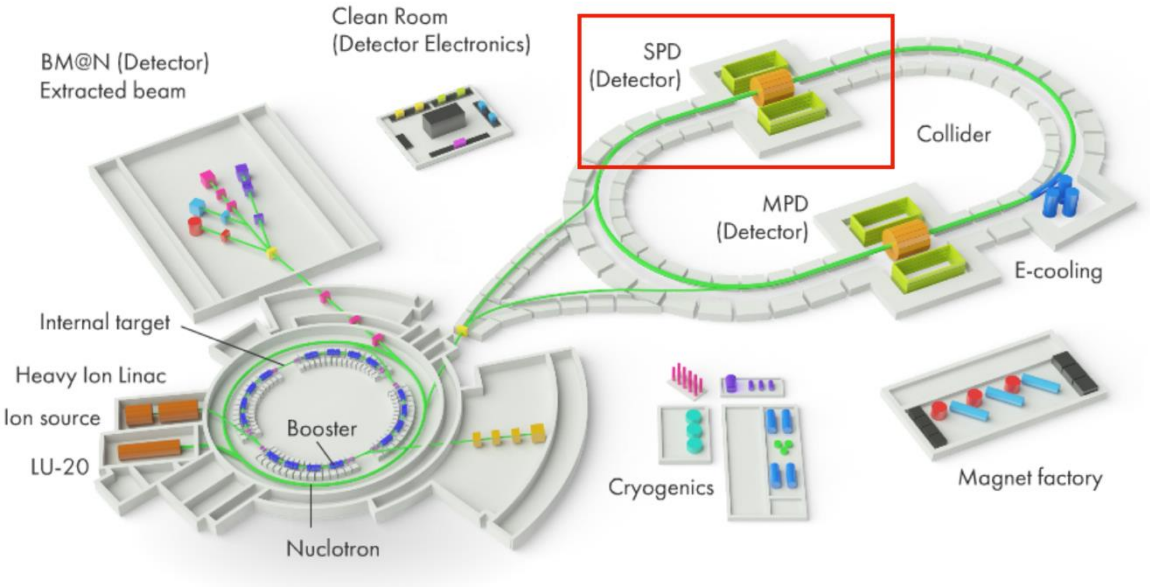
Data Flow and Data Control in the SPD Online Filter Middleware

Korshunova Polina
JINR MLIT

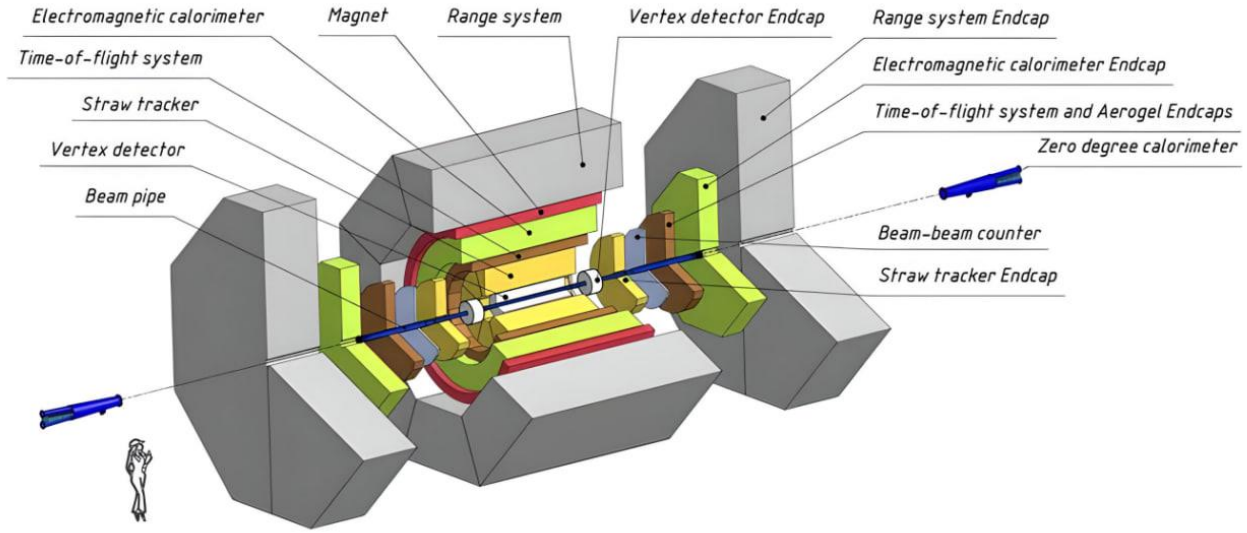
AYSS-2026

22.10.2026

Spin Physics Detector



The main purpose of SPD experiment is a comprehensive study of the unpolarized and polarized gluon component of nucleon

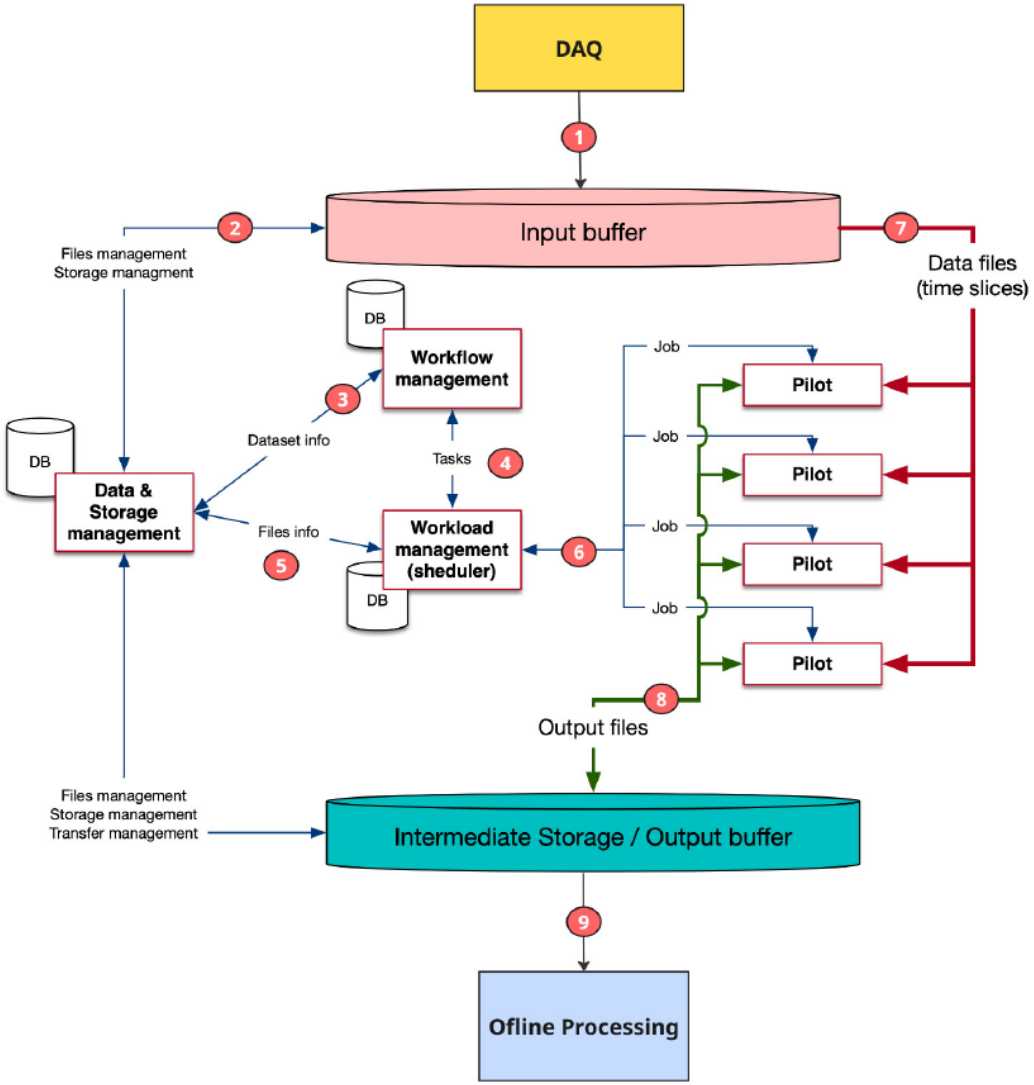


The SPD detector will be used to study the spin structure of the proton and deuteron and other spin-related phenomena

