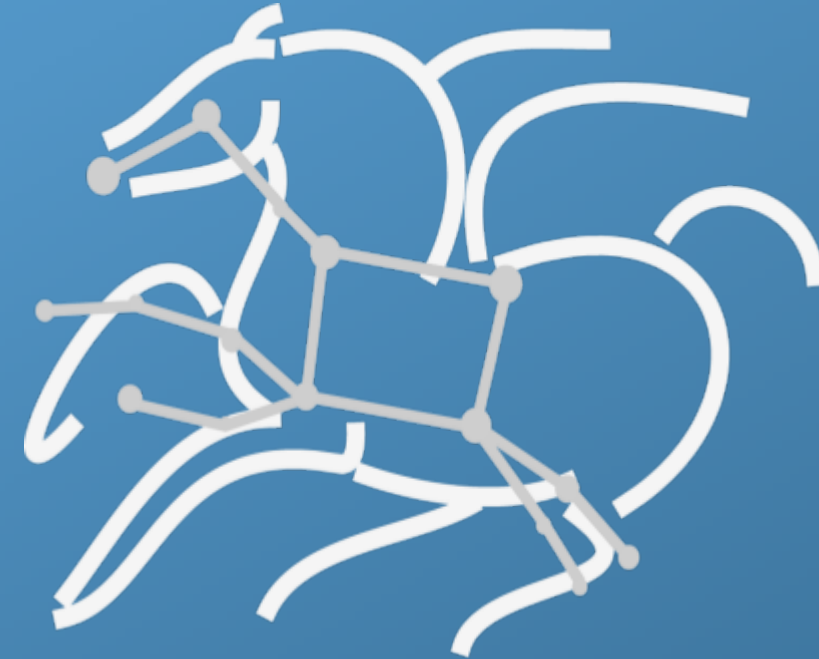




U.S. DEPARTMENT OF
ENERGY



Pegasus WMS : An Introduction and upcoming features



Karan Vahi

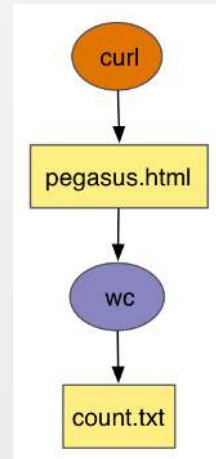
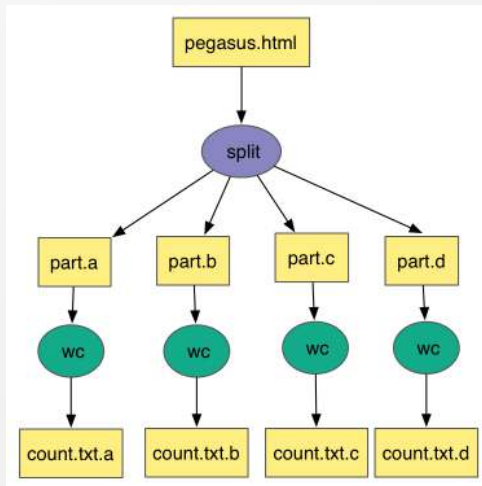
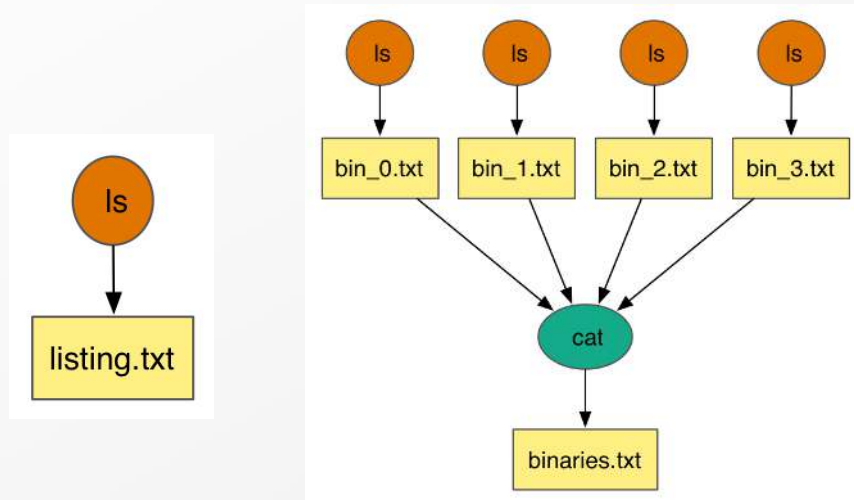
vahi@isi.edu

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

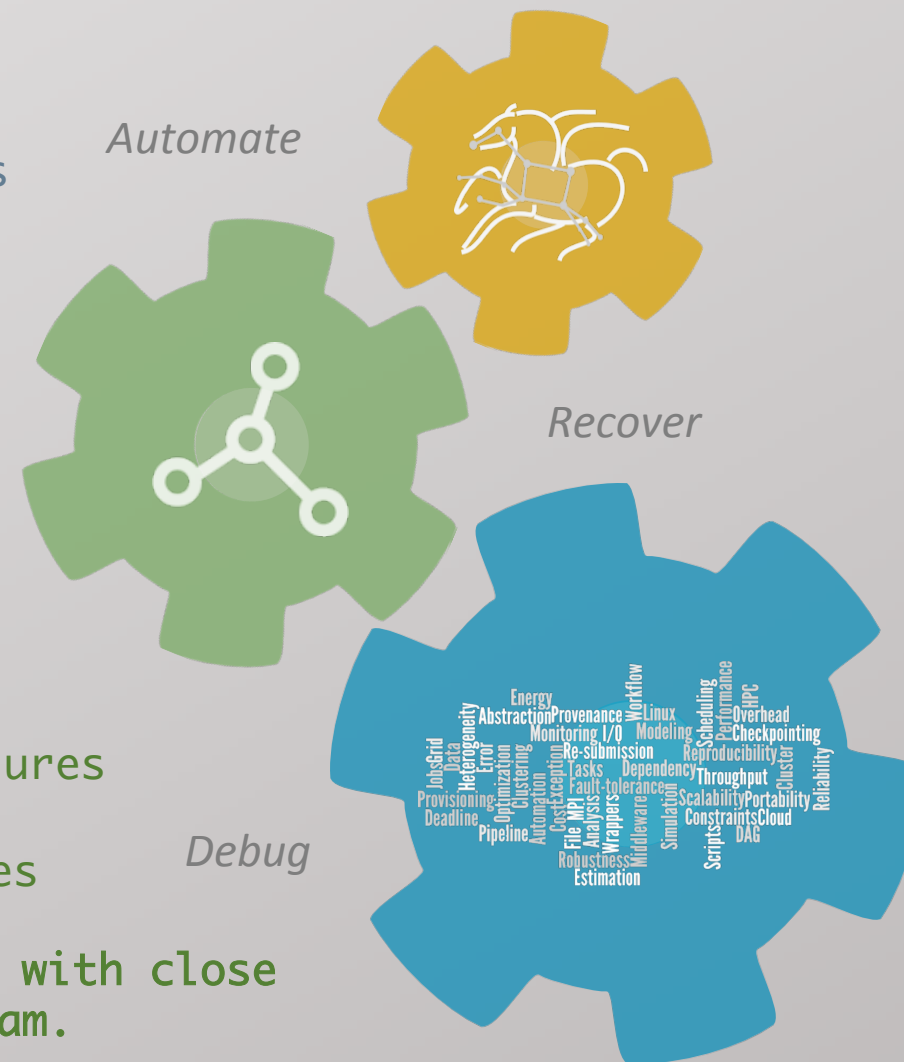
Reusable, aids reproducibility

Records how data was produced (provenance)

