# Enabling End-to-End Experiment Sharing and Reuse with Workflows via Jupyter Notebooks

Rafael Ferreira da Silva, Karan Vahi, Mats Rynge, Rajiv Mayani, Ewa Deelman
University of Southern California – Information Sciences Institute
***In collaboration with the HTCondor Team – University of Wisconsin, Madison*

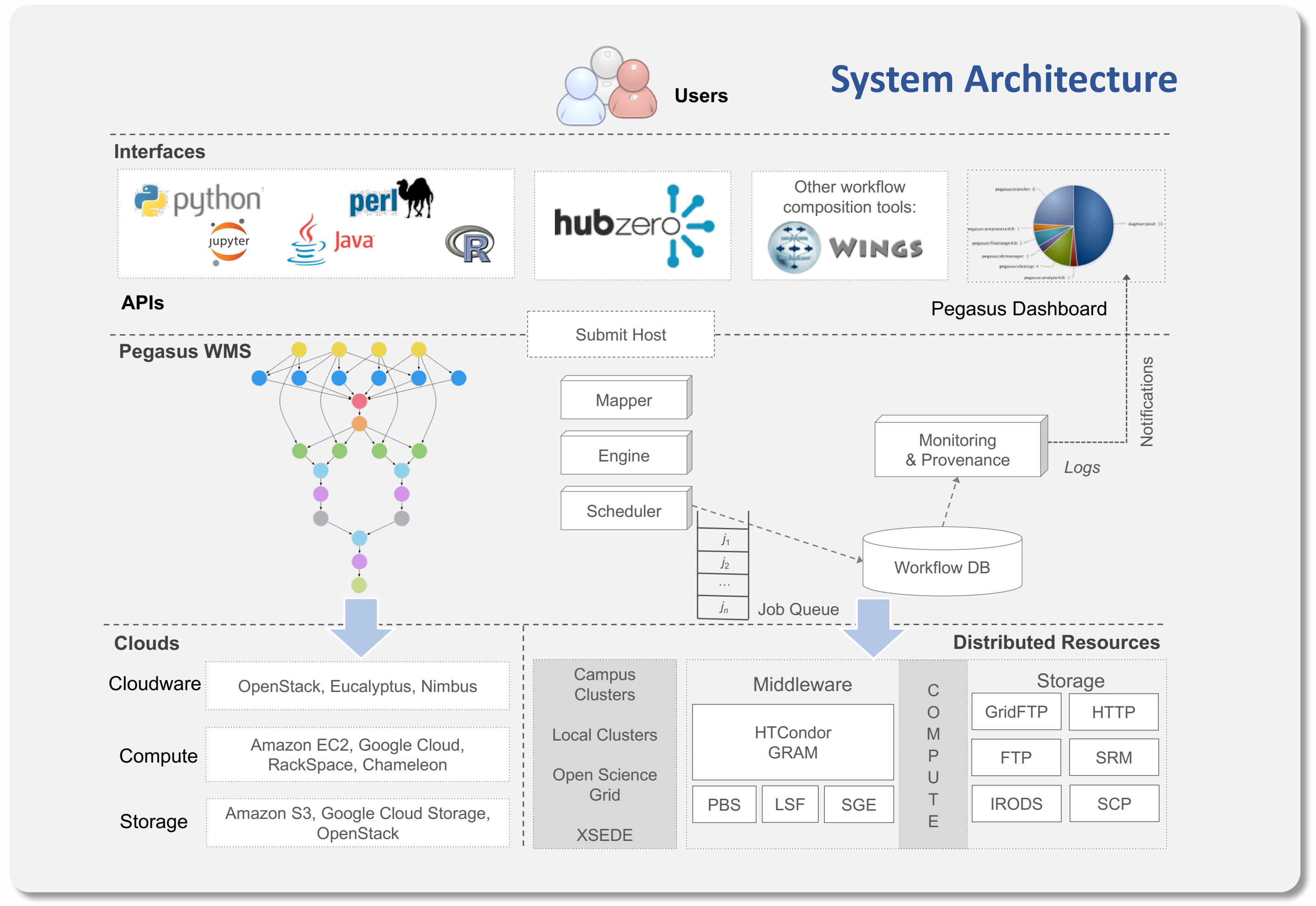
Rafael Ferreira da Silva
*rafsilva@isi.edu*

USC Viterbi School of Engineering *Information Sciences Institute*

## PEGASUS WORKFLOW MANAGEMENT SYSTEM

*Overview of the Pegasus WMS*

- Pegasus is a system for mapping and executing abstract **application workflows** over a range of execution environments
- The same abstract workflow can, at different times, be mapped **different execution environments** such as XSEDE, OSG, commercial and academic clouds, campus grids, and clusters
- Pegasus can easily scale both the size of the workflow, and the resources that the workflow is distributed over. Pegasus runs workflows ranging from just a few computational tasks **up to 1 million**
- Stores static and runtime **metadata** associated with workflow, files and tasks. Accessible via command line tools and **web based dashboard**
- Pegasus-MPI-Cluster enables fine-grained task graphs to be executed **efficiently on HPC** resources



## CANONICAL WORKFLOW EXAMPLE

