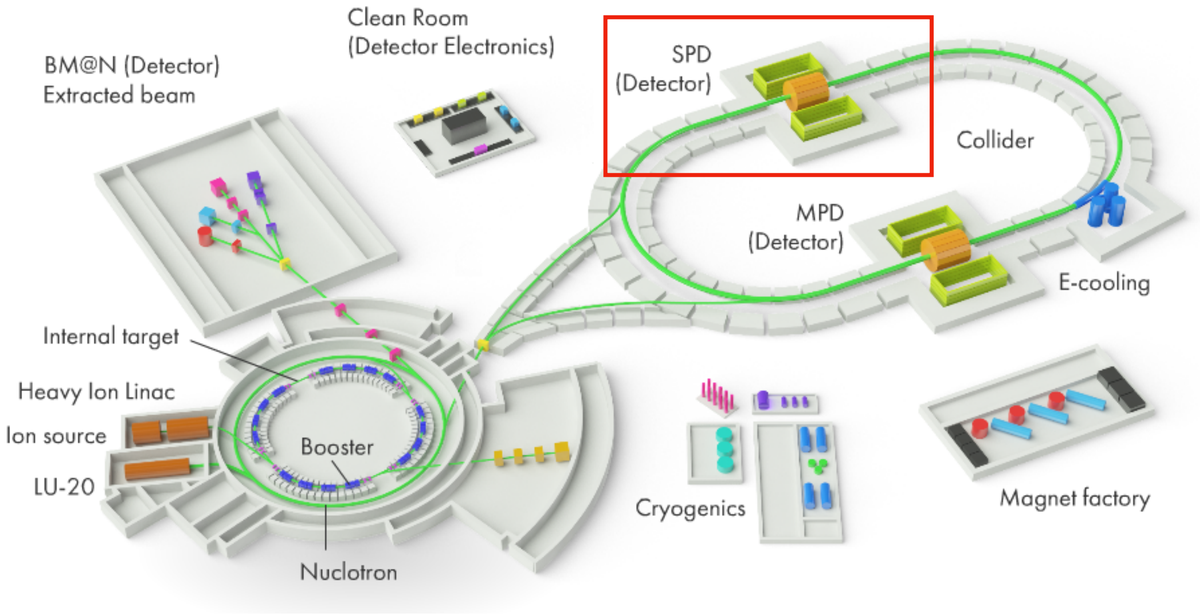
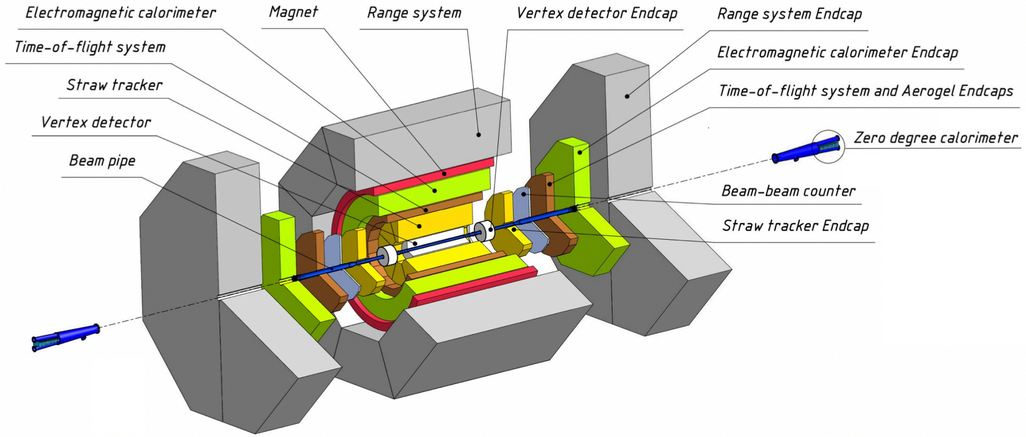
The SPD Online Filter: primary data processing facility of the SPD Experiment.

Danila Oleynik, JINR MLIT, 25.06.2024

SPD Spin Physics Detector

Study of the nucleon spin structure and spin-related phenomena in polarized *p-p*, *d-d* and *p-d* collisions





- SPD - a universal facility for comprehensive study of gluon content in proton and deuteron
- CDR presented in 2021
- TDR presented in 2023