Provides to tools to handle and debug failures

Keeps track of data and files

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

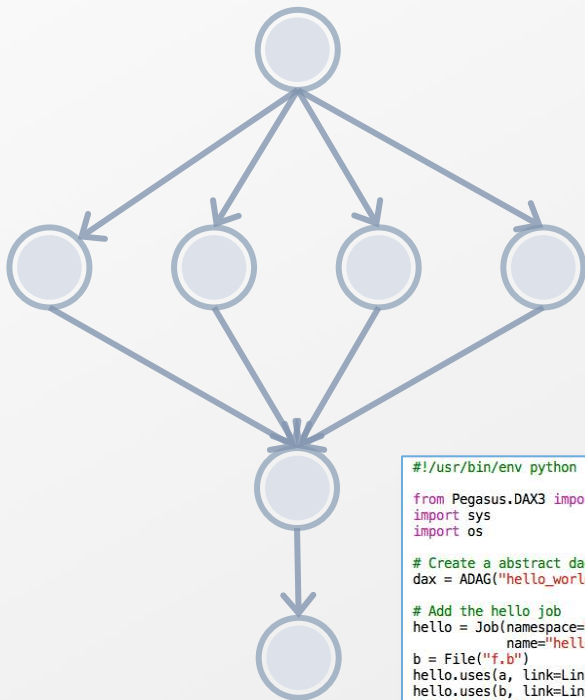


DAX

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 an 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))

