

ACCESS Pegasus

A hosted workflow system for ACCESS users,
bringing their own capacity

Mats Rynge

University of Southern California
Information Sciences Institute



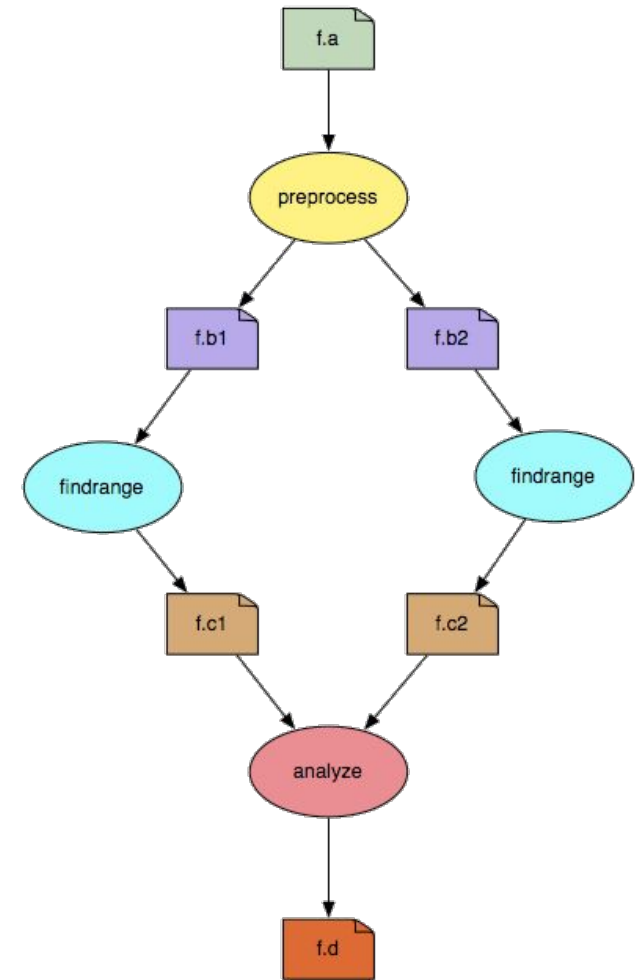
Supported by NSF #2138286

The image features decorative geometric patterns in the corners. The top-right and bottom-left corners contain clusters of shapes including triangles, circles, and semi-circles in shades of teal, orange, and yellow. Some shapes are filled with concentric lines. The central text is positioned between these decorative elements.

Pegasus Workflow Management System

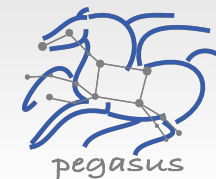
Scientific Workflows

- An abstraction to express ensemble of complex computational operations
 - *Eg: retrieving data from remote storage services, executing applications, and transferring data products to designated storage sites*
- A workflow is represented as a directed acyclic graph (DAG)
 - *Nodes: tasks or jobs to be executed*
 - *Edges: depend between the tasks*
- Have a monolithic application/experiment?
 - *Find the inherent DAG structure in your application to convert into a workflow*



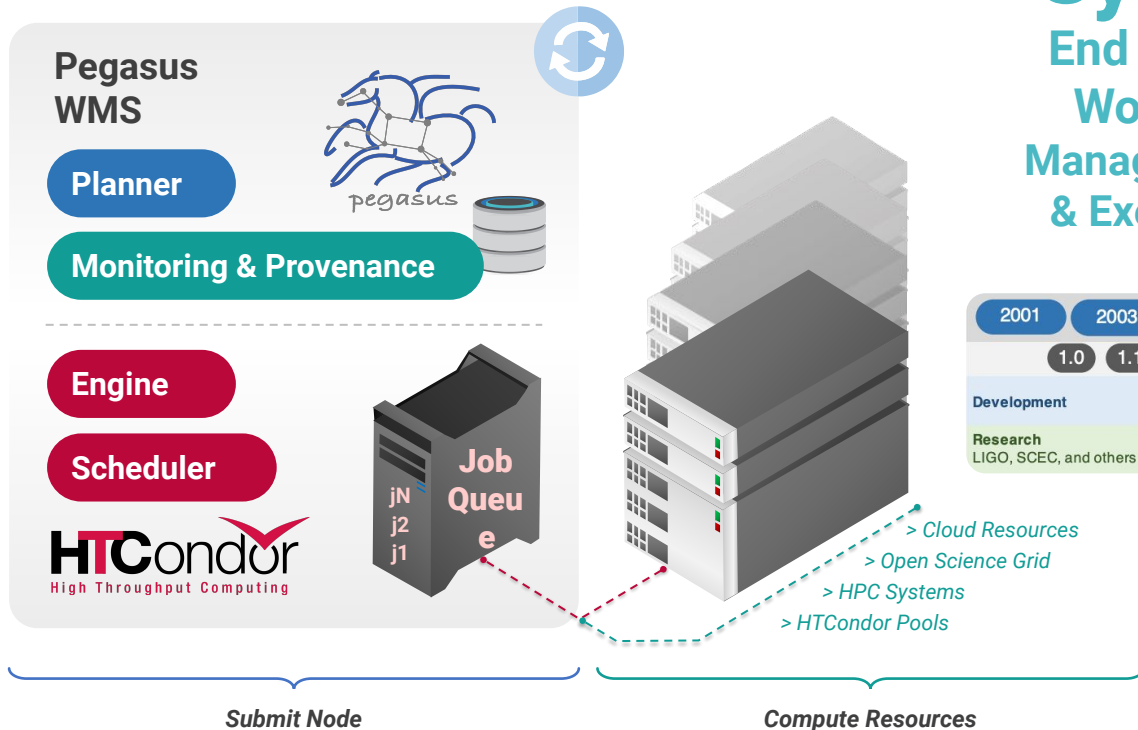


Pegasus Workflow Management System



End to End Workflow Management & Execution

- ▶ Develop portable scientific workflows in Python, Java, and R
- ▶ Compile workflows to be run on heterogeneous resources
- ▶ Monitor and debug workflow execution via CLI and web-based tools
- ▶ Recover from failures with built-in fault tolerance mechanisms
- ▶ Regular release schedule incorporating latest research and development

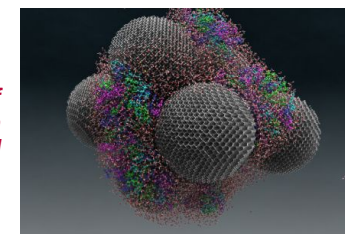


| Year | 2001 | 2003 | 2005 | 2007 | 2009 | 2011 | 2013 | 2015 | 2017 | 2018 | 2020 | | | | | | | | | | | | | |
|-------------|------------------------|-------------------------|------|-----------------|-----------------|----------------------------|-----------------|----------------------------------|---------------------|------------------------------------|------------------|-----|--------------------------|------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|--|
| Version | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 | 2.0 | 2.1 | 2.2 | 2.3 | 2.4 | 3.0 | 3.1 | 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 4.5 | 4.6 | 4.7 | 4.8 | 4.9 | 5.0 | |
| Development | | | | support for GT4 | task clustering | | support for AWS | hierarchical workflows | pegasus-lite engine | monitoring dashboard | ensemble manager | | support for containers | redesign of APIs | | | | | | | | | | |
| Research | LIGO, SCEC, and others | data cleanup algorithms | | data footprint | | cloud computing evaluation | | MPI-based workflow engine design | | Real time performance data capture | metadata capture | | data integrity assurance | | | | | | | | | | | |

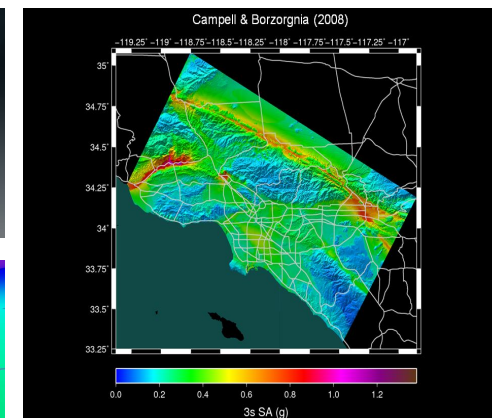
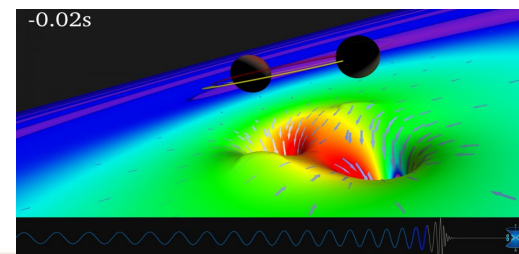


- ▶ Laser Interferometer Gravitational Wave Observatory (LIGO) develops large scale analysis pipelines used for gravitational wave detection.
- ▶ Southern California Earthquake Center (SCEC) CyberShake project generates hazard maps using hierarchical workflows .
- ▶ Oak Ridge National Lab (ORNL) conducted studies on tRNA and nanodiamonds to improve drug delivery design principles.

Visualization of water on nanodiamond spheres from ORNL



LIGO observation of colliding black holes



Hazard map indicating maximum amount of shaking at a particular geographic location generated from SCEC's CyberShake Pegasus workflow



Key Pegasus Concepts

▲ Pegasus WMS == Pegasus planner (mapper) + DAGMan workflow engine + HTCondor scheduler/broker

- Pegasus maps workflows to infrastructure
- DAGMan manages dependencies and reliability
- HTCondor is used as a broker to interface with different schedulers

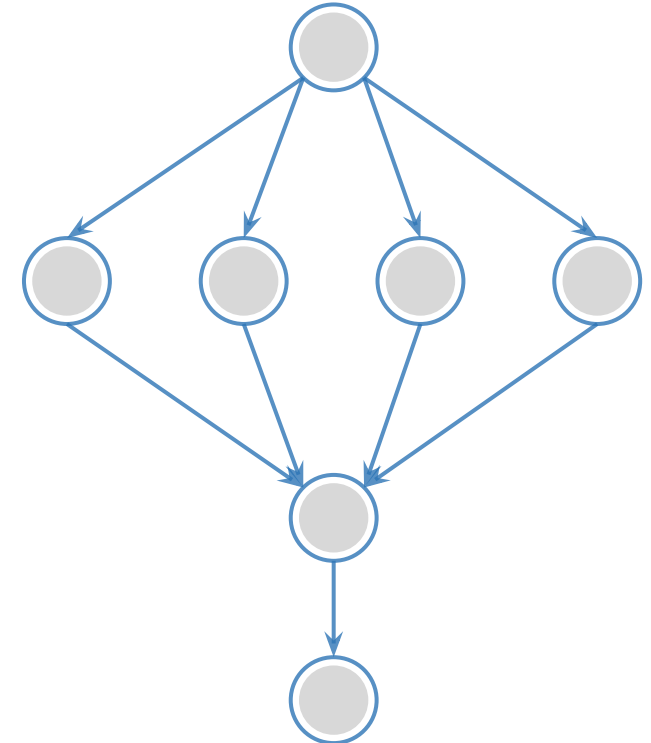
▲ Workflows are DAGs

- Nodes: jobs, edges: dependencies
- No while loops, no conditional branches
- Jobs are standalone executables