SPD as a data source

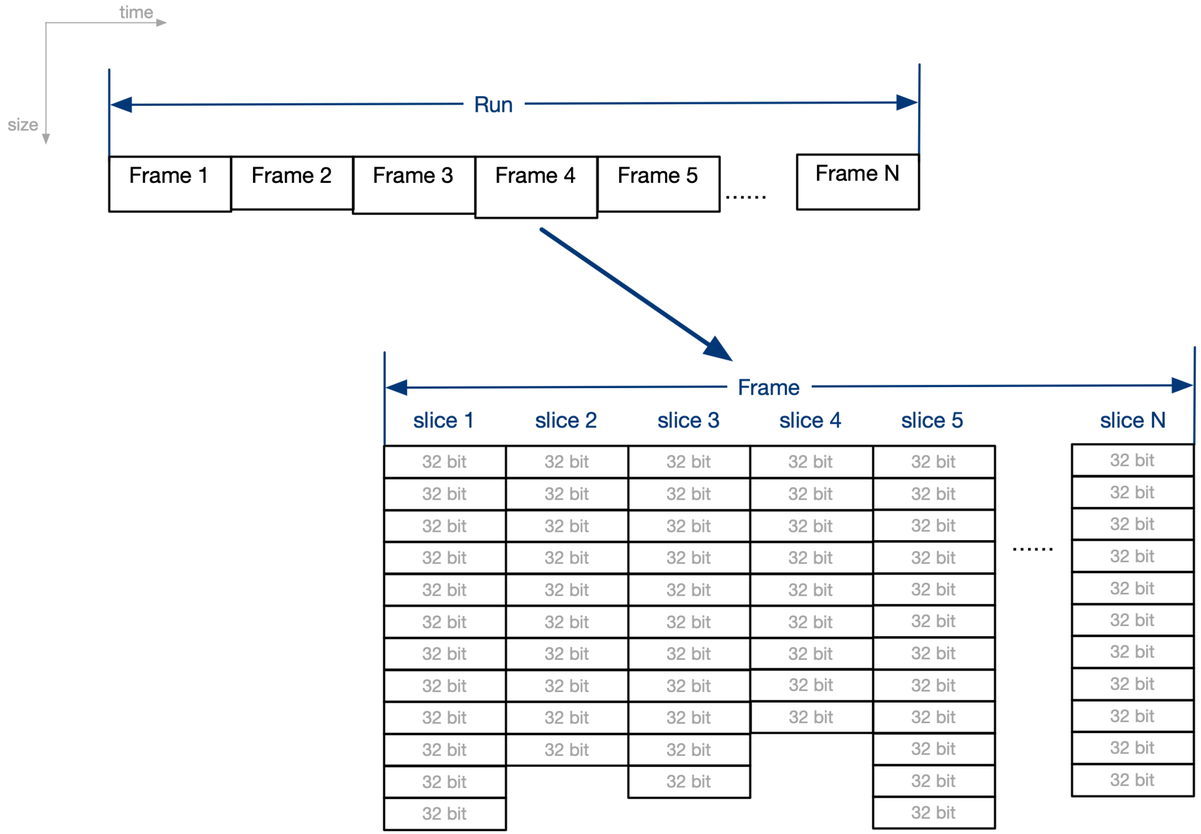
The SPD detector is a medium scale setup in size, but a large scale one in data rate!

- Bunch crossing every 76,3 ns = crossing rate 13 MHz
- ~ 3 MHz event rate (at 10^{32} $cm^{-2}s^{-1}$ design luminosity) = **pileups**
- **20 GB/s** (or **200 PB/year** "raw" data, **~3*10^{13}** events/year)
  - Selection of physics signal requires momentum and vertex reconstruction → no **simple trigger** is possible

Free run DAQ and (raw) data format

No trigger = No classical events anymore

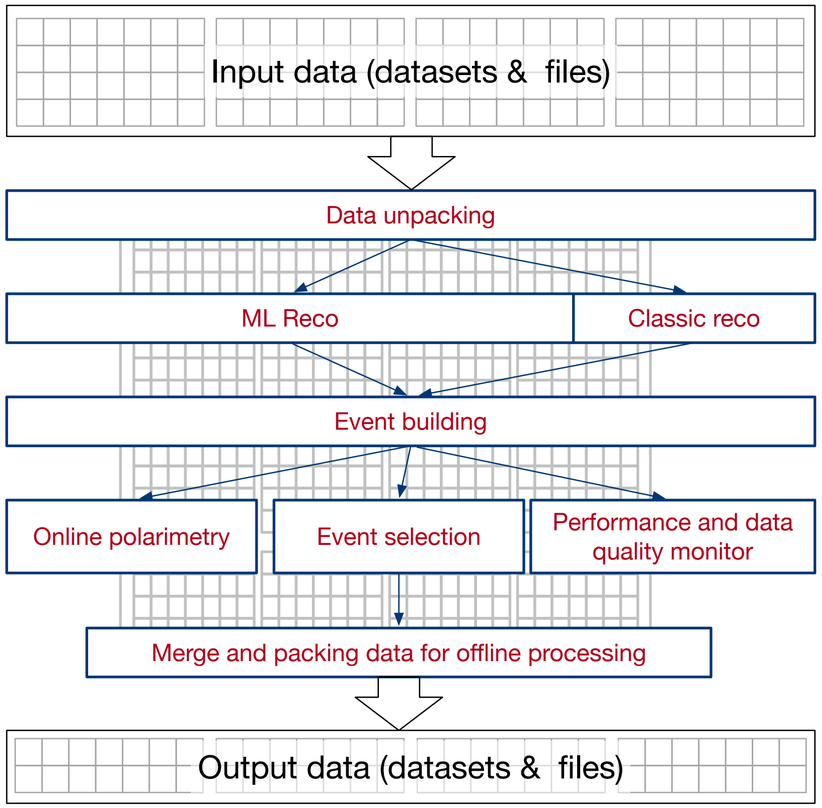
- Free run DAQ, means that the output of the system will not be a dataset of raw events, but a set of signals from detectors organized in time slices
- Primary data unit: time slice (1 μs — 8.3 ms)
  Time slices combined in time frames (up to 549 s, 16 GB max, < 160 MB to fullfil 20 GB/s limit)
- Intermediate units — time chunks of 0.1-0.2 s (2-4 GB)
- Every time slices will contain signals from a few to many collisions (events)
- Event building have to unscramble events from a series of time slices



