Challenges of Managing Scientific Workflows in High-Throughput and High-Performance Computing Environments

Ewa Deelman, Ph.D.

USC Viterbi
School of Engineering
Information Sciences Institute

http://pegasus.isi.edu

LIGO Experiment: Searching for Gravitational Waves

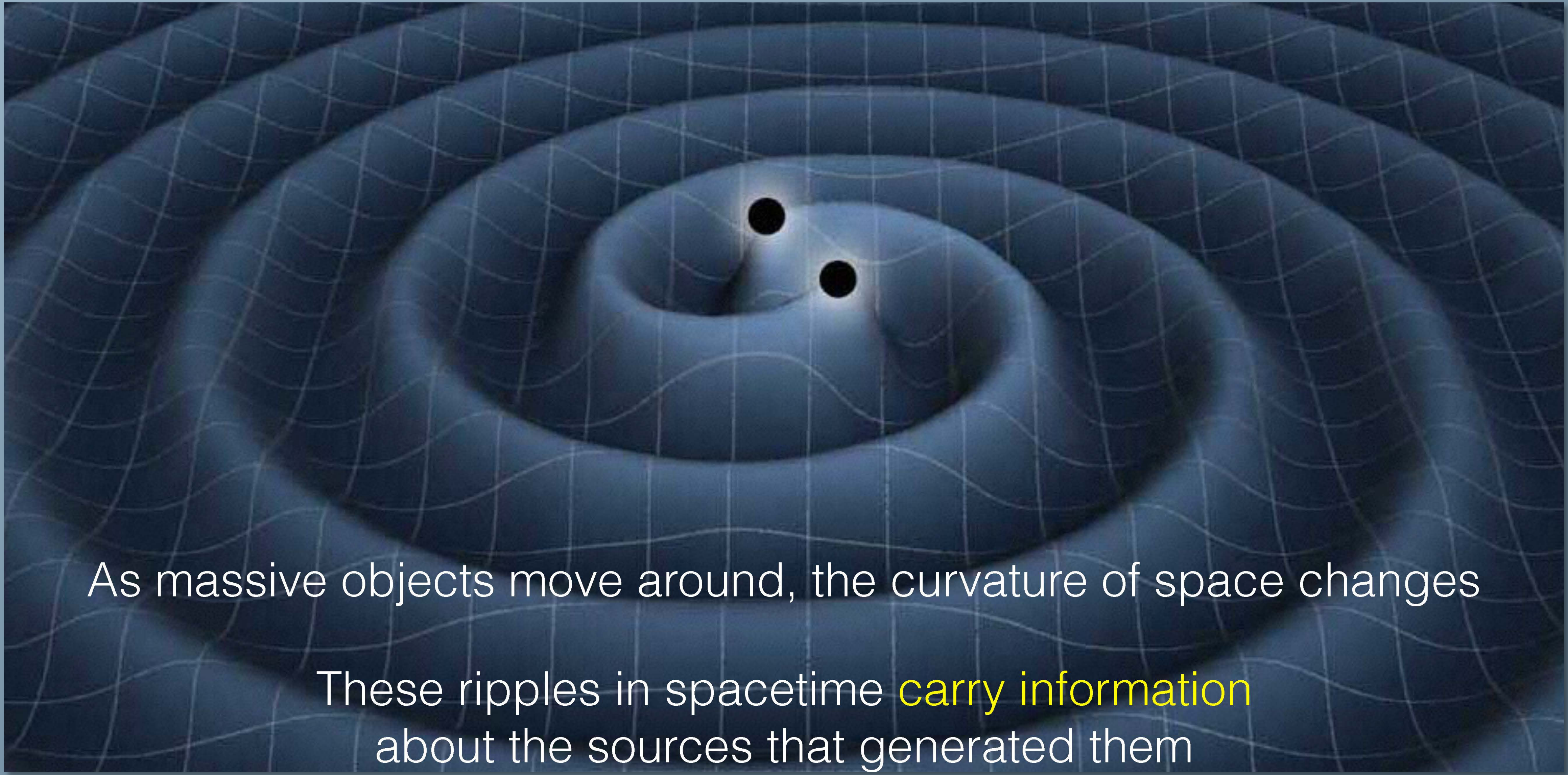


Image courtesy of LSC

LIGO (Laser Interferometer Gravitational-Wave Observatory)

LSC (LIGO Scientific Collaboration)

- Collaboration involved in research of the data coming out of the detectors
- 1000 scientists from universities in US and 14 other countries
- 250 students
- Responsible for developing analysis methodologies and detector technology.

Background

- Largest ever NSF funded project
- Two 4km long detectors in the US (Hanford, Washington, and Livingston, Louisiana)

Aerial View of the LIGO Livingston Laboratory
Image Credit: Caltech/MIT/LIGO Lab

Phase I

(Initial LIGO 2002 – 2010)

- No gravitational waves detected
- But a lot of analysis pipelines and computing infrastructure were setup
- **Late 2010 – Passed Blind Injection Test**

Upgrade of the detectors

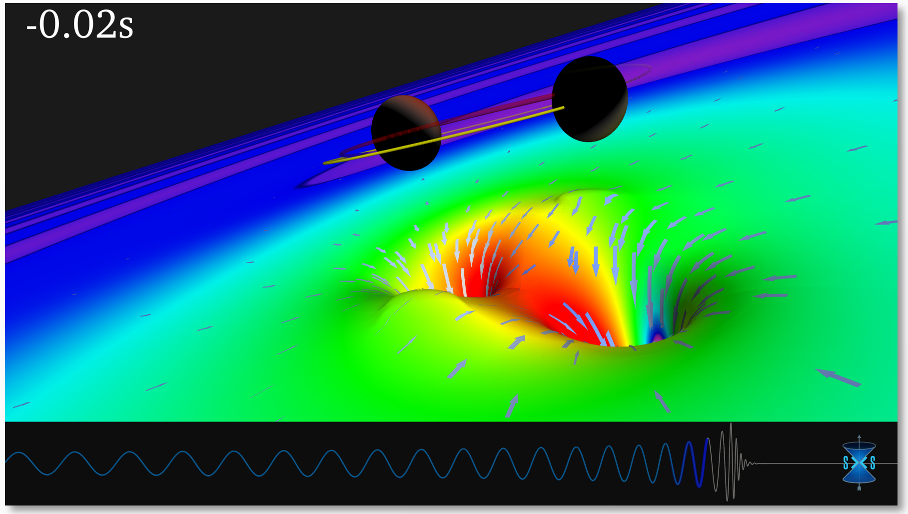
- Designed to be 10 times more sensitive than Phase I

Phase II

- Currently operating at 4 times the Initial LIGO sensitivity

(Advanced LIGO September 2015 onwards)

LIGO's Gravitational Wave Detection



LIGO announced first ever detection of gravitational waves *Feb 2016*

Created as a result of coalescence of a pair of dense, massive black holes

Confirms major prediction of **Einstein Theory of Relativity**

Detection Event

Detected by both of the operational Advanced LIGO detectors
(4km long L shaped interferometers)

Event occurred at September 14, 2015 at 5:51 a.m. Eastern Daylight Time

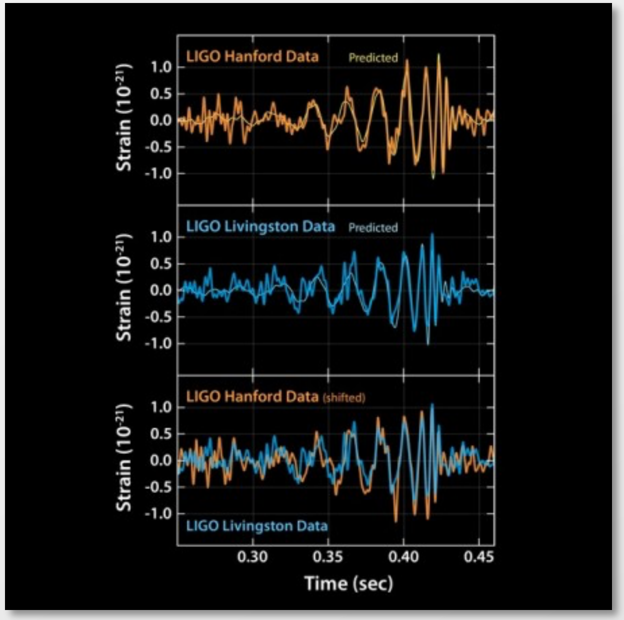
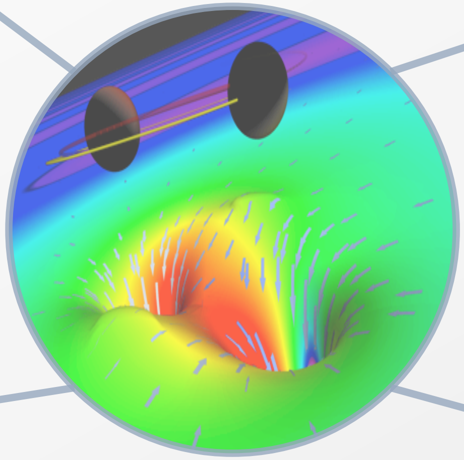


Image Credits: *0.2 Second before the black holes collide: SXS/LIGO*
Signals of Gravitational Waves Detected: Caltech/MIT/LIGO Lab

LIGO Detection

Behind the Scenes

A variety of complex analysis pipelines were executed



Some were low latency that initially alerted people to look at a specific piece of data containing the signal

However, to verify that signal is a valid candidate

- *A large amount of data needed to be analyzed*
- *Statistical significance of the detection should be at 5-sigma level*

Pipelines/Workflows are mainly executed on LSC Data Grid

- *Consists of approximately 11 large clusters at various LIGO institutions and affiliates*
- *Data is replicated at sites in the US and Europe*
- *Each cluster has Grid middleware and HTCondor installed*
- *GridFTP used for data transfers*

