



Заявка на соискание гранта ОМУС, стипендий им. М.Г. Мещерякова и Н.Н. Говоруна

Разработка системы управления рабочей нагрузкой
SPD Online Filter

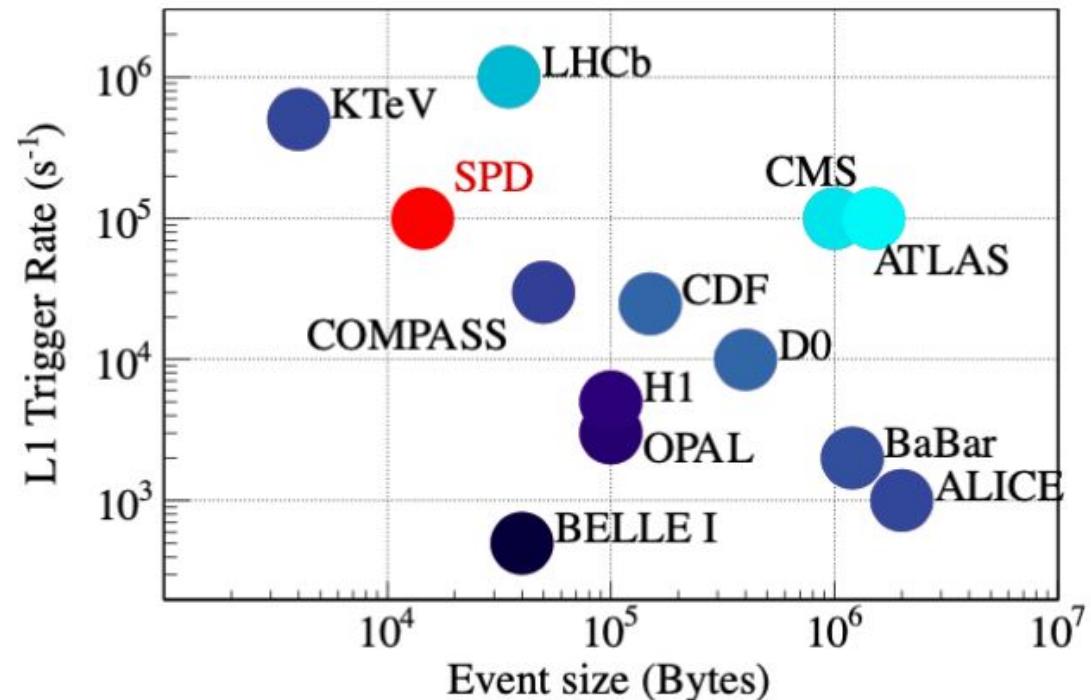
Гребень Никита
стажер-исследователь
НТО ВКиРИС

Непосредственный руководитель
Олейник Данила Анатольевич
старший научный сотрудник ЛИТ ОИЯИ

Актуальность работы

«SPD Online filter» будет представлять собой программно-аппаратный комплекс высокопропускной обработки первичных данных эксперимента SPD коллайдере NICA, с целью уменьшения их объема для последующей обработки и долговременного хранения. Аппаратная часть будет состоять из совокупности многоядерных вычислительных узлов, высокопроизводительных систем хранения данных и ряда управляющих серверов; программная часть будет состоять не только из прикладного программного обеспечения, но и из комплекса промежуточного ПО – «SPD Online filter» «Visor», роль которого заключается в реализации многоступенчатой обработки данных.

Система управления нагрузкой, является ключевым компонентом комплекса промежуточного ПО

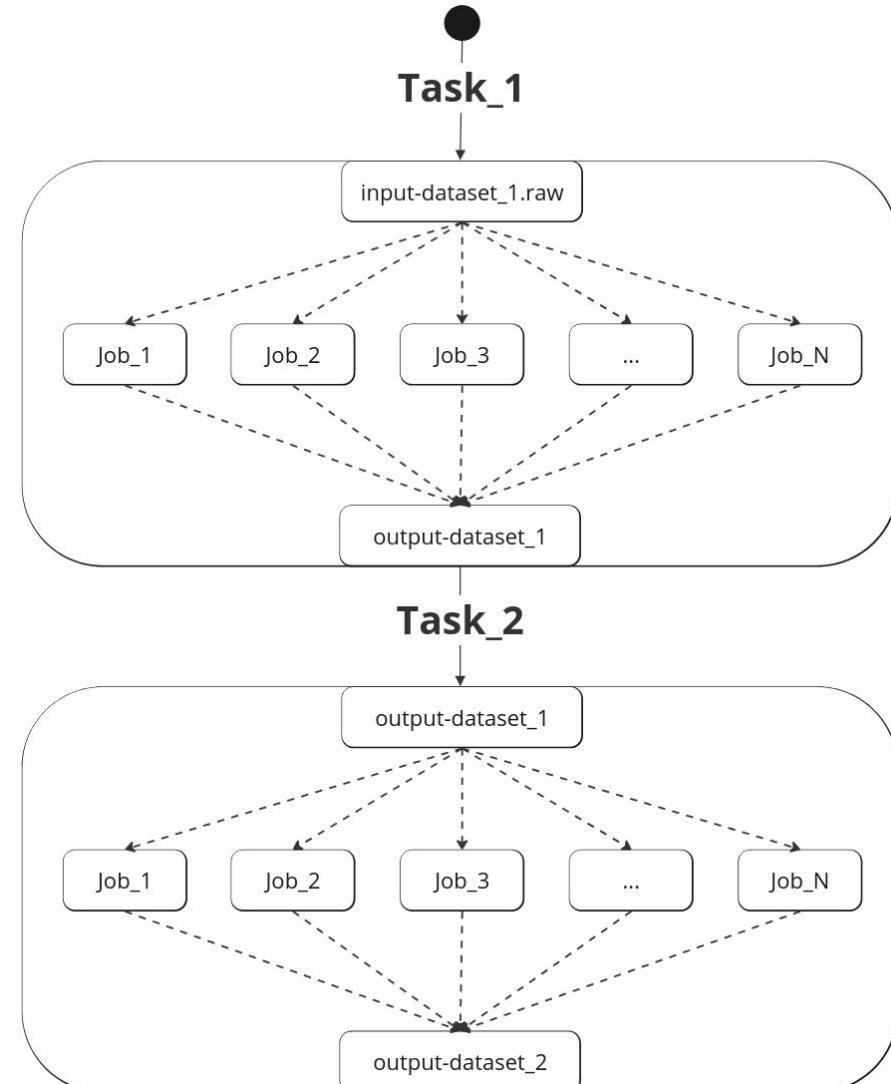
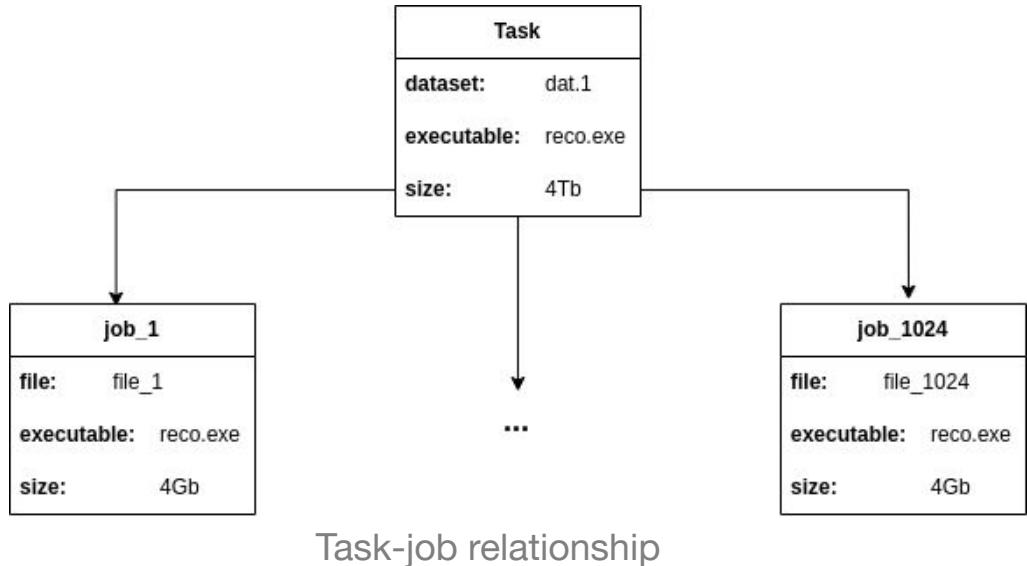


- ~ 20 ГБ/с (или 200 ПБ/год) «сырых» данных
- безтриггерный DAQ (Data Acquisition System)

Высокопропускные вычисления



- НТС определяется как тип вычислений, при котором одновременно выполняется множество простых и независимых друг от друга задач для выполнения задания обработки данных.
- Поскольку каждый элемент данных может обрабатываться одновременно, это может быть применено к данным, агрегированным системой сбора данных (DAQ).
- Обработка данных многоступенчатая:
 - Один этап обработки → **task**
 - Обработка блока данных (файла) → **job**

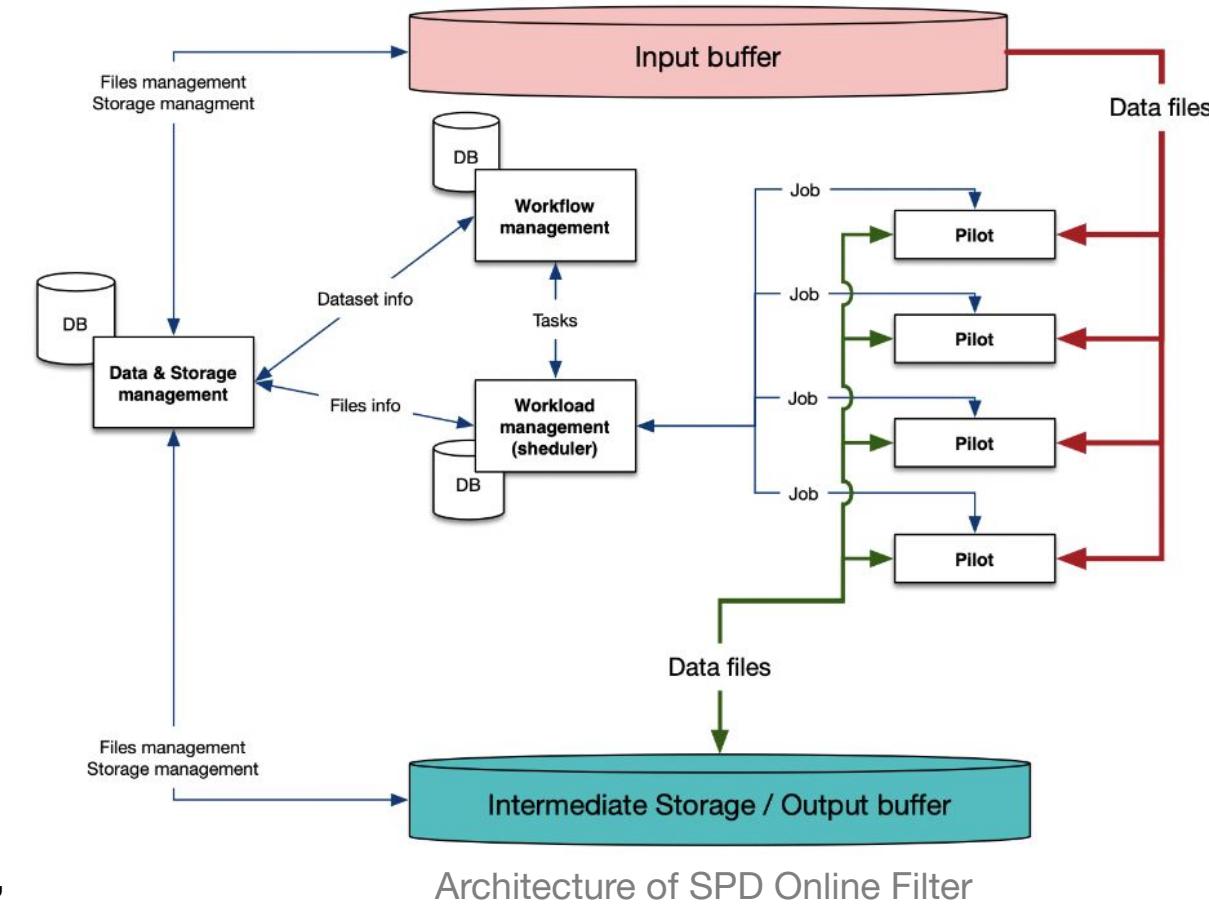


Data processing workflow example

Промежуточное программное обеспечение

«SPD OnLine filter» – аппаратно-программный комплекс, обеспечивающий многоступенчатую высокопропускную обработку первичных данных эксперимента SPD.

