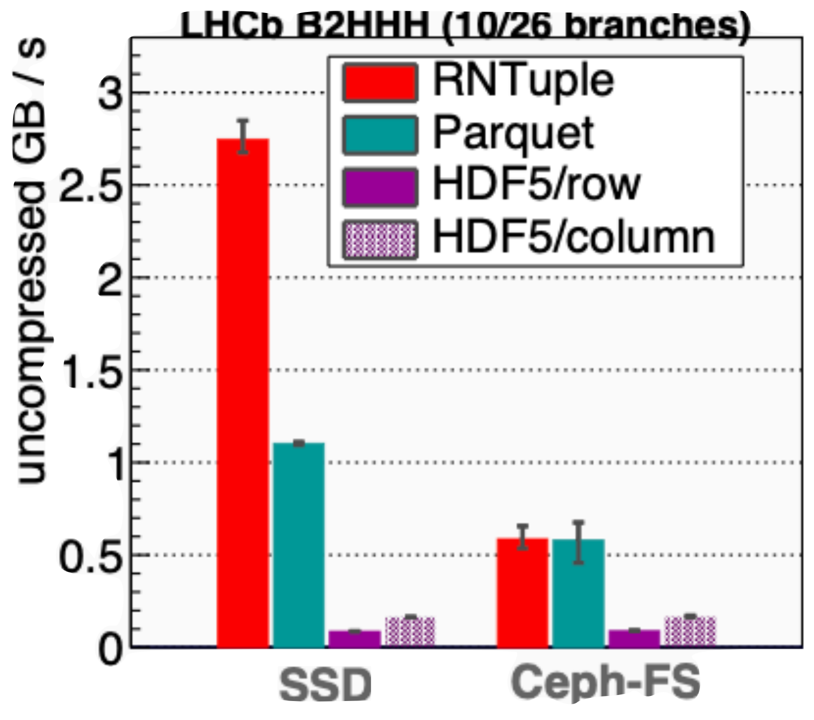
Ref: https://indico.jlab.org/event/459/contributions/11586/

ROOT RNTuple

- HEP analysis level data is quite complex
    - Nested, inter-dependent collections of variable size
- We also need to keep our data preserved for multi-decade timescales
- Data volumes are prodigious - and therefore expensive
- RNTuple is a new modern I/O system in ROOT
    - **Smaller** size on disk than old TTree or any industry alternative
    - **Faster** read speeds from local SSDs or from network file system
    - Also can be stored in modern object stores (DAOS)
    - More robust API with proper error handling







