Before we start

1. Grab the slide deck.
   a. Navigate to https://pegasus.isi.edu/documentation/tutorials/
   b. Under year 2025, click on the **AI Workflows on ACCESS Resources** tutorial.

1. Make sure you have an ACCESS account if you plan to do the hands on exercises
   a. An ACCESS account is required for the hands-on portion of the tutorial. If you already have an account, use that one.
   b. New account can be requested here. We strongly recommend going with the Option 1 “Register with an existing identity”.
   c. Note that the account does not have to be part of an allocation. Having an account is sufficient.
   d. Verify that you can log in to https://pegasus.access-ci.org

AI Workflows on ACCESS Resources

Karan Vahi, Mats Rynge
Information Sciences Institute
University of Southern California
vahi@isi.edu , rynge@isi.edu

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

Tutorial Goals

1. **Use a workflow system for your AI workloads**
2. **Scheduler overlay to simplify resource management**

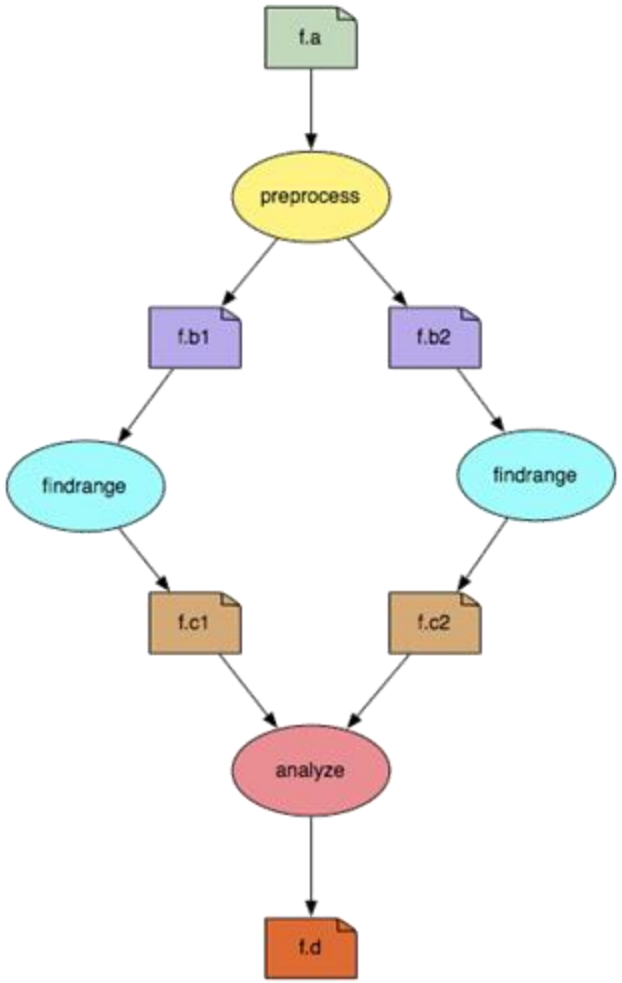
Why ACCESS Pegasus?

Pegasus can be deployed and used in a variety of different ways, such as cli/Jupyter, local install on your machine, a cluster or cloud. One option is a hosted system like the ACCESS Pegasus one we are using today.

Introduction

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*



Workflow Challenges Across Domains

- Describe complex workflows in a simple way
- Access distributed, heterogeneous data and resources (heterogeneous interfaces)
- Deal with resources/software that change over time
- Ease of use. Ability to debug and monitor large workflows

Our Focus

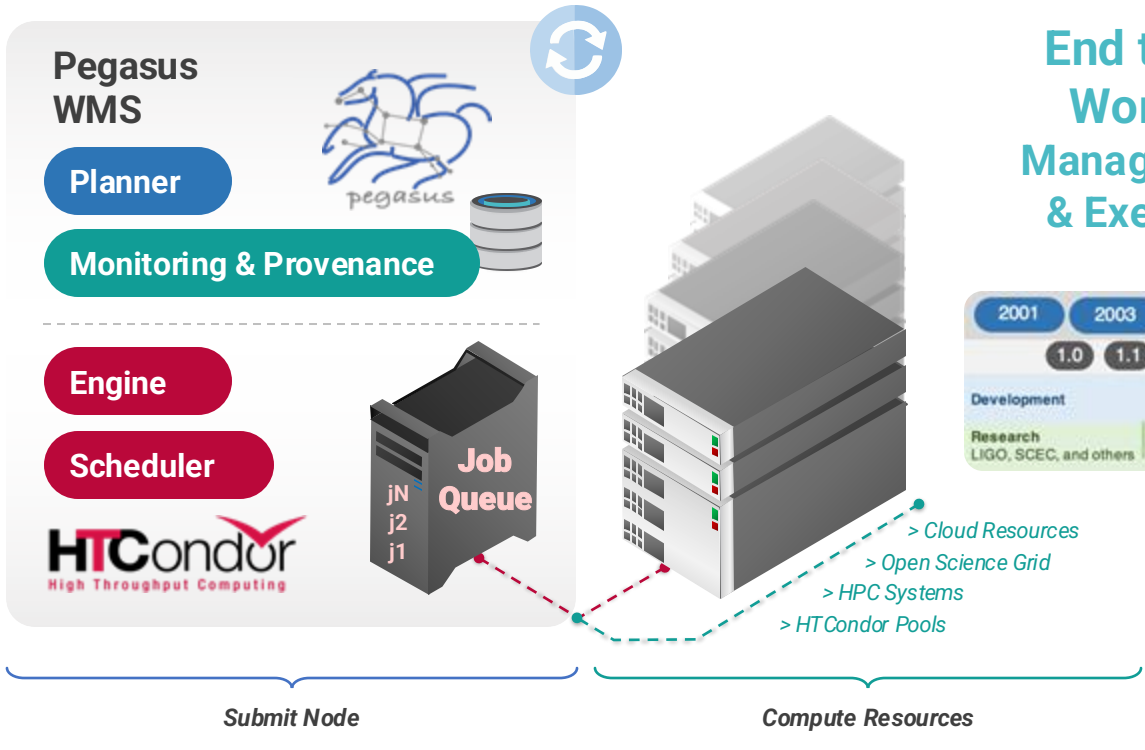
- Separation between workflow description and workflow execution
- Workflow planning and scheduling (scalability, performance)
- Task execution (monitoring, fault tolerance, debugging, web dashboard)
- Provide additional assurances that a scientific workflow is not accidentally or maliciously tampered with during its execution.

ACCESS Support

HTCondor High Throughput Computing

Pegasus is a NSF funded project since 2001, with close collaboration with HTCondor team

Pegasus Workflow Management System



End to End Workflow Management & Execution

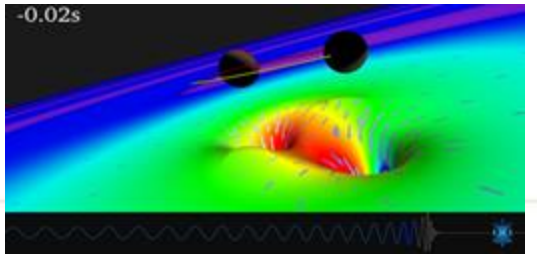
- Develop portable scientific workflows in Python, Java, and R
- Compile workflows to be run on heterogeneous resources
- Monitor and debug workflow execution via CLI and web-based tools
- Recover from failures with built-in fault tolerance mechanisms
- Regular release schedule incorporating latest research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

