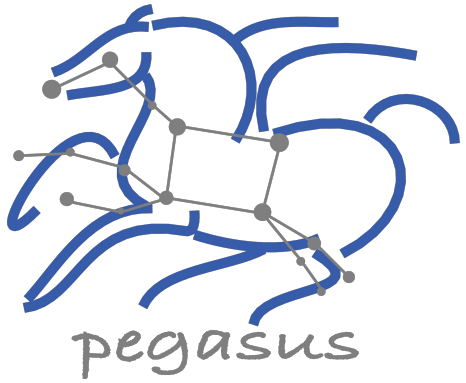
https://pegasus.isi.edu



Scientific Data Processing with the Pegasus Workflow Management System

Karan Vahi, Mats Rynge

Information Sciences Institute
University of Southern California, School of Engineering
vahi@isi.edu , rynge@isi.edu

USC | Advanced Research Computing
Enabling scientific breakthroughs at scale

ACCESS | Advancing Innovation

NSF U.S. DEPARTMENT OF ENERGY NIH DARPA

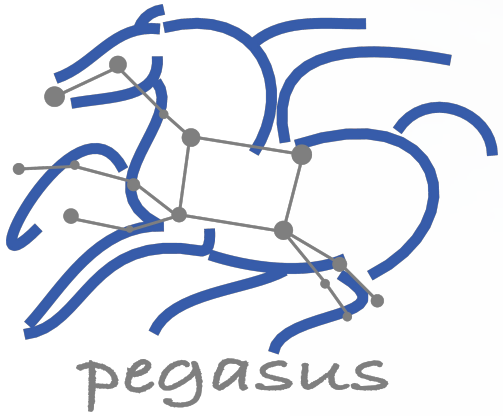
Preparations and Outline

- To do the online training exercises you need an ACCESS/XSEDE login. You can create one using your academic institution credentials.
  - https://adass2023.lpl.arizona.edu/sites/adass2023.lpl.arizona.edu/files/documents/ADASS23-Instructions-for-Pegasus-Tutorial-v2.pdf (short url: https://shorturl.at/FJL06)
- Use a web browser and log on to ACCESS Pegasus at https://access.pegasus.isi.edu

  Backup (If problem with ACCESS Accounts): https://tinyurl.com/pegasus-adass23-backup

---

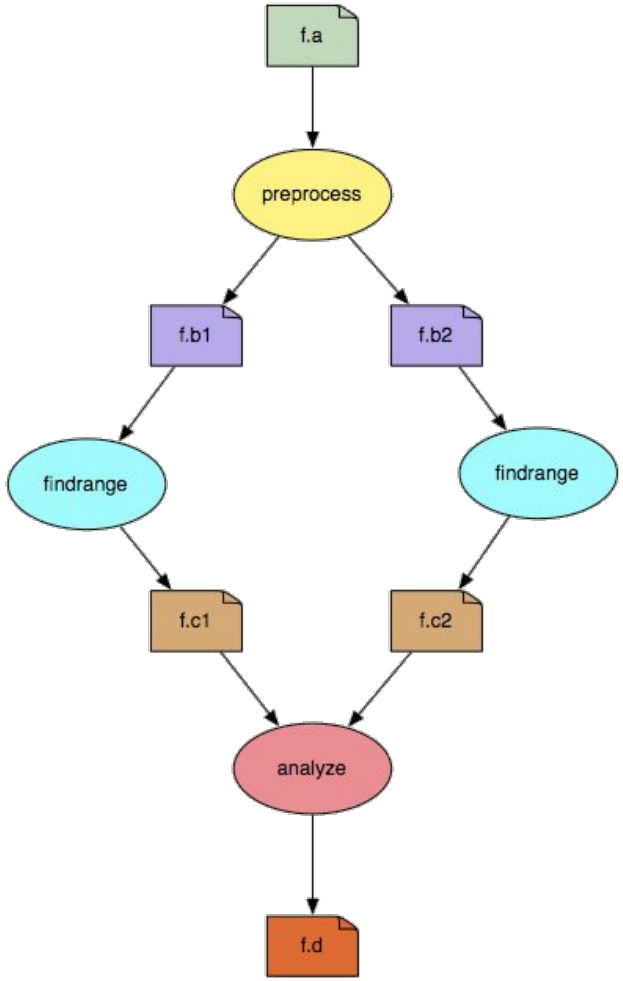
- SlideDeck: https://tinyurl.com/pegasus-adass23
- Slides - Introduction to Pegasus
- Hands on (API, Debugging, Command Line)
- Montage Example



1. 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

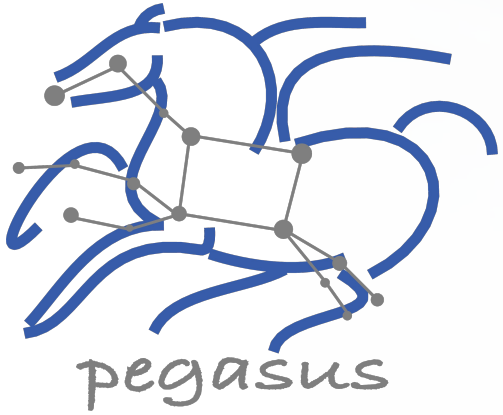
- Describe complex workflows in a simple way
- Access distributed, heterogeneous data and resources (heterogeneous interfaces)
- Deal with resources/software that change over time
- Ease of use. Ability to debug and monitor large workflows

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.

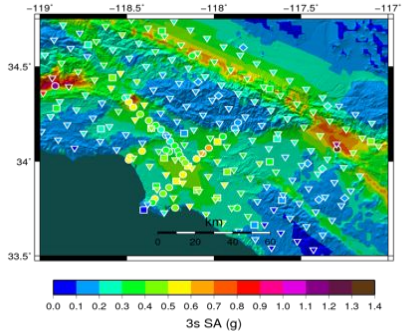
HTCondor High Throughput Computing

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



Some of The Success Stories...

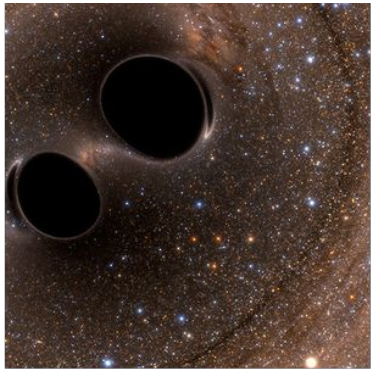
Southern California Earthquake Center's CyberShake



Mix of MPI and single-core jobs, mix of CPU, GPU codes.
Large data sets (10s of TBs), ~300 workflows with 420,000 tasks each
Supported since 2005: changing CI, x-platform execution

First Physics-Based "Shake map" of Southern California

Laser Interferometer Gravitational-Wave Observatory (LIGO)

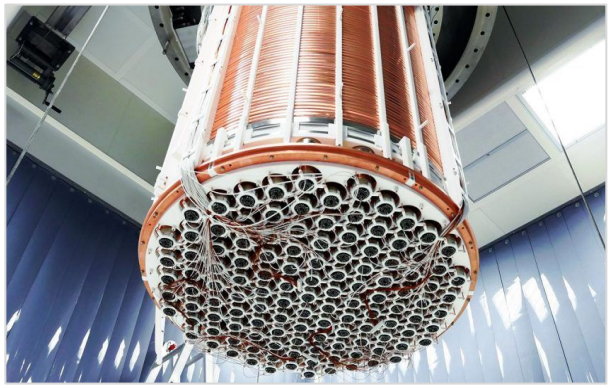


High-throughput computing workload, access to HPC resources, ~ 21K Pegasus workflows, ~ 107M tasks

Supported since 2001, distributed data, opportunistic computing resources

First direct detection of a gravitational wave (colliding black holes)

XENONnT - Dark Matter Search



- Custom data management
- Rucio for data management
- MongoDB instance to track science runs and data products.

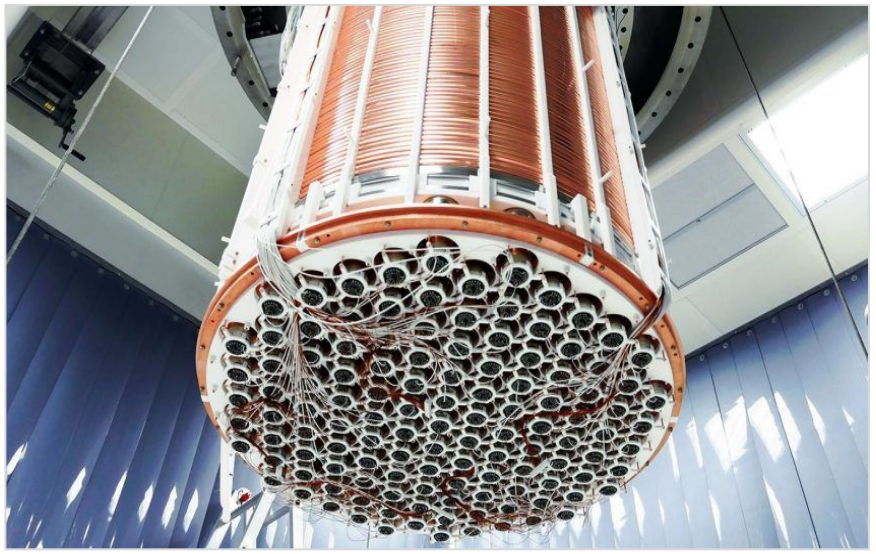
Monte Carlo simulations and the main processing pipeline.

