# Pegasus

Automate, recover, and debug scientific computations.

**Mats Rynge**
**rynge@isi.edu**

USC Viterbi
School of Engineering
*Information Sciences Institute*

*https://pegasus.isi.edu*

# *Why* Pegasus ?

Automates complex, multi-stage processing pipelines

Enables parallel, distributed computations

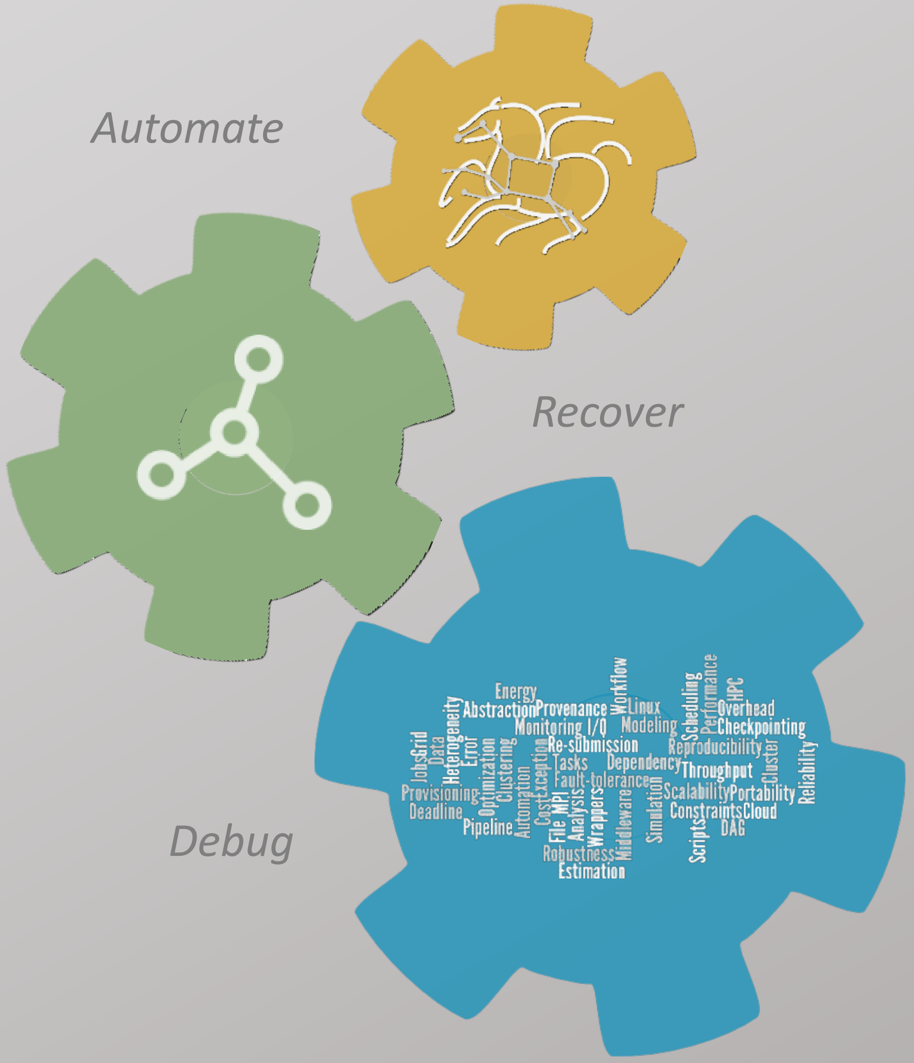
Automatically executes data transfers

Reusable, aids reproducibility

Records how data was produced (provenance)

Handles failures with to provide reliability

Keeps track of data and files



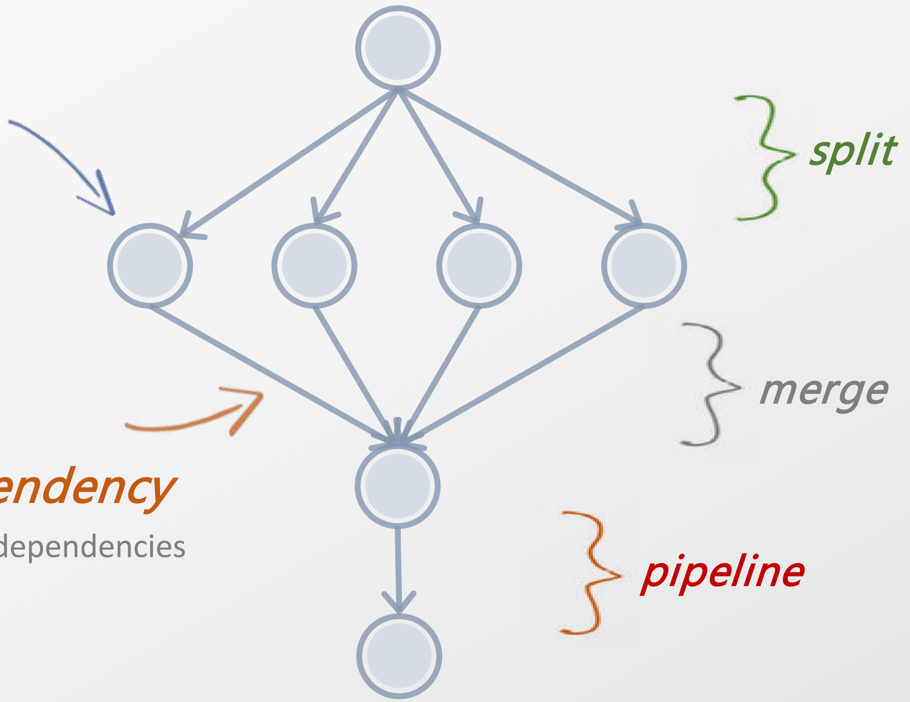
HTCondor
High Throughput Computing

Pegasus

# Taking a closer look into a workflow…

DAG

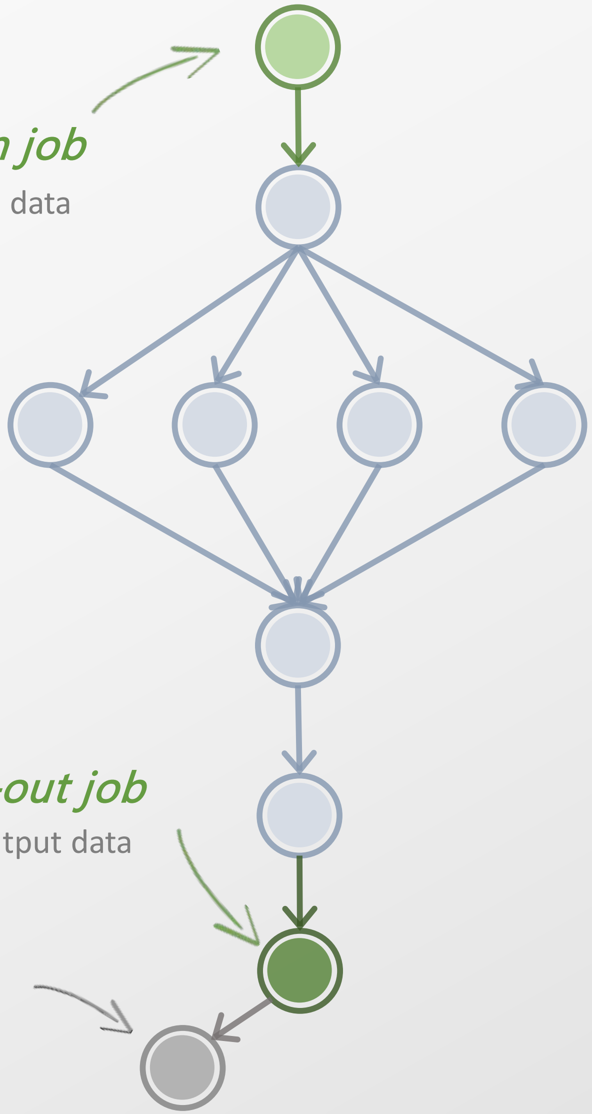
directed-acyclic graphs



DAX

DAG in XML

# From the abstraction to execution!



*stage-in job*

Transfers the workflow input data

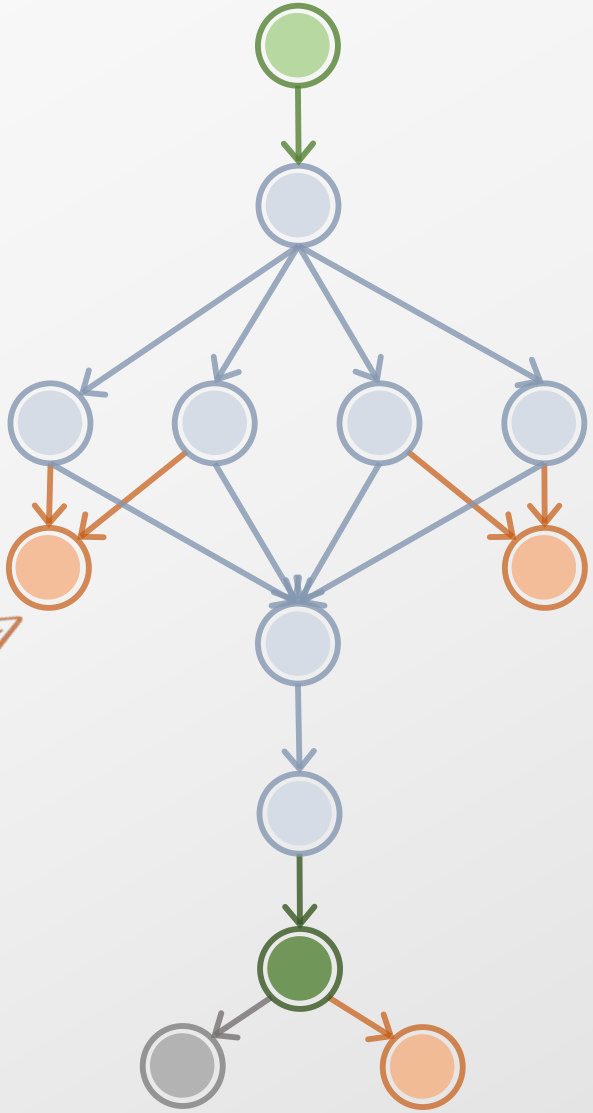
*stage-out job*

Transfers the workflow output data

*registration job*

Registers the workflow output data

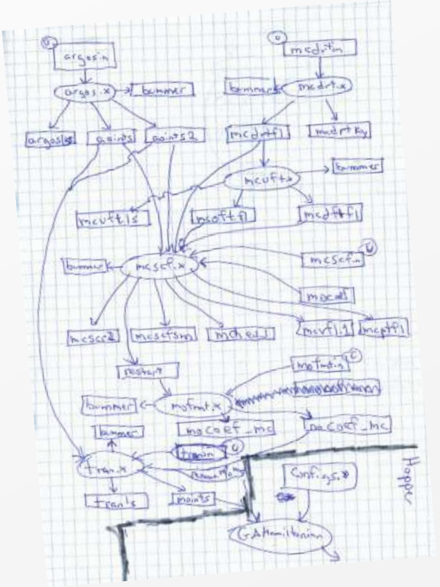
# Optimizing storage usage…



*cleanup job*

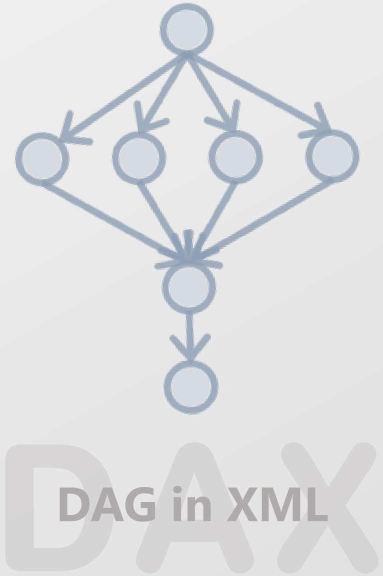
Removes unused data

# Pegasus also provides tools to generate the abstract workflow



python Java perl

```
dax = ADAG("test_dax")
firstJob = Job(name="first_job")
firstInputFile = File("input.txt")
firstOutputFile = File("tmp.txt")
firstJob.addArgument("input=input.txt", "output=tmp.txt")
firstJob.uses(firstInputFile, link=Link.INPUT)
firstJob.uses(firstOutputFile, link=Link.OUTPUT)
dax.addJob(firstJob)
for i in range(0, 5):
    simulJob = Job(id="%s" % (i+1), name="simul_job”)
    simulInputFile = File("tmp.txt”)
    simulOutputFile = File("output.%d.dat" % i)
    simulJob.addArgument("parameter=%d" % i, "input=tmp.txt”,
        output=%s" % simulOutputFile.getName())
    simulJob.uses(simulInputFile, link=Link.INPUT)
    simulJob.uses(simulOutputFile, line=Link.OUTPUT)
dax.addJob(simulJob)
dax.depends(parent=firstJob, child=simulJob)
fp = open("test.dax", "w”)
dax.writeXML(fp)
fp.close()
```