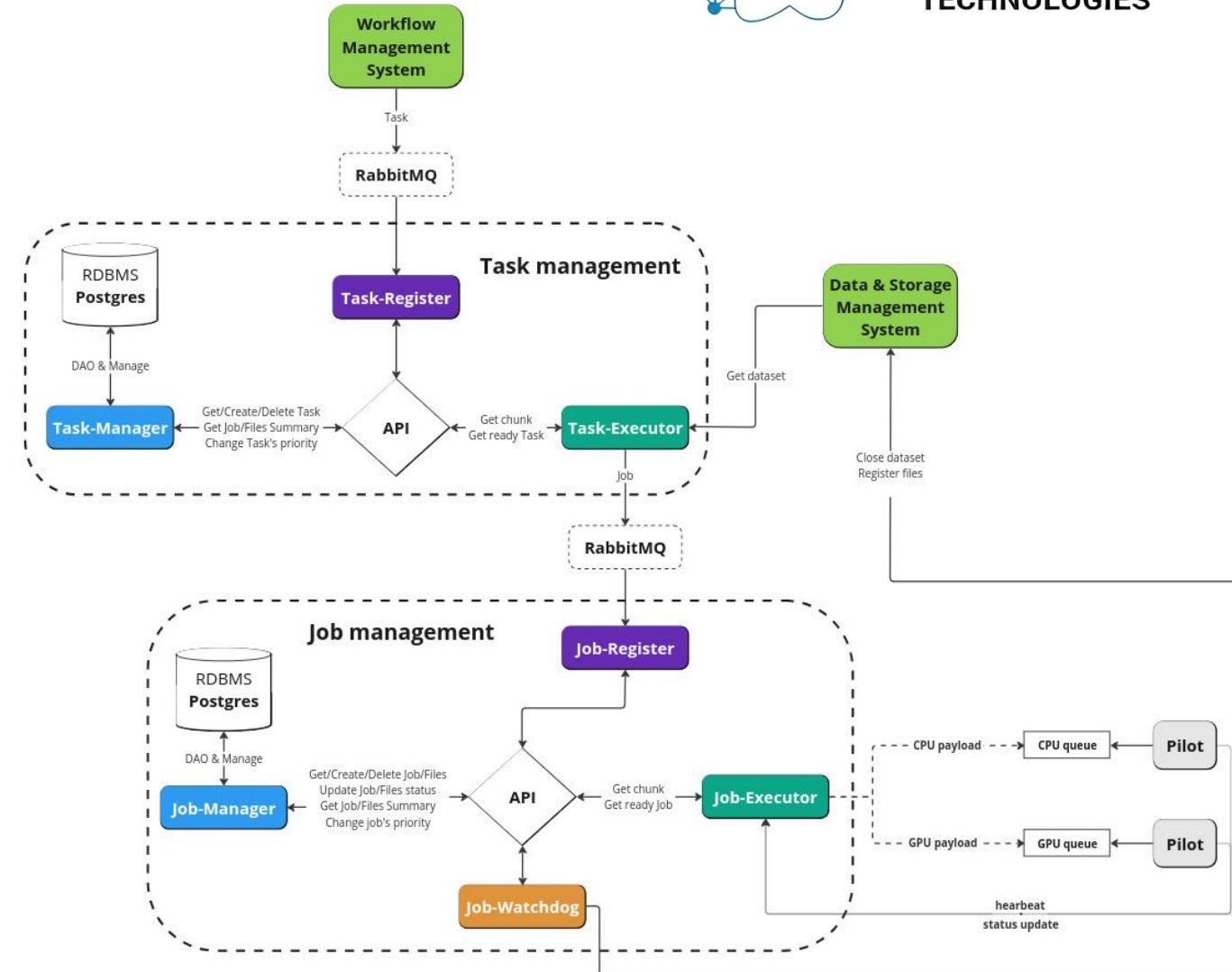
- Система управления данными
 - Поддержка жизненного цикла данных (каталог данных, проверка согласованности, очистка, хранение);
- Система управления процессами
 - Определение и выполнение цепочки обработки, сгенерирував необходимое количество вычислительных задач;
- Система управления нагрузкой
 - Генерация необходимого количества задач для выполнения задания;
 - Контроль выполнения задач с помощью пилотов, работающих на вычислительных узлах;
- Координатор: **Данила Олейник**



Требования к системе управления нагрузкой



- **Регистрация задачий:** формализованное описание задания, включая параметры задач и требуемые метаданные;
- **Определение задач:** генерация необходимого количества задач для выполнения задания путем контролируемой загрузки доступных вычислительных ресурсов;
- **Управление исполнения задач:** контроль состояния задач, повторные попытки выполнения задач в случае сбоев, завершение выполнения задачи;
- **Контроль согласованности:** контроль согласованности данных в отношении заданий, файлов и задач;
- **Планировщик:** реализация алгоритма планирования для распределения заданий/задач;



SPD Workload Management System High Level Architecture

Pilot Agent

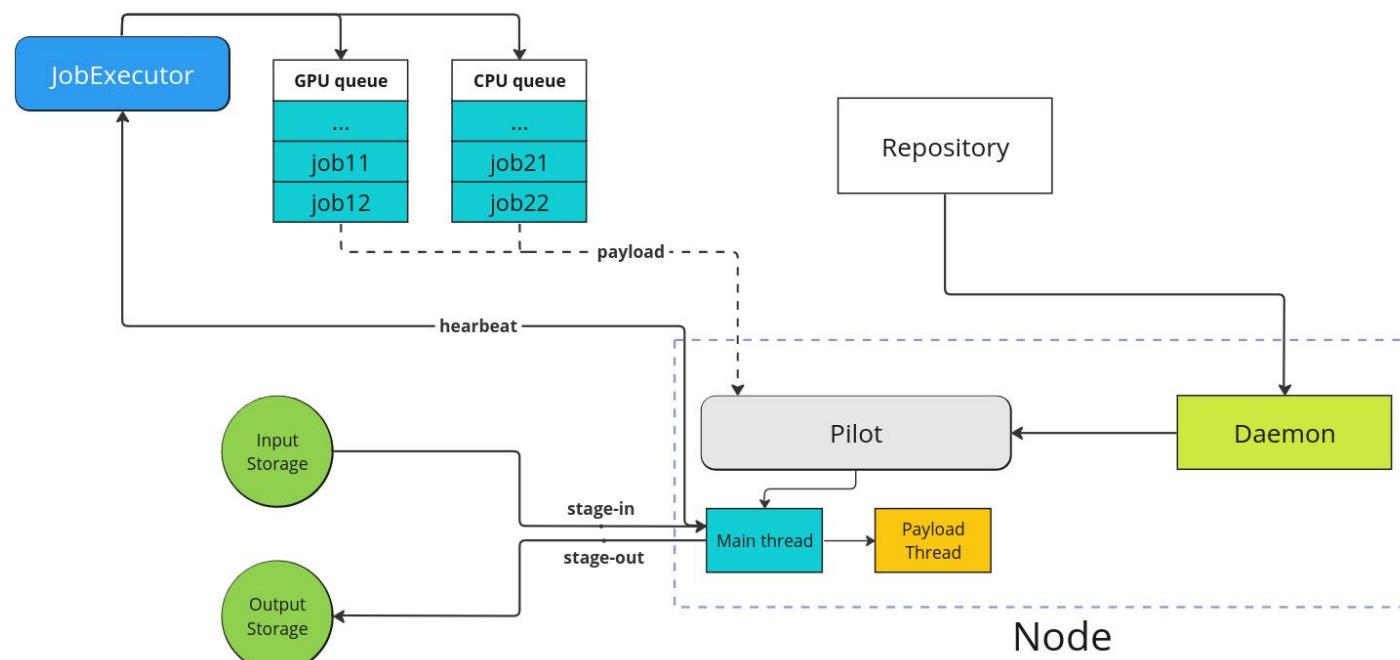
- Агентское приложение развертывается на вычислительном узле и состоит из следующих двух компонентов: UNIX демона и самого пилота.
- Задача демона UNIX - запустить следующий пилот, загрузив актуальную версию из хранилища.
- Сам Pilot представляет собой многопоточное приложение, отвечающее за
 - Получение и валидацию задач из брокера сообщений;
 - Загрузка входных файлов и выгрузка файлов с результатами в выходное хранилище;
 - Запуск подпроцесса для выполнения полезной нагрузки (декодирование формата DAQ, алгоритм распознавания треков и т. д.)
 - Информирование вышестоящей системы о текущем статусе полезной нагрузки и самого пилота

Два типа узлов:

- Multi-CPU
- Multi-CPU + GPU

Два канала связи:

- HTTP (aiohttp)
- AMQP (message broker - RabbitMQ)



Текущий статус

Проектирование:

- Разработан и реализован набор необходимых REST API методов (DB API сервис);
- Реализован механизм объявления модели данных в базе данных на основе ORM и скриптов миграции;
- Настройка инструментов CD (сборка и развертывание) в облачной инфраструктуре ЛИТ;
- Разработка сценариев межсервисного взаимодействия - определение контрактов API;
- Реализована модель взаимодействия с пилотом на базе описанной статусной модели задач.

Прототипирование:

- Большинство микросервисов реализовано;
- Подсистема управления задачами является наиболее продвинутой: реализовано и тестируется большинство взаимодействий;
- Переезд на производственный СУБД Postgres;
- Пилот обрабатывает все этапы выполнения задачи;
- Реализуется разбиение задания;
- Переезд на производственный сервер RabbitMQ.

Публикации

1. Greben, N., Romanychev, L., Oleynik, D., Degtyarev, A. SPD On-Line Filter: Workload Management System and Pilot Agent. Phys. Part. Nuclei 55, 612–614 (2024).
2. V.M. Abazov et.al [SPD Collaboration] Technical Design Report of the Spin Physics Detector at NICA // arXiv: 2404.08317 [hep-ex]

Выступления

1. 59th meeting of the PAC for Particle Physics – «Workload Management System for SPD Online Filter» – ОИЯИ – Дубна – Россия – 2024. – Постерная сессия
2. VII SPD Collaboration meeting – «Workload Management System for SPD Online filter» – Institute of Nuclear Physics – Almaty – Kazakhstan – 2024. – Устный доклад
3. JINR Association of Young Scientists and Specialists Conference "Alushta-2024" – «Система управления нагрузкой специализированной вычислительной системы SPD Online Filter» – JINR – Alushta – Russia – 2024. – Устный доклад
4. The 28th International Scientific Conference of Young Scientists and Specialists (AYSS-2024) – «Workload Management System Development for SPD Online Filter» – JINR – Dubna – Russia – 2024. – Устный доклад
5. VIII SPD Collaboration Meeting – «SOF Middleware development status» – JINR – Dubna – Russia – 2024. Устный доклад

План работы на 2025 год

□ Обработка заданий и рабочих процессов

- Выполнение всего рабочего процесса, установленного на уровне системы управления процессами;
- Весь рабочий процесс - цепочка зависимых заданий.

□ Интеграция с прикладным программным обеспечением

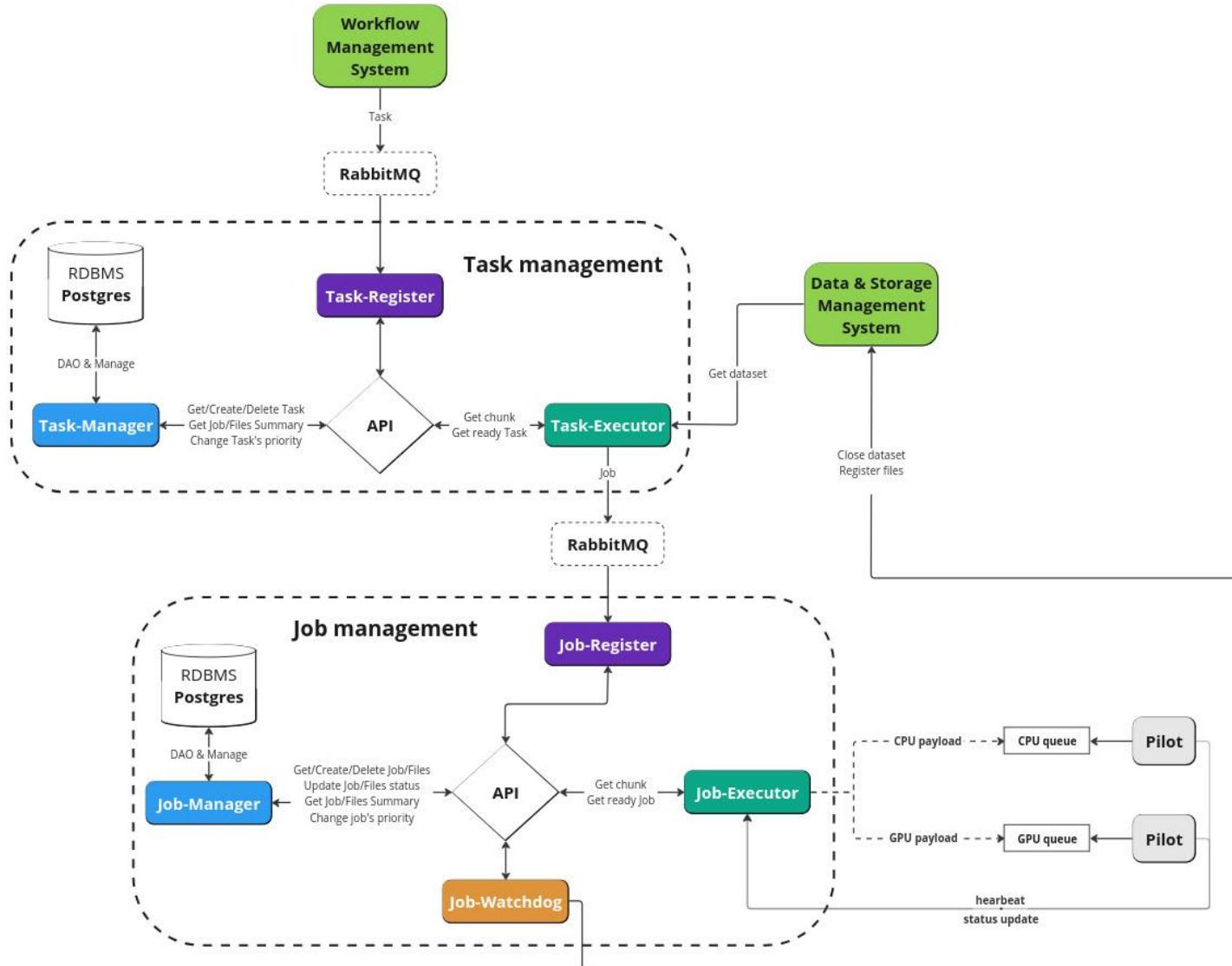
- Требуется прикладное программное обеспечение (в процессе разработки) и смоделированные данные;
- Нефункциональные требования к прикладным программам.

