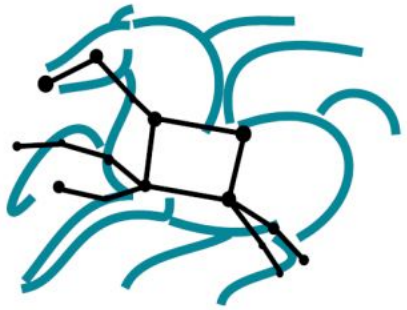


Learn Pegasus Workflows

Five Tutorials Across Time Zones

pegasus-support@isi.edu
Pegasus Users Slack: https://pegasus.isi.edu/contact/
Office Hours: https://pegasus.isi.edu/office-hours/

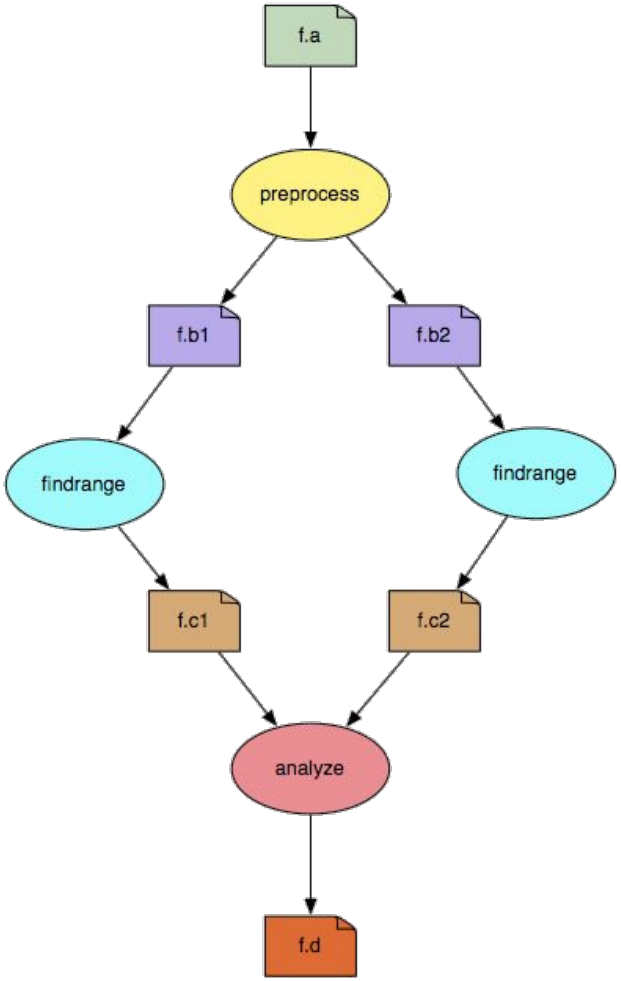
Supported by NSF #2138286

Tutorial Background

1. High Throughput Computing (HTC) / HPC
   a. This tutorial is mostly HTC, but Pegasus can run HPC workloads as well
2. Why ACCESS?
   a. Pegasus can be used in a variety of different ways, such as cli/Jupyter deployed on your machine, a cluster or cloud. One options is a hosted system like the ACCESS Pegasus one we are using today.
   b. ACCESS Pegasus acts as a gateway to some of the largest open access resources
3. Why AI/GPUs in the tutorial?
   a. A fun example, which in addition to teaching Pegasus, demonstrates how to use GPUs on ACCESS
4. There will be homework!

Scientific Workflows

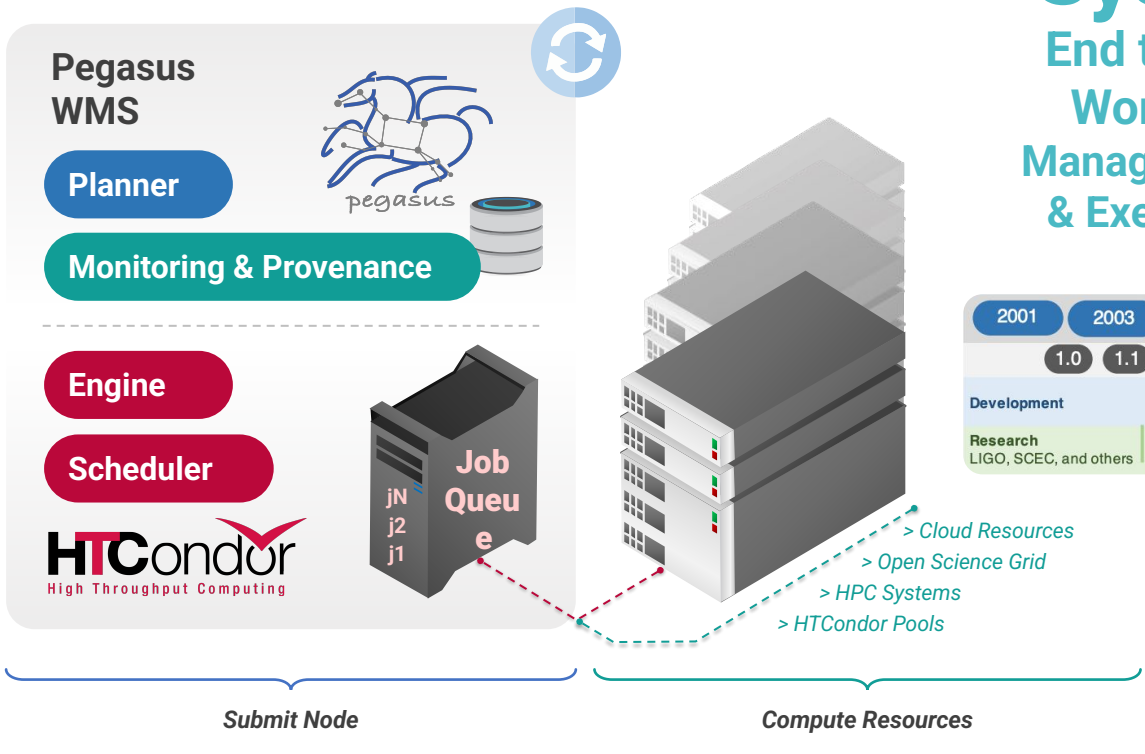
- An abstraction to express ensemble of complex computational operations
  - *Eg: retrieving data from remote storage services, executing applications, and transferring data products to designated storage sites*
- A workflow is represented as a directed acyclic graph (DAG)
  - *Nodes: tasks or jobs to 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*

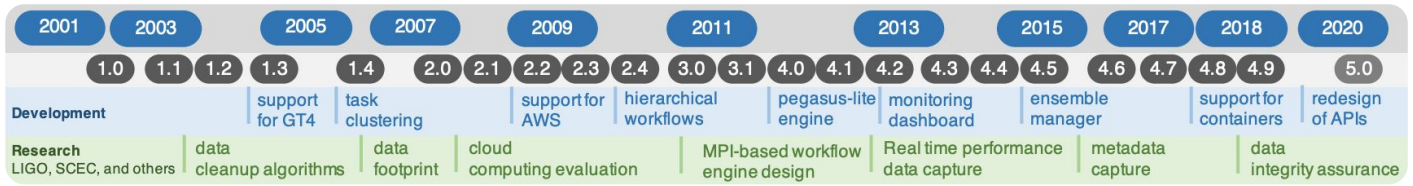


Pegasus Workflow Management System

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

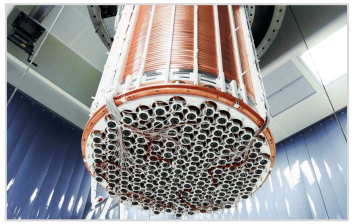




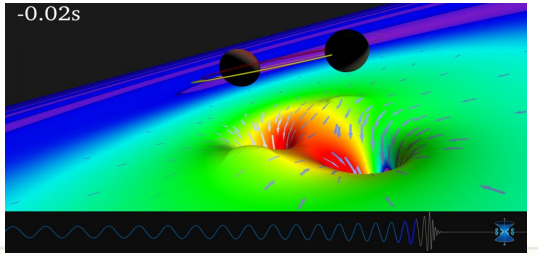
Pegasus in practice

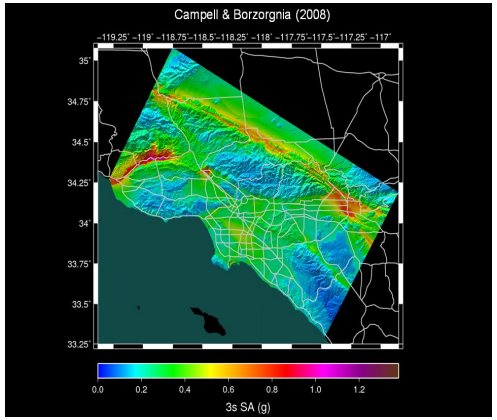
- Laser Interferometer Gravitational Wave Observatory (LIGO) develops large scale analysis pipelines used for gravitational wave detection.
- Southern California Earthquake Center (SCEC) CyberShake project generates hazard maps using hierarchical workflows .
- The XENONnT project uses Pegasus for processing and monte carlo workflows, searching for dark matter

