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

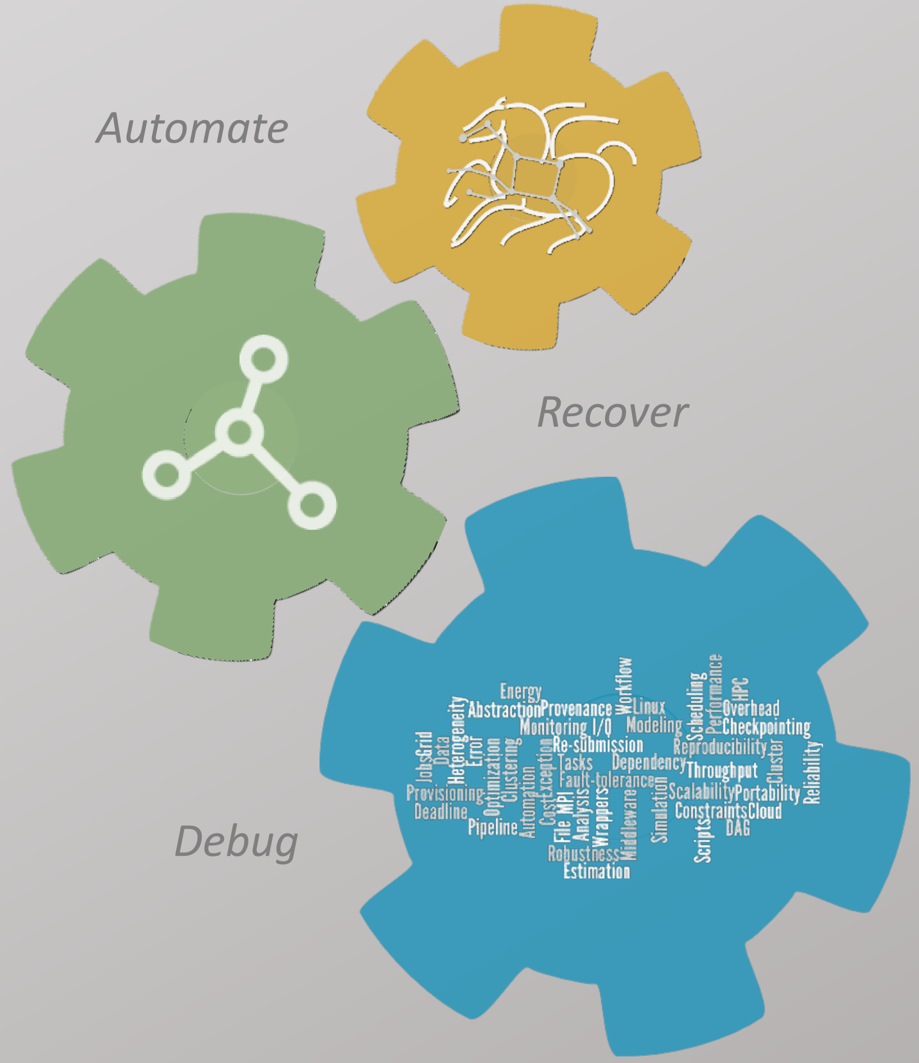
Automatically executes data transfers

Reusable, aids reproducibility

Records how data was produced (provenance)

Handles failures with to provide reliability

Keeps track of data and files

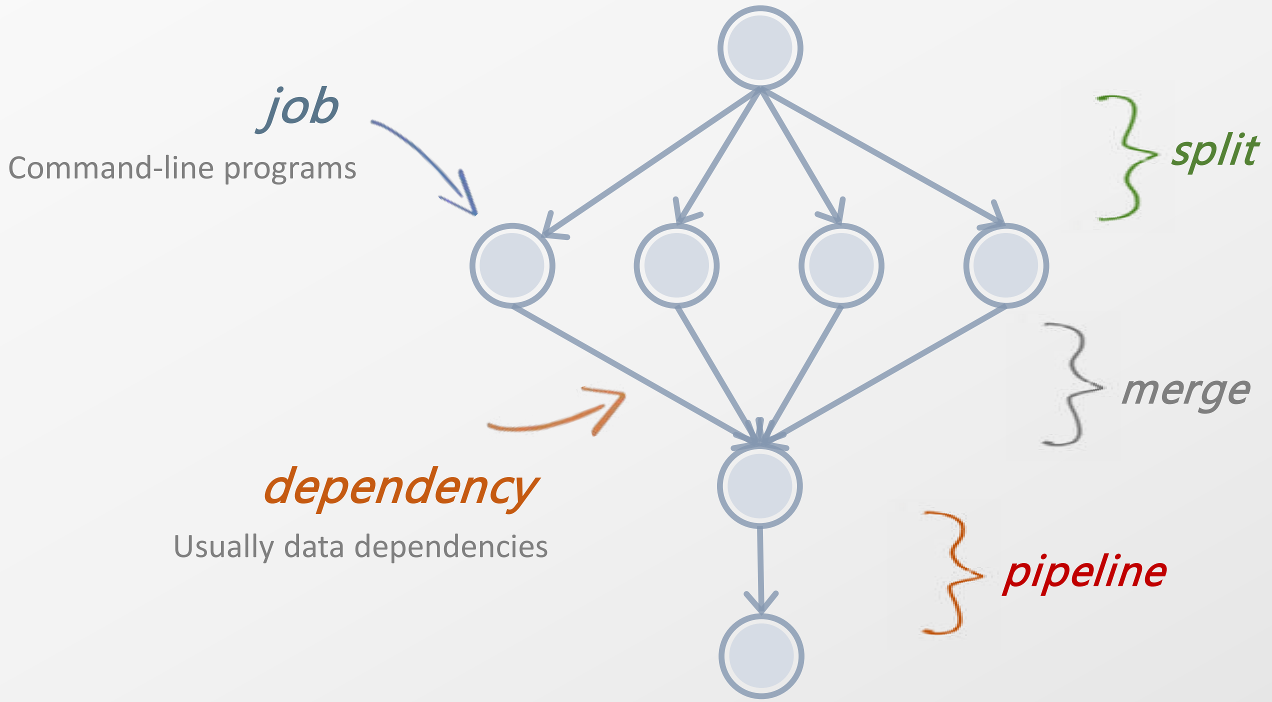


HTCondor
High Throughput Computing

Taking a closer look into a workflow…

DAG

directed-acyclic graphs



job

Command-line programs

dependency

Usually data dependencies

split

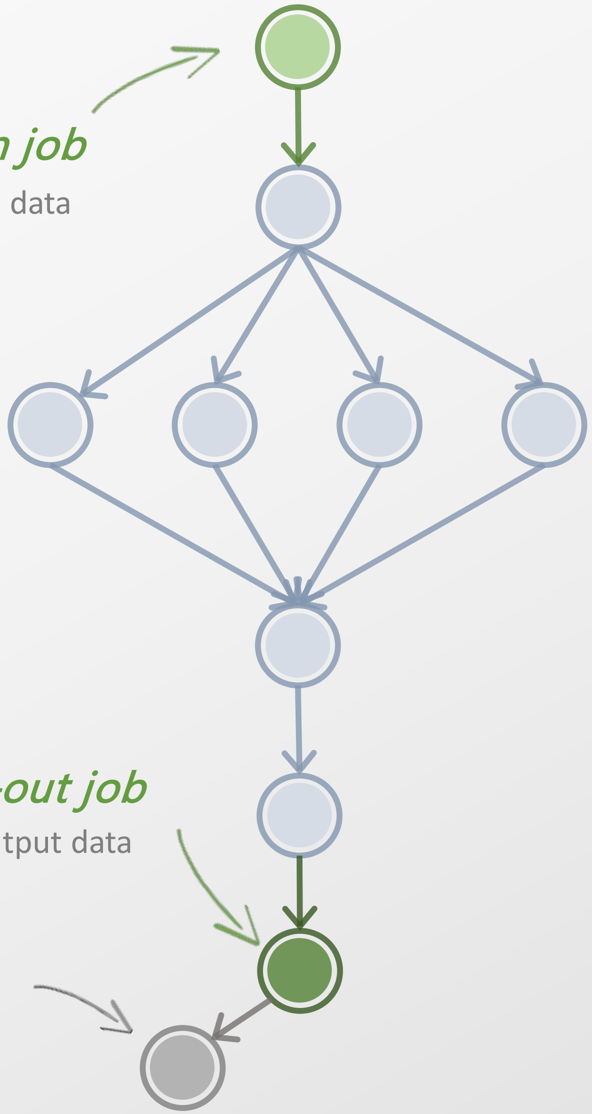
merge

pipeline

DAX

DAG in XML

From the abstraction to execution!



