Workflows using Pegasus

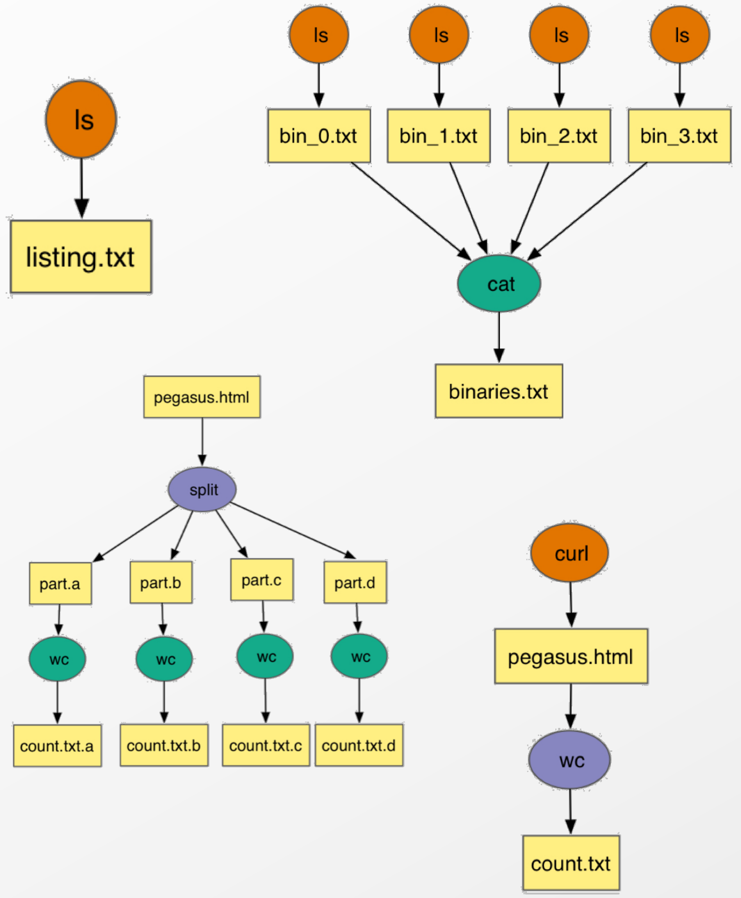
Pegasus Workflow Management System

Karan Vahi

USC Viterbi
School of Engineering
Information Sciences Institute

http://pegasus.isi.edu

Compute Pipelines Building Blocks



Compute Pipelines

- Allows scientists to connect different codes together and execute their analysis
- Pipelines can be very simple (independent or parallel) jobs or complex represented as DAG's
- Helps users to automate scale up

However, it is still up-to user to figure out

Data Management

- How do you ship in the small/large amounts data required by your pipeline and protocols to use?

How best to leverage different infrastructure setups

- OSG has no shared filesystem while XSEDE and your local campus cluster has one!

Debug and Monitor Computations

- Correlate data across lots of log files
- Need to know what host a job ran on and how it was invoked

Restructure Workflows for Improved Performance

- Short running tasks? Data placement

Pegasus

Why Pegasus ?

Automates complex, multi-stage processing pipelines

Enables parallel, **distributed computations**

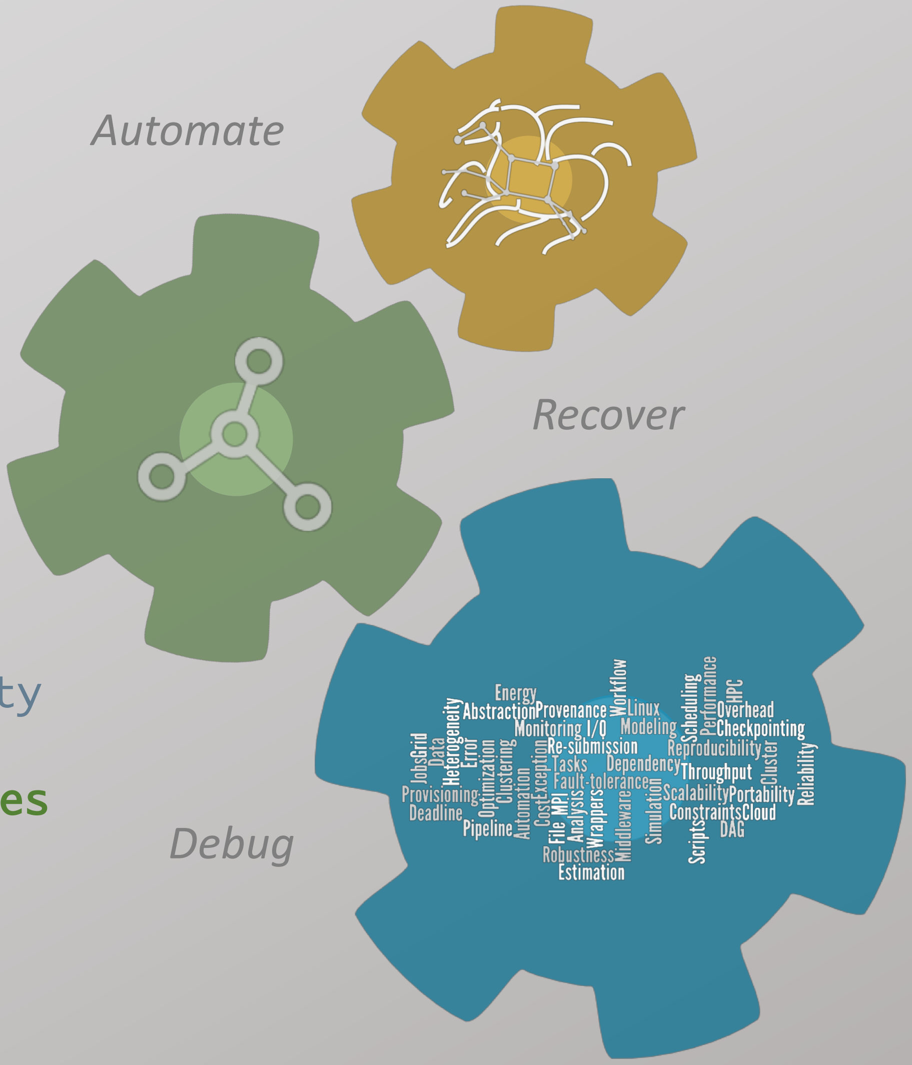
Automatically executes data transfers

Reusable, aids **reproducibility**

Records how data was produced (**provenance**)

