SPD OnLine Filter.

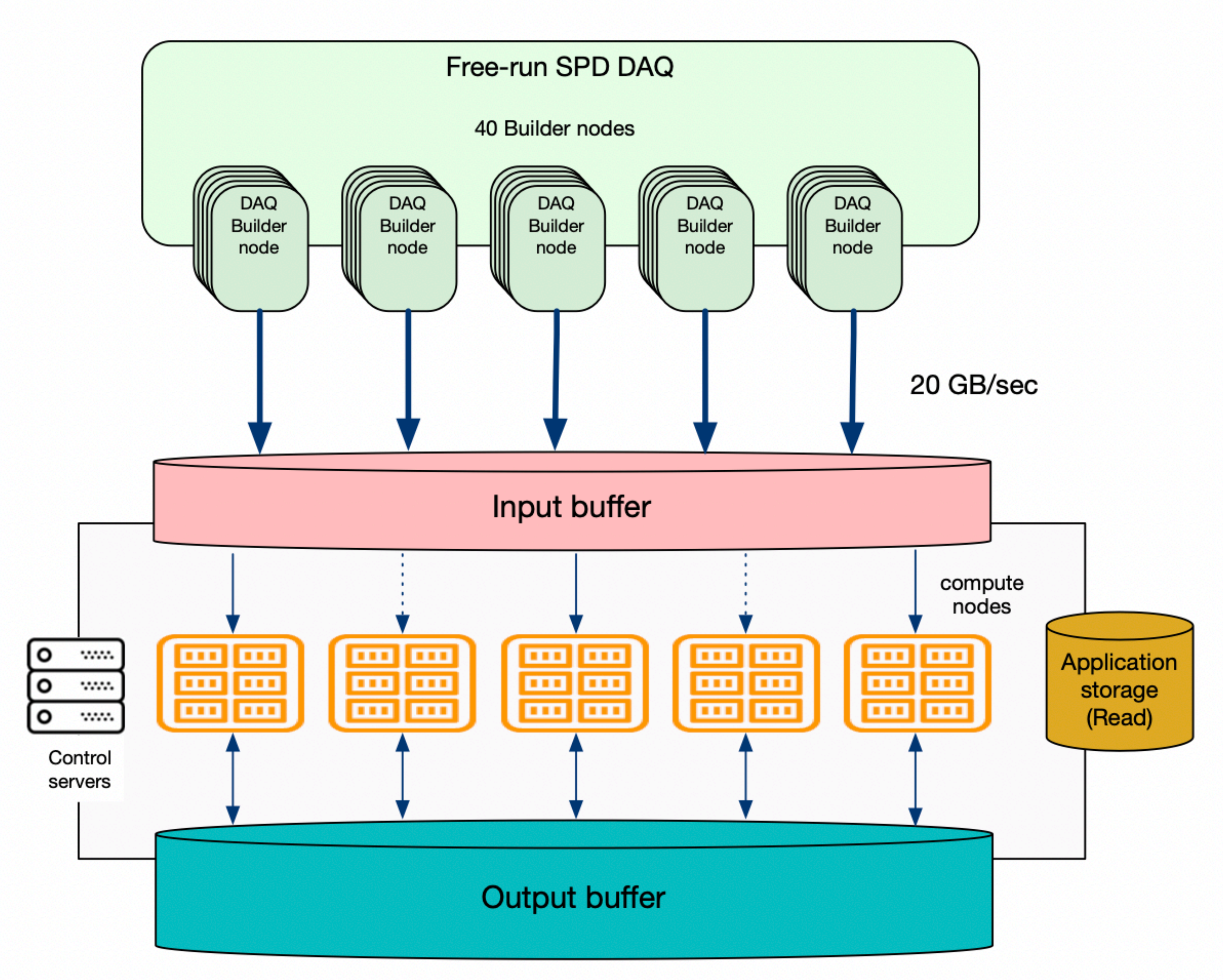
Current status and next steps.

VI SPD Collaboration Meeting. 26.10.2023
Oleynik Danila.

