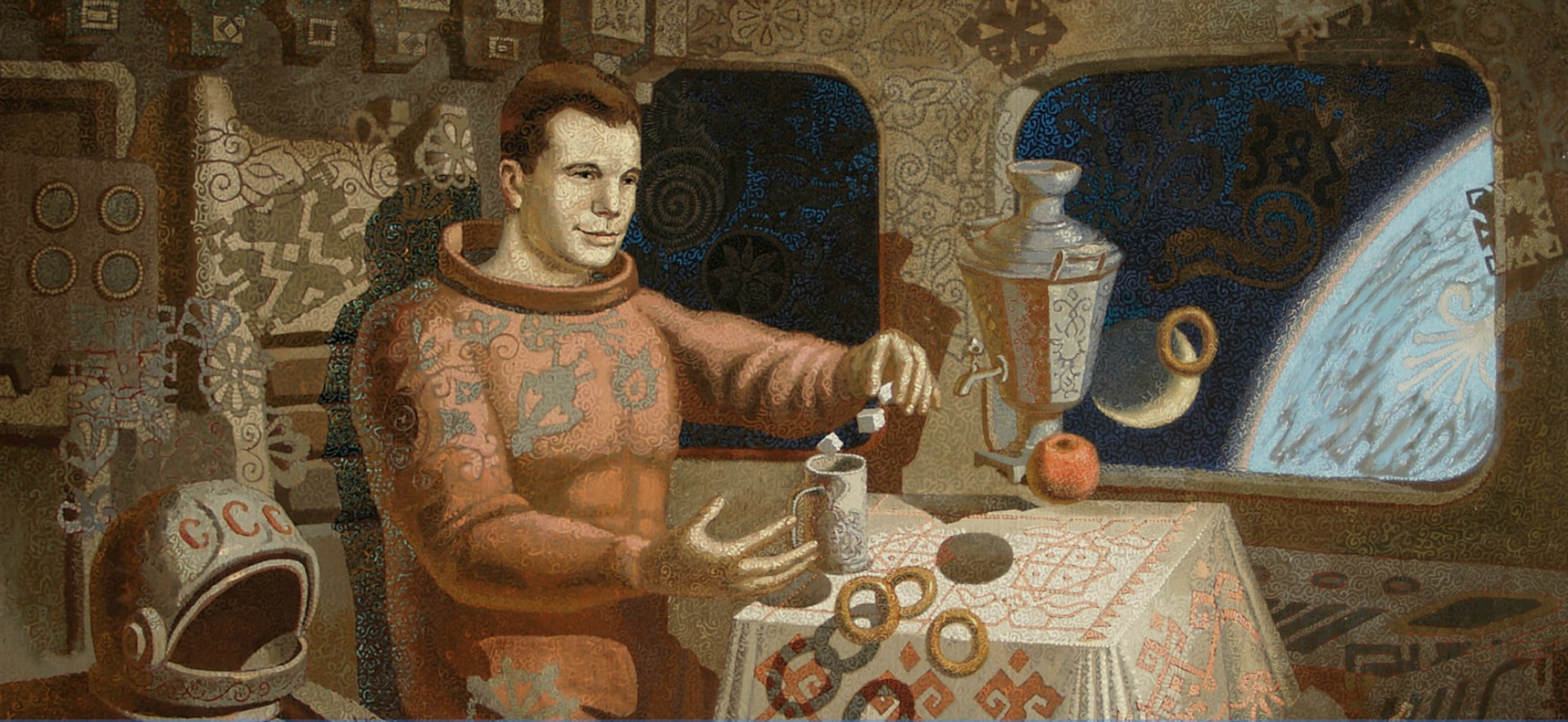
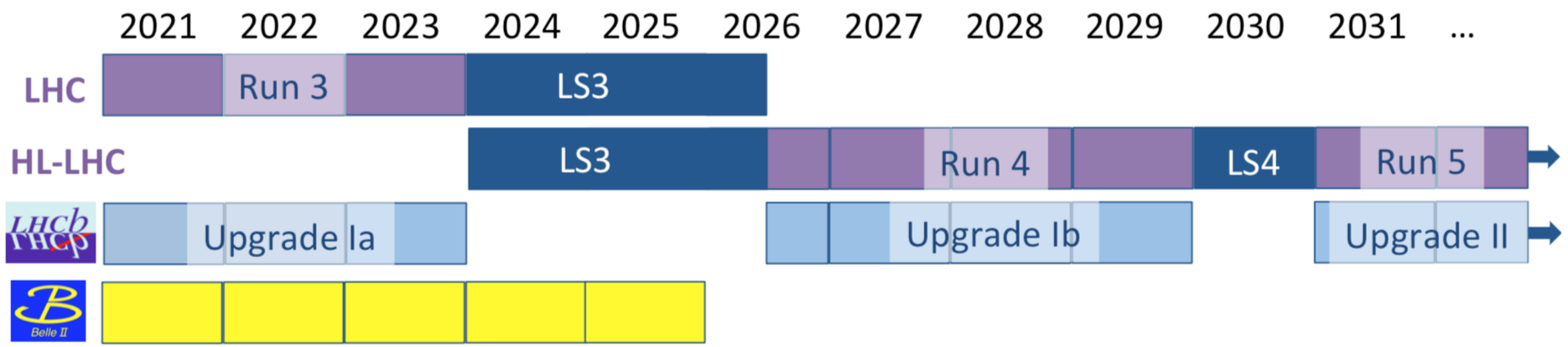


