



# EXA4MIND

## Supercomputing with versatile data backends (2)

Stuart Gordon(BADW-LRZ)



Funded by  
the European Union

Visit us at [exa4mind.eu](https://exa4mind.eu)

# How do we apply the AQIS and create a data-driven workflow?

- The AQIS is the core module enabling:
  - Extreme Data workflows in the EXA4MIND architecture.
- The AQIS engine
  - is a submodule specifically set-up for processing workflows using extreme data
- We focus on:
  - Workflow orchestration
  - Parallelization within workflow
  - How to manage access to different data backends (AQIS Data System Adaptor submodules)

# How do we apply the AQIS and create a data-driven workflow?

- **Airflow:** An open-source **workflow orchestrator** fused for building, scheduling, and monitoring data pipelines
- **Core components:**
  - **Web UI**
    - logs, lineage, task states, Gantt views
  - **Scheduler**
    - decides **when** each task should run and **moves tasks through states**
  - **Executor**
    - is the component that actually **runs your task instances**
  - **Metadata DB**
    - the **system of record** for your Airflow deployment

# How can we define Airflow workflows as code?

- A DAG is a **Directed Acyclic Graph**: a set of **nodes** connected by **directed edges**.
  - **Directed**: Each edge has a direction ( $A \rightarrow B$  means “A must happen before B”).
  - **Acyclic**: You can't loop back to where you started. **Topological order**.
  - **Graph**: A structure for expressing dependencies and concurrency.
  - **Written in Python**
- **Providers** optional, installable packages that **extend Airflow**
  - **Operators**
  - **Hooks**
  - **Sensors**
  - **Extras**
- **Apply AQIS with Airflow**

# DEMO 1: Airflow

- What will be covered:
  - Installing Airflow
  - Navigating the UI
    - DAGs view (homepage)
    - Inside a DAG (tabs you'll use most)
    - Browse menu (system-wide tables)
    - Admin menu (platform settings)
  - DAG example code


# How can we introduce HPC into the workflow pipeline?

- Dask is a Python-native parallel computing library
- What Dask gives us:
  - Auto parallelization of codes
  - Two APIs (high level/low level)
  - Easy Deployment from a single laptop to a multi-node cluster
  - **HPC deployment patterns**
    - SLURM
    - Adaptive scaling
    - Containers
    - Diagnostics

# When is Dask not appropriate?

- When **not to use** Dask
  - The application runs on a single-node or fits in memory
    - Consider using an alternative Python library such as Pandas
  - Runtime per task < 100–200 ms (micro-tasks)
    - Dask has per-task overhead (ms–10s ms).
  - Heavy communication between Dask workers
    - Heavy communication (e.g., frequent syncing) scales better with MPI.

# When is Dask appropriate?

- Your data **doesn't fit into memory** on a single machine
  - You want to **scale existing NumPy / Pandas / scikit-learn** workflows
  - You need to **parallelize computations** across multiple cores or nodes
  - Your workflow involves **large datasets or many files** (ETL, ML pipelines)
  - You need **lazy evaluation**, **fault tolerance**, or **cluster scheduling** with MPI.
- 

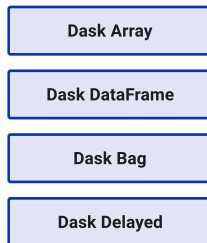


# What are the core components of Dask?

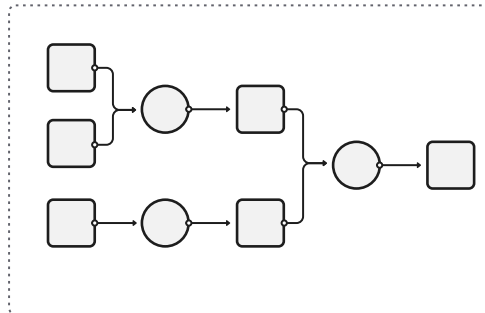
- Core components:
  - Collections
  - Lazy task graphs
  - Schedulers

## Collections

(create task graphs)

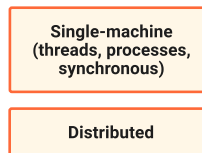


## Task Graph



## Schedulers

(execute task graphs)



# How does Dask integrate with the AQIS Engine?

- The AQIS engine serves as a framework for parallel task deployment
- The engine leverages Dask capabilities for parallel computing in data processing or machine learning.
  - It facilitates **flexible management of distributed computing tasks.**
  - Functions can either be **executed locally or remotely via a Dask cluster**

# AQIS Engine Component Overview

- **AQIS Engine Interface**
  - Facilitates Dask cluster connections
  - Execution mode management for either local or remote Dask cluster orchestration
  - Provides utility methods to submit distributed tasks and expose REST APIs
- **AQIS Dataframe**
  - For data handling. Abstracts input data via a unified interface
- **Data System Adaptors / operation packs**
  - These encapsulate basic operations for vector DBs, relation DBs, and object stores
  - **AQIS Airflow providers:**
    - iRODS
    - Milvus

## DEMO 2 Airflow AQIS/Dask

- What will be covered:
  - Dask diagnostics
  - Airflow example DAG: Extract, Transform, and Load (ETL) process
    - Extract the data from s3 storage
    - Deploy a distributed transformation (Map/Reduce) computation (HPC Slurm)
      - Using a Dask scheduler and workers we clean the data
    - Load transformed data



# About this Two-Part Webinar / Further Resources

# Aim of the two-part webinar

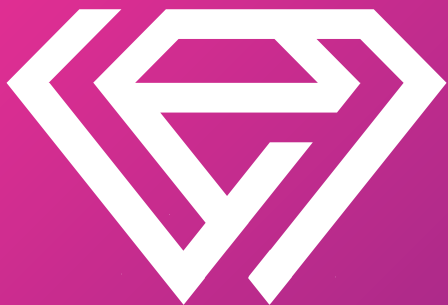
## ... of which you just saw the second part

With our webinars “Supercomputing with optimum data backends (1/2)” in the EXAKI training series, we presented you:

- modern object-store and database backends for data handling,
- how EXA4MIND helps you install these systems, and
- how EXA4MIND and its AQIS help you set up data analytics on top of this
  - using the AQIS-engine-Airflow option
  - using the AQIS-engine-dask option / a combination of dask and Airflow

These techniques can be used on systems from laptop to supercomputing scale.

To understand AQIS setup, backend setup and the combined utilisation for Extreme Data workflows in more detail, visit our [documentation and repositories](#)!



Visit us at [exa4mind.eu](https://exa4mind.eu)

# Thank You

Documentation  
[docs.exa4mind.eu](https://docs.exa4mind.eu)



Open Source  
Modules Repository



Project Site



E4M Kickstarter  
Initiative EXAKI



Funded by  
the European Union

Keep in touch!



@exa4mind



/exa4mind