Funded by the National Science Foundation, Grant #2320373

Scientific Workflows with Pegasus @ CHESS

Karan Vahi

USC Information Sciences Institute

X-CITE & DIALS Workshops

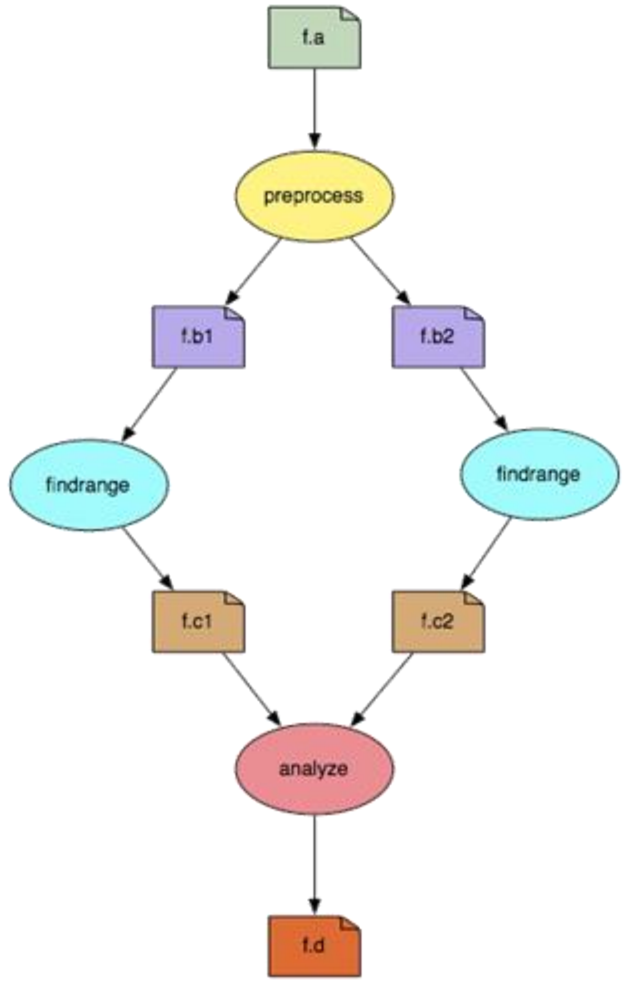
X-CITE and DIALS Workshops 2025

NSF OAC Award Numbers: 2320373, 2320375, 2320374

Funded by the National Science Foundation, Grant #2320373

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?
- The tasks in a scientific workflow can be everything from short serial tasks to very large parallel tasks (MPI for example) surrounded by a large number of small, serial tasks used for pre- and post-processing.
- Find the inherent DAG structure in your application to convert into a workflow



Funded by the National Science Foundation, Grant #2320373

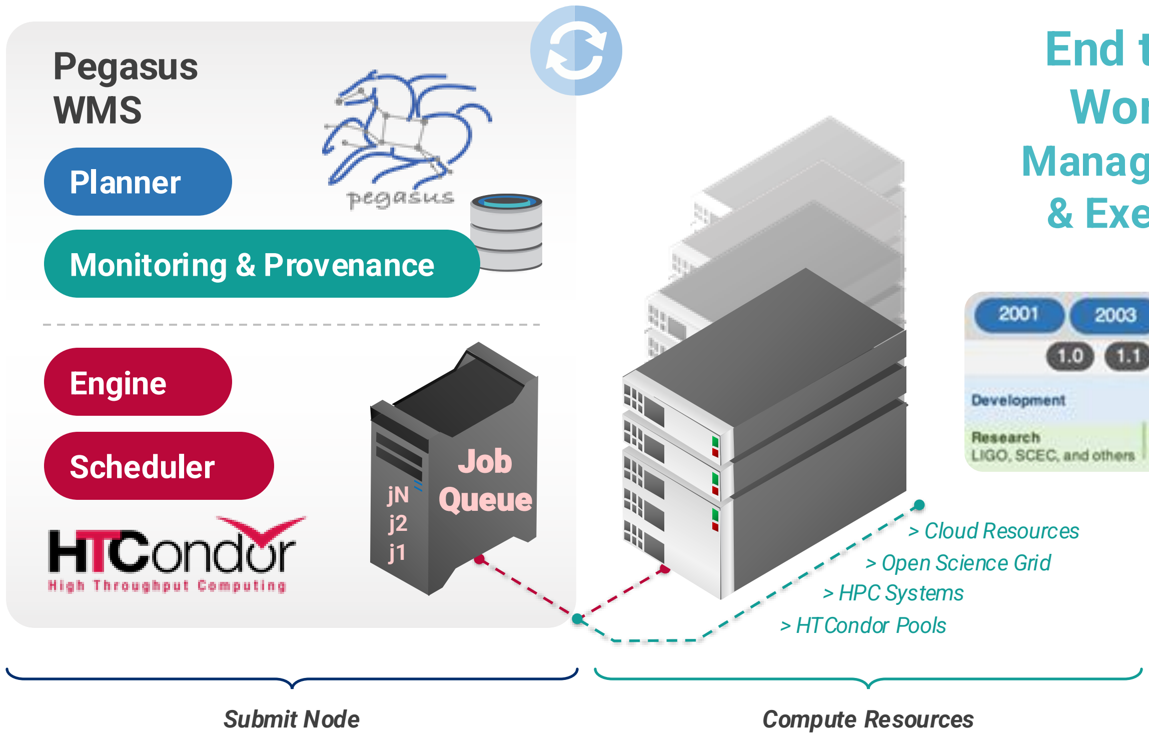
Workflow Challenges Across Domains

- 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

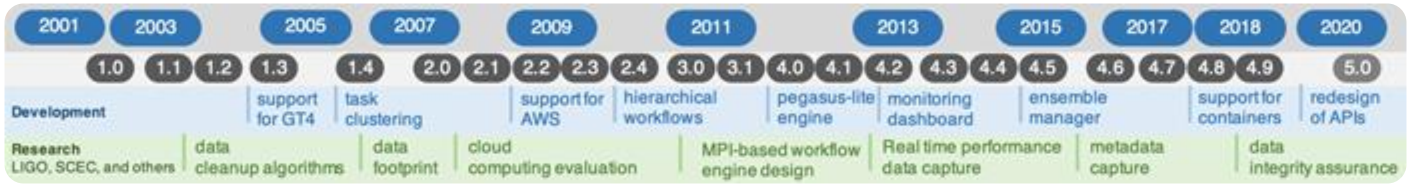
- Separation between workflow description and workflow execution
- Workflow planning and scheduling (scalability, performance)
- Task execution (monitoring, fault tolerance, debugging, web dashboard)
- Provide additional assurances that a scientific workflow is not accidentally or maliciously tampered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