The XENONnT detector



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

ACCESS Support

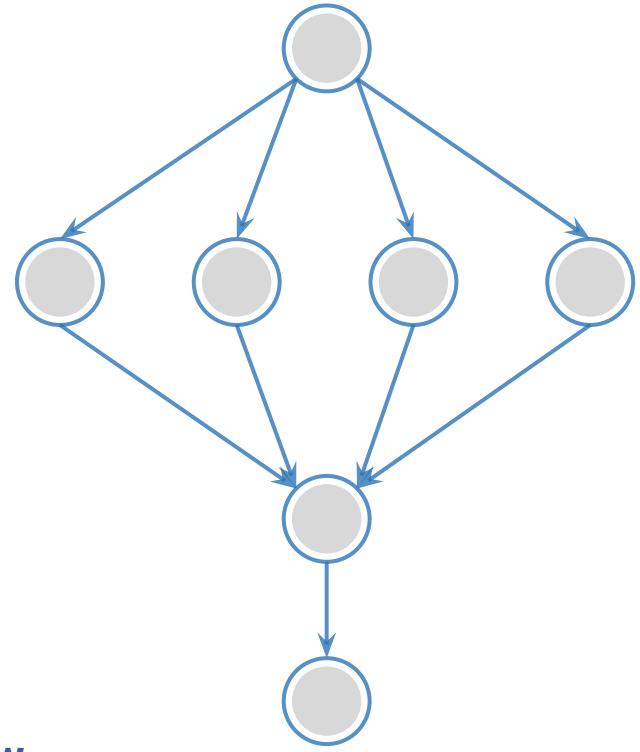
Key Pegasus Concepts

- **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

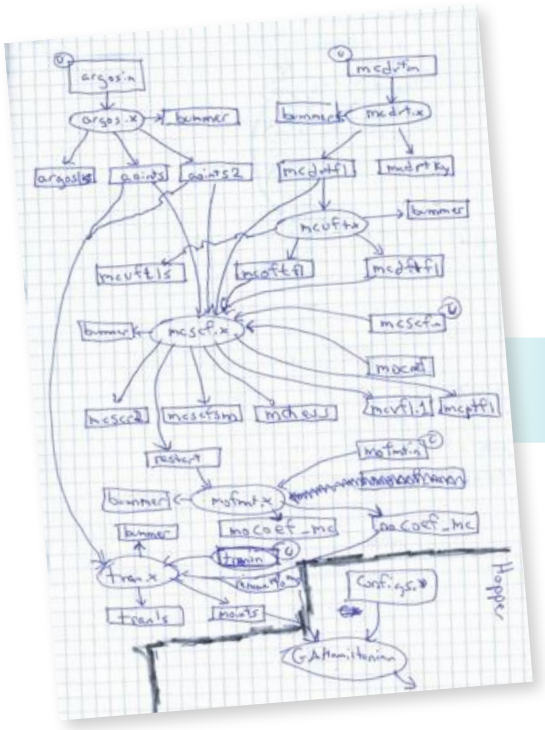
- **Workflows are DAGs**
  - Nodes: jobs, edges: dependencies
  - No while loops, no conditional branches
  - Jobs are standalone executables

- **Planning occurs ahead of execution**

- **Planning converts an abstract workflow into a concrete, executable workflow**
  - Planner is like a compiler



Pegasus provides APIs to generate the Abstract Workflow

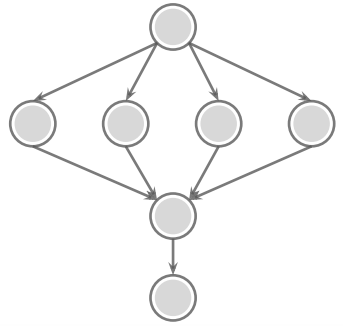


```python
#!/usr/bin/env python3

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()
```

python™

Java

jupyter



Abstract Workflow

```yaml
x-pegasus:
  apiLang: python
  createdBy: vahi
  createdOn: 11-19-20T14:57:58Z
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
```

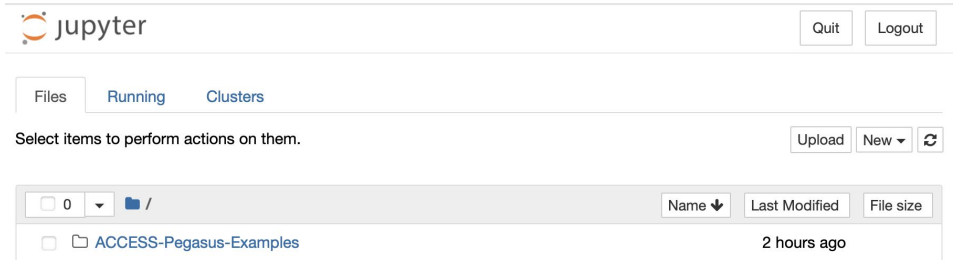
YAML Formatted

Tutorial Setup

Set of Jupyter Notebooks, contains a mix of overview, pointers, workflows. Step through them one cell at the time, starting from the top.

Log in to ACCESS Pegasus, and start a Jupyter Notebook.

Tutorials can be found in the ***ACCESS-Pegasus-Examples*** directory.



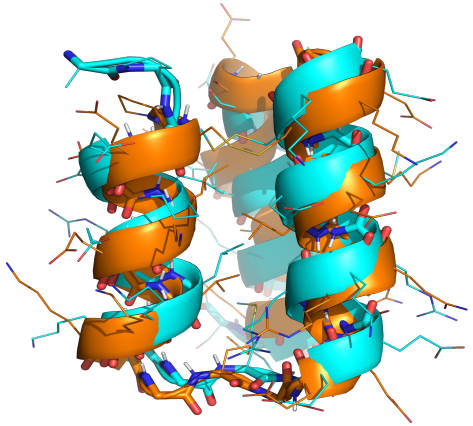
If you do not already have an ACCESS account: https://operations.access-ci.org/identity/new-user

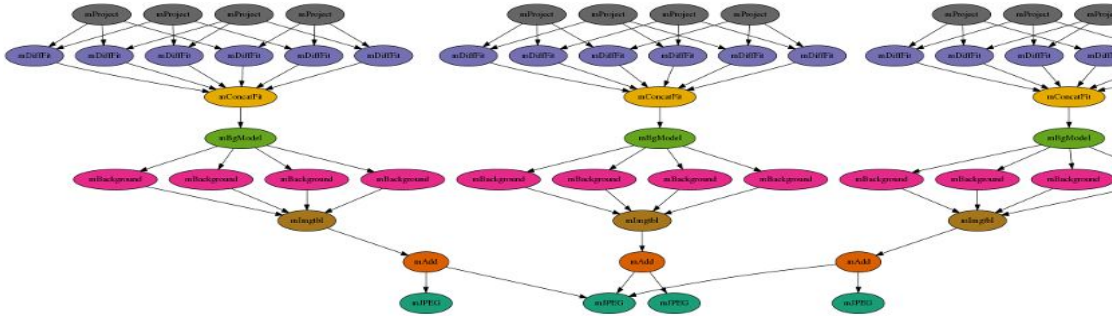
Using your ACCESS account, verify that you can use a web browser to log into https://pegasus.access-ci.org

