ACCESS | Advancing Innovation

ACCESS Pegasus

A hosted scientific workflow system part of ACCESS Support

Mats Rynge, USC/ISI

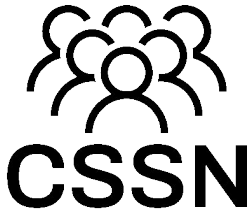
OpenCI Award: OCI 21-38296

NSF

ACCESS Support Strategy

Powerful Tools & Workflows

Dynamic Knowledge Base



Community Expertise and Experience

Powerful Tools & Workflows

OnDemand

INTEGRATED WEB-BASED INTERFACES

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

Pegasus

AUTOMATED WORKFLOWS

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

Knowledge Base

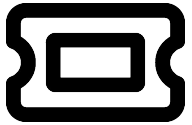
Self-service resources reduce the learning curve

DOCS

RP guides, code and best practices

LINKS

Provided and vetted by the community



TICKETS

Answers build the Knowledge Base

ASK.CI

Community Q&A forum

Computational Science & Support Network

CSSN

Utilizing community expertise

MATCH Engagements

Assistance is available from CSSN experts matched to Researcher needs.

MATCHPlus

Researchers assisted by a Student and their Mentor

< 6 MONTHS

MATCHPremier

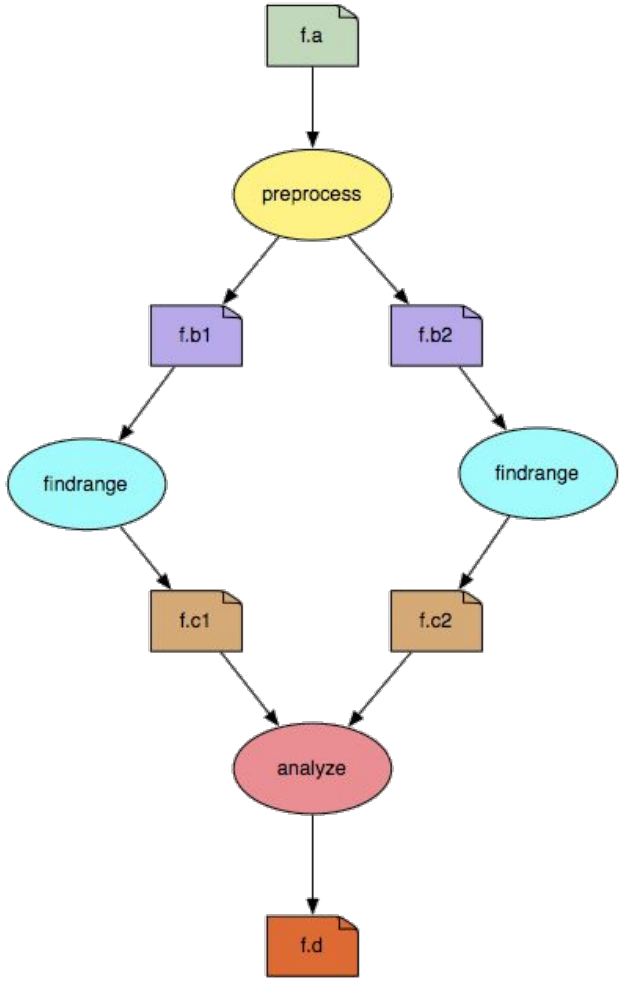
Researchers assisted by an expert Consultant

