Design of the online filter

A. Zhemchugov, A.Belova, D.Oleynik, A.Petrosyan

9 June 2021

Online Filter

High-performance heterogeneous computing cluster

- The goal is to reduce the data rate by a factor of 50 or more
- Partial reconstruction
  - Fast tracking
  - Fast ECAL clustering
- Event unscrambling
- Software trigger
  - several data streams

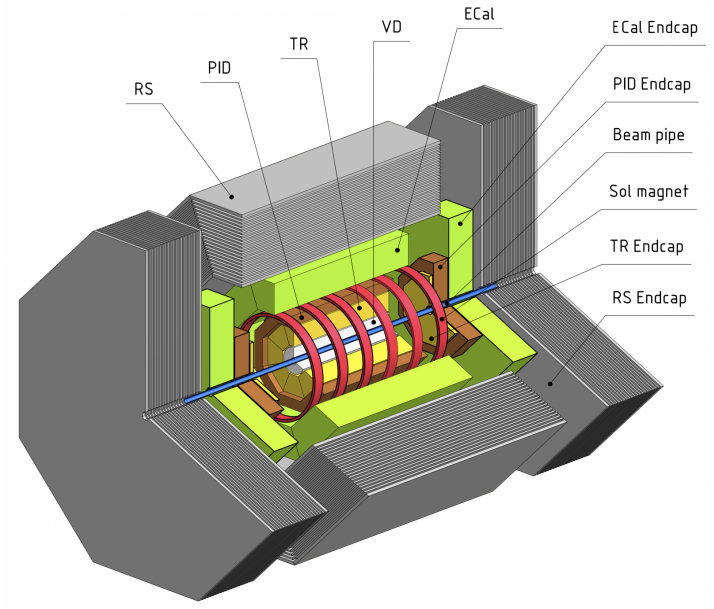
Machine learning is a key technology

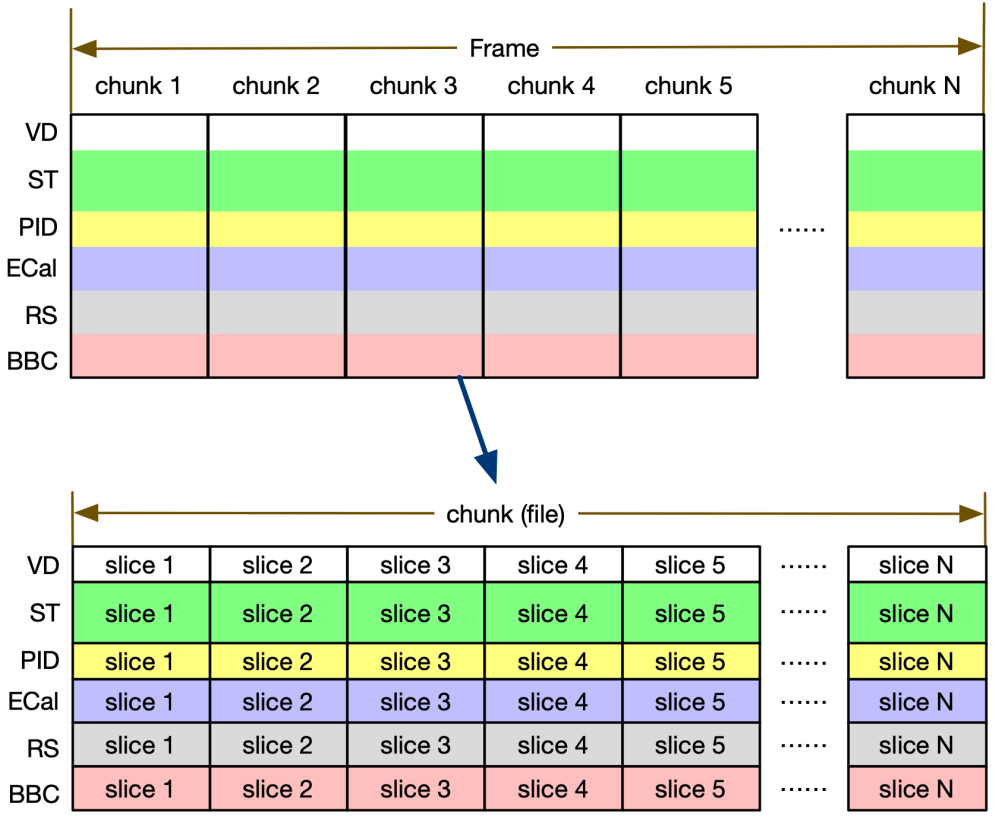
- Monitoring and Data quality assessment
- Local polarimetry

