Scaling AI Science with ACCESS Pegasus

From Notebooks to National Cyberinfrastructure: A Guide for Novice Users

Submit Locally, Run Globally.
University of Southern California, Information Sciences Institute

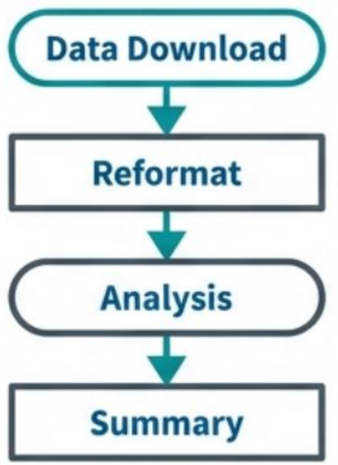
This work was supported by NSF grants #2138286 and #2513101

The Challenge: Moving Beyond the Laptop

The Scripting Trap

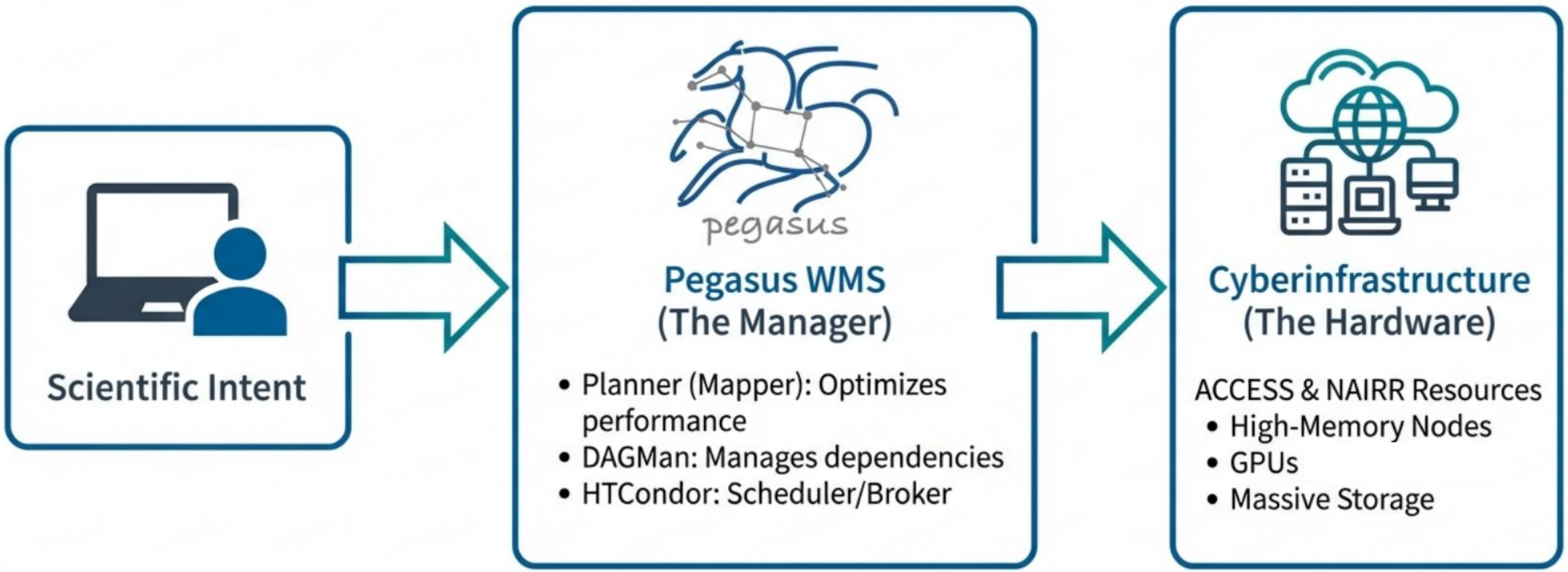
- Quick to code, but hard to scale
- Difficult to maintain and error-prone
- Hard to move between resources (Laptop vs. Cluster)
- “Works on my machine” syndrome

The Workflow Advantage



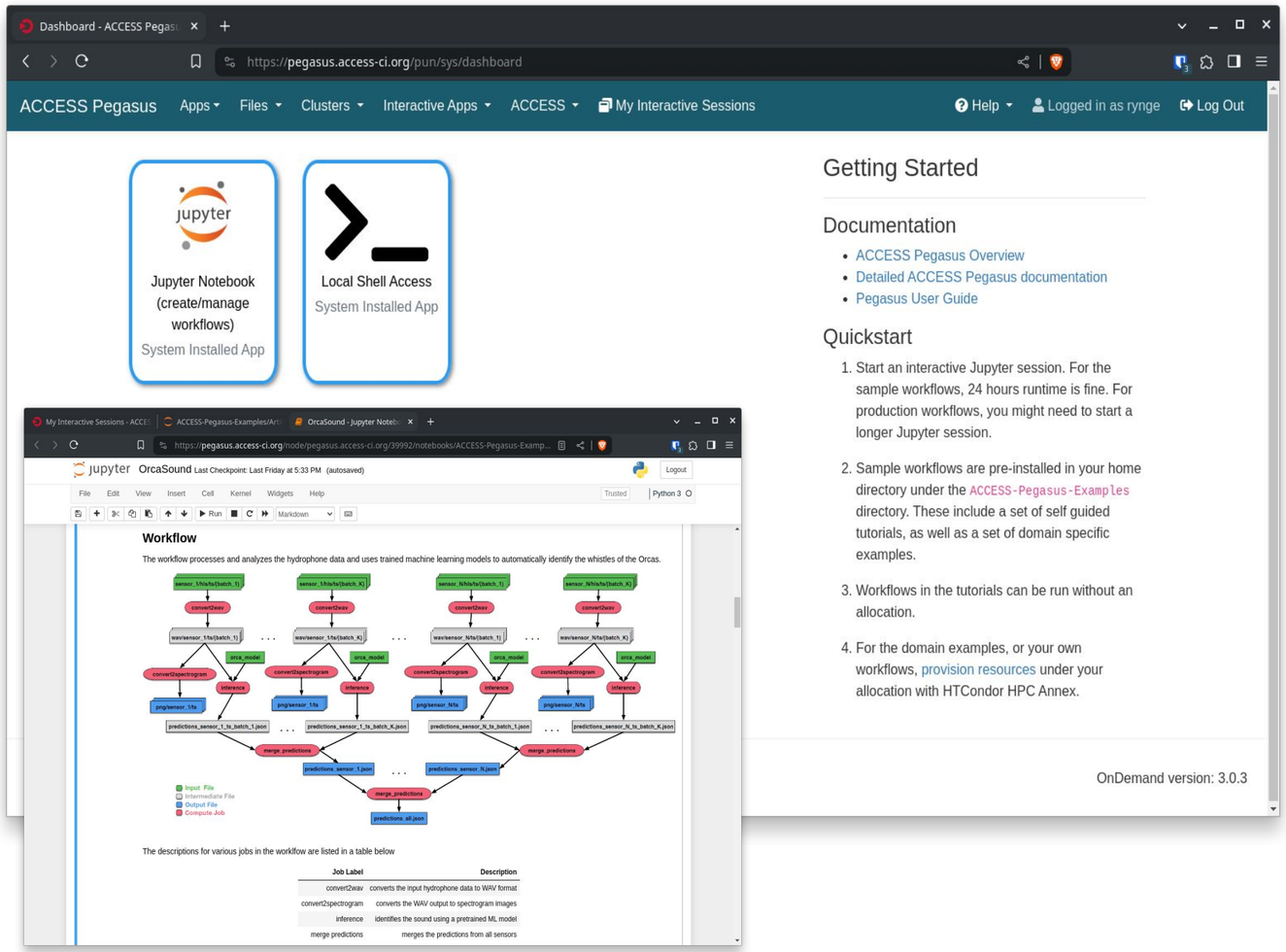
- Declarative: Describe *what* to do, not *how*
- Automated: Minimizes human involvement
- Provenance: Tracks history for reproducibility
- Portable: Runs on ACCESS, NAIRR, cloud, …

Your Toolkit for Scale



Pegasus bridges the gap between your code and national-scale hardware.