6+ MONTHS

Pegasus

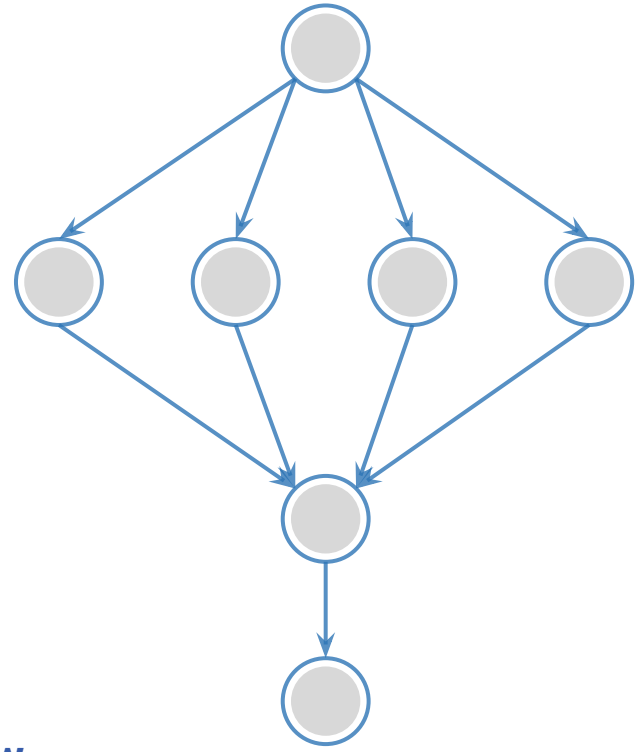
Scientific Workflows

- 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*



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

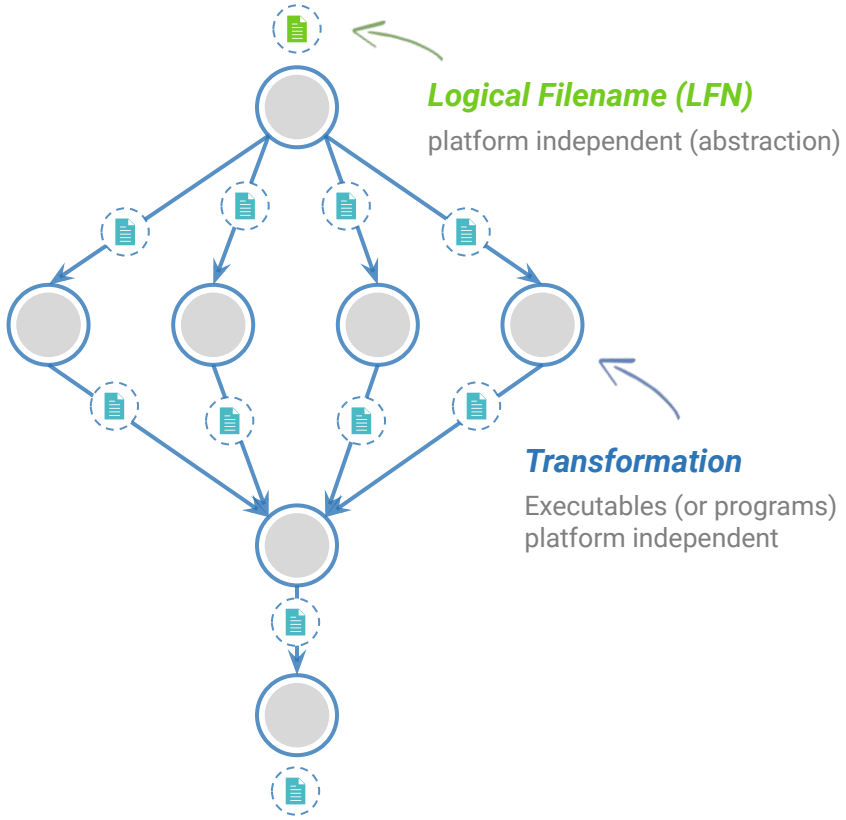
directed-acyclic graphs

Output Workflow

Portable Description

Users do not worry about low level execution details

ABSTRACT WORKFLOW

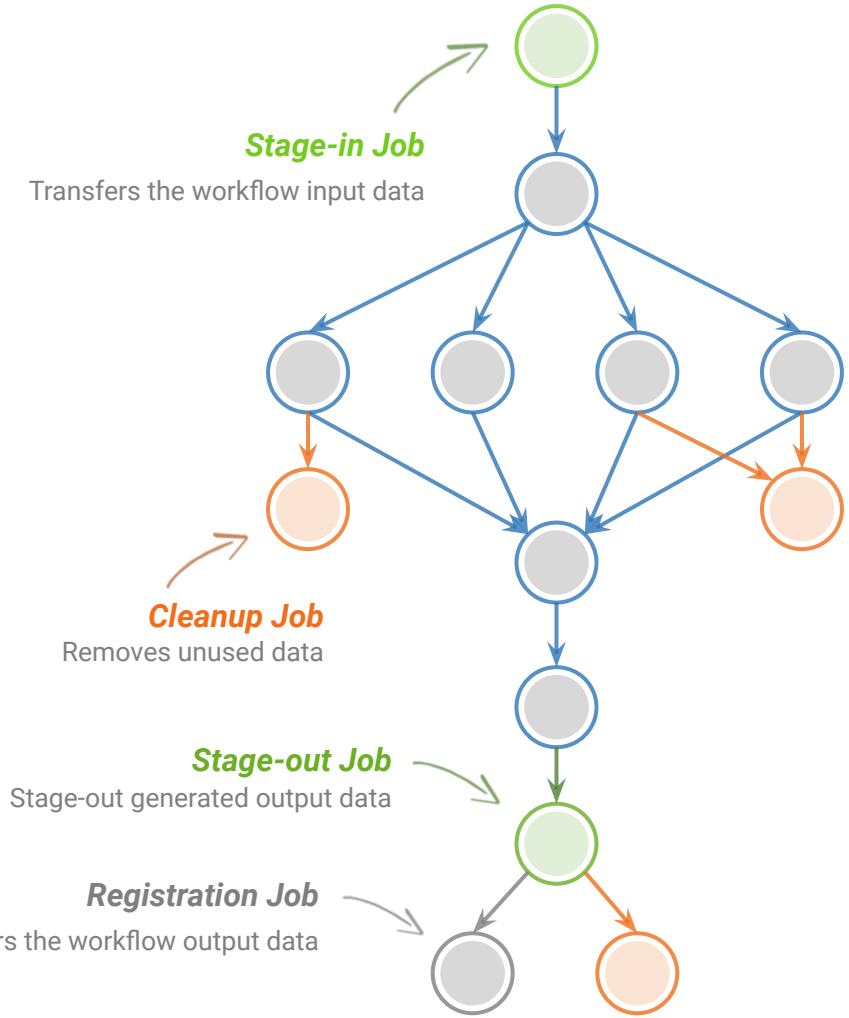


Logical Filename (LFN)