▲ Planning occurs ahead of execution

▲ Planning converts an abstract workflow into a concrete, executable workflow

- Planner is like a compiler

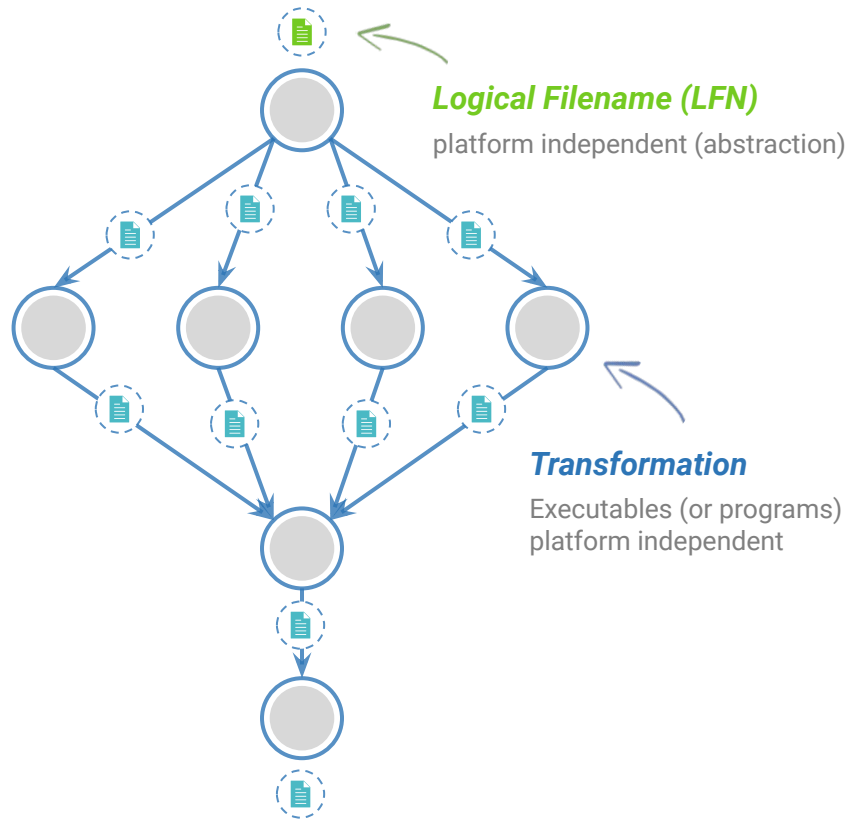


Input Workflow Specification **YAML formatted**

Portable Description

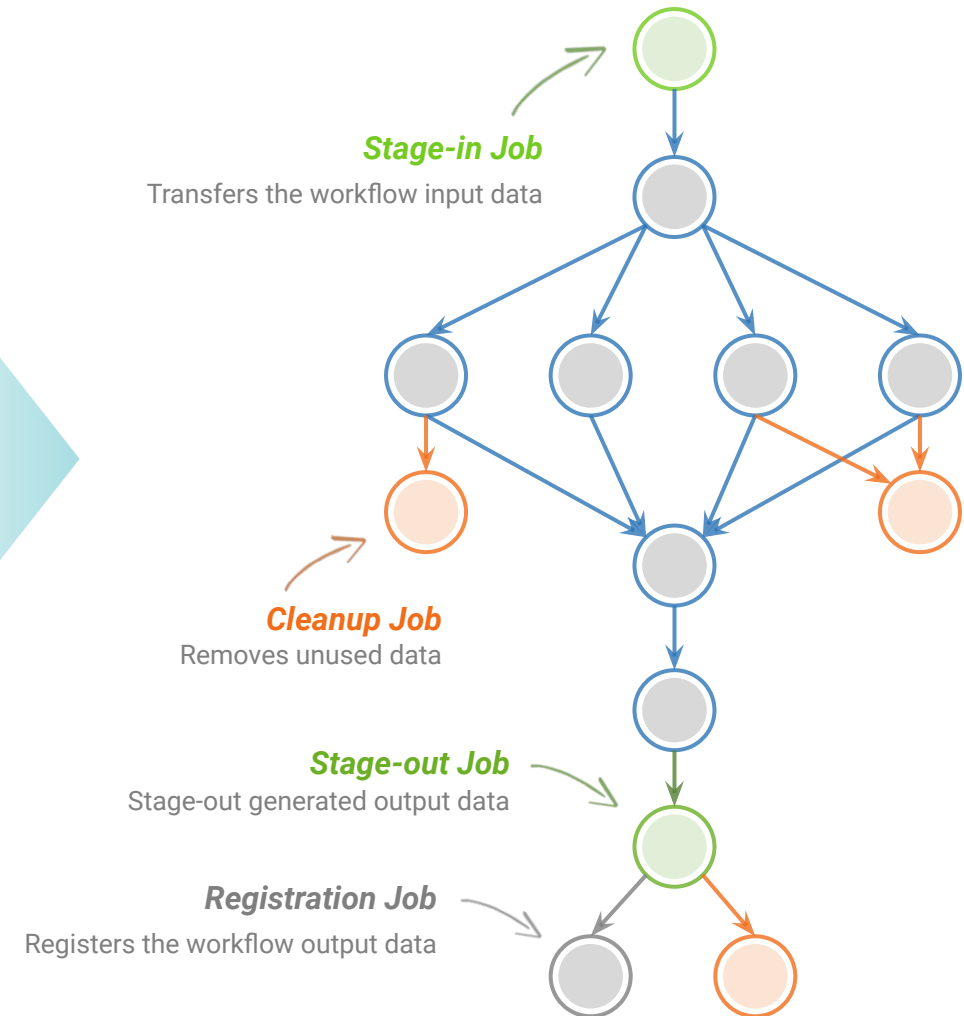
Users do not worry about low level execution details

ABSTRACT WORKFLOW

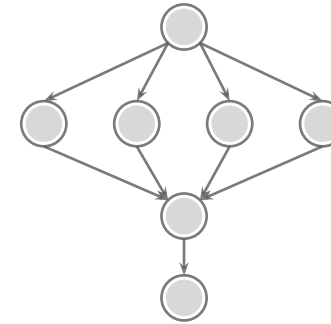


Output Workflow

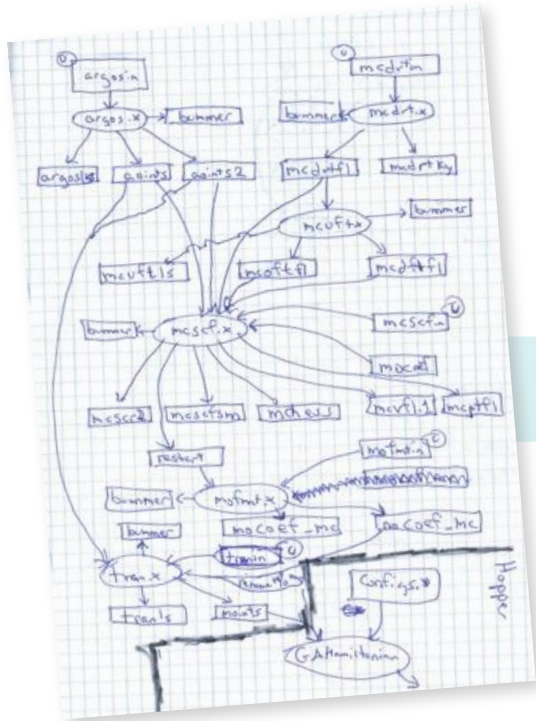
EXECUTABLE WORKFLOW



Pegasus provides APIs to generate the Abstract Workflow



Abstract Workflow



```
#!/usr/bin/env python3

import os
import logging
from pathlib import Path
from argparse import ArgumentParser

logging.basicConfig(level=logging.DEBUG)

# --- Import Pegasus API -----
from Pegasus.api import *

# --- Create Abstract Workflow -----
wf = Workflow("pipeline")

webpage = File("pegasus.html")

# --- Create Parent Job -----
curl_job = (
    Job("curl")
    .add_args("-o", webpage, "http://pegasus.isi.edu")
    .add_outputs(webpage, stage_out=False, register_replica=False)
)

count = File("count.txt")

# --- Create Dependent Job -----
wc_job = (
    Job("wc")
    .add_args("-l", webpage)
    .add_inputs(webpage)
    .set_stdout(count, stage_out=True, register_replica=True)
)

# --- Add jobs to the Abstract Workflow -----
wf.add_jobs(curl_job, wc_job)

# --- Add control flow dependency -----
wf.add_dependency(wc_job, parents=[curl_job])

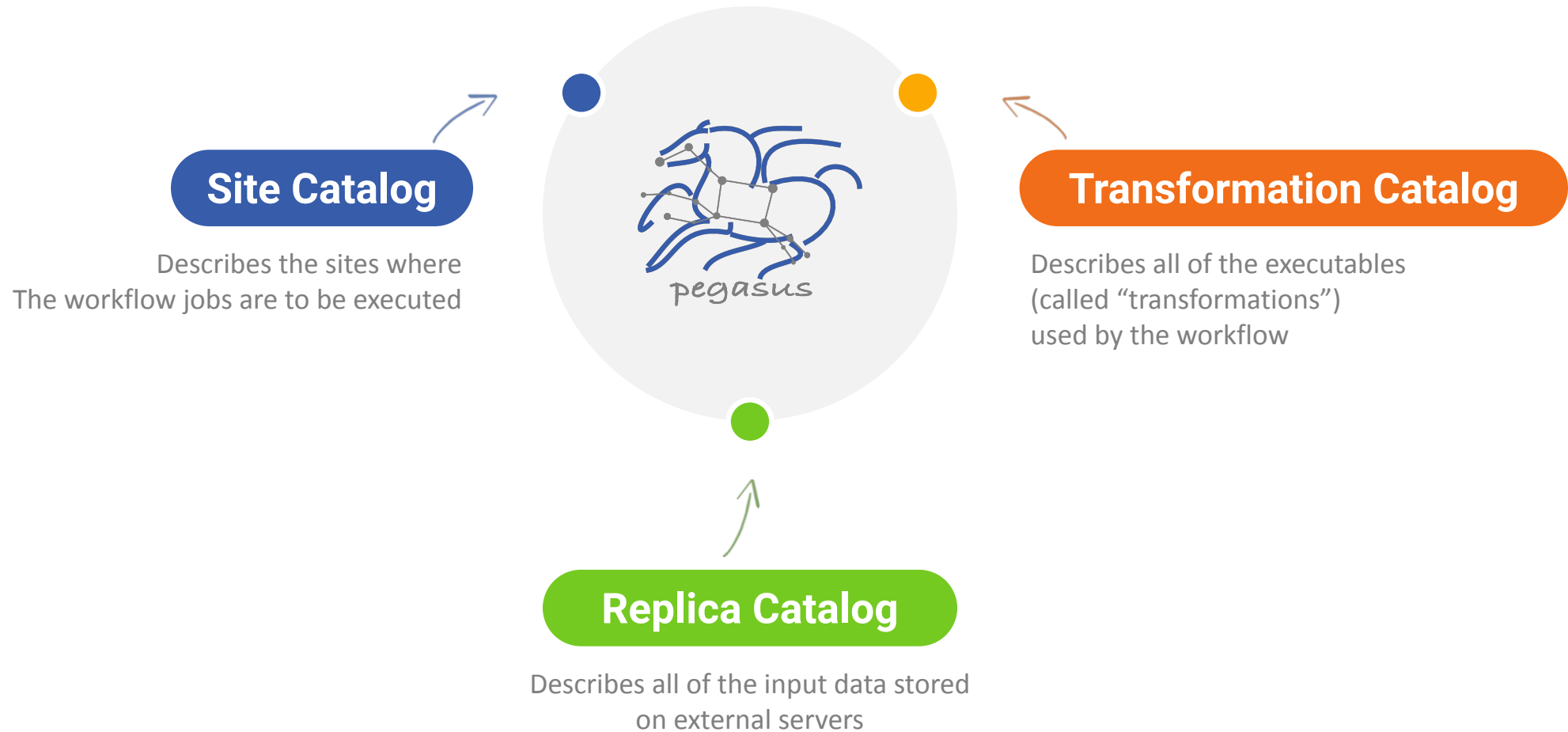
# --- Write out the Abstract Workflow -----
wf.write()
```