Ref: https://indico.cern.ch/event/1294815/

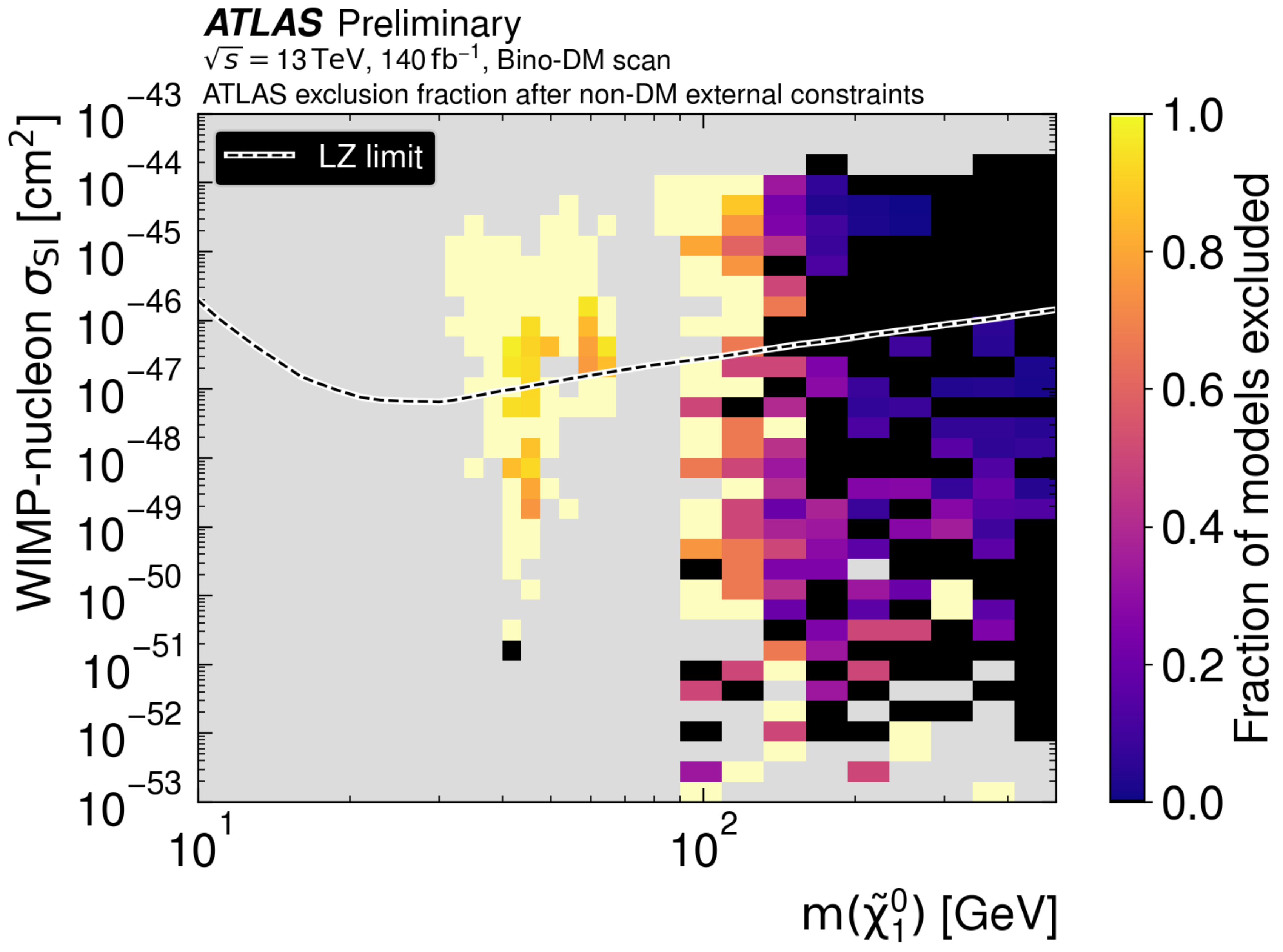
Analysis - A Grand Challenge

- Large scale analysis sits at the interface between software and computing
  - Generally an I/O bound problem (cf. simulation, reconstruction, which are not)
- A lot of interest in Analysis Facilities, as a possible evolution of our computing facilities
  - High speed access to large amounts of data, with fast turnaround, reliable and possibly interactive access
- Test analysis (t-tbar cross section with CMS open data) written in different frameworks: Coffea, ROOT RDataFrame, Julia
  - Declarative style often used, no event loop:

```
electron_reqs = (elecs.pt > 30) & (np.abs(elecs.eta) < 2.1) & (elecs.cutBased == 4) & (elecs.sip3
muon_reqs = ((muons.pt > 30) & (np.abs(muons.eta) < 2.1) & (muons.tightId) & (muons.sip3d < 4) &
             (muons.pfRelIso04_all < 0.15))
jet_reqs = (jets.pt > 30) & (np.abs(jets.eta) < 2.4) & (jets.isTightLeptonVeto)
```

Analysis: From Fitting to Preservation

- Fitting can be difficult and resource consuming
  - New improved fitters can run on GPUs (RooFit, zfit, GooFit)
- Analysis also needs to be *preserved* and *FAIR*
  - To work well this needs to be built into the workflow from the start
  - When this is done it really pays off
    - ATLAS analysis of 25k SUSY models in one go
- HEP Statistics Serialisation Standard (HS3) helps bridge a gap here
  - Inspired by reusable HistFactory models from pyhf
  - Uses JSON and extended to the full RooFit feature set