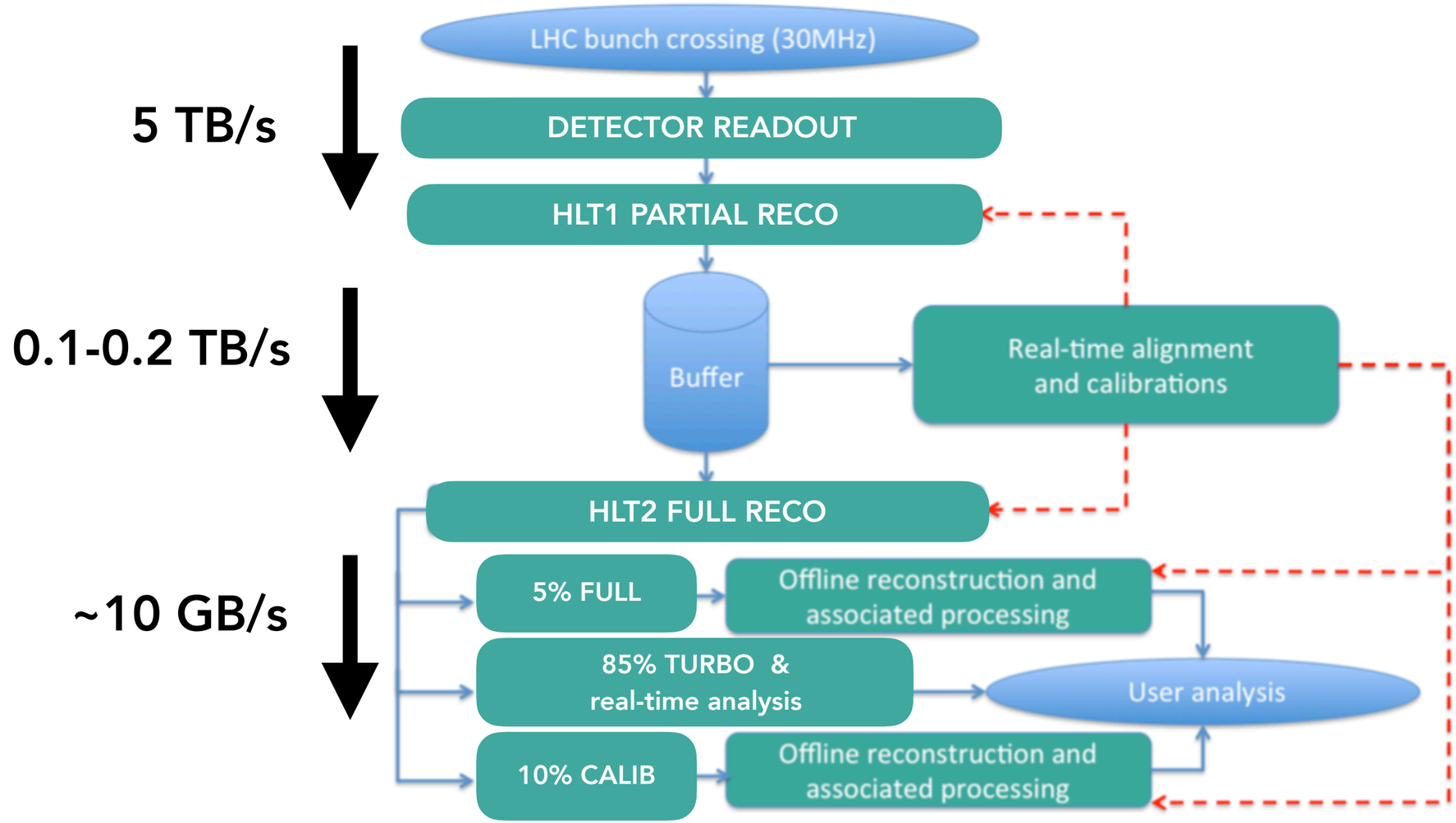
# Comment peuvent-en profiter des algorithmes AI sur les cartes FPGA au avenir?