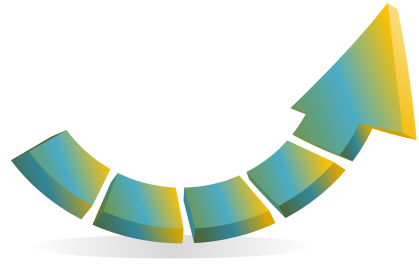


```
x-pegasus:
  apiLang: python
  createdBy: vahi
  createdOn: 11-19-2014:57:58Z
  pegasus: '5.0'
  name: pipeline
  jobs:
  - type: job
    name: curl
    id: ID0000001
    arguments:
    - -o
    - pegasus.html
    - http://pegasus.isi.edu
    uses:
    - lfn: pegasus.html
      type: output
      stageOut: false
      registerReplica: false
  - type: job
    name: wc
    id: ID0000002
    stdout: count.txt
    arguments:
    - -l
    - pegasus.html
    uses:
    - lfn: count.txt
      type: output
      stageOut: true
      registerReplica: true
    - lfn: pegasus.html
      type: input
  jobDependencies:
  - id: ID0000001
    children:
    - ID0000002
```

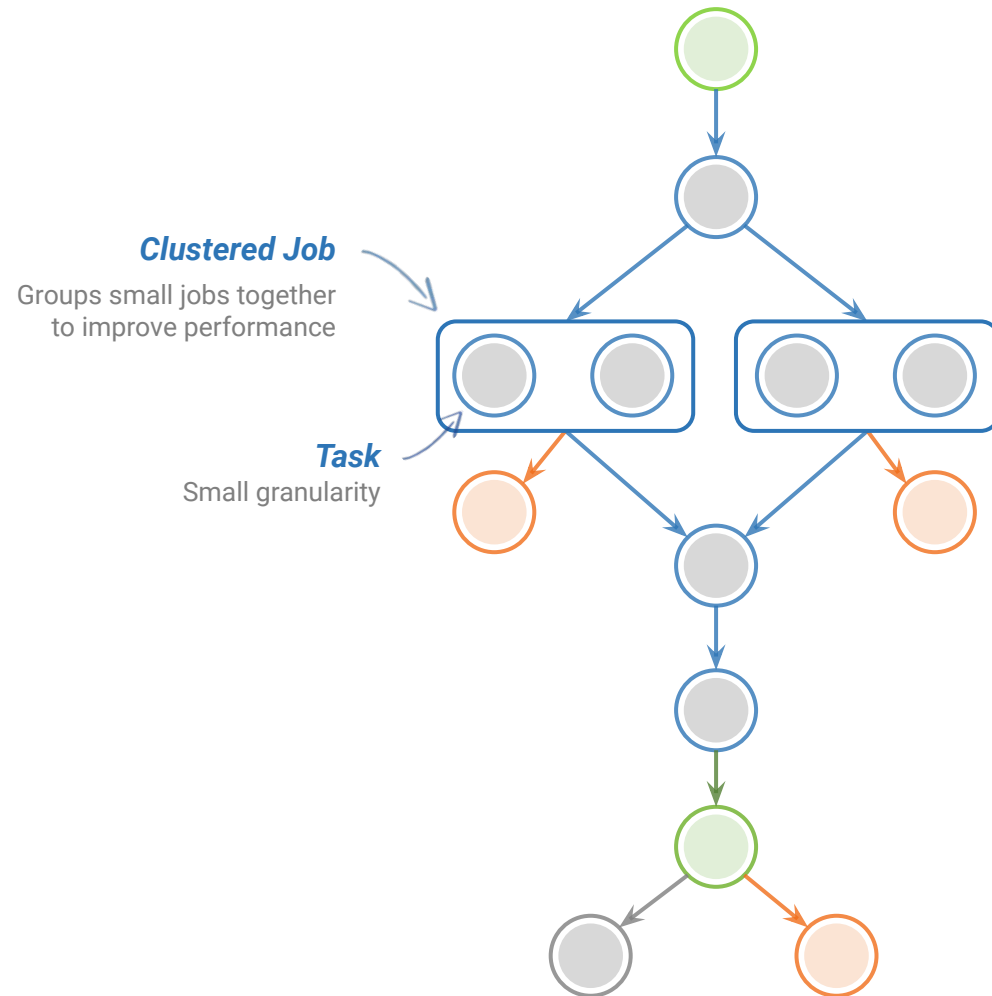
YAML Formatted

So, what other information does Pegasus need?

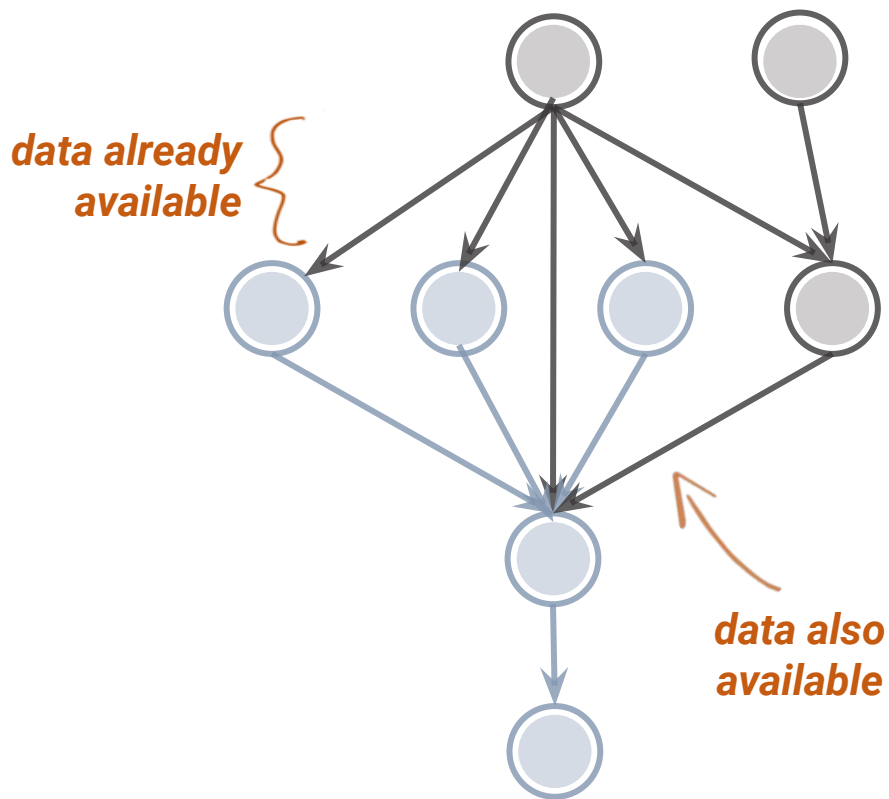




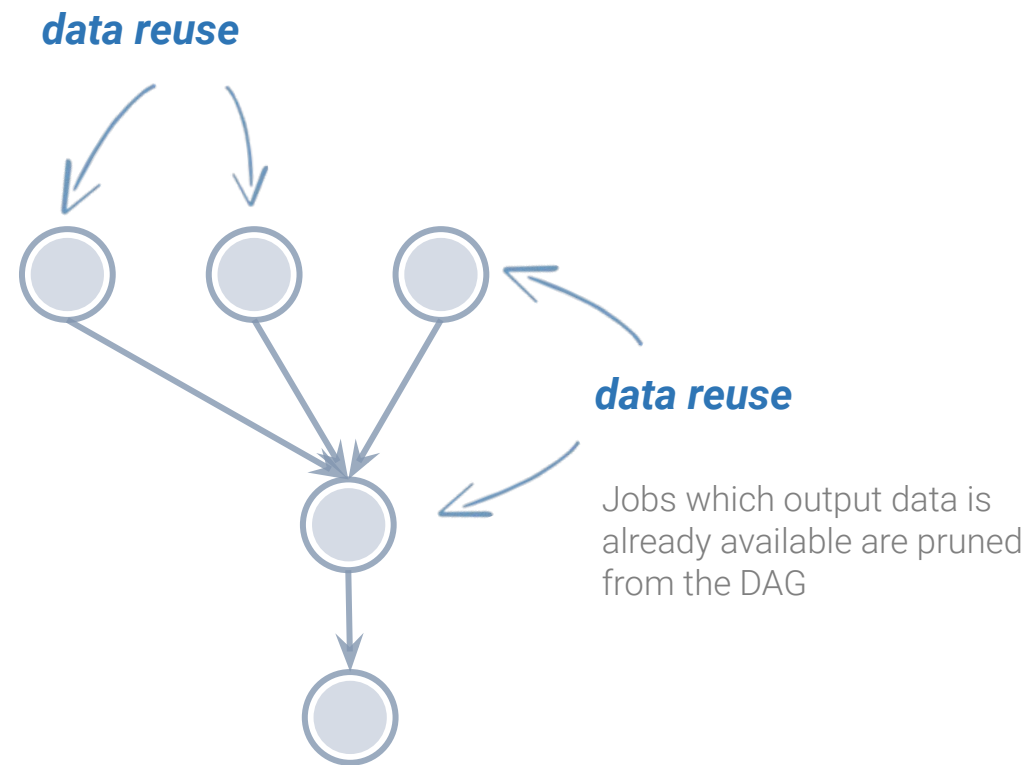
Performance. Why not improve it?



Data Reuse **prune jobs if output data already exists**



workflow
reduction



Pegasus-transfer

Pegasus' internal data transfer tool with support for a number of different protocols

- 🕒 **Directory creation, file removal**
 - If protocol can support it, also used for cleanup

- 🕒 **Two stage transfers**
 - e.g., SCP to S3 = SCP to local file, local file to S3

- 🕒 **Parallel transfers**

- 🕒 **Automatic retries**

- 🕒 **Credential management**
 - Uses the appropriate credential for each site and each protocol (even 3rd party transfers)

```
HTTP
webdav
SCP
GridFTP
Globus Online
iRods
Amazon S3
Google Cloud Storage
SRM
FDT
OSDF / stashcp
Rucio
cp
ln -s
```