Pegasus Workflow Management System

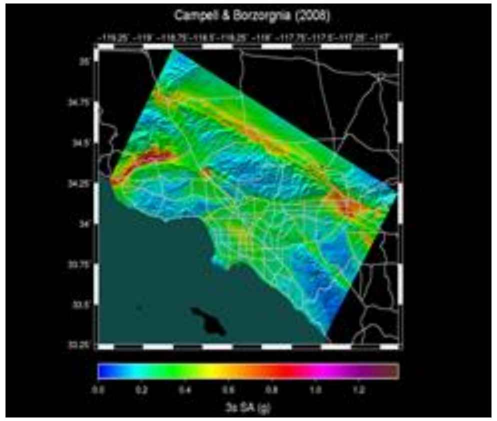


End to End Workflow Management & Execution

- 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



The XENONnT detector

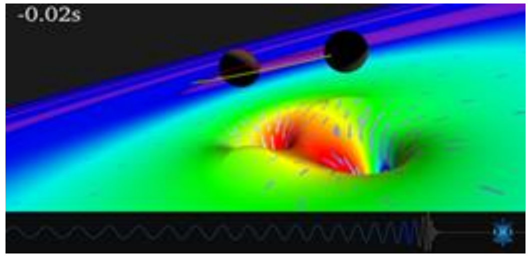


Hazard map indicating maximum amount of shaking at a particular geographic location generated from SCEC's CyberShake Pegasus workflow

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

LIGO observation of colliding black holes



renci USC Viterbi School of Engineering Information Sciences Institute CHESS CORNELL HIGH ENERGY SYNCHROTRON SOURCE

Funded by the National Science Foundation, Grant #2320373

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

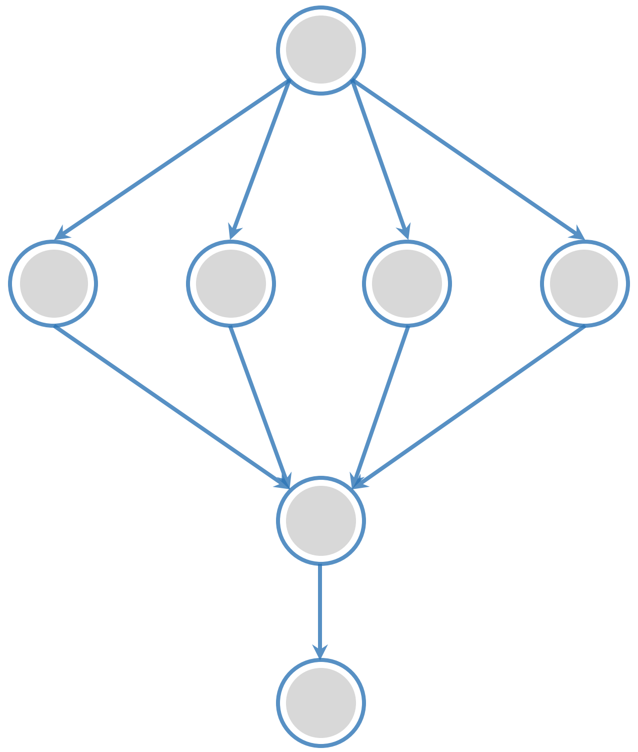
Workflows are DAGs

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

Planning occurs ahead of execution

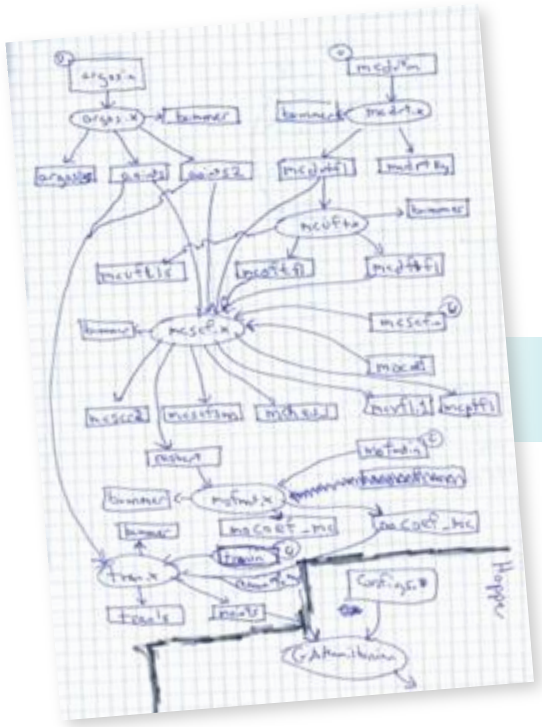
Planning converts an abstract workflow into a concrete, executable workflow