stage-in job

Transfers the workflow input data

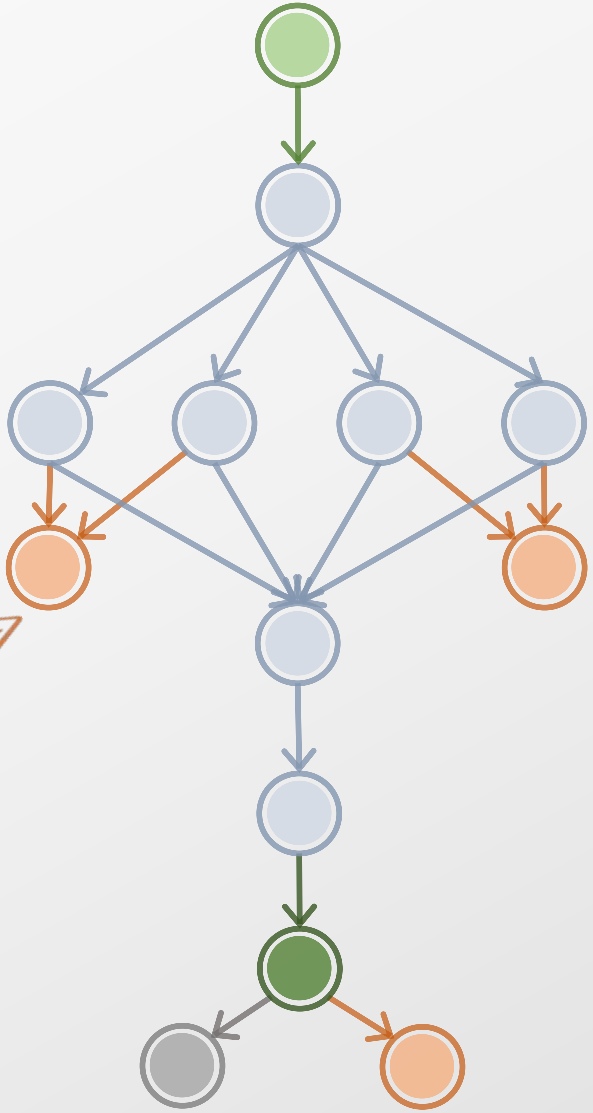
stage-out job

Transfers the workflow output data

registration job

Registers the workflow output data

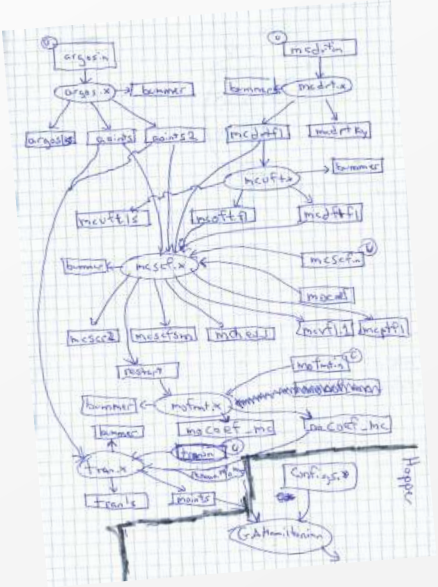
Optimizing storage usage…



cleanup job

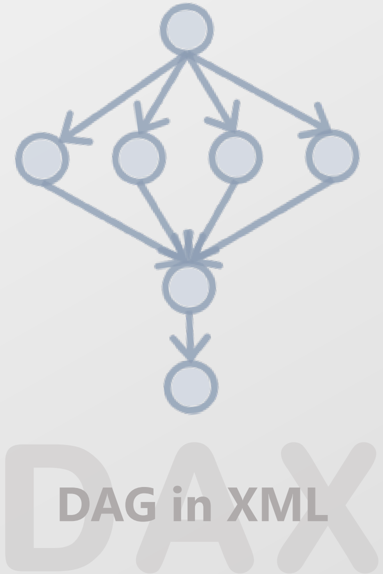
Removes unused data

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()
```



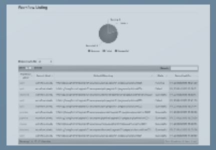
While you wait…

…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

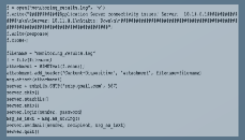
Workflow execution and job performance metrics

Debug

Set of debugging tools to unveil issues

RESTful API

Monitoring and reporting information on your own application interface



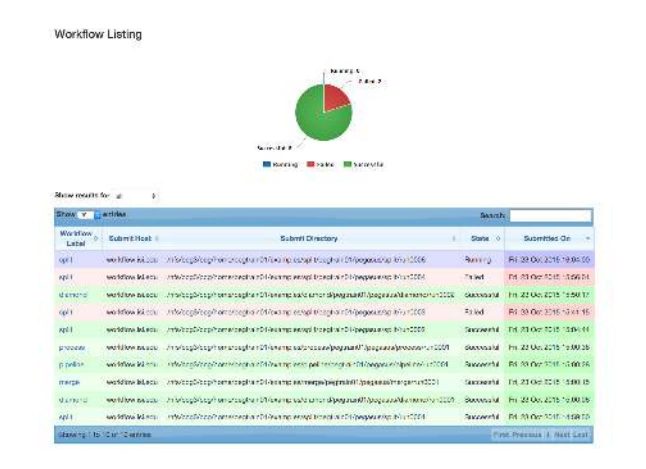
```
f = open("monitoring_results.log", "w")
f.write([illegible]
[illegible]
[illegible]
f.write(response)
f.close()

filename = "monitoring_results.log"
f = file(filename)
attachment = MIMEText(f.read())
attachment.add_header('Content-Disposition', 'attachment', filename=filename)
msg.attach(attachment)
server = smtplib.SMTP('smtp.gmail.com', 587)
server.ehlo()
server.starttls()
server.ehlo()
server.login(sender, password)
msg_as_text = msg.as_string()
server.sendmail(sender, recipient, msg_as_text)
server.quit()
```

Command-line tools

Tools for monitor and debug workflows

Pegasus dashboard

web interface for monitoring and debugging workflows



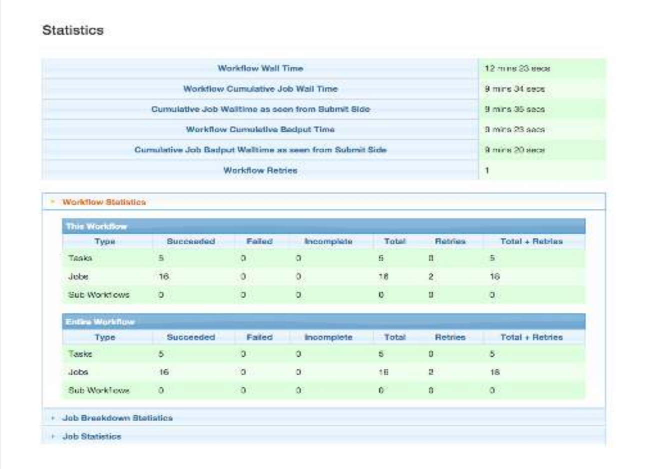
Workflow Listing

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 16:04:00
split	workflow.isi.edu	/nfs/ccg3/ccg/home/pegtrain01/examples/split/pegtrain01/pegasus/split/run0004	Failed	Fri, 23 Oct 2015 15:56:04
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/run0005	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 15:00:35
pipeline	workflow.isi.edu	/nfs/ccg3/ccg/home/pegtrain01/examples/pipeline/pegtrain01/pegasus/pipeline/run0001	Successful	Fri, 23 Oct 2015 15:00:26
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 15:00:05
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



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

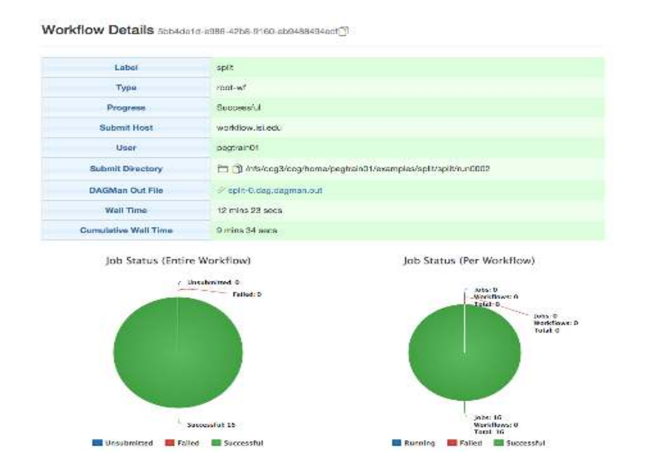
Entire Workflow

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 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 23 secs
Cumulative Wall Time	9 mins 34 secs

Real-time Monitoring

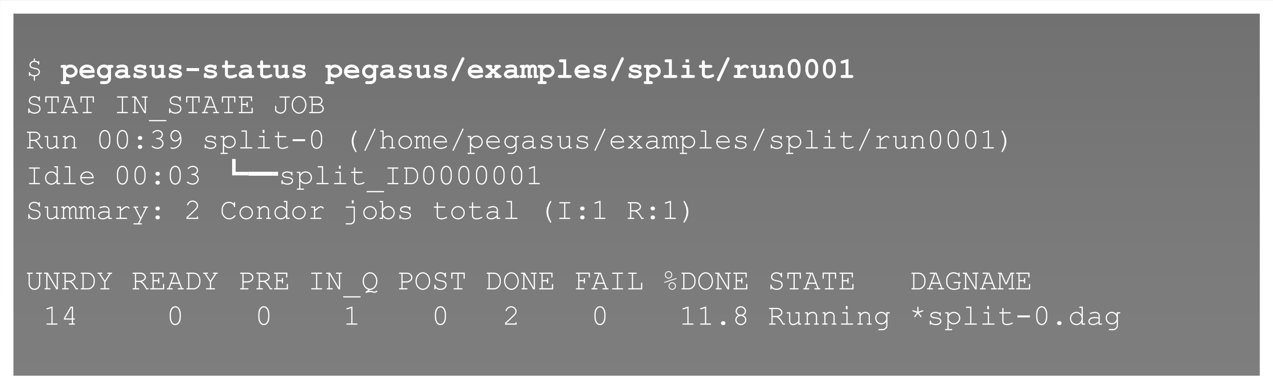
Reporting

Debugging

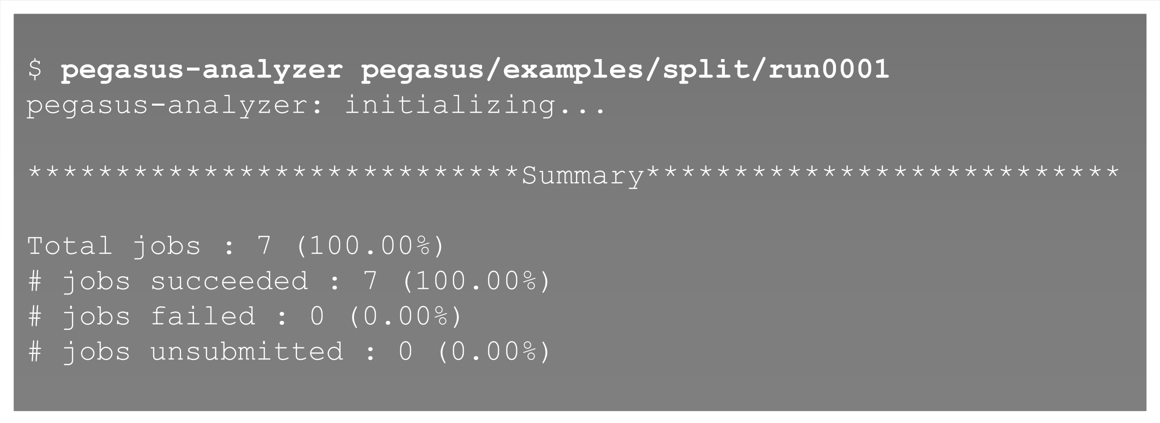
Troubleshooting

RESTful API

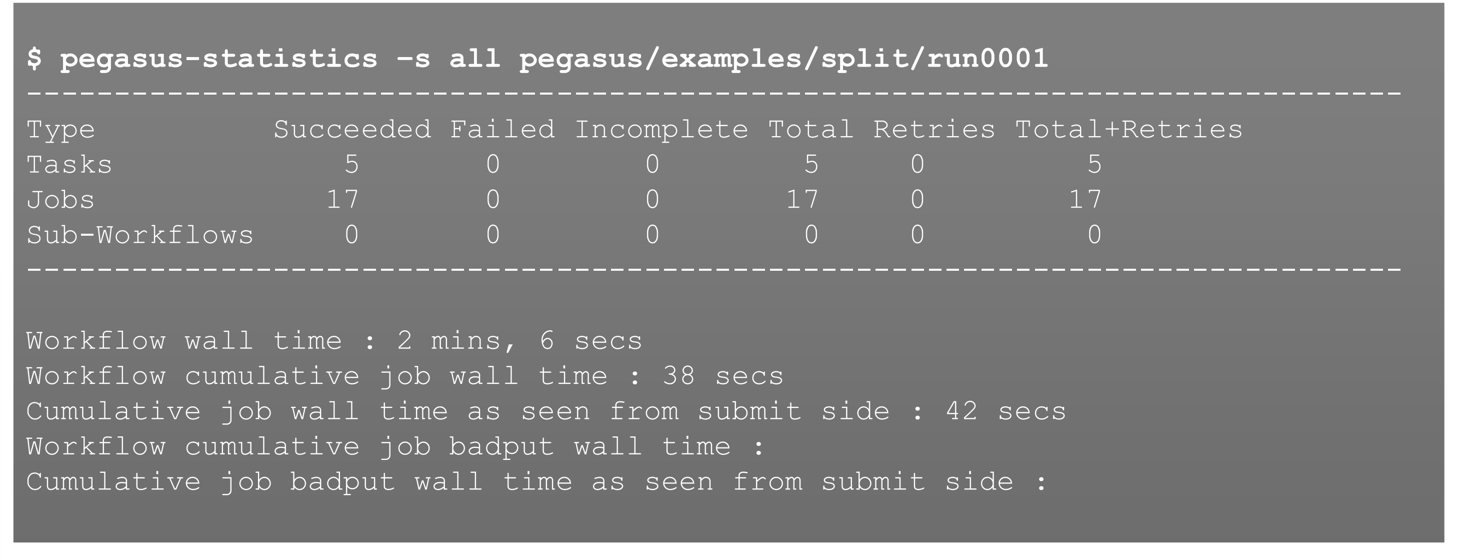
>_ But, if you prefer the command-line…



```
$ 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
```



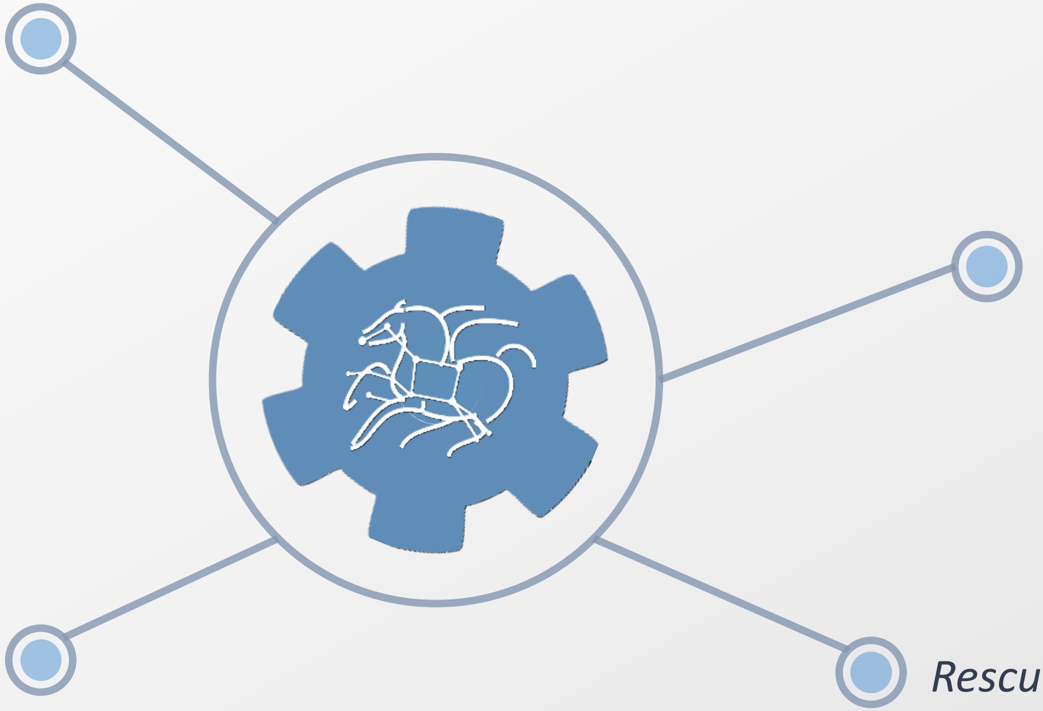
```
$ pegasus-analyzer pegasus/examples/split/run0001
pegasus-analyzer: initializing...