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
```

Main processing pipeline is being developed for XENONnT - data taking will start at the end of 2019. Workflow in development:

Event Horizon Telescope

Bringing Black Holes into Focus

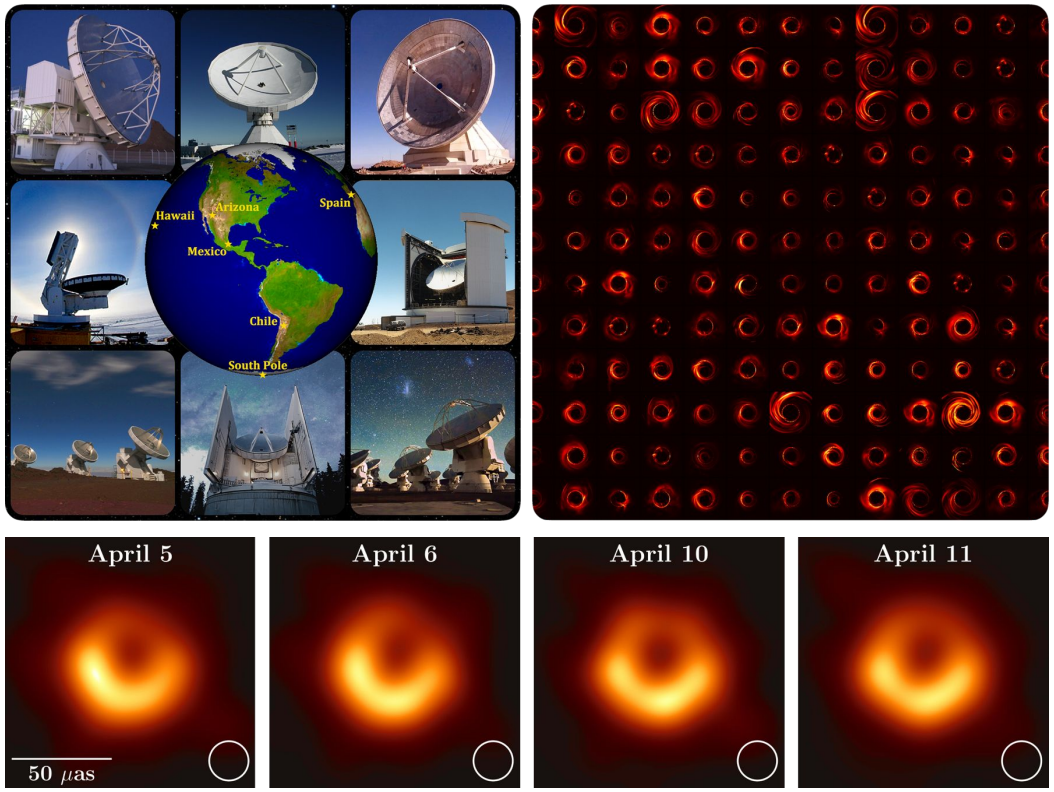
480,000 jobs - 2,600,000 core hours

#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

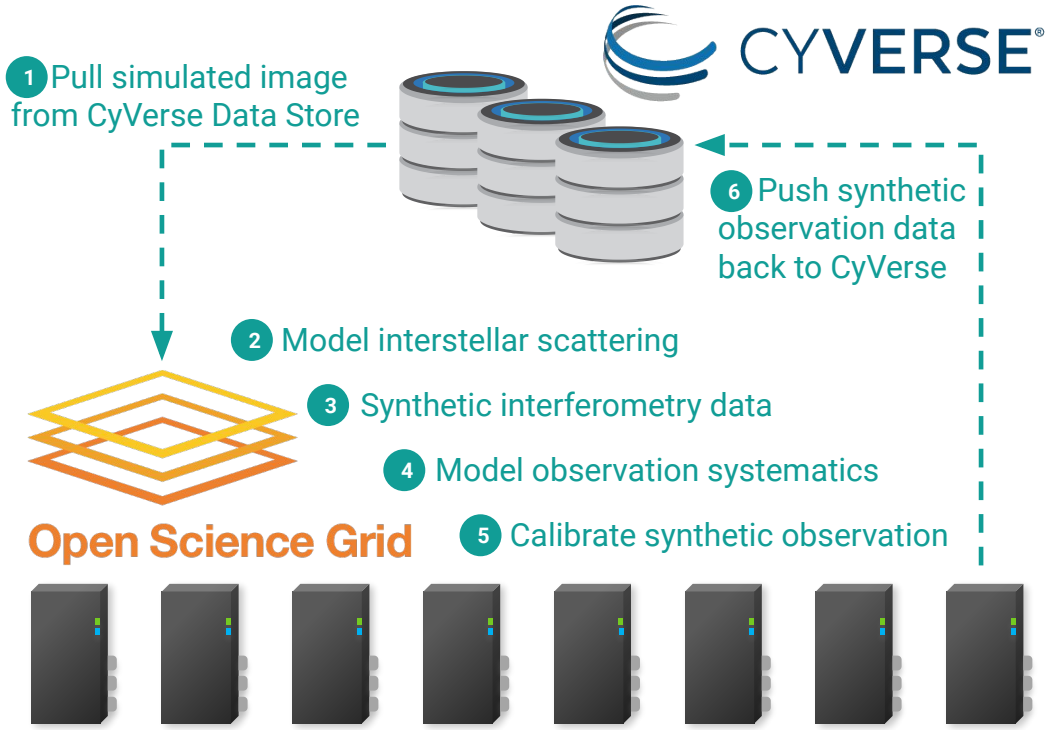


First images of black hole at the center of the M87 galaxy

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

Pegasus-SYMBA Pipeline

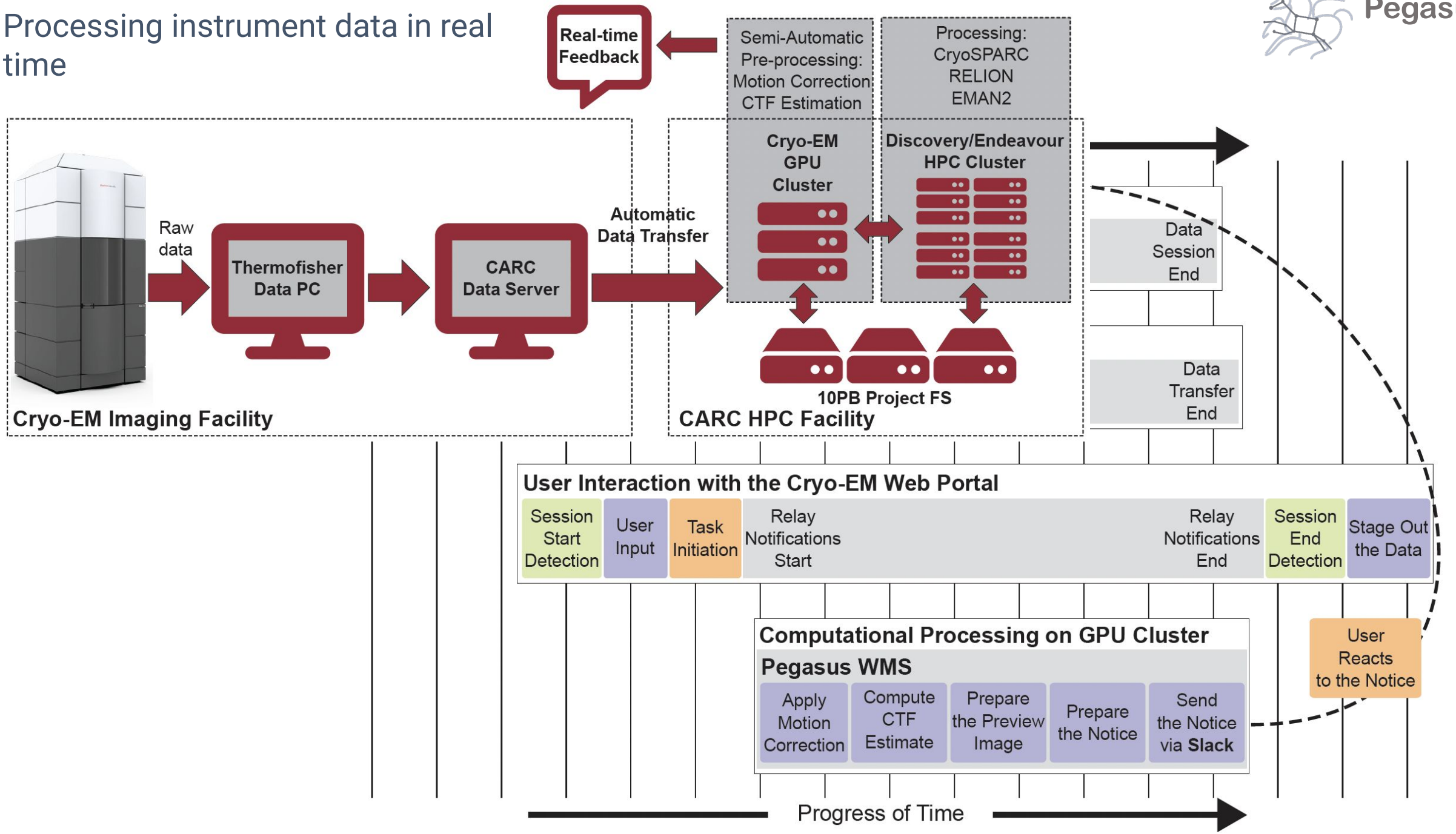
Physically accurate synthetic observation data from simulations are keys to develop calibration and imaging algorithms, as well as comparing the observation with theory and interpreting the results.



USC Viterbi School of Engineering Information Sciences Institute

https://pegasus.isi.edu

Processing instrument data in real time



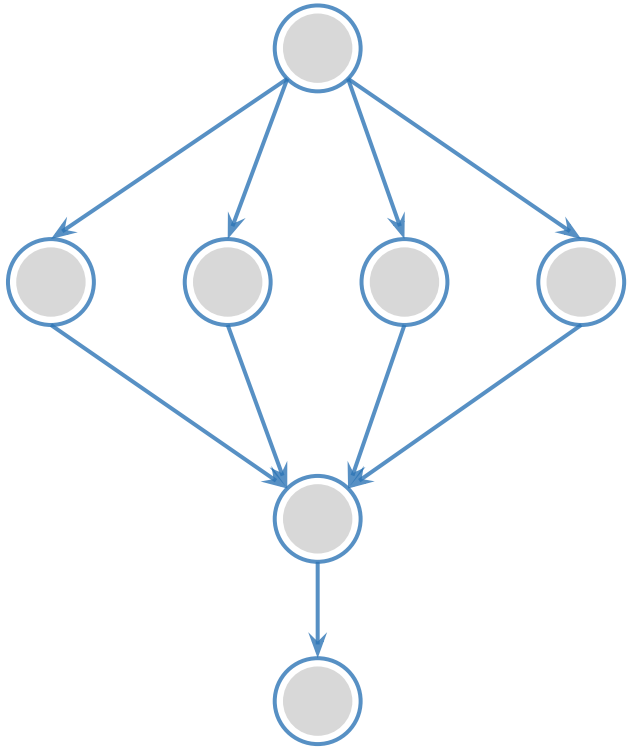
Key Pegasus Concepts

- **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

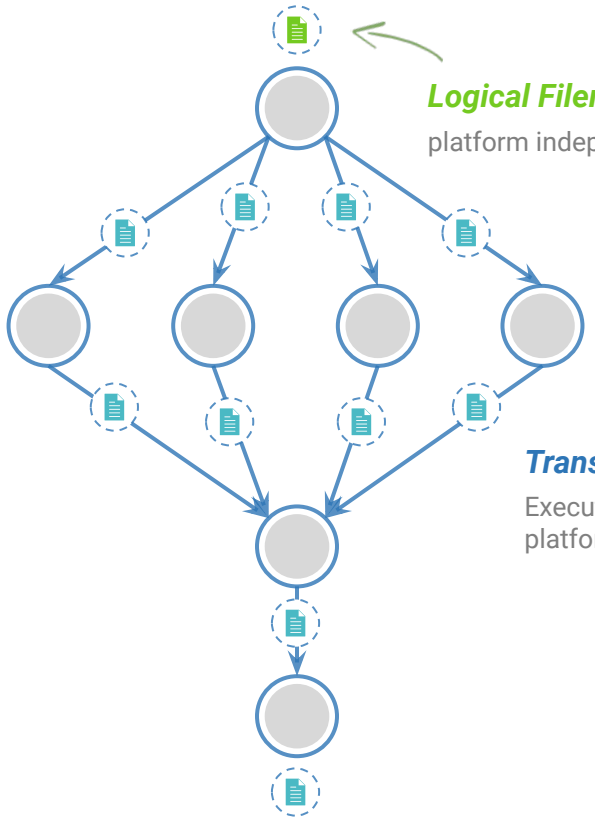
directed-acyclic graphs

Output Workflow

Portable Description

Users do not worry about low level execution details

ABSTRACT WORKFLOW

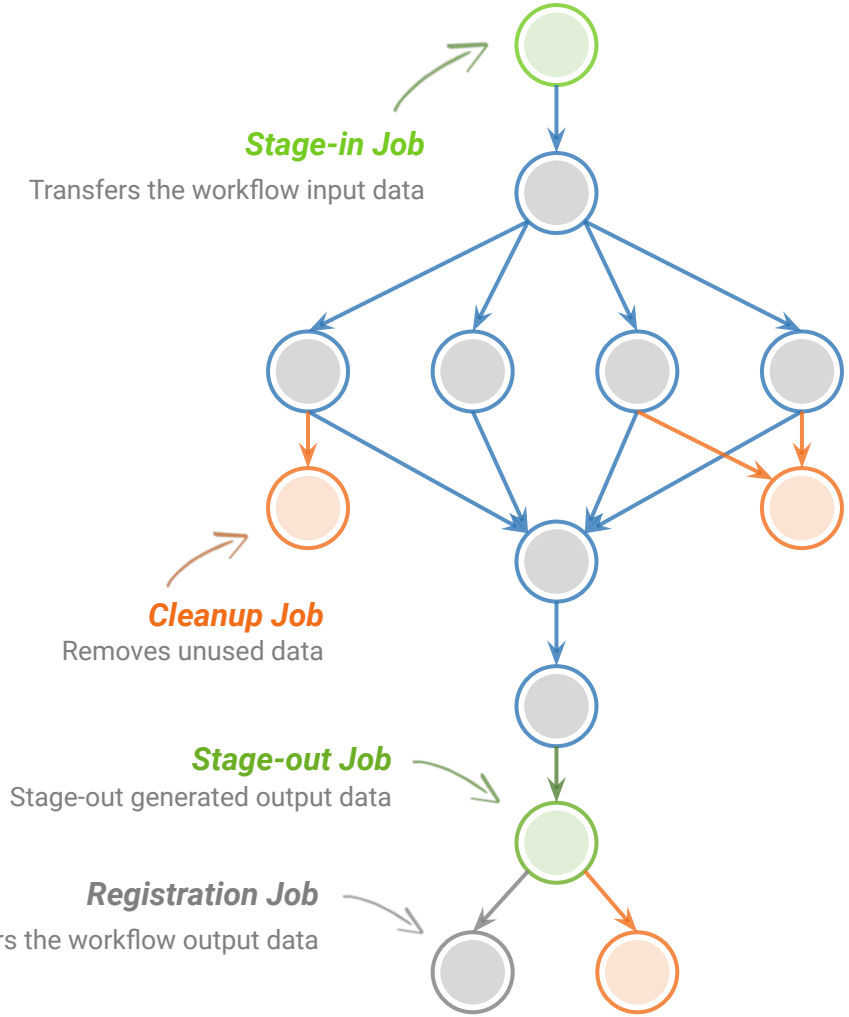


Logical Filename (LFN)

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

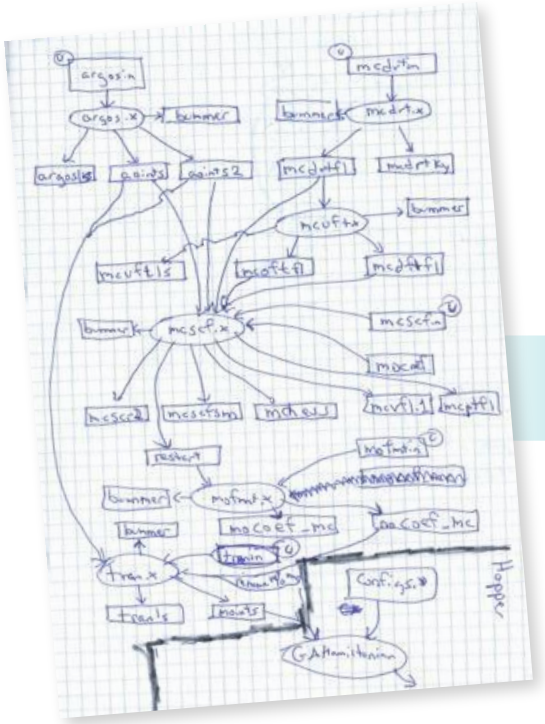
Stage-out generated output data

Registration Job

Registers the workflow output data

EXECUTABLE WORKFLOW

Pegasus provides tools to generate the Abstract Workflow



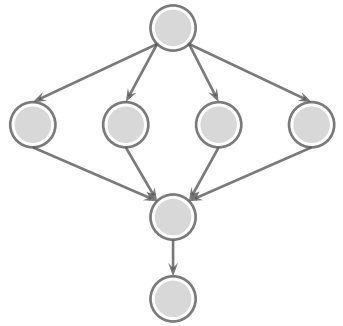
```python
#!/usr/bin/env python3