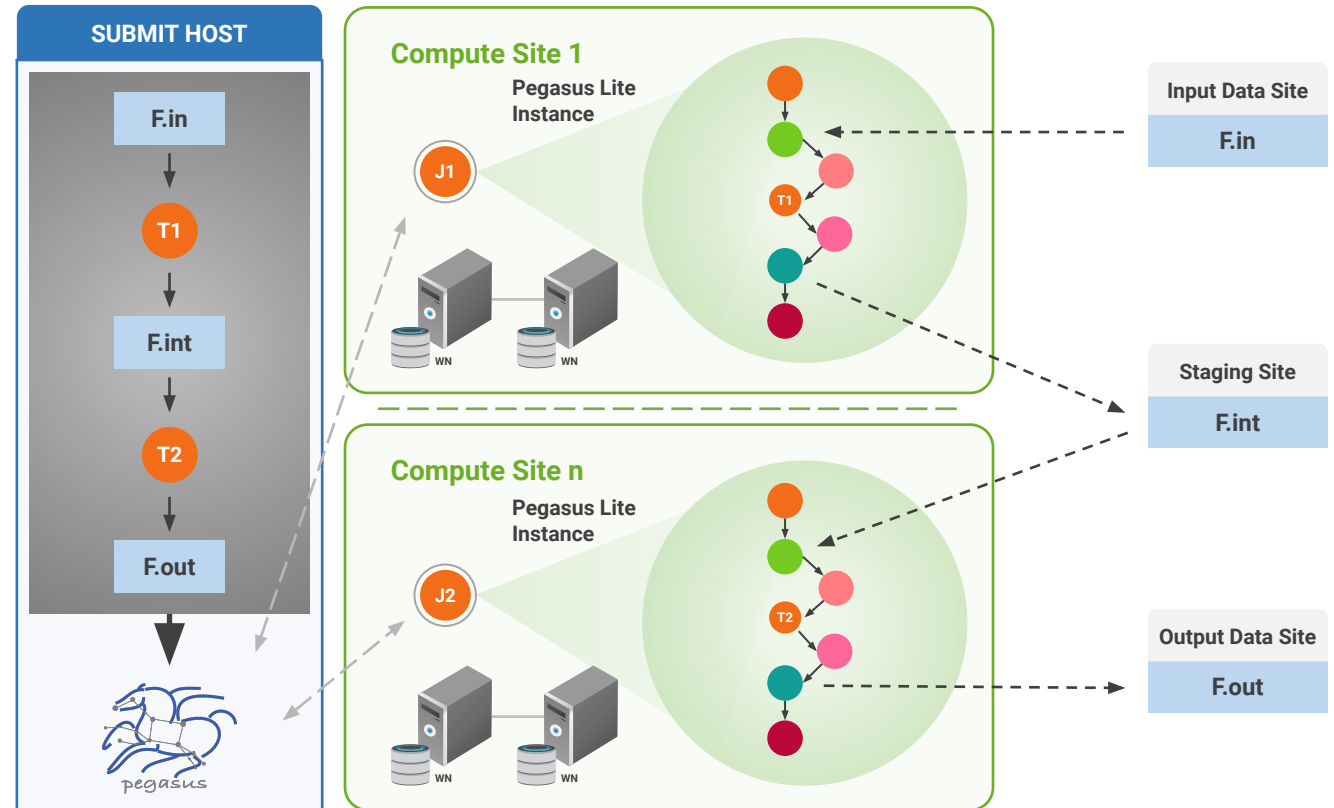


Automatic Integrity Checking

Pegasus performs integrity checksums on input files right before a job starts on the remote node.

- ▶ For raw inputs, **checksums specified in the input replica catalog** along with file locations
- ▶ All **intermediate** and **output** files checksums are generated and tracked within the system.
- ▶ Support for **sha256** checksums

Job failure is triggered if checksums fail



| LEGEND | | | | | |
|---------------|-----------------------|-----------------------|-------------------------|---------------------------|----------------------------|
| ← - - - - - → | Task flow + Checksums | ● Directory Setup Job | ● Data Stageout Job | ● Check Integrity Job | ● Pegasus Lite Compute Job |
| - - - - - → | Data Flow | ● Data Stagein Job | ● Directory Cleanup Job | ● Checksum Generation Job | ● Worker Node (WN) |



And if a job fails?



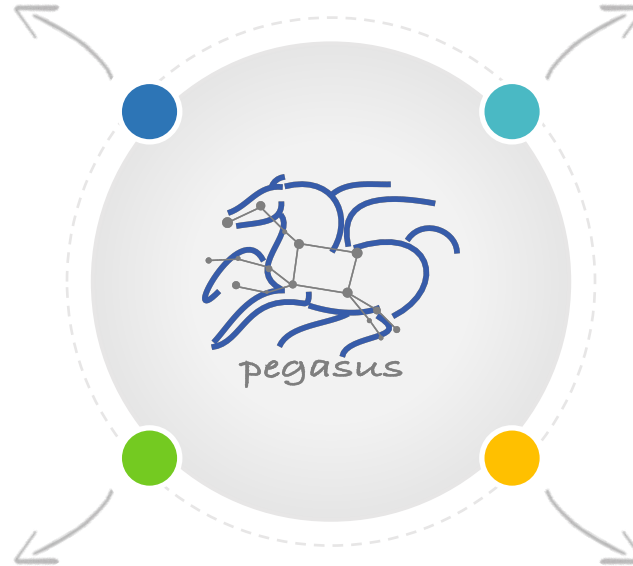
Postscript

detects non-zero exit code output
parsing for success or failure
message exceeded timeout do not
produced expected output files



Checkpoint Files

job generates checkpoint files
staging of checkpoint files is
automatic on restarts



Job Retry



helps with transient failures
set number of retries per job
and run

Rescue DAGs





workflow can be restarted from
checkpoint file recover from
failures with minimal loss

The image features decorative geometric patterns in the corners. The top-right and bottom-left corners contain clusters of shapes including triangles, circles, semi-circles, and concentric arcs in shades of teal, orange, and yellow. The central area is white and contains the text 'Success Stories'.


Success Stories

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



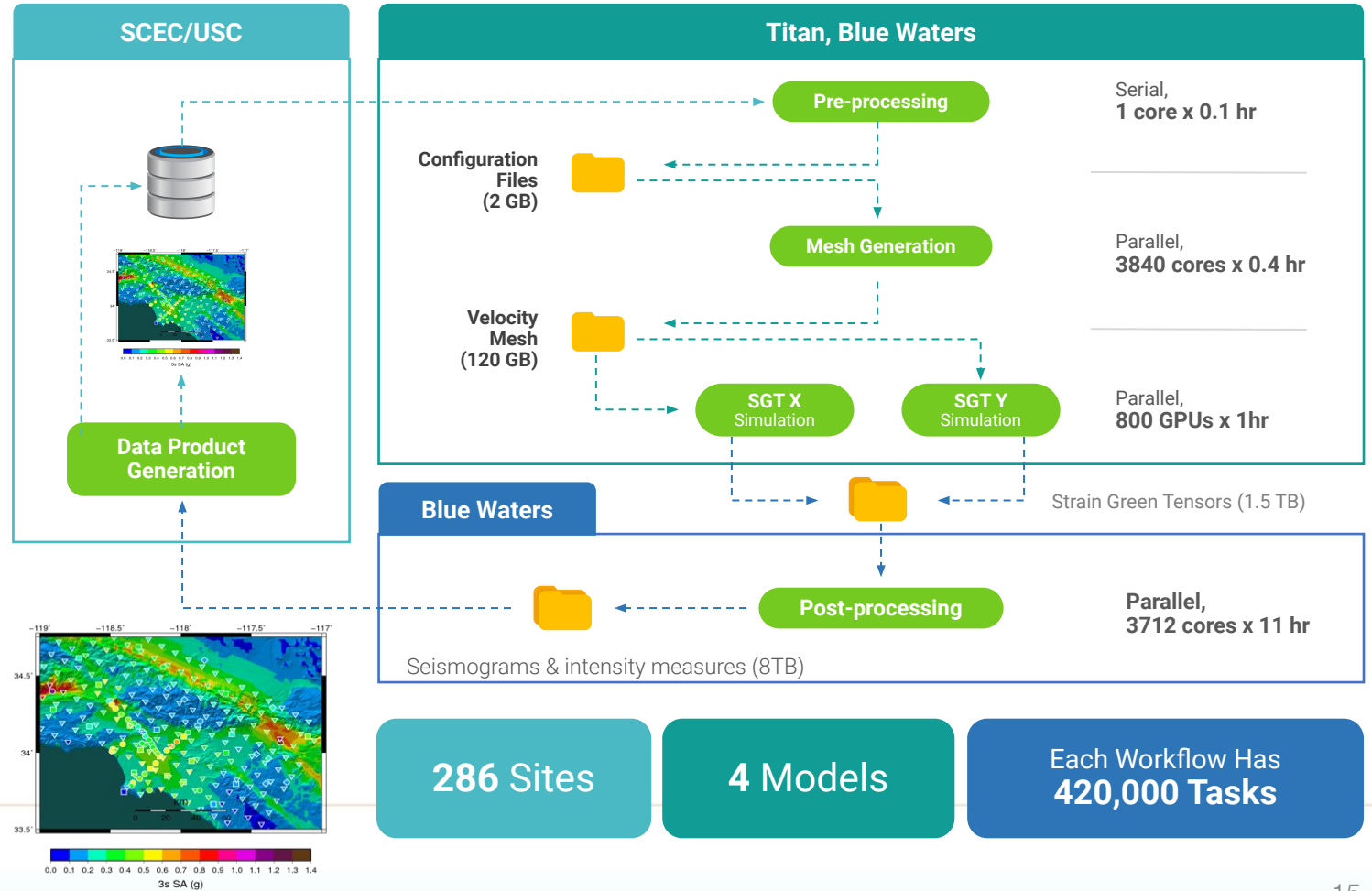
Builders ask seismologists:

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



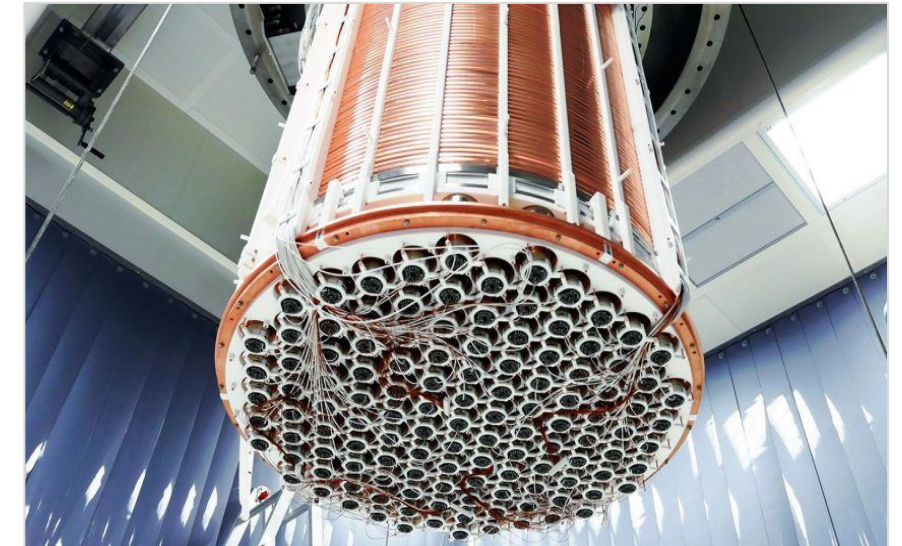
XENONnT - Dark Matter Search



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
    
```

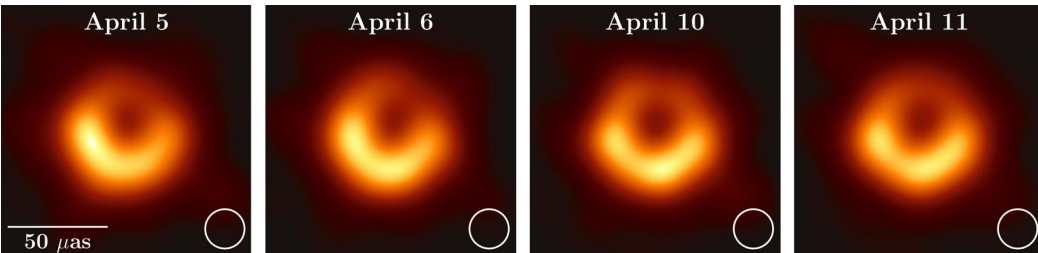
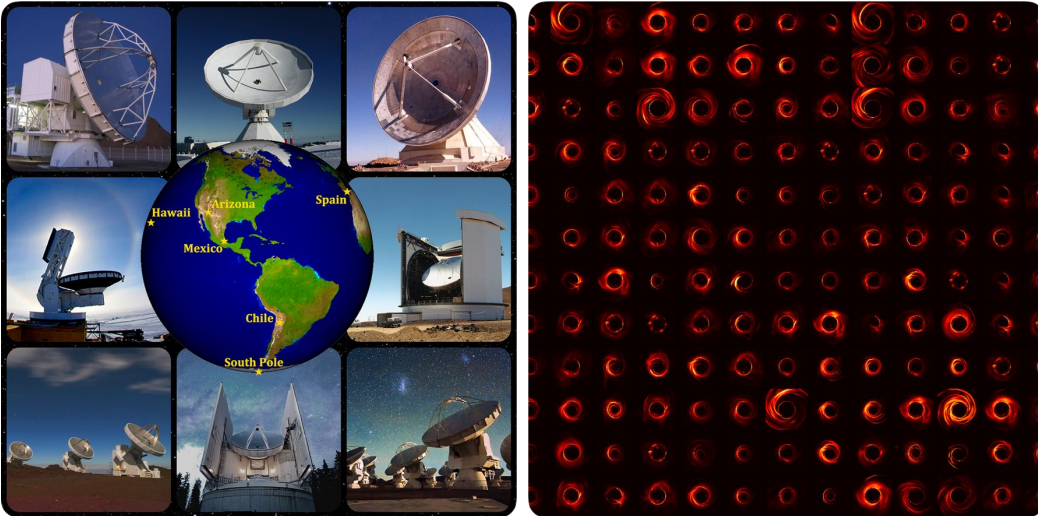


Event Horizon Telescope

Bringing Black Holes into Focus

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

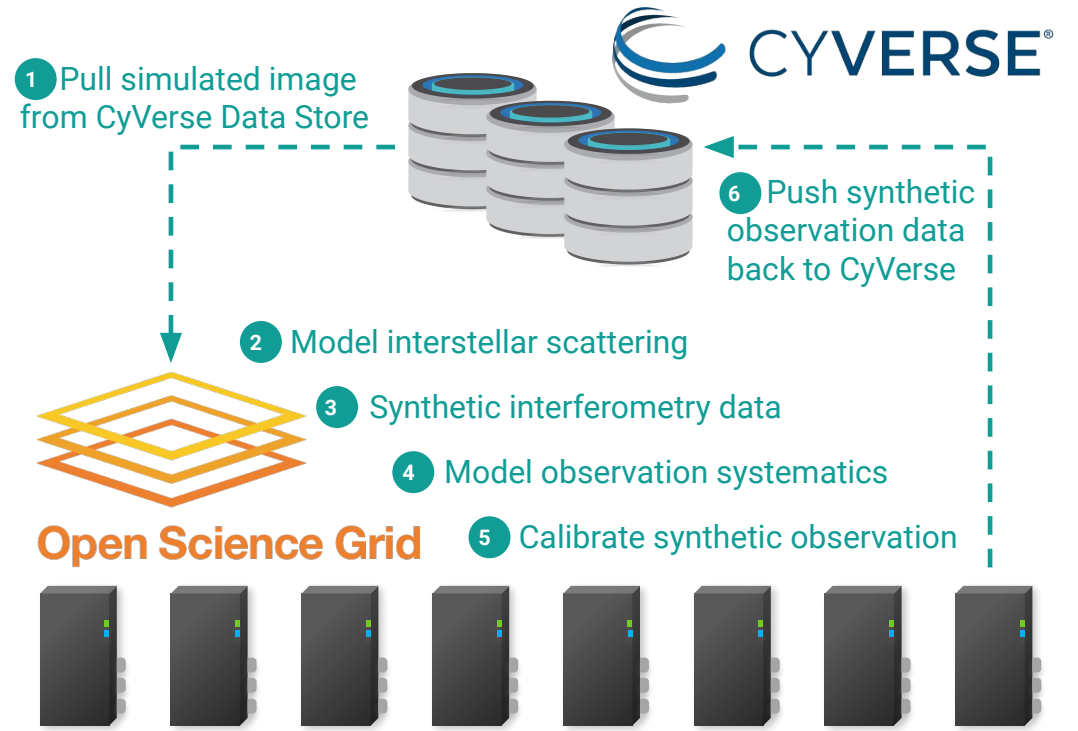
480,000 jobs - 2,600,000 core hours

#15 in all OSG projects in last 6 months

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

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.

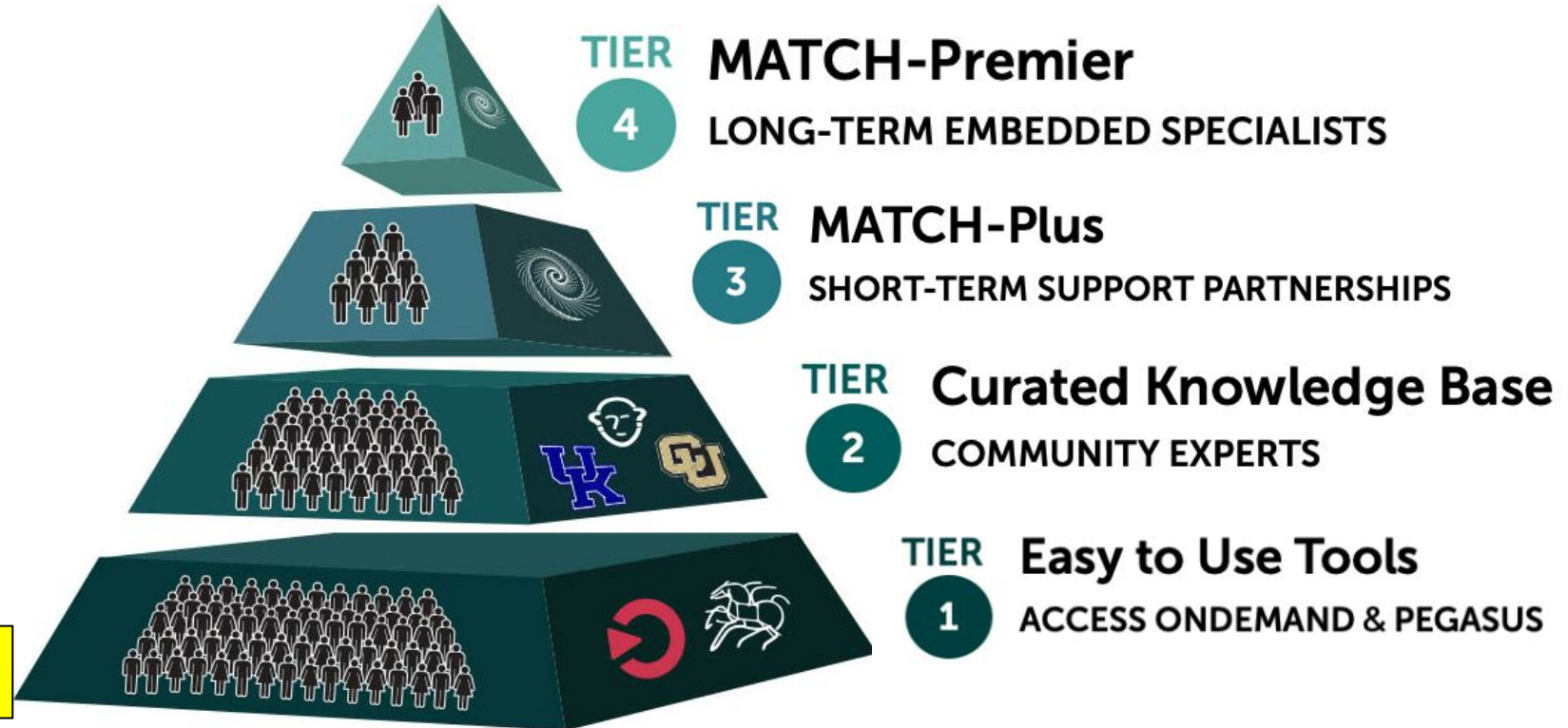


The image features decorative geometric patterns in the corners. The top-right and bottom-left corners contain clusters of shapes including triangles, circles, semi-circles, and concentric arcs in shades of teal, orange, and yellow. The text 'ACCESS Pegasus' is centered in the white space between these patterns.

ACCESS Pegasus

ACCESS Researcher Support Services

- Enable innovative research through equitable and scalable support
- Four tiers of support
- Tools, growing knowledge base
- Match-making with experts
- Student engagement
- Engagement from community
- CSSN incentives



<https://support.access-ci.org>



ACCESS Support Strategy

Reduce the need for support by simplifying access to resources



Powerful Tools & Workflows

nDemand

■ INTEGRATED WEB-BASED INTERFACES

Schedule jobs, manage files, create remote visualizations and use a host of other valuable services.



Pegasus

■ AUTOMATED WORKFLOWS

Simplify complex data workflows on distributed computing resources, such as clusters, grids, and clouds.



ACCESS Pegasus

Leveraging Open OnDemand and Jupyter Notebooks

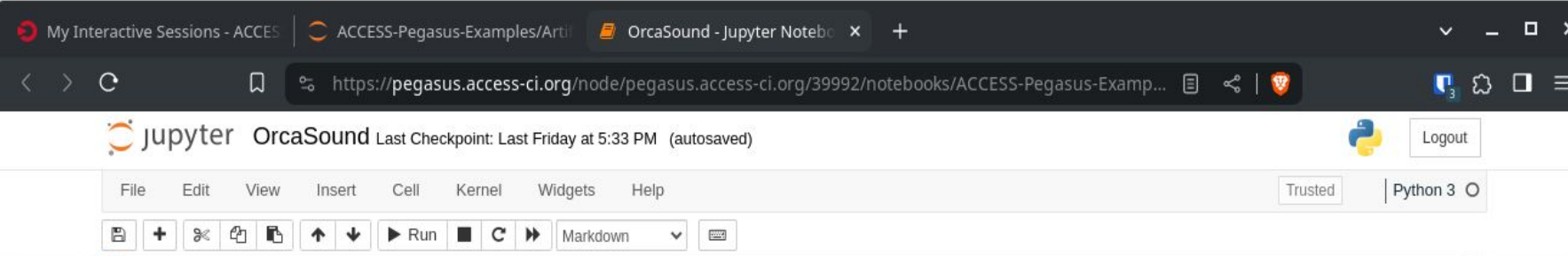
The screenshot shows the ACCESS Pegasus dashboard in a web browser. The browser address bar shows the URL <https://pegasus.access-ci.org/pun/sys/dashboard>. The dashboard has a dark teal header with navigation menus: ACCESS Pegasus, Apps, Files, Clusters, Interactive Apps, ACCESS, and My Interactive Sessions. On the right of the header, there is a Help icon, a user profile icon labeled 'Logged in as ryng', and a Log Out button. The main content area features two large blue-bordered cards. The first card has the Jupyter logo and text: 'Jupyter Notebook (create/manage workflows) System Installed App'. The second card has a terminal icon and text: 'Local Shell Access System Installed App'. To the right of these cards is a 'Getting Started' section with sub-sections for 'Documentation' and 'Quickstart'. The 'Documentation' section lists three links: 'ACCESS Pegasus Overview', 'Detailed ACCESS Pegasus documentation', and 'Pegasus User Guide'. The 'Quickstart' section contains a four-step list: 1. Start an interactive Jupyter session. For the sample 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. 3. Workflows in the tutorials can be run without an allocation. 4. For the domain examples, or your own workflows, provision resources under your allocation with HTCondor HPC Annex. At the bottom of the dashboard, it says 'powered by OPEN OnDemand' and 'OnDemand version: 3.0.3'.

<https://pegasus.access-ci.org/>



ACCESS Pegasus

Leveraging Open OnDemand and Jupyter Notebooks

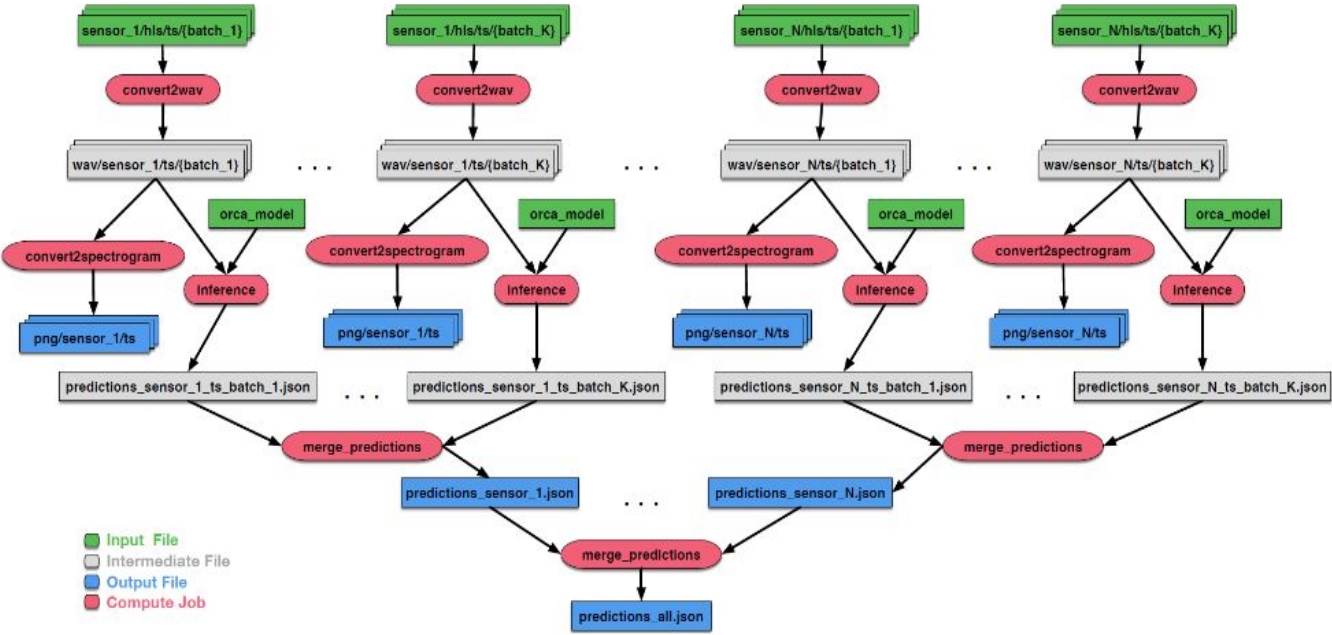


The screenshot shows a Jupyter Notebook interface with the following elements:

- Browser tabs: "My Interactive Sessions - ACCESS", "ACCESS-Pegasus-Examples/Arti", "OrcaSound - Jupyter Notebo", and a plus sign for more tabs.
- Address bar: <https://pegasus.access-ci.org/node/pegasus.access-ci.org/39992/notebooks/ACCESS-Pegasus-Examp...>
- Page title: "OrcaSound Last Checkpoint: Last Friday at 5:33 PM (autosaved)"