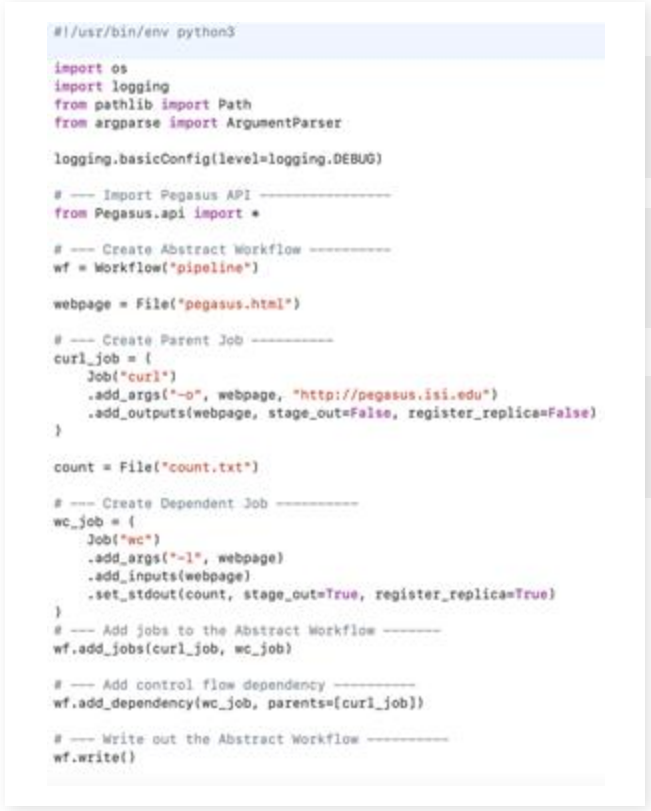
- Planner is like a compiler.



Funded by the National Science Foundation, Grant #2320373

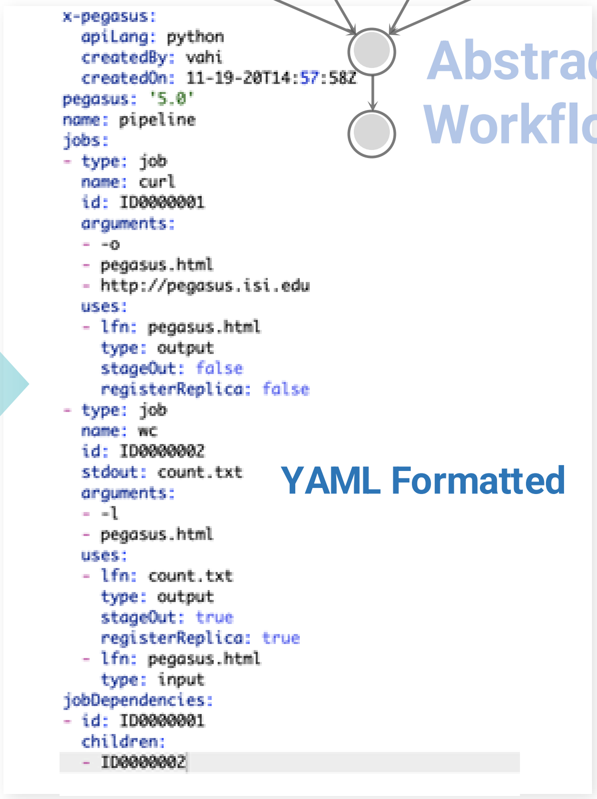
Pegasus provides APIs to generate the Abstract Workflow



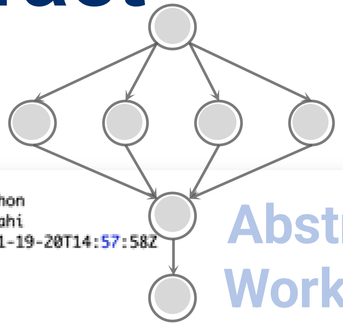


```python
#!/usr/bin/env python3

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