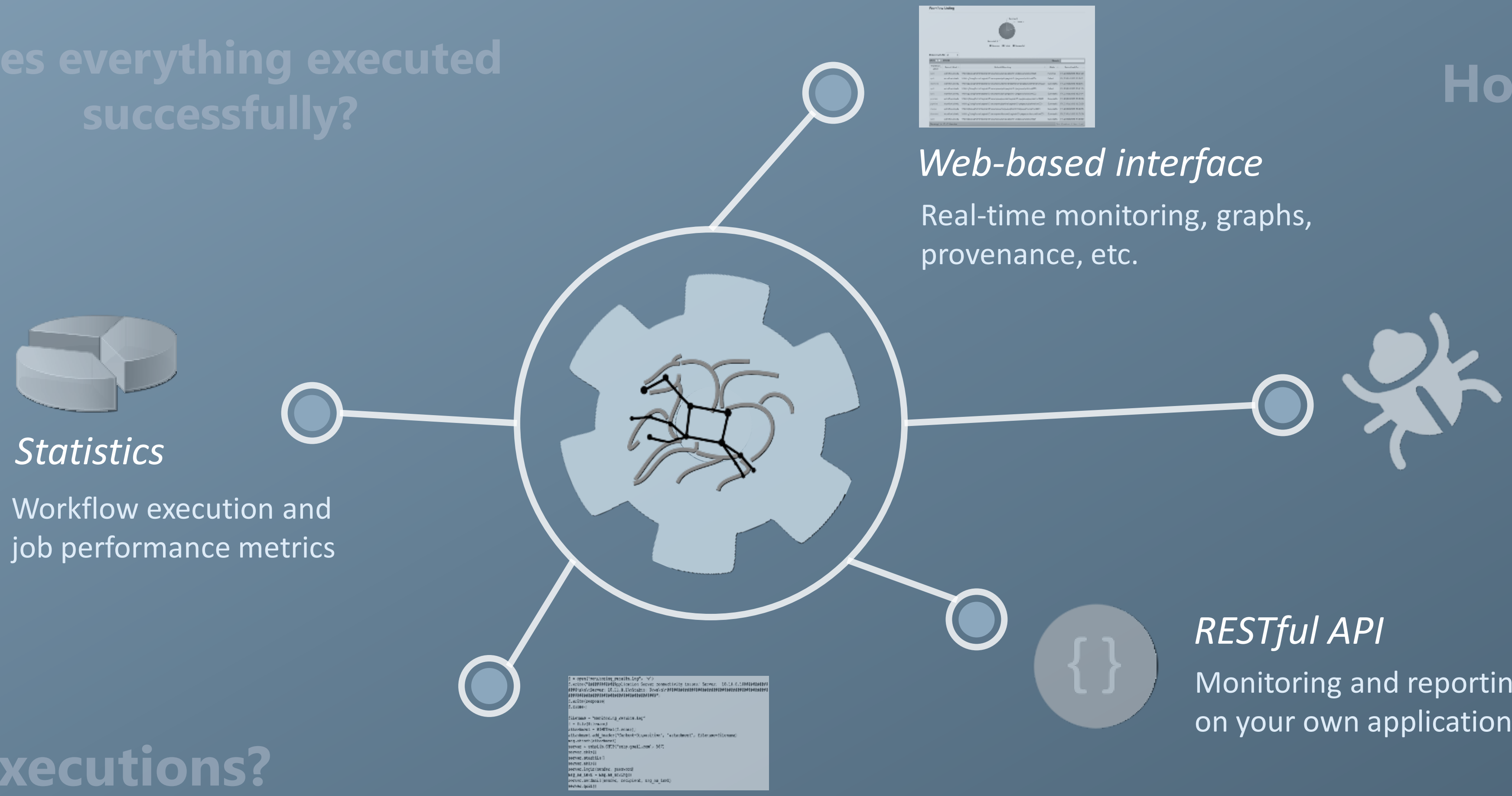
# While you wait…

# …or the execution is finished.

Does everything executed successfully?

How my workflow behaves?

Past executions?



*Web-based interface*

Real-time monitoring, graphs, provenance, etc.

*Statistics*

Workflow execution and job performance metrics

*Debug*

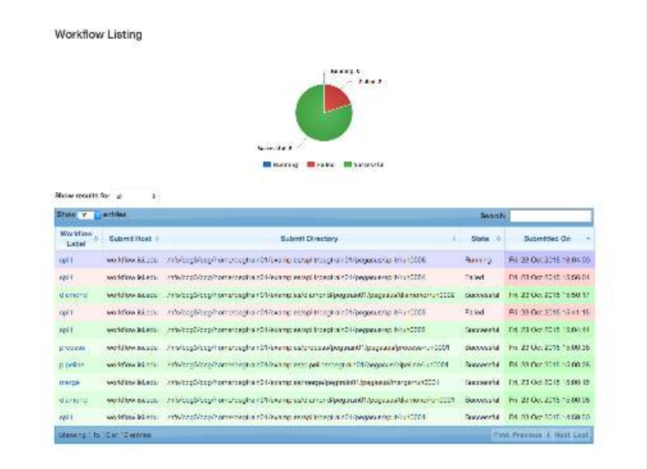
Set of debugging tools to unveil issues

*RESTful API*

Monitoring and reporting information on your own application interface

*Command-line tools*

Tools for monitor and debug workflows



Workflow Listing

Show results for: all

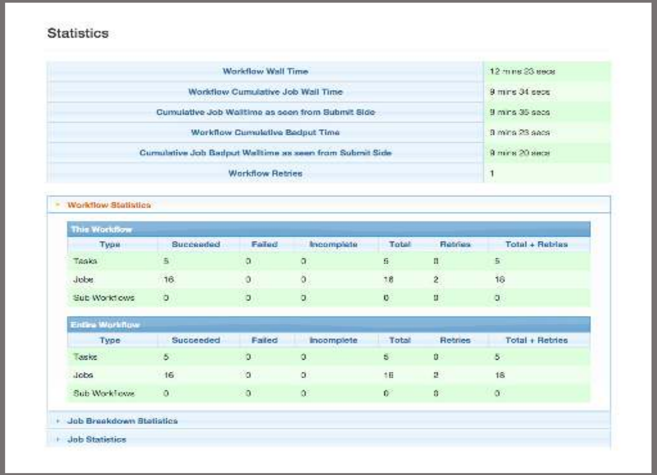
Show 10 entries Search:

| 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 19:04:00 |
| split | workflow.isi.edu | /nfs/ccg3/ccg/home/pegtrain01/examples/split/pegtrain01/pegasus/split/run0004 | Failed | Fri 23 Oct 2015 18:56:04 |
| diamond | workflow.isi.edu | /nfs/ccg3/ccg/home/pegtrain01/examples/diamond/pegtrain01/pegasus/diamond/run0002 | Successful | Fri 23 Oct 2015 18:50:17 |
| split | workflow.isi.edu | /nfs/ccg3/ccg/home/pegtrain01/examples/split/pegtrain01/pegasus/split/run0003 | Failed | Fri 23 Oct 2015 18:41:15 |
| split | workflow.isi.edu | /nfs/ccg3/ccg/home/pegtrain01/examples/split/pegtrain01/pegasus/split/run0003 | Successful | Fri 23 Oct 2015 18:04:44 |
| process | workflow.isi.edu | /nfs/ccg3/ccg/home/pegtrain01/examples/process/pegtrain01/pegasus/process/run0001 | Successful | Fri 23 Oct 2015 16:00:36 |
| pipeline | workflow.isi.edu | /nfs/ccg3/ccg/home/pegtrain01/examples/pipeline/pegtrain01/pegasus/pipeline/run0001 | Successful | Fri 23 Oct 2015 16:00:26 |
| merge | workflow.isi.edu | /nfs/ccg3/ccg/home/pegtrain01/examples/merge/pegtrain01/pegasus/merge/run0001 | Successful | Fri 23 Oct 2015 16:00:15 |
| diamond | workflow.isi.edu | /nfs/ccg3/ccg/home/pegtrain01/examples/diamond/pegtrain01/pegasus/diamond/run0001 | Successful | Fri 23 Oct 2015 16:00:05 |
| split | workflow.isi.edu | /nfs/ccg3/ccg/home/pegtrain01/examples/split/pegtrain01/pegasus/split/run0001 | Successful | Fri 23 Oct 2015 14:58:50 |

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 | 18 | 2 | 18 |
| Sub Workflows | 0 | 0 | 0 | 0 | 0 | 0 |

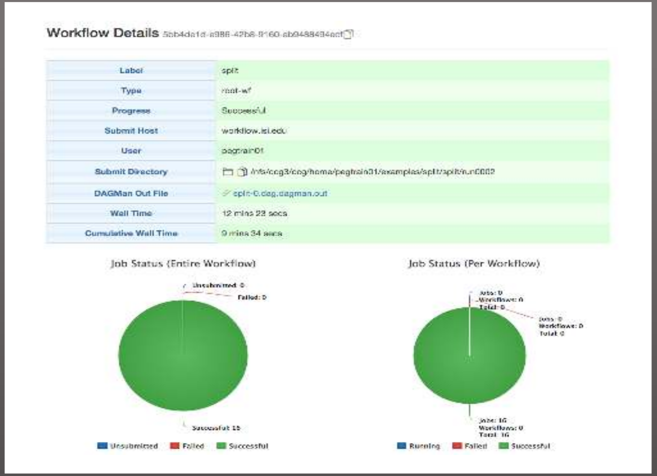
Entire Workflow

| Type | Succeeded | Failed | Incomplete | Total | Retries | Total + Retries |
|---|---|---|---|---|---|---|
| Tasks | 5 | 0 | 0 | 5 | 0 | 5 |
| Jobs | 16 | 0 | 0 | 18 | 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 5bb4da1d-a986-42b6-9160-ab9488494acf

| | |
|---|---|
| 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 22 secs |
| Cumulative Wall Time | 9 mins 34 secs |

Real-time Monitoring

Reporting

Debugging

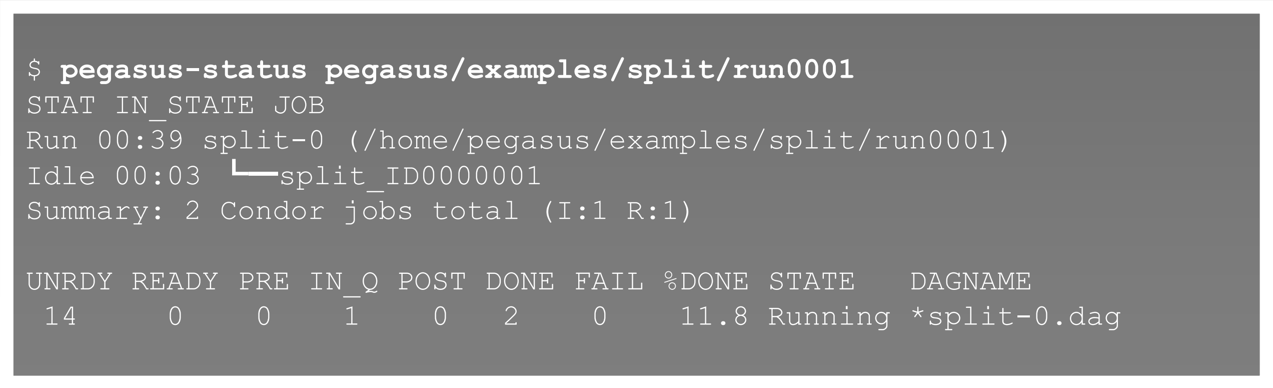
Troubleshooting

RESTful API

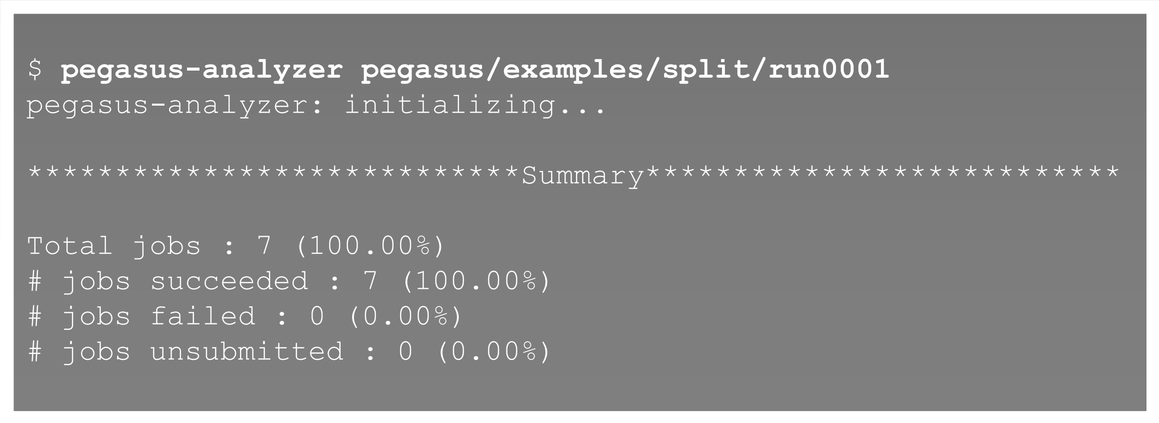
Pegasus

http://pegasus.isi.edu

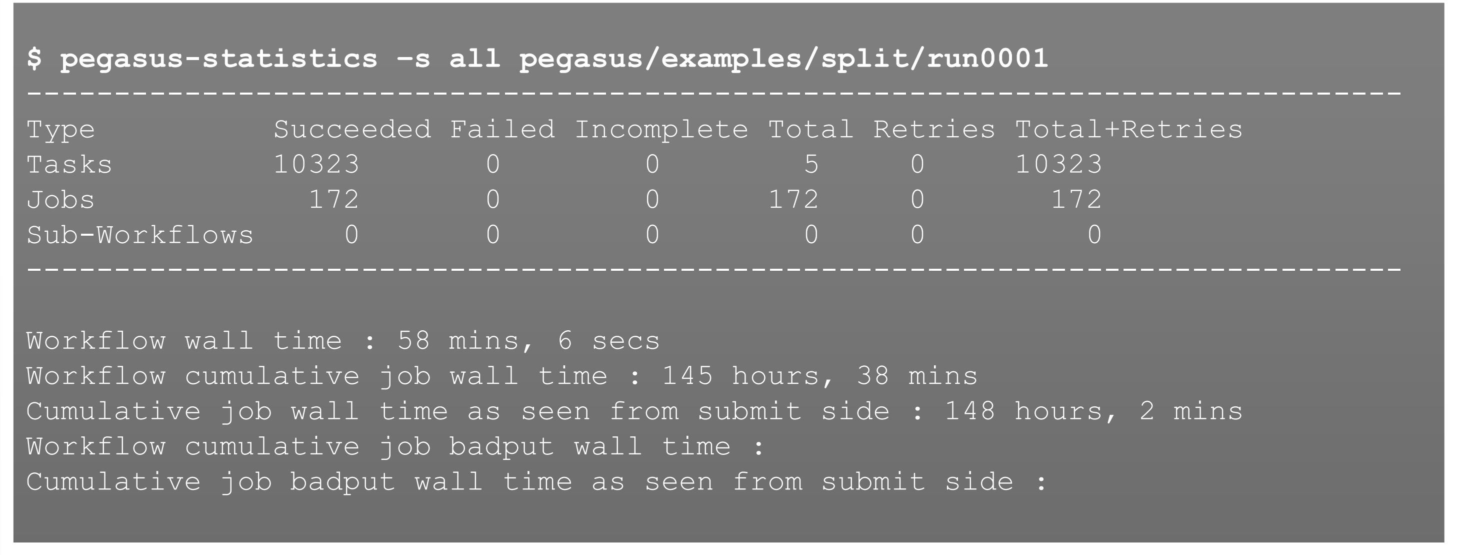
# But, if you prefer the command-line…



```
$ 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
```



