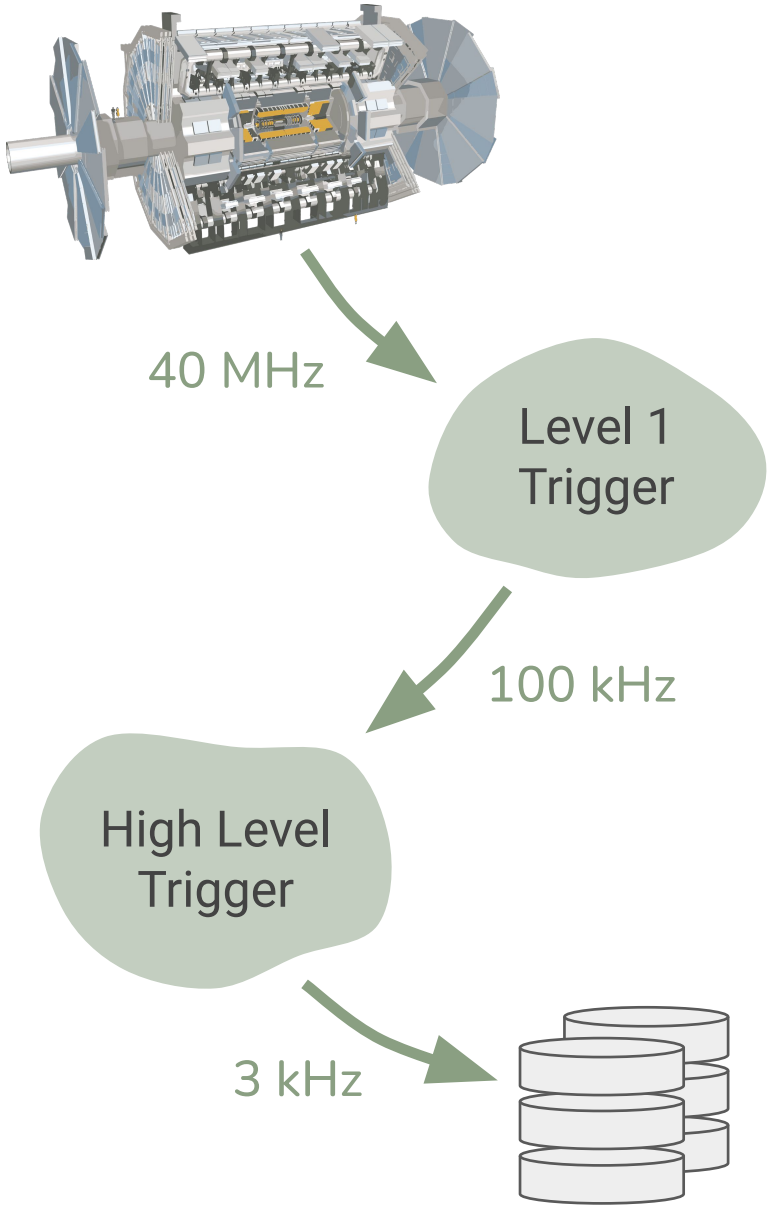
# *The ATLAS Trigger system: the current and future HLT*

Francesco Giuli (on behalf of the ATLAS Collaboration)


CEPC workshop
Nanjing (China)
26/10/2023

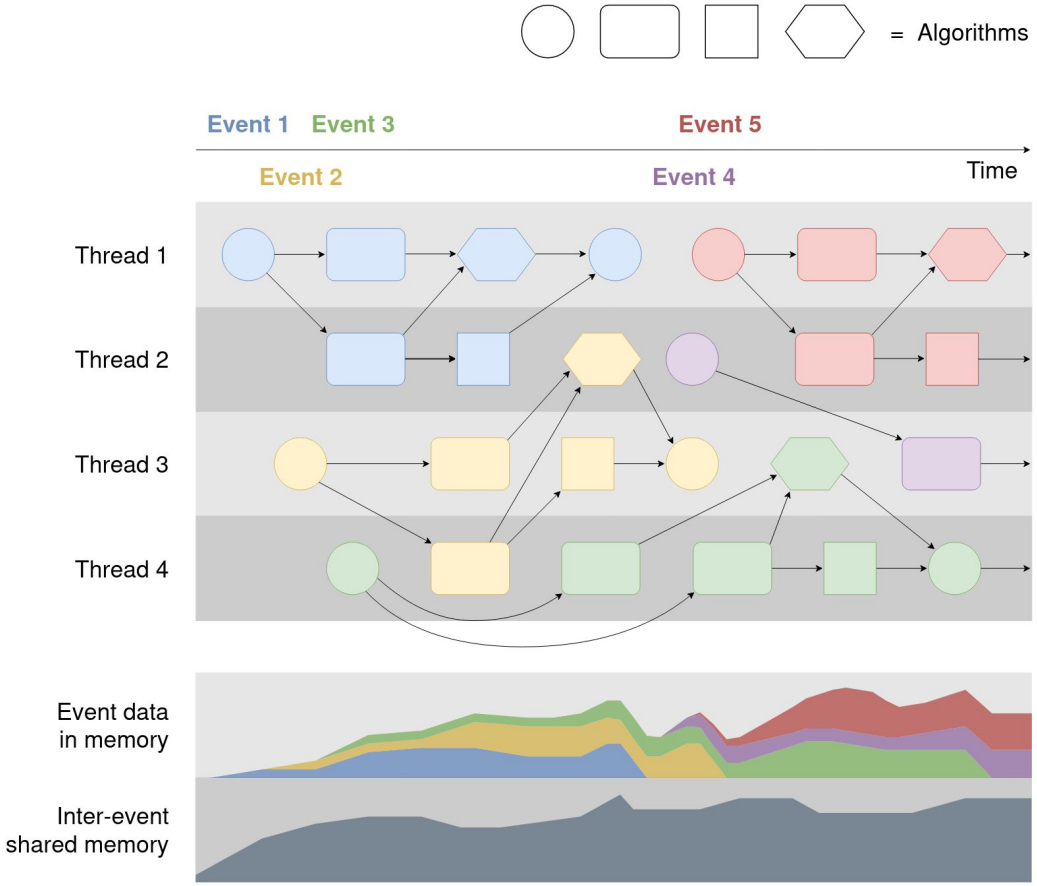
# Introduction

- The **ATLAS High Level Trigger (HLT)** selects events based on
  - Detector readout
  - Level-1 Trigger decision, applying a coarse selection
- Event selection is achieved by a set of selection **chains**
- **Early algorithms** within a chain reject as many events as possible
  - More CPU-intensive algorithms are executed only on a small subset of events
- The data is processed on a computing farm with **~60k real CPU cores** (2023)



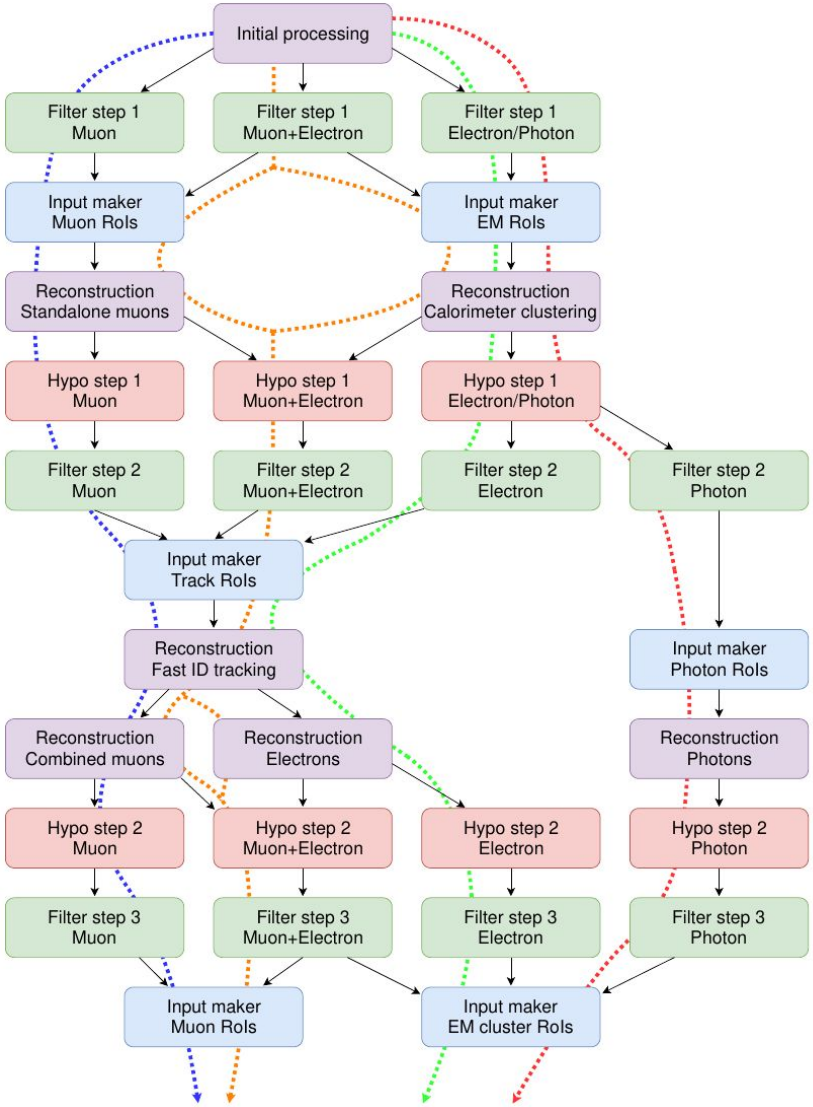
# Run 3 Multi-Threaded HLT

- The HLT was redesigned to share the same code with offline reconstruction
  - Support the Multi-Threaded mode
  - Reduce the memory footprint of the code (not an issue for online operation)
- The upgrade benefits:
  - Simplified maintenance of the code
  - General performance improvements
  - Integration of computing accelerators for future running periods
- More details can be found in ATL-DAQ-PROC-2019-004



# Run 3 HLT configuration

- The Run 3 HLT Control Flow is generated based on a list of algorithms organised in **steps**, performing reconstruction and selection
- The steps are combined in **chains** and organised in a selection **menu**
- The configuration is stored in JSON format and can be provided transparently to HLT in different ways:
  - From a database
  - From a file
  - From a configuration in Python
  - From 'in-file meta-data' (mostly used for offline reconstruction)




ATL-DAQ-PROC-2019-004

# Online performance

- The configuration of the HLT Processing Unit and its CPU resource utilisation is defined by 3 parameters
- Number of process **forks**
- Number of **threads** within the process
- Number of event **slots** defining how many events can be executed in parallel per node