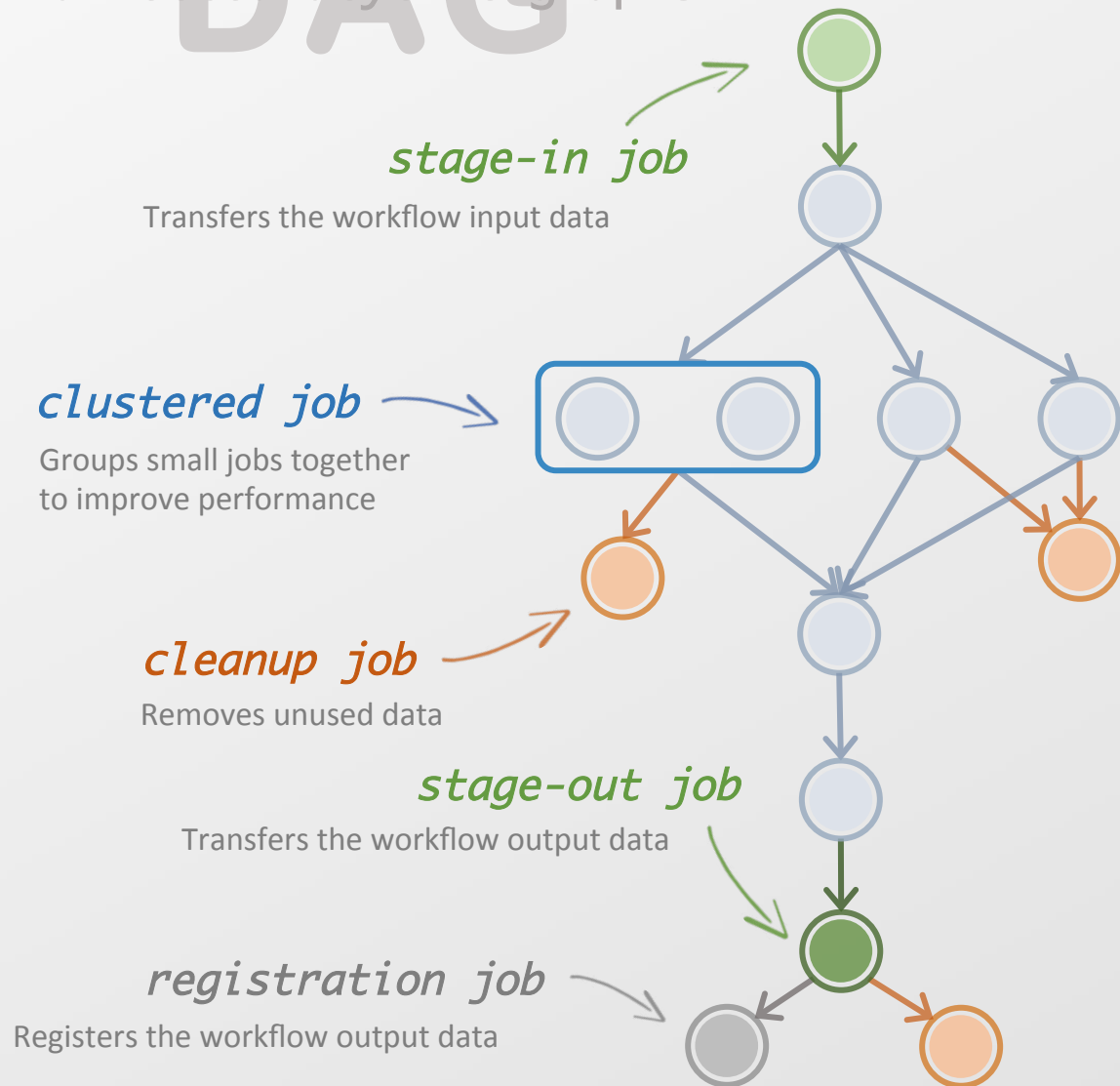
# Write the DAX to stdout
dax.writeXML(sys.stdout)
```



Pegasus

DAG

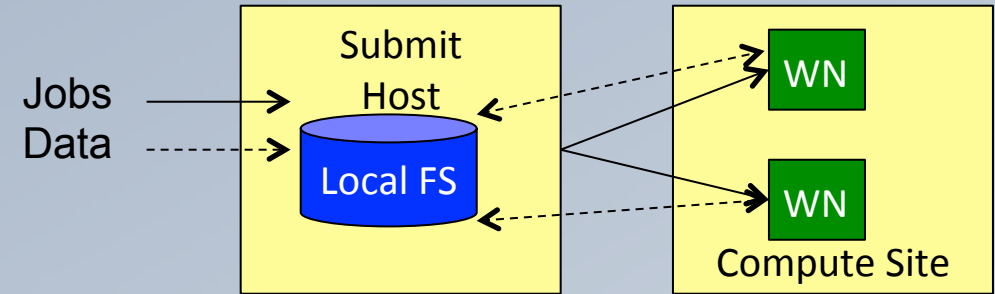
directed-acyclic graphs



Data Staging Configurations

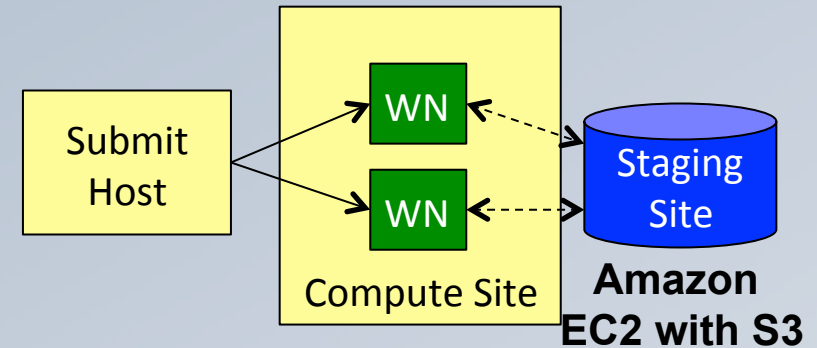
Condor I/O (HTCondor pools, OSG, ...)

- 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, ...)

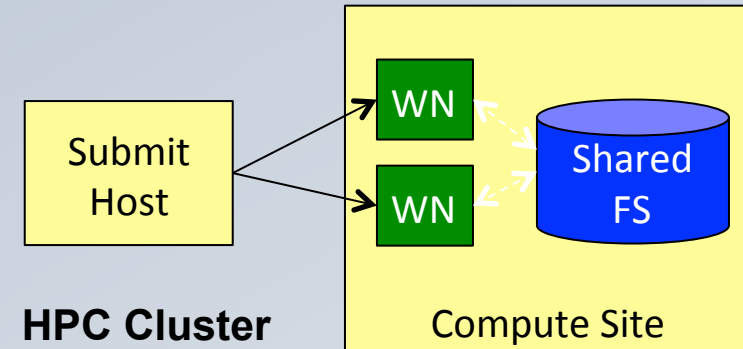
- Worker nodes do not share a file system
- Data is pulled / pushed from a staging site, possibly not co-located with the computation



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

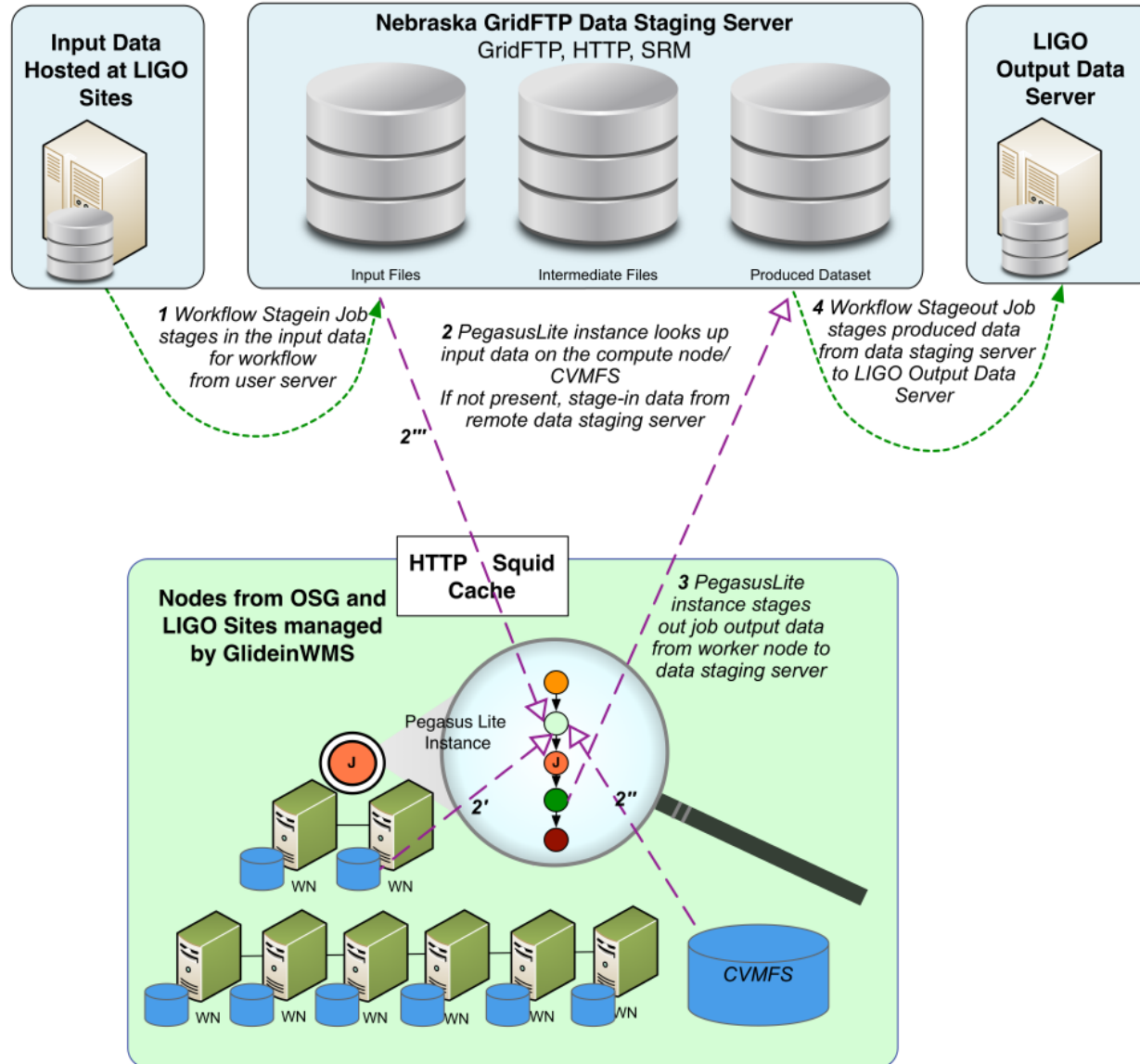
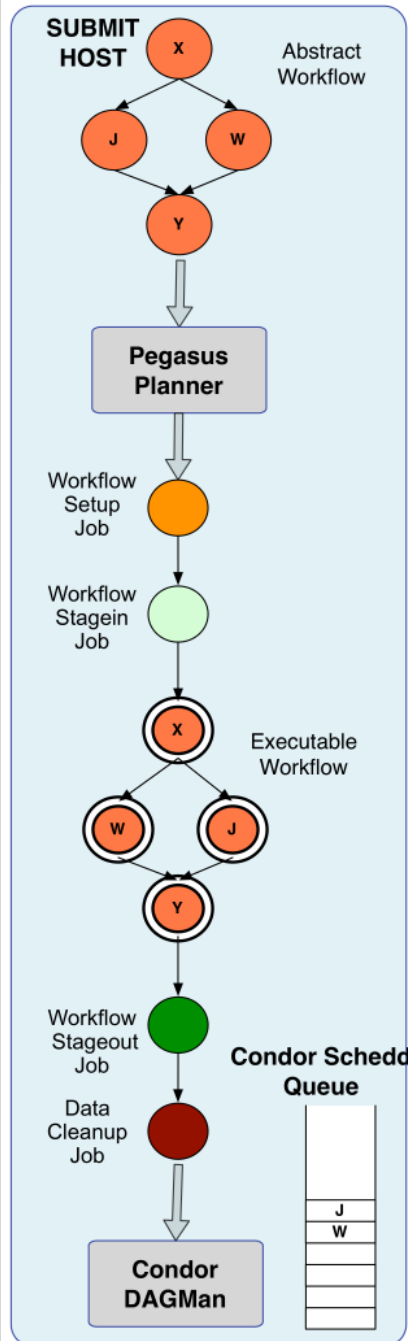
FDT

stashcp

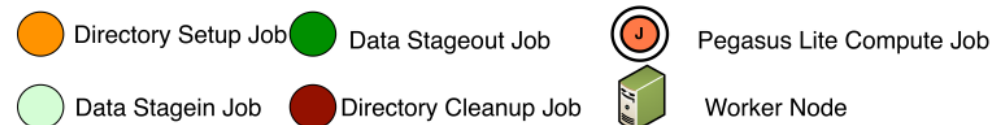
cp

ln -s

Data Flow for LIGO Pegasus Workflows in OSG



LEGEND



**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

Problem?

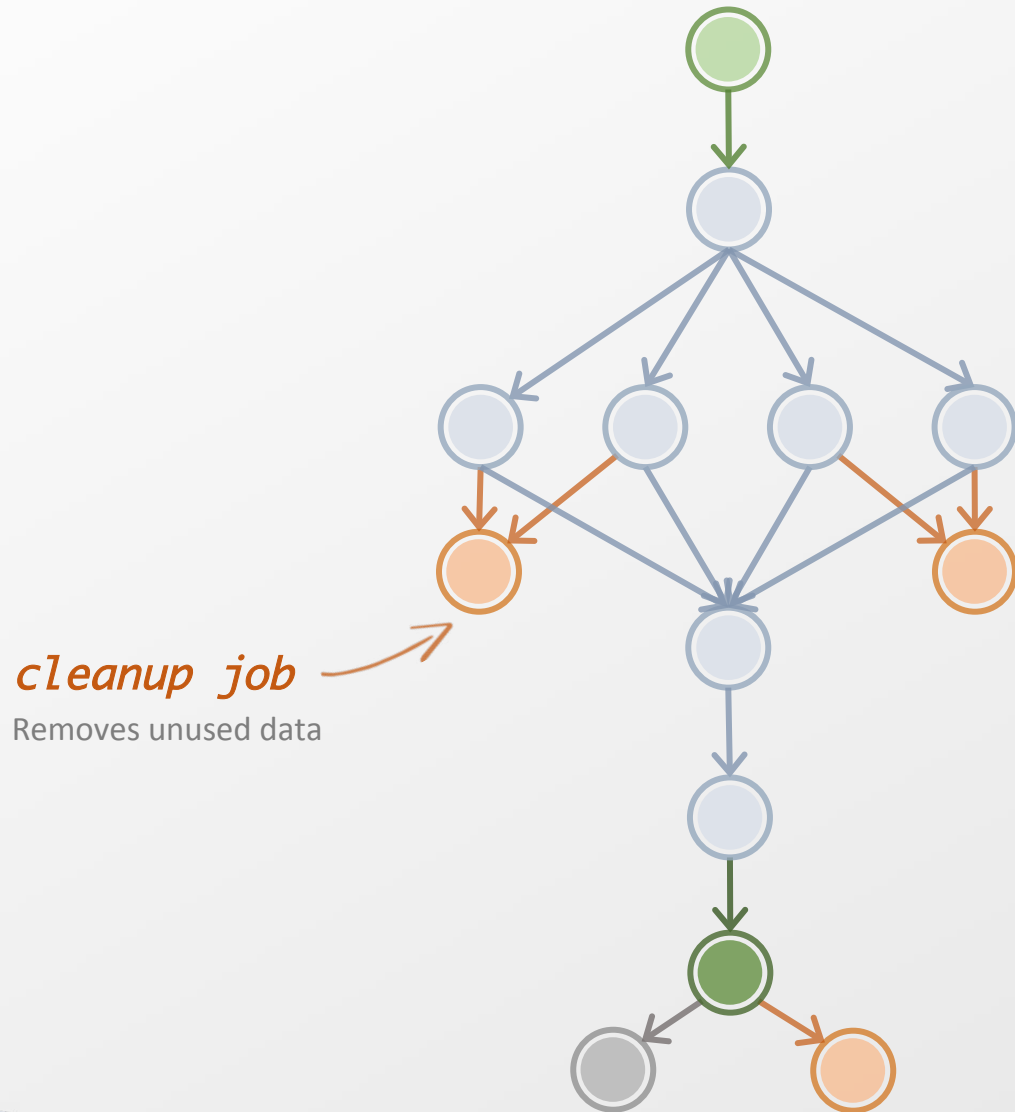
- Users run out of disk space while running workflows