Спасибо за внимание!

Backup slides

Architecture and functionality of 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.



Workload management system requirements - reminder

The key requirement - systems must meet the **high-throughput paradigm**.

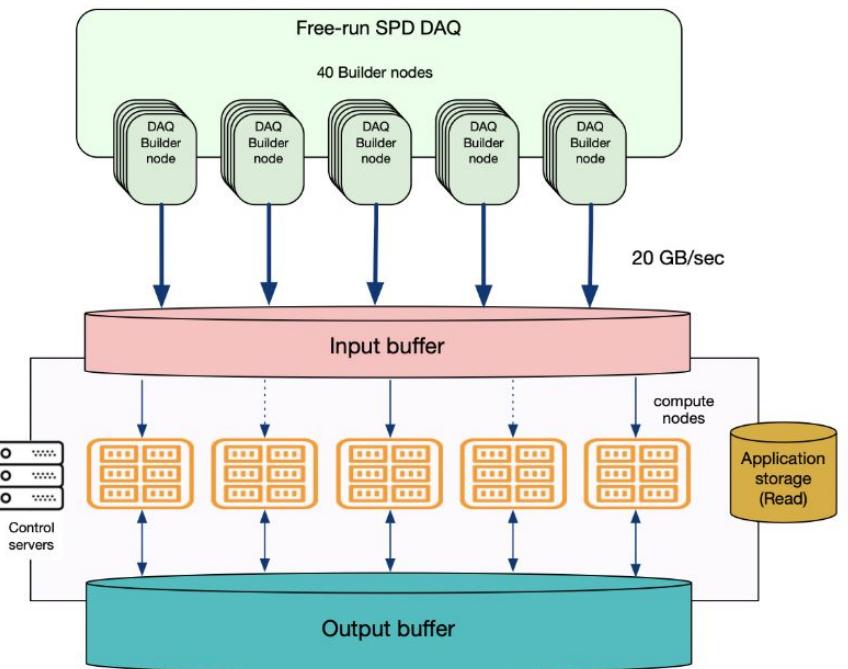
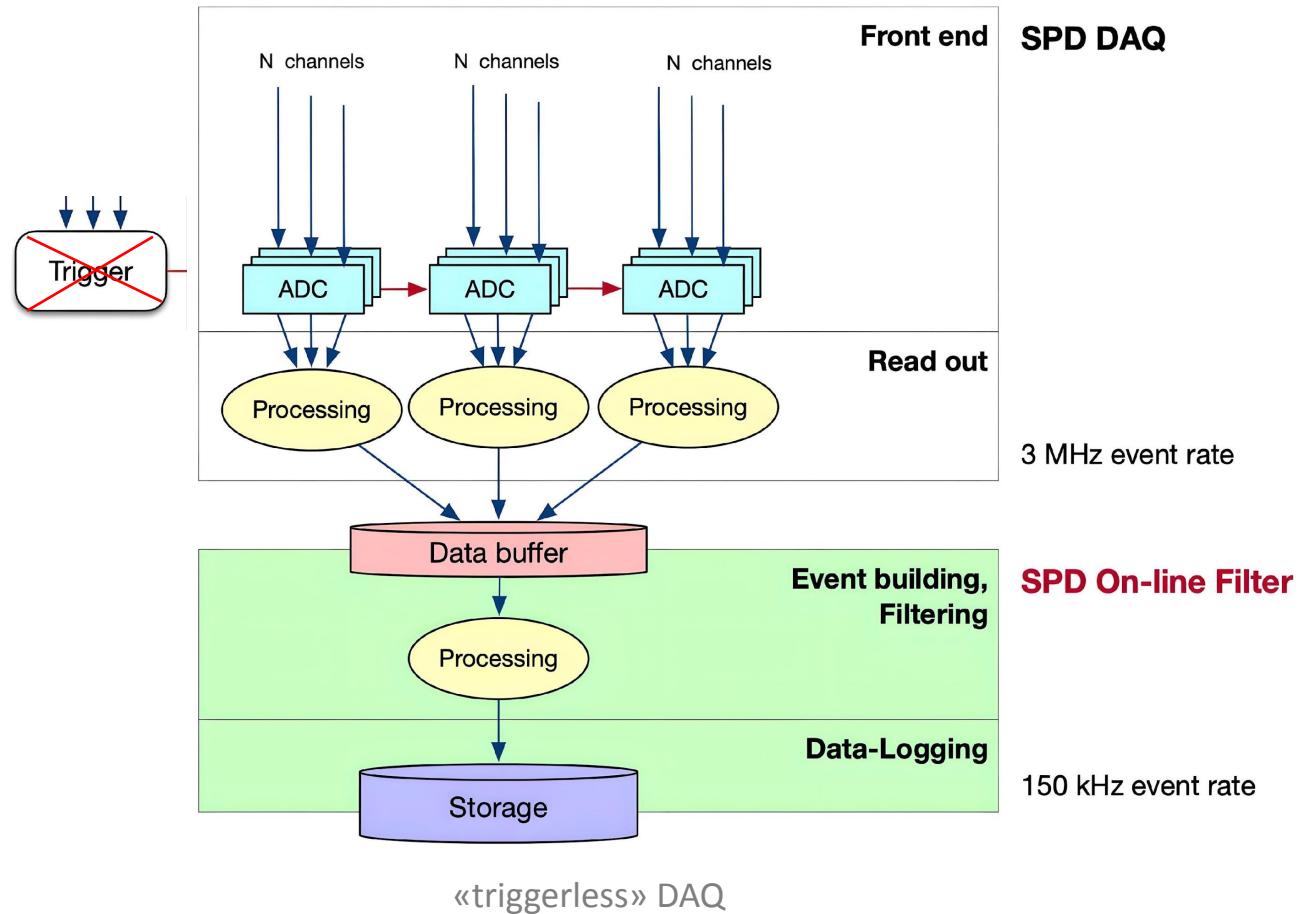
- ❑ **Task registration:** formalized task description, including job options and required metadata registration;
- ❑ **Jobs definition:** generation of required number of jobs to perform task by controlled loading of available computing resources;
- ❑ **Jobs execution management:** continuous job state monitoring by communication with pilot, job retries in case of failures, job execution termination;
- ❑ **Consistency control:** control of the consistency of information in relation to the tasks, files and jobs;
- ❑ **Scheduling:** implementing a scheduling principle for task/job distribution;



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

Triggerless DAQ

Triggerless DAQ means that the output of the system is not a set of raw events, but a set of signals from sub-detectors organized into time slices.



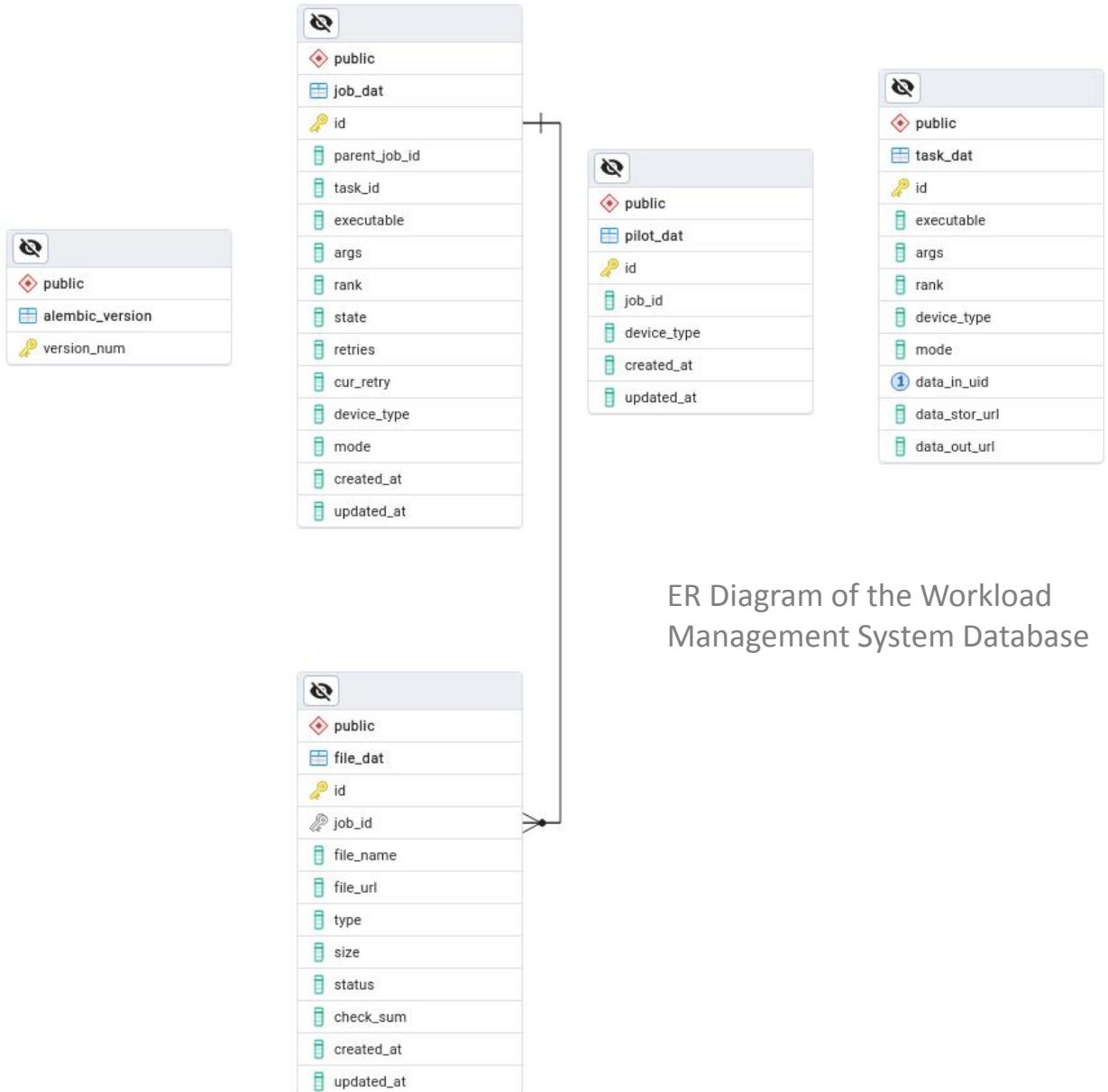
- DAQ provide data organized in time frames which placed in **files** with reasonable size (a few GB).
- Each of these file may be processed independently as a part of top-level **workflow chain**.
- No needs to exchange of any information during handling of each initial file, but results of may be used as input for next step of processing.