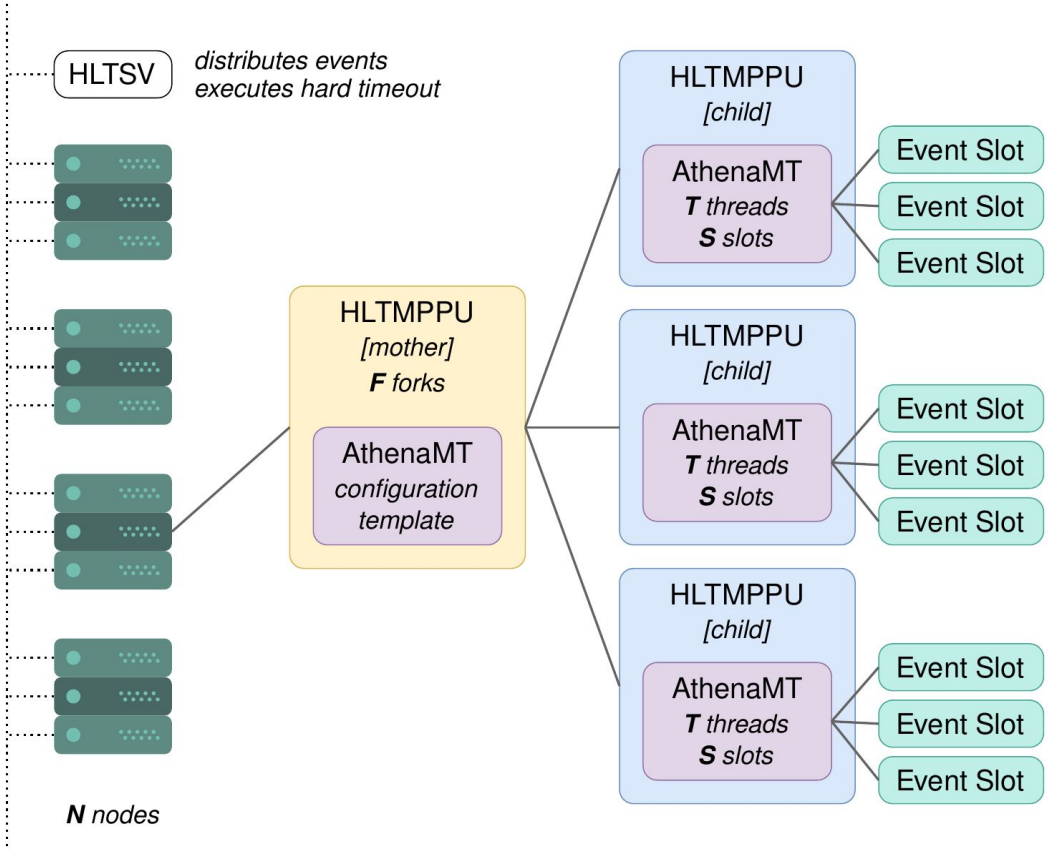
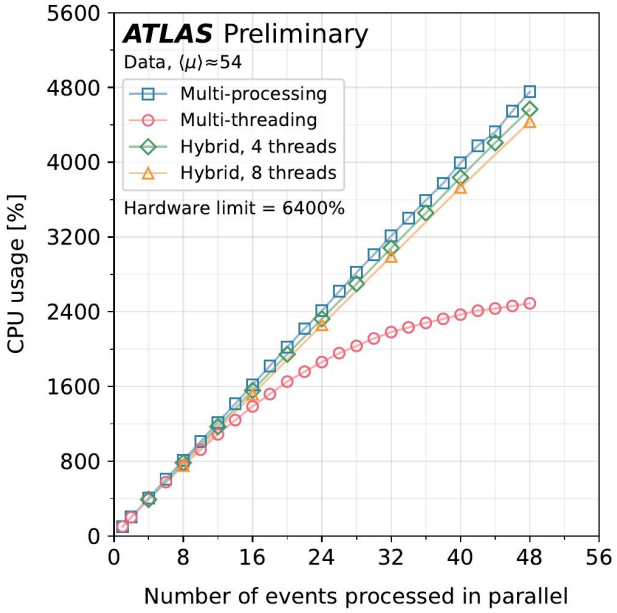


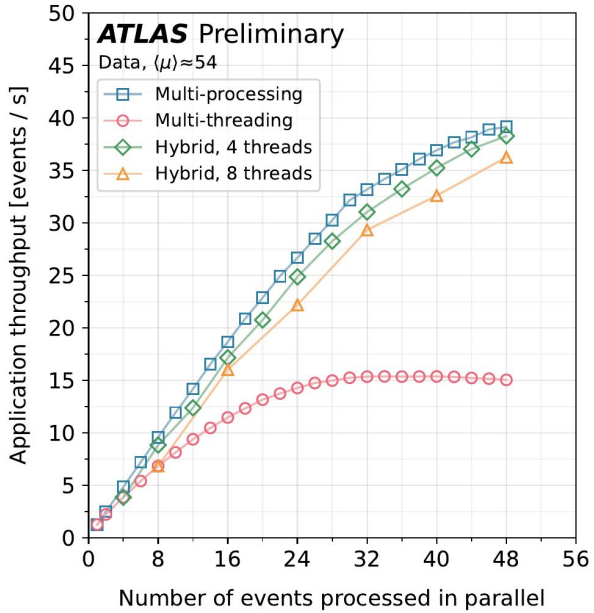

Figure from TriggerCoreSWPublicResults

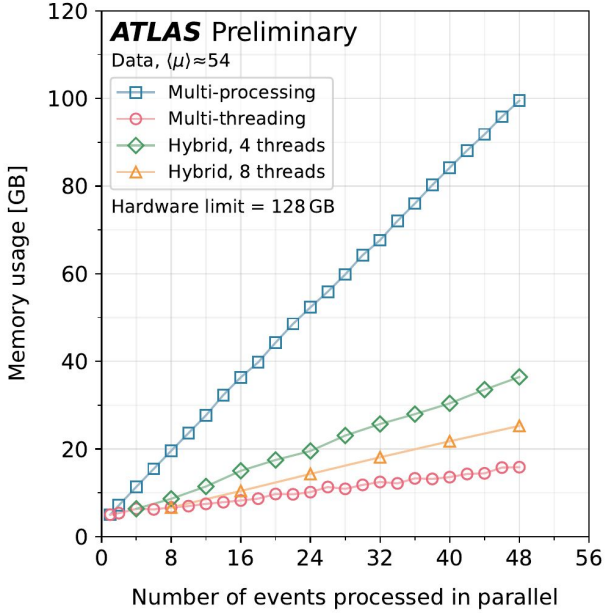
# Online performance

- In 2022 the online event processing was maximized with a **pure Multi-Processing** configuration
- A **pure Multi-Threaded** configuration shows lower throughput
  - It is still used for MC production, where memory savings are necessary
- **Hybrid configurations** also considered, giving similar gains in memory usage without throughput penalty

Figure from TriggerCoreSWPublicResults



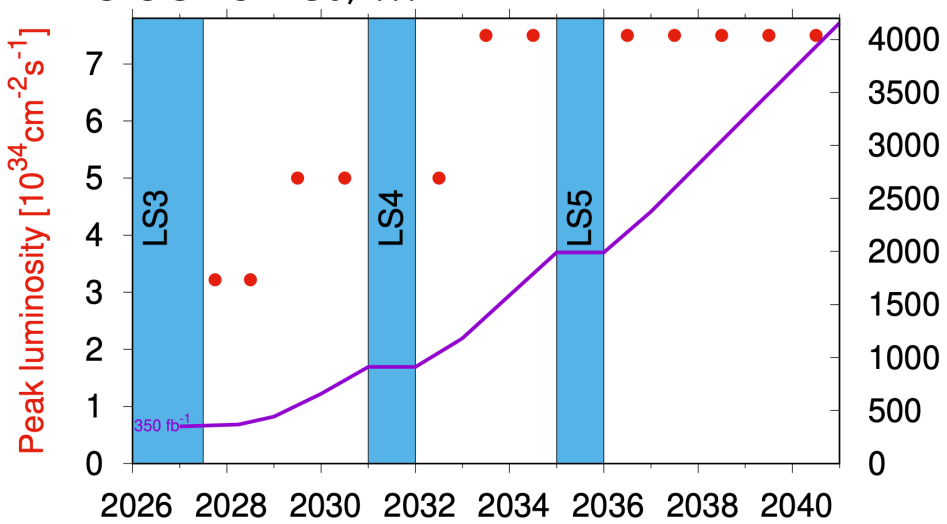



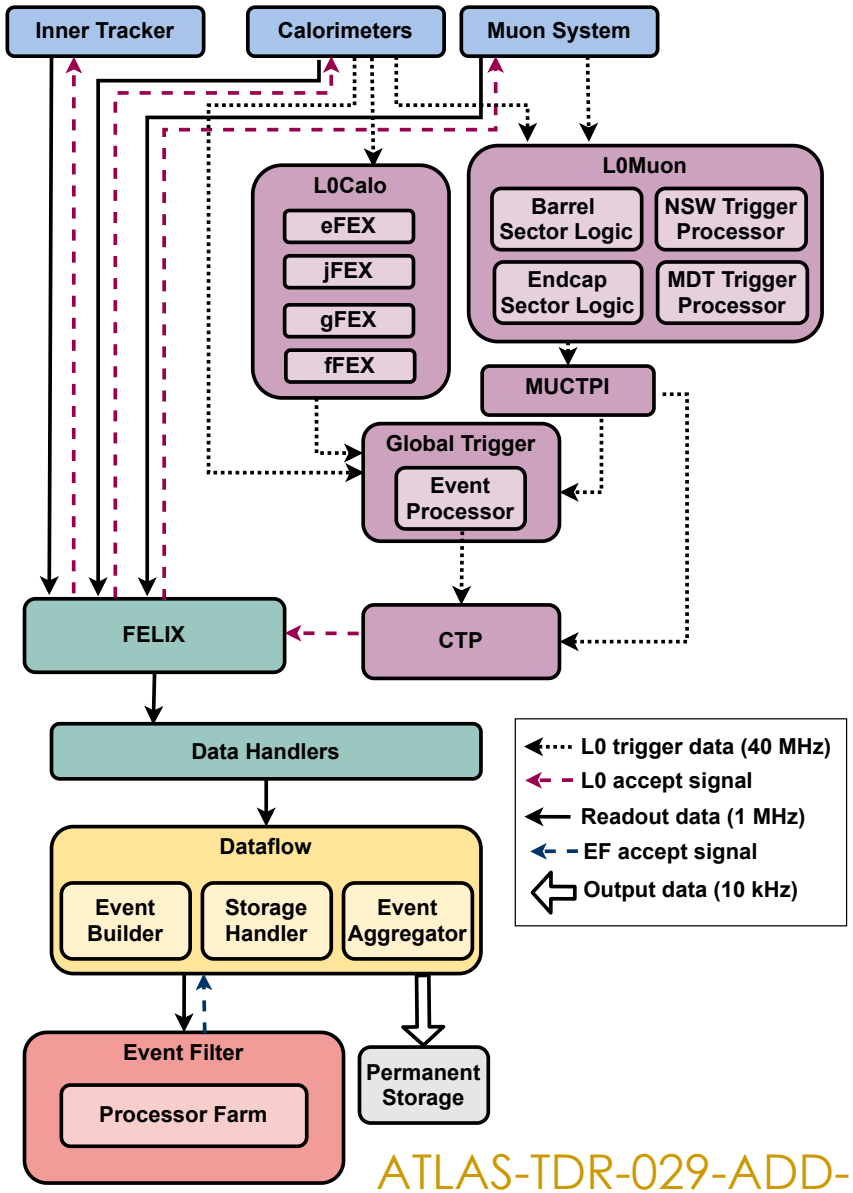
# ATLAS HL-HLC Trigger System

- Trigger decisions much more challenging at HL-LHC
  - **Luminosity: 2 → 7.5 · $10^{34}$ $cm^{-2}$ $s^{-1}$**
  - **Pile-up: 60 → 200**
  - **From 100kHz (2kHz) to 1MHz (10 kHz) for L1 (HLT)**

- **ATLAS detector upgrade**

- New Tracker, new Timing Detector, additional muon chambers, new Tile electronics, …

