

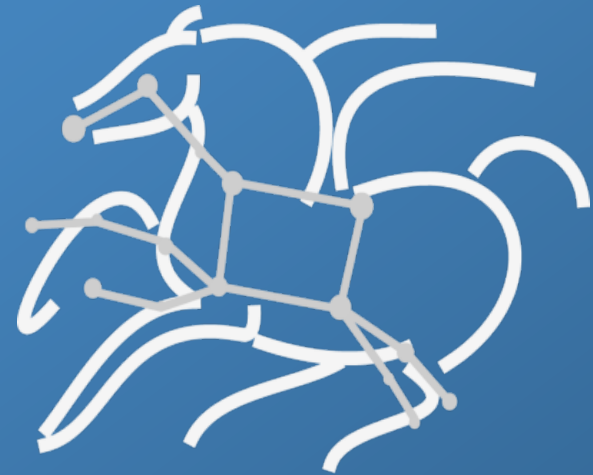


U.S. DEPARTMENT OF
ENERGY



Pegasus

Pegasus Workflow Management System



Mats Rynge
rynge@isi.edu

USC Viterbi
School of Engineering
Information Sciences Institute

<https://pegasus.isi.edu>

Why workflows?

Automates complex, multi-stage processing pipelines

Enables parallel, distributed **computations**

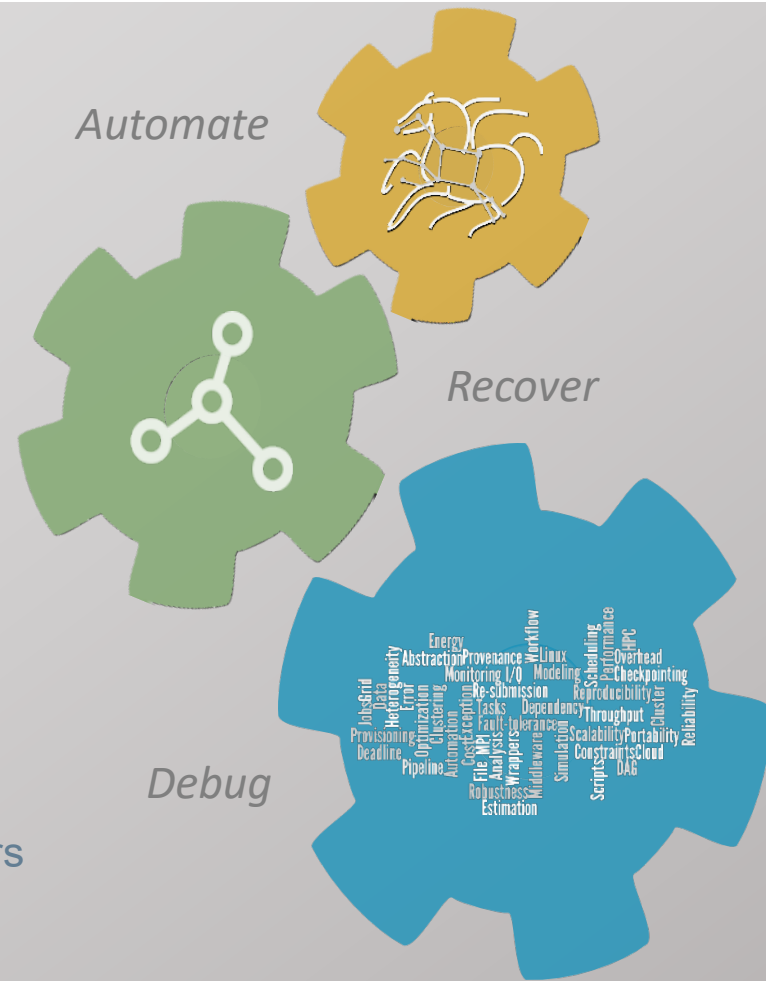
Automatically executes data transfers

Reusable, aids **reproducibility**

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

Handles **failures** with to provide reliability

Keeps track of data and **files**



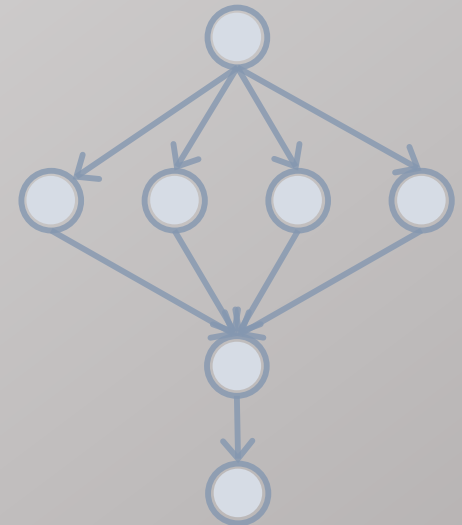
Key Pegasus Concepts

Pegasus WMS == Pegasus planner (mapper) + DAGMan workflow engine + HTCCondor scheduler/broker

- Pegasus maps workflows to infrastructure
- DAGMan manages dependencies and reliability
- HTCCondor is used as a broker to interface with different schedulers

Workflows are DAGs (or hierarchical DAGs)

- Directed acyclic graph
- Nodes: jobs, edges: dependencies
- No while loops, no conditional branches



Planning occurs ahead of execution

- (Except hierarchical workflows)

Planning converts an abstract workflow into a concrete, executable workflow

- Planner is like a compiler

Taking a closer look into a workflow...