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

R

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

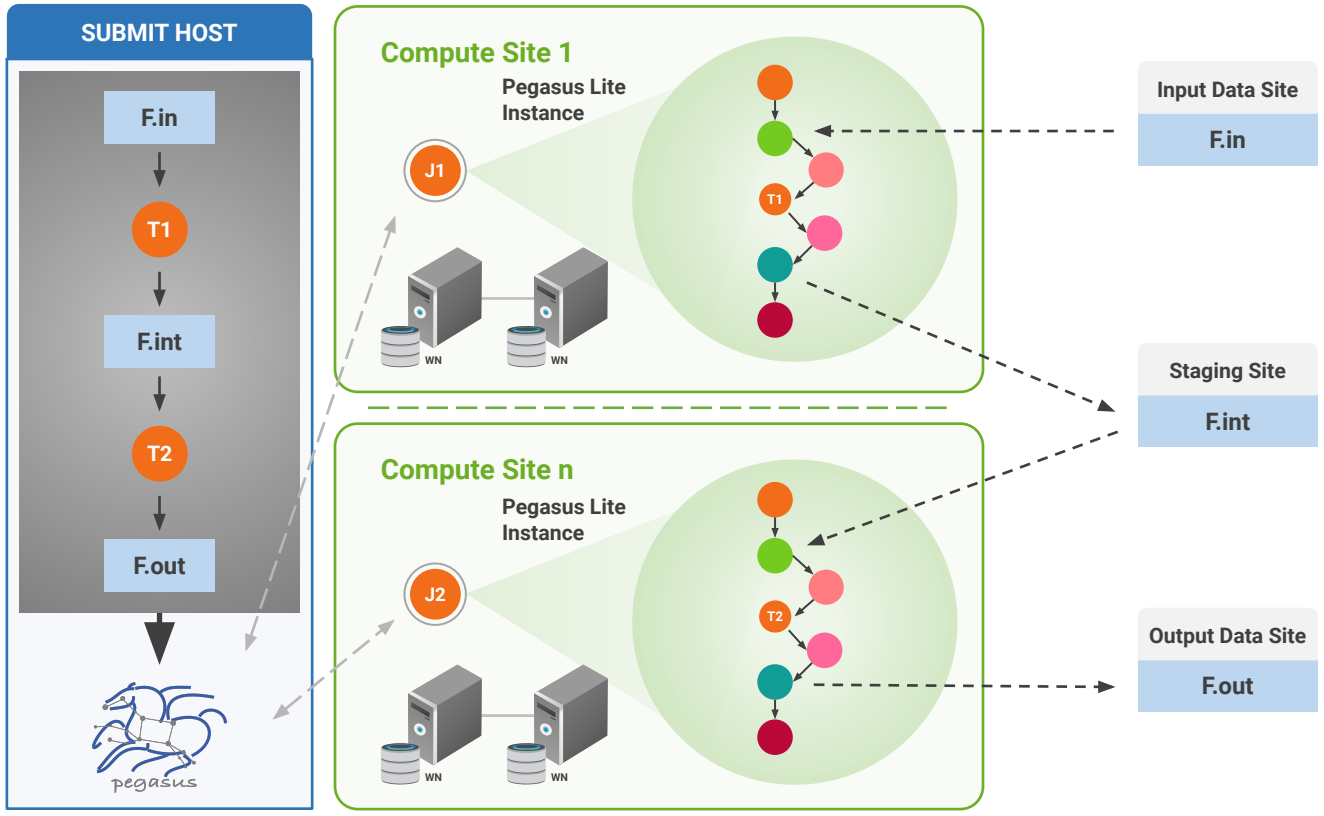
- Host Input Data

Data Staging Site

- Coordinate data movement for workflow

Output Site

- Where output data is placed

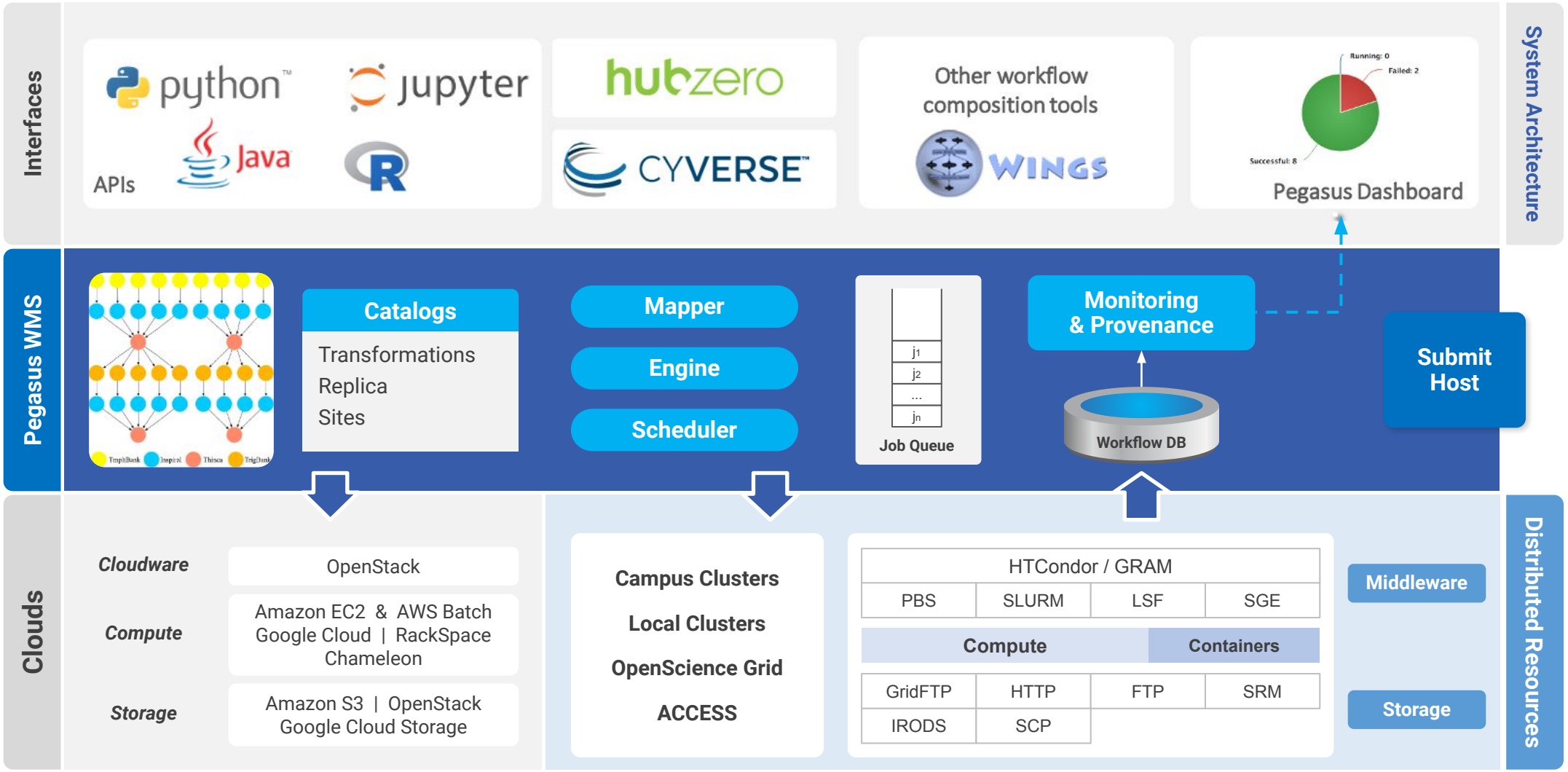


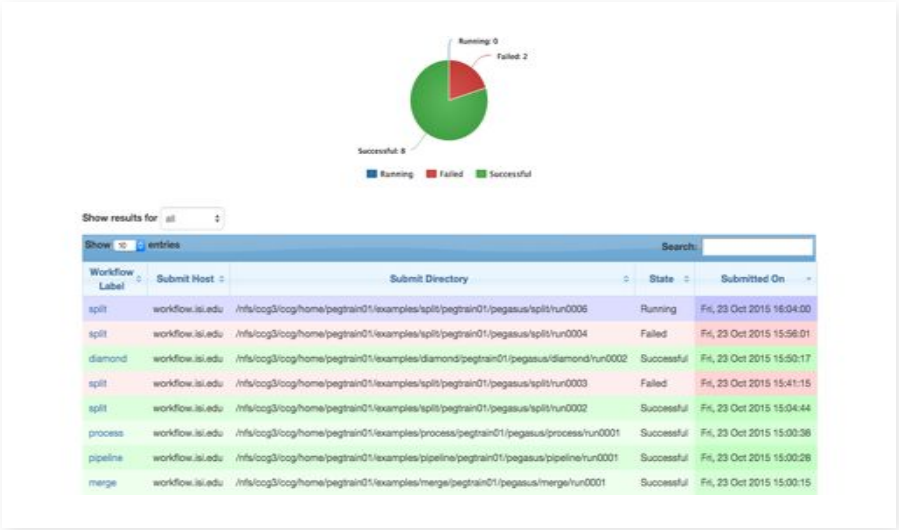
Pegasus-transfer

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., 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
```


Interfaces
python
jupyter
APIs
Java
R
hubzero
CYVERSE
Other workflow composition tools
WINGS
Running: 0
Failed: 2
Successful: 8
Pegasus Dashboard
System Architecture
Pegasus WMS
Catalogs
Transformations
Replica
Sites
Mapper
Engine
Scheduler
j1
j2
...
jn
Job Queue
Monitoring & Provenance
Workflow DB
Submit Host
Clouds
Cloudware
OpenStack
Compute
Amazon EC2 & AWS Batch
Google Cloud | RackSpace
Chameleon
Storage
Amazon S3 | OpenStack
Google Cloud Storage
Campus Clusters
Local Clusters
OpenScience Grid
ACCESS
HTCondor / GRAM
PBS
SLURM
LSF
SGE
Compute
Containers
GridFTP
HTTP
FTP
SRM
IRODS
SCP
Middleware
Storage
Distributed Resources

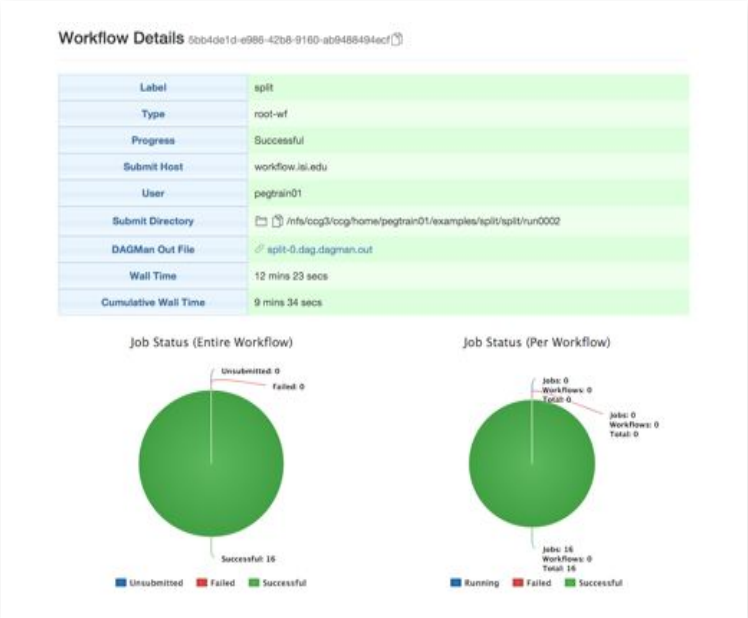


PEGASUS DASHBOARD

web interface for monitoring and debugging workflows

Real-time **monitoring** of workflow executions. It shows the **status** of the workflows and jobs, job **characteristics, statistics** and **performance** metrics.

Provenance data is stored into a relational database.



- Real-time Monitoring
- Reporting
- Debugging
- Troubleshooting
- RESTful API

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

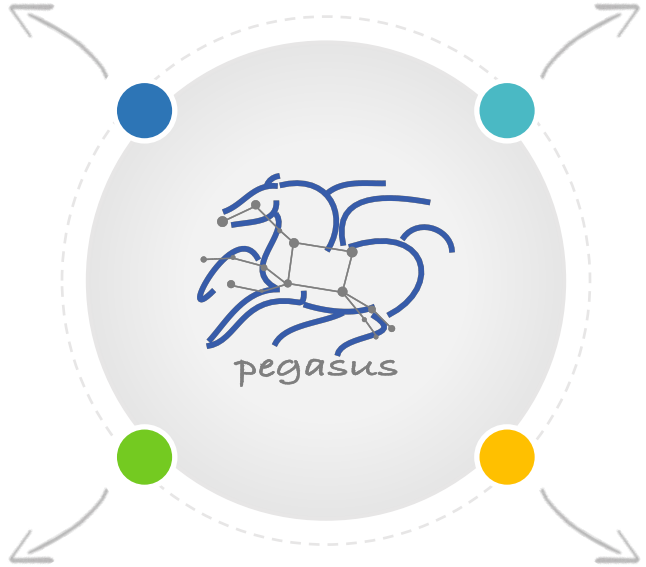
And if a job fails?

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

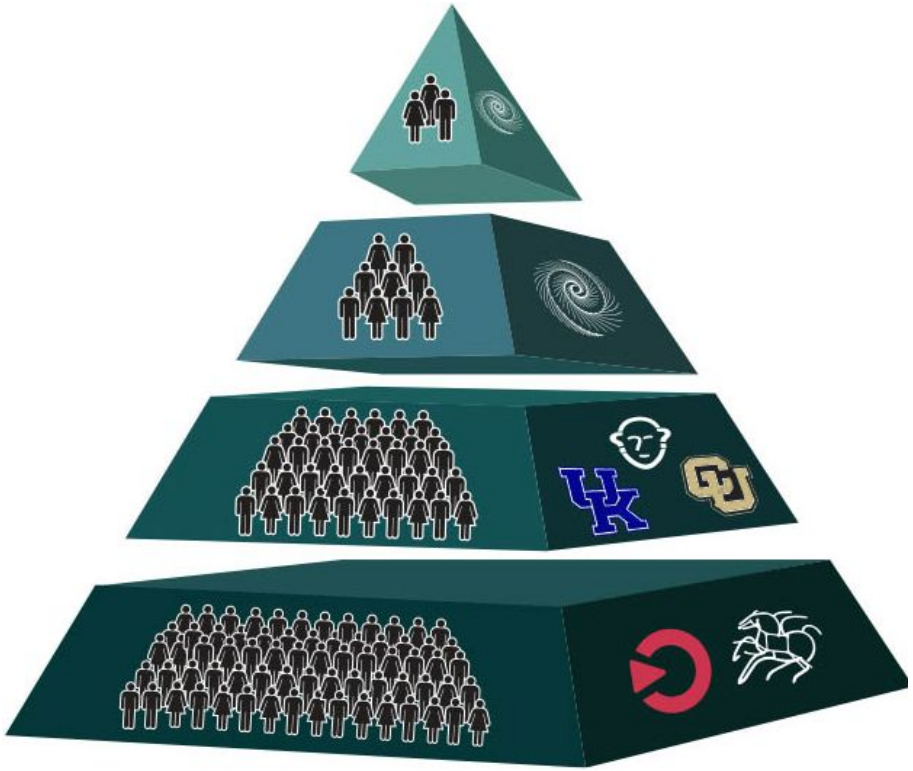
workflow can be restarted from checkpoint file recover from failures with minimal loss

ACCESS Pegasus

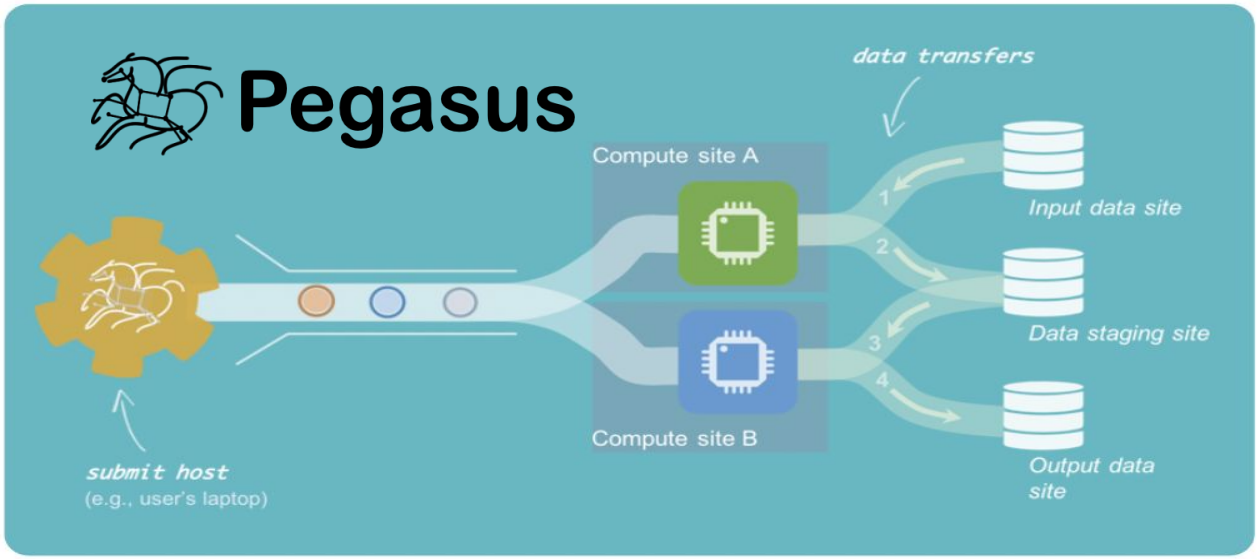
https://support.access-ci.org/pegasus

Democratizing Access to Resources

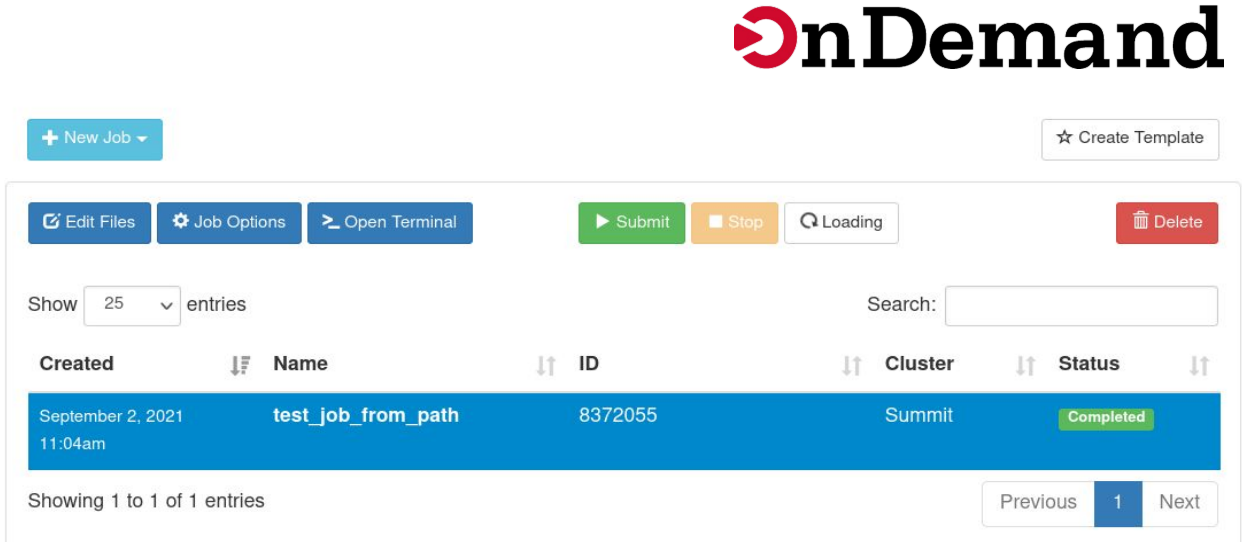
- Decrease time to science
- Allow researchers to self-serve
- Leverage existing tools
- Comprehensive Knowledge Base (KB)
- Events, training