Initial SPD data processing

- 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
    - Call the block of information associated with each vertex an event
    - Store reconstructed events
- Software trigger
  - several data streams
- Monitoring and Data quality assessment
- Local polarimetry

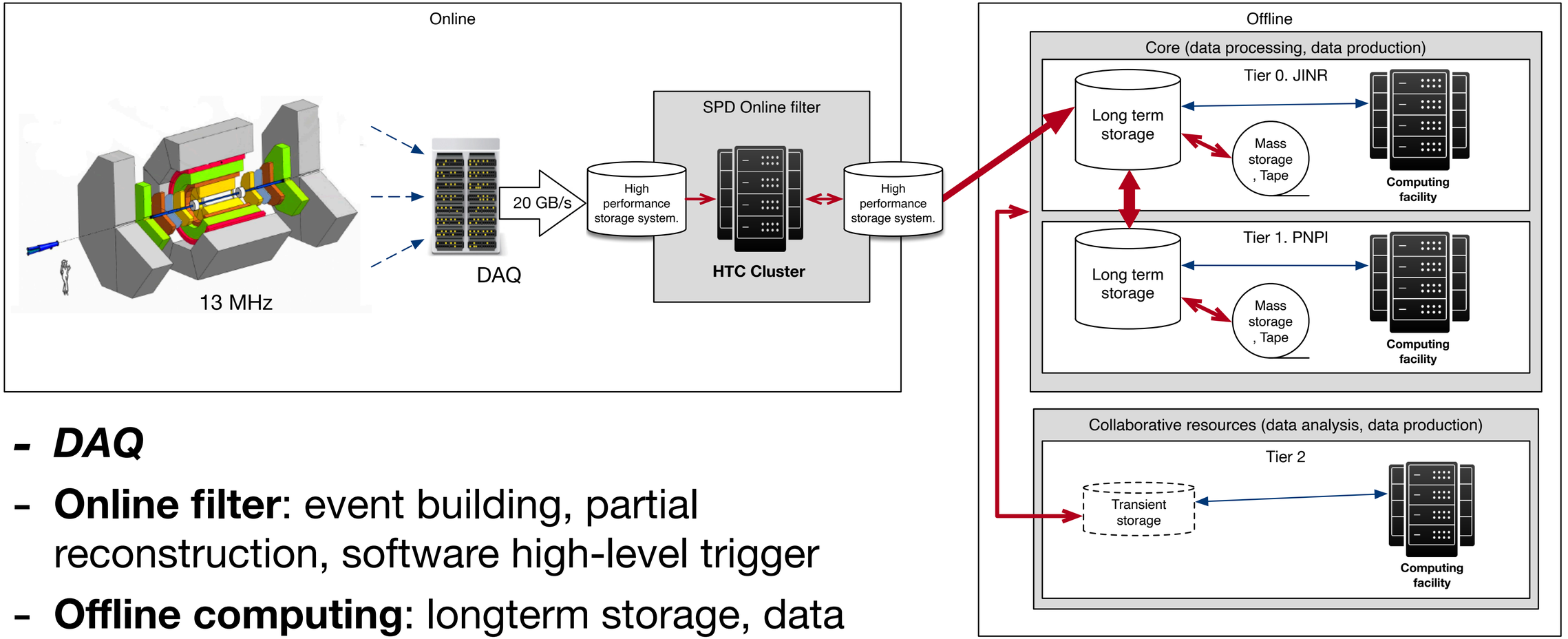
Meaning of data processing



- **DATA PROCESSING** – the execution of a systematic sequence of operations, performed with data, e.g. handling, merging, sorting, computing.
  - Note: Where data processing is performed in order to increase the value or significance (from a certain point of view) of the information conveyed by the data, it may be called **INFORMATION PROCESSING**.

THE TERMINOLOGY WORK OF IFIP (International Federation for Information Processing) AND ICC (International Computing Centre)
I. H. GOULD and G. C. TOOTILL

Data flow



- ***DAQ***
- **Online filter**: event building, partial reconstruction, software high-level trigger
- **Offline computing**: longterm storage, data production, data processing and analysis

SPD Online Filter

Online filter is the first stage in data processing chain for SPD Experiment (right after DAQ)