# WHAT PROBLEM DO WE NEED TO SOLVE?

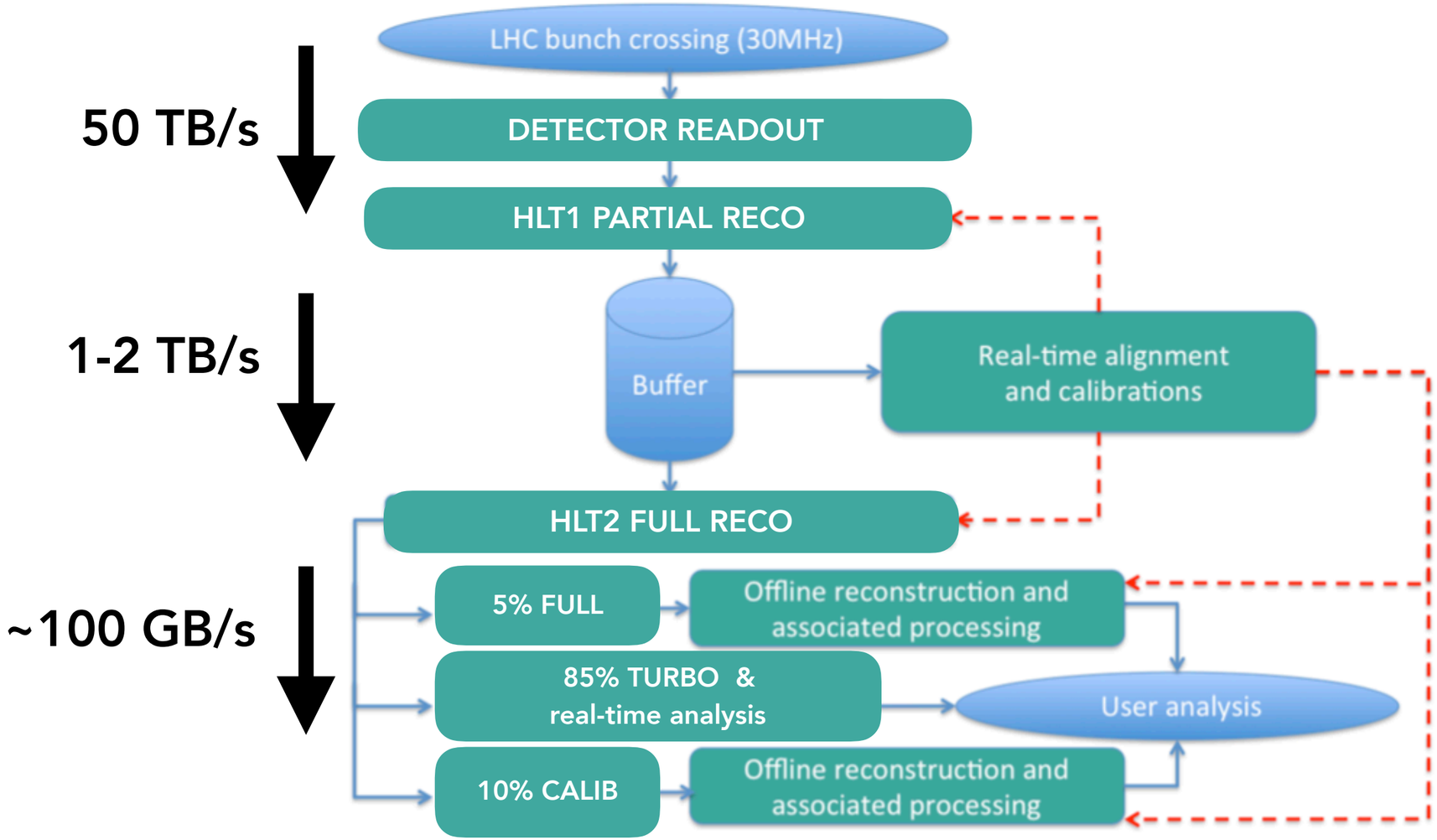
# Timelines and jargon



# LHCb real-time architecture 2021



# LHCb real-time architecture 2030?

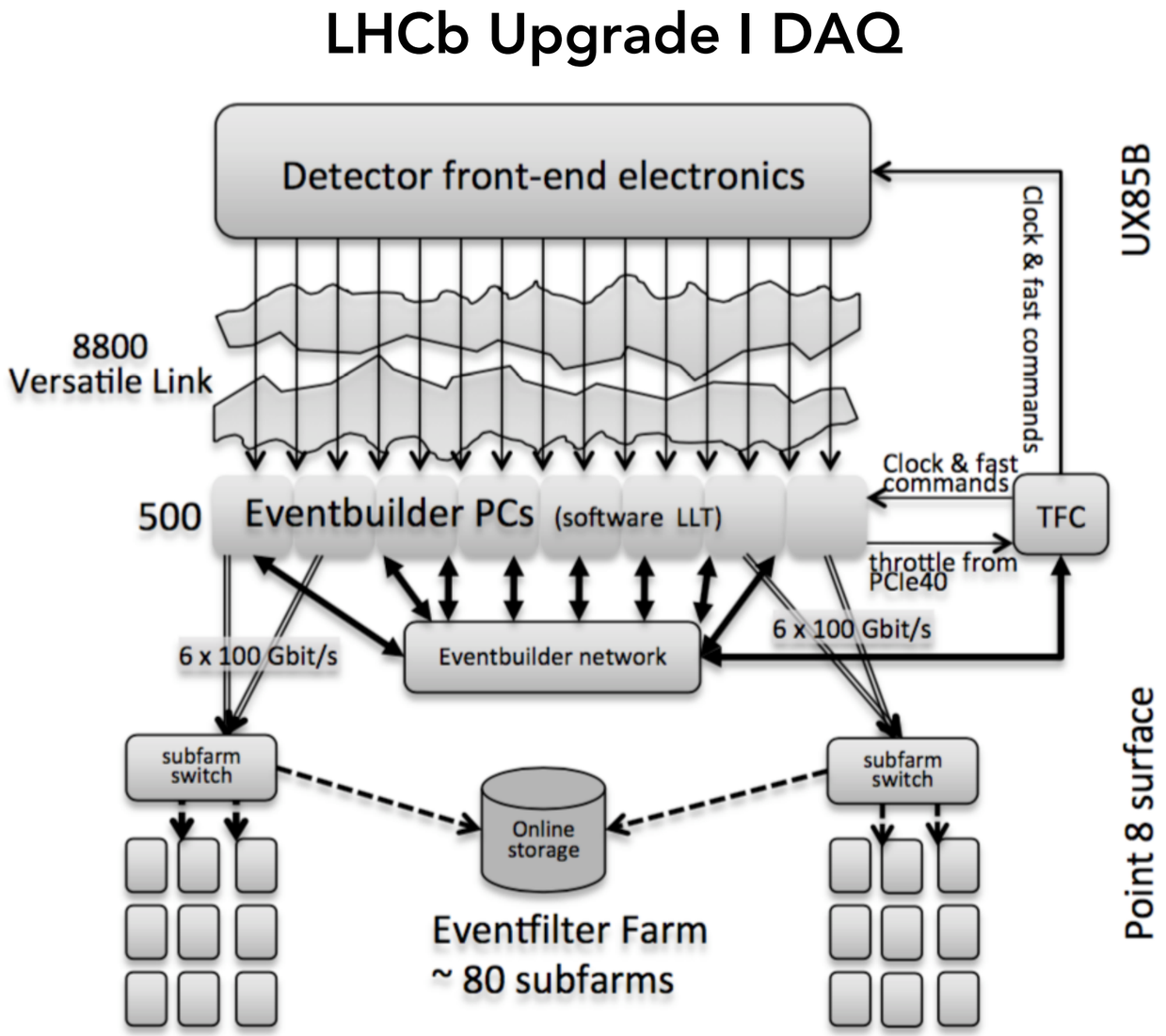


# Challenges & evolution of DAQ

GBT link : 4.8 Gb/s Upgrade I
Assume evolution to 10 Gb/s for HL-LHC using aggressive error handling : missing factor 5 compared to data rate growth.

Event-building : current network is 500 servers with 100 Gb/s links. 200 Gb/s readily available, keep an eye on price/ performance scaling beyond this?

Farm : carry out R&D in next years on optimal use of hybrid architectures (GPU/ CPU/FPGA), remain flexible