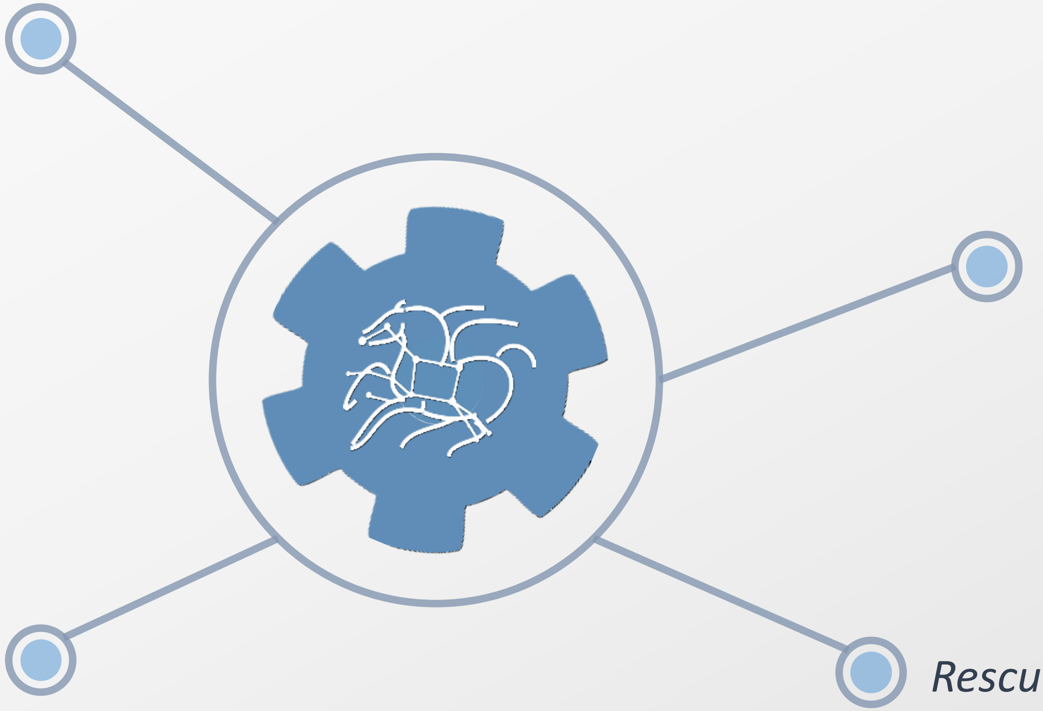
```
$ pegasus-analyzer pegasus/examples/split/run0001
pegasus-analyzer: initializing...

****************************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         10323       0        0        5     0     10323
Jobs            172       0        0      172     0       172
Sub-Workflows     0       0        0        0     0         0
------------------------------------------------------------------------------

Workflow wall time : 58 mins, 6 secs
Workflow cumulative job wall time : 145 hours, 38 mins
Cumulative job wall time as seen from submit side : 148 hours, 2 mins
Workflow cumulative job badput wall time :
Cumulative job badput wall time as seen from submit side :
```

…Pegasus provides a set of <u>concise</u> and <u>powerful</u> tools

# And if a job fails?



*Job Failure Detection*

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

SRM

http

Local disk

Amazon S3

Worried about

# data?

Let Pegasus manage it for you

GridFTP

Shared filesystem

ftp

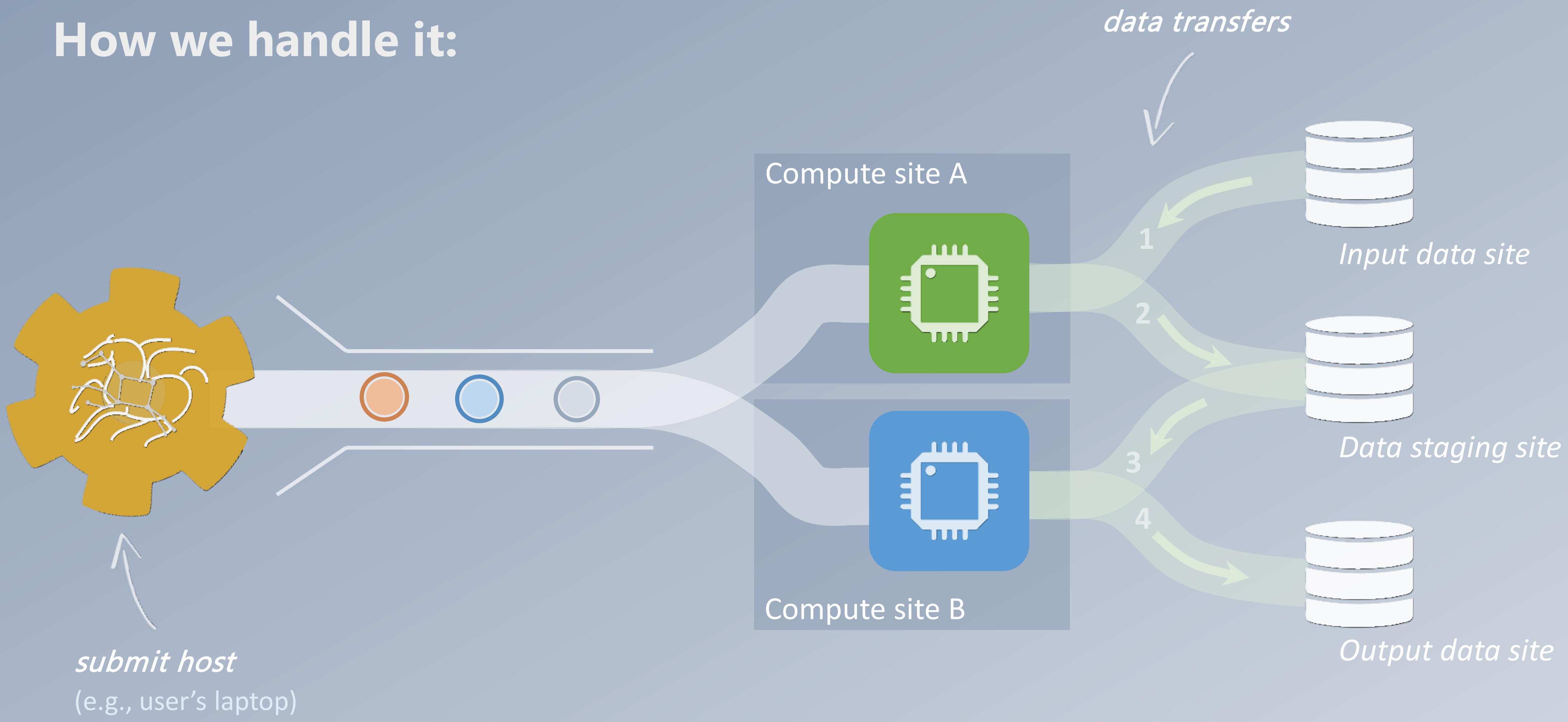
scp

Google Storage

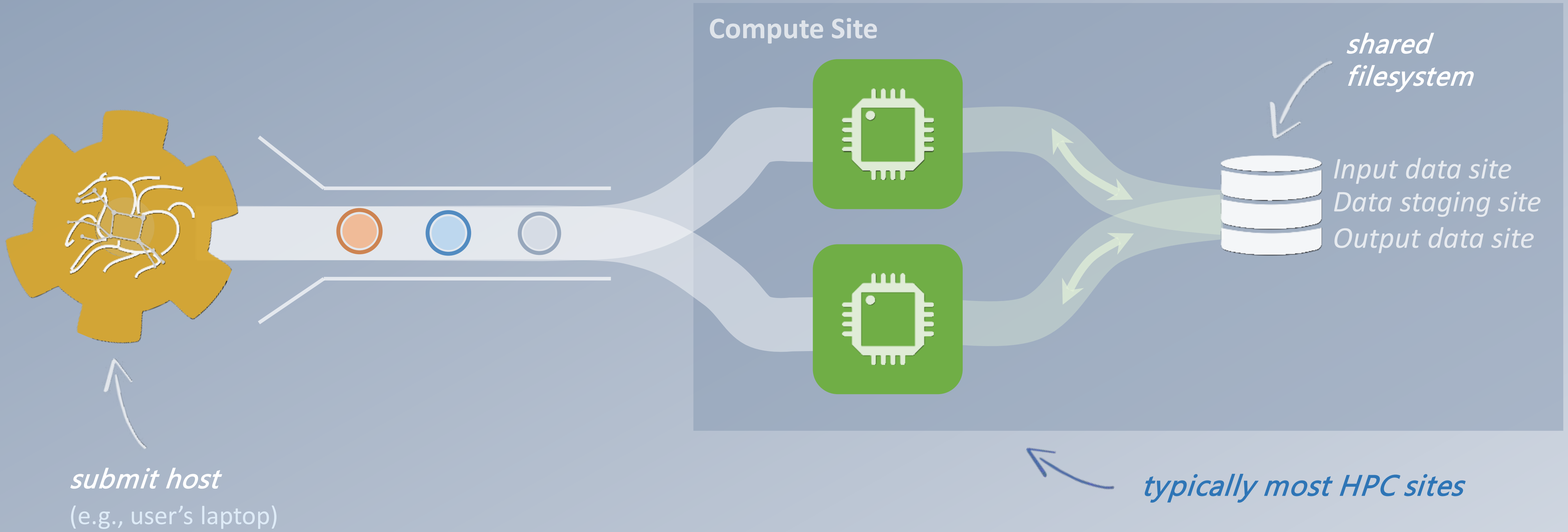
StashCache

iRODS

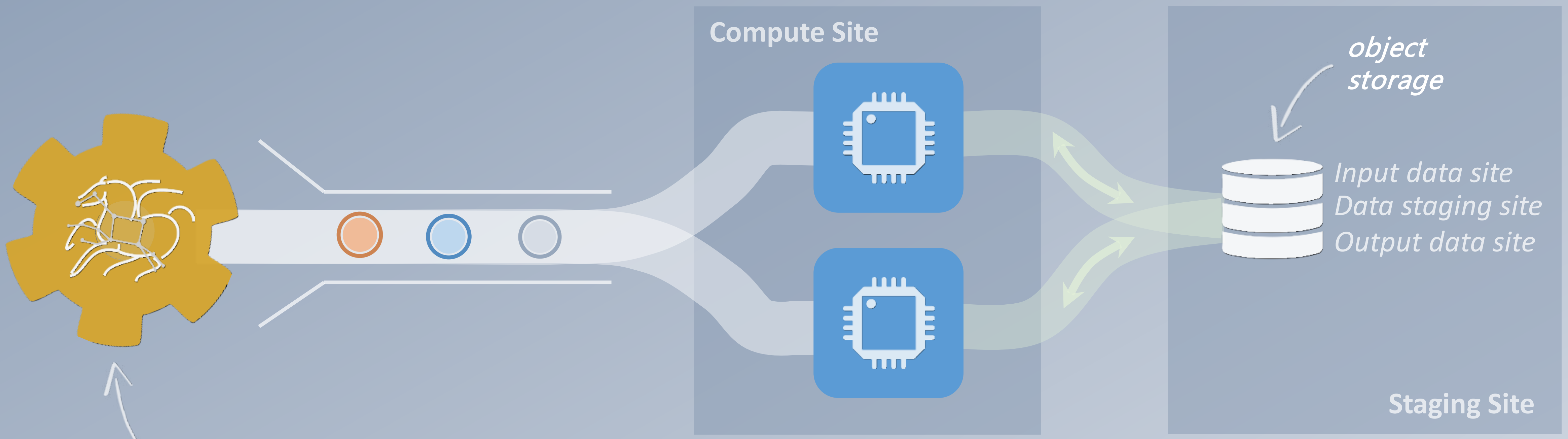
# How we handle it:



# However, there are several possible configurations for data sites…



# Pegasus also handles high-scalable object storages

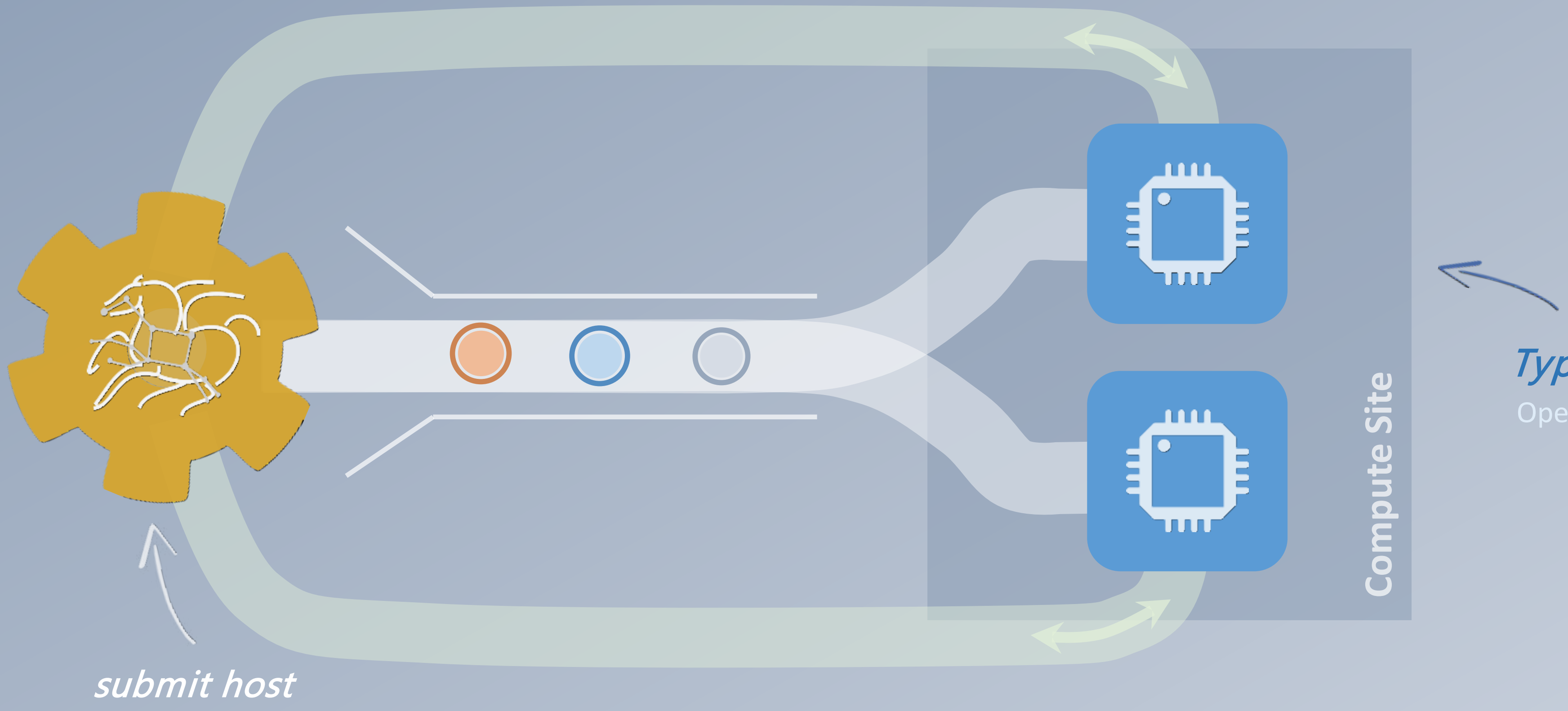


*submit host*

(e.g., user's laptop)

*Typical cloud computing deployment (Amazon S3, Google Storage)*

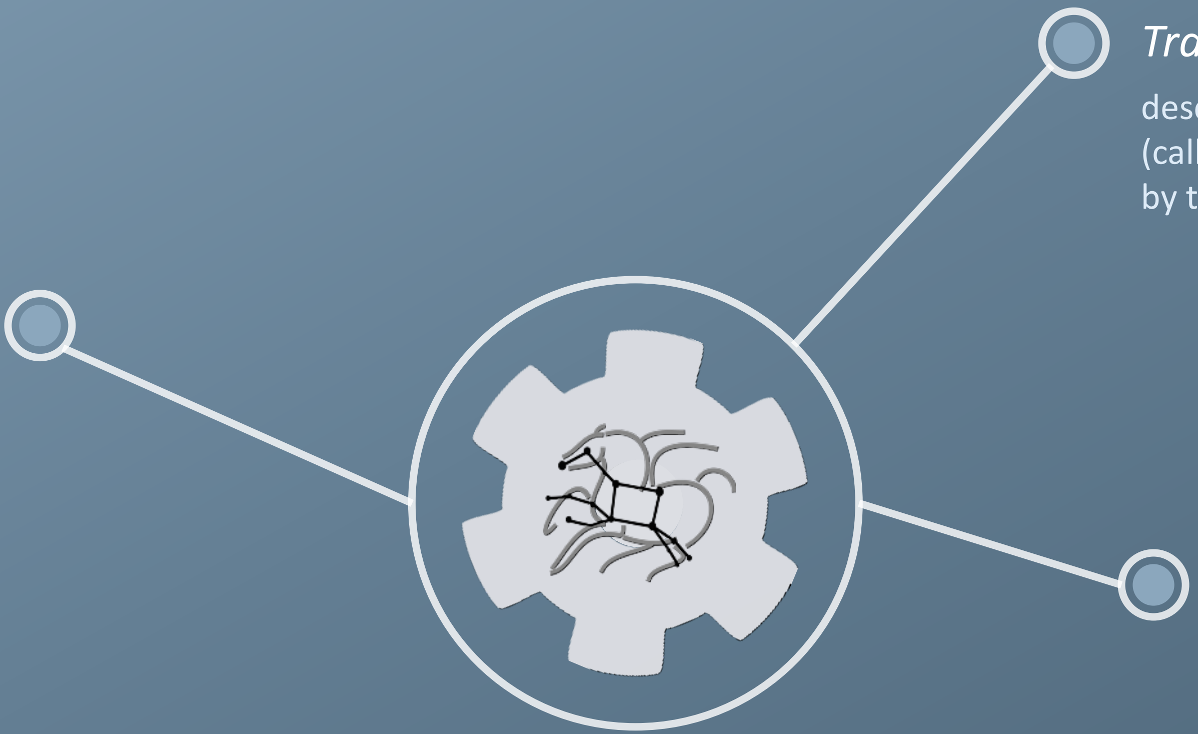
# Pegasus can also manage data over the submit host…



*Typical OSG sites*

Open Science Grid

# So, what information does Pegasus need?



*Transformation Catalog*

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

*Site Catalog*

describes the sites where the workflow jobs are to be executed

*Replica Catalog*

describes all of the input data stored on external servers

# A few more features…

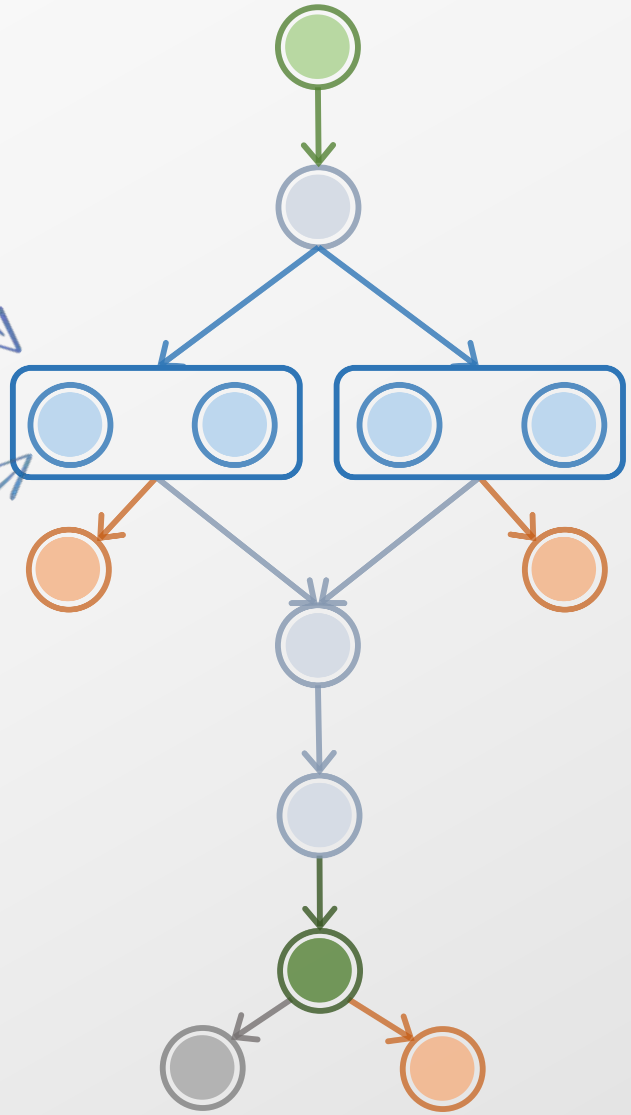
# Performance, why not improve it?

workflow restructuring

workflow reduction

hierarchical workflows

pegasus-mpi-cluster



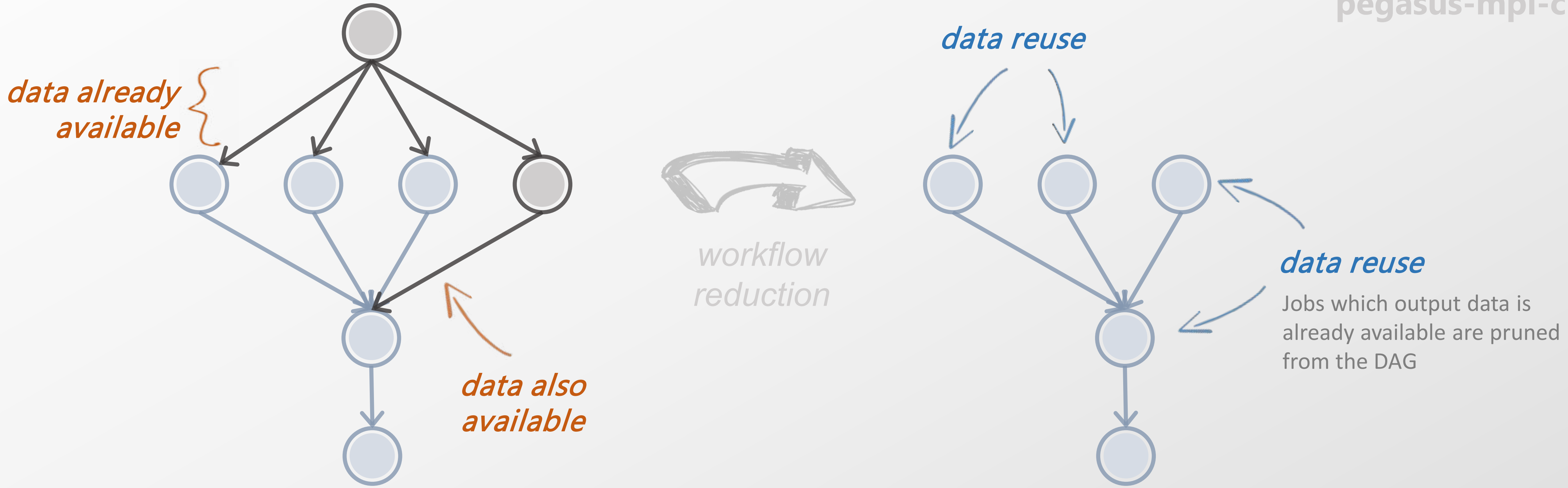
*clustered job*

Groups small jobs together to improve performance

*task*

small granularity

# What about **data reuse**?



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

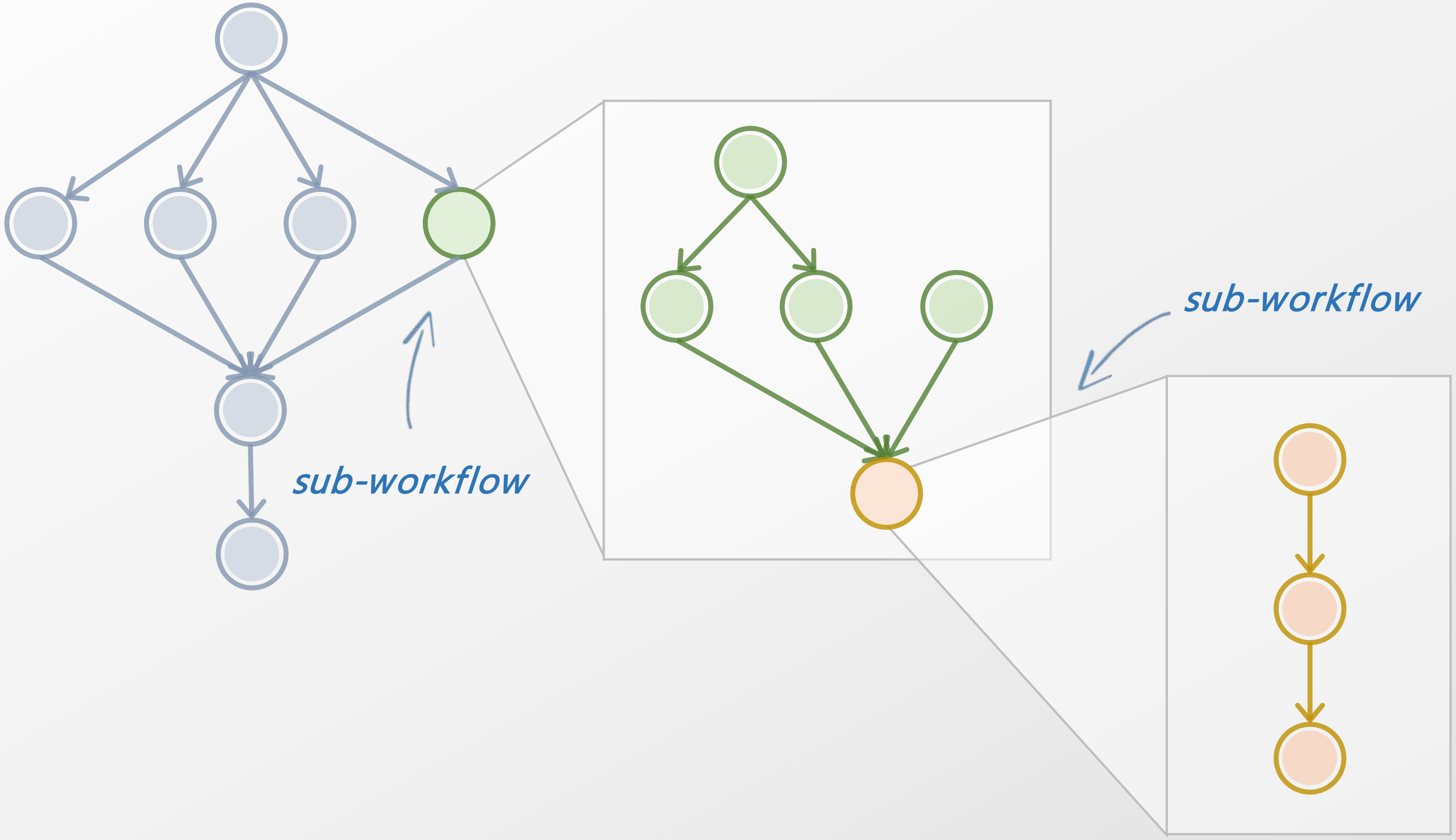
# Pegasus also handles **large-scale workflows**