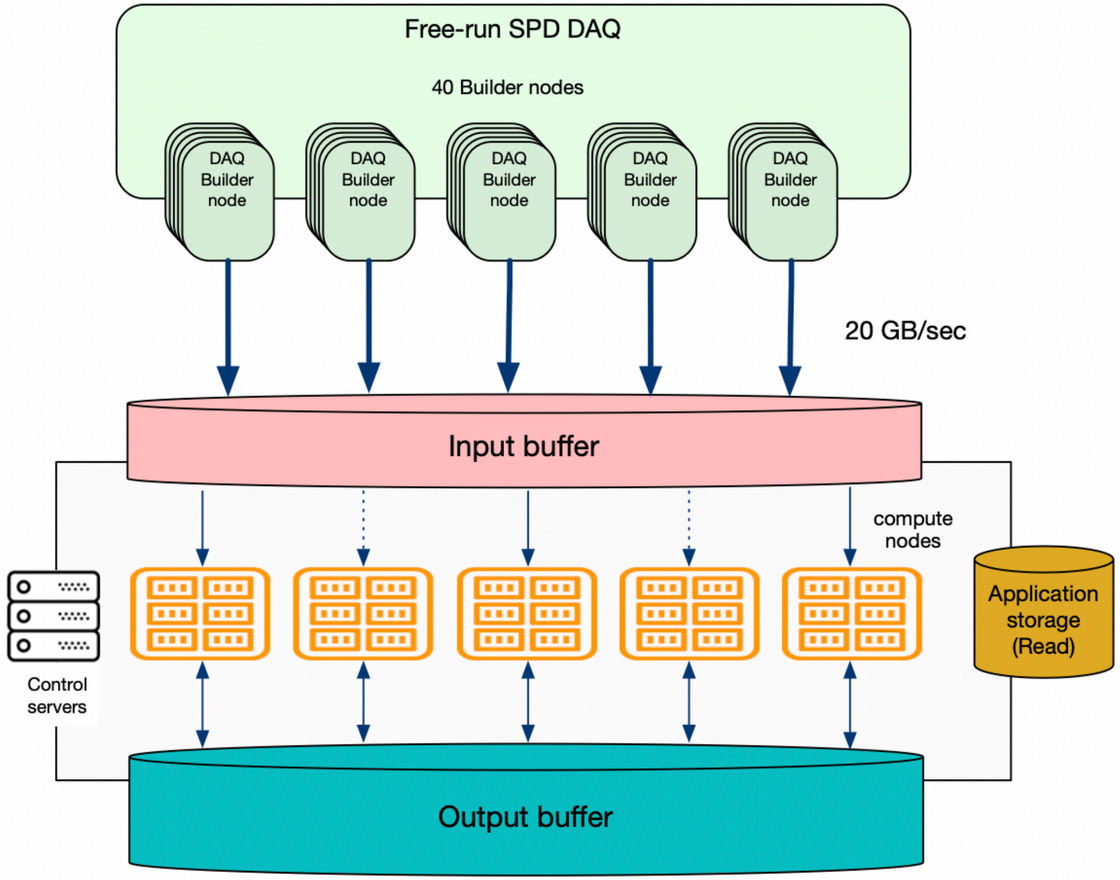
Main goals:

- Events unscrambling through partial reconstruction
- Software trigger, which essentially is event filter

- SPD Online Filter is a high performance computing system for high throughput processing
  - **Hardware component:** dedicated compute cluster
  - **Middleware component:** software complex for management of multistep data processing and efficient loading (usage) of computing facility.
  - **Applied software**: performs informational processing of data

SPD Online Filter cluster

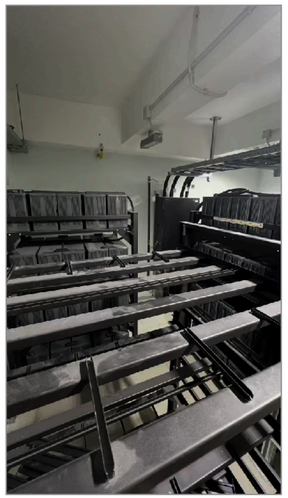
- SPD Online Filter is a high performance computing system for high throughput processing
  - High speed (parallel) storage system for input data written by DAQ.
  - Compute cluster with two types of units: multi-CPU and hybrid multi CPU + Neural network accelerators (GPU, FPGA etc.)
  - A set of dedicated 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.

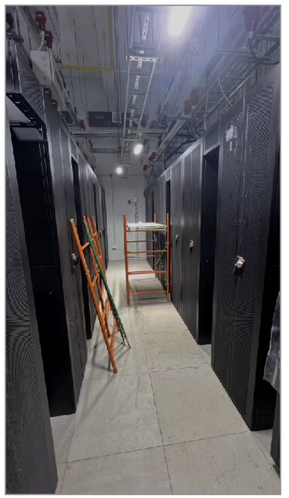


SPD Online Filter cluster prototype

Dream version

- Was expected as a part of NICA computing data center (building 14)
  - 32 Compute units -> 2816 Cores with ~10GB RAM per core
  - 2 Storage units -> 1,8 PB Storage raw volume
  - 1 GPU unit -> 8 NVIDIA H100 GPU





Compute unit	Storage unit	GPU unit	Control units
2 CPU * 44 Core (FMA3, AVX512) 1024 GB RAM 6 * 4 TB NVMe - 24 TB Ethernet: 100 Gb/s InfiniBand: 200 Gb/s	2 CPU * 44 Core (FMA3, AVX512) 512 GB RAM 32 * 30,72 TB NVMe - 983 TB InfiniBand: 2 * 200 Gb/s	2 CPU * 48 Core (FMA3, AVX512) 1024 GB RAM 8 GPU (NVIDIA H100) 4 * 4 TB NVMe - 16 TB per node InfiniBand: 2 * 200 Gb/s	2 CPU * 44 Core (FMA3, AVX512) 512 GB RAM 2 * 4 TB NVMe Ethernet: 100 Gb/s InfiniBand: 200 Gb/s

SPD Online Filter cluster prototype

Reality

- In progress purchasing of hardware for prototyping of compute cluster: 256 CPU Cores, 1TB RAM, 120TB HDD across four quite fat servers (without GPU).