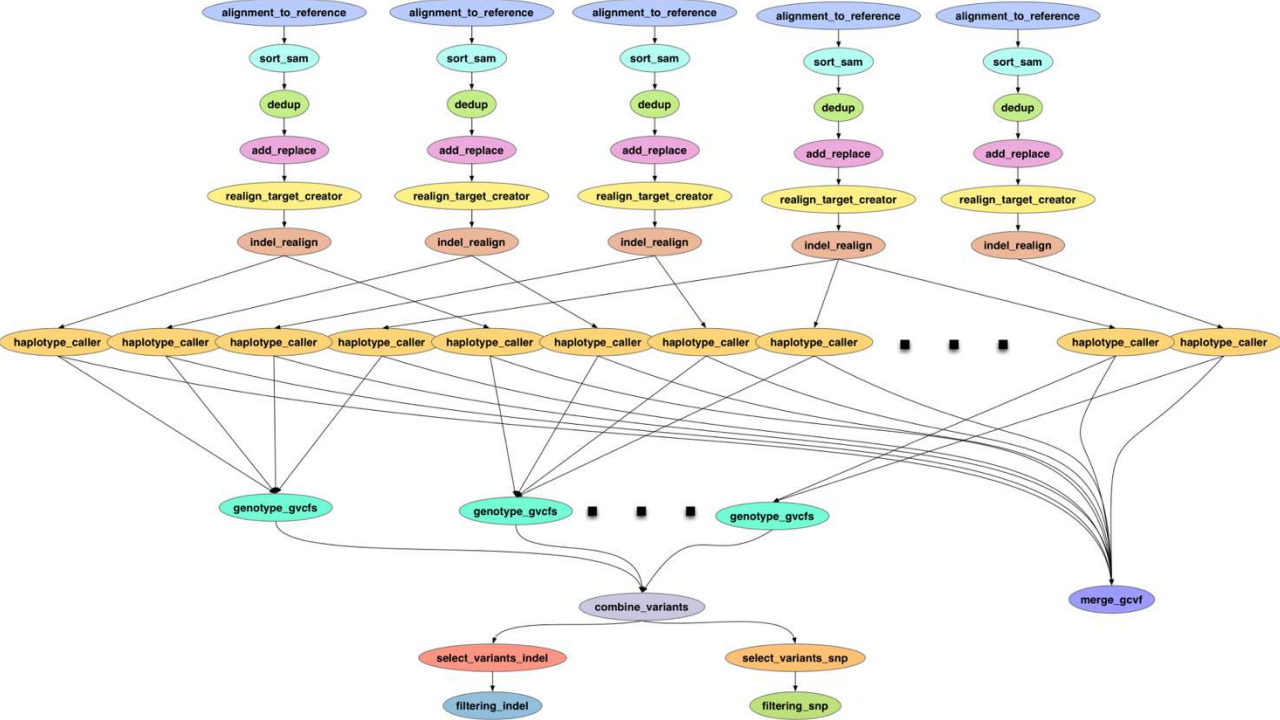
Why does it occur

- 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





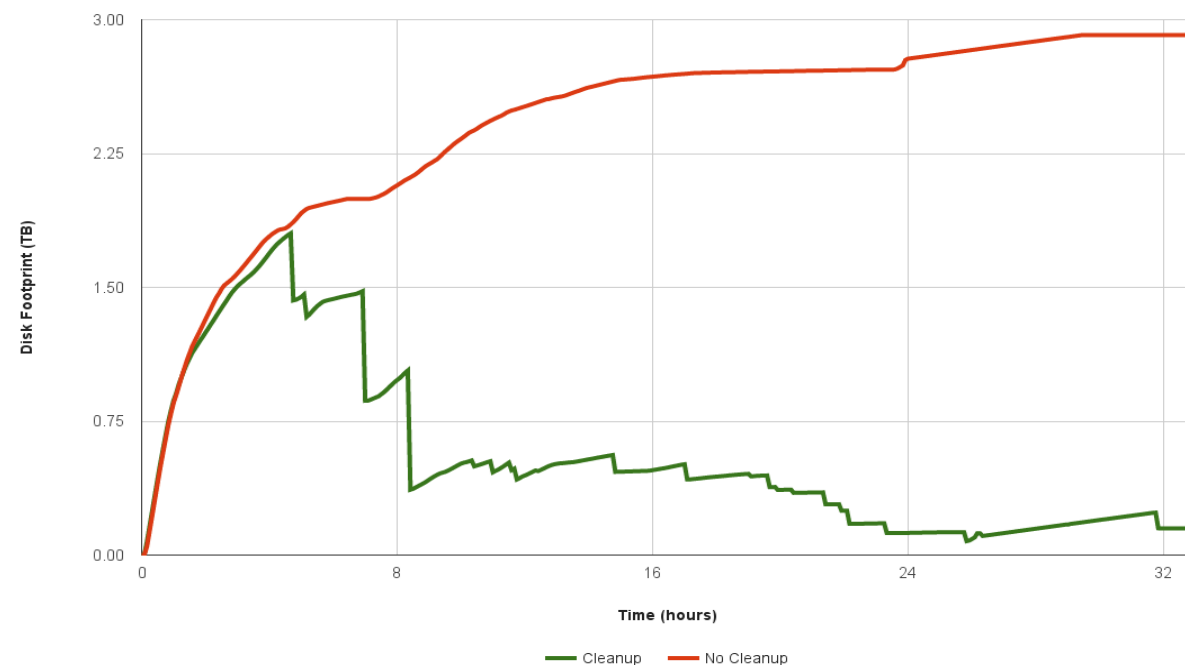
Task	Base Code	Cores (Threads)	Memory (GB)
Alignment_to_reference	BWA	7	8
Sort_sam	Picard	1	21
Dedup	Picard	1	21
Add_replace	Picard	1	21
Realign_target_creator	GATK	15	10
Indel_realign	GATK	1	10
Haplotype_caller	GATK	1	3
Genotype_gvcfs	GATK	1	10
Merge_gvcf	GATK	10	20
Combine_variants	GATK	1	10
Select_variants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

TACC Wrangler as Execution Environment

Flash Based Shared Storage

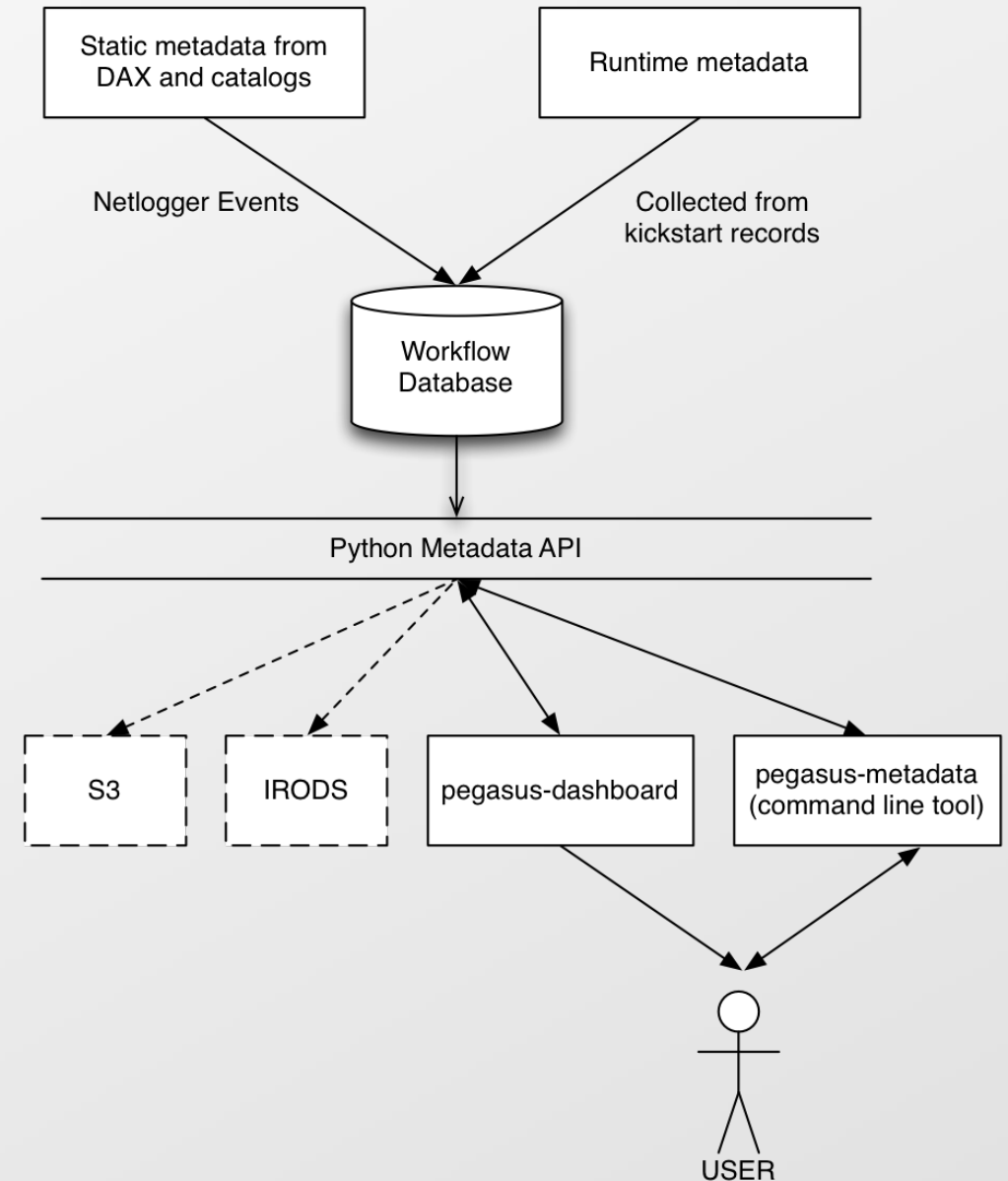
Switched to glideins (pilot jobs) - Brings in remote compute nodes and joins them to the HTCondor pool on in the submit host - Workflow runs at a finer granularity

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



Metadata

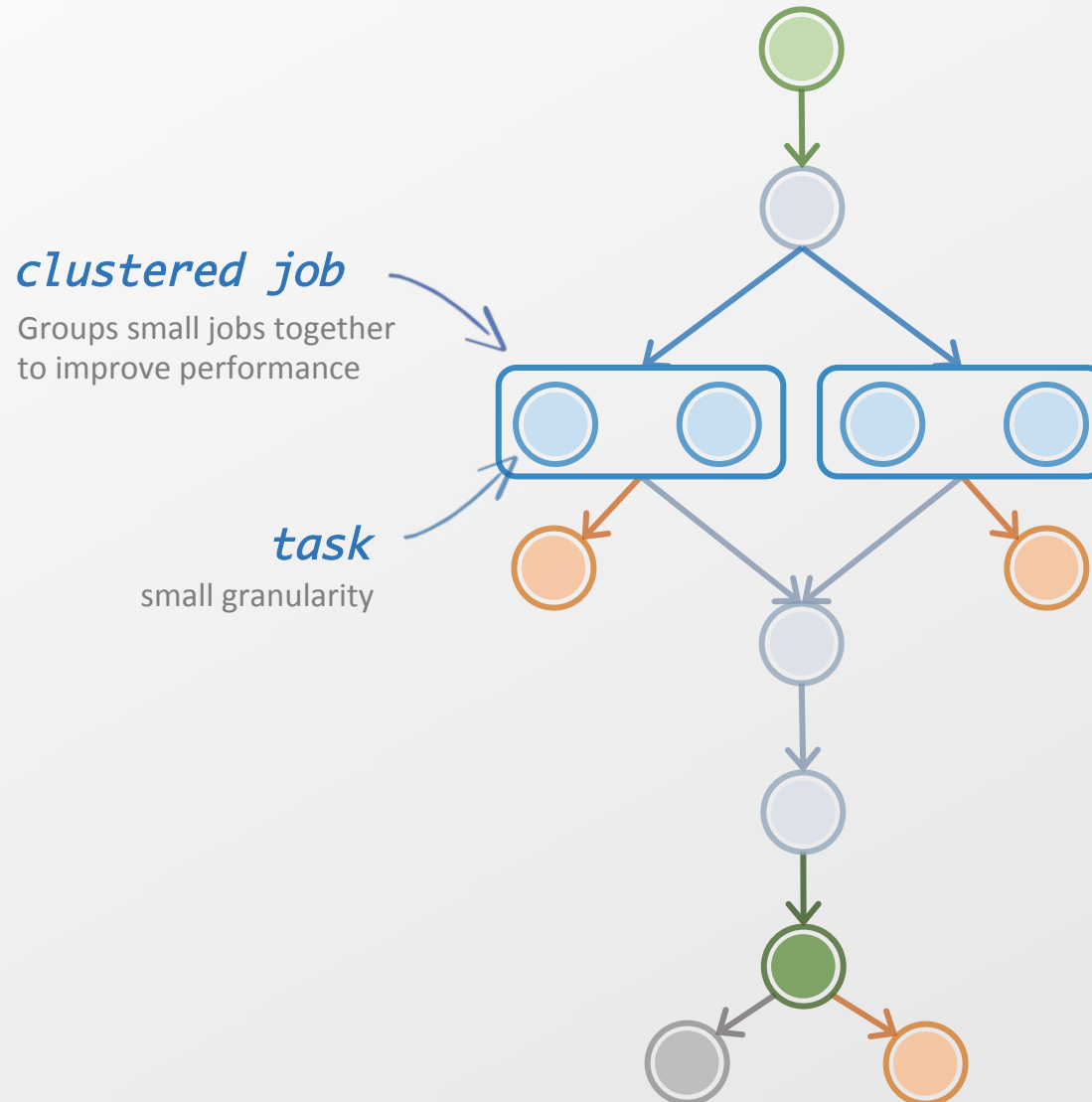