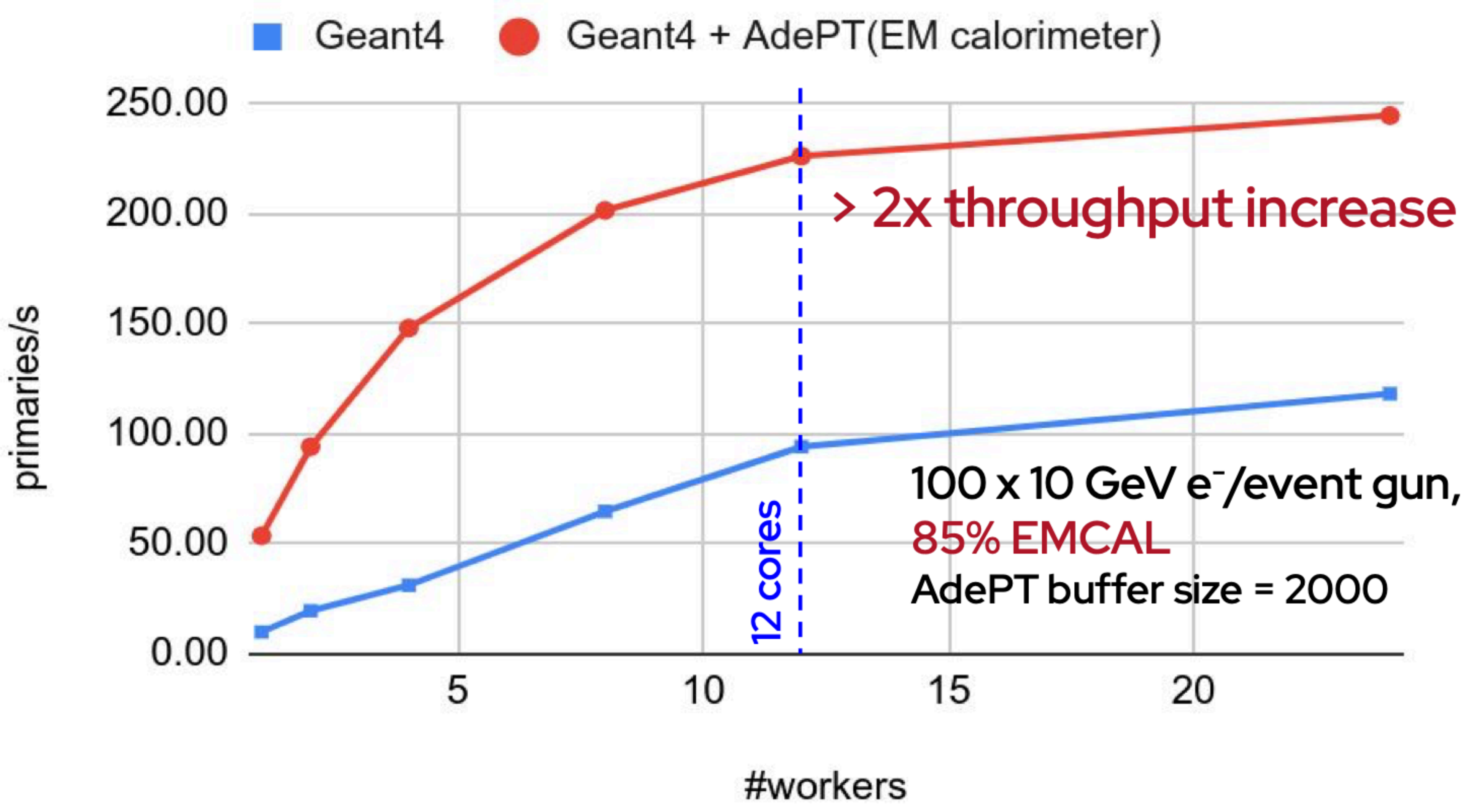
HS3 pyhf ROOT Data Analysis Framework BAT.jl Bayesian Analysis Toolkit zfit recast reana