- Question and answer forum – ask.ci
- Concierge level support



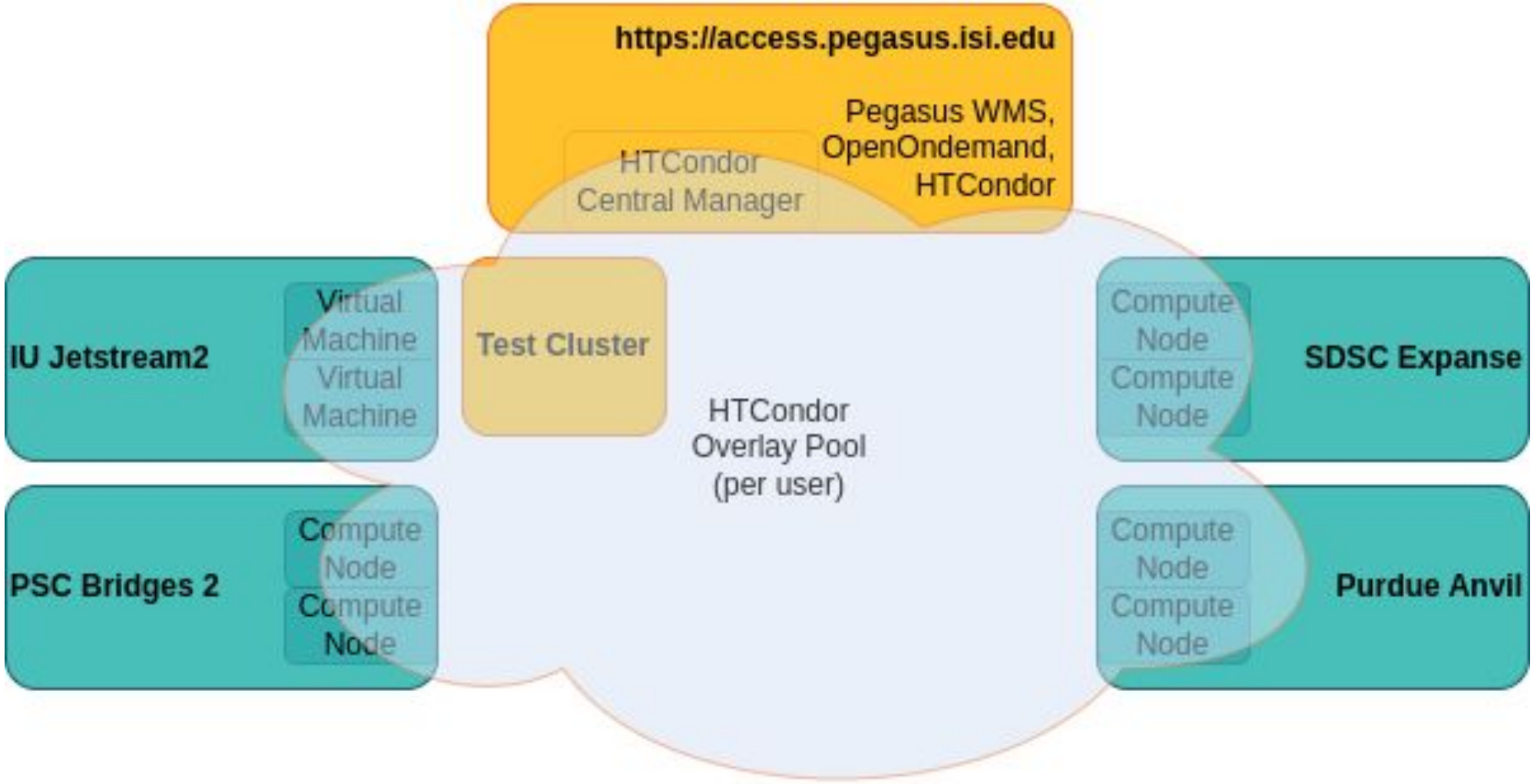
Tools to Make Large-Scale Computing Easier

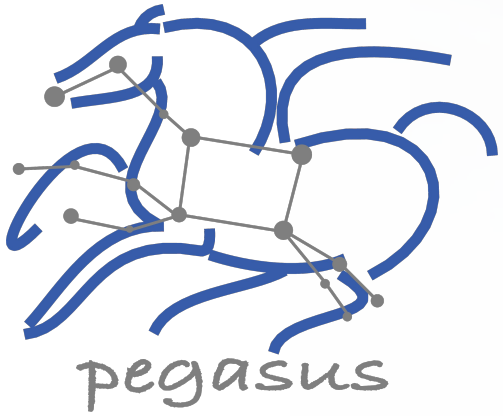


- Input data -> Compute Job -> Output data
- Complex data workflows
- Reproducible
- Provenance which ensures data integrity



- Improve user experience with an easy-to-use interface to access complex cyberinfrastructure
- Templates to run jobs/transfer data


https://access.pegasus.isi.edu
Pegasus WMS,
OpenOndemand,
HTCondor
HTCondor
Central Manager
IU Jetstream2
Virtual
Machine
Virtual
Machine
Test Cluster
HTCondor
Overlay Pool
(per user)
Compute
Node
Compute
Node
SDSC Expanse
PSC Bridges 2
Compute
Node
Compute
Node
Compute
Node
Compute
Node
Purdue Anvil



2. Hands on Exercises

Hands on Tutorial Exercises: Login to ACCESS Pegasus

- To do the online training exercises you need an ACCESS/XSEDE login. You can create one using your academic institution credentials.
  - https://adass2023.lpl.arizona.edu/sites/adass2023.lpl.arizona.edu/files/documents/ADASS23-Instructions-for-Pegasus-Tutorial-v2.pdf (short url: https://shorturl.at/FJL06)

- Use a web browser and log on to ACCESS Pegasus at https://access.pegasus.isi.edu

  Backup (If problem with ACCESS Accounts): https://tinyurl.com/pegasus-adass23-backup

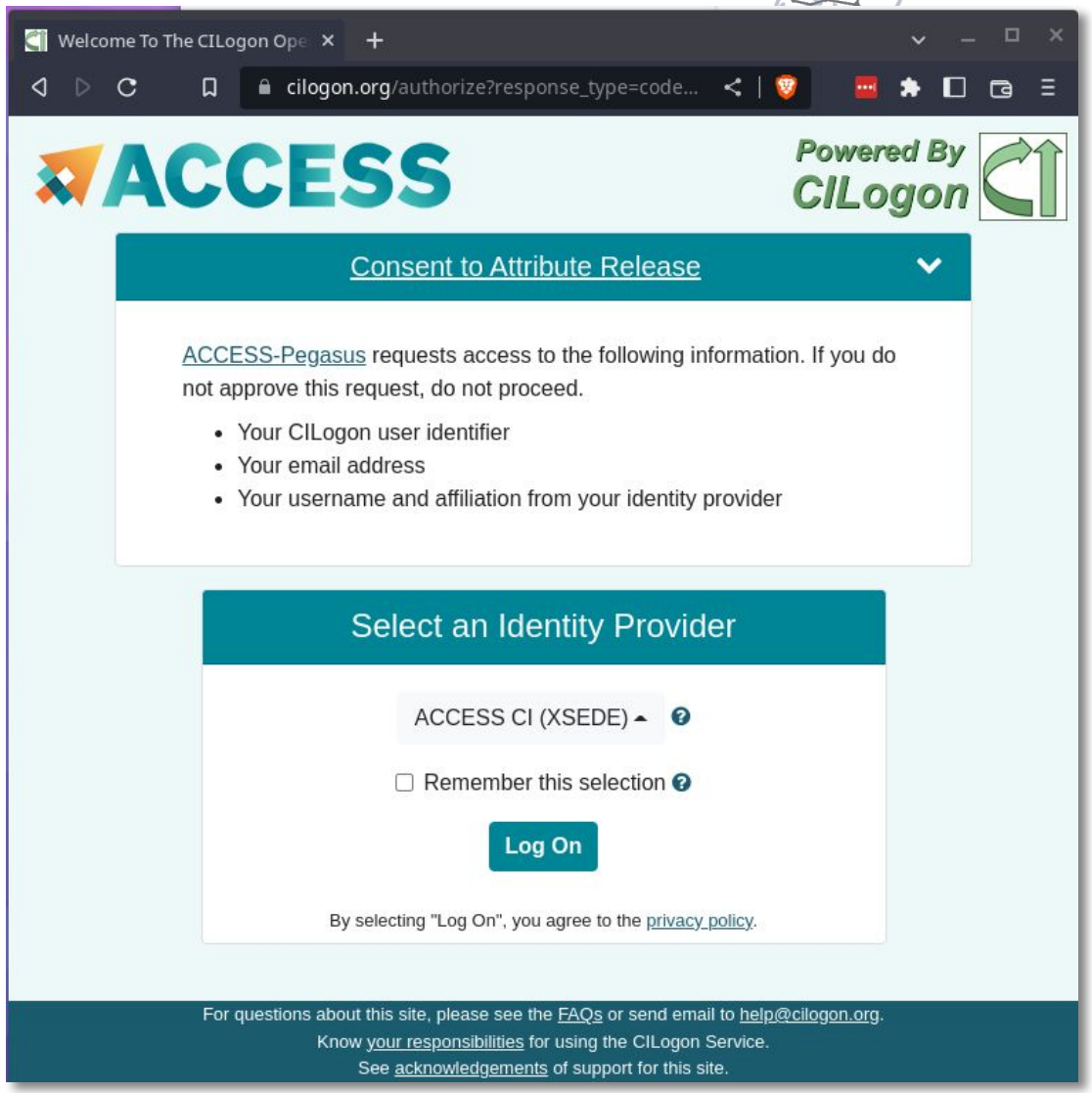
Prepare

Logging In

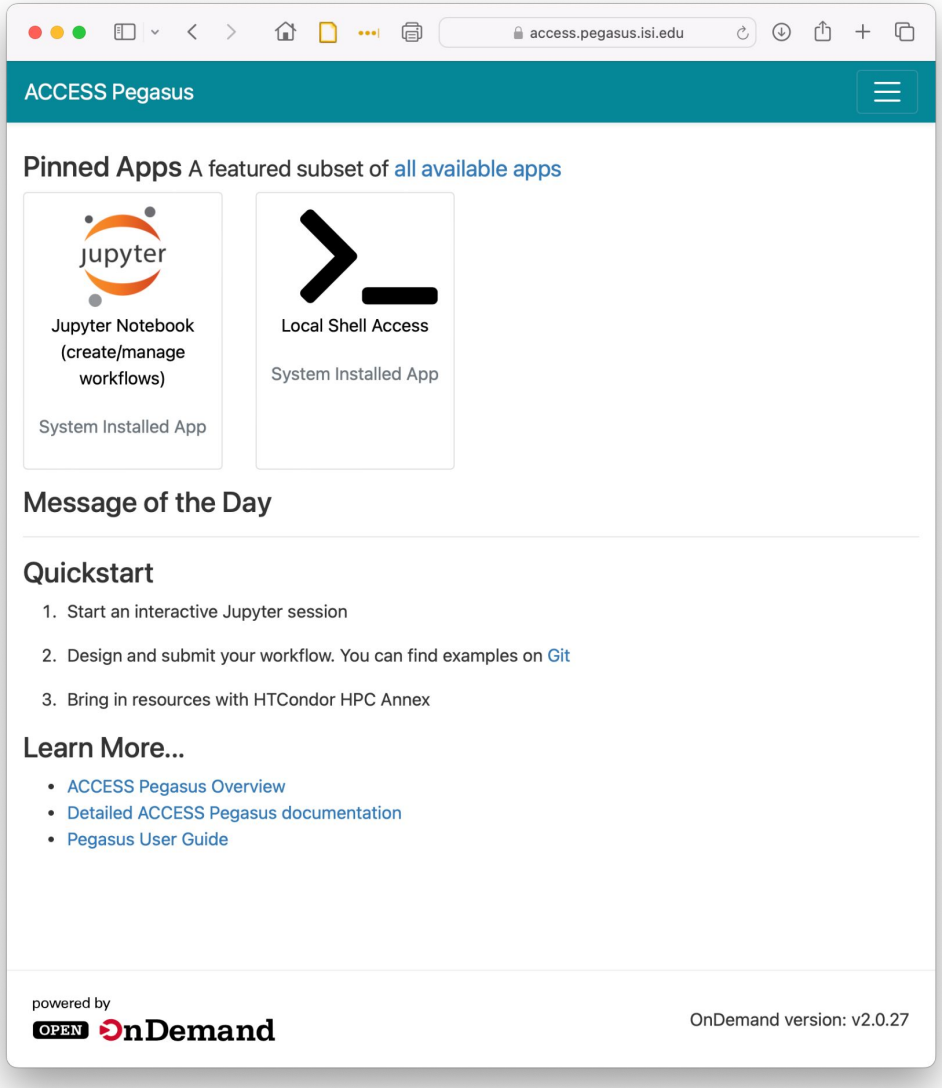
CILogin with your ACCESS ID and institutional login

- https://access.pegasus.isi.edu

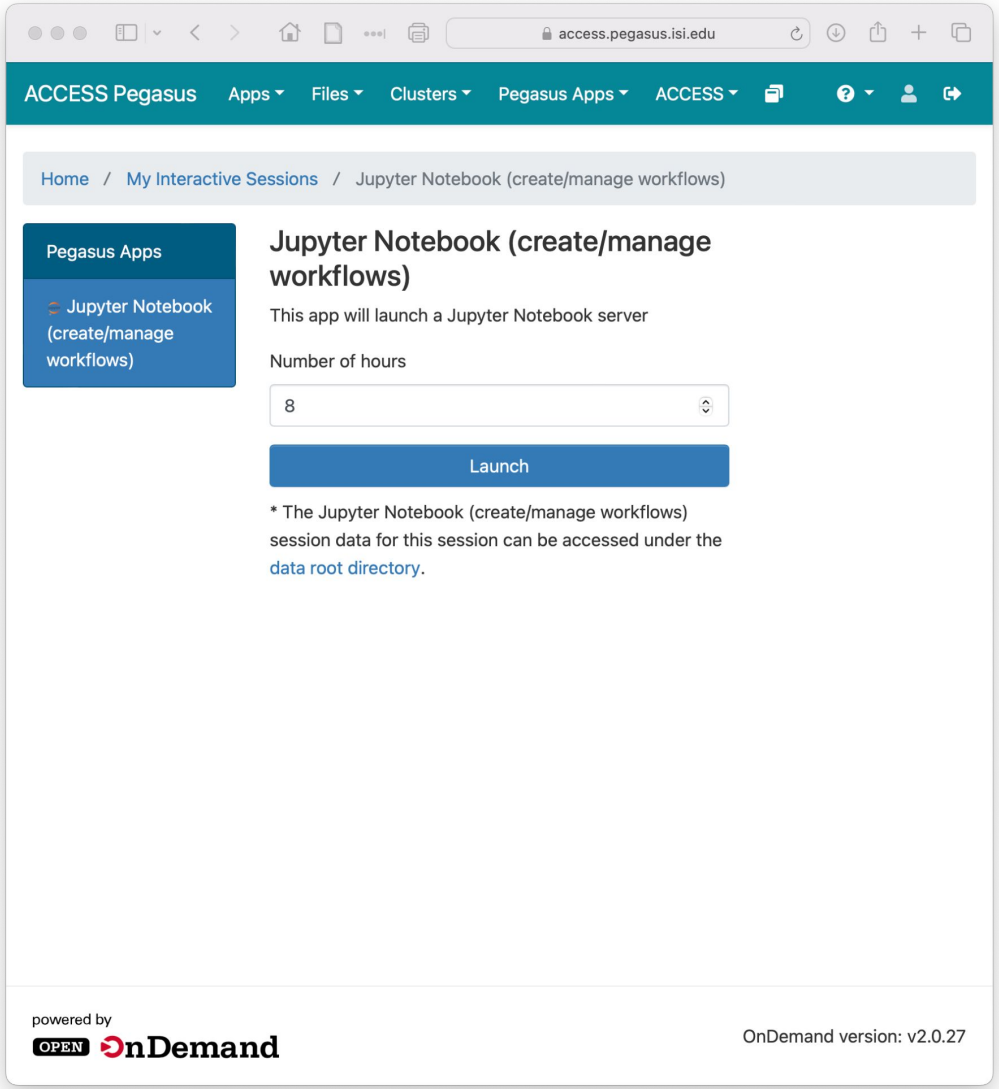
All registered ACCESS users with an active allocation automatically have access



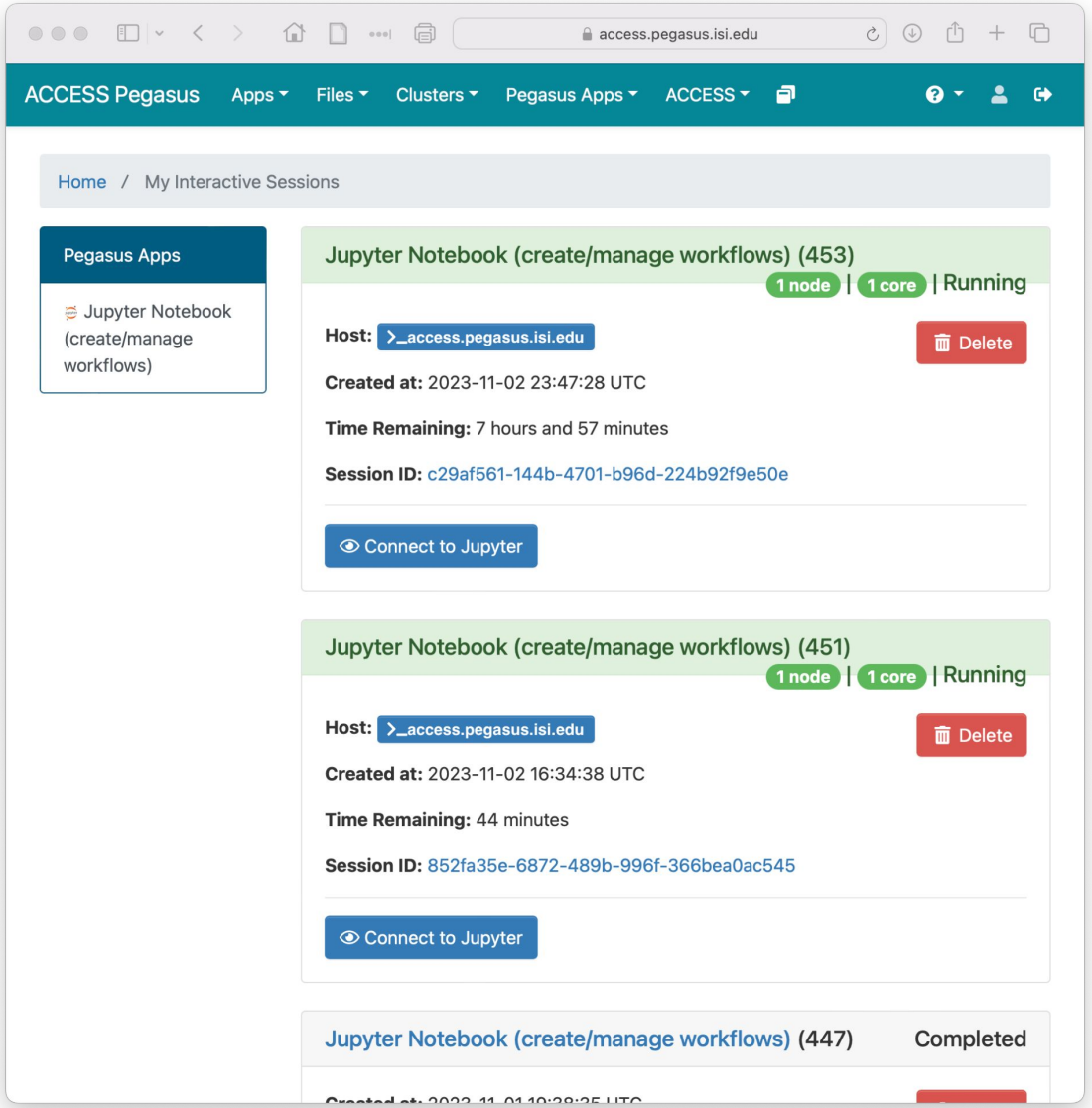
Hands on Tutorial Exercises: Start a Jupyter Server



Hands on Tutorial Exercises: Start a Jupyter Server

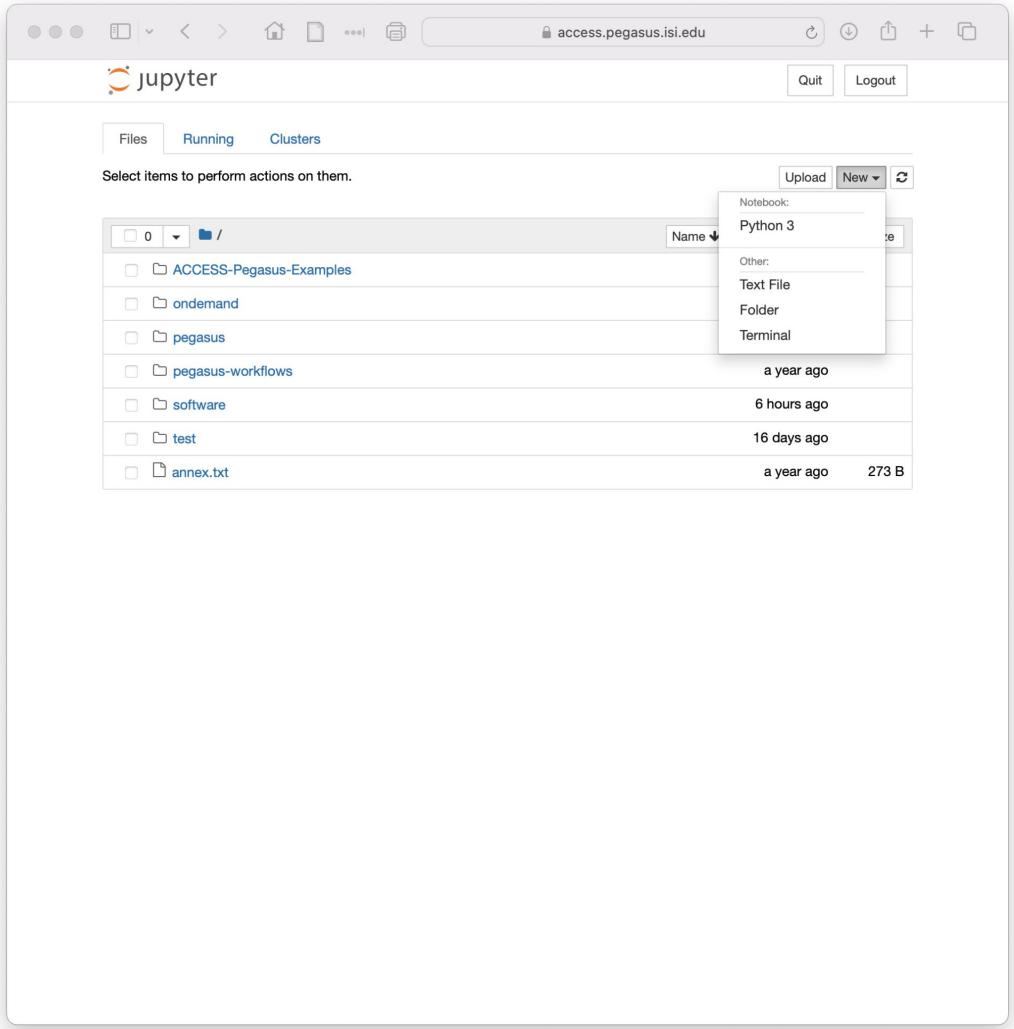


Hands on Tutorial Exercises: Connect to JupyterLab

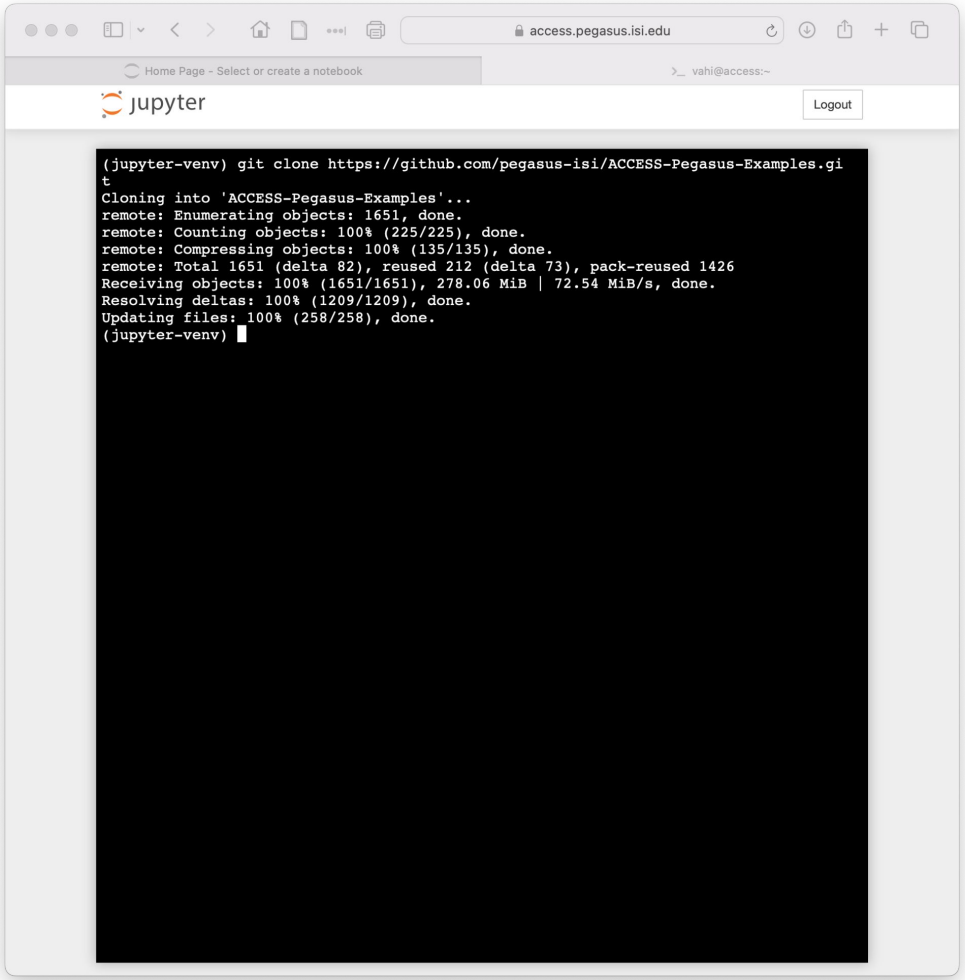


Hands on Tutorial Exercises: Start a Terminal

- In JupyterLab, Click on File -> New and then click on Terminal to get the terminal



Hands on Tutorial Exercises: Clone Repository

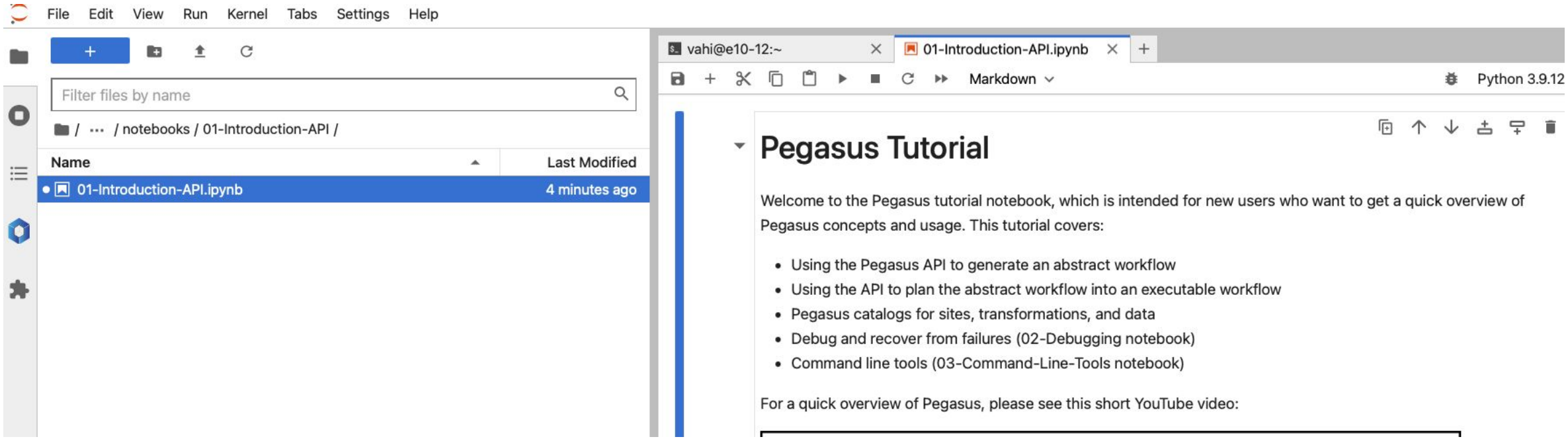