The Interface: Tools You Already Know

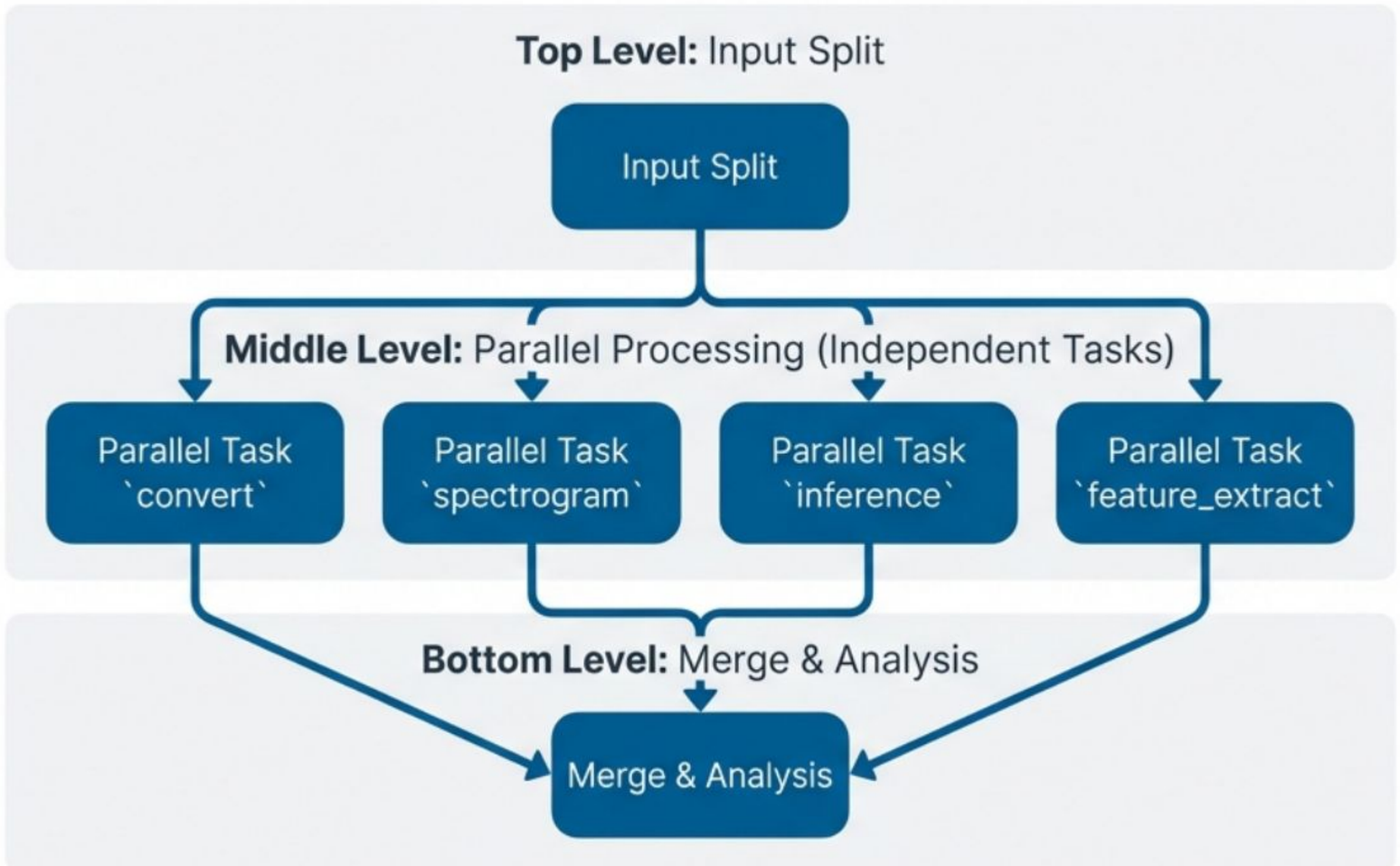


Open OnDemand: Web-based access, no complex command line setup.

Interactive Development

- Develop and monitor workflows directly in Jupyter.
- Pre-configured environment: Log in, Start Jupyter, Clone Examples.
- Zero installation required on your local machine.

Visualizing the Workflow

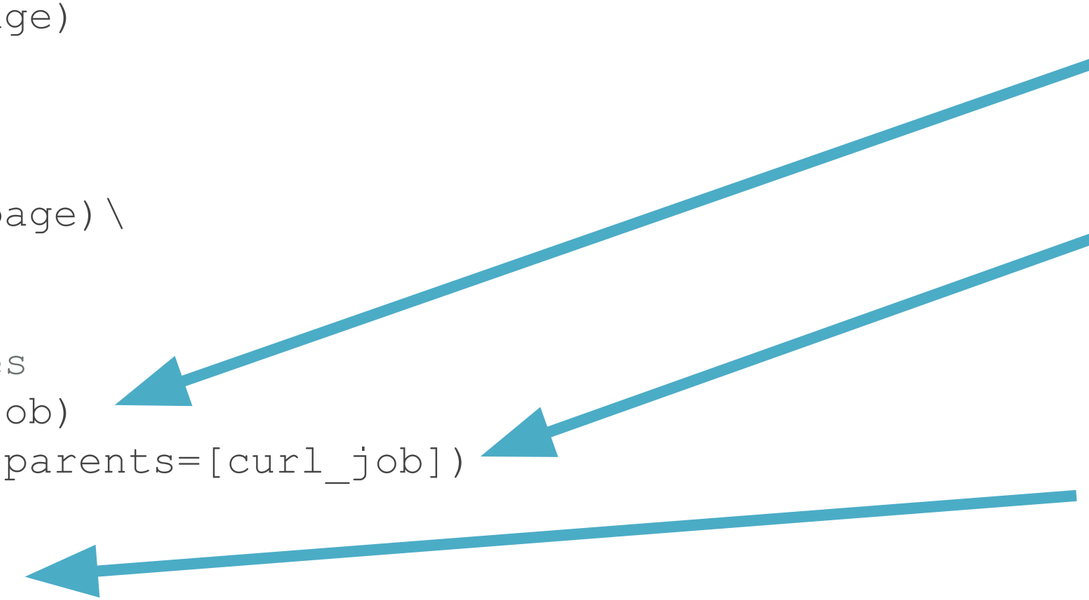


The DAG (Directed Acyclic Graph)

- **Nodes:** The tasks (e.g., 'inference', 'convert').
- **Edges:** The dependencies. Task B waits for Task A.
- **Benefit:** Pegasus detects independent branches and runs them in parallel automatically.

Defining Workflows in Python

```
from Pegasus.api import *

# Create abstract workflow
wf = Workflow("pipeline")

# Track the data
page = File("page.html")

# Create a Job
curl_job = Job("curl")\
           .add_args("-o", page, "https://pegasus.isi.edu")\
           .add_outputs(page)

# Create a Dependent Job
wc_job = Job("wc")\
         .add_args("-l", page)\
         .add_inputs(page)

# Add jobs and dependencies
wf.add_jobs(curl_job, wc_job)
wf.add_dependency(wc_job, parents=[curl_job])

wf.write("pipeline.yml")
```



Declarative job addition

Simple dependency definition

Generates the abstract workflow (DAG)

The AI-Assisted Development Pattern

Transforming scientific ideas into production workflows in minutes, not days.

01. Human Input

Describe the "What", not the "How". Define data inputs, analysis goals, and expected outputs in plain English. Ignore infrastructure.

02. Context Injection

Feed Claude a "Gold Standard" Pegasus repo (e.g., Earthquake or Air Quality). This forces valid valid syntax and prevents hallucinations.

03. AI Processing

Claude maps science to task definitions. Output is a working `'workflow_generator.py'` script—logical structure, not pseudocode.

04. Human-in-the-Loop

Validate on local/small data. Ask targeted fixes: "How do I parallelize this?" Safe iteration separated from execution.

05. Pegasus Execution

Deploy at scale. The validated workflow automatically handles retries, provenance, and global seismic/sensor datasets.

Core Philosophy: The user focuses on the **Science** (parameters and logic), while the AI handles the **Syntax** (structure and dependencies).

The ‘Grounding Technique: Your Recipe for Success

Claude becomes an expert developer only when guided by the right examples.