workflow restructuring

workflow reduction

**hierarchical workflows**

pegasus-mpi-cluster



*recursion ends when DAX with only compute jobs is encountered*

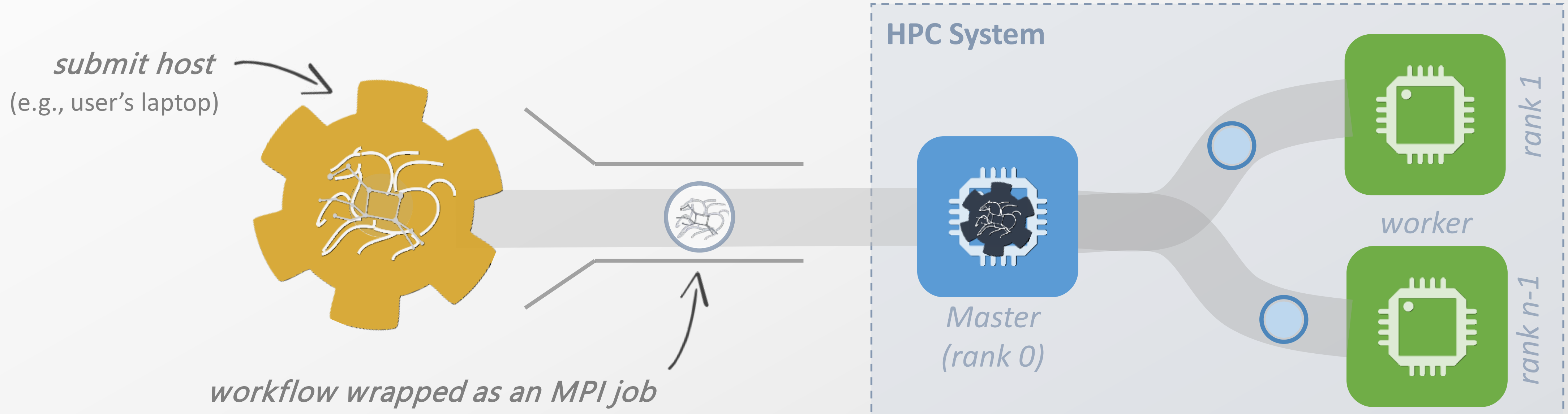
# Running **fine-grained** workflows on HPC systems…

workflow restructuring

workflow reduction

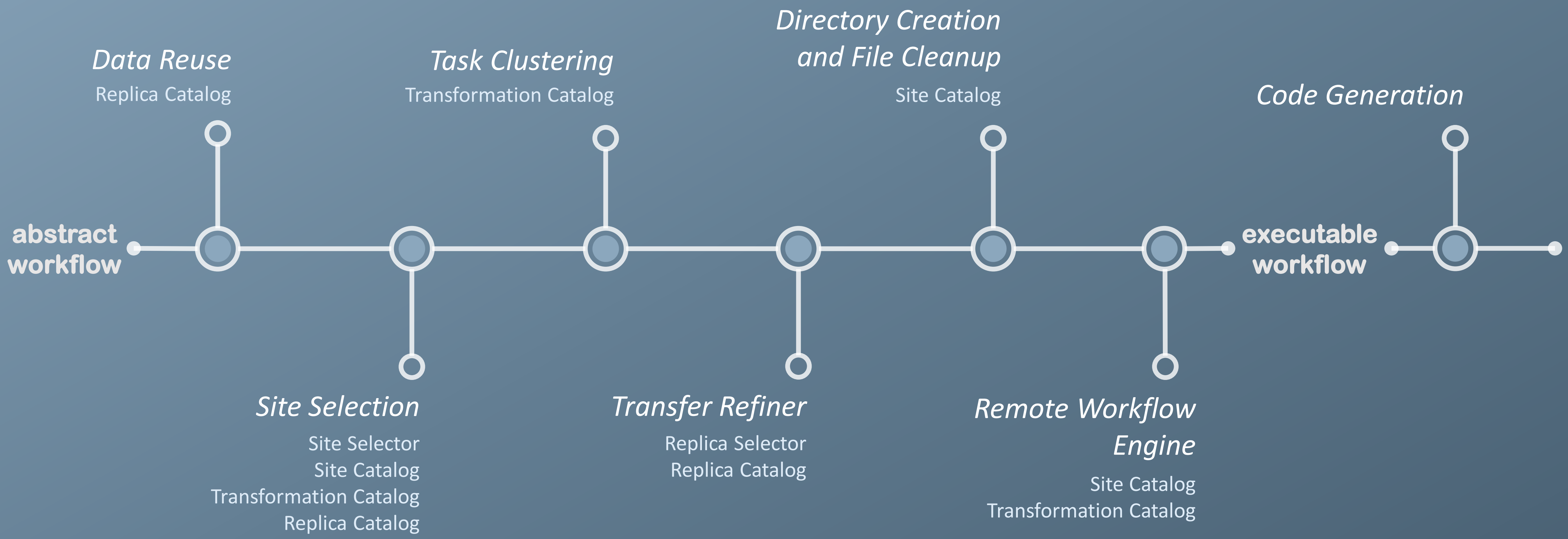
hierarchical workflows

**pegasus-mpi-cluster**



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

# Pegasus’ flow at a glance



**abstract workflow**

*Data Reuse*

Replica Catalog

*Site Selection*

Site Selector
Site Catalog
Transformation Catalog
Replica Catalog

*Task Clustering*

Transformation Catalog

*Transfer Refiner*

Replica Selector
Replica Catalog

*Directory Creation and File Cleanup*

Site Catalog

*Remote Workflow Engine*

Site Catalog
Transformation Catalog

**executable workflow**

*Code Generation*

# Applications…

# Galactic Plane - Montage

Multi-wavelength image atlas of the Galactic Plane, with coverage of 360° along the galactic plane and ±20° on either side

16 different wavelengths from 1 to 24 μm

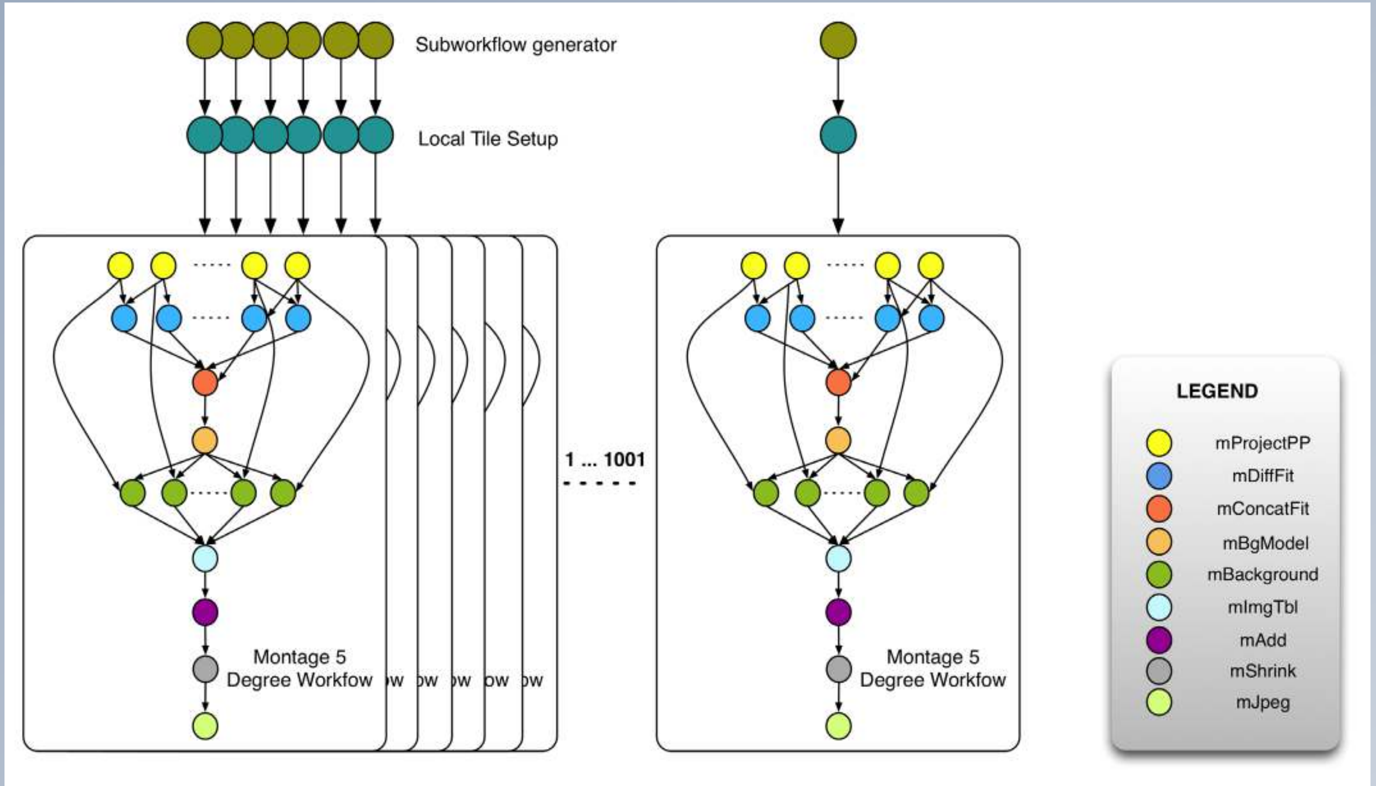
Each output image is 5° by 5° in size, and have an overlap of 1° with neighboring tiles

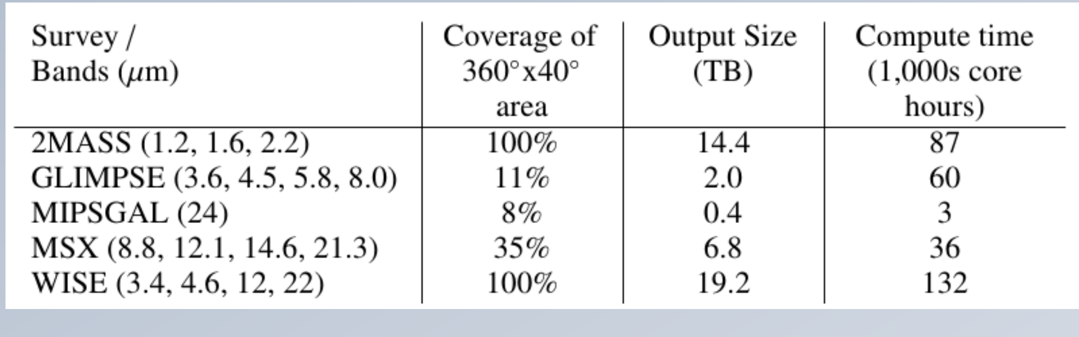
Processed so that they appear to have been measured with a single instrument observing all 16 wavelengths - Cartesian projection

18 million input images (~2.5TB)

16 workflows, each of which contains 1,001 sub-workflows (hierarchical workflows)

10.5 million tasks





| Survey / Bands ($\mu$m) | Coverage of 360°x40° area | Output Size (TB) | Compute time (1,000s core hours) |
|---|---|---|---|
| 2MASS (1.2, 1.6, 2.2) | 100% | 14.4 | 87 |
| GLIMPSE (3.6, 4.5, 5.8, 8.0) | 11% | 2.0 | 60 |
| MIPSGAL (24) | 8% | 0.4 | 3 |
| MSX (8.8, 12.1, 14.6, 21.3) | 35% | 6.8 | 36 |
| WISE (3.4, 4.6, 12, 22) | 100% | 19.2 | 132 |