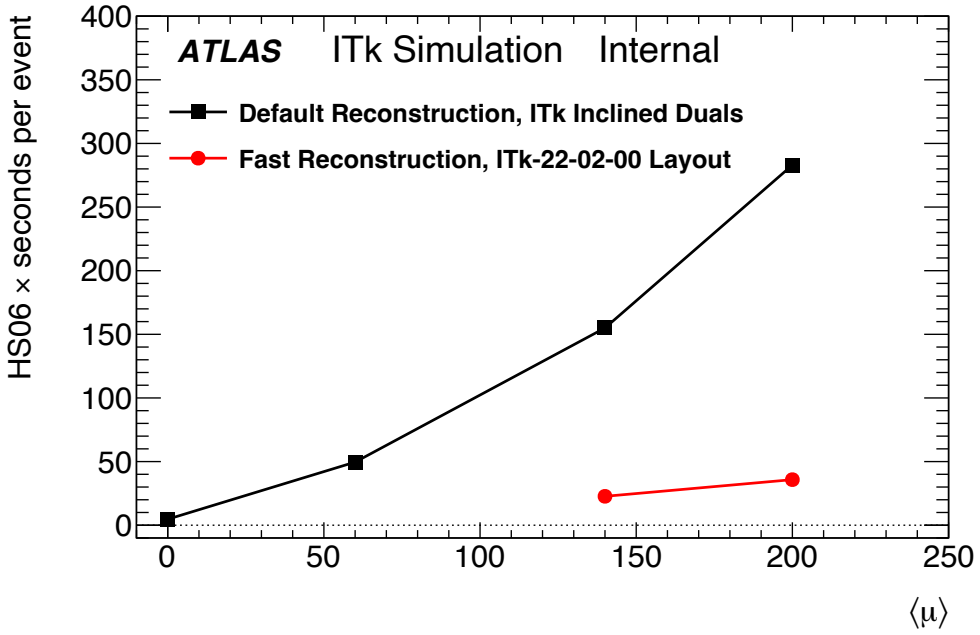


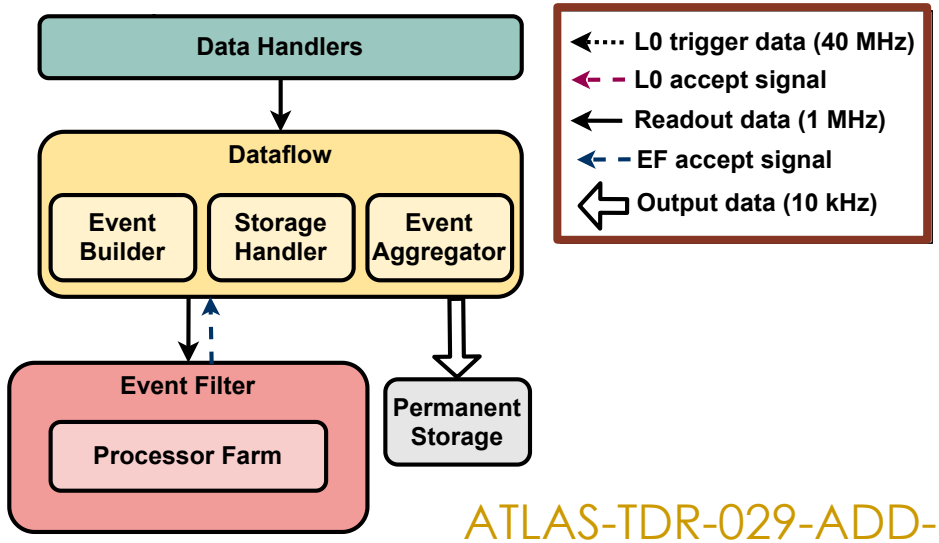




ATLAS-TDR-029-ADD-1

# Event Filter

- Commodity hardware:
  - CPU (7.8/11.4 MHS06 for Run4/5)
  - Possibility w/ accelerators: GPU, FPGA
- Preliminary feasibility studies
  - **CPU showed x8 speed-up**
  - **Use of GPU/FPGA looked promising**
- First demonstrators started
  - Tracking, calorimeters, muons
- Better software, new algorithms and commercial accelerators to handle the computer challenges
- Technology decision about the use of accelerators in 2025

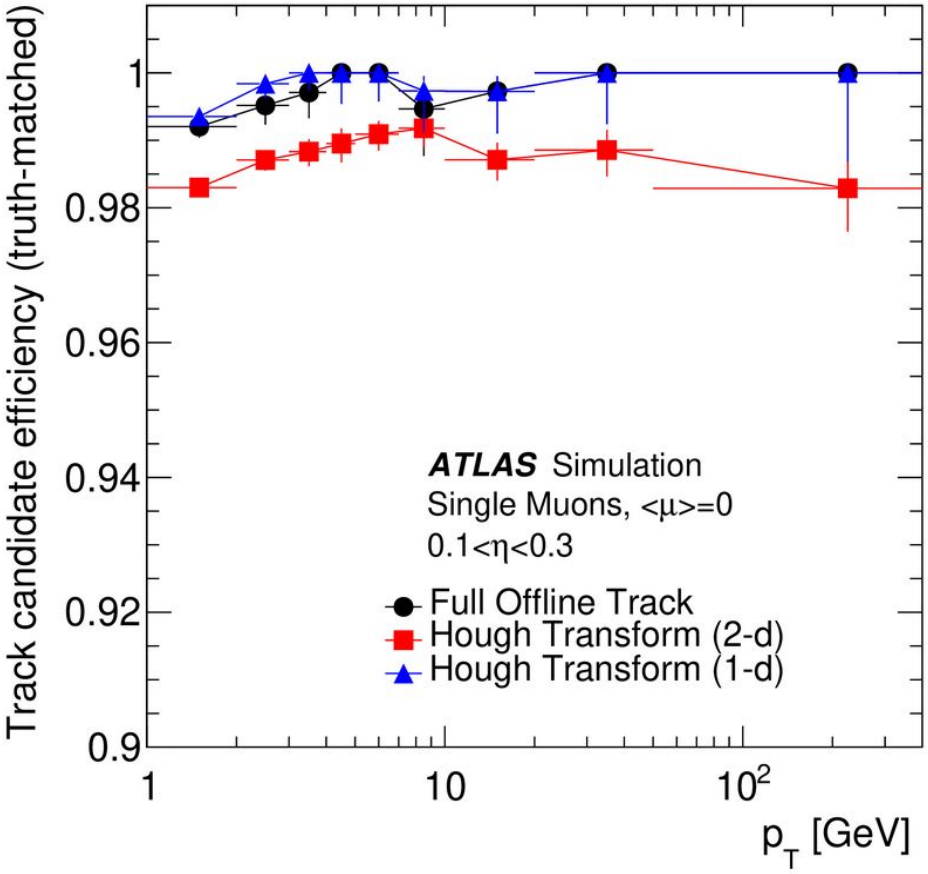



ATLAS-TDR-029-ADD-1

# EF tracking

- **Many many ongoing FPGA and GPU studies**
- Track seeding, pattern recognition, track fitting, ambiguity removal, …
- Exploring use of High Level Synthesis
- Hough transforms with FPGAs and Graph Neural Network
- Plan to use Acts Common Tracking Software
- Experiment independent toolkit for track reconstruction
- **Support for accelerators and heterogenous options**

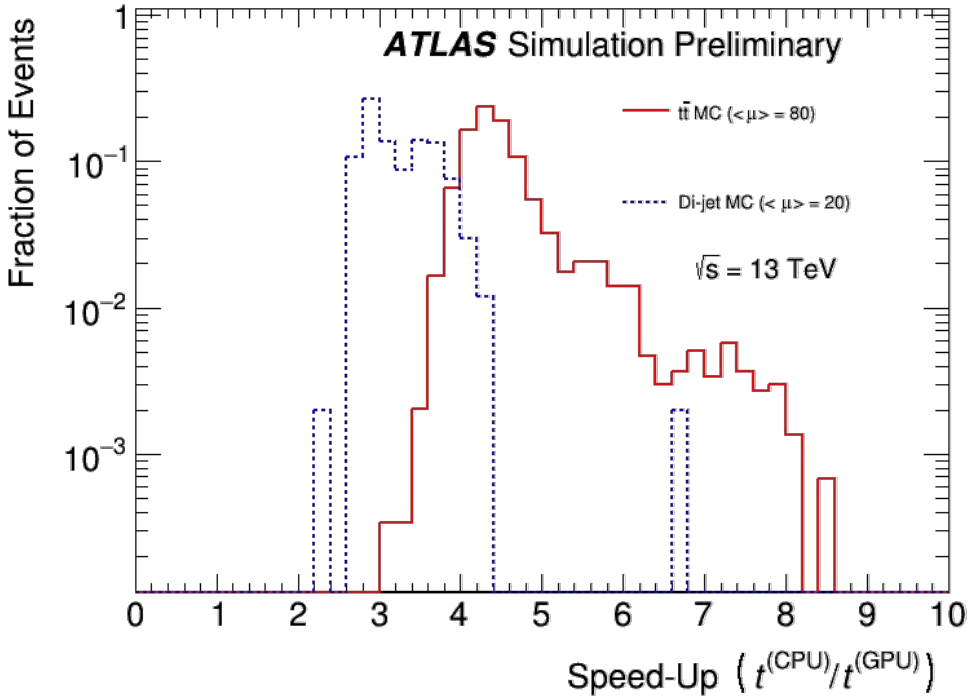


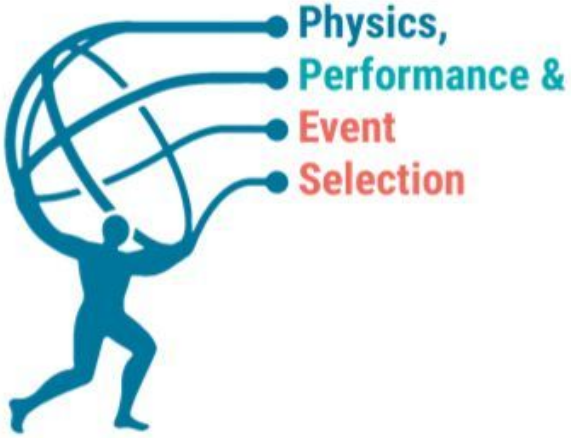

EF Tracking

a:ts

# EF Calo and PPES

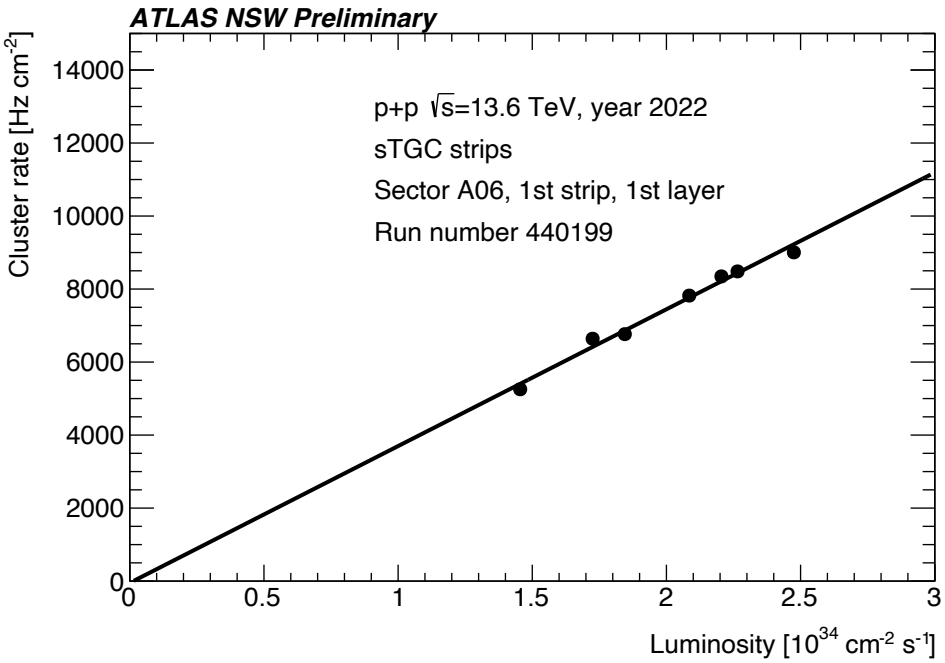
- EF Calo: demonstrated topological Cell Clustering with GPUs
- **Speed-up wrt CPU**
  - ~3.5 for di-jets at $< \mu > \sim 20$
  - ~5.5 for $t\bar{t}$ at $< \mu > \sim 80$
- Exploring FPGAs alternatives
- Physics, Performance & Event Selection group coordinates simulation, performance and trigger menu development for Level-0 and EF algorithms