The Ingredients (Gold Standard Templates)

Earthquake Workflow
`pegasus-isi/earthquake-workflow`
- Complex branching pipelines
- USGS API data fetching
- ML-based aftershock prediction

Air Quality Workflow
`pegasus-isi/airquality-workflow`
- Sensor data & forecasting
- OpenAQ ingestion
- LSTM modeling & containerized viz

Orcasound Workflow
`pegasus-isi/orcasound-workflow`
- Media processing
- S3 bucket integration
- Hardware sensor inputs

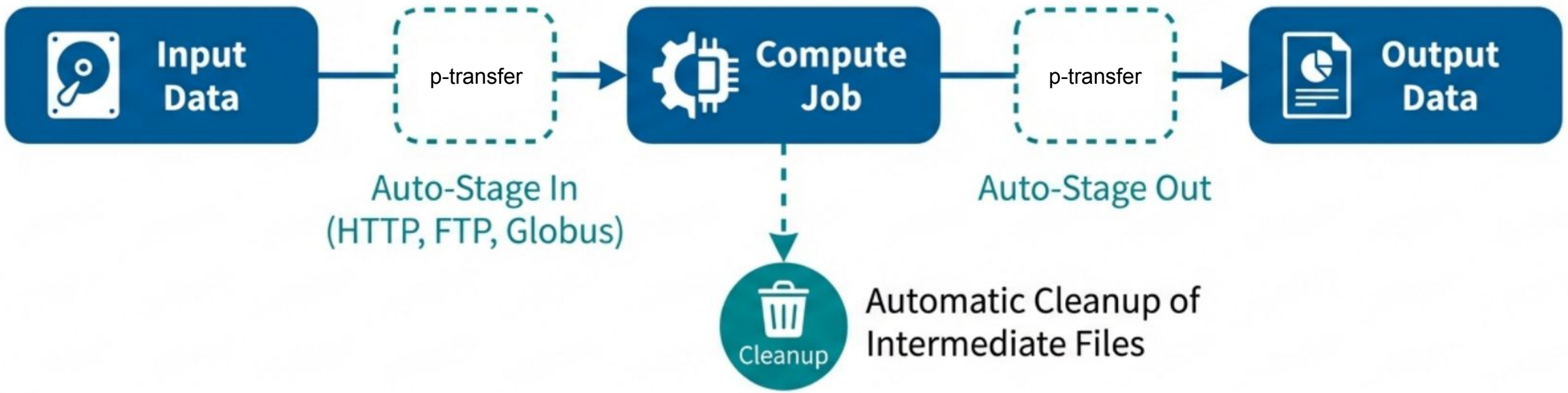
The Execution (The Golden Prompt)

```
Context: I am building a workflow to [Analyze X
Data] in order to [Produce Y Result].

Grounding: Here is an existing Pegasus workflow
[Insert Link to workflow_generator.py from
Earthquake repo]. Please adapt this structural
pattern to my new use case.

Request: Generate the 'workflow_generator.py'
script that defines the tasks, input files, and
dependencies.
```

Understood. Based on the structural pattern in the Earthquake repository, here is the adapted Python script for your [X Data] analysis...

The Mechanism: Claude reuses the **Structure** (Logic, Dependencies, Best Practices) from the repo, but swaps the **Content** (the specific scientific domain) from your prompt. This yields Pegasus-native code immediately.

Automated Data Management



Pegasus handles the file transfers so you don't have to write scp, Globus, s3cmd, … scripts manually.

Resilience: When Things Go Wrong

Automatic Retries

Jobs are automatically retried upon failure.

Checkpointing

Workflow-level checkpointing saves progress. Never restart from scratch.

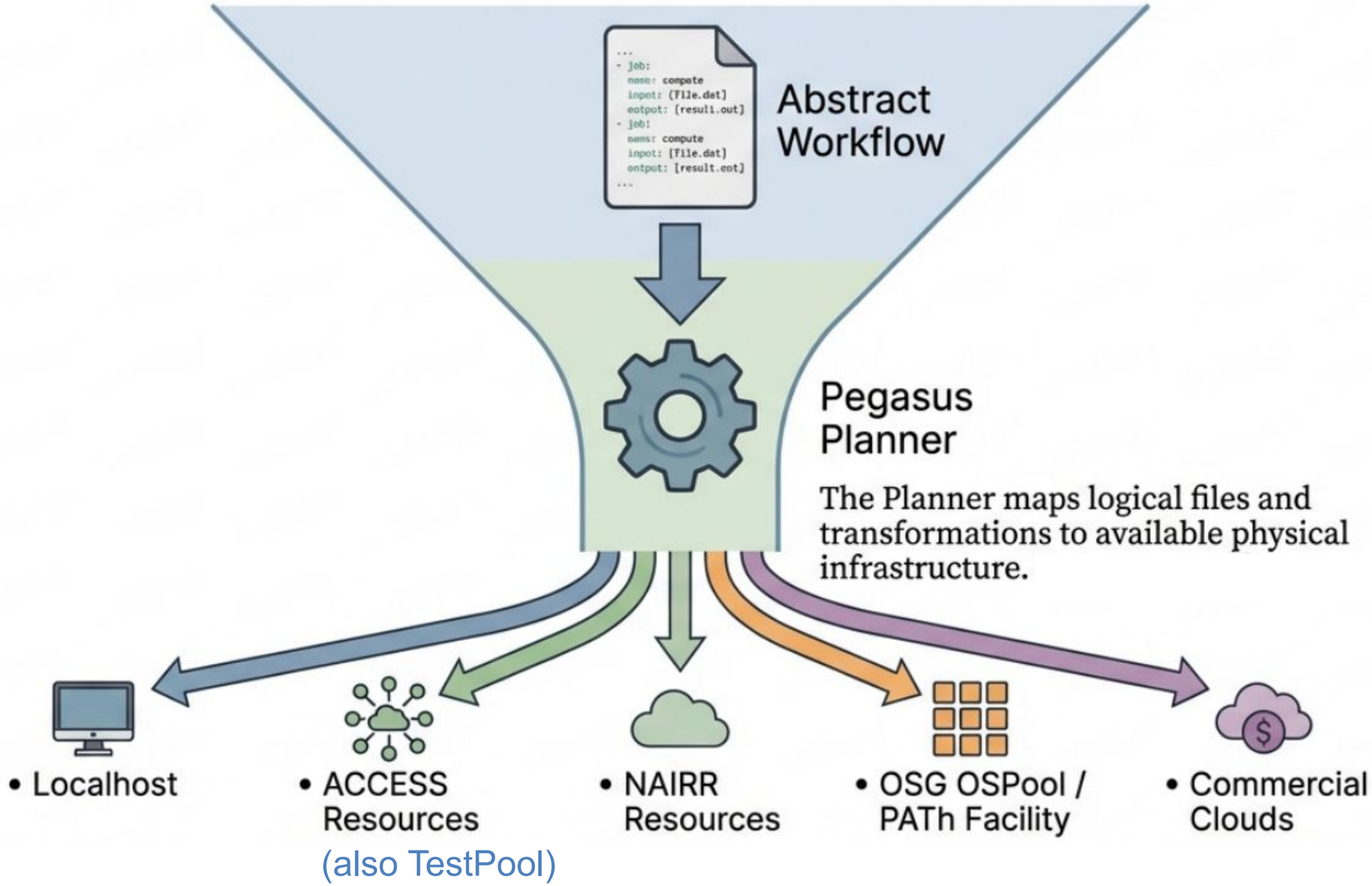
Rescue DAGs

Generates a plan containing only the remaining work after a crash.

Debug Tools

`pegasus-analyzer` pinpoints specific errors (data vs. code).