# Use of AI/ML today in LHCb

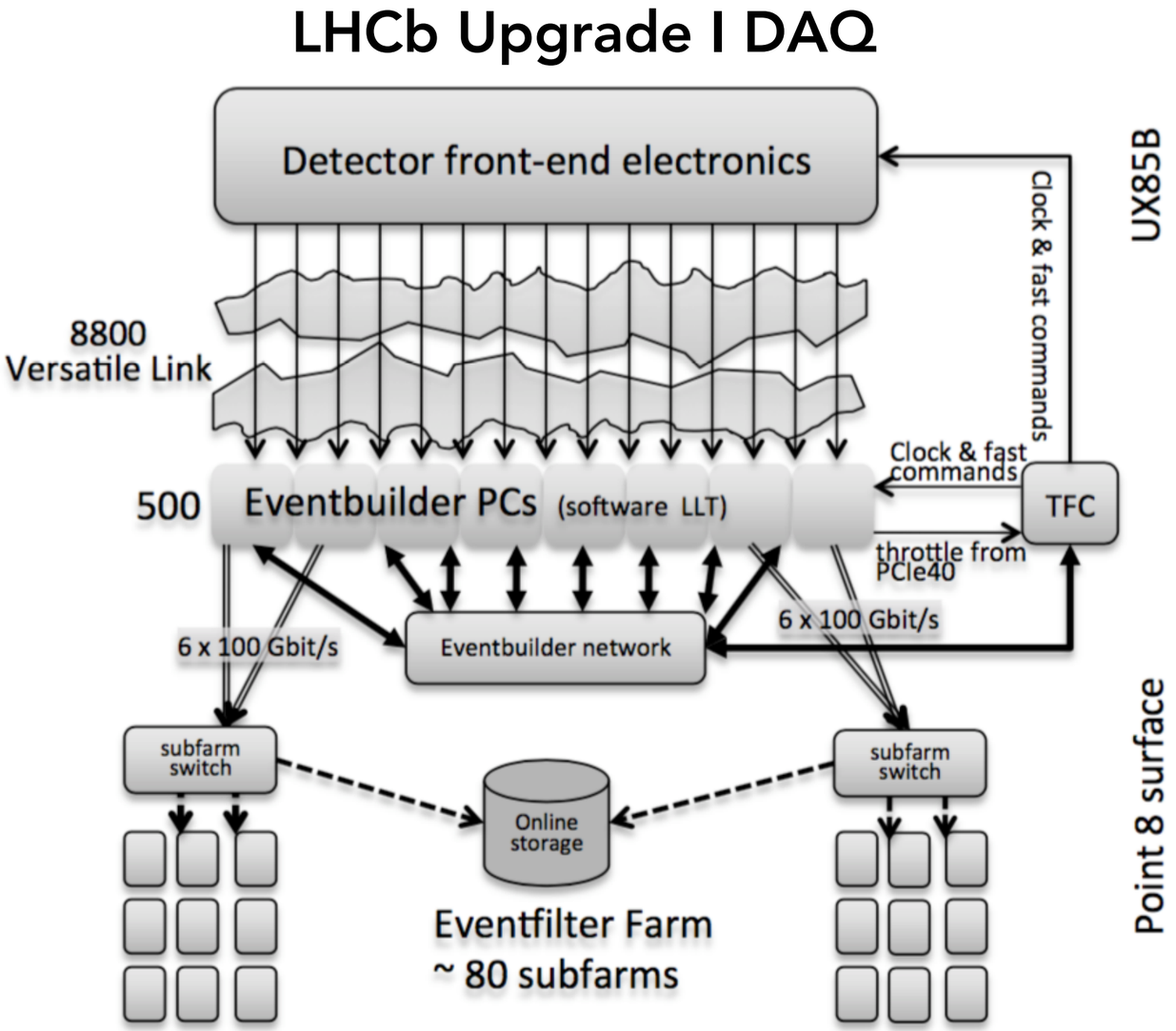
1. Many trigger lines used ML classifiers in Run 2 — expected that most will use ML classifiers in Run 3. Typically for classification at our working point neural nets gain little over BDTs.
2. Main particle identification and flavour tagging algorithms are based on neural nets, both for charged and neutral particles
3. Neural nets used in parts of the track reconstruction to discriminate between good and fake hit combinations, speeds up reconstruction significantly