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

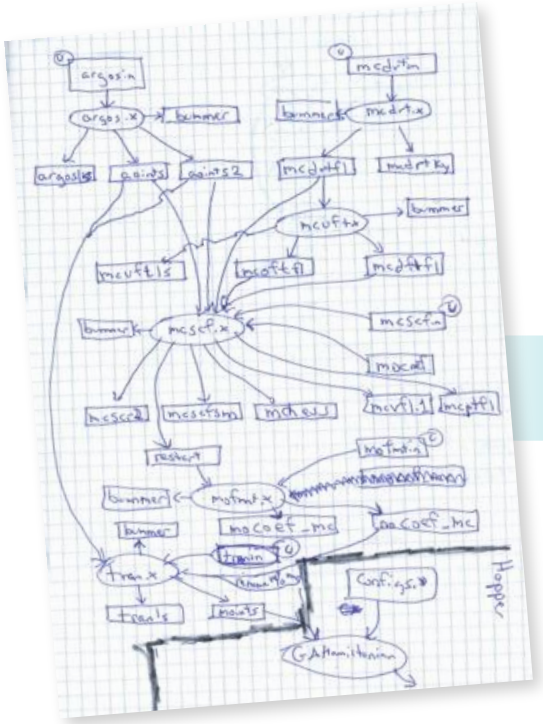
Stage-out generated output data

Registration Job

Registers the workflow output data

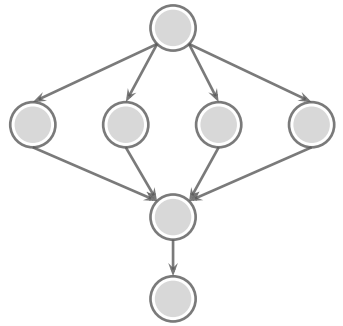
EXECUTABLE WORKFLOW

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
```

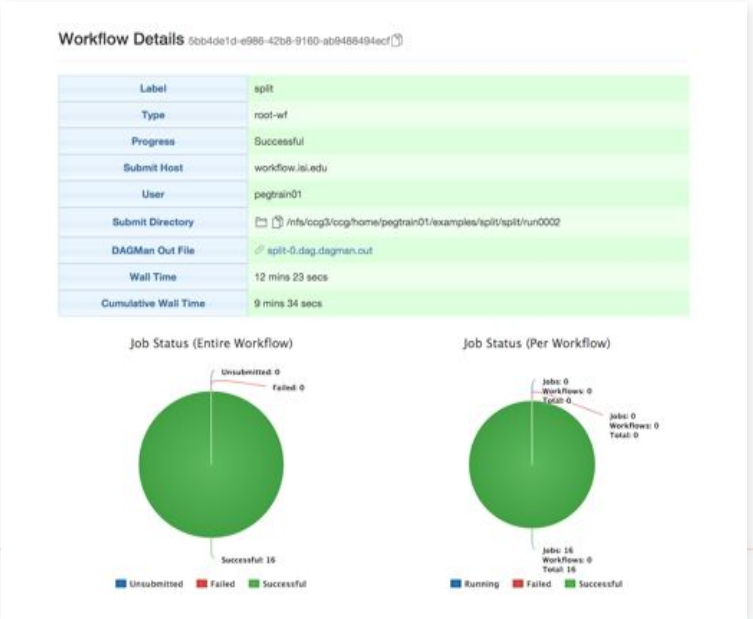
YAML Formatted

PEGASUS 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

ACCESS | Advancing Innovation

NSF

command-line…

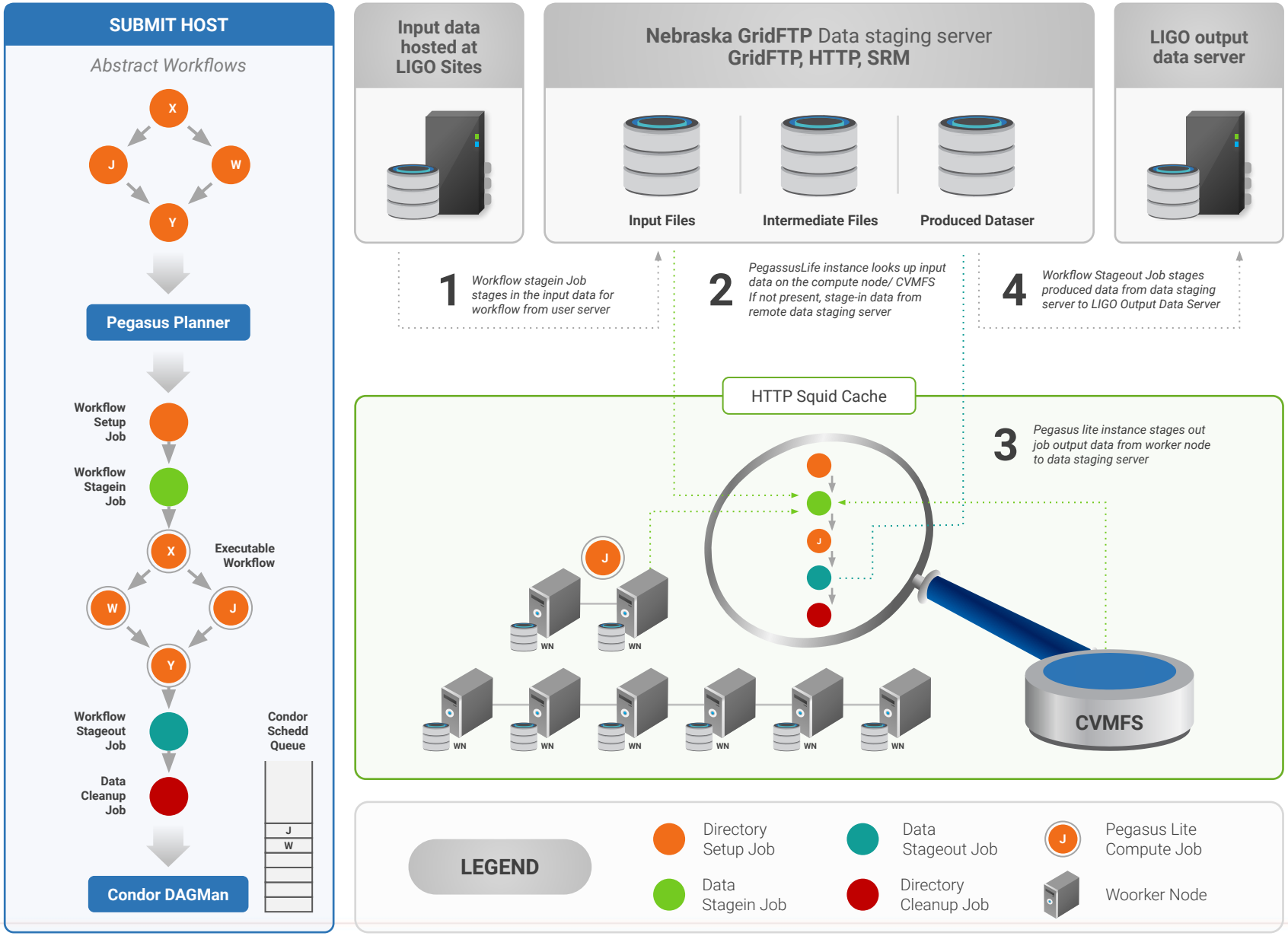
```
$ pegasus-status pegasus/examples/split/run0001
STAT IN_STATE JOB
Run 00:39 split-0 (/home/pegasus/examples/split/run0001)
Idle 00:03 └─split_ID0000001
Summary: 2 Condor jobs total (I:1 R:1)

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

Success Stories



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

Advanced LIGO

Laser **I**nterferometer
Gravitational Wave **O**bservatory

60,000 Compute Tasks

Input Data: 5000 files (10GB total)

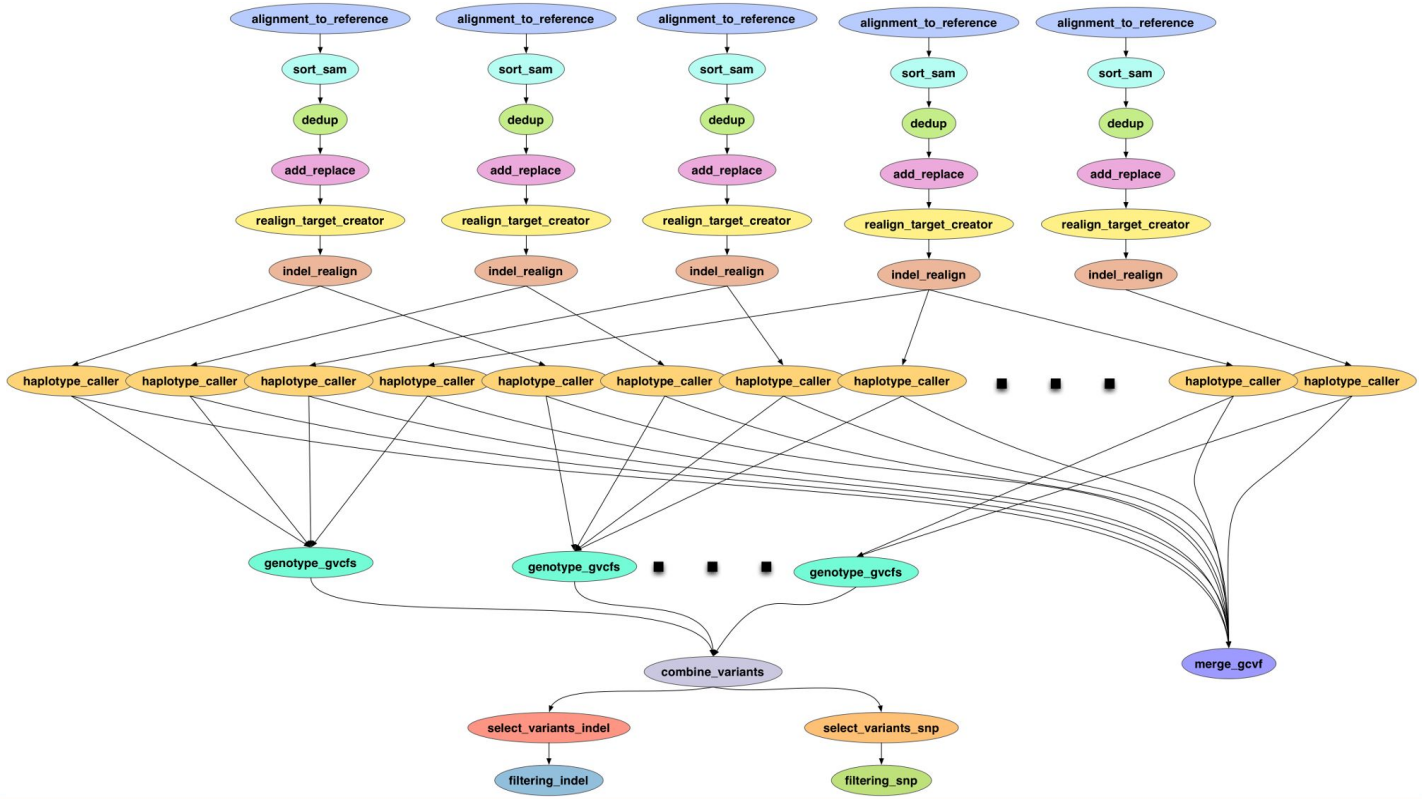
Output Data: 60,000 files (60GB total)