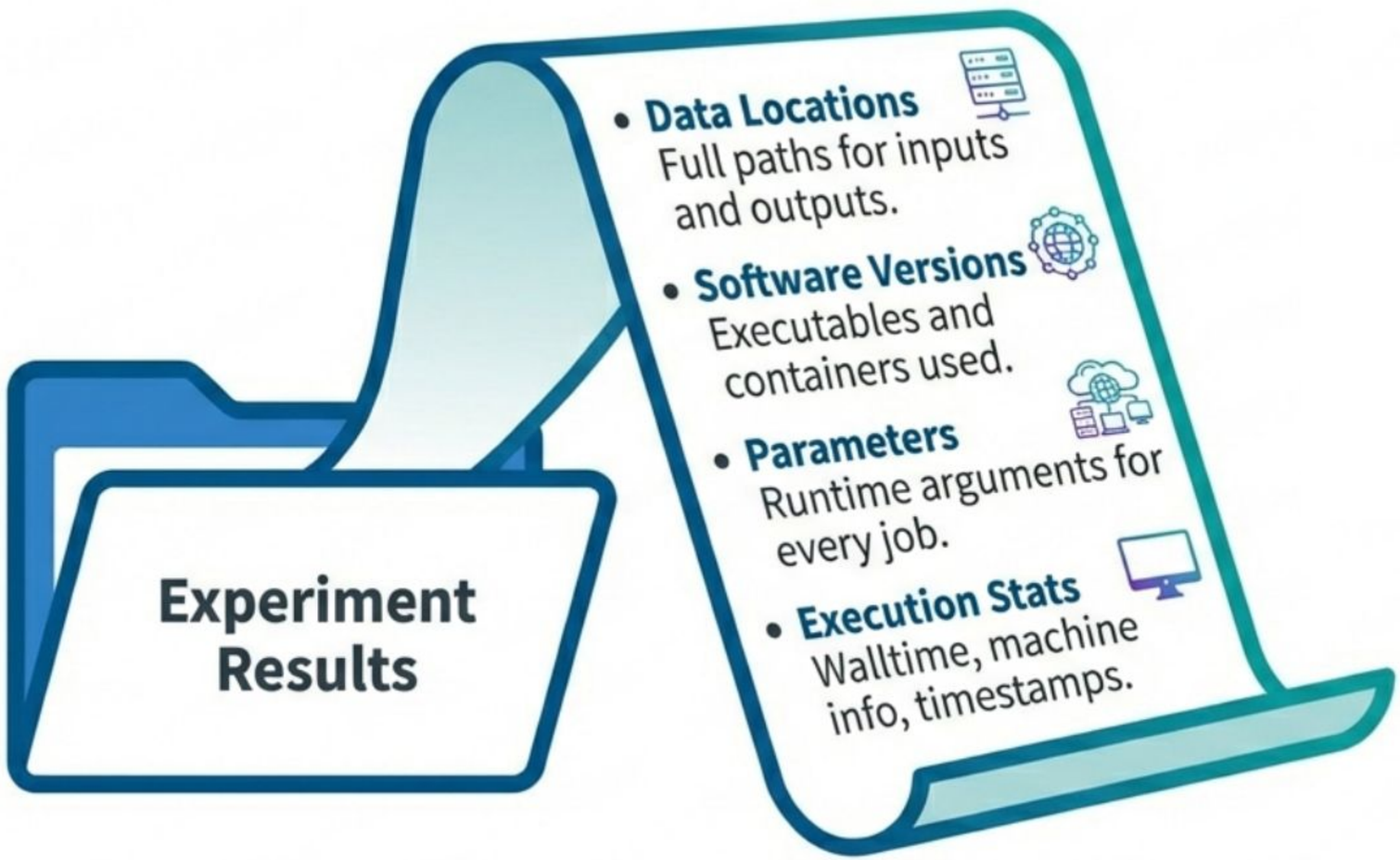
Write Once, Run Anywhere



Requirement: Any HPC/cloud/… system with outbound network connectivity.
Includes your own campus cluster or lab machines!

Provenance & Reproducibility

Good science requires proof.

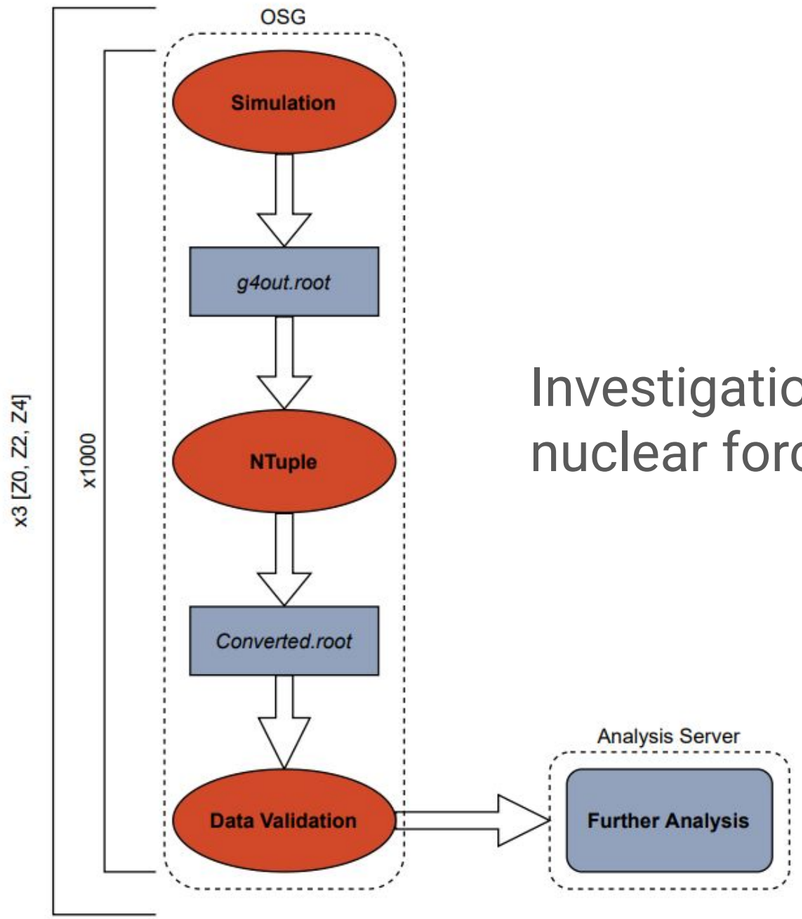


Pegasus tracks every bit of **data and every compute** step, ensuring your results are **verifiable** and **reproducible.**

Proven at Scale

From a proof of concept to winning the 2022 David Swanson Memorial Award

Connor Natzke,
Physics Student
Colorado School of Mines

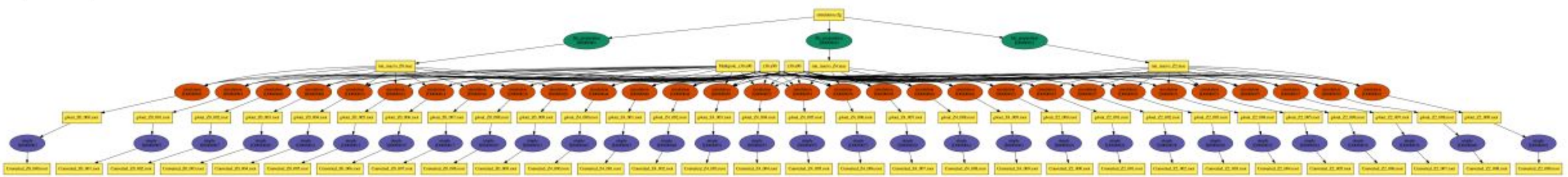


Investigation of the strong nuclear force

Pegasus-based Monte-Carlo simulation, 590,000 jobs, 15 years total wall time, 4 hours wall time on OSG

Pegasus provided:

1) Automation and scaling
2) Automatic job retries
3) Automation of file transfers
4) Managed disk space at execution sites

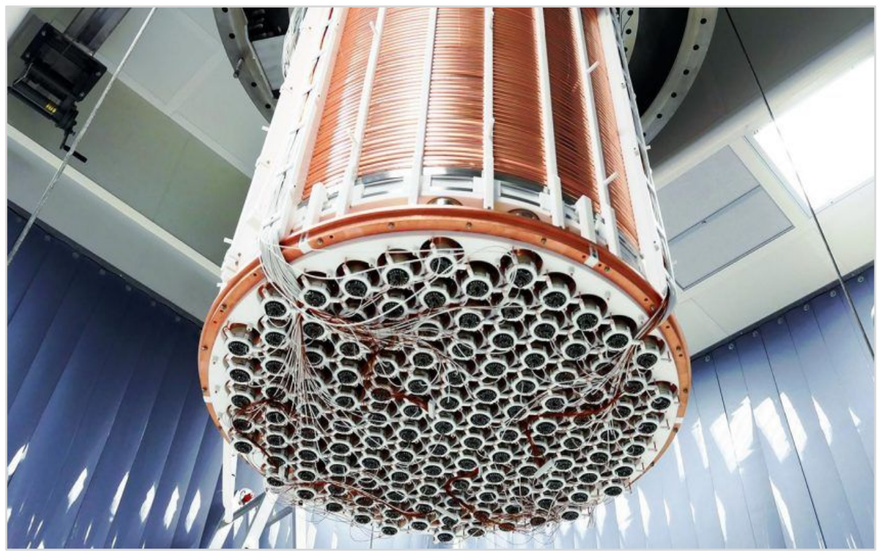


XENONnT - Dark Matter Search

Two Workflows

Monte Carlo simulations and the main processing pipeline.

