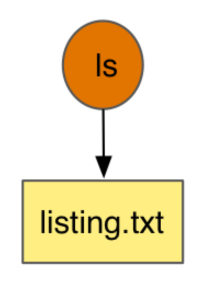
Pegasus + HTCondor DAGMan: Expanding Scientific Workflows Horizons

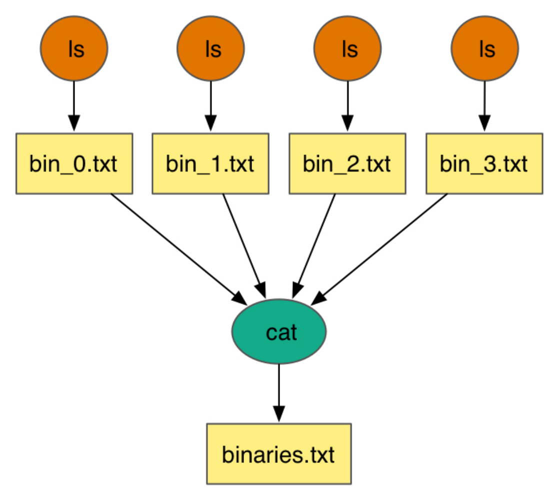
Karan Vahi
vahi@isi.edu

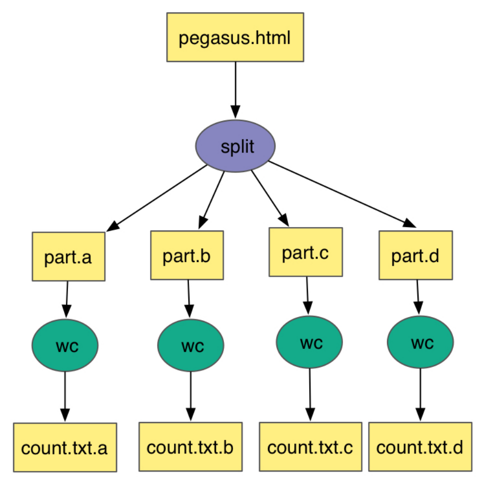
USC Viterbi
School of Engineering
Information Sciences Institute

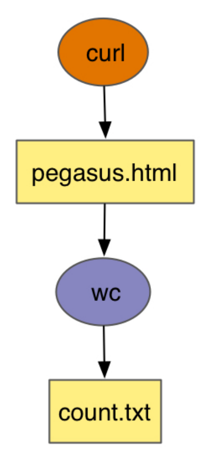
https://pegasus.isi.edu

Compute Pipelines – Building Blocks









HTCondor DAGMan

- DAGMan is a reliable and a scalable workflow executor
  - Sits on top of HTCondor Schedd
  - Can handle very large workflows
- Has useful reliability features in-built
  - Automatic job retries and rescue DAG’s (recover from where you left off in case of failures)
- Throttling for jobs in a workflow

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

Why Pegasus?

Automates complex, multi-stage processing pipelines

Enables parallel, distributed computations

Portable: Describe once; execute multiple times

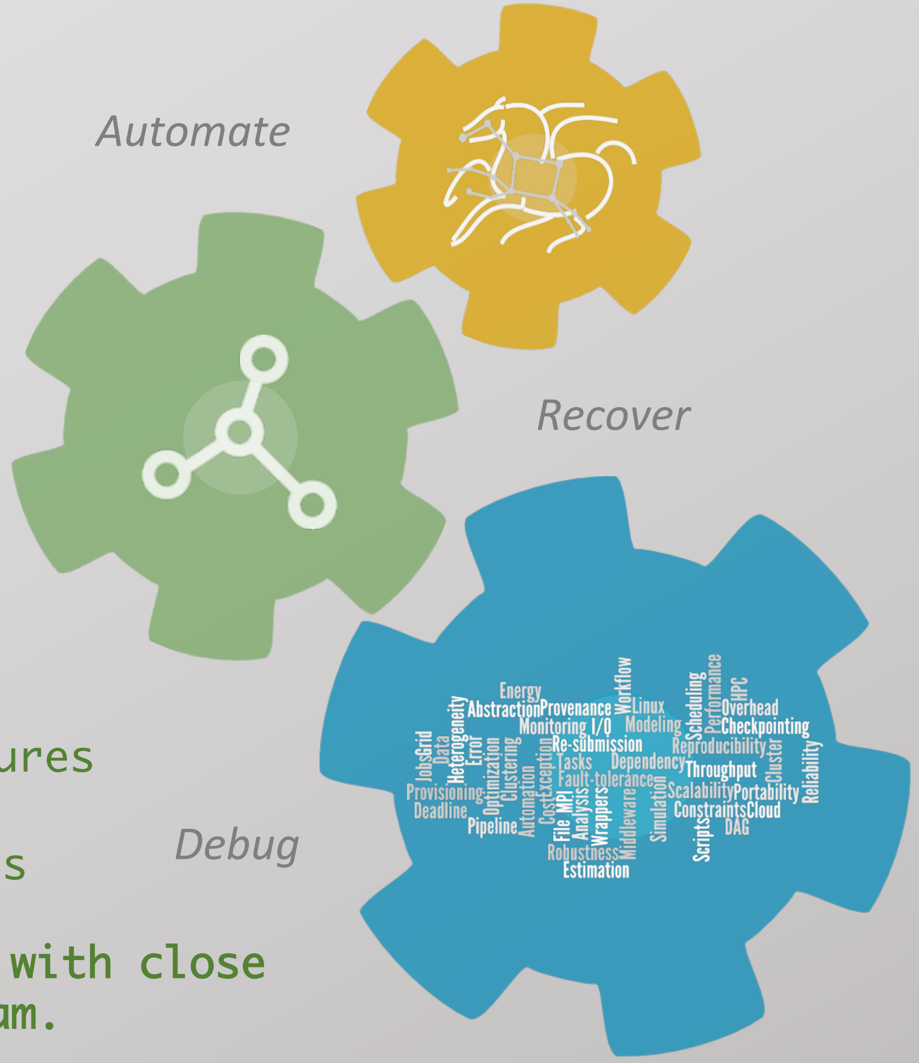
Automatically executes data transfers

Reusable, aids reproducibility

Records how data was produced (provenance)

Provides to tools to handle and debug failures

Keeps track of data and files



HTCondor
High Throughput Computing

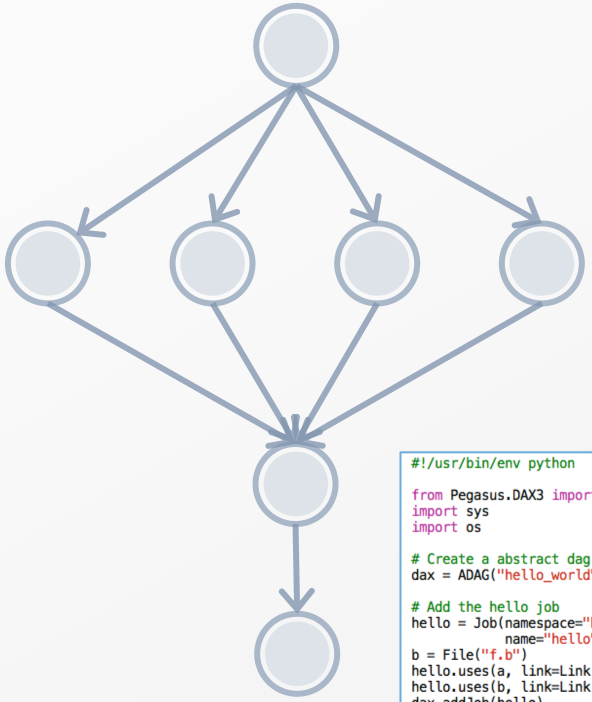
NSF funded project since 2001, with close Collaboration with HTCondor team.