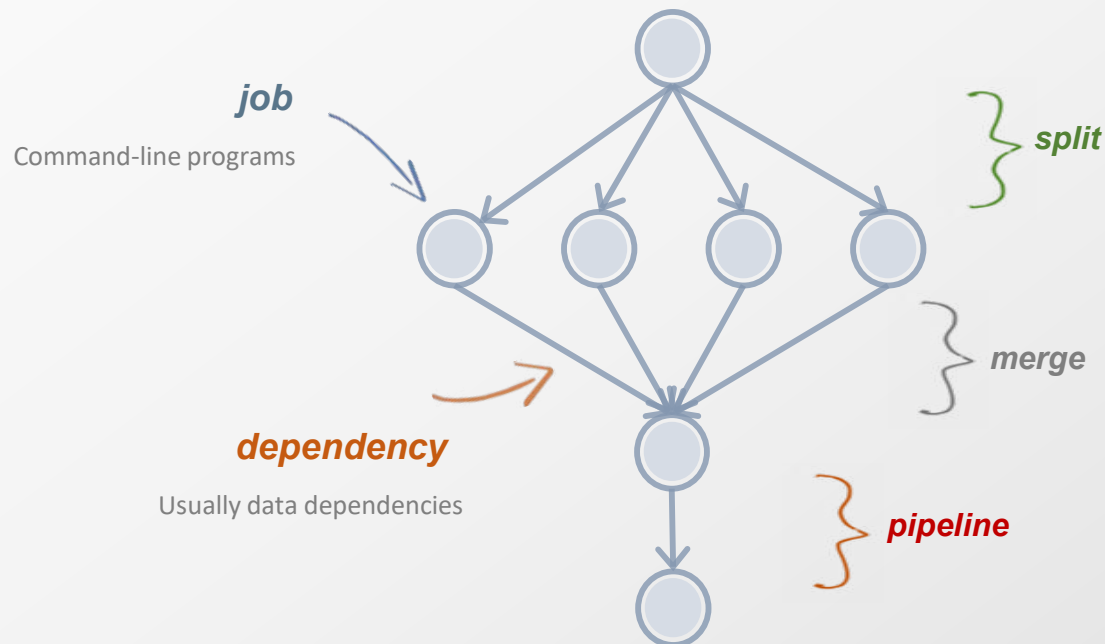
abstract workflow

executable workflow

optimizations

storage constraints

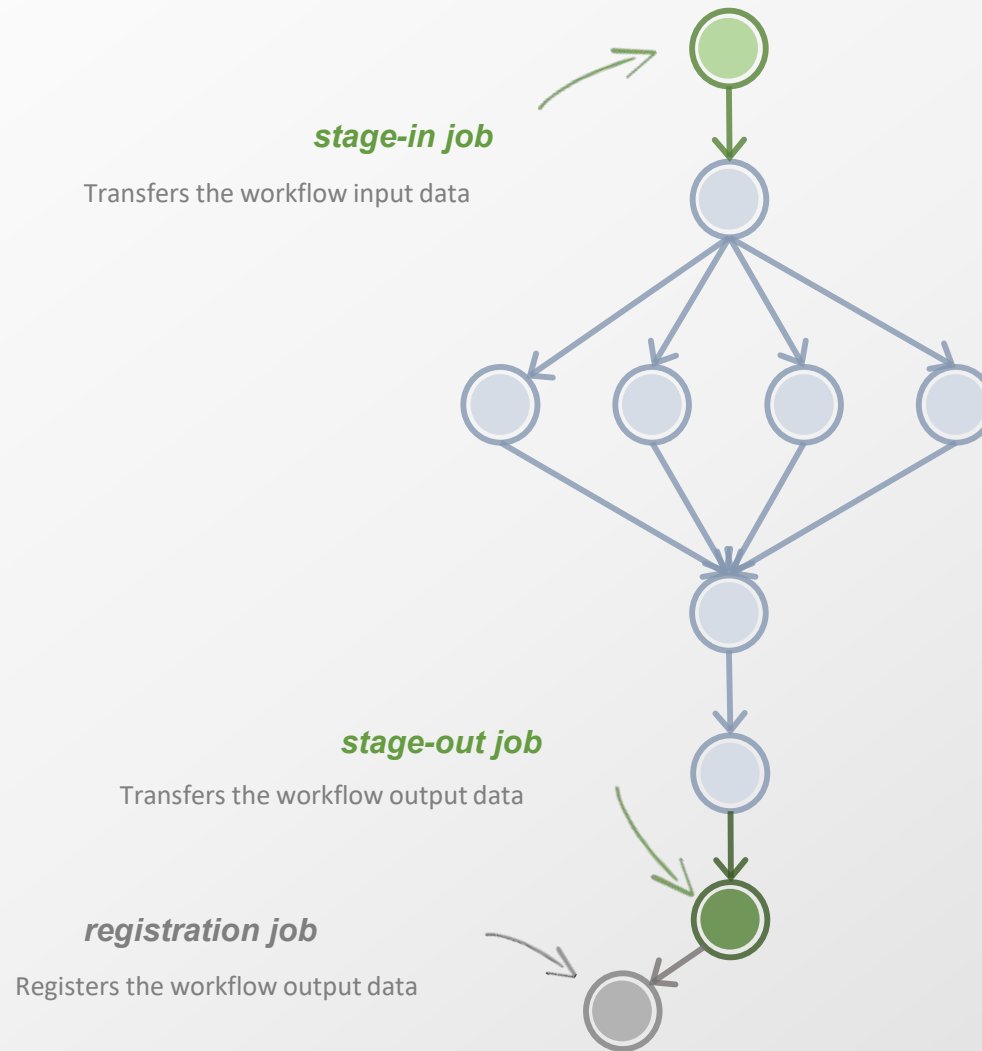
directed-acyclic graphs



DAG in XML



From the abstraction to execution!

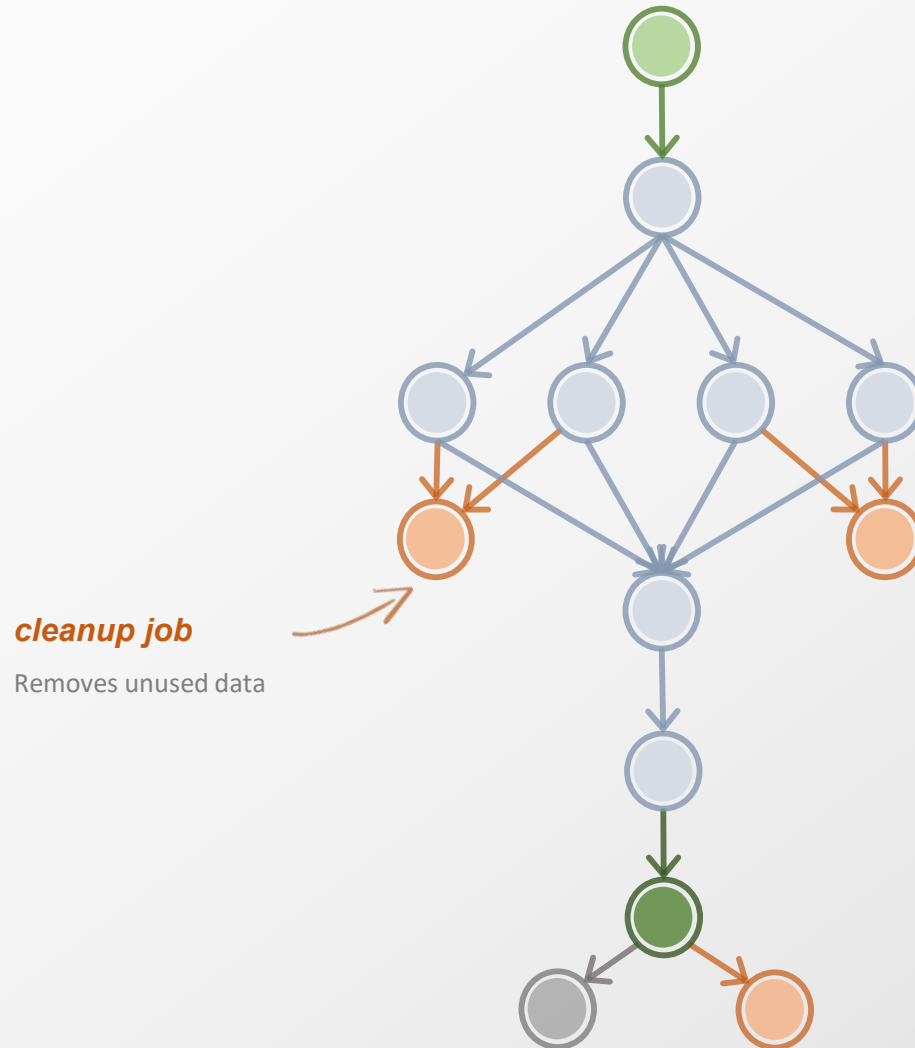


Optimizing storage usage...

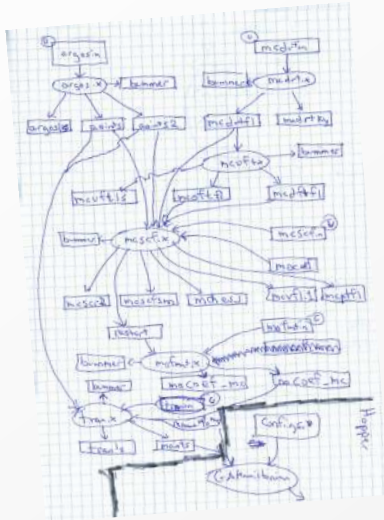
abstract workflow

executable workflow

storage constraints



Pegasus also provides tools to generate the abstract workflow



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

from Pegasus.DAX3 import *
import sys
import os

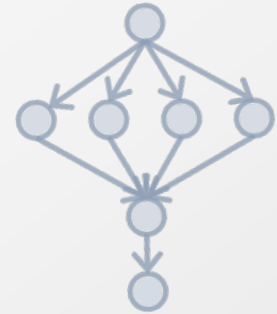
# Create an abstract dag
dax = ADAG("hello_world")

# 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)
```



```
<?xml version="1.0" encoding="UTF-8"?>
<!-- generator: python -->
<adag xmlns="http://pegasus.isi.edu/schema/DAX"
      version="3.4" name="hello_world">