The online filter operation

Input data structure

No trigger = No classical events anymore





Input data structure

Primary data unit: **time slice** (1 us — 8.3 ms) with a timestamp

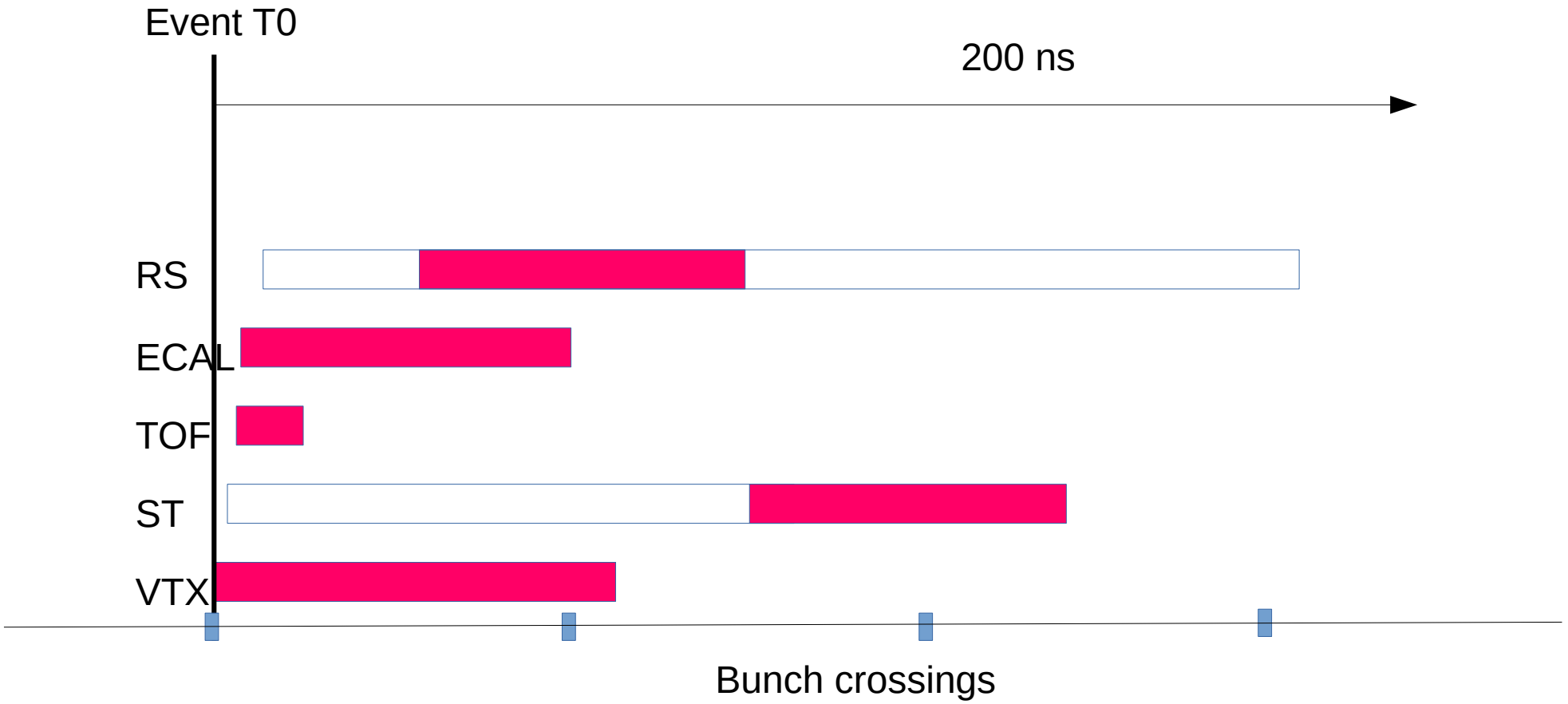
Time slices combined in **time frames** (up to 549 s, 16 GB max, < 160 MB to fullfil 20 GB/s limit)