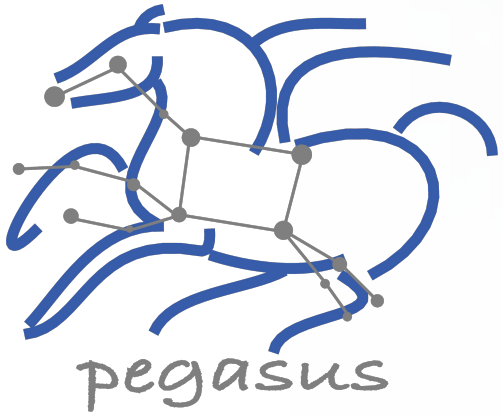
- Clone Tutorial Repository in the terminal

git clone https://github.com/pegasus-isi/ACCESS-Pegasus-Examples.git

Hands on Tutorial Exercises: Navigate to Notebooks

- In Jupyter, navigate to the example you are interested in, and step through the notebook.
- For first time users, we highly recommend to do the notebooks in order, as they build up on concepts in the previous notebooks.

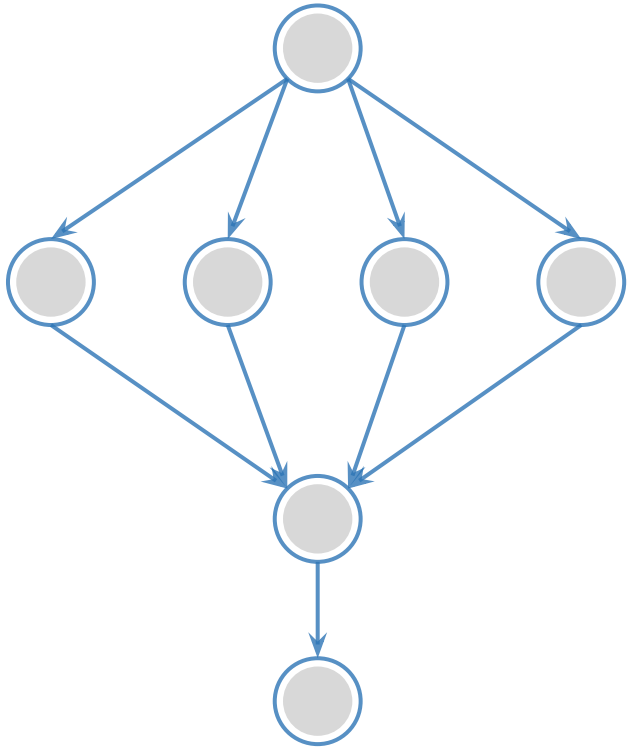




2.1 API

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

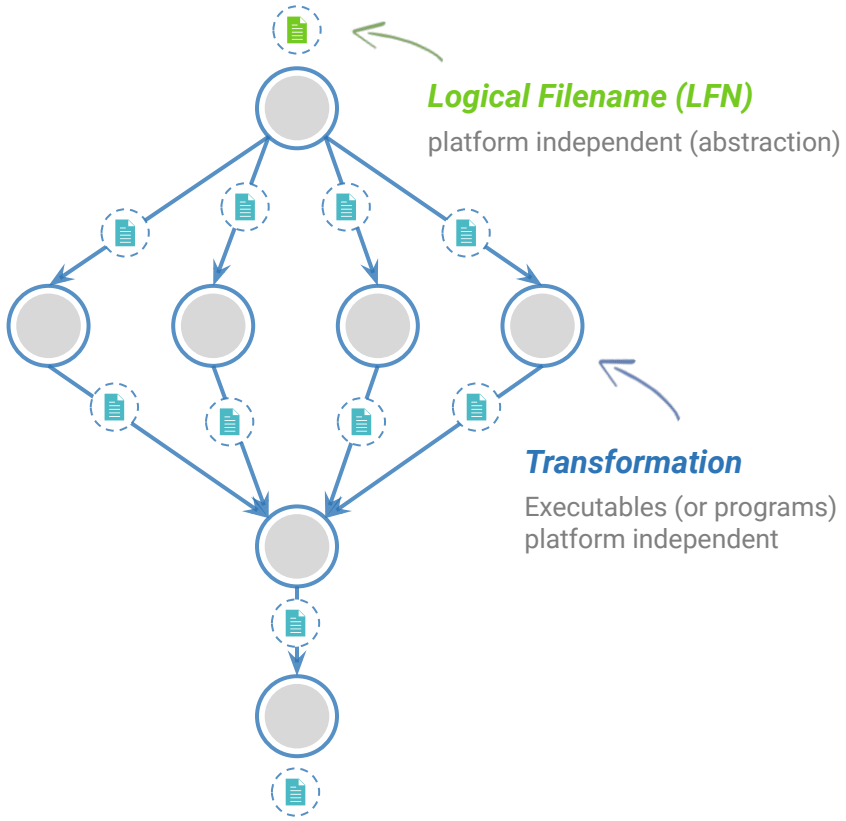
directed-acyclic graphs

Output Workflow

Portable Description

Users do not worry about low level execution details

ABSTRACT WORKFLOW

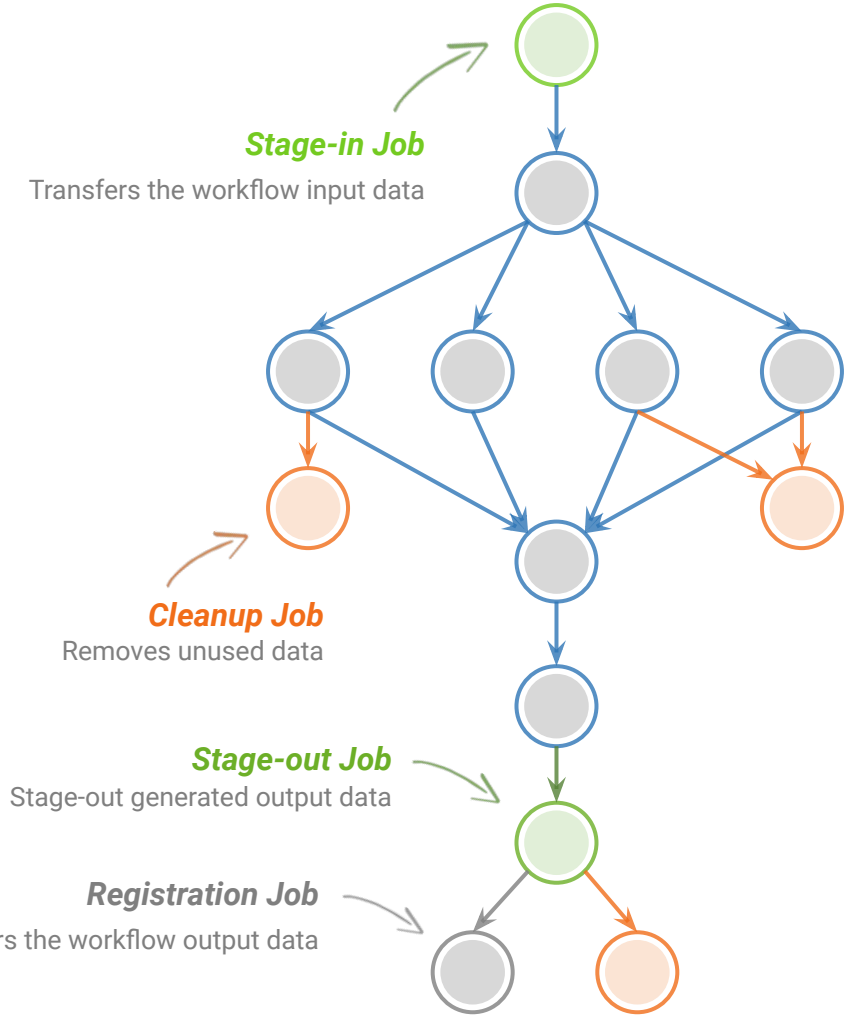


Logical Filename (LFN)

platform independent (abstraction)

Transformation

Executables (or programs)
platform independent



Stage-in Job

Transfers the workflow input data

Cleanup Job

Removes unused data

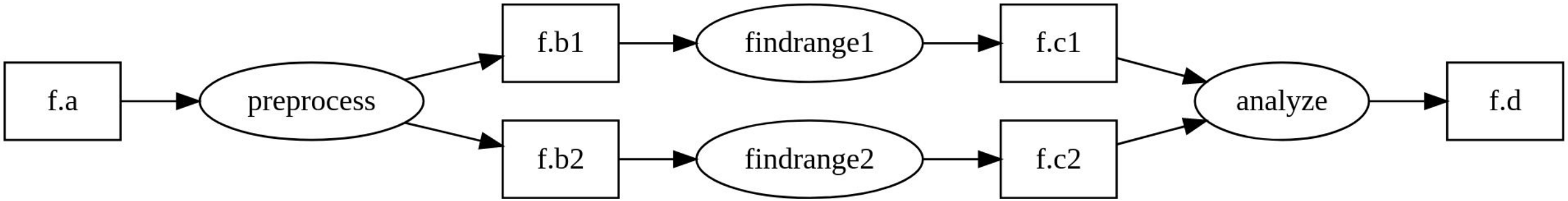
Stage-out Job

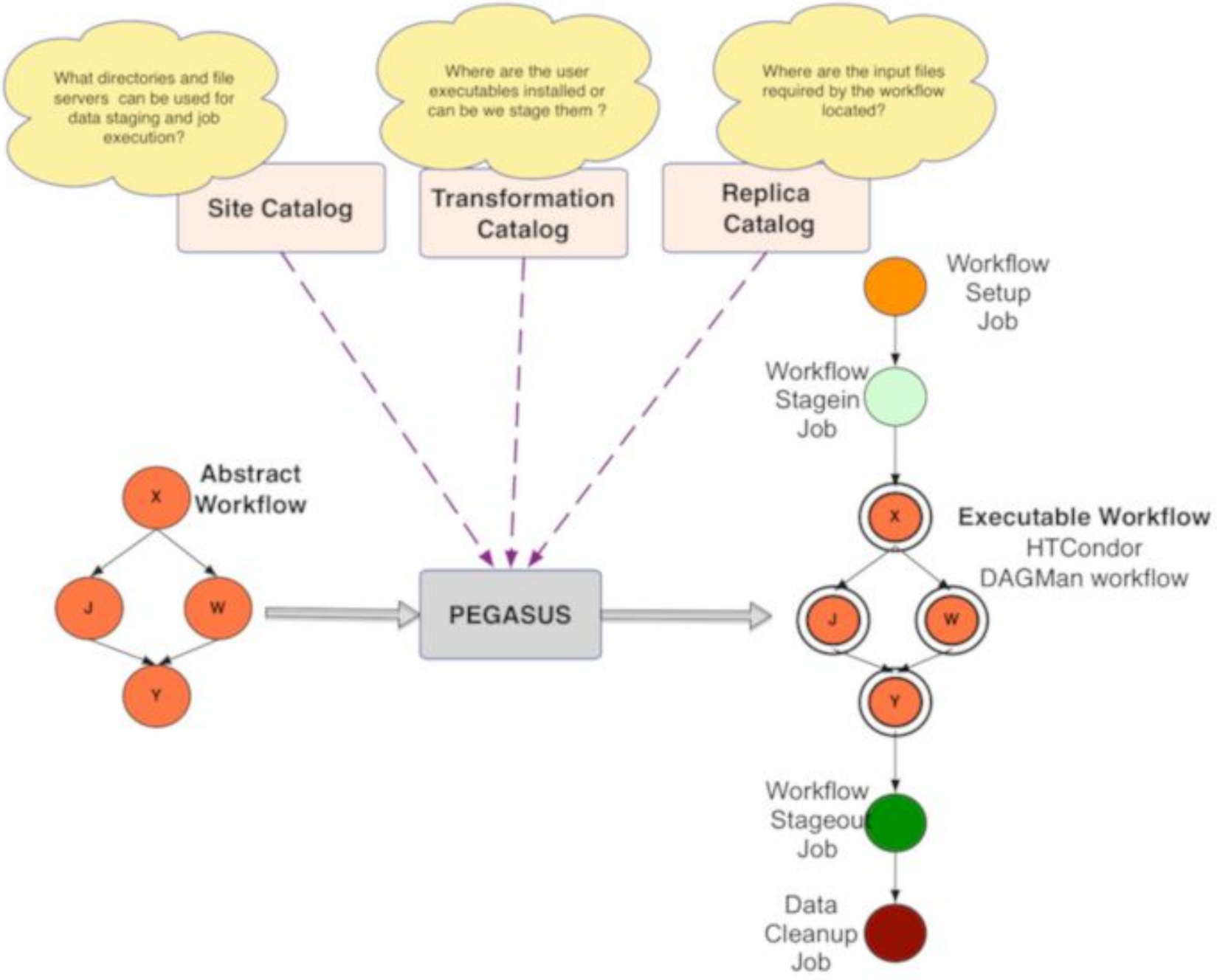
Stage-out generated output data

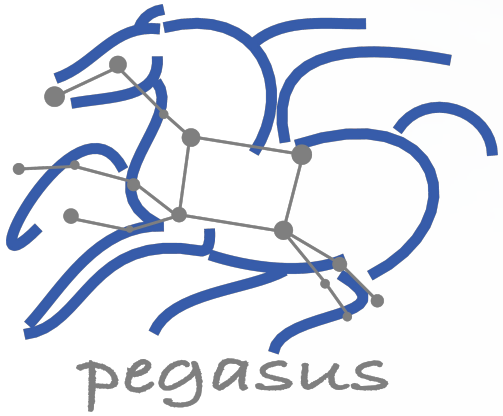
Registration Job

Registers the workflow output data

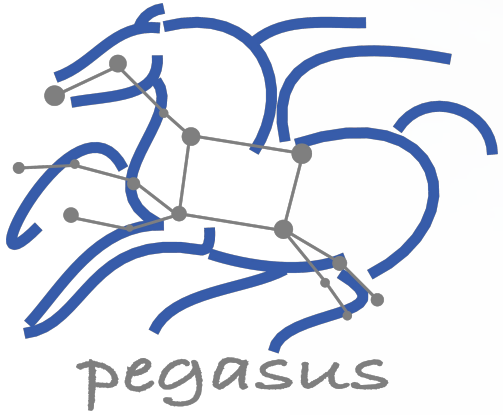
EXECUTABLE WORKFLOW


f.a
preprocess
f.b1
f.b2
findrange1
findrange2
f.c1
f.c2
analyze
f.d


What directories and file servers can be used for data staging and job execution?
Where are the user executables installed or can be we stage them ?
Where are the input files required by the workflow located?
Site Catalog
Transformation Catalog
Replica Catalog
Workflow Setup Job
Workflow Stagein Job
Abstract Workflow
X
J
W
Y
PEGASUS
Executable Workflow
HTCondor
DAGMan workflow
Workflow Stageout Job
Data Cleanup Job



2.2 Debugging



2.3 Command Line Tools

Pegasus Container Support

Pegasus

docker SHIFTER

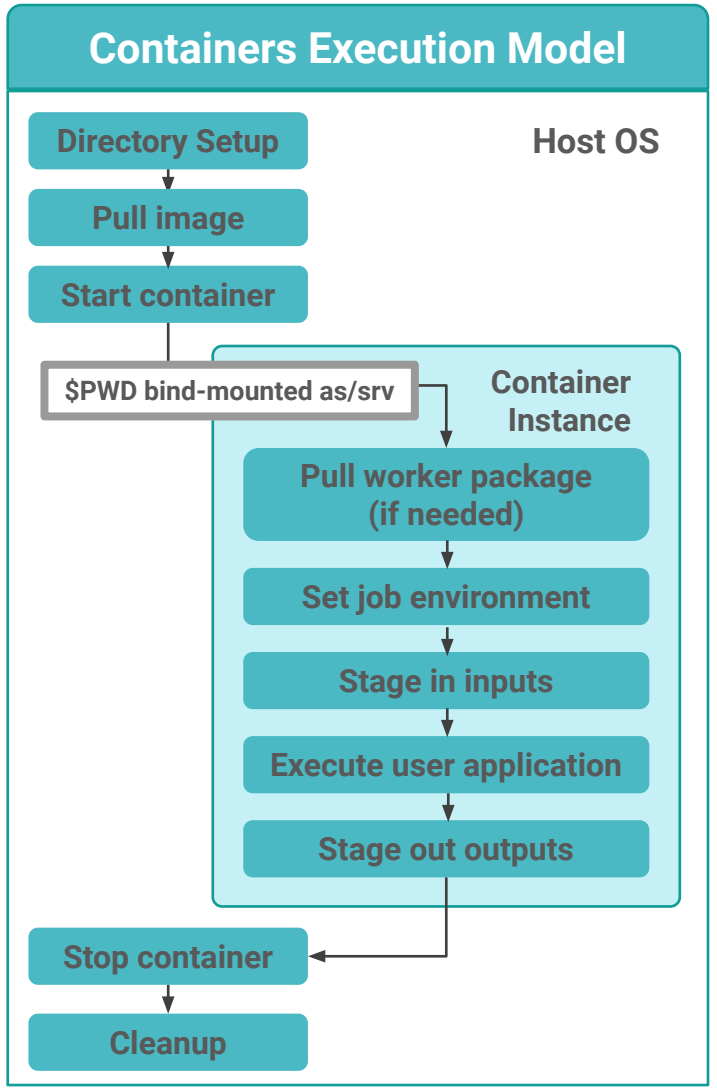
Users can refer to **containers** in the **Transformation Catalog** with their executable preinstalled

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

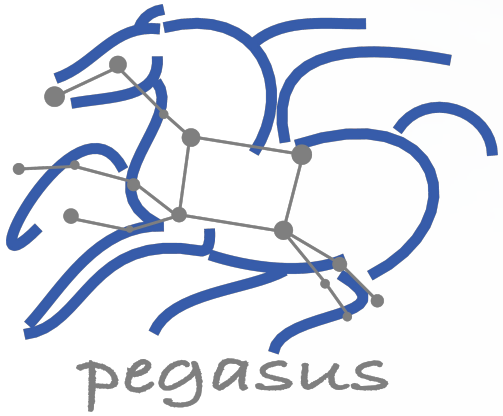
- Staged to compute node if not present
- Docker or Singularity Hub URL's