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
  - VarientCalling





Hands on: Running our first workflow

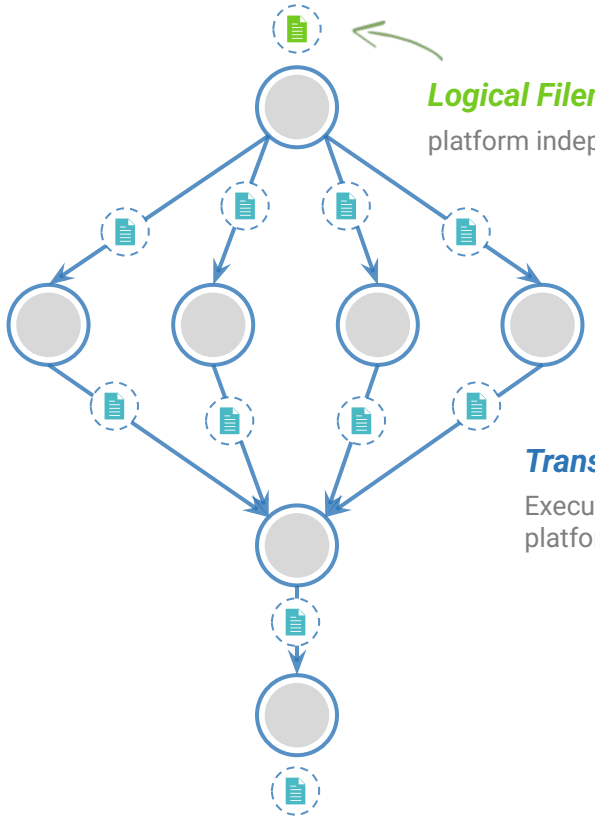
Input Workflow Specification YAML formatted

Output Workflow

Portable Description

Users do not worry about low level execution details

ABSTRACT WORKFLOW

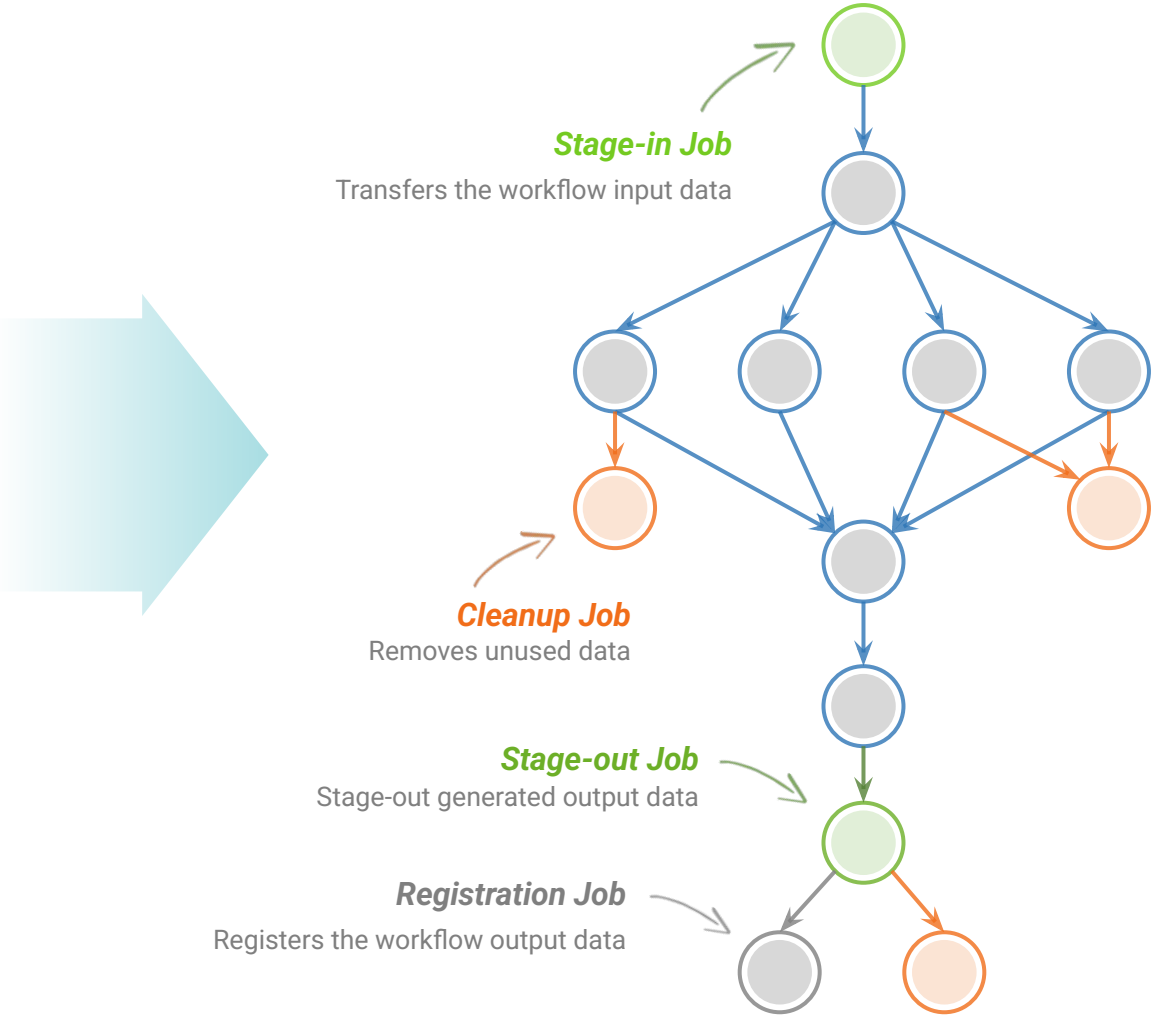


Logical Filename (LFN)

platform independent (abstraction)

Transformation

Executables (or programs)
platform independent



Stage-in Job

Transfers the workflow input data

Cleanup Job

Removes unused data

Stage-out Job

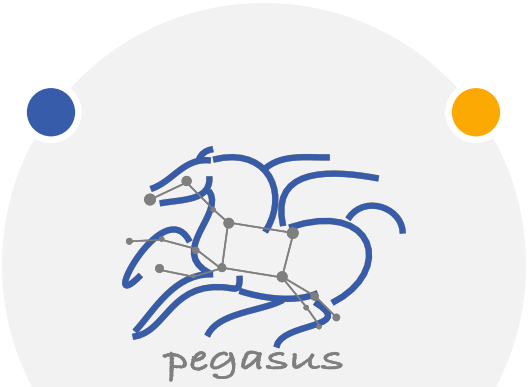
Stage-out generated output data

Registration Job

Registers the workflow output data

EXECUTABLE WORKFLOW

So, what other information does Pegasus need?



Site Catalog

Describes the sites where
The workflow jobs are to be executed

Transformation Catalog

Describes all of the executables
(called “transformations”)
used by the workflow

Replica Catalog

Describes all of the input data stored
on external servers

Hands on: API and Catalogs

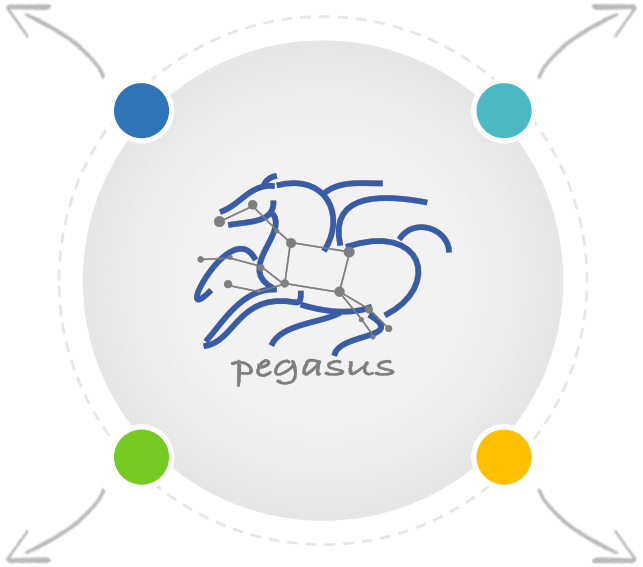
And if a job fails?

