2023 SUMMER BOOT CAMP

Scientific Data Processing with Pegasus Workflows

pegasus

USC | Advanced Research Computing
Enabling scientific breakthroughs at scale

Karan Vahi [1], Mats Rynge [1], Tomasz Osinski [2]

[1]Information Sciences Institute, University of Southern California
[2]USC Center for Advanced Research Computing (CARC)

vahi@isi.edu , rynge@isi.edu , osinski@usc.edu

NSF U.S. DEPARTMENT OF ENERGY NIH DARPA

1. Introduction

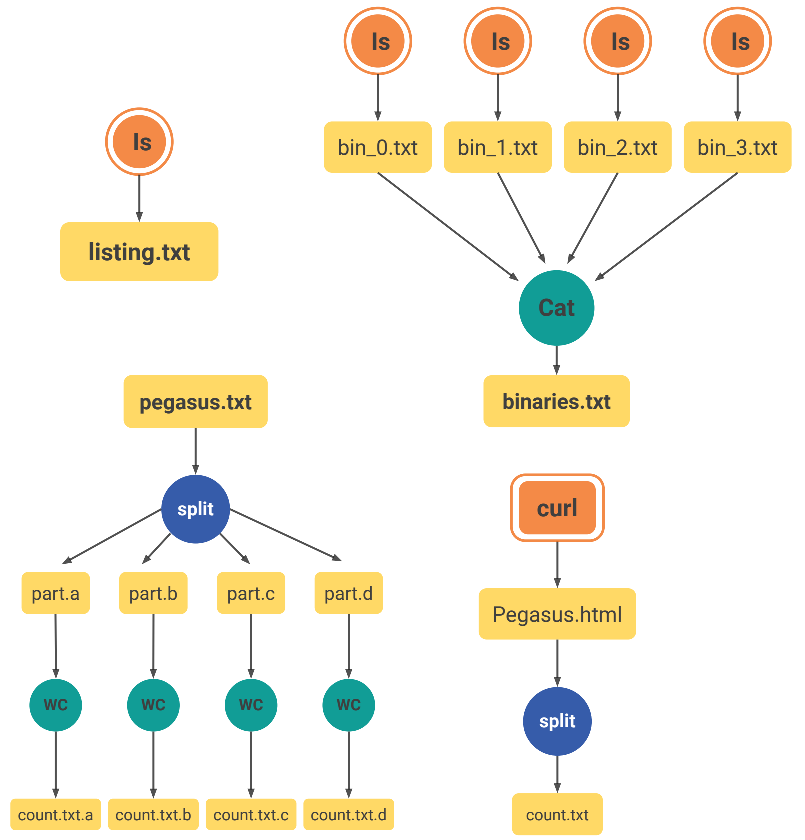
Workflow Systems and USC CACR / HPC?

- We will talk about:
  - Multiple job workloads
  - Relationship between jobs
  - Automatic data management
  - … and more
- HPC is not just parallel jobs
  - High throughput computing (HTC)

What are Scientific Workflows

- **Conducts a series of computational tasks.**
  - Resources distributed across Internet.
- **Chaining (outputs become inputs) replaces manual hand-offs.**
  - Accelerated creation of products.
- **Ease of use - gives non-developers access to sophisticated codes.**
  - Resources distributed across Internet.
- **Provides framework to host or assemble community set of applications.**
  - Honors original codes. Allows for heterogeneous coding styles.
- **Framework to define common formats or standards when useful.**
  - Promotes exchange of data, products, codes. Community metadata.
- **Multi-disciplinary workflows can promote even broader collaborations.**
  - E.g., ground motions fed into simulation of building shaking.
- **Certain rules or guidelines make it easier to add a code into a workflow.**

Workflow Building Blocks



Slide Content Courtesy of David Okaya, SCEC, USC

Why Pegasus?

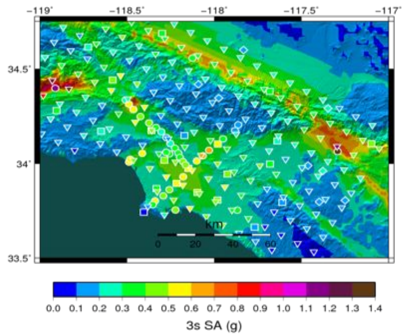
- **Automates Complex**, Multi-stage Processing Pipelines
- Enables Parallel, **Distributed Computations**
- **Automatically Executes** Data Transfers
- Reusable, Aids **Reproducibility**
- Records How Data was Produced **(Provenance)**
- Handles **Failures** with to Provide Reliability
- Keeps Track of Data and **Files**
- Ensures **Data Integrity** during workflow execution

HTCondor High Throughput Computing

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

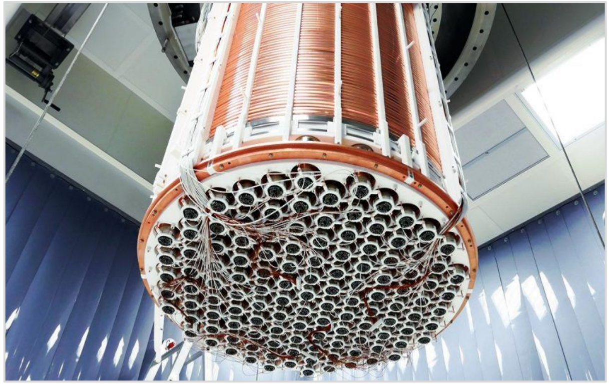
Laser Interferometer Gravitational-Wave Observatory (LIGO)

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.

Southern California Earthquake Center's CyberShake

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

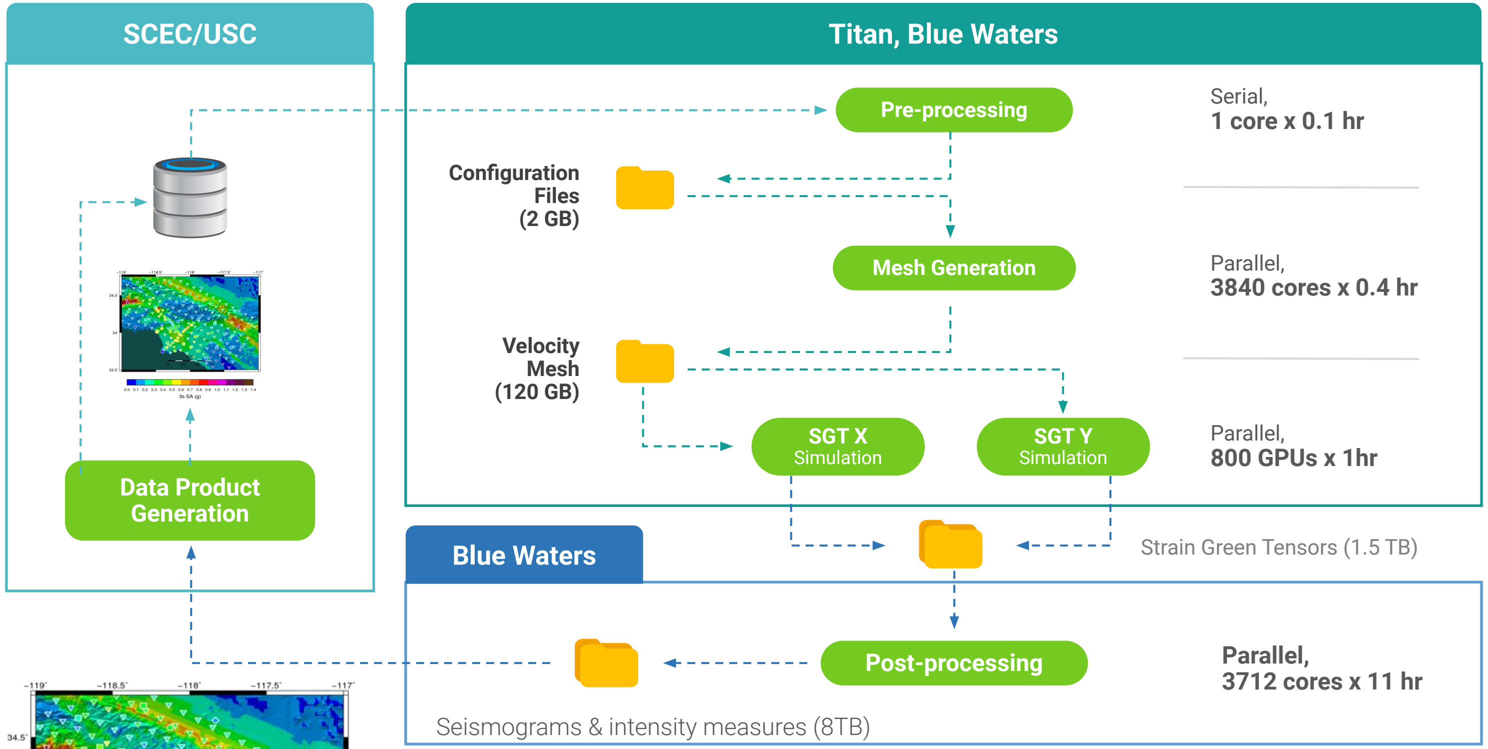
Seismologists answer this question
using Probabilistic Seismic Hazard Analysis (PSHA)

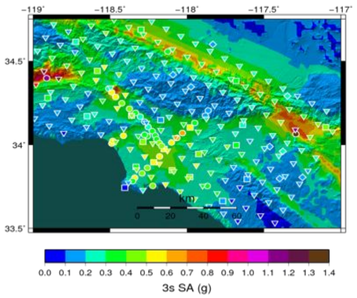
CPU jobs
(Mesh generation, seismogram synthesis)
1,094,000 node-hours

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

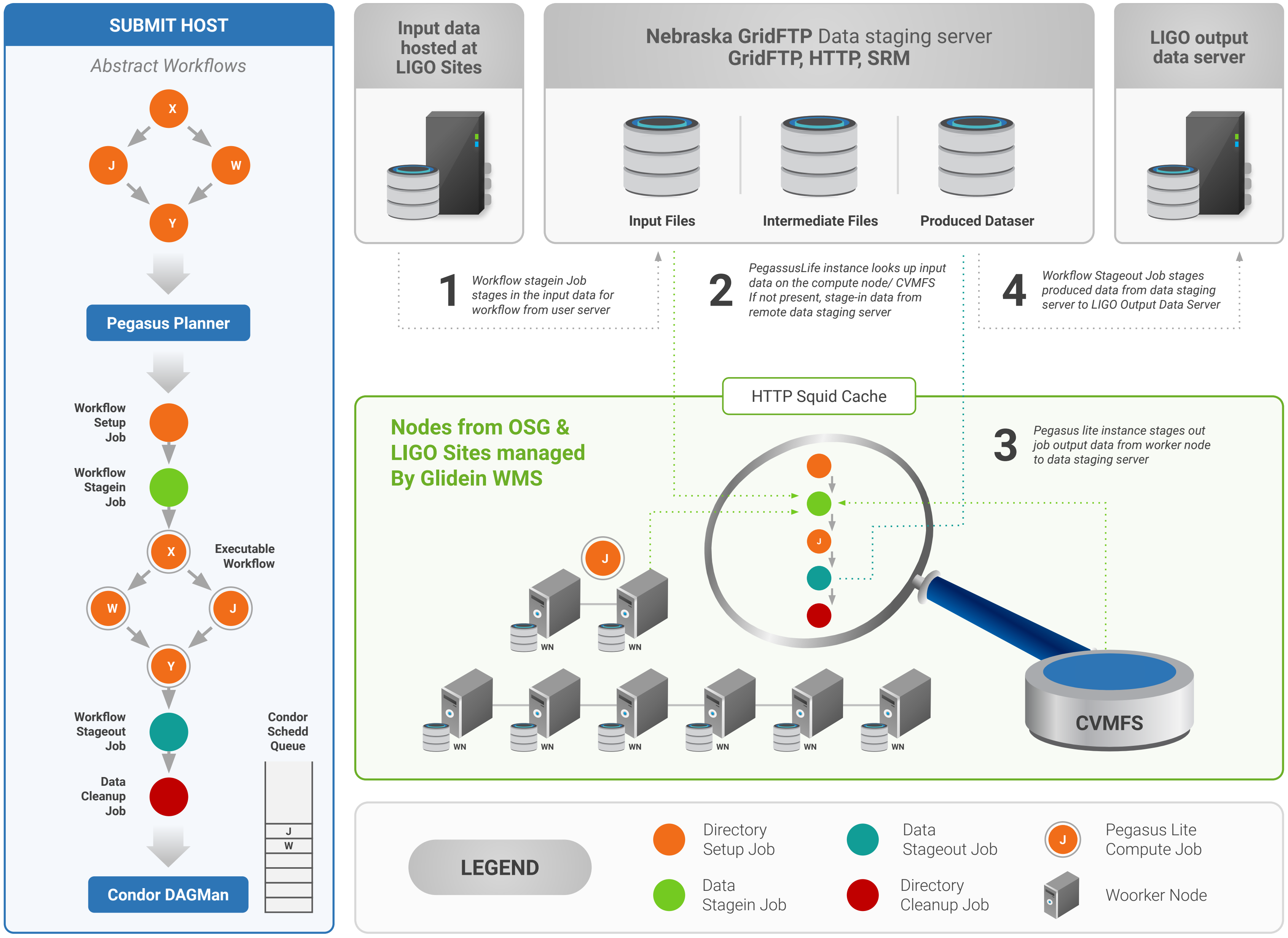




286 Sites

4 Models

Each Workflow Has
420,000 Tasks



Data Flow for **LIGO** **Pegasus Workflows** in OSG

Advanced LIGO

Laser **I**nterferometer **G**ravitational Wave **O**bservatory

60,000 Compute Tasks

Input Data: 5000 files (10GB total)

Output Data: 60,000 files (60GB total)

Processed Data: 725 GB

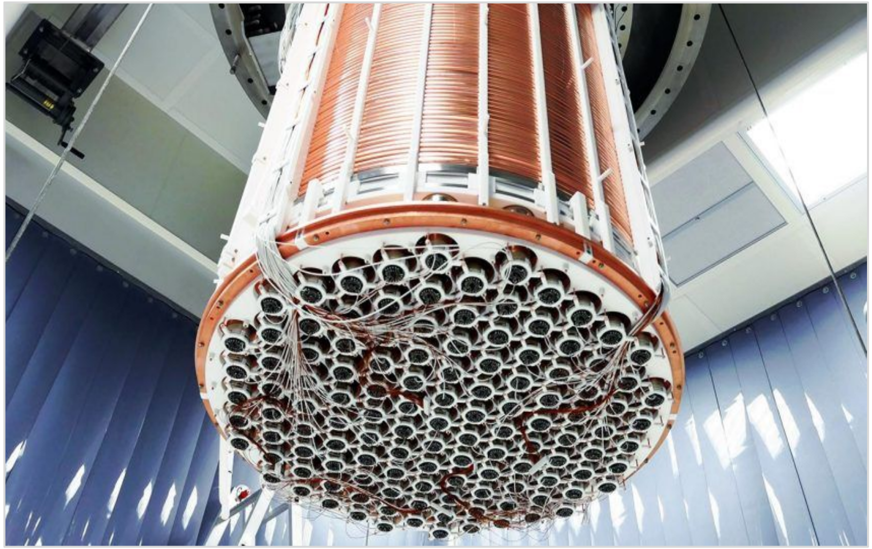
Executed on **LIGO Data Grid, EGI, Open Science Grid and XSEDE**

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.