- Navigation menu: File, Edit, View, Insert, Cell, Kernel, Widgets, Help
- Trust status: "Trusted" and "Python 3" environment.
- Toolbar: Run, Stop, Refresh, and other notebook controls.

Workflow

The workflow processes and analyzes the hydrophone data and uses trained machine learning models to automatically identify the whistles of the Orcas.



The workflow diagram illustrates the following steps:

- Input Files:** sensor_1/hls/ts/(batch_1) through sensor_N/hls/ts/(batch_K).
- Convert to WAV:** convert2wav jobs produce intermediate files: wav/sensor_1/ts/(batch_1) through wav/sensor_N/ts/(batch_K).
- Convert to Spectrogram:** convert2spectrogram jobs produce intermediate files: png/sensor_1/ts through png/sensor_N/ts.
- Inference:** Inference jobs use the orca_model to produce prediction files: predictions_sensor_1_ts_batch_1.json through predictions_sensor_N_ts_batch_K.json.
- Merge Predictions:** merge_predictions jobs combine intermediate predictions into sensor-specific prediction files: predictions_sensor_1.json through predictions_sensor_N.json.
- Final Output:** A final merge_predictions job produces the aggregated predictions_all.json.

Legend:

- Green box: Input File
- Grey box: Intermediate File
- Blue box: Output File
- Red box: Compute Job

The descriptions for various jobs in the workflow are listed in a table below

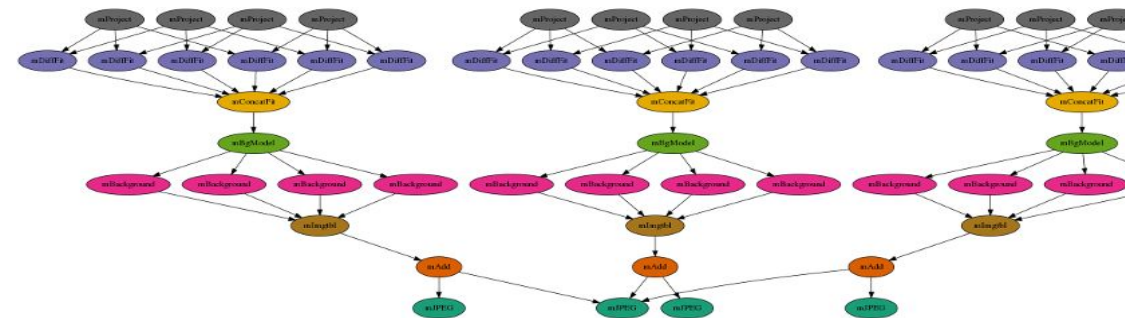
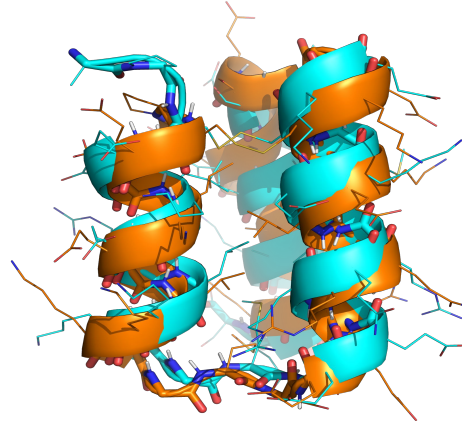
| Job Label | Description |
|---------------------|--|
| convert2wav | converts the input hydrophone data to WAV format |
| convert2spectrogram | converts the WAV output to spectrogram images |
| inference | identifies the sound using a pretrained ML model |
| merge predictions | merges the predictions from all sensors |



Example Workflows

In addition to tutorial workflows, a set of example workflows are automatically installed into each user account - easy to explore, execute and modify!

- Artificial Intelligence
 - Lung Segmentation
 - Mask Detection
 - Orca Sound
 - LLM + RAG
- Astronomy
 - Montage
- Bioinformatics
 - Alphafold
 - Rosetta
 - VariantCalling

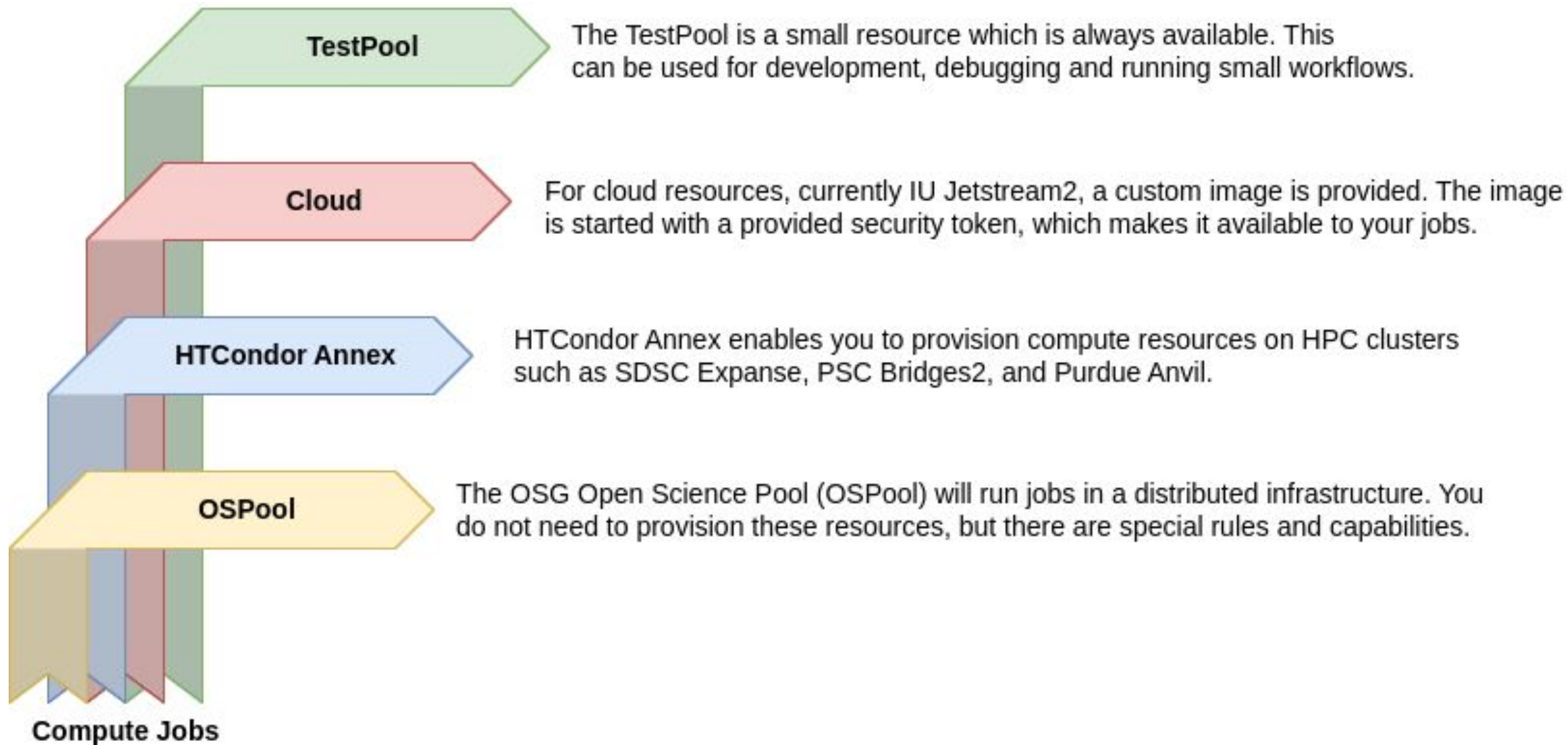


Bring Your Own Capacity

- ACCESS CI consists of allocatable resources
- Motivations
 - Increased / known capacity
 - GPUs
 - Parallel filesystems, I/O
 - Clouds / configurable
- ACCESS Pegasus connects to both shared and BYOC resources

| Project Type | Explore | Discover | Accelerate | Maximize |
|---------------------------------|--|--|--|--------------------------------|
| ACCESS Credits* | 400,000 | 1,500,000 | 3,000,000 | Awarded in resource units |
| Project duration | Supporting grant duration or 12 months | Supporting grant duration or 12 months | Supporting grant duration or 12 months | 12 months |
| Requests accepted | Anytime | Anytime | Anytime | Every 6 months |
| | Multiple requests allowed | Multiple requests allowed | Multiple requests allowed | 1 allowed (some exceptions) |
| Requirements and review process | Overview | 1-page proposal | 3-page proposal (max. length) | 10-page proposal (max. length) |
| | Confirmation of eligibility and suitability of requested resources | Confirmation of eligibility and suitability of requested resources | Panel merit review | Panel merit review |





Shared Capacity - TestPool

Helps users with an ACCESS account but no allocation try 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)




BYOC - 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

Boot Script

This `cloud-init` config describes how to provision the instance. It's provided here to permit specific changes in rare circumstances; please modify it cautiously.

 By editing this it's possible to break various Exosphere features like web desktop, web shell, usage graphs, setup status, etc.