Postscript

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

Job Retry

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



Checkpoint Files

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

Rescue DAGs

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

command-line…

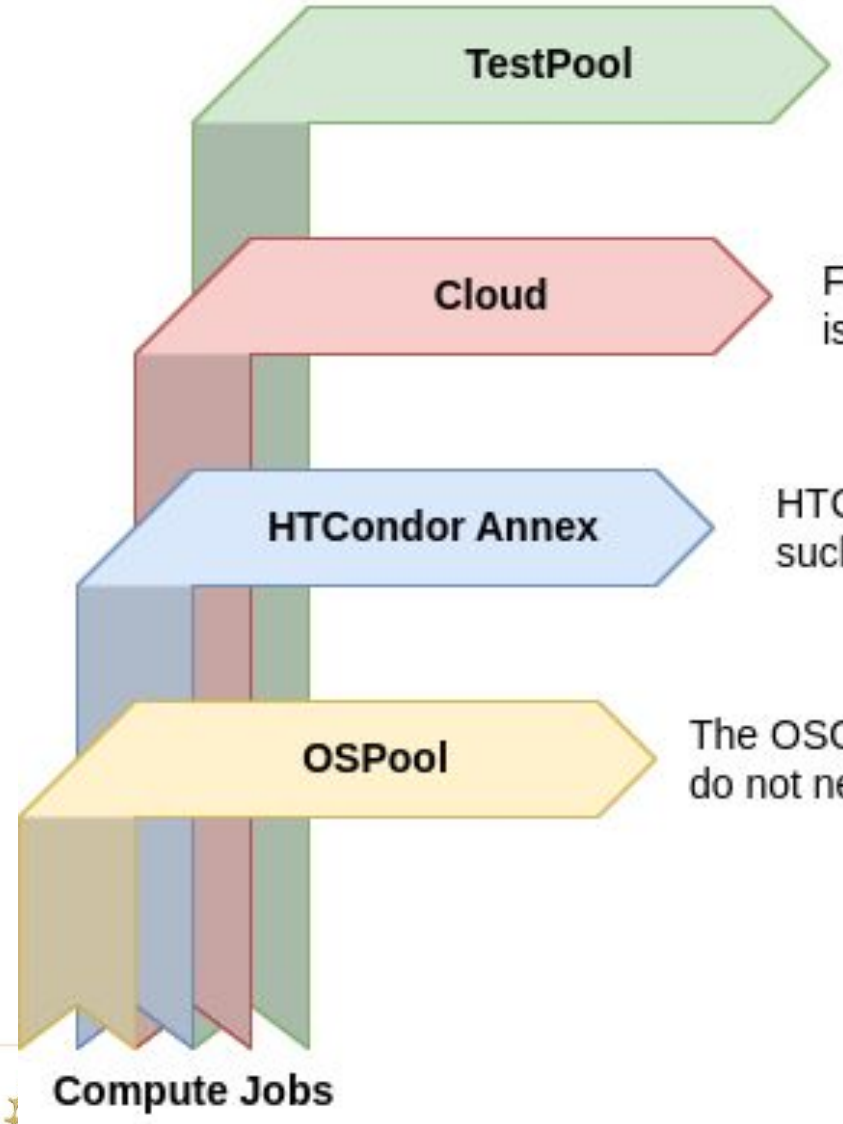
```
$ pegasus-status pegasus/examples/split/run0001
STAT IN_STATE JOB
Run 00:39 split-0 (/home/pegasus/examples/split/run0001)
Idle 00:03 └─split_ID0000001
Summary: 2 Condor jobs total (I:1 R:1)

UNRDY READY PRE IN_Q POST DONE FAIL %DONE STATE   DAGNAME
 14     0    0    1    0   2    0    11.8 Running *split-0.dag
```

```
$ pegasus-analyzer pegasus/examples/split/run0001
pegasus-analyzer: initializing...

****************************Summary***************************

Total jobs : 7 (100.00%)
# jobs succeeded : 7 (100.00%)
# jobs failed : 0 (0.00%)
# jobs unsubmitted : 0 (0.00%)
```

```
$ pegasus-statistics -s all pegasus/examples/split/run0001
------------------------------------------------------------------------------
Type           Succeeded Failed Incomplete Total Retries Total+Retries
Tasks              5       0        0        5     0         5
Jobs              17       0        0       17     0        17
Sub-Workflows      0       0        0        0     0         0
------------------------------------------------------------------------------

Workflow wall time : 2 mins, 6 secs
Workflow cumulative job wall time : 38 secs
Cumulative job wall time as seen from submit side : 42 secs
Workflow cumulative job badput wall time :
Cumulative job badput wall time as seen from submit side :
```

Provenance Data can be Summarized pegasus-statistics or Used for Debugging pegasus-analyzer

Hands on: Statistics and Debugging



The TestPool is a small resource which is always available. This can be used for development, debugging and running small workflows.

For cloud resources, currently IU Jetstream2, a custom image is provided. The image is started with a provided security token, which makes it available to your jobs.

HTCondor Annex enables you to provision compute resources on HPC clusters such as SDSC Expanse, PSC Bridges2, and Purdue Anvil.

The OSG Open Science Pool (OSPool) will run jobs in a distributed infrastructure. You do not need to provision these resources, but there are special rules and capabilities.

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)

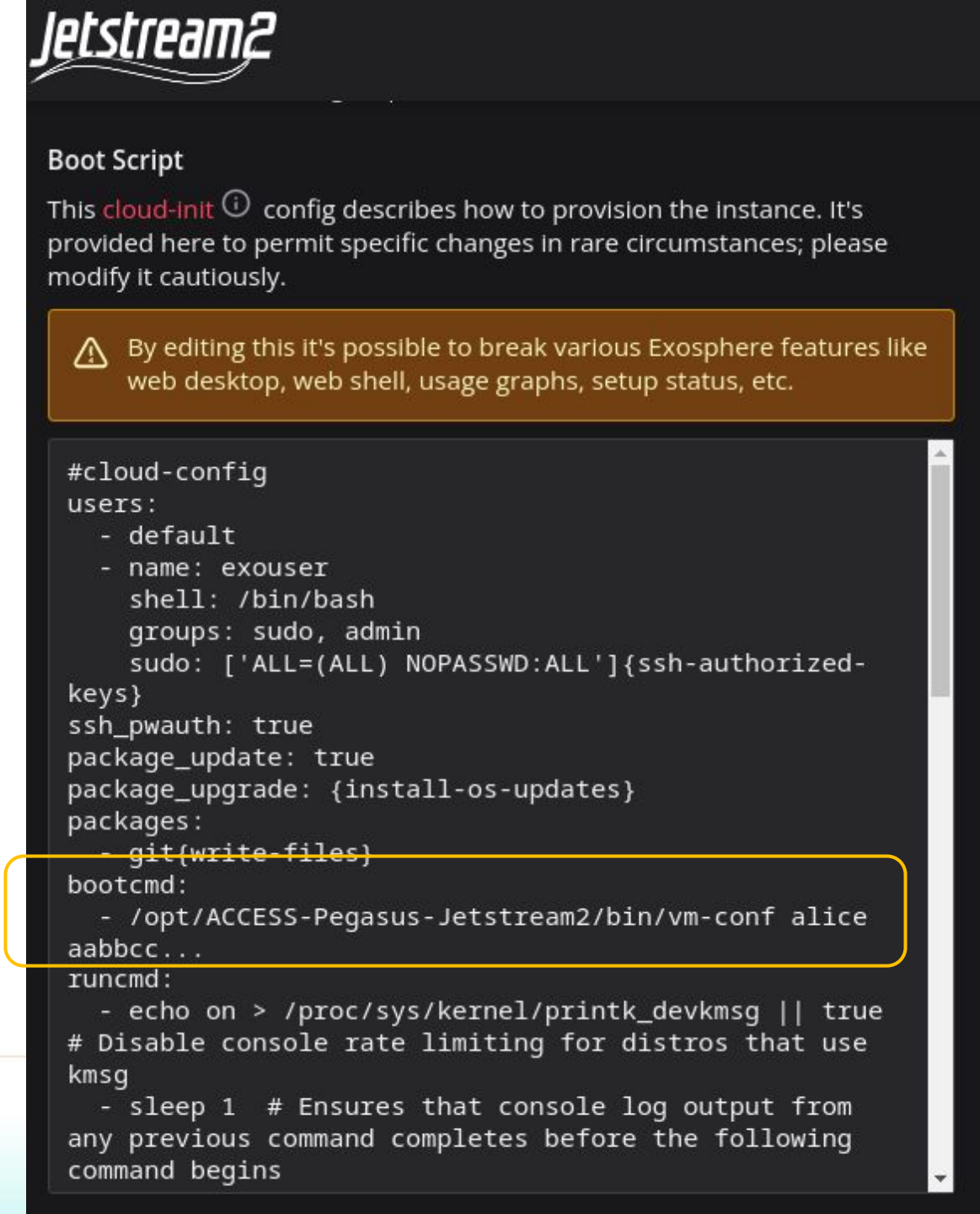
Bring Your Own Capacity

- 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

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



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

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

Hands on: Provisioning

Thank You!