Advanced LIGO PyCBC Workflow

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

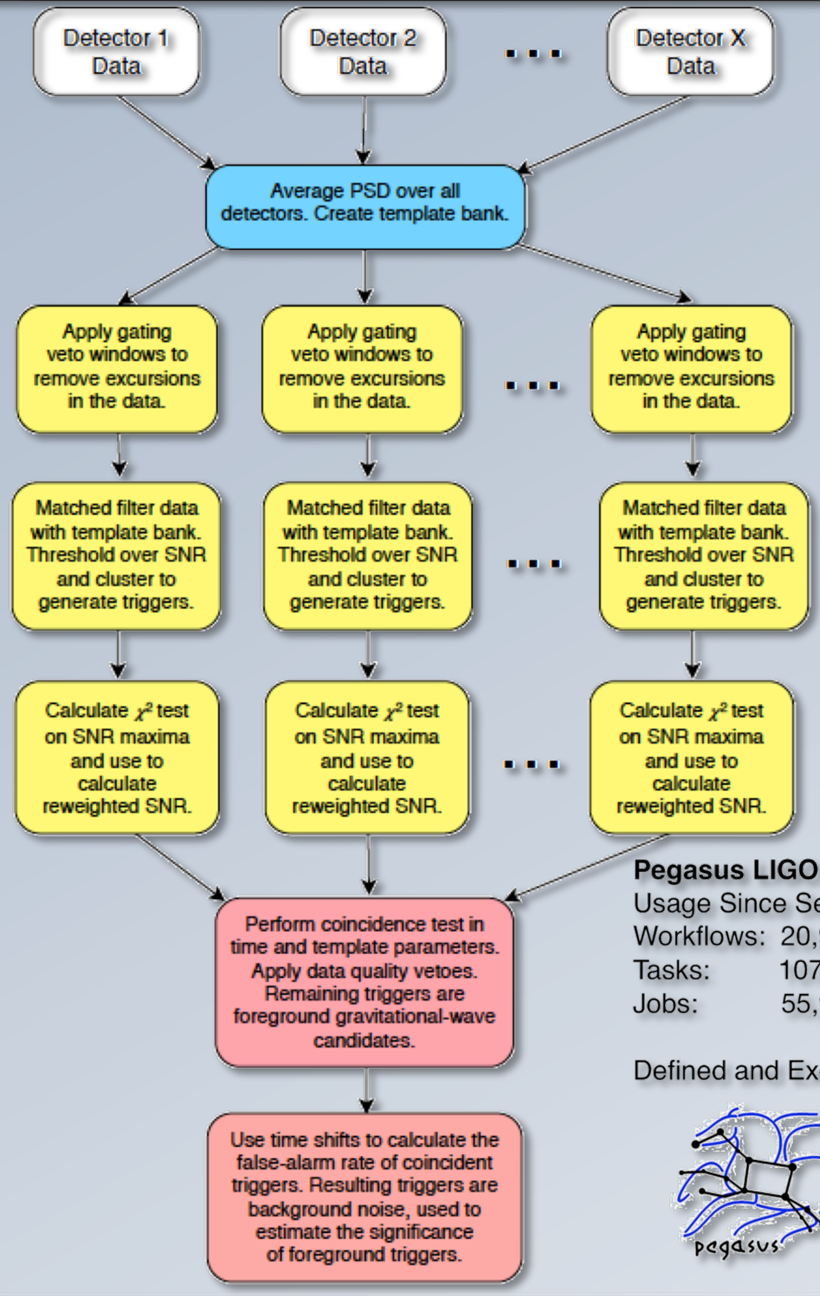
One of the main pipelines to measure the statistical significance of data needed for discovery

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

Use our Pegasus software to automate the execution of tasks and data access



Pegasus LIGO PyCBC Workflow

Usage Since Sept 2015
Workflows: 20,942
Tasks: 107,576,294
Jobs: 55,915,928

Defined and Executed by Pegasus

Outline

1. Example Pegasus Workflows
2. Pegasus Workflow Management System
3. Challenges and solutions for workflow execution in clusters, distributed systems, and clouds
4. Future Directions

Why Scientific Workflows?

Automates complex, multi-stage processing pipelines

Enables parallel, distributed computations

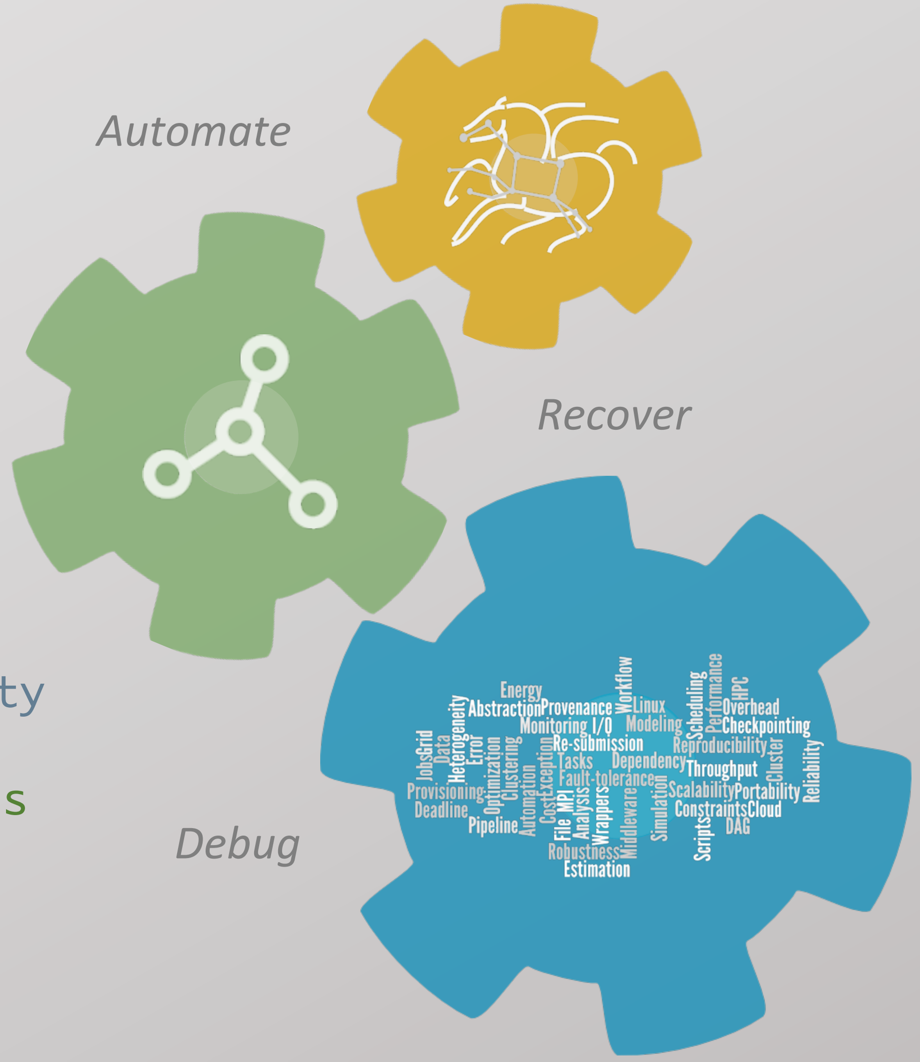
Automatically executes data transfers

Reusable, aids reproducibility

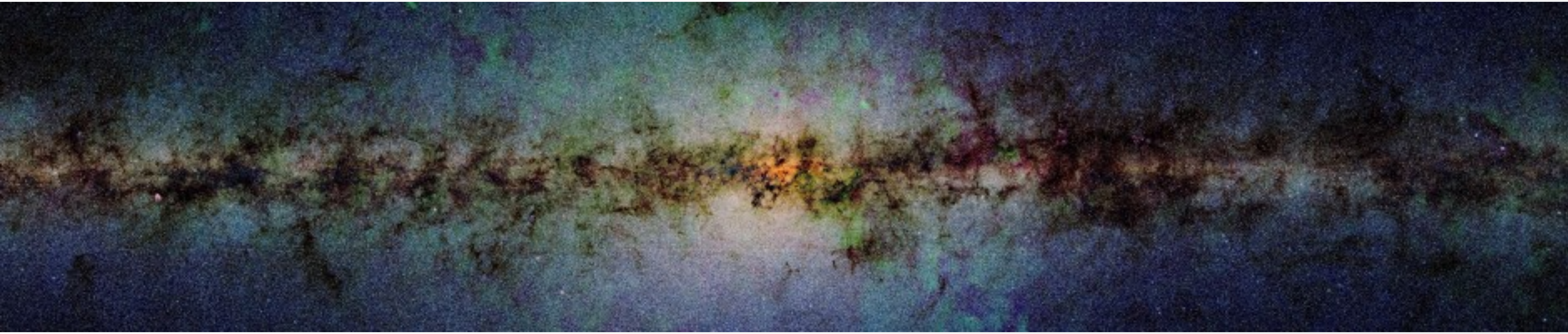
Records how data was produced (provenance)

Handles failures with to provide reliability

Keeps track of data and files



Community Archives: Galactic Plane Atlas



- 18 million input images (~2.5 TB)
- 900 output images (2.5 GB each, 2.4 TB total)

} × 17

- Measuring the global star formation rate in the galaxy
- Studying the energetics of the interaction of molecular clouds with the interstellar medium
- Determining whether coagulation or fragmentation governs the formation of massive stars
- Assessing the supernova rate in the Galaxy

International Rice Research Institute

Liya Wang

IRRI

The 3000 rice genome: ~15.4 TBs of 83 bp paired-end genomic reads for a diverse set of rice varieties

Want to capture genetic differences using well known bioinformatics applications

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

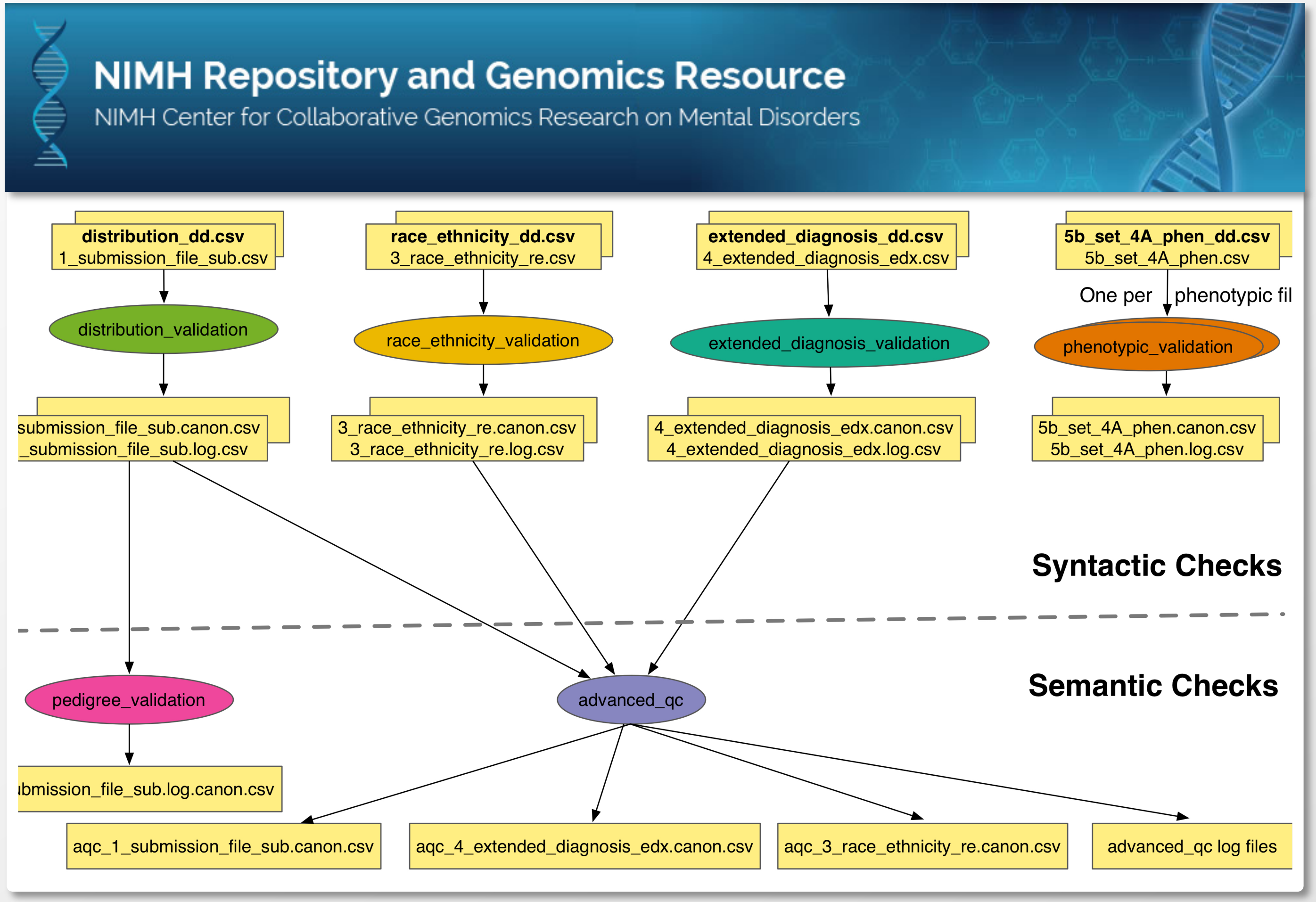
HTCondor
High Throughput Computing

Workflow supports 3 different execution environments

- SDSC Comet (glideins)
- TACC Stampede (pegasus-mpi-cluster)
- Distributed (local HTCondor pool)