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

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

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

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

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

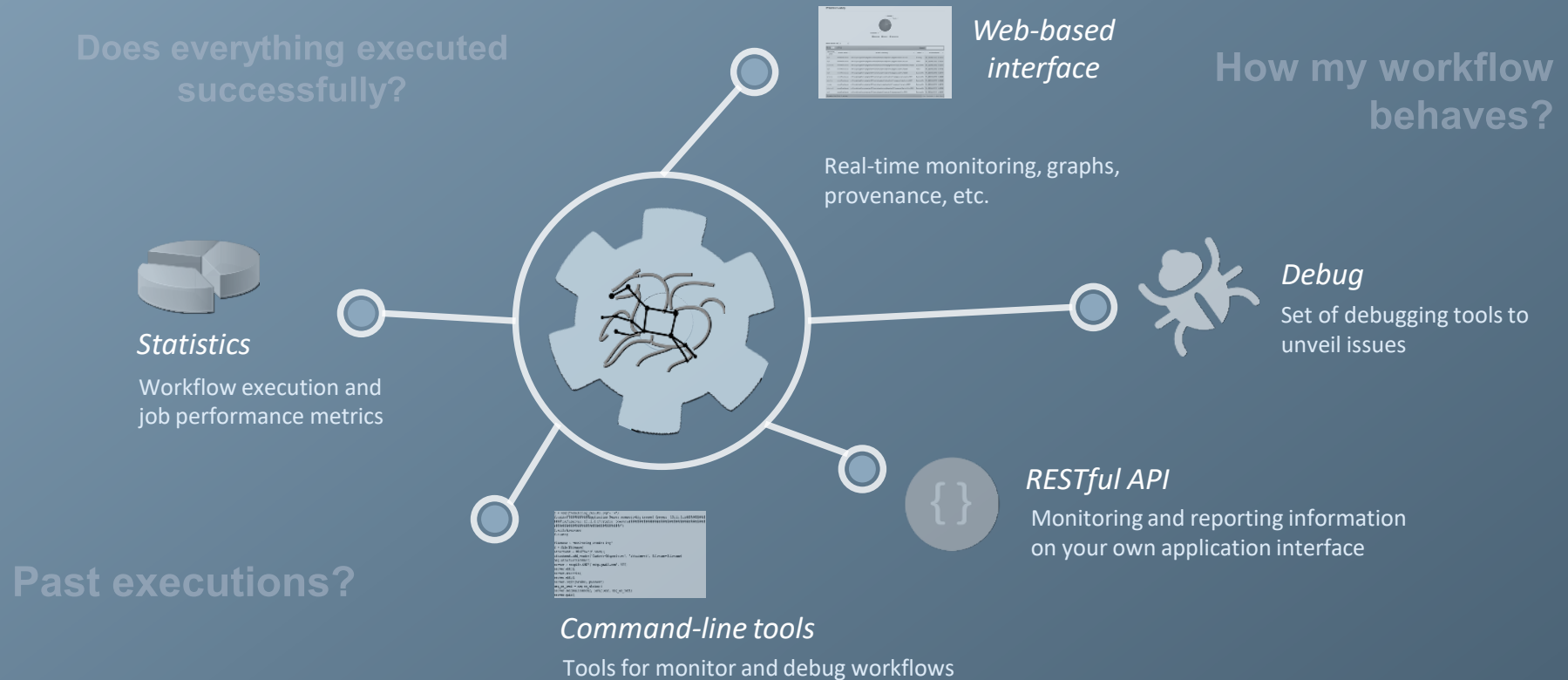


DAG in XML

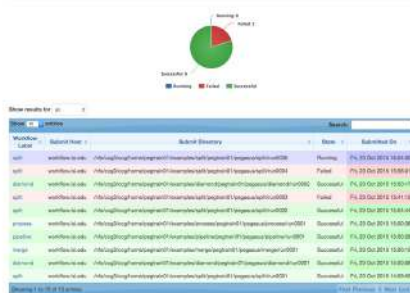


While you wait...

...or the execution is finished.



Workflow Listing



Pegasus dashboard

web interface for monitoring and debugging workflows

Statistics

Workflow Wall Time	12 mins 20 secs
Workflow Cumulative Job Wall Time	6 mins 50 secs
Cumulative Job Walltime as seen from Submit Side	6 mins 50 secs
Workflow Cumulative Budget Time	6 mins 20 secs
Cumulative Job Budget Walltime as seen from Submit Side	6 mins 20 secs
Workflow Retries	1

Workflow Statistics						
Type	Successful	Failed	Incomplete	Total	Retries	Total + Retries
Tasks	5	0	0	5	0	5
Jobs	10	0	0	10	0	10
Sub Workflows	0	0	0	0	0	0

Running Workflows						
Type	Successful	Failed	Incomplete	Total	Retries	Total + Retries
Tasks	5	0	0	5	0	5
Jobs	10	0	0	10	0	10
Sub Workflows	0	0	0	0	0	0

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



Real-time Monitoring
Reporting
Debugging
Troubleshooting
RESTful API





Command-line...

```
$ pegasus-statistics
```

```
-----  
Type           Succeeded Failed Incomplete Total    Retries Total+Retries  
Tasks          100000    0      0         100000   543      100543  
Jobs           20206    0      0         20206   604      20810  
Sub-Workflows  0          0      0          0        0         0  
-----
```

```
Workflow wall time           : 19 hrs, 37 mins  
Cumulative job wall time     : 1 year, 5 days  
Cumulative job wall time as seen from submit side : 1 year, 27 days  
Cumulative job badput wall time : 2 hrs, 42 mins  
Cumulative job badput wall time as seen from submit side : 2 days, 2 hrs
```

```
$ 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%)
```

Provenance data can
be summarized
(*pegasus-statistics*)
or used for
debugging
(*pegasus-analyzer*)



And if a job fails?



pegasus-kickstart

```
() login02.osgconnect.net — Konsole
File Edit View Bookmarks Settings Help

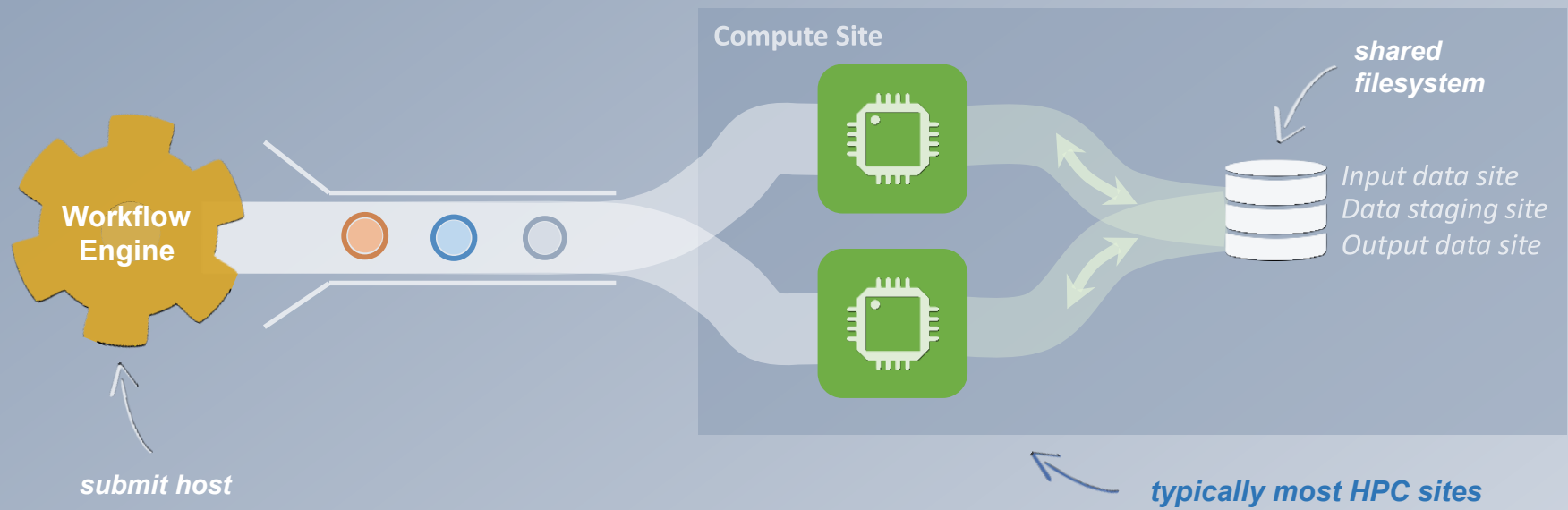
<?xml version="1.0" encoding="UTF-8"?>
<invocation xmlns="http://pegasus.isi.edu/schema/invocation" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://pegasus.isi.edu/schema/invocation http://pegasus.isi.edu/schema/iv-2.3.xsd" version="2.3" start="2016-11-28T14:27:48.909-06:00" duration="11200.691" transformation="job-wrapper.sh" derivation="ID0013214" resource="condorpool" wf-label="particleshower" wf-stamp="2016-11-22T21:14:13-06:00" interface="eth0" hostaddr="131.225.208.240" hostname="fnpc4593.fnal.gov" pid="1725084" uid="12740" user="osg" gid="9652" group="osg" umask="0022">
  <mainjob start="2016-11-28T14:27:49.007-06:00" duration="11200.593" pid="1725089">
    <usage utime="10921.591" stime="30.304" maxrss="395820" minflt="128741" majflt="18" nswap="0" inblock="85776" outblock="1717424" msgsnd="0" msgrcv="0" nsignals="0" nvcs="7676" nivcs="185495"/>
    <status raw="0"><regular exitcode="0"/></status>
    <statcall error="0">
      <file name="/storage/local/data1/condor/execute/dir_1227464/glide_bSxwfe/execute/dir_1724937/pegasus.XRZlp3/job-wrapper.sh">23212F62696E2F626173680A0A736574</file>
      <statinfo mode="0100755" size="1305" inode="16648869" nlink="1" blksize="4096" blocks="8" mtime="2016-11-28T12:10:53-06:00" atime="2016-11-28T14:27:48-06:00" ctime="2016-11-28T14:27:48-06:00" uid="12740" user="osg" gid="9652" group="osg"/>
    </statcall>
    <argument-vector>
      <arg nr="1">100</arg>
      <arg nr="2">0</arg>
      <arg nr="3">gamma</arg>
      <arg nr="4">62</arg>
      <arg nr="5">VERITAS</arg>
      <arg nr="6">corsika.tar.gz</arg>
      <arg nr="7">corsika75000Linux_QGSII_urqmd</arg>
      <arg nr="8">13213</arg>
    </argument-vector>
  </mainjob>
  <jobids condor="547839.0"/>
  <cwd>/storage/local/data1/condor/execute/dir_1227464/glide_bSxwfe/execute/dir_1724937/pegasus.XRZlp3</cwd>
  <usage utime="0.013" stime="0.085" maxrss="828" minflt="2448" majflt="0" nswap="0" inblock="0" outblock="0" msgsnd="0" msgrcv="0" nsignals="0" nvcs="1" nivcs="12"/>
  <machine page-size="4096">
    <stamp>2016-11-28T14:27:48.909-06:00</stamp>
    <uname system="linux" nodename="fnpc4593.fnal.gov" release="2.6.32-642.6.2.el6.x86_64" machine="x86_64">#1 SMP Tue Oct 25 15:06:33 CDT 2016</uname>
    <linux>
      <ram total="65319608" free="1071948" shared="0" buffer="148224"/>
      <swap total="8388604" free="7741364"/>
      <boot idle="45893257.760">2016-11-09T16:40:54.260-06:00</boot>
      <cpu count="32" speed="2000" vendor="AuthenticAMD">AMD Opteron(tm) Processor 6128</cpu>
      <load min1="26.35" min5="27.70" min15="24.33"/>
      <procs total="881" running="23" sleeping="854" waiting="3" zombie="1" vmsize="65009304" rss="14780272"/>
      <task total="1273" running="24" sleeping="1243" waiting="5" zombie="1"/>
    </linux>
  </machine>
</invocation>
```