Handles **failures** with to provide reliability

Keeps track of data and **files**



NSF funded project since 2001, with close collaboration with HTCondor team

Basic concepts…

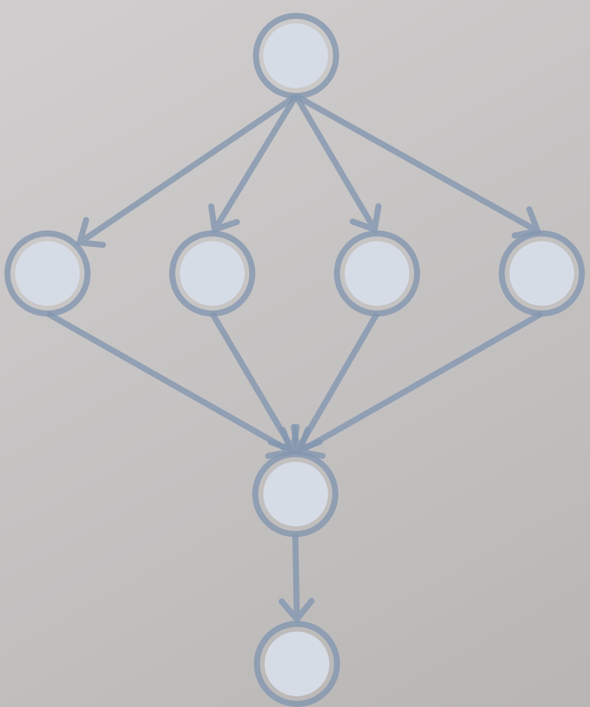
Key Pegasus Concepts

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

DAX

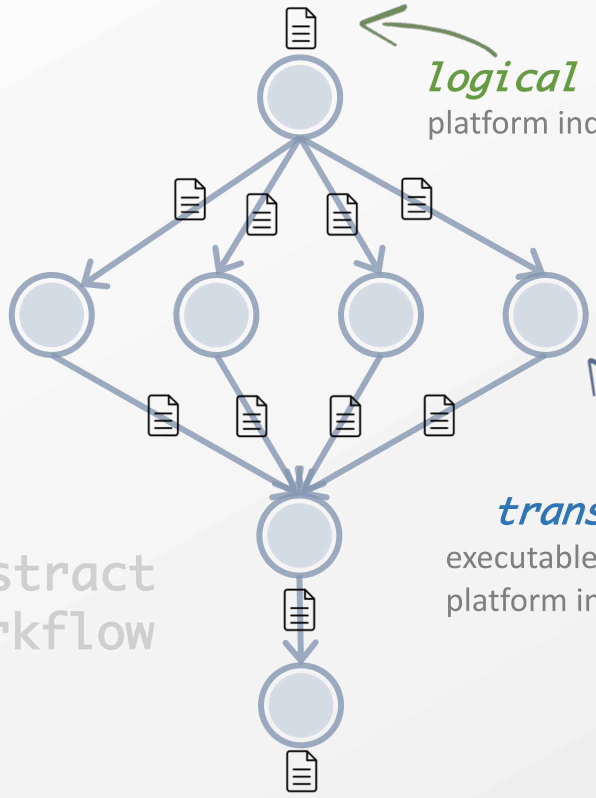
DAG in XML

DAG

directed-acyclic graphs

Portable Description

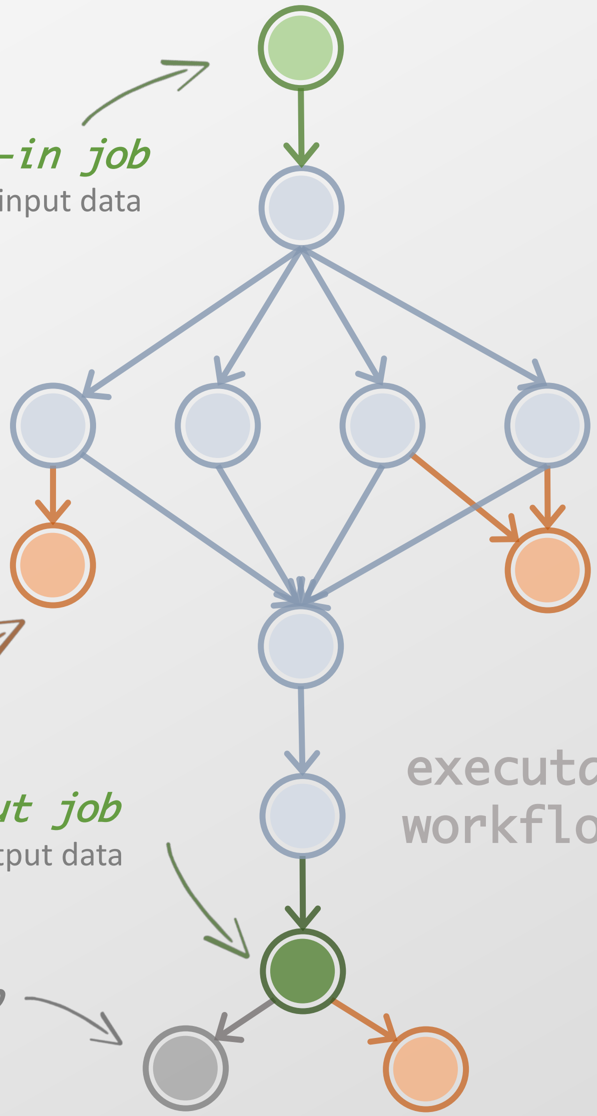
Users do not worry about
low level execution details



logical filename (LFN)
platform independent (abstraction)

transformation
executables (or programs)
platform independent

abstract
workflow



stage-in job
Transfers the workflow input data

cleanup job
Removes unused data

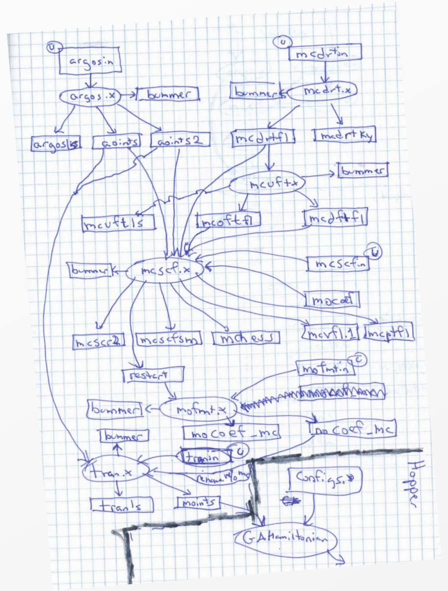
stage-out job
Transfers the workflow output data

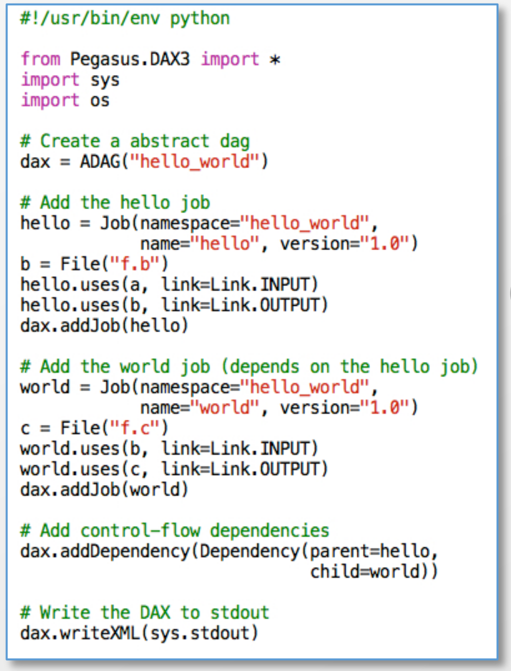
registration job
Registers the workflow output data

executable
workflow

Pegasus

Pegasus also provides tools to generate the abstract workflow





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

from Pegasus.DAX3 import *
import sys
import os

# Create a abstract dag
dax = ADAG("hello_world")

# Add the hello job
hello = Job(namespace="hello_world",
            name="hello", version="1.0")
b = File("f.b")
hello.uses(a, link=Link.INPUT)
hello.uses(b, link=Link.OUTPUT)
dax.addJob(hello)

# Add the world job (depends on the hello job)
world = Job(namespace="hello_world",
            name="world", version="1.0")
c = File("f.c")
world.uses(b, link=Link.INPUT)
world.uses(c, link=Link.OUTPUT)
dax.addJob(world)

# Add control-flow dependencies
dax.addDependency(Dependency(parent=hello,
                             child=world))

# Write the DAX to stdout
dax.writeXML(sys.stdout)
```

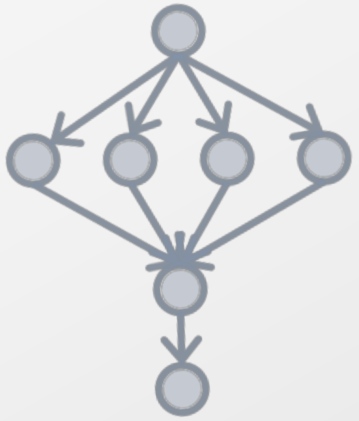
python™

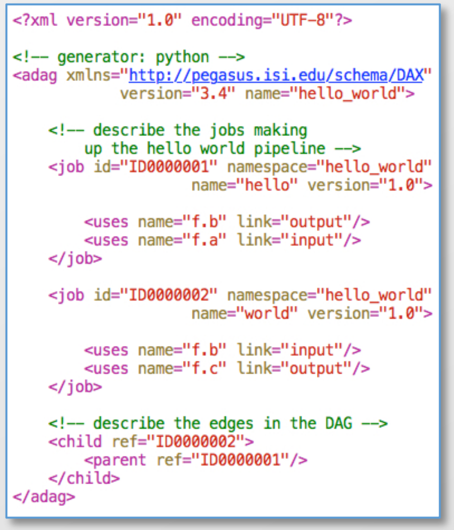
Java

R

perl

jupyter





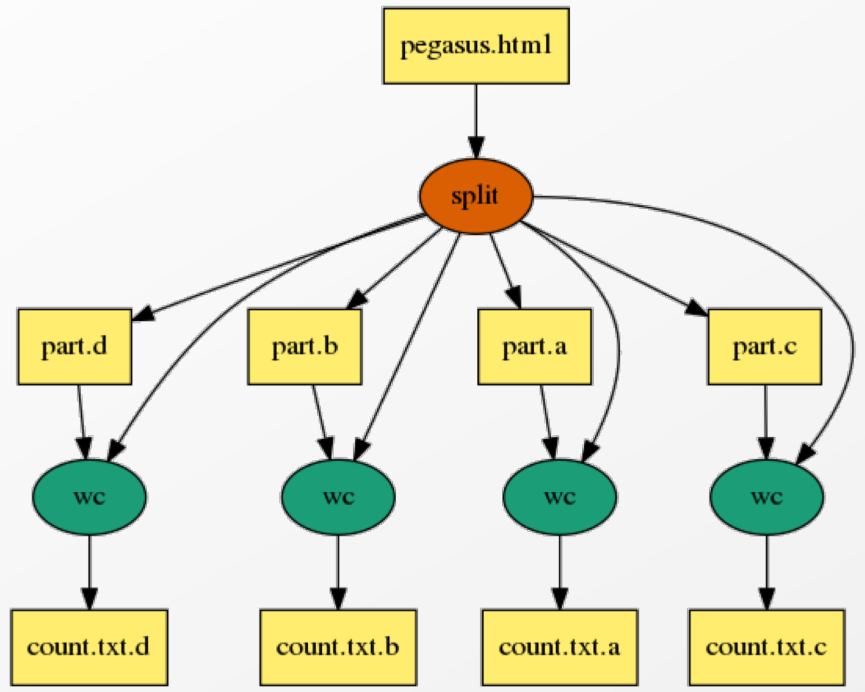
```
<?xml version="1.0" encoding="UTF-8"?>

