Empowering weather & climate forecast:

ML Apps & Datasets
ML Workflow Tools
Hardware Systems

Fabian Emmerich
4Cast

“The MAELSTROM project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955513. The JU receives support from the European Union’s Horizon 2020 research and innovation programme and United Kingdom, Germany, Italy, Luxembourg, Switzerland, Norway.”
Motivation

Different cultures, approaches & solutions. Unification and extension are needed!

Problem

- Complicated workflow
- Enormous data sets
- Lack of unified infrastructure
- Lack of unified software

Solution

- Ease the workflow
- Pave the way towards exascale data
- Abstract away infrastructure
- Build software with unified interface
Unified infrastructure missing today:

- Benchmarking
- Collaboration
- Monitoring
- Framework integration
- Reproducibility
- User interface
Our vision for the workflow

Provenance & Reproducibility – Strictly Open Source

Collaborative Data and ML Model Repository

Data Hub - ETL
- Import Data
- Tidy up Data
  - Secure import
  - Automated cleaning

ML Hub
- AI Model Cycle
- Transform
- Visualize
  - ML-Powered Benchmarking
  - W&C ML Community
  - Library Integration

Model

Deploy & Monitor
- Deployment Hub
- Business Intelligence
- HPC environments
- Automatic App Extraction

Deploy & Monitor
- Interpret & Infer

Our vision for the workflow
Achievements to date

Survey of typical ML workflow in W&C

Requirements for workflow tools

MAELSTROM workflow tools v1.0
- Based on the Mantik ML tool\(^1\): open-source software to execute ML workflows on any hardware infrastructure (local machine, cloud computers or HPC architectures)
- Will be used to benchmark different software solutions for MAELSTROM apps
- Store the configurations and results of experiments in the Model Database

\(^1\) www.mantik.ai
Mantik software architecture

Cloud Service

- Engine
  - Core Component
  - Interacts with users
  - Creates and stores DAGs
  - Triggers DAG execution
  - Collects results

Model Database

- Storage
  - Workflows
  - Experiments
  - ML models

Local/Cloud/HPC

Coordinator

- Pipeline Execution
  - Starts Bridges
  - Triggers pipelines
  - Monitors Bridges
  - Fetches results

Bridges

- Pipeline Steps
  - Run as microservices
  - Wrap frameworks
  - Implement algorithms or provide data
  - Apply algorithms on incoming data
  - Train and evaluate ML models
Workflow tools key features

- Reproducible ML solutions
- Share ML solutions across user base
- Recommendation of ML solutions to users with specific problems
- Interface to cloud computing and HPC
- Manipulation of execution graphs leading to optimal execution of W&C workflows
Current interfaces

MAELSTROM platform as a ML workflow platform

- Command Line Interface
- Model Database
- HPC interface
- GUI for execution and job status
Code examples

# main.py
import mantik

# Connect to the Mantik Engine.
with mantik.Client("mantik.ai") as client:
    # Add pipeline steps to the Engine's registry.
    dataset = client.add_artifact("dataset/")
    transform = client.add_artifact("transform/")
    model = client.add_artifact("ml-model/")

    with client.enter_session():
        # Train the model.
        pipeline = [transform, model]
        model, stats = client.train(pipeline, data=dataset)

        # Use the model for inference.
        inference = client.apply(model, data=dataset)

# transform/algorithm.py

# Access input parameter or data.
variable = meta.get("variable")
data = bundle.value

# Transform the data.
transformed = ...

# Pass data to the next pipeline step.
return mantik.Bundle(transformed)

# ml-model/train.py

# Train the model on the data (`bundle.value`).
model = ...

# Get statistics about the model.
metrics = ...

# Return the trained model and any other information.
return model, mantik.types.Bundle(metrics)

# Apply the model on the input data.
result = model.predict(bundle.value)

# Return the result of the inference.
return mantik.Bundle(result)

# dataset/dataset.py

def get(meta: mantik.MetaVariables):
    # Access input parameters.
    variable = meta.get("variable")

    # Create the dataset.
    data = ...

    # Pass the dataset to the next step.
    return mantik.Bundle(data)
Vision for the next two years

- **6 months**: MAELSTROM protocol and ML requirements
- **12 months**: Workflow tools v1.0
- **30 months**: Improved data processing tools and weather data loading pipeline
- **36 months**: Final version of workflow tools

Software performance benchmarking
The MAELSTROM project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955513. The JU receives support from the European Union’s Horizon 2020 research and innovation programme and United Kingdom, Germany, Italy, Luxembourg, Switzerland, Norway.

Questions?