```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
```



Abstract Workflow

YAML Formatted

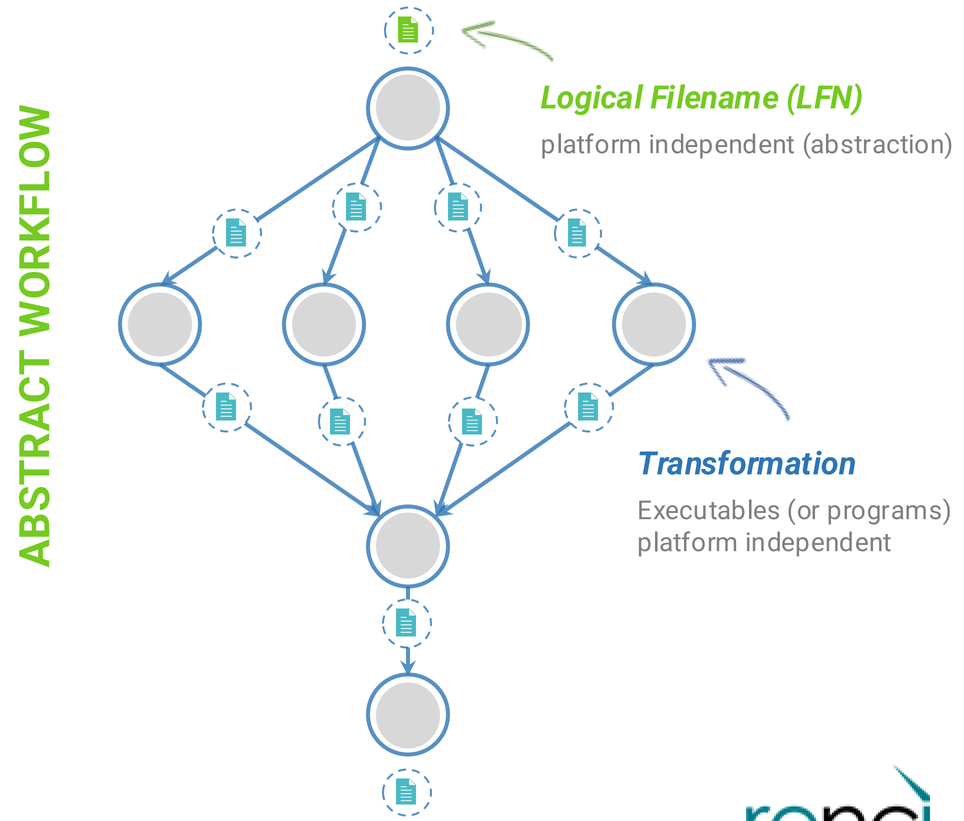
Funded by the National Science Foundation, Grant #2320373

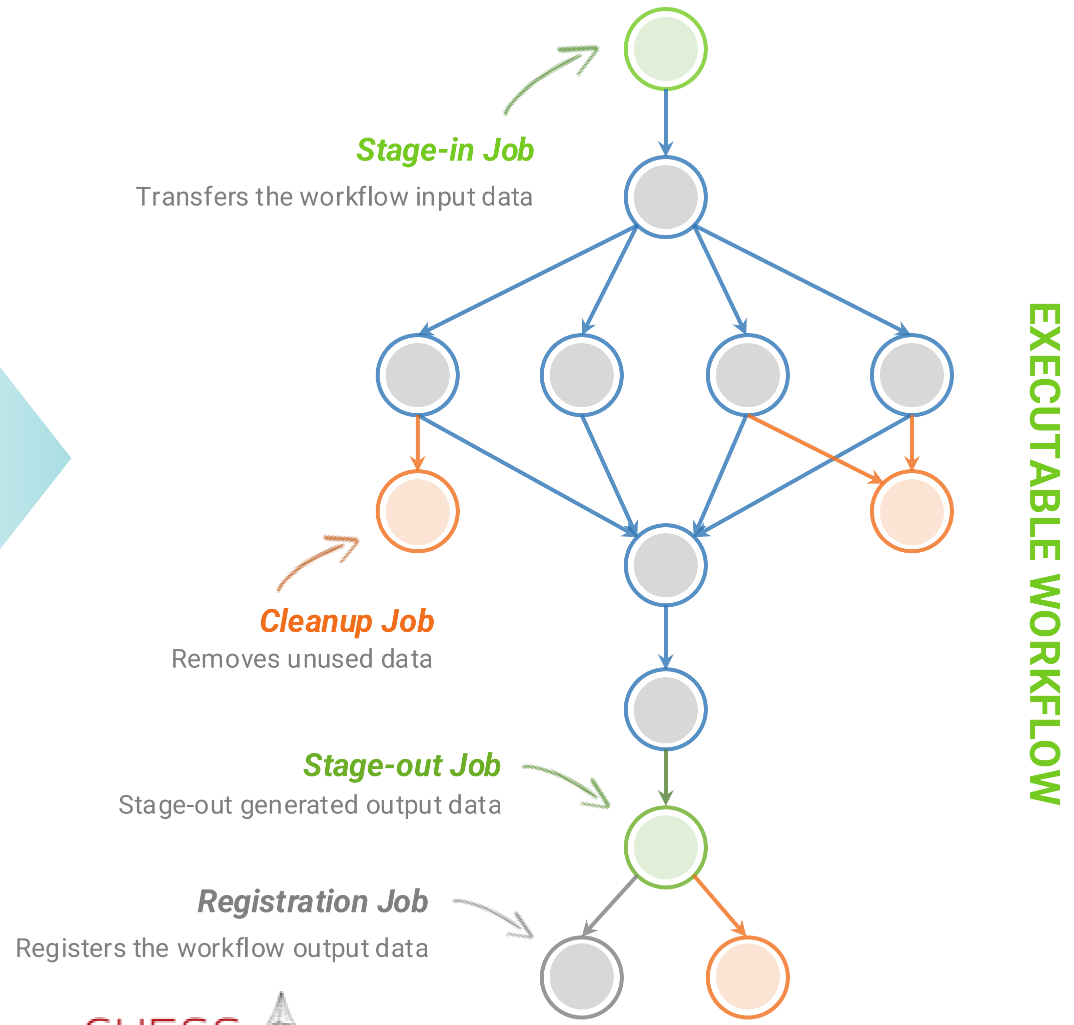
Input Workflow Specification YAML formatted

Output Workflow

Portable Description

Users do not worry about low level execution details




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

Funded by the National Science Foundation, Grant #2320373

So, what other information does Pegasus need?

Site Catalog

Describes the sites where
The workflow jobs are to be executed

Transformation Catalog

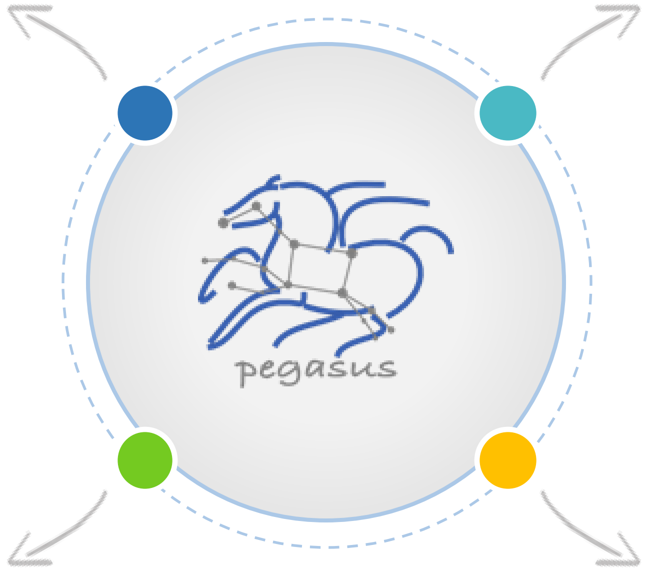