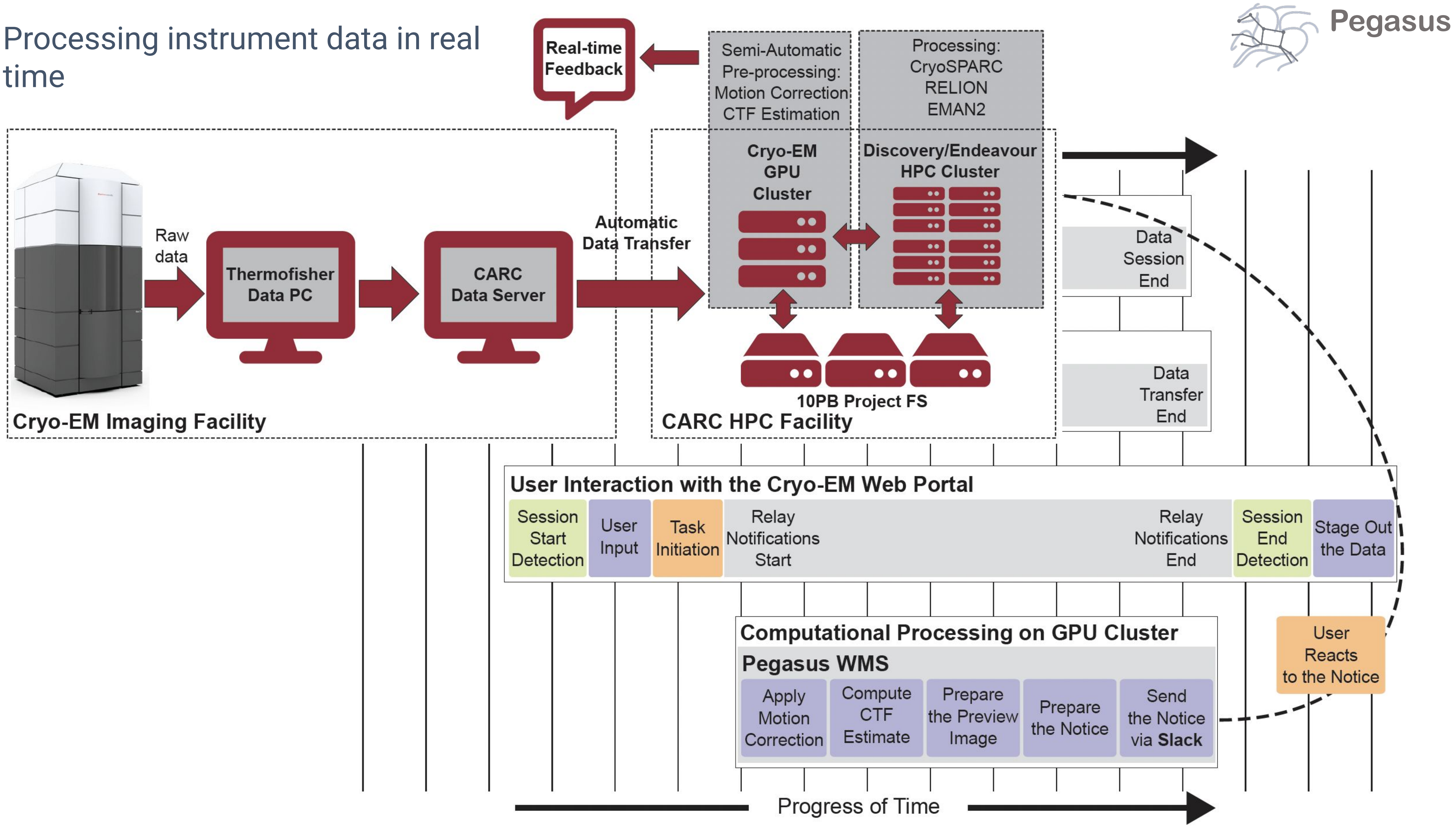


Type	Succeeded	Failed	Incomplete	Total	Retries	Total+Retries
Tasks	4000	0	0	4000	267	4267
Jobs	4484	0	0	4484	267	4751
Sub-Workflows	0	0	0	0	0	0

```
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:

Processing instrument data in real time



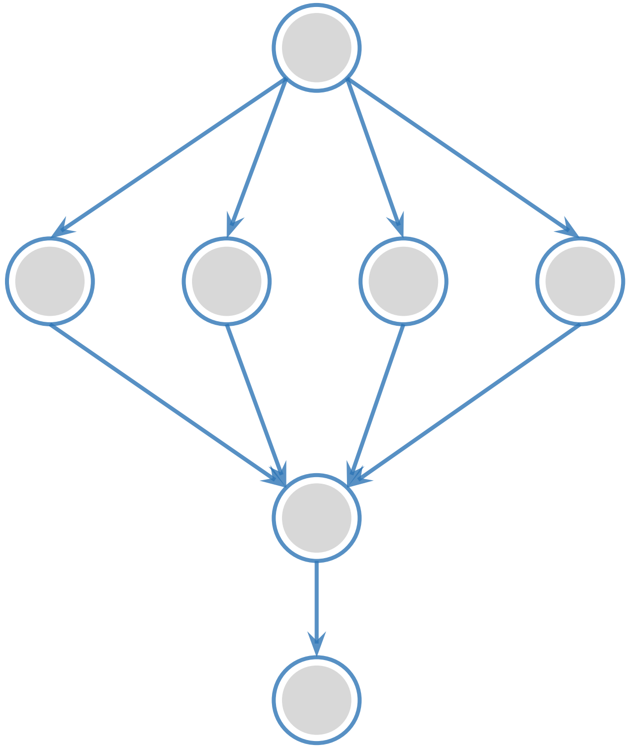
Key Pegasus Concepts

- **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



Pegasus

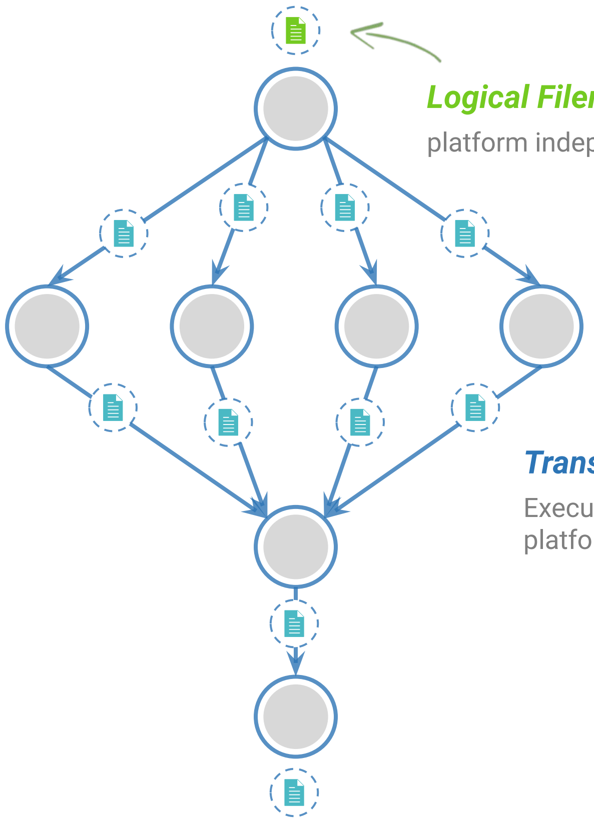
Input Workflow Specification **YAML formatted**

directed-acyclic graphs Output Workflow

Portable Description

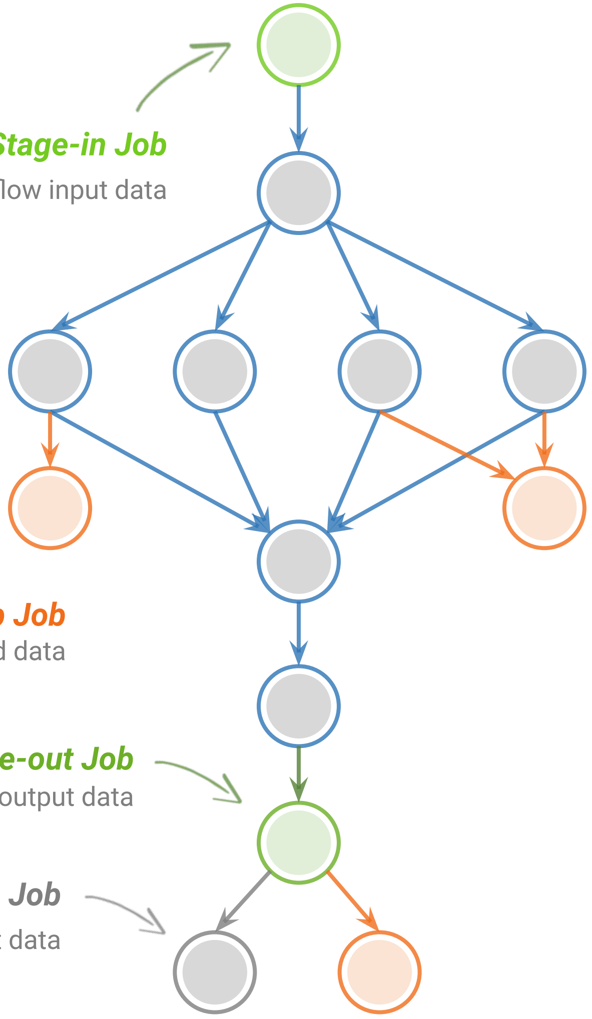
Users do not worry about low level execution details

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

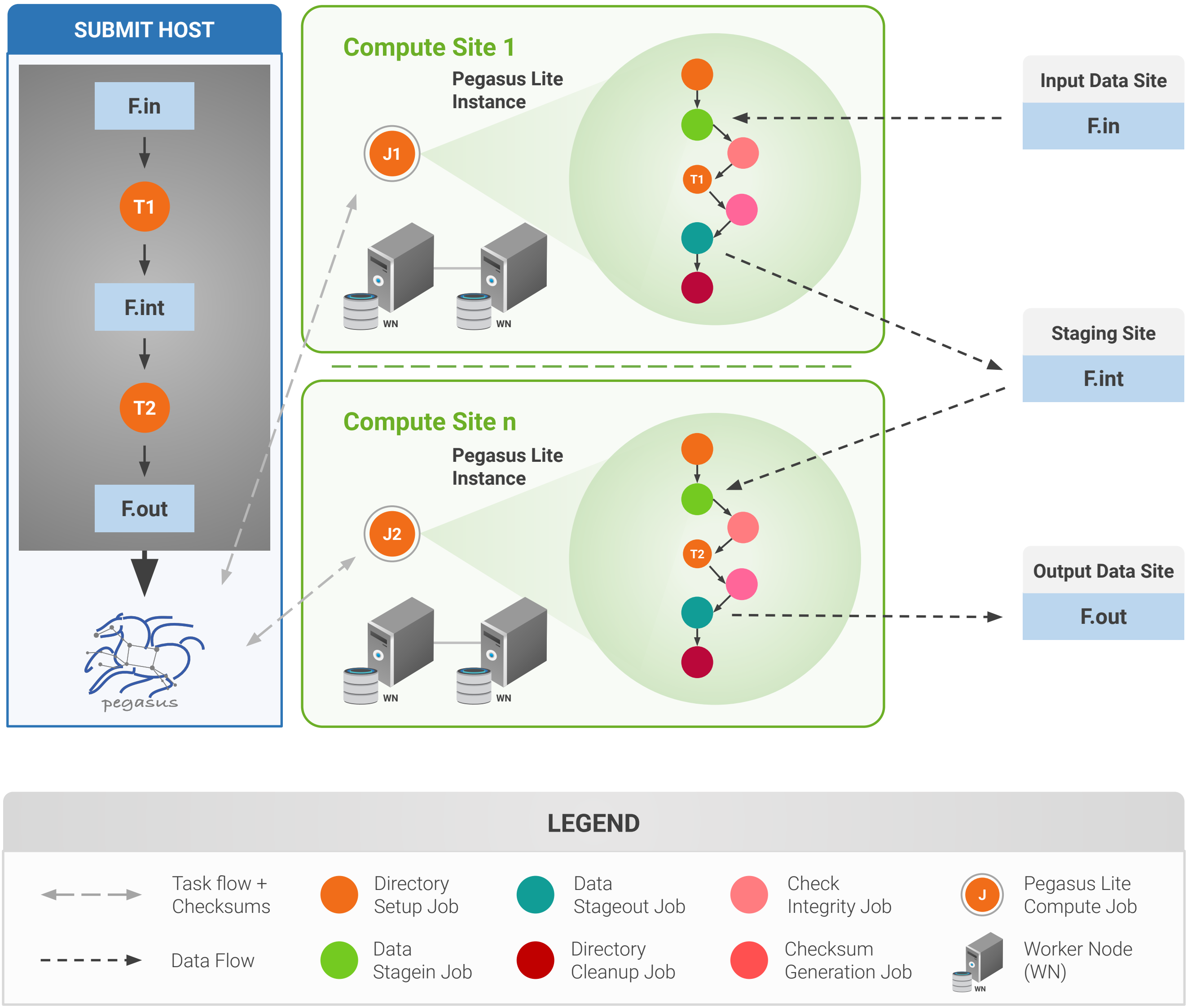
Stage-out Job
Stage-out generated output data

Registration Job
Registers the workflow output data

EXECUTABLE WORKFLOW

Pegasus 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

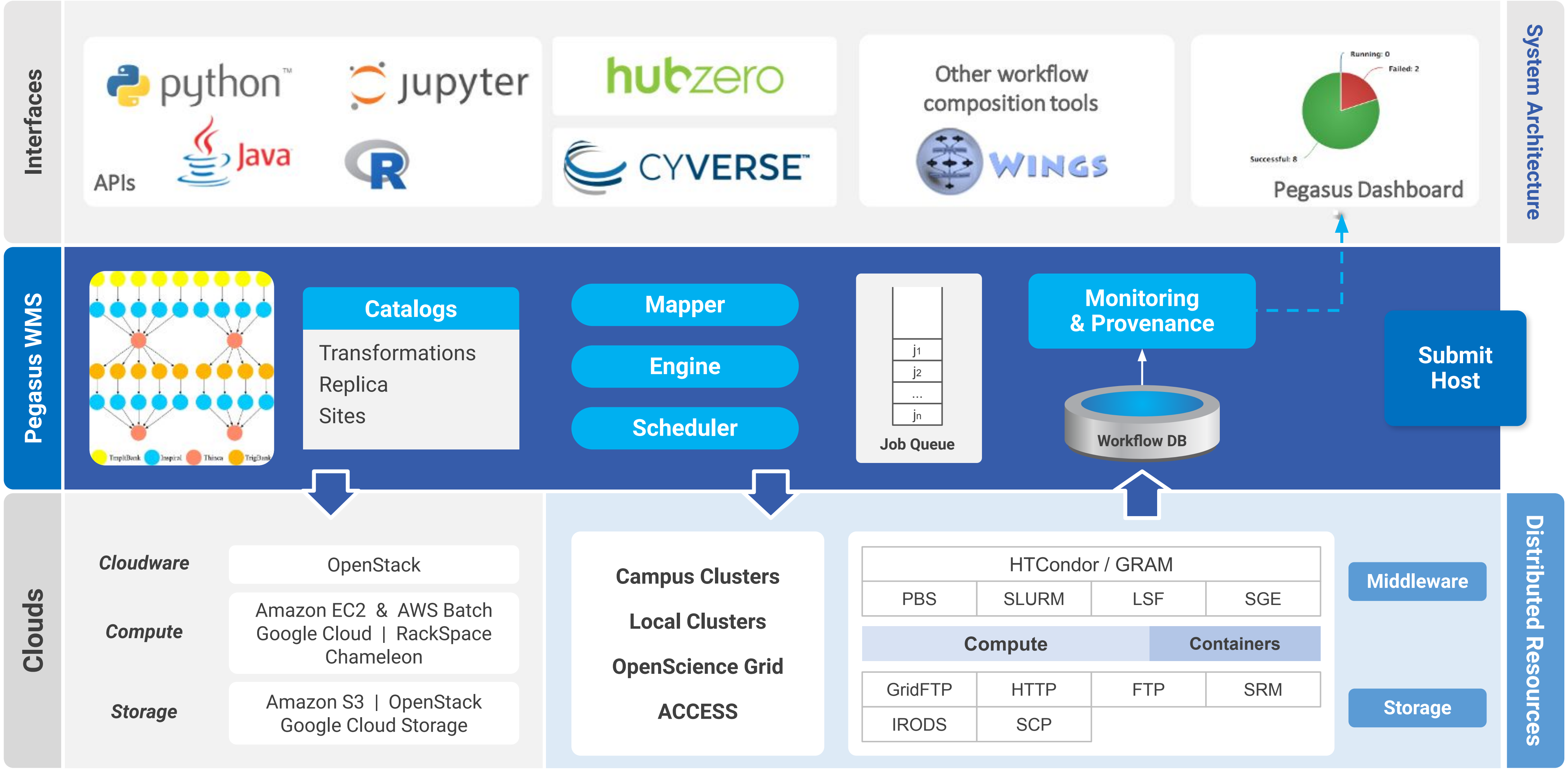


Pegasus-transfer