****************************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

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?

GridFTP

Let Pegasus manage it for you

ftp

Shared filesystem

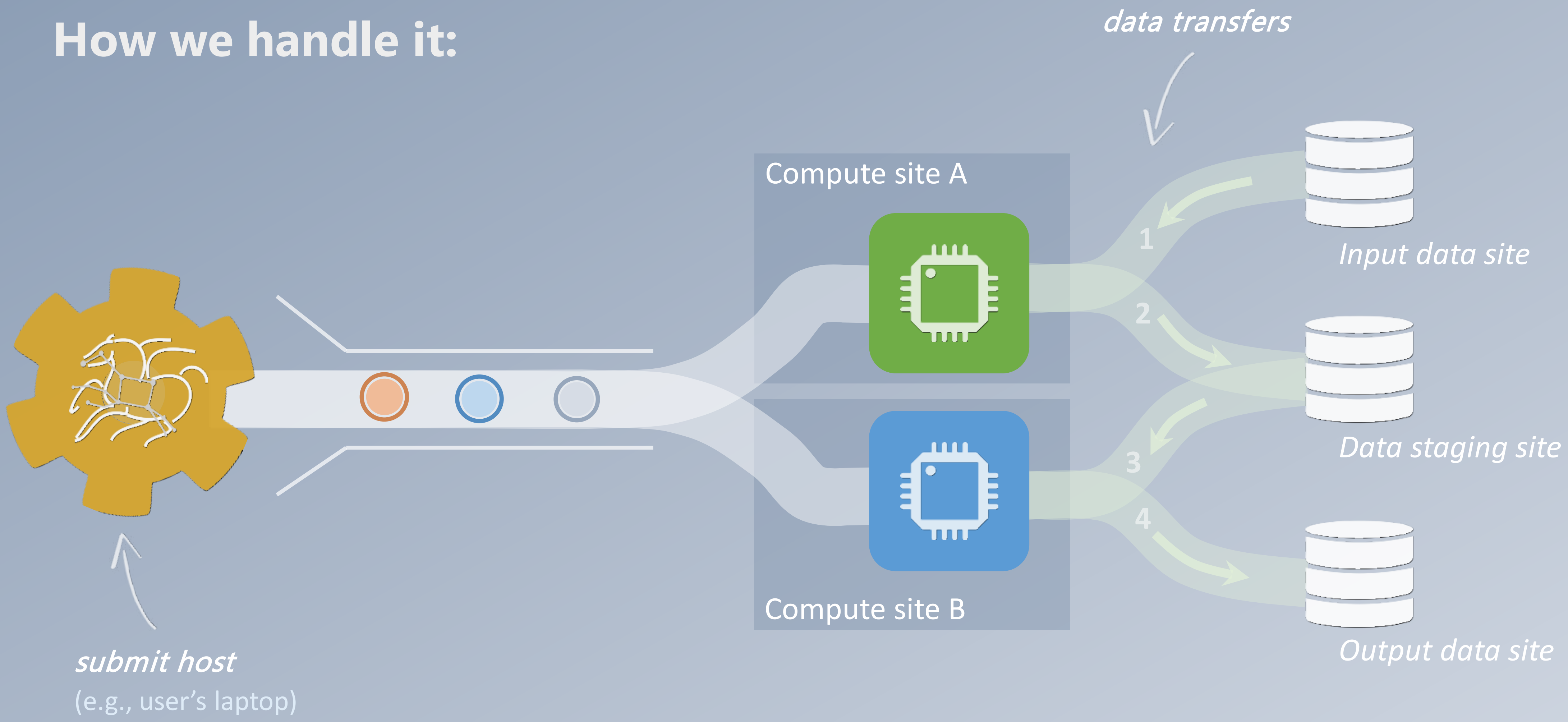
scp

Google Storage

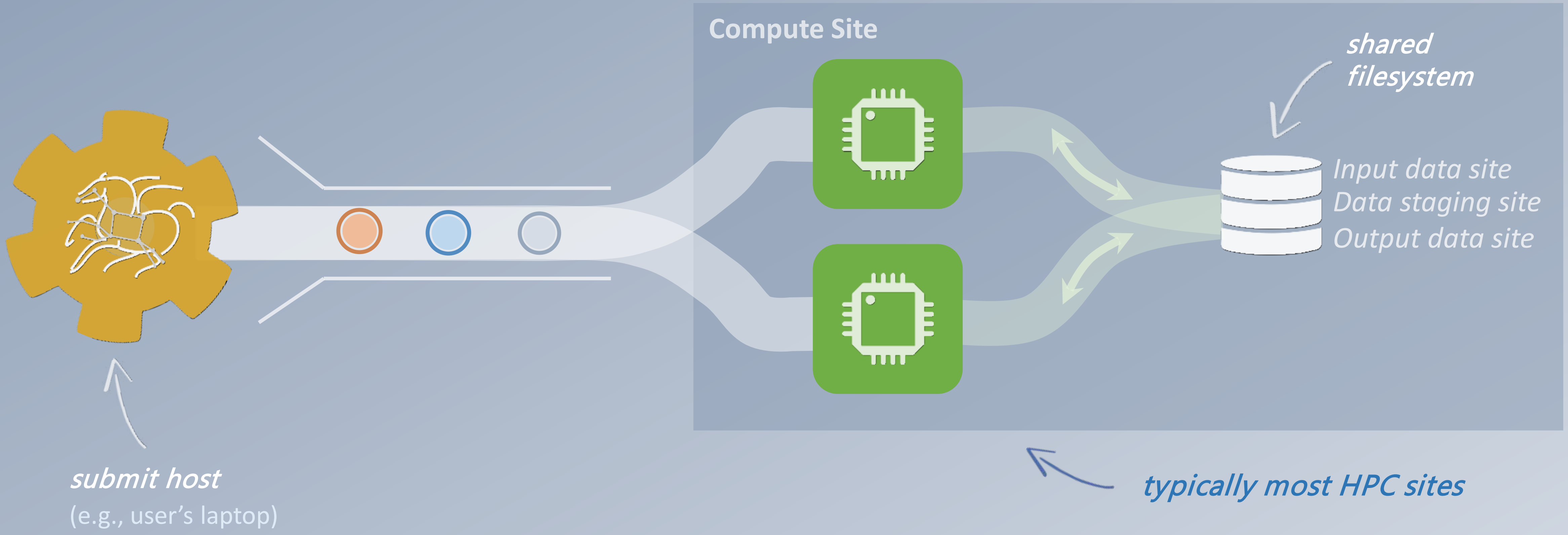
StashCache

iRODS

How we handle it:

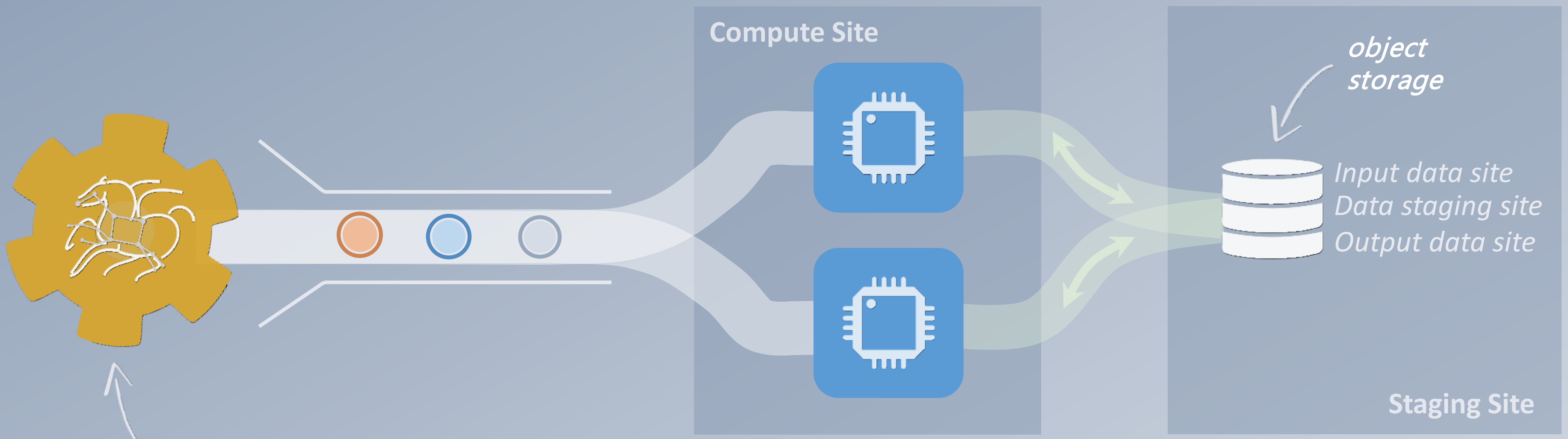


However, there are several possible configurations for data sites…



typically most HPC sites

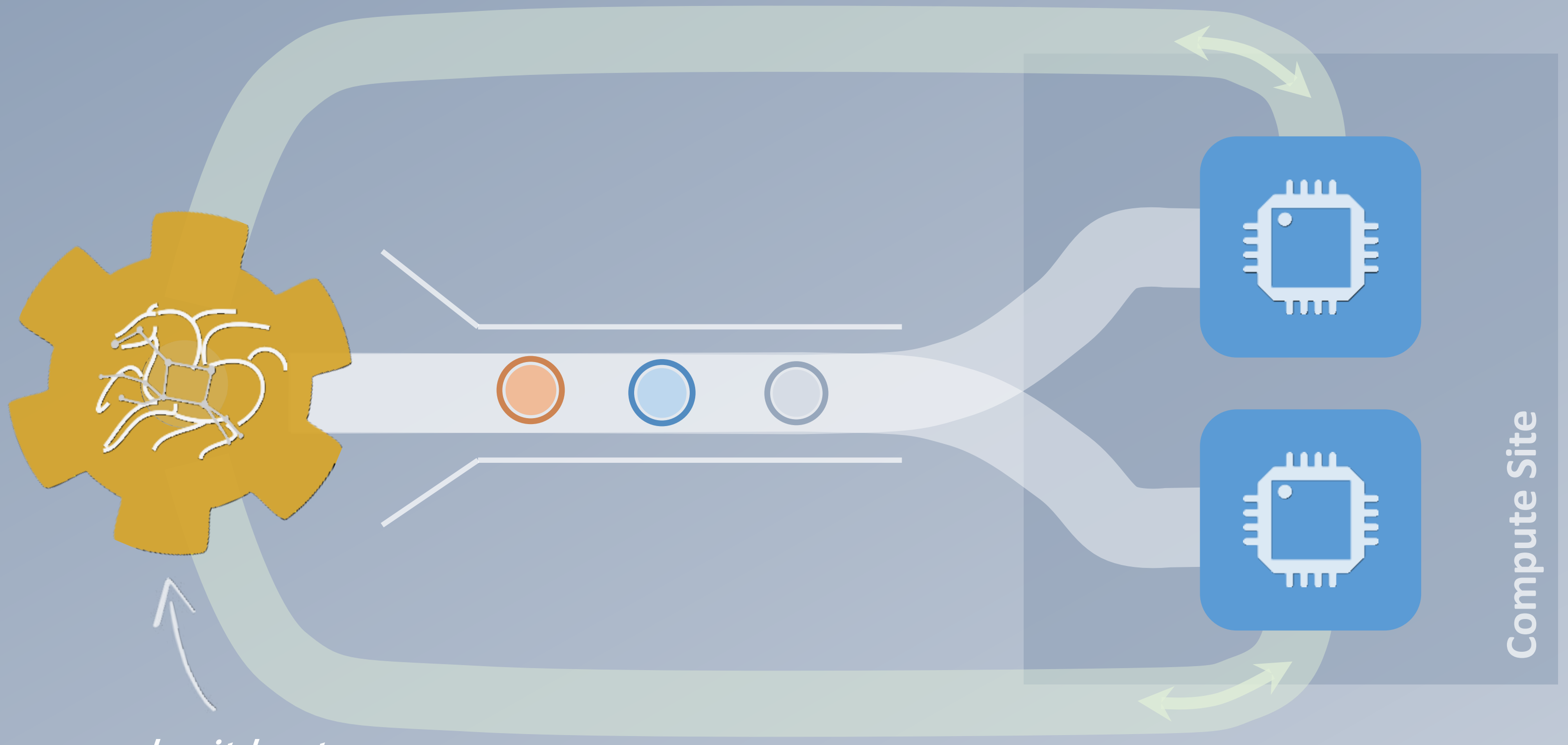
Pegasus also handles high-scalable object storages



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

Typical cloud computing deployment (Amazon S3, Google Storage)

Pegasus can also manage data over the submit host…



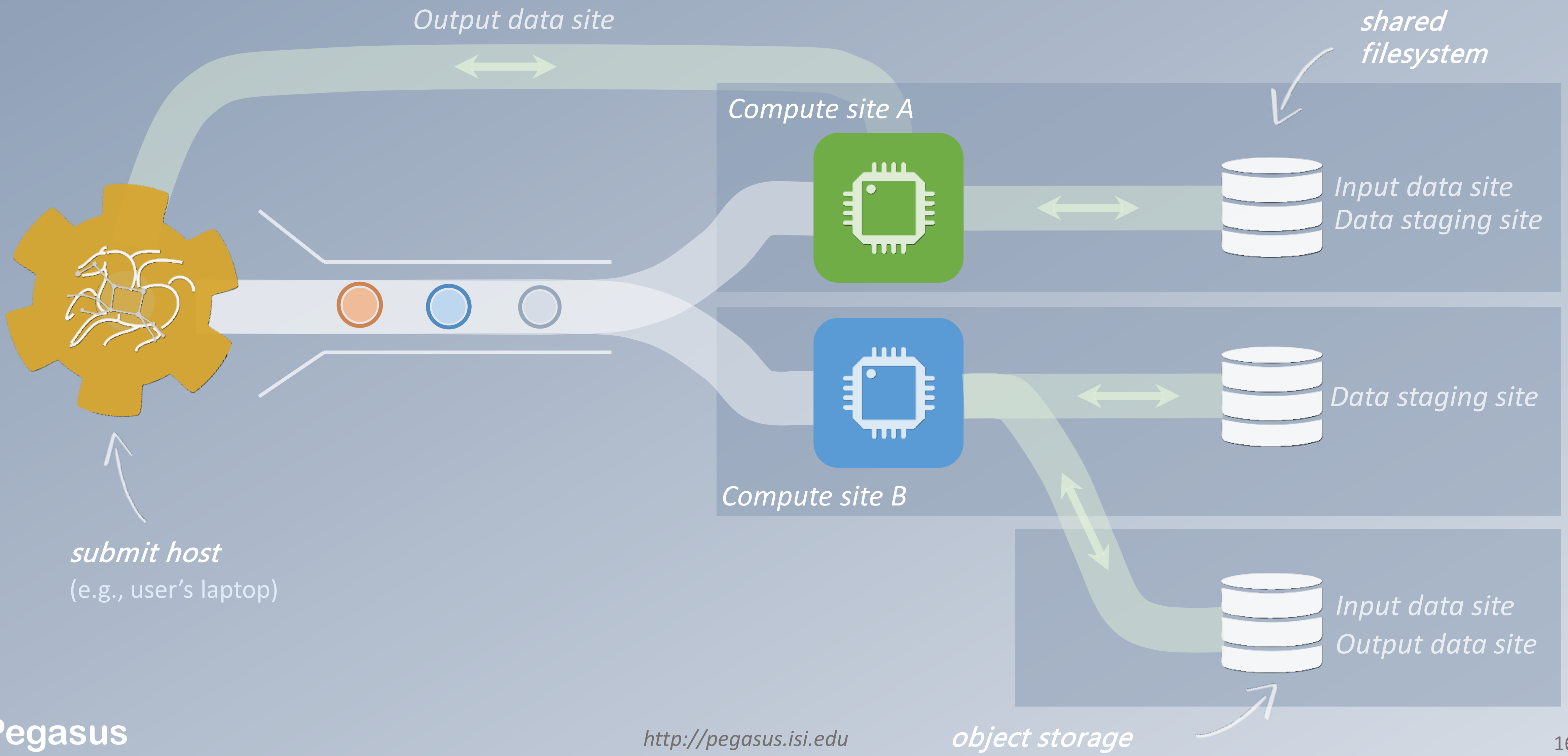
submit host

(e.g., user's laptop)

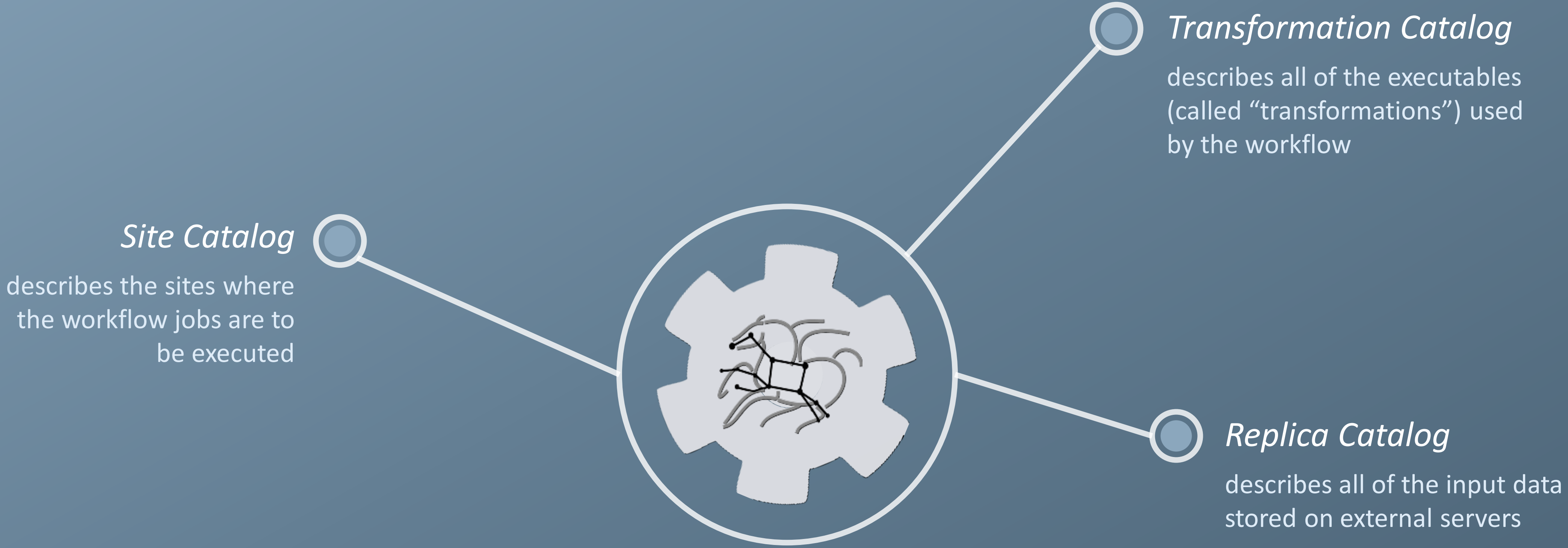
Typical OSG sites

Open Science Grid

And yes… you can mix everything!



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

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.)

A few more features…

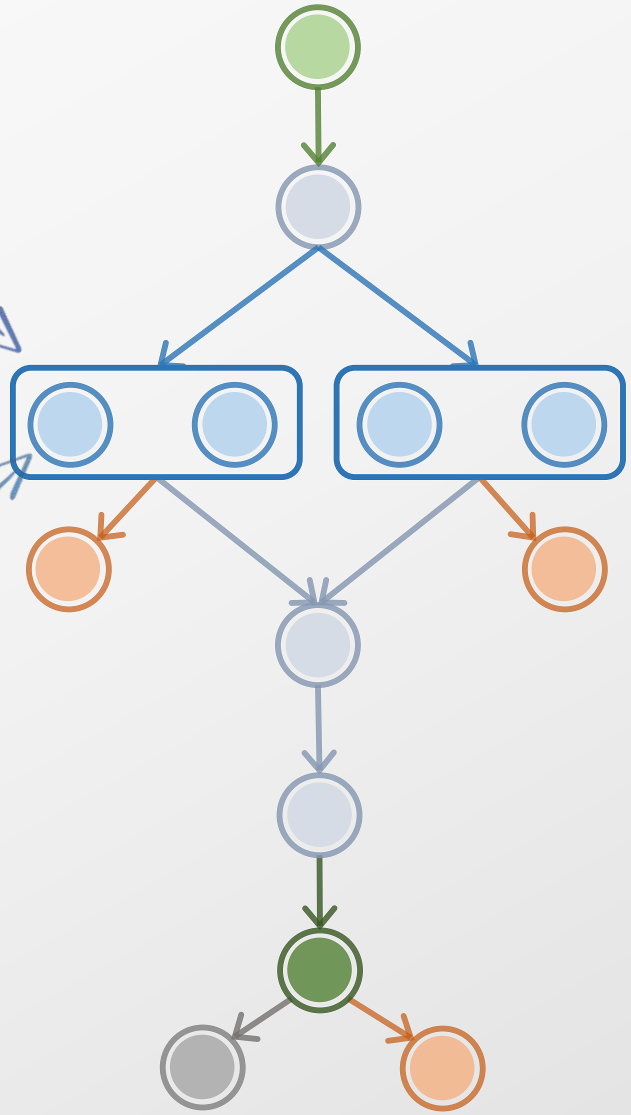
Performance, why not improve it?

