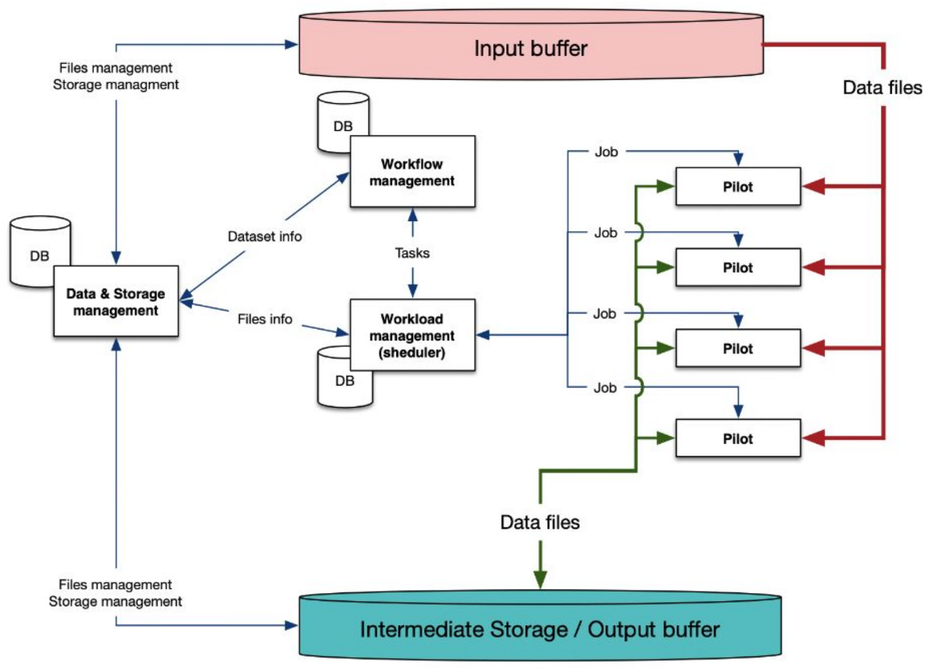
Workload Management System for SPD Online Filter

Greben N.V., Oleynik D.A., Romanychev L.R.

The SPD online filter is the special computing facility for processing of primary data collected by SPD detector. The main goal of the online filter is a fast reconstruction of the SPD events and suppression of the background events by at least a factor of 20 [1]. The special middleware is required for managing of multi-stage high throughput processing on this facility. Workload management system is one of the key components of this middleware.

«SPD ONLINE FILTER» ARCHITECTURE



SPD Online Filter – Middleware: Workflow Management; Data & Storage Management; Workload Management

Workload Management System

- Create the required number of jobs to perform the task;
- Dispatch jobs to working nodes via pilots;
- Control job execution;
- Pilot control (identification of "dead" pilots);
- Efficient resource management;

Data & Storage Management System

- Data lifecycle support (data catalog, consistency check, cleanup, storage)

Workflow Management System

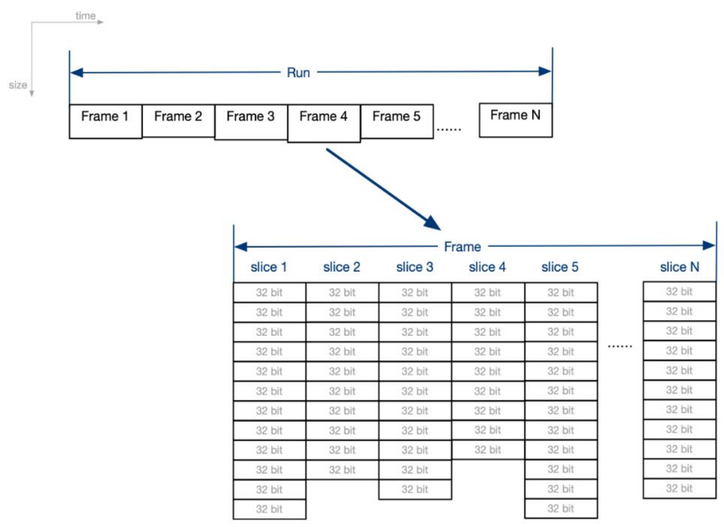
- Define and execute processing chains by generating the required number of computational tasks;

Pilot

- Determine the resource type and takes the job from the appropriate queue (GPU's or CPU's);
- Monitor and manage job execution;
- Error and job status notification;

DATA ORGANIZATION

The data acquisition (DAQ) system organizes data into slices based on time, with each slice lasting 10÷100 μs. These slices are then grouped into frames, which are sequences of slices. The frames are stored in reasonably large files (several GB). Each of these files can be processed independently as part of a chain of higher-level workflows. There is no need to exchange information during the processing of each file with physical events, the results can be used as input for the next processing step. As each data frame can be processed independently, a high throughput paradigm can be applied [2]. The complete processing cycle is a complex process and can be organized into multiple steps to make efficient use of computational resources.



File structure from "triggerless" DAQ

The system will require extensive data processing, and the overall main data flow can be described as follows: the data acquisition (DAQ) system aggregates the signals from the detectors and transfers the data to the **input buffer** as files of an agreed size; the workflow management system creates and deletes intermediate **datasets** at each processing step [3]; the workload management system **populates the datasets** with information about the resulting files; at each data processing step, the pilots read and write files to the storage, creating secondary data.