- Can associate arbitrary key-value pairs with workflows, jobs, and files
- Data registration
 - Output files get tagged with metadata on registration in the workflow database.
- Static and runtime metadata
 - Static: application parameters
 - Runtime: performance metrics



Introduced in Pegasus 4.6

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

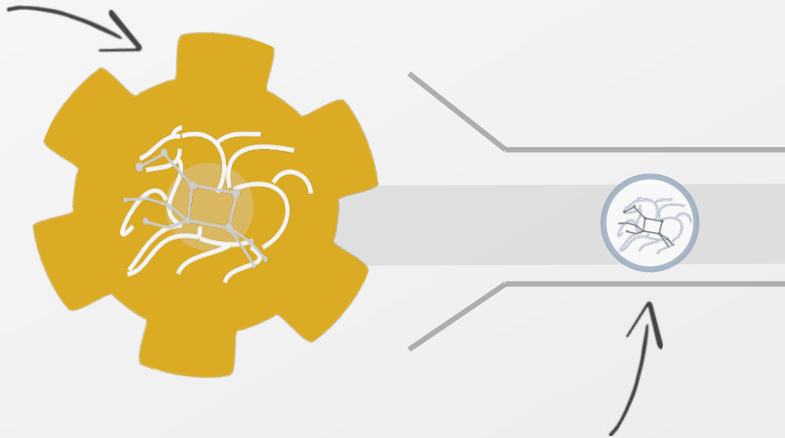
• Pegasus Solutions

- 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...

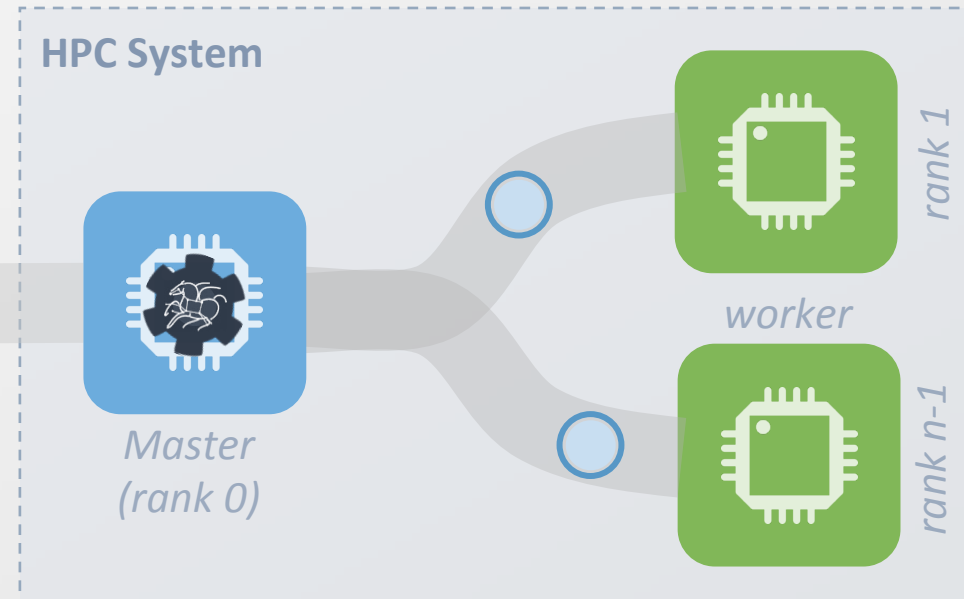
workflow restructuring
workflow reduction
hierarchical workflows
pegasus-mpi-cluster

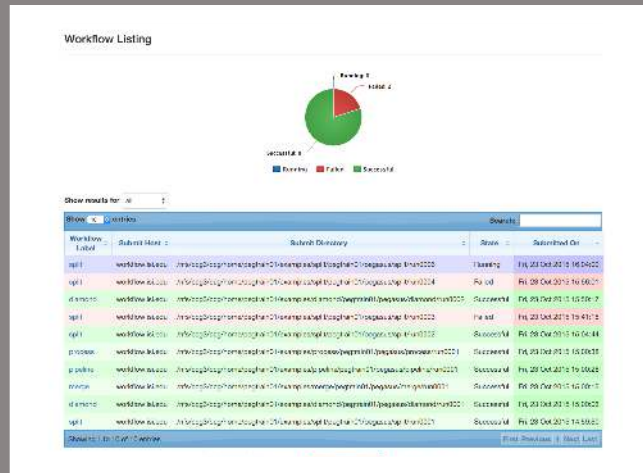
submit host
(e.g., user's laptop)



workflow wrapped as an MPI job

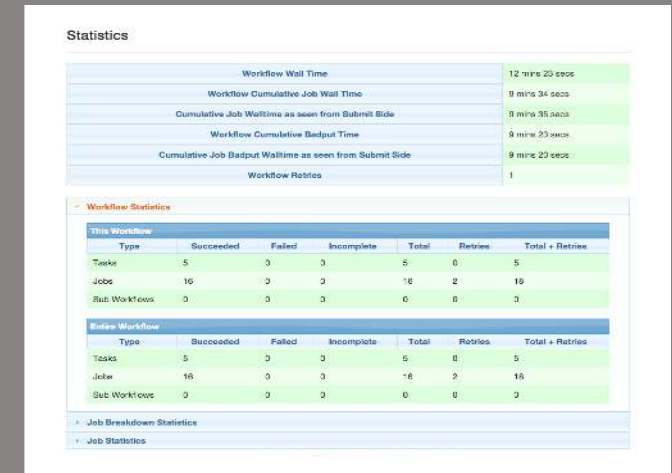
Allows sub-graphs of a Pegasus workflow to be submitted as monolithic jobs to remote resources






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

> 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
-----
```