workflow restructuring

workflow reduction

hierarchical workflows

pegasus-mpi-cluster



clustered job

Groups small jobs together to improve performance

task

small granularity

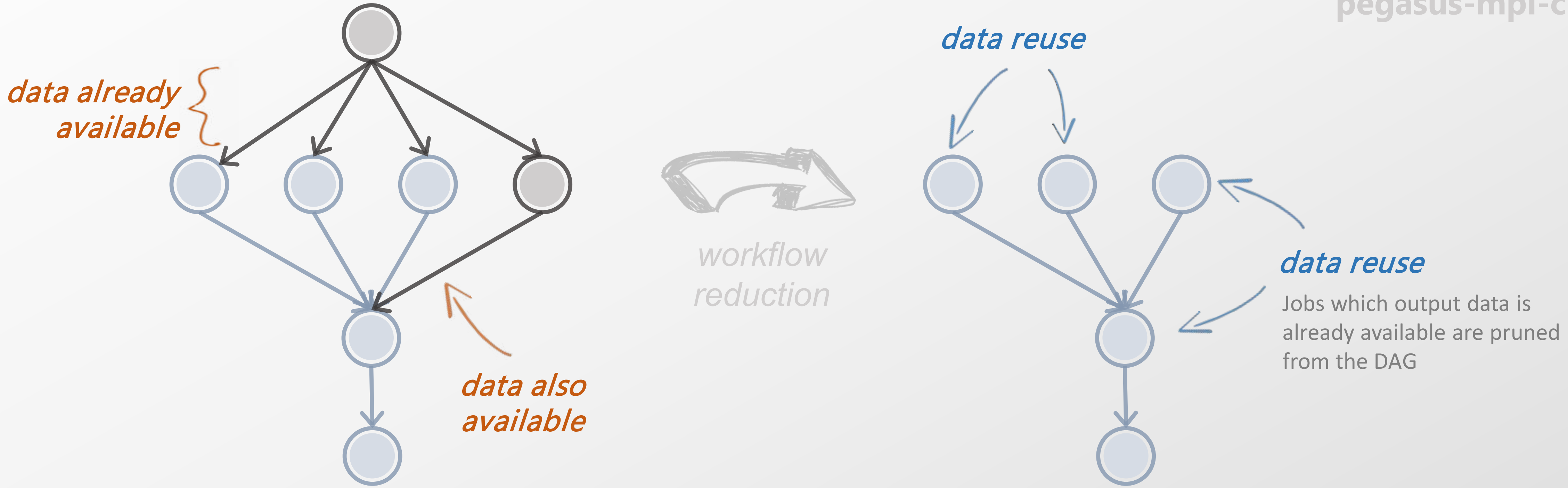
What about **data reuse**?

workflow restructuring

workflow reduction

hierarchical workflows

pegasus-mpi-cluster



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

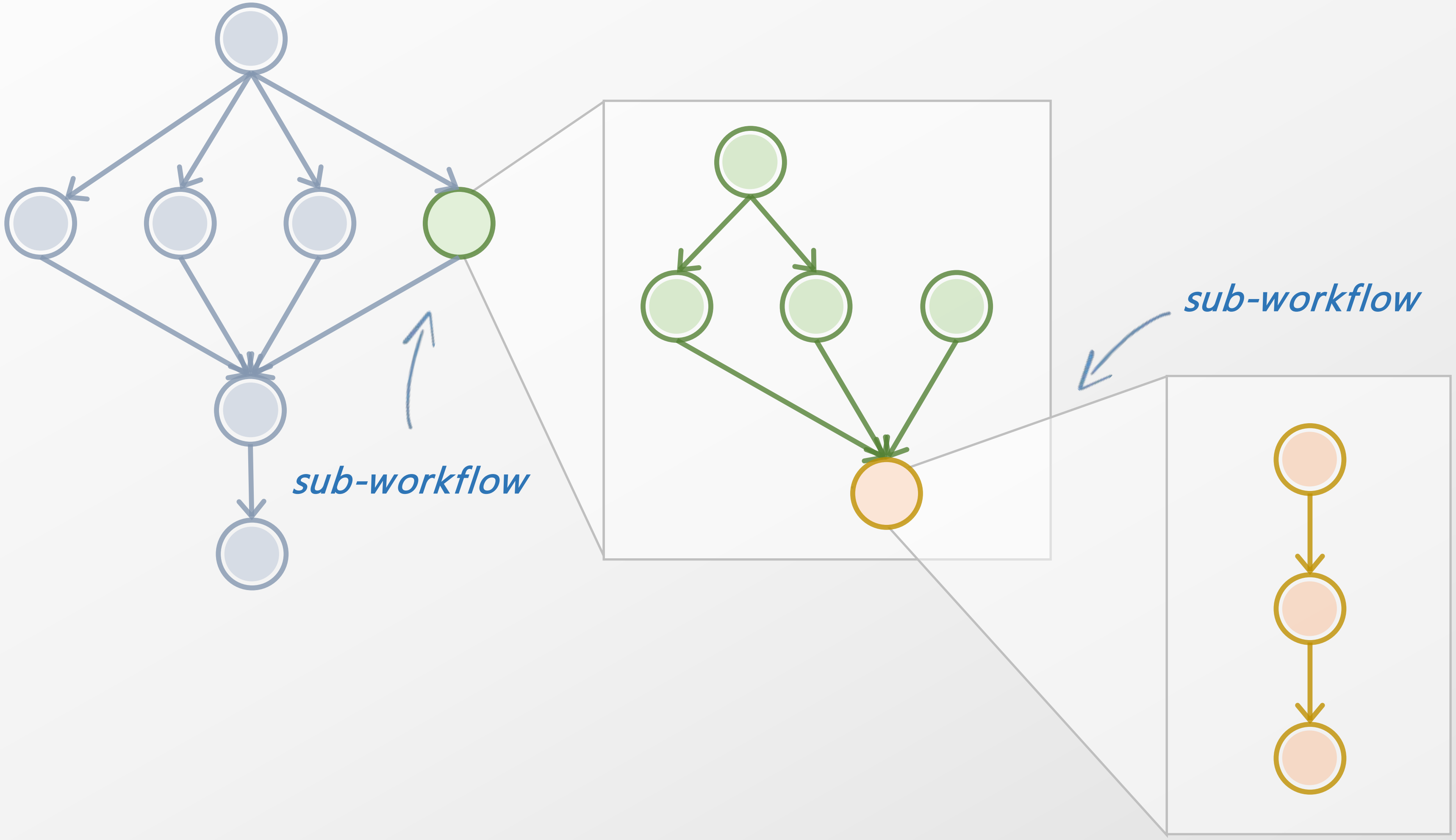
Pegasus also handles **large-scale workflows**

workflow restructuring

workflow reduction

hierarchical workflows

pegasus-mpi-cluster



recursion ends when DAX with only compute jobs is encountered

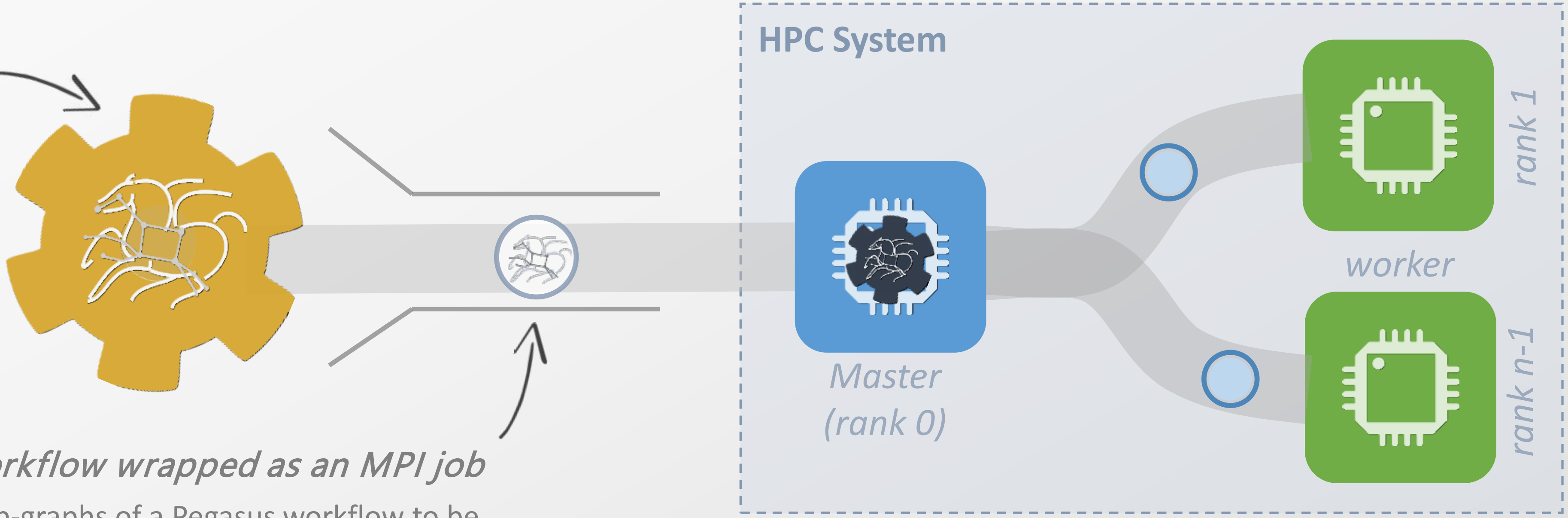
Running **fine-grained** workflows on HPC systems…

workflow restructuring

workflow reduction

hierarchical workflows

pegasus-mpi-cluster

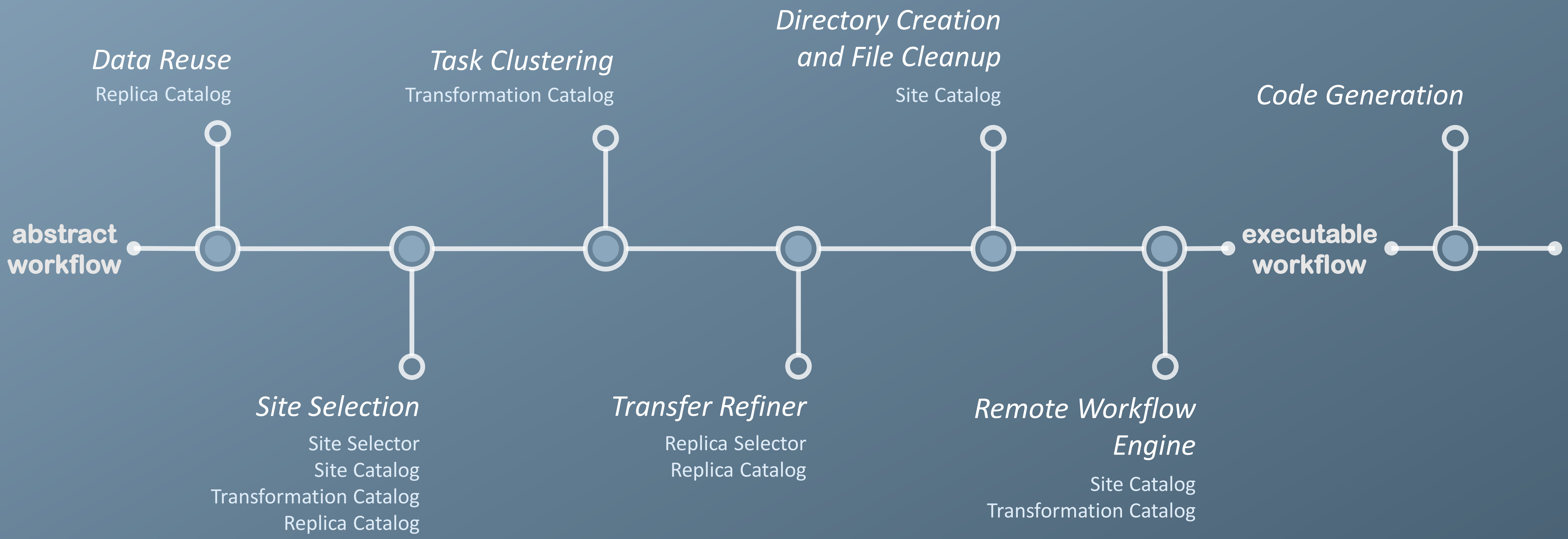


submit host
(e.g., user's laptop)