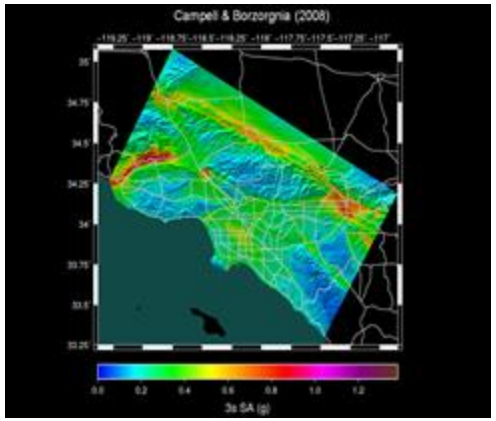
Pegasus in practice

- 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



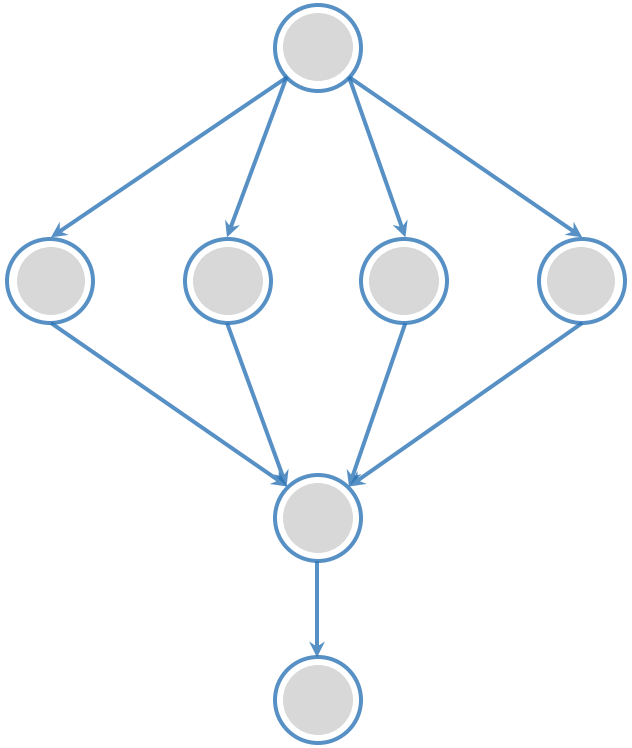


Hazard map indicating maximum amount of shaking at a particular geographic location generated from SCEC's CyberShake Pegasus workflow

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

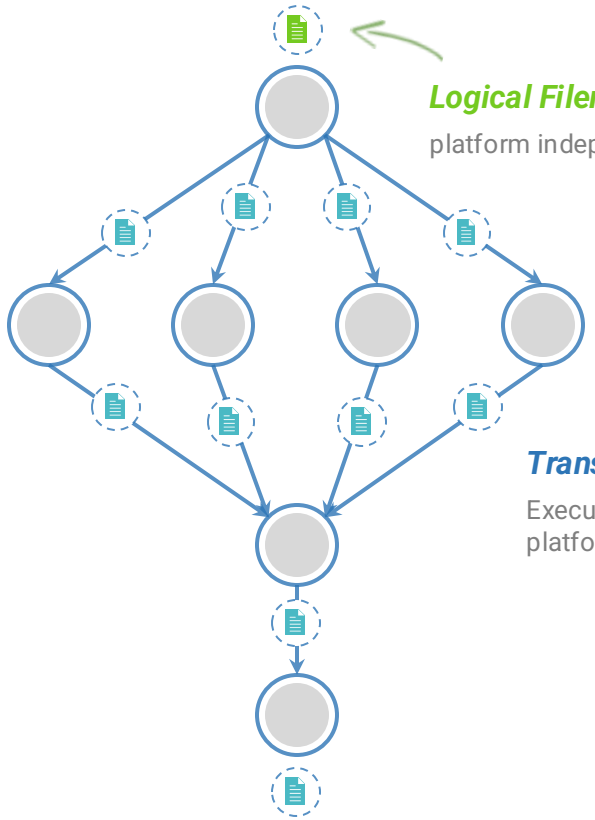
Input Workflow Specification YAML formatted

Output Workflow

Portable Description

Users do not worry about low level execution details

ABSTRACT WORKFLOW

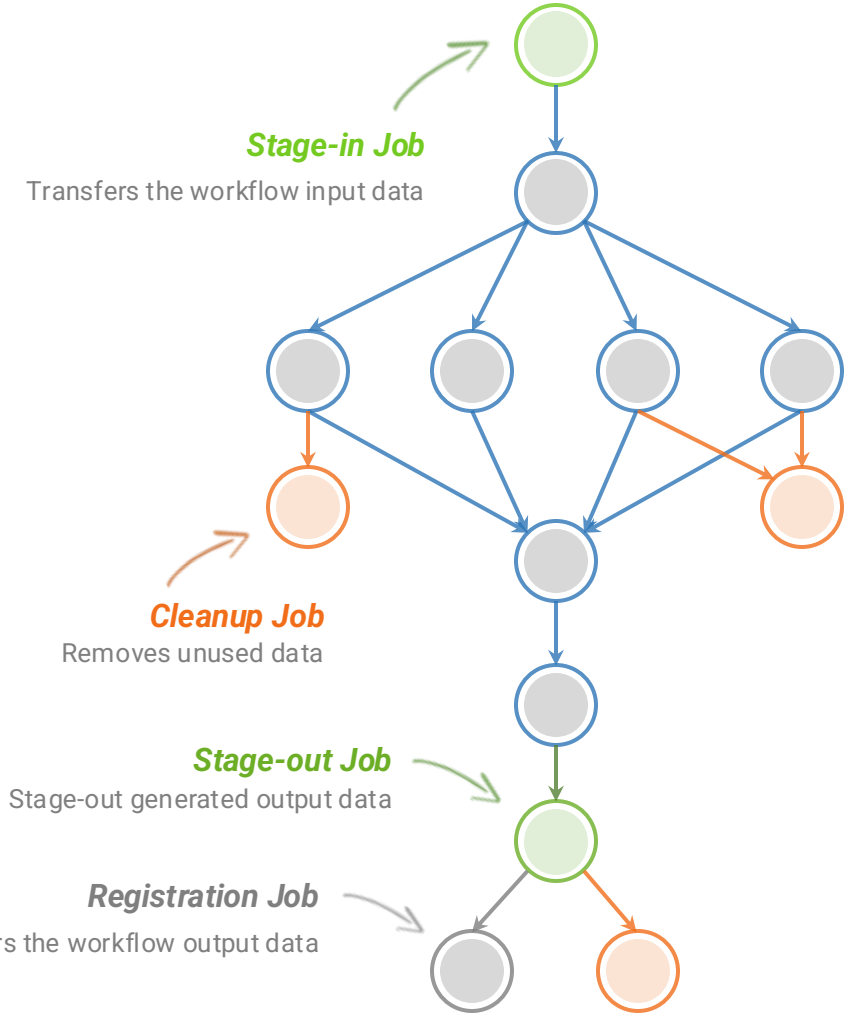


Logical Filename (LFN)