Processed Data: 725 GB

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

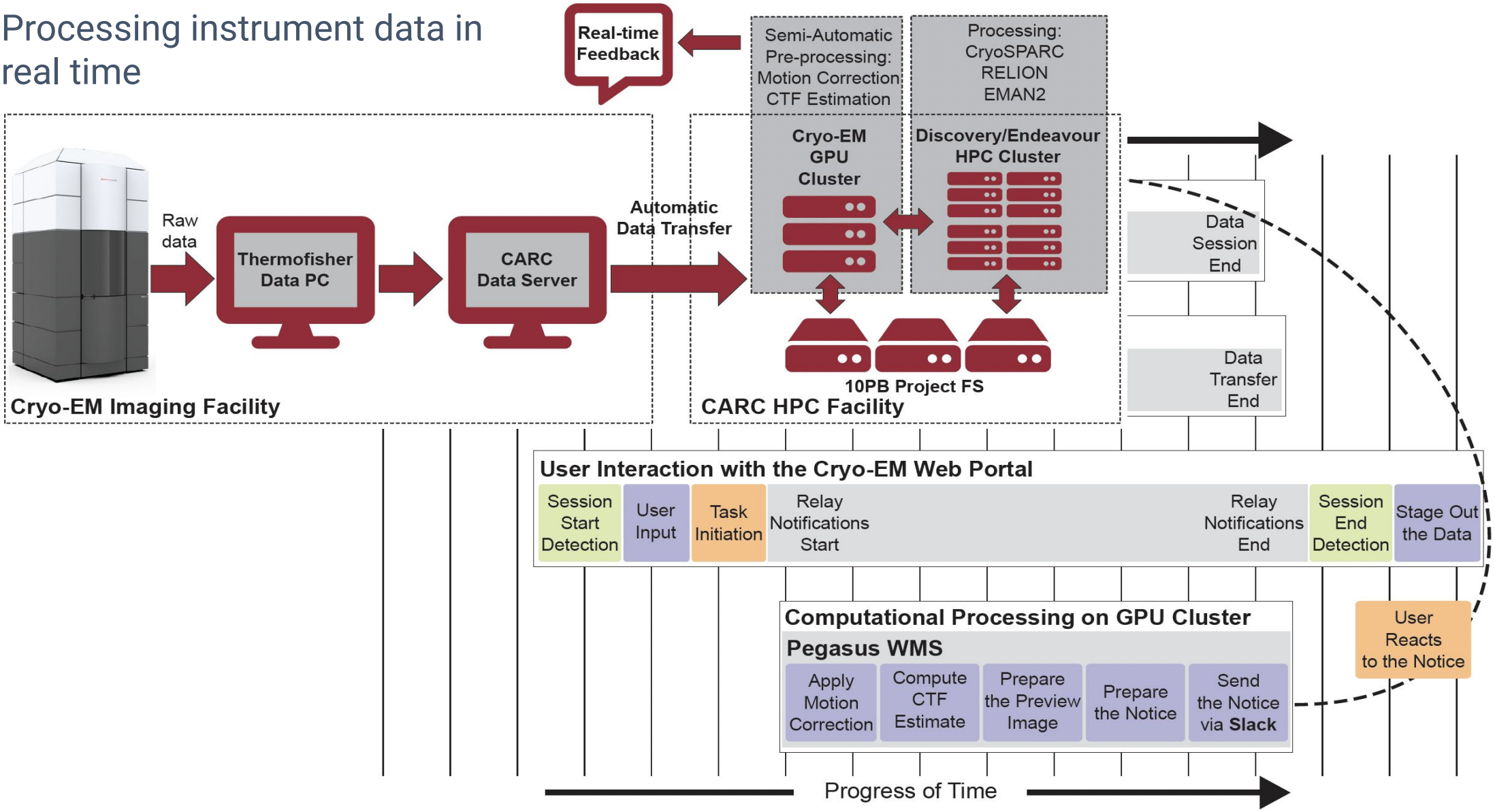
SoyKB

PI: Dong Xu
Trupti Joshi, Saad Kahn, Yang Liu, Juexin Wang, Badu Valliyodan, Jiaojiao Wang



Task	Base Code	Cores (Threads)	Memory (GB)
Alignment_to_reference	BWA	7	8
Sort_sam	Picard	1	21
Dedup	Picard	1	21
Add_replace	Picard	1	21
Realign_target_creator	GATK	15	10
Indel_realign	GATK	1	10
Haplotype_caller	GATK	1	3
Genotype_gvcfs	GATK	1	10
Merge_gvcf	GATK	10	20
Combine_variants	GATK	1	10
Select_variants	GATK	14	10
Filtering	GATK	1	10