<!-- generator: python -->
<adag xmlns="http://pegasus.isi.edu/schema/DAX"
            version="3.4" name="hello_world">

    <!-- describe the jobs making
        up the hello world pipeline -->
    <job id="ID0000001" namespace="hello_world"
                name="hello" version="1.0">

        <uses name="f.b" link="output"/>
        <uses name="f.a" link="input"/>
    </job>

    <job id="ID0000002" namespace="hello_world"
                name="world" version="1.0">

        <uses name="f.b" link="input"/>
        <uses name="f.c" link="output"/>
    </job>

    <!-- describe the edges in the DAG -->
    <child ref="ID0000002">
        <parent ref="ID0000001"/>
    </child>
</adag>
```

DAX

DAG in XML

Pegasus

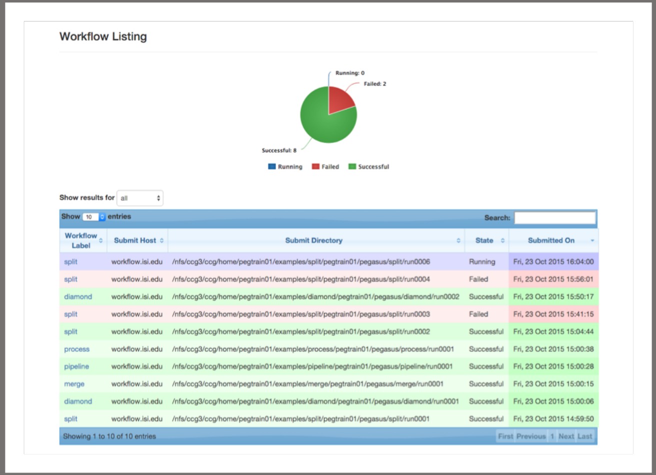
An example
Split Workflow



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

import os, pwd, sys, time
from Pegasus.DAX3 import *

# Create an abstract dag
dax = ADAG("split")

webpage = File("pegasus.html")

# the split job that splits the webpage into smaller chunks
split = Job("split")
split.addArguments("-l","100","-a","1",webpage,"part.")
split.uses(webpage, link=Link.INPUT)
# associate the label with the job. all jobs with same label
# are run with PMC when doing job clustering
split.addProfile( Profile("pegasus","label","p1"))
dax.addJob(split)

# we do a parmeter sweep on the first 4 chunks created
for c in "abcd":
    part = File("part.%s" % c)
    split.uses(part, link=Link.OUTPUT, transfer=False, register=False)
    count = File("count.txt.%s" % c)
    wc = Job("wc")
    wc.addProfile( Profile("pegasus","label","p1"))
    wc.addArguments("-l",part)
    wc.setStdout(count)
    wc.uses(part, link=Link.INPUT)
    wc.uses(count, link=Link.OUTPUT, transfer=True, register=True)
    dax.addJob(wc)

    #adding dependency
    dax.depends(wc, split)

f = open(“split.dax”, "w")
dax.writeXML(f)
f.close()
```

Visualization Tools:
pegasus-graphviz
pegasus-plots

https://pegasus.isi.edu/documentation/tutorial_submitting_wf.php



Workflow Listing

Show results for all

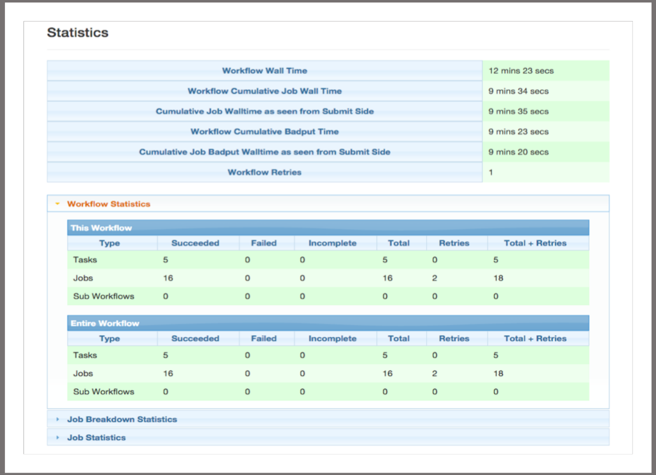
Show 10 entries Search:

Workflow Label	Submit Host	Submit Directory	State	Submitted On
split	workflow.isi.edu	/nfs/ccg3/ccg/home/pegtrain01/examples/split/pegtrain01/pegasus/split/run0006	Running	Fri, 23 Oct 2015 16:04:00
split	workflow.isi.edu	/nfs/ccg3/ccg/home/pegtrain01/examples/split/pegtrain01/pegasus/split/run0004	Failed	Fri, 23 Oct 2015 15:56:01
diamond	workflow.isi.edu	/nfs/ccg3/ccg/home/pegtrain01/examples/diamond/pegtrain01/pegasus/diamond/run0002	Successful	Fri, 23 Oct 2015 15:50:17
split	workflow.isi.edu	/nfs/ccg3/ccg/home/pegtrain01/examples/split/pegtrain01/pegasus/split/run0003	Failed	Fri, 23 Oct 2015 15:41:15
split	workflow.isi.edu	/nfs/ccg3/ccg/home/pegtrain01/examples/split/pegtrain01/pegasus/split/run0002	Successful	Fri, 23 Oct 2015 15:04:44
process	workflow.isi.edu	/nfs/ccg3/ccg/home/pegtrain01/examples/process/pegtrain01/pegasus/process/run0001	Successful	Fri, 23 Oct 2015 15:00:38
pipeline	workflow.isi.edu	/nfs/ccg3/ccg/home/pegtrain01/examples/pipeline/pegtrain01/pegasus/pipeline/run0001	Successful	Fri, 23 Oct 2015 15:00:28
merge	workflow.isi.edu	/nfs/ccg3/ccg/home/pegtrain01/examples/merge/pegtrain01/pegasus/merge/run0001	Successful	Fri, 23 Oct 2015 15:00:15
diamond	workflow.isi.edu	/nfs/ccg3/ccg/home/pegtrain01/examples/diamond/pegtrain01/pegasus/diamond/run0001	Successful	Fri, 23 Oct 2015 15:00:06
split	workflow.isi.edu	/nfs/ccg3/ccg/home/pegtrain01/examples/split/pegtrain01/pegasus/split/run0001	Successful	Fri, 23 Oct 2015 14:59:50

Showing 1 to 10 of 10 entries First Previous 1 Next Last

Pegasus dashboard

web interface for monitoring and debugging workflows



Statistics

Workflow Wall Time	12 mins 23 secs
Workflow Cumulative Job Wall Time	9 mins 34 secs
Cumulative Job Walltime as seen from Submit Side	9 mins 35 secs
Workflow Cumulative Badput Time	9 mins 23 secs
Cumulative Job Badput Walltime as seen from Submit Side	9 mins 20 secs
Workflow Retries	1

Workflow Statistics

This Workflow

Type	Succeeded	Failed	Incomplete	Total	Retries	Total + Retries
Tasks	5	0	0	5	0	5
Jobs	16	0	0	16	2	18
Sub Workflows	0	0	0	0	0	0

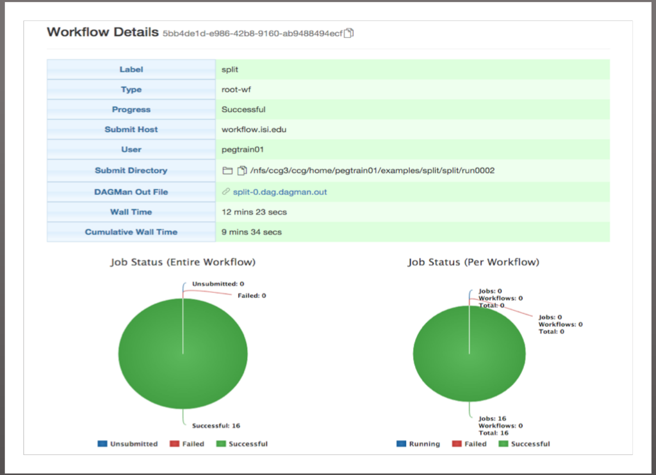
Entire Workflow

Type	Succeeded	Failed	Incomplete	Total	Retries	Total + Retries
Tasks	5	0	0	5	0	5
Jobs	16	0	0	16	2	18
Sub Workflows	0	0	0	0	0	0

Job Breakdown Statistics

Job Statistics

Real-time <u>monitoring</u> of workflow executions. It shows the <u>status</u> of the workflows and jobs, job <u>characteristics</u>, <u>statistics</u> and <u>performance</u> metrics. <u>Provenance</u> data is stored into a relational database.