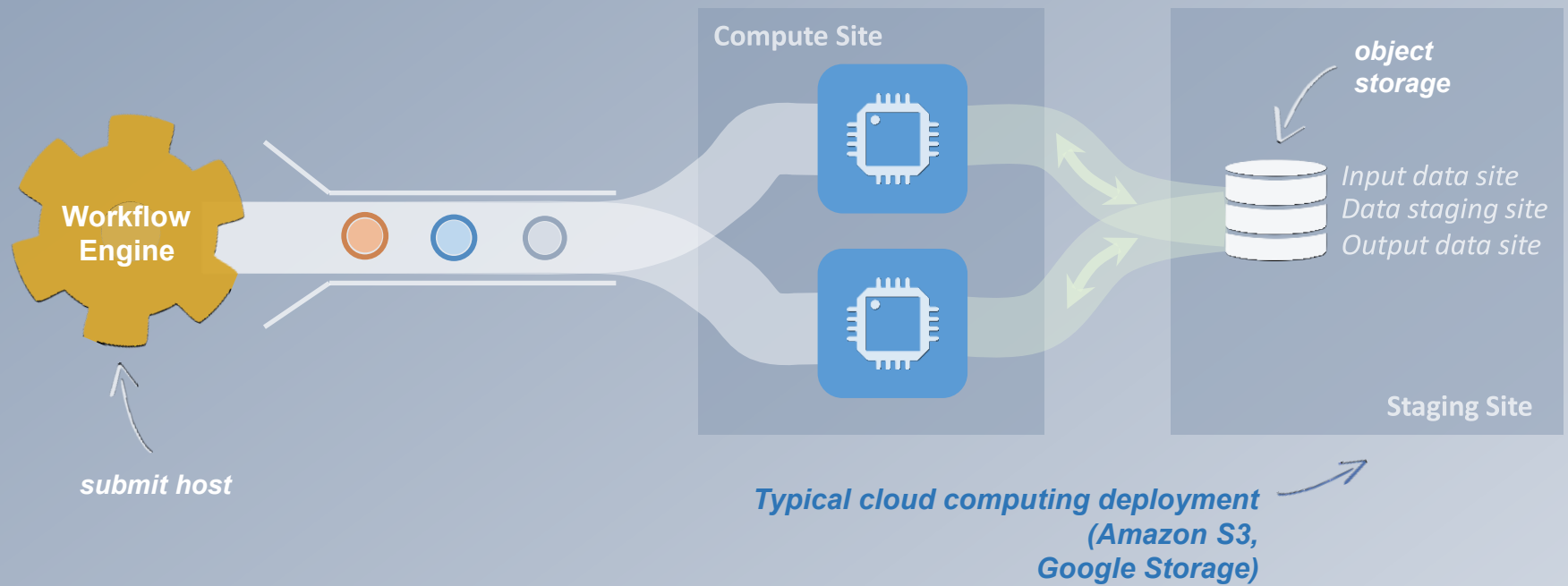
Data Staging Configurations

- HTCondor 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

High Performance Computing

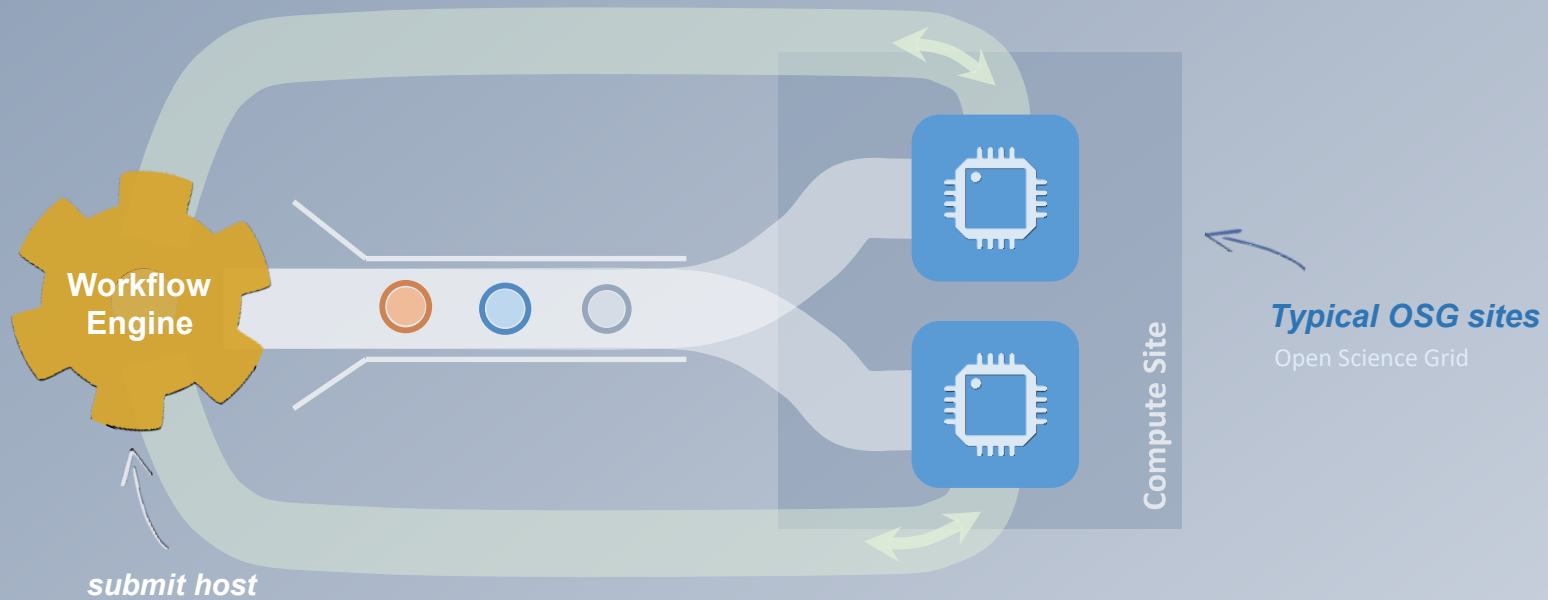
There are several possible configurations...