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)

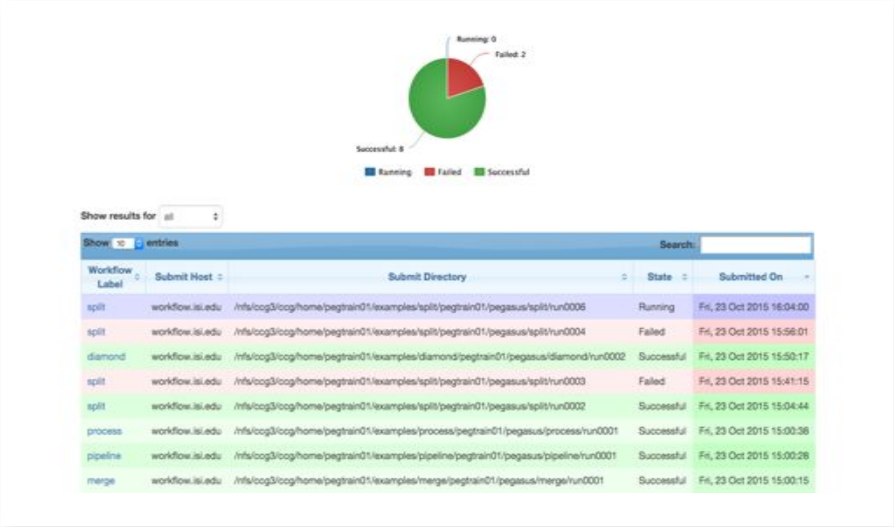
```
HTTP
SCP
GridFTP
Globus
Online
iRods
Amazon S3
Google
Storage
SRM
FDT
Stashcp
Rucio
cp
ln -s
```


Interfaces
APIs
python
jupyter
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

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

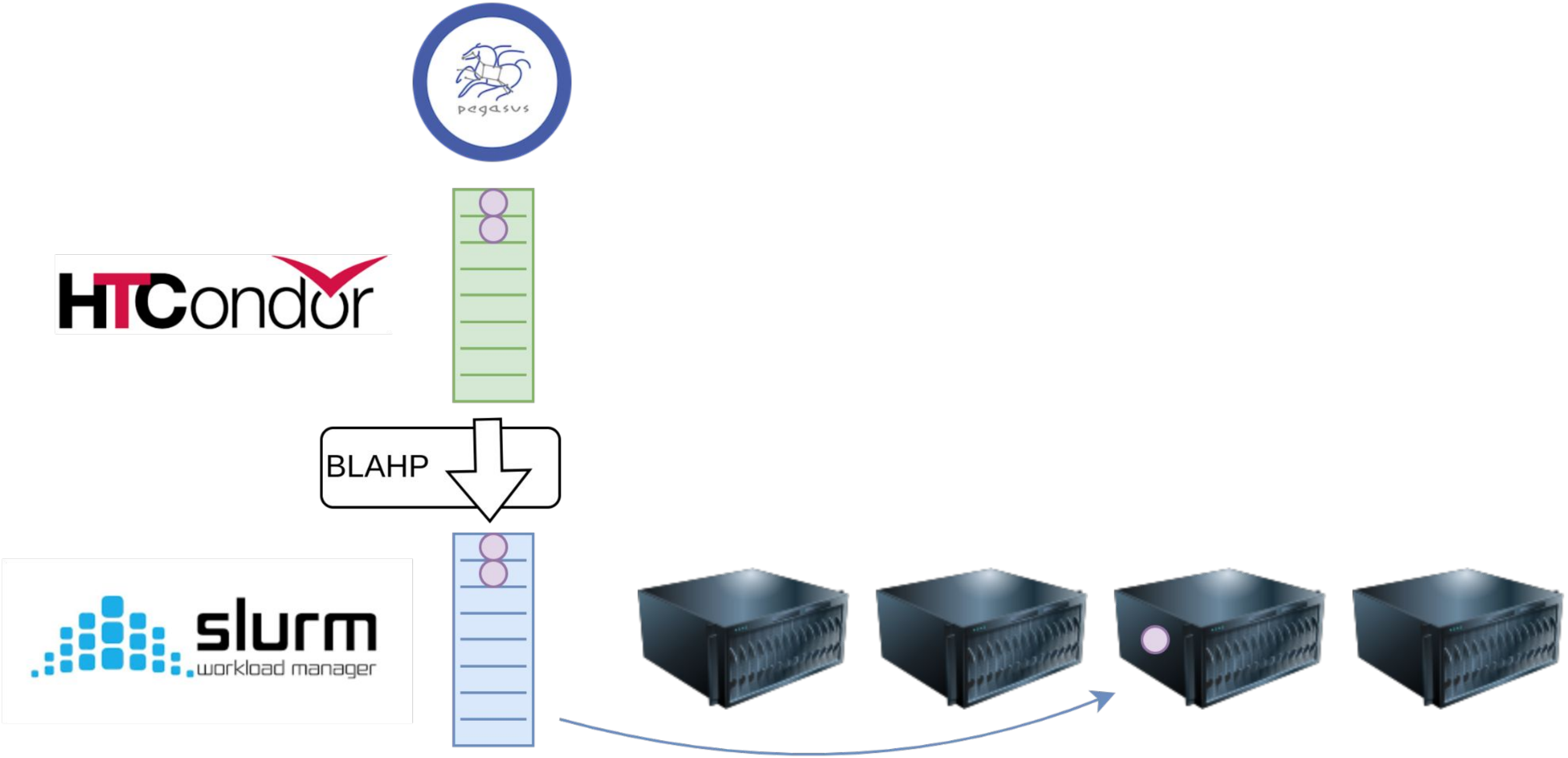
Checkpoint Files

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

HTCondor with BLAHP translation layer

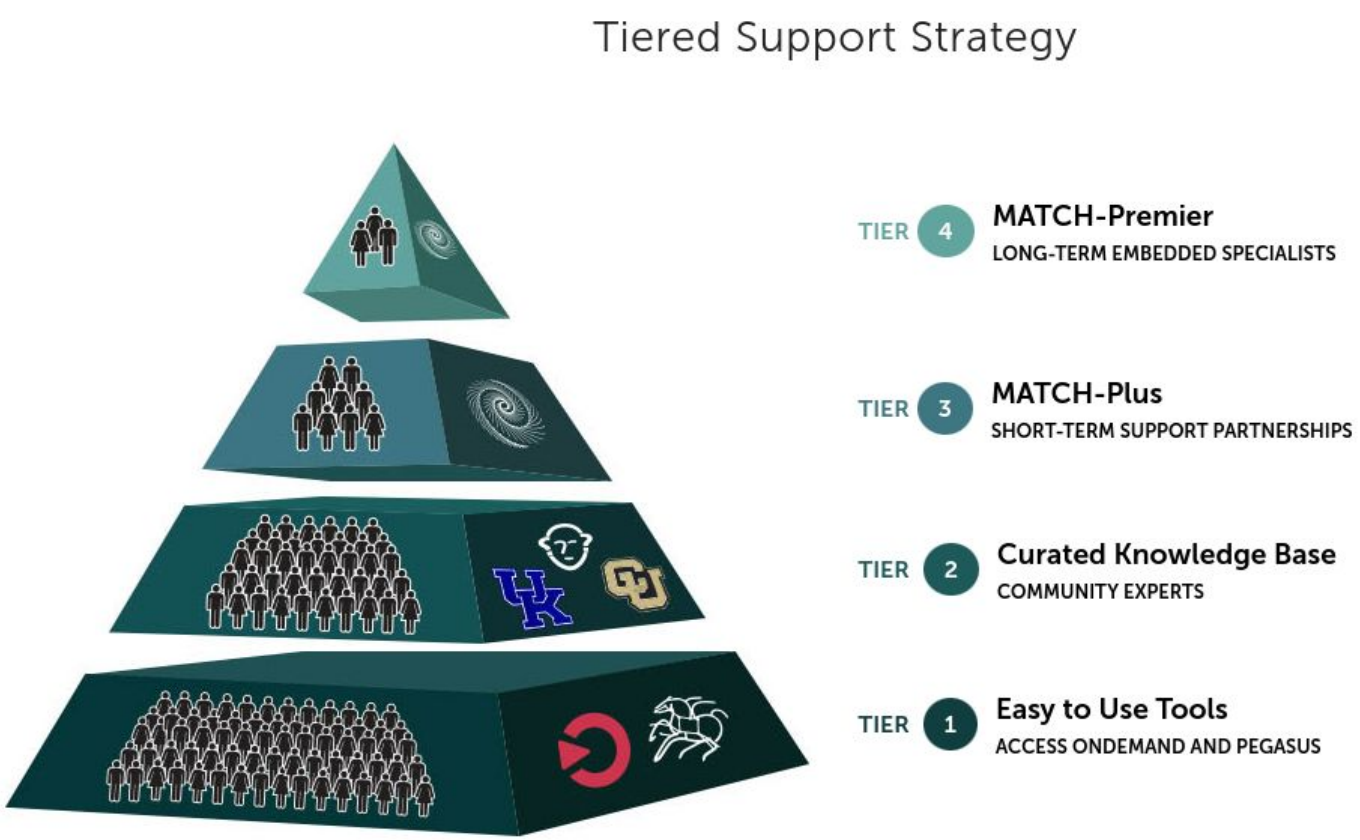


Pegasus is part of the ACCESS support strategy

Pegasus is be used as a tier 1 tool

Central Open OnDemand instance with Pegasus, HTCondor and Jupyter

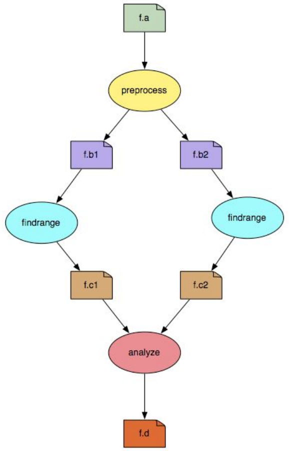
It is be easy to run HTC workflows across ACCESS sites

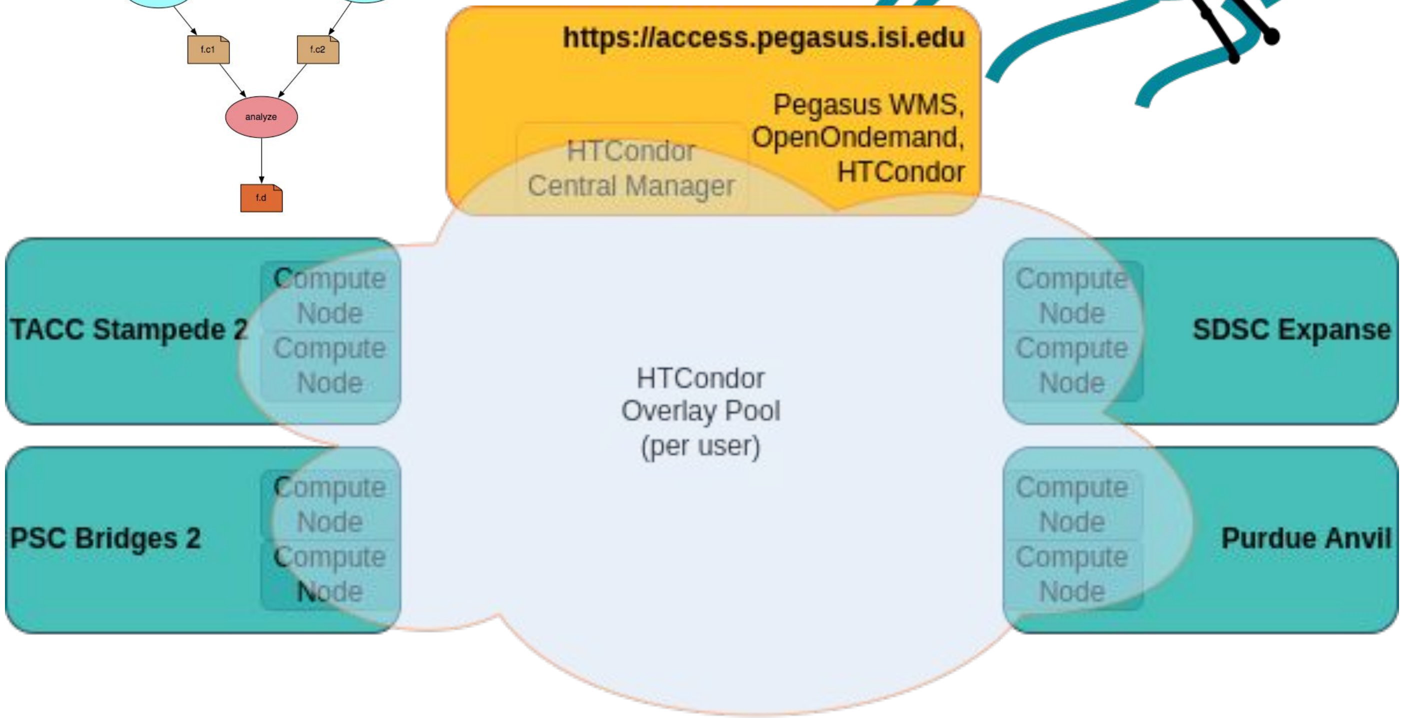


ACCESS Pegasus

Bring your workflows to ACCESS!

- **Execute scientific workflows across ACCESS resources**