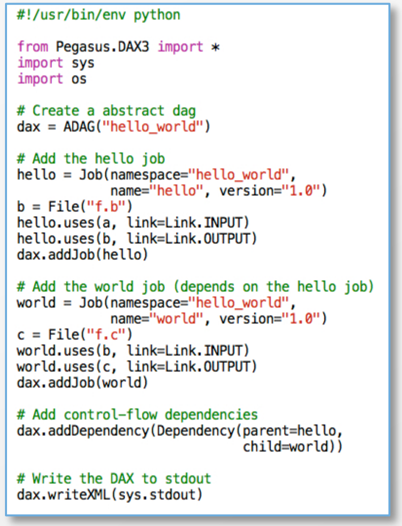
DAX

DAG in XML

Portable Description

Users don’t worry about low level execution details

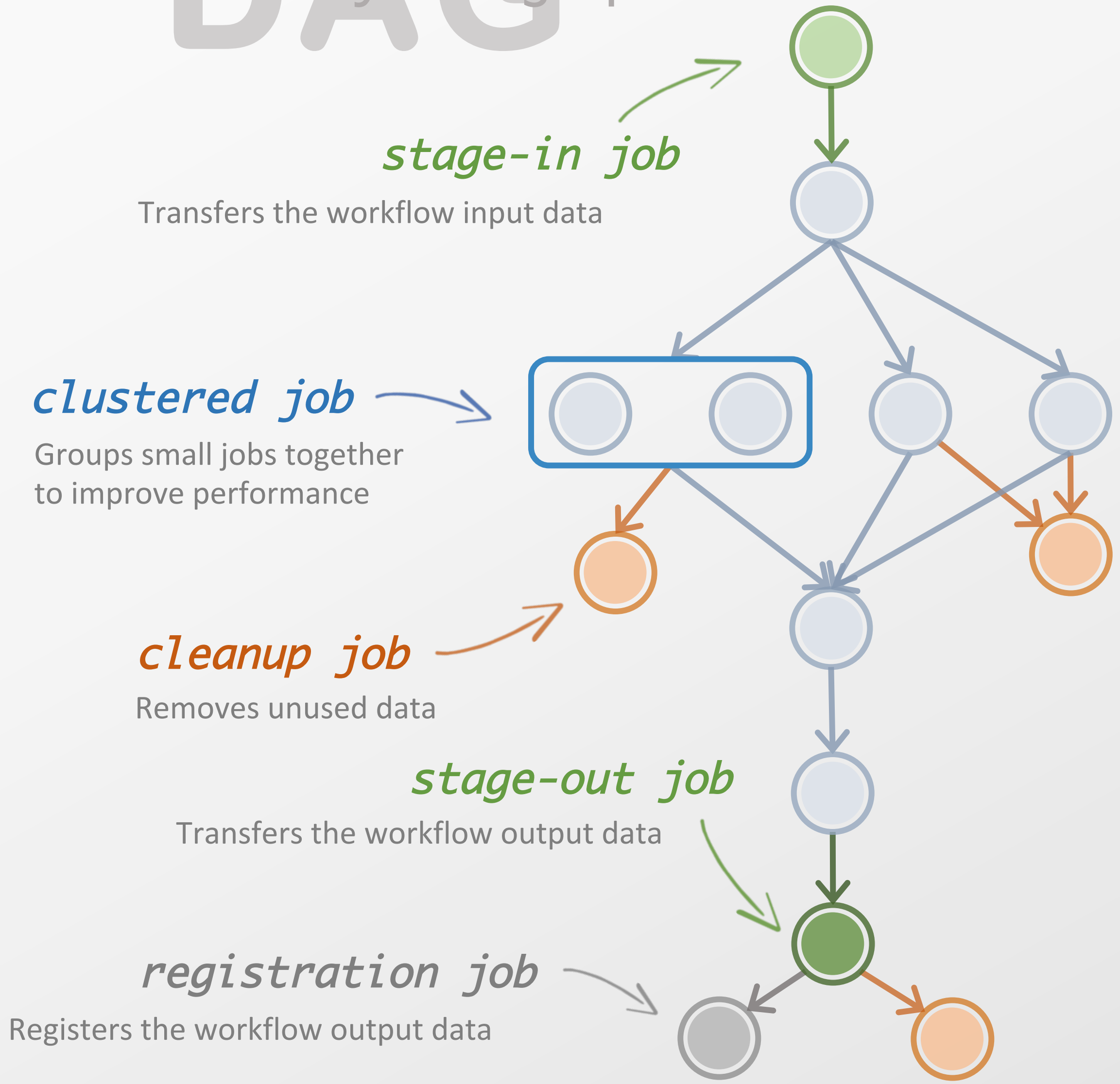




```
#!/usr/bin/env python

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)
```

DAG

directed-acyclic graphs



stage-in job

Transfers the workflow input data

clustered job

Groups small jobs together to improve performance

cleanup job

Removes unused data

stage-out job

Transfers the workflow output data

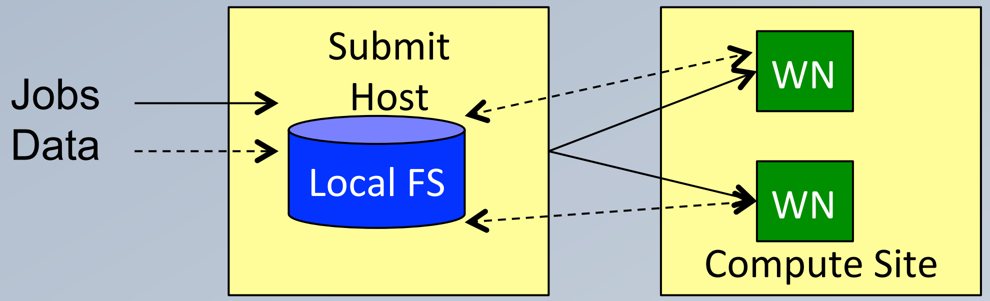
registration job

Registers the workflow output data

Data Staging Configurations

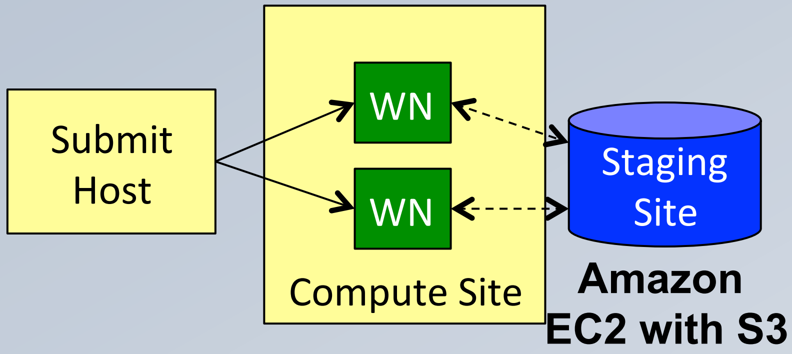
Condor I/O (HTCondor pools, OSG, …)

- 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, …)

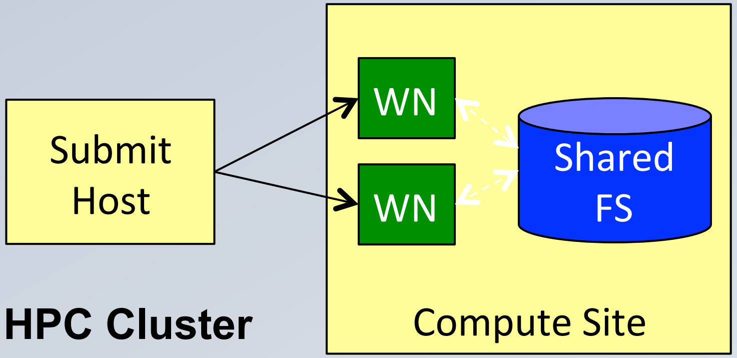
- Worker nodes do not share a file system
- Data is pulled / pushed from a staging site, possibly not co-located with the computation



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

- I/O is directly against the shared file system

Pegasus Guarantee - Wherever and whenever a job runs it’s inputs will be in the directory where it is launched.



pegasus-transfer

- Pegasus' internal data transfer tool with support for a number of different protocols
- Directory creation, file removal
  - If protocol supports, 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

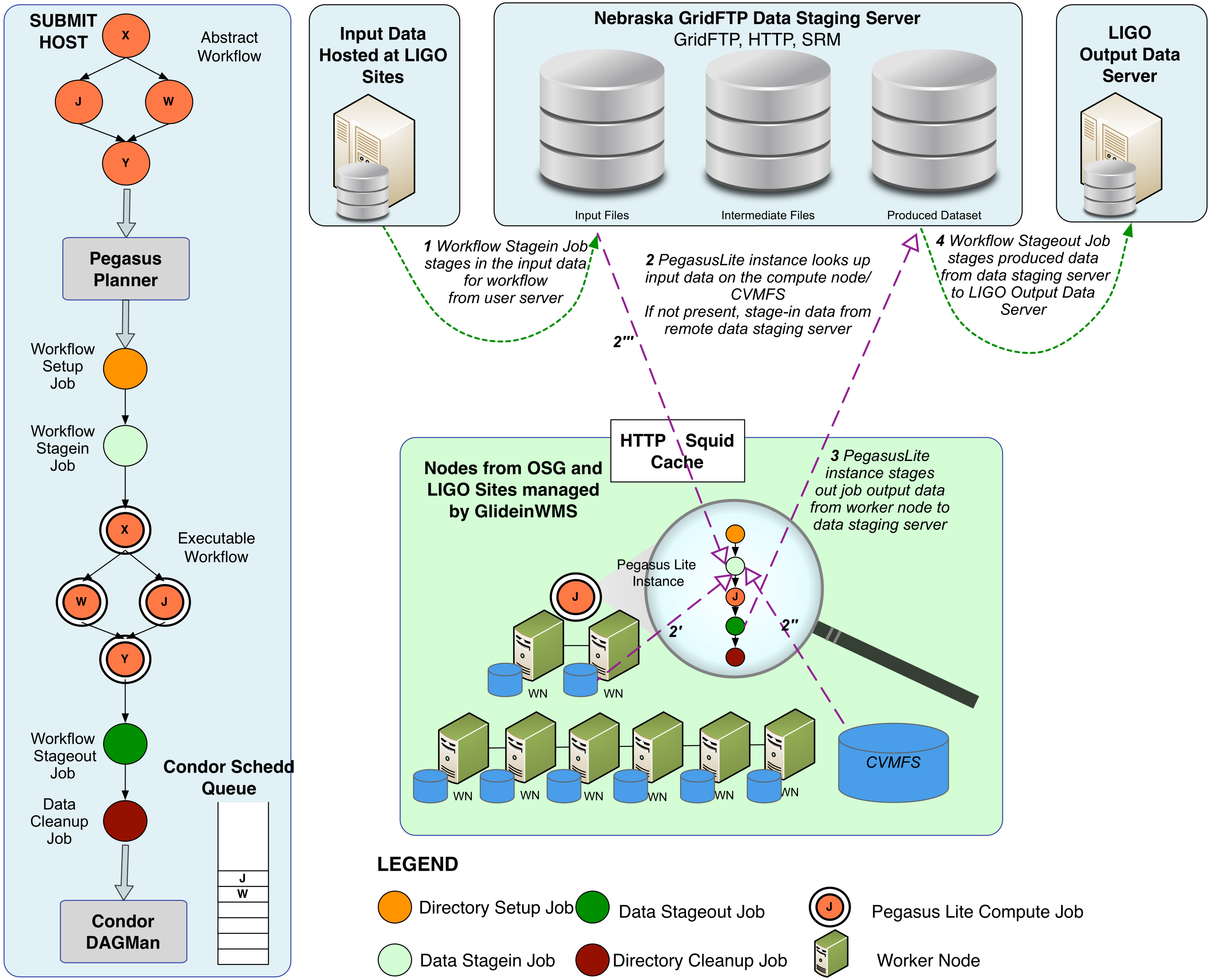
FDT

stashcp

cp

ln -s

Data Flow for LIGO Pegasus Workflows in OSG



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

3 PegasusLite instance stages out job output data from worker node to data staging server

4 Workflow Stageout Job stages produced data from data staging server to LIGO Output Data Server

Advanced LIGO – Laser Interferometer Gravitational Wave Observatory

Benefits to LIGO provided by Pegasus- Expanded Computing Horizons

- No longer limited to a single execution resource
  - Non Pegasus LIGO pipelines can often only run on LIGO clusters
  - Input is replicated out of band , in a rigid directory layout.
  - Rely on the shared filesystem to access data.