workflow wrapped as an MPI job

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

Pegasus’ 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

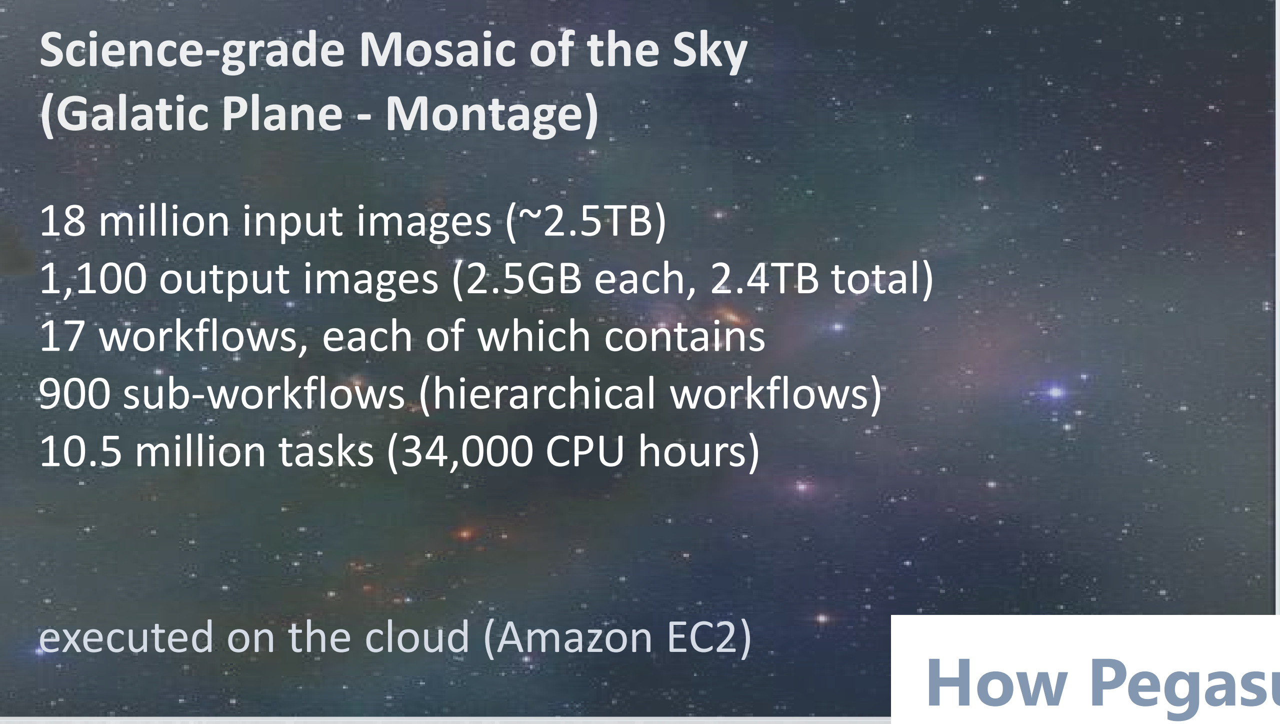
Remote Workflow Engine

Site Catalog
Transformation Catalog

executable workflow

Code Generation

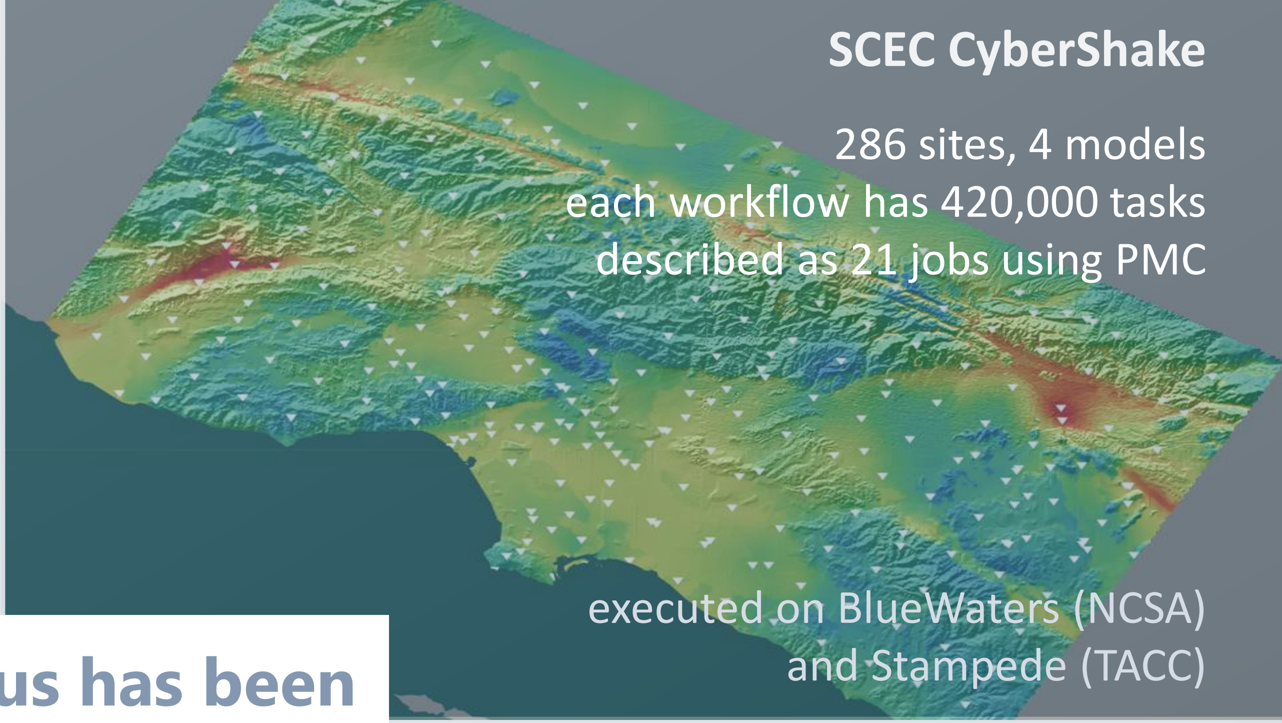
How Pegasus has been used?



Science-grade Mosaic of the Sky (Galatic Plane - Montage)

18 million input images (~2.5TB)
1,100 output images (2.5GB each, 2.4TB total)
17 workflows, each of which contains
900 sub-workflows (hierarchical workflows)
10.5 million tasks (34,000 CPU hours)

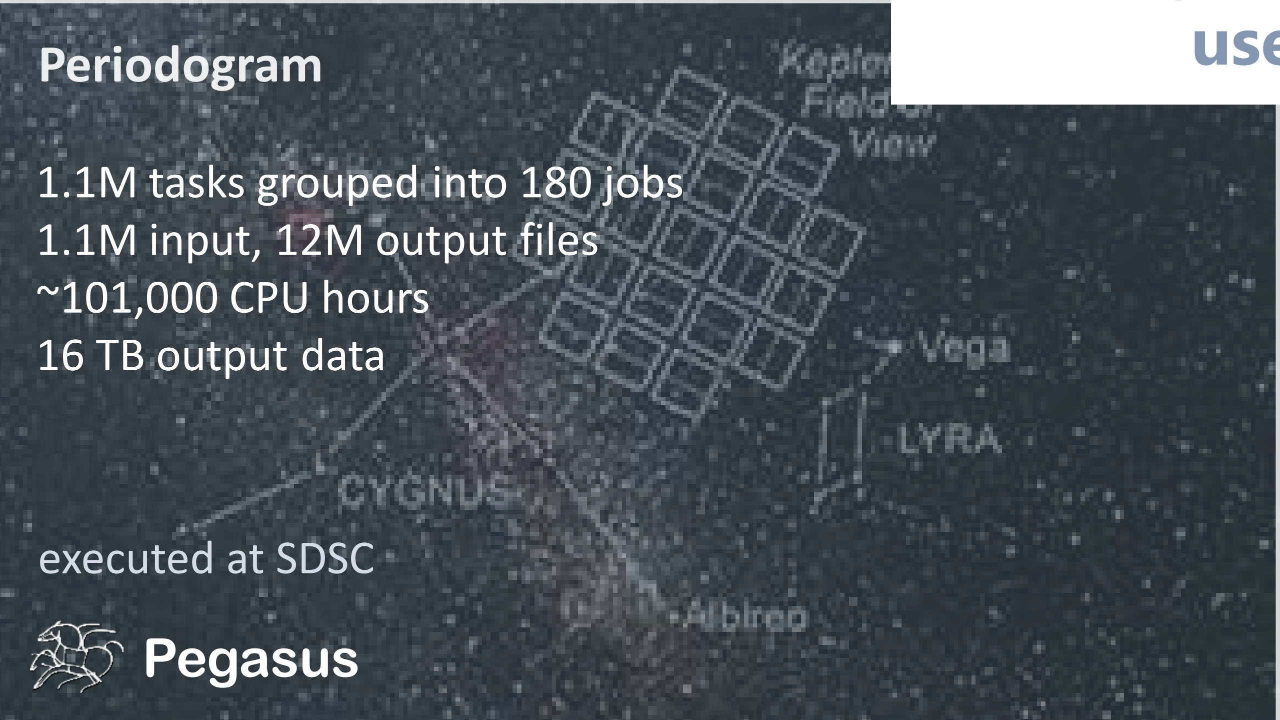
executed on the cloud (Amazon EC2)



SCEC CyberShake

286 sites, 4 models
each workflow has 420,000 tasks
described as 21 jobs using PMC

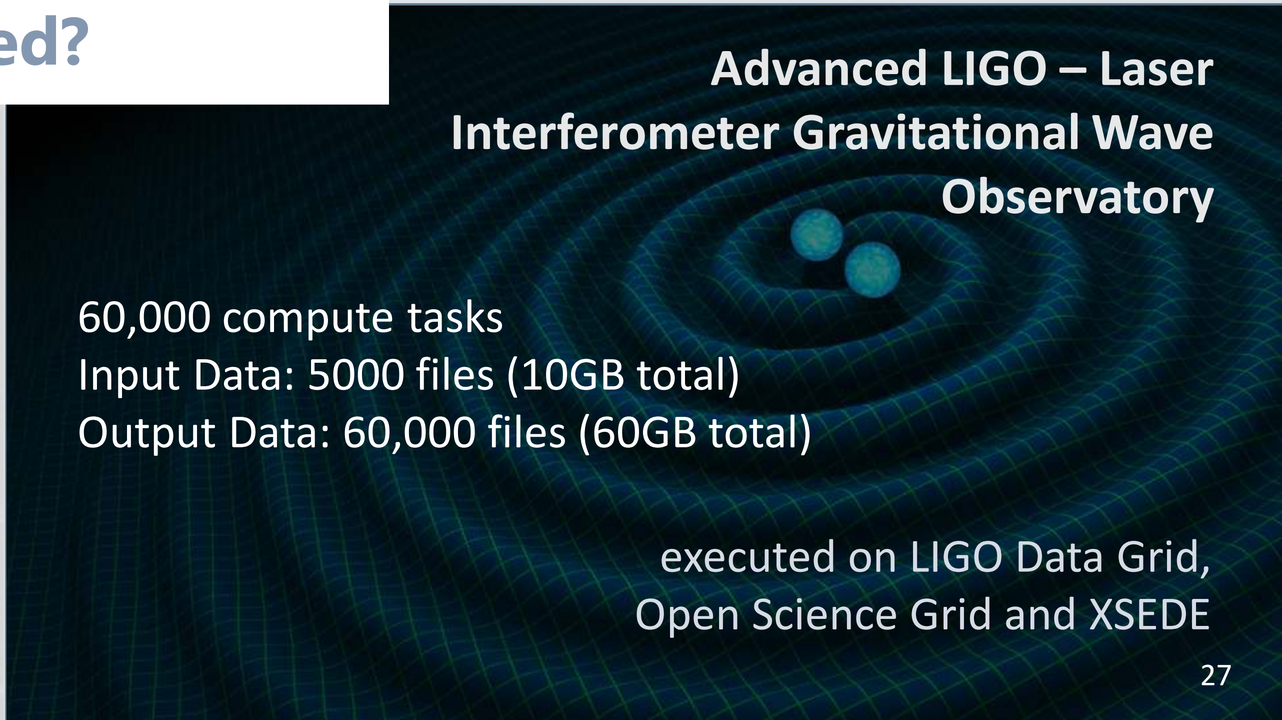
executed on BlueWaters (NCSA)
and Stampede (TACC)



Periodogram

1.1M tasks grouped into 180 jobs
1.1M input, 12M output files
~101,000 CPU hours
16 TB output data