https://pegasus.isi.edu

Pegasus Users Slack and mailing lists:
https://pegasus.isi.edu/contact/

E-mail Support:
pegasus-support@isi.edu

Office Hours:
https://pegasus.isi.edu/office-hours/

Additional slides

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

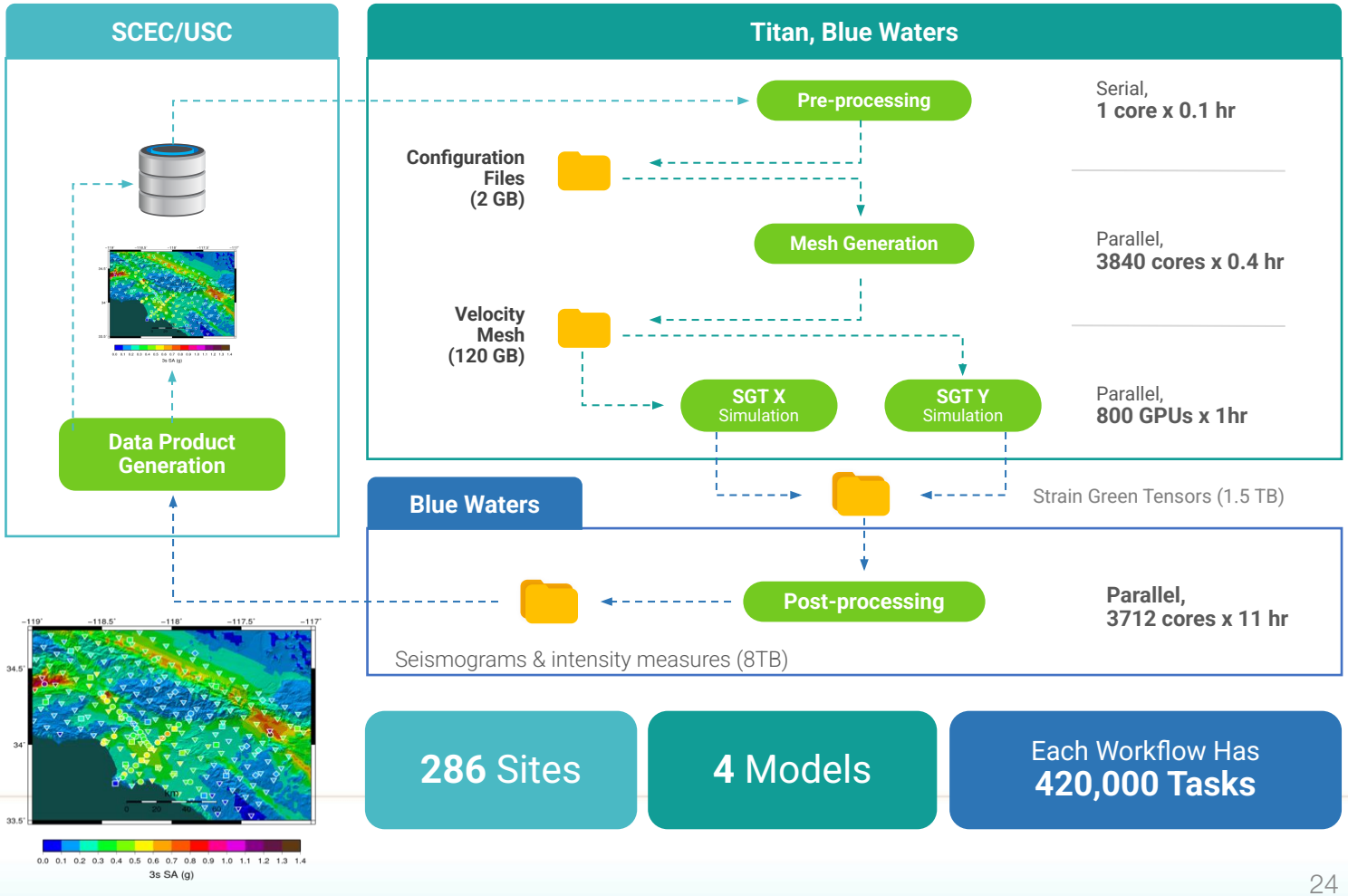
Blue Waters:
673,000 CPU node-hours, 329,000 GPU node-hours

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

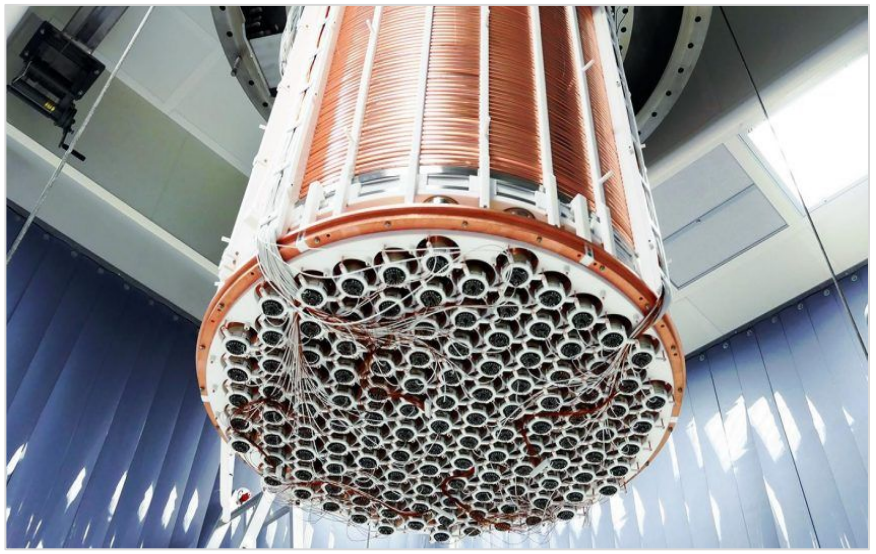
Each Workflow Has **420,000 Tasks**

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

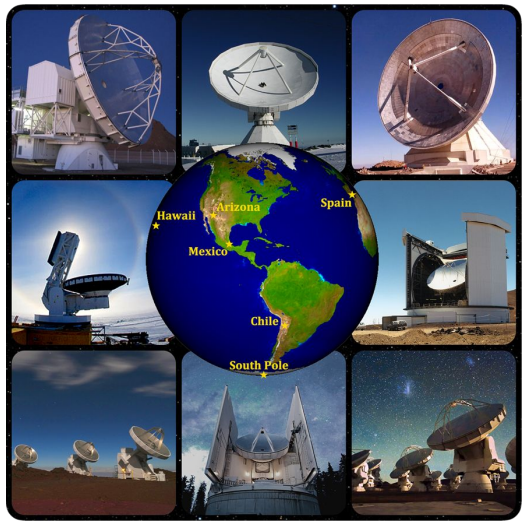
480,000 jobs - 2,600,000 core hours

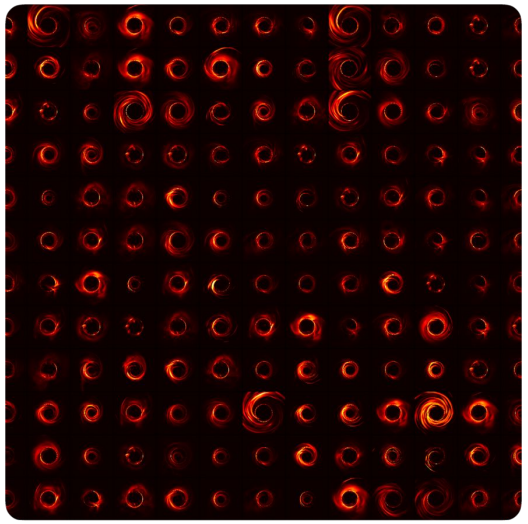
#15 in all OSG projects in last 6 months