# Galactic Plane - Montage

Amazon Web Services contributed the computations and storage

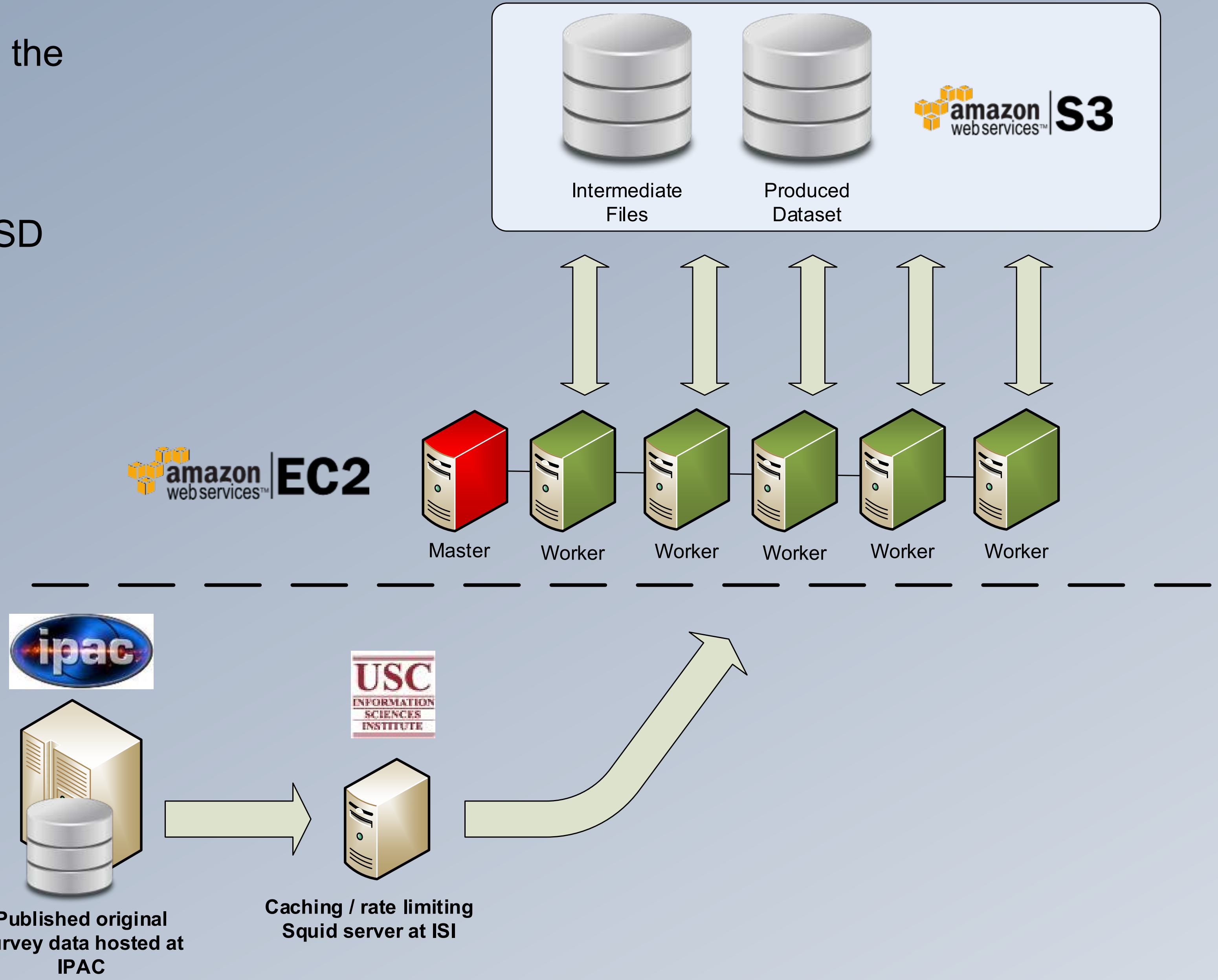
hi1.4xlarge instance

Memory optimized, with 2 x SSD ephemeral drives

318,000 core hours

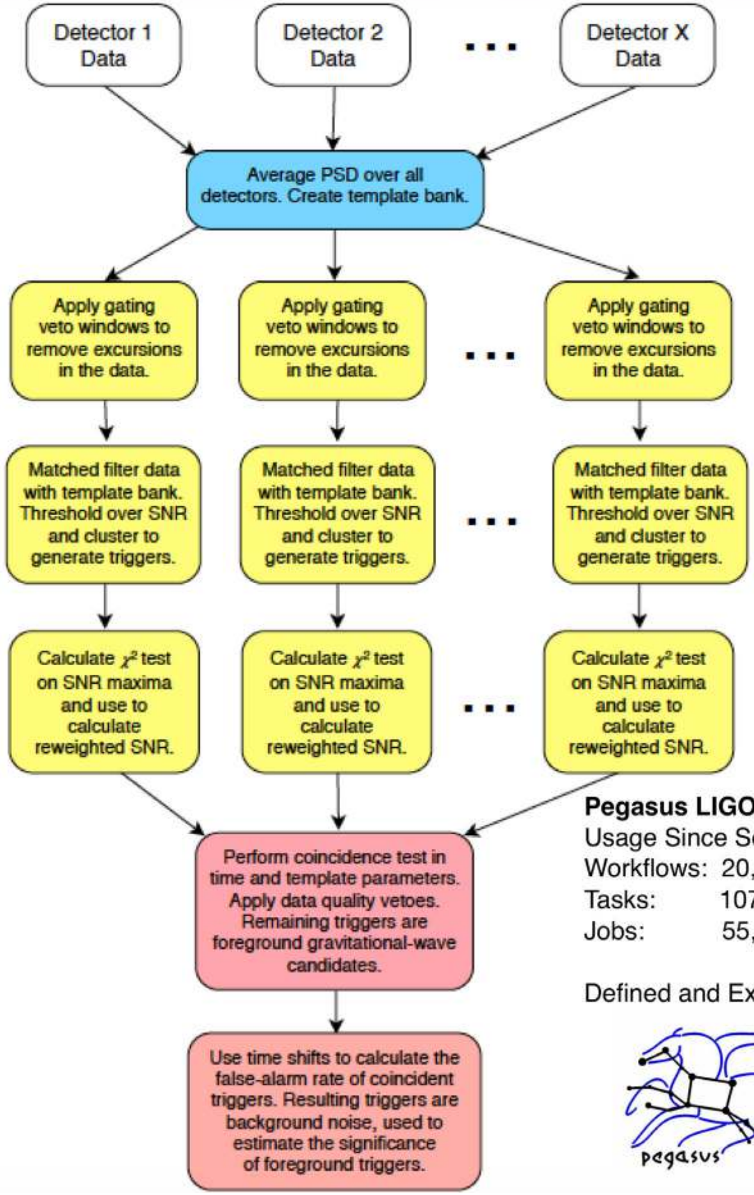
Spot instance price: $5,950

Note: this is from 2013!


Intermediate Files
Produced Dataset
amazon web services S3
amazon web services EC2
Master
Worker
Worker
Worker
Worker
Worker
ipac
USC INFORMATION SCIENCES INSTITUTE
Published original survey data hosted at IPAC
Caching / rate limiting Squid server at ISI

LIGO Laser Interferometer Gravitational-Wave Observatory
Supported by the National Science Foundation
Operated by Caltech and MIT

# Advanced LIGO – Laser Interferometer Gravitational Wave Observatory



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 $\chi^2$ 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.

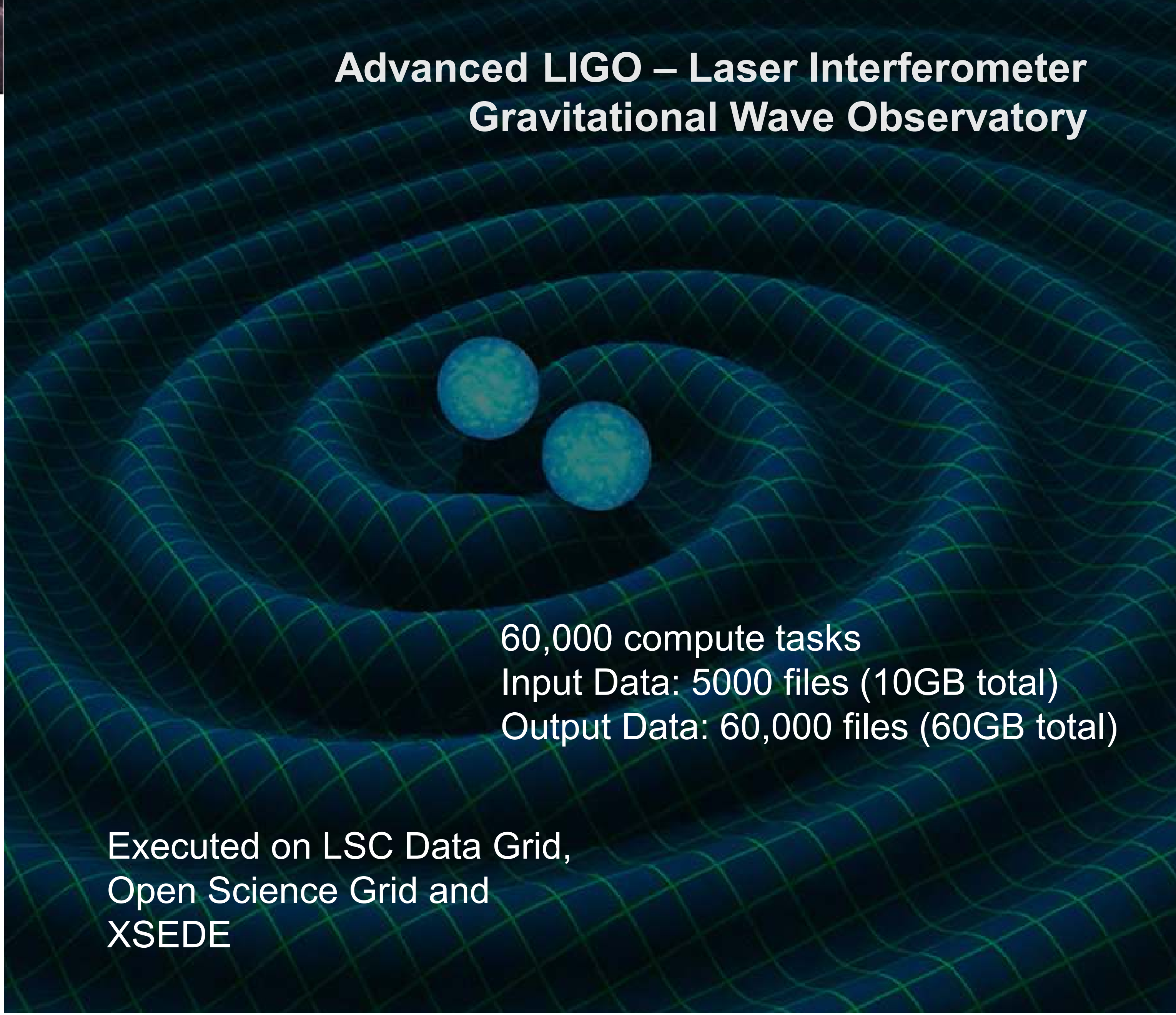
**Pegasus LIGO PyCBC Workflow**

Usage Since Sept 2015

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

Defined and Executed by Pegasus

pegasus



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

Executed on LSC Data Grid, Open Science Grid and XSEDE

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

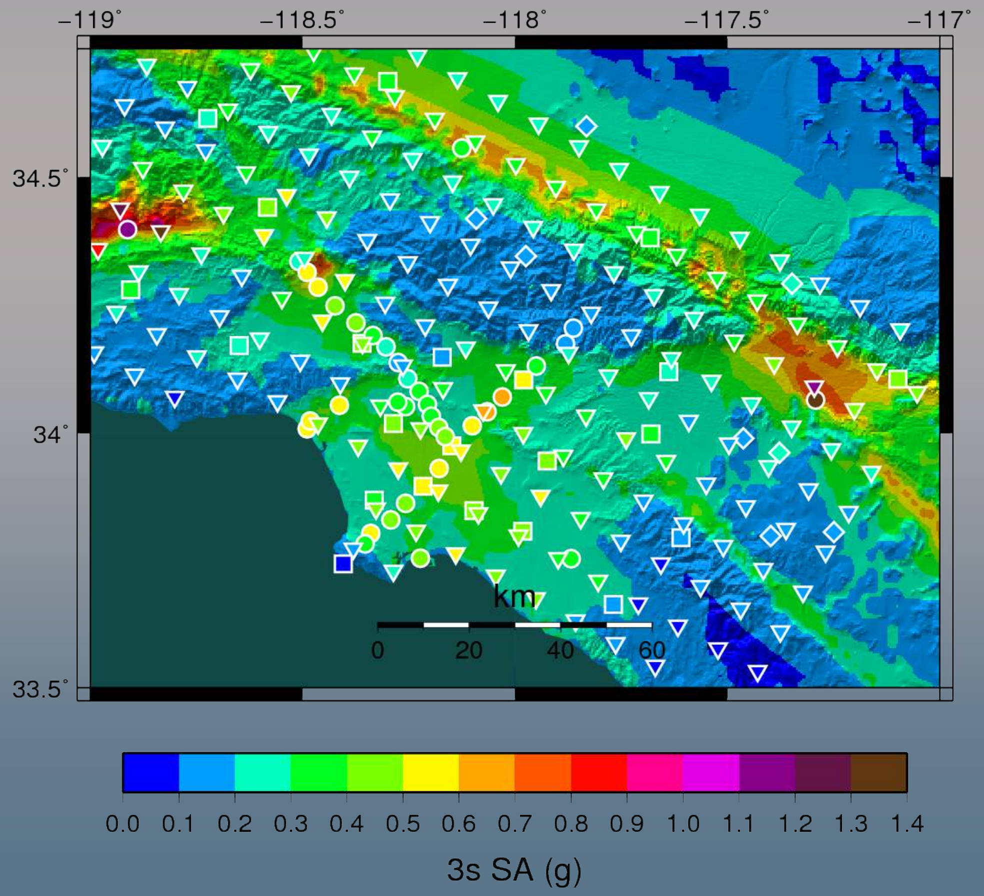
**PyCBC Detection GW150914:** First results from the search for binary black hole coalescence with Advanced LIGO. *B. P. Abbott et al.*