- Pegasus made it possible to leverage Non LDG Computing Resources
  - Open Science Grid
    - Dynamic – Best Effort Resource with no shared filesystem available
  - Large NSF Supercomputing Clusters XSEDE
    - No HTCondor
    - Geared for Large MPI jobs, not thousands of single node jobs
    - LIGO tried to setup XSEDE cluster as a LDG site but mismatch in setup.
    - Pegasus enabled LIGO to use XSEDE without changes at LIGO or at XSEDE
  - VIRGO Resources in Europe
    - Clusters with no shared filesystem and different storage management infrastructure than LDG
    - No HTCondor

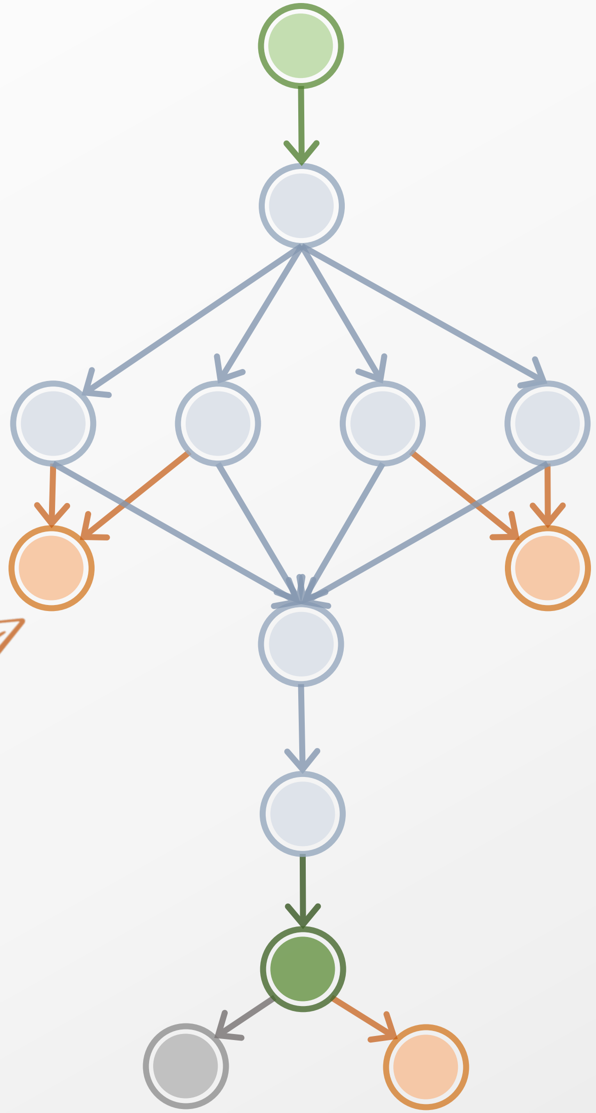
Optimizing storage usage…

abstract workflow

executable workflow

optimizations

storage constraints



cleanup job

Removes unused data

Problem?

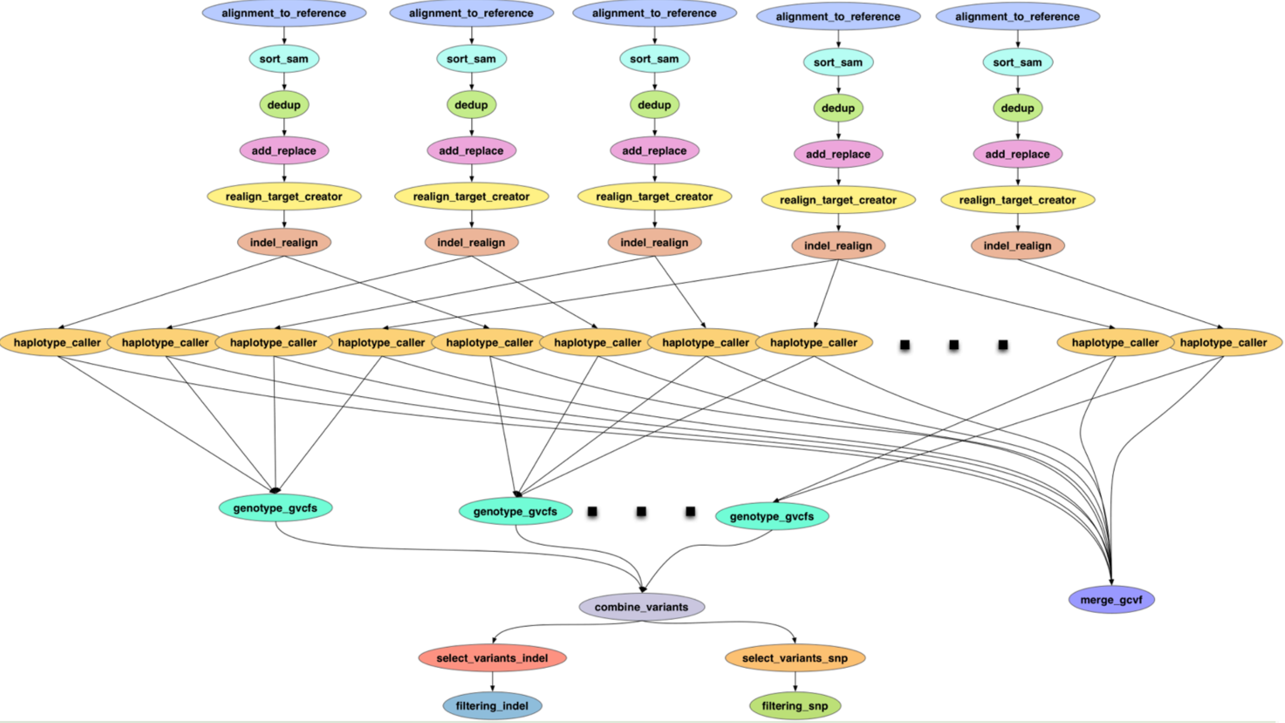
- Users run out of disk space while running workflows

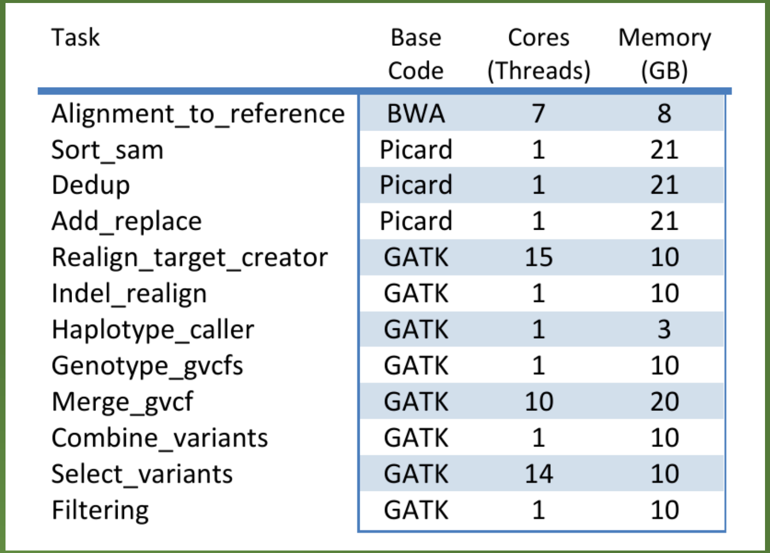
Why does it occur

- Workflows could bring in huge amounts of data
- Data is generated during workflow execution
- Users don’t worry about cleaning up after they are done

- Pegasus Solutions
  - Add leaf cleanup nodes to cleanup after workflow finishes.
  - Interleave cleanup nodes
  - Cluster cleanup nodes per level to improve performance