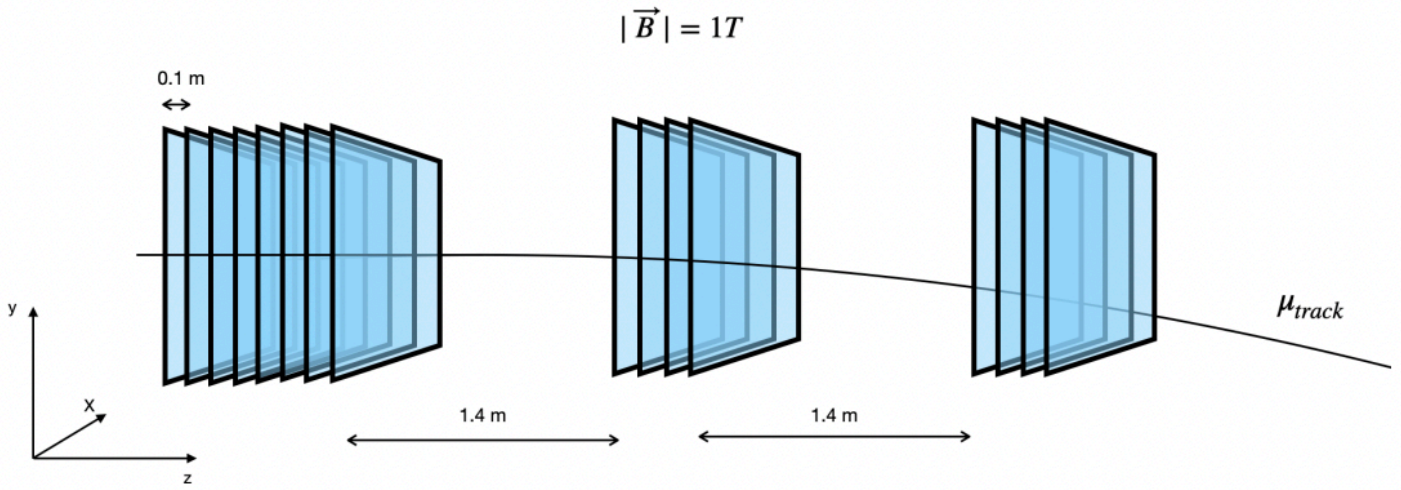


# Muon trigger tracking studies

- Toy model inspired by the New Small Wheel in ATLAS
- **4 samples** produced with different noise rates: 2, 5, 10 and 15 kHz/cm$^2$
- A target is used to emulate effect from correlated background
- Evaluation of clustering and pattern recognition performance on **CPU, GPU and FPGAs**



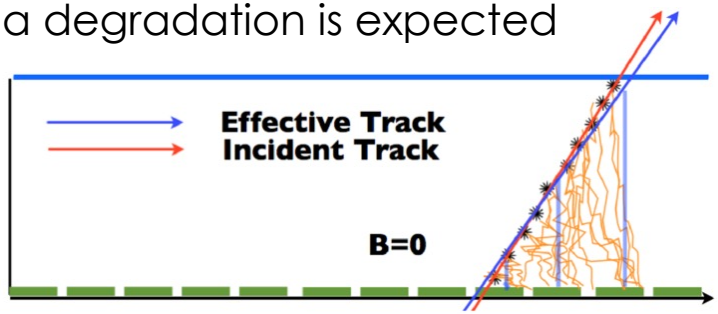



# Muon cluster reconstruction

MC's talk

- A cluster is formed from neighbouring hits and typically the weighted centroid of the cluster is used : $x_{cluster} = \frac{\sum_{strips} q_{strips} \cdot x_{strips}}{q_{strips}}$

- The known challenges with the standard approach are:
  - Depending on the incidence angle of the muon, a degradation is expected
  - "Correlated" background that originates from interactions with material prior to the active layers

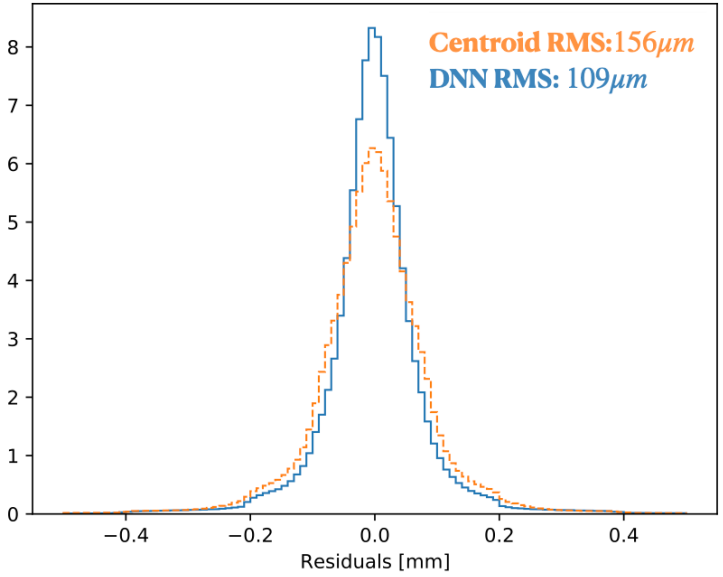



ML is good candidate to improve clustering performance

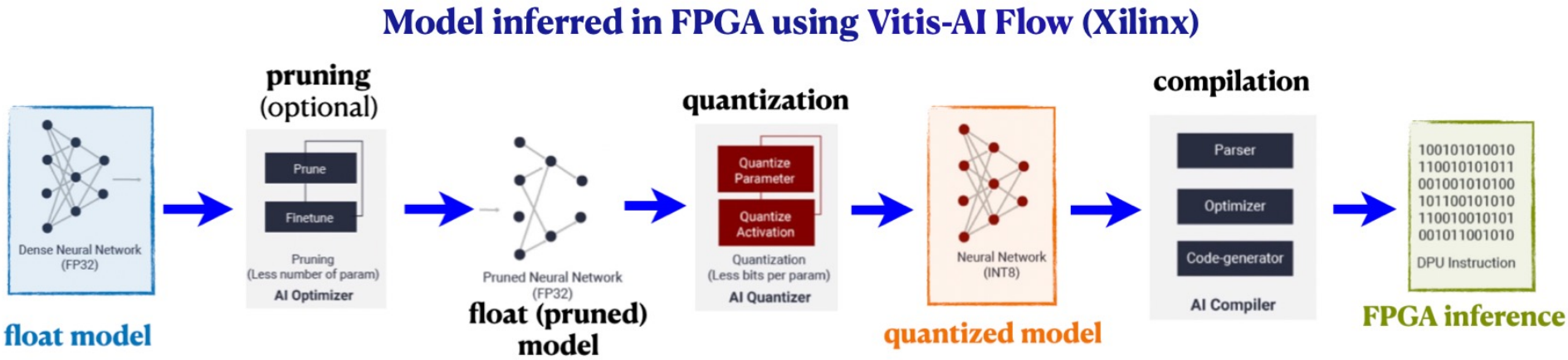
**Deep Neural Network**

Input variables:

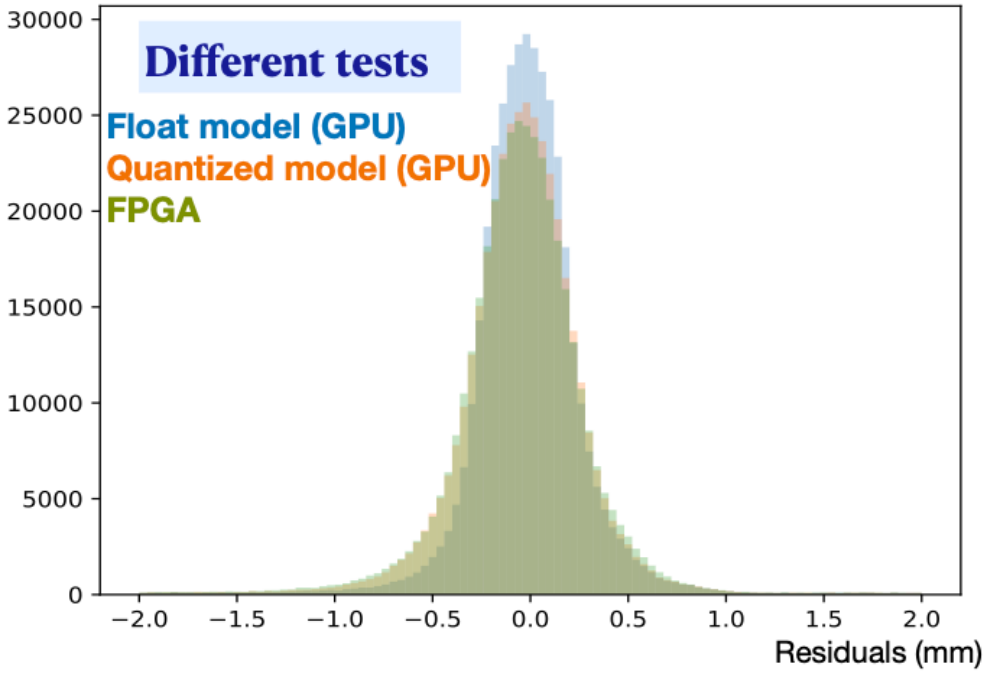
- Total number of hits belonging to the cluster
- The charge/position of the strip with highest charge
- The charge/position of its two left-right closest neighbours



# Muon cluster reconstruction

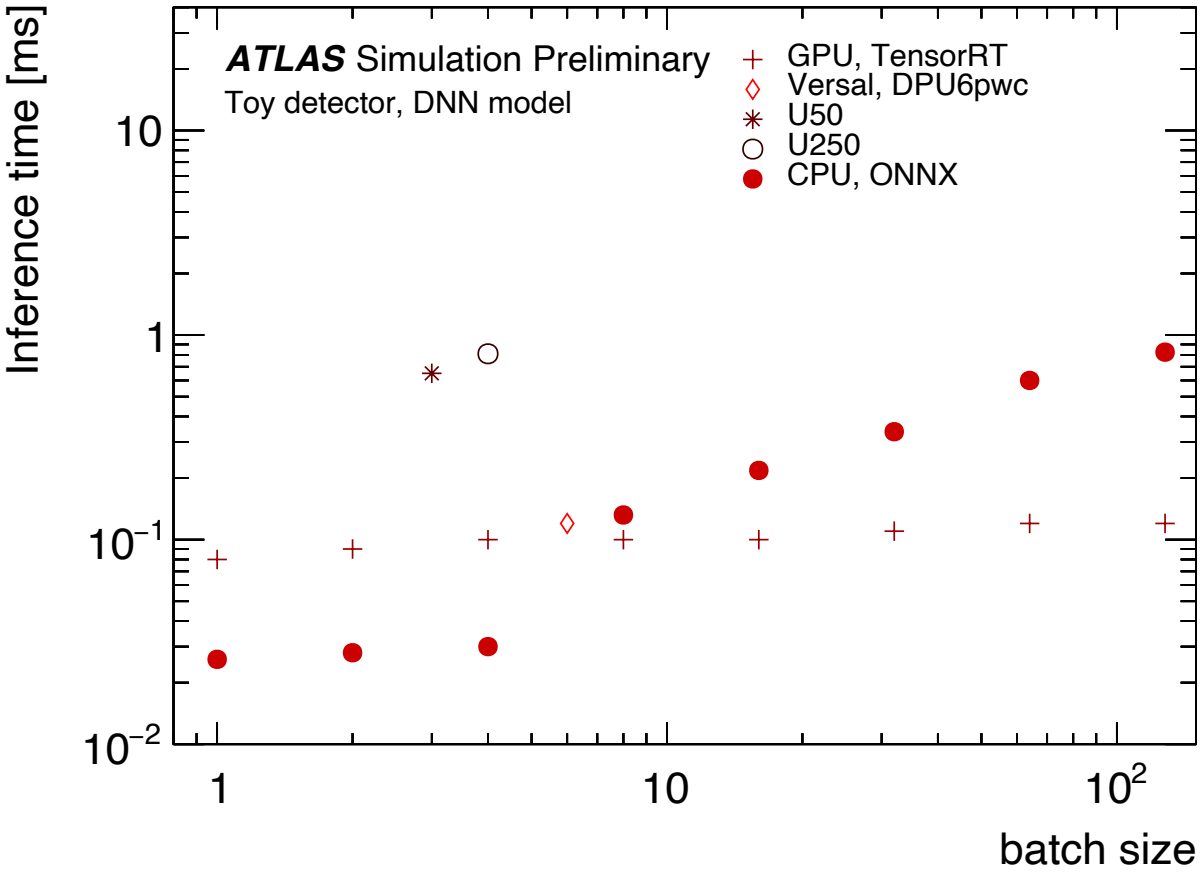



- Quantization converts 32-bit floating point weights and activations to fixed-point INT8
- Many quantization models available
- No re-training was performed, accepting a small degradation of performance



# DNN: inference time results

- CPU is already well within the latency requirements
- GPU with TensorRT improves a bit further
- No significant gain observed over both architectures