Sometimes you want to "hide" the workflow…

Automated QC Workflow, Pedigree checks, Consistency checks



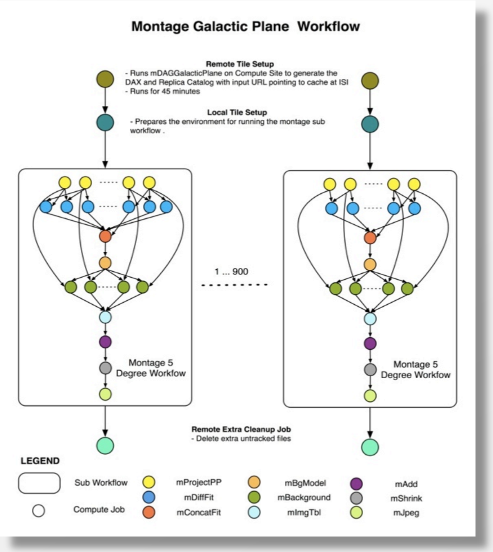
Outline

1. Example Pegasus Workflows
2. Pegasus Workflow Management System
3. Challenges and solutions for workflow execution in clusters, distributed systems, and clouds
4. Future Directions

Sometimes the environment is complex…

Data Storage

Local Resource

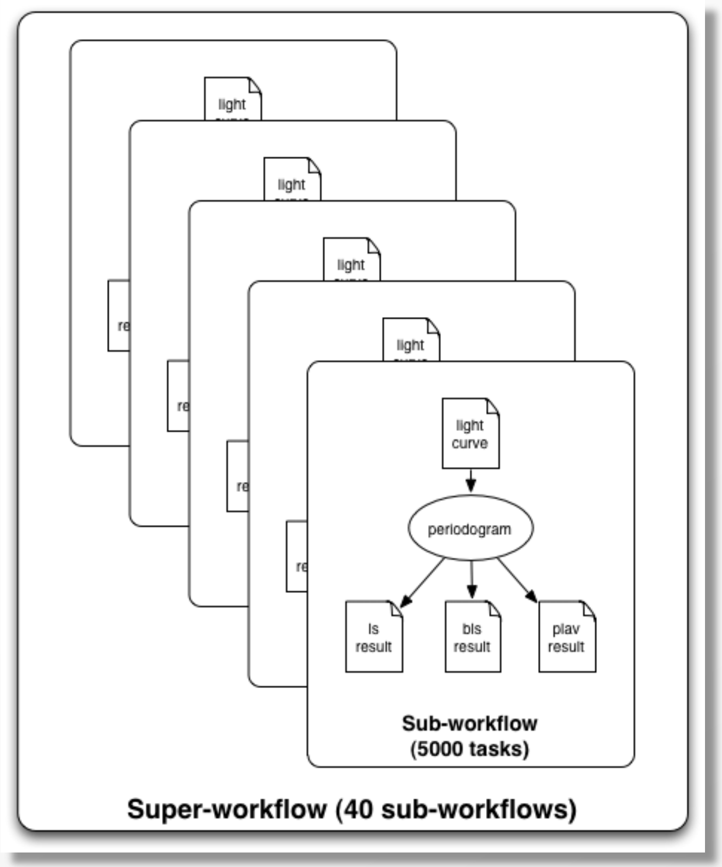


Work Definition

Campus Cluster | XSEDE | NERSC | ALCF | OLCF | OSG | Chameleon | Amazon Cloud

Sometimes the environment is just not exactly right

Single core workload

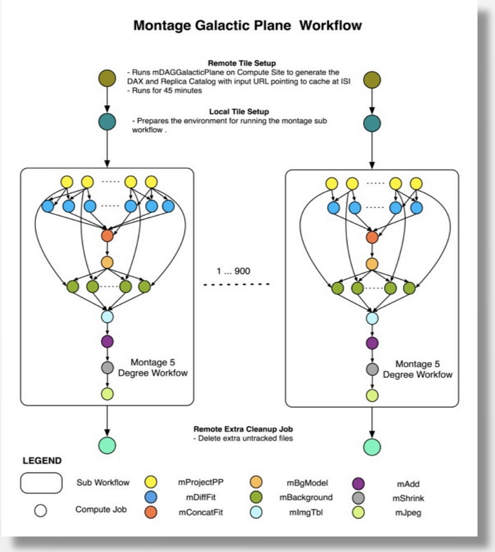


Cray XK7 System Environment / Designed for MPI codes

Sometimes you want to change or combine resources

Data Storage

Local Resource



data

Work Definition

work

You don't want to recode your workflow

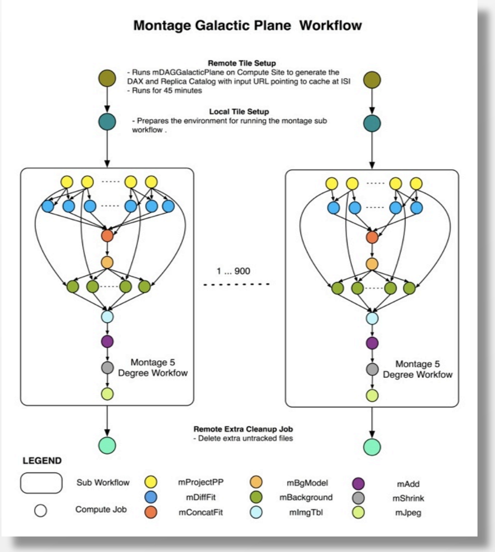
Campus Cluster	XSEDE	NERSC	ALCF	OLCF	OSG	Chameleon	Amazon Cloud

Our Approach: Submit locally, Compute globally

Data Storage

Local Resource

Workflow Management System



data

Work Definition

work

Campus Cluster | XSEDE | NERSC | ALCF | OLCF | OSG | Chameleon | Amazon Cloud

Workflow Management System (WMS) Functions

Discover what resources (computation, data, software) are available

Select the appropriate resources based on a architecture, availability of software, performance, reliability, availability of cycles, storage,..

Devise a plan

– What resources to use
– How to best adapt the workflow to the resources
– What protocols to use to access the data,
to schedule jobs
– What data to save

Execute the plan

– In a reliable way
– Keep track of what data was accessed,
generated and how

Currently outside of the WMS functions

– Resource provisioning

Pegasus Workflow Management System (est. 2001)

Collaboration with HTCondor, UW Madison

A workflow "compiler"/planner

– Input: abstract workflow description, resource-independent

– Auxiliary Info (catalogs): available resources, data, codes

– Output: executable workflow with concrete resources

– Automatically locates physical locations for both workflow tasks and data

– Transforms the workflow for performance and reliability

A workflow engine (DAGMan)

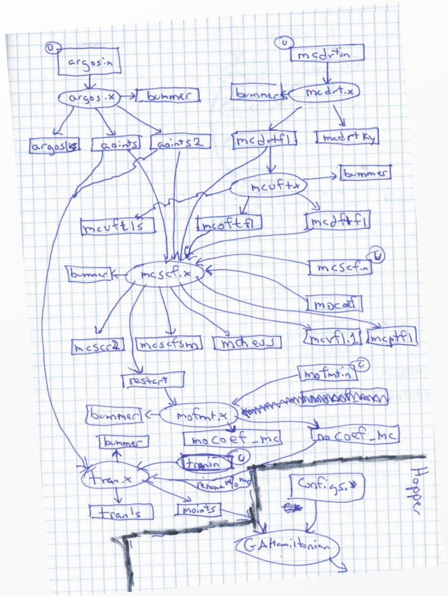
– Executes the workflow on local or distributed resources (HPC, clouds)

– Task executables are wrapped with *pegasus-kickstart* and managed by *Condor schedd*

Provenance and execution traces are collected and stored

Traces and DB can be mined for performance and overhead information

Pegasus provides tools to generate the abstract workflow



python Java R perl

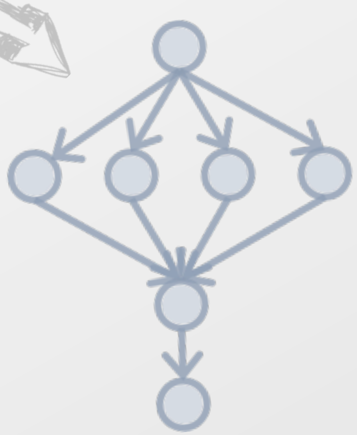
API

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

from Pegasus.DAX3 import *
import sys
import os

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

# Add the hello job
hello = Job(namespace="hello_world",
            name="hello", version="1.0")
b = File("f.b")
hello.uses(a, link=Link.INPUT)
hello.uses(b, link=Link.OUTPUT)
dax.addJob(hello)

# Add the world job (depends on the hello job)
world = Job(namespace="hello_world",
            name="world", version="1.0")
c = File("f.c")
world.uses(b, link=Link.INPUT)
world.uses(c, link=Link.OUTPUT)
dax.addJob(world)

# Add control-flow dependencies
dax.addDependency(Dependency(parent=hello,
                             child=world))

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

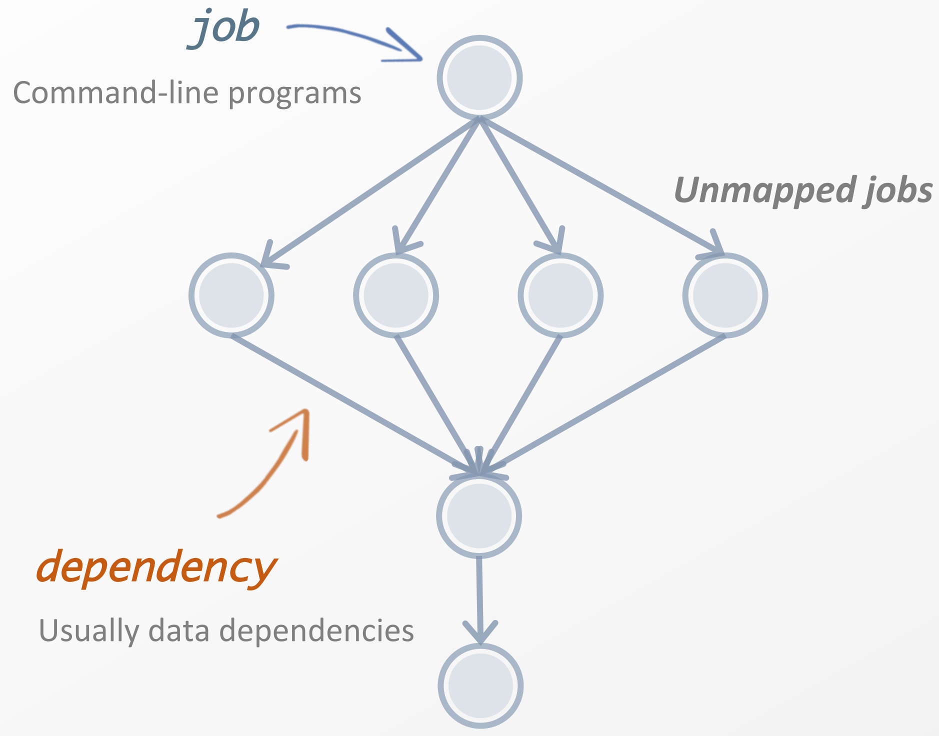
```xml
<?xml version="1.0" encoding="UTF-8"?>

<!-- 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>
```