Horizons in Simulation

- Simulation has been a very expensive piece of LHC computing
  - Even small gains in Geant4 speeds for the LHC experiments can give significant savings
  - Some significant speed improvements in recent releases for Geant4
    - G4HepEM for combined electromagnetic processes
    - Woodcock tracking for segmented calorimeters
- Both ATLAS and CMS have made significant improvements in overall simulation time

	Physics List	Tracking Manager	difference
G4NativeEm	473 s	405 s	-14.4 %
G4HepEm	414 s	337 s	-18.6 %
difference	-12.5 %	-16.8 %	**-28.7 %**

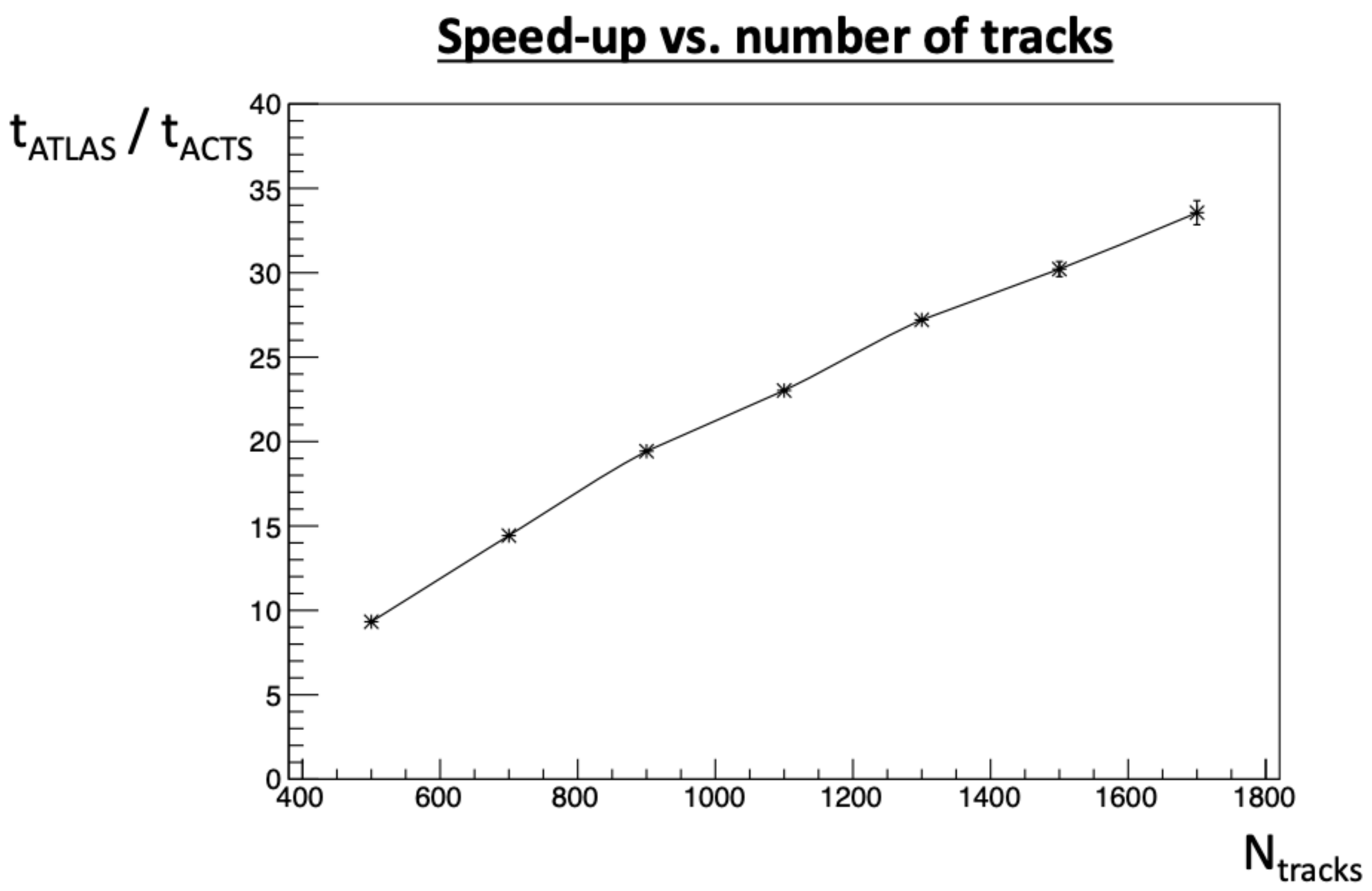
		"Normal way"	Gamma-general	Woodcock(+GG)
E_{dep} [MeV]	Pb	7726.3	7725.9	7735.4
	lAr	2145.6	2145.9	2145.6
#secondary	γ	5215.7	5216.2	5215.4
	e^-	8963.3	8931.2	8928.5
	e^+	538.5	538.3	538.3
#steps	charged	36548.4	36522	36860.5
	neutral	**36963.4**	**36952.7**	**9546.8**
Rel. perf. gain		0	~ 5 [%]	**~15 [%]**