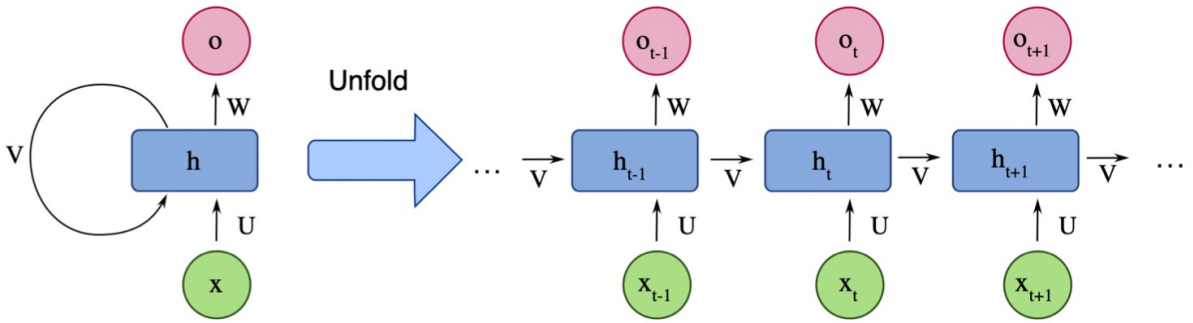


# Pattern recognition and tracking

- **AIM:** we want to identify which hits belong to the track and which are background hits
- Current algorithm for offline reconstruction are not optimized for NSW:
  - Large number of fakes with high occupancy, with consequent time increase

Current pattern finding algorithm: **Hough transform (HT)**

Implementing new machine learning algorithm to test on FPGA

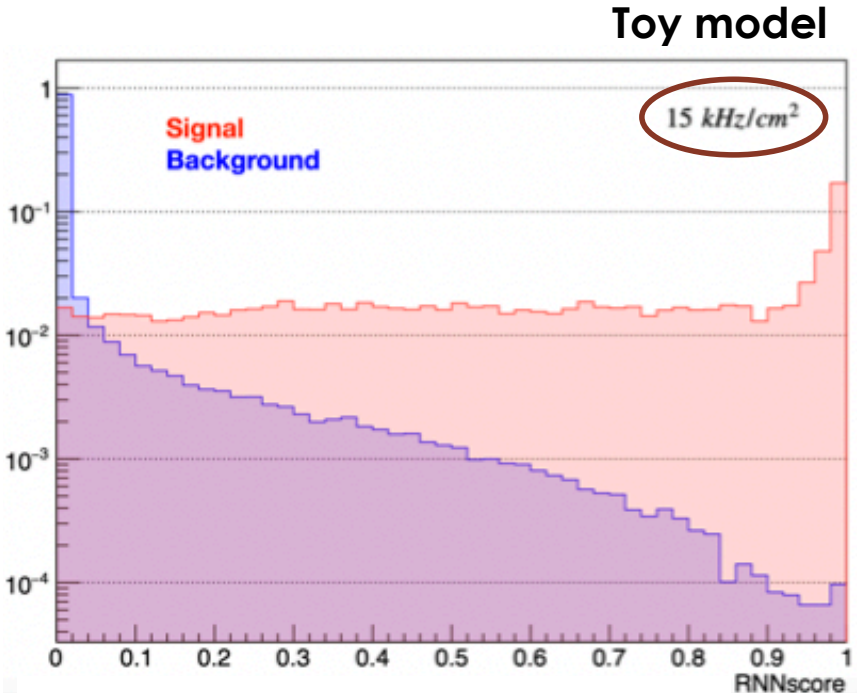


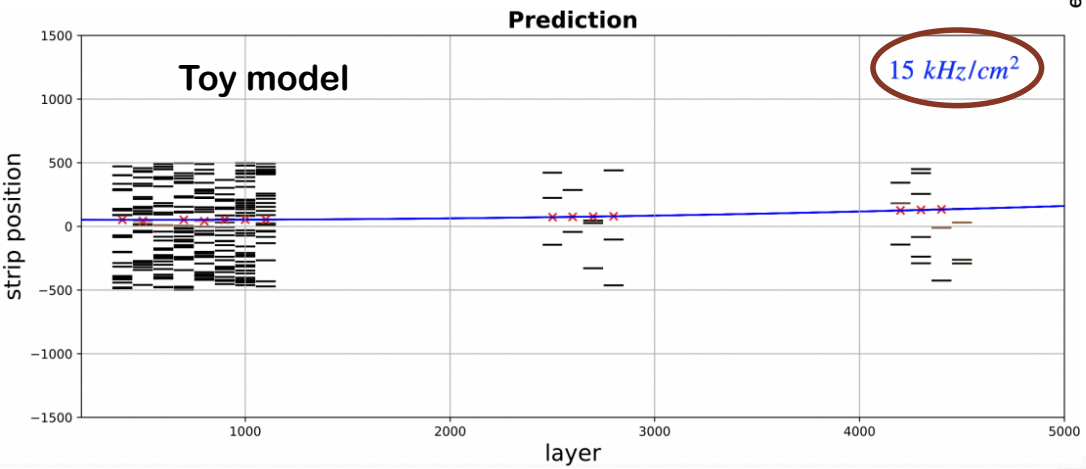

**Recurrent Neural Network**

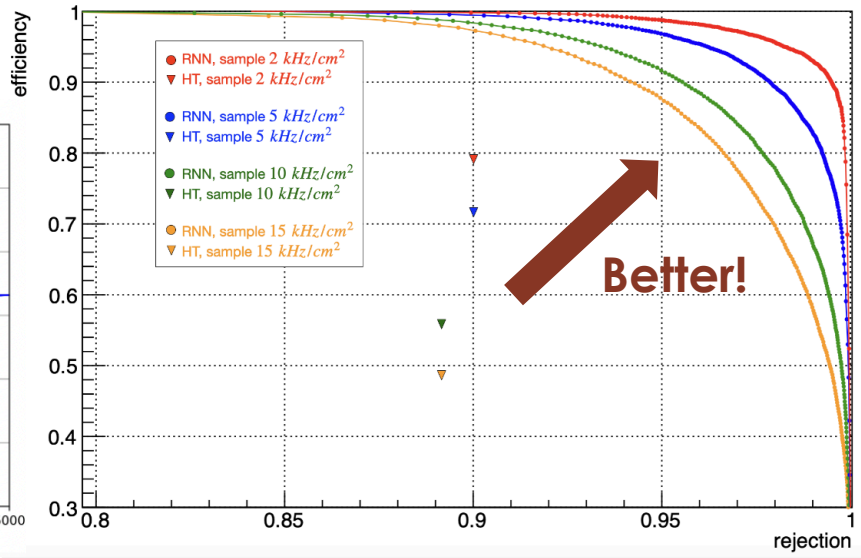
- Designed to deal with sequential data
- At each step, it takes as input at time t the output at time t-1

# RNN performance results

- Performance evaluated for different rates
- Decrease of performance seen at higher rates (expected)
- Performance show improvements over a naive HT model
- Model evaluated on CPU only ☹
- Using ONNX on CPU, inference for a single event of **O(1 ms)** – well within the lately HLT requirement

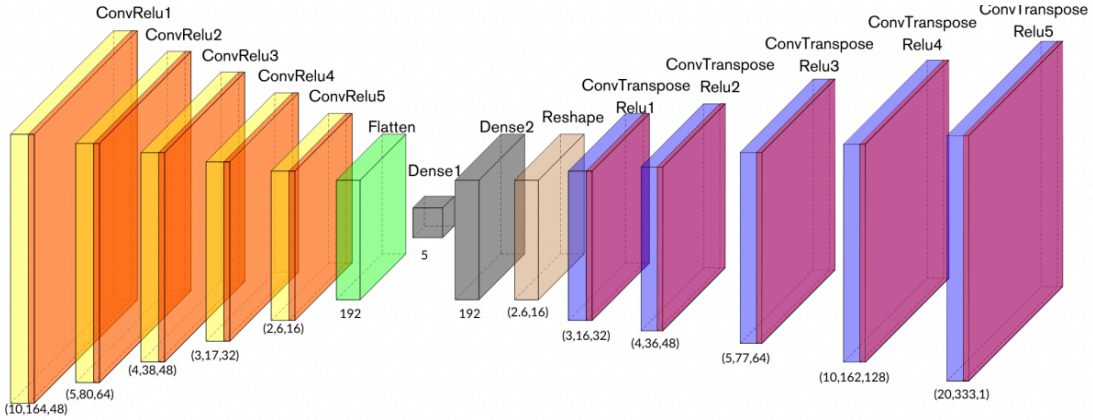


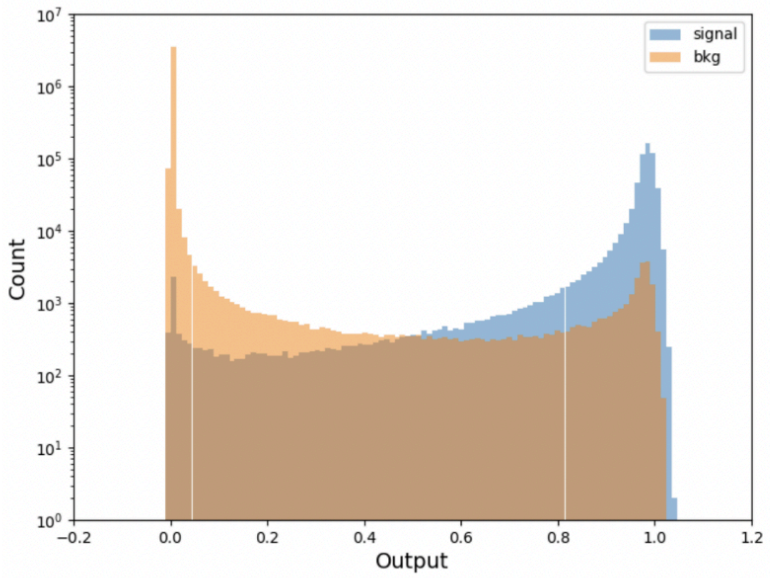



# Alternative: a CNN approach

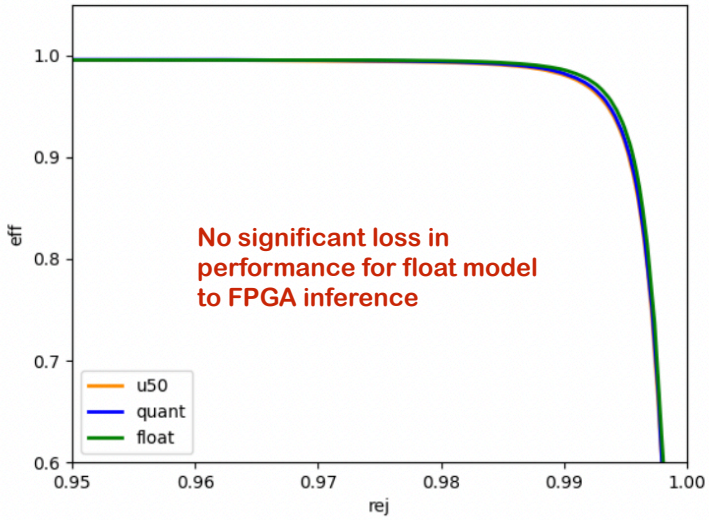
- In order to test the algorithm with Alveo cards, a CNN was also developed
- Not optimal approach for pattern recognition but useful for testing FPGA performance
- An event display translated into a 3000x16 pixel 2D image
- Convolution/deconvolution operations are used