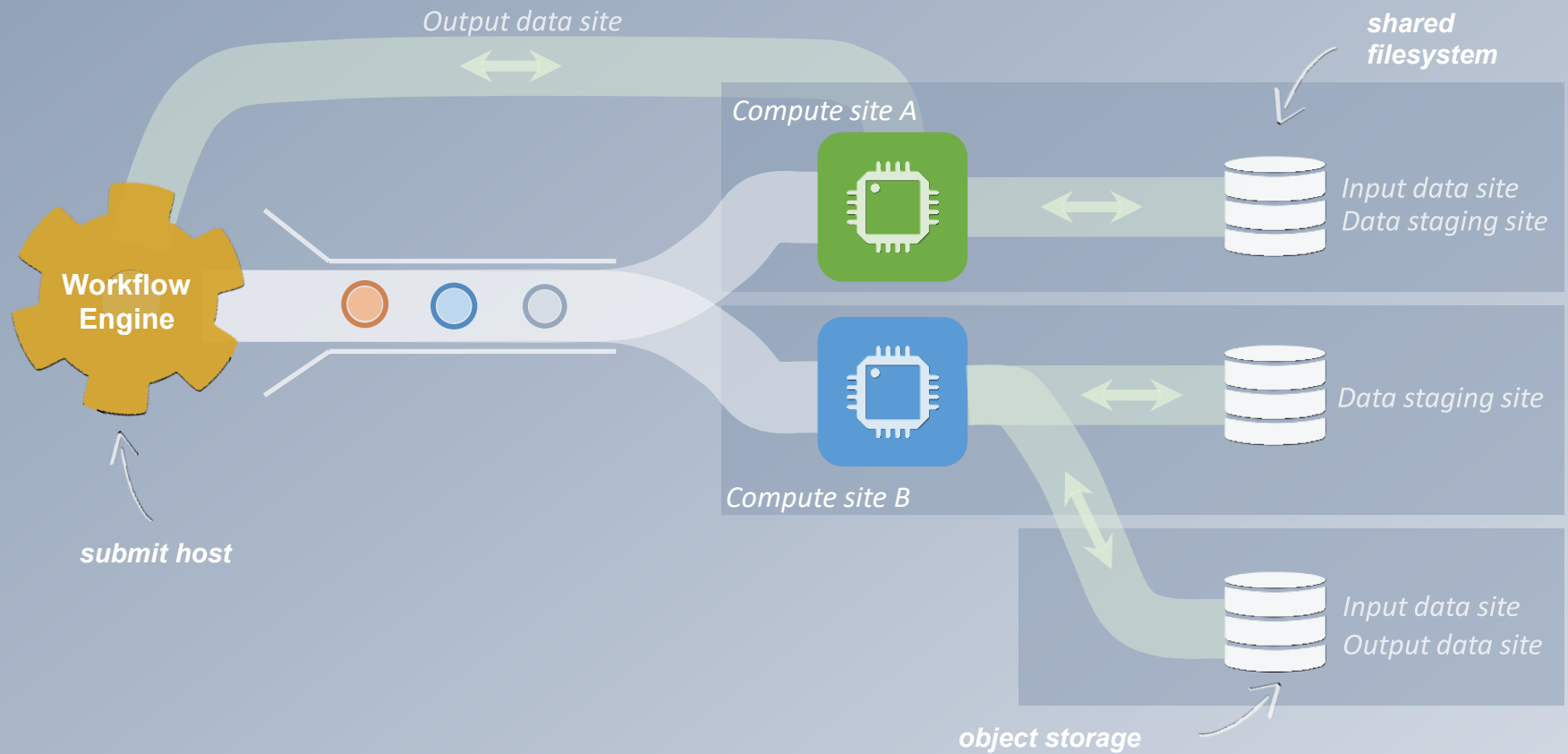


Grid Computing

local data management



And yes... you can mix everything!



pegasus-transfer

- Pegasus' internal data transfer tool
- Supports many 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
- Checkpoint and restart transfers
- Credential management
 - Uses the appropriate credential for each site and each protocol (even 3rd party transfers)

Protocols

- HTTP
- SCP
- GridFTP
- iRods
- Amazon S3
- Google Storage
- SRM
- FDT
- stashcp
- cp
- ln -s
- Docker
-

So, what information does Pegasus need?



How does Pegasus decide where to execute?

site catalog

transformation catalog

replica catalog

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 where 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">/wf/key.priv</profile>

</site>
...
```



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

site catalog

transformation catalog

replica catalog

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?

site catalog

transformation catalog

replica catalog

```
# 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    http://storage.mysite.edu/examples/diamond/input/f.a    site="storage"
```



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.)



site name

in which site the file is available



Replica catalog – multiple sources

site catalog

transformation catalog

replica catalog

pegasus.conf

```
# Add Replica selection options so that it will try URLs first, then
# XrootD for OSG, then gridftp, then anything else
pegasus.selector.replica=Regex
pegasus.selector.replica.regex.rank.1=file:///cvmfs/*.
pegasus.selector.replica.regex.rank.2=file://.*
pegasus.selector.replica.regex.rank.3=root://.*
pegasus.selector.replica.regex.rank.4=gridftp://.*
pegasus.selector.replica.regex.rank.5=.*
```

rc.data