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

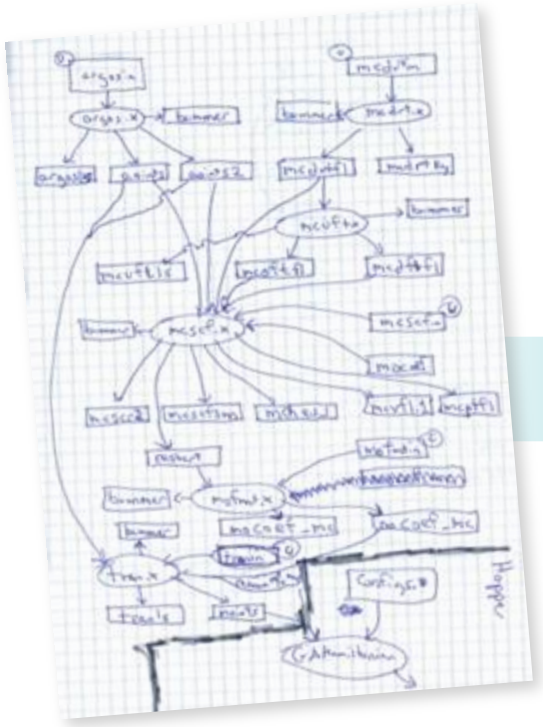
Stage-out generated output data

Registration Job

Registers the workflow output data

EXECUTABLE WORKFLOW

Pegasus provides APIs to generate the Abstract Workflow

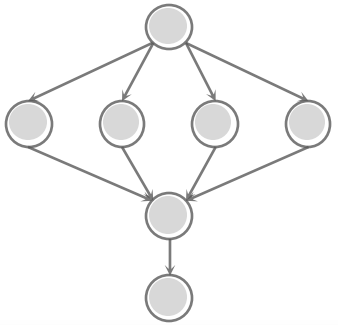


```python
#!/usr/bin/env python3

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



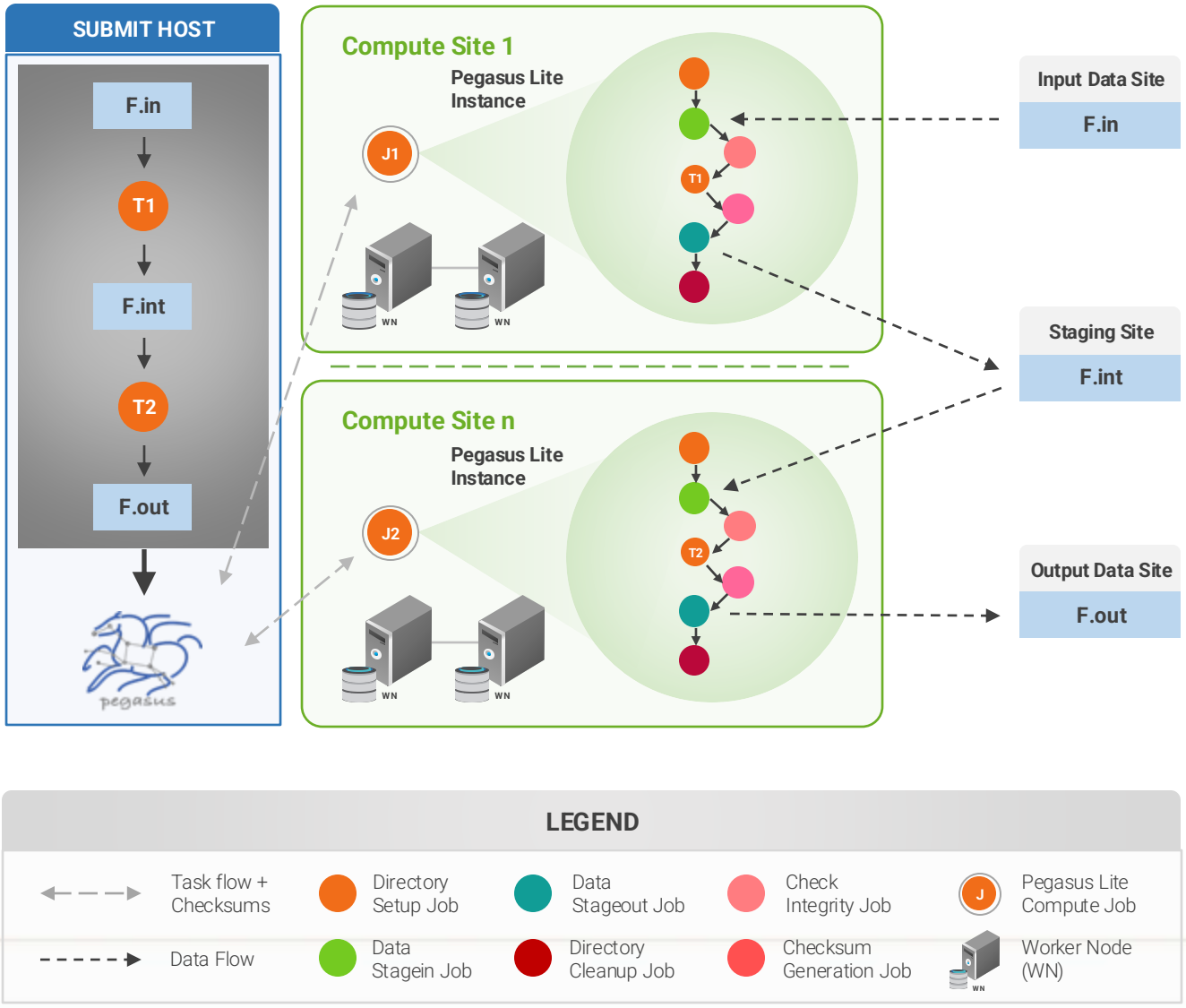
Abstract Workflow

```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

Pegasus Deployment

- **Workflow Submit Node**
  - Pegasus WMS
  - HTCondor
- **One or more Compute Sites**
  - Compute Clusters
  - Cloud
  - OSG
- **Input Sites**
  - Host Input Data
- **Data Staging Site**
  - Coordinate data movement for workflow
- **Output Site**
  - Where output data is placed



Pegasus-transfer

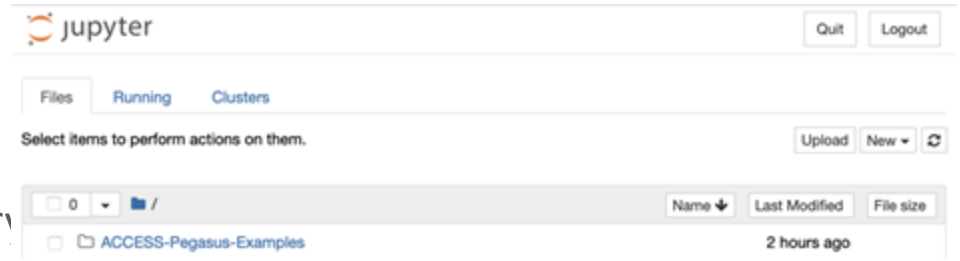
Pegasus' internal data transfer tool with support for a number of different protocols