- OpenOnDemand Portal: **has all you need**: Jupyter Notebooks, ACCESS authentication, Pegasus workflow management, and HTCondor job management
- **Bring your own ACCESS capacity**: HTCondor Annex - pilot jobs automatically create a virtual HTCondor pool





https://access.pegasus.isi.edu

More at: support.access-ci.org/pegasus

pegasus

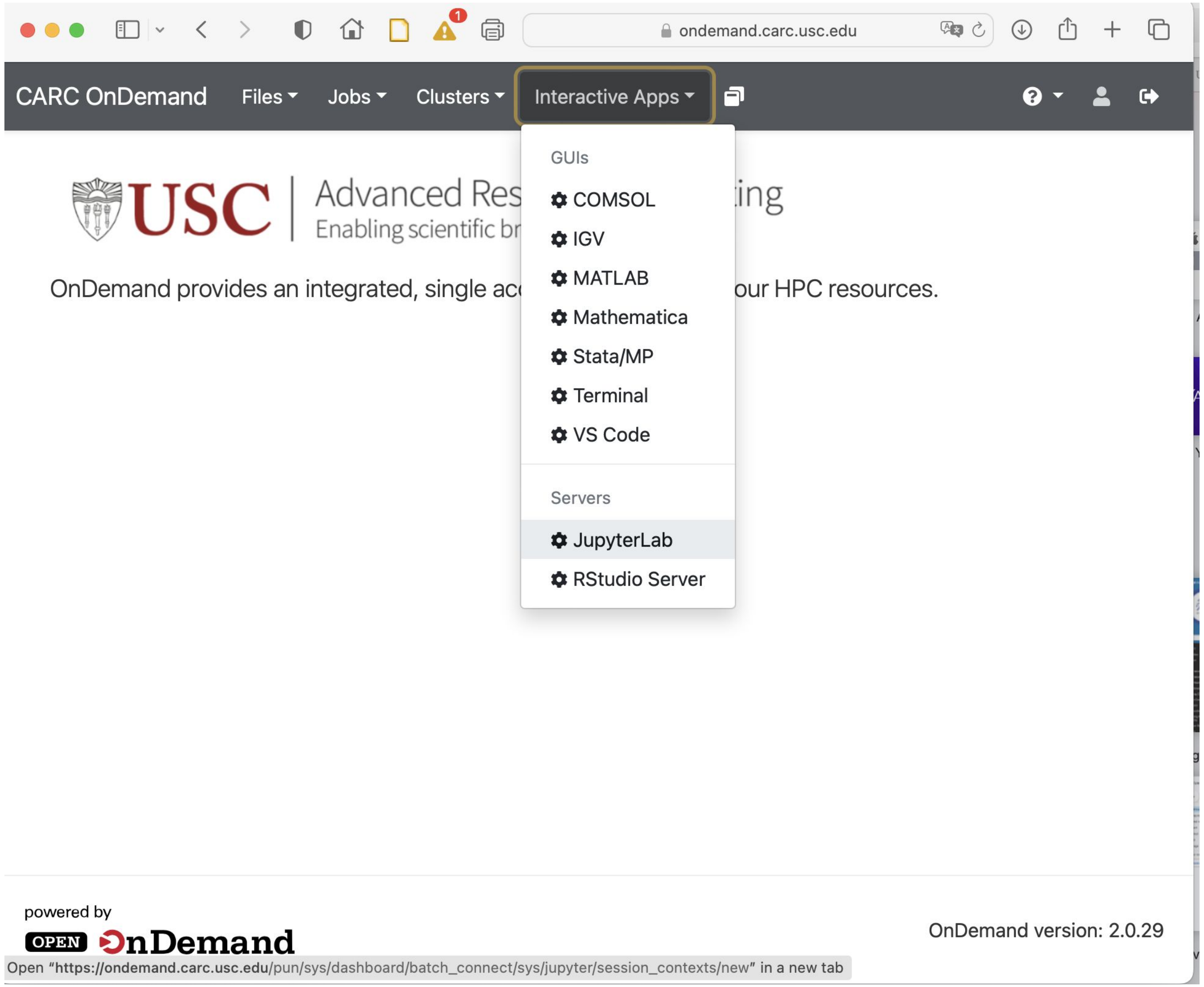
2. Hands on Exercises

Hands on Tutorial Exercises: Login to Open OnDemand

- You need to be on USC Network and need to use your USC credentials to log in
- Use a web browser and log on to USC OnDemand Instance at https://ondemand.carc.usc.edu .

Hands on Tutorial Exercises: Start a Juypter Server

- Start a Jupyter notebook server, Click on Interactive Apps and then select JupyterLab



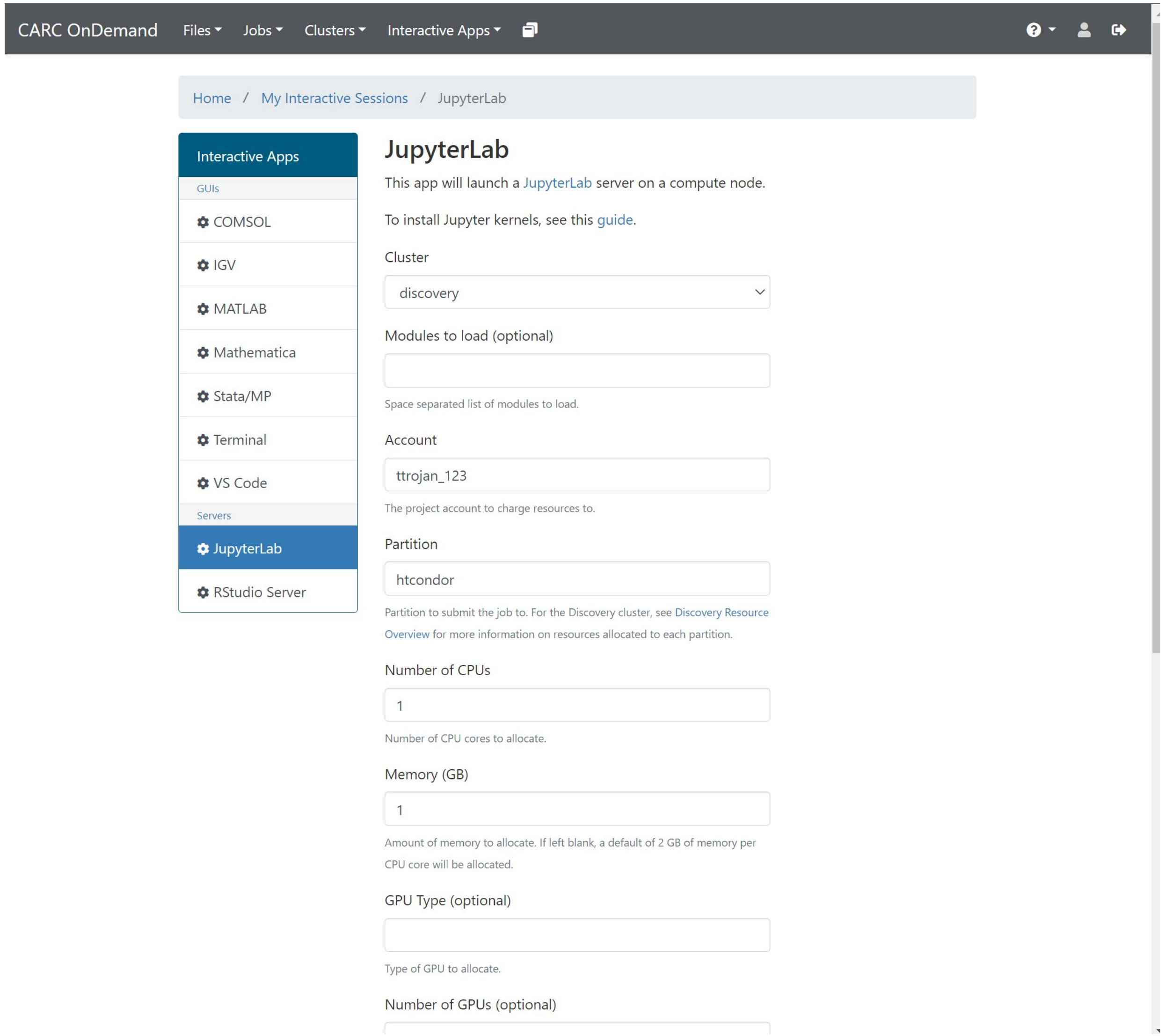
Hands on Tutorial Exercises: Jupyter Lab Configuration

- When launching the Jupyter Lab, it is important to select the following

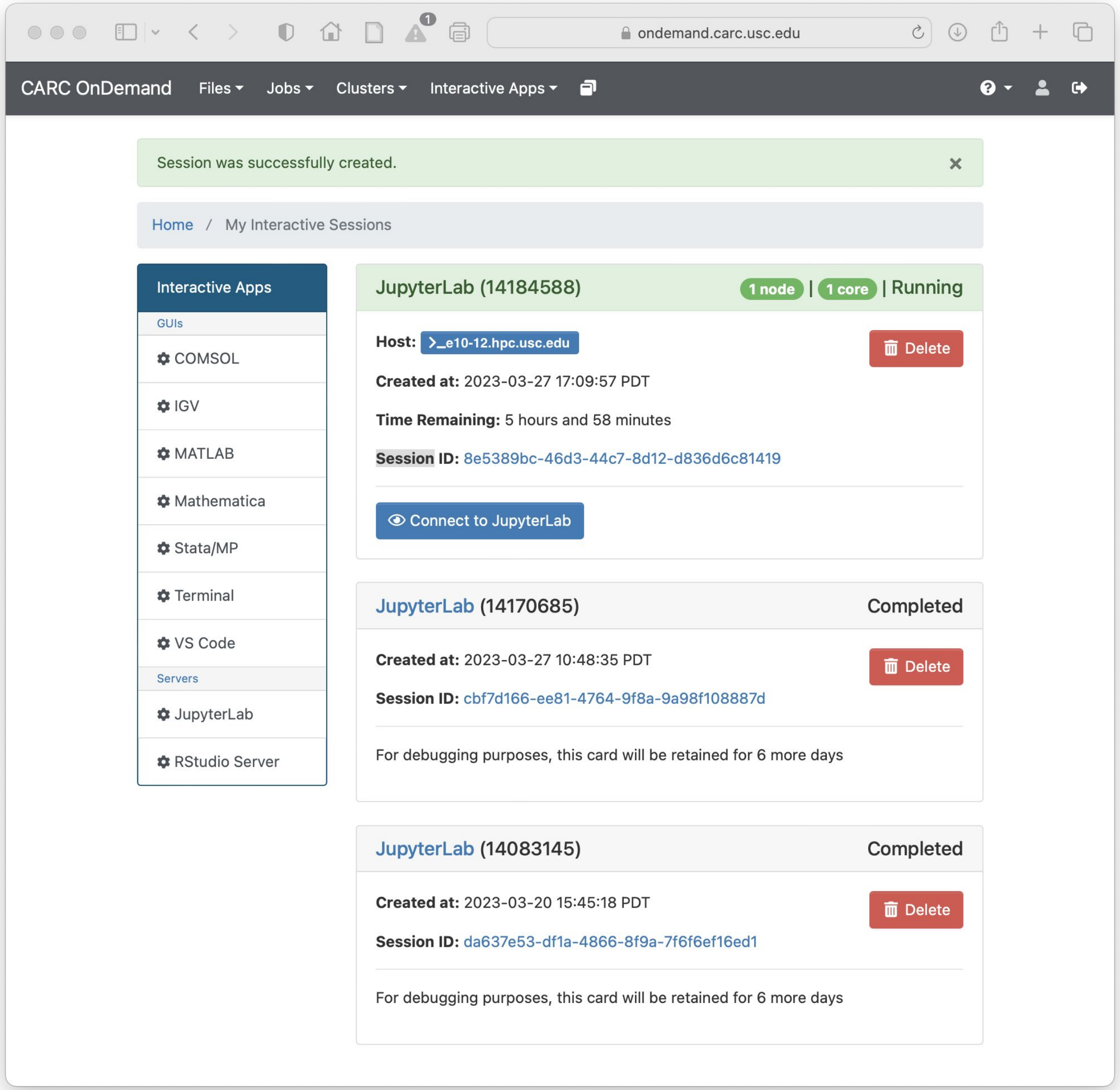
- For Cluster: specify Discovery

- For Account: specify the account ttrojan_123