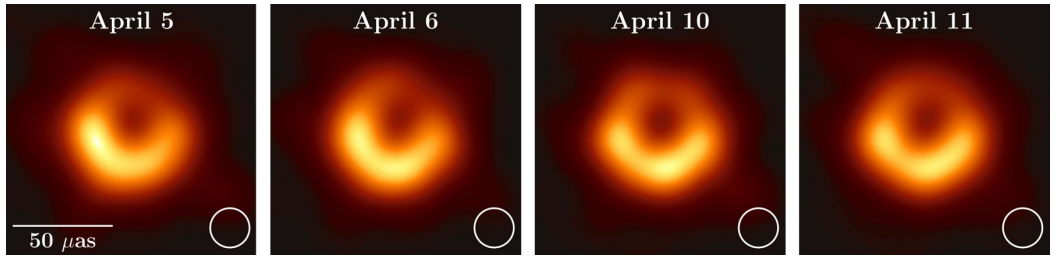
#2 in all OSG astronomy projects in the last 6 months

8 telescopes: 5 PB of data

60 simulations: 35 TB data





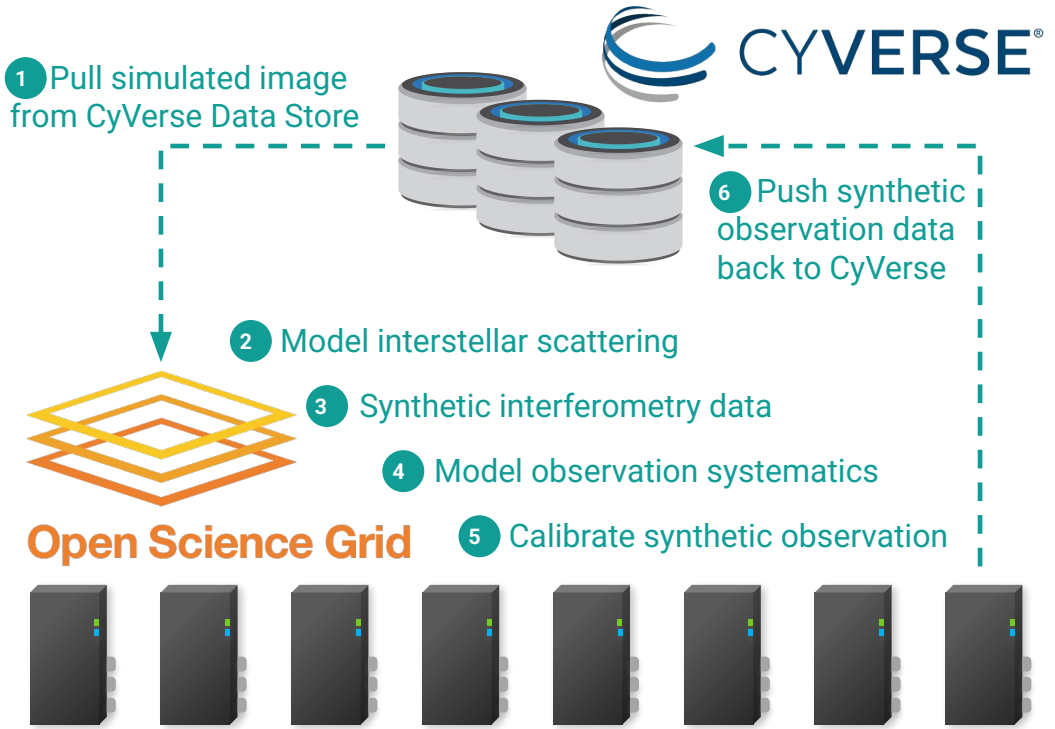


First images of black hole at the center of the M87 galaxy

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

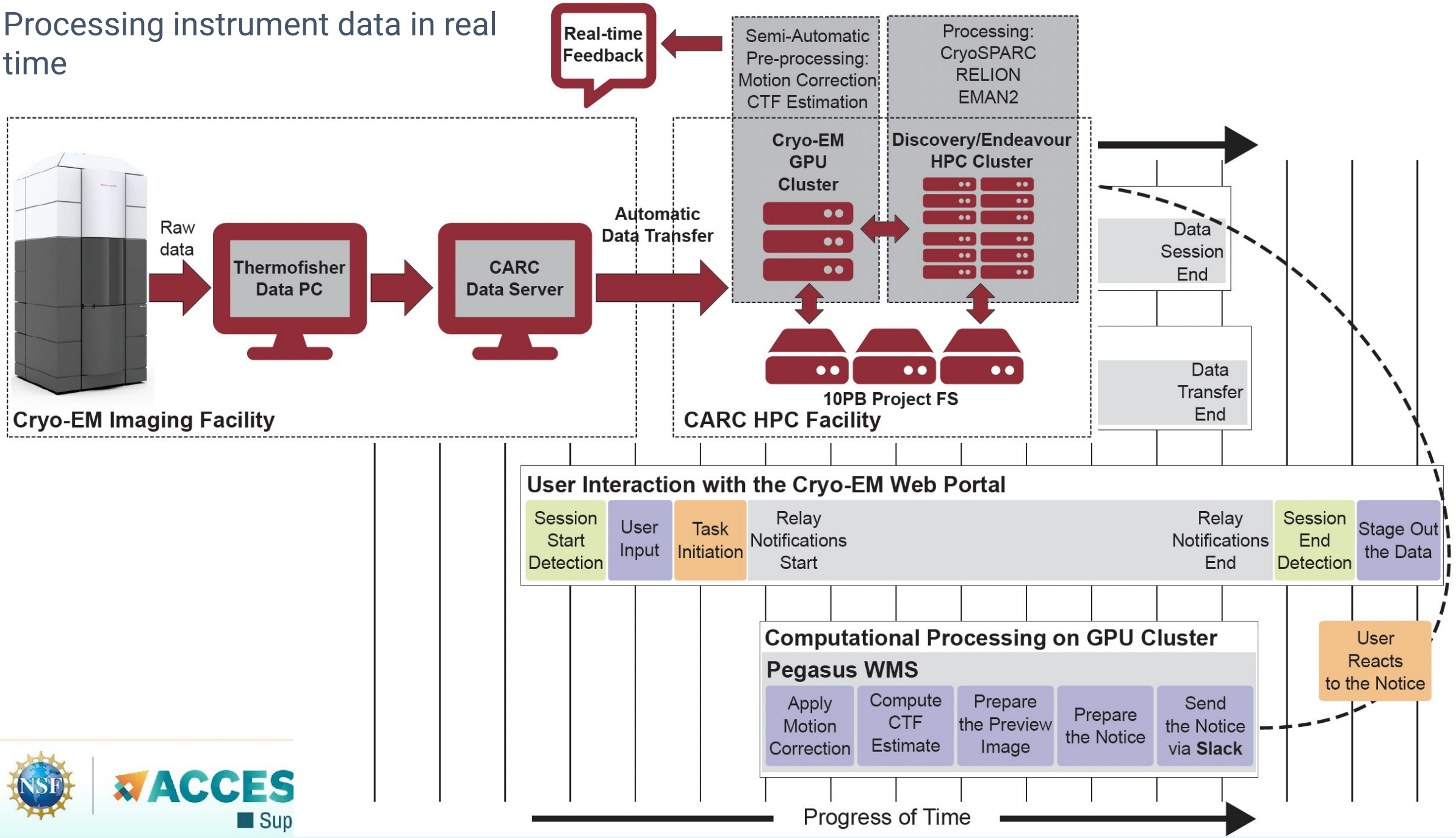
Pegasus-SYMBA Pipeline

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.



ACCESS Support

Processing instrument data in real time



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.