DAX

DAG in XML

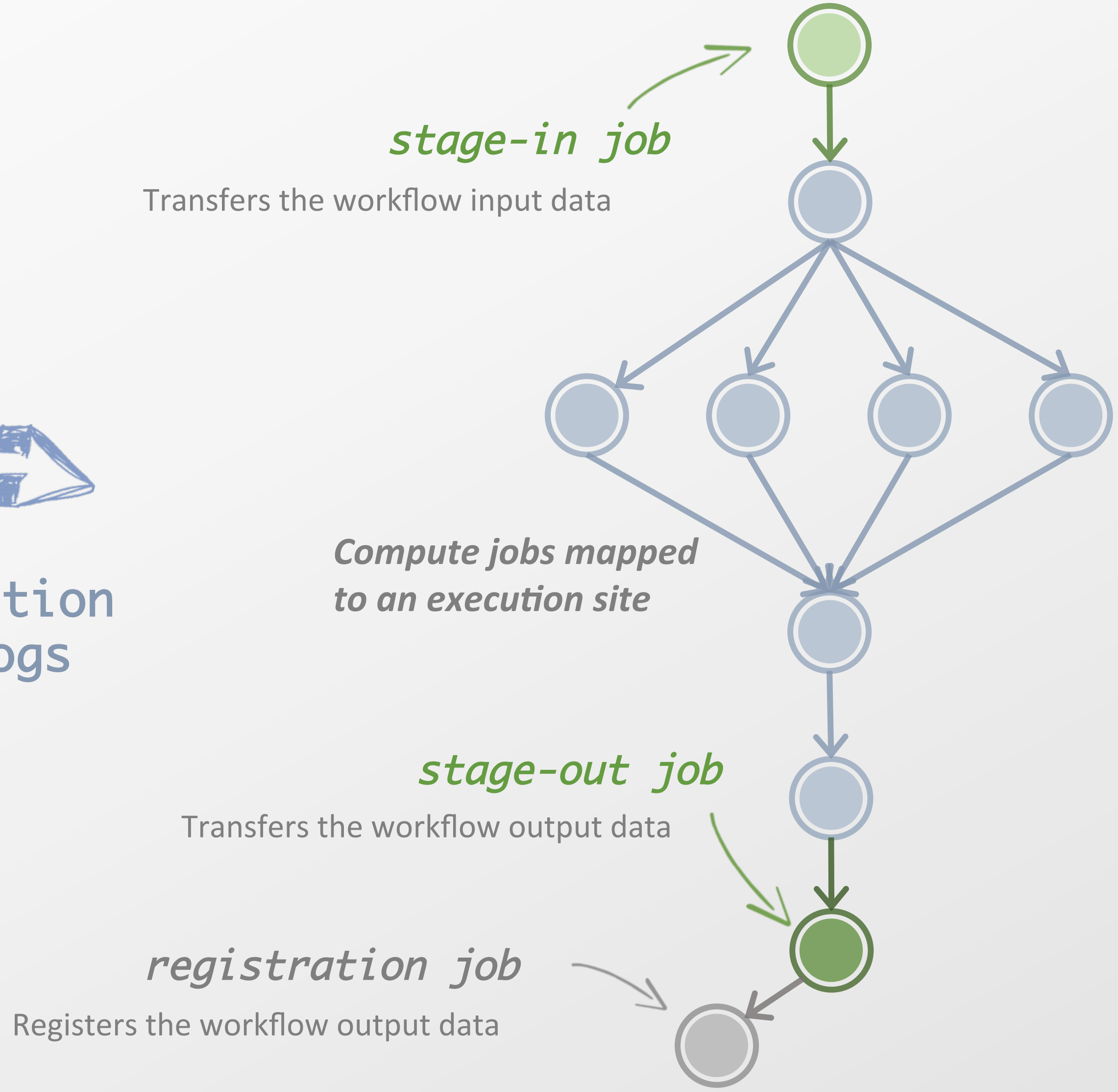


From the abstraction to execution!

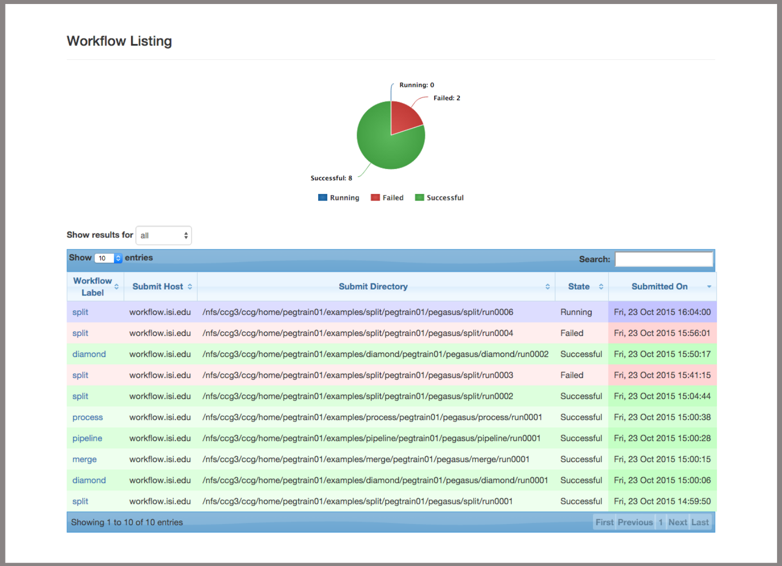


abstract workflow

Information Catalogs

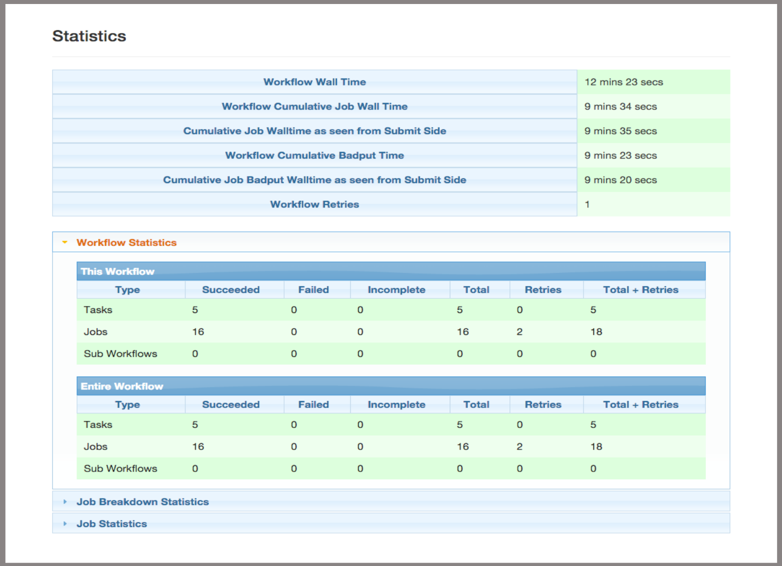


executable workflow

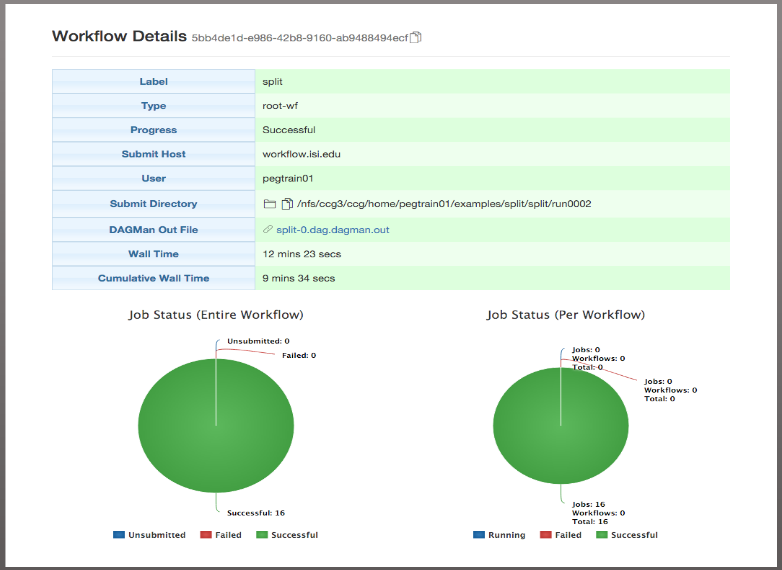


Pegasus dashboard

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

Tools to calculate job statistics

Workflow makespan, Cumulative time

Task Type	Count	Runtime(s)	IO Read (MB)	IO Write (MB)	Memory Peak(MB)	CPU Utilization(%)
mProjectPP	2102	1.73	2.05	8.09	11.81	86.96
mDiffFit	6172	0.66	16.56	0.64	5.76	28.39
mConcatFit	1	143.26	1.95	1.22	8.13	53.17
mBgModel	1	384.49	1.56	0.10	13.64	99.89
mBackground	2102	1.72	8.36	8.09	16.19	8.46
mImgtbl	17	2.78	1.55	0.12	8.06	3.48
mAdd	17	282.37	1102	775.45	16.04	8.48
mShrink	16	66.10	412	0.49	4.62	2.30
mJPEG	1	0.64	25.33	0.39	3.96	77.14

Execution profile of the Montage workflow, averages calculated

Outline

1. Example Pegasus Workflows
2. Pegasus Workflow Management System
3. Challenges and solutions for workflow execution in clusters, distributed systems, and clouds
4. Future Directions