Workflow Details 5bb4de1d-e986-42b8-9160-ab9488494ecf

Label	split
Type	root-wf
Progress	Successful
Submit Host	workflow.isi.edu
User	pegtrain01
Submit Directory	/nfs/ccg3/ccg/home/pegtrain01/examples/split/split/run0002
DAGMan Out File	split-0.dag.dagman.out
Wall Time	12 mins 23 secs
Cumulative Wall Time	9 mins 34 secs

Job Status (Entire Workflow)
Unsubmitted: 0
Failed: 0
Successful: 16
🟦 Unsubmitted 🟥 Failed 🟩 Successful
Job Status (Per Workflow)
Jobs: 0 Workflows: 0 Total: 0
Jobs: 0 Workflows: 0 Total: 0
Jobs: 16 Workflows: 0 Total: 16
🟦 Running 🟥 Failed 🟩 Successful

Real-time Monitoring

Reporting

Debugging

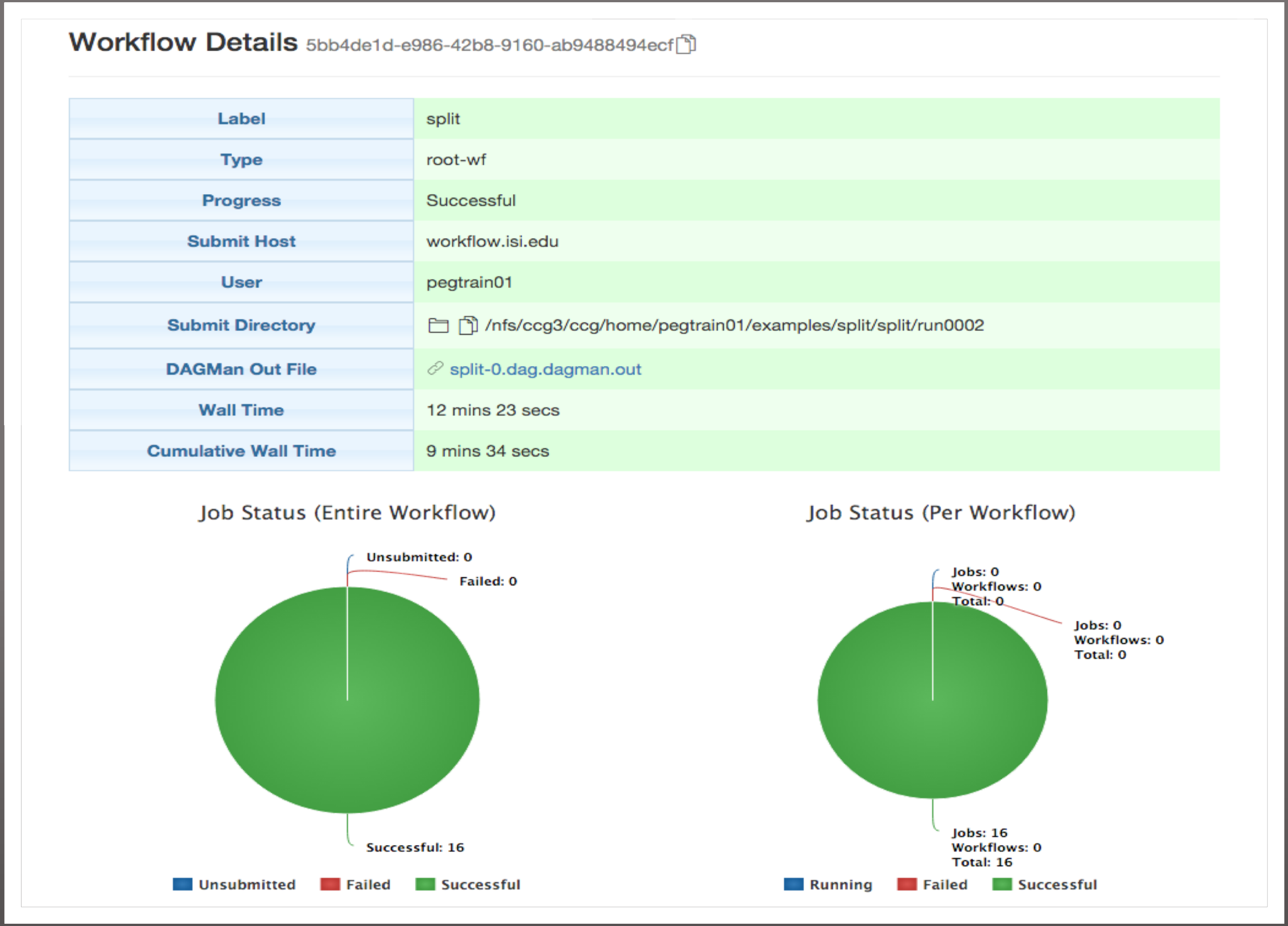
Troubleshooting

RESTful API

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.



Workflow Details 5bb4de1d-e986-42b8-9160-ab9488494ecf

Label	split
Type	root-wf
Progress	Successful
Submit Host	workflow.isi.edu
User	pegtrain01
Submit Directory	/nfs/ccg3/ccg/home/pegtrain01/examples/split/split/run0002
DAGMan Out File	split-0.dag.dagman.out
Wall Time	12 mins 23 secs
Cumulative Wall Time	9 mins 34 secs

>_ command-line…

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

UNRDY READY PRE IN_Q POST DONE FAIL %DONE STATE   DAGNAME
 14     0     0    1    0    2    0    11.8 Running *split-0.dag
```

```
$ pegasus-analyzer pegasus/examples/split/run0001
pegasus-analyzer: initializing...

****************************Summary***************************

Total jobs : 7 (100.00%)
# jobs succeeded : 7 (100.00%)
# jobs failed : 0 (0.00%)
# jobs unsubmitted : 0 (0.00%)
```

```
$ pegasus-statistics –s all pegasus/examples/split/run0001
------------------------------------------------------------------------------
Type           Succeeded Failed Incomplete Total Retries Total+Retries
Tasks               5        0          0       5      0            5
Jobs               17        0          0      17      0           17
Sub-Workflows       0        0          0       0      0            0
------------------------------------------------------------------------------