- For Partition specify htcondor

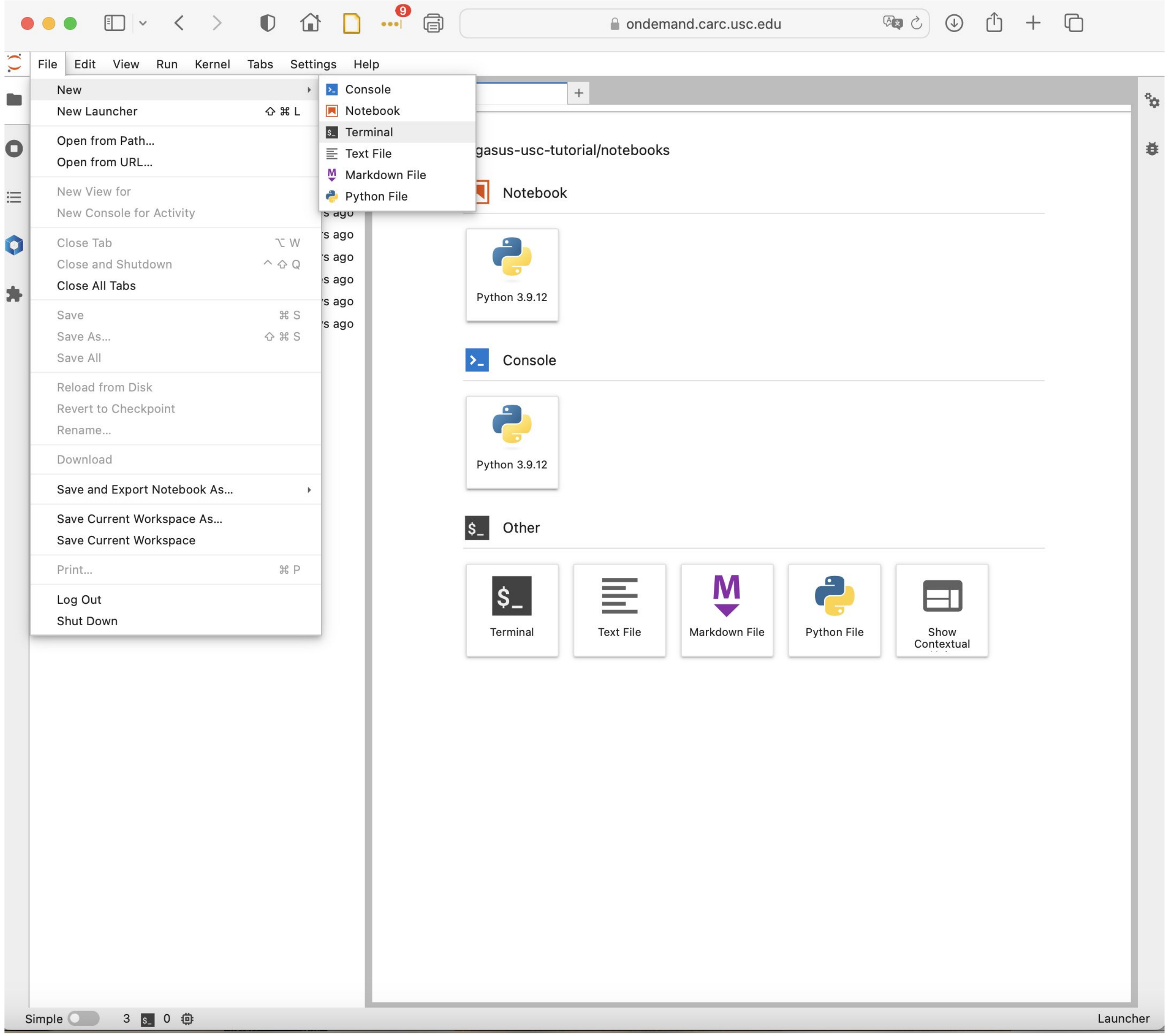


Hands on Tutorial Exercises: Connect to JupyterLab



Hands on Tutorial Exercises: Start a Terminal

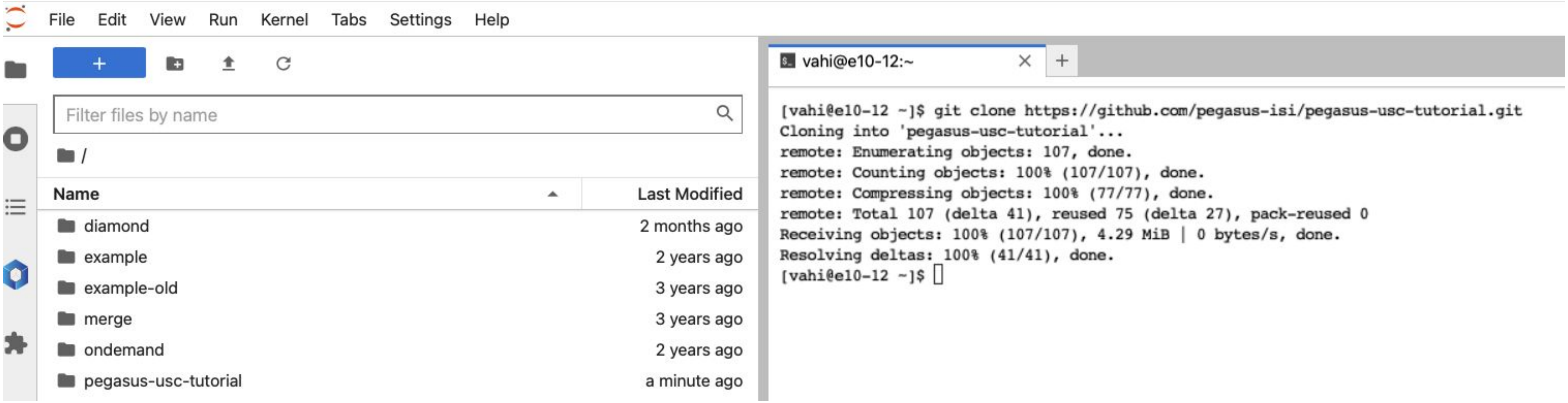
- In JupyterLab, Click on File -> New and then click on Terminal to get the terminal
-



Hands on Tutorial Exercises: Clone Repository

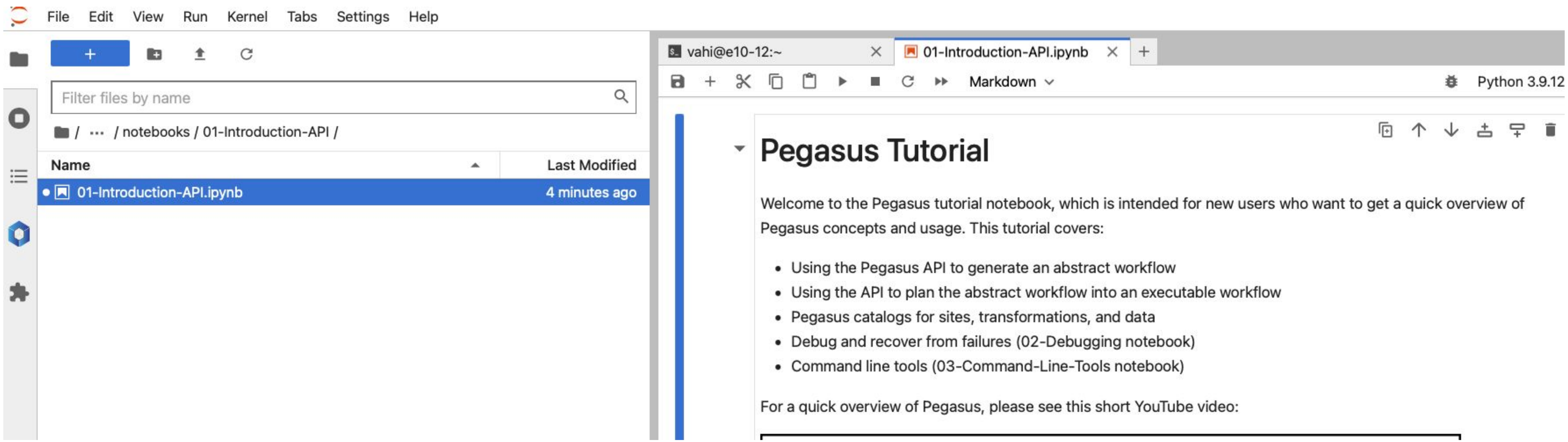
- Clone Tutorial Repository in the terminal

git clone https://github.com/pegasus-isi/pegasus-usc-tutorial.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

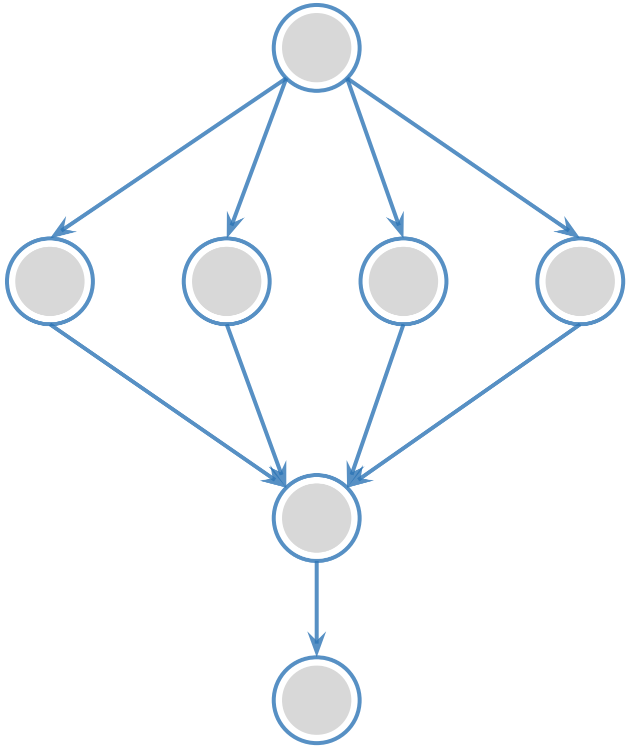
Key Pegasus Concepts

- **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



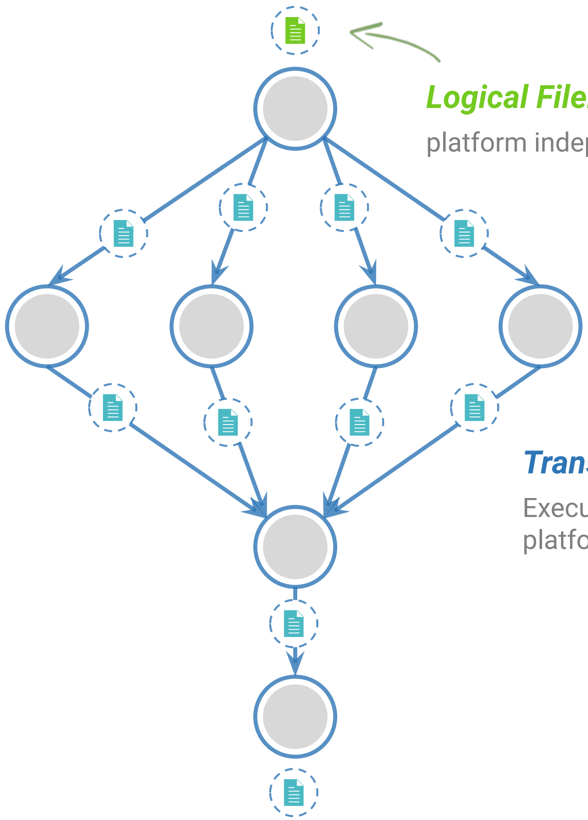
Input Workflow Specification YAML formatted

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

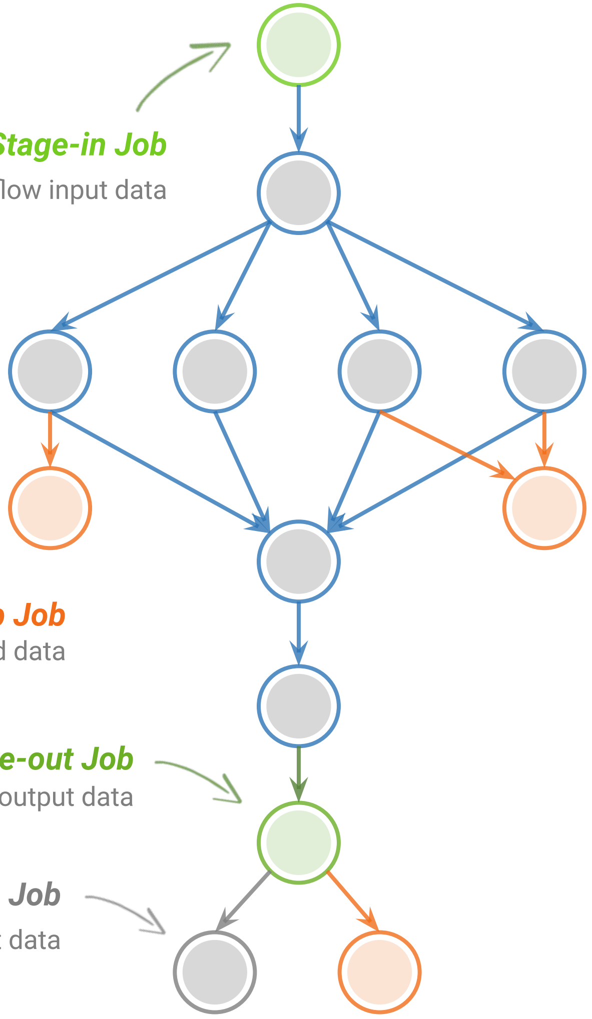
Removes unused data

Stage-out Job

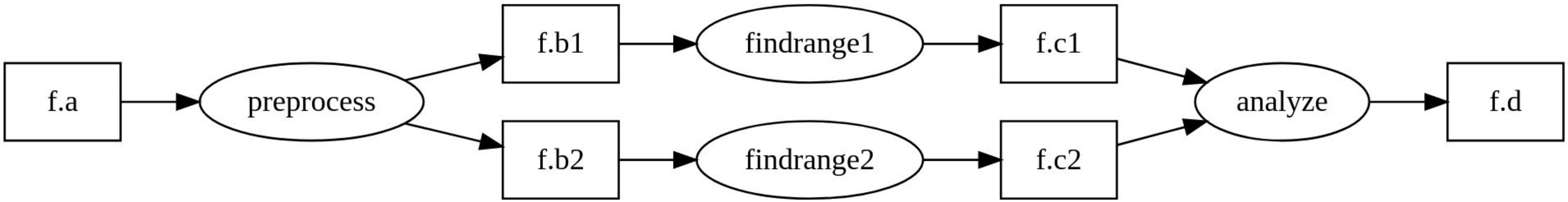
Stage-out generated output data

Registration Job

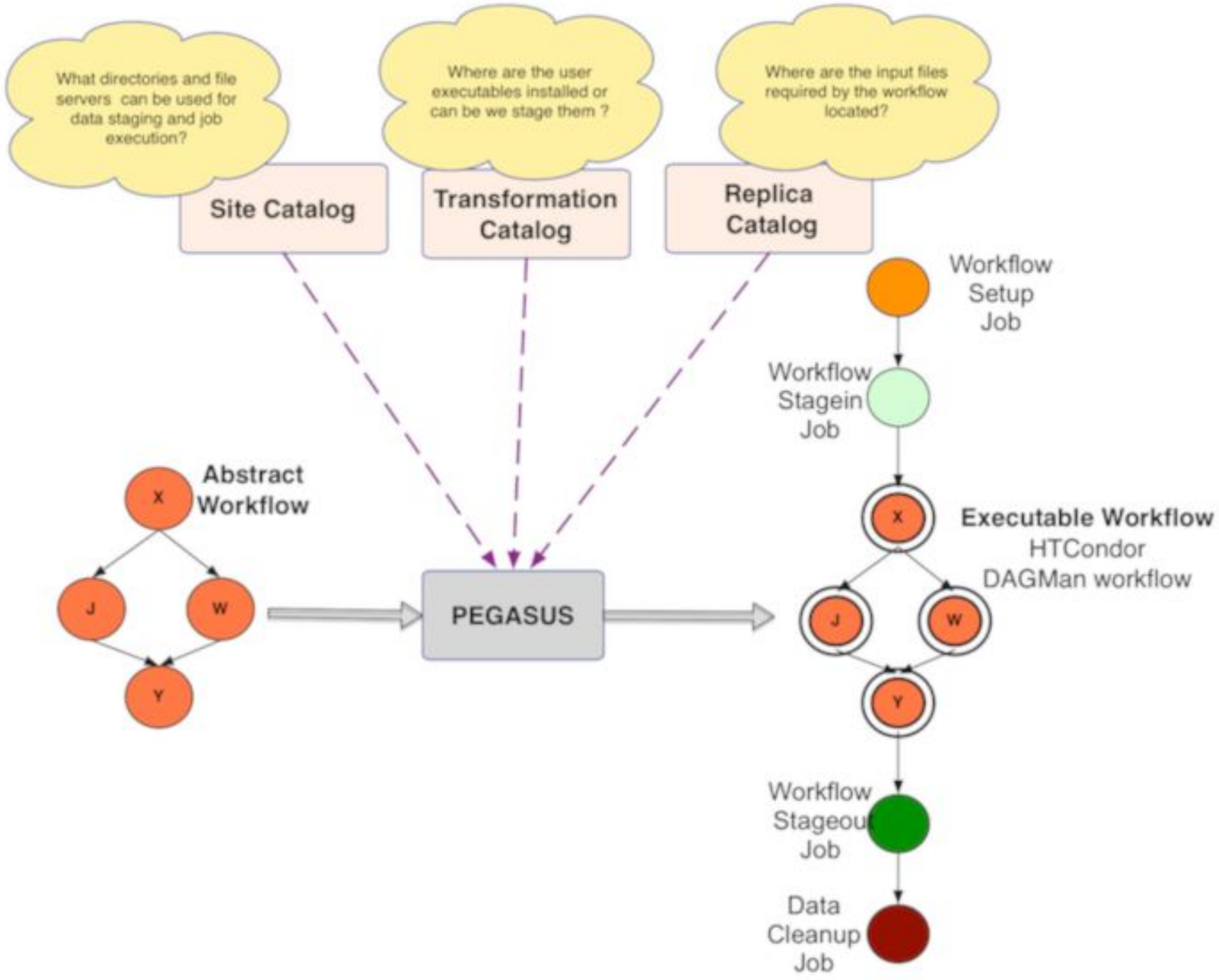