SPD Online filter

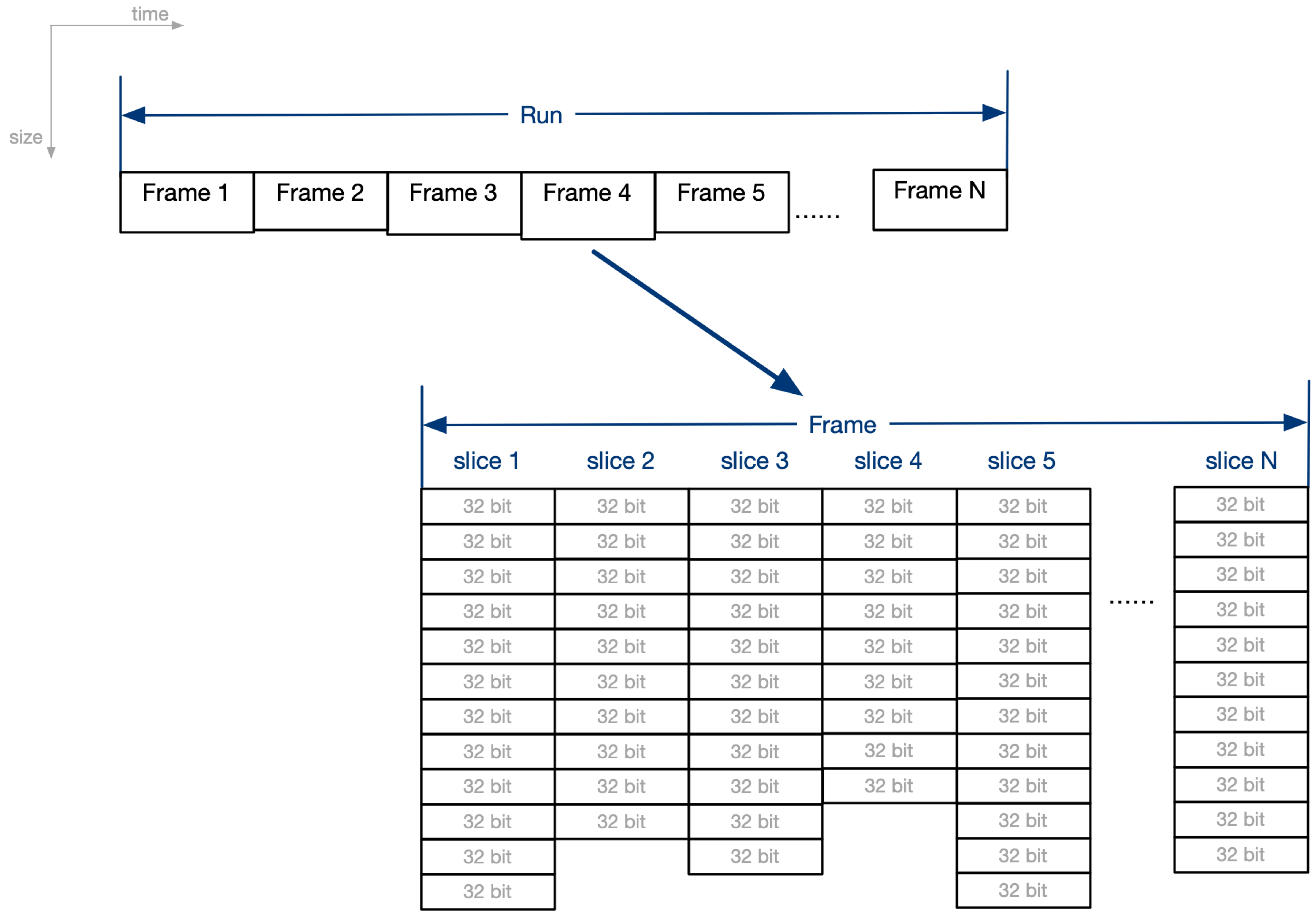
Reminder :-)

- SPD Online Filter is a high performance computing system for high throughput processing
  - High speed (parallel) storage system for input data written by DAQ.
  - Compute cluster with two types of units: multi-CPU and hybrid multi CPU + Neural network accelerators (GPU, FPGA etc.)
  - A set of dedicated servers for middleware which will manage processing workflow, monitoring and other service needs.
  - Buffer for intermediate output and for data prepared for transfer to long-term storage and future processing.