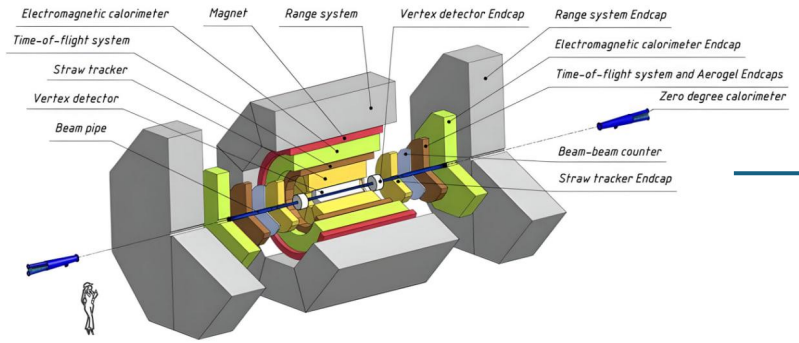
Online Filter

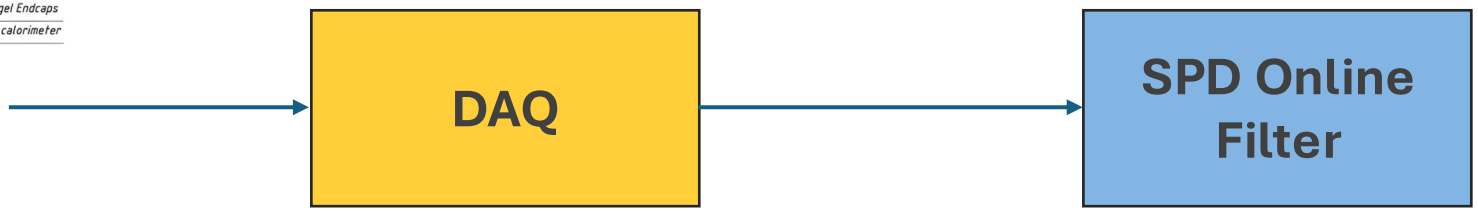
SPD Online Filter is a high-performance computing system for high-throughput data processing

A special feature of **high-throughput data processing** is the large amount of data, both primary that needs to be processed and intermediate that occurs during processing

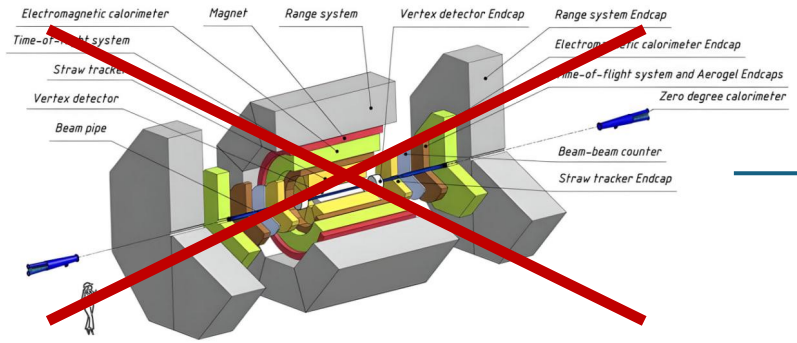


Data Flow


Electromagnetic calorimeter
Magnet
Range system
Vertex detector Endcap
Range system Endcap
Time-of-flight system
Electromagnetic calorimeter Endcap
Straw tracker
Time-of-flight system and Aerogel Endcaps
Vertex detector
Zero degree calorimeter
Beam pipe
Beam-beam counter
Straw tracker Endcap