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., GridFTP to S3 = GridFTP 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
SCP
GridFTP
Globus
Online
iRods
Amazon S3
Google
Storage
SRM
FDT
Stashcp
Rucio
cp
ln -s
```

Pegasus Container Support

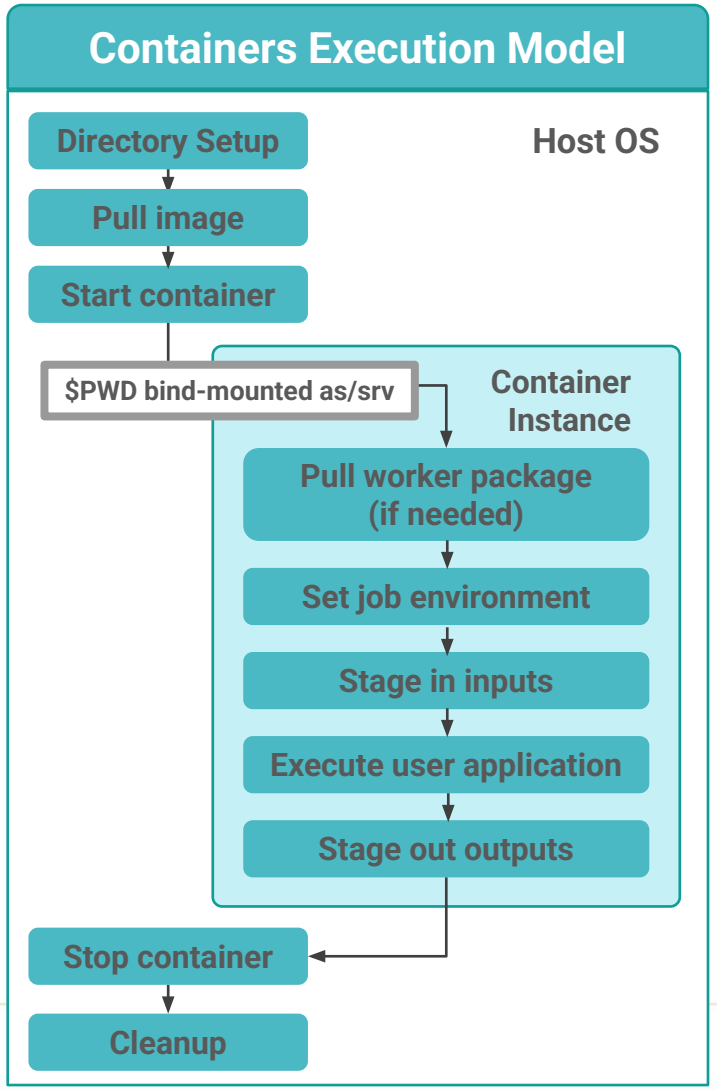
Users can refer to **containers** in the **Transformation Catalog** with their executable preinstalled

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

Data Staging Configurations

HTCondor I/O (HTCondor pools, OSG, …)

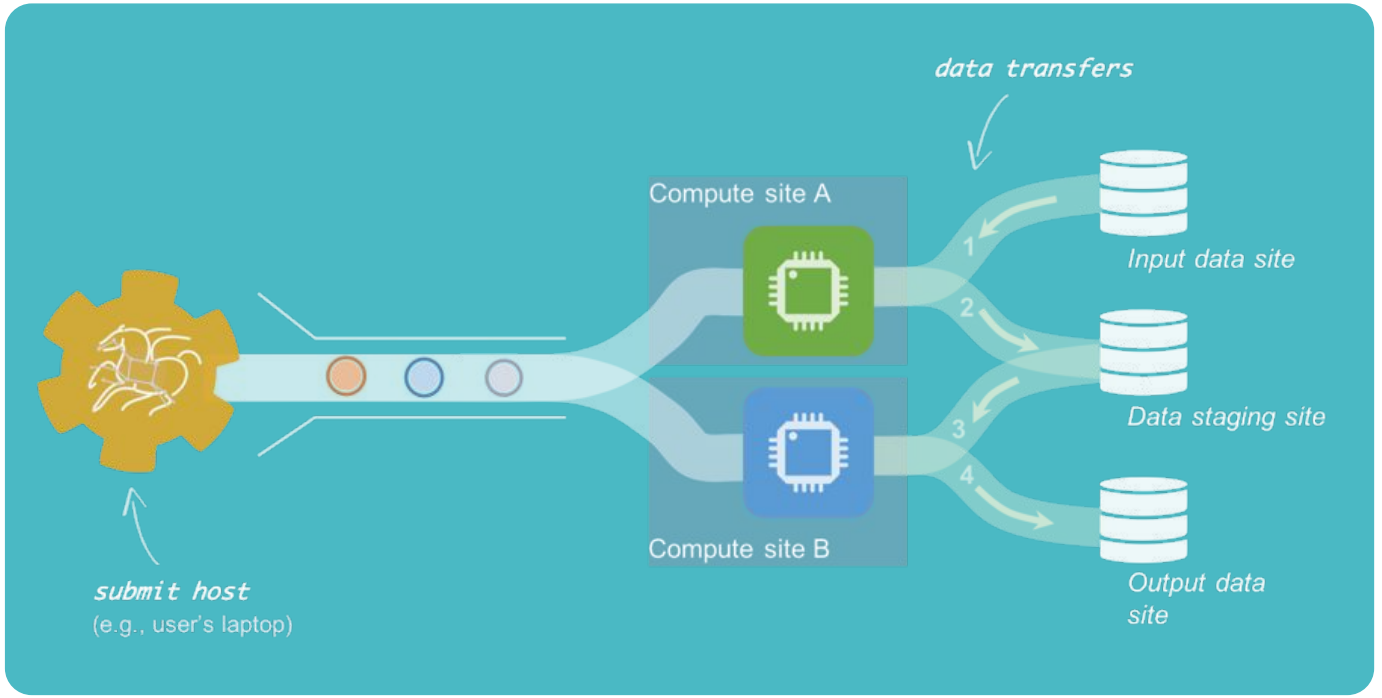
- Worker nodes do not share a file system
- Data is pulled from / pushed to the submit host via HTCondor file transfers
- Staging site is the submit host

Non-shared File System (clouds, OSG, …)

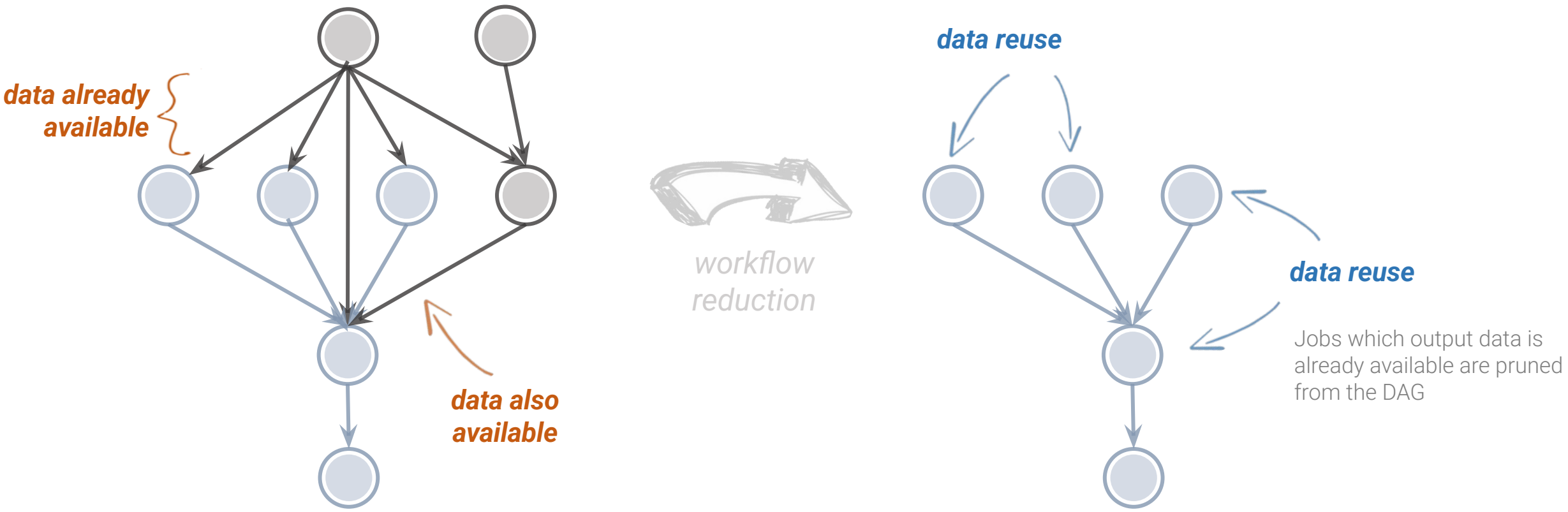
- Worker nodes do not share a file system
- Data is pulled / pushed from a staging site, possibly not co-located with the computation

Shared File System (HPC sites, XSEDE, Campus clusters, …)

- I/O is directly against the shared file system



Data Reuse prune jobs if output data already exists



Metadata

Can associate arbitrary key-value pairs with workflows, jobs, and files

Data Registration

Output files get tagged with metadata on registration in the workflow database

Static and Runtime Metadata

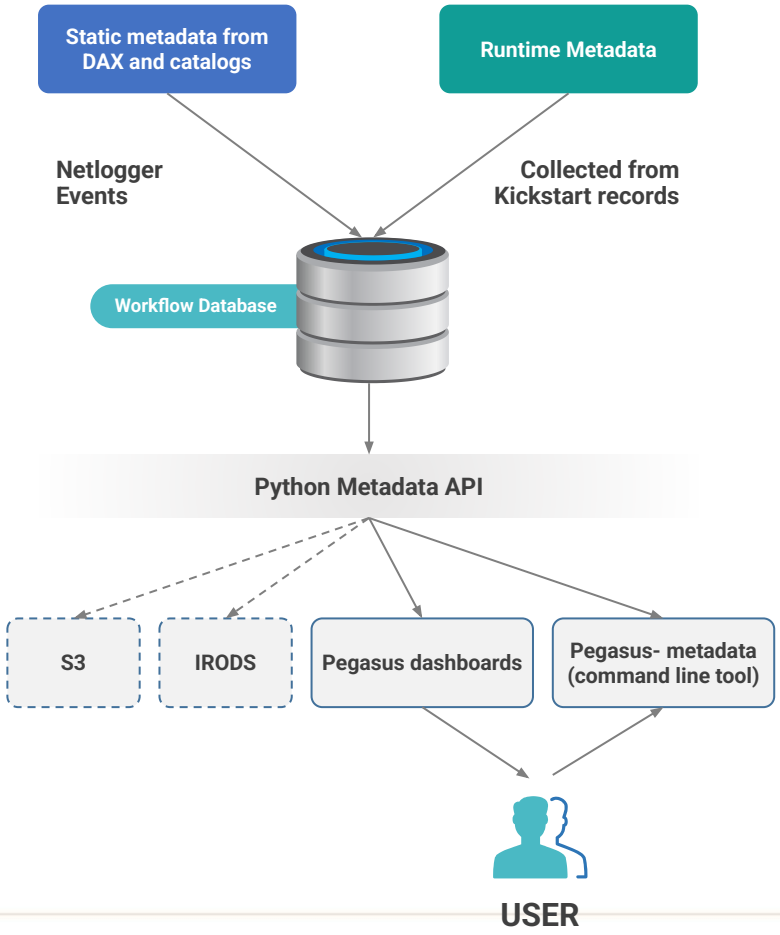
Static: application parameters
Runtime: performance metrics

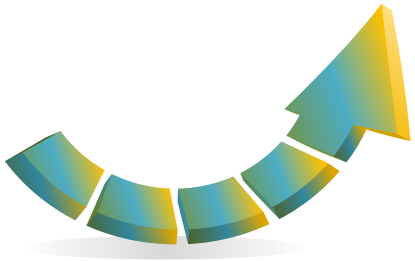
Workflow, Job, File