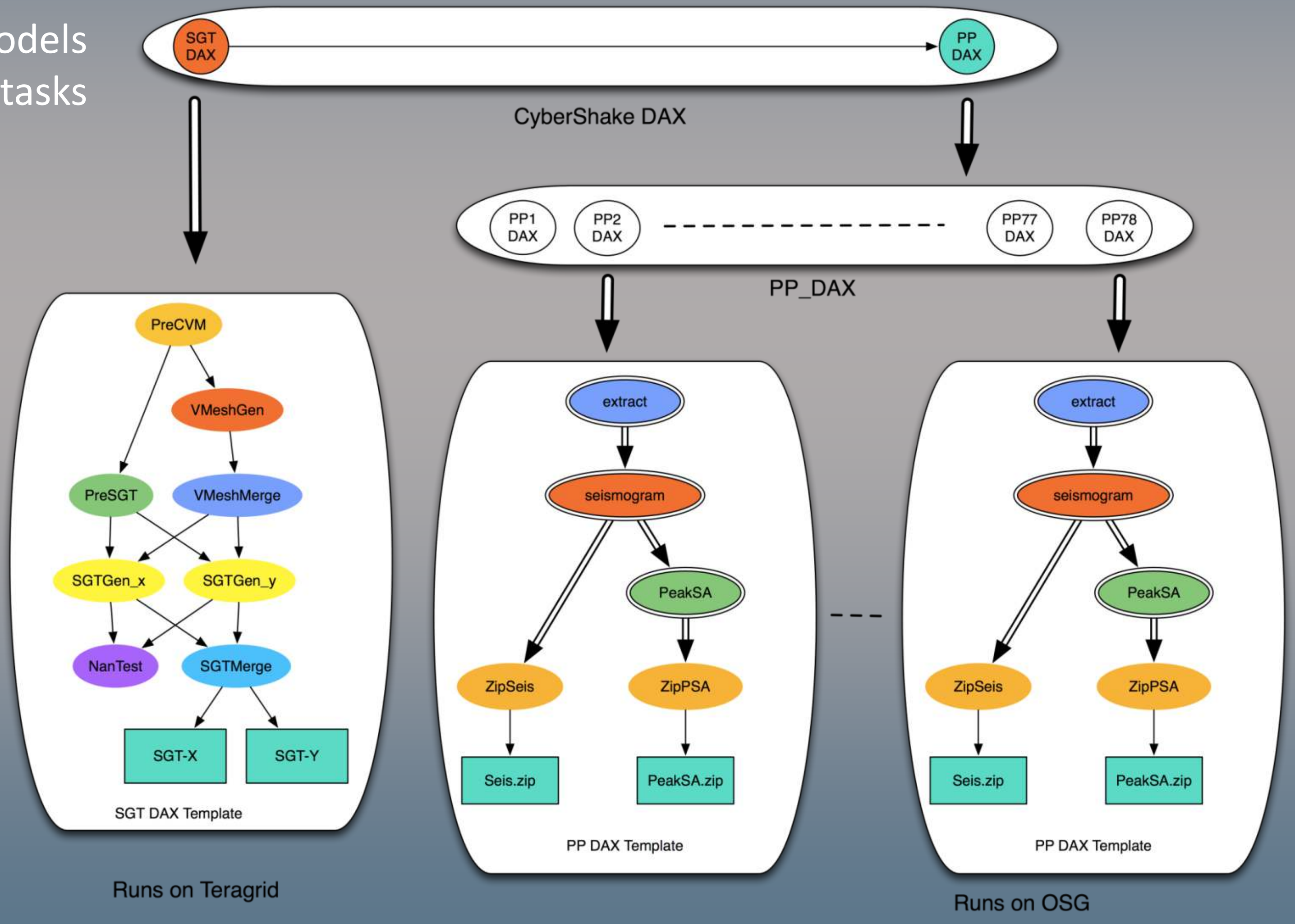
# Southern California Earthquake Center's CyberShake

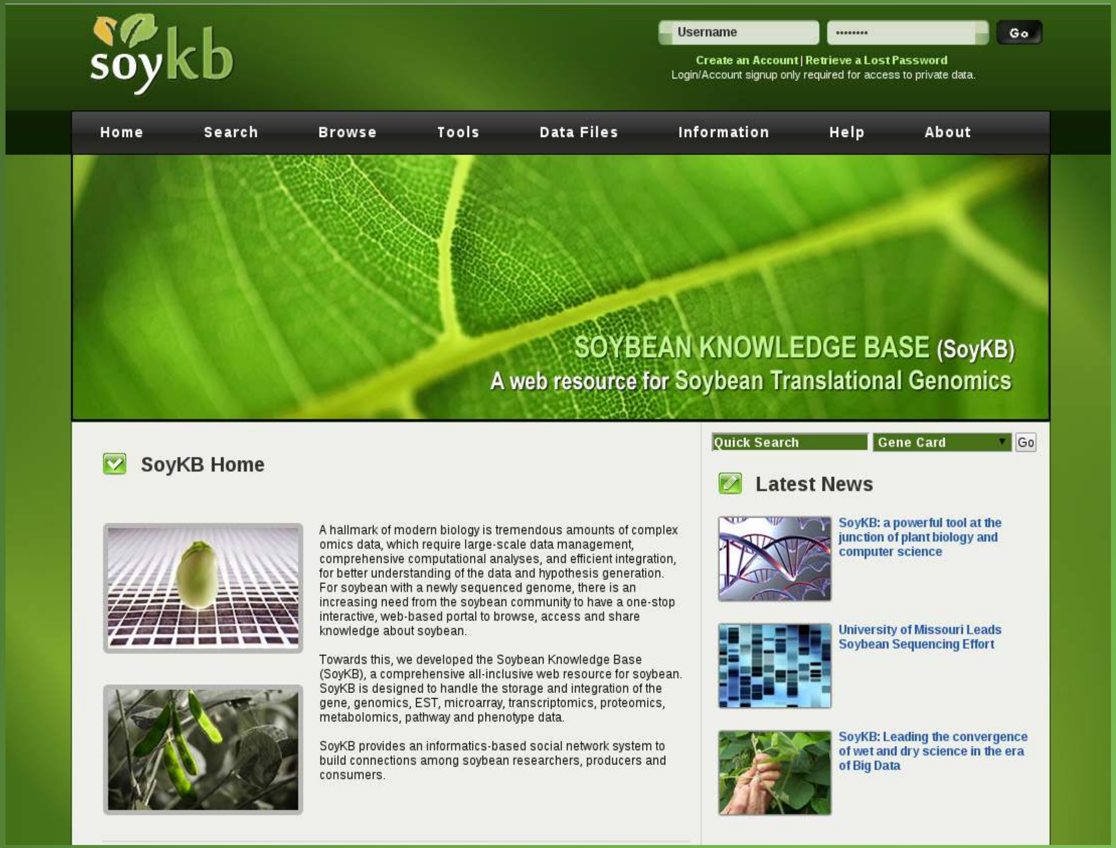
Builders ask seismologists: "What will the peak ground motion be at my new building in the next 50 years?"

Seismologists answer this question using Probabilistic Seismic Hazard Analysis (PSHA)

286 sites, 4 models
each workflow has 420,000 tasks







Login/Account signup only required for access to private data.

SOYBEAN KNOWLEDGE BASE (SoyKB)
A web resource for Soybean Translational Genomics

SoyKB Home

A hallmark of modern biology is tremendous amounts of complex omics data, which require large-scale data management, comprehensive computational analyses, and efficient integration, for better understanding of the data and hypothesis generation. For soybean with a newly sequenced genome, there is an increasing need from the soybean community to have a one-stop interactive, web-based portal to browse, access and share knowledge about soybean.

Towards this, we developed the Soybean Knowledge Base (SoyKB), a comprehensive all-inclusive web resource for soybean. SoyKB is designed to handle the storage and integration of the gene, genomics, EST, microarray, transcriptomics, proteomics, metabolomics, pathway and phenotype data.

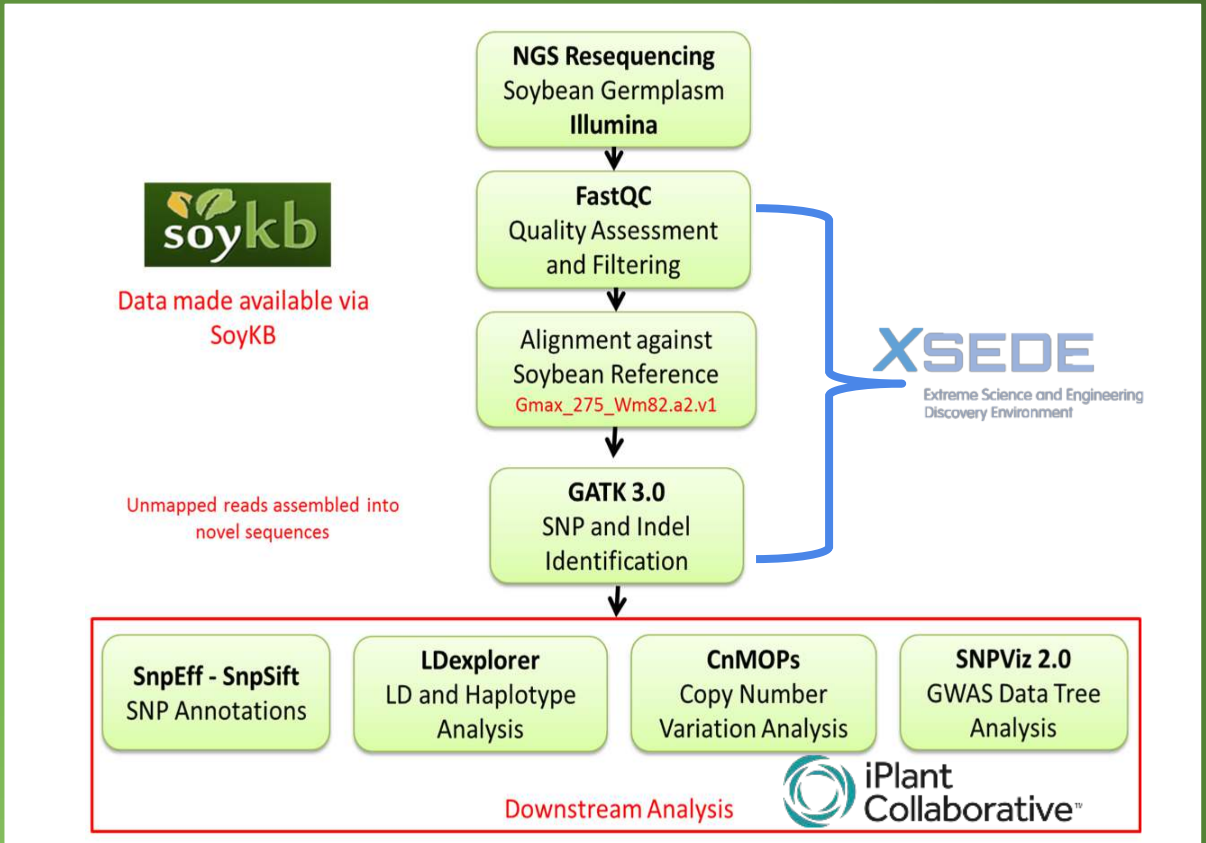
SoyKB provides an informatics-based social network system to build connections among soybean researchers, producers and consumers.

Latest News

SoyKB: a powerful tool at the junction of plant biology and computer science

University of Missouri Leads Soybean Sequencing Effort

SoyKB: Leading the convergence of wet and dry science in the era of Big Data



Data made available via SoyKB

Unmapped reads assembled into novel sequences

Downstream Analysis

# http://soykb.org

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

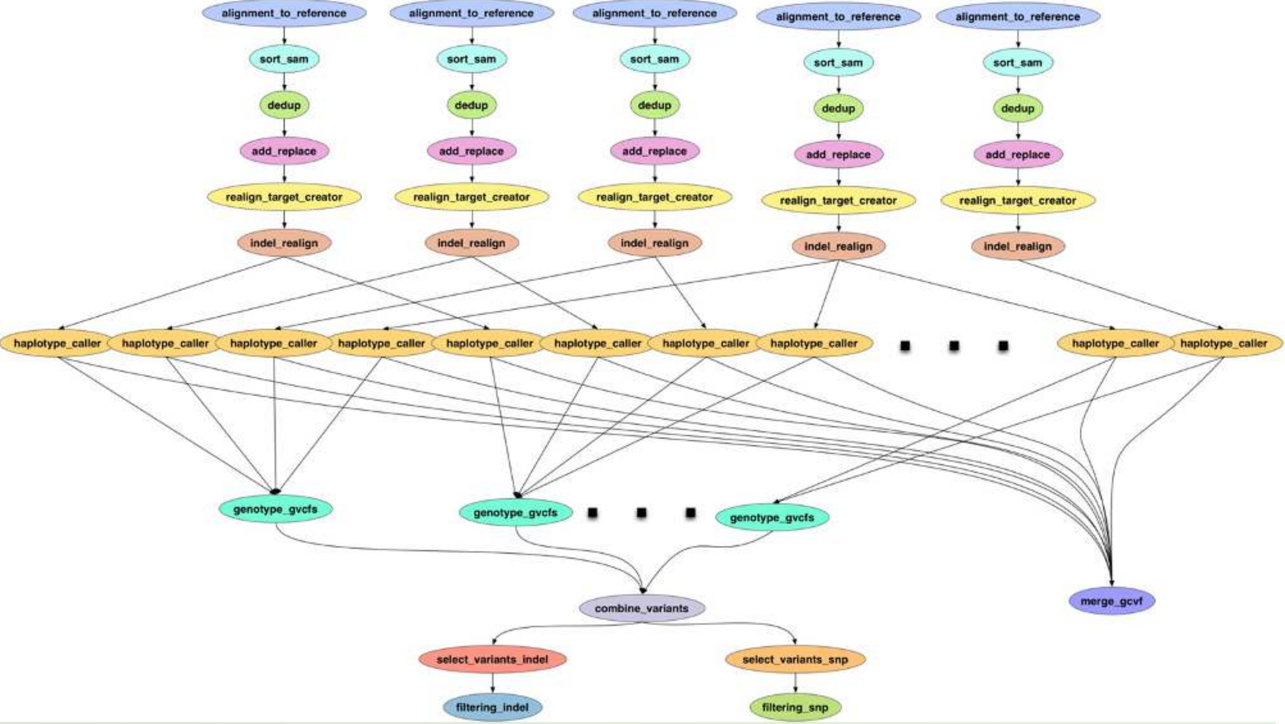
https://github.com/pegasus-isi/Soybean-Workflow