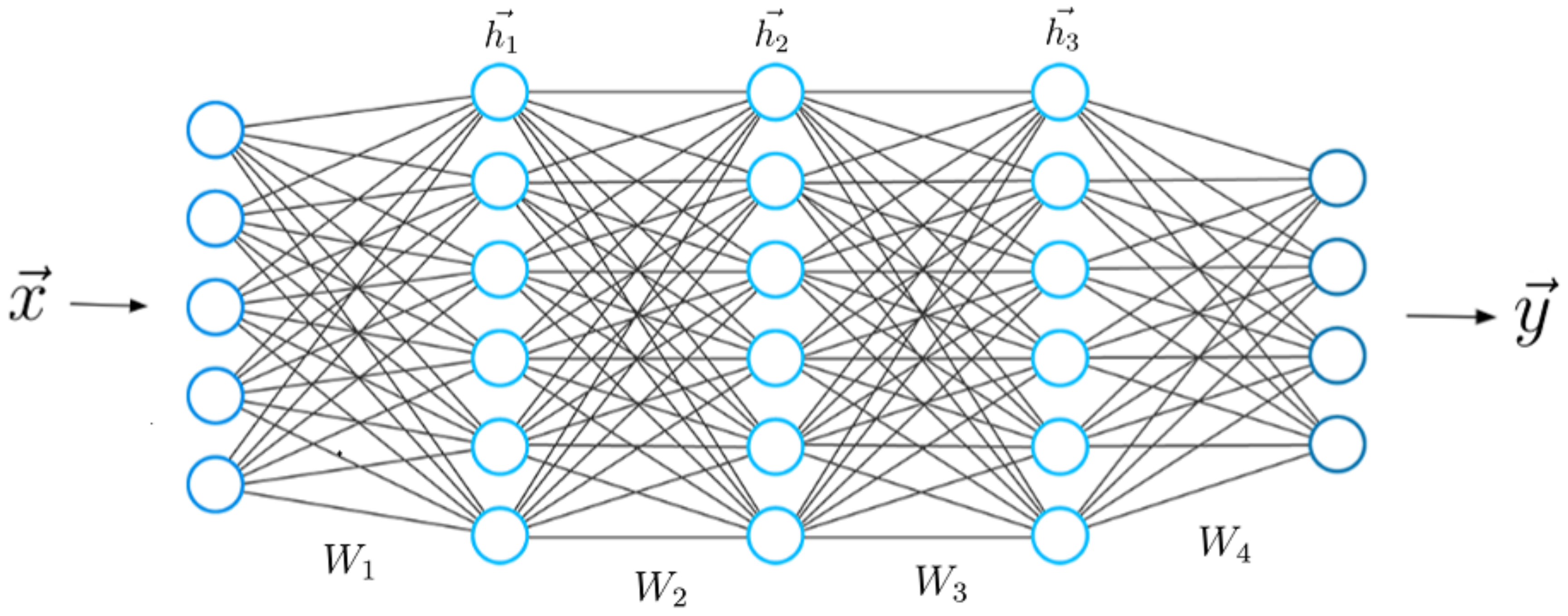


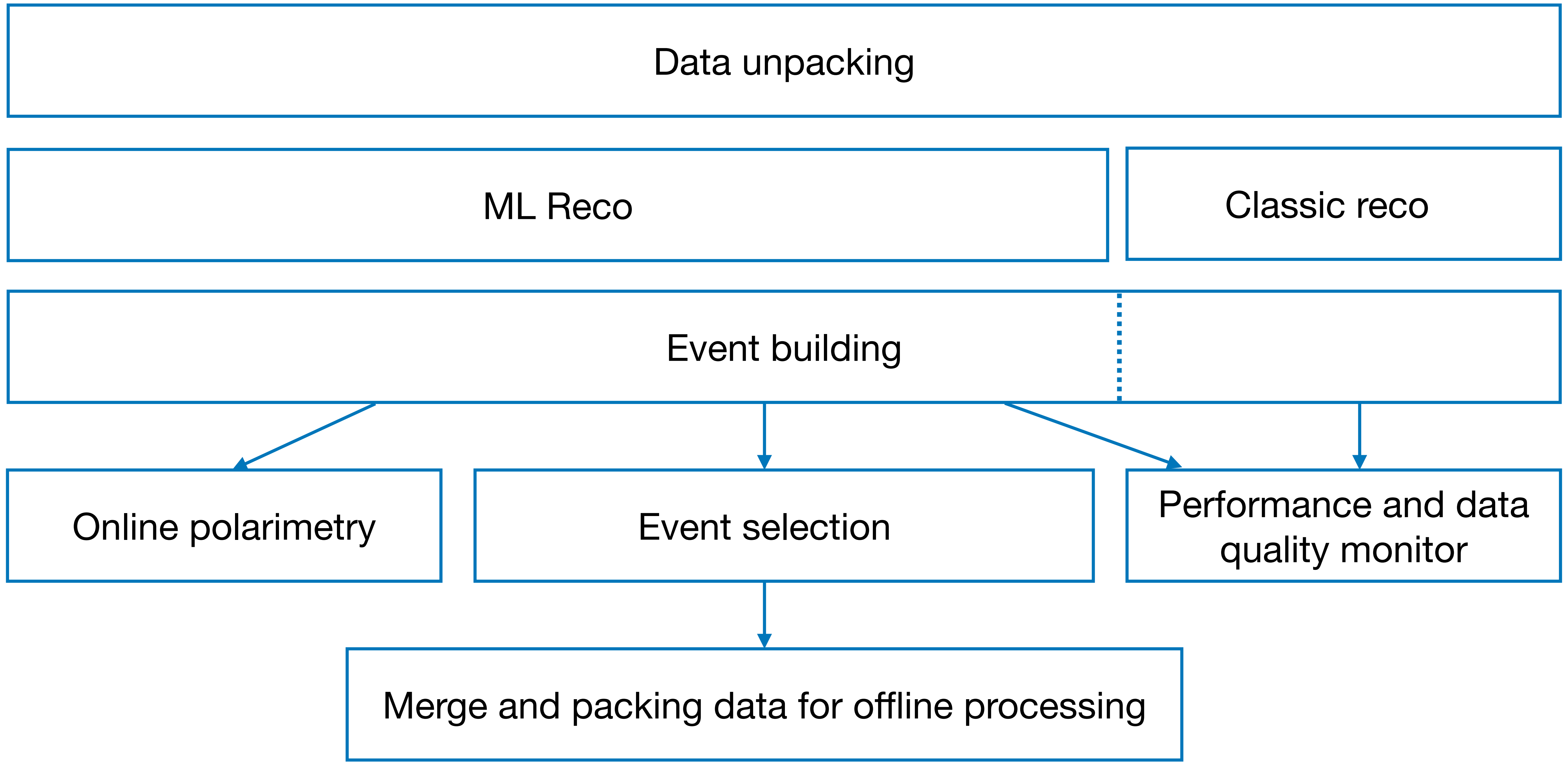
Initial data

- Free run DAQ, means that the output of the system will not be a dataset of raw events, but a set of signals from detectors organized in time slices
- Primary data unit: time slice (~10 μs)
  Time slices combined in time frames (1-10 sec.)
- Every slice will contain signals from a few to many collisions (events)
- Event building have to unscramble events from a series of time slices



Base payload





Data unpacking

ML Reco

Classic reco

Event building

Online polarimetry

Event selection

Performance and data quality monitor

Merge and packing data for offline processing

Online filter

Software part

- **Middleware** - software complex for management of multistep data processing and efficient loading (usage) of computing facility
  - Workflow management
  - Data management
  - Workload management
- **Applied software** - performs actual data processing
  - Framework - responsible for unified algorithm interfaces, IO, multithreading etc.
  - Algorithms - responsible for a single pieces of processing

Middleware

Data management;