Pegasus





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

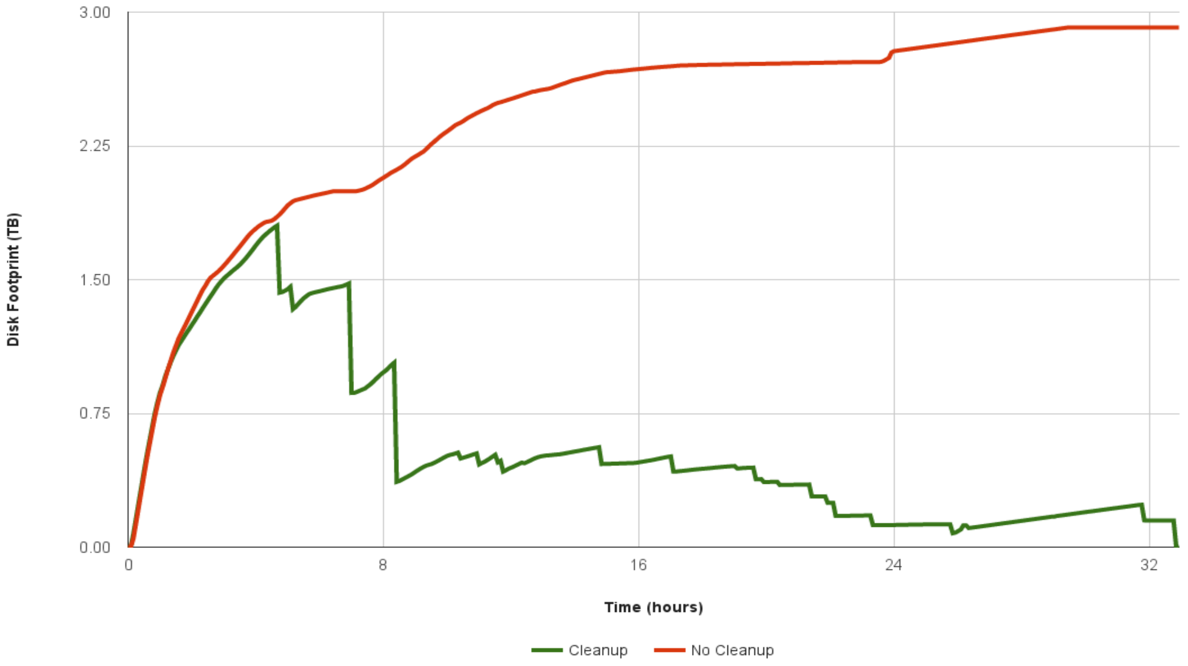
TACC Wrangler as Execution Environment

Flash Based Shared Storage

Switched to glideins (pilot jobs) - Brings in remote compute nodes and joins them to the HTCondor pool on in 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)

Pegasus

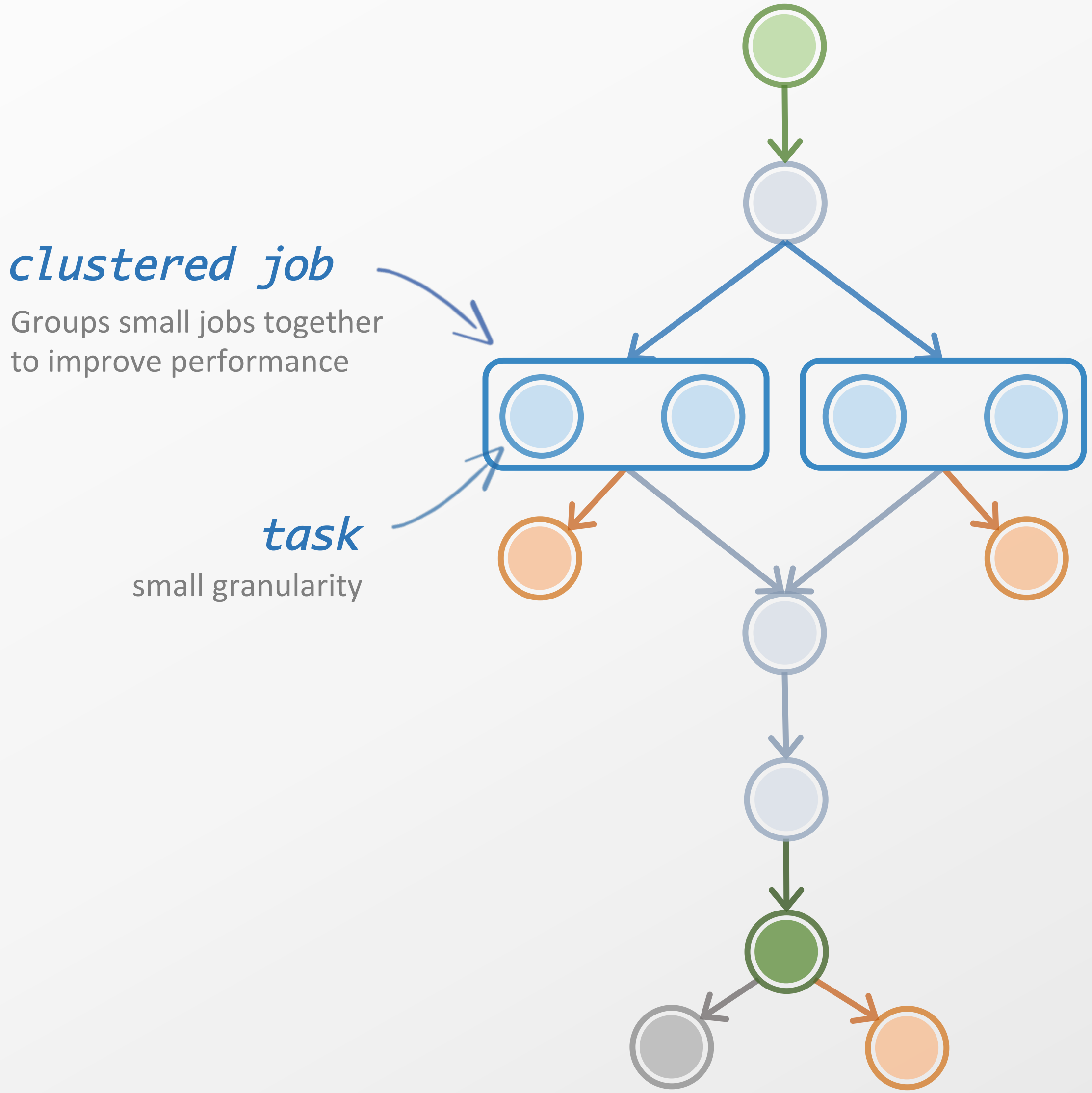


Performance, why not improve it?

workflow restructuring

workflow reduction

hierarchical workflows



Problem?

- Users can have short running tasks that increase workflow walltime

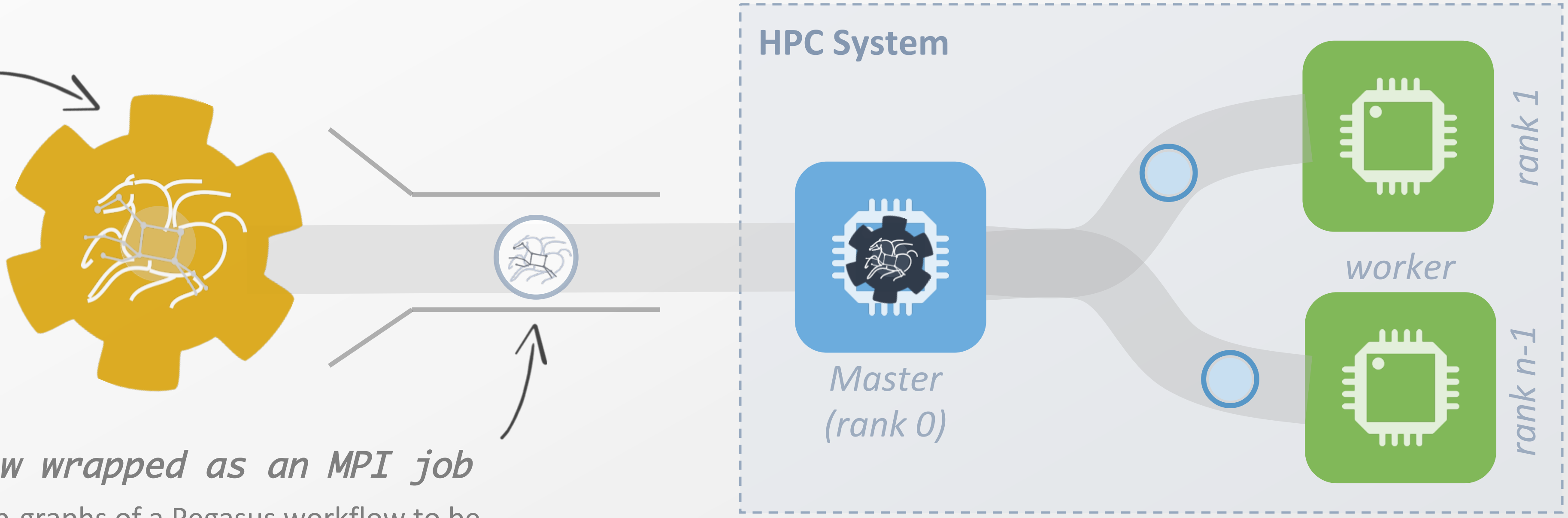
Why does it occur

- Each job has a scheduling delay associated with it

- Pegasus Solutions
  - Cluster tasks together resulting in improved performance and better data placement
  - Ability to run clustered tasks as a single MPI job

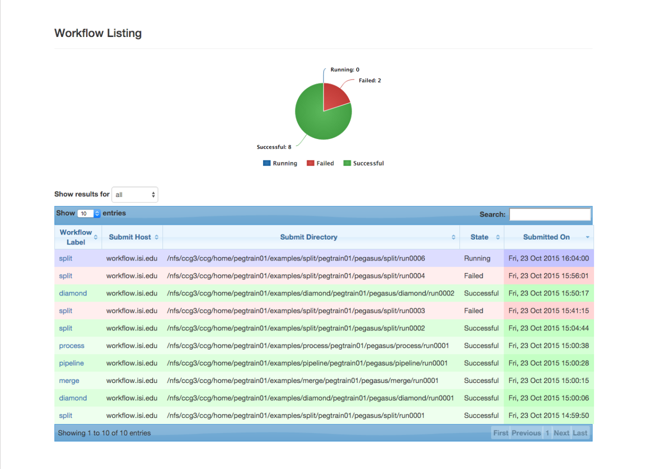
Running fine-grained workflows on HPC systems…

submit host
(e.g., user's laptop)



workflow wrapped as an MPI job

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



Workflow Listing

Show results for all

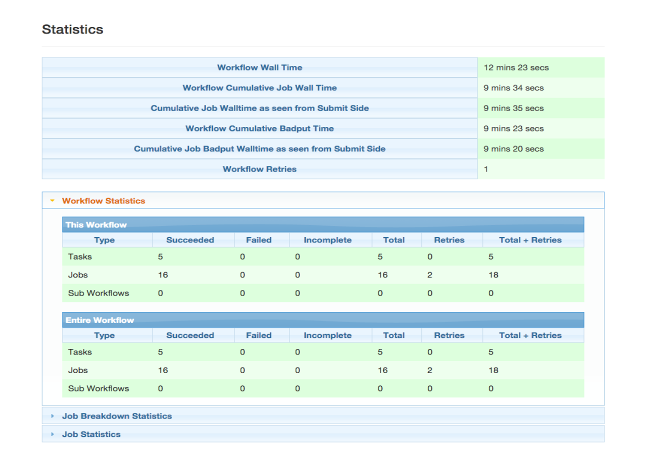
Show 10 entries | Search:

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

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

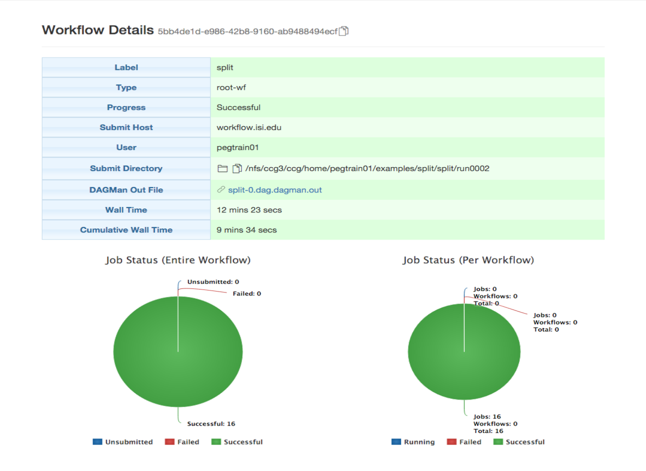
Entire Workflow

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 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

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

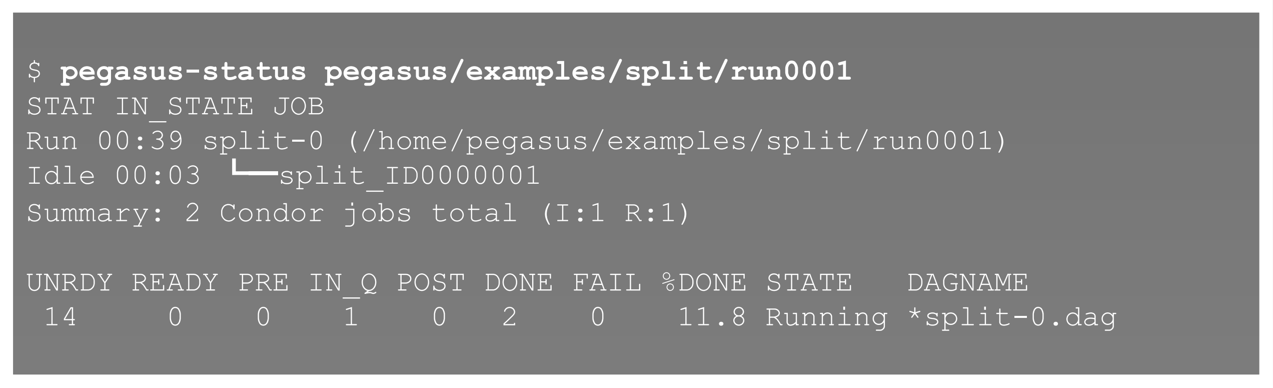
Reporting

Debugging

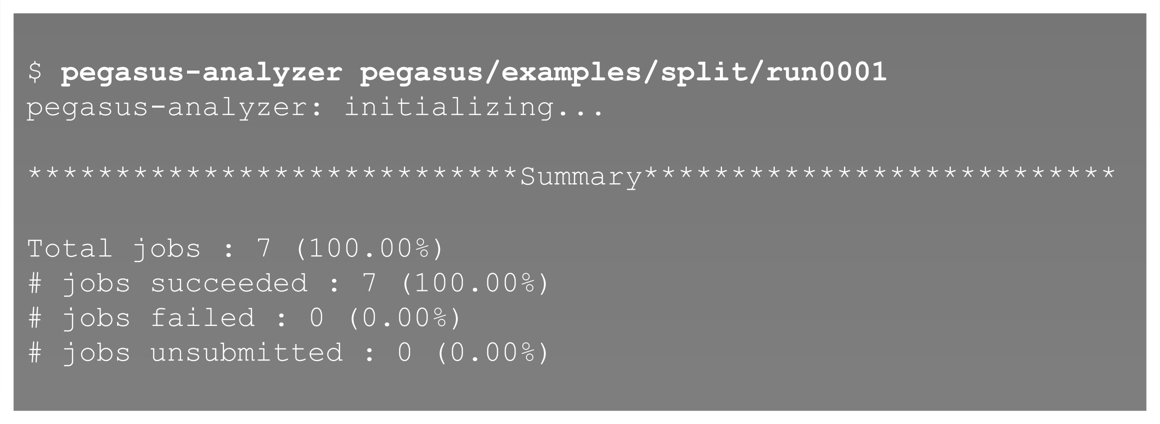
Troubleshooting

RESTful API

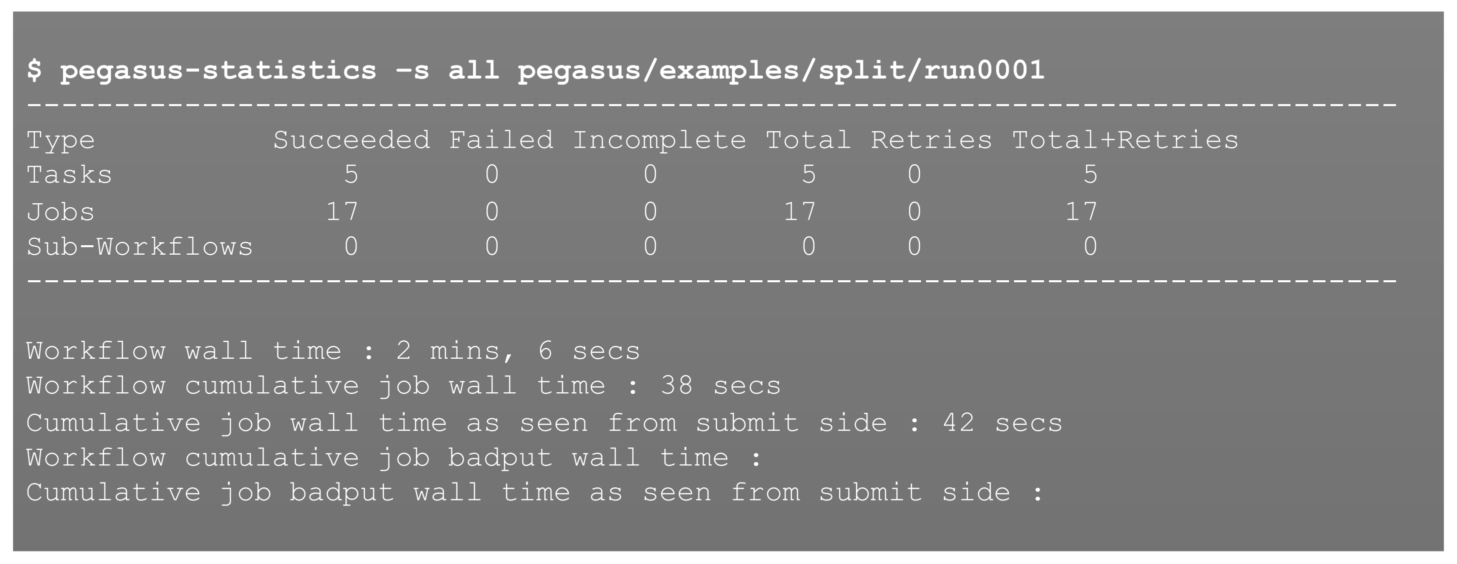
But, if you prefer the command-line…



```
$ 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 :
```

…Pegasus provides a set of concise and powerful tools

Pegasus Container Support

- Support for
  - Docker
  - Singularity – Widely supported on OSG

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

- Users can refer to a container they want to use. However, they let Pegasus stage their executable 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 at tar file and ship them around* (planned for 4.8.X)

Data Management for Containers

- Users can refer to container images as
  - Docker or Singularity Hub URL’s
  - Docker Image exported as a TAR file and available at a server , just like any other input dataset.

- We want to avoid hitting Docker/Singularity Hub repeatedly for large workflows
  - Extend pegasus-transfer to pull image from Docker Hub and then export it as tar file, that can be shipped around in the workflow.

- Ensure pegasus worker package gets installed at runtime inside the user container.