Challenges in workflow execution

Failures in the execution environment or application

Data storage limitations on execution sites

Performance

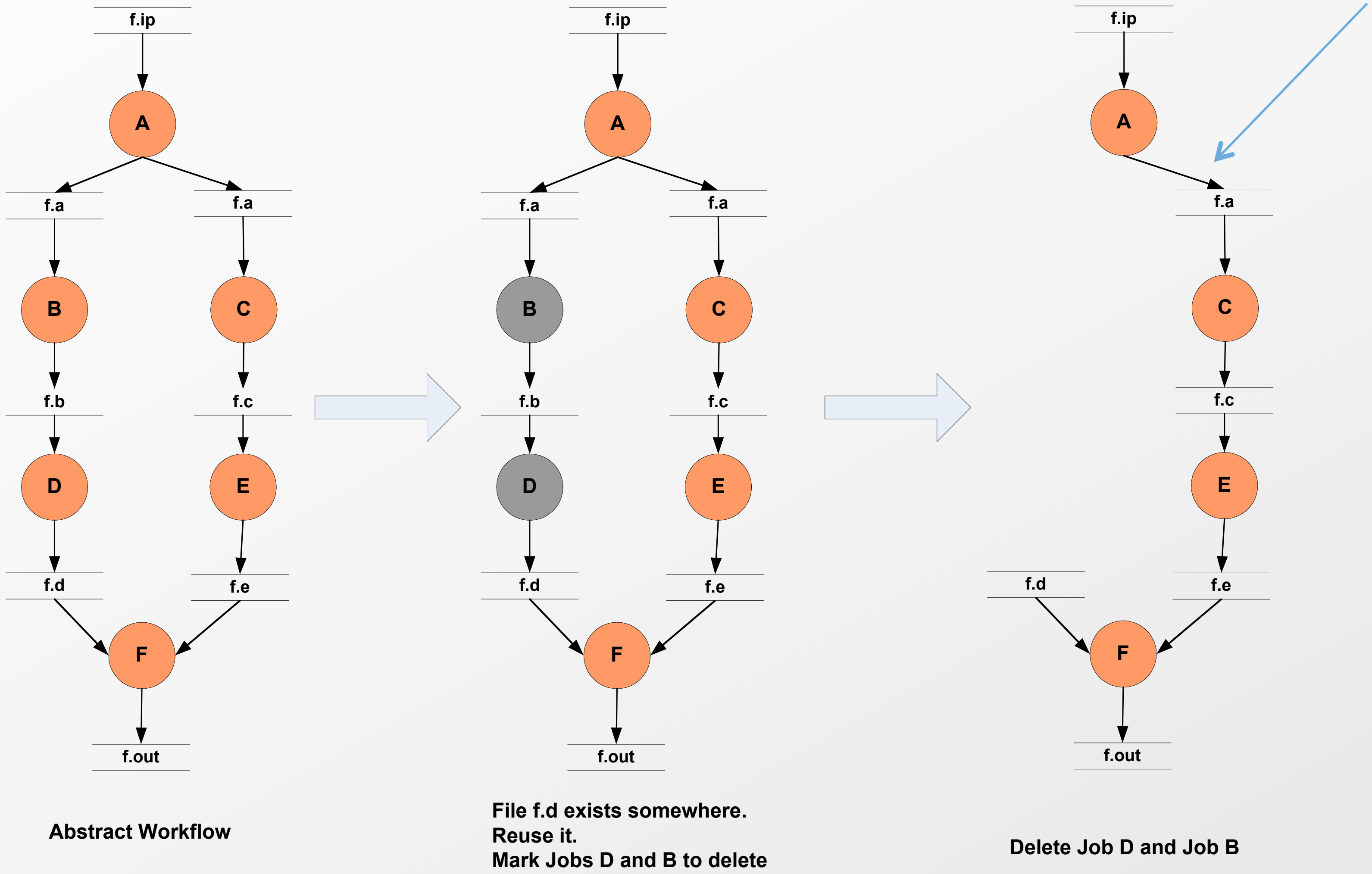
– Small workflow tasks

Heterogeneous execution architectures

– Different file systems (shared/ non-shared)
– Different system architectures (Cray XT, Blue Gene, …)
– Mismatch between tasks and architectures

Sometimes fatal errors occur during workflow execution

Want to restart the workflow from where it left off
Sometimes intermediate data is already available



- *workflow reduction*
- *data reuse*
- *workflow-level checkpointing*

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)

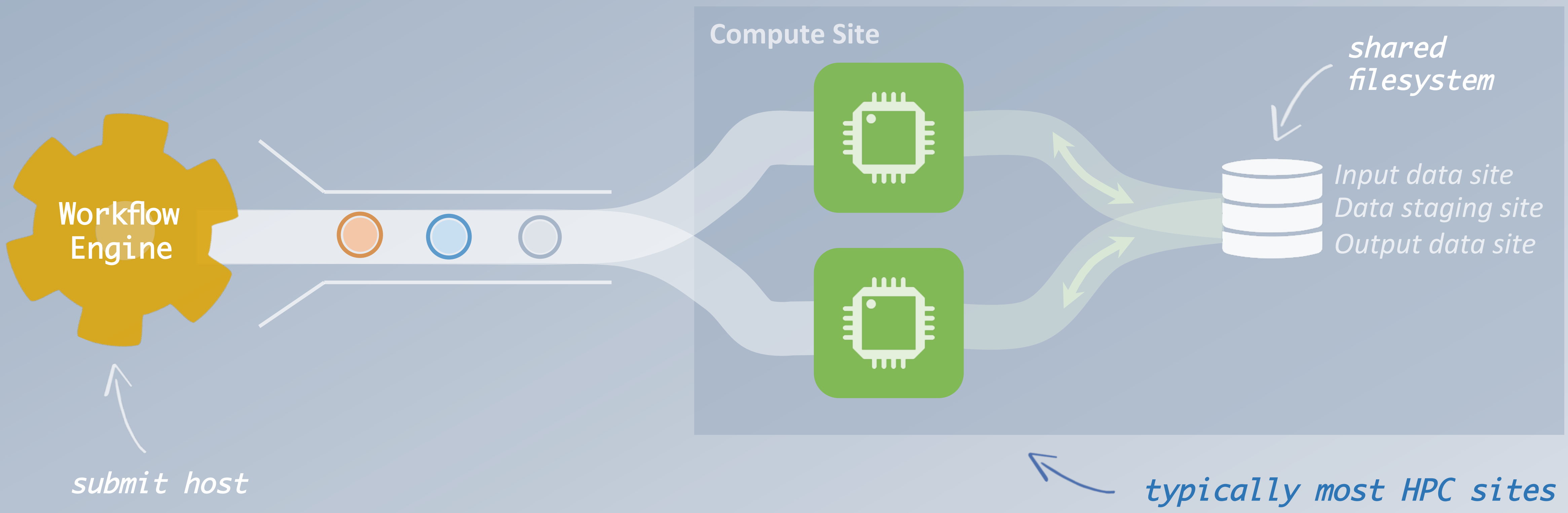
- Worker nodes do not share a file system
- Data is pulled / pushed from a staging site, possibly not co-located with the computation

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

- I/O is directly against the shared file system

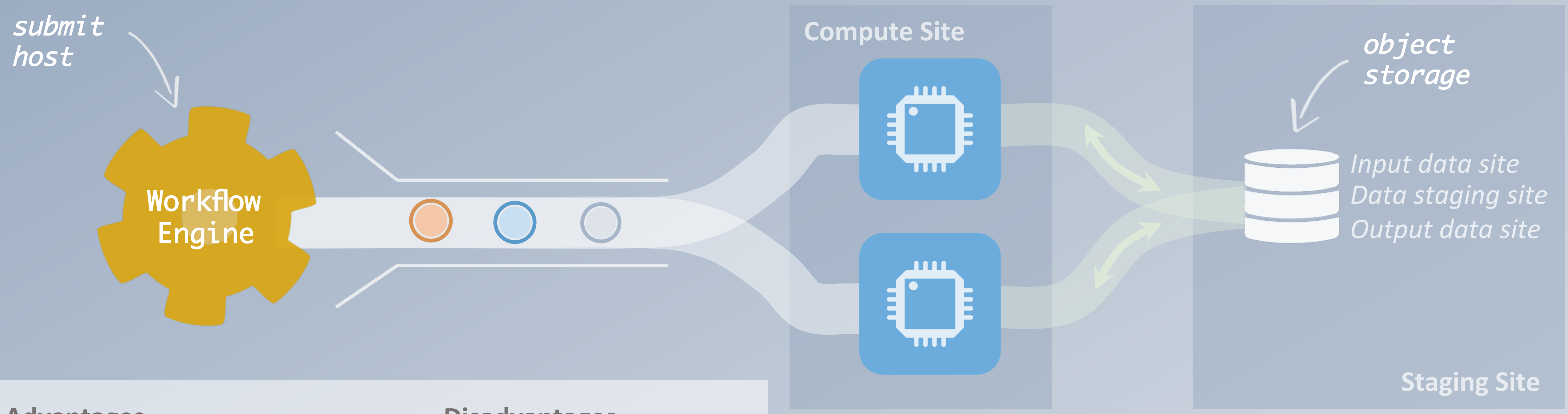
High Performance Computing

There are several possible configurations…



Cloud Computing

High-scalable object storages



Typical cloud computing deployment (Amazon S3, Google Storage)

Advantages

– Can leverage scalable stores

– Distribute computations across resources, such as supporting spillover from local resources to cloud resources

– Great bandwidth

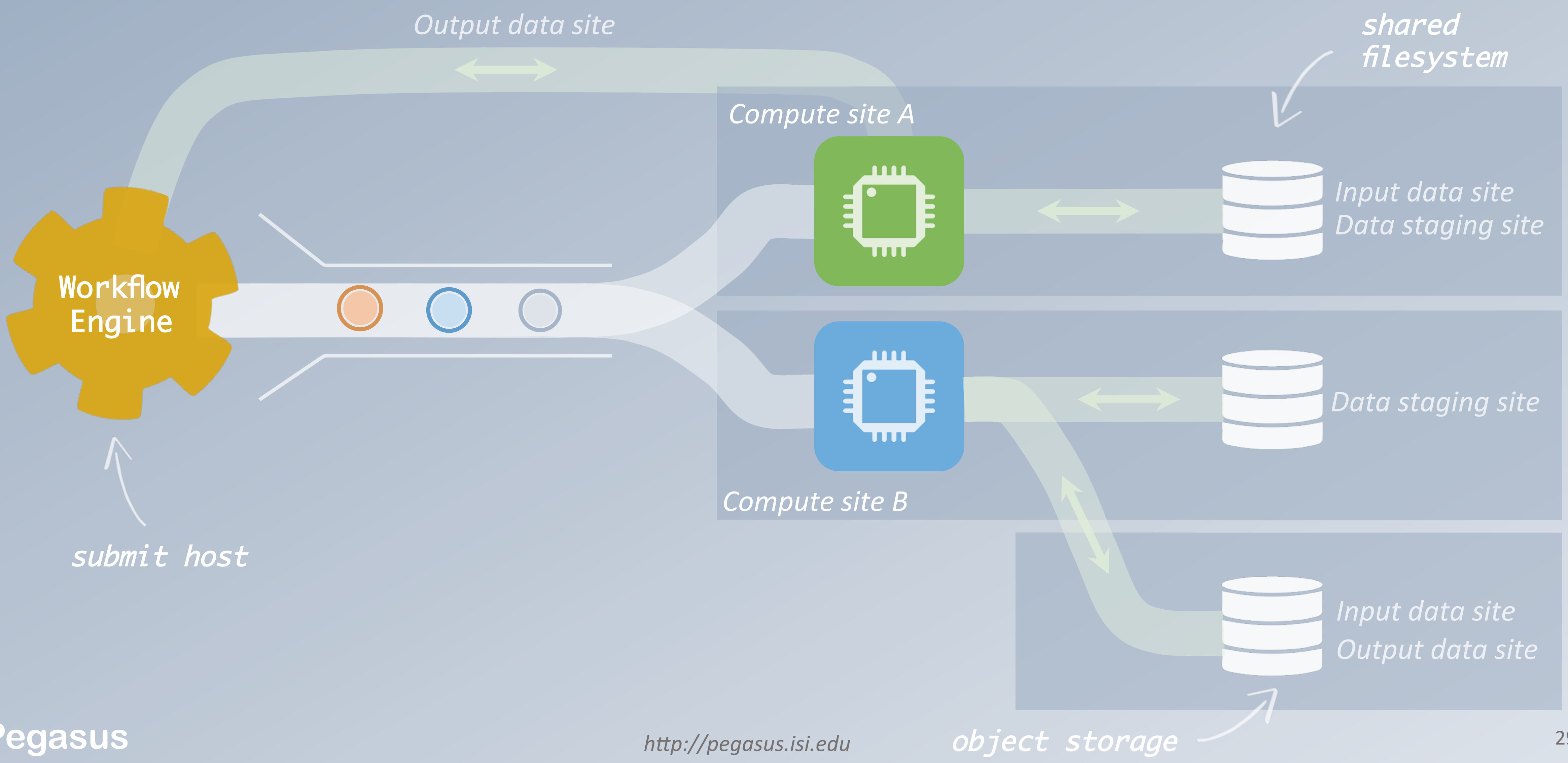
Disadvantages

– Duplicate Transfers

– Latencies in transferring large number of files

– Added costs for duplicate transfers

And yes… you can mix everything!



pegasus-transfer

subsystem for various storage systems

Pegasus' internal data transfer tool

Supports many different protocols

Directory creation, file removal
– If protocol supports, used for cleanup

Credential management
– Uses the appropriate credential for each site and each protocol (even 3rd party transfers)