```
# 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:///cvmfs/oasis.opensciencegrid.org/diamond/input/f.a    site="cvmfs"
f.a    file:///local-storage/diamond/input/f.a    site="prestaged"
f.a    gridftp://storage.mysite/edu/examples/diamond/input/f.a    site="storage"
```

A few more features...

Performance, why not improve it?

workflow restructuring

workflow reduction

hierarchical workflows

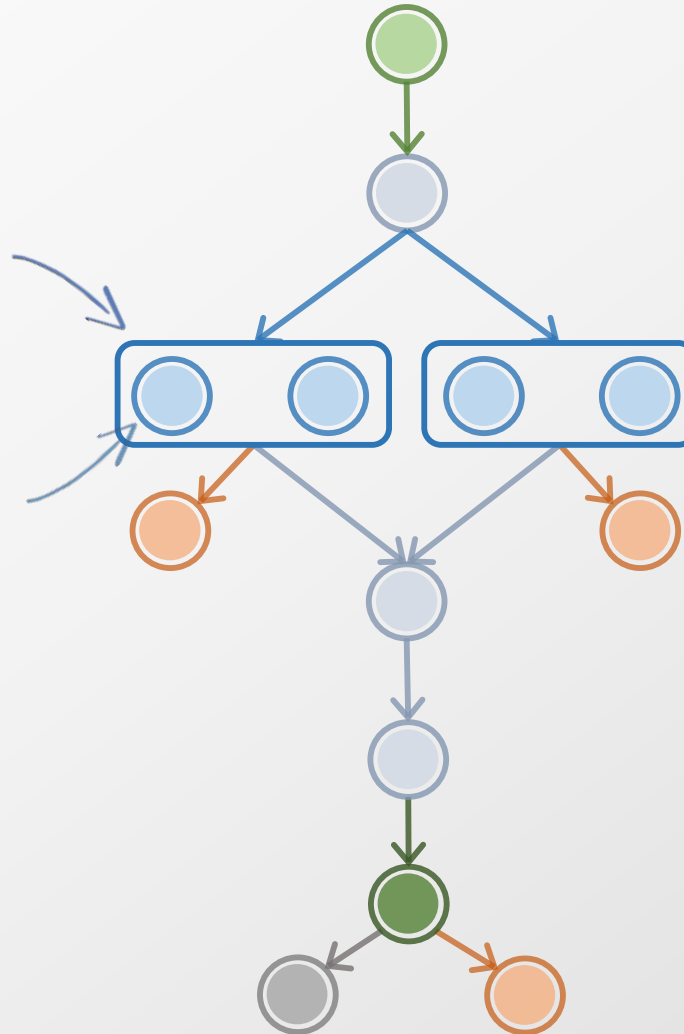
pegasus-mpi-cluster

clustered job

Groups small jobs together to improve performance

task

small granularity



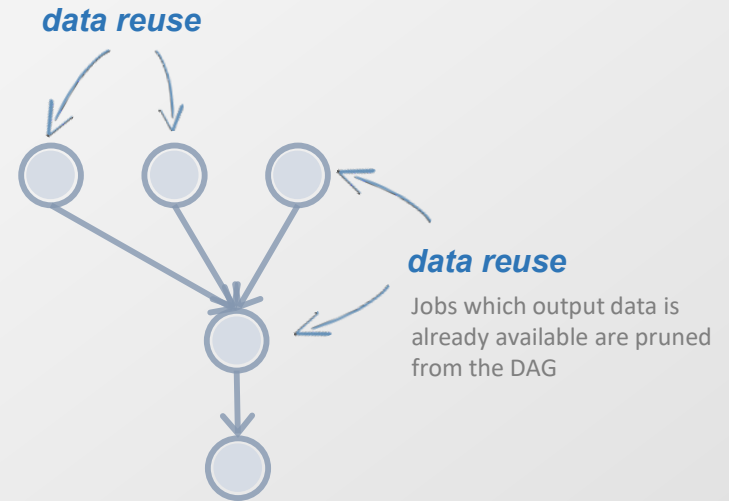
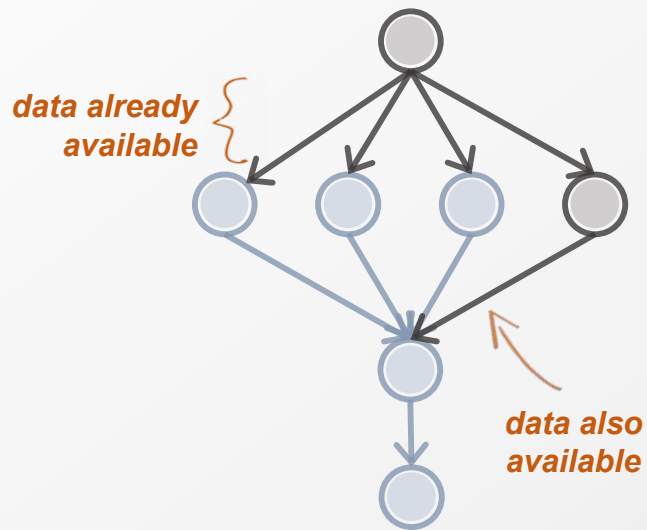
What about **data reuse**?

workflow restructuring

workflow reduction

hierarchical workflows

pegasus-mpi-cluster



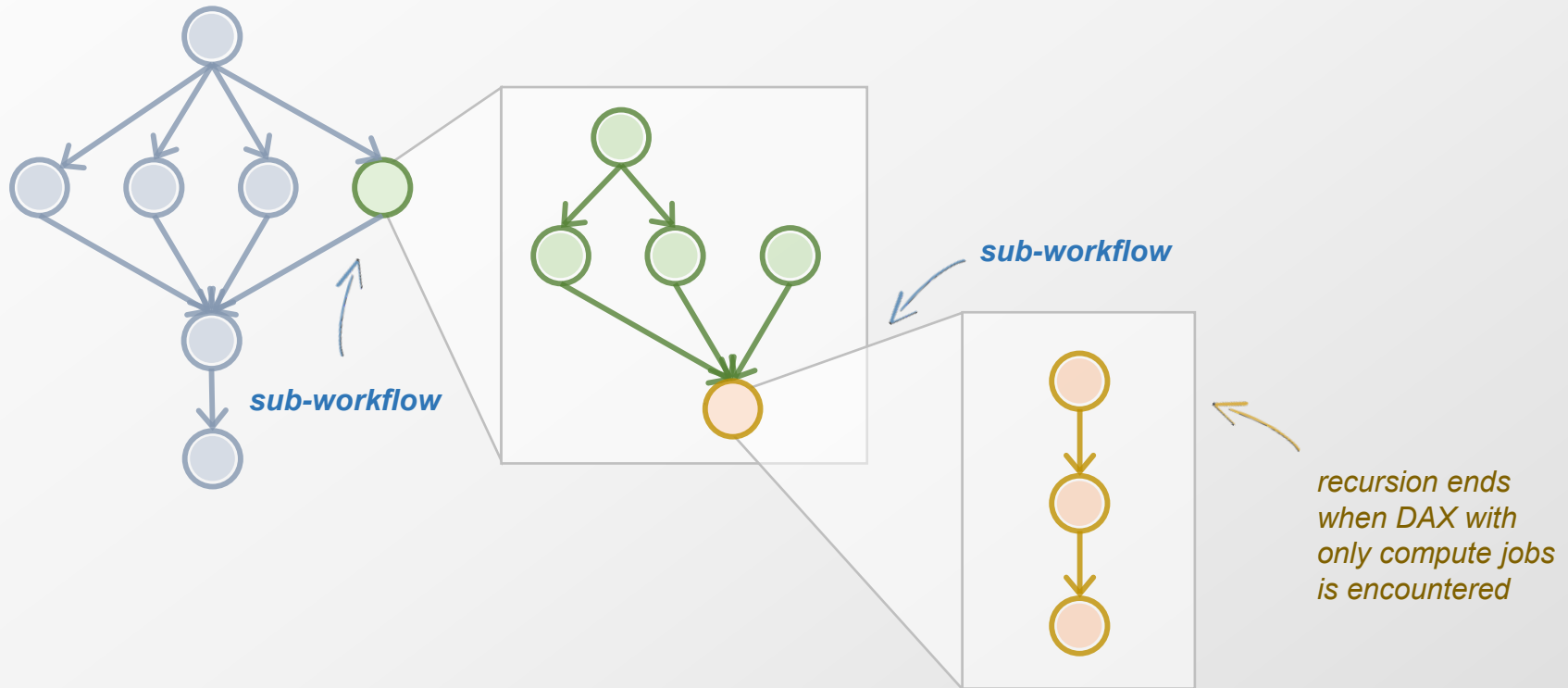
Pegasus also handles **large-scale workflows**

workflow restructuring

workflow reduction

hierarchical workflows

pegasus-mpi-cluster



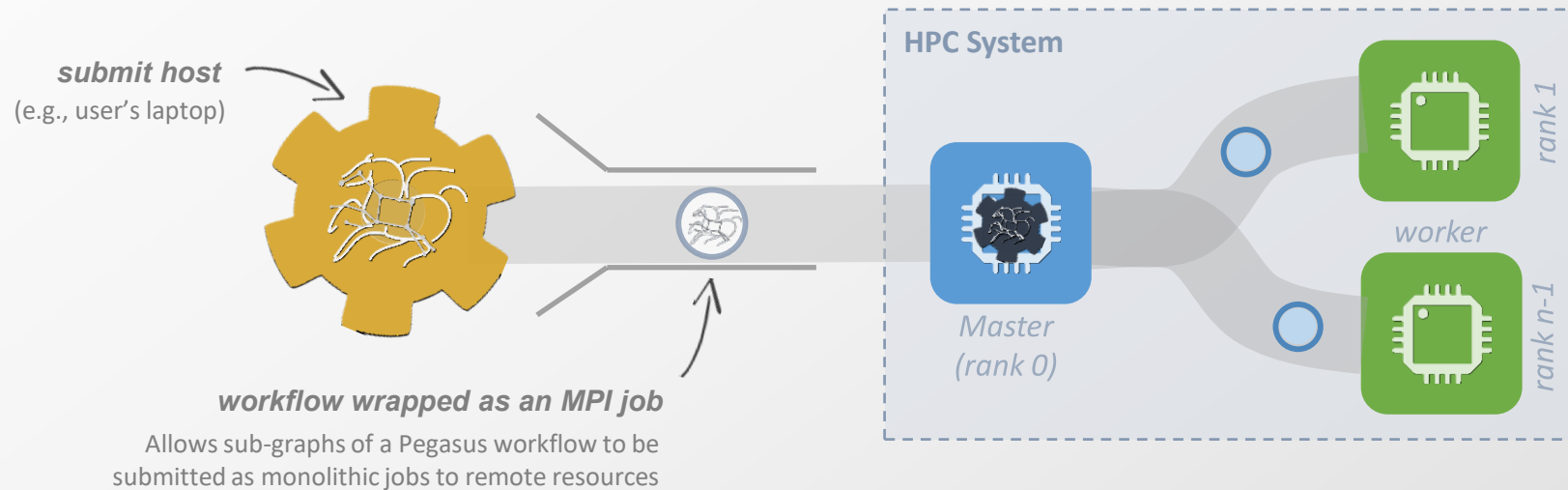
Running **fine-grained** workflows on HPC systems...

workflow restructuring

workflow reduction

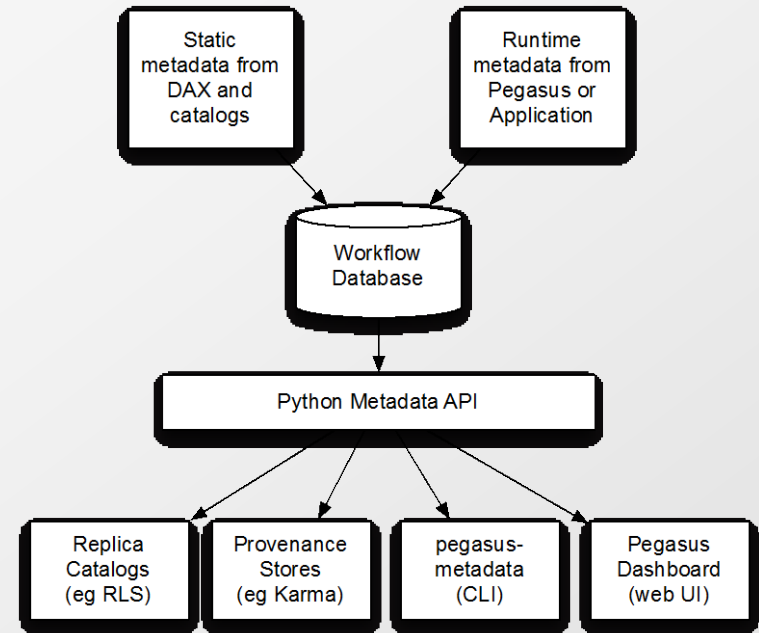
hierarchical workflows

pegasus-mpi-cluster



Metadata

- Can associate arbitrary key-value pairs with workflows, jobs, and files
- Replica selection
 - Input files are selected based on metadata attributes
- Data registration
 - Output files get tagged with metadata on registration
- Static and runtime metadata
 - Static: application parameters
 - Runtime: performance metrics