DAQ
SPD Online Filter

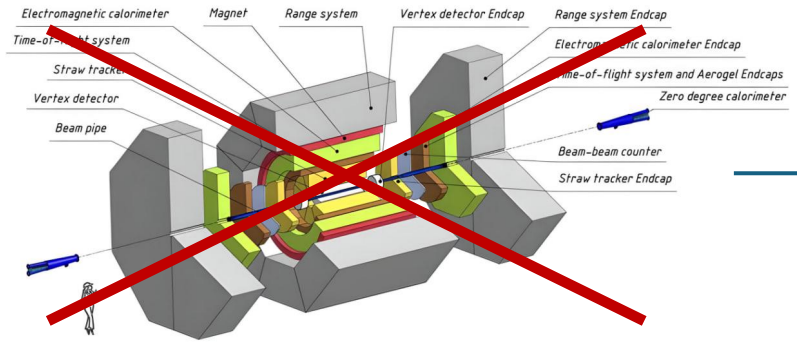
Data Flow



DAQ

SPD Online Filter

Data Flow



DAQ

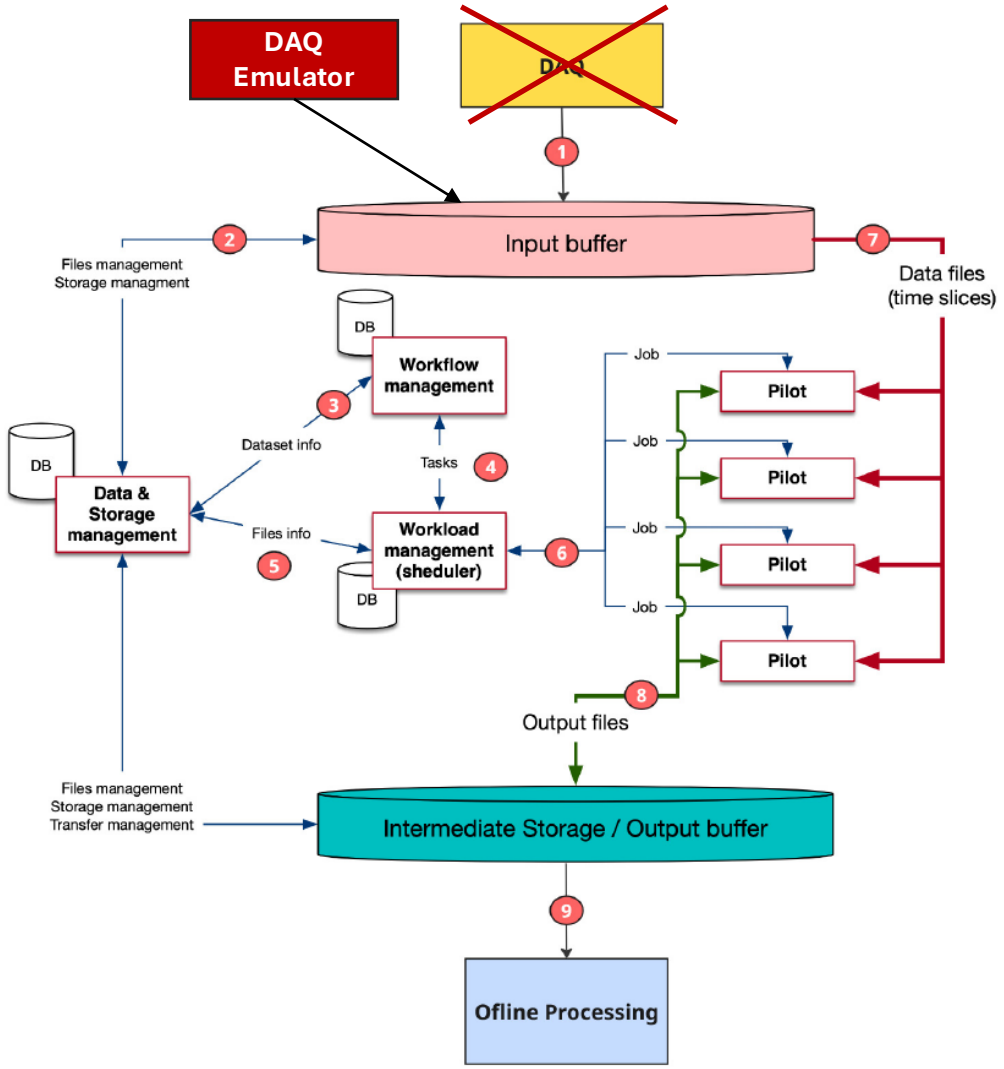
SPD Online Filter

DAQ Emulator

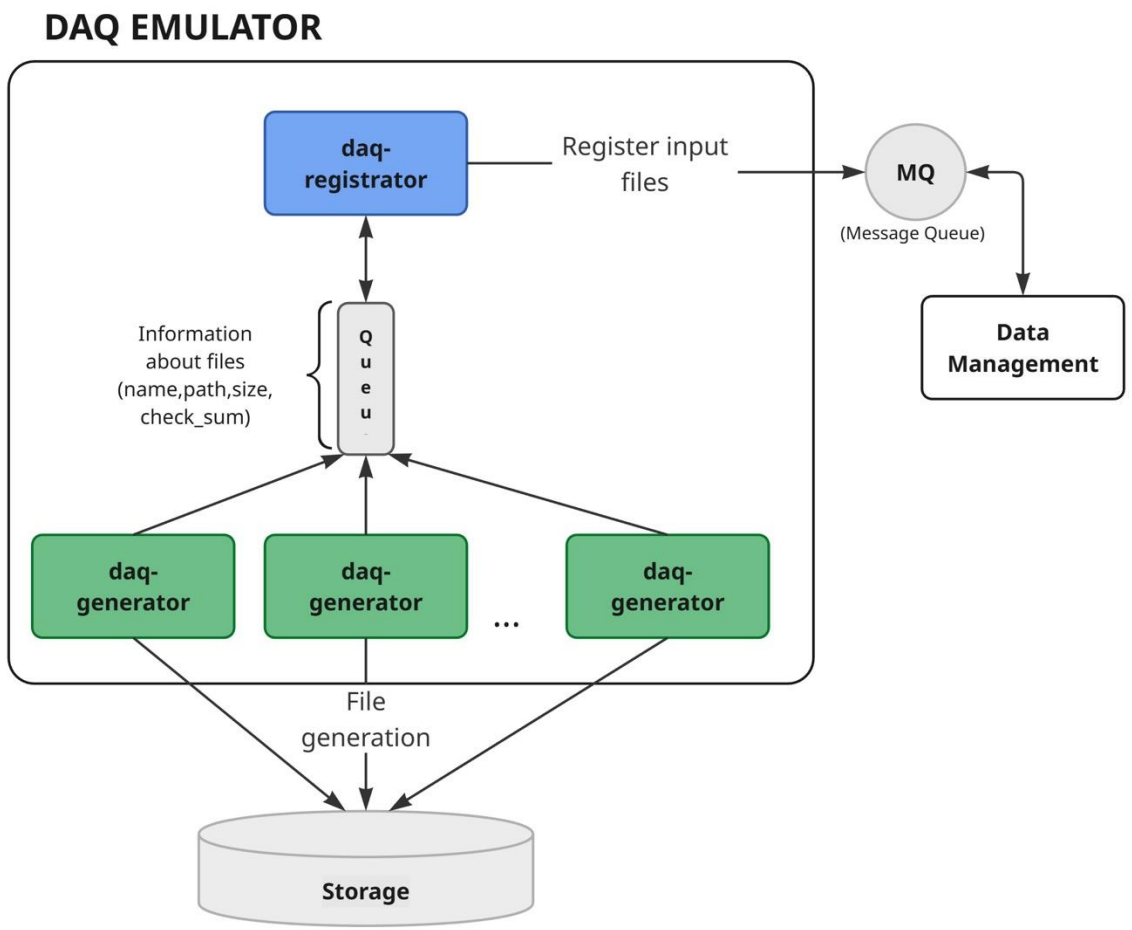
DAQ Emulator: Purpose

- Provide a continuous data stream for testing the SPD Online Filter Middleware
- Simulate the operation of the DAQ while real detector data is not available yet
- Enable load testing of the entire middleware

  - Emulates the load, but not the content!



DAQ Emulator: Architecture



daq-generator

- creates files in the storage and place corresponding information into queue, multiple generators can run simultaneously

daq-registrator

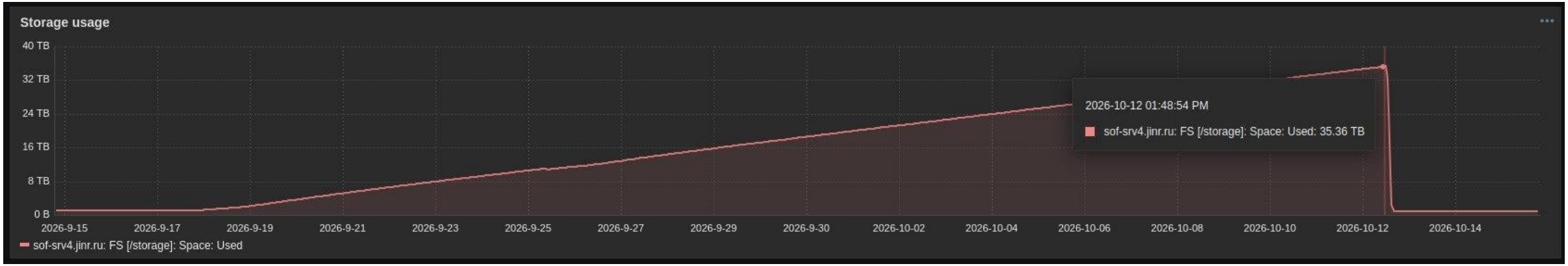
- register input files in ONF via a message broker when the number of accumulated files reaches the threshold required to form a dataset

Test load

Current prototype:

- Produce 100 files (10 files in one dataset) each run
- Each file 512 MB
- Run every 3 hours

400 GB per day of raw data



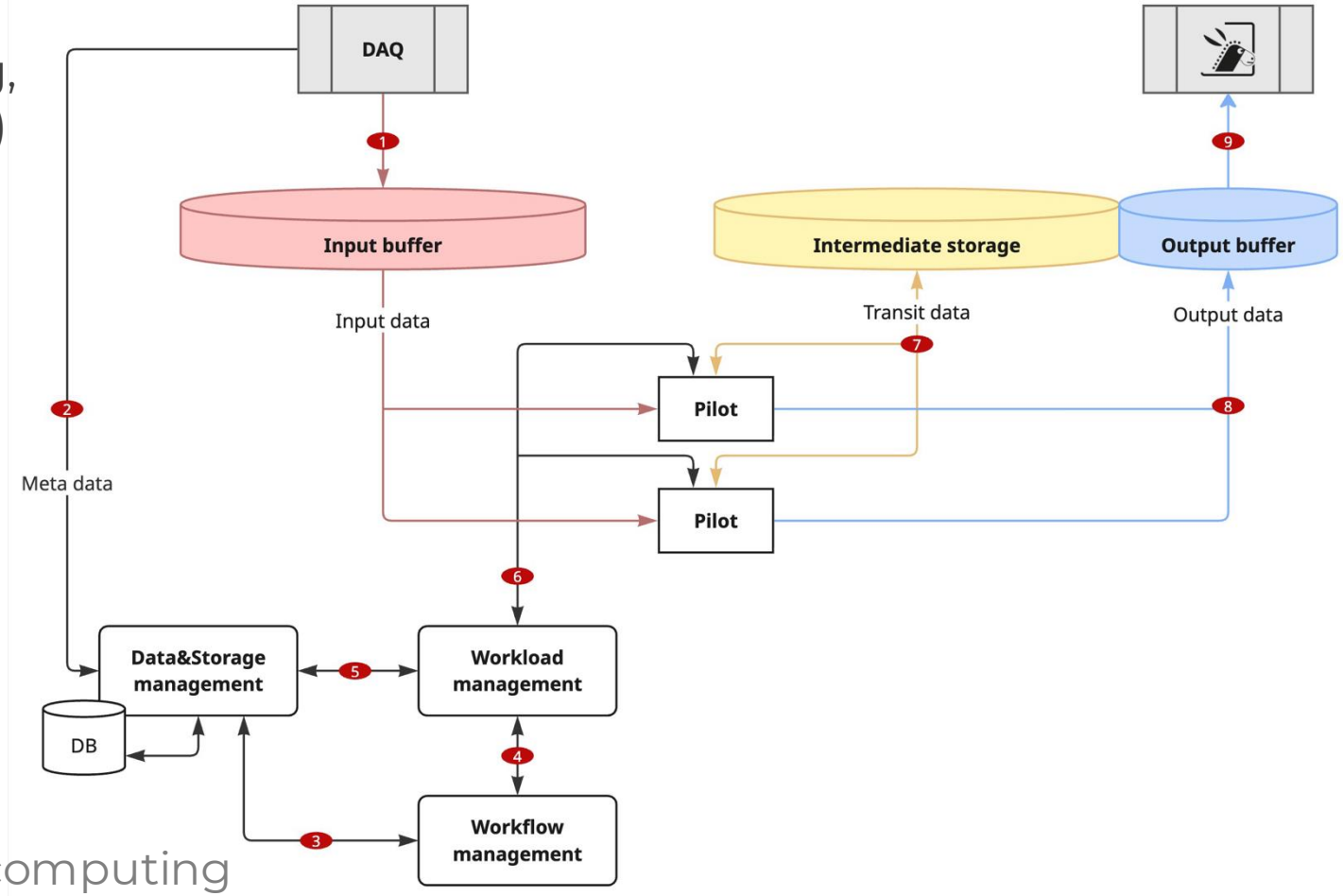
Middleware

Data 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

Workload Management System
- implementation of processing stages (task generation, sending tasks to pilots)