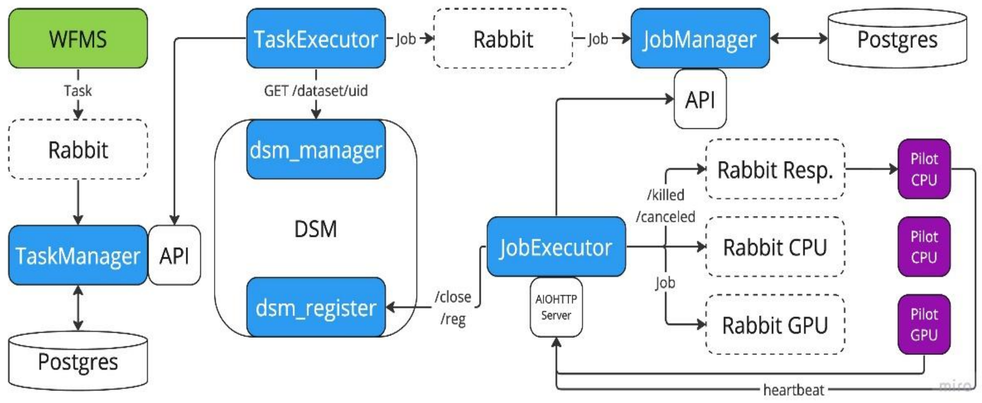
Forming jobs based on dataset contents, one file per one job

REFERENCES

1. V.M. Abazov et.al [SPD Collaboration] Conceptual design of the Spin Physics Detector // arXiv:2102.00442 [hep-ex].
2. Oleynik D. Data processing in HEP experiments [Electronic resource]. Available at: https://indico-hlit.jinr.ru/event/329/contributions/1972/attachments/575/1032/HEPNICA_computing_2022.pdf (accessed 10.12.2022)
3. SPD On-line Filter: workflow and data management systems [Electronic resource]. Available at: https://indico.jinr.ru/event/3505/contributions/21778/ (accessed 06.07.2023)
4. N. Greben, L. Romanychev, D. Oleynik, A. Degtyarev. SPD On-line Filter: Workload management system and Pilot Agent. // Physics of Particles and Nuclei.

WORKLOAD MANAGEMENT SYSTEM

- **Task** - unit of workload that is responsible for processing a block of homogeneous data.
- A processing request is a set of input data, which may consist of multiple files, and a handler (an execution file).
- **Task** completion criteria is processing the whole block of data.
- **Job** (payload) is a unit of work which process a unit of data.
- The block responsible for processing a single file in terms of workload is called a **job**.
- **Job** is an abstraction that can perform an arbitrary objective at each step of multistep processing, whether it is track recognition, data verification, event filtering, or result file merging.



SPD Workload Management System architecture

The responsibility of generating tasks, dispatching them to compute nodes and executing them lies with the Workload Management System [4]. Its objectives are as follows:

- **Task registration**: formalized task description, including job options and required metadata registration.
- **Jobs definition**: generation of required number of jobs to perform task by controlled loading of available computing resources.
- **Jobs execution management**: continuous job state monitoring by communication with pilot, job retries in case of failures, job execution termination.

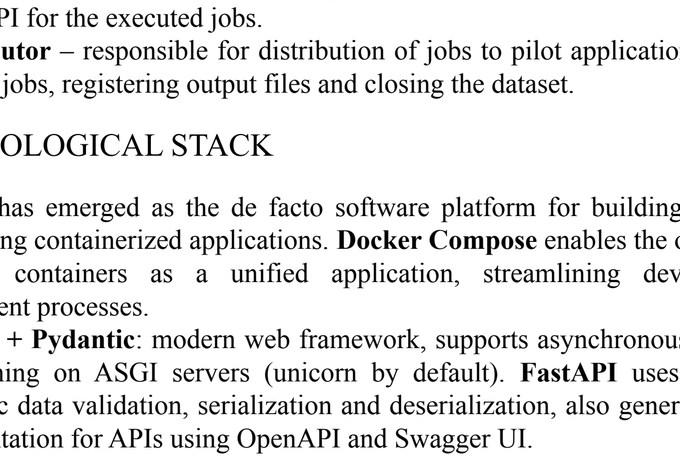
MICROSERVICES

- **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, registering output files and closing the dataset.

TECHNOLOGICAL STACK

- **Docker** has emerged as the de facto software platform for building, running, and distributing containerized applications. **Docker Compose** enables the orchestration of multiple containers as a unified application, streamlining development and deployment processes.
- **FastAPI + Pydantic**: modern web framework, supports asynchronous programming and running on ASGI servers (unicorn by default). **FastAPI** uses **Pydantic** for automatic data validation, serialization and deserialization, also generates interactive documentation for APIs using OpenAPI and Swagger UI.
- **SQLAlchemy**: a library that implements the ORM principle with relational databases, in our case PostgreSQL, using the Python language.
- **Alembic**: a Python database migration tool designed to work with **SQLAlchemy**, allowing changes to the database structure (tables, columns, etc.) via migration scripts.
- **aio-pika**: a high-performance asynchronous Python client library for **RabbitMQ**.
- **aiohttp**: a lightweight HTTP client/server that executes multiple requests asynchronously without blocking the main thread.
- **asyncpg**: PostgreSQL asynchronous database communication library, provides a powerful API for connecting to PostgreSQL databases and executing queries asynchronously.

PILOT AGENT



Pilots are an integral part of the *WMS* and are responsible for executing jobs on compute nodes, organizing their execution and communicating various information about the progress and state of the *WMS* node to the services. Compute nodes differ only in the availability of specialized co-processors (GPUs) and are assigned to the appropriate message broker based on the computational needs of the job.

CONCLUSIONS

The components of the workload management system were designed with consideration of the characteristics and internal requirements of both the *WFMS* and *DSM* systems. The goal is to complete the prototyping phase and fully integrate with the application layer components of the «SPD OnLine Filter» platform.