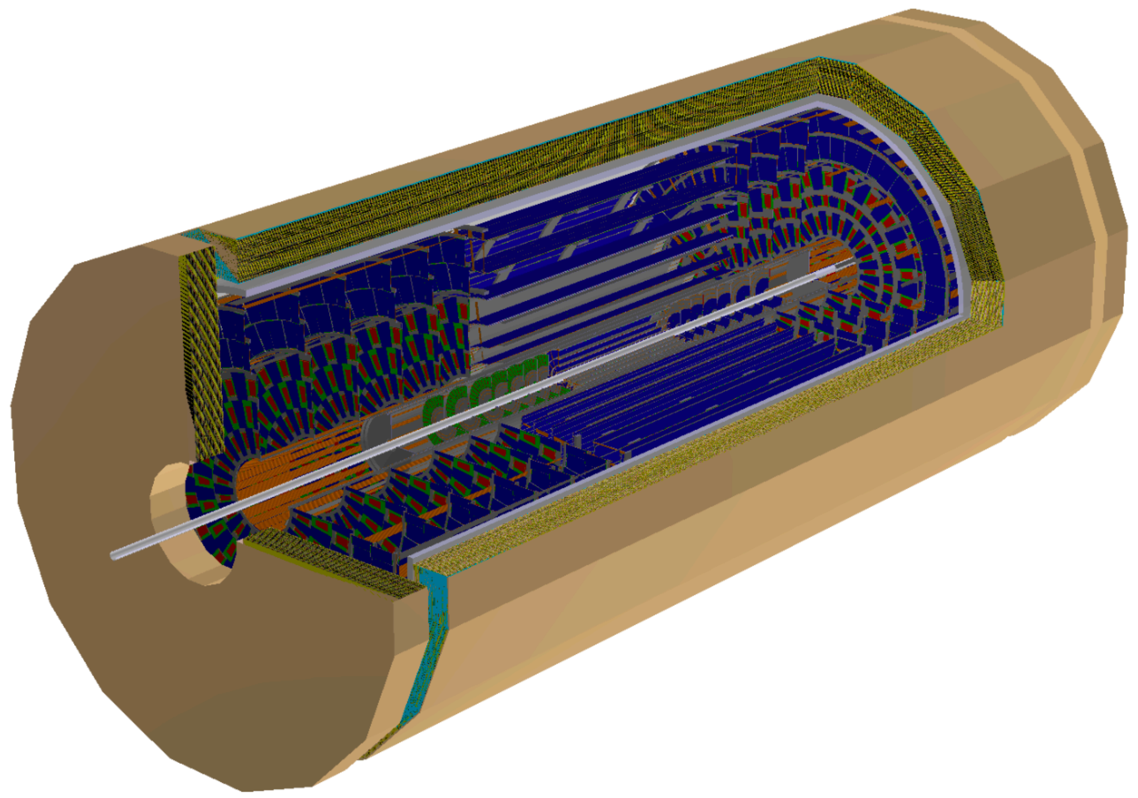
- More ambitiously, can we use GPU devices for simulation?
  - Two R&D projects have been investigating this Celeritas and AdePT
- It's not easy - divergence is something of a killer for GPUs, inherent to the problem
  - Encouraging results in running a hybrid workflow, EM physics on GPU for calorimeter simulations
    - 4x performance per Watt for EM test, x1.8 speedup for ATLAS test application (Celeritas)
    - x2 throughput improvement in simple CMS setup (AdePT)
- Geometry seems to be a significant bottleneck at the moment
  - New R&D to move from volume models to surface models, which should improve GPU performance

