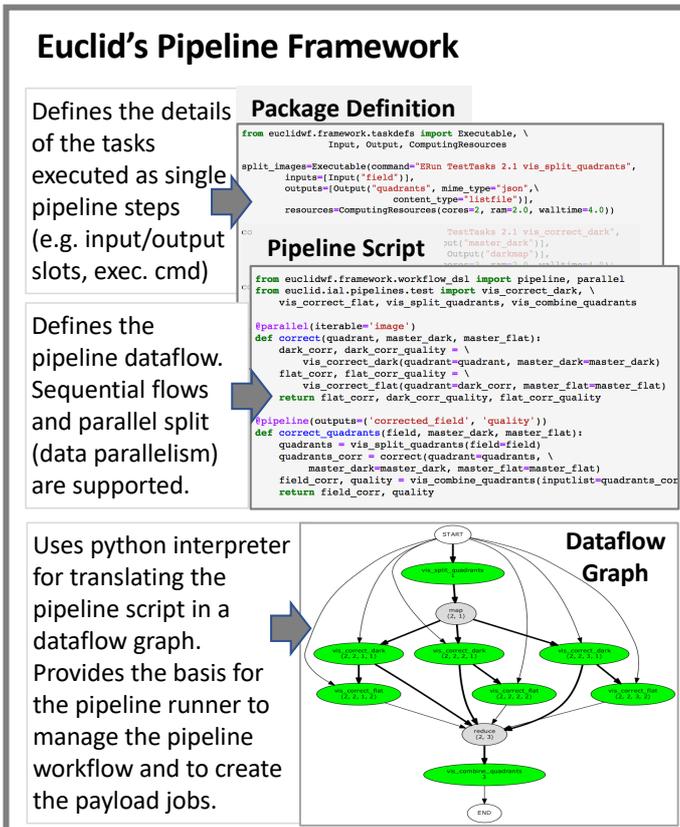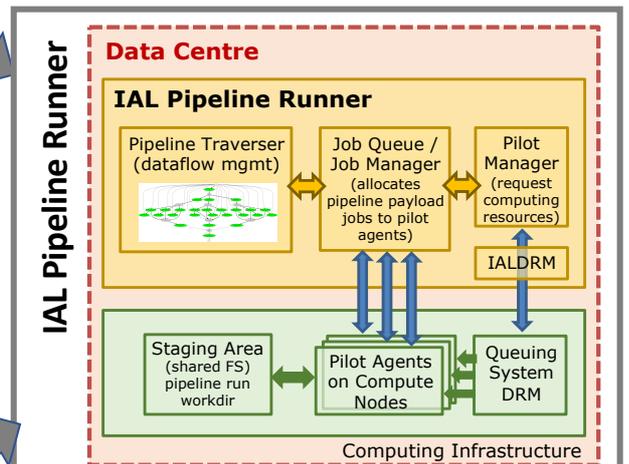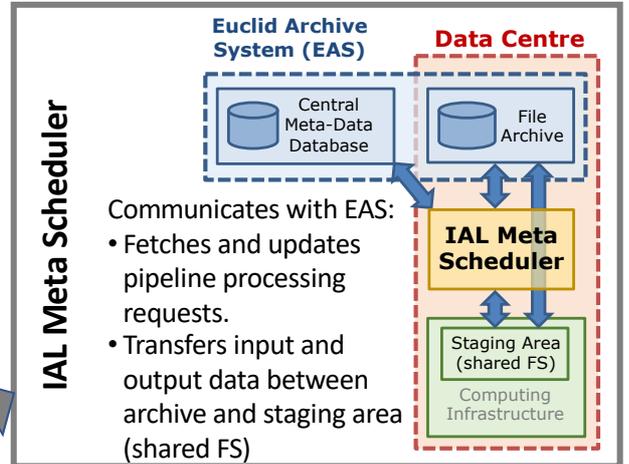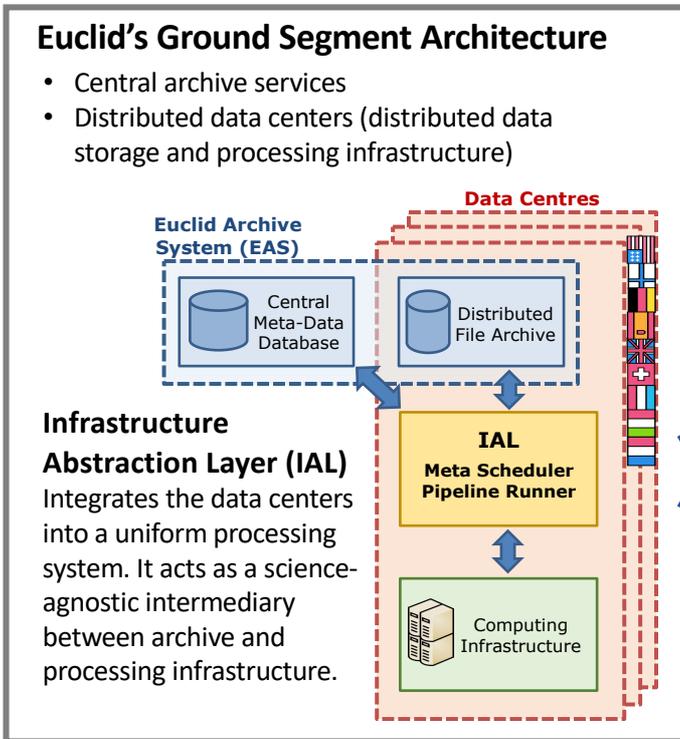# Scheduling the Euclid Pipeline in the Ground Segment Processing Infrastructure