Processing instrument data in real time



ACCESS Pegasus

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

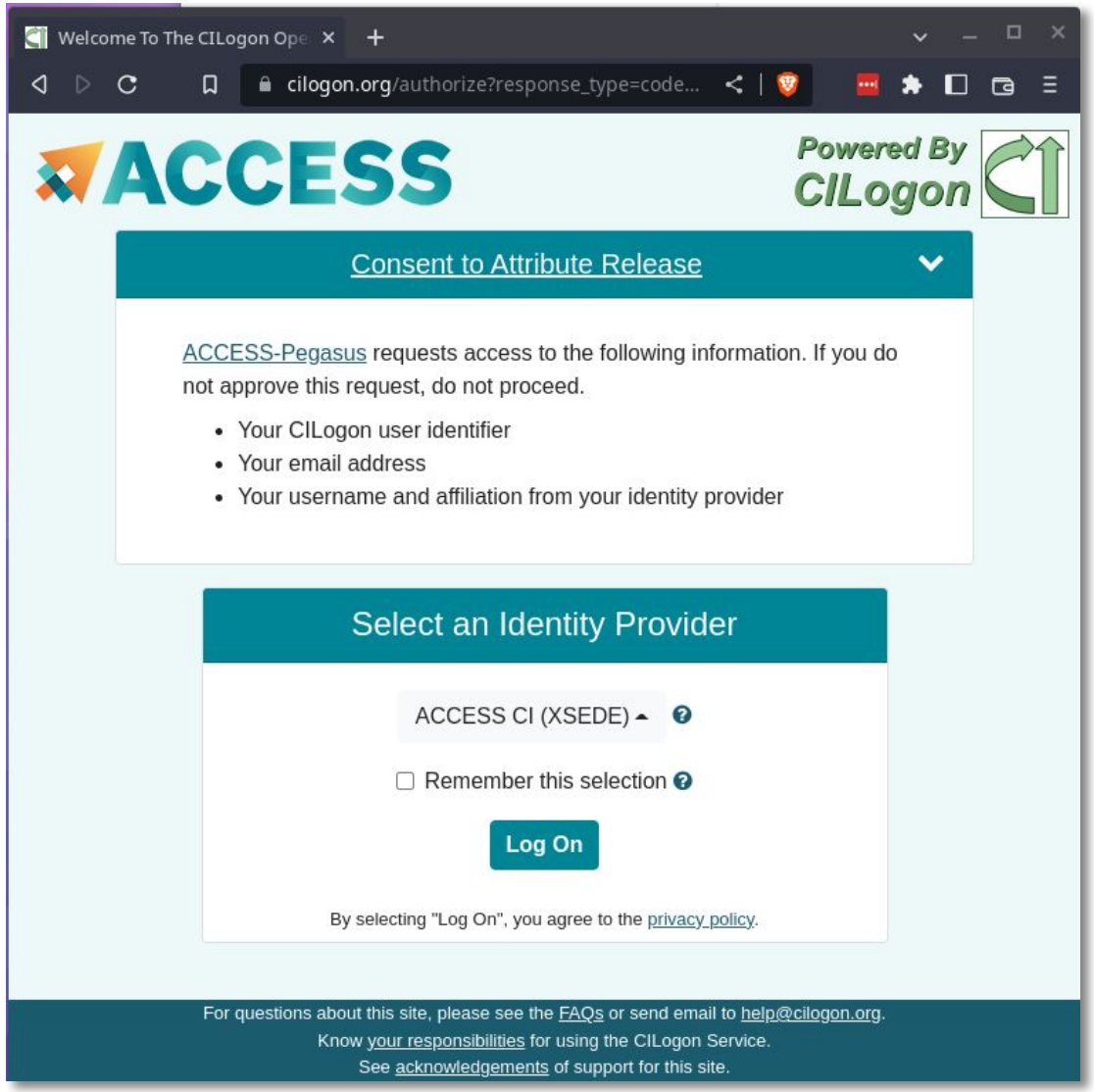
Prepare

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