New in Pegasus 4.6, added to support users who want to select data based on attributes rather than names (e.g. LIGO)

DAX Metadata Example

**Workflow,
Job, File**

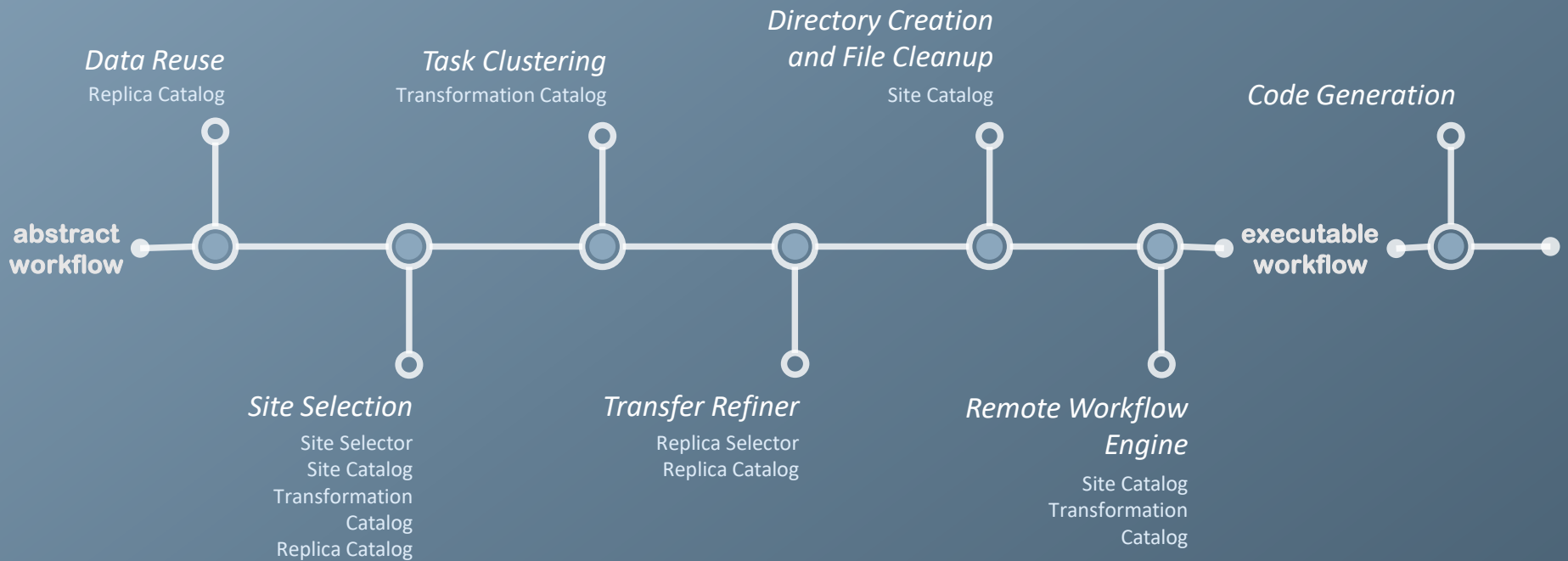
```
1 <adag ...>
  <metadata key="experiment">par_all27_prot_lipid</metadata>
  <job id="ID0000001" name="namd">
    <argument><file name="equilibrate.conf"/></argument>
    <metadata key="timesteps">500000</metadata>
    <metadata key="temperature">200</metadata>
    <metadata key="pressure">1.01325</metadata>
    <uses name="Q42.psf" link="input">
      <metadata key="type">psf</metadata>
      <metadata key="charge">42</metadata>
    </uses>
    ...
    <uses name="eq.restart.coord" link="output" transfer="false">
      <metadata key="type">coordinates</metadata>
    </uses>
    ...
  </job>
</adag>
```

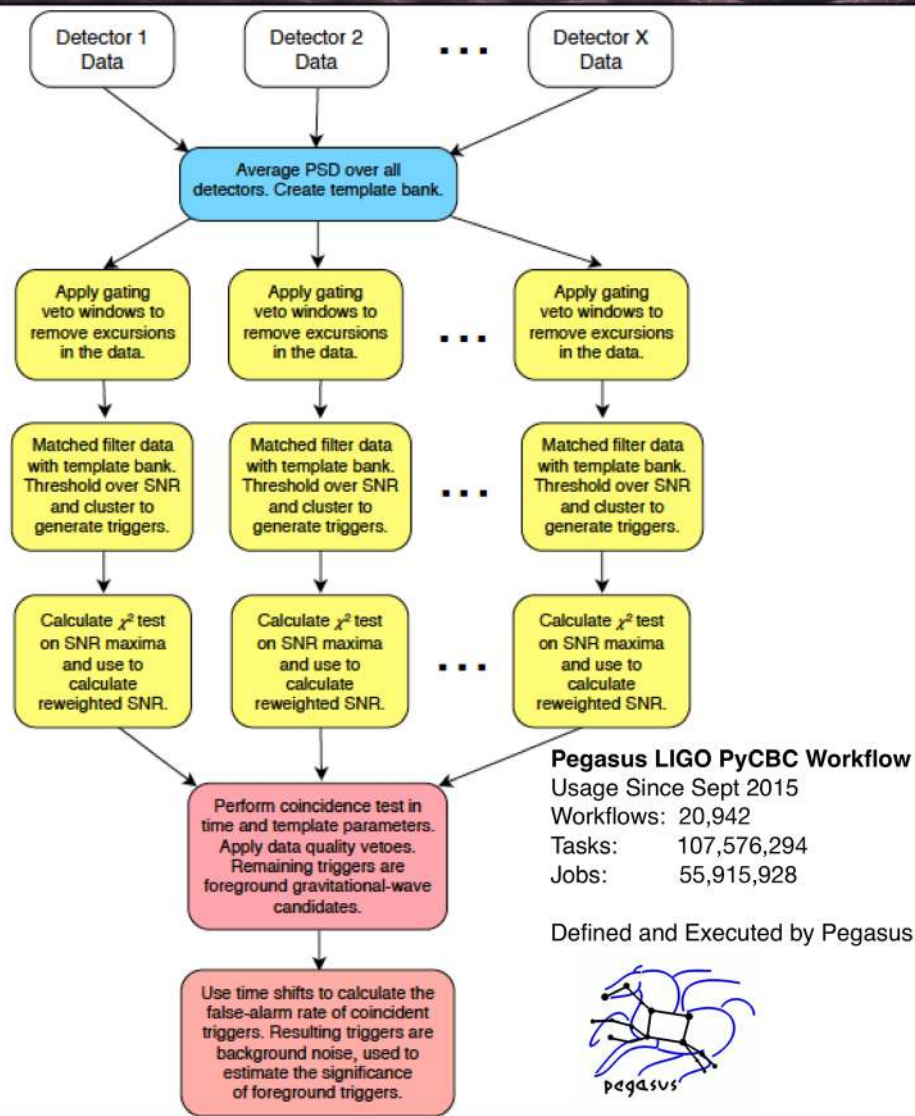
**Select data based
on metadata**

**Register data with
metadata**



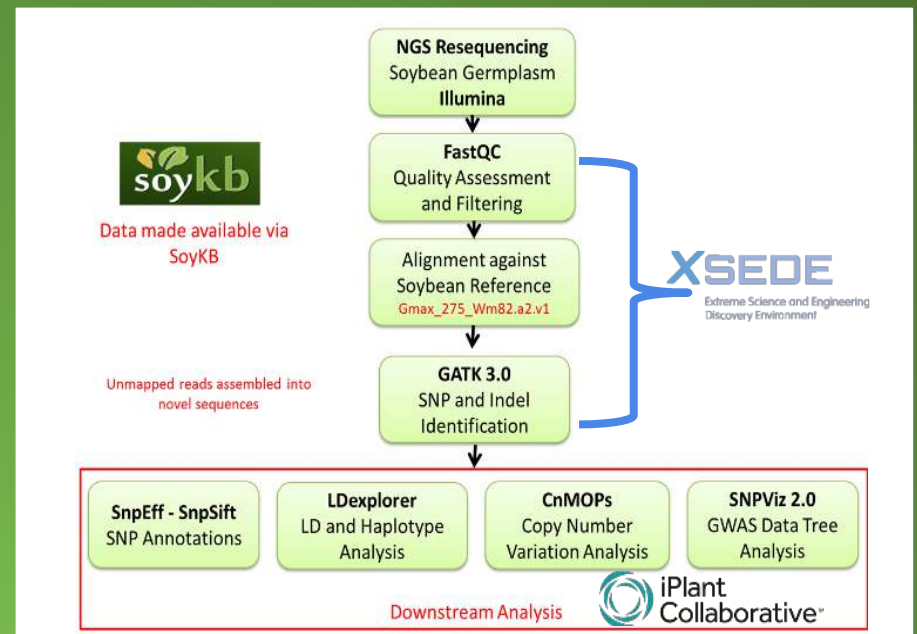
Pegasus' flow at a glance





Advanced LIGO PyCBC Workflow

- One of the main pipelines to measure the statistical significance of data needed for discovery.
- Contains 100's of thousands of jobs and accesses on order of terabytes of data.
- Uses data from multiple detectors.
- For the detection, the pipeline was executed on Syracuse and Albert Einstein Institute Hannover
- A single run of the binary black hole + binary neutron star search through the O1 data (about 3 calendar months of data with 50% duty cycle) requires a workflow with 194,364 jobs. Generating the final O1 results with all the review required for the first discovery took about 20 million core hours