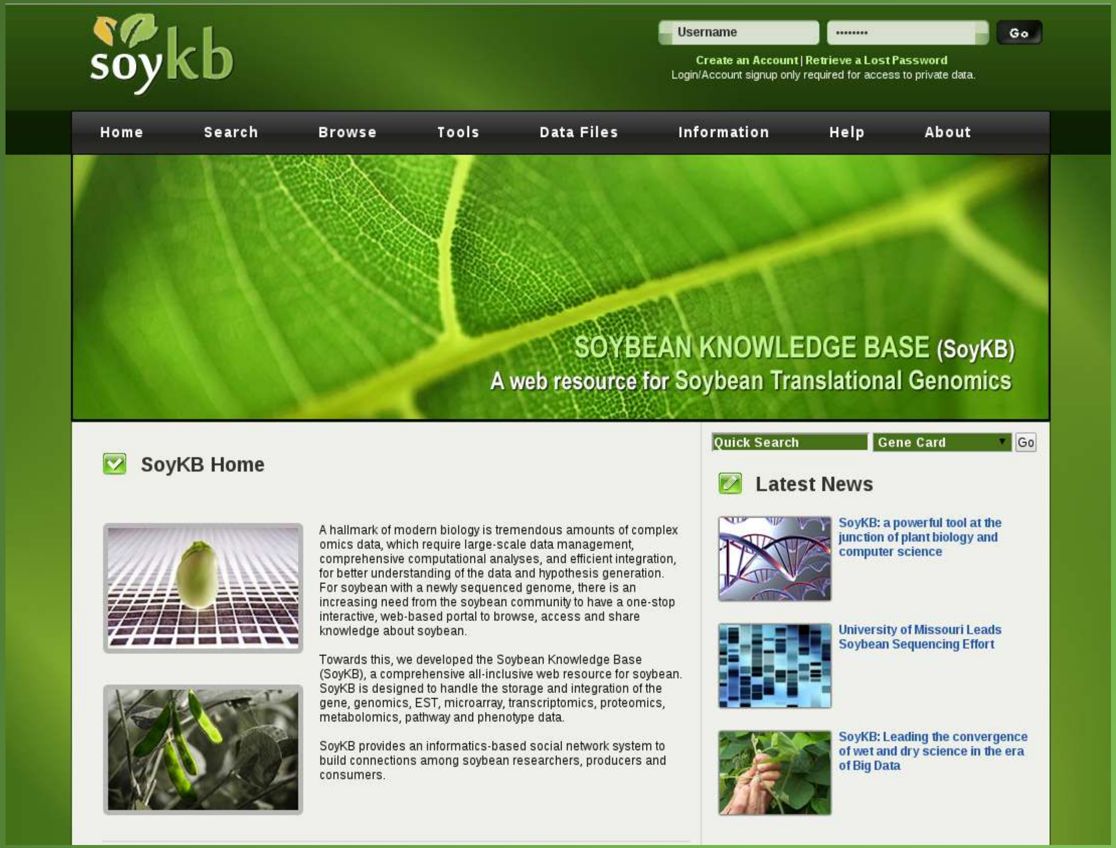
executed at SDSC



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


soykb
Username
••••••••
Go
Create an Account | Retrieve a Lost Password
Login/Account signup only required for access to private data.
Home
Search
Browse
Tools
Data Files
Information
Help
About
SOYBEAN KNOWLEDGE BASE (SoyKB)
A web resource for Soybean Translational Genomics
Quick Search
Gene Card
Go

SoyKB Home

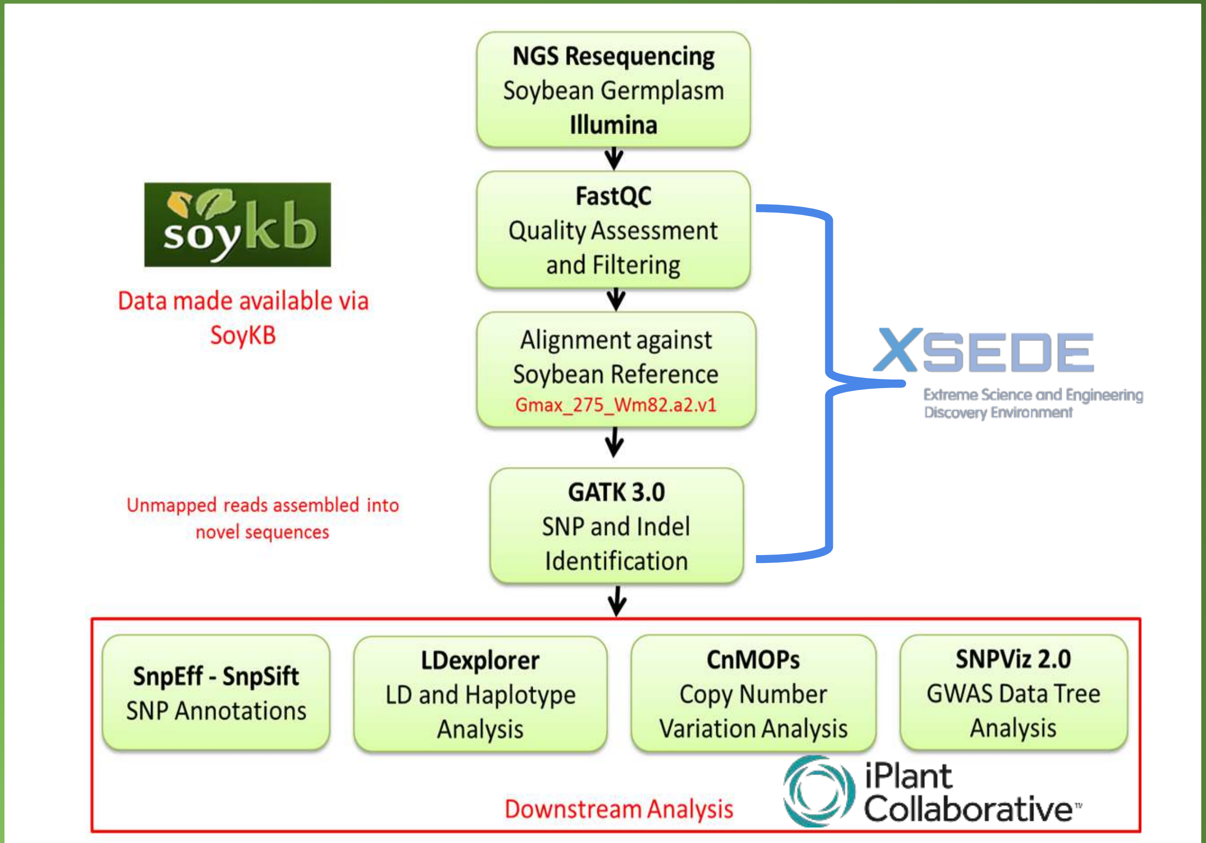
A hallmark of modern biology is tremendous amounts of complex omics data, which require large-scale data management, comprehensive computational analyses, and efficient integration, for better understanding of the data and hypothesis generation. For soybean with a newly sequenced genome, there is an 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