Upcoming Features

To be released with:

Pegasus 4.9

Scientific Workflow Integrity with Pegasus

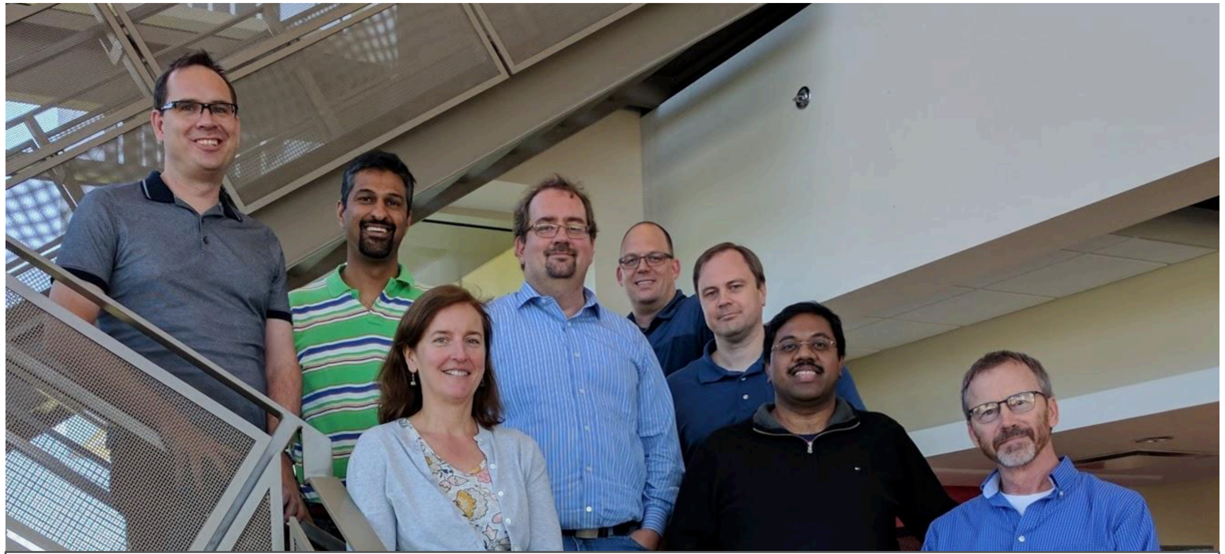
NSF CICI Awards 1642070, 1642053, and 1642090

GOALS

Provide additional assurances that a scientific workflow is not accidentally or maliciously tampered with during its execution

Allow for detection of modification to its data or executables at later dates to facilitate reproducibility.

Integrate cryptographic support for data integrity into the Pegasus Workflow Management System.



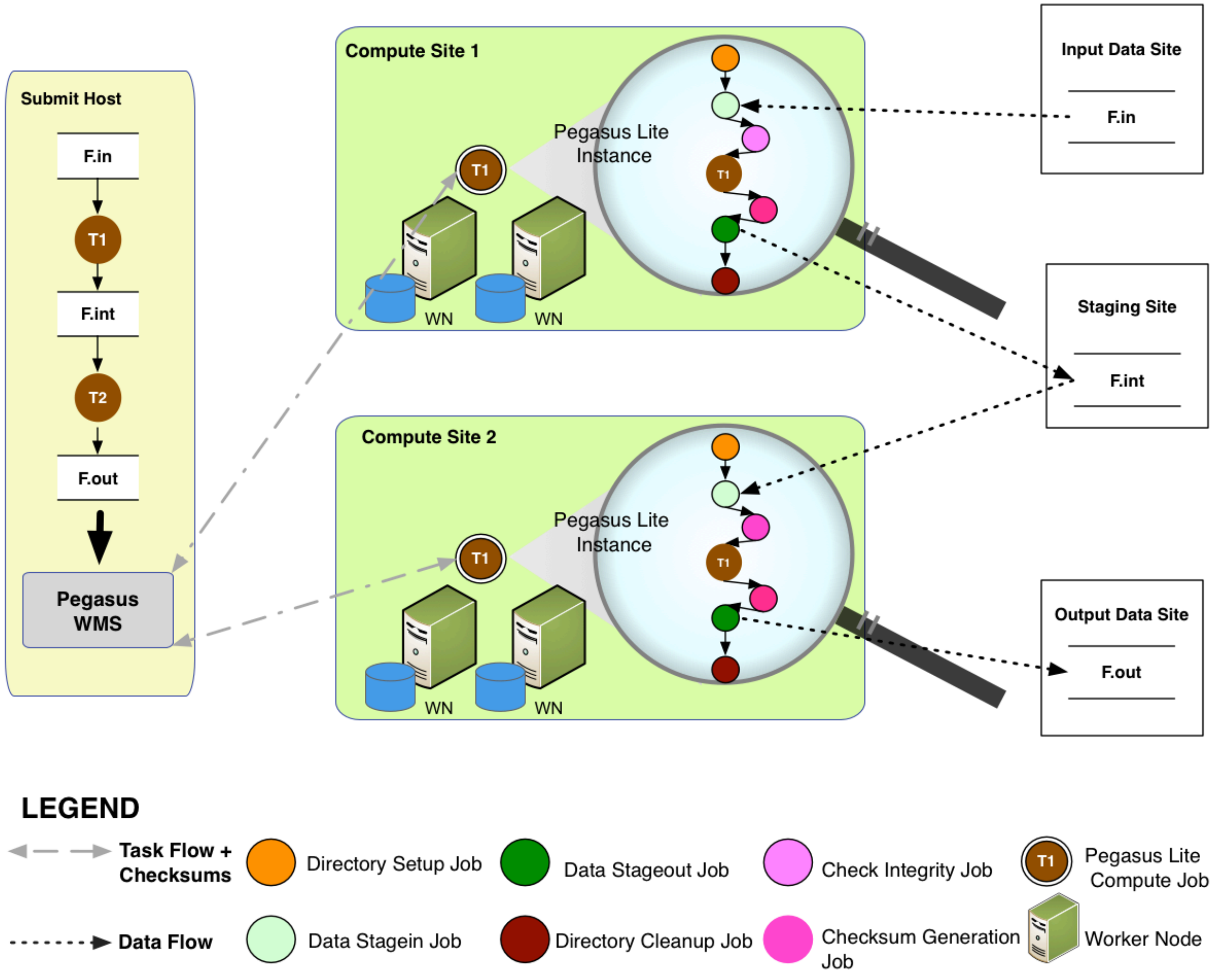
PIs: Von Welch, Ilya Baldin, Ewa Deelman, Steve Myers
Team: Omkar Bhide, Rafael Ferrieira da Silva, Randy Heiland, Anirban Mandal, Rajiv Mayani, Mats Rynge, Karan Vahi

Automatic Integrity Checking

Pegasus will perform integrity checksums on input files before a job starts on the remote node.

- For raw inputs, checksums specified in the input replica catalog along with file locations. *Can compute checksums while transferring if not specified*.
- All intermediate and output files checksums are generated and tracked within the system.
- Support for sha256 checksums

Failure is triggered if checksums fail



Initial Results with Integrity Checking on

- OSG-KINC workflow (50606 jobs) encountered 60 integrity errors in the wild (production OSG). The problematic jobs were automatically retried and the workflow finished successfully.
- The 60 errors took place on 3 different hosts. The first one at UColorado, and group 2 and 3 at UNL hosts.

Error Analysis

- Host 2 had 3 errors, all the same bad checksum for the "kinc" executable with only a few seconds in between the jobs.
- Host 3 had 56 errors, all the same bad checksum for the same data file, and over the timespan of 64 minutes. The site level cache still had a copy of this file and it was the correct file. Thus we suspect that the node level cache got corrupted.

Integration with

- 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

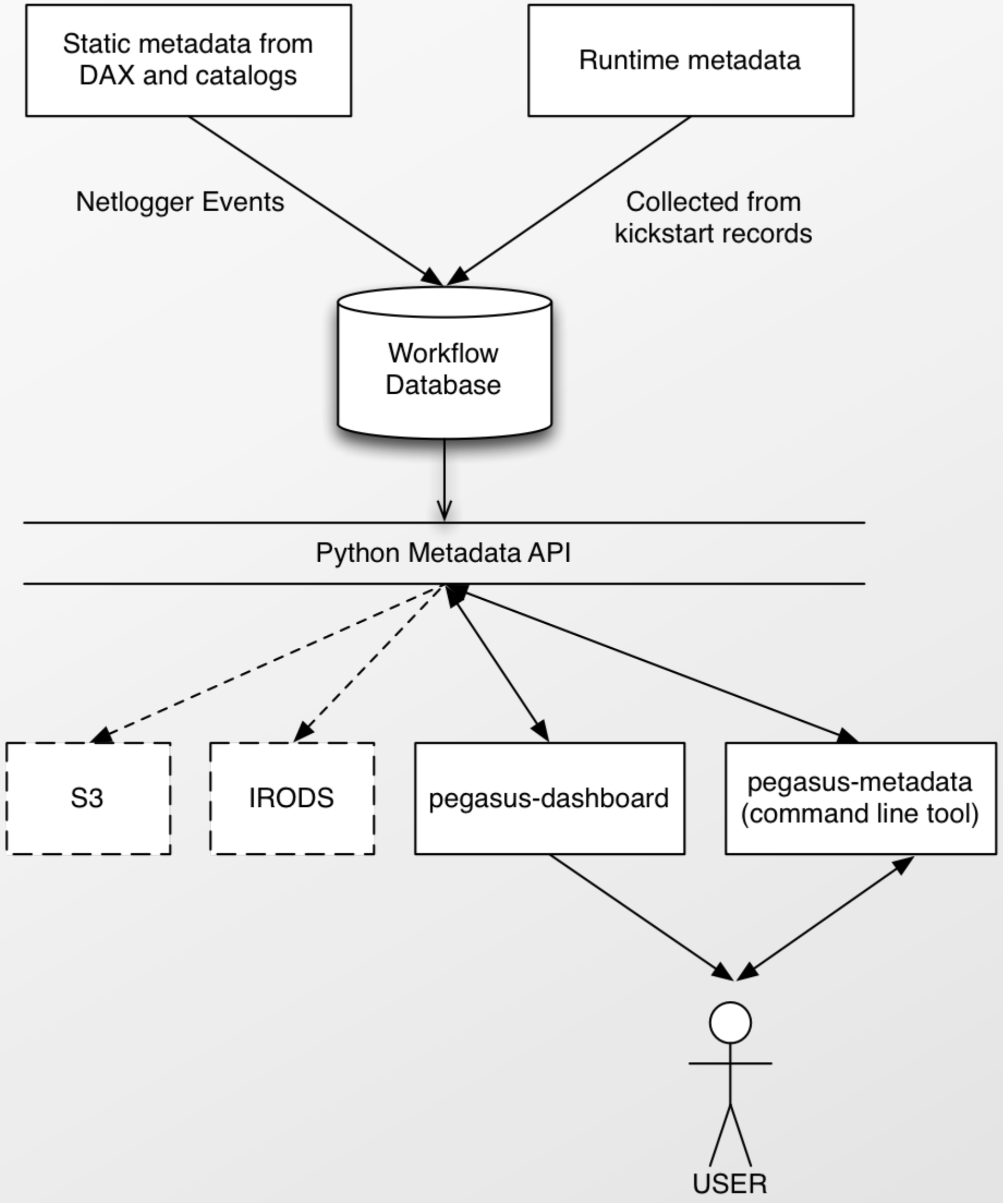
  Automates most of the batch setup programmatically