Intermediate units — **time chunks** of 0.1-0.2 s (2-4 GB or ~10^5-10^6 events) — good to store in a separate file

Every time slice will contain signals from one or few collisions (events)

Event building have to unscramble events from a series of time slices

Time scale



The procedure

- Data unpacking
- Reconstruction
- Event unscrambling
- Event selection

The main challenge is the performance: 20 GB/s = 5 files/s. Each file (4 GB, 10^6 events, ~10^7 tracks) should be copied to a working node and processed within few minutes!

Data unpacking

- **Input:** bytestream from the DAQ
- **Output:** raw hits (channel-signal) grouped by time slices, with a timestamp

After the event selection, we are going to keep these unpacked raw data as an input for the offline reconstruction

Fast reconstruction

- Machine learning is a promising technique for this work
- We need to invent, to develop and to train neural networks for:
  - *tracking*
  - *primary vertex reconstruction*
  - *ECAL clusters*
  - *π^0 reconstruction, ZDC, BBC for online polarimetry*
  - *RS tracks and showers*
- python → C++
- Continuous monitoring of the ML reco performance is needed to keep control of systematics
- We need a simulation of time slices with raw hits (and possible pile-ups), labelled for ML training!

Classic reconstruction

- Needed to verify the ML one
- The same as ML reconstruction but using traditional algorithms
- Assume that calibration constants and alignment are not readily available
- Assume that noise level is not known *a priori*

Event unscrambling

- **Input:** several (2 or 3) consecutive time slices selected by a sliding window, with reconstructed data
- Event building is based on timing and reconstructed primary vertex position
- **Output:** event structure, consisting of a set of raw hits and reconstructed information (primary vertex, tracks, clusters)