Database design

RDBMS - PostgreSQL 16

Tables:

- ❖ **alembic_version** – managing and tracking database schema changes
- ❖ **file_dat** – a directory specifying the output files and logs generated on the pilot
- ❖ **job_dat** – jobs currently being processed in the system
- ❖ **task_dat** – current tasks in the system



Extra mechanisms:

- ❖ **Indexes** – on filter fields for optimization of operations (B-tree);
- ❖ **Procedures** – task and job generation for test purposes;
- ❖ **Triggers** – rank update logic;
- ❖ **Decomposition** – single database per microservice (Postgres in Docker initially)

ER Diagram of the Workload Management System Database

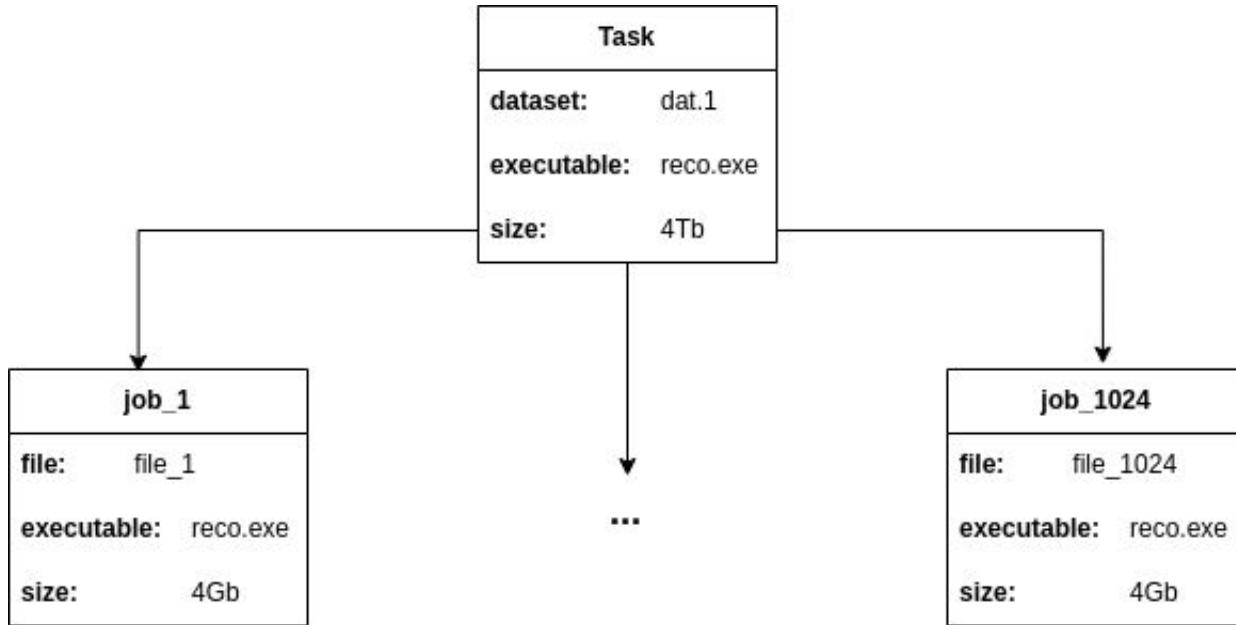
Tech stack

<p>Common</p> <ul style="list-style-type: none"> ➤ Python 3.12 ➤ docker compose - running multi-container applications 	<p>Frameworks</p> <ul style="list-style-type: none"> ➤ aio-pika (RabbitMQ + asyncio) - asynchronous API with RabbitMQ ➤ FastAPI + uvicorn
<p>DB</p> <ul style="list-style-type: none"> ➤ PostgreSQL - RDBMS ➤ Alembic (Migration) ➤ SQLAlchemy 2.0 ➤ asyncpg - Postgres DBAPI 	<p>Extra</p> <ul style="list-style-type: none"> ➤ aiohttp - asynchronous HTTP client/server framework ➤ Pydantic - validate and serialize data schemes ➤ pytest-asyncio - test purposes

Task and job definition

- A **task** is a workload unit responsible for processing a block of homogeneous data - **dataset**.
- A processing request is a set of input data, which may consist of multiple files, and a handler.
- The criterion for the completion of the task is the processing of the entire block of data.
- The **Workflow Management System** is responsible for defining and executing workflows, as well as defining a processing request, which is a **task**.

- A **job** (payload) is a unit of work that processes a unit of data (**file**).
- The unit responsible for processing a single **file** in terms of workload is called a **job**.
- The **Workload Management System** is responsible for generating **jobs**, sending them to compute nodes, and executing them.



Task-job relationship

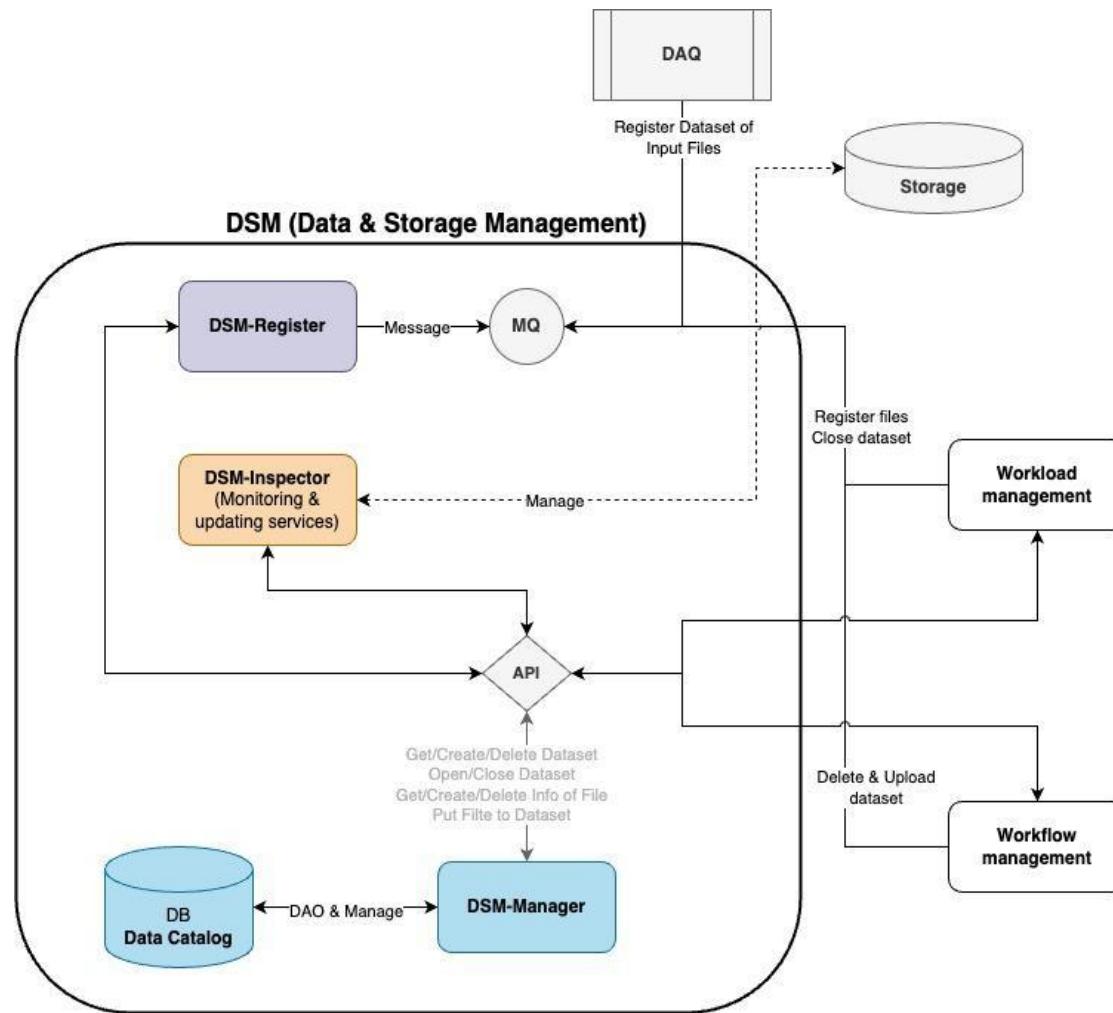
Interaction with Data Management System

Routing Key	Msg	Algo
dataset.close	Dataset info <ul style="list-style-type: none"> • Dataset UID • File check list (file names) 	Request the registered files in the dataset. If they match the checklist, set the status to CLOSED . Otherwise, return the messages back to the queue for deferred execution.
dataset.upload	Dataset UID	Marking dataset for uploading (TO_UPLOAD)
dataset.delete	Dataset UID	Marking dataset for deletion (TO_DELETE)

Signature and algorithm of message receiving gateways for the **dsm-register** service

Within a **Workload Management System**, there are several scenarios for interacting with the data management system:

- Obtain information about dataset contents for forming jobs from **DSM-Manager (Data Catalog REST API)**
- Register files in datasets after executing payload on compute node – **DSM-Register (Data Registration)**
- Close dataset after cancellation or sufficient number of successfully processed files – **DSM-Register***

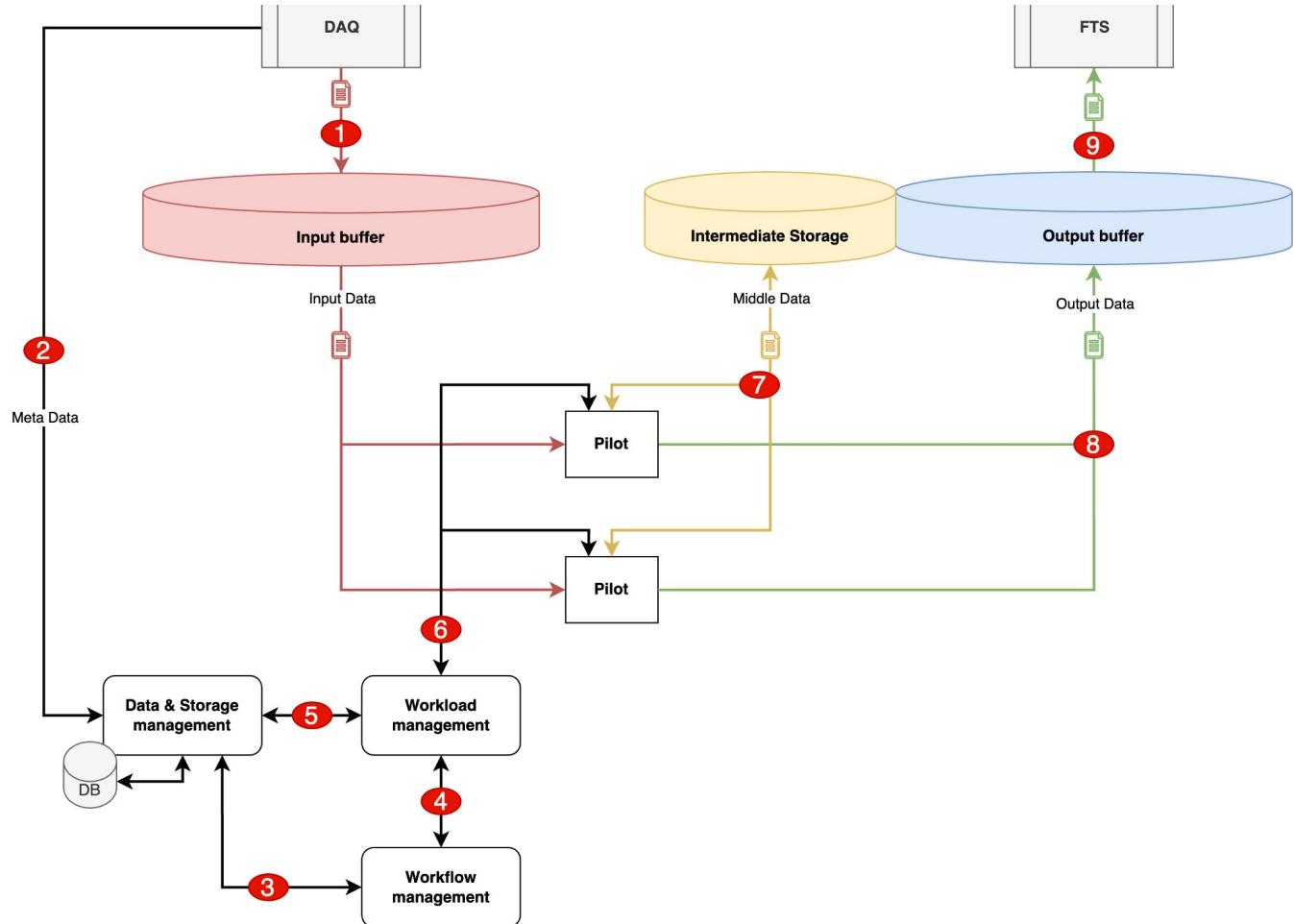


Architecture of Data Management

Dataflow and data processing concept

Main data streams:

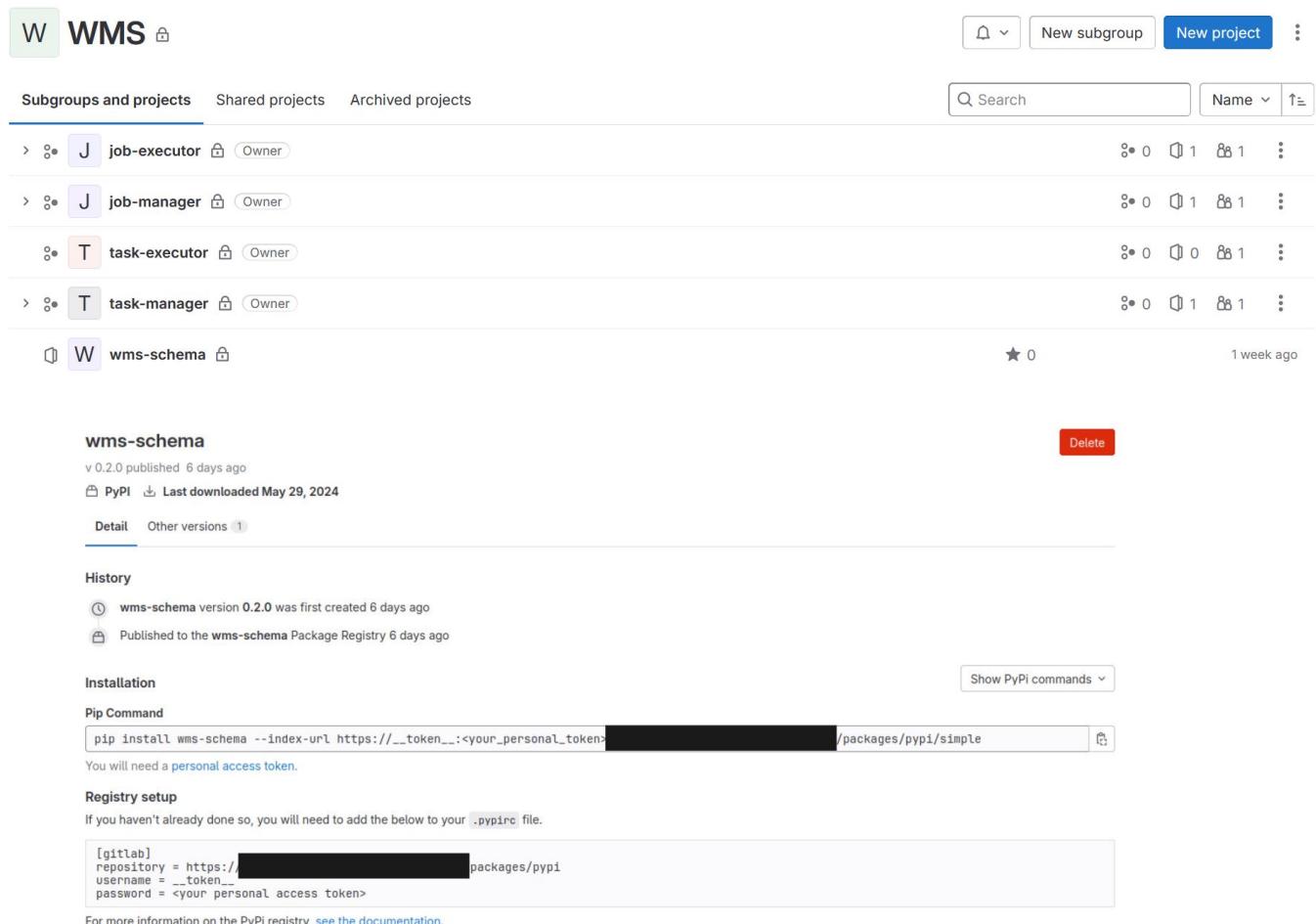
- ❖ SPD DAQs, after dividing sensor signals into time blocks, send data to the SPD Online Filter input buffer as files of a consistent size.
- ❖ The workflow management system creates and deletes intermediate and final data sets
- ❖ The **workload management system** “populates” the data sets with information about the resulting files
- ❖ At each stage of data processing, pilots will read and write files to storage and create secondary data



Modularization: deploying and using own packages

Following tools are used:

- ❖ Poetry
 - Particularly good at handling complex dependency trees and ensuring that the different modules can integrate with each other without version conflicts
- ❖ Python packages
 - Separate GitLab repositories for each package
 - Poetry for packaging and dependency management
- ❖ Gitlab
 - Access *Tokens* used as kind of credentials for scripts and other tools
 - CI/CD for automate testing and building



The screenshot shows a GitLab project named 'wms-schema'. The project page includes a table of contents with sections like 'Subgroups and projects', 'Shared projects', and 'Archived projects'. Below the table, there's a list of files and their details. One file, 'wms-schema', is highlighted. Its details page shows it was published on May 29, 2024, via PyPI. It has a 'Delete' button. The 'Installation' section contains a 'Pip Command' field with the command: `pip install wms-schema --index-url https://__token__:<your_personal_token>@packages/pypi/simple`. A note says you will need a personal access token. The 'Registry setup' section provides a configuration snippet for a .pypirc file:

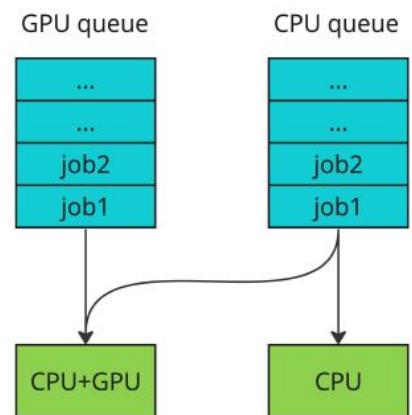
```
[gitlab]
repository = https://__token__:<your_personal_access_token>@packages/pypi
username = __token__
password = <your personal access token>
```

For more information on the PyPi registry, see the documentation.

wms-schema is a package that contains a scheme for task and job data that is used in almost every other service

Interaction with the Pilot Agent

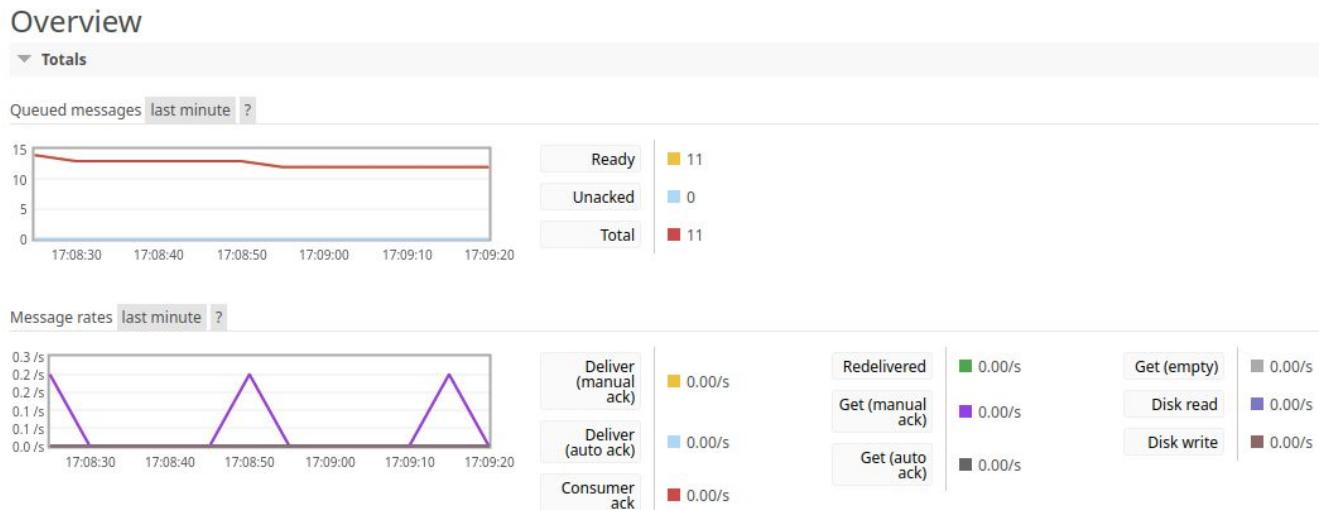
- ❖ Pilot has a series of prepossessing stages before running a job itself:
 - a. start logging
 - b. read configuration
 - c. getting a job from message queue
 - d. validation
 - ❖ After those steps the Pilot launches another thread where it does
 - a. environment setup script
 - b. copying files locally from the input storage
 - c. starts execution of a job itself in a separate sub-process
 - d. analysis of the result of a job
 - e. copying output data and logs to storage
 - f. sends regular messages to **WMS**
 - g. cleaning up the local environment
 - ❖ Pilot sends status-update message at any point of internal changes
 - ❖ **WMS** may terminate the job if the corresponding task is cancelled or if an error occurs.



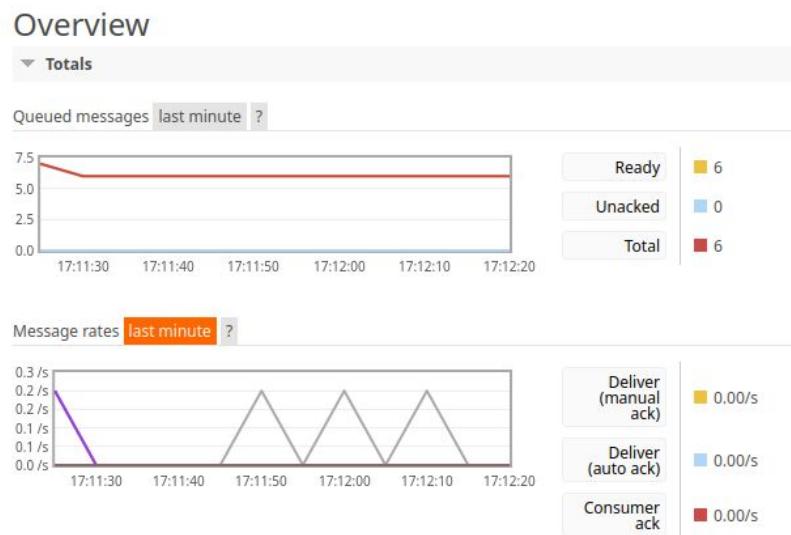
Система управления нагрузкой - Пилот



- ✓ Описана подробная модель состояния задач;
- ✓ Введены коды ошибок;
- ✓ Pilot прошел все этапы выполнения задачи;
- ✓ Демон реализован и запущен;
- Больше нет эмулятора пилота!**
- ✗ Требуется большой цикл тестирования и рефакторинга;
- ✗ Отладка выполнения всего задания (всех задач, связанных с заданием);