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

Other Pegasus Capabilities…

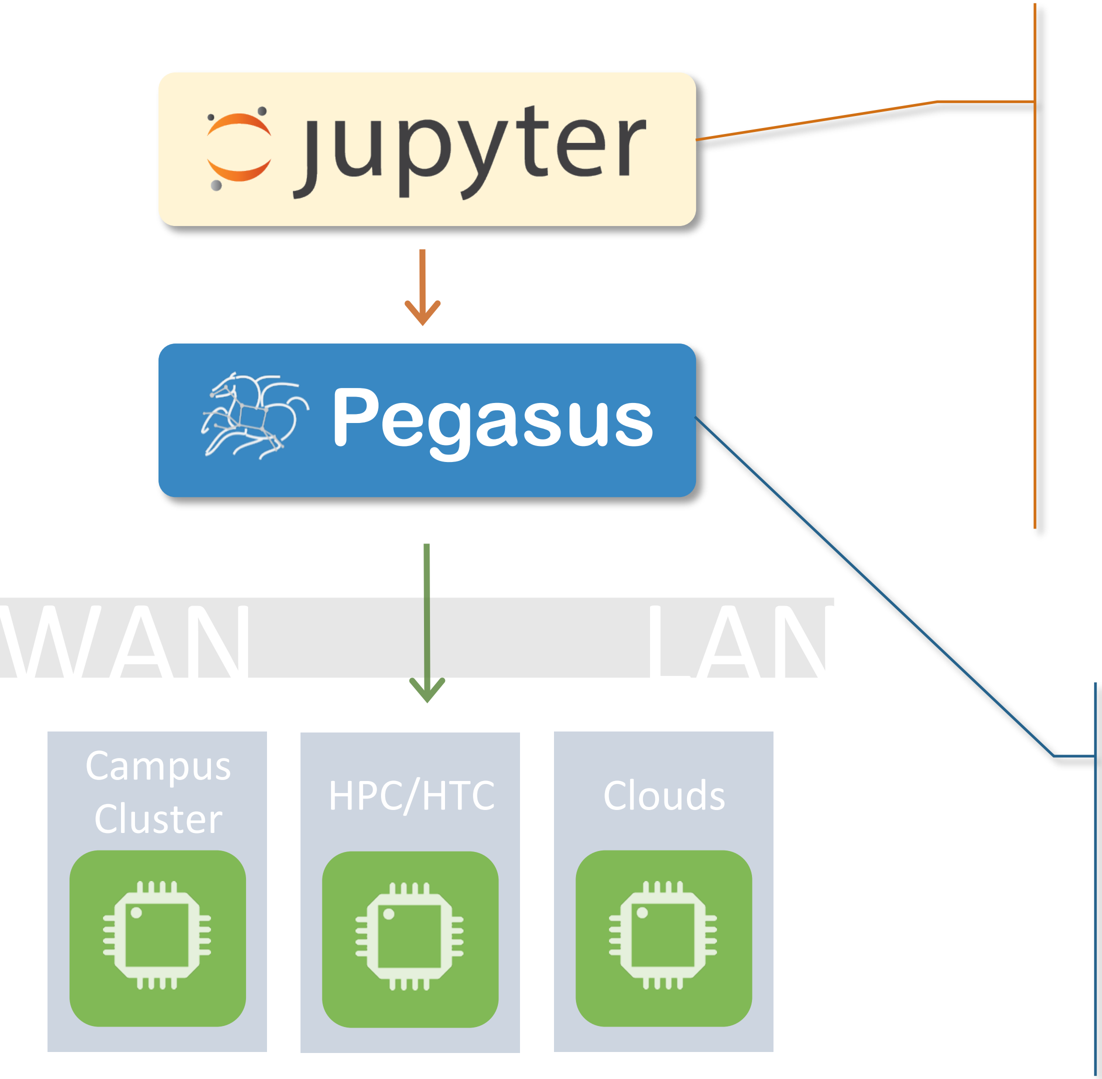
Metadata

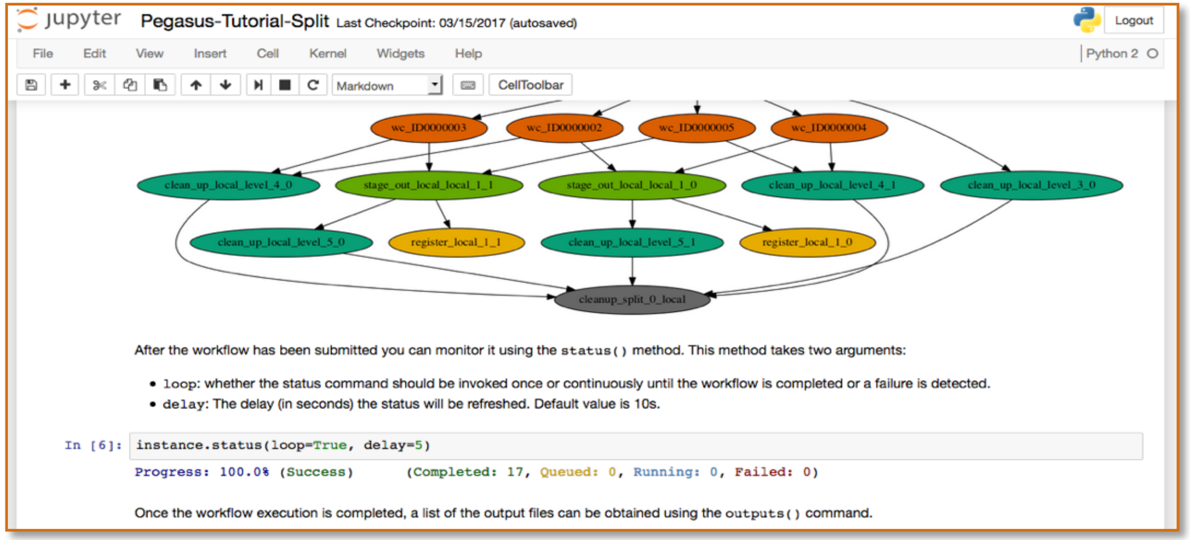
- Can associate arbitrary key-value pairs with workflows, jobs, and files
- Data registration
  - Output files get tagged with metadata on registration in the workflow database.
- Static and runtime metadata
  - Static: application parameters
  - Runtime: performance metrics


Static metadata from DAX and catalogs
Runtime metadata
Netlogger Events
Collected from kickstart records
Workflow Database
Python Metadata API
S3
IRODS
pegasus-dashboard
pegasus-metadata (command line tool)
USER

Introduced in Pegasus 4.6

Running Pegasus workflows with Jupyter



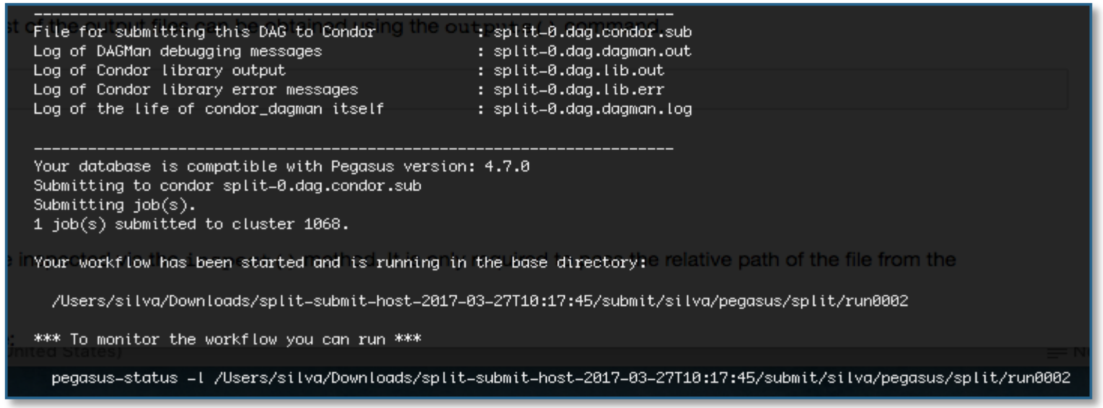


After the workflow has been submitted you can monitor it using the `status()` method. This method takes two arguments:

- `loop`: whether the status command should be invoked once or continuously until the workflow is completed or a failure is detected.
- `delay`: The delay (in seconds) the status will be refreshed. Default value is 10s.

```
In [6]: instance.status(loop=True, delay=5)

        Progress: 100.0% (Success)      (Completed: 17, Queued: 0, Running: 0, Failed: 0)
```