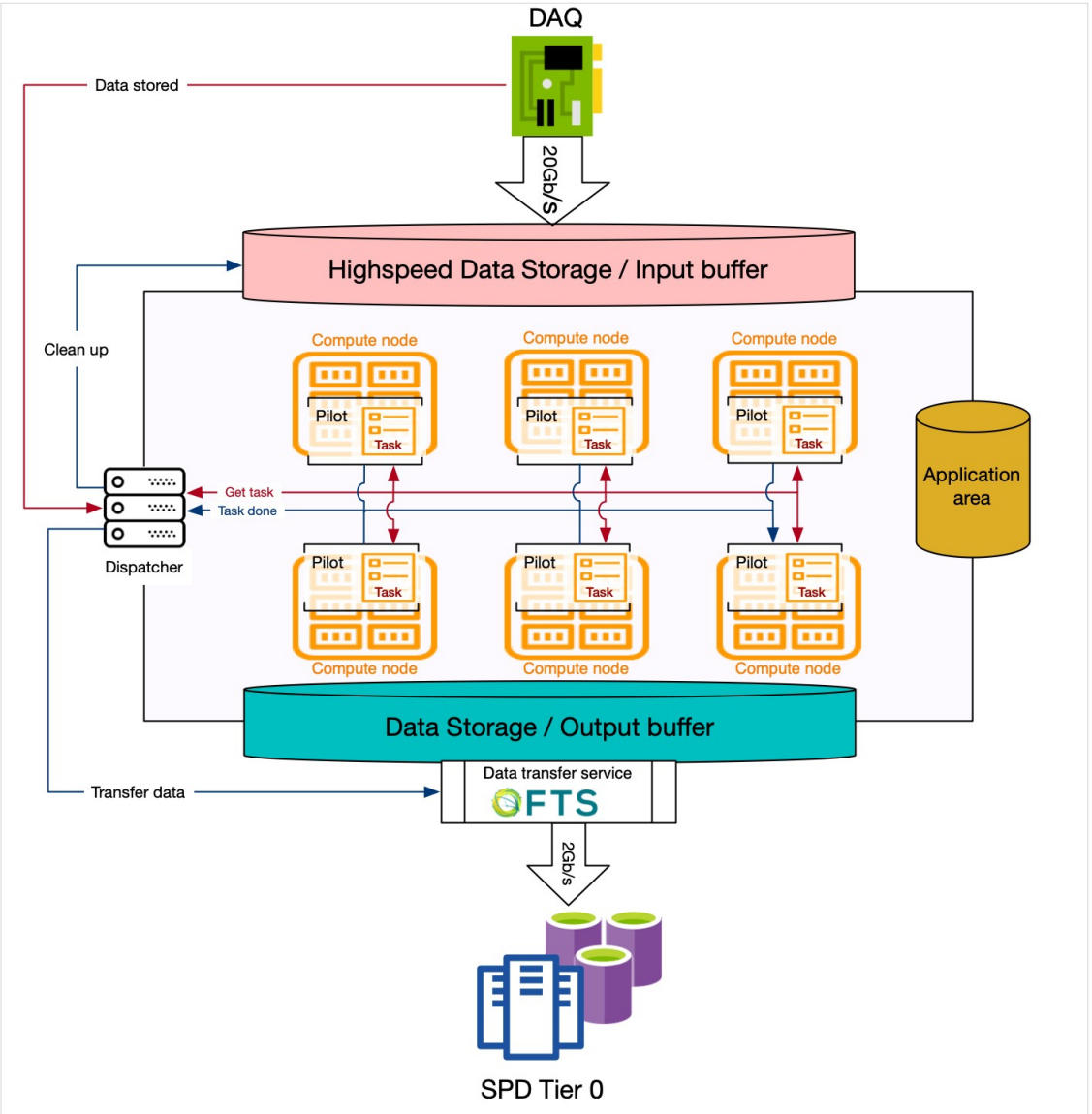
Event selection

- We need a preliminary set of physics criteria to select interesting events, and relevant pre-scale factors for the output data streams
- **The work is in progress and the result defines whether the whole concept is viable! Most important task for today.**
- Decision of the event selector is an input for the data management system (datasets, metadata)