UNIX Daemon's running Pilot



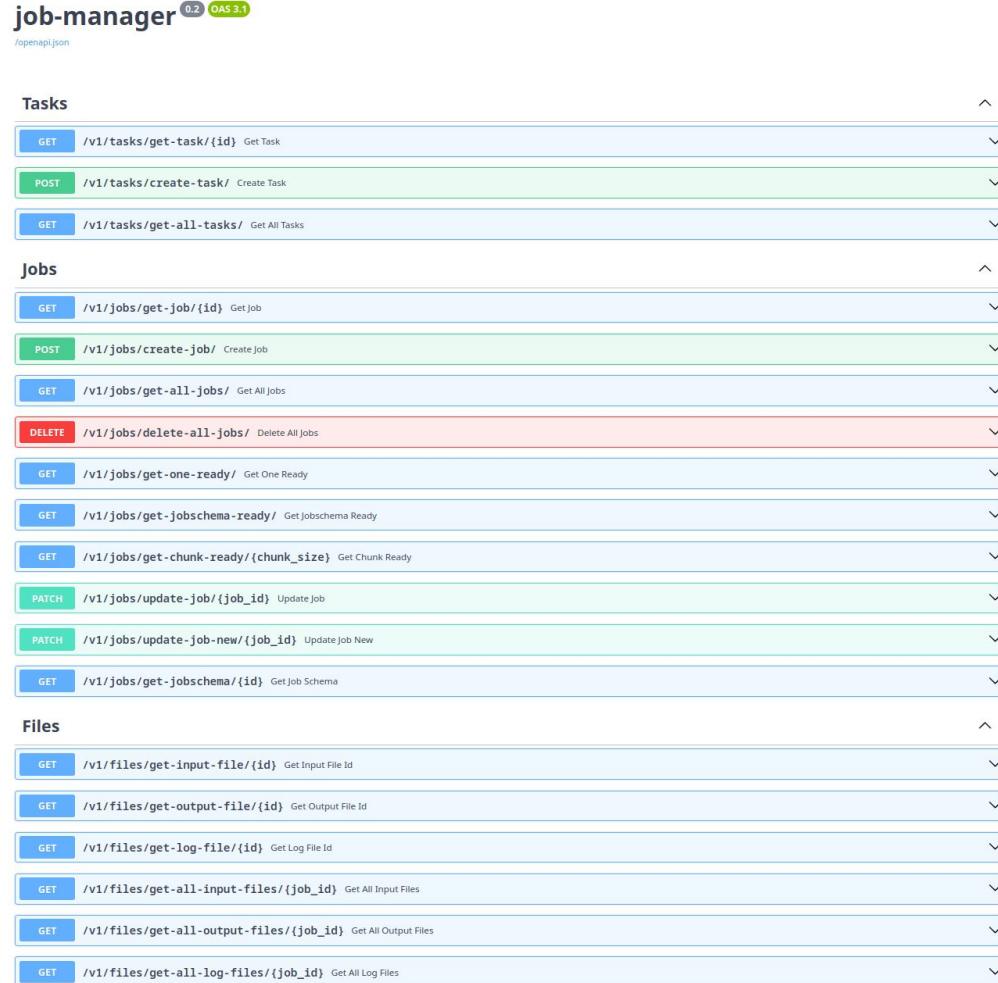
Aftermath of executing the entire queue

What it all looks like

```
spitter-collector-1 | INFO: 2024-11-05 09:50:13,959 | aiohttp.access | web_log.py:211 | 1 >>> [05/Nov/2024:09:50:13 +0000] "POST /status HTTP/1.1" 200 301 "-" "Python-urllib/3.12"
spitter-collector-1 | INFO: 2024-11-05 09:50:15,064 | collector.src.job_manager | job_manager.py:27 [ 1 >>> json={'job_id': '272', 'pilot_id': '1', 'info': '', 'state': 'stage_out', 'error_code': 0, 'exit_code': 0, 'datetime': '2024-11-05 09:50:15.062660', 'files_in_url': [{'id': 353, 'job_id': 272, 'file_name': 'file_in_13', 'file_url': '/data/SPDOF-buffers/input/builder_01/file_in_13', 'type': 'input', 'size': 26, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'finished', 'error_code': 0, 'info': ''}, {'id': 173, 'job_id': 272, 'file_name': 'file_out.task_test1.job_272_12', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_out.task_test1.job_272_12', 'type': 'output', 'size': 894, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'ready', 'error_code': 0, 'info': ''}, {'id': 405, 'job_id': 272, 'file_name': 'file_out.task_test1.job_272_28', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_out.task_test1.job_272_28', 'type': 'output', 'size': 412, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'ready', 'error_code': 0, 'info': ''}, {'id': 978, 'job_id': 272, 'file_name': 'file_out.task_test1.job_272_15', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_out.task_test1.job_272_15', 'type': 'output', 'size': 419, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'ready', 'error_code': 0, 'info': ''}, {'id': 482, 'job_id': 272, 'file_name': 'file_out.task_test1.job_272_13', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_out.task_test1.job_272_13', 'type': 'output', 'size': 99, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'ready', 'error_code': 0, 'info': ''}], 'files_log_url': [{"id": 923, 'job_id': 272, 'file_name': 'file_log.task_test1.job_272.tar.gz', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_log.task_test1.job_272.tar.gz', 'type': 'log', 'size': 809, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'finished', 'error_code': 0, 'info': ''}], 'metrics': {'info': 'Job metrics', 'cpu_load': 0.3, 'memory_load': 11.2, 'network_load': 117138999, 'io': 12211}, type=<class 'dict'>
spitter-collector-1 | INFO: 2024-11-05 09:50:15,065 | aiohttp.access | web_log.py:211 | 1 >>> [05/Nov/2024:09:50:15 +0000] "POST /status HTTP/1.1" 200 295 "-"
spitter-collector-1 | INFO: 2024-11-05 09:50:16,170 | collector.src.job_manager | job_manager.py:27 [ 1 >>> json={'job_id': '272', 'pilot_id': '1', 'info': '', 'state': 'completed', 'error_code': 0, 'exit_code': 0, 'datetime': '2024-11-05 09:50:16.169291', 'files_in_url': [{'id': 353, 'job_id': 272, 'file_name': 'file_in_13', 'file_url': '/data/SPDOF-buffers/input/builder_01/file_in_13', 'type': 'input', 'size': 26, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'finished', 'error_code': 0, 'info': ''}, {'files_out_url': [{"id": 173, 'job_id': 272, 'file_name': 'file_out.task_test1.job_272_12', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_out.task_test1.job_272_12', 'type': 'output', 'size': 894, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'finished', 'error_code': 0, 'info': ''}, {"id": 405, 'job_id': 272, 'file_name': 'file_out.task_test1.job_272_28', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_out.task_test1.job_272_28', 'type': 'output', 'size': 412, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'ready', 'error_code': 0, 'info': ''}, {"id": 978, 'job_id': 272, 'file_name': 'file_out.task_test1.job_272_15', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_out.task_test1.job_272_15', 'type': 'output', 'size': 419, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'finished', 'error_code': 0, 'info': ''}, {"id": 482, 'job_id': 272, 'file_name': 'file_out.task_test1.job_272_13', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_out.task_test1.job_272_13', 'type': 'output', 'size': 99, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'ready', 'error_code': 0, 'info': ''}], 'files_log_url': [{"id": 923, 'job_id': 272, 'file_name': 'file_log.task_test1.job_272.tar.gz', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_log.task_test1.job_272.tar.gz', 'type': 'log', 'size': 809, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'finished', 'error_code': 0, 'info': ''}], 'metrics': {'info': 'Job metrics', 'cpu_load': 0.3, 'memory_load': 11.2, 'network_load': 117138999, 'io': 12211}, type=<class 'dict'>
spitter-collector-1 | INFO: 2024-11-05 09:50:16,170 | aiohttp.access | web_log.py:211 | 1 >>> [05/Nov/2024:09:50:16 +0000] "POST /status HTTP/1.1" 200 295 "-"
spitter-collector-1 | INFO: 2024-11-05 09:50:16,170 | collector.src.job_manager | job_manager.py:27 [ 1 >>> json={'job_id': '272', 'pilot_id': '1', 'info': '', 'state': 'completed', 'error_code': 0, 'exit_code': 0, 'datetime': '2024-11-05 09:50:16.169291', 'files_in_url': [{"id": 353, 'job_id': 272, 'file_name': 'file_in_13', 'file_url': '/data/SPDOF-buffers/input/builder_01/file_in_13', 'type': 'input', 'size': 26, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'finished', 'error_code': 0, 'info': ''}, {"files_out_url": [{"id": 173, 'job_id': 272, 'file_name': 'file_out.task_test1.job_272_12', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_out.task_test1.job_272_12', 'type': 'output', 'size': 894, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'finished', 'error_code': 0, 'info': ''}, {"id": 405, 'job_id': 272, 'file_name': 'file_out.task_test1.job_272_28', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_out.task_test1.job_272_28', 'type': 'output', 'size': 412, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'finished', 'error_code': 0, 'info': ''}, {"id": 978, 'job_id': 272, 'file_name': 'file_out.task_test1.job_272_15', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_out.task_test1.job_272_15', 'type': 'output', 'size': 419, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'finished', 'error_code': 0, 'info': ''}, {"id": 482, 'job_id': 272, 'file_name': 'file_out.task_test1.job_272_13', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_out.task_test1.job_272_13', 'type': 'output', 'size': 99, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'finished', 'error_code': 0, 'info': ''}], 'files_log_url': [{"id": 923, 'job_id': 272, 'file_name': 'file_log.task_test1.job_272.tar.gz', 'file_url': '/data/SPDOF-buffers/output/task_822/job_272/file_log.task_test1.job_272.tar.gz', 'type': 'log', 'size': 809, 'check_sum': 'e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855', 'state': 'finished', 'error_code': 0, 'info': ''}], 'metrics': {'info': 'Job metrics', 'cpu_load': 2.2, 'memory_load': 11.2, 'network_load': 117278374, 'io': 12211}, type=<class 'dict'>
spitter-collector-1 | INFO: 2024-11-05 09:50:16,171 | aiohttp.access | web_log.py:211 | 1 >>> [05/Nov/2024:09:50:16 +0000] "POST /status HTTP/1.1" 200 295 "-> Python-urllib/3.12"
```

Prototyping Job-Manager (API)

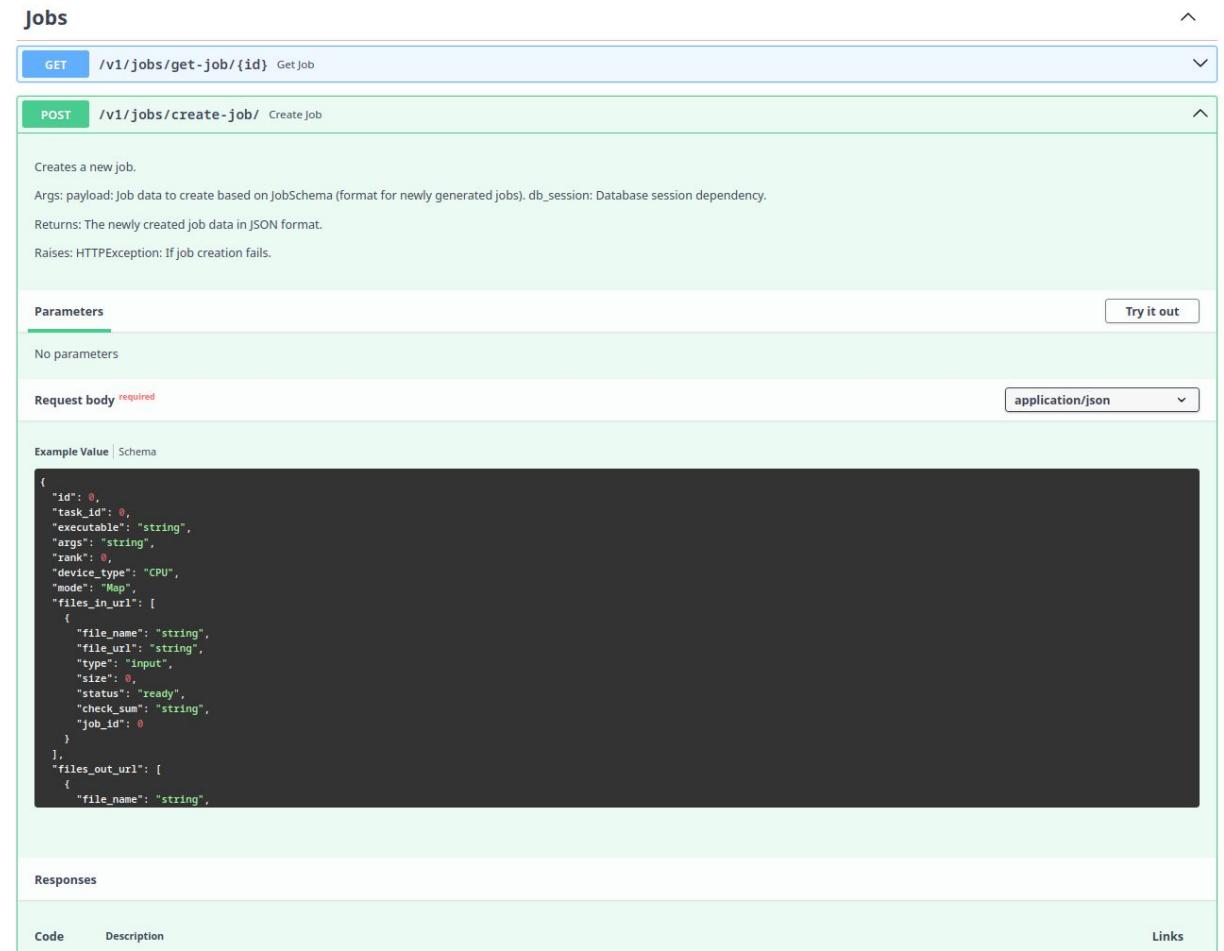
- The chosen framework for building the service is FastAPI + Unicorn asynchronous framework
- A basic set of CRUD operations on data in the form of REST API is developed.
- API description autogeneration according to OpenAPI 3.0 specification is implemented (available in Swagger UI at <server address>/docs)



The screenshot shows the Swagger UI interface for the job-manager service. It displays a hierarchical list of API endpoints categorized by resource type:

- Tasks** (GET /v1/tasks/get-task/{id}, POST /v1/tasks/create-task/, GET /v1/tasks/get-all-tasks/)
- Jobs** (GET /v1/jobs/get-job/{id}, POST /v1/jobs/create-job/, GET /v1/jobs/get-all-jobs/, DELETE /v1/jobs/delete-all-jobs/, GET /v1/jobs/get-one-ready/, GET /v1/jobs/get-jobschema-ready/, GET /v1/jobs/get-chunk-ready/{chunk_size}, PATCH /v1/jobs/update-job/{job_id}, PATCH /v1/jobs/update-job-new/{job_id}, GET /v1/jobs/get-jobschema/{id})
- Files** (GET /v1/files/get-input-file/{id}, GET /v1/files/get-output-file/{id}, GET /v1/files/get-log-file/{id}, GET /v1/files/get-all-input-files/{job_id}, GET /v1/files/get-all-output-files/{job_id}, GET /v1/files/get-all-log-files/{job_id})

Swagger UI with job-manager service API description



The screenshot shows a detailed view of the 'Create Job' endpoint from the Swagger UI. The endpoint is defined as follows:

```

    POST /v1/jobs/create-job/ Create Job
    Creates a new job.
    Args: payload: Job data to create based on JobSchema (format for newly generated jobs). db_session: Database session dependency.
    Returns: The newly created job data in JSON format.
    Raises: HTTPException: If job creation fails.
  
```

Parameters: No parameters

Request body (required): application/json

Example Value | Schema

```

{
  "id": 0,
  "task_id": 0,
  "executable": "string",
  "args": "string",
  "rank": 0,
  "device_type": "CPU",
  "mode": "Map",
  "files_in_url": [
    {
      "file_name": "string",
      "file_url": "string",
      "type": "input",
      "size": 0,
      "status": "ready",
      "check_sum": "string",
      "job_id": 0
    }
  ],
  "files_out_url": [
    {
      "file_name": "string",
      "file_url": "string"
    }
  ]
}
  
```

Responses

Code	Description

Links

Example of a service call to post a new job

Prototyping Job-Executor - Pilot (RabbitMQ queues)

- RabbitMQ is selected as the message broker
- Queues are defined using the declarative notation of the aio-pika tool
- At the start of the application their unfolding is performed

 RabbitMQ™ RabbitMQ 3.11.28 Erlang 25.3.2.9

Overview Connections Channels Exchanges **Exchanges** Queues Admin

Exchange: jobs

Message rates last minute ?