- Docker Image exported as a TAR file and available at a server, just like any other input dataset

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

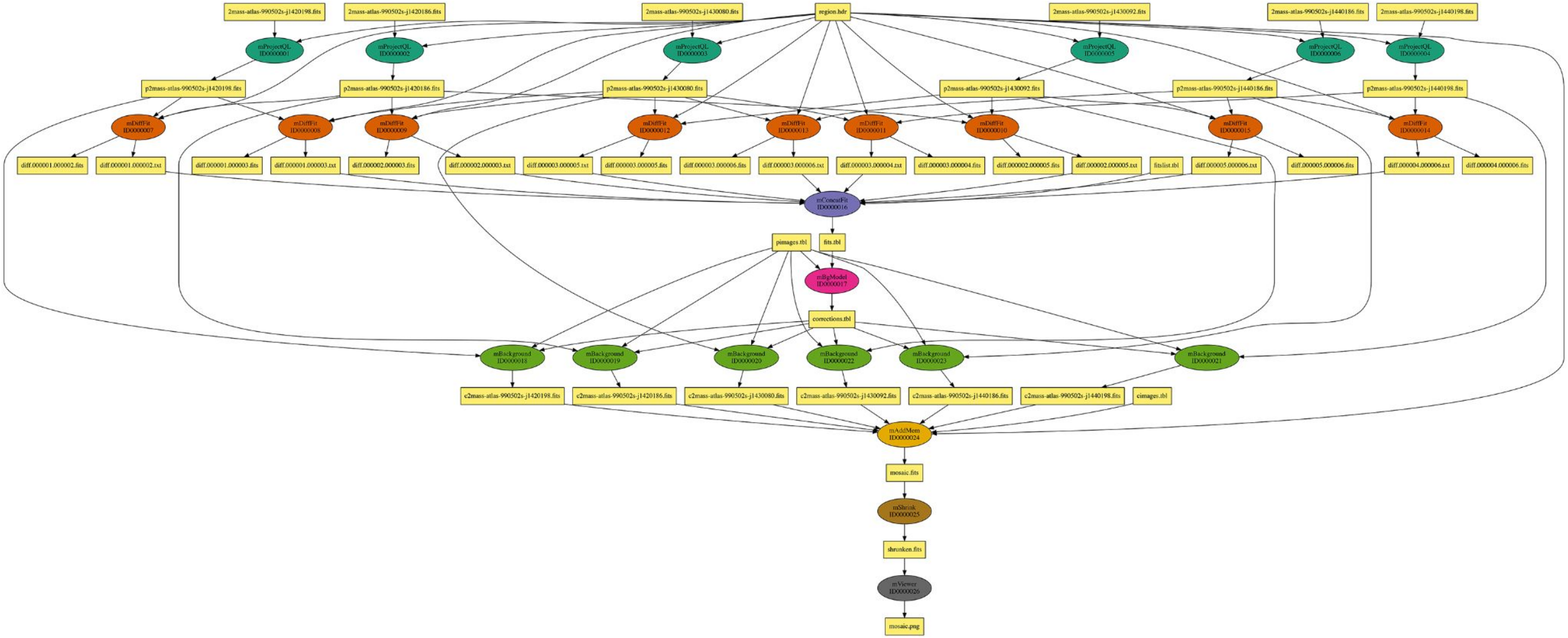


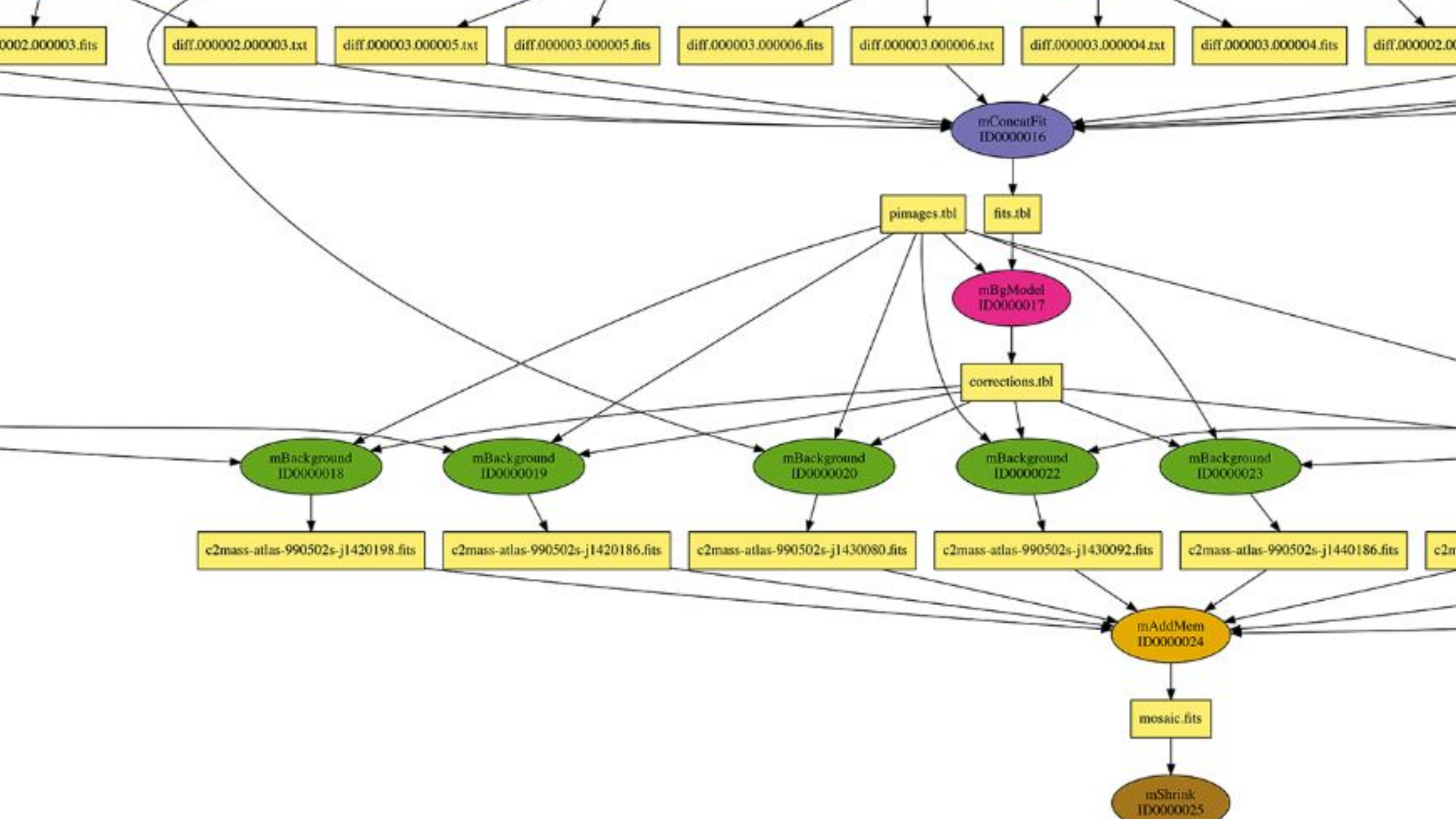
2.4 Montage Workflow

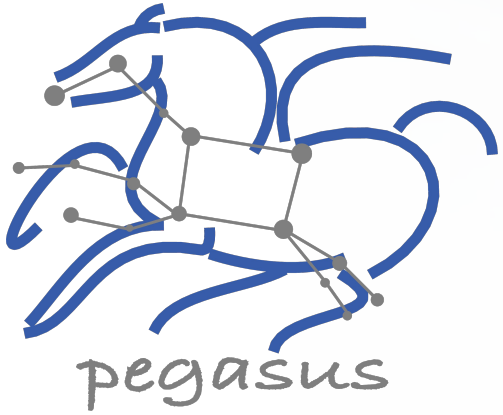
Montage Workflow

- Create a mosaic of M17 (the Omega Nebula, 1 degree x 1 degree, in the 2MASS J-band)
- Reproject, background correct, coadd
- Using Montage in two contexts and in slightly different ways.
  - Montage modules through their Python bindings to gather the information we need to create the workflow.
  - Then later, when the workflow is running it uses the C binding to do the compute-intensive image processing.

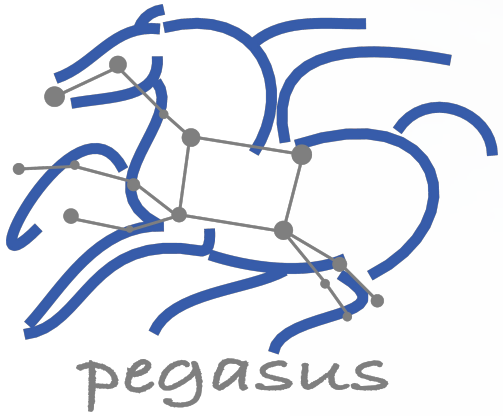
Pegasus


2mass-atlas-990502s-j1420198.fits
2mass-atlas-990502s-j1420186.fits
2mass-atlas-990502s-j1430080.fits
region.hdr
2mass-atlas-990502s-j1430092.fits
2mass-atlas-990502s-j1440186.fits
2mass-atlas-990502s-j1440198.fits
mProjectQL ID0000001
mProjectQL ID0000002
mProjectQL ID0000003
mProjectQL ID0000005
mProjectQL ID0000006
mProjectQL ID0000004
p2mass-atlas-990502s-j1420198.fits
p2mass-atlas-990502s-j1420186.fits
p2mass-atlas-990502s-j1430080.fits
p2mass-atlas-990502s-j1430092.fits
p2mass-atlas-990502s-j1440186.fits
p2mass-atlas-990502s-j1440198.fits
mDiffFit ID0000007
mDiffFit ID0000008
mDiffFit ID0000009
mDiffFit ID0000012
mDiffFit ID0000013
mDiffFit ID0000011
mDiffFit ID0000010
mDiffFit ID0000015
mDiffFit ID0000014
diff.000001.000002.fits
diff.000001.000002.txt
diff.000001.000003.fits
diff.000001.000003.txt
diff.000002.000003.fits
diff.000002.000003.txt
diff.000003.000005.txt
diff.000003.000005.fits
diff.000003.000006.fits
diff.000003.000006.txt
diff.000003.000004.txt
diff.000003.000004.fits
diff.000002.000005.fits
diff.000002.000005.txt
fitslist.tbl
diff.000005.000006.txt
diff.000005.000006.fits
diff.000004.000006.txt
diff.000004.000006.fits
mConcatFit ID0000016
pimages.tbl
fits.tbl
mBgModel ID0000017
corrections.tbl
mBackground ID0000018
mBackground ID0000019
mBackground ID0000020
mBackground ID0000022
mBackground ID0000023
mBackground ID0000021
c2mass-atlas-990502s-j1420198.fits