File merging

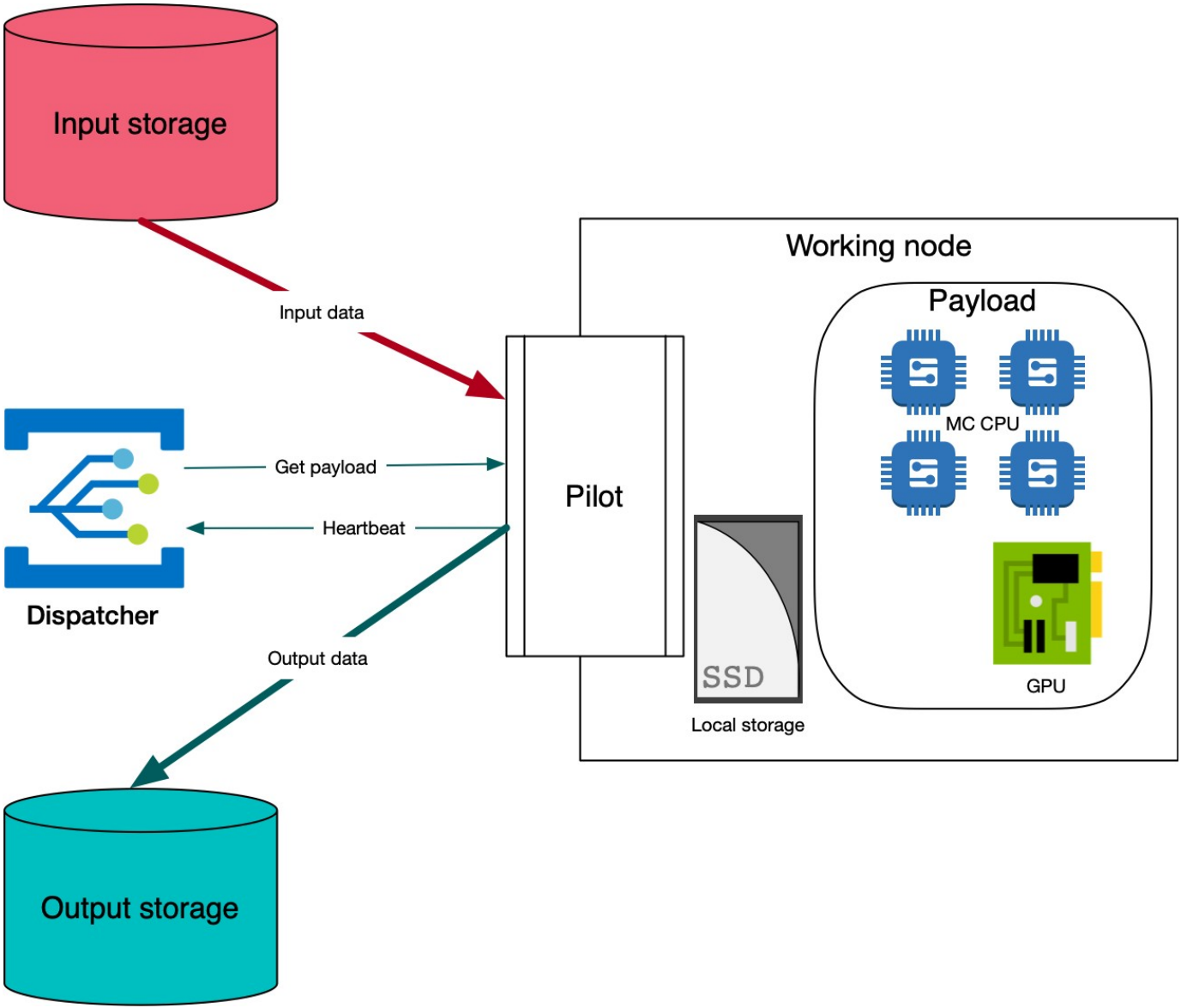
- 4 GB input file with one data chunk will reduce down to O(100 MB) file with the output data (unpacked raw data + reconstructed objects)
- Too small to handle by the offline data management system
- We need an auxillary merging task to combine 30-50 files together
- Shall we consider HDF5 as an output data format?

The online filter design


DAQ
Data stored
20Gb/s
Highspeed Data Storage / Input buffer
Clean up
Compute node
Pilot
Task
Application area
Get task
Task done
Dispatcher
Data Storage / Output buffer
Data transfer service
FTS
Transfer data
2Gb/s
SPD Tier 0