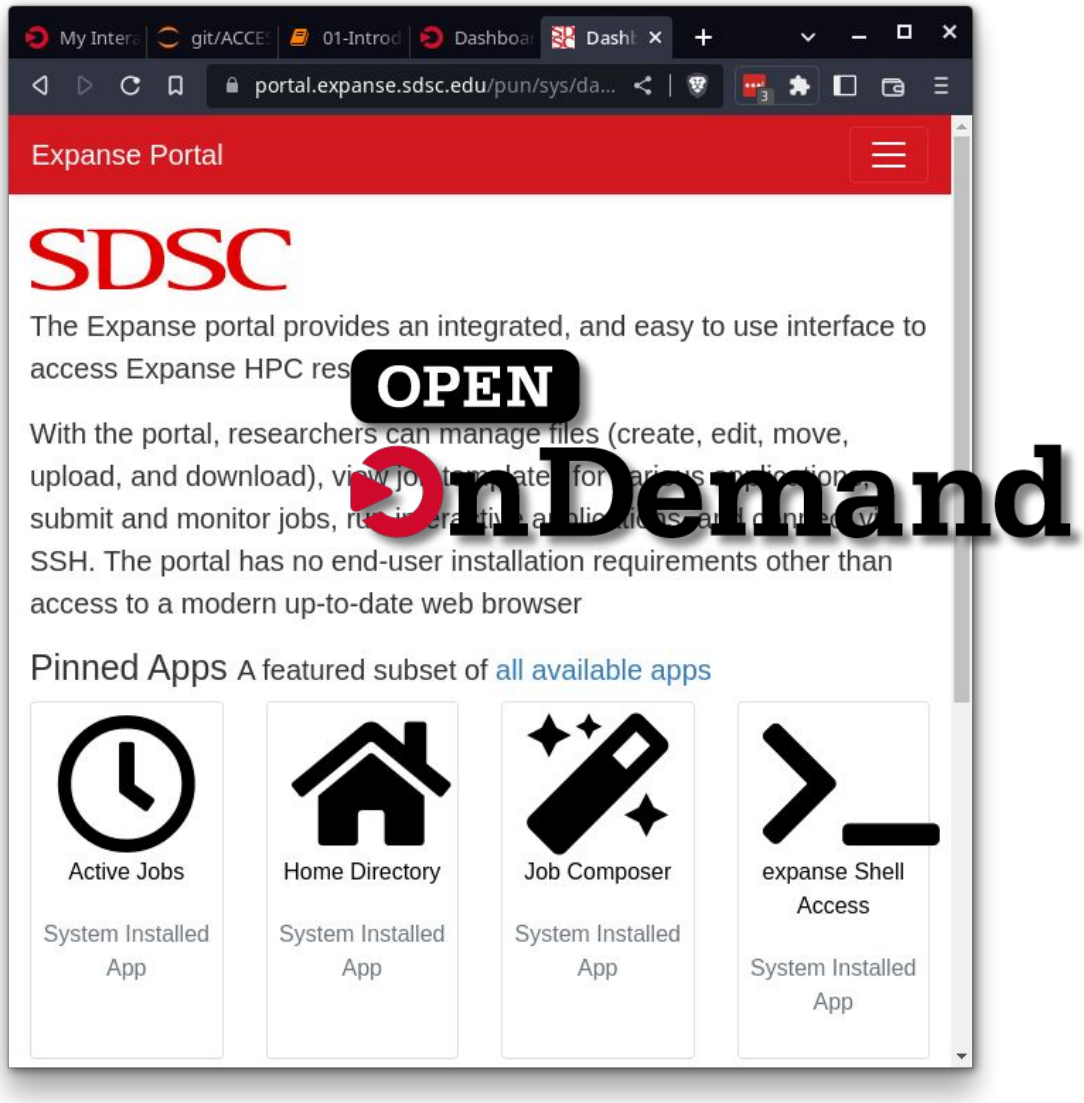


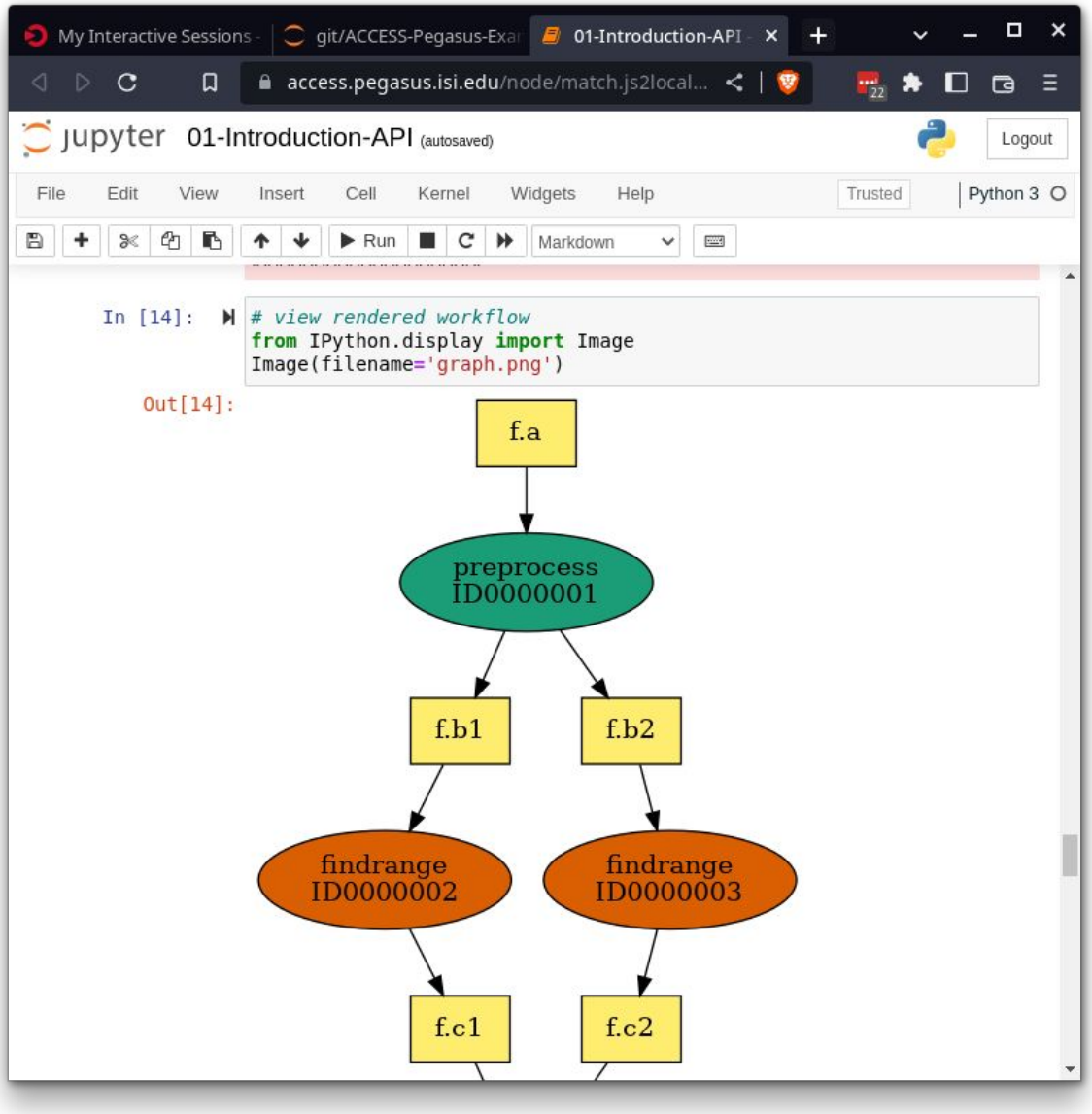
Prepare

Setting Up Resources

One time setup

Use Open Ondemand instances at resource providers to install ssh keys, and determine local allocation id