```
x-pegasus:
apiLang: python
createdBy: vahi
createdOn: 12-08-20T10:08:48Z
pegasus: "5.0"
name: diamond
metadata:
    experiment:"par_all27_prot_lipid"
jobs:
- type: "job"
  name: "namd"
  id: "ID0000001"
  arguments: ["equilibrate.conf"]
  uses:
    - lfn: "Q42.psf"
      metadata:
        type: "psf"
        charge: "42"
      type: "input"
    - lfn: "eq.restart.coord"
      type: "output"
      metadata:
        type: "coordinates"
      stageOut: true
      registerReplica: true
  metadata:
    timesteps:500000
    temperature:200
    pressure:1.01353
```

Select Data Based on Metadata

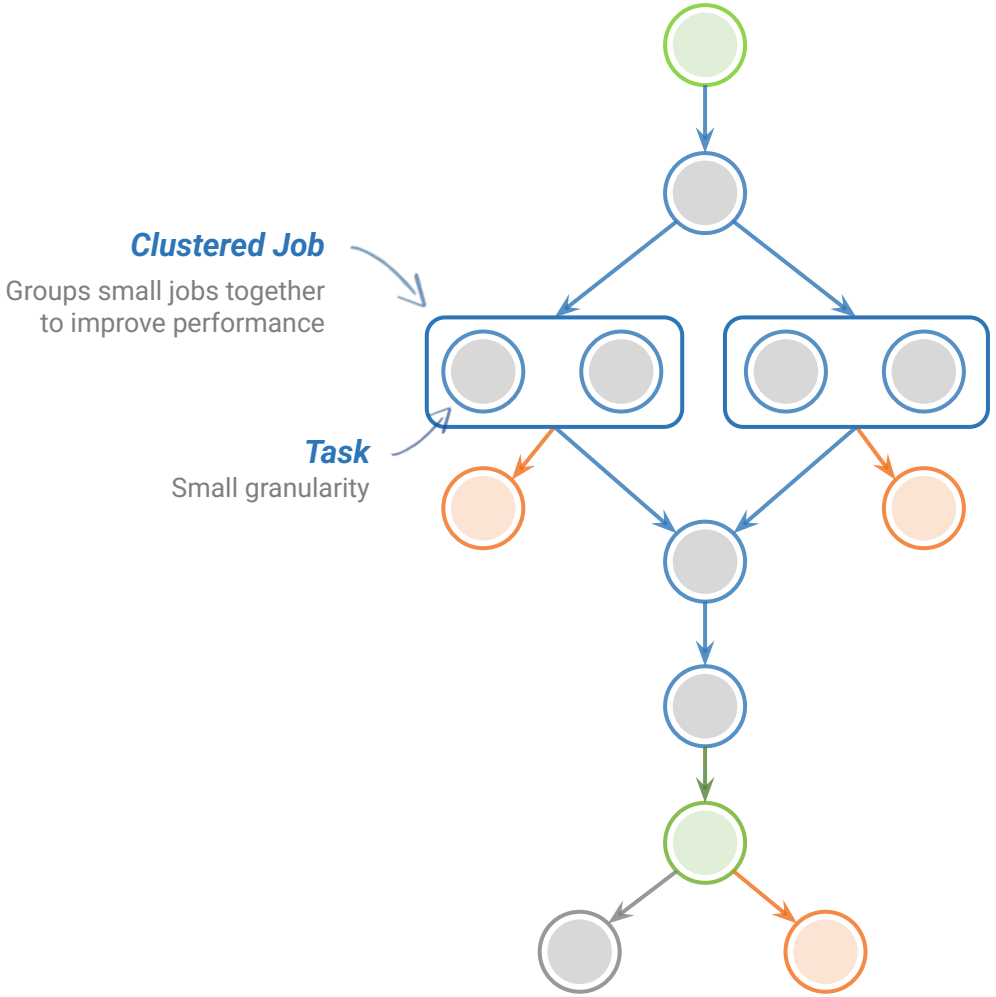
Register Data With Metadata





Performance.

Why not improve it?



Challenges to Scientific Data Integrity

Modern IT systems are not perfect
- errors creep in.

At modern **"Big Data"** sizes we are starting to see checksums breaking down.

Plus there is the threat of intentional changes:
malicious attackers, insider threats, etc.

User Perception: "Am I not already protected? I have heard about TCP checksums, encrypted transfers, checksum validation, RAID and erasure coding – is that not enough?"

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