**Toy model**

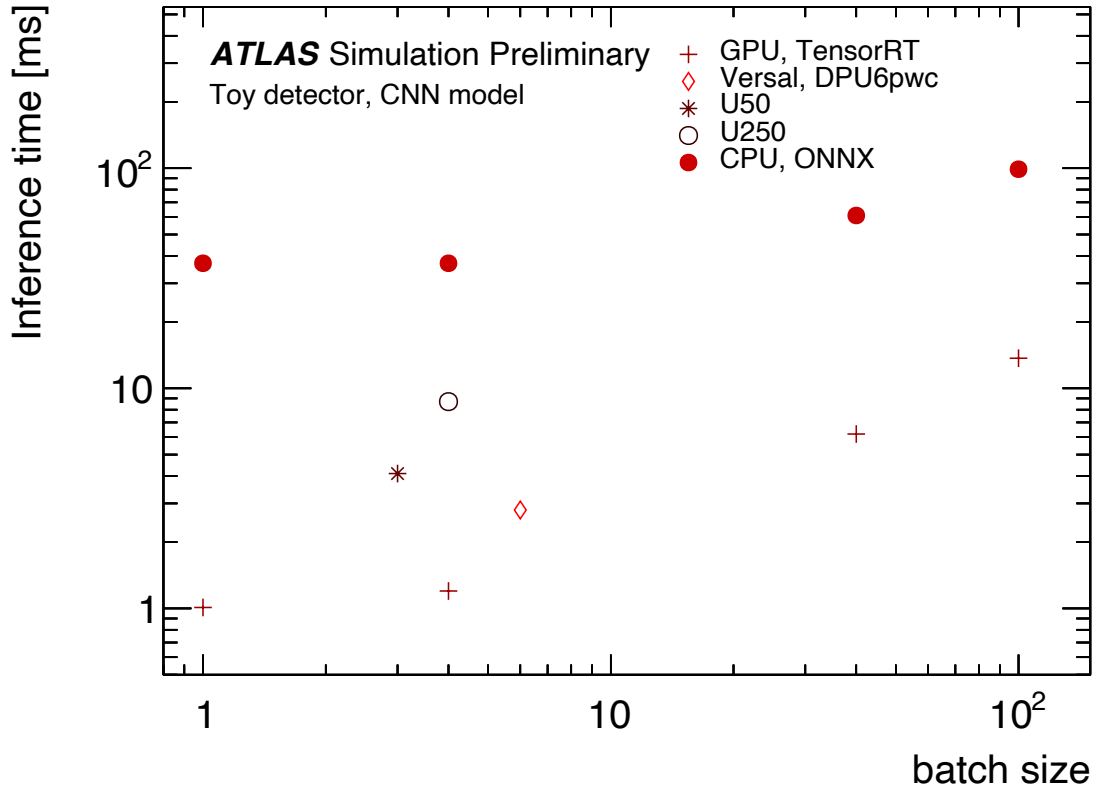



**Toy model**



# CNN: inference time results

- CNN model successfully tested on **CPU, GPU and several FPGAs**
- Overall CPU already meets the requirement imposed by the HLT latency
- **Largest improvements is seen with TensorRT on GPU**
- CPU load to be studied, as well as the power dissipations



# Conclusion & outlook

- During the current Run 3, redesign of ATLAS HLT framework to support the Multi-Threaded mode and to share reconstruction modules with offline
  - HLT farm upgrade, increasing the performance to 2.0M HS06 (start of 2023)
- HLT upgrade based on a mix of commodity and custom solutions for HL-LHC
  - Most projects already passes many reviews
  - Prototypes available for many projects
- Event Filter
  - Investigating accelerator options, technology decision in 2025
- Toy models have been implemented in order to investigate timing performance on commercial accelerator cards
- Choice will be a balance between:
  - Implementation complexity of novel technologies to the HLT farm and costs
  - Gain in power consumption and CPU load (yet to be studied)
  - Time performance… as well as cost
- Preliminary studies suggest that CPU are already suitable for this task

# Backup Slides

# Hardware platforms

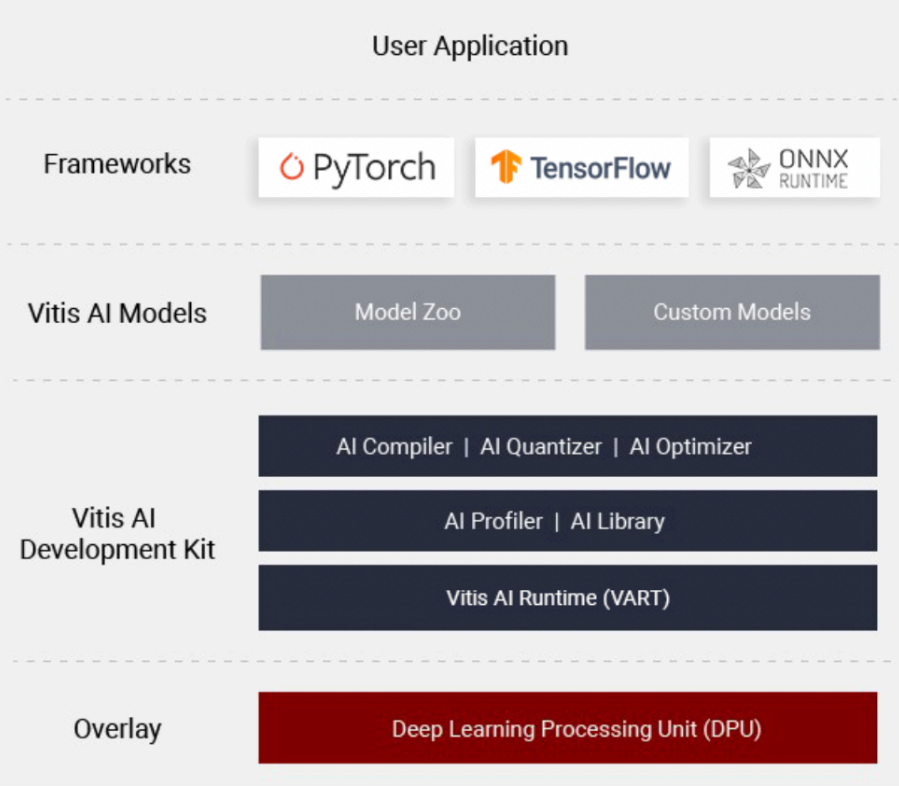


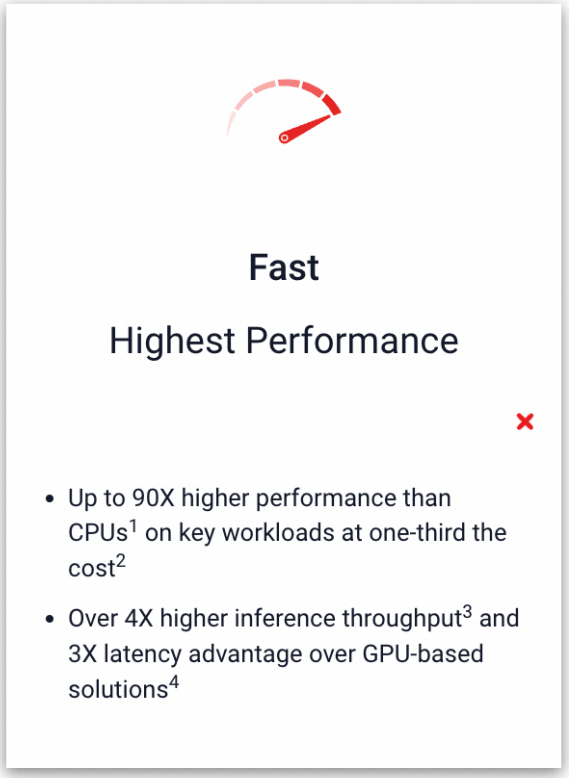
- **Direct HLC implementation into FPGAs**
- Require implementation of a neural network in VHDL or similar
- Significant effort to do so
- Platform developed and maintained for HEP community exists: hls4ml
- Fand suitable for a Level-0 trigger

- **Use commercial accelerator cards which offer integrated platform for deployment**
- Commercially available, no ad-hoc maintenance
- Dedicated hardware and related software to translate from high level python codes, into code executable in dedicated hardware
- Not as fast as the other approach, but suitable for a HLT trigger

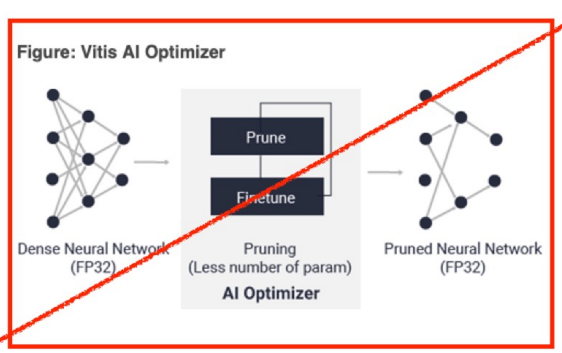
# Vitis-AI overview

- Xilinx offers several accelerator card designed and built to accelerate ML algorithms (mostly CNN)
- The claim is that inference and throughput are improved over standard COU and GPU
- Improvements also expected in terms of power consumption

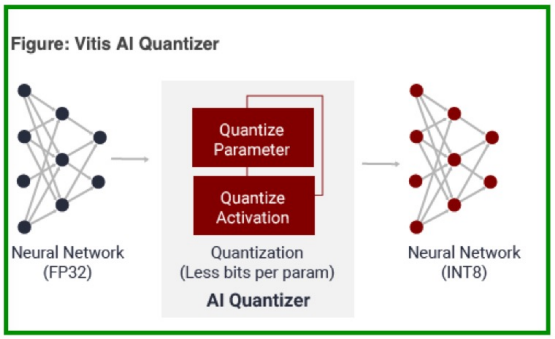


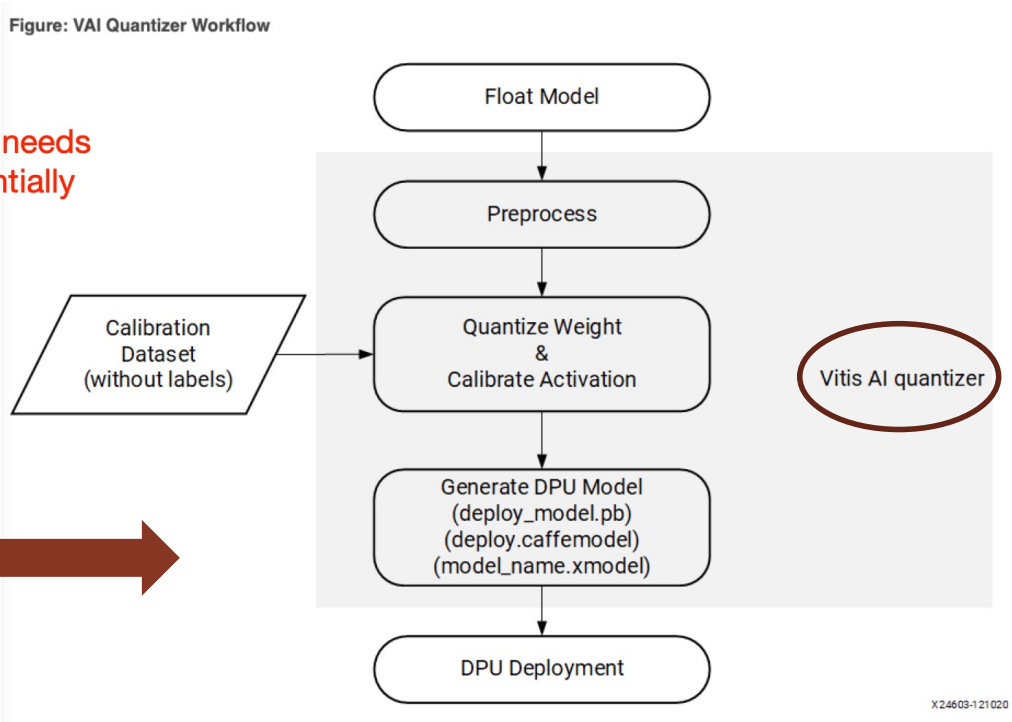



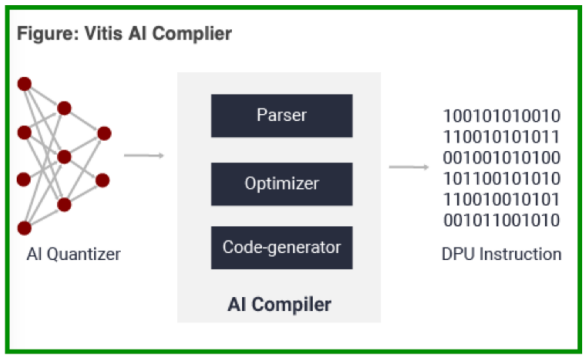
# Quantisation & compilation




Not used so far, needs a license, potentially useful




**DPUs**

Table: DPUs on Different Hardware Platforms

| DPU Name | Hardware platform |
|---|---|
| DPUCZDX8G | Zynq® UltraScale+™ MPSoC |
| DPUCAHX8H | Alveo™ U50LV, U55C Data Center accelerator cards |
| DPUCADF8H | Alveo U200, U250 Data Center accelerator cards |
| DPUCVDX8G | Versal® ACAP VCK190 evaluation board, Versal AI Core Series |
| DPUCVDX8H | Versal ACAP VCK5000 evaluation kit |