Pilot – an application running on a computing node and performing tasks



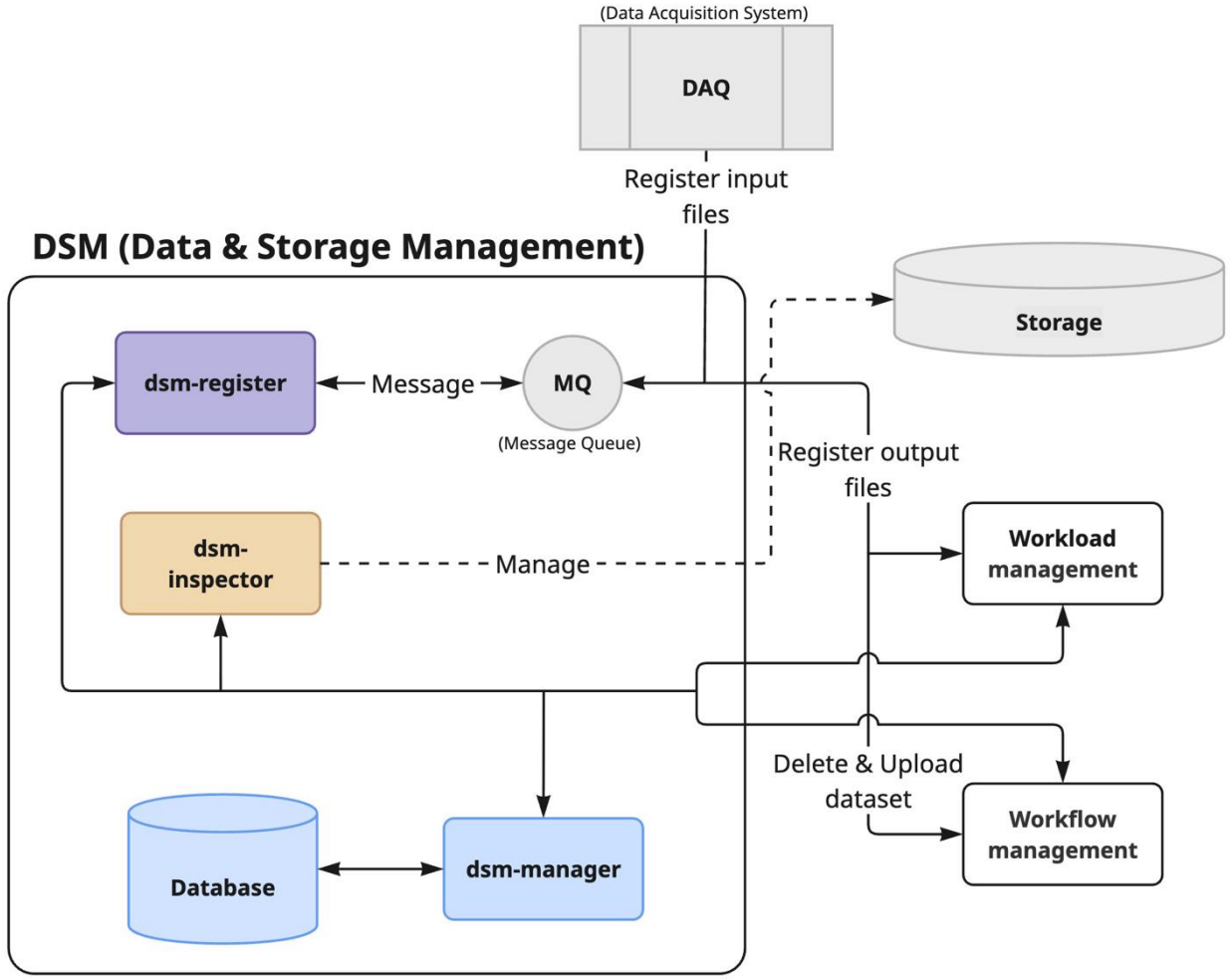
Key Features of DMS

Purpose: Automation of lifecycle management of experimental data at the stage of primary processing (decoding data, partial reconstruction of events, filtering of events)

Key Features:

- Separation of metadata and physical file storage (cataloging using a database)
- Data lifecycle tracking (maintaining a status model of file and dataset)
- Registration of files and datasets via API
- Ensuring consistency between the physical storage and the database using background services

Microservice architecture in DMS



According to the microservice architecture approach, we split a large application into independent services, each of which performs a separate function

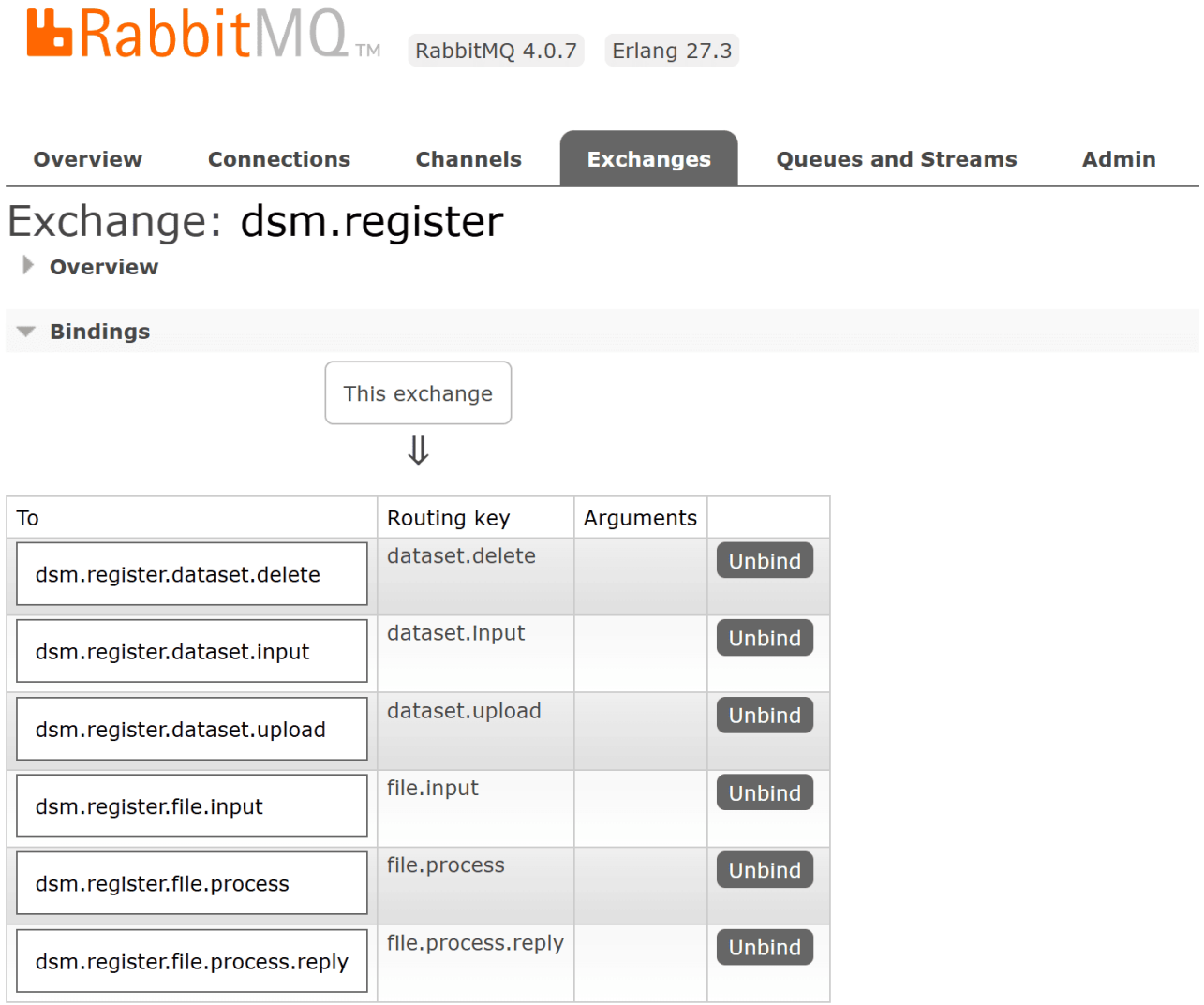
- **dsm-register** (data registration)
- **dsm-manager** (REST API of data catalog)
- **dsm-inspector** (daemon tasks for monitoring)

Dsm-register

The service should check the message queue and process requests for adding/deleting data in the system

RabbitMQ is used as an AMQP broker that performs routing and subscribing to the necessary queues

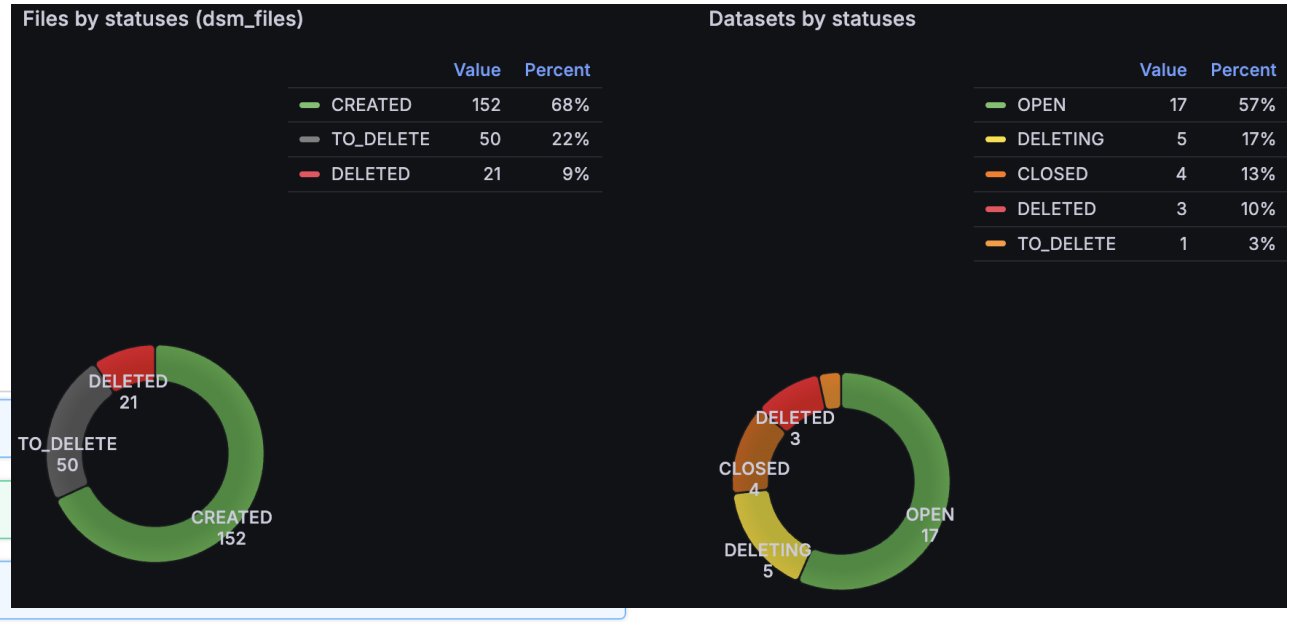
- All queues are fully operational
- Implement retry mechanism (exponential backoff) in case an error occurs when connecting to dsm-manager
- Reduce number of requests to dsm-manager