Summary: very extensive use, well developed framework for deployment on CPU. Deployment on GPU is in a much less advanced stage but no “real” difference with respect to CPU, simply we started using GPUs much later so framework less mature

Some attempts to have full reconstruction algorithms replaced by neural nets, but so far none have been completely successful — usable reconstruction algorithms are still a mixture of classical Kalman filter and combinatorics based steps and neural nets in certain specific parts. But this may change in the future.

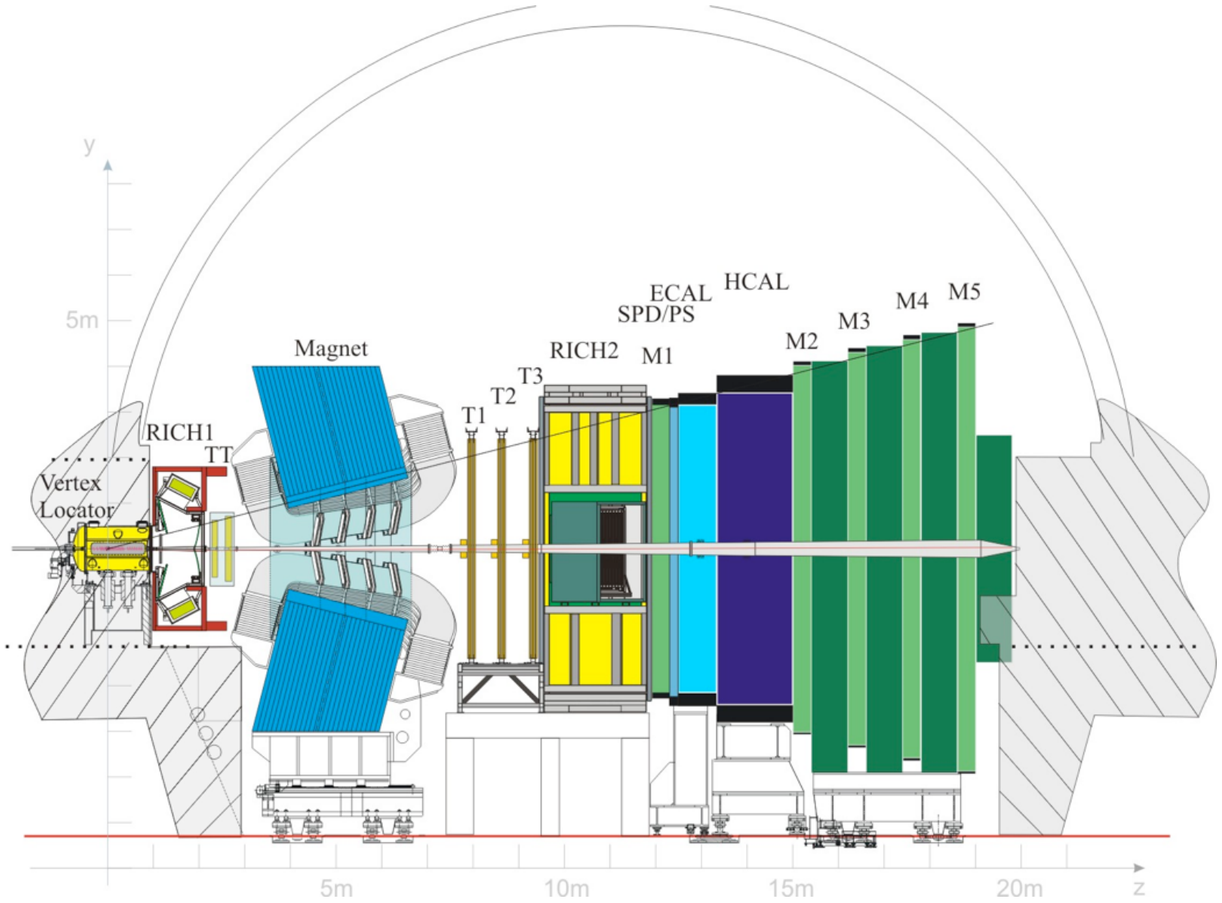
Key point on CPU/GPU: all detector data available to all algorithms! Very different on FPGA