n|w

Martin Melchior[1], Simon Marcin[1], Christophe Dabin[2], Marco Frailis[3], Mark Holliman[4], Andrey Belikov[5]

(1) University of Applied Science of Northwestern Switzerland, Switzerland, (2) CNES Toulouse Centre National d'Etudes Spatiales, France, (3) INAF - Osservatorio Astronomico di Trieste, Italy, (4) University of Edinburgh, United Kingdom, (5) University of Groningen, Netherlands

Euclid's Ground Segment Architecture

- Central archive services
- Distributed data centers (distributed data storage and processing infrastructure)

Euclid Archive System (EAS)

Central Meta-Data Database

Distributed File Archive

Data Centres

IAL Meta Scheduler Pipeline Runner

Computing Infrastructure

Infrastructure Abstraction Layer (IAL)

Integrates the data centers into a uniform processing system. It acts as a science-agnostic intermediary between archive and processing infrastructure.

IAL Meta Scheduler

Euclid Archive System (EAS)

Central Meta-Data Database

Data Centre

File Archive

IAL Meta Scheduler

Staging Area (shared FS)

Computing Infrastructure

Communicates with EAS:
- Fetches and updates pipeline processing requests.
- Transfers input and output data between archive and staging area (shared FS)

IAL Pipeline Runner

Data Centre

IAL Pipeline Runner

Pipeline Traverser (dataflow mgmt)

Job Queue / Job Manager (allocates pipeline payload jobs to pilot agents)

Pilot Manager (request computing resources)

IALDRM

Staging Area (shared FS) pipeline run workdir

Pilot Agents on Compute Nodes

Queuing System DRM

Computing Infrastructure

- Adopts dataflow paradigm to manage pipeline workflow (at data centre level).
- Abstracts the systems setup at the different data centers: DRM (IALDRM), FS, VM
- Adopts pilot job concept for more flexible and efficient job submission: Submits pilot agents to DRM, once started on compute nodes asks job manager for payload jobs and executes them.

Euclid's Pipeline Framework

Defines the details of the tasks executed as single pipeline steps (e.g. input/output slots, exec. cmd)

Package Definition

```
from euclidwf.framework.taskdefs import Executable, \
        Input, Output, ComputingResources

split_images=Executable(command="ERun TestTasks 2.1 vis_split_quadrants",
        inputs=[Input("field")],
        outputs=[Output("quadrants", mime_type="json",\
                content_type="listfile")],
        resources=ComputingResources(cores=2, ram=2.0, walltime=4.0))

        TestTasks 2.1 vis_correct_dark",
        put("master_dark")],
        Output("darkmap")],
```

Pipeline Script

```
from euclidwf.framework.workflow_dsl import pipeline, parallel
from euclid.ial.pipelines.test import vis_correct_dark, \
    vis_correct_flat, vis_split_quadrants, vis_combine_quadrants

@parallel(iterable='image')
def correct(quadrant, master_dark, master_flat):
    dark_corr, dark_corr_quality = \
        vis_correct_dark(quadrant=quadrant, master_dark=master_dark)
    flat_corr, flat_corr_quality = \
        vis_correct_flat(quadrant=dark_corr, master_flat=master_flat)
    return flat_corr, dark_corr_quality, flat_corr_quality

@pipeline(outputs=('corrected_field', 'quality'))
def correct_quadrants(field, master_dark, master_flat):
    quadrants = vis_split_quadrants(field=field)
    quadrants_corr = correct(quadrant=quadrants, \
        master_dark=master_dark, master_flat=master_flat)
    field_corr, quality = vis_combine_quadrants(inputlist=quadrants_corr)
    return field_corr, quality
```

Defines the pipeline dataflow. Sequential flows and parallel split (data parallelism) are supported.

Uses python interpreter for translating the pipeline script in a dataflow graph. Provides the basis for the pipeline runner to manage the pipeline workflow and to create the payload jobs.

Dataflow Graph

Technologies

- Pipeline Runner: microservices architecture; python 3, twisted, sqlalchemy, flask
- Meta-Scheduler: finite state machine, multi-threaded; java 8, spring boot, hibernate, jaxb
- IALDRM: wrapper scripts (python), supports SLURM, PBS, SGE, local, (…)

Both components passed several software development cycles and have been successfully tested at scale in a series of Euclid ground segment challenges.