Workflow wall time : 2 mins, 6 secs
Workflow cumulative job wall time : 38 secs
Cumulative job wall time as seen from submit side : 42 secs
Workflow cumulative job badput wall time :
Cumulative job badput wall time as seen from submit side :
```

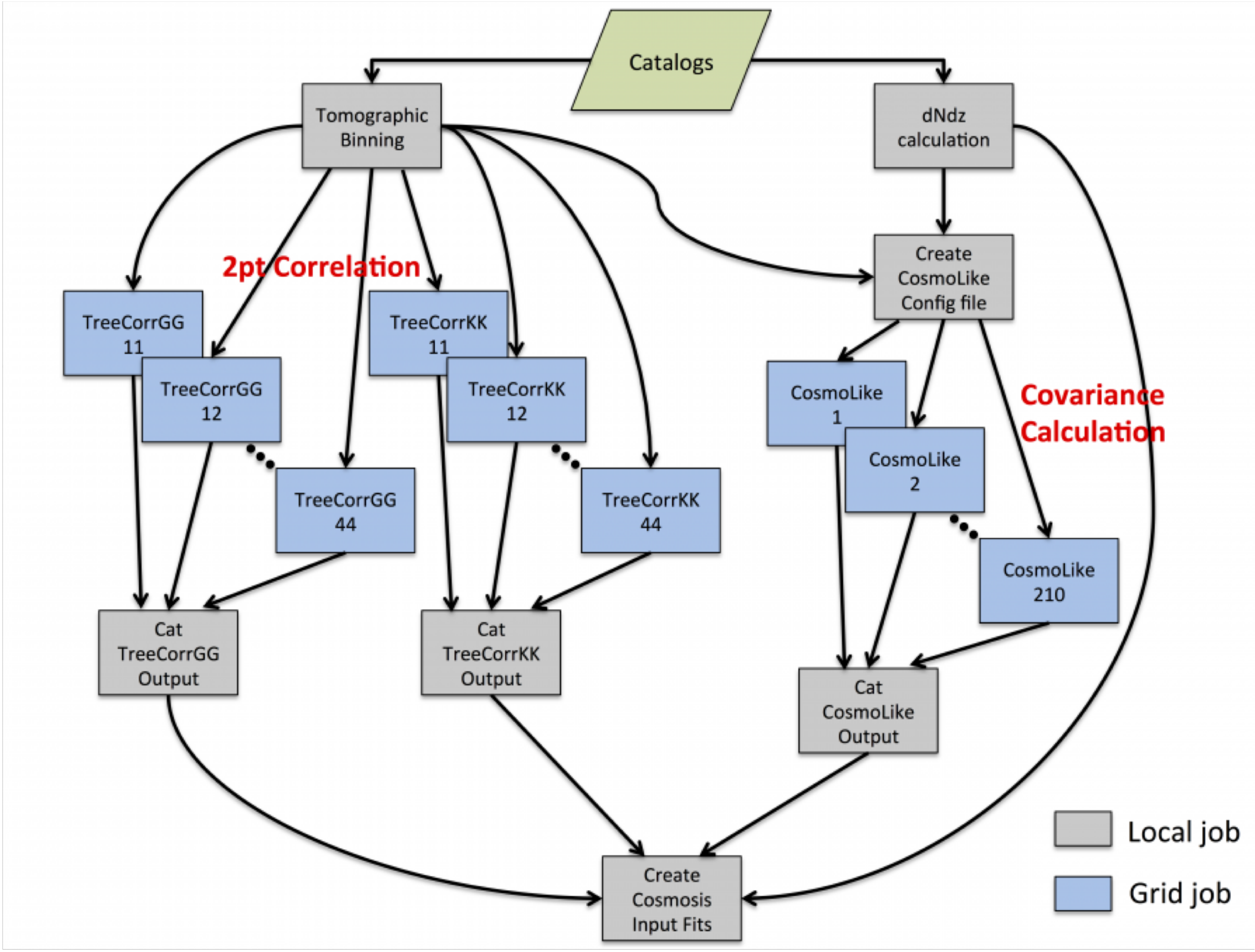
Provenance data can be summarized
pegasus-statistics

or used for debugging
pegasus-analyzer

Hands-on Demo…

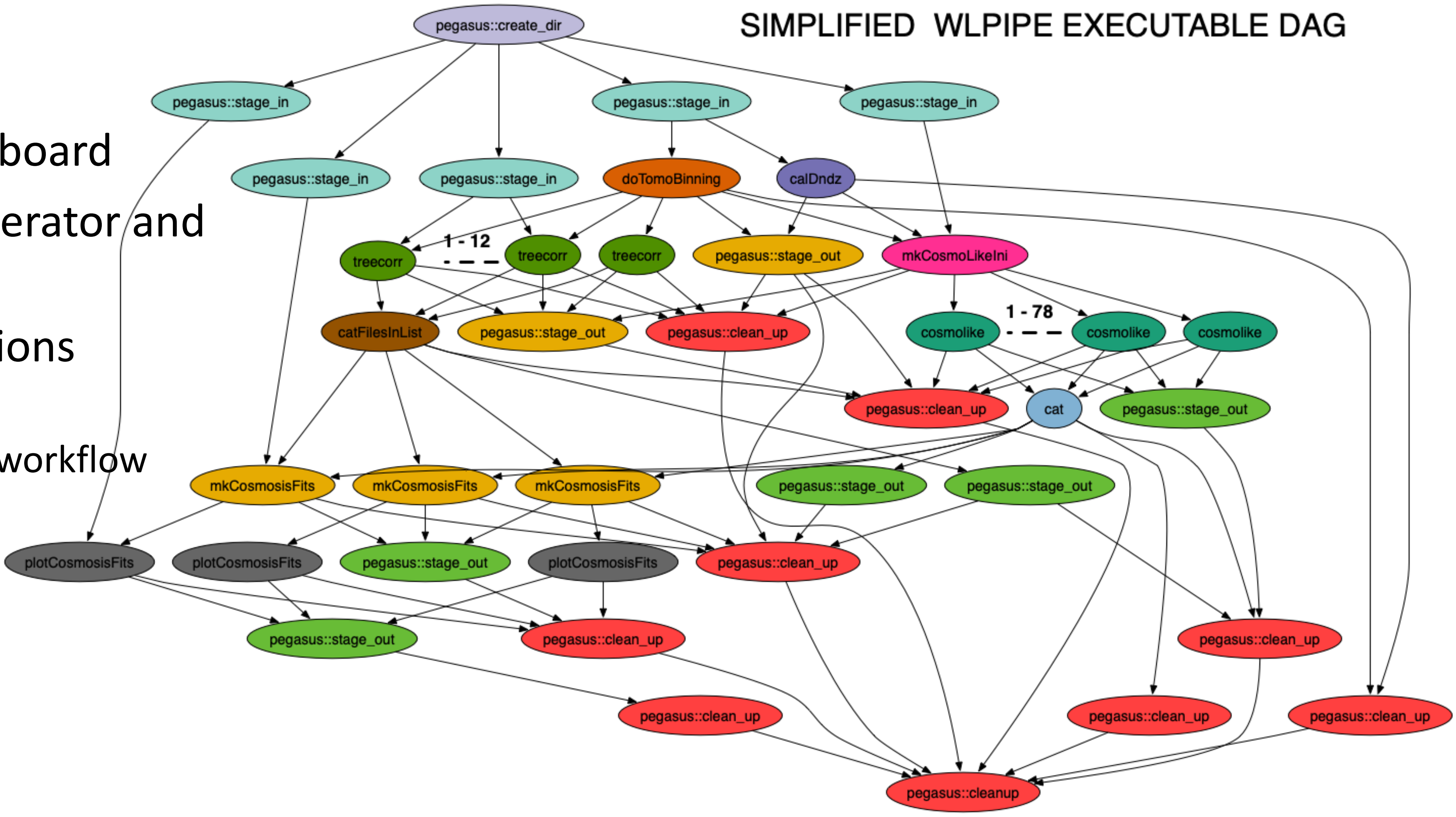
Hands On Demo

- Weak Lensing Pipeline
  - https://github.com/pegasus-isi/pegasus-wlpipe
- An example of a typical gravitational weak lensing analysis. It uses publicly available Science Verification catalogs the Dark Energy Survey (DES-SV).
- The pipeline is run currently at Fermi Grid
- We will run the example version at a cluster at ISI
- Science Codes are bundled into a Singularity Container
- This is an interactive session. Please interrupt at anytime to ask questions.

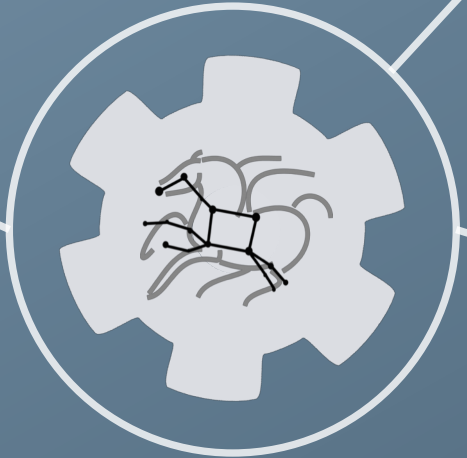


Outline

- Submit a workflow
- Go through the dashboard
- Go over the DAX generator and catalogs
- Show debugging options
  - pegasus-analyzer
  - Recover from failed workflow
- pegasus-statistics



So, what information does Pegasus need?



Site Catalog

describes the sites where the workflow jobs are to be executed

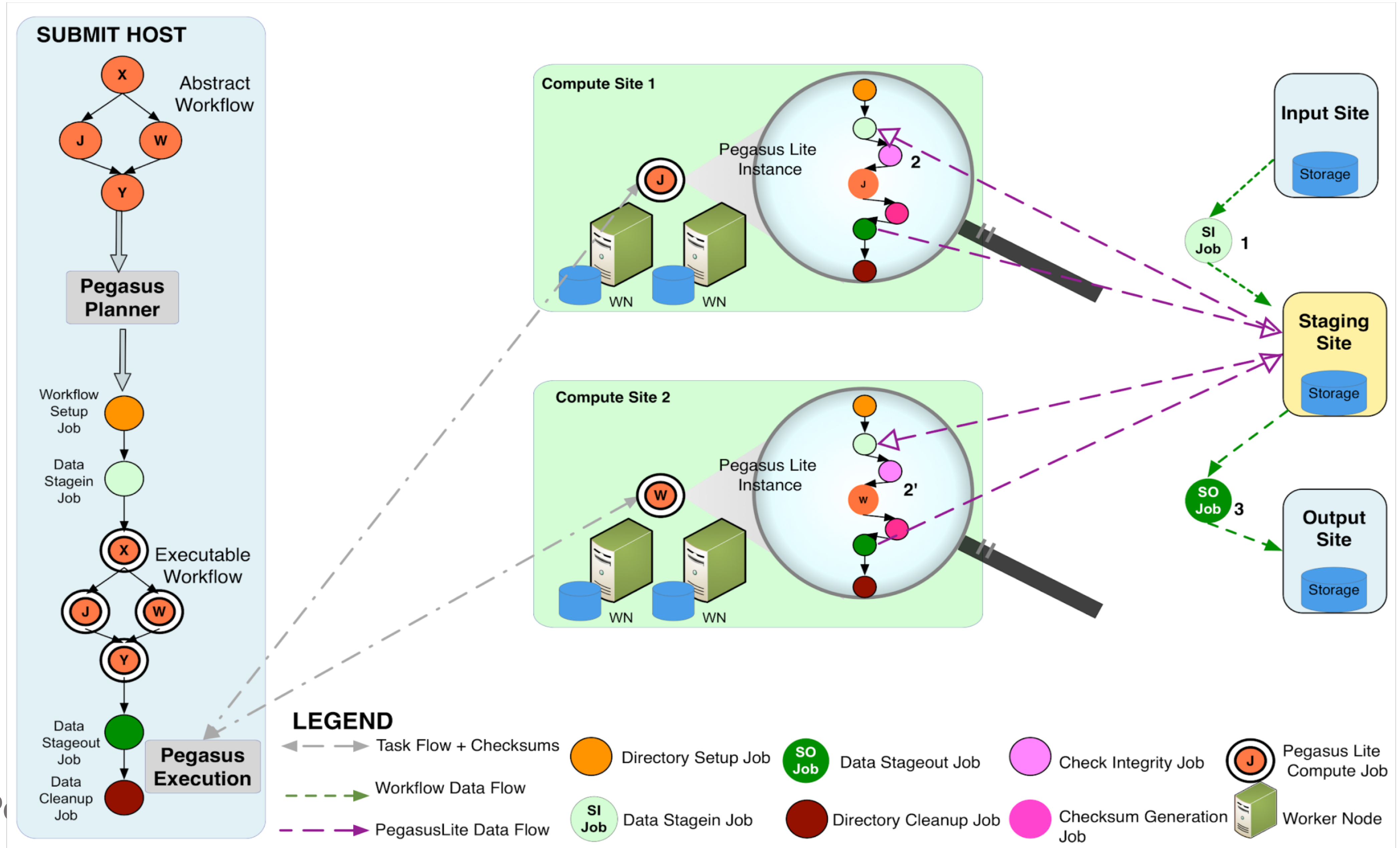
Transformation Catalog

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

Replica Catalog

describes all of the input data stored on external servers

Distributed Execution



Job Submissions

Local

Submit Machine
Personal HTCondor

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

Remote

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