# Testbed installation

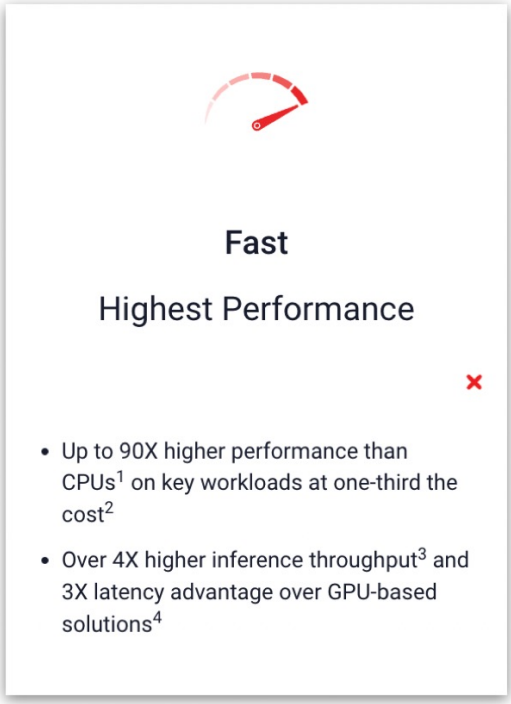
- Hardware:
  - Supermicro Server, installed in Bat40-5-D05
  - Graphics card: NVIDIA Quadro RTX A5000 PB 24 GB
  - FPGA cards : Xilinx U50, U250 and Alveo VCK5000 Accelerator Card
- Study of new ML algorithms for clusters positions reconstruction and pattern recognition with the NSW (HLT Run-3)
- Performance studies of ML algorithms in FPGA(Run-4)

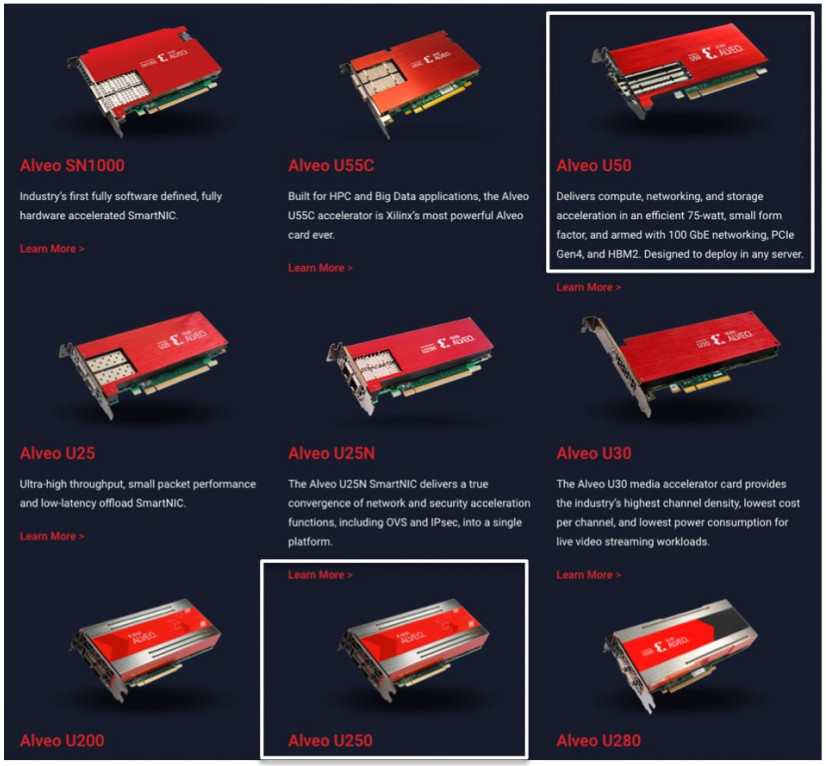
- SW
  - Ubuntu 18.04
    - OS needs to be validated against Xilinx tools
  - Docker
    - Docker GPU support
  - Nvidia drivers, CUDA tools
  - Xilinx Vitis (2021.2) development tool
  - Xilinx Vitis-AI (2.0) docker images (Tensorflow, Pytorch, compilers, Realtime support)
  - Xilinx Realtime environment (XRT) and platform support.
  - Samba
  - X2Go server
  - DHCP server for local network
- Support
  - many tickets opened to Xilinx Support to reach this stage.
    - only a subset of all potentially interesting NN architectures is supported by the current tool, but a significant improvement is found with new releases.

# Inference studies

- Xilinx offers several accelerator card designed and built to accelerate ML algorithms (mostly CNN)

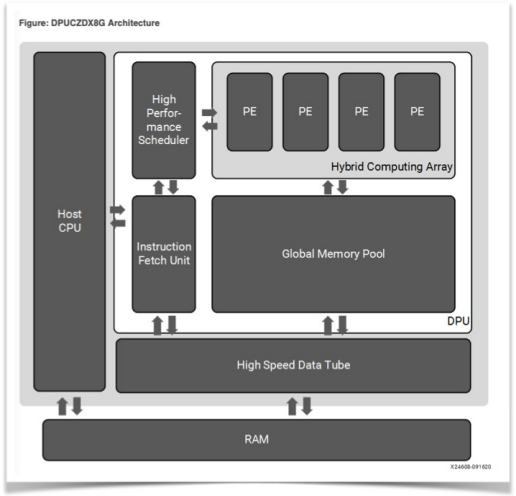

Also a Versal card has been tested



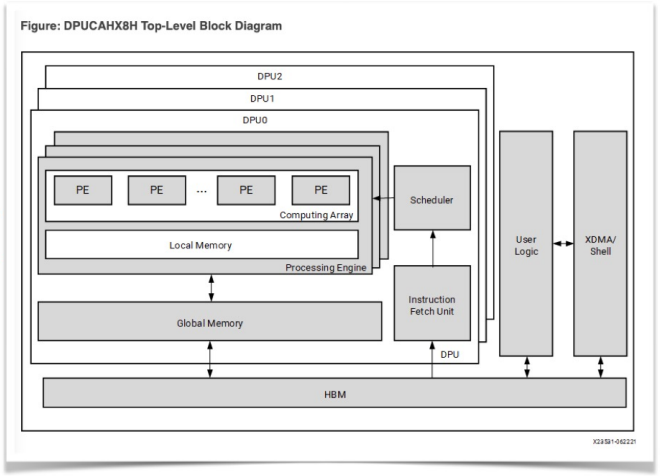



# The processing units

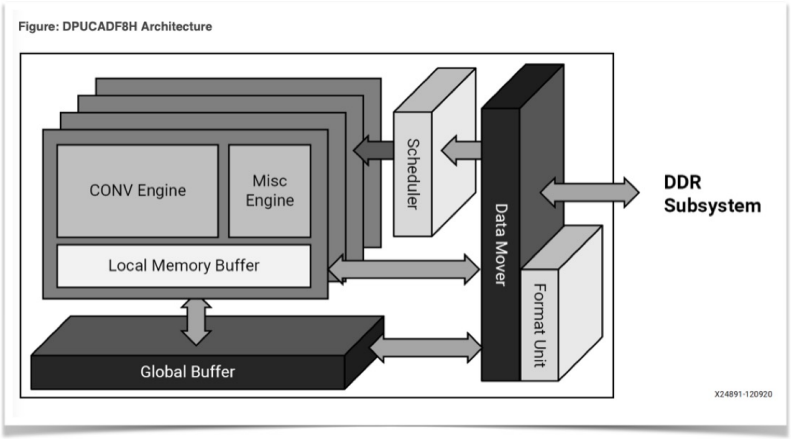
ZYNQ



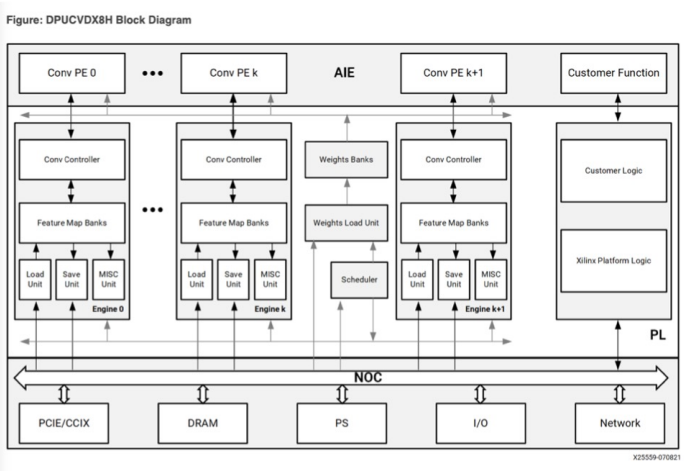

U50




U250




Versal




More about the Alveo VCK5000 accelerator board later in the talk

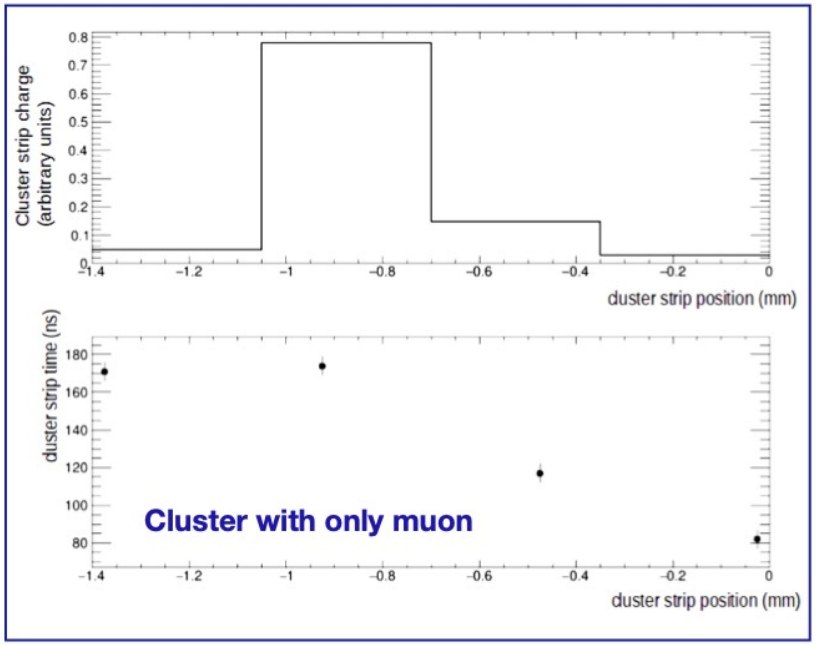
# Muon identification and tracking

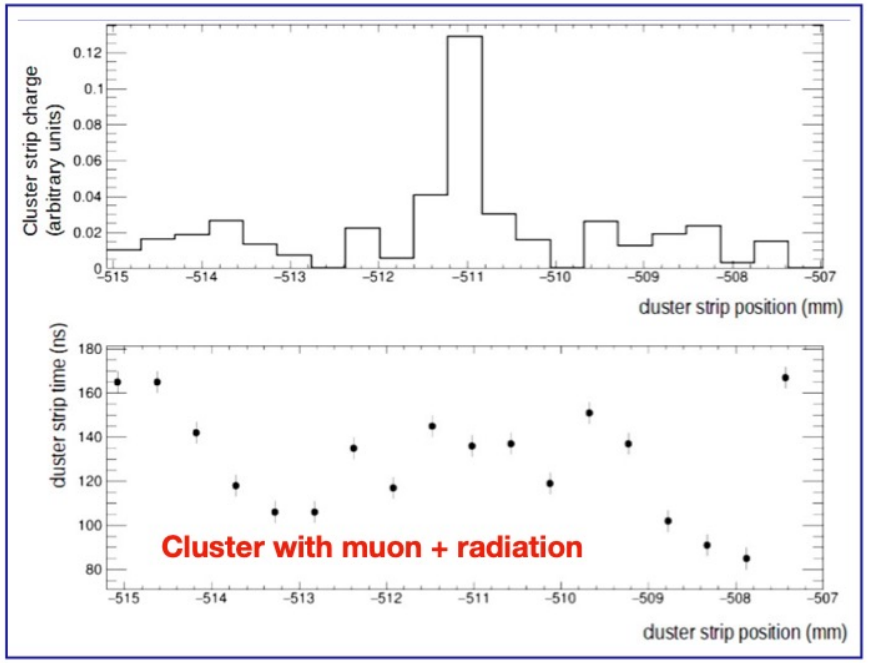
- Work started during software development for the New Small Wheel (NSW)
- Study the implementation of Deep Neural Network (DNN) for:
  - Identification of cluster produced by muons in Micromega (MM) and Small TGC (STGC) chambers
  - Optimization of the single-point (i.e. by layer) position resolution
  - Cancel or mitigate the effect of backgrounds, in particular those correlated to the muon track
- Study Recurrent Neural Network (RNN) or other possible AI-based approaches (i.e. Convoluted Neural Network – CNN) for pattern recognition, muon identification and momentum measurement
  - Combine the layers info in a pattern across all muons stations
  - Compare to other existing approached i.e. Hough Transform
- Models initially thought for the offline, proved to be a good benchmark for studies on AI-accelerators