Dsm-manager

The service should provide a REST API to the database

The asynchronous **FastAPI** framework is used as a web framework



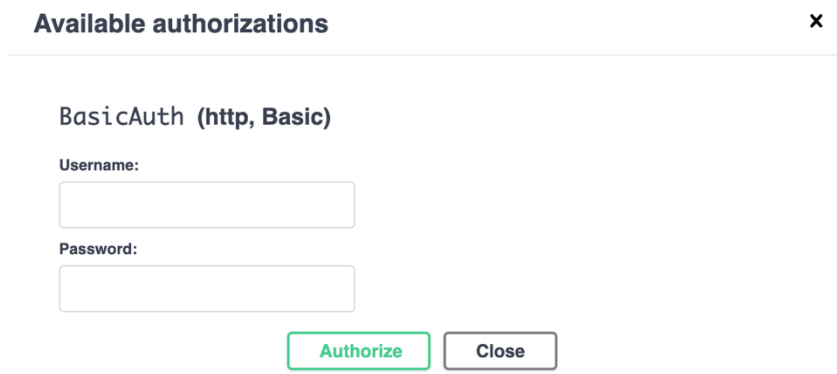
file

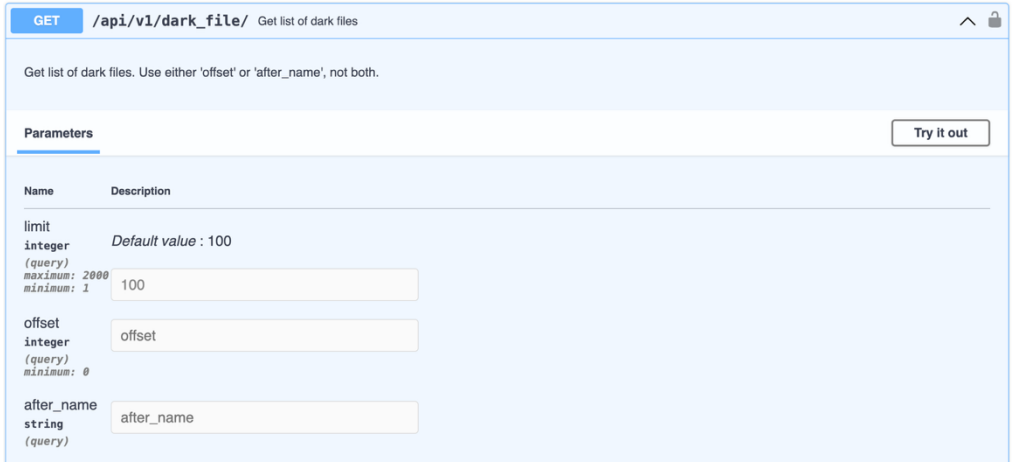
- GET `/api/v1/file/` Get list of files
- POST `/api/v1/file/` Add file
- GET `/api/v1/file/dataset/{dataset_id}` Get dataset content
- GET `/api/v1/file/{file_id}` Get file by id
- PUT `/api/v1/file/{file_id}` Update file
- DELETE `/api/v1/file/{file_id}` Remove file
- PATCH `/api/v1/file/{file_id}` Partial file update
- GET `/api/v1/file/file_name/{file_name}` Get file by name
- PATCH `/api/v1/file/{file_id}/status` Update file status

FastAPI

Dsm-manager: new features

- Add authorization to dsm-manager API and Swagger
- Add offset/keyset pagination to GET requests
- Add new functionality (partial update)
- Code refactoring with the implementation of the Unit Of Work pattern (it became necessary to perform several operations within a single transaction)





Dsm-inspector

The service consists of a set of **background** tasks:

- Deleting files on storages
  - Deleting datasets and files
  - Deleting dark files

Dsm-inspector

The service consists of a set of **background** tasks:

- Deleting files on storages
  - Deleting datasets and files
  - Deleting dark files

- File consistency check
  - Reduce number of request to dsm-manager

Dsm-inspector

The service consists of a set of **background** tasks:

- Deleting files on storages
  - Deleting datasets and files
  - Deleting dark files

- File consistency check
  - Reduce number of request to dsm-manager

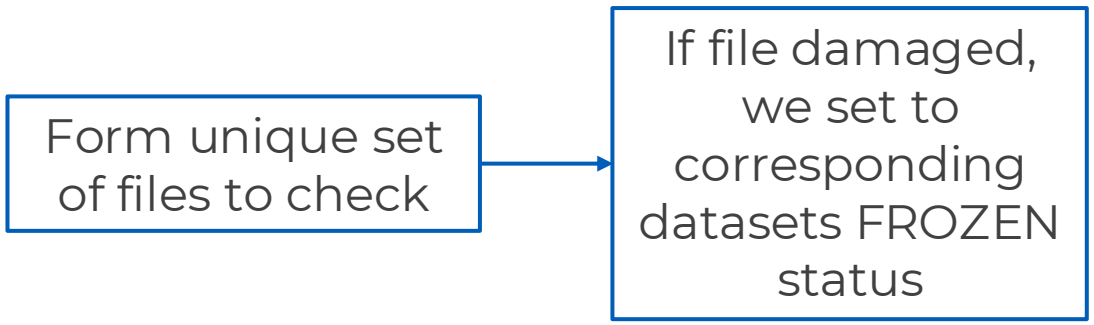
Form unique set of files to check

Dsm-inspector

The service consists of a set of **background** tasks:

- Deleting files on storages
  - Deleting datasets and files
  - Deleting dark files

- File consistency check
  - Reduce number of request to dsm-manager

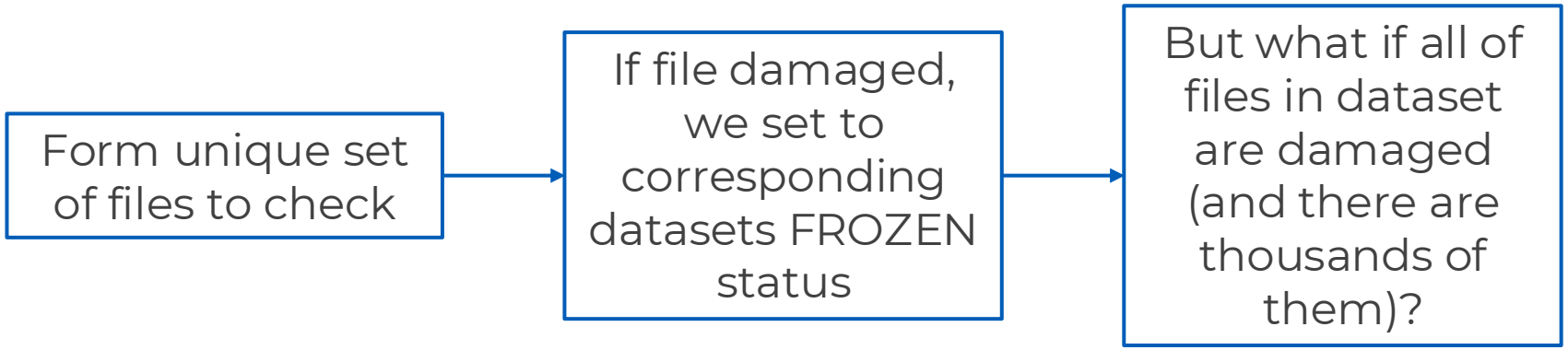


Dsm-inspector

The service consists of a set of **background** tasks:

- Deleting files on storages
  - Deleting datasets and files
  - Deleting dark files

- File consistency check
  - Reduce number of request to dsm-manager

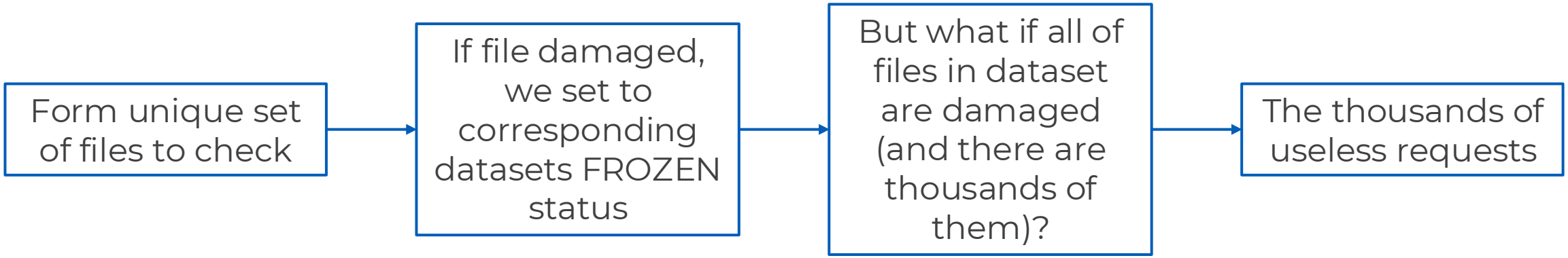


Dsm-inspector

The service consists of a set of **background** tasks:

- Deleting files on storages
  - Deleting datasets and files
  - Deleting dark files

- File consistency check
  - Reduce number of request to dsm-manager



Dsm-inspector

The service consists of a set of **background** tasks:

- Deleting files on storages
  - Deleting datasets and files
  - Deleting dark files