Parallel transfers

Automatic retries

Checkpoint and restart transfers

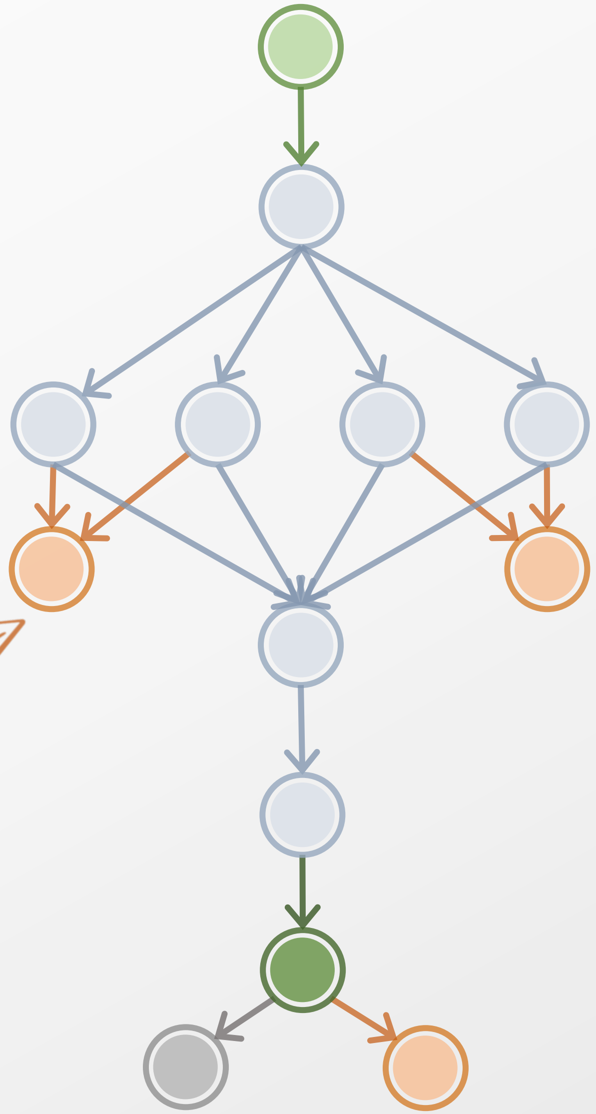
Two stage transfers
– e.g. GridFTP to S3 = GridFTP to local file, local file to S3

PROTOCOLS

HTTP SCP GridFTP iRods AmazonS3 Google Storage SRM FDT stashcp cp ln -s

Optimizing storage usage…

Small amount of space



cleanup job

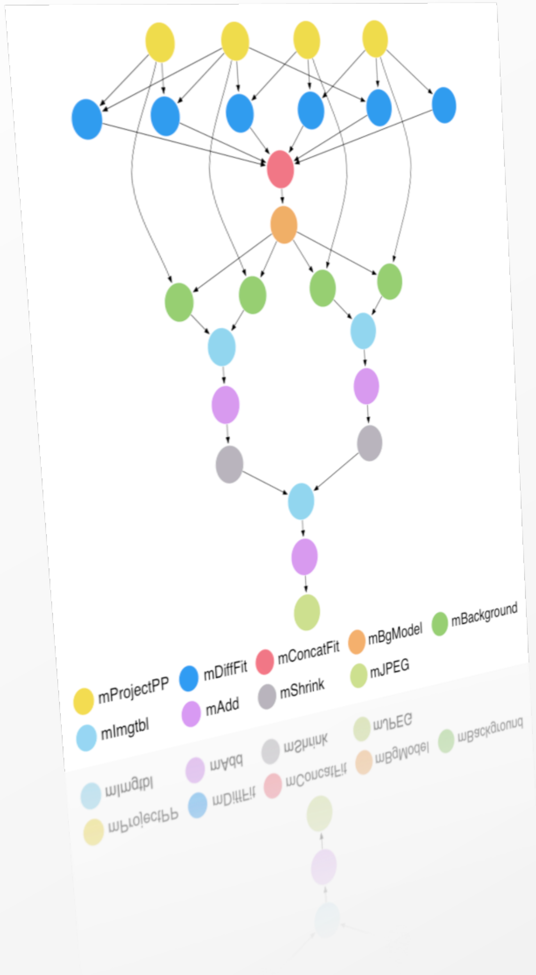
Removes unused data

Automatically add tasks to "clean up" data no longer needed

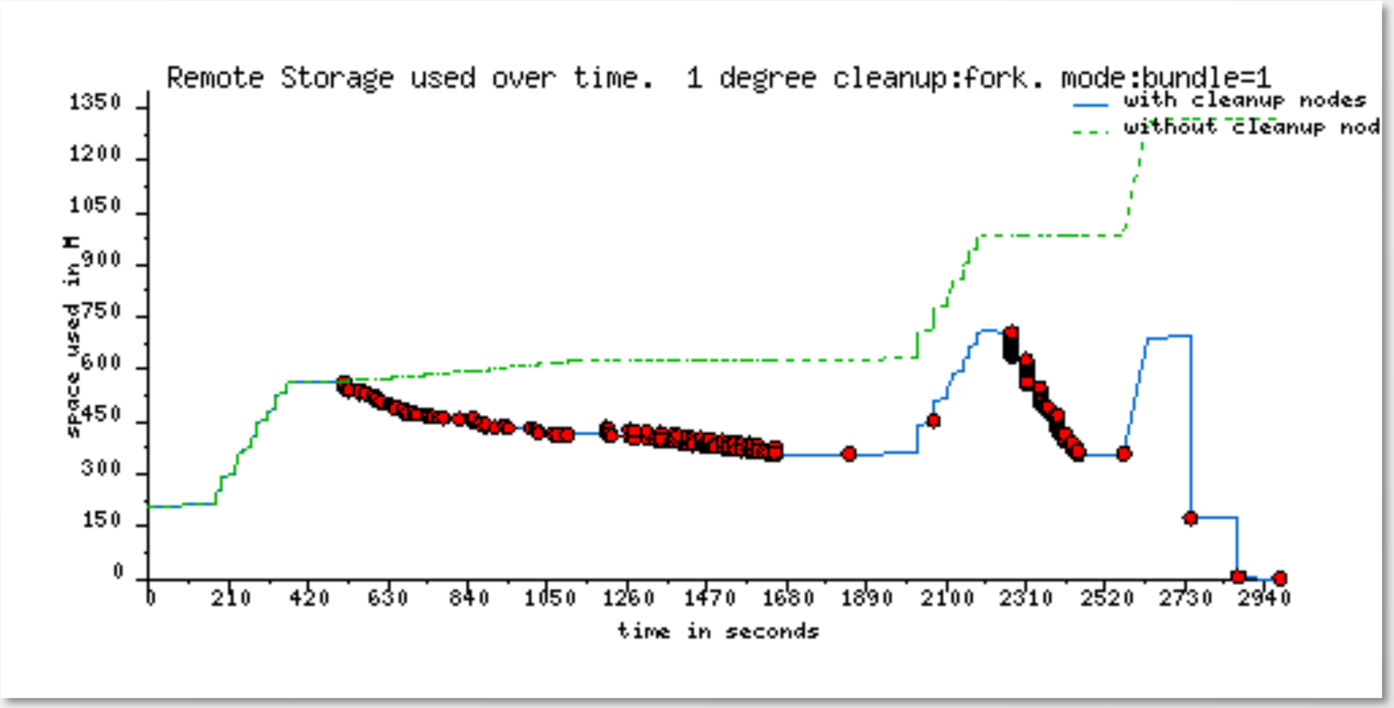
LIGO was running on Open Science Grid resources, processing TBs of data within a single workflow

Optimizing storage usage…

Example of the Montage Astronomy Workflow



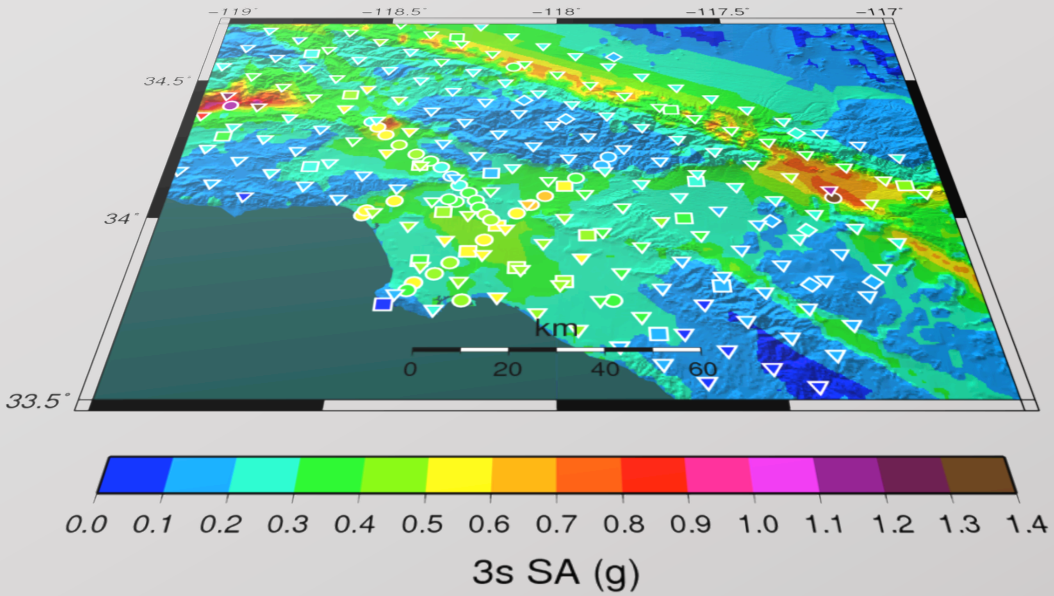
1.25GB versus 700 MB



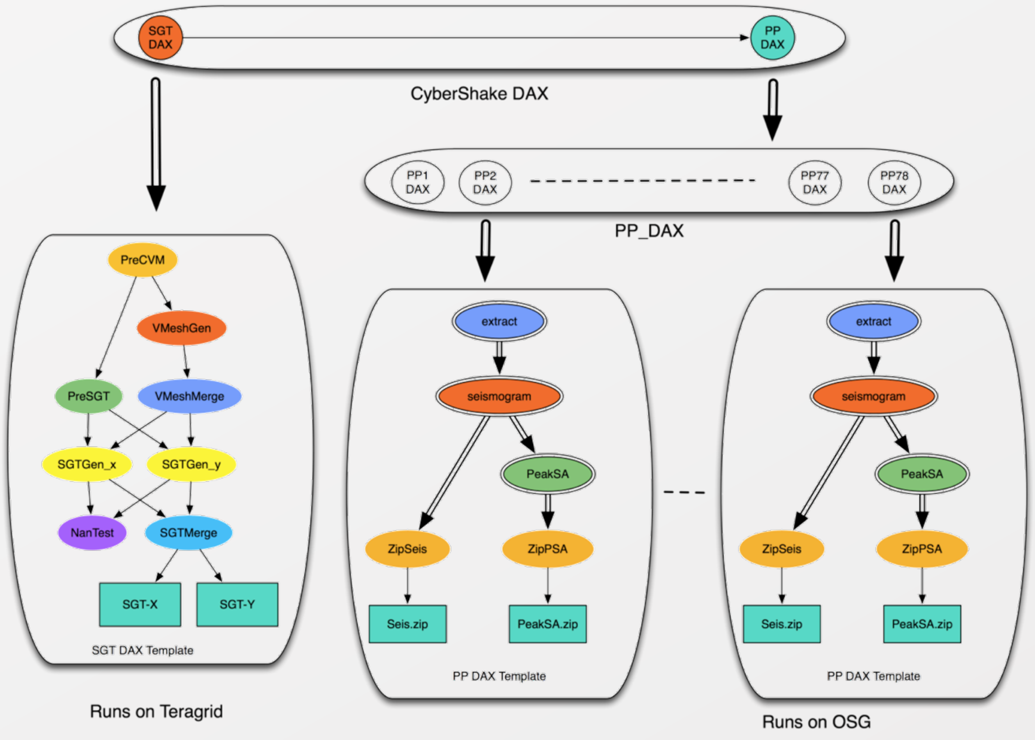
Southern California Earthquake Center's CyberShake PSHA Workflow