Registers the workflow output data



EXECUTABLE WORKFLOW


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

Pegasus


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
X
J
W
Y
Workflow Stageout Job
Data Cleanup Job

2.2 Debugging

2.3 Command Line Tools

Pegasus Container Support

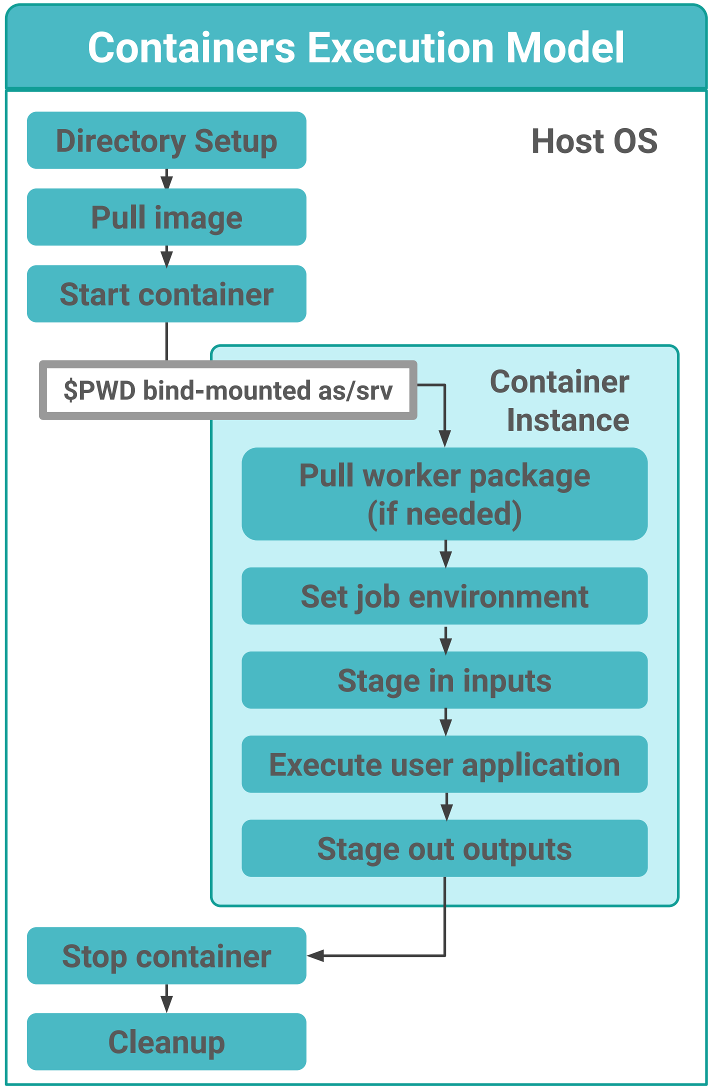
Users can refer to **containers** in the **Transformation Catalog** with their executable preinstalled

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

2.4 Containers

2.5 Summary

3. Advanced Topics

Data Staging Configurations

HTCondor I/O (HTCondor pools, OSG, …)

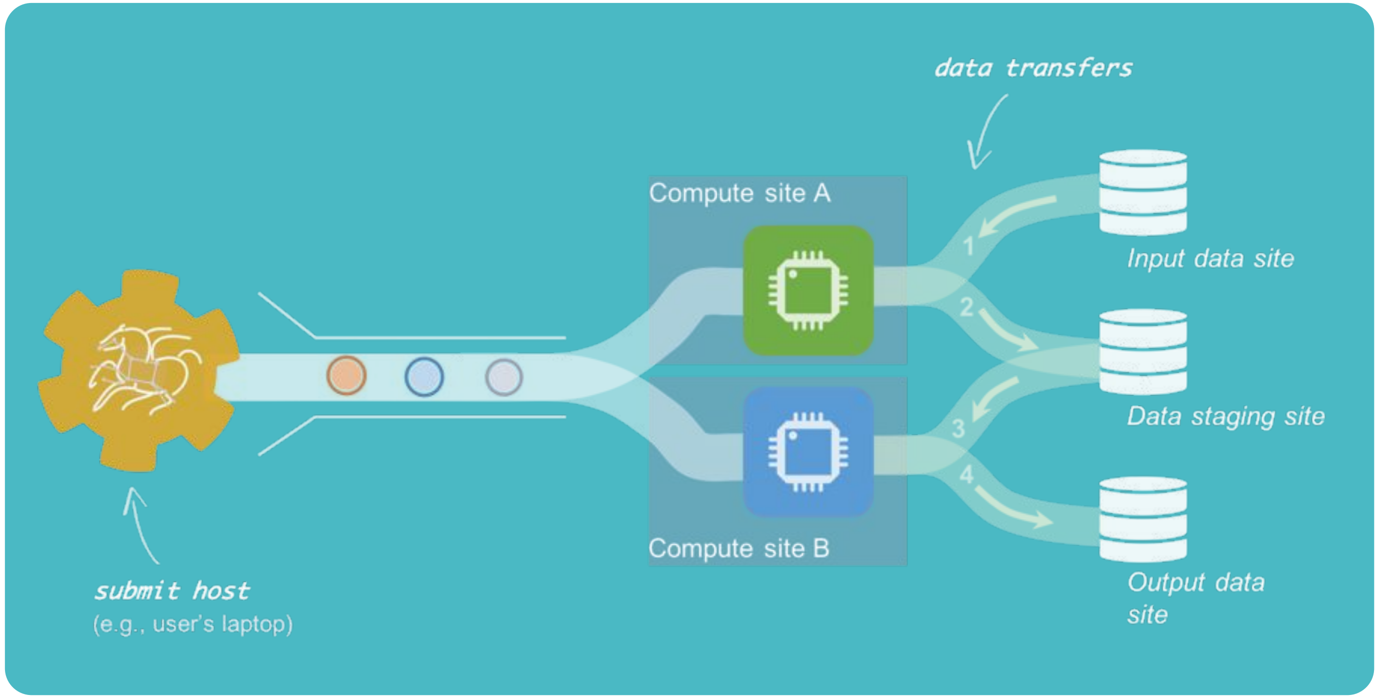
- Worker nodes do not share a file system
- Data is pulled from / pushed to the submit host via HTCondor file transfers
- Staging site is the submit host

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

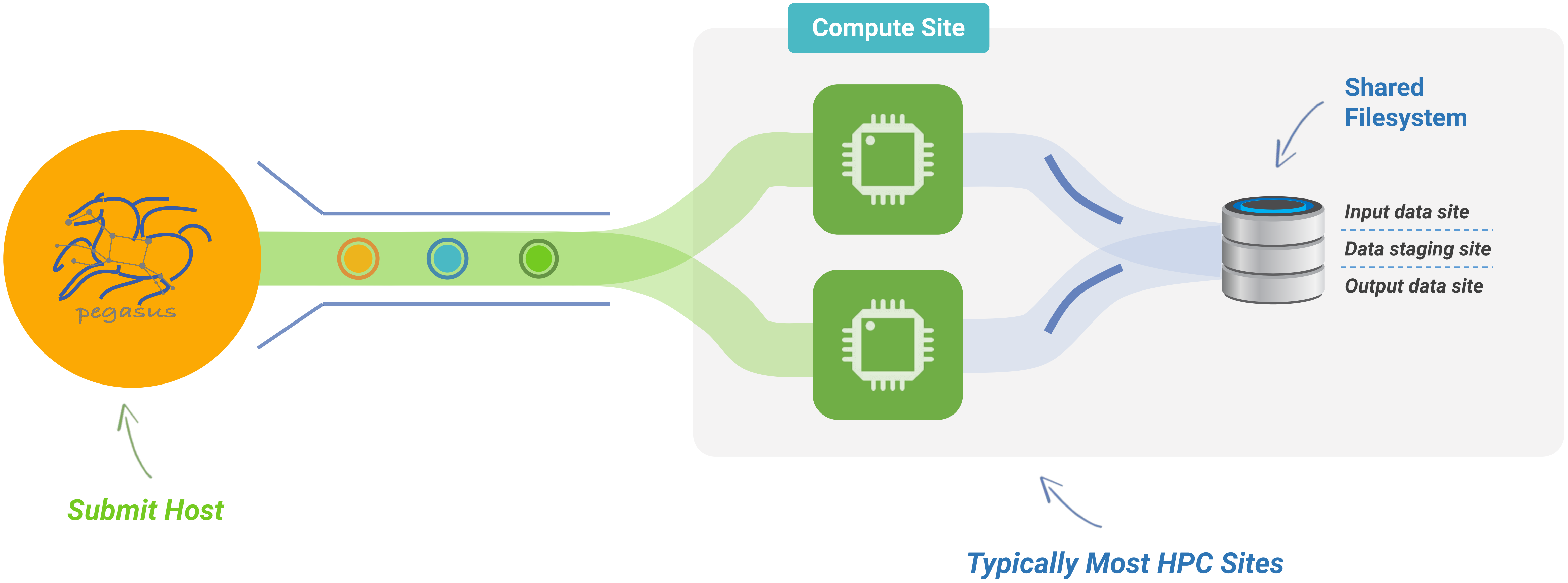
Shared File System (HPC sites, XSEDE, Campus clusters, …)

- I/O is directly against the shared file system