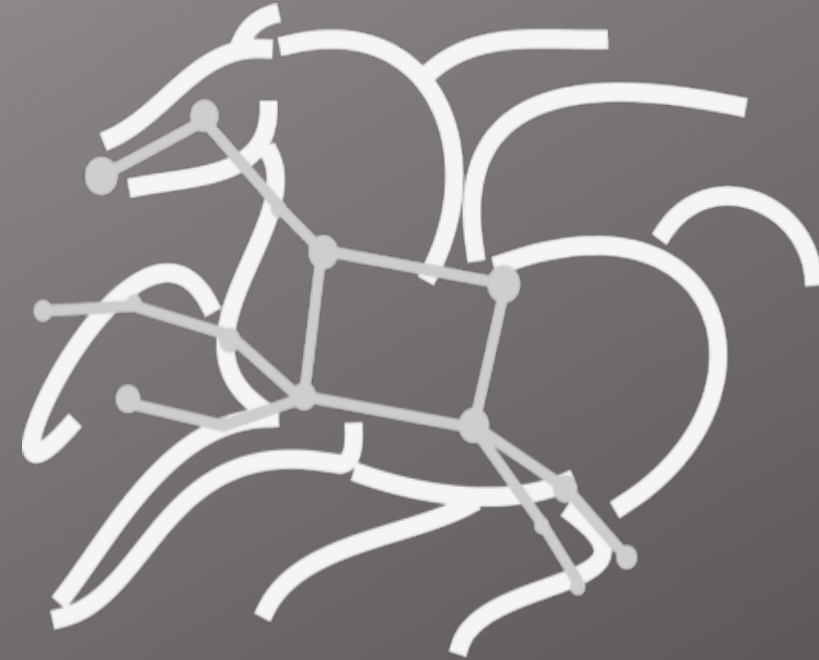
```
Workflow wall time : 2 mins, 6 secs
Workflow cumulative job wall time : 38 secs
Cumulative job wall time as seen from submit side : 42 secs
Workflow cumulative job badput wall time :
Cumulative job badput wall time as seen from submit side :
```