Describes all of the executables
(called “transformations”)
used by the workflow

Replica Catalog

Describes all of the input data stored
on external servers

Funded by the National Science Foundation, Grant #2320373

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

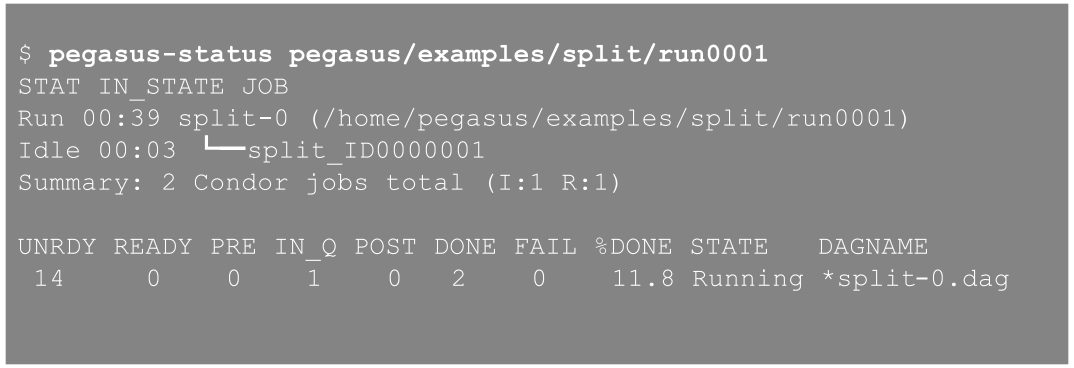
job generates checkpoint files staging of checkpoint files is automatic on restarts

Rescue DAGs

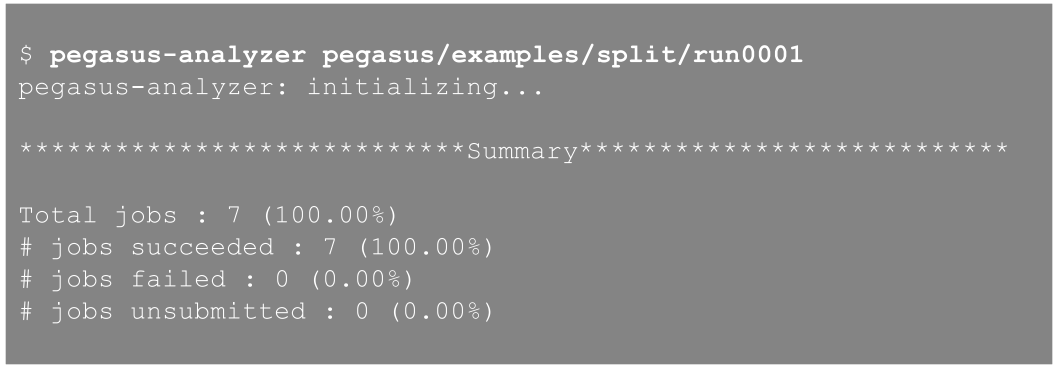
workflow can be restarted from checkpoint file recover from failures with minimal loss

Funded by the National Science Foundation, Grant #2320373

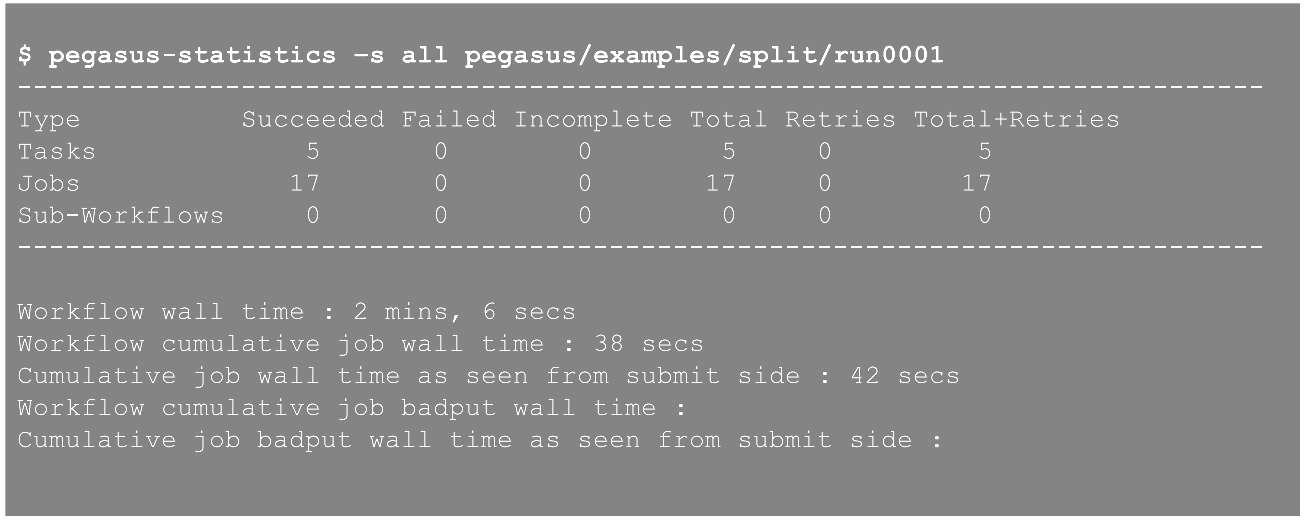
command-line…