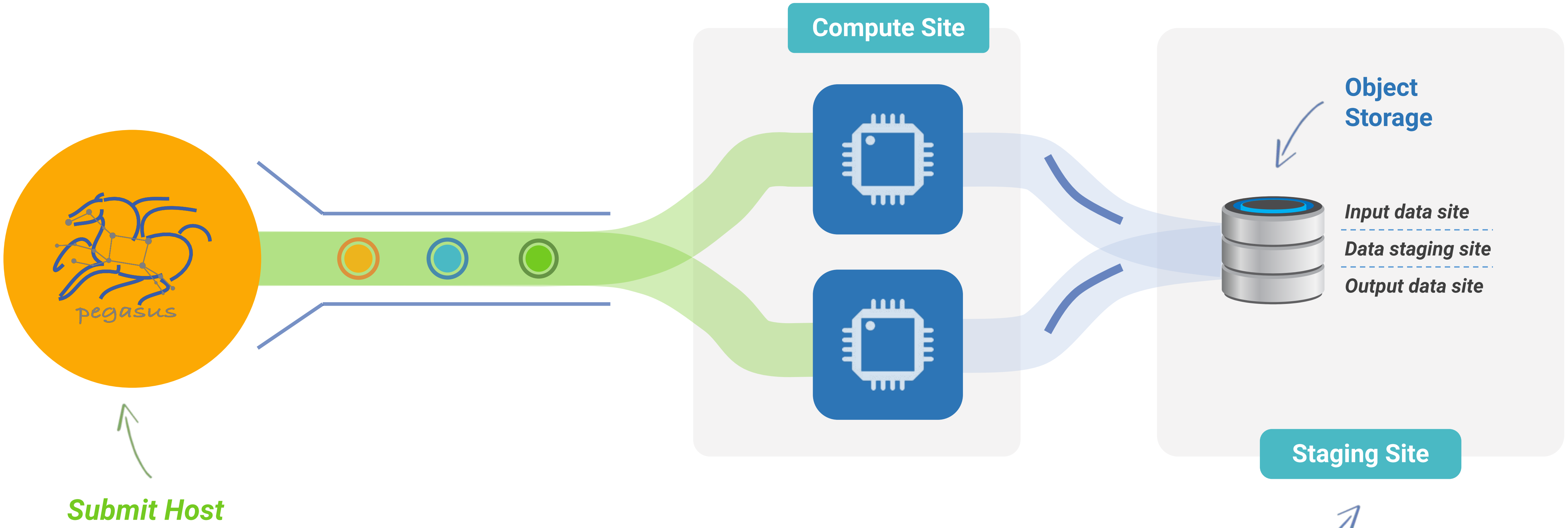
High Performance Computing

There are several possible configurations…



Cloud Computing

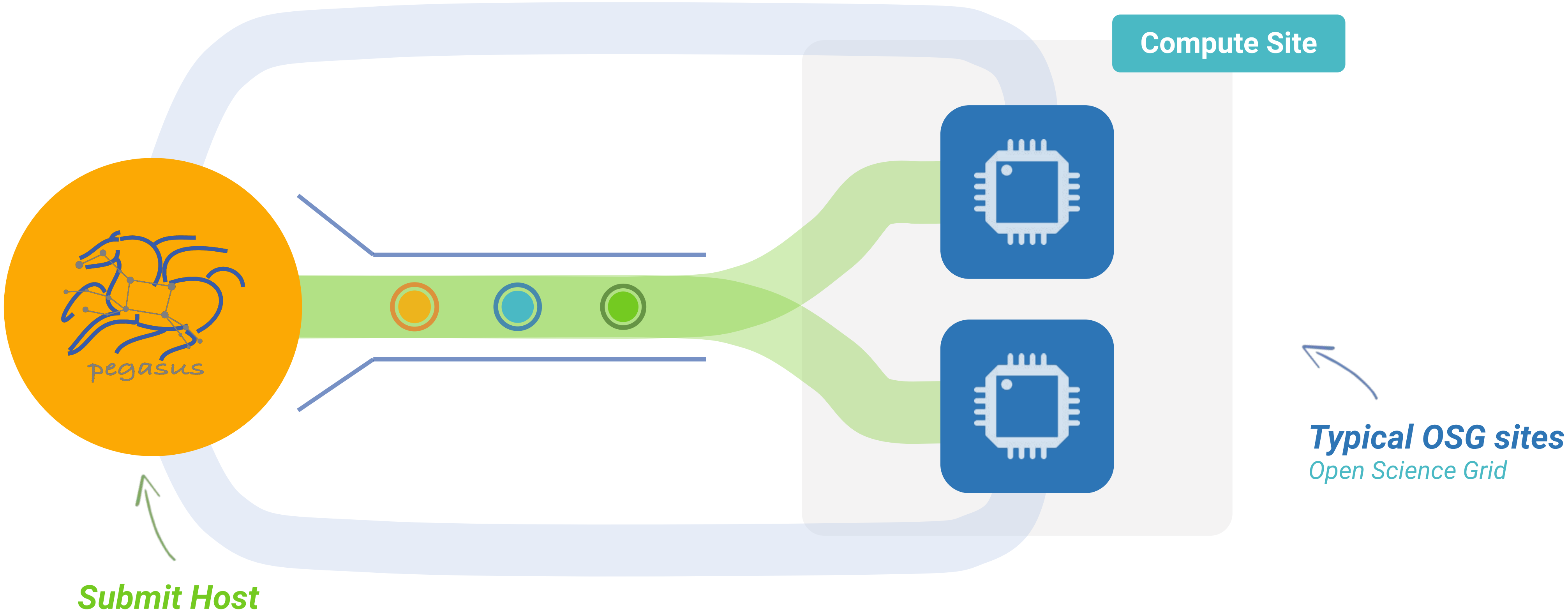
High-scalable object storages



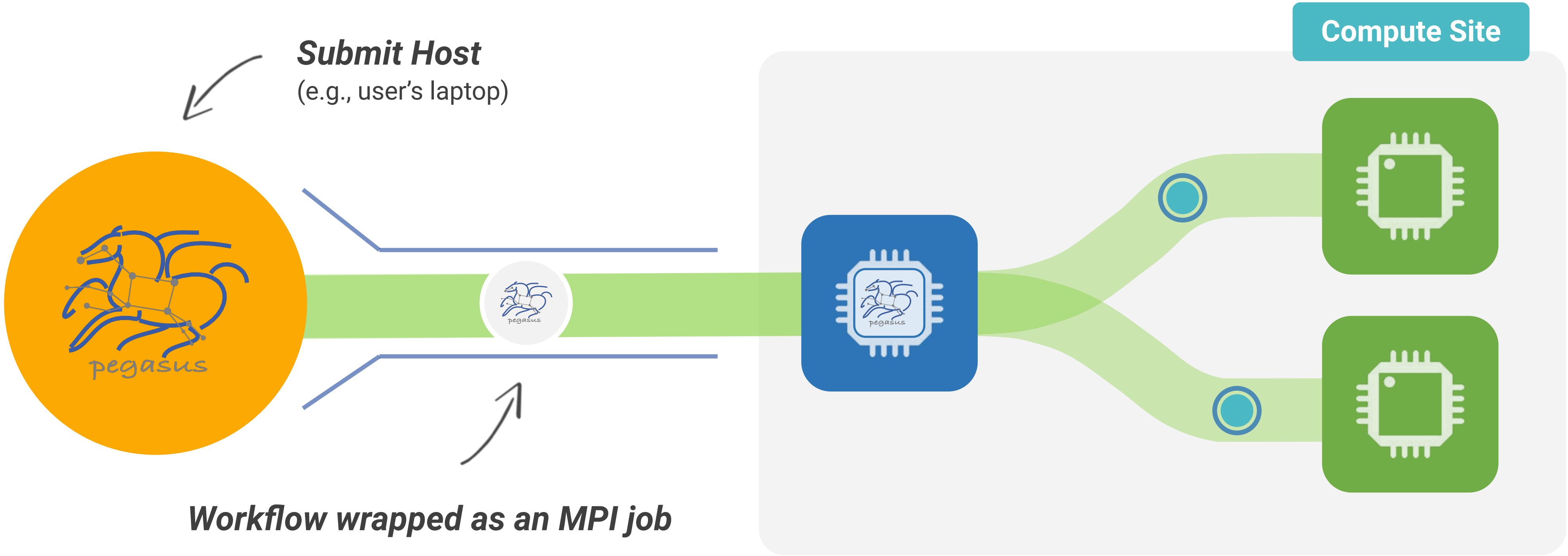
Typical cloud computing deployment
(Amazon S3, Google Storage)

Grid Computing

Local data management

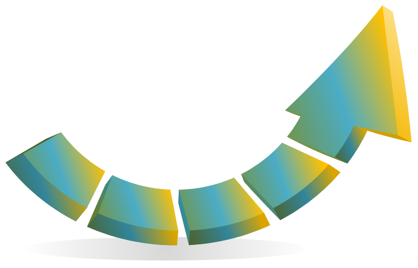


Running fine-grained workflows on HPC systems…

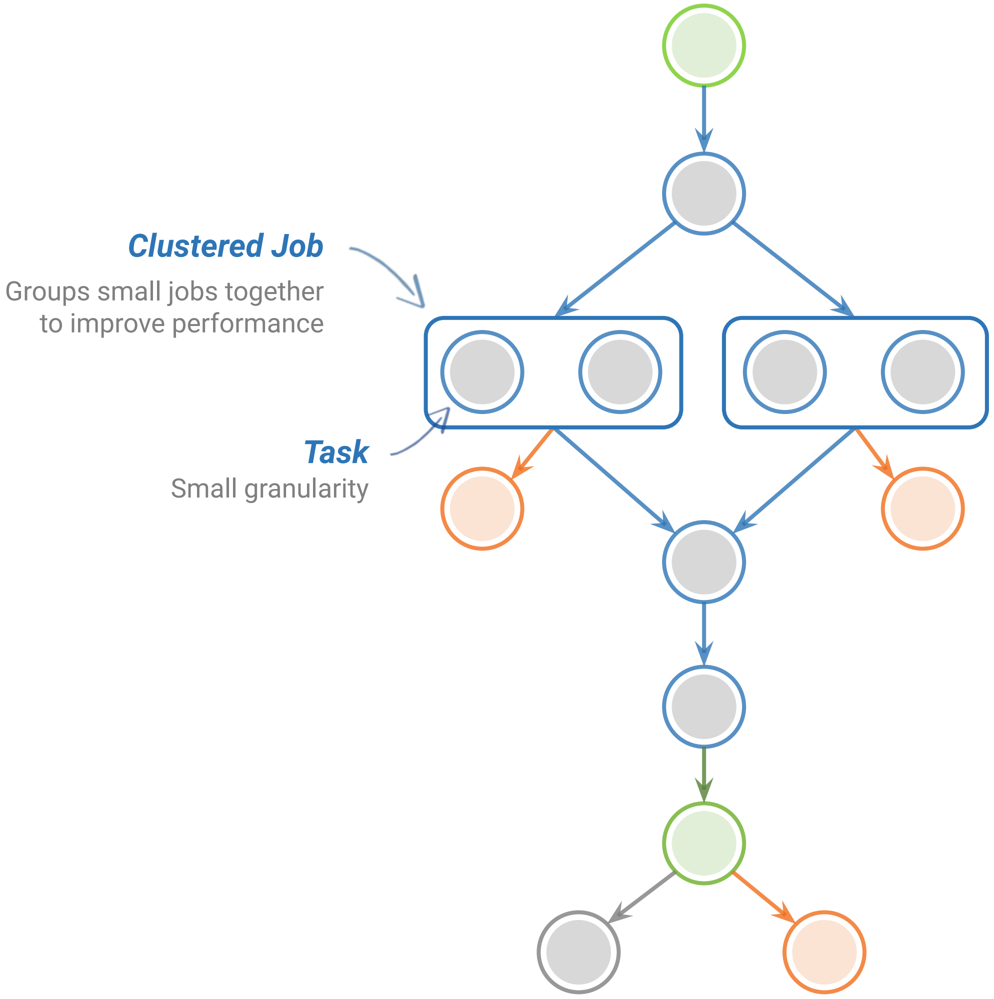


Workflow wrapped as an MPI job

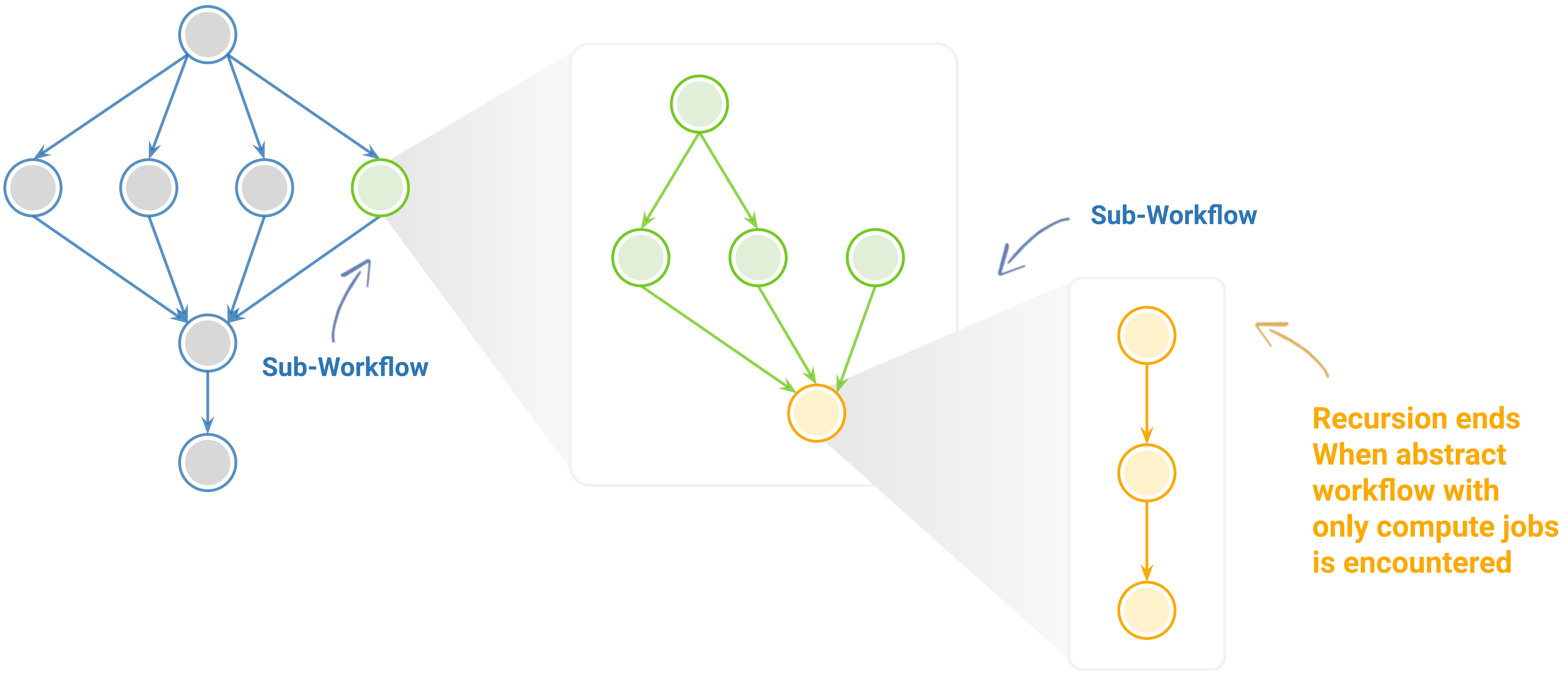
Allows sub-graphs of a Pegasus workflow to be submitted as monolithic jobs to remote resources



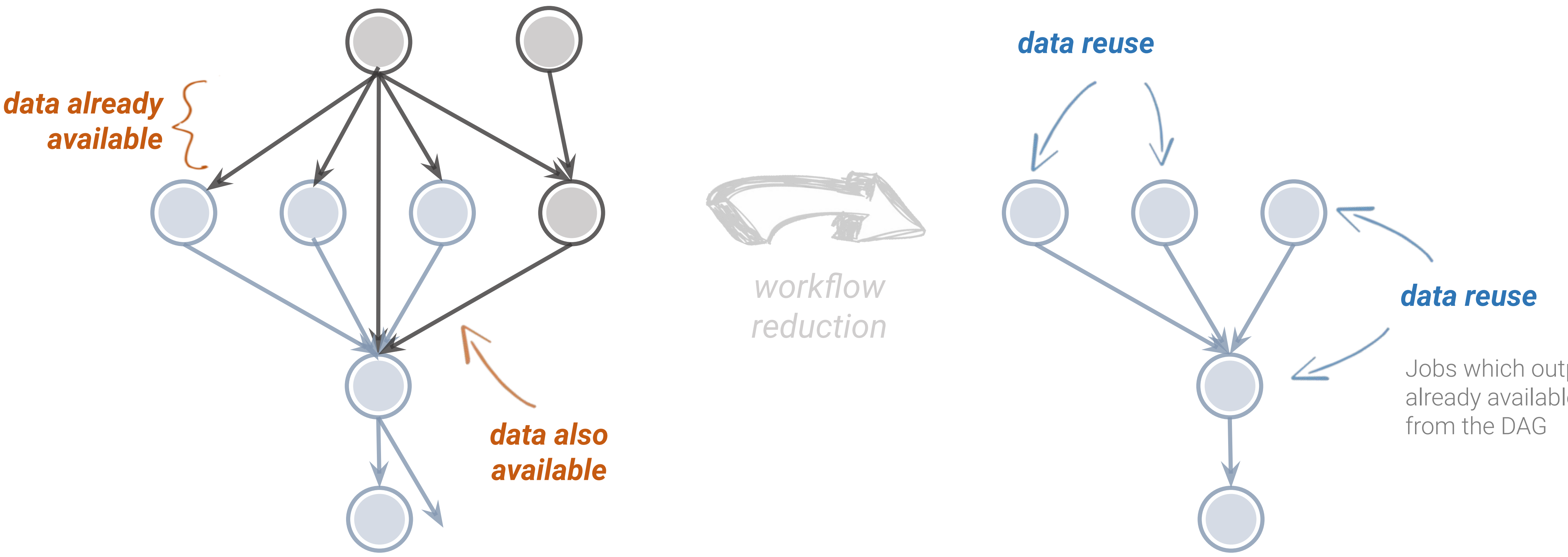
Performance.
Why not improve it?



Pegasus also handles large-scale workflows



Data Reuse prune jobs if output data already exists



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

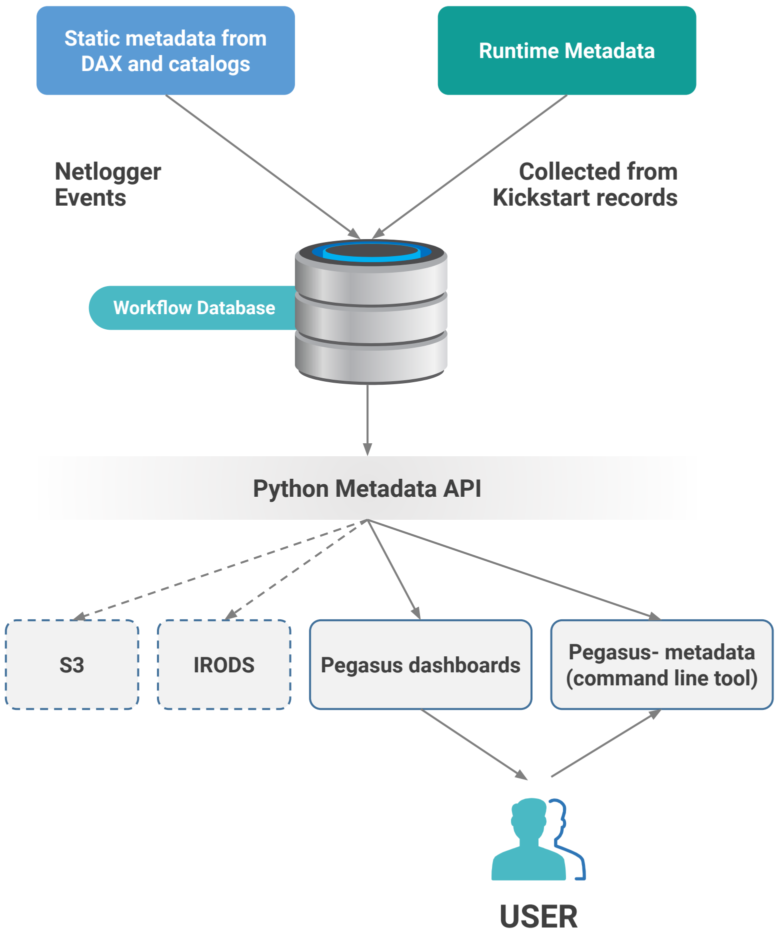
Static: application parameters
Runtime: performance metrics