- **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
```

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

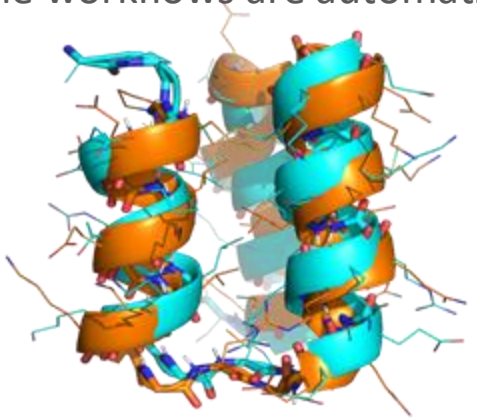
If you do not already have an ACCESS account: https://operations.access-ci.org/identity/new-user

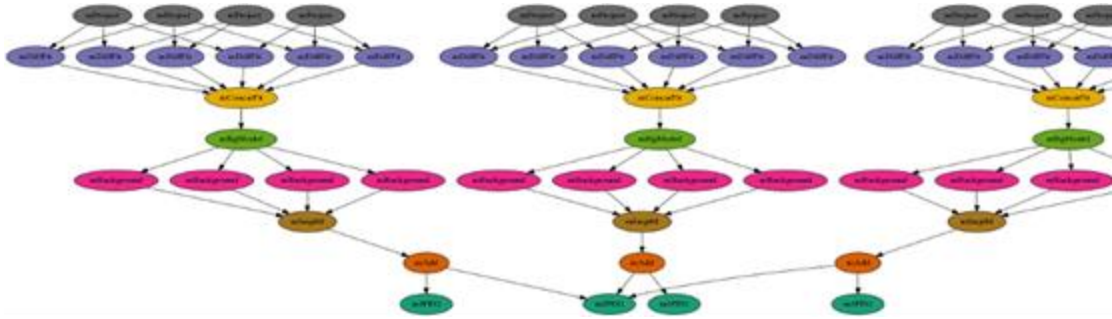
Using your ACCESS account, verify that you can use a web browser to log into https://pegasus.access-ci.org

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
  - VariantCalling





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.

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.

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

Tutorial: LLM - RAG

Goal: Execute a set of LLMs, locally on NAIRR resources, and use RAG to augment the models and answer a set of predefined prompts.

Container includes full LLM and toolchain (Mistral, LangChain, Chroma)

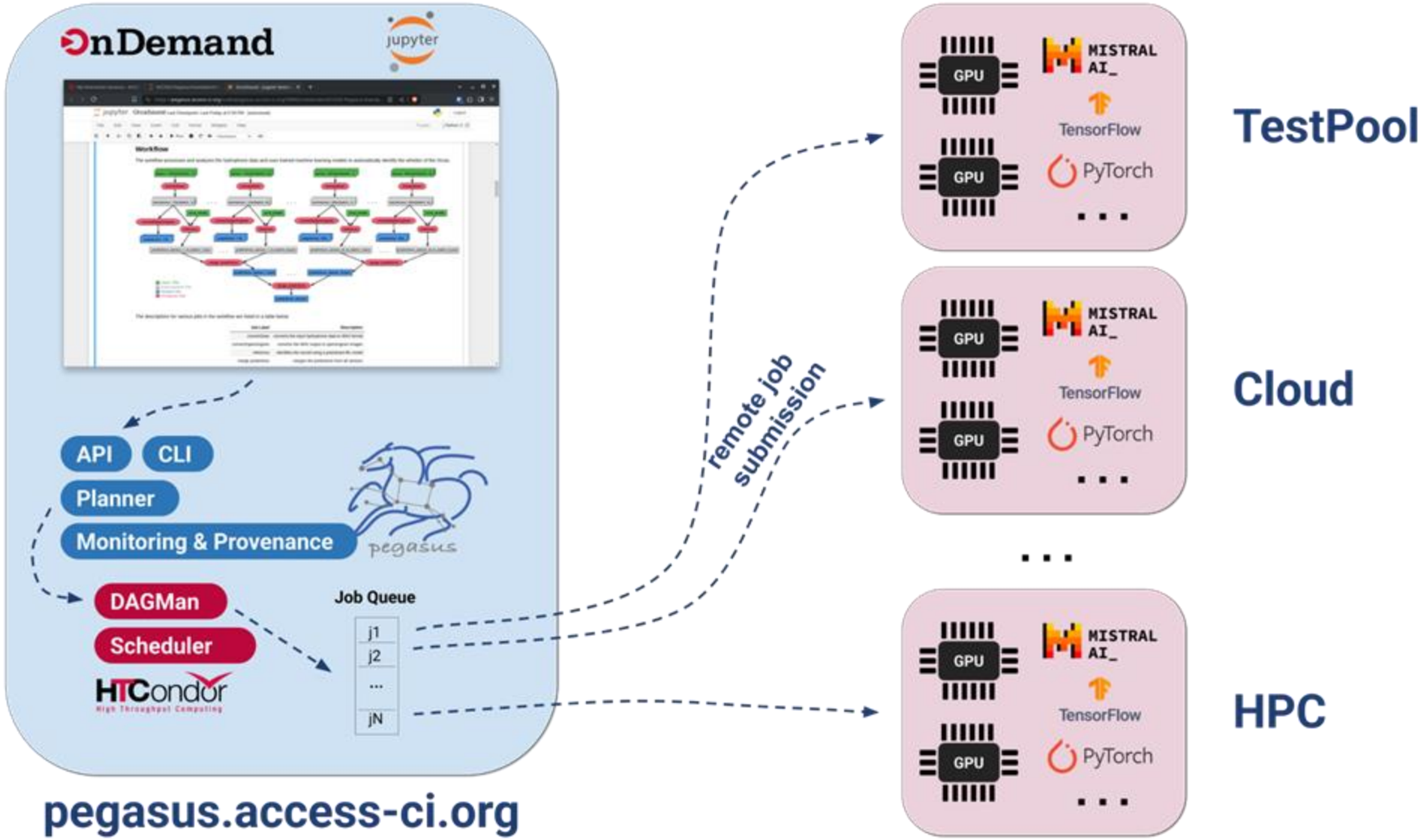
Workflow Input: Public domain books (Project Gutenberg)

Prompts:

- Please provide a one paragraph summary of the book.
- Who is the protagonist in the book?
- Who is the antagonist in the book?
- What time period is the book set in?

Can execute on GPUS from any of the capacity providers: TestPool, Cloud, HTCondor Annex, OSPool

Chroma


OnDemand
jupyter
API
CLI
Planner
Monitoring & Provenance
pegasus
DAGMan
Scheduler
HTCondor
High Throughput Computing
Job Queue
j1
j2
...
jN
remote job submission
GPU
GPU
MISTRAL AI_
TensorFlow
PyTorch
TestPool
Cloud
HPC
pegasus.access-ci.org