...Pegasus provides
a set of concise
and powerful tools

Upcoming Features

To be released with:

Pegasus 4.8



Automatic Integrity Checking

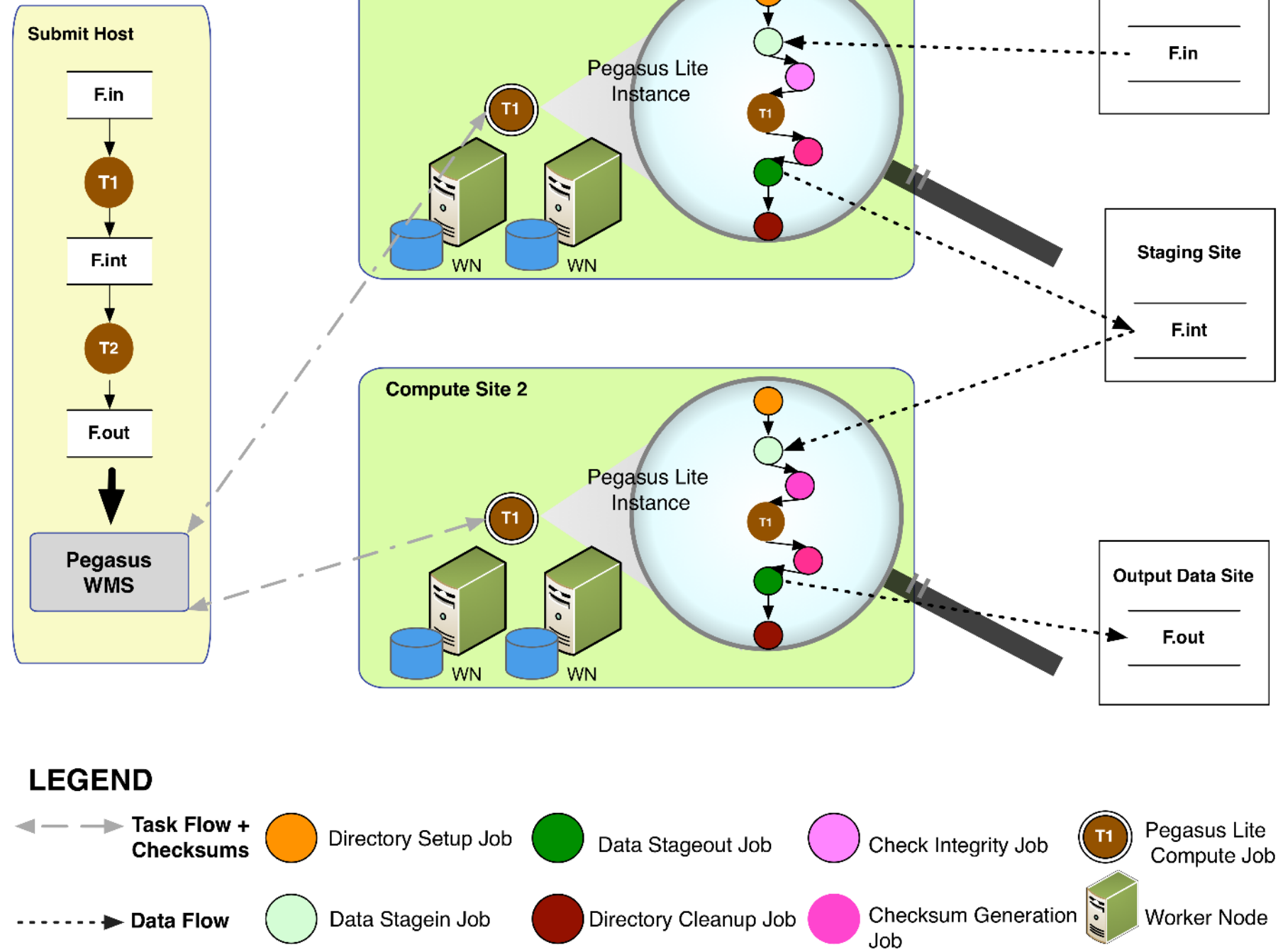
- Pegasus will perform integrity checksums on input files before a job starts on the remote node.

For raw inputs, checksums specified in the input replica catalog along with file locations

All intermediate and output files checksums are generated and tracked within the system.

Support for sha256 checksums

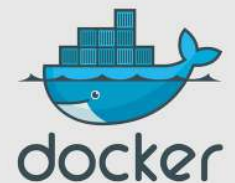
- Failure** is triggered if checksums fail



Proposed 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**. 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.
- 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)*

Plan to support Docker containers initially (due to users preference), but will be supporting Singularity



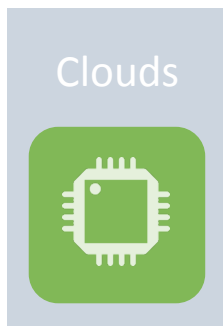
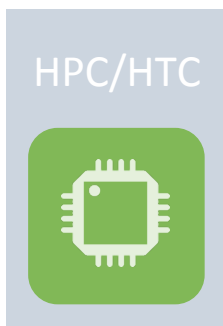
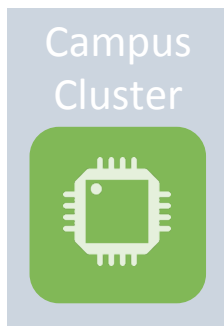
Data Management for Containers

- Users can refer to container images as
 - Docker 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 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.

Running Pegasus workflows with Jupyter



WAN LAN



Jupyter Pegasus-Tutorial-Split Last Checkpoint: 03/15/2017 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Python 2

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: a relative path of the file from the
/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*

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

running a workflow

```
# 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"))
```

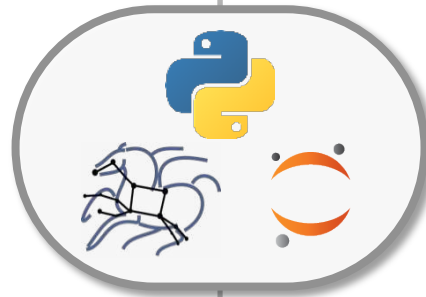
```
dax.addJob(split)
```

*using the Pegasus DAX3 API
to write a workflow*

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

monitoring a workflow execution

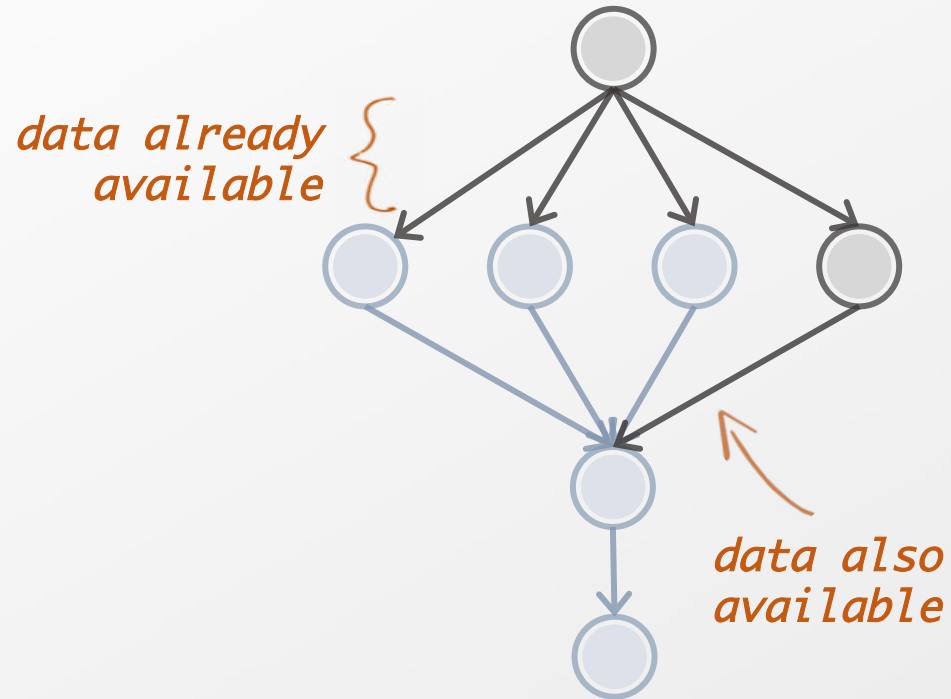
```
Progress: 100.0% (Success) (Completed: 17, Queued: 0, Running: 0, Failed: 0)
```