```
$ 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

Funded by the National Science Foundation, Grant #2320373

Pegasus Deployment

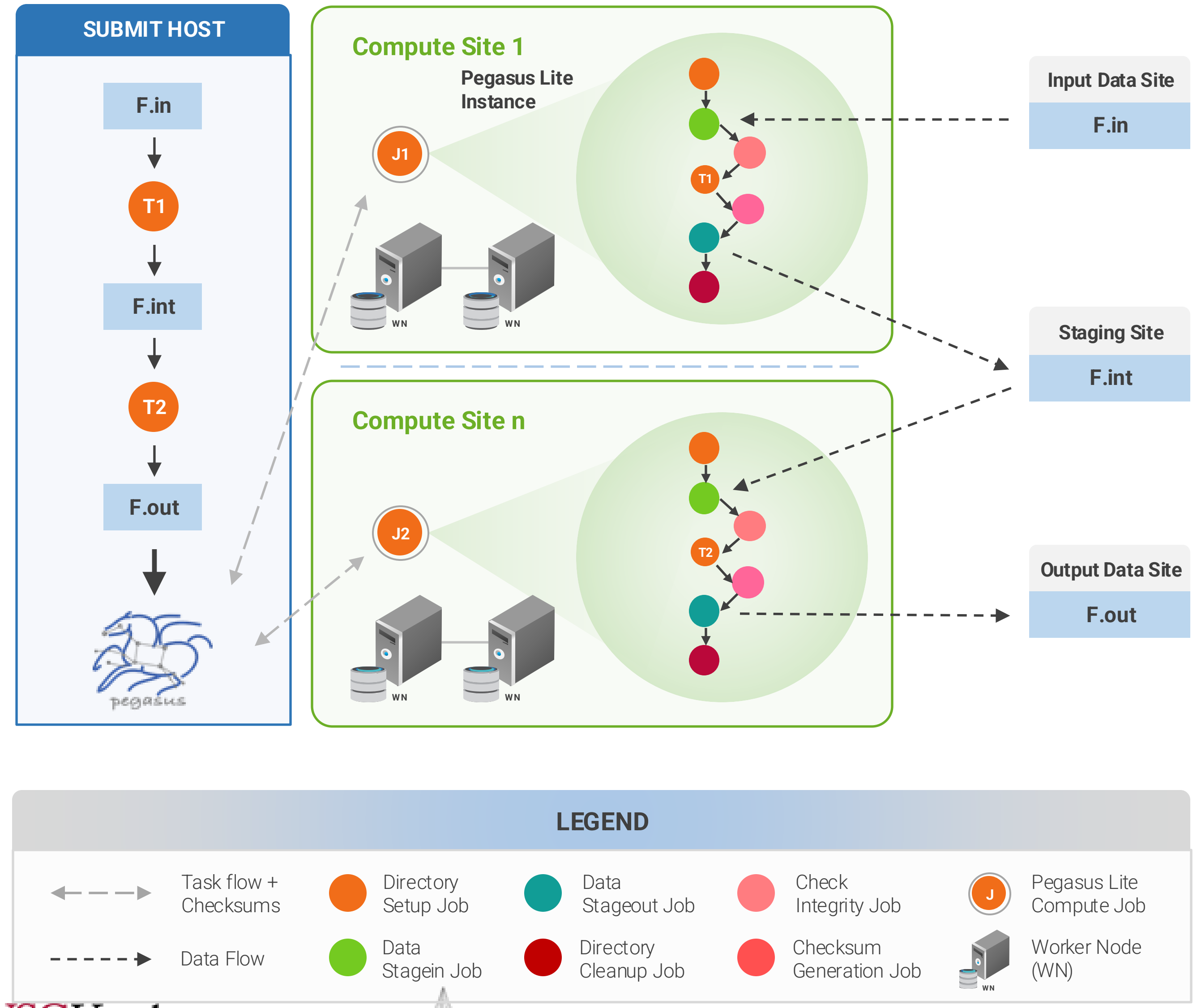
Workflow Submit Node
- Pegasus WMS
- HTCondor
- *lnx201.classe.cornell.edu*

One or more Compute Sites
- CHESS Compute Cluster
- Cloud
- OSG

Input Sites
- Host Input Data
- Local Shared Filesystem

Data Staging Site
- Coordinate data movement for workflow

Output Site
- Where output data is placed
- Globus Online Endpoint
- Directory on the filesystem



Funded by the National Science Foundation, Grant #2320373

Hands on: Running our first workflow

Submit Host: Logon to *lnx201.classe.cornell.edu*

Reference Materials:
https://xcitecourse.org/theme3/DC101/scientific-workflow-management.html#pegasus-workflows

Jump to : Getting Started with Pegasus @ CHESS section.

Funded by the National Science Foundation, Grant #2320373

Data Staging Configurations

HTCondorIO (HTCondor pools, OSG, ...)

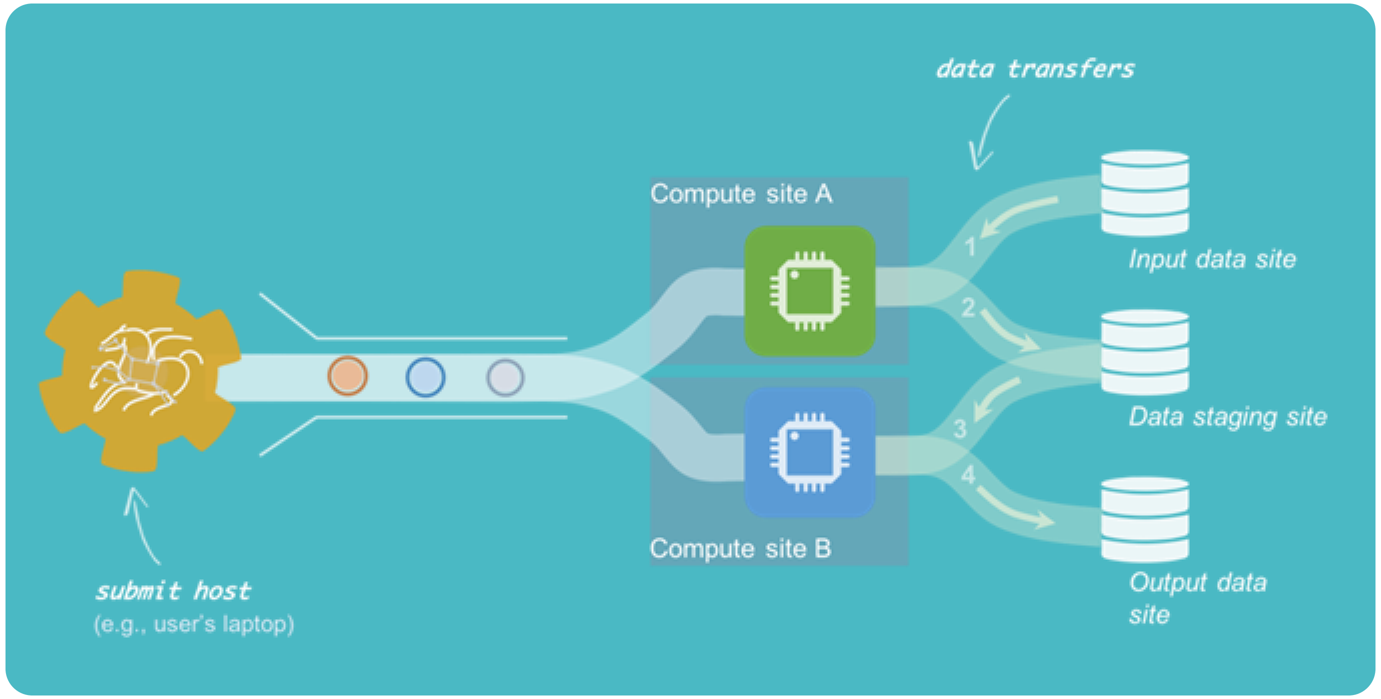
- Worker nodes do not share a file system
- Data is pulled / pushed from a staging site, possibly not co-located with the computation
- Staging site is the submit host

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
- Staging site is the submit host