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)



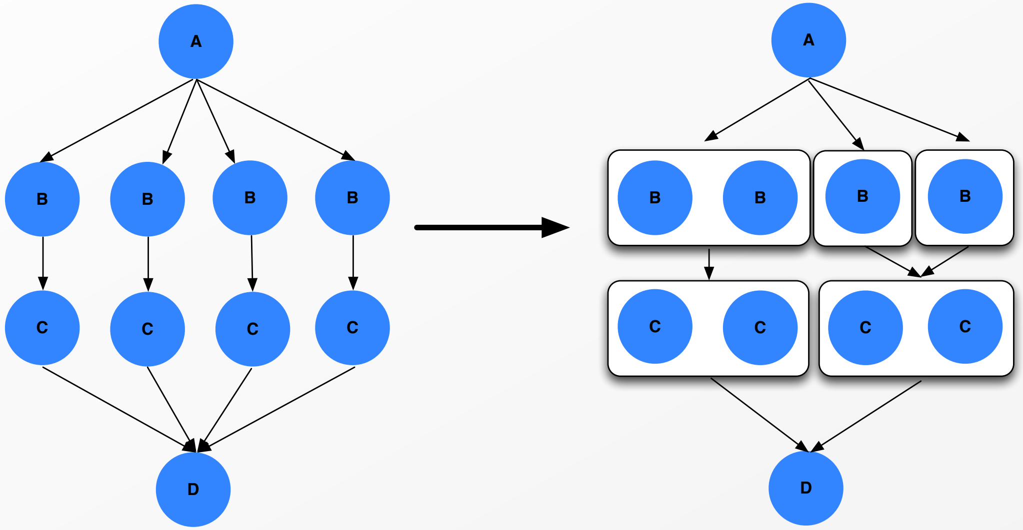
Workload does not match the infrastructure



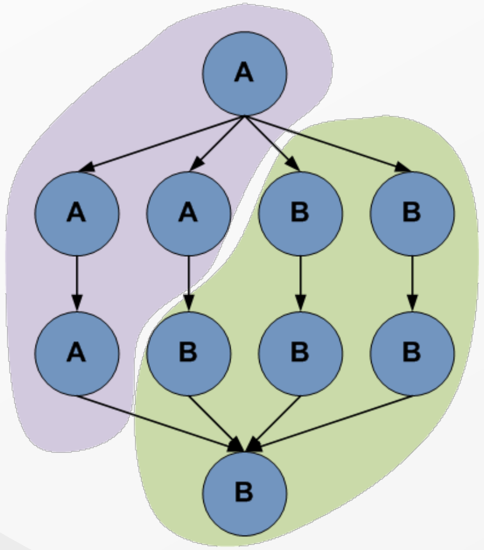
293 workflows
each workflow has 820,000 tasks

Solutions:

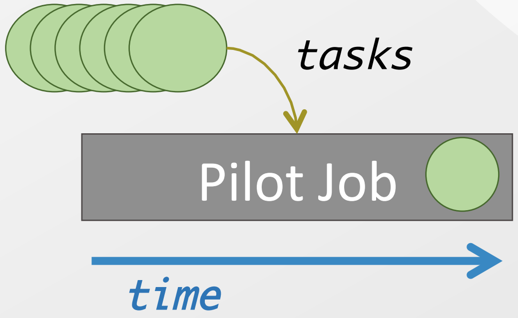
Task clustering



Partition the workflow into sub-workflows and send them for execution to the target system

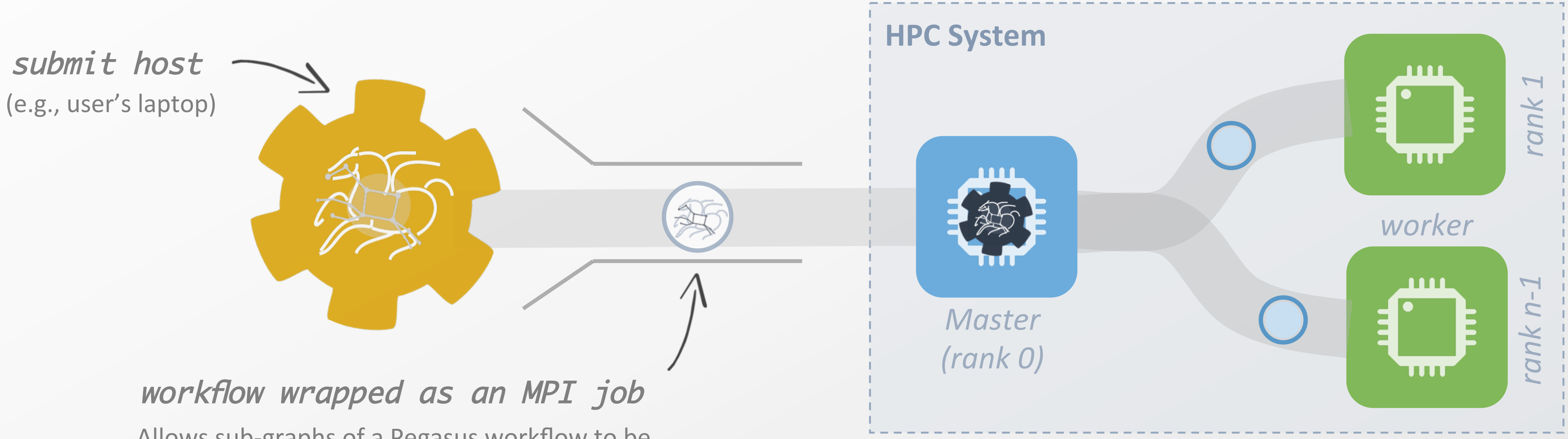


Use "pilot" jobs to dynamically provision a number of resources at a time



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

Specialized Workflow Engines Needed for Different Execution Sites



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

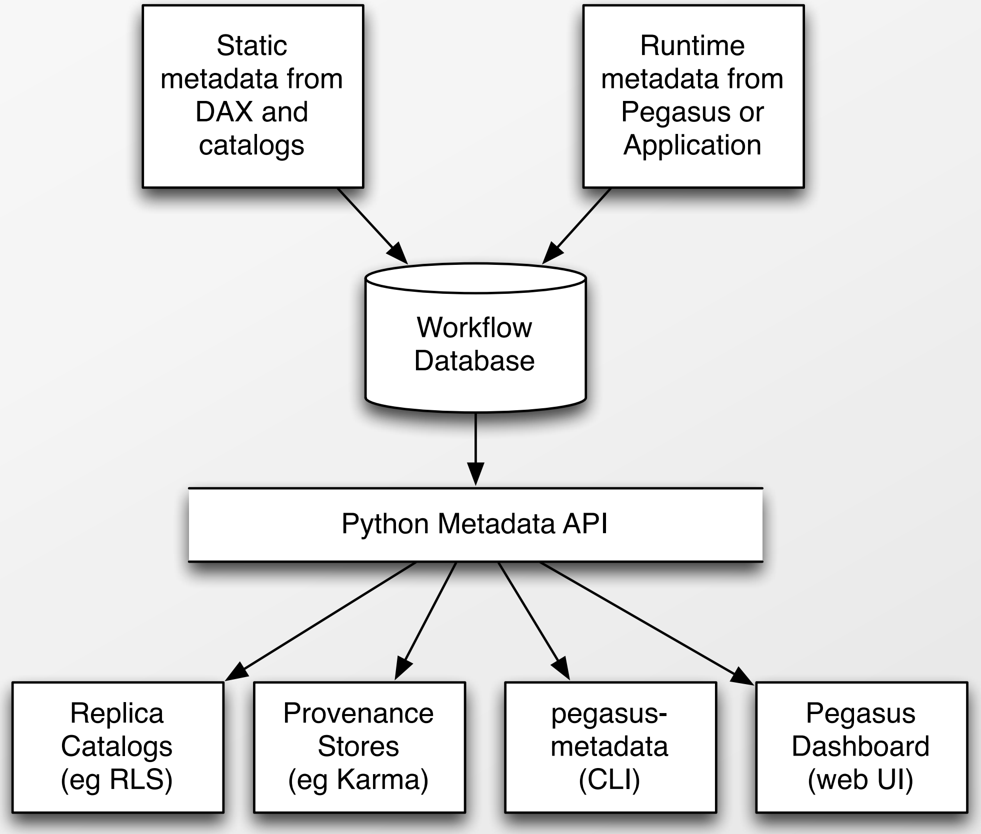
Pegasus MPI-Cluster

Outline

1. Example Pegasus Workflows
2. Pegasus Workflow Management System
3. Challenges and solutions for workflow execution in clusters, distributed systems, and clouds
4. Future Directions

Metadata

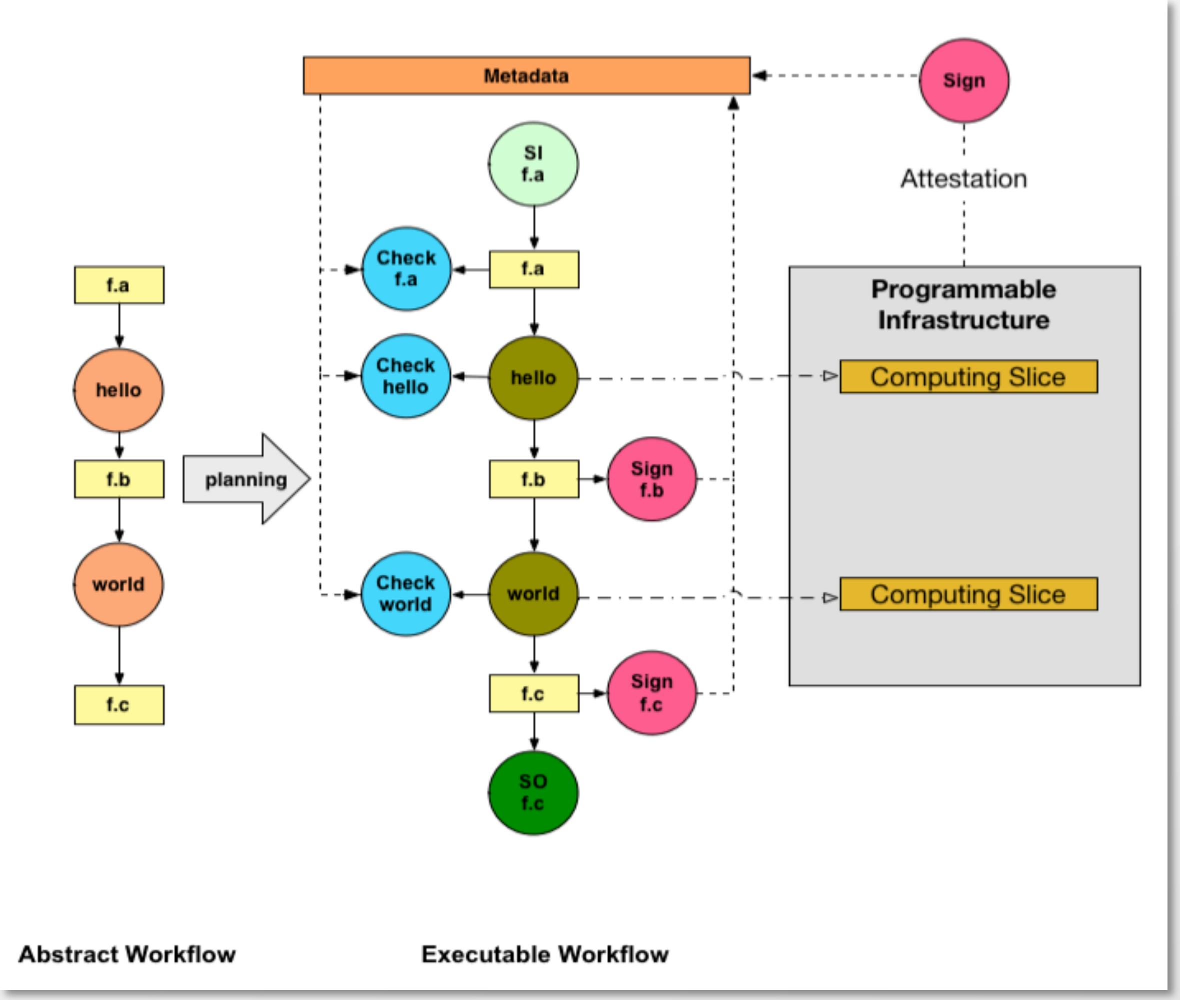
- Can associate arbitrary key-value pairs with workflows, jobs, and files
- Replica selection
  - Input files are selected based on metadata attributes
- Data registration
  - Output files get tagged with metadata on registration
- Static and runtime metadata
  - Static: application parameters
  - Runtime: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