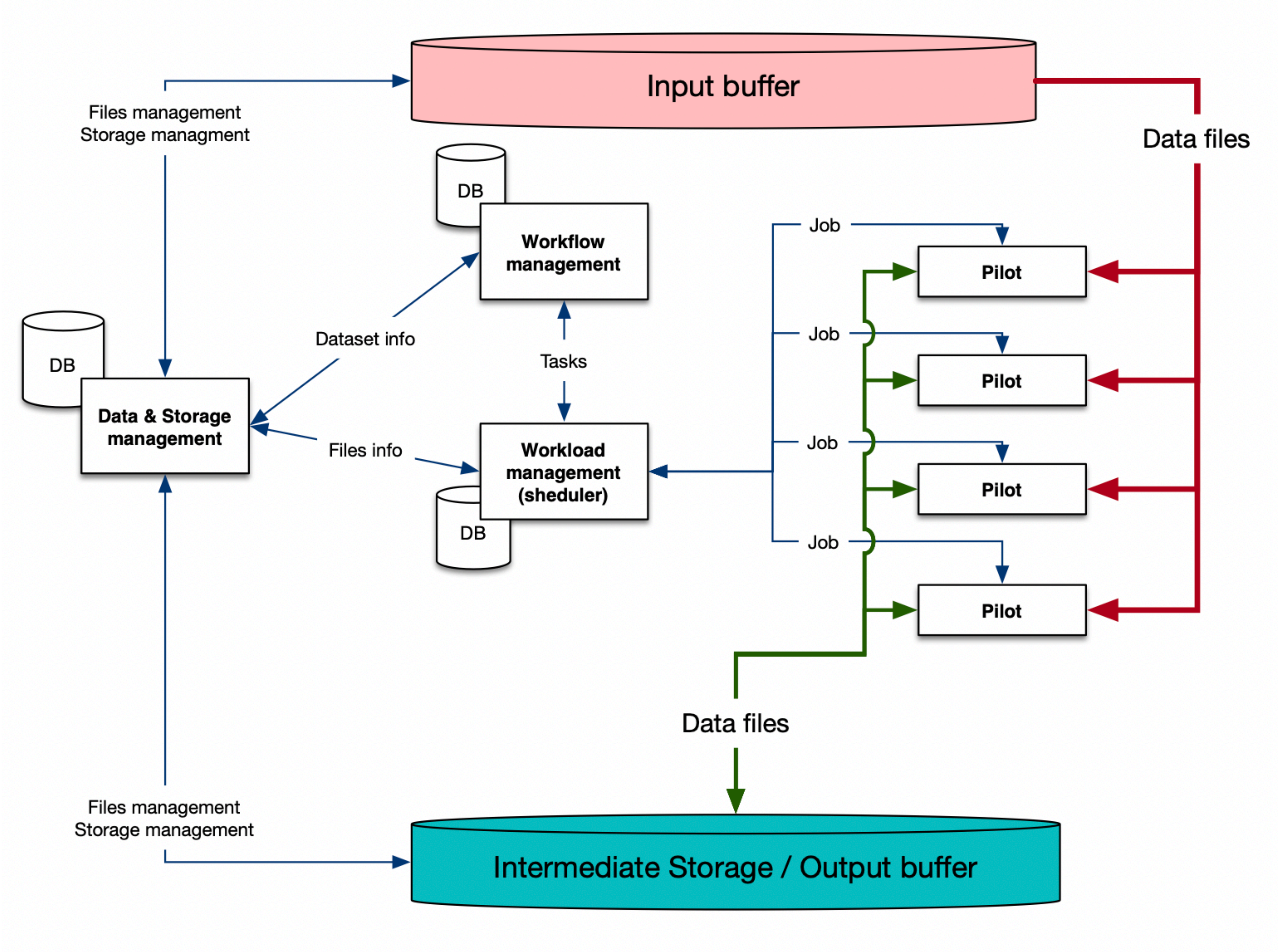
- *Support of data life-cycle and storage usage;*

Workflow management;

- *Definition of processing chains;*
- *Realisation of processing chains as set of computations tasks;*
- *Management of tasks execution;*

Workload management:

- *Generation of required number of processing jobs for performing of task;*
- *Control of jobs executions through pilots, which works on compute nodes;*



Middleware

Current status

- Each subsystem were engineered and partially prototyped
- Microservice architecture with domain driven design was chosen
  - Flexibility, scalability, easy for long-term support
- Data management
  - dsm-register – responsible for registration of input data from DAQ in the catalogue
  - dsm-manager – realise interfaces to the catalogue for subsystems
  - dsm-inspector – realise auxiliary tools for storage management (consistency check, cleanup, dark data identification)

Middleware

Current status 2

- Workflow management
  - “Chain definer” - user oriented application which allow define sequences of processing steps
  - “Processing starter” - microservice responsible for triggering of processing chains
  - “Chain executor” - microservice responsible for control of execution of processing chain
- Workload management
  - Realize a task execution process by shredding a required number of jobs to provide controlled loading to compute facility, tacking into account priority of tasks and associated jobs. A task is one step in a processing chain of a block of data. Job is a processing of a single piece of data (file or few files).
  - Microservices: task manager, task executor, job manager, job executor
  - Base architecture and initial functionality of pilot application is defined. It is a multithread application with interactions between threads through queues