*BOSCO based Glideins***
SSH based submission of Glideins

PyGlidein
ICE Cube Glidein service

OSG using glideinWMS

CREAMCE
Uses CondorG

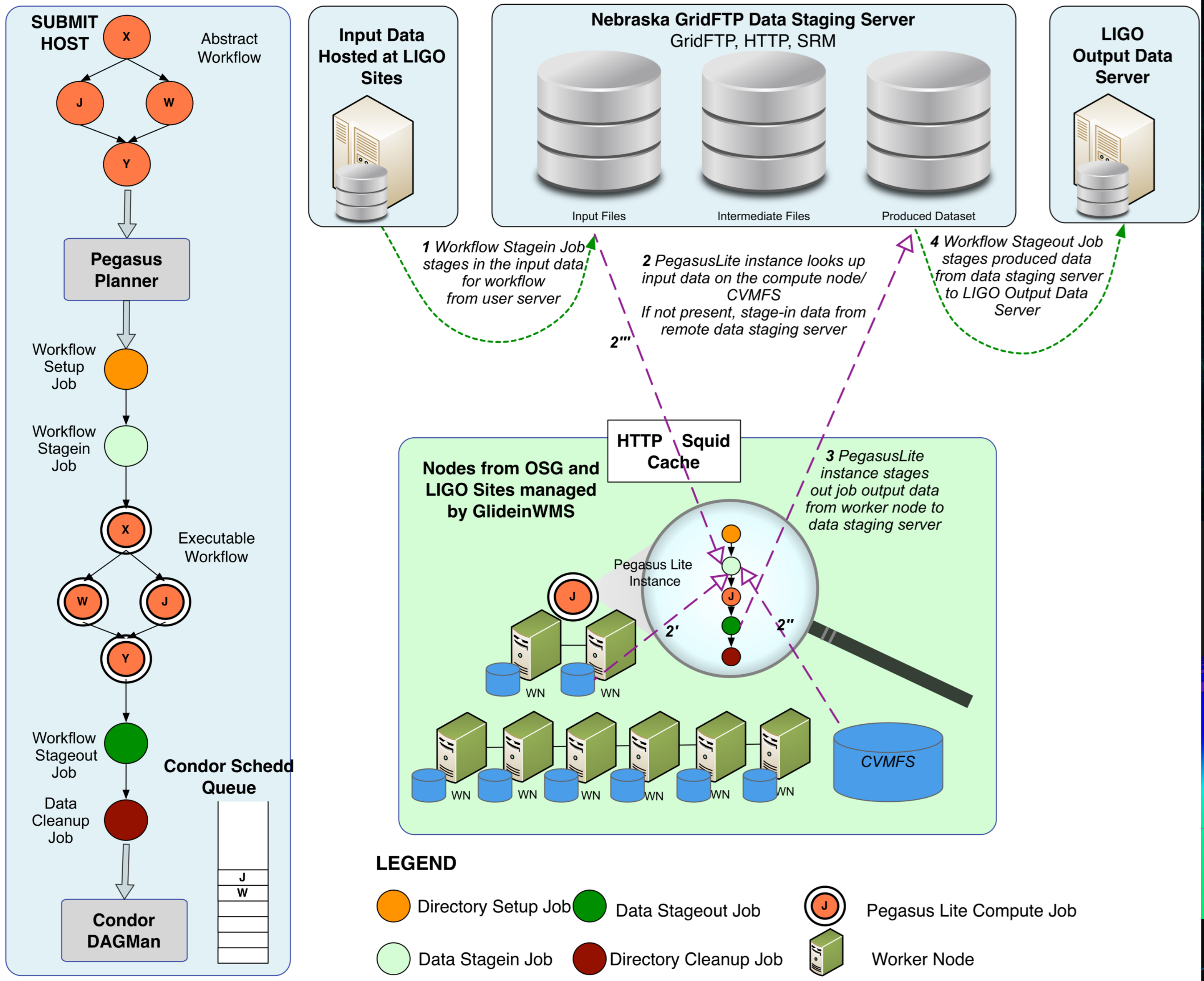
Globus GRAM
Uses CondorG

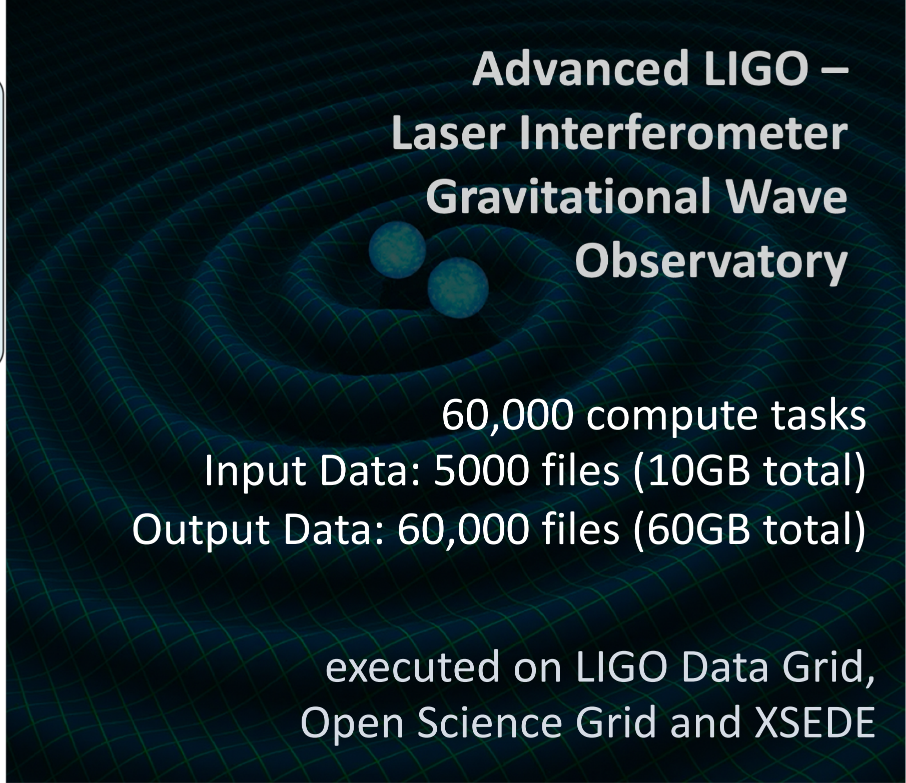
*** Both Glite and BOSCO build on HTCondor BLAHP Support.*
Supported schedulers

PBS SGE SLURM MOAB

Some of the successful stories…

Data Flow for LIGO Pegasus Workflows in OSG

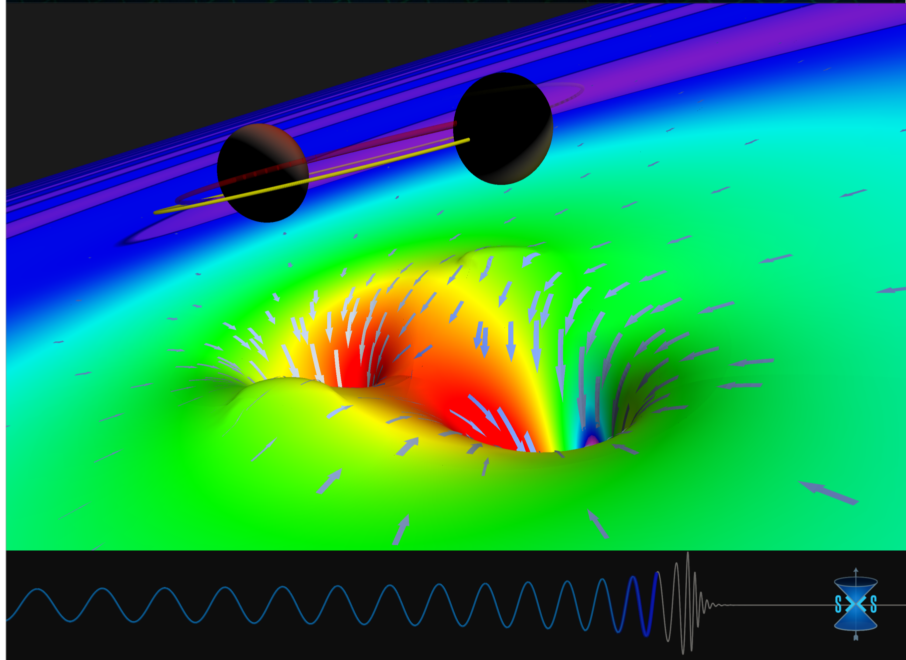

SUBMIT HOST
Abstract Workflow
X
J
W
Y
Pegasus Planner
Workflow Setup Job
Workflow Stagein Job
Executable Workflow
Workflow Stageout Job
Data Cleanup Job
Condor Schedd Queue
Condor DAGMan
Input Data Hosted at LIGO Sites
Nebraska GridFTP Data Staging Server
GridFTP, HTTP, SRM
Input Files
Intermediate Files
Produced Dataset
LIGO Output Data Server
1 Workflow Stagein Job stages in the input data for workflow from user server
2 PegasusLite instance looks up input data on the compute node/ CVMFS
If not present, stage-in data from remote data staging server
4 Workflow Stageout Job stages produced data from data staging server to LIGO Output Data Server
2'''
HTTP Squid Cache
Nodes from OSG and LIGO Sites managed by GlideinWMS
3 PegasusLite instance stages out job output data from worker node to data staging server
Pegasus Lite Instance
2'
2''
WN
CVMFS
LEGEND
Directory Setup Job
Data Stageout Job
Pegasus Lite Compute Job
Data Stagein Job
Directory Cleanup Job
Worker Node



Advanced LIGO – Laser Interferometer Gravitational Wave Observatory

60,000 compute tasks
Input Data: 5000 files (10GB total)
Output Data: 60,000 files (60GB total)

executed on LIGO Data Grid, Open Science Grid and XSEDE



Advanced LIGO PyCBC Workflow