- Workflows execute across Open Science Grid (OSG) & **European Grid Infrastructure (EGI)**
- Rucio for data management
- MongoDB instance to track science runs and data products.



```
------------------------------------------------------------------------------
Type           Succeeded Failed  Incomplete  Total     Retries   Total+Retries
Tasks          4000      0       0           4000      267       4267
Jobs           4484      0       0           4484      267       4751
Sub-Workflows  0         0       0           0         0         0
------------------------------------------------------------------------------

Workflow wall time                                       : 5 hrs, 2 mins
Cumulative job wall time                                 : 136 days, 9 hrs
Cumulative job wall time as seen from submit side        : 141 days, 16 hrs
Cumulative job badput wall time                          : 1 day, 2 hrs
Cumulative job badput wall time as seen from submit side : 4 days, 20 hrs
```

Event Horizon Telescope

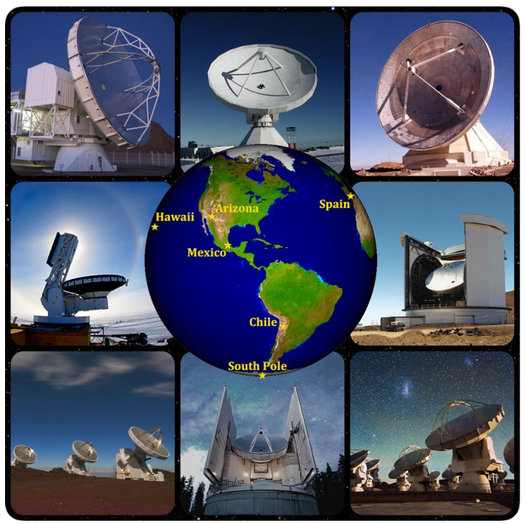
Bringing Black Holes into Focus

480,000 jobs - 2,600,000 core hours

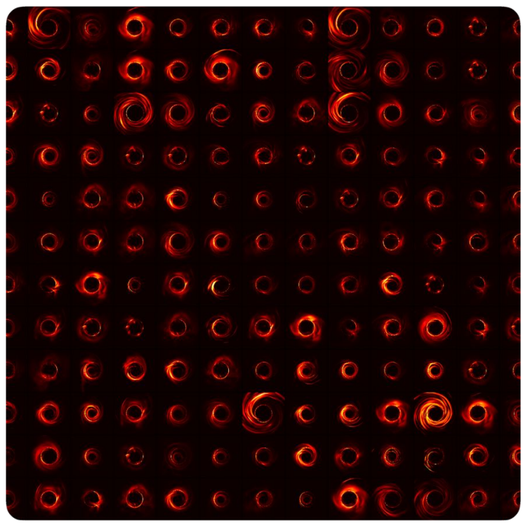
#15 in all OSG projects in last 6 months

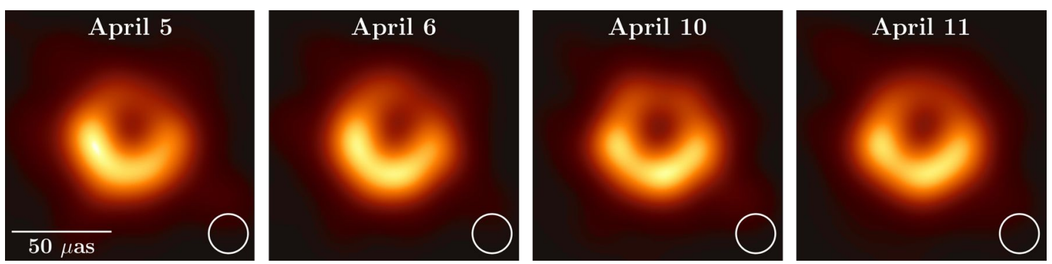
#2 in all OSG astronomy projects in the last 6 months

8 telescopes: 5 PB of data



60 simulations: 35 TB data



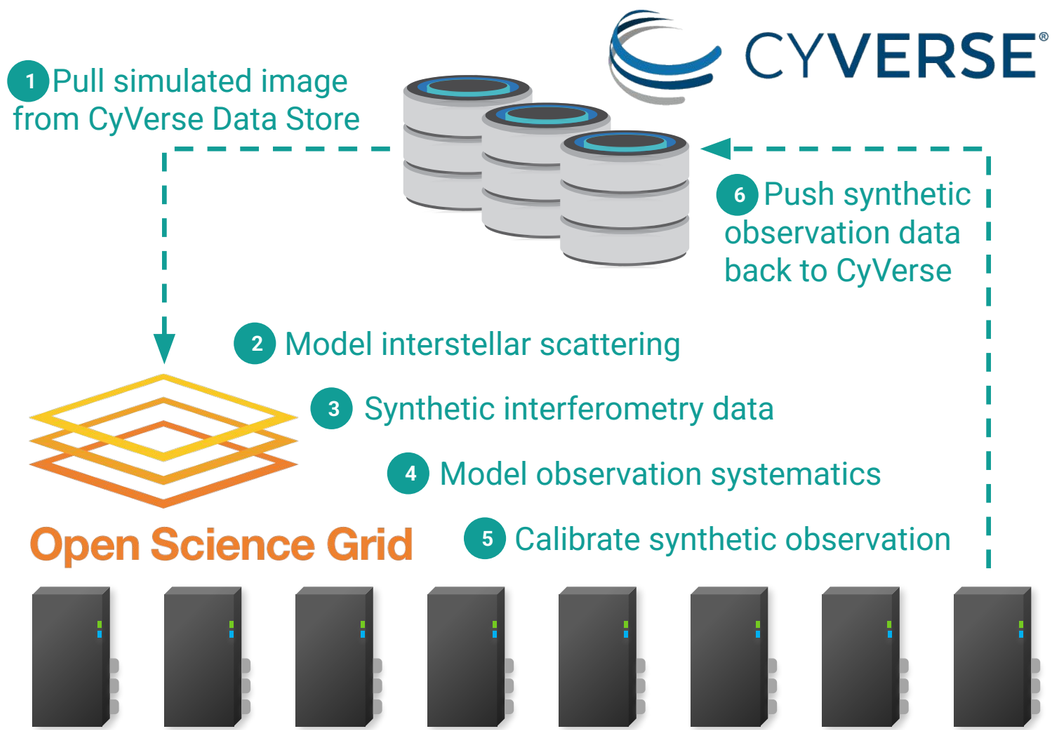


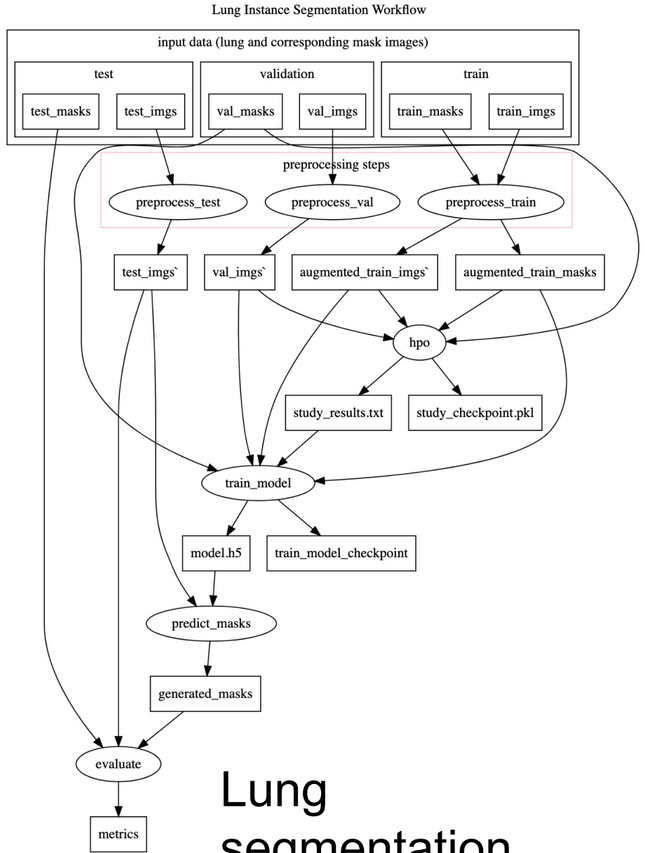
First images of black hole at the center of the M87 galaxy

Improve constraints on Einstein's theory of general relativity by 500x

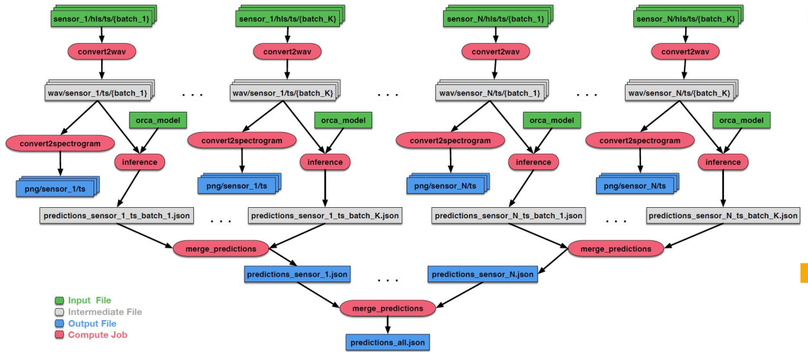
Pegasus-SYMBA Pipeline

Physically accurate synthetic observation data from simulations are keys to develop calibration and imaging algorithms, as well as comparing the observation with theory and interpreting the results.