```
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
```

Workflow, Job, File

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?"

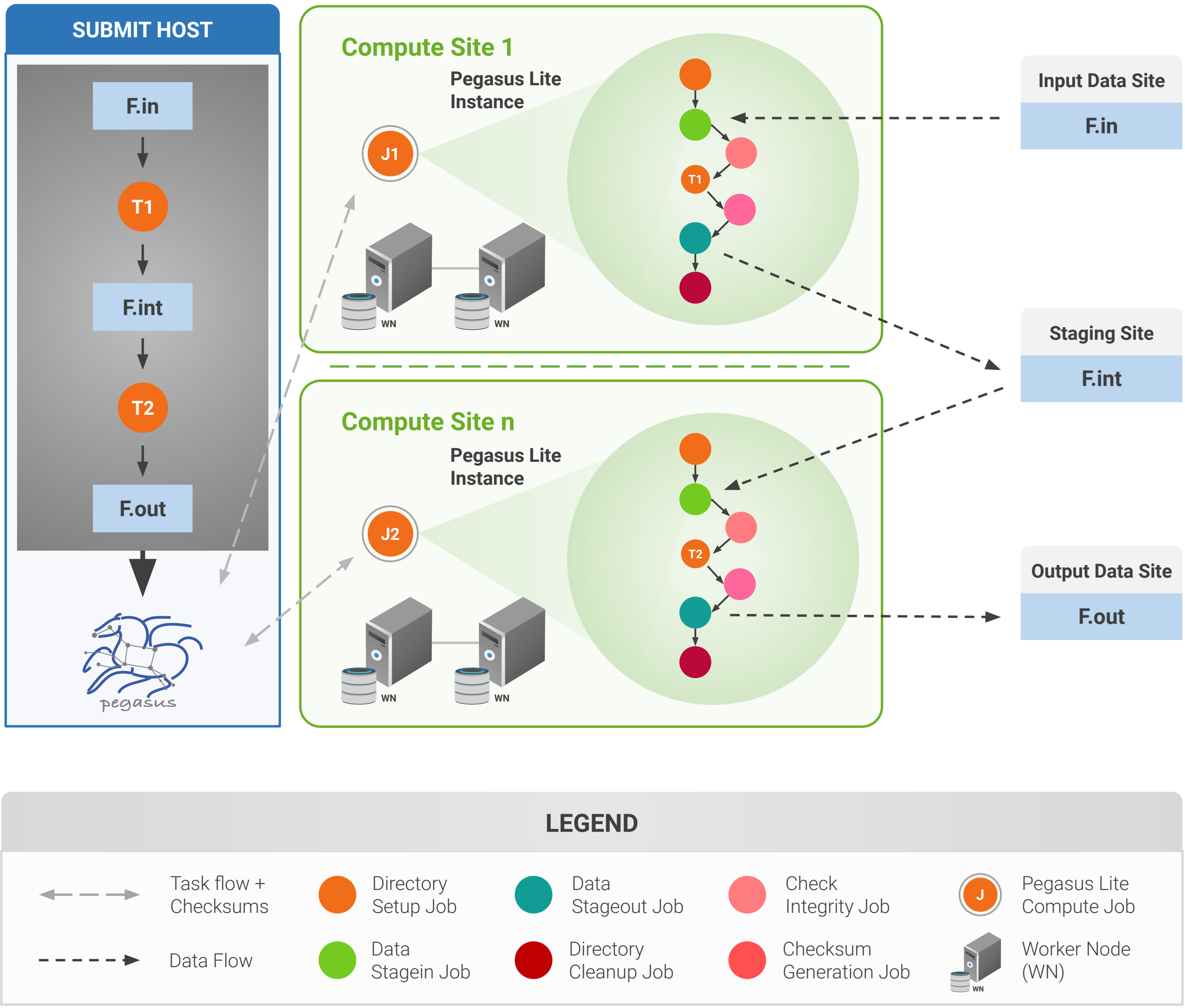
USC Viterbi
School of Engineering
Information Sciences Institute

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

Credentials Management

Credentials required for two purposes

- Job Submission
- Data transfers to **stage-in** input and **stage-out** generated outputs when a job executes

Specifying Credentials

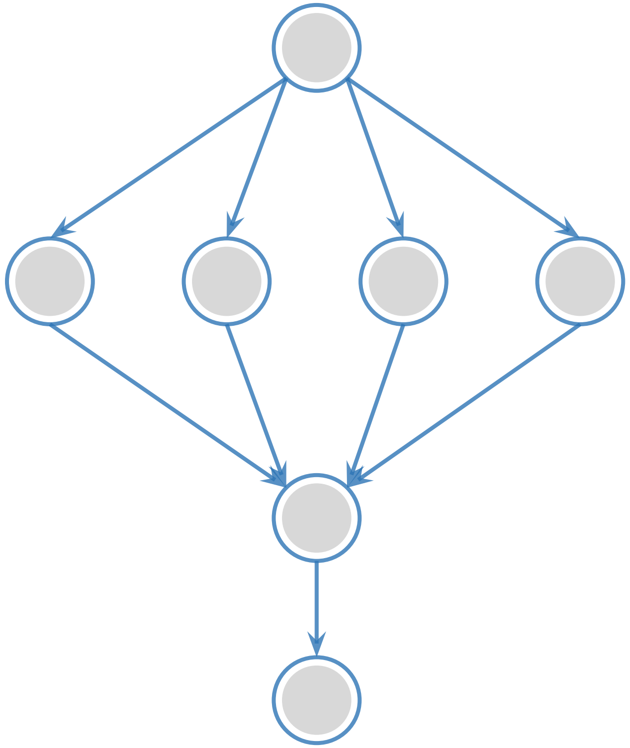
- Users can specify credentials in a **generic credentials file** on submit host
- Associate credentials with sites in site catalog

Approach

- Planner will **automatically** associate the **required credentials** with each job
- The credentials are **transferred** along with the job
- Usually available **only for the duration** of the job **execution**

Supported Credentials

- X.509 grid proxies
- Amazon AWS S3 keys,
- Google Cloud Platform OAuth token (.boto file),
- iRods password
- SSH keys
- Web Dav



USC Viterbi School of Engineering Information Sciences Institute

AWS BATCH

Amazon AWS Batch

AWS Batch

- Container based, dynamically scaled and efficient batch computing service

- Automatically launches compute nodes in Amazon based on demand in the associated job queue

- Users can specify compute environment that dictates what type of VM's are launched

Pegasus will **allow clusters of jobs** to be run on **Amazon EC2** using **AWS Batch Service**

New command line tool: **pegasus-aws-batch**

Automates most of the batch setup programmatically

- **Sets up and Deprovisions**
  - Compute Environment
  - Job Queues
- **Follows AWS Batch HTTP specification**

Pegasus

est. 2001

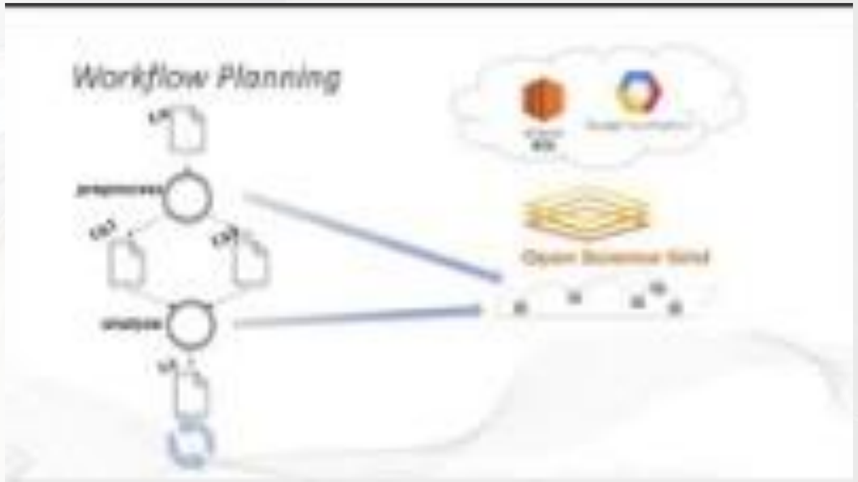
Automate, recover, and debug scientific computations.

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