Main ingredients

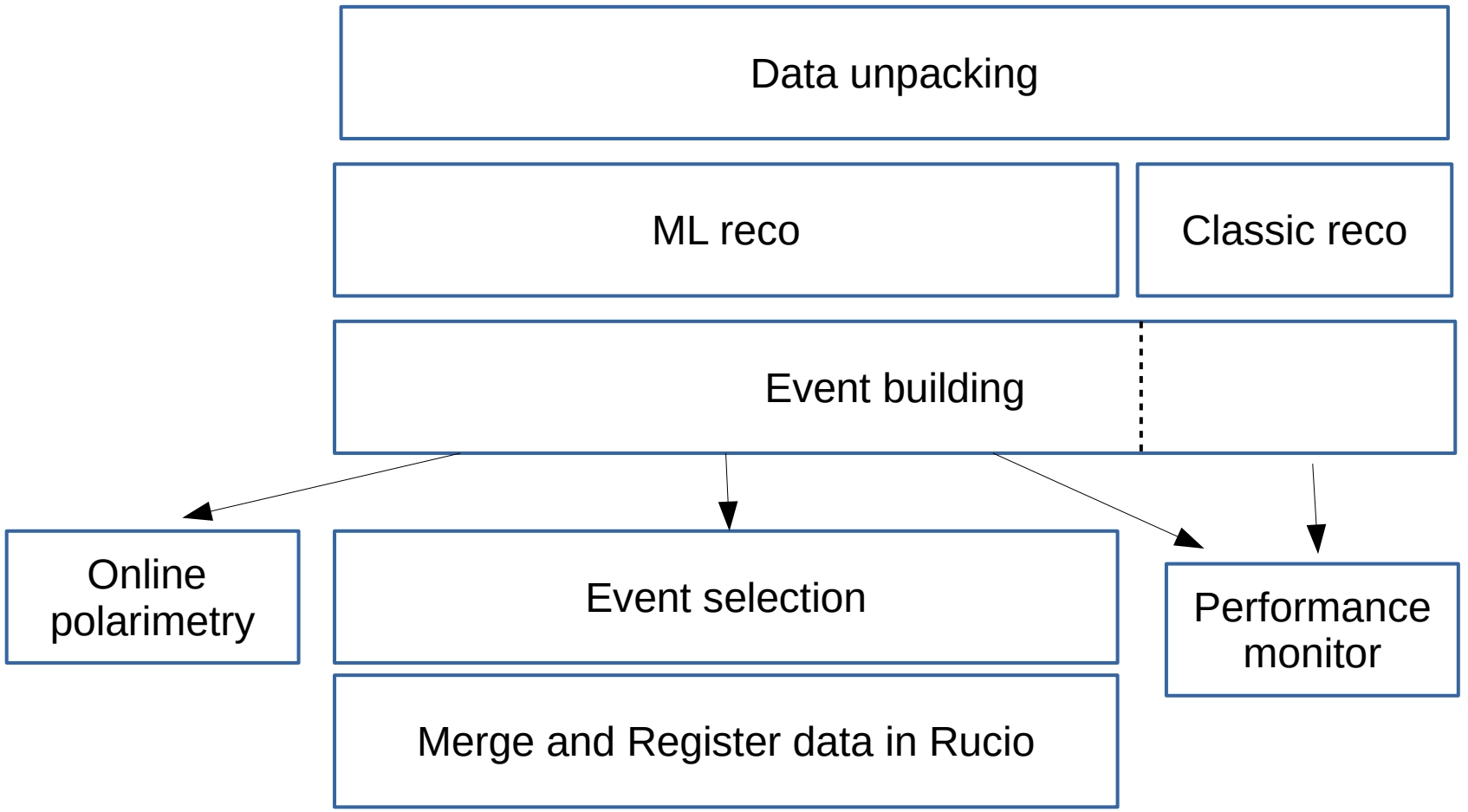
- **Input buffer:** 20 GB/s write, 20 GB/s read, delete 5 files/s
- **Output buffer:** 2x400 MB/s write, 2x400 MB/s read
- **Dispatcher**
- **Identical workers:** multicore nodes with GPUs or FPGA co-processors. 1000 or 5000 WNs ?— depends on the performance of our algorithms!
- *We should foresee using these computing resources for offline data processing between the data taking campaigns*


Input storage
Input data
Working node
Payload
MC CPU
Get payload
Heartbeat
Pilot
Dispatcher
Output data
SSD
Local storage
GPU
Output storage

The pilot

- Constantly runs at a WN
- Communicates with the dispatcher
- Copies data from the input buffer to the WN
- Calls the reconstruction software (ML, classic, merging — depends on the dispatcher’s instruction)
- Copies the resulting file to the output buffer

The payload


Data unpacking
ML reco
Classic reco
Event building
Online polarimetry
Event selection
Performance monitor
Merge and Register data in Rucio

Preliminary task list

- Data reading and unpacking
- Physics selection criteria and data streams, physics requirements, performance monitoring
- Dedicated MC simulation (time slices with labels for ML training, noise)
- Framework
  - Dispatcher, Pilot, Core framework, Workflows
  - ML implementation (C++)
  - Event selector, File merger, HDF5 IO, Interface to Rucio
  - Cluster simulation
- ML reconstruction
  - Tracking and Primary vertex , ECAL, RS
- Classic reconstruction (+GPU ?)
  - Tracking and Primary vertex, ECAL, RS
- Event unscrambling
- Local polarimetry, monitoring and DQA

Summary

- The online filter is a very important player for the SPD data processing
- A lot of interesting problems to be solved
- A lot of MSc/PhD theses to be prepared 
- The primary goal for 2021 is to develop the prototype of two key components:
  - ML reconstruction
  - Computing infrastructure