One of the main pipelines to measure the statistical significance of data needed for discovery

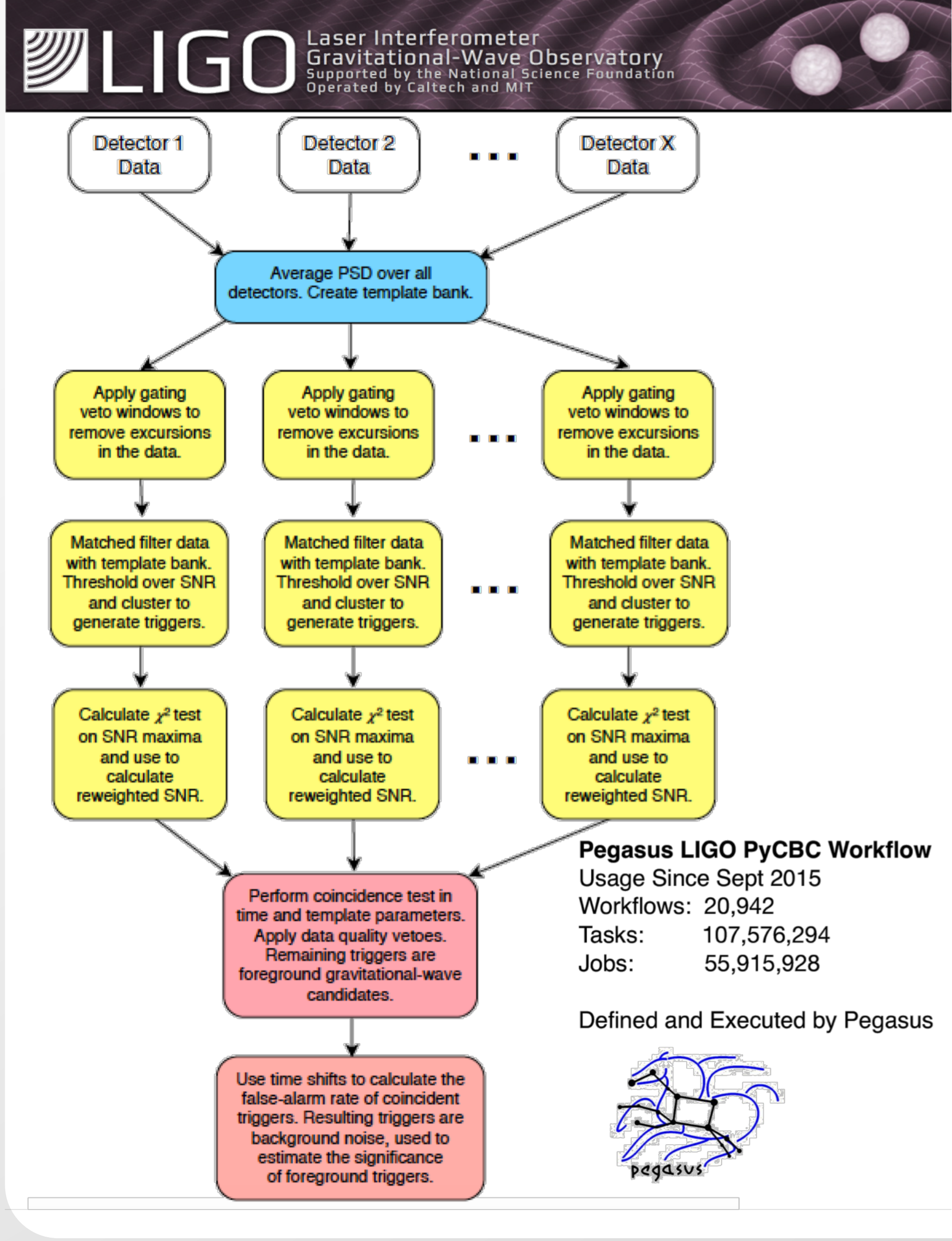
Contains **100’s of thousands of jobs** and accesses on order of **terabytes of data**

Uses data from multiple detectors

For the detection, the pipeline was executed on Syracuse and Albert Einstein Institute Hannover

A single run of the binary black hole + binary neutron star search through the O1 data (about 3 calendar months of data with 50% duty cycle) requires a **workflow** with **194,364 jobs**

Generating the final O1 results with all the review required for the first discovery took about **20 million core hours**


LIGO
Laser Interferometer Gravitational-Wave Observatory
Supported by the National Science Foundation
Operated by Caltech and MIT
Detector 1 Data
Detector 2 Data
Detector X Data
Average PSD over all detectors. Create template bank.
Apply gating veto windows to remove excursions in the data.
Matched filter data with template bank. Threshold over SNR and cluster to generate triggers.
Calculate χ^2 test on SNR maxima and use to calculate reweighted SNR.
Perform coincidence test in time and template parameters. Apply data quality vetoes. Remaining triggers are foreground gravitational-wave candidates.
Use time shifts to calculate the false-alarm rate of coincident triggers. Resulting triggers are background noise, used to estimate the significance of foreground triggers.
pegasus

Pegasus LIGO PyCBC Workflow
Usage Since Sept 2015

Workflows:	20,942
Tasks:	107,576,294
Jobs:	55,915,928

Defined and Executed by Pegasus

Pegasus

PyCBC Papers: An improved pipeline to search for gravitational waves from compact binary coalescence. *Samantha Usman, Duncan Brown et al.*