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

- I/O is directly against the shared file system



Funded by the National Science Foundation, Grant #2320373

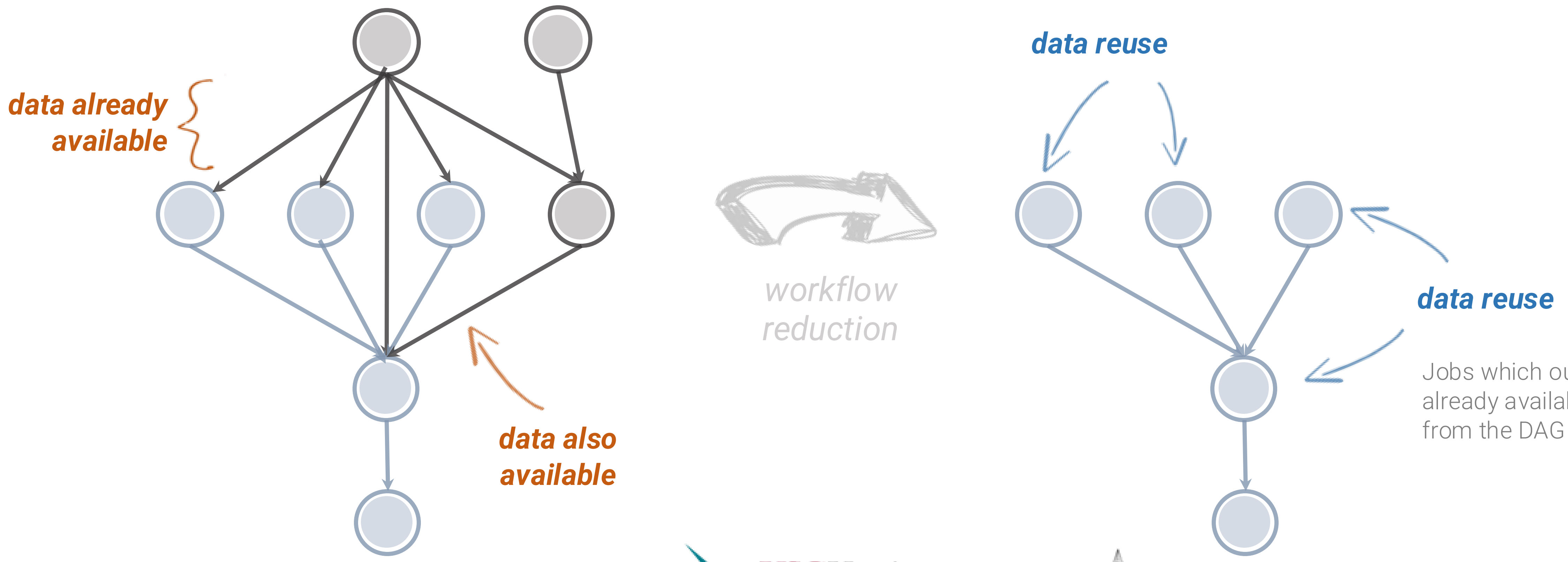
pegasus-transfer

- **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
```

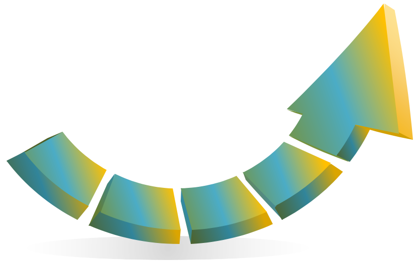
Funded by the National Science Foundation, Grant #2320373

Data Reuse prune jobs if output data already exists

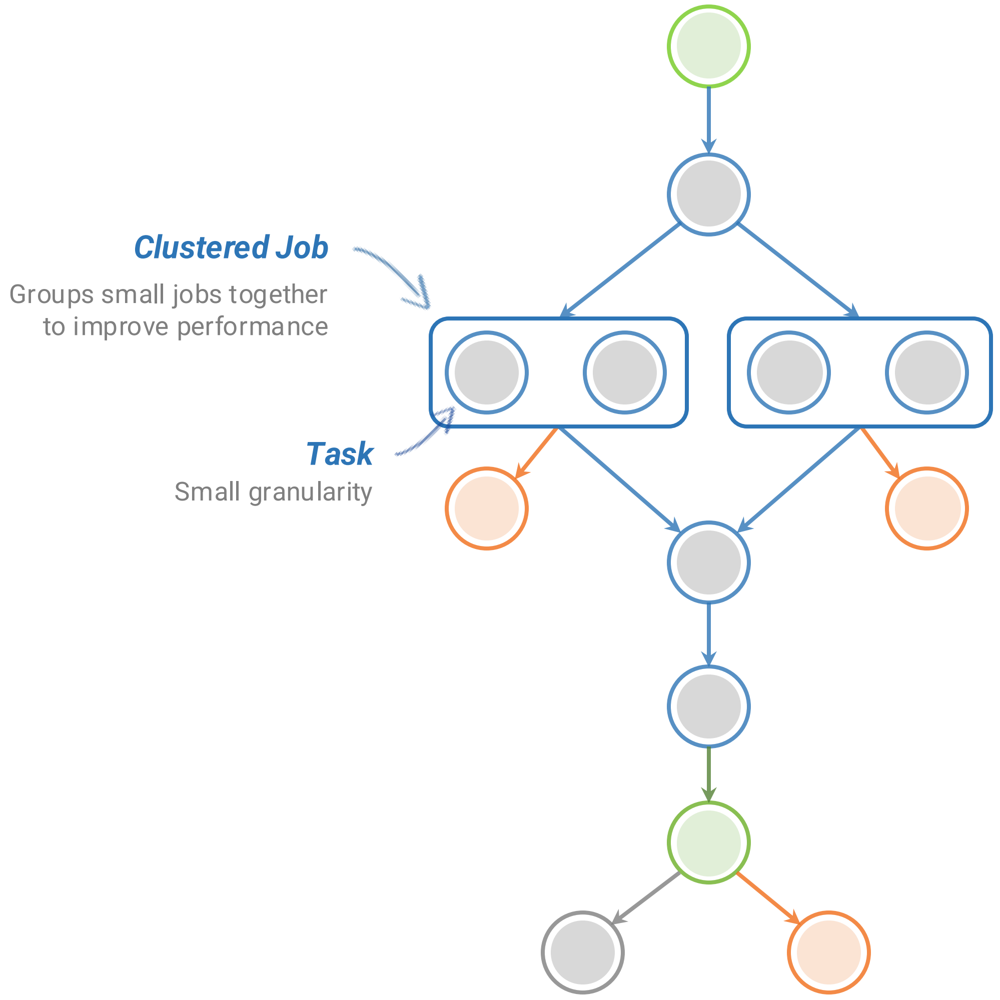


Jobs which output data is already available are pruned from the DAG

Funded by the National Science Foundation, Grant #2320373



Performance.
Why not improve it?



Funded by the National Science Foundation, Grant #2320373

QM2 Beamline

Introductory Video:
https://drive.google.com/file/d/1JCcf66AzHM3XdPb1MLkNhQxcyU85WDtM/view

Funded by the National Science Foundation, Grant #2320373

Pegasus CHESS QM Workflow

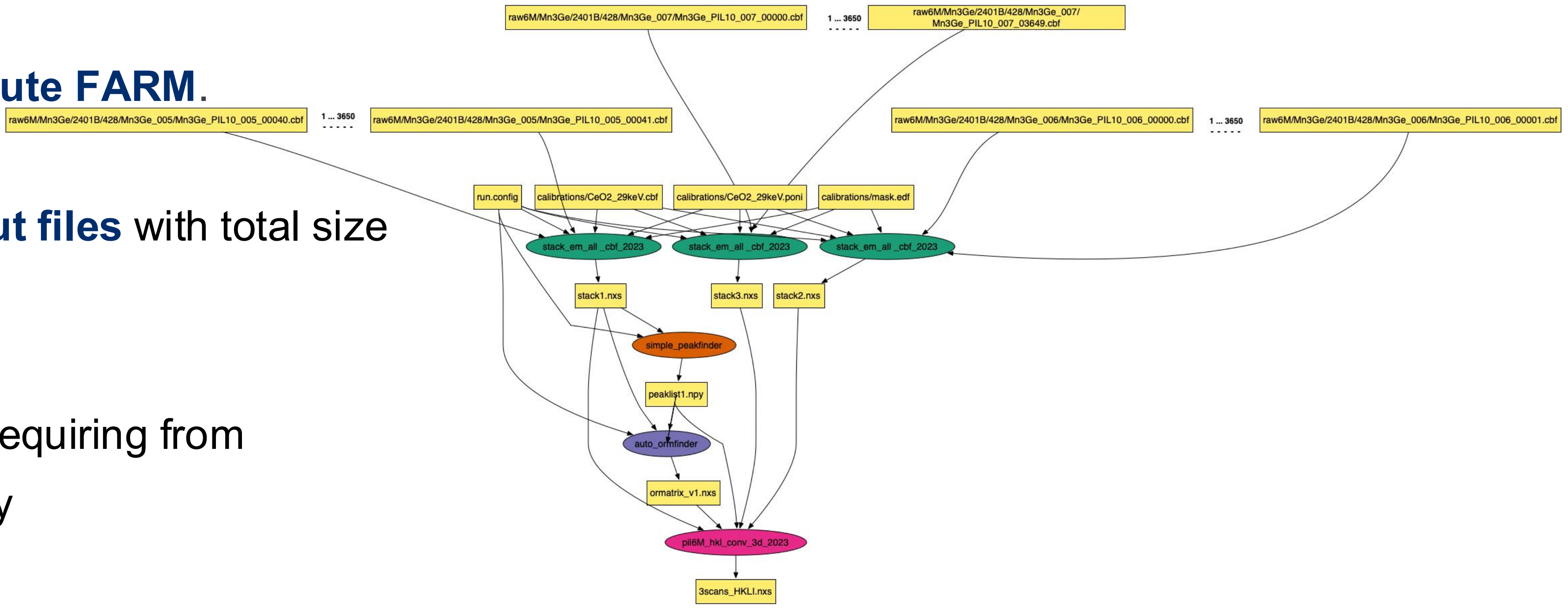
GitHub Repository: https://github.com/pegasus-isi/chess-qmb-workflow

Runs on the **CHESS Compute FARM**.

Requires about **11,000 input files** with total size of approximately **570GB**.