c2mass-atlas-990502s-j1420186.fits
c2mass-atlas-990502s-j1430080.fits
c2mass-atlas-990502s-j1430092.fits
c2mass-atlas-990502s-j1440186.fits
c2mass-atlas-990502s-j1440198.fits
cimages.tbl
mAddMem ID0000024
mosaic.fits
mShrink ID0000025
shrunken.fits
mViewer ID0000026
mosaic.png


0002.000003.fits
diff.000002.000003.txt
diff.000003.000005.txt
diff.000003.000005.fits
diff.000003.000006.fits
diff.000003.000006.txt
diff.000003.000004.txt
diff.000003.000004.fits
mConcatFit
ID0000016
pimages.tbl
fits.tbl
mBgModel
ID0000017
corrections.tbl
mBackground
ID0000018
mBackground
ID0000019
mBackground
ID0000020
mBackground
ID0000022
mBackground
ID0000023
c2mass-atlas-990502s-j1420198.fits
c2mass-atlas-990502s-j1420186.fits
c2mass-atlas-990502s-j1430080.fits
c2mass-atlas-990502s-j1430092.fits
c2mass-atlas-990502s-j1440186.fits
mAddMem
ID0000024
mosaic.fits
mShrink
ID0000025



2.5 Summary



3. Advanced Topics

Data Staging Configurations

HTCondor I/O (HTCondor pools, OSG, …)

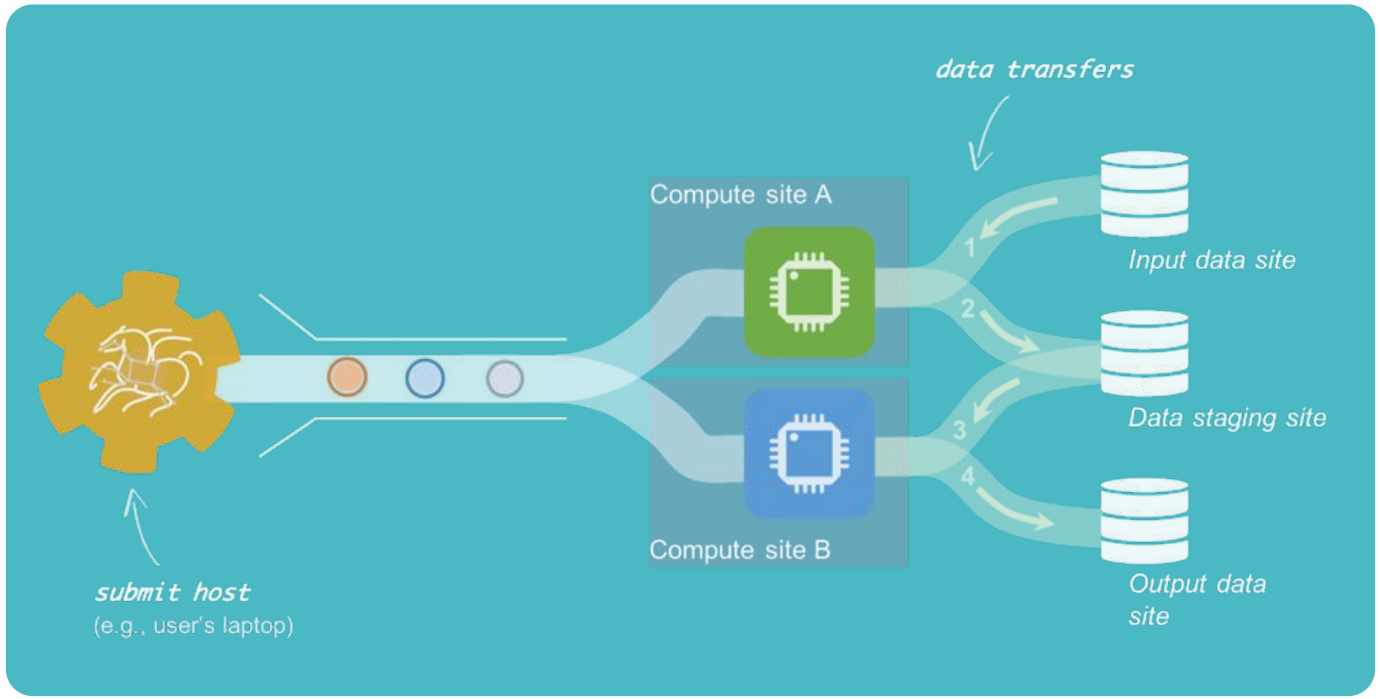
- Worker nodes do not share a file system
- Data is pulled from / pushed to the submit host via HTCondor file transfers
- Staging site is the submit host

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

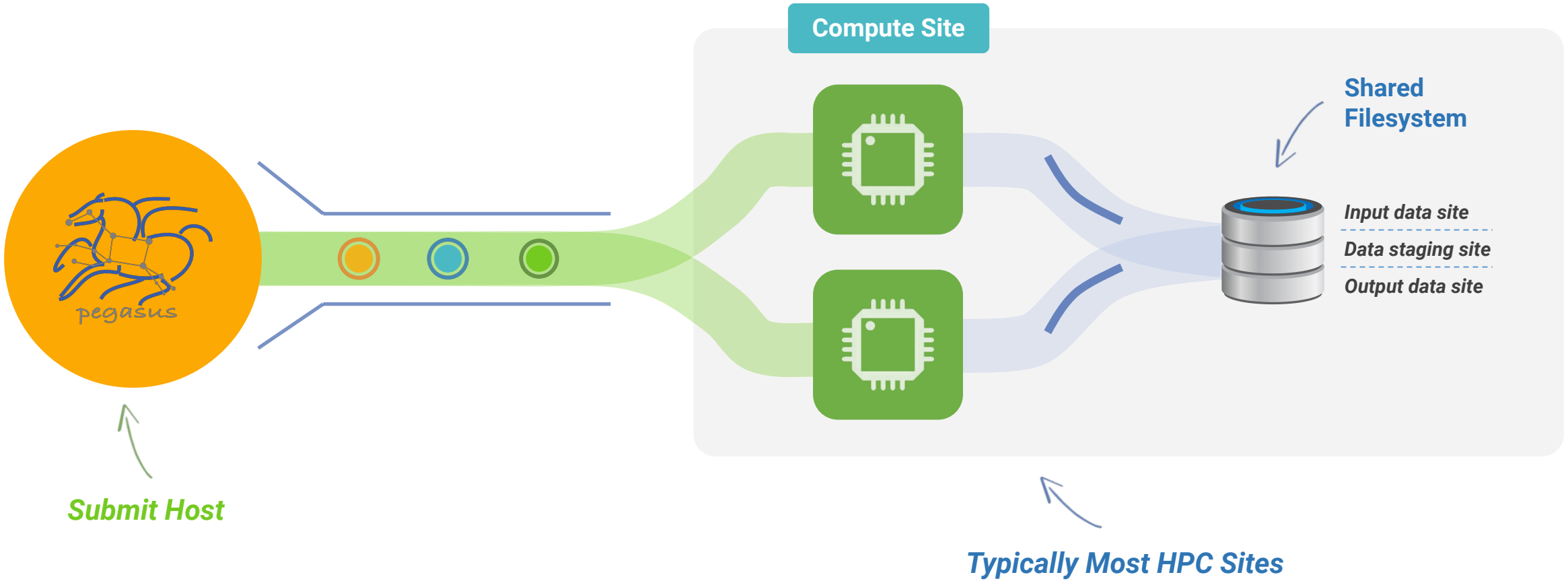
Shared File System (HPC sites, XSEDE, Campus clusters, …)

- I/O is directly against the shared file system



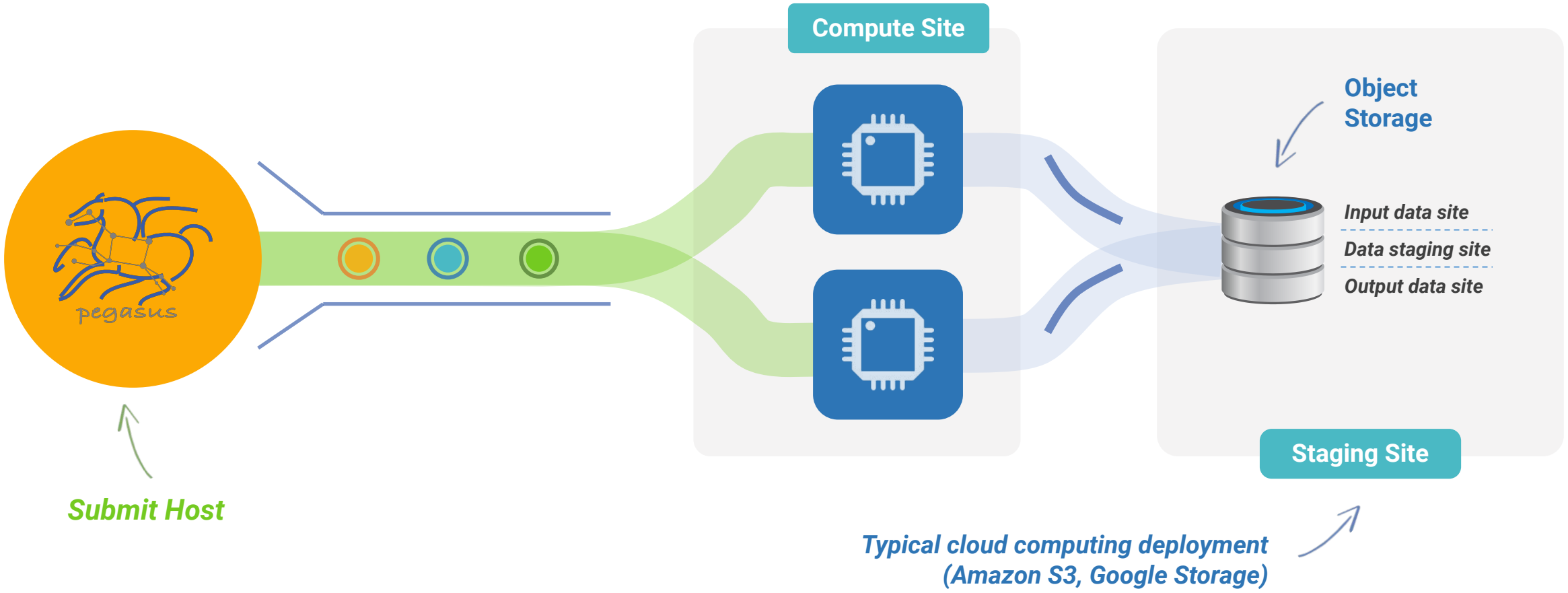
High Performance Computing

There are several possible configurations…



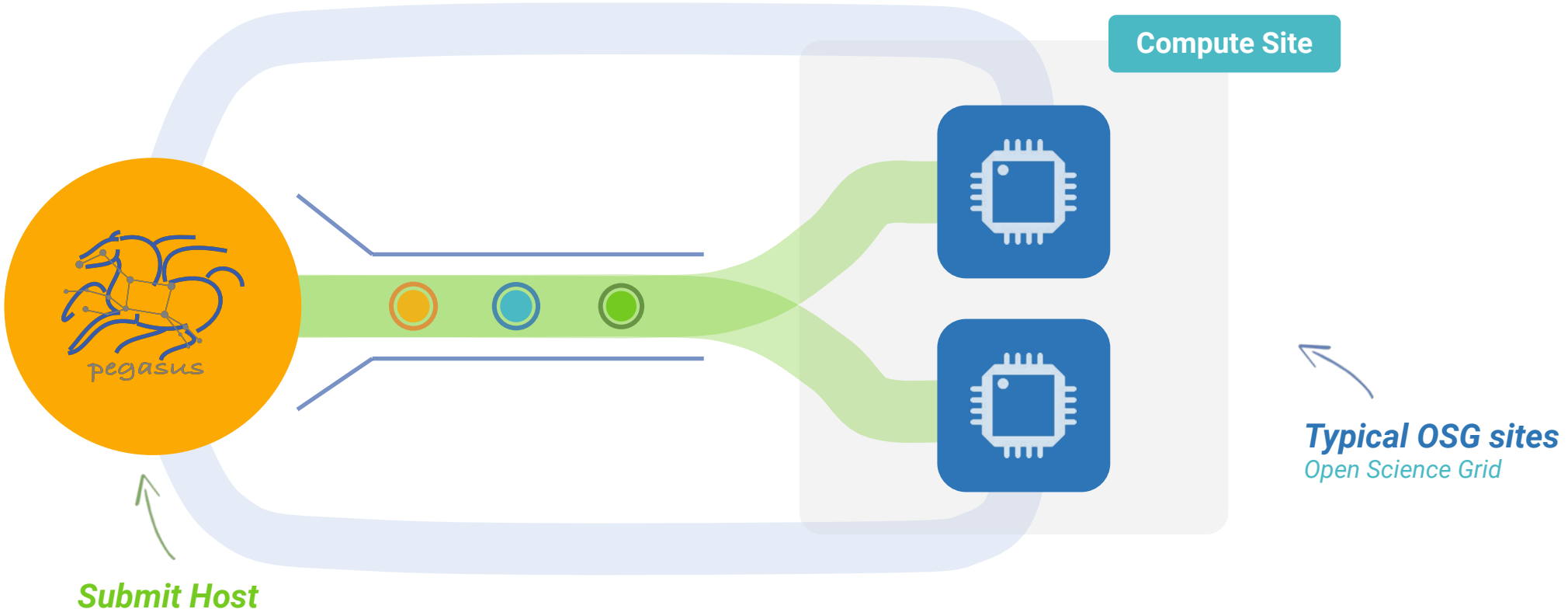
Cloud Computing

High-scalable object storages

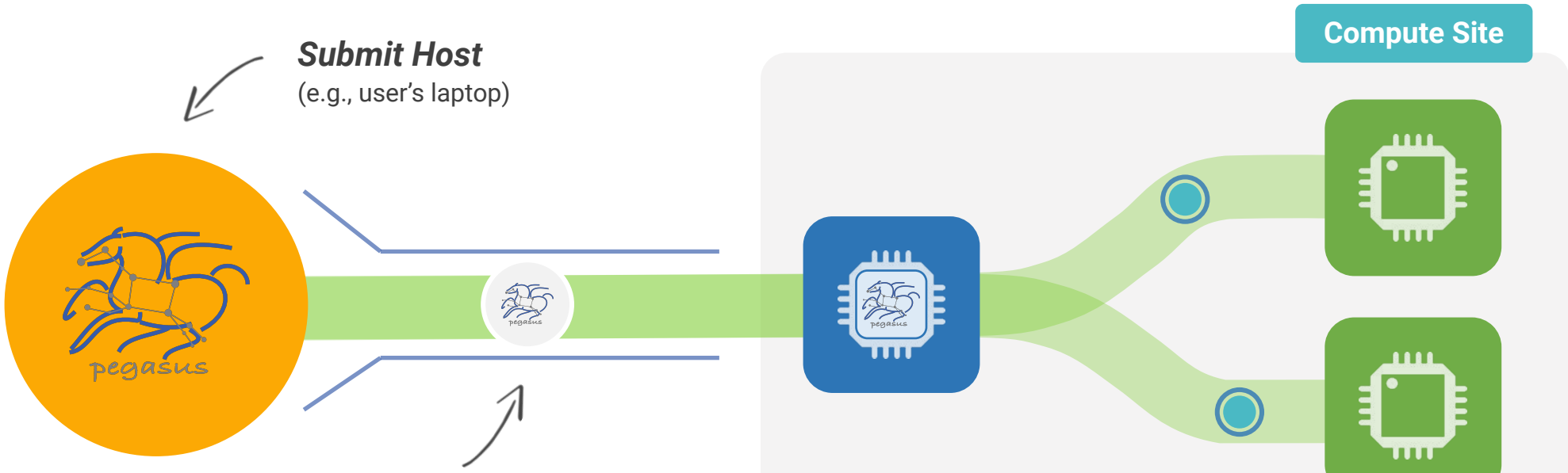


Grid Computing

Local data management

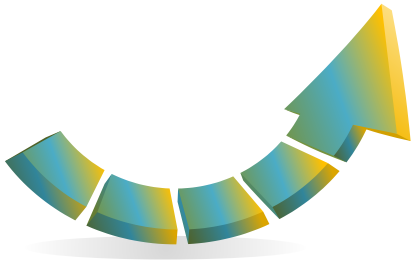


Running fine-grained workflows on HPC systems…

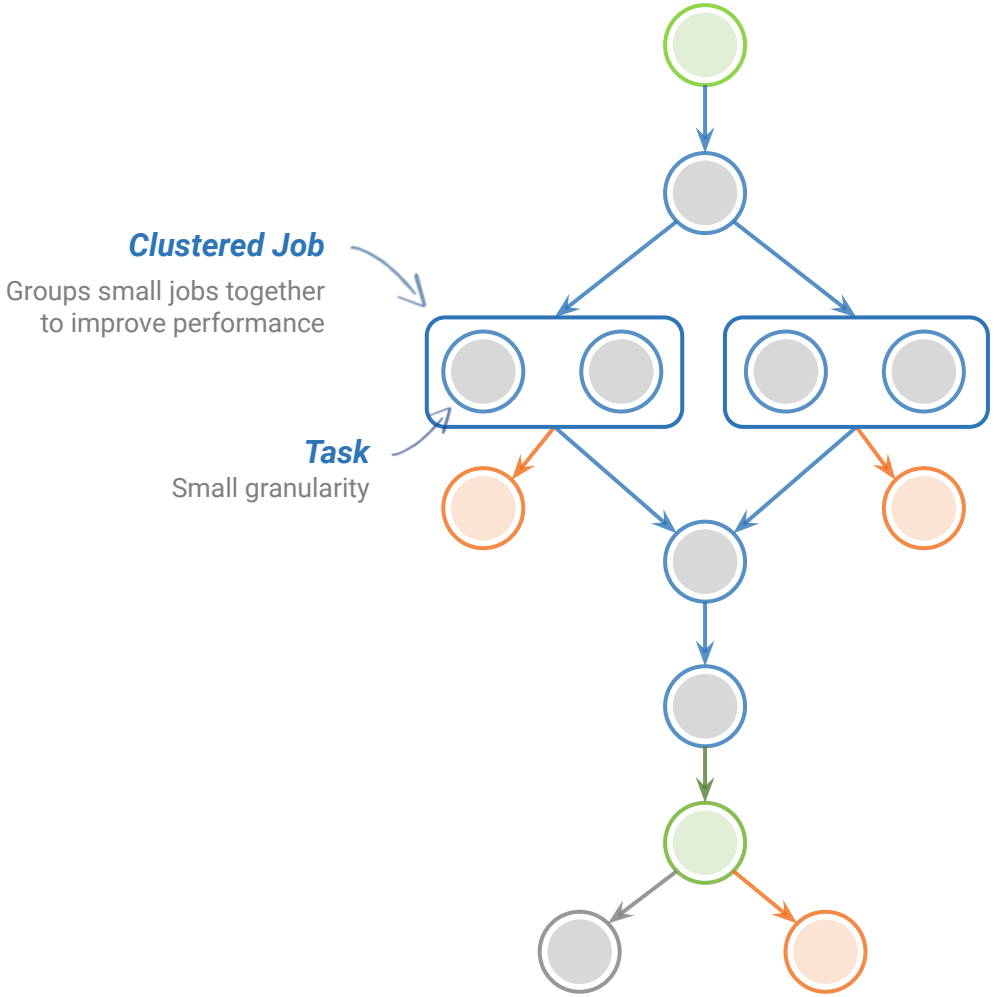


Workflow wrapped as an MPI job

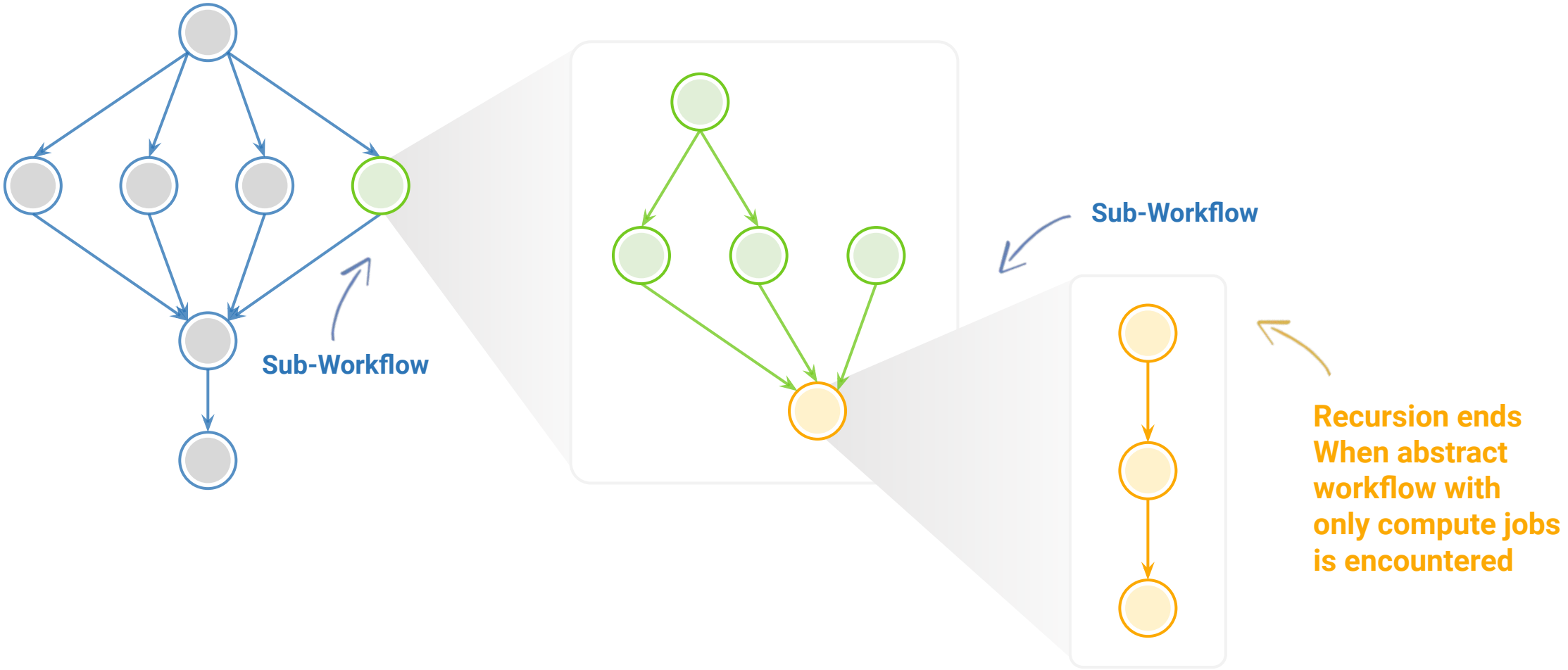
Allows sub-graphs of a Pegasus workflow to be submitted as monolithic jobs to remote resources



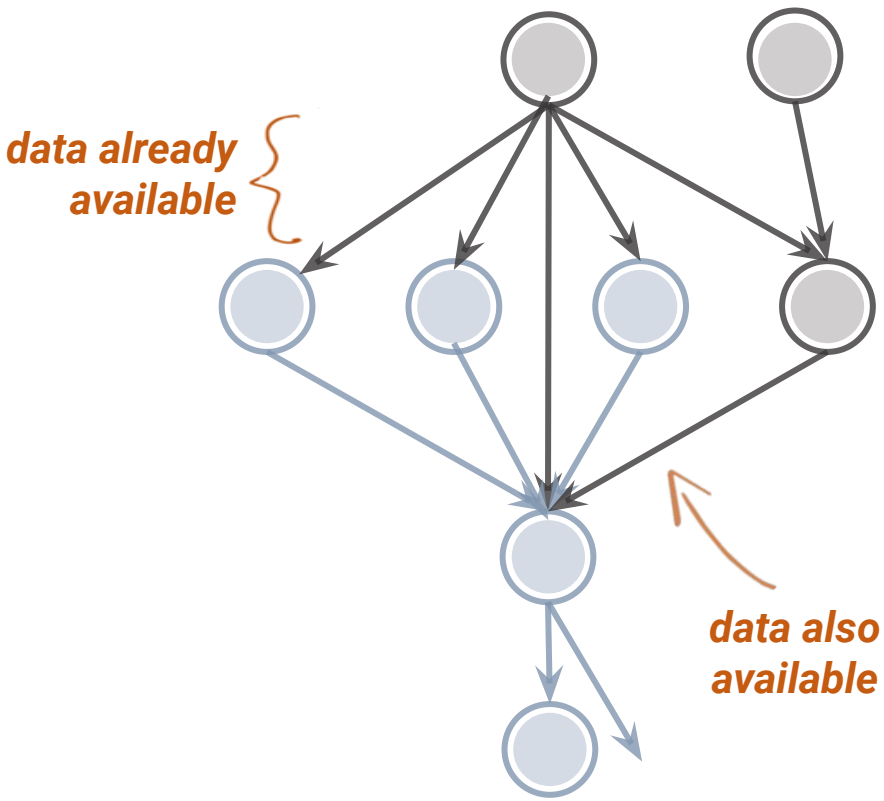
Performance.
Why not improve it?



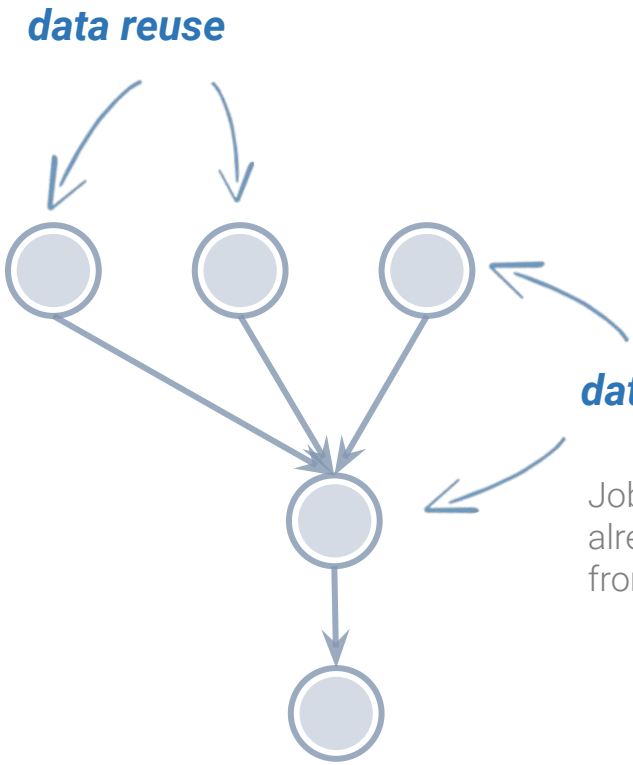
Pegasus also handles large-scale workflows



Data Reuse prune jobs if output data already exists