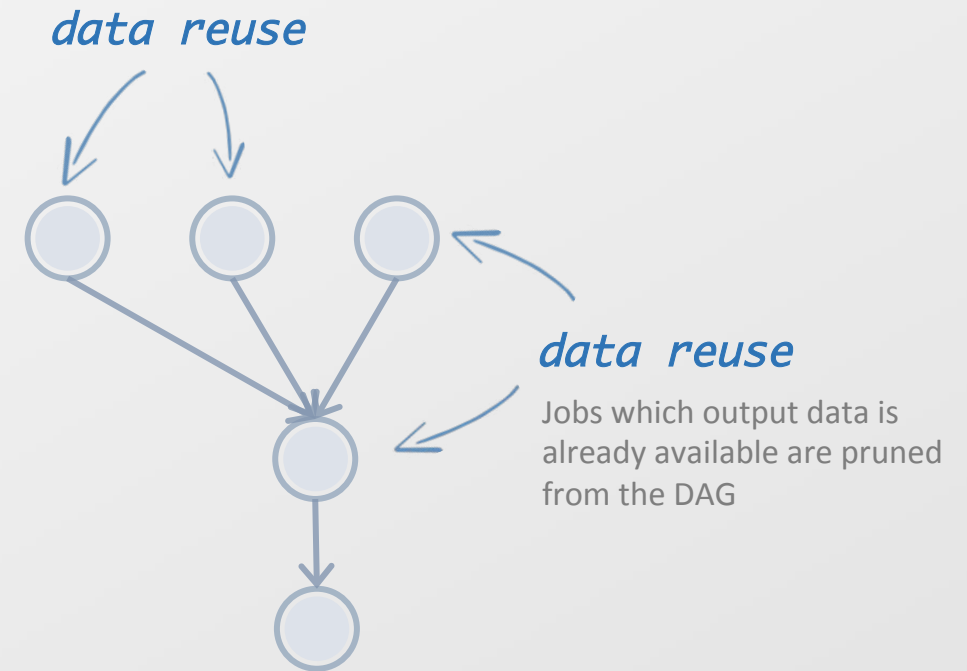
Other Pegasus Capabilities...

What about data reuse?

workflow restructuring
workflow reduction
hierarchical workflows

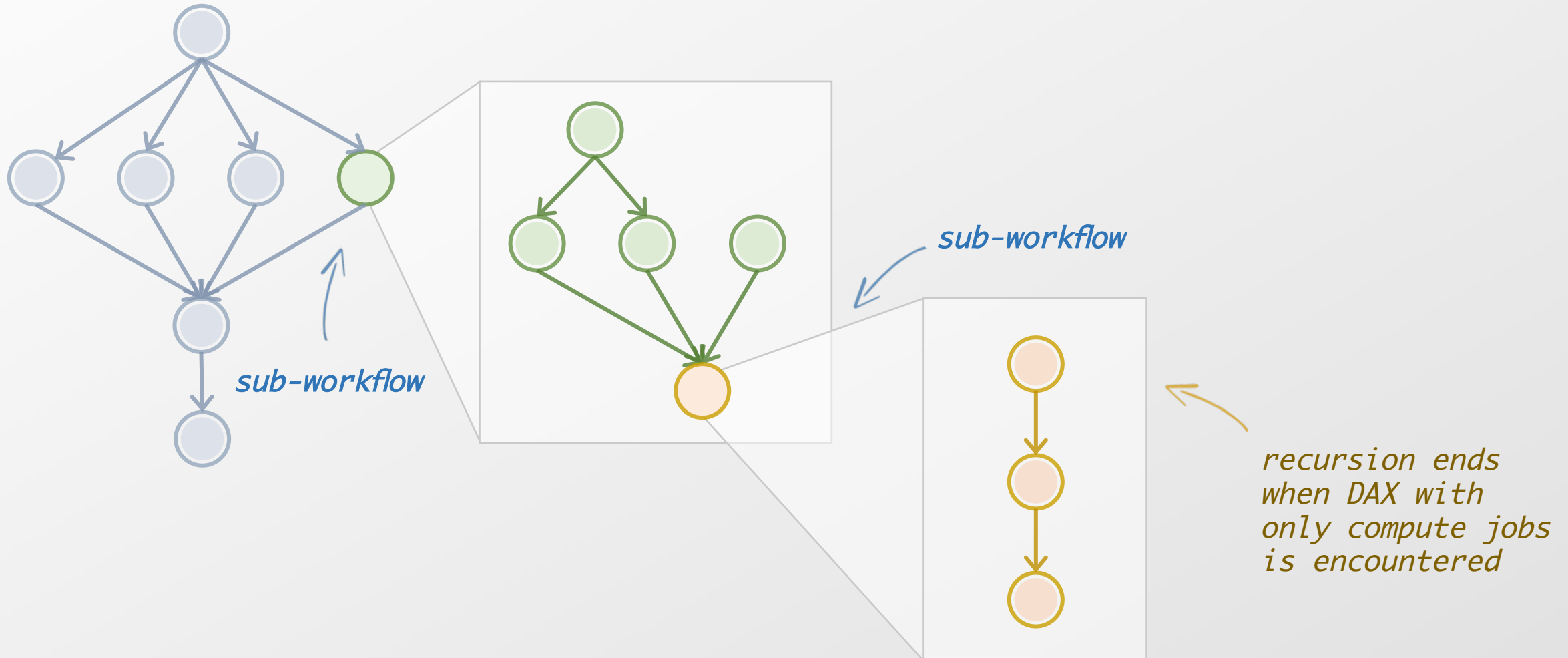


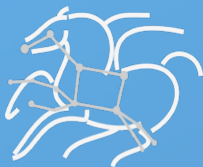
workflow
reduction



Pegasus also handles large-scale workflows

workflow restructuring
workflow reduction
hierarchical workflows





Pegasus

est. 2001

Automate, recover, and debug scientific computations.

Get Started

Pegasus Website

<http://pegasus.isi.edu>

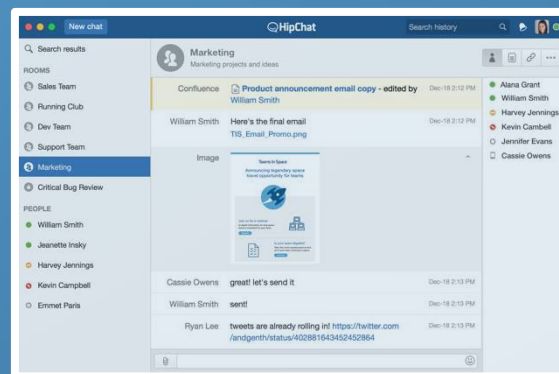
Users Mailing List

pegasus-users@isi.edu

Support

pegasus-support@isi.edu

HipChat





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



U.S. DEPARTMENT OF
ENERGY