1.0 /s Publish (In) 0.00/s
0.0 /s Publish (Out) 0.00/s

Details

Type	direct
Features	
Policy	

Bindings

This exchange

↓

To	Routing key	Arguments	Unbind
pilot-cpu	CPU		Unbind
pilot-gpu	GPU		Unbind

Configured RabbitMQ queues

Add binding from this exchange

To queue: *

Routing key:

Arguments: = String

Bind

Publish message

Routing key: CPU

Headers: ? = String

Properties: ? = String

Payload:

```
{
  "id": 11,
  "task_id": 44,
  "executable": "mock_job.py",
  "args": "-in file_in_1157 file_in_1158 -out file_out_1157 file_out_1158",
  "rank": 1,
  "device_type": "CPU",
  "mode": "Map",
  "files_in_url": [
    {
      "file_name": "file_in_1157",
      "file_url": "/data/SPDOF-buffers/input/file_in_1157",
      "type": "INPUT",
      "size": 1073741824,
      "status": {"state": "ready", "error_code": "", "info": "File is ready to be processed"},
      "check_sum": "a9f8c7e6-4a9b-4f0a-8f2c-9c4a9a3f0f0f"
    },
    {
      "file_name": "file_in_1158",
      "file_url": "/data/SPDOF-buffers/input/file_in_1158",
      "type": "INPUT",
      "size": 1073741824,
      "status": {"state": "ready", "error_code": "", "info": "File is ready to be processed"},
      "check_sum": "3fa85f64-5717-4562-b3fc-2c963f66afa7"
    }
  ],
  "files_out_url": [
    {
      "file_name": "file_out_1157",
      "file_url": "/data/SPDOF-buffers/output/file_out_1157",
      "type": "OUTPUT",
      "size": 0,
      "status": {"state": "ready", "error_code": "", "info": "This file should be created"}
    }
  ]
}
```

Payload encoding: String (default)

Publish message

Jobs could be delivered manually

Examples of Templates and Tasks



- Registration and authorization
 - Template and task output
 - CWL template creation by user
 - Preliminary validation and writing of CWL templates to the database

Template Manager				Templates	Tasks	a@aaa.aaa	Logout		
				Create template					
template_id	name	inner_dataset_mask	description				status		
1	template1	.test.	{"steps": {"decoding": {"run": {"class": "CommandLineTool", "baseCommand": "echo", "inputs": {"dataset_name": {"type": "string"}, "processing_program": {"type": "string"}, "processing_program_version": {"type": "string"}, "cable_map": {"type": "File"}, "input_params": {"type": "File"}}, "outputs": {"output_dataset": {"type": "File"}, "log_dataset": {"type": "File"}}, "in": {"dataset_name": ".test."}, "processing_program": "processing_program", "processing_program_version": "processing_program_version", "cable_map": "cable_map", "input_params": "input_params"}, "out": {"[output_dataset, log_dataset]"}, "reconstruction": {"run": {"class": "CommandLineTool", "baseCommand": "echo", "inputs": {"dataset_name": {"type": "string"}, "processing_program": {"type": "string"}, "processing_program_version": {"type": "string"}, "cable_map": {"type": "File"}, "input_params": {"type": "File"}}, "outputs": {"output_dataset": {"type": "File"}, "log_dataset": {"type": "File"}}, "in": {"dataset_name": ".test."}, "processing_program": "processing_program", "processing_program_version": "processing_program_version", "cable_map": "cable_map", "input_params": "input_params"}, "out": "[output_dataset, log_dataset]"}}				ACTUAL		
2	template2	.test.	{"steps": {"decoding": {"run": {"class": "CommandLineTool", "baseCommand": "echo", "inputs": {"dataset_name": {"type": "string"}, "processing_program": {"type": "string"}, "processing_program_version": {"type": "string"}, "cable_map": {"type": "File"}, "input_params": {"type": "File"}}, "outputs": {"output_dataset": {"type": "File"}, "log_dataset": {"type": "File"}}, "in": {"dataset_name": ".test."}, "processing_program": "processing_program", "processing_program_version": "processing_program_version", "cable_map": "cable_map", "input_params": "input_params"}, "out": "[output_dataset, log_dataset]"}}				ARCHIVED		

Created template

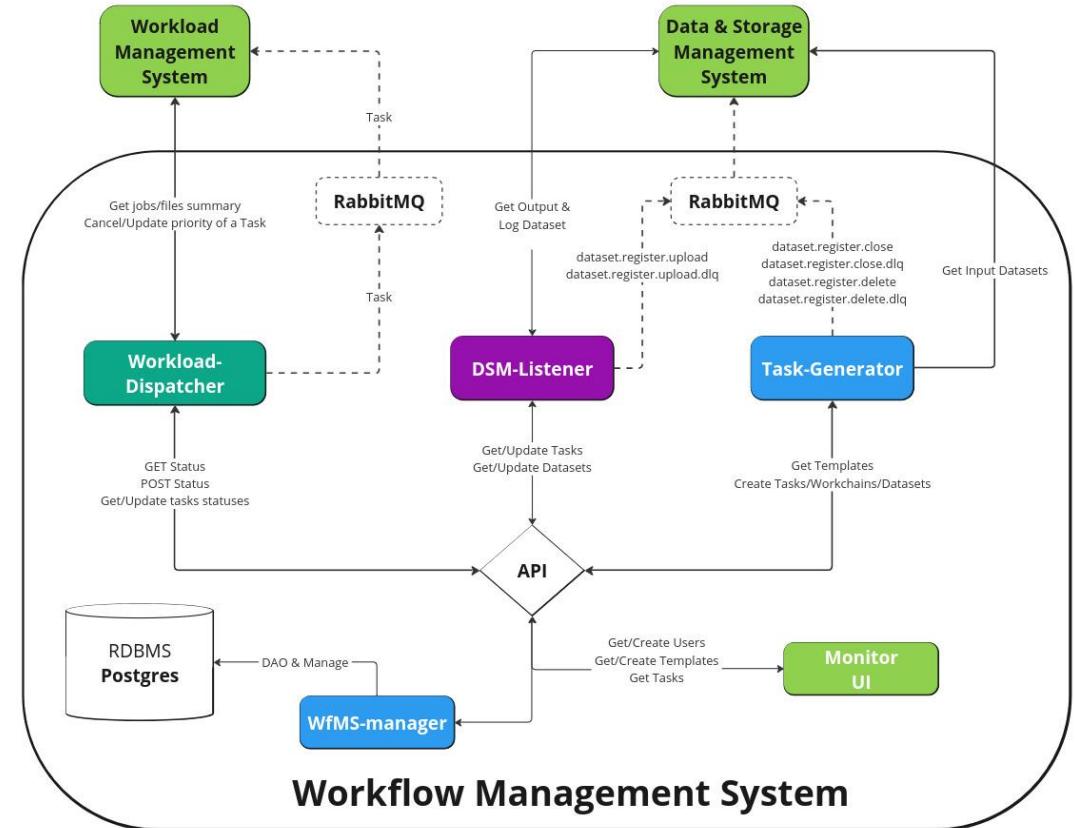
Template Manager				Templates			Tasks					Actions	
task_id	wflow_id	exec	args	rank	device	mode	retry	datas_in_id	datas_out_id	datas_log_id	status		
11	6	processing_program	cable_map	1	CPU	map	5	26	27	28	IN_PROGRESS	View	Edit
12	6	processing_program	cable_map	1	CPU	map	5	27	29	30	IN_PROGRESS	View	Edit
13	7	processing_program	cable_map	1	CPU	map	5	31	32	33	IN_PROGRESS	View	Edit
14	7	processing_program	cable_map	1	CPU	map	5	32	34	35	IN_PROGRESS	View	Edit
15	8	processing_program	cable_map	1	CPU	map	5	36	37	38	IN_PROGRESS	View	Edit
16	8	processing_program	cable_map	1	CPU	map	5	37	39	40	IN_PROGRESS	View	Edit
17	9	processing_program	cable_map	1	CPU	map	5	41	42	43	IN_PROGRESS	View	Edit
18	9	processing_program	cable_map	1	CPU	map	5	42	44	45	IN_PROGRESS	View	Edit
19	10	processing_program	cable_map	1	CPU	map	5	46	47	48	IN_PROGRESS	View	Edit
20	10	processing_program	cable_map	1	CPU	map	5	47	49	50	IN_PROGRESS	View	Edit
21	11	processing_program	cable_map	1	CPU	map	5	51	52	53	IN_PROGRESS	View	Edit
22	11	processing_program	cable_map	1	CPU	map	5	52	54	55	IN_PROGRESS	View	Edit
23	12	processing_program	cable_map	1	CPU	map	5	56	57	58	IN_PROGRESS	View	Edit
24	12	processing_program	cable_map	1	CPU	map	5	57	59	60	IN_PROGRESS	View	Edit

WfMS task description

Interaction with Workflow Management System

The following interaction scenarios have been identified with the Workflow Management System

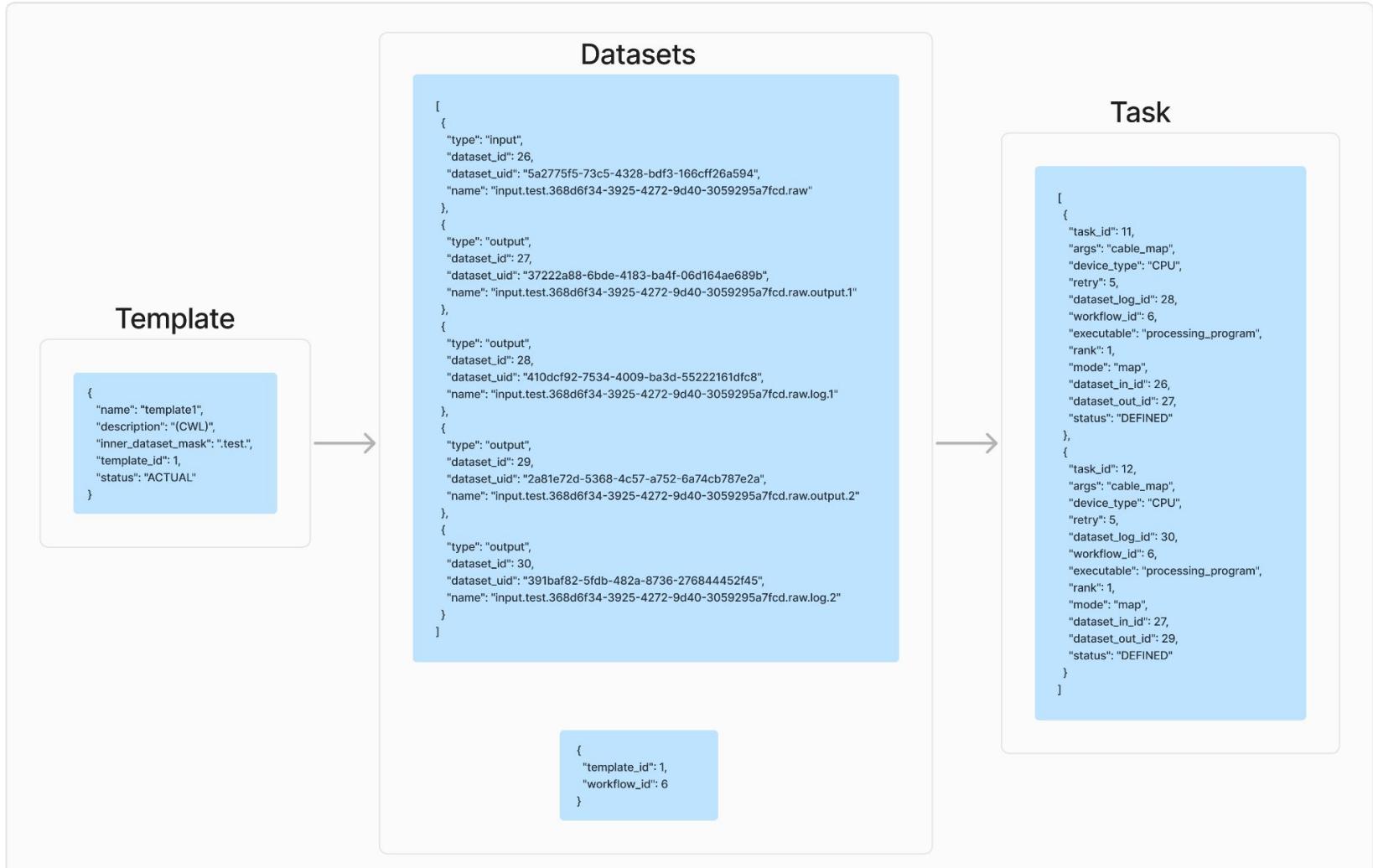
- Registration of a task for processing: **WfMS** passes the task description into the message queue;
- Summary of current intermediate properties of jobs/files in the system: aggregated information about the status of each job/file for further decision making;
- Task cancellation: based on the decision made on the **WfMS** (*too many errors occurring*) on operator side;
- Change priority of a task: is used to accelerate the rate at which the corresponding dataset is being processed;