# Where could we use AI on FPGA?



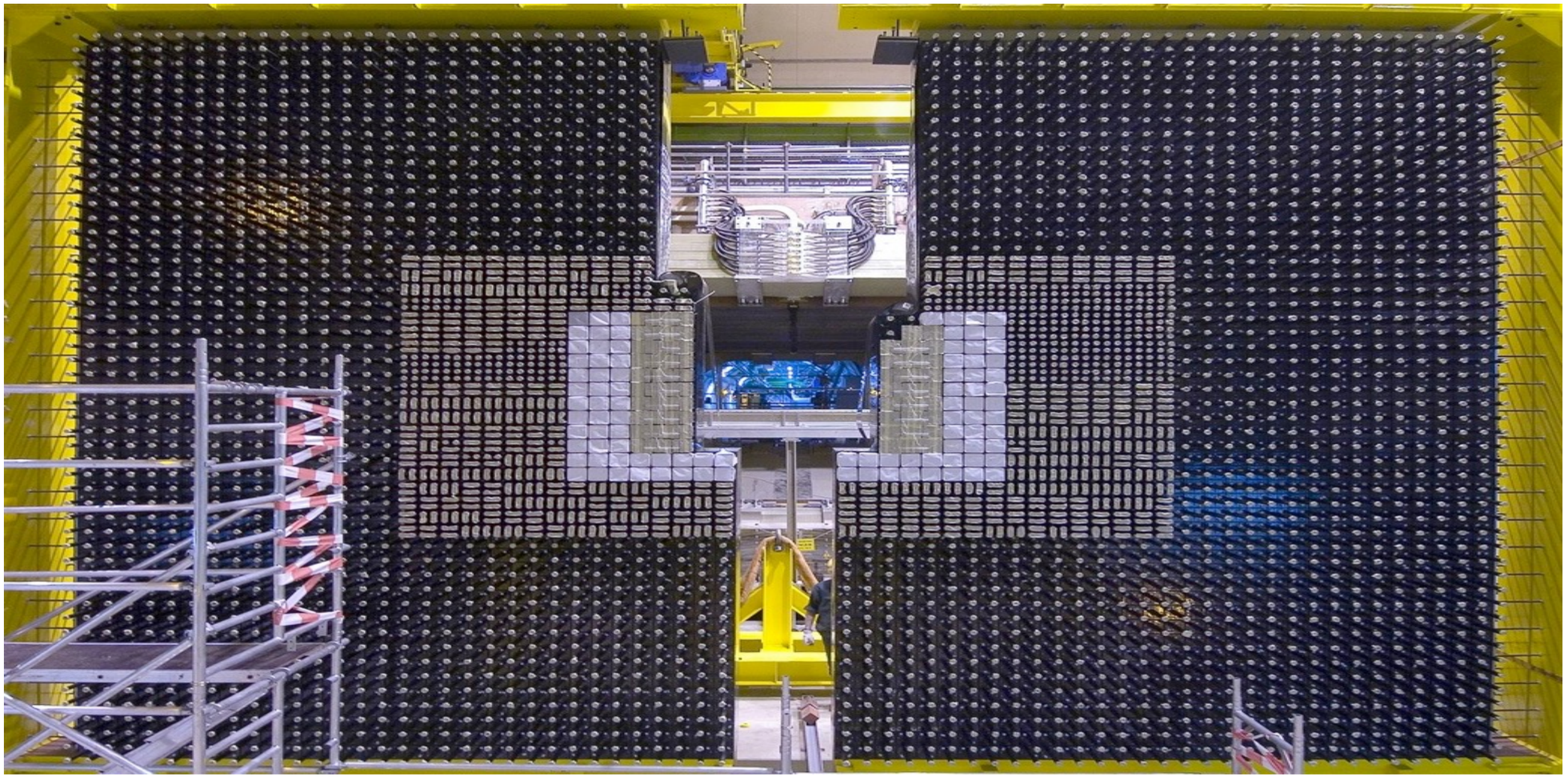

No inherent latency in system — so FPGAs have to compete on cost-benefit with GPU/CPU