```
#cloud-config
users:
  - default
  - name: exouser
    shell: /bin/bash
    groups: sudo, admin
    sudo: ['ALL=(ALL) NOPASSWD:ALL'] {ssh-authorized-
keys}
ssh_pwauth: true
package_update: true
package_upgrade: {install-os-updates}
packages:
  - git(write-files)
bootcmd:
  - /opt/ACCESS-Pegasus-Jetstream2/bin/vm-conf alice
  aabbcc...
runcmd:
  - echo on > /proc/sys/kernel/printk_devkmsg || true
# Disable console rate limiting for distros that use
kmsg
  - sleep 1 # Ensures that console log output from
any previous command completes before the following
command begins
```

BYOC - 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

expans

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



Shared or BYOC - OSPool



- pegasus.access-ci.org is an OSPool access point, via flocking
- Jobs have to set +ProjectName, which can be an ACCESS allocation (if the user have one), or any registered OSG project name
- No OSDF origin
 - Can use the ACCESS allocated Open Storage Network (OSN), which is a S3 compatible object store



Example AI Workflow - LLM + RAG

LLM RAG (Large Language Model Retrieval-Augmented Generation) is a technique that enhances large language models by incorporating information retrieval mechanisms. It involves retrieving relevant information from a database or document corpus and combining it with the original query to provide additional context. This augmented input is then processed by the large language model to generate more accurate and contextually relevant responses. LLM RAG offers benefits such as improved accuracy, access to up-to-date information, and better contextual understanding, making it useful for applications like question answering, summarization, and conversational AI.

- Container includes LLM
- Inputs: IU Jetstream2 documentation PDF
- Same query twice, once without RAG and one with RAG
- “Please provide a summary of the GPU capabilities of the IU Jetstream2 system. Include the instance flavors, and any details about the GPUs.”
- Can execute on GPUS from any of the capacity providers: TestPool, Cloud, HTCondor Annex, OSPool

Summary without RAG:

The IU Jetstream2 system is a high-performance computing resource provided by Indiana University for research purposes. Regarding its GPU capabilities, Jetstream2 offers several instance flavors that include GPUs:

1. `gpu96`: This flavor provides 4 NVIDIA V100 GPUs (with 32GB of memory each) and 128GB of RAM on the host CPU node.
2. `gpu60`: This flavor offers 2 NVIDIA V100 GPUs (each with 32GB of memory) and 64GB of RAM on the host CPU node.

...

Summary with RAG:

The IU Jetstream2 system has 90 GPU-enabled nodes with four NVIDIA A100 GPUs each. These are divided using NVIDIA virtual GPU (vGPU) to allow allocations to utilize from 1/5th of a GPU to an entire GPU in instances, facilitating use for educational and research workloads requiring varying amounts of GPU processing power.

The Jetstream2 system offers several GPU instance flavors: g3.small, g3.medium, g3.large, and g3.xl. Here's a summary of their specifications:

- g3.small: 4 vCPUs, 15 GB RAM, 60 GB local storage, 20% GPU compute, 20 GB GPU RAM
- g3.medium: 8 vCPUs, 30 GB RAM, 60 GB local storage, 25% GPU compute, 10 GB GPU RAM
- g3.large: 16 vCPUs, 60 GB RAM, 60 GB local storage, 50% GPU compute, 20 GB GPU RAM
- g3.xl: 32 vCPUs, 125 GB RAM, 60 GB local storage, 100% GPU compute, 40 GB GPU RAM



The image features decorative geometric patterns in the top right and bottom left corners. These patterns consist of various shapes including circles, triangles, and semi-circles in shades of teal, yellow, and orange, some with concentric line patterns.

Thank You!

<https://support.access-ci.org/tools/pegasus>

<https://pegasus.isi.edu>

Deployment Scenarios

HTCondor Pool

Workflow jobs are submitted to an existing HTCondor pool, and handled directly by HTCondor

Slurm (+HTCondor)

Common setup on campus clusters. Pegasus and HTCondor are installed on a head node. Workflow jobs are submitted to HTCondor, which translates the jobs to Slurm jobs. Good solution for parallel jobs.