The PyCBC search for gravitational waves from compact binary coalescence, *Samantha Usman et al* (https://arxiv.org/abs/1508.02357)

PyCBC Detection GW150914: First results from the search for binary black hole coalescence with Advanced LIGO. *B. P. Abbott et al.*

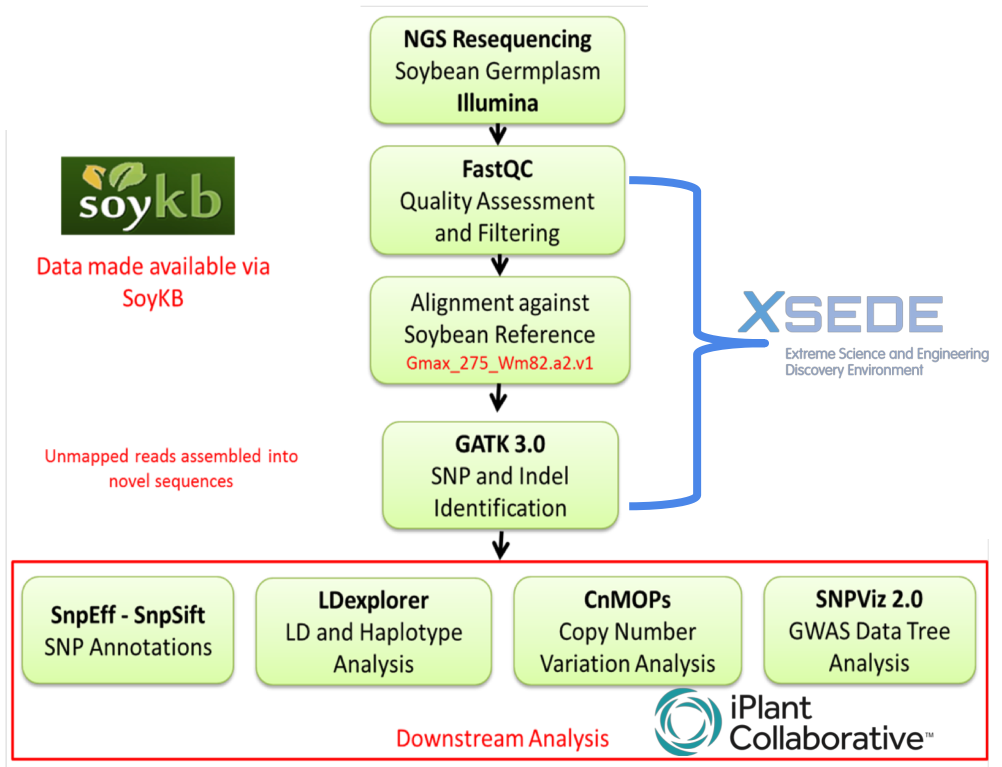
Soybean Workflow

TACC Wrangler as Execution Environment

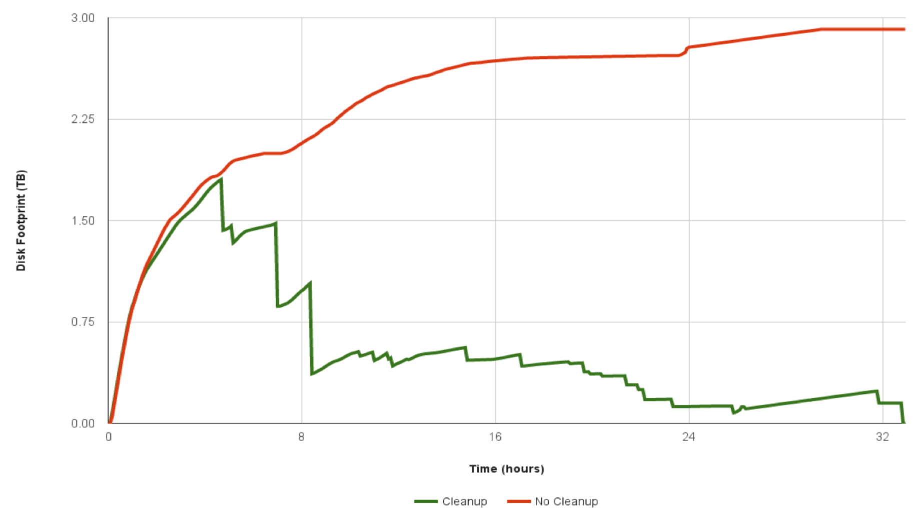
Flash Based Shared Storage

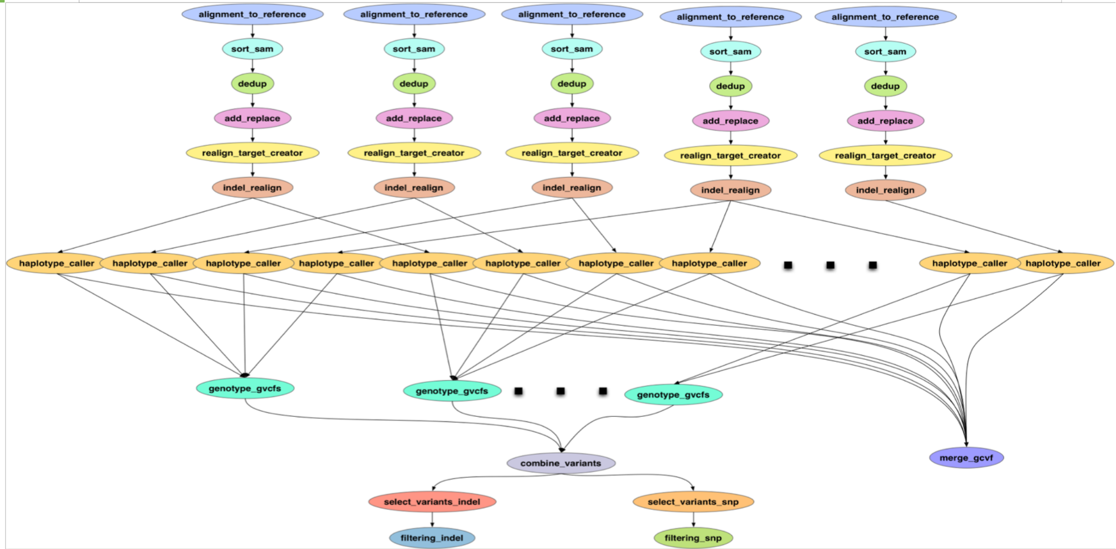
Switched to glideins (pilot jobs) - Brings in remote compute nodes and joins them to the HTCondor pool on the submit host - Workflow runs at a finer granularity

Works well on Wrangler due to more cores and memory per node (48 cores, 128 GB RAM)



NGS Resequencing
Soybean Germplasm
Illumina
FastQC
Quality Assessment and Filtering
Alignment against Soybean Reference
Gmax_275_Wm82.a2.v1
GATK 3.0
SNP and Indel Identification
soykb
Data made available via SoyKB
Unmapped reads assembled into novel sequences
XSEDE
Extreme Science and Engineering Discovery Environment
SnpEff - SnpSift
SNP Annotations
LDexplorer
LD and Haplotype Analysis
CnMOPs
Copy Number Variation Analysis
SNPViz 2.0
GWAS Data Tree Analysis
iPlant Collaborative
Downstream Analysis





Pegasus est. 2001

Automate, recover, and debug scientific computations.

Get Started

Pegasus Website

http://pegasus.isi.edu

Users Mailing List

pegasus-users@isi.edu

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

Support

pegasus-support@isi.edu

Pegasus
Workflow Management System

USC University of Southern California
Information Sciences Institute

https://pegasus.isi.edu/jobs

We're Hiring!