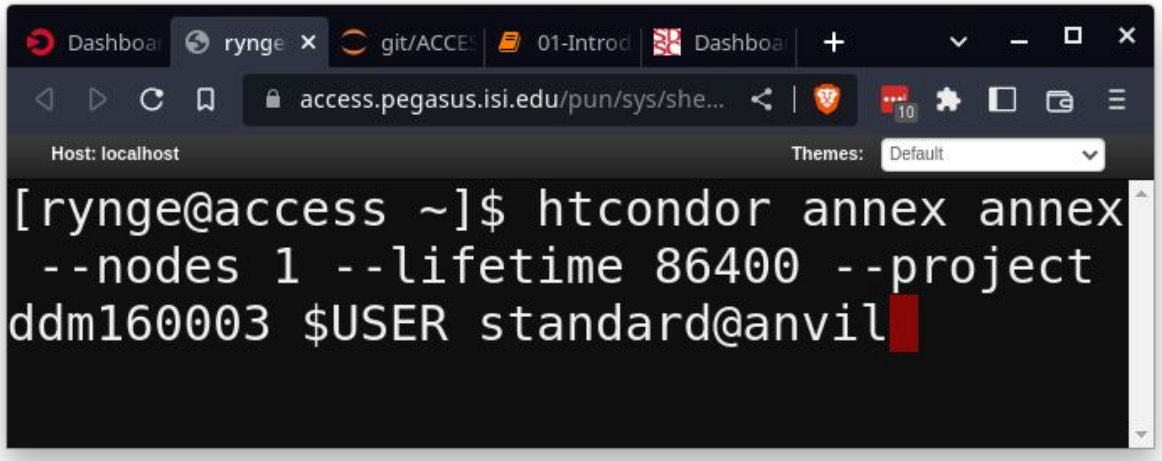


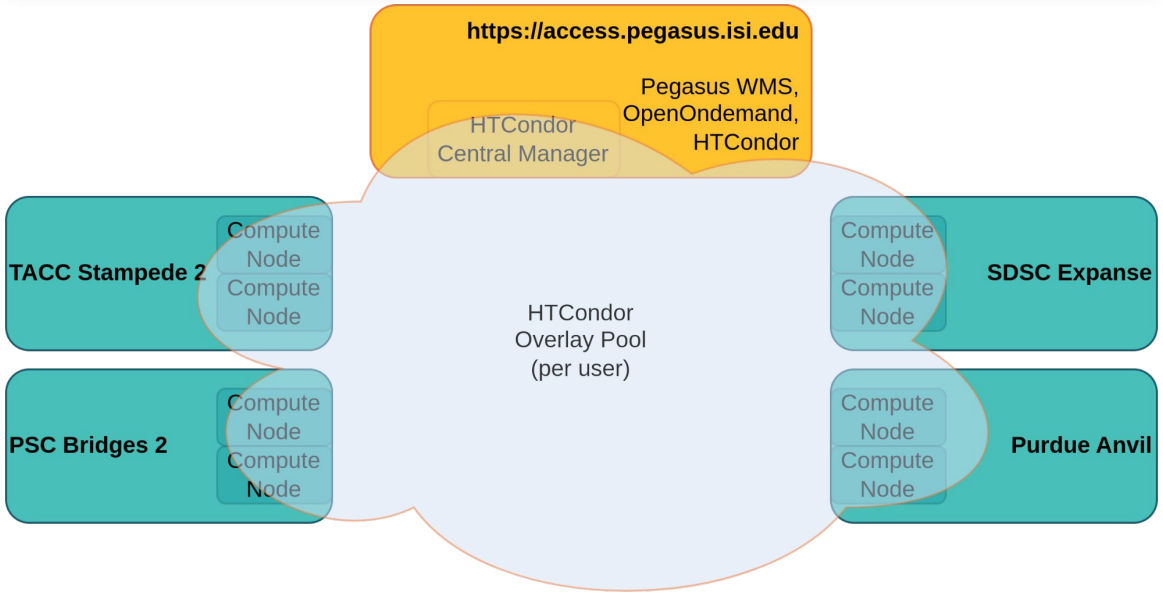
Step 1

Designing Workflow

Pegasus API in Jupyter Notebook

Fully hosted environment, based on Open Ondemand





Step 2

Provision Resources

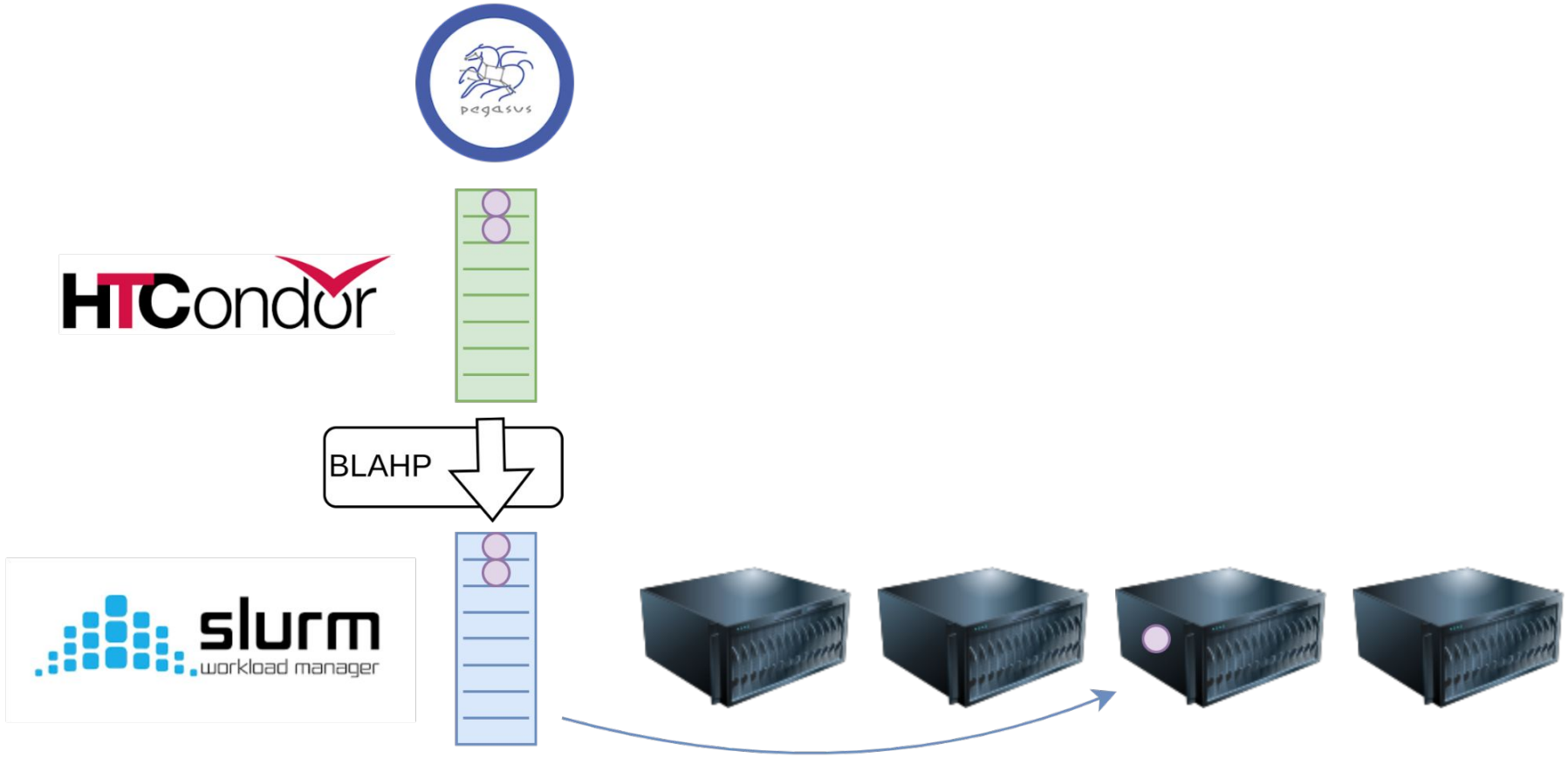
Use the HTCondor Annex tool to dynamically bring in compute nodes from one or more resource providers