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.

Using your ACCESS account, verify that you can use a web browser to log into:

https://pegasus.access-ci.org

Hands on: Running our first workflow

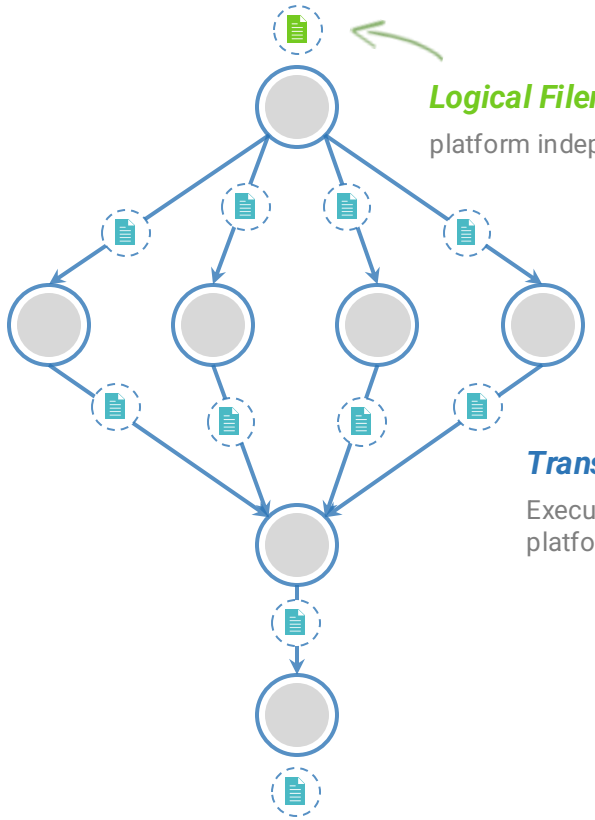
Input Workflow Specification YAML formatted

Output Workflow

Portable Description

Users do not worry about low level execution details

ABSTRACT WORKFLOW

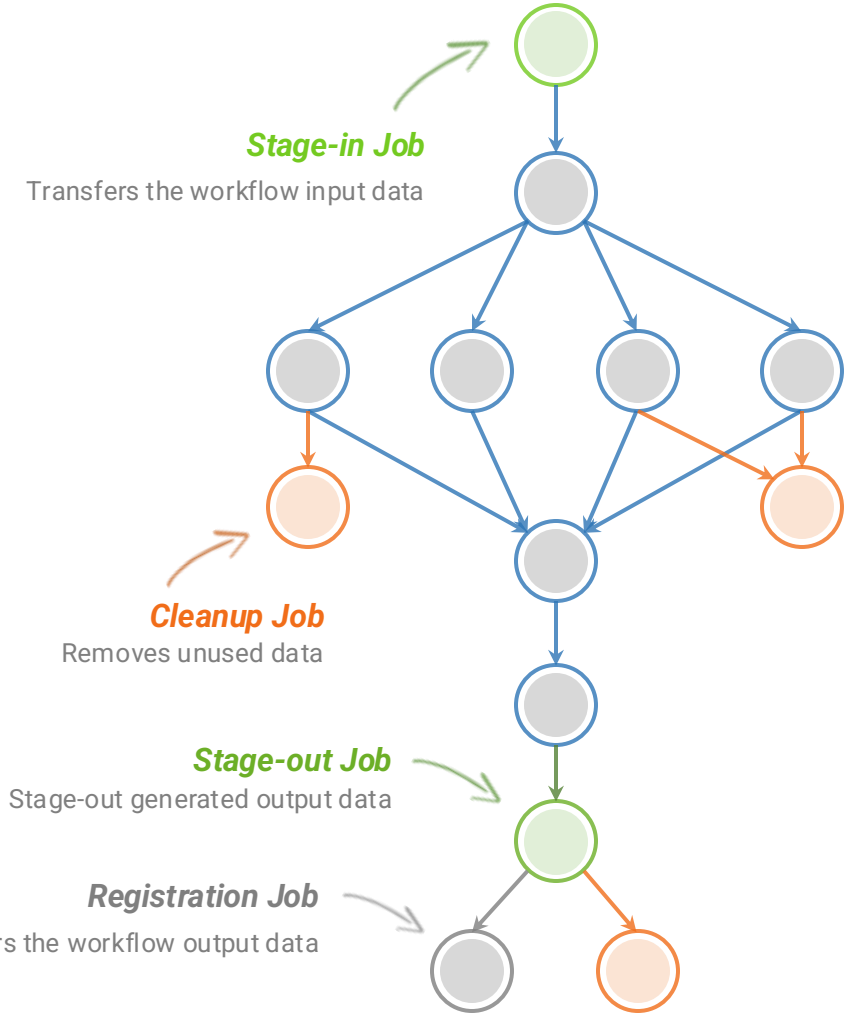


Logical Filename (LFN)

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: Pegasus Catalogs

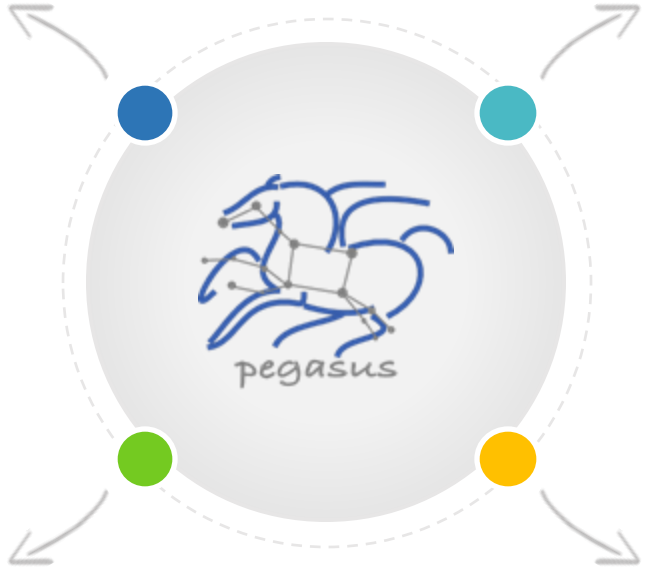
And if a job fails?

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: Generating statistics about your workflow

Resources Overlay

Advantages of Glidein (Pilot) Jobs in HTCondor

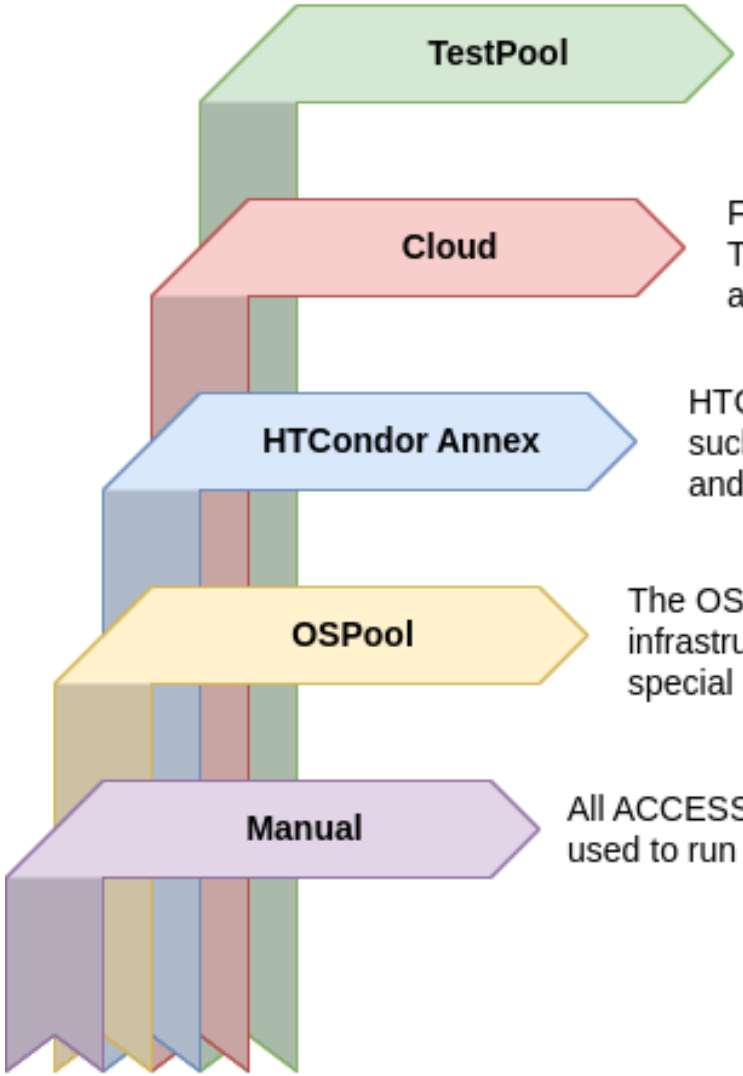
A glidein (pilot) job in HTCondor is a placeholder job that starts on a remote resource, sets up the HTCondor environment, and transforms the resource into a temporary worker node ready to run user jobs from the central HTCondor pool.

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.



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 clusters such as NCSA Delta, SDSC Expanse, PSC Bridges2, Purdue Anvil and PATh Facility

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.

All ACCESS Pegasus users are automatically issued a token which can be used to run glideins on any resource.

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

Jetstream2

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
```