Task generation service

1. Getting registered datasets from **DMS** from RabbitMQ;
2. Matching datasets by name mask to the desired template;
3. Registration of input dataset in the system;
4. Creating a workflow from a template;
5. Creating output dataset and log dataset in the system;
6. Creating a task;

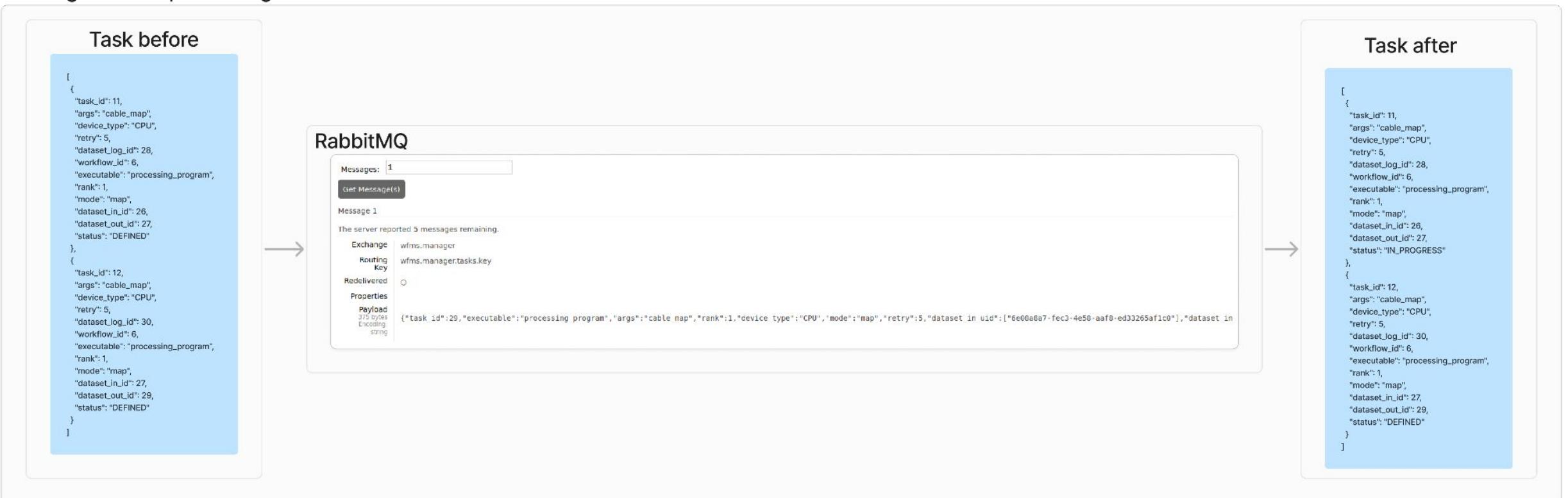
From template to tasks



Task management service

- Iterate on tasks in “DEFINED” status;
- Querring DMS about the status of the input dataset (“CLOSED”);
- Creating output datasets and log datasets in DMS;
- Sending the task to RabbitMQ for further processing in WMS;
- Change task status to "IN_PROGRESS".

Sending tasks for processing



Data access service

- Data access service is implemented and provides all necessary endpoints at this stage;
- Test coverage is required;

Templates		Workflows		auth	
<code>GET</code>	/template/all Get All Templates	<code>GET</code>	/workflow/all Get All Workflows	<code>POST</code>	/auth/jwt/login Auth:Jwt.Login
<code>GET</code>	/template/actual Get Actual Templates	<code>GET</code>	/workflow/{workflow_id} Workflow Response	<code>POST</code>	/auth/jwt/logout Auth:Jwt.Logout
<code>GET</code>	/template/{template_id} Template Response	<code>POST</code>	/workflow/create Create Workflow	<code>POST</code>	/auth/register Register:Register
<code>POST</code>	/template/create Create Template	<code>GET</code>	/task/all Get All Tasks	<code>POST</code>	/auth/forgot-password Reset:Forgot Password
<code>PUT</code>	/template/{template_id}/change Change Template	<code>GET</code>	/task/{task_id} Task Response	<code>POST</code>	/auth/reset-password Reset:Reset Password
<code>DELETE</code>	/template/{template_id}/delete Delete Template	<code>GET</code>	/task/status/{status_name} Get Defined Tasks	<code>POST</code>	/auth/request-verify-token Verify:Request-Token
Datasets		Tasks		users	
<code>GET</code>	/dataset/all Get All Datasets	<code>POST</code>	/task/create Create Task	<code>GET</code>	/users/me Users:Current User
<code>GET</code>	/dataset/{dataset_id} Dataset Response	<code>PUT</code>	/task/{task_id}/rank Change Rank	<code>PATCH</code>	/users/me Users:Patch Current User
<code>POST</code>	/dataset/create Create Dataset	<code>PUT</code>	/task/{task_id}/status Change Rank	<code>GET</code>	/users/{id} Users:User
<code>PUT</code>	/dataset/{dataset_id}/dataset_uid Change Rank	<code>DELETE</code>	/task/{task_id}/delete Delete Task	<code>PATCH</code>	/users/{id} Users:Patch User
				<code>DELETE</code>	/users/{id} Users:Delete User

Description of the current implemented API's

Service for interaction with user

- Registration and authorization of users with different rights;
- CWL template/tasks output;
- Creation of CWL templates by superuser;
- CWL template status changes by superuser;
- Store template in the database;
- FastAPI Users
- JWT-token

Template Manager

[Login](#) [Registration](#)

Registration

[Complete registration](#)

Template Manager

[Templates](#) [Tasks](#)

a@aaa.aaa [Logout](#)

CWL Template

Enter CWL here...

[Complete](#)

Examples of Templates and Tasks

- Viewing templates and tasks is available to all users who have completed the authorization process;
- Template creation is only available to superusers;

Template Manager

Templates				Tasks			
				a@aaa.aaa	Logout		
Create template							
template_id	name	inner_dataset_mask	description	status			
1	template1	.test.	<pre>{"steps": {"decoding": {"run": {"class": "CommandLineTool", "baseCommand": "echo", "inputs": [{"dataset_name": {"type": "string"}, "processing_program": {"type": "string"}, "processing_program_version": {"type": "string"}, "cable_map": {"type": "File"}, "input_params": {"type": "File"}}, "outputs": {"output_dataset": {"type": "File"}, "log_dataset": {"type": "File"}}, "in": {"dataset_name": ".test.", "processing_program": "processing_program", "processing_program_version": "processing_program_version", "cable_map": "cable_map", "input_params": "input_params"}, "out": "[output_dataset, log_dataset]"}, "reconstruction": {"run": {"class": "CommandLineTool", "baseCommand": "echo", "inputs": [{"dataset_name": {"type": "string"}, "processing_program": {"type": "string"}, "processing_program_version": {"type": "string"}, "cable_map": {"type": "File"}, "input_params": {"type": "File"}}, "outputs": {"output_dataset": {"type": "File"}, "log_dataset": {"type": "File"}}, "in": {"dataset_name": ".test.", "processing_program": "processing_program", "processing_program_version": "processing_program_version", "cable_map": "cable_map", "input_params": "input_params"}, "out": "[output_dataset, log_dataset]"]}}</pre>	ACTUAL			
2	template2	.test.	<pre>{"steps": {"decoding": {"run": {"class": "CommandLineTool", "baseCommand": "echo", "inputs": [{"dataset_name": {"type": "string"}, "processing_program": {"type": "string"}, "processing_program_version": {"type": "string"}, "cable_map": {"type": "File"}, "input_params": {"type": "File"}}, "outputs": {"output_dataset": {"type": "File"}, "log_dataset": {"type": "File"}}, "in": {"dataset_name": ".test.", "processing_program": "processing_program", "processing_program_version": "processing_program_version", "cable_map": "cable_map", "input_params": "input_params"}, "out": "[output_dataset, log_dataset]"]}}</pre>	ARCHIVED			

Created template

Template Manager

Templates				Tasks							
				a@aaa.aaa	Logout						
task_id	wflow_id	exec	args	rank	device	mode	retry	datas_in_id	datas_out_id	datas_log_id	status
11	6	processing_program	cable_map	1	CPU	map	5	26	27	28	IN_PROGRESS
12	6	processing_program	cable_map	1	CPU	map	5	27	29	30	IN_PROGRESS
13	7	processing_program	cable_map	1	CPU	map	5	31	32	33	IN_PROGRESS
14	7	processing_program	cable_map	1	CPU	map	5	32	34	35	IN_PROGRESS
15	8	processing_program	cable_map	1	CPU	map	5	36	37	38	IN_PROGRESS
16	8	processing_program	cable_map	1	CPU	map	5	37	39	40	IN_PROGRESS
17	9	processing_program	cable_map	1	CPU	map	5	41	42	43	IN_PROGRESS
18	9	processing_program	cable_map	1	CPU	map	5	42	44	45	IN_PROGRESS
19	10	processing_program	cable_map	1	CPU	map	5	46	47	48	IN_PROGRESS
20	10	processing_program	cable_map	1	CPU	map	5	47	49	50	IN_PROGRESS
21	11	processing_program	cable_map	1	CPU	map	5	51	52	53	IN_PROGRESS
22	11	processing_program	cable_map	1	CPU	map	5	52	54	55	IN_PROGRESS
23	12	processing_program	cable_map	1	CPU	map	5	56	57	58	IN_PROGRESS
24	12	processing_program	cable_map	1	CPU	map	5	57	59	60	IN_PROGRESS

WfMS task description

RabbitMQ configured queues

Exchange: dsm.register

▶ Overview

▼ Bindings

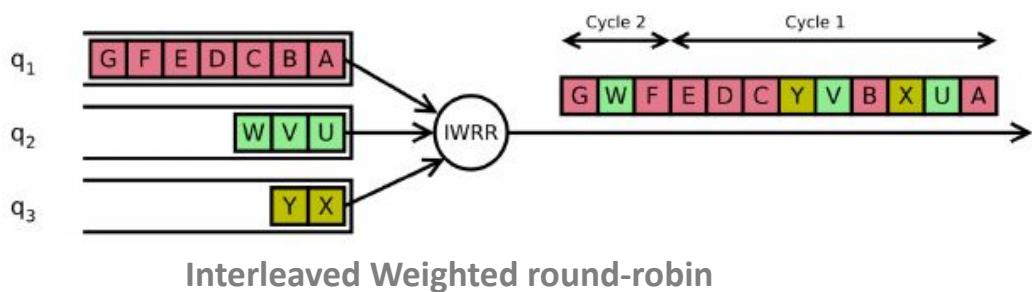
This exchange			
To	Routing key	Arguments	
dsm.register.dataset.close	dataset.close		<button>Unbind</button>
dsm.register.dataset.delete	dataset.delete		<button>Unbind</button>
dsm.register.dataset.input	dataset.input		<button>Unbind</button>
dsm.register.dataset.upload	dataset.upload		<button>Unbind</button>
dsm.register.file.input	file.input		<button>Unbind</button>
dsm.register.file.process	file.process		<button>Unbind</button>
dsm.register.file.process.reply	file.process.reply		<button>Unbind</button>

Exchange	Routing Key	Appointment
	file.input	Receiving information about incoming files to the input buffer
dsm.register	file.process (direct)	Receiving information about new files, received during processing
	dataset.close	Accepting a request to close a dataset
	dataset.upload	Accepting an application to upload files in a dataset to an external storage
	dataset.delete	Accepting a request to delete files in a dataset on the internal storage

- Jobs scheduling algorithm
- Partitioning of a task
 - Imagine a multitasking operating system.
 - Each dataset represents a process, and each record within a dataset is like a thread within that process.
 - The algorithm acts as the operating system's scheduler, allocating processing time to threads based on their priority.
- Chunk size and rank/priority of a job as a basic control unit:

$$\text{rank}_{i+1} = \alpha \times x_i + \beta \times y_i + \gamma \times \text{rank}_i$$

x_i – aging, y_i – retries



Algorithm 1 Task Scheduling Algorithm

Variables:

global_queue – global queue with tasks
 dataset – array of datasets
 N – number of datasets
 rank_max – maximum task priority
 heap – binary heap storing maximum task priorities
 rank – array with task priorities

Algorithm:

```

1: initialize_datasets(dataset)
2: build_heap(rank)
3: while true do
4:   rank_max = heap.top()
5:   for r = 1 to rank_max do
6:     for i = 1 to N do
7:       if not dataset[i].chunk.empty() and rank[i] ≥ r then
8:         await dataset[i].chunk.cur_item
9:         update(dataset[i].chunk - i cur_item)
10:        else if dataset[i].chunk.empty() then
11:          if dataset[i].chunk.cur_item then
12:            dataset[i] = global_queue.head()
13:          end if
14:          update(rank[i])
15:          update(heap)
16:        end if
17:      end for
18:    end for
19:  end while
  
```

Proposed task-partitioning algorithm