HTCondor Software Suite

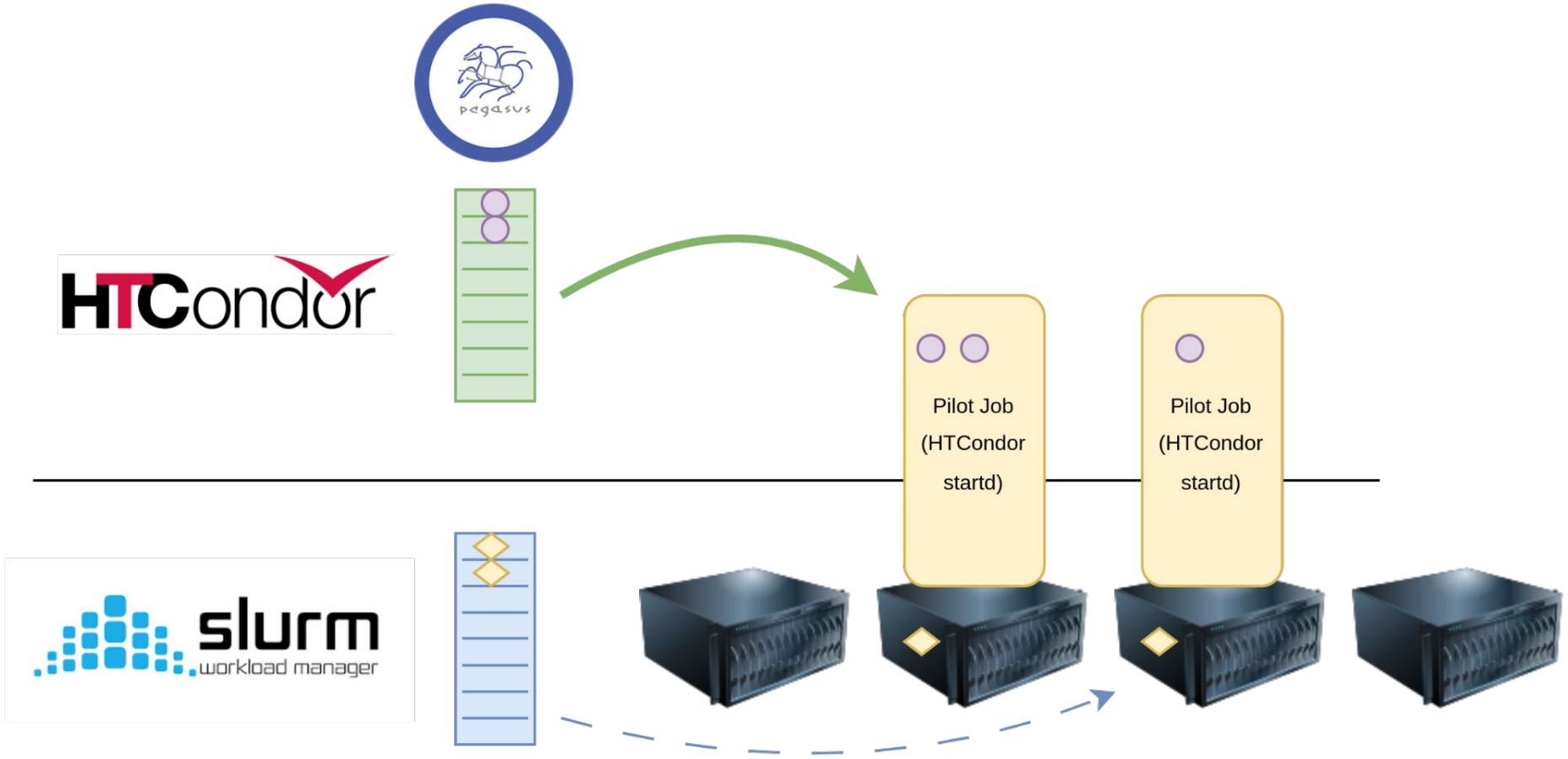
HTCondor Annex / Pilot Jobs

- **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.**

HTCondor with BLAHP translation layer



HTCondor Pilot Jobs

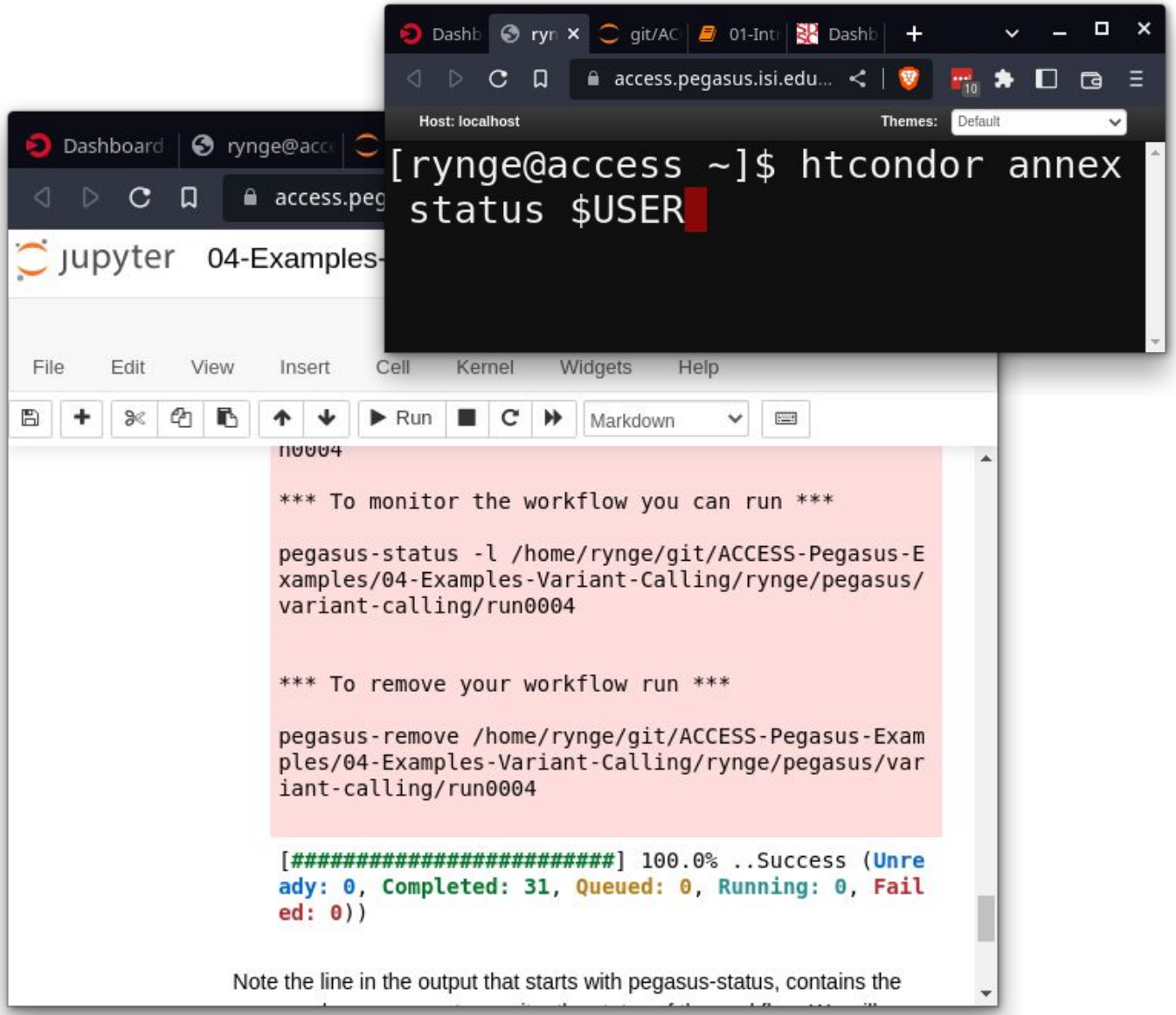


Step 3

Monitoring Workflow and Resources

Workflows can be monitored from within the Jupyter notebook, or via command line

HTCondor Annex can be monitored on the command line



Try it out!

Documentation:

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

Open a ticket:

https://support.access-ci.org/open-a-ticket

Questions?

Tutorial

Tutorial

This is **not** using ACCESS resources - jobs are staying local in the container.

If we are not finishing here today, feel free to keep exploring on your own

In-person: handout
Remote: https://tinyurl.com/pegasus-ern