Scientific Workflow Integrity with Pegasus

SWIP
Scientific Workflow Integrity with Pegasus

- Provide additional assurances that a scientific workflow is not accidentally or maliciously tampered with during its execution
- Allow for detection of modification to its data or executables at later dates to facilitate reproducibility
- Integrate cryptographic support for data integrity into the Pegasus Workflow Management System.



Collaboration with Von Welch (IU) and Ilya Baldin (RENCI)

Cloud-based Computing on Jetstream

Simple tool to quickly bring up a HTCondor/Pegasus cluster within Jetstream: Bootstraps master/compute nodes

Maintains configuration/credentials using SaltStack

No autoscaling, but nodes can be added/ removed at runtime manually

Used by ***Upendra Kumar Devisett*** *and* ***Susan Miller*** *to run a workflow derived from the OSG Gene Expression Matrix*

Future directions in clouds

Support for containers

Better management of elasticity of cloud resources

– Worrying about runaway resources

– Develop monitoring algorithms for scaling up and down based on upcoming workflow needs

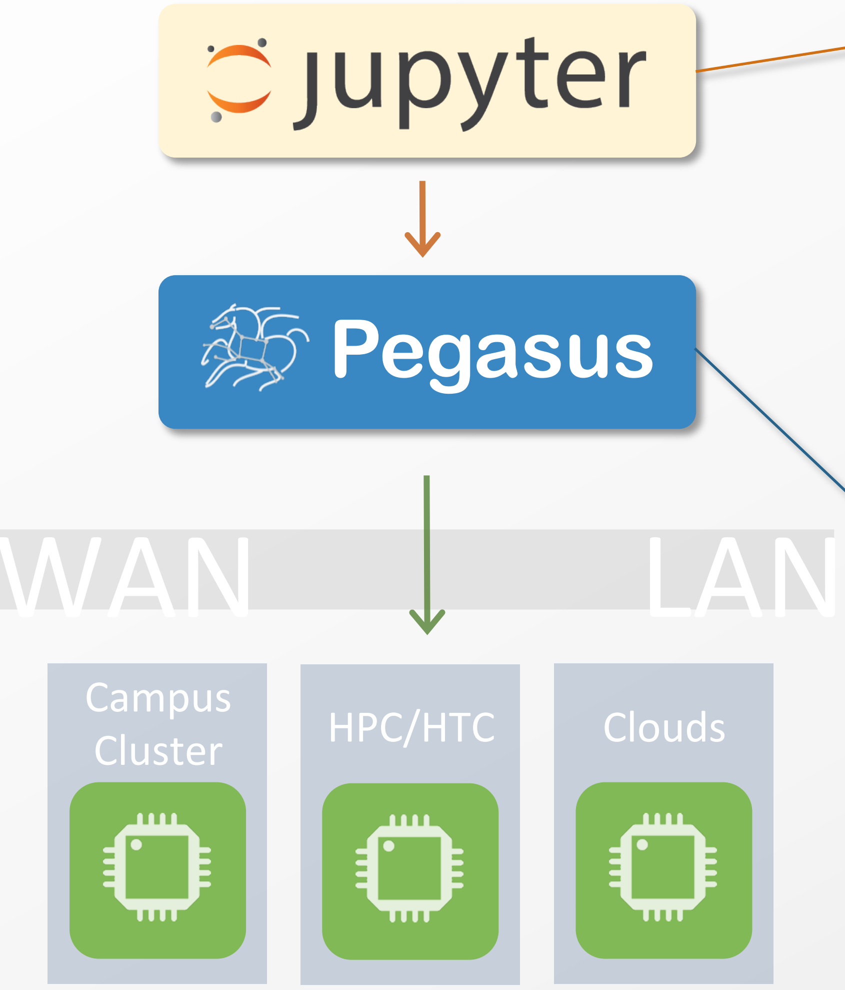
On the fly deployment through workflows

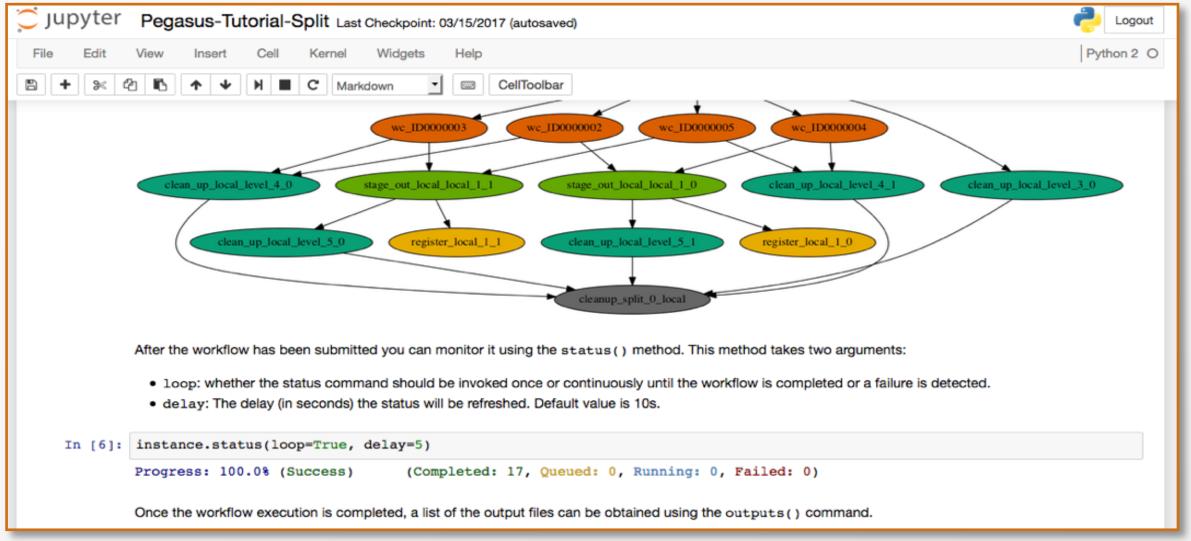
Develop a "cloud cost" calculator to help estimate the cost of workflow execution on the cloud

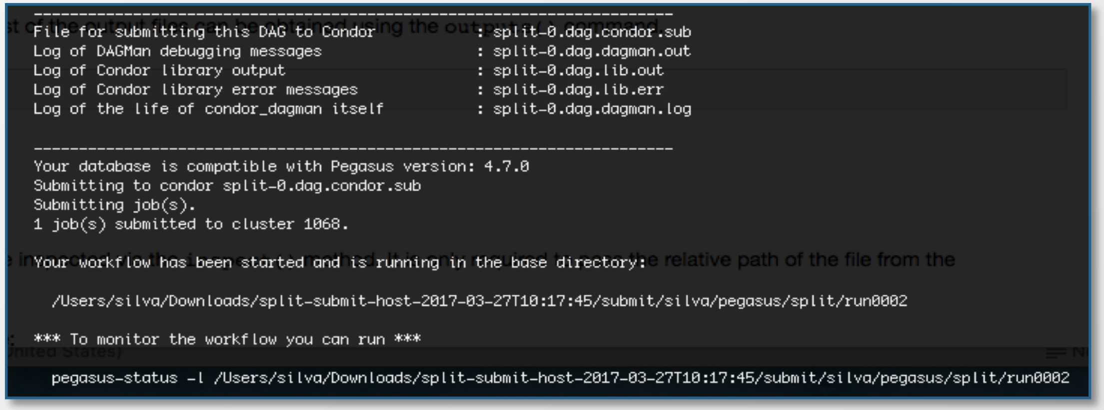
– Including different types of resources

– Including workflow ensembles

Running Pegasus workflows with Jupyter







Pegasus-Jupyter Python API

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

importing the API

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

creating an instance of the DAX

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

# the split job that splits the webpage into smaller chunks
split = Job("split")
split.addArguments("-l","100","-a","1",webpage,"part.")
split.uses(webpage, link=Link.INPUT)
# associate the label with the job. All jobs with same label
# are run with PMC when doing job clustering
split.addProfile( Profile("pegasus","label","p1"))
dax.addJob(split)
```

using the Pegasus DAX3 API to write a workflow

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

running a workflow

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

monitoring a workflow execution

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

Το βε ρελεασεδ ωιτη:

Pegasus 4.8

Ease of use and data focus

Enhancing ease of use of tools

– Workflow composition via R – available now
– Easier reuse
– Exploring integration with Jupyter Notebook

Enhanced data management

– Collecting and archiving data from sensors and instruments
– Adding data management primitives

Focus on *live* processing

– Analysis of instrumental data on the fly
– Coupling simulation and data mining/visualization "in-situ" – within an HPC system

Reproducibility, transparency, reuse

– Explore how we can capture, quantify, publish workflows in a reproducible way

Pegasus est. 2001

Automate, recover, and debug scientific computations.

Get Started

Pegasus Website

http://pegasus.isi.edu

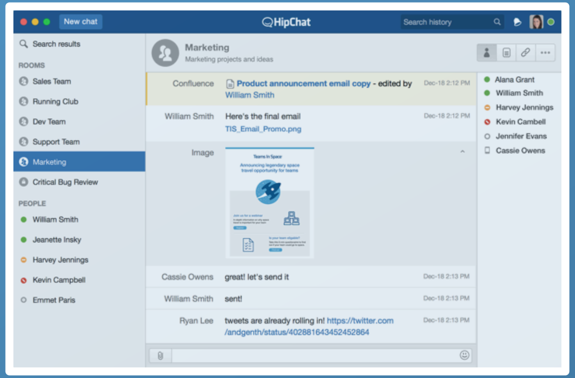
Users Mailing List

pegasus-users@isi.edu

Support

pegasus-support@isi.edu

HipChat



We welcome the opportunity to work with new applications and enhance our solutions based on user's needs.