# LHCb geometry constraints



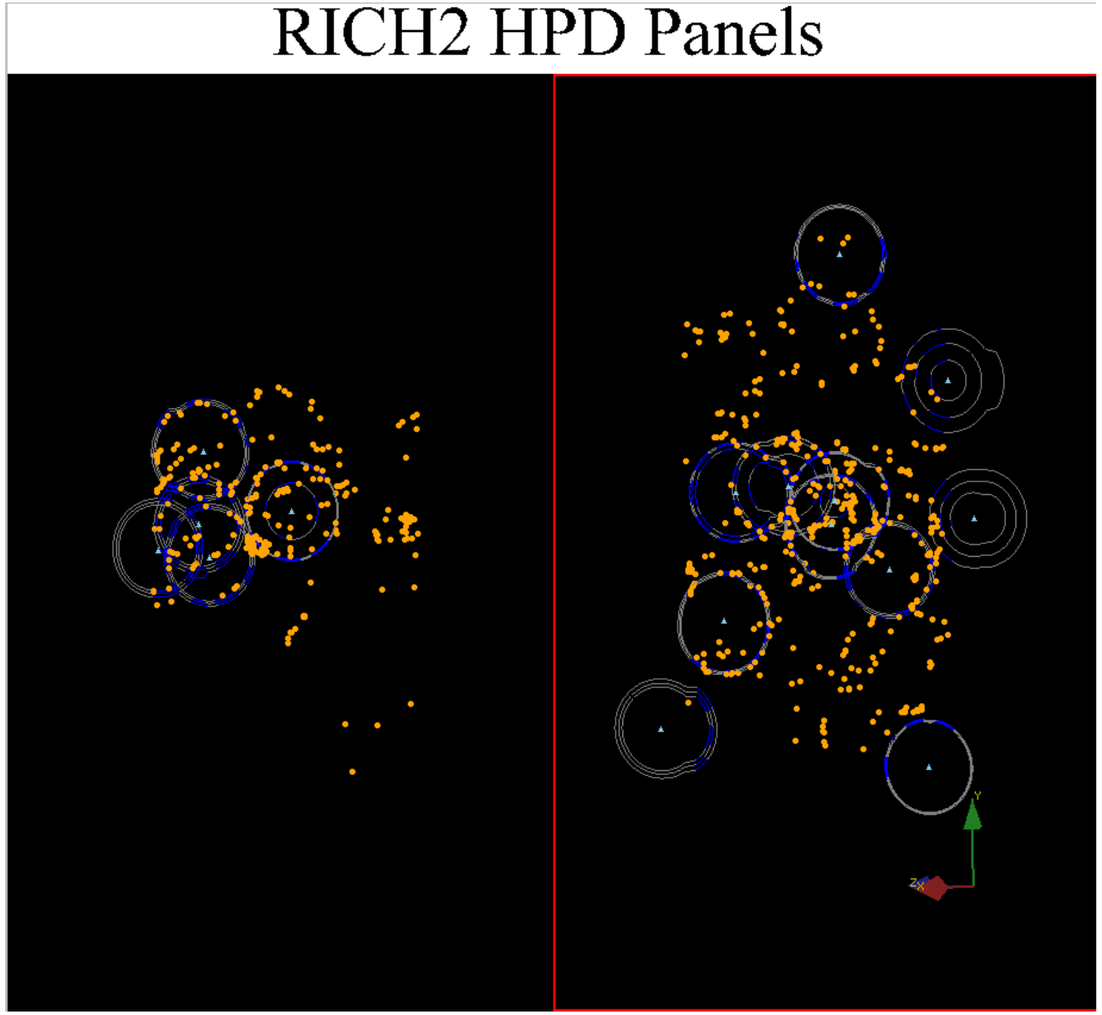

Want a problem which is inherently local, to minimize FPGA to FPGA communication

# Potential “local” problems



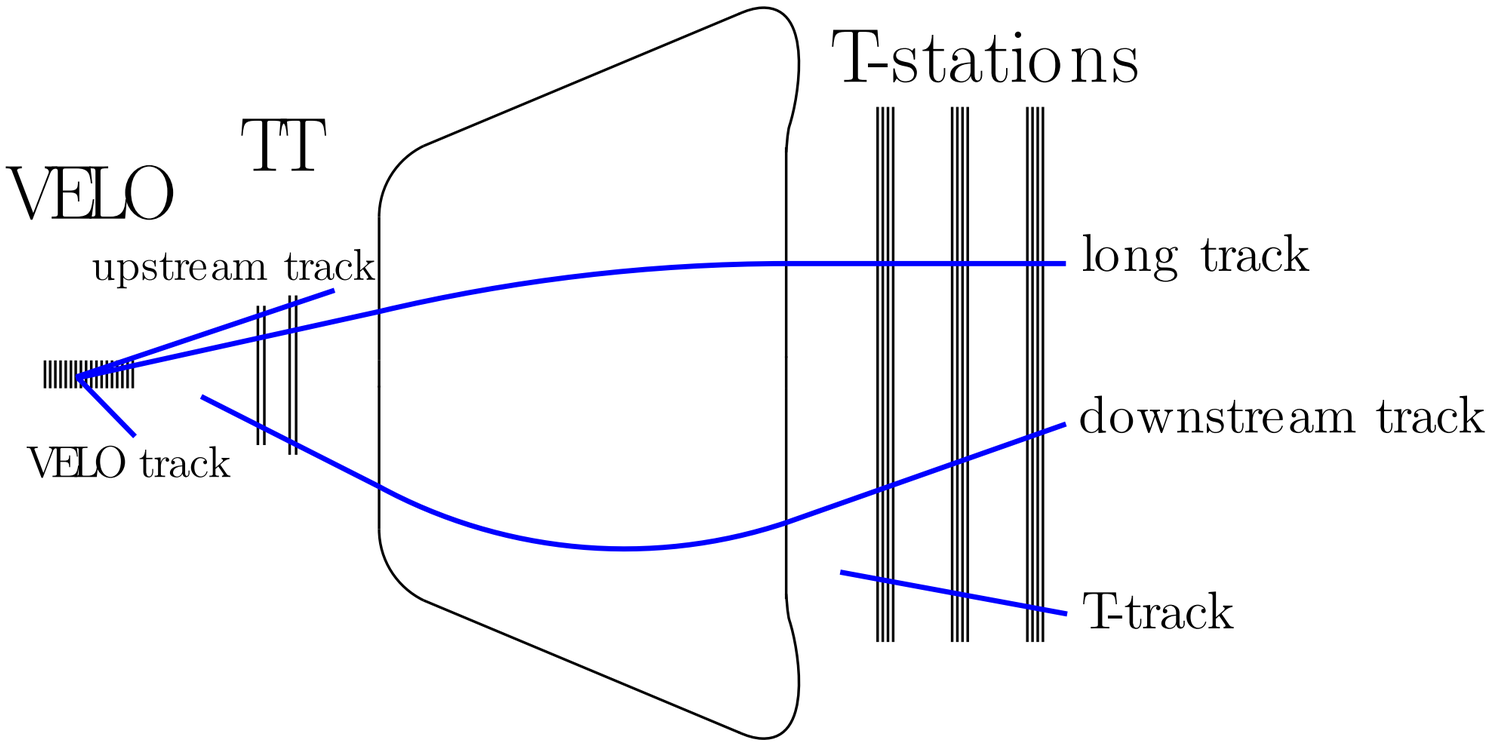
**Calorimeter reconstruction — best candidate in my opinion. Use AI to find clusters and simultaneously determine cluster types from the shower shape (photon/electron/$\pi^0$)**

# Potential “local” problems continued




RICH detectors — classic image recognition problem. However very high hit density. Could be interesting if we can project in time with O(10ps) precision to reduce the pileup

# Why not tracking?




**LHCb's magnetic field means you have to bring information from different trackers together in a highly non-local way to find tracks. Could find stubs in individual pieces of the tracker, but unconvinced this will ever be cost-effective compared to GPU/CPU if priced fairly.**

# Concrete projects

1. Understand the way to optimally use hybrid (FPGA, GPU, x86) architectures considering the dataflow as a whole.
2. Integration of as much data processing as possible into the FPGA detector readout, ettiquetage des donnees, reconstruction des objets locaux (e.g. dans calorimetre). Develop a coherent approach to the readout of *all* subdetectors.
3. Can machine learning algorithms deployed on FPGA give the same physics *faster* on highly parallel architectures?

If we want to have a useful outcome, we have to consider the whole dataflow from the start, and embed the proposed FPGA algorithms within a highly parallel CPU/GPU processing scheme. Work together with the detector readout, not against it!

There are even people who have ideas of using Ethernet directly from the front-end ASICS to the server farm — in which case no “free” backend readout FPGAs, and price-performance becomes even more complicated.