SPD Online filter middleware

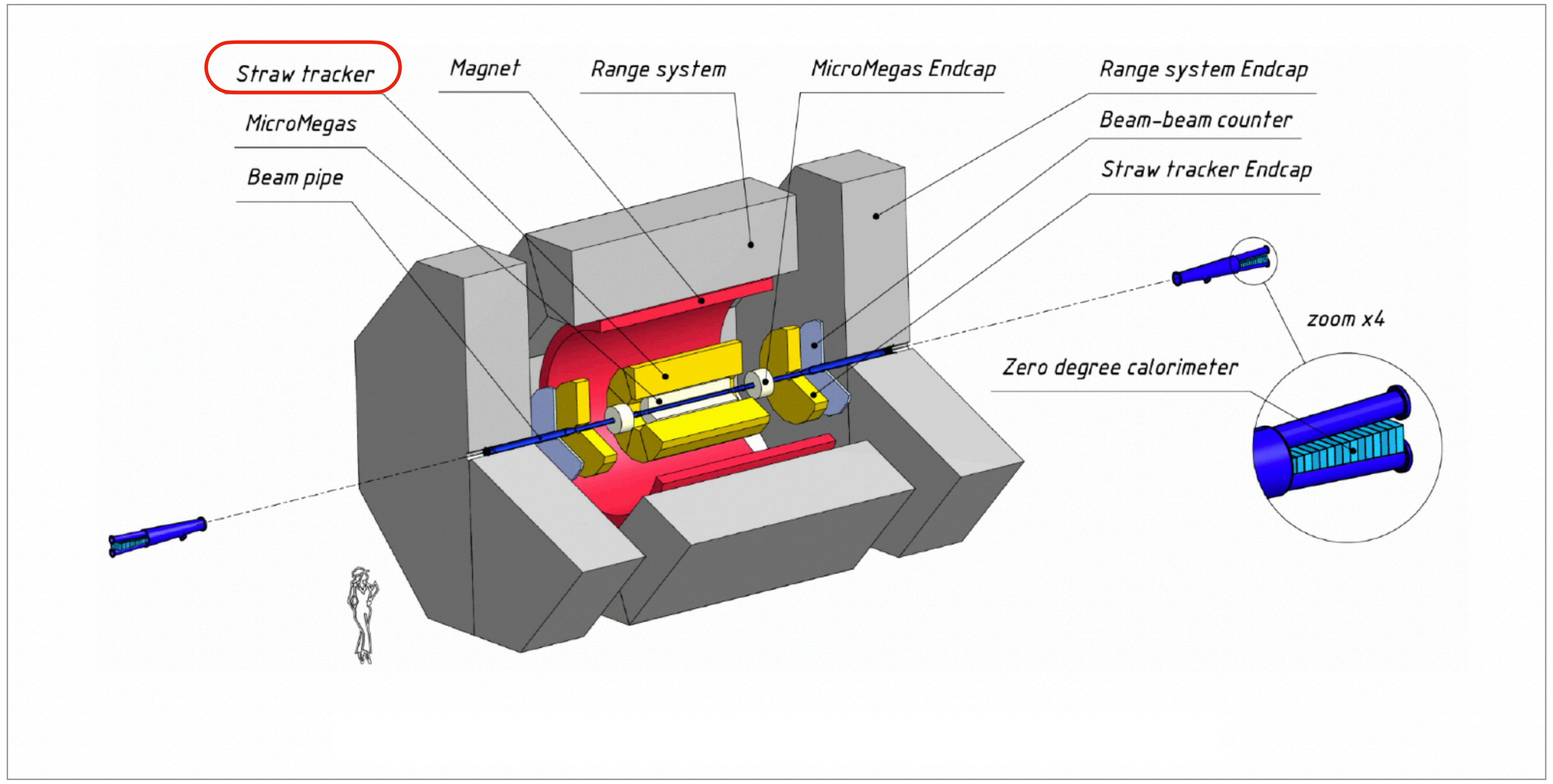
next steps

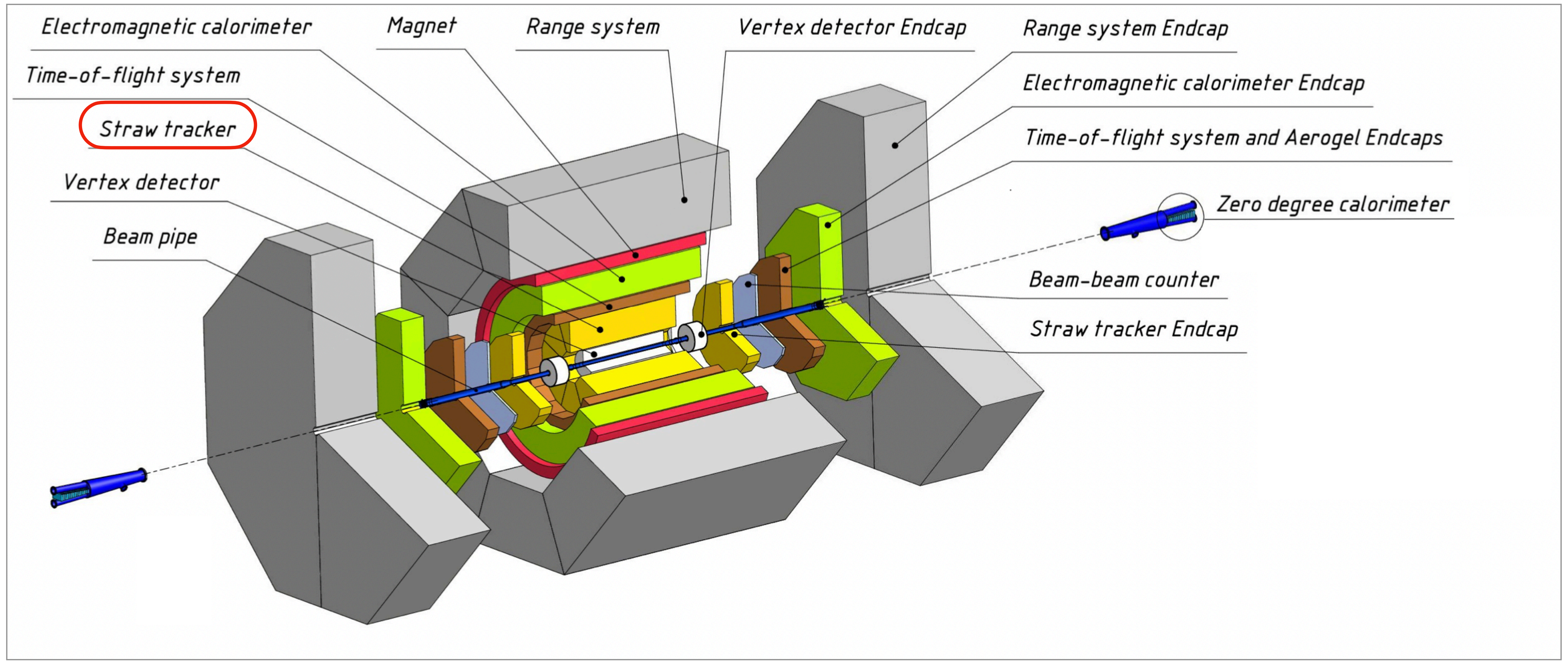
- Manpower: two PHD students, one full time researcher
  - A couple of master students recently joined
- Development, testing, integration infrastructure
- Deployment procedures etc.
- Integration with applied software (framework)

Event unscrambling

For each time slice

- Reconstruct tracks and associate them with vertices
- Determine bunch crossing time for each vertex
- Associate ECAL and RS hits with each vertex (by timestamp)
- Attach unassociated tracker hits in a selected time window according to bunch crossing time
- Attach raw data from other subdetectors according to bunch crossing time
- Name the block of information associated with each vertex an event
- Store reconstructed events





Debugging requirements

- Initial testing:
  - Agreed interfaces and data formats
  - Simplified simulated data: properly packed “white noise”
  - Low amount of data (<<0,1% of expected average)
- Functional testing:
  - Simulated data partially close to real data, which will allows debugging of some algorithms, and some workflows
  - Data amount (0,1 - 1% of expected average)
- Pre-production testing:
  - Simulated data of whole systems
  - Data amount (1 - 10% of expected average)