Working Together - Acts

- Long tradition in HEP of the community supporting common software projects
  - ROOT (foundation and analysis) and Geant4 (simulation) best known, but also newer software linking to Python data science, Scikit-HEP
- Reconstruction has traditionally been a very experiment specific area of work
- However **A Common Tracking Software** (Acts) is attempting to generalise tracking software to multiple experiments
  - Geometry and Event Data Model are the tricky bits!
- Born at ATLAS, but being used/evaluated by many experiments: sPhenix, Faser, ALICE, LDMX, ePIC, STFC, FCCee/hh
- ACTS provides also a great generic testbed for new algorithms and techniques, e.g., graph neural network tracking
  - Gave birth to the *OpenDataDetector*



Vertex finding speed-ups with ACTS cf. ATLAS Run-2 vertexing (Ref)

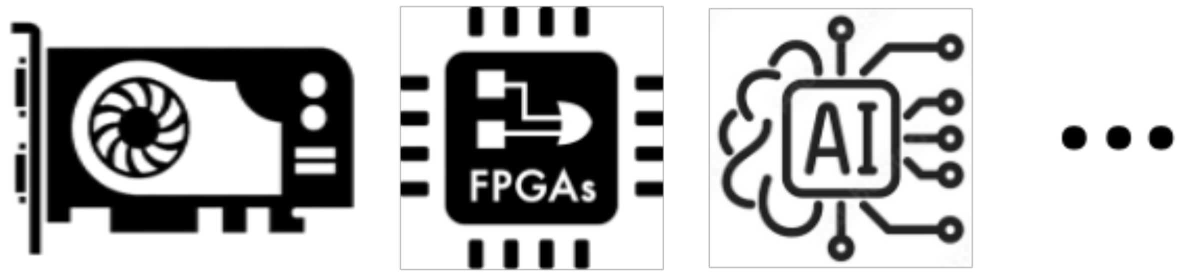


Open Data Detector - experiment neutral realistic detector

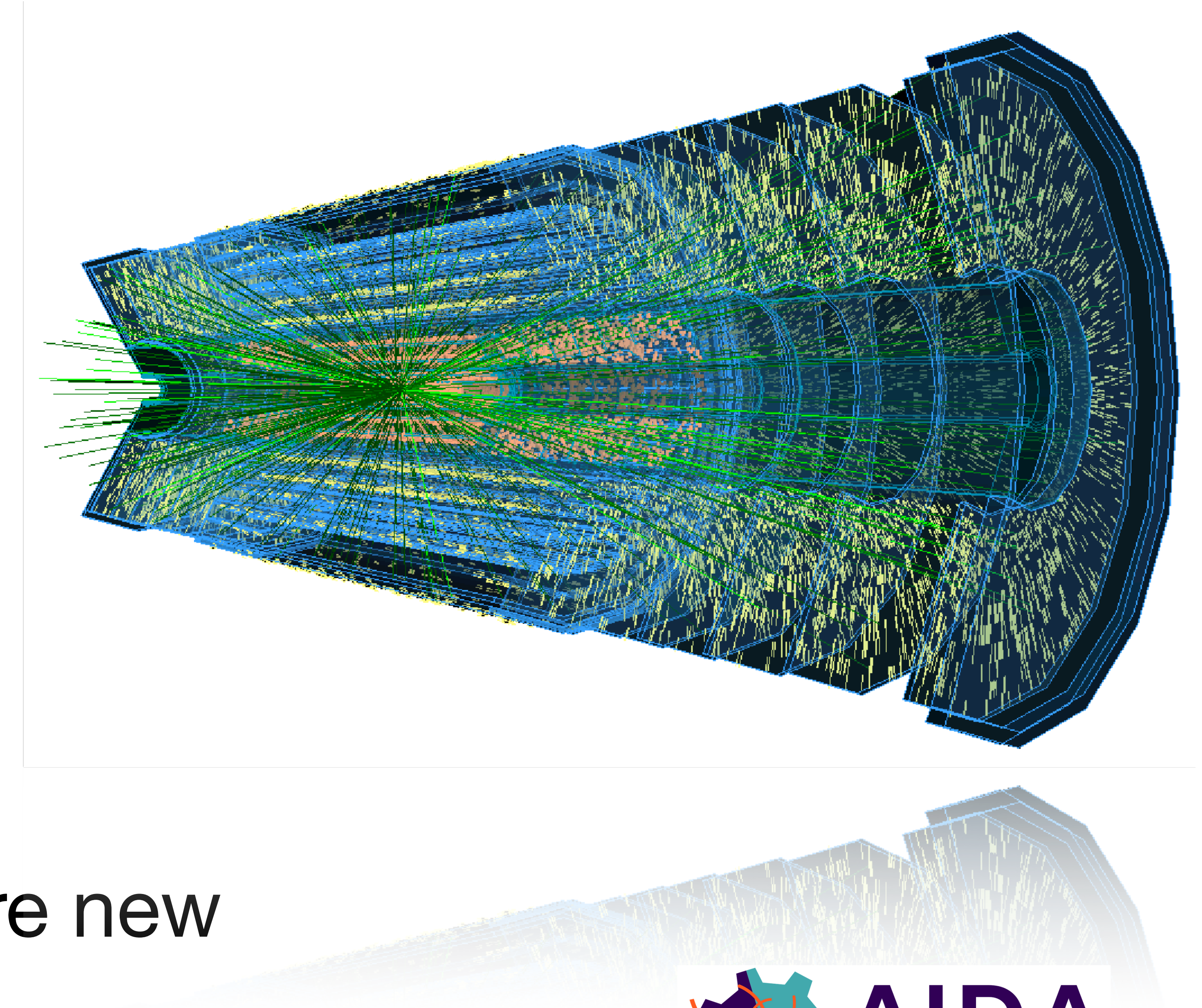
Training, Communities and Summary

Training and Communities

- Good software is mission critical for all large science these days
  - Need to follow industry best practices and standards
  - Significant changes in tooling and techniques
- HSF and IRIS-HEP trying to create a **training community**
  - Gather and develop training materials
  - In-person, hybrid and self-study options
- Many success stories: Software Essentials, C++ Course, Analysis Preservation and CI/CD
  - Excellent addition to in-person schools (CSC, Bertinoro, GridKa, …)
- This extends to **building communities** that support our software in a way that encourages coherent development
  - PyHEP is now rather mature
  - JuliaHEP is just starting



Conclusions



- Exciting physics programs ahead in many areas
  - Software (and computing) is a *mission critical area* for exploitation
- Investment in current software base is high and needs to be ongoing
- Vigorous R&D program is required to investigate new solutions and explore new avenues
  - IRIS-HEP, Swift-HEP, HEP-CCE and CERN EP R&D are all important examples
  - HSF provides a forum to exchange ideas, discuss and foster communities
- The use of ***compute accelerators*** and ***heterogeneous platforms*** enables us to keep up with high data rates and maximise physics output
  - Optimal data handling is vital to achieve necessary throughput
  - Development of new software skills is required
    - Plus we need viable career paths for our experts!
- But need to keep software *simple and accessible* for Physicists
- ***Machine learning and AI*** are having a increasing impact - Lukas' talk is next!