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

Data made available via SoyKB

Unmapped reads assembled into novel sequences

SnpEff - SnpSift SNP Annotations

LDexplorer LD and Haplotype Analysis

CnMOPs Copy Number Variation Analysis

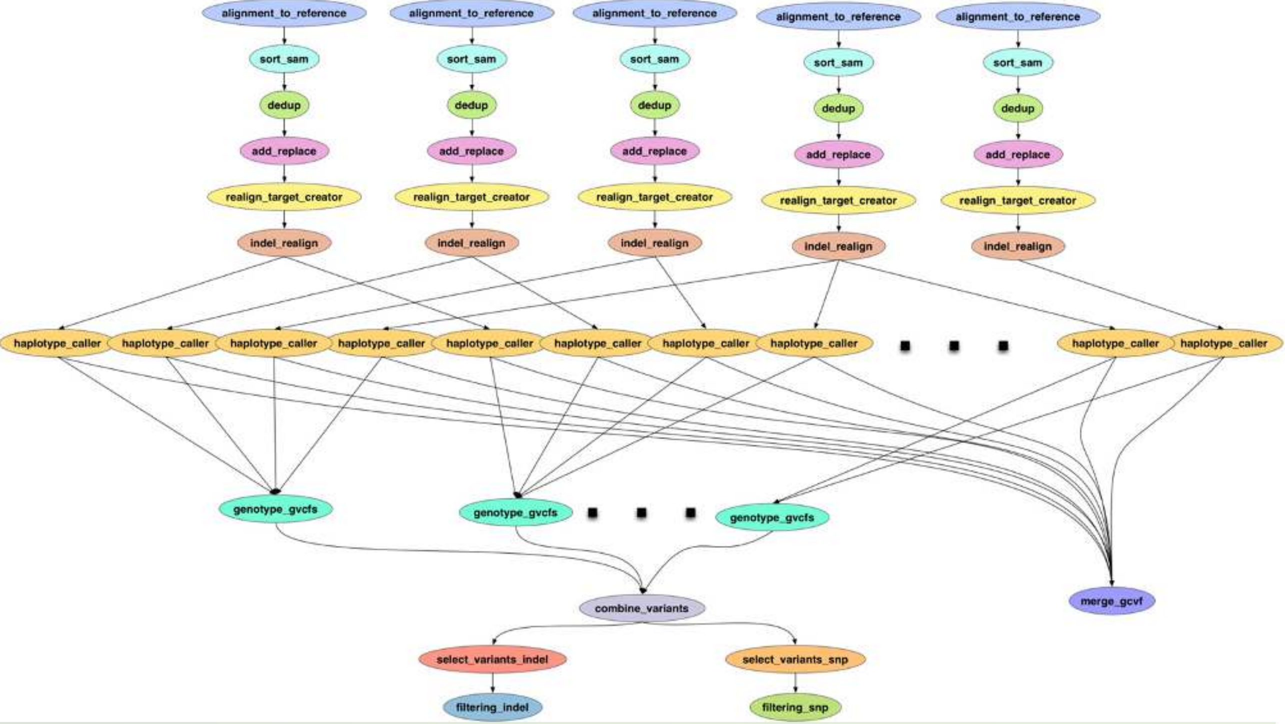
SNPViz 2.0 GWAS Data Tree Analysis

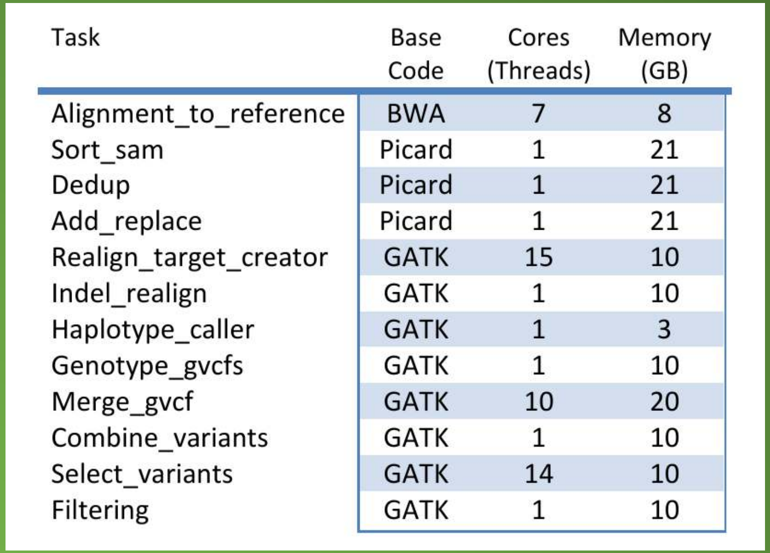
Downstream Analysis

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





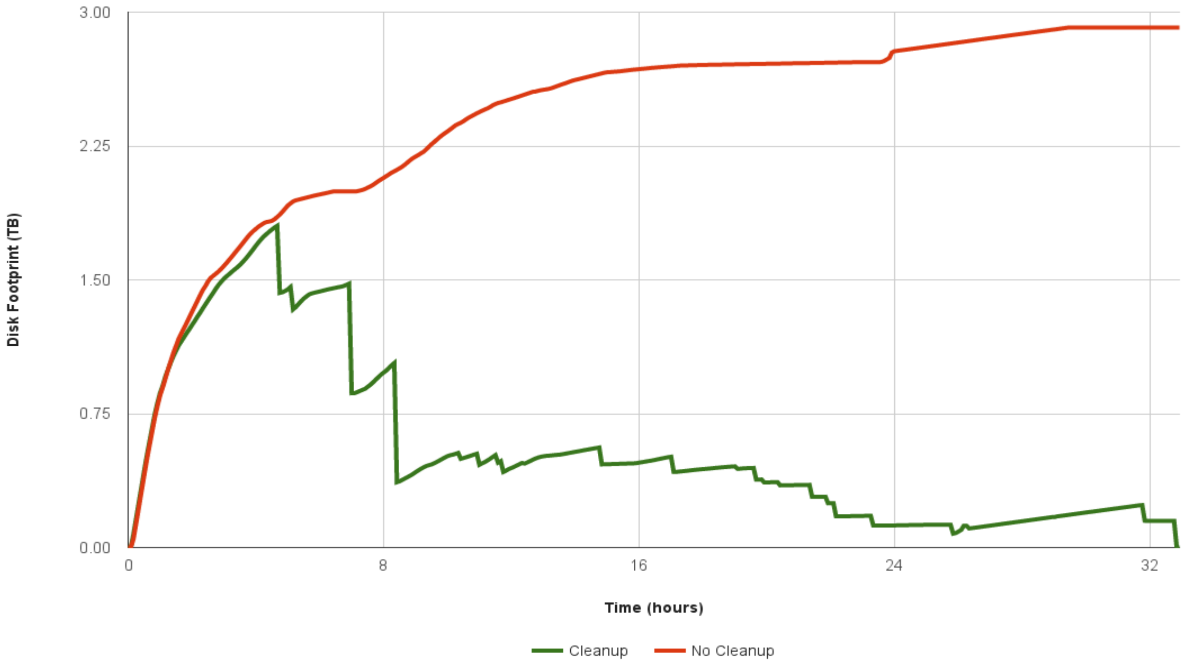
Task	Base Code	Cores (Threads)	Memory (GB)
Alignment_to_reference	BWA	7	8
Sort_sam	Picard	1	21
Dedup	Picard	1	21
Add_replace	Picard	1	21
Realign_target_creator	GATK	15	10
Indel_realign	GATK	1	10
Haplotype_caller	GATK	1	3
Genotype_gvcfs	GATK	1	10
Merge_gvcf	GATK	10	20
Combine_variants	GATK	1	10
Select_variants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



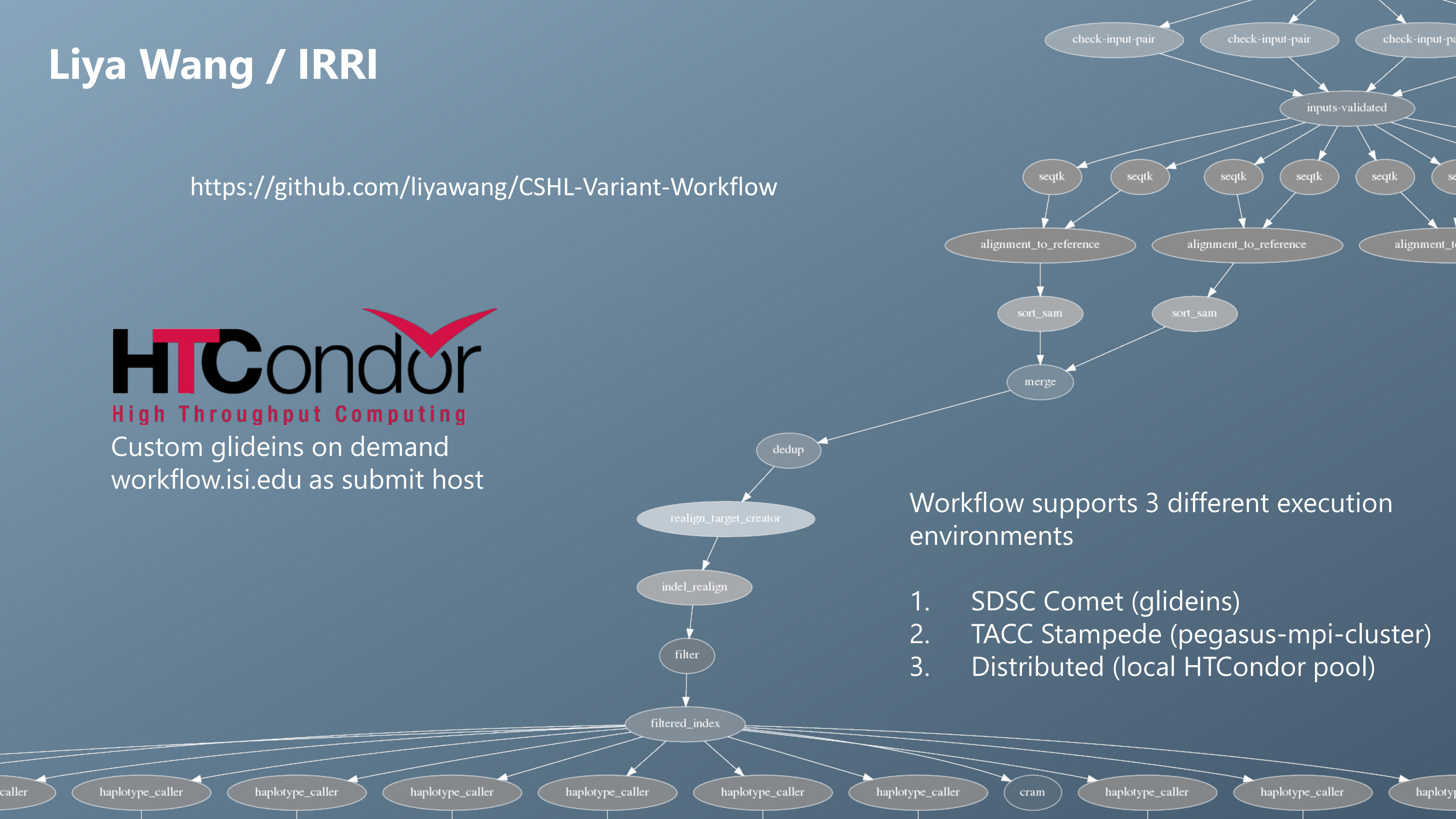
Liya Wang / IRRI

https://github.com/liyawang/CSHL-Variant-Workflow

HTCondor

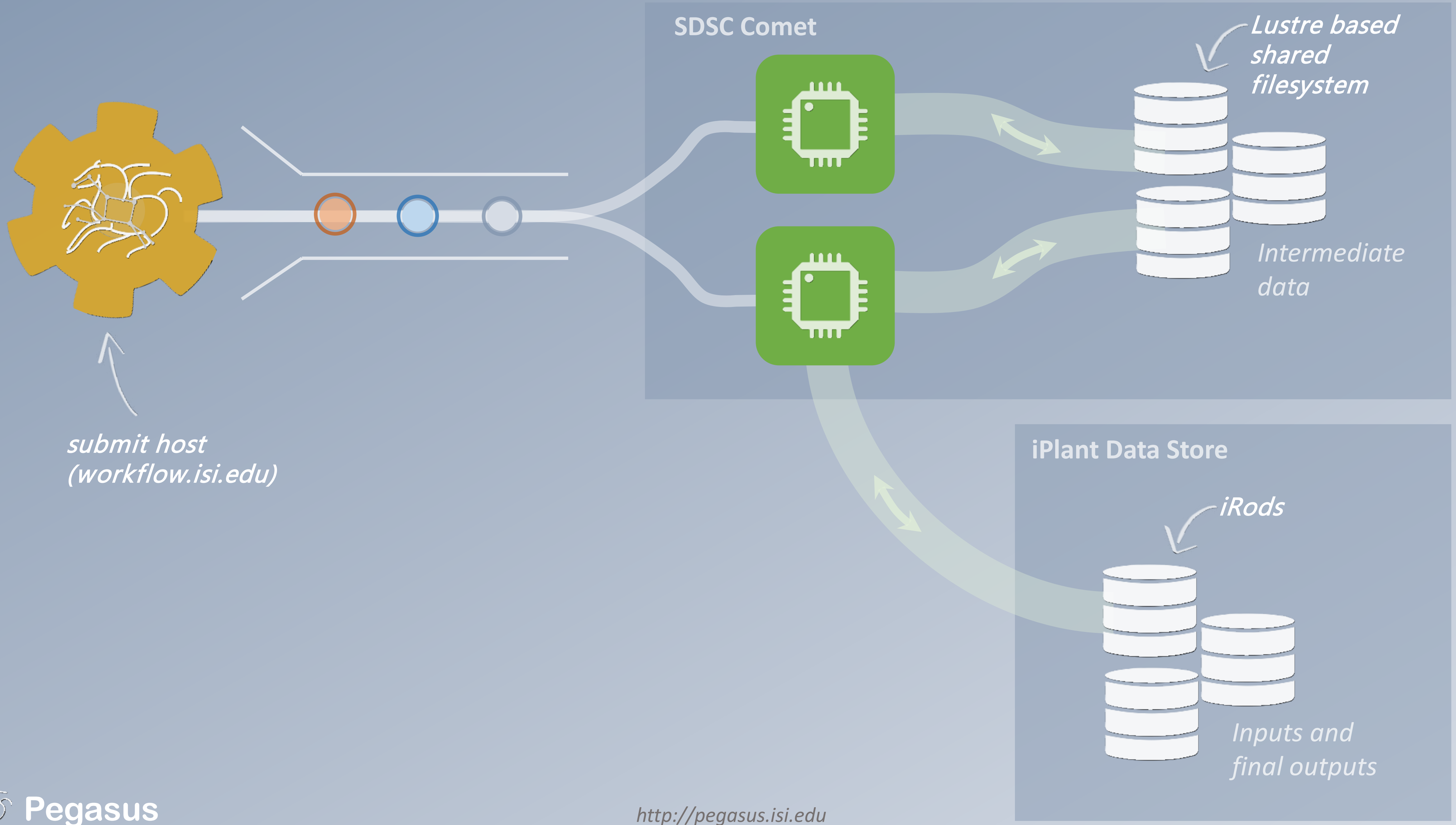
High Throughput Computing

Custom glideins on demand
workflow.isi.edu as submit host



Workflow supports 3 different execution environments

1. SDSC Comet (glideins)
2. TACC Stampede (pegasus-mpi-cluster)
3. Distributed (local HTCondor pool)


SDSC Comet
Lustre based shared filesystem
Intermediate data
submit host (workflow.isi.edu)
iPlant Data Store
iRods
Inputs and final outputs

Pegasus est. 2001

Automate, recover, and debug scientific computations.

Get Started

Pegasus Website

http://pegasus.isi.edu

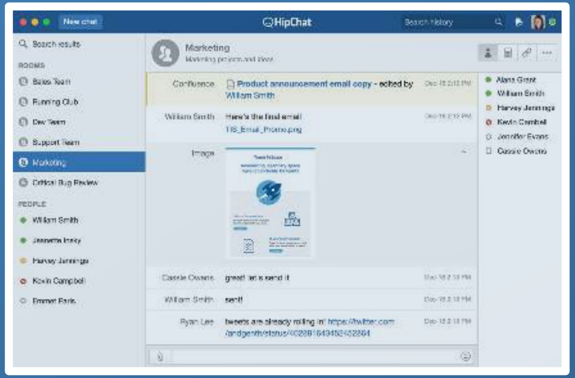
Users Mailing List

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 2:18 PM

William Smith sent! Dec 18 2:18 PM

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

NSF U.S. DEPARTMENT OF ENERGY NIH

Meet our team

Ewa Deelman

Karan Vahi

Gideon Juve

Mats Rynge

Rajiv Mayani

Rafael Ferreira da Silva