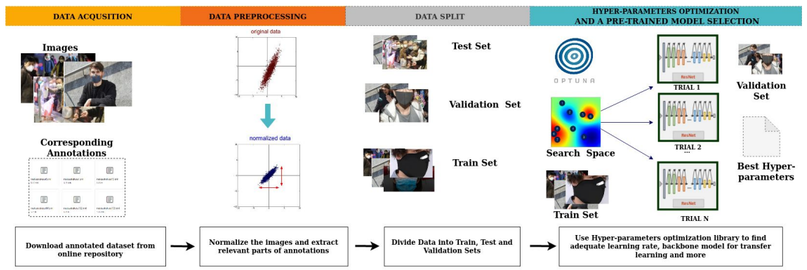


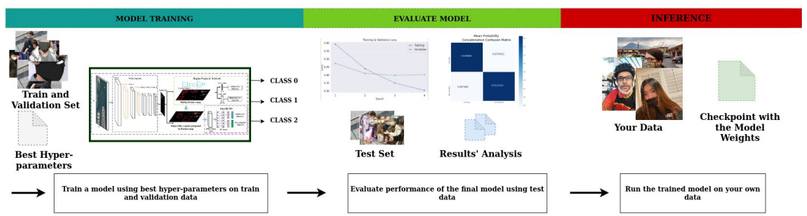


Lung segmentation

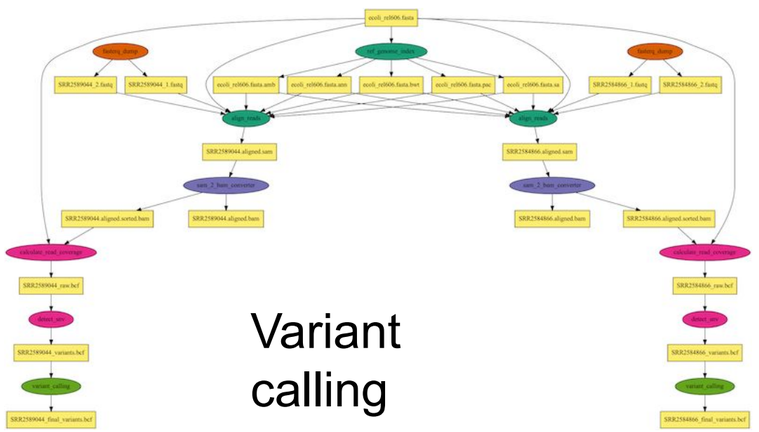


Orca Sound

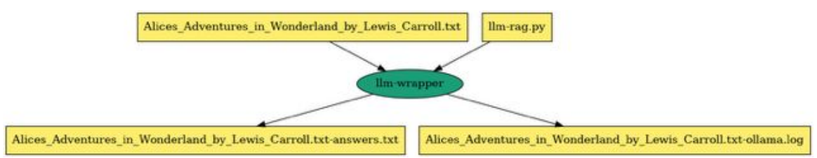




Mask detection

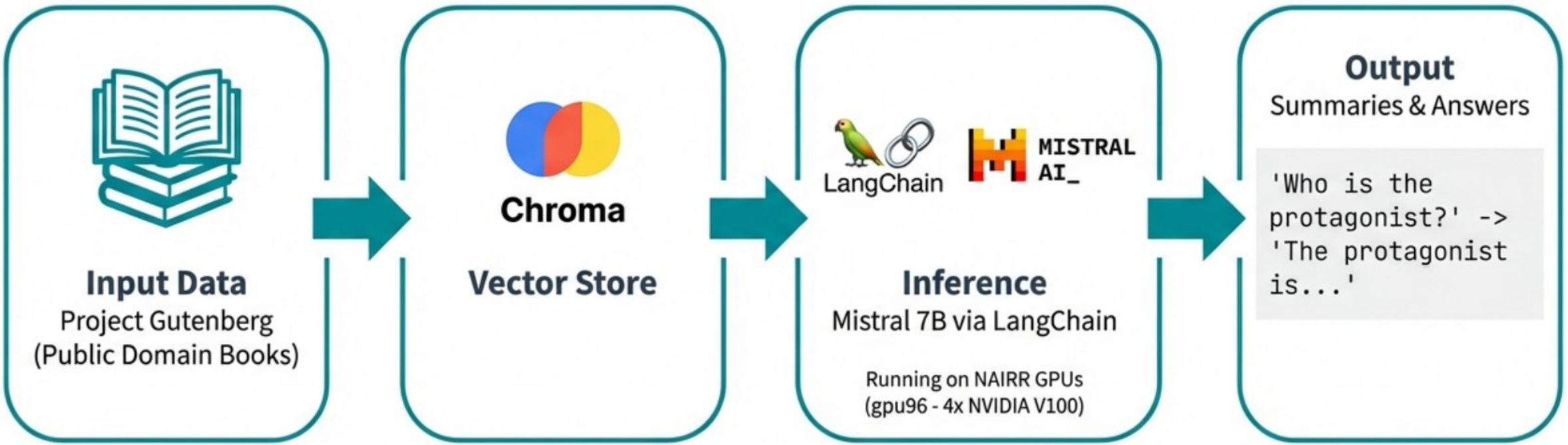


Variant calling



LLM + RAG

Real-World Example: LLM & RAG



Get Started Today

Login

https://pegasus.access-ci.org

Launch

Start a Jupyter session.

Run

Run the numbered tutorials

Documentation

- ACCESS Pegasus Overview
- Detailed ACCESS Pegasus documentation
- Pegasus User Guide

Getting Started

1. Start an interactive Jupyter session. For the small workflows, 24 hours runtime is fine. For production workflows, you might need to start a longer Jupyter session.
2. Sample workflows are pre-installed in your home directory under the `ACCESS-Pegasus-Examples` directory. These include a set of self guided tutorials, as well as a set of domain specific examples (described below).
3. Tutorials, example workflows, and your own simple workflows can be run without an allocation.
4. For larger workflows, provision resources under your allocation with HTCondor HPC Annex.

Demo

We are here to help!

Slack channel

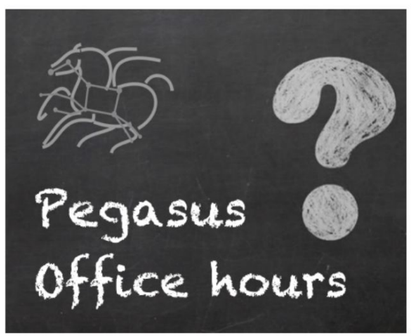
Email: pegasus-support@isi.edu

Office hours every Friday, 11am PT

Office Hours

Join the Pegasus team every Friday for virtual office hours at 11 AM Pacific / 2 PM Eastern.

Do you have questions about workflows or need guidance on organizing and implementing them? Join our weekly office hours – designed to support both new and experienced users in learning and engaging with Pegasus. Here's what to expect:



ACCESS

Tutorial walkthrough First Friday of the month

http://pegasus.isi.edu

Share Share Post Reddit Like 1

Slack

Add to Slack We encourage you to join the Slack Workspace as it is an on-going, open forum for all Pegasus users to share ideas, experiences, and talk out issues with the Pegasus Development team. Please click the button above or ask for an invite by trying to join *pegasus-users.slack.com* in the Slack app, or send an email to pegasus-support@isi.edu to request an invite.

Mailing Lists

`pegasus-users@isi.edu`

This list is the main support vehicle for Pegasus. Please note that the following subscription link requires a Google-linked email address. If you want to subscribe with non-linked email address, please contact us at pegasus-support@isi.edu and we will add you.

- Subscribe / unsubscribe / archive

`pegasus-announce@isi.edu`

Messages about new releases and updates. Low traffic. Please note that the following subscription link requires a Google-linked email address. If you want to subscribe with non-linked email address, please contact us at pegasus-support@isi.edu and we will add you.

- Subscribe / unsubscribe / archive

Automate, Recover, and Debug your Scientific Computations.