<http://soykb.org>

XSEDE Allocation

PI: Dong Xu

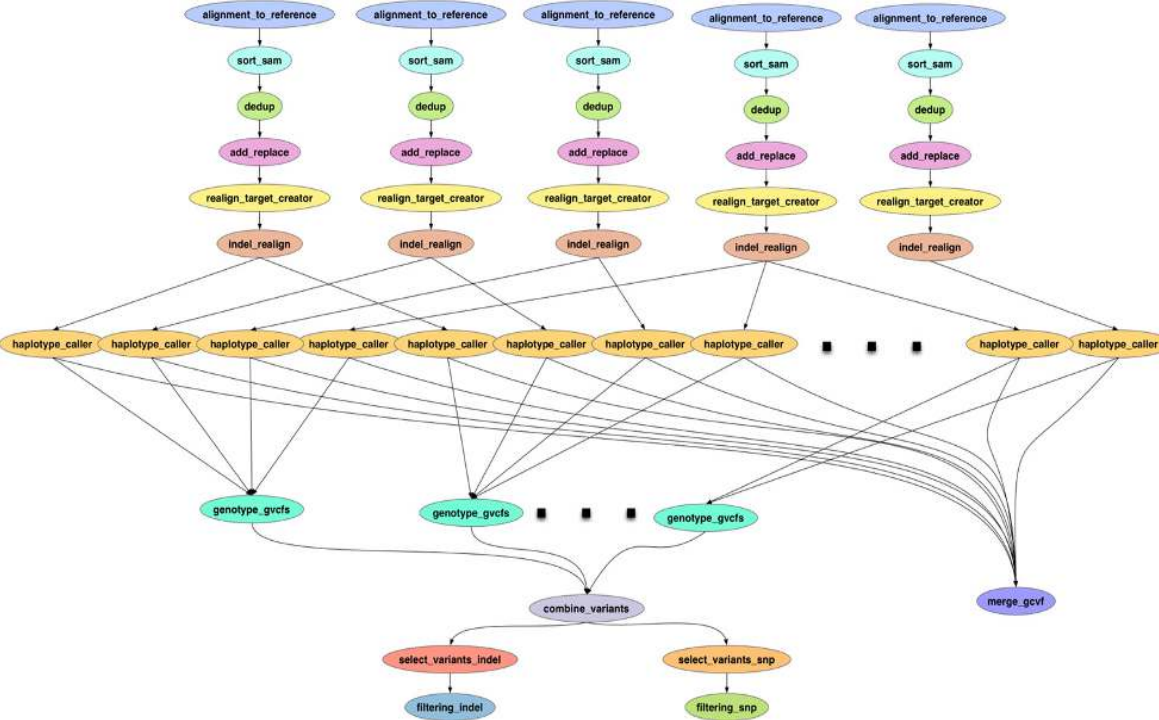
Trupti Joshi, Saad Kahn, Yang Liu, Juexin Wang, Badu Valliyodan, Jiaojiao Wang

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



Pegasus

<http://pegasus.isi.edu>



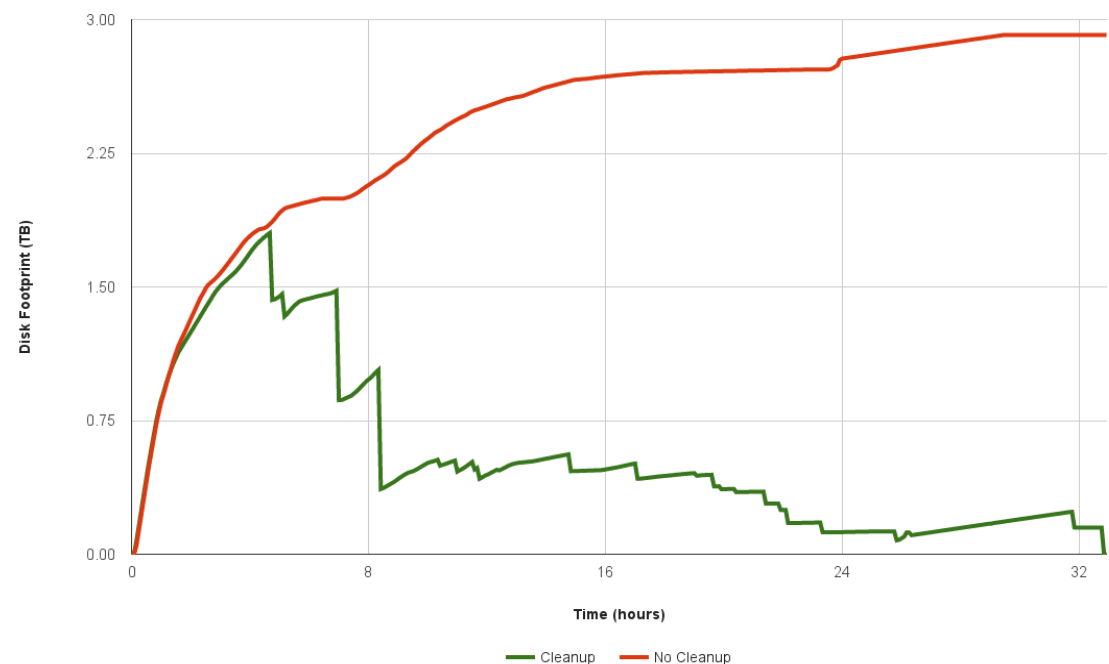
Task	Base Code	Cores (Threads)	Memory (GB)
Alignment_to_reference	BWA	7	8
Sort_sam	Picard	1	21
Dedup	Picard	1	21
Add_replace	Picard	1	21
Realign_target_creator	GATK	15	10
Indel_realign	GATK	1	10
Haplotype_caller	GATK	1	3
Genotype_gvcfs	GATK	1	10
Merge_gvcf	GATK	10	20
Combine_variants	GATK	1	10
Select_variants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

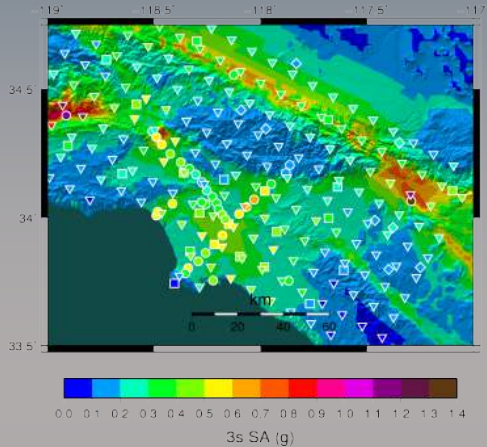
Works well on TACC Wrangler due to more cores and memory per node (48 cores, 128 GB RAM)



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)



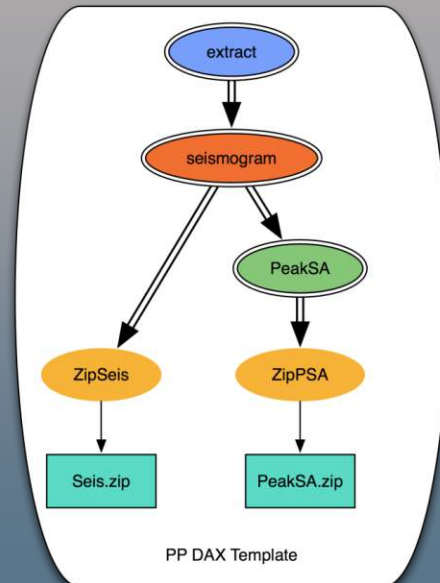
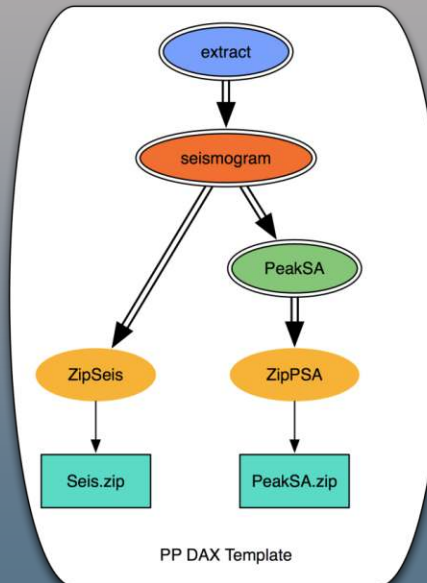
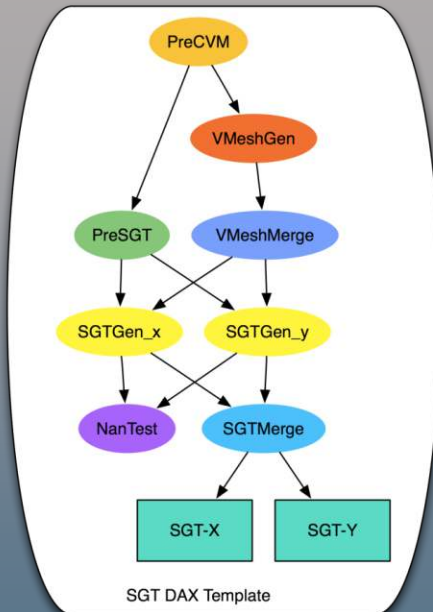
286 sites, 4
models
each workflow
has 420,000
tasks



CyberShake DAX



PP_DAX



Upcoming Features

4.8 – Late summer 2017

- Containers
- Jupyter

4.9 – Early 2018?

- Data integrity



Pegasus

est. 2001

Automate, recover, and debug scientific computations.

Get Started

Pegasus Website

<https://pegasus.isi.edu>

Users Mailing List

pegasus-users@isi.edu

Support

pegasus-support@isi.edu

HipChat