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

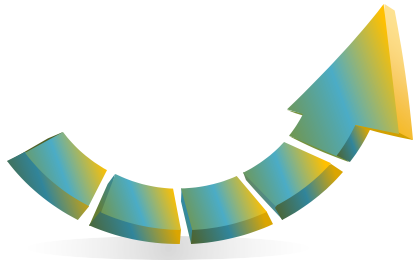
pegasus-glidein

- 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"'
```

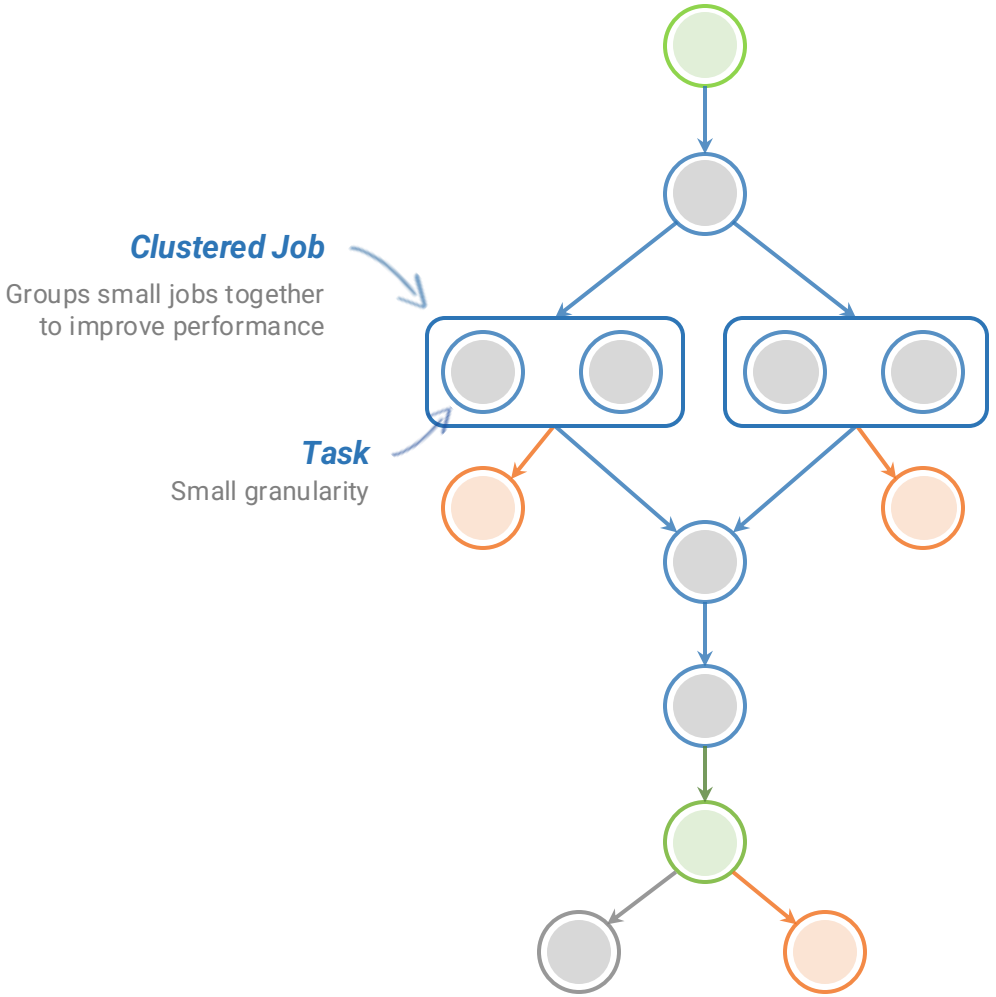
Optional Hands On: Resource Provisioning

Pegasus Features

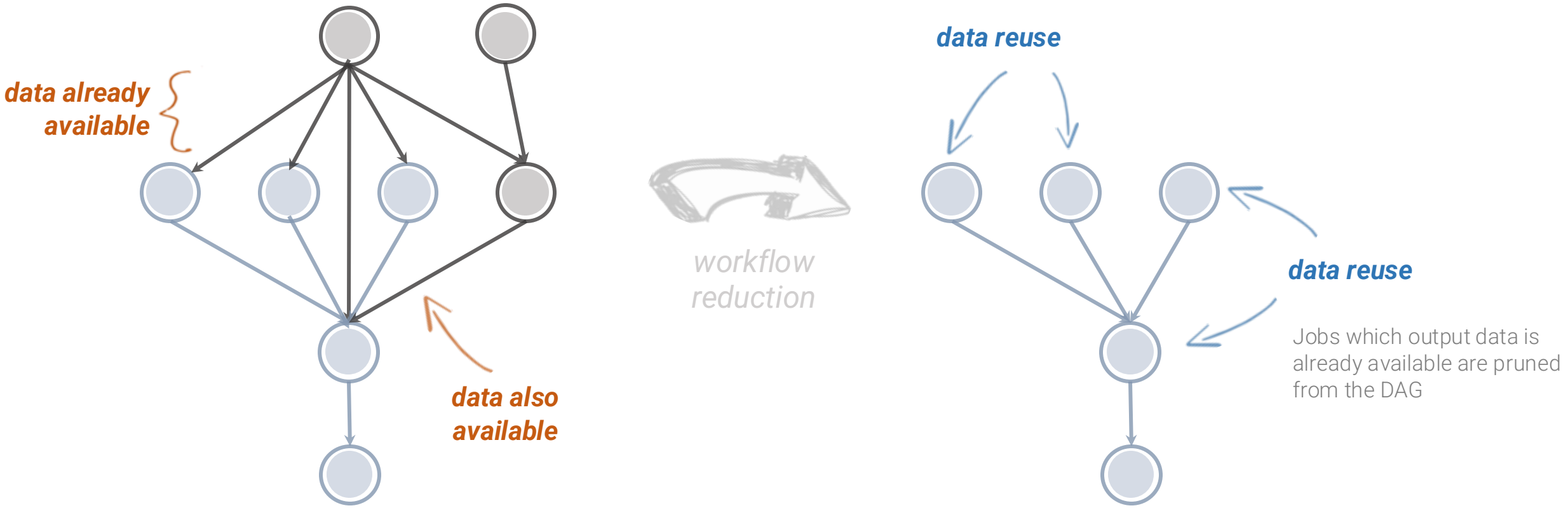


Performance.

Why not improve it?



Data Reuse prune jobs if output data already exists

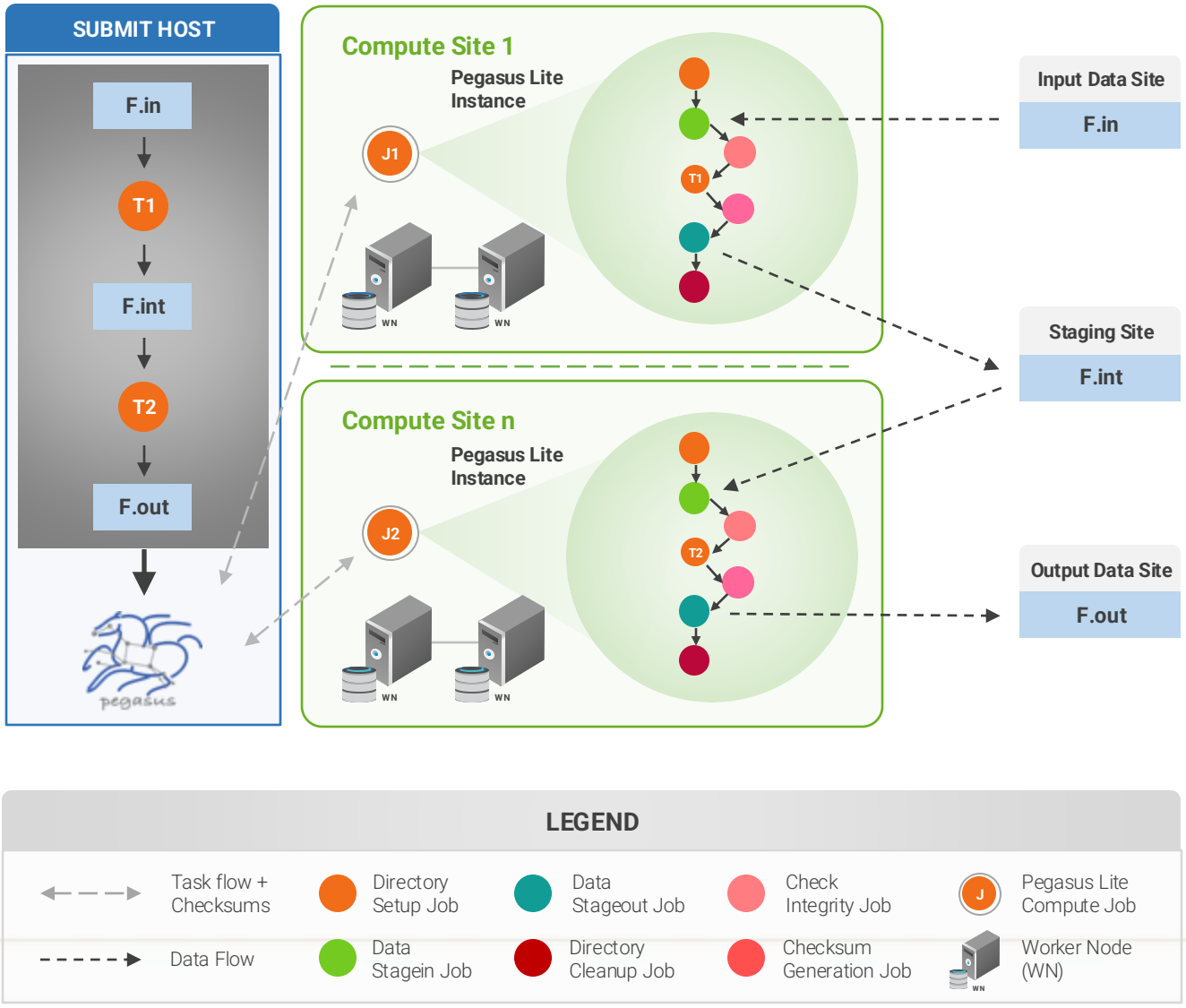


Automatic Integrity Checking

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