Extra

Southern California Earthquake Center's CyberShake

CPU jobs
(Mesh generation, seismogram synthesis)
1,094,000 node-hours

GPU jobs:
439,000 node-hours
AWP-ODC finite-difference code
5 billion points per volume, 23,000 timesteps
200 GPUs for 1 hour

Titan:
421,000 CPU node-hours, 110,000 GPU node-hours

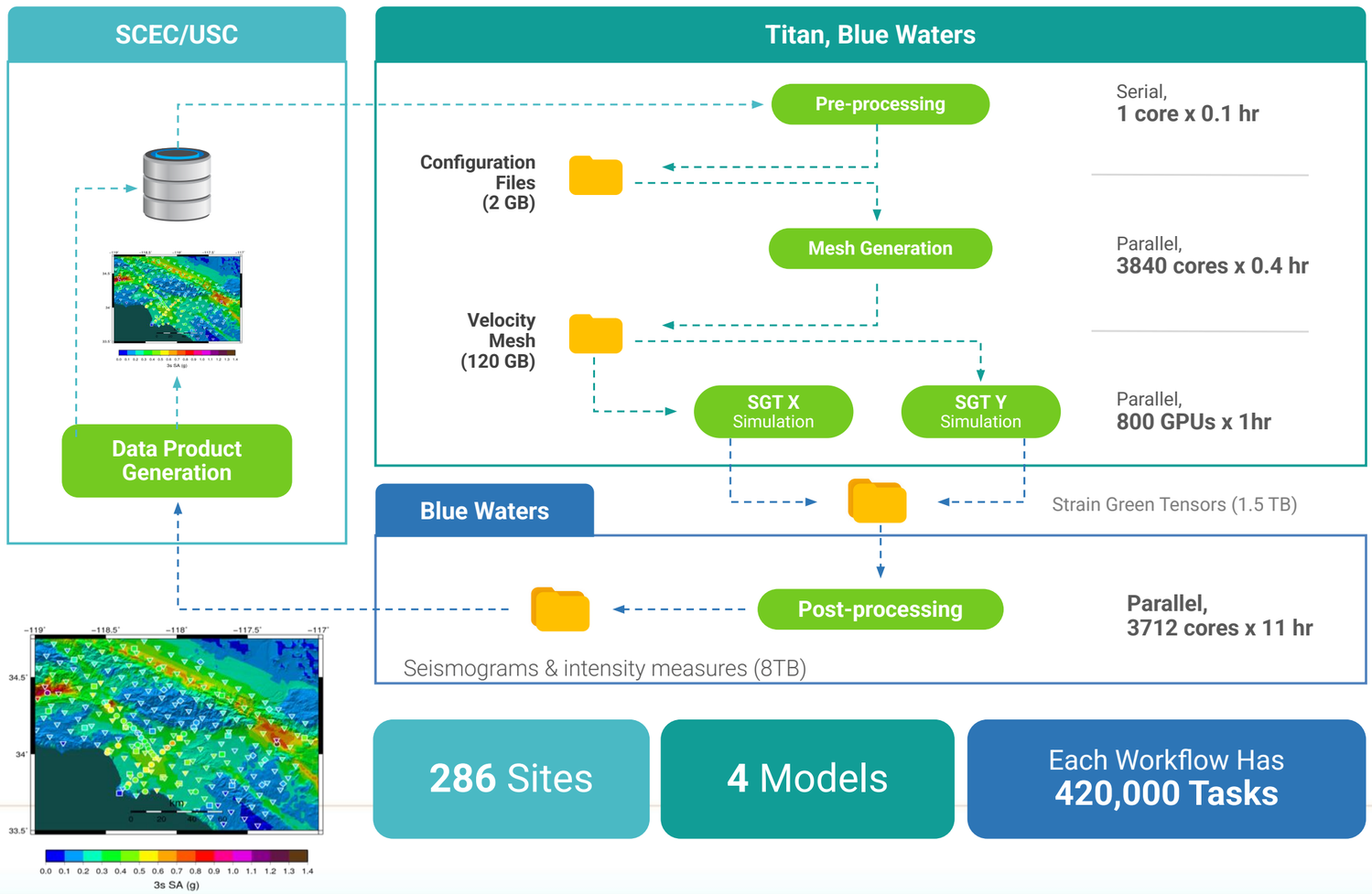
Blue Waters:
673,000 CPU node-hours, 329,000 GPU node-hours

Builders ask seismologists:

What will the peak ground motion be at my new building in the next 50 years?

Seismologists answer this question

using Probabilistic Seismic Hazard Analysis (PSHA)



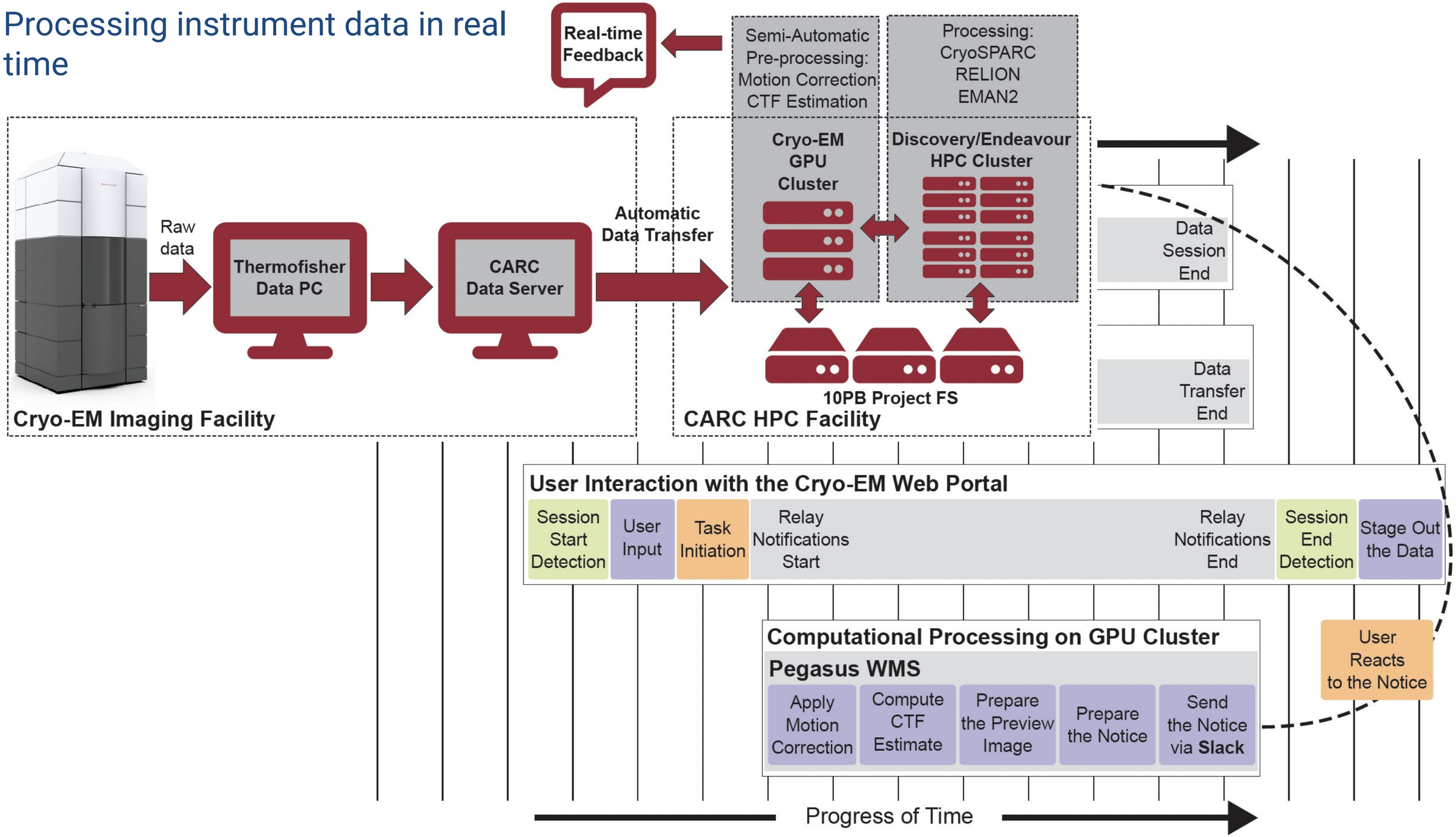
286 Sites

4 Models

Each Workflow Has **420,000 Tasks**

NSF | ACCESS Support

Processing instrument data in real time



Next major version of Pegasus will have AI augmentation

AI-augmented *pegasus-analyzer* automatically interprets workflow logs and job histories, providing users with intelligent, human-readable explanations of why a workflow failed and suggesting likely fixes or optimizations.