# Why ML?

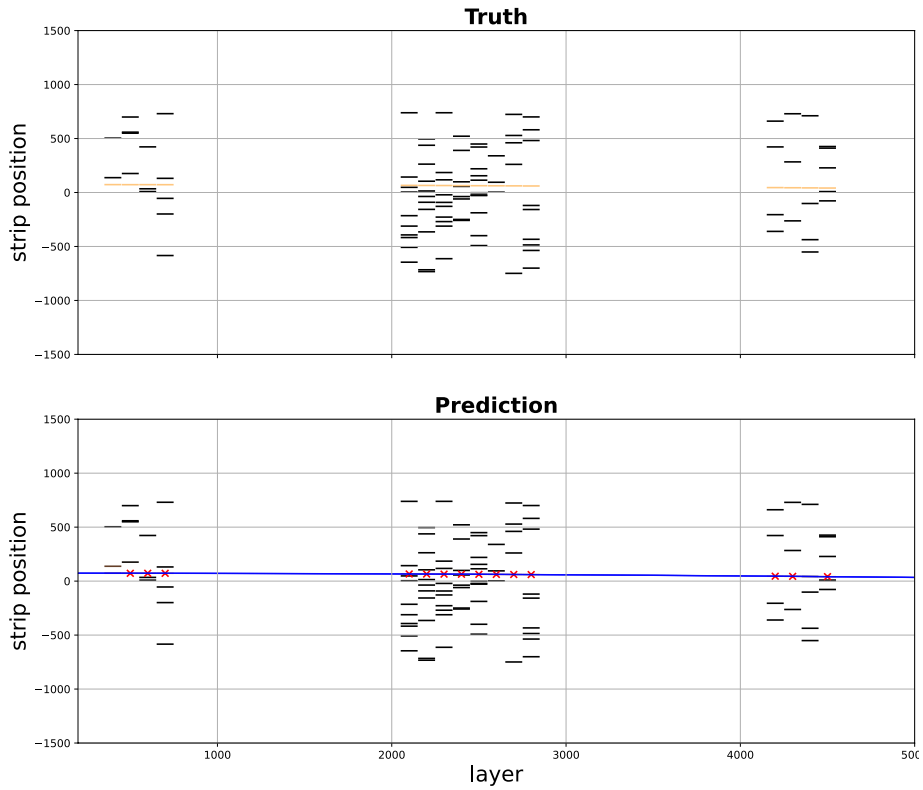
- Current methods performances worsen with correlated background cause of showers produced by muons in the last part of the JD shielding

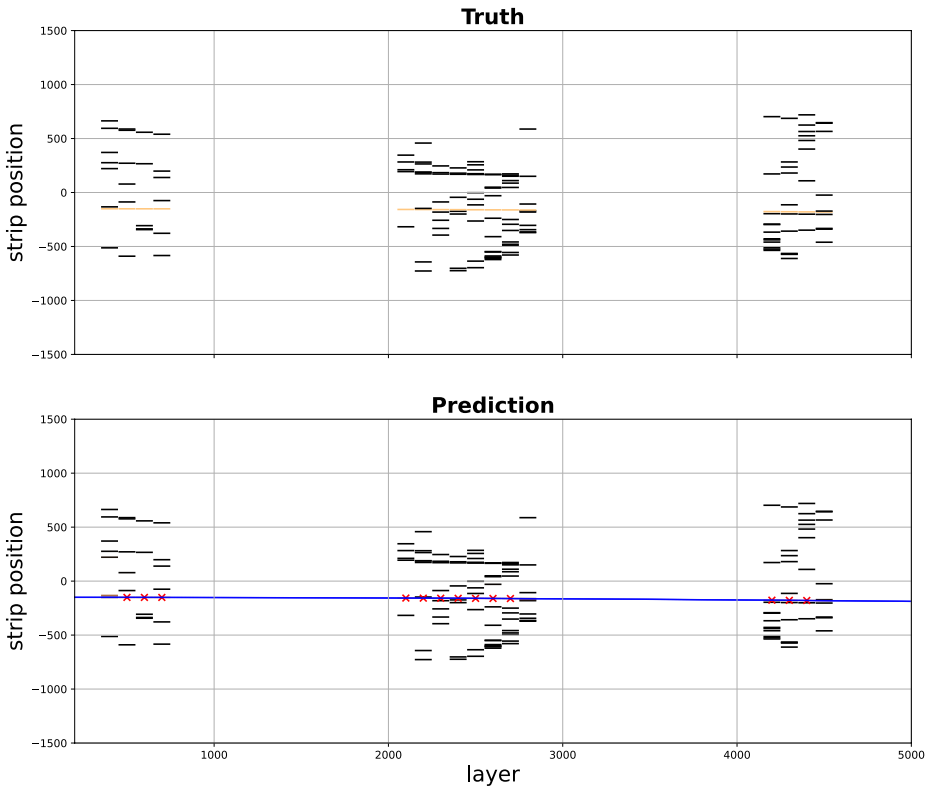
- ML algorithms implemented to:
  - Classify clusters → assign different errors to the fit in track reconstruction
  - Increase the single point resolution
  - Mitigate effects of correlated background with dedicated NN training



# Pattern recognition and tracking



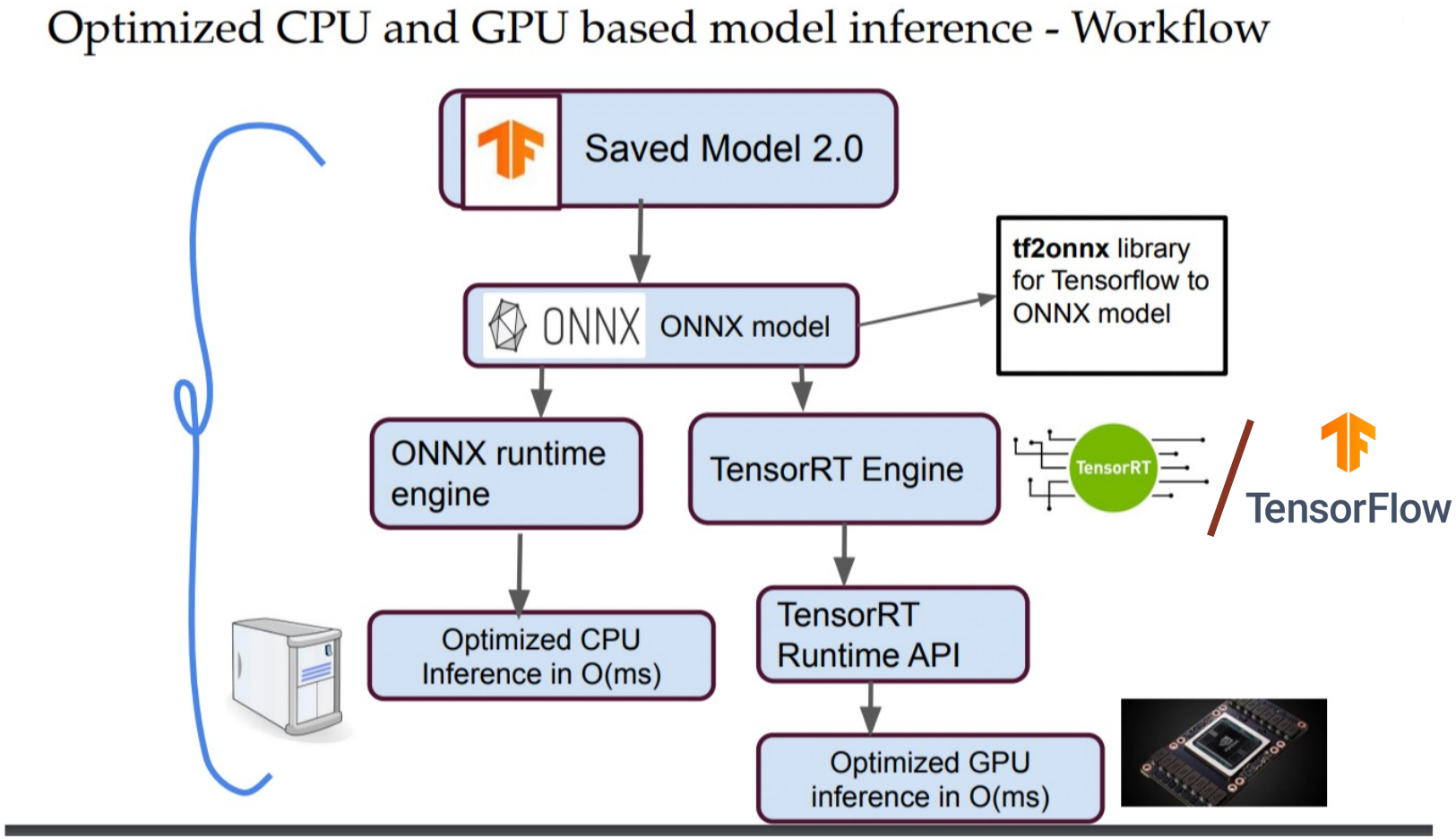




- Truth on top, black points represent background hits, while orange ones muon signal
- Predictions from the RNN at the bottom (from black to orange, according to the RNN output)
- The 'x' represents the hits which the output cuts (i.e. the hits used in the fit to determine $p_T$), while the blue line the reconstructed $p_T$

**$p_T$ = 0.3BR**

# Inference studies

- To run the inference:
  - ONNX: **"session.Run(input_names, batch_input_tensors, output_names)"**
  - TensorFlow: **"Output = model.predict(input)"**



# Inference studies

- Main point is that tensorRT does not work with dynamic batch sizes
- We want to study the inference time when varying the batch size
- Below is report the general workflow to change the batch size


```python
import onnx
onnx_model = onnx.load_model('model_singleLoss.onnx')

BATCH_SIZE = 1
inputs = onnx_model.graph.input
for input in inputs:
    dim1 = input.type.tensor_type.shape.dim[0]
    dim1.dim_value = BATCH_SIZE

model_name = "model_singleLoss_mod.onnx"
onnx.save_model(onnx_model, model_name)
```

Example script

# Alveo VCK5000 accelerator board

- We have 1 **Alveo VCK5000 accelerator board**, which allows us to run AI algorithms on Vector processor Arrays
- To look at new features, like pipelining together different engine Kernels
- Vitis 2022.3.5 have been installed
- **Preliminary studies** on the DNN with batch size = 8:
  - **2x (3x) slower than CPU (GPU)**
  - **2x faster than U50**
- Timing seems to be independent from the complexity of the model (number of layers, weights, etc.)
- Next steps? Same studies for the CNN (not supported currently)