Pegasus Container Support

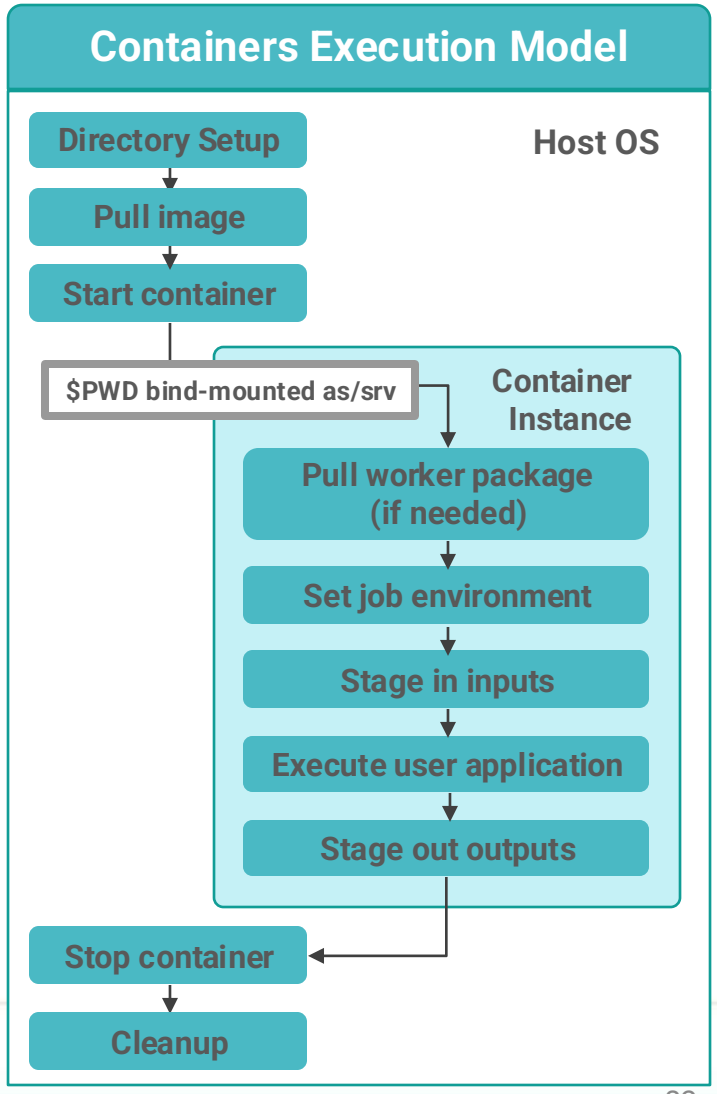
Users can refer to **containers** in the **Transformation Catalog** with their executable preinstalled

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

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

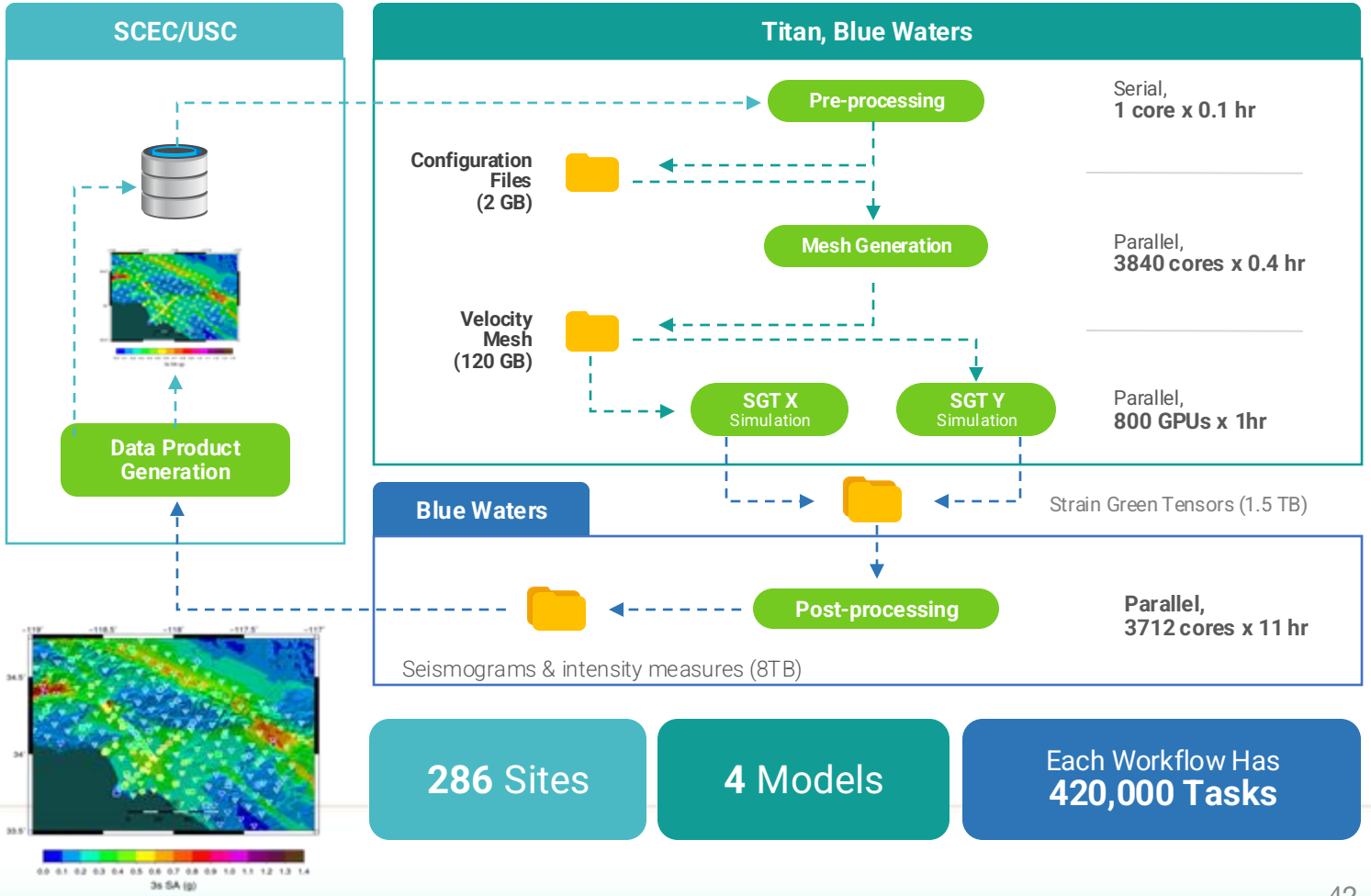
Blue Waters:
673,000 CPU node-hours, 329,000 GPU node-hours

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

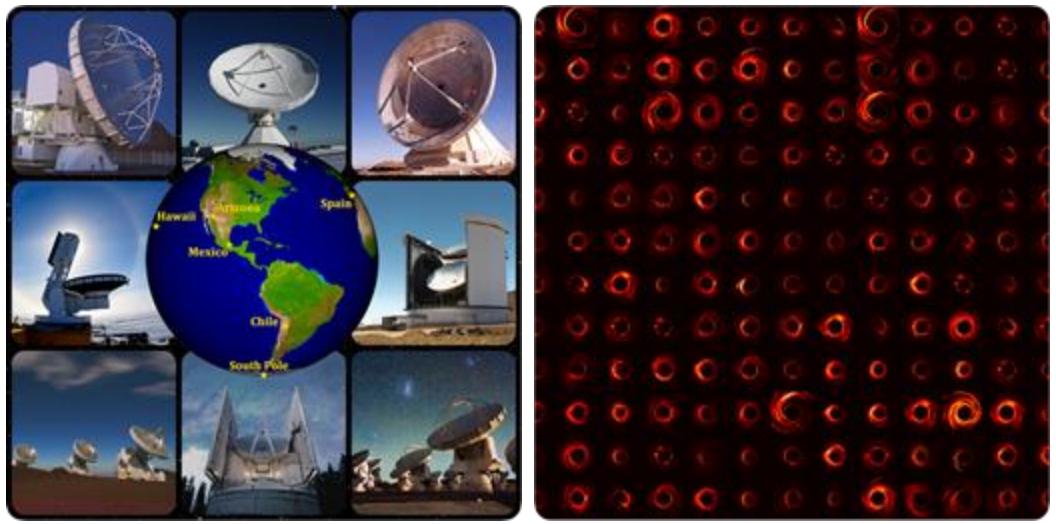
480,000 jobs - 2,600,000 core hours