Next major version of Pegasus will have AI assistant

Pegasus Analyzer

Current version of analyzer pinpoints and highlight failures.

```
2025-09-25 19:10:16,985    INFO:  /bin/cp -f -R -L
'/home/rynge/ACCESS-Pegasus-Examples/04-Tutorial-Debugging-Statistics/bin/llm-rag.py'
'/home/rynge/ACCESS-Pegasus-Examples/04-Tutorial-Debugging-Statistics/scratch/rynge/pegasus/ll
m-rag-books/20250925T190853+0000/./llm-rag.py'
2025-09-25 19:10:16,991    ERROR:  Expected local file does not exist:
/home/rynge/ACCESS-Pegasus-Examples/04-Tutorial-Debugging-Statistics/inputs/Alices_Adventures
_in_Wonderland_by_Lewis_Carroll.txt
```

We are adding a Pegasus AI assistant to help interpret the errors.

Next major version of Pegasus will have AI assistant

========== Pegasus AI Analysis ==========

The workflow failed due to a missing input file. The job `stage_in_local_local_0_0` encountered an error:
"Expected local file does not exist: /path/to/Alices_Adventures.txt"

Root Cause:
- The required input file is missing from the specified path.
- This prevents the transfer process from completing, causing the workflow to fail.

Next Steps:
1. Verify the file exists at the specified path.
2. Ensure the file path in the workflow configuration matches the actual location.
3. Resubmit the workflow after resolving the file issue.

The remaining unsubmitted jobs (7 total) likely depend on this staged file, so fixing this error will enable further execution.

Advantages of Glidein (Pilot) Jobs in HTCondor

A glidein (pilot) job in HTCondor is a placeholder job that starts on a remote resource, sets up the HTCondor environment, and transforms the resource into a temporary worker node ready to run user jobs from the central HTCondor pool.

Dynamic Resource Provisioning: Jobs start only when needed, reducing idle time.

Late Binding: Matches jobs to the best resources after provisioning.

Central Management: Unified view of all jobs from a single scheduler.

Efficient Setup: Reuses pilots to run multiple jobs, reducing overhead.

TestPool

Helps users with an ACCESS account but no allocation explore the capability

Small amount of compute resources attached to pegasus.access-ci.org

- **CPU: 32 cores, 128 GB RAM, 256 GB disk**
- **GPU: 32 cores, 2 GPUs, 128 GB RAM, 256 GB disk**
- **Hosted on IU Jetstream2, provisioned when needed**

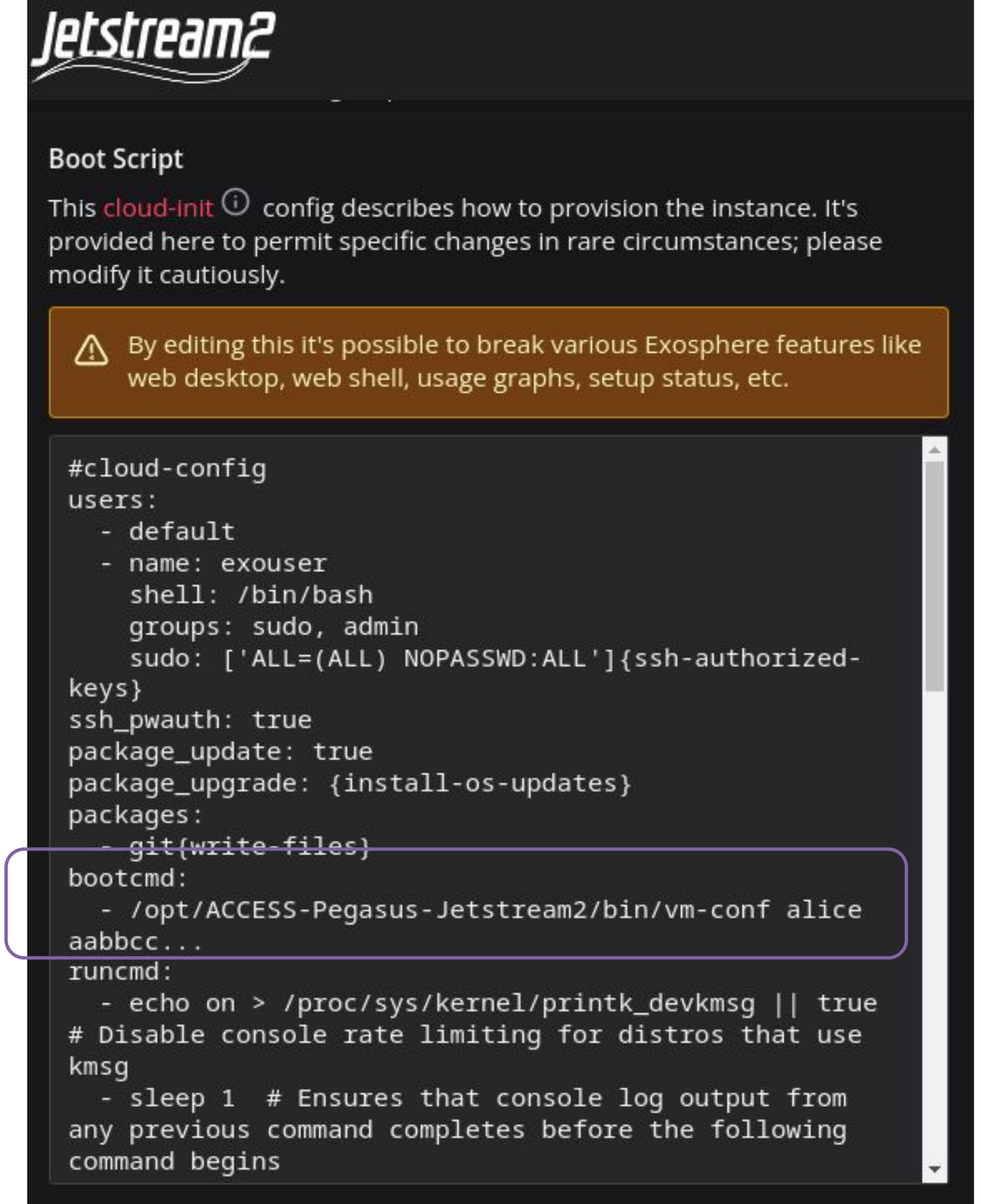
Always available, no allocation needed

Can be used for quick turnaround jobs

- **workflow development and debugging**
- **tutorials (not all users might have an allocation at the time of the tutorial)**

Cloud

- IU JetStream2
- Provided VM image
- Users have to add pegasus.access-ci.org username and token in the cloud-init yaml
- Instances self-terminates when there are no more jobs



HTCondor Annex

- **Bring your own HPC allocation**
- **Semi-managed, submits glideins via SSH.**
  - A glidein can run multiple user jobs - it stays active until no more user jobs are available or until end of life has been reached, whichever comes first.
  - A glidein is partitionable - job slots will dynamically be created based on the resource requirements in the user jobs. This means you can fit multiple user jobs on a compute node at the same time.
  - A glidein will only run jobs for the user who started it.
- **Documentation: https://htcondor.org/experimental/ospool/byoc/**

```
$ htcondor annex create --nodes 1 --lifetime 7200 \
  --project sta230005p --gpu-type v100-16 $USER GPU@bridges2
```

delta
- cpu
- cpu-interactive
- gpuA100x4
- gpuA100x4-preempt
- gpuA100x8
- gpuA40x4
- gpuA40x4-preempt

stampede2
- normal
- development
- skx-normal

expanse
- compute
- gpu
- shared
- gpu-shared

anvil
- wholenode
- wide
- shared
- gpu
- gpu-debug

bridges2
- RM
- RM-512
- RM-shared
- EM
- GPU
- GPU-shared

path-facility
- cpu

pegasus-glidein

- Simple glidein which can be run anywhere, as long as you have outbound network connectivity
- Ties in well with ACCESS Pegasus
- https://github.com/pegasus-isi/pegasus-glidein

```
#!/bin/bash
#SBATCH --job-name=glidein
#SBATCH --nodes=10
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=48
#SBATCH --time=24:00:00

curl -o pegasus-glidein https://raw.githubusercontent.com/pegasus-isi/pegasus-glidein/main/pegasus-glidein
chmod a+x pegasus-glidein
srun ./pegasus-glidein -c pegasus.access-ci.org \
                       -t mytoken \
                       -s 'RemoteOwner == "myusername"'
```