Glideins / Pilot Jobs

Variation of HTCondor Pool, but the capacity is dynamically added to the pool by submitting pilot jobs. The pilot jobs will become part of the pool, and will start executing workflow jobs.

Hosted

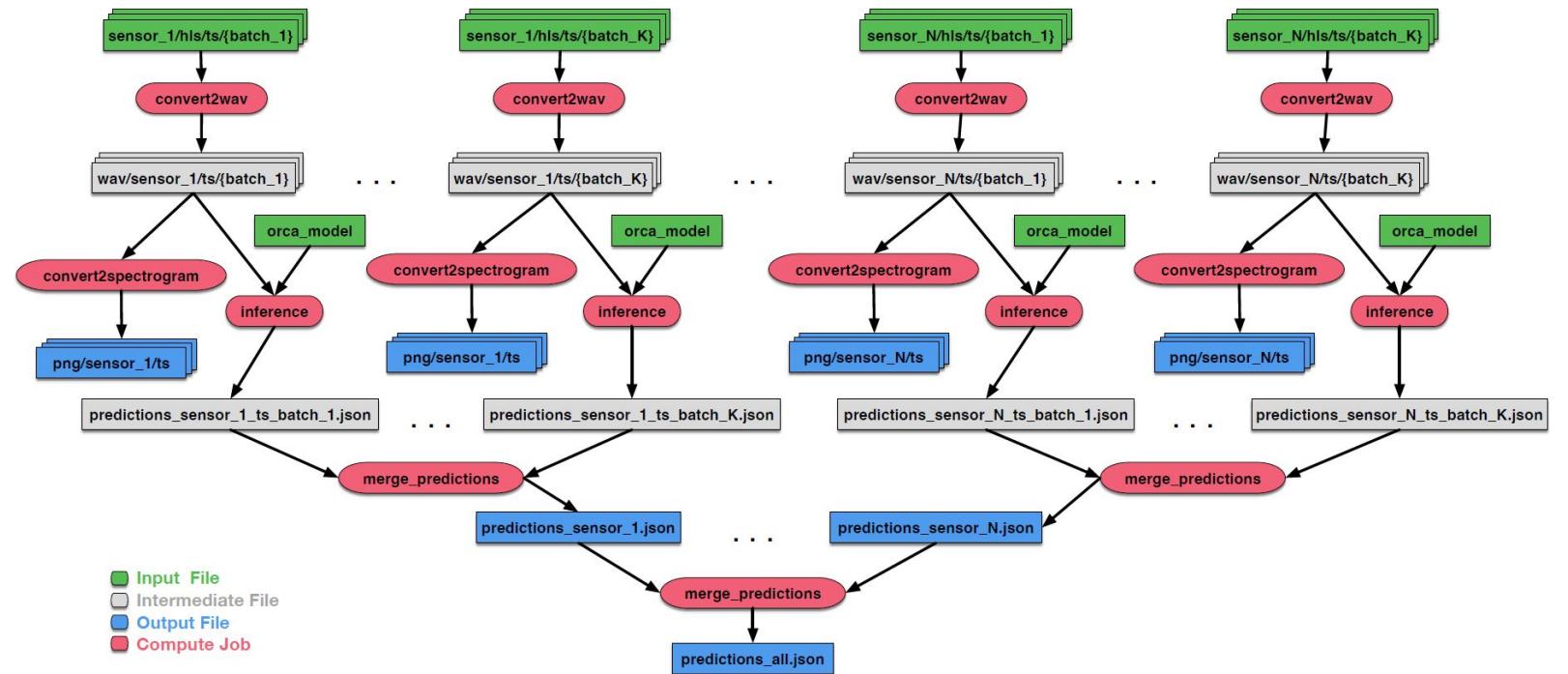
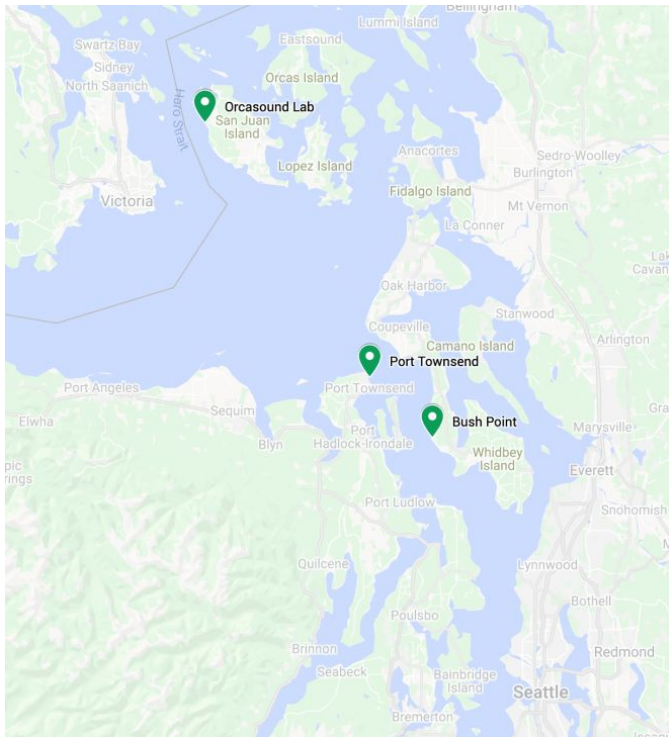
Larger infrastructure projects might have hosted instances, commonly built on top of either a static pool or glideins. [ACCESS Pegasus](#) described later is an example.



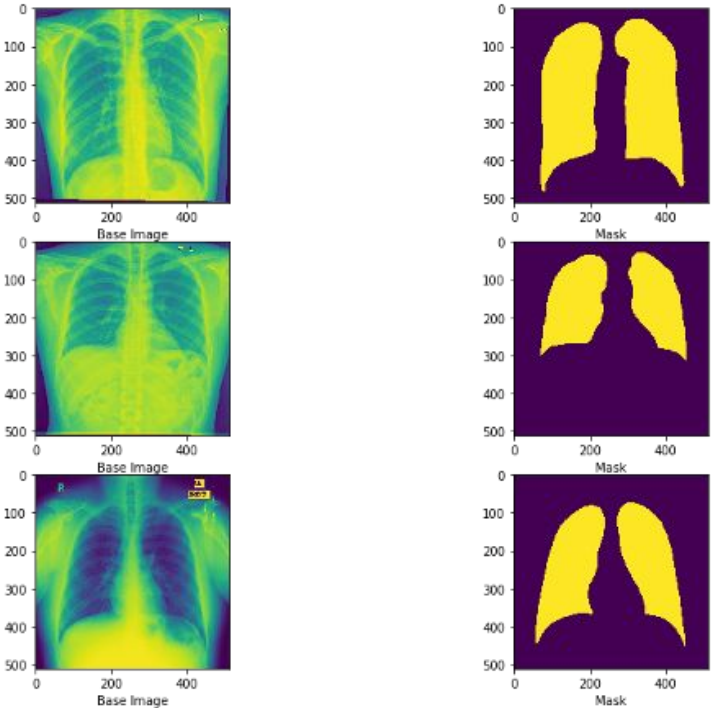
AI Workflows - Orcasound

The Ocean Observatories Initiative (OOI), through a network of sensors, supports critical research in ocean science and marine life. Orcasound is a community driven project that leverages hydrophone sensors deployed in three locations in the state of Washington (San Juan Island, Point Bush, and Port Townsend as shown in the figure below) in order to study Orca whales in the Pacific Northwest region.

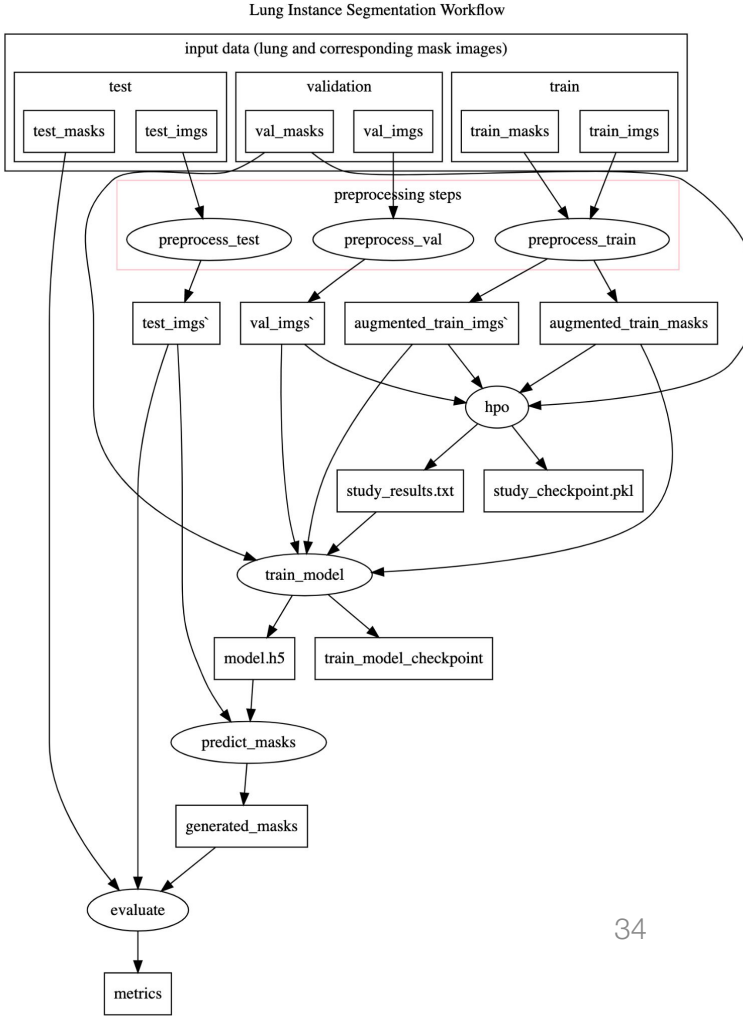
The workflow processes the hydrophone data of one or more sensors in batches for each timestamp, and converts them to a WAV format. Using the WAV output it creates spectrogram images that are stored in the final output location. Furthermore, using the pretrained Orcasound model, the workflow scans the WAV files to identify potential sounds produced by the orcas.



AI Workflows - Lung Segmentation



Precise detection of the borders of organs and lesions in medical images such as X-rays, CT, or MRI scans is an essential step towards correct diagnosis and treatment planning. We implement a workflow that employs supervised learning techniques to locate lungs on X-ray images. Lung instance segmentation workflow uses Chest X-ray for predicting lung masks from the images using U-Net model.

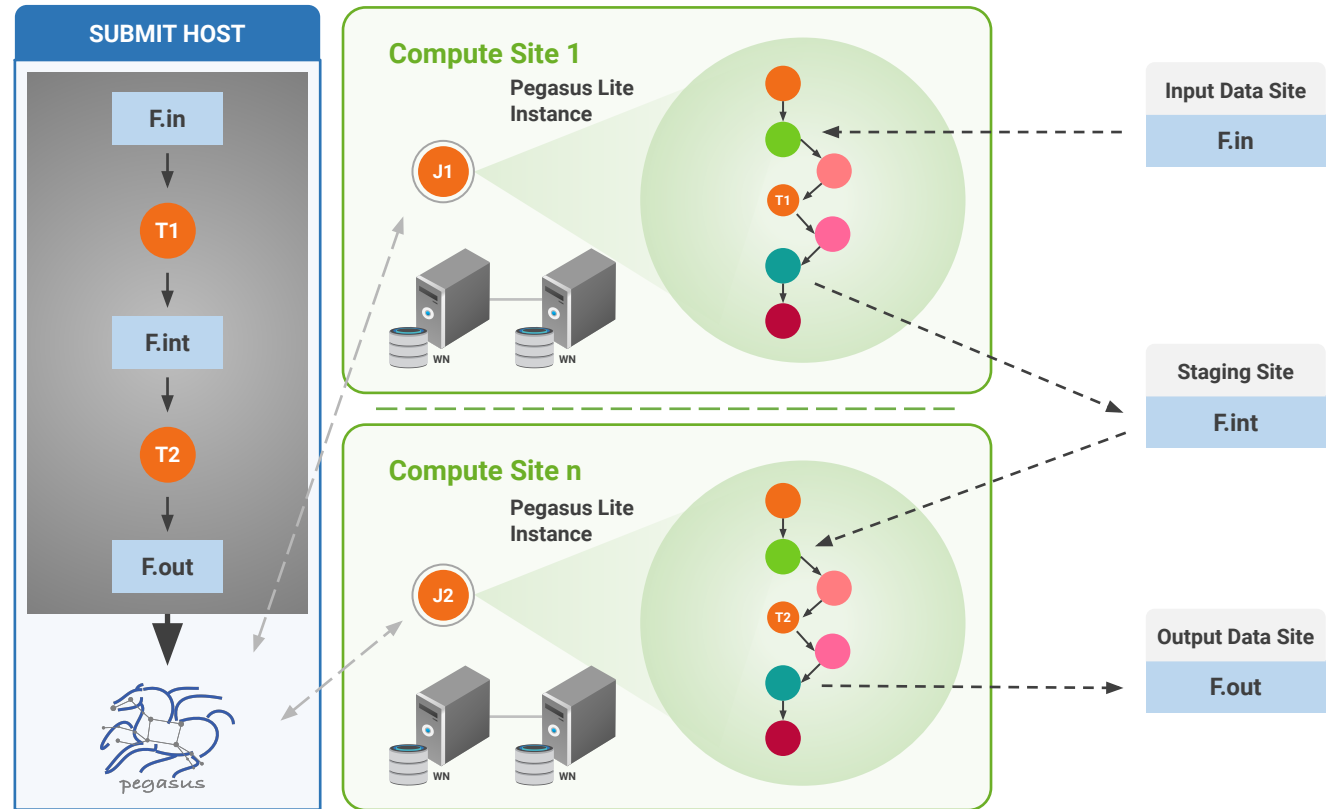


Automatic Integrity Checking in Pegasus

Pegasus performs integrity checksums on input files right before a job starts on the remote node.

- ▶ For raw inputs, **checksums specified in the input replica catalog** along with file locations
- ▶ All **intermediate** and **output** files checksums are generated and tracked within the system.
- ▶ Support for **sha256** checksums

Job failure is triggered if checksums fail



| LEGEND | | | | | | | | | |
|--------|-----------------------|--|---------------------|--|-----------------------|--|-------------------------|--|--------------------------|
| | Task flow + Checksums | | Directory Setup Job | | Data Stageout Job | | Check Integrity Job | | Pegasus Lite Compute Job |
| | Data Flow | | Data Stagein Job | | Directory Cleanup Job | | Checksum Generation Job | | Worker Node (WN) |



Pegasus Container Support



Users can refer to **containers** in the **Transformation Catalog** with their executable preinstalled



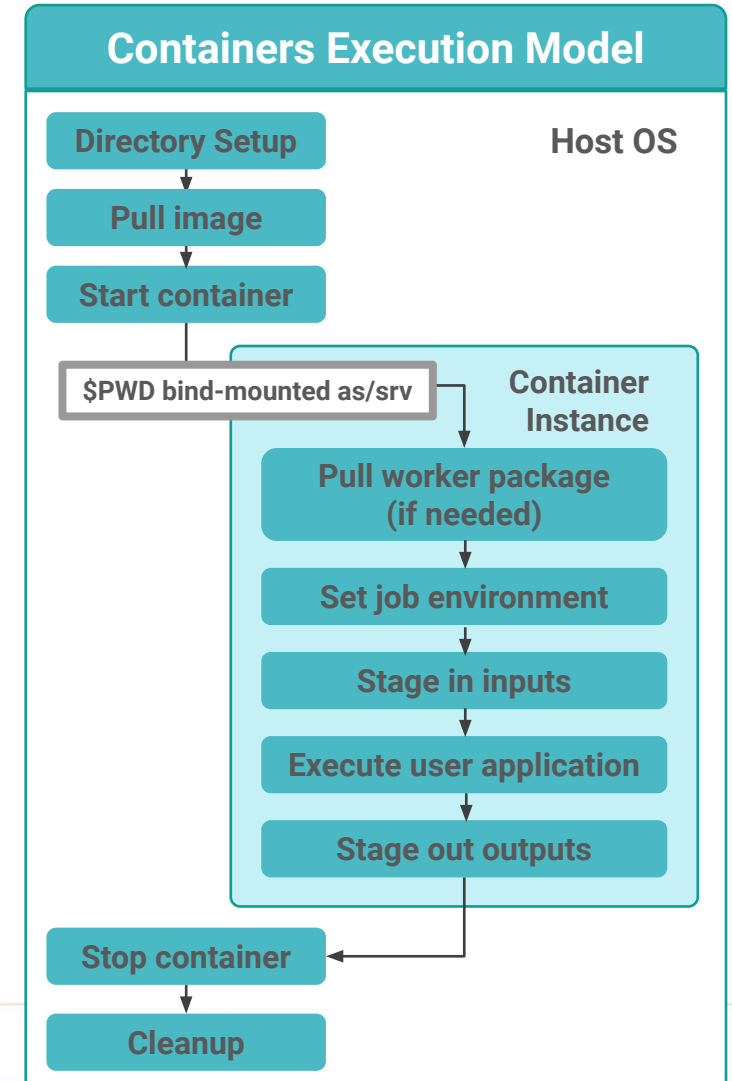
Users can **refer** to a **container** they want to **use** – **Pegasus stages** their executables and containers to the node

- Useful if you want to use a site recommended/standard container image.
- Users are using generic image with executable staging.



Future Plans

- Users can **specify an image buildfile** for their jobs.
- *Pegasus will build the Docker image as separate jobs in the executable workflow, export them as a tar file and ship them around*



Data Management for Containers



Containers are data too!

Pegasus treats containers as input data dependency

- Staged to compute node if not present
- Docker or Singularity Hub URL's
- Docker Image exported as a TAR file and available at a server, just like any other input dataset

Scaling up for larger workflows

- The image is pulled down as a tar file as part of data stage-in jobs in the workflow
- The exported tar file is then shipped with the workflow and made available to the jobs
- Pricing considerations. You are now charged if you exceed a certain rate of pulls from Hubs

Other Optimizations

- **Symlink** against **existing images** on shared file system such as **CVMFS**
- The exported tar file is then shipped with the workflow and made available to the jobs