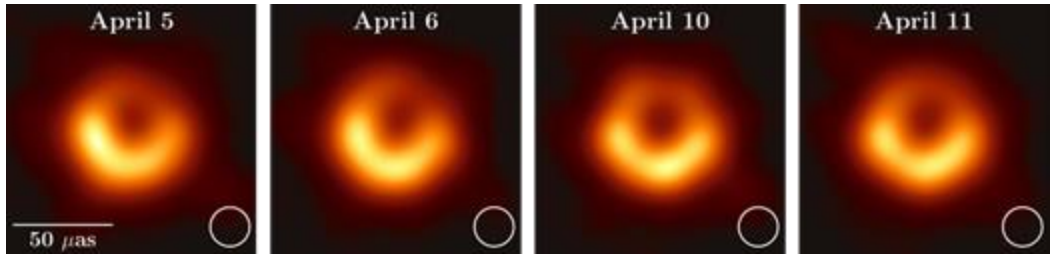
#15 in all OSG projects in last 6 months

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

8 telescopes: 5 PB of data

60 simulations: 35 TB data



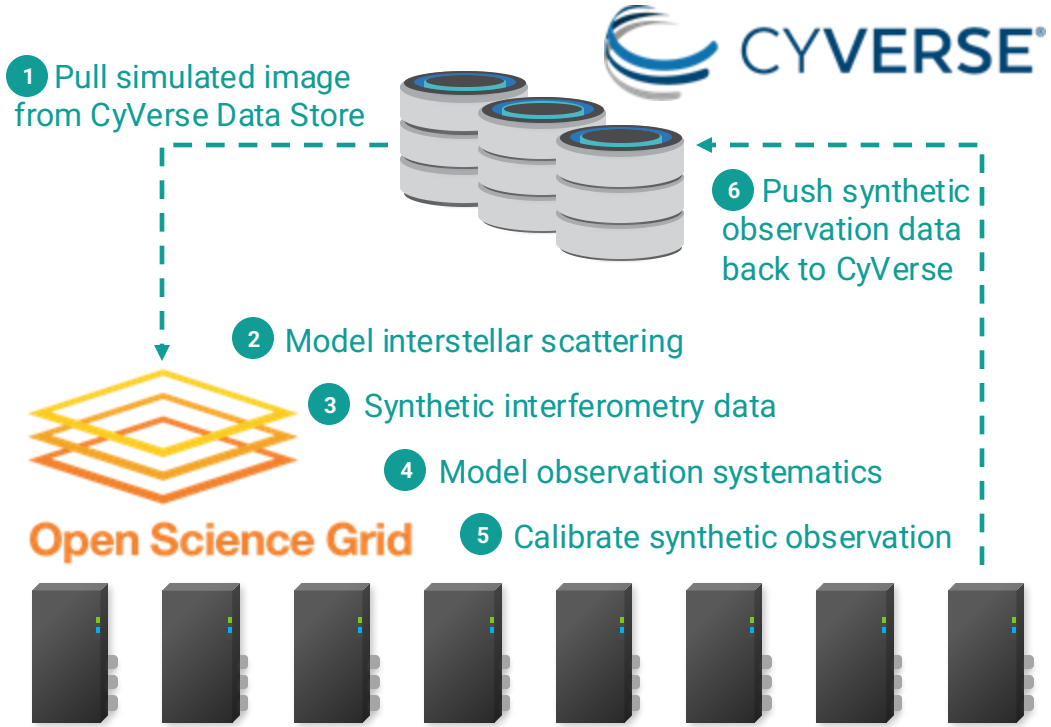


First images of black hole at the center of the M87 galaxy

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

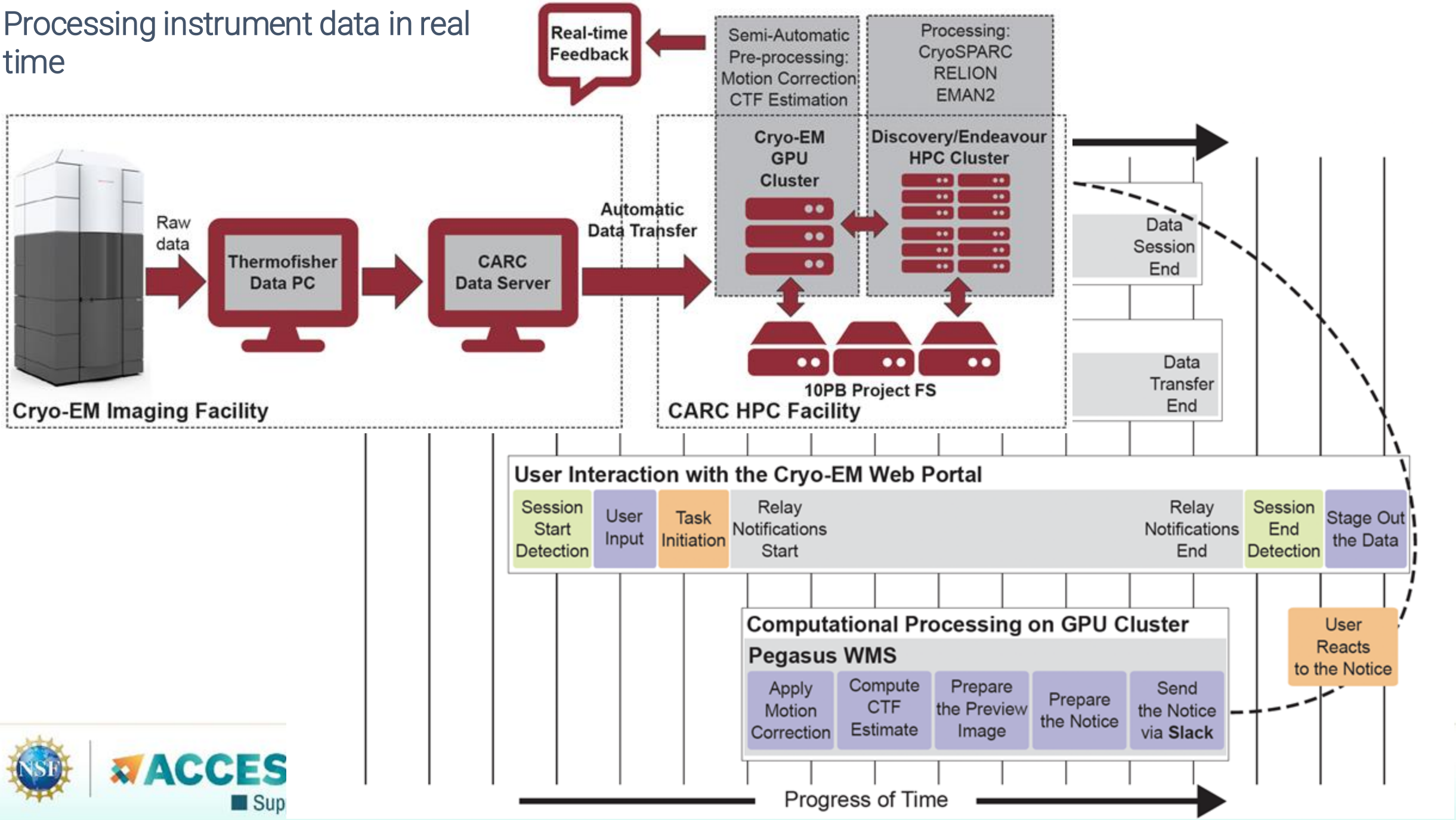
Pegasus-SYMBA Pipeline

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



Optional Hands On: API

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/