workflow reduction



Jobs which output data is already available are pruned from the DAG

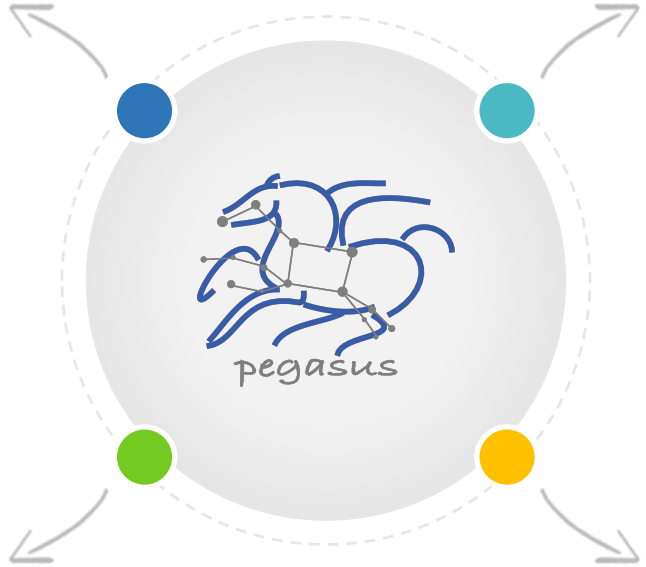
And if a job fails?

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

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

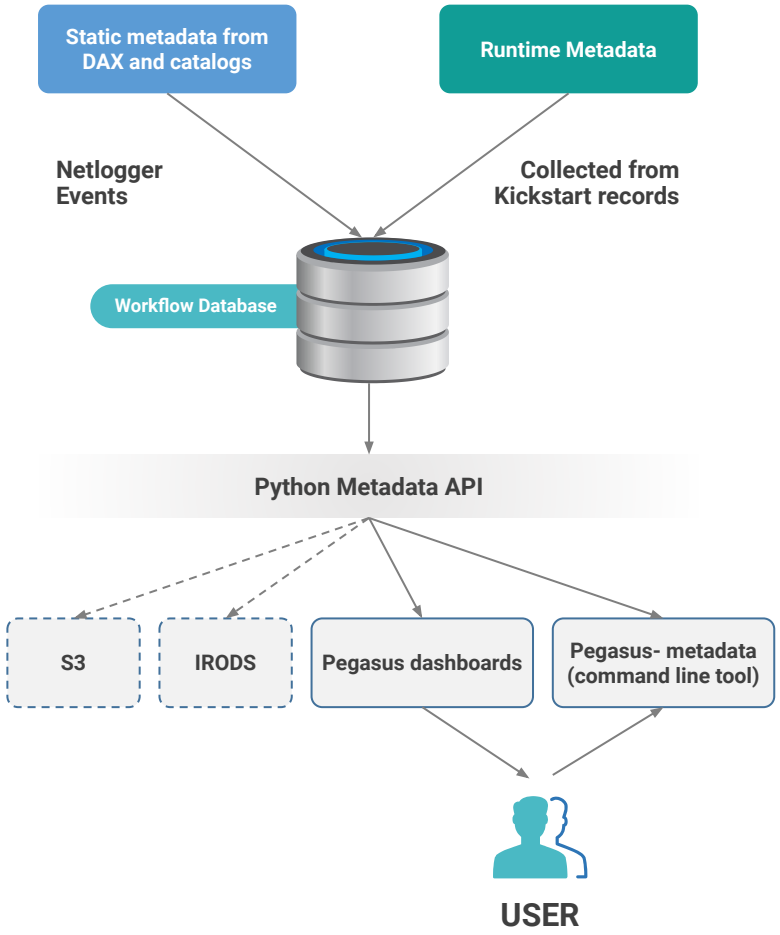
Static: application parameters
Runtime: performance metrics

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



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.

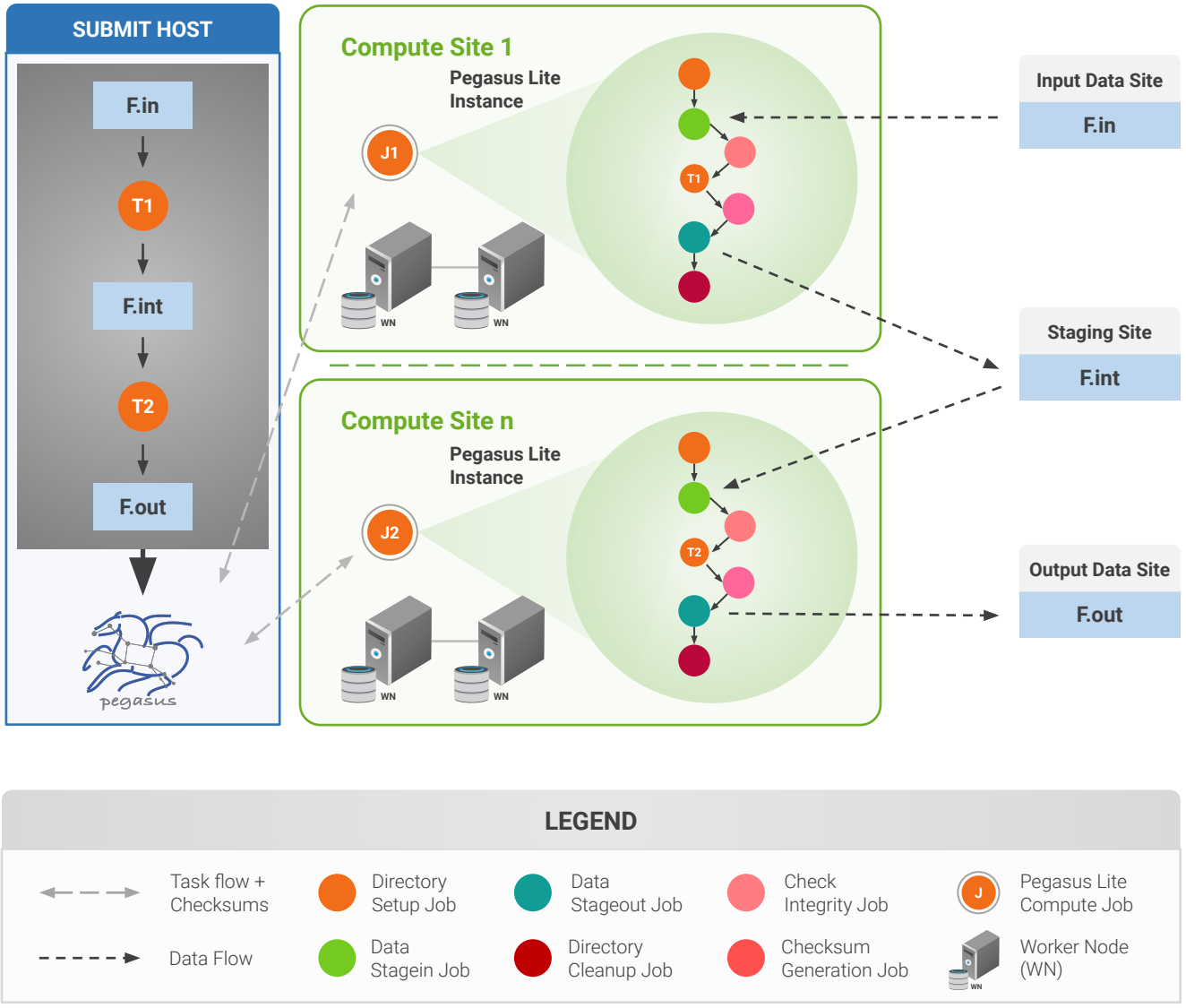
User Perception: "Am I not already protected? I have heard about TCP checksums, encrypted transfers, checksum validation, RAID and erasure coding – is that not enough?"

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



Job Submissions

LOCAL

Submit Machine
Personal HTCondor

Local Campus Cluster accessible via Submit Machine **
HTCondor via BLAHP

*** Both Glite and BOSCO build on HTCondor BLAHP*

Currenty supported schedulers:
SLURM SGE PBS MOAB

REMOTE

BOSCO + SSH**
Each node in executable workflow submitted via SSH connection to remote cluster

BOSCO based Glideins**
SSH based submission of glideins

PyGlidein
IceCube glidein service

OSG using glideinWMS
Infrastructure provisioned glideins

CREAMCE
Uses CondorG

Globus GRAM
Uses CondorG

HTCondor Annex / Pilot Jobs

- **A pilot 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 pilot 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 pilot will only run jobs for the user who started it.**

Pegasus

est. 2001

Automate, recover, and debug scientific computations.

Get Started

Pegasus Website
https://pegasus.isi.edu

Users Mailing List
pegasus-users@isi.edu

Support
pegasus-support@isi.edu

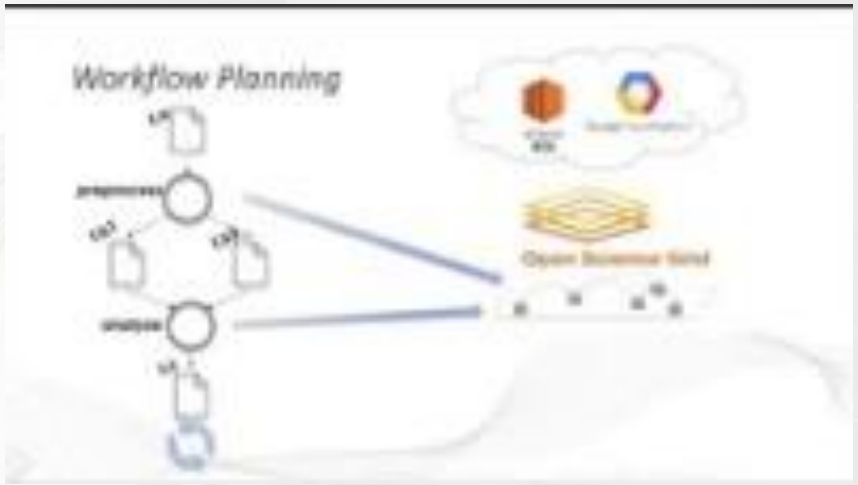
Slack
Ask for an invite by trying to join pegasus-users.slack.com in the Slack app

Pegasus Online Office Hours
https://pegasus.isi.edu/blog/online-pegasus-office-hours/

Bi-monthly basis on second Friday of the month, where we address user questions and also apprise the community of new developments

YouTube Channel

https://www.youtube.com/channel/UCwJQln1CqBvTJqiNr9X9F1Q/featured



Pegasus in 5 Minutes