Once the workflow execution is completed, a list of the output files can be obtained using the `outputs()` command.



```
--------------------------------------------------------------------------------
File for submitting this DAG to Condor           : split-0.dag.condor.sub
Log of DAGMan debugging messages                 : split-0.dag.dagman.out
Log of Condor library output                     : split-0.dag.lib.out
Log of Condor library error messages             : split-0.dag.lib.err
Log of the life of condor_dagman itself          : split-0.dag.dagman.log

--------------------------------------------------------------------------------
Your database is compatible with Pegasus version: 4.7.0
Submitting to condor split-0.dag.condor.sub
Submitting job(s).
1 job(s) submitted to cluster 1068.

Your workflow has been started and is running in the base directory:

  /Users/silva/Downloads/split-submit-host-2017-03-27T10:17:45/submit/silva/pegasus/split/run0002

*** To monitor the workflow you can run ***

  pegasus-status -l /Users/silva/Downloads/split-submit-host-2017-03-27T10:17:45/submit/silva/pegasus/split/run0002
```

Pegasus-Jupyter Python API

```
from Pegasus.jupyter.instance import *
```

importing the API

```
instance = Instance(dax)
```

creating an instance of the DAX

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

# 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)
```

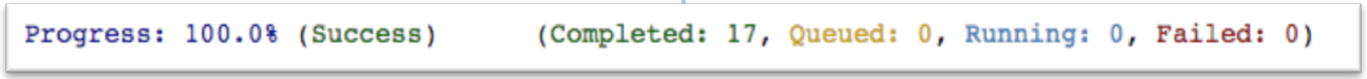
using the Pegasus DAX3 API to write a workflow

```
instance.run(site='condorpool')
```

running a workflow

```
instance.status(loop=True, delay=5)
```

monitoring a workflow execution



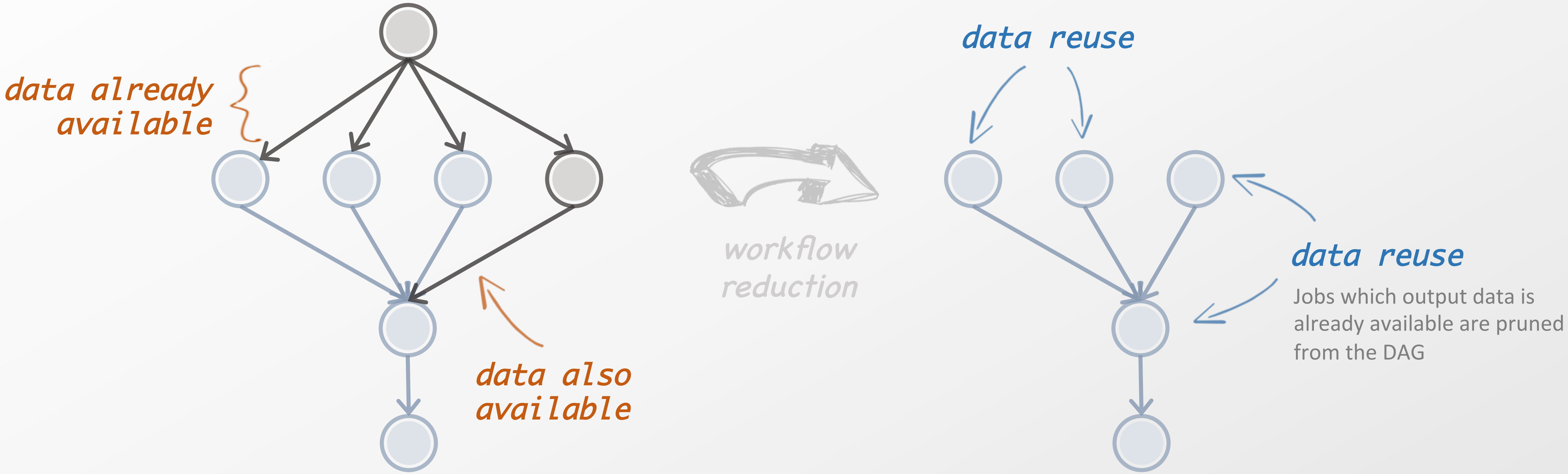
```
Progress: 100.0% (Success)      (Completed: 17, Queued: 0, Running: 0, Failed: 0)
```

What about **data reuse**?

workflow restructuring

workflow reduction

hierarchical workflows



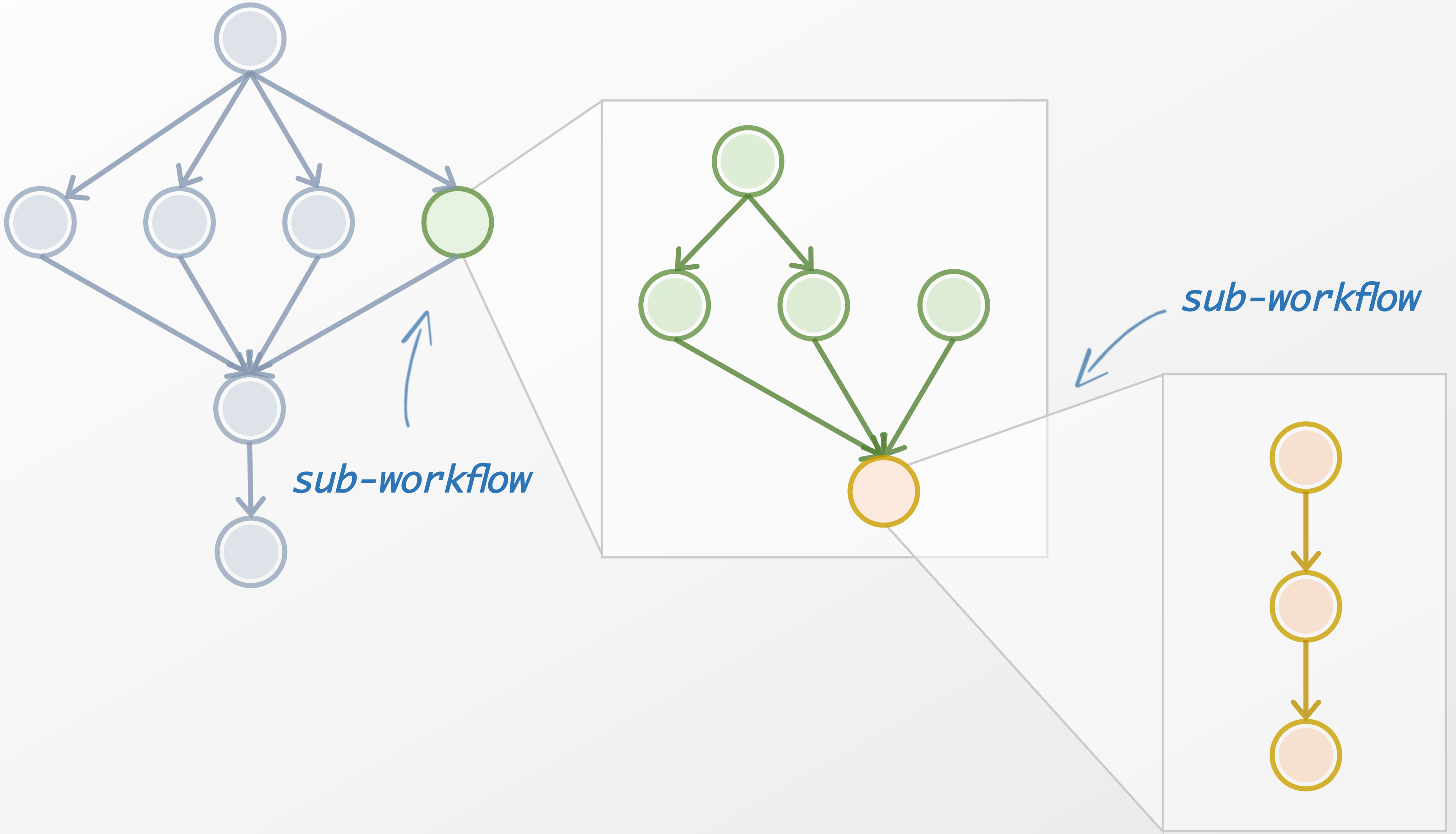
Jobs which output data is already available are pruned from the DAG

Pegasus also handles **large-scale workflows**

workflow restructuring

workflow reduction

hierarchical workflows



recursion ends when DAX with only compute jobs is encountered

Pegasus est. 2001

Automate, recover, and debug scientific computations.

Get Started

Pegasus Website

http://pegasus.isi.edu

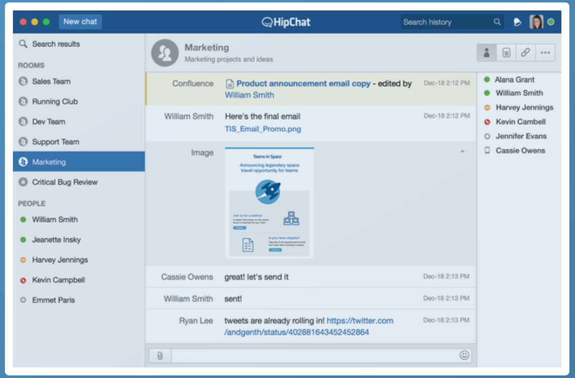
Users Mailing List

pegasus-users@isi.edu

Support

pegasus-support@isi.edu

HipChat



Confluence Product announcement email copy - edited by William Smith Dec-18 2:12 PM

William Smith Here's the final email TIS_Email_Promo.png Dec-18 2:12 PM

Image

Cassie Owens great! let's send it Dec-18 2:13 PM

William Smith sent! Dec-18 2:13 PM

Ryan Lee tweets are already rolling in! https://twitter.com/andgenth/status/402881643452452864 Dec-18 2:13 PM

Pegasus est. 2001

Automate, recover, and debug scientific computations.

Thank You

Questions?

Mats Rynge
rynge@isi.edu

USC Viterbi
School of Engineering
Information Sciences Institute

Meet our team

Ewa Deelman

Karan Vahi

Mats Rynge

Rajiv Mayani

Rafael Ferreira da Silva

NSF U.S. DEPARTMENT OF ENERGY NIH