Debugging workflow

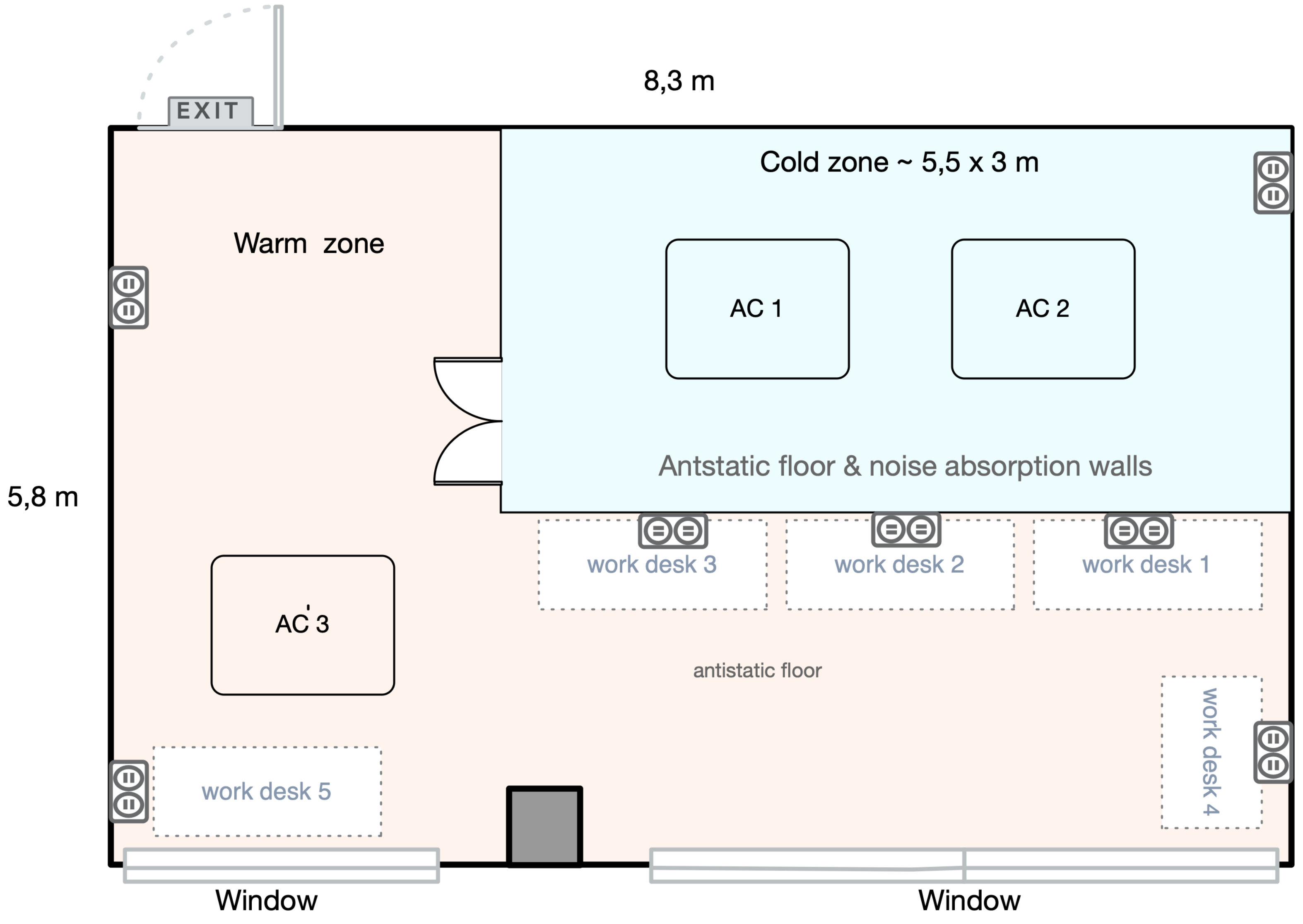
- Offline system:
  - MC production with incremental growth of simulated data
  - Agreed data organisation (to allow different types of debugging)
  - Physics group: algorithms and data production control
- Step by step improvement of Online filter prototype
  - Estimation of required set of services: software distribution and deployment
  - Formalisation of workflows
  - Subsidiary data sources: mapping, geometry etc.
  - VM to real HW (on small scale)

General plan for next 6 months

- OnLine filter
  - More attention to framework and reco. algorithms: simulated data is needed
  - MC production workflow in offline system:
    - Agreed data model and data organization
    - Data management system in place
    - MC data production policies
  - Definition and implementation of obvious data processing pipelines
- Running up of the SOF-DAQ testbed

Thank you!

DAQ & Online filter testbed in MLIT


EXIT
8,3 m
Cold zone ~ 5,5 x 3 m
Warm zone
AC 1
AC 2
Antstatic floor & noise absorption walls
5,8 m
work desk 3
work desk 2
work desk 1
AC 3
antistatic floor
work desk 4
work desk 5
Window
Window