Mainly **high memory** jobs requiring from **10GB to 350 GB** of memory

The **pil6M_hkl_conv** job requries 56 cores.


raw6M/Mn3Ge/2401B/428/Mn3Ge_007/Mn3Ge_PIL10_007_00000.cbf
1 ... 3650
raw6M/Mn3Ge/2401B/428/Mn3Ge_007/Mn3Ge_PIL10_007_03649.cbf
raw6M/Mn3Ge/2401B/428/Mn3Ge_005/Mn3Ge_PIL10_005_00040.cbf
1 ... 3650
raw6M/Mn3Ge/2401B/428/Mn3Ge_005/Mn3Ge_PIL10_005_00041.cbf
raw6M/Mn3Ge/2401B/428/Mn3Ge_006/Mn3Ge_PIL10_006_00000.cbf
1 ... 3650
raw6M/Mn3Ge/2401B/428/Mn3Ge_006/Mn3Ge_PIL10_006_00001.cbf
run.config
calibrations/CeO2_29keV.cbf
calibrations/CeO2_29keV.poni
calibrations/mask.edf
stack_em_all _cbf_2023
stack_em_all _cbf_2023
stack_em_all _cbf_2023
stack1.nxs
stack3.nxs
stack2.nxs
simple_peakfinder
peaklist1.npy
auto_ormfinder
ormatrix_v1.nxs
pil6M_hkl_conv_3d_2023
3scans_HKLI.nxs

Funded by the National Science Foundation, Grant #2320373

Thank you!

XCITE Workflows Module:
https://xcitecourse.org/theme3/DC101/scientific-workflow-management.html

Website: https://pegasus.isi.edu

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

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

National Science Foundation

NSF Cybertraining Award # 2320373