- File consistency check
  - Reduce number of request to dsm-manager

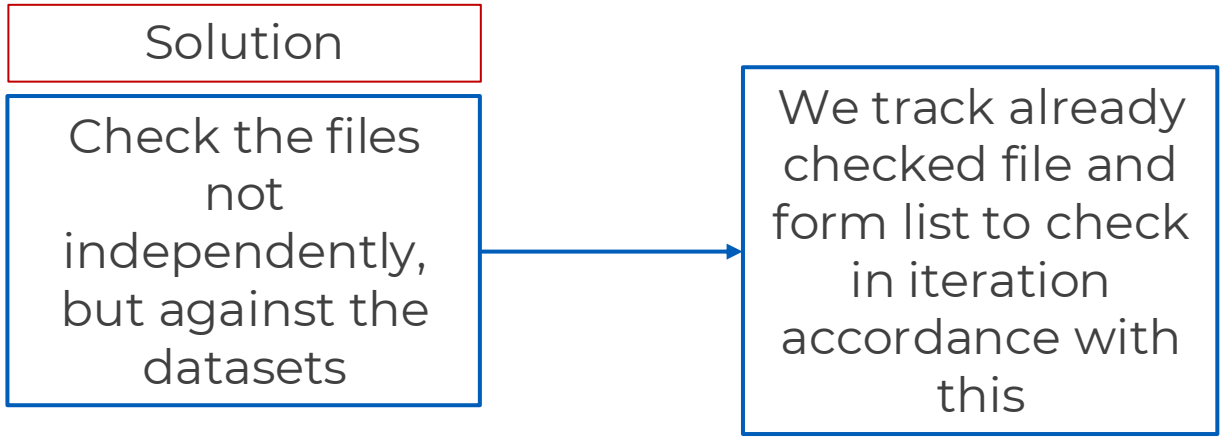
Solution
Check the files not independently, but against the datasets

Dsm-inspector

The service consists of a set of **background** tasks:

- Deleting files on storages
  - Deleting datasets and files
  - Deleting dark files

- File consistency check
  - Reduce number of request to dsm-manager

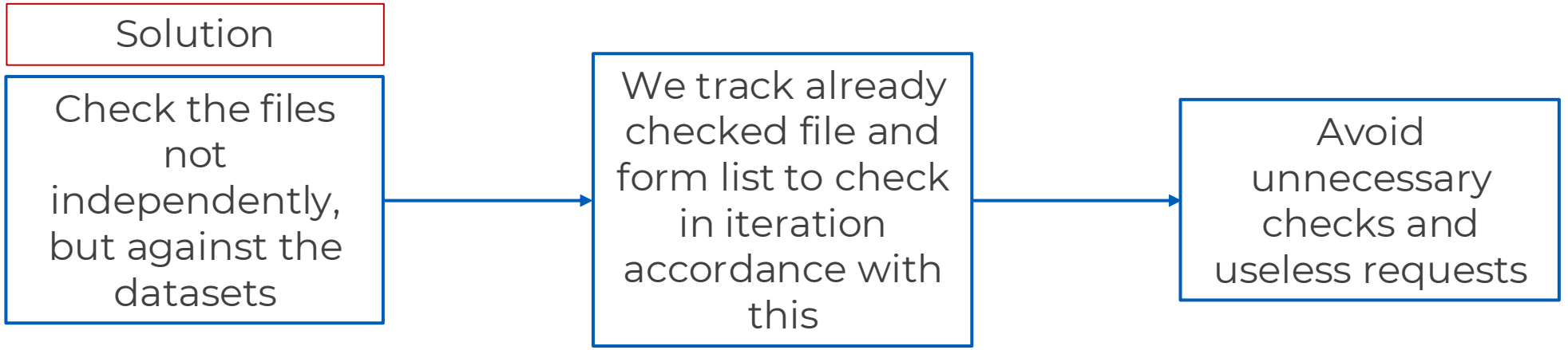


Dsm-inspector

The service consists of a set of **background** tasks:

- Deleting files on storages
  - Deleting datasets and files
  - Deleting dark files

- File consistency check
  - Reduce number of request to dsm-manager



Dsm-inspector

The service consists of a set of **background** tasks:

- Deleting files on storages
  - Deleting datasets and files
  - Deleting dark files

- File consistency check

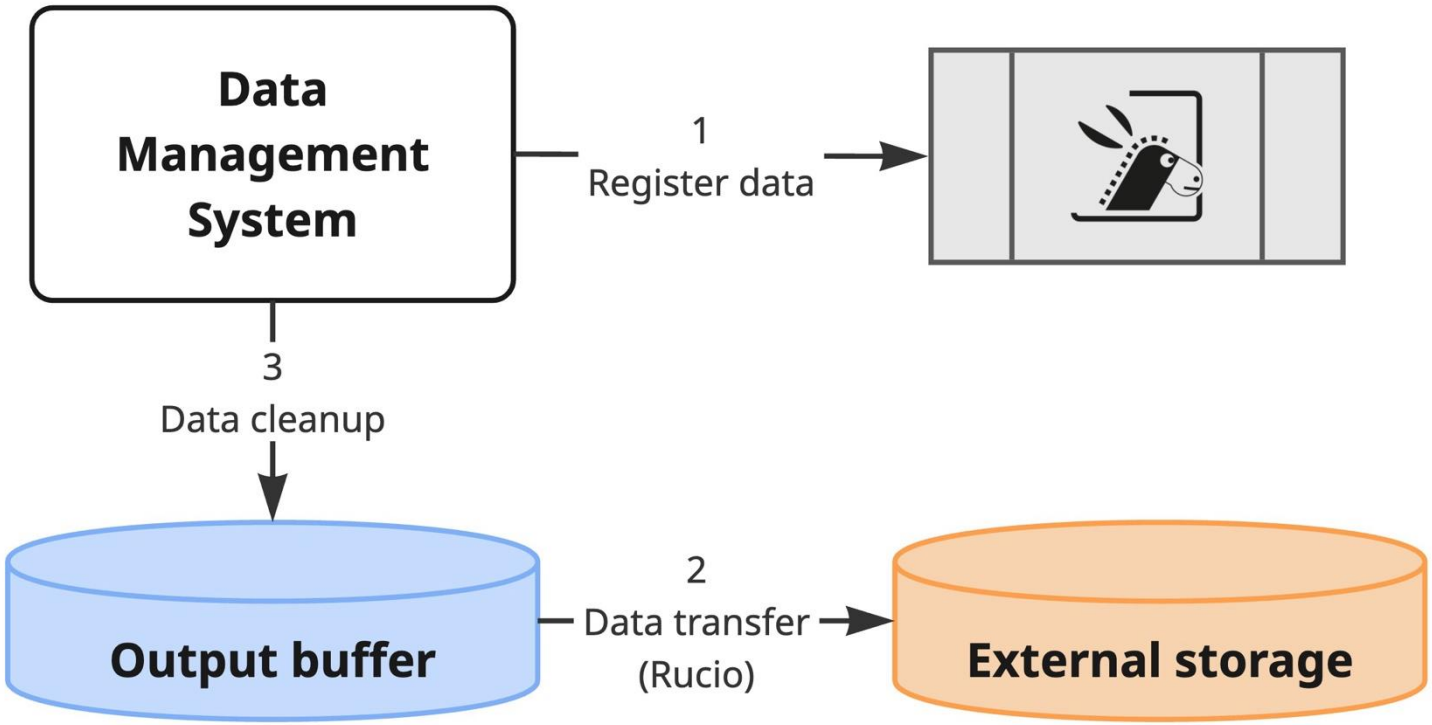
- Storage monitoring
  - Dark files monitoring
  - Monitoring storage usage

Dsm-inspector

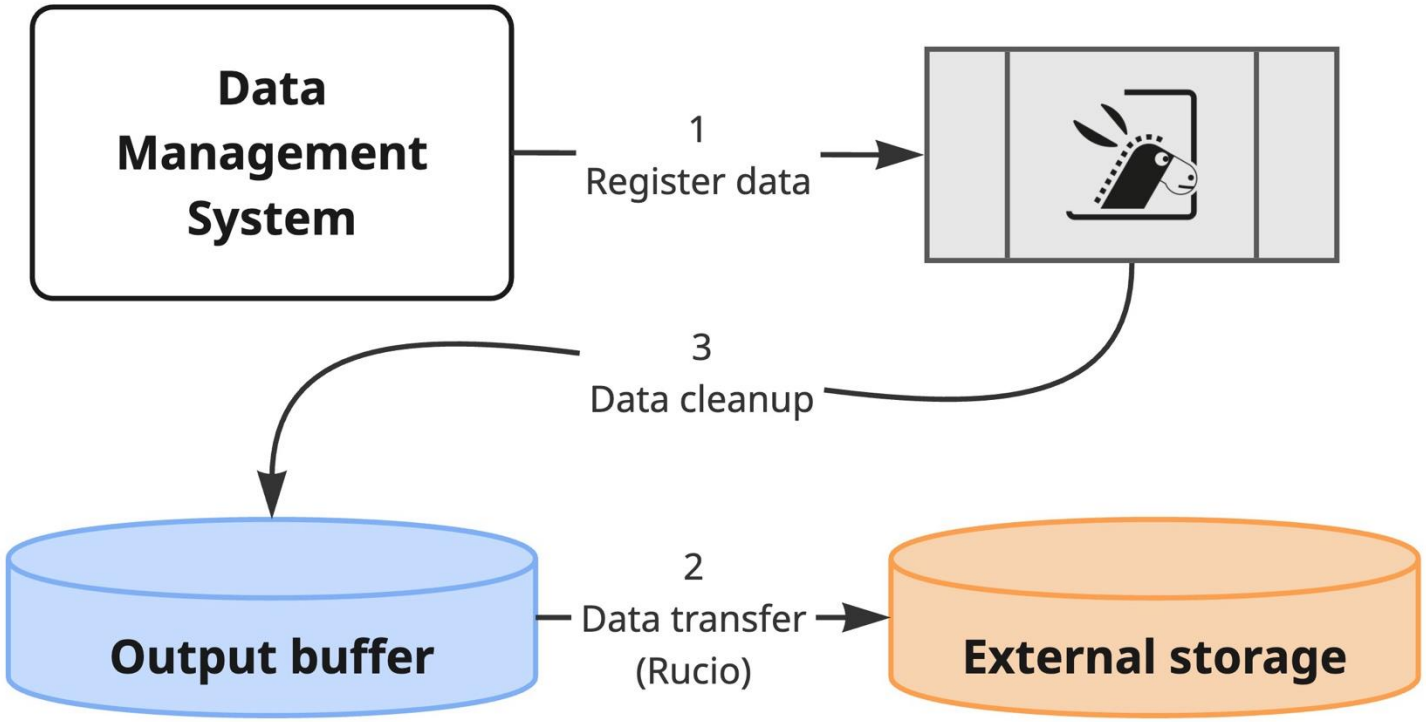
The service consists of a set of **background** tasks:

- Deleting files on storages
  - Deleting datasets and files
  - Deleting dark files
- File consistency check
- Storage monitoring
  - Dark files monitoring
  - Monitoring storage usage
- File upload control (in mock state)
  - Setting tasks for uploading datasets
  - Uploading monitoring

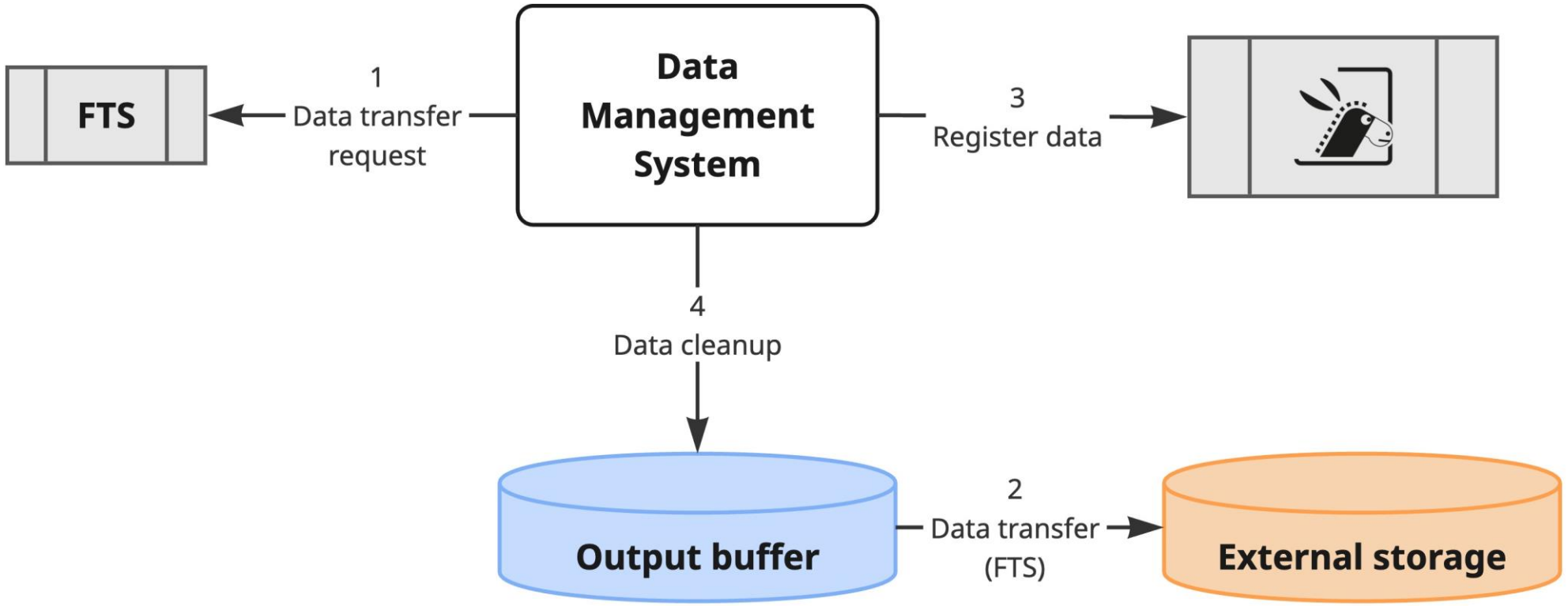
Data upload to external storage (1/3 methods)



Data upload to external storage (2/3 methods)



Data upload to external storage (3/3 methods)



Next steps

- Implement all three ways to uploading data, test them, and choose the best one

Thank you for your attention!

Backup slides

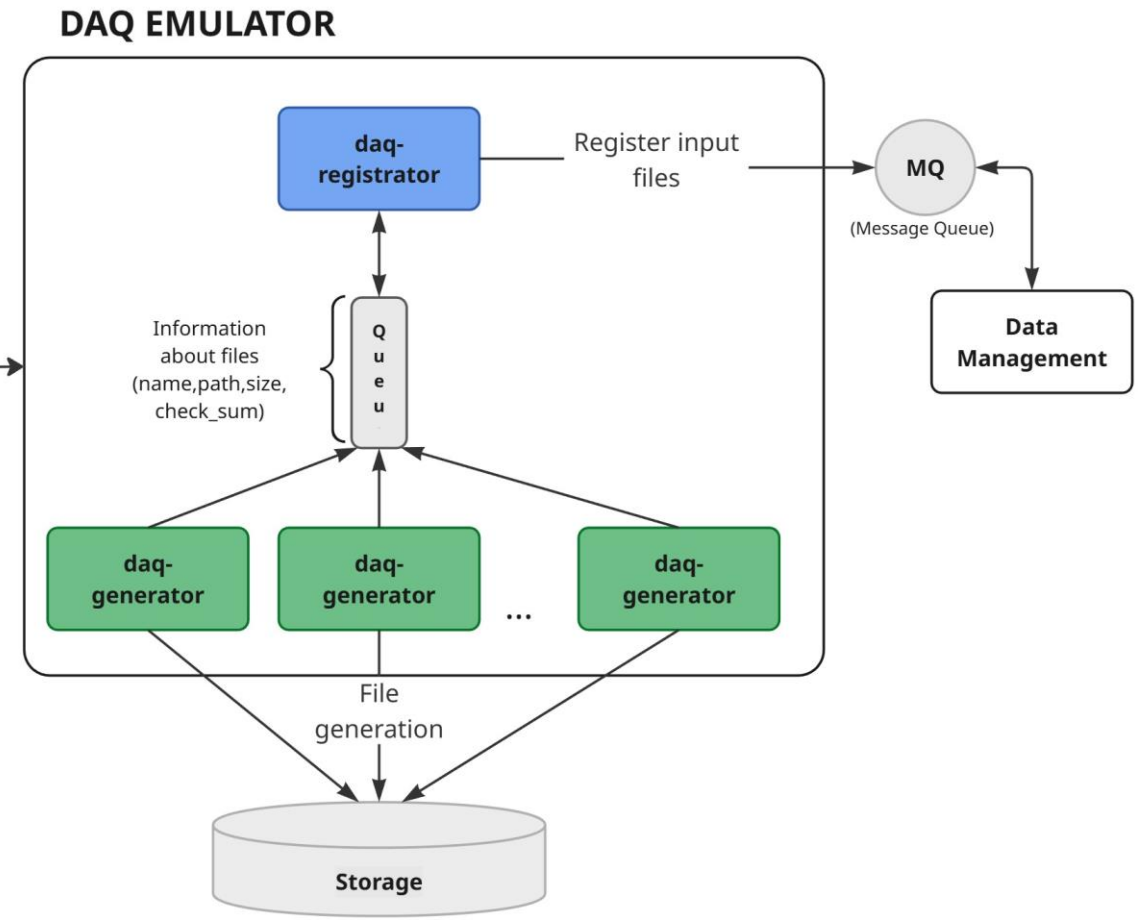
DAQ Emulator: Key Features

config.ini

```
[General]
number_of_generators = 5
number_of_runs = 10
sleep = 7200
input_storage =
/path/to/input/storage

[Files]
file_size = 50
number_of_files = 100

[Datasets]
name_template = "_run_"
dataset_size = 10
```



- Multiple parallel data generators
- Read the configuration file every new run (you can change the configuration without restarting the application)
- Provides an API for quick launch of the run

DAQ Emulator: Configuration Parameters

config.ini

```
[General]
number_of_generators = 5
number_of_runs = 10
sleep = 7200
input_storage =
/path/to/input/storage

[Files]
file_size = 50
number_of_files = 100

[Datasets]
name_template = "_run_"
dataset_size = 10
```

Parameter	Description
Number of generators	How many daq-generator instances run simultaneously
Number of runs	How many generation cycles are performed
Number of files per run and their size	Volume of generated data
Dataset size	Number of files per dataset
Dataset name template	Naming convention for datasets

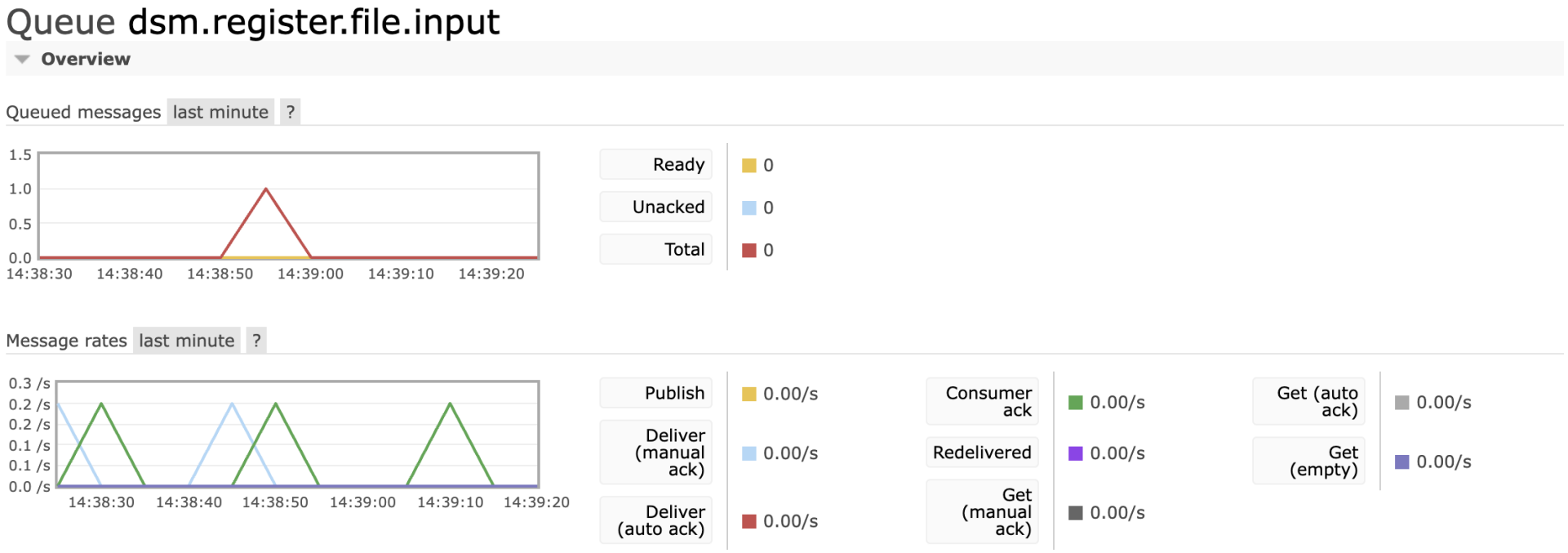
Load Testing (1/2)

daq_data table

run	fileName	fileSize	sessionStart	File generation time
1	run-1-100-20260423_114144.dat	537 MB	2026-04-23_11:37:59	2.02 s
1	run-1-99-20260423_114138.dat	537 MB	2026-04-23_11:37:59	3.51 s
1	run-1-98-20260423_114138.dat	537 MB	2026-04-23_11:37:59	3.59 s
1	run-1-97-20260423_114138.dat	537 MB	2026-04-23_11:37:59	3.74 s
1	run-1-96-20260423_114132.dat	537 MB	2026-04-23_11:37:59	3.71 s
1	run-1-95-20260423_114131.dat	537 MB	2026-04-23_11:37:59	3.01 s
1	run-1-94-20260423_114131.dat	537 MB	2026-04-23_11:37:59	3.04 s
1	run-1-93-20260423_114126.dat	537 MB	2026-04-23_11:37:59	2.94 s
1	run-1-92-20260423_114126.dat	537 MB	2026-04-23_11:37:59	2.99 s
1	run-1-91-20260423_114125.dat	537 MB	2026-04-23_11:37:59	3.45 s

- Produce 10 datasets of 10 files
- File generation time ~ 3-4 seconds

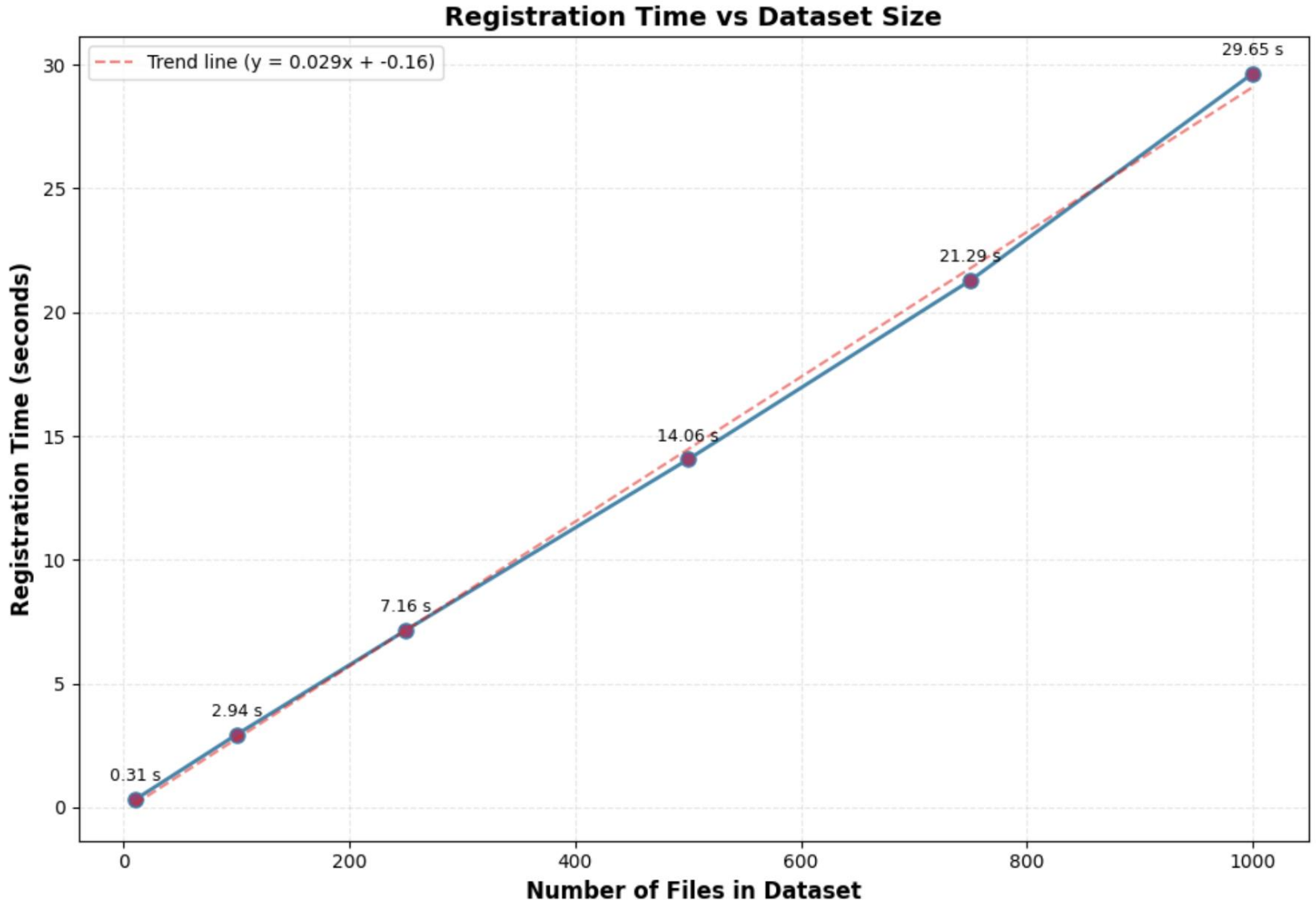
Load Testing (2/2)




```
dsm-register | 2026-04-23 11:38:30 INFO: Registration time = 0.32 sec
dsm-register | 2026-04-23 11:38:50 INFO: Registration time = 0.31 sec
dsm-register | 2026-04-23 11:39:11 INFO: Registration time = 0.3 sec
dsm-register | 2026-04-23 11:39:52 INFO: Registration time = 0.35 sec
dsm-register | 2026-04-23 11:39:55 INFO: Registration time = 0.31 sec
dsm-register | 2026-04-23 11:40:18 INFO: Registration time = 0.3 sec
dsm-register | 2026-04-23 11:40:47 INFO: Registration time = 0.29 sec
dsm-register | 2026-04-23 11:41:09 INFO: Registration time = 0.31 sec
dsm-register | 2026-04-23 11:41:29 INFO: Registration time = 0.31 sec
dsm-register | 2026-04-23 11:41:51 INFO: Registration time = 0.28 sec
```

- Send message to dsm-register
- Dsm-register creates a new dataset and registers files
- Registration time for one dataset ~ 0.3 seconds

Dataset Registration Time



Database Structure