Middleware components

Data management

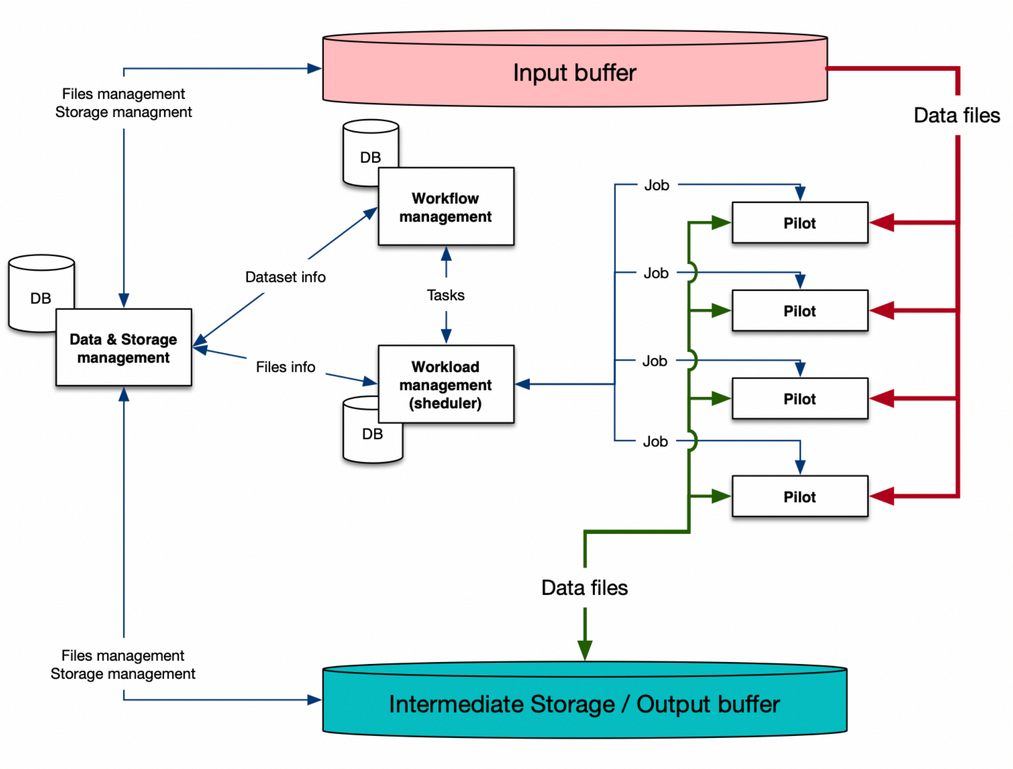
- *Support of data life-cycle and storage usage;*