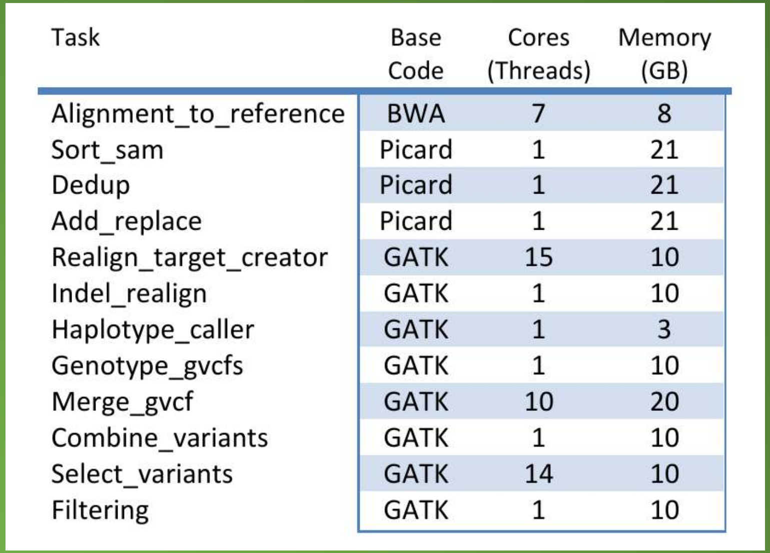
# TACC Wrangler as Execution Environment

Flash Based Shared Storage

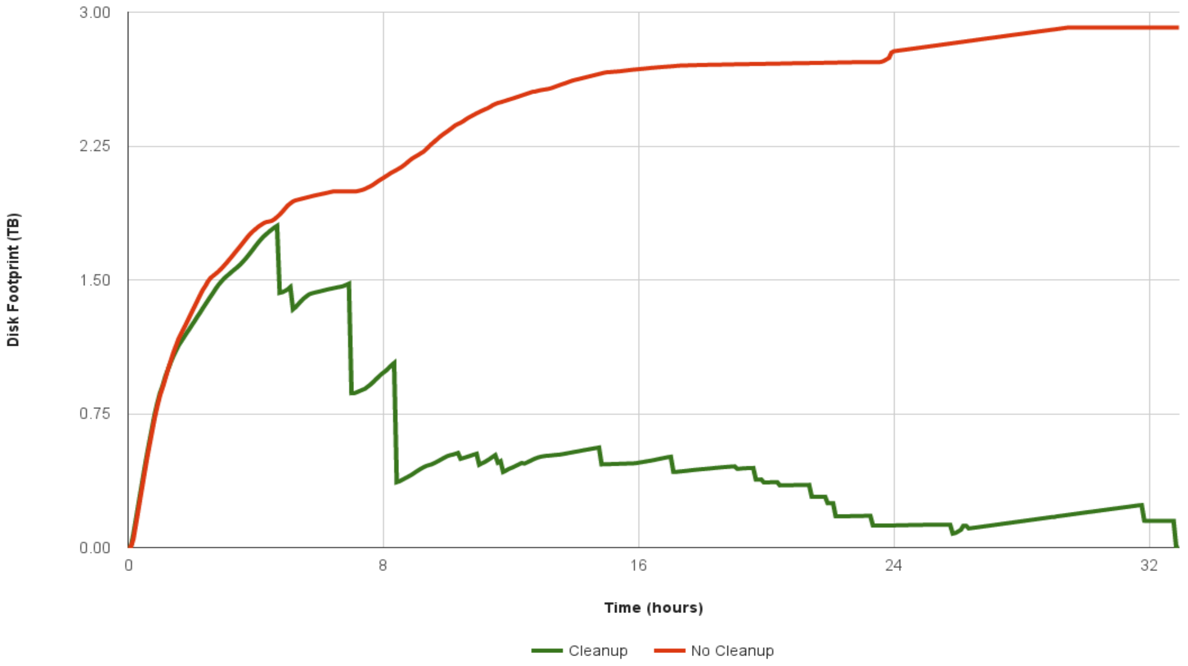
Switched to glideins (pilot jobs) - Brings in remote compute nodes and joins them to the HTCondor pool on in the submit host - Workflow runs at a finer granularity

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





| 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 |



# Pegasus est. 2001

Automate, recover, and debug scientific computations.

## Get Started

**Pegasus Website**

http://pegasus.isi.edu

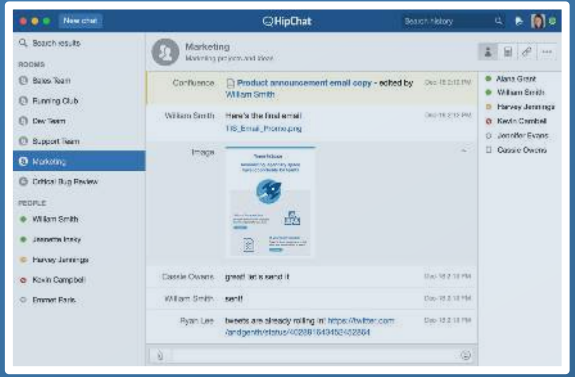
**Users Mailing List**

pegasus-users@isi.edu

**Support**

pegasus-support@isi.edu

**HipChat**



Marketing
Marketing projects and ideas

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

# Extra…

# How does Pegasus decide where to execute?

*site description*
describes the compute resources

*scratch*
tells where temporary data is stored

*storage*
tells where output data is stored

*profiles*
key-pair values associated per job level

```
...
  <!-- The local site contains information about the submit host -->
    <!-- The arch and os keywords are used to match binaries in the transformation
catalog -->
    <site handle="local" arch="x86_64" os="LINUX">

      <!-- These are the paths on the submit host were Pegasus stores data -->
      <!-- Scratch is where temporary files go -->
      <directory type="shared-scratch" path="/home/tutorial/run">
        <file-server operation="all" url="file:///home/tutorial/run"/>
      </directory>

      <!-- Storage is where pegasus stores output files -->
      <directory type="local-storage" path="/home/tutorial/outputs">
        <file-server operation="all" url="file:///home/tutorial/outputs"/>
      </directory>

      <!-- This profile tells Pegasus where to find the user's private key for SCP
transfers -->
      <profile namespace="env" key="SSH_PRIVATE_KEY">/home/tutorial/.ssh/id_rsa</profile>

    </site>
...
```

# How does it know where the executables are or which ones to use?

*executables description*

list of executables locations per site

*physical executables*

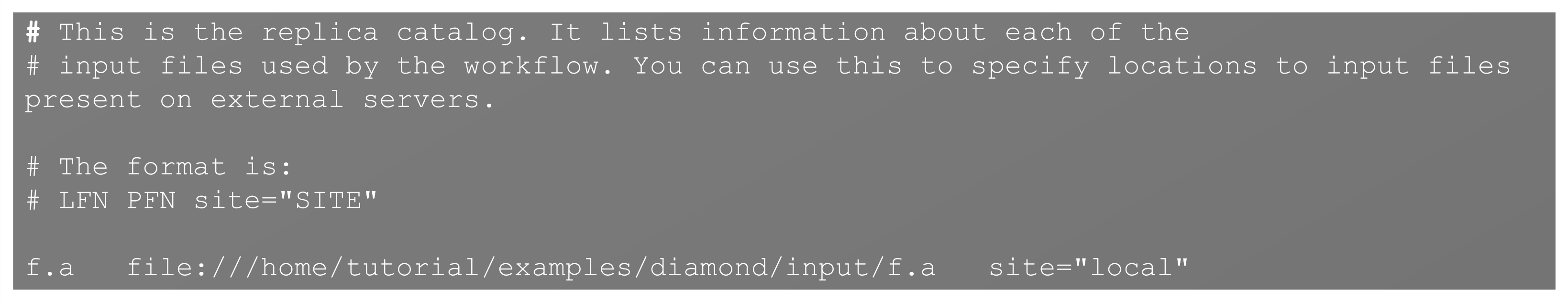
mapped from logical transformations

*transformation type*

whether it is installed or available to stage

```
...
# This is the transformation catalog. It lists information about each of the
# executables that are used by the workflow.

tr ls {
  site PegasusVM {
    pfn "/bin/ls"
    arch "x86_64"
    os "linux"
    type "INSTALLED”
  }
}
...
```

# What if data is not local to the submit host?



```
# This is the replica catalog. It lists information about each of the
# input files used by the workflow. You can use this to specify locations to input files
present on external servers.

# The format is:
# LFN PFN site="SITE"

f.a   file:///home/tutorial/examples/diamond/input/f.a   site="local"
```

*site name*

in which site the file is available

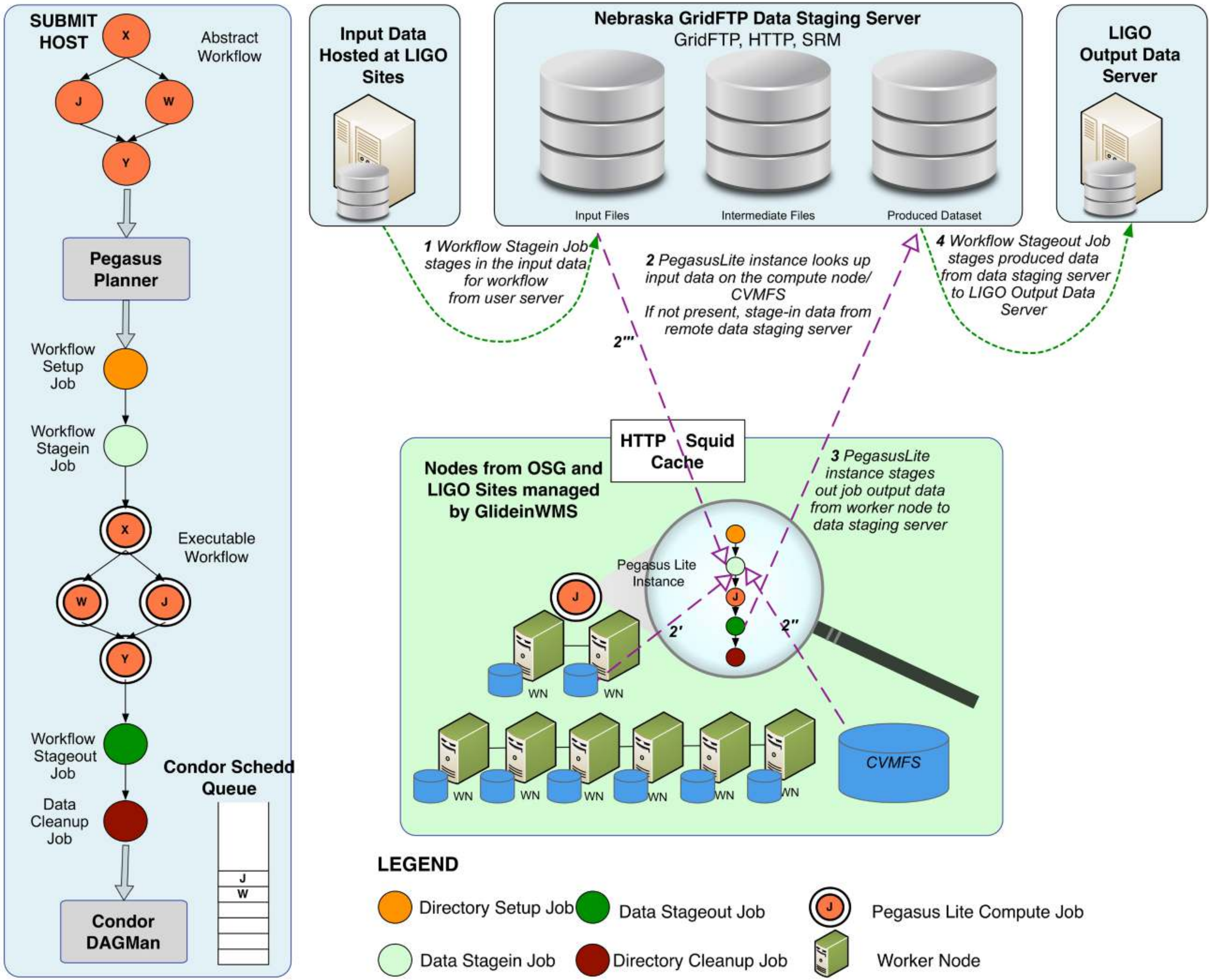
*logical filename*

abstract data name

*physical filename*

data physical location on site
different transfer protocols
can be used (e.g., scp, http,
ftp, gridFTP, etc.)

# Data Flow for LIGO Pegasus Workflows in OSG



SUBMIT HOST

Abstract Workflow

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*

*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*

HTTP Squid Cache

Nodes from OSG and LIGO Sites managed by GlideinWMS

Pegasus Lite Instance

CVMFS

LEGEND

- Directory Setup Job
- Data Stagein Job
- Data Stageout Job
- Directory Cleanup Job
- Pegasus Lite Compute Job
- Worker Node