Workflow management

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

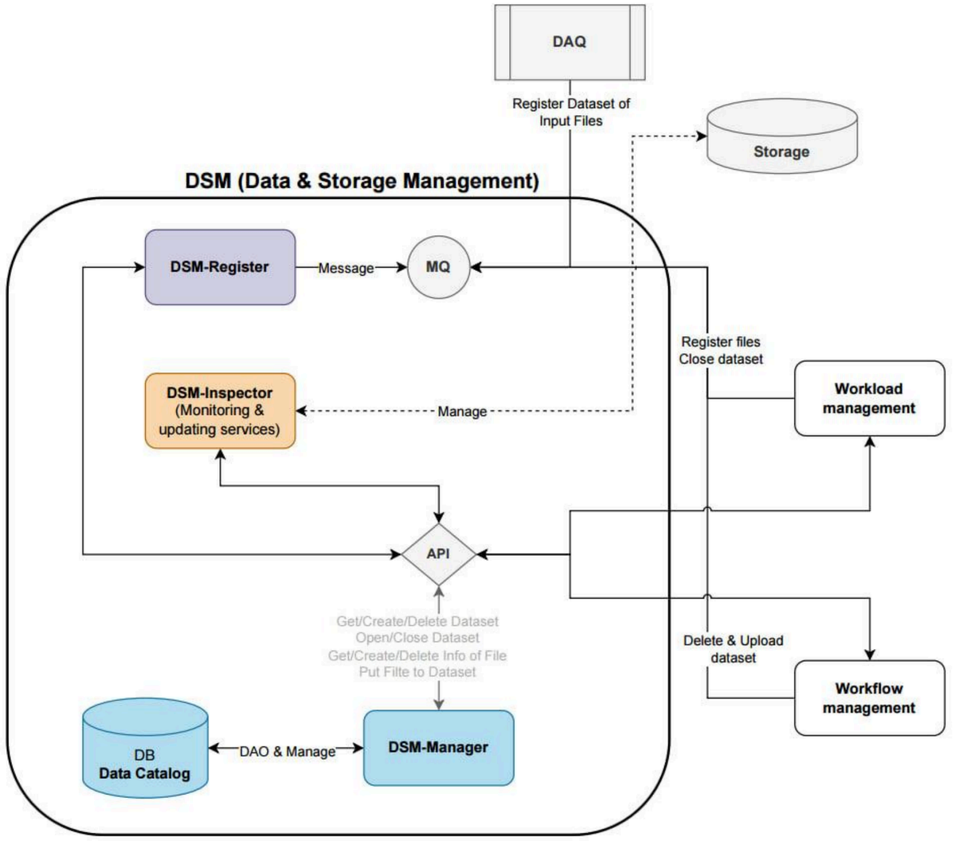
Workload management

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



Data management system

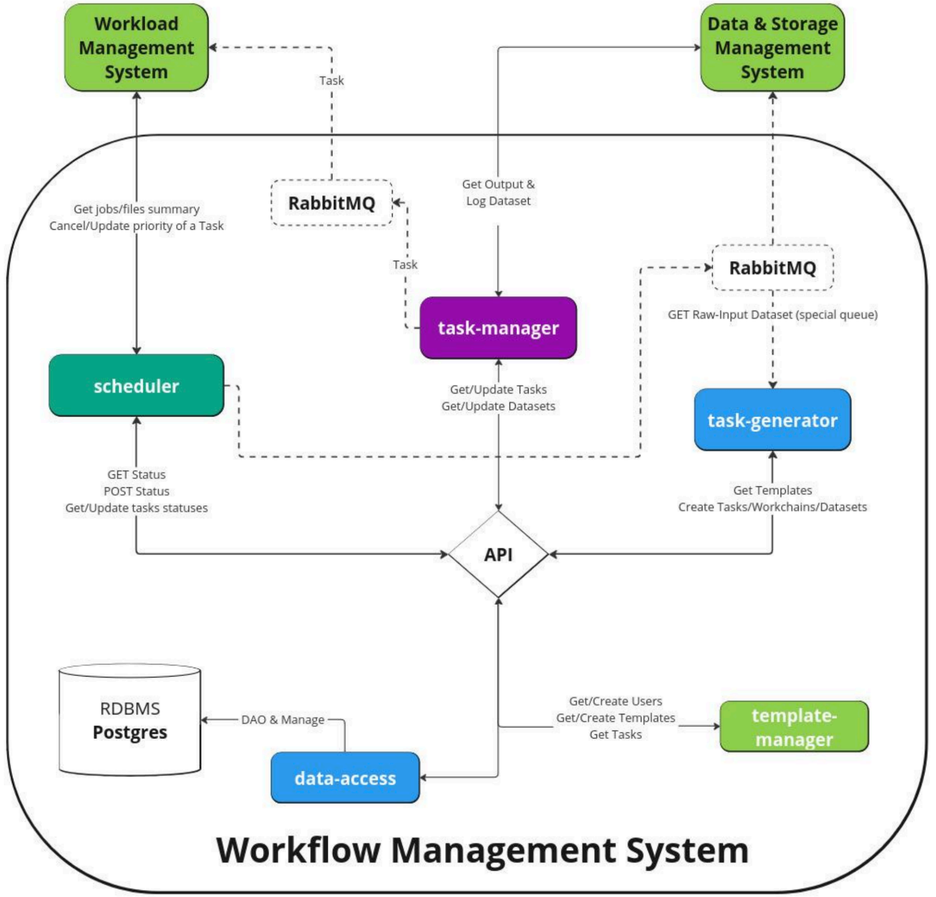
- **dsm-register (data registration)**
  - a service that receive requests for adding/ deleting data in the system asynchronously (via MQ). Then the service makes changes to the data catalog via the API of the dsm-manager
- **dsm-manager (REST API of data catalog)**
  - file and dataset catalogue management
- **dsm-inspector (daemon tasks)**
  - delete files on storage, check consistency of files, monitoring the use of storage, “dark data” identification



Architecture of Data Management System

Workflow Management System

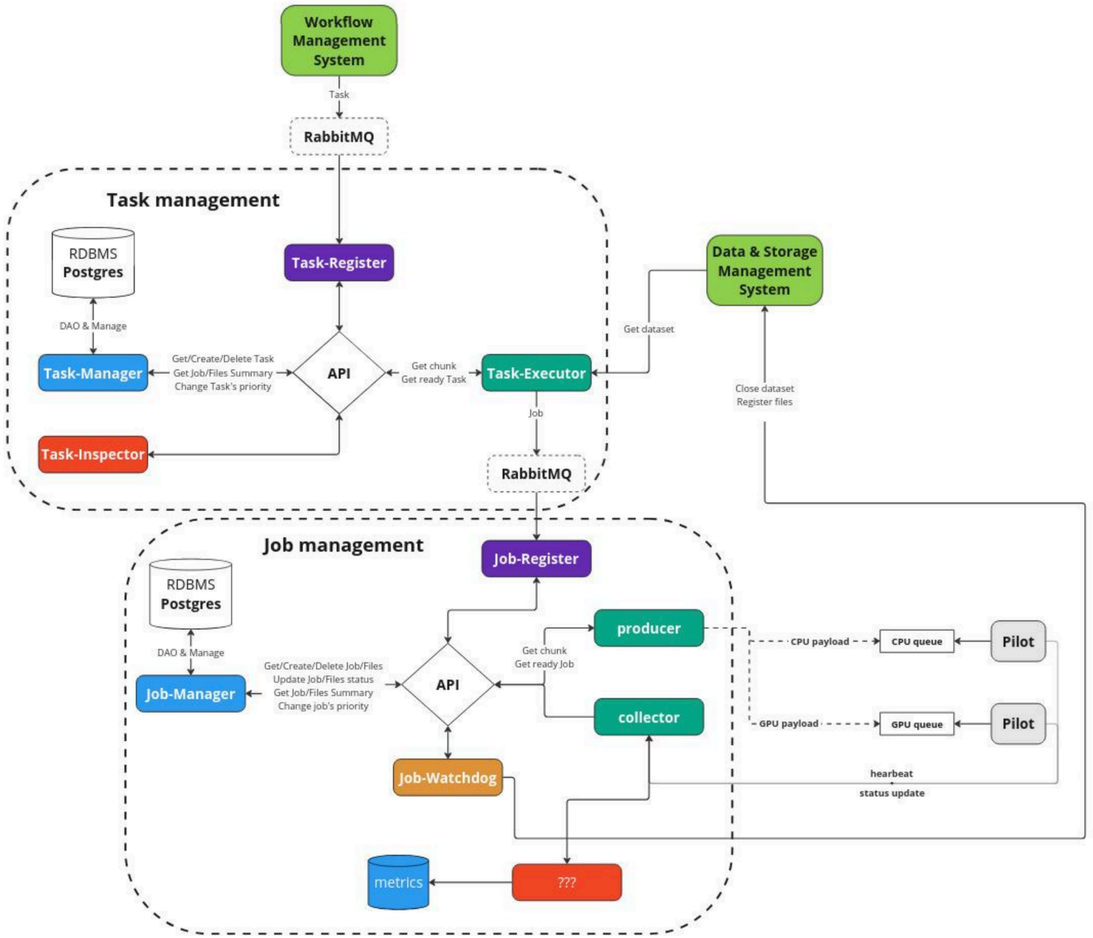
- **task-manager** – a service that requests the last dataset created in the previous step of the workflow chain, populates it, and sends the next task to the WMS.
- **task-generator** – responsible for starting the workflows based on the available templates.
- **template-manager** – service for interaction with the data processing operator/user.
- **data access** – a service that encapsulates direct database access, provides a RESTful API's through endpoints.
- **scheduler** – a services responsible for making decision on when to close datasets, cancel or change a priority of a task



Workflow Management System High-Level Architecture

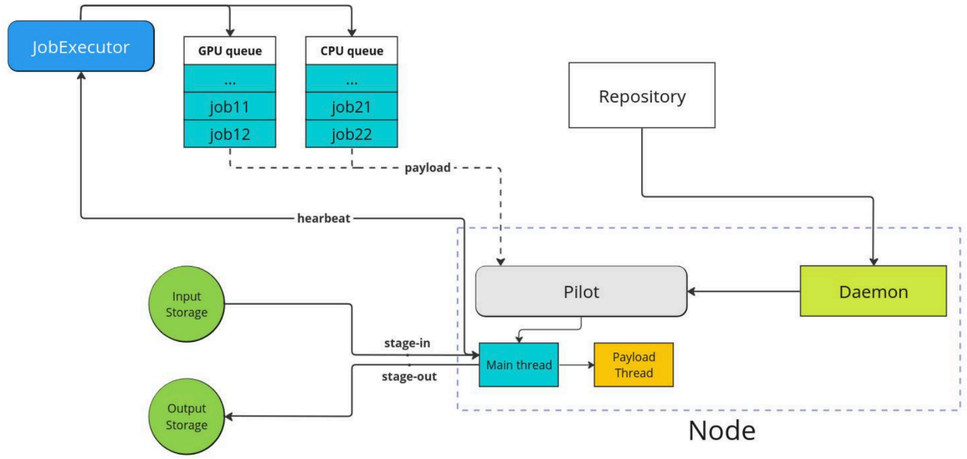
Workload Management System

- **task-manager** – implements both external and internal REST APIs. Responsible for registering tasks for processing, cancelling tasks, reporting on current output files and tasks in the system.
- **task-executor** – responsible for forming jobs in the system by dataset contents.
- **job-manager** – accountable for storing jobs and files metadata, as well as providing a REST API for the executed jobs.
- **job-executor** – responsible for distribution of jobs to pilot applications, updating the status of jobs
- **pilot** – responsible for running jobs on compute nodes, organizing their execution, and communicating various information about their progress and status



Pilot Software

- The agent application is deployed on a compute node and consists of the following two components: a UNIX daemon and the pilot itself.
- The UNIX daemon's objective is to run the next pilot by downloading an up-to-date version from the repository.
- Pilot itself is a multi-threaded Python application responsible for
  - Receiving and validating jobs from the message broker;
  - Downloading input files for the payload stage and uploading the result files to the output storage;
  - Launching a subprocess to execute a payload (job)
  - Keeping the upstream system informed of the current status of the payload and the pilot itself via heartbeat/status updates during each phase of pilot execution;



First “load testing”

- 100 concurrently running pilots
- ~2100 jobs completed in 7 min
- Pilot works for ~15 seconds
- Workload Management System generates ~5000 jobs in less than a minute
- Must be tested on meaningful data and payload, the system may not need to be over engineered more

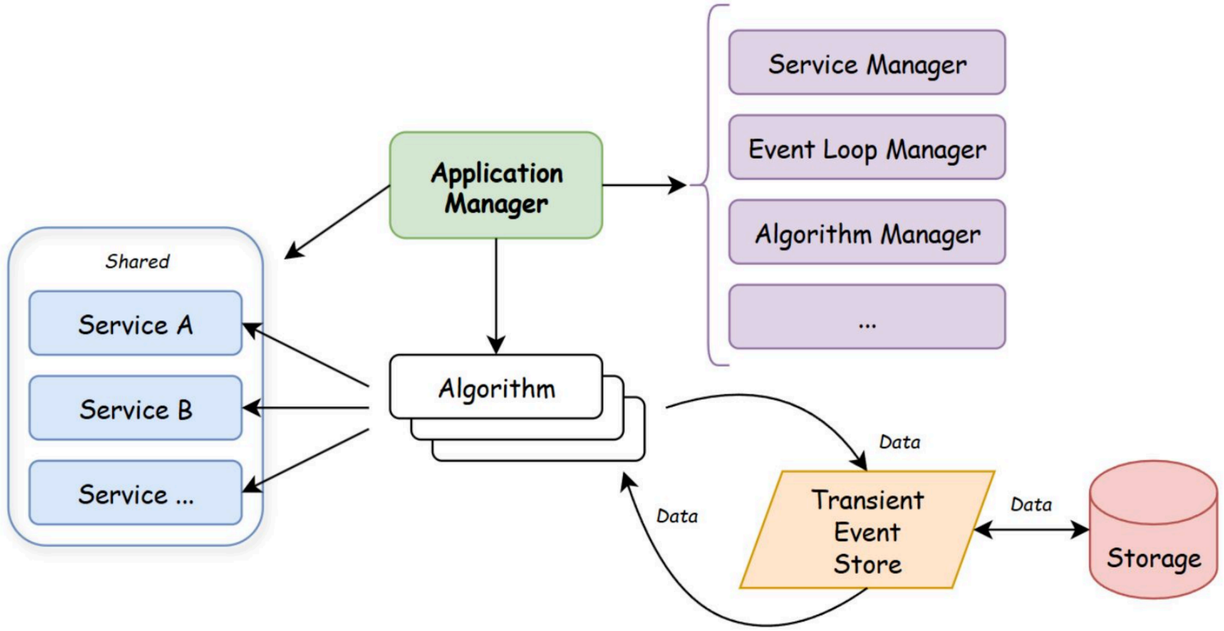
DAQ emulator

- Using SPD DAQ Data Generator software , we’ve generated 50 files, each ~2Gb;
- Input dataset has been registered with these files;
- Task has been processed (or 50 jobs);
- The payload for Pilot is simple: compute the MD5/BLAKE3 hash, as there is no actual computation involved at this stage.;
- Takes about ~7 min to generate a file, using JINR Cloud VM: 12x 1-core (Intel Xeon E5-2650)
- Registration of the entire dataset: ~10 sec

SPD applied software framework

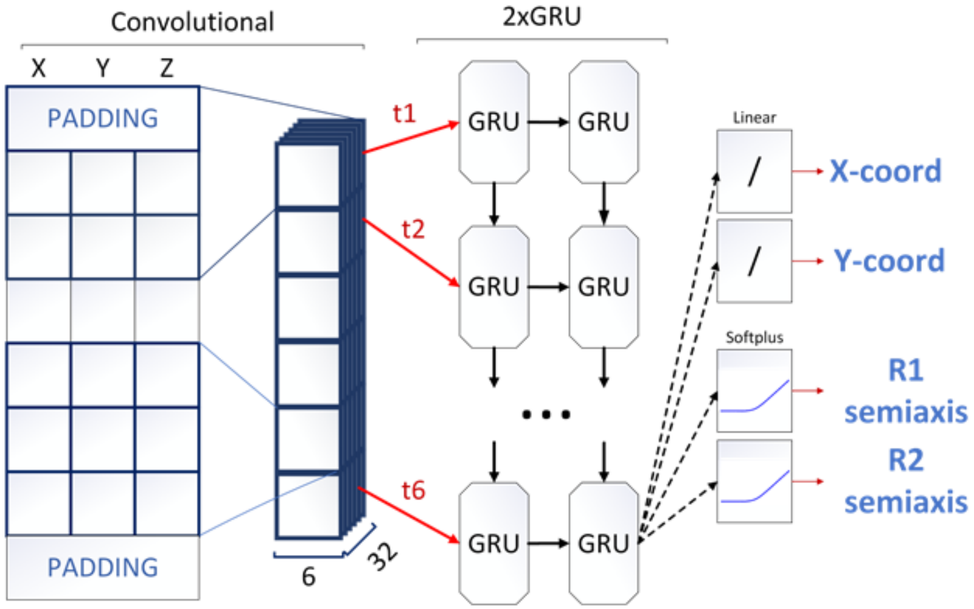
A Gaudi-based software framework is being developed.

- Gaudi architecture: **Data objects** manipulated by **Algorithms** that are launched on per-event basis. **Algorithms**, **Services** and **Tools** are dynamically configurable via **Properties** mechanism.



Machine learning in SPD Software

Example: TrackNETv3 for track recognition



- Network predicts an area at the next detector layer where to search for the track continuation
- If continuation is found the hit is added to the track candidate and the procedure repeats again
- Essentially reproduces the idea of the Kalman filter: track parameters are predicted by synaptic weights determined by network training

	Time slices of 40 events
Track efficiency (recall) (%)	96,54
Track purity (precision) (%)	94.75
Time slices / sec	63.74 (*40 = 2549.6)

Preliminary

Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz + GPU Nvidia V100 32Gb

Next steps/milestones

- Setup of hardware for prototype of facility
- Middleware deployment and release management
- Full description of initial processing chain in details
  - use it as one of inputs for applied software specification
  - specify sources of supplementary input data for proper processing (mapping, conditions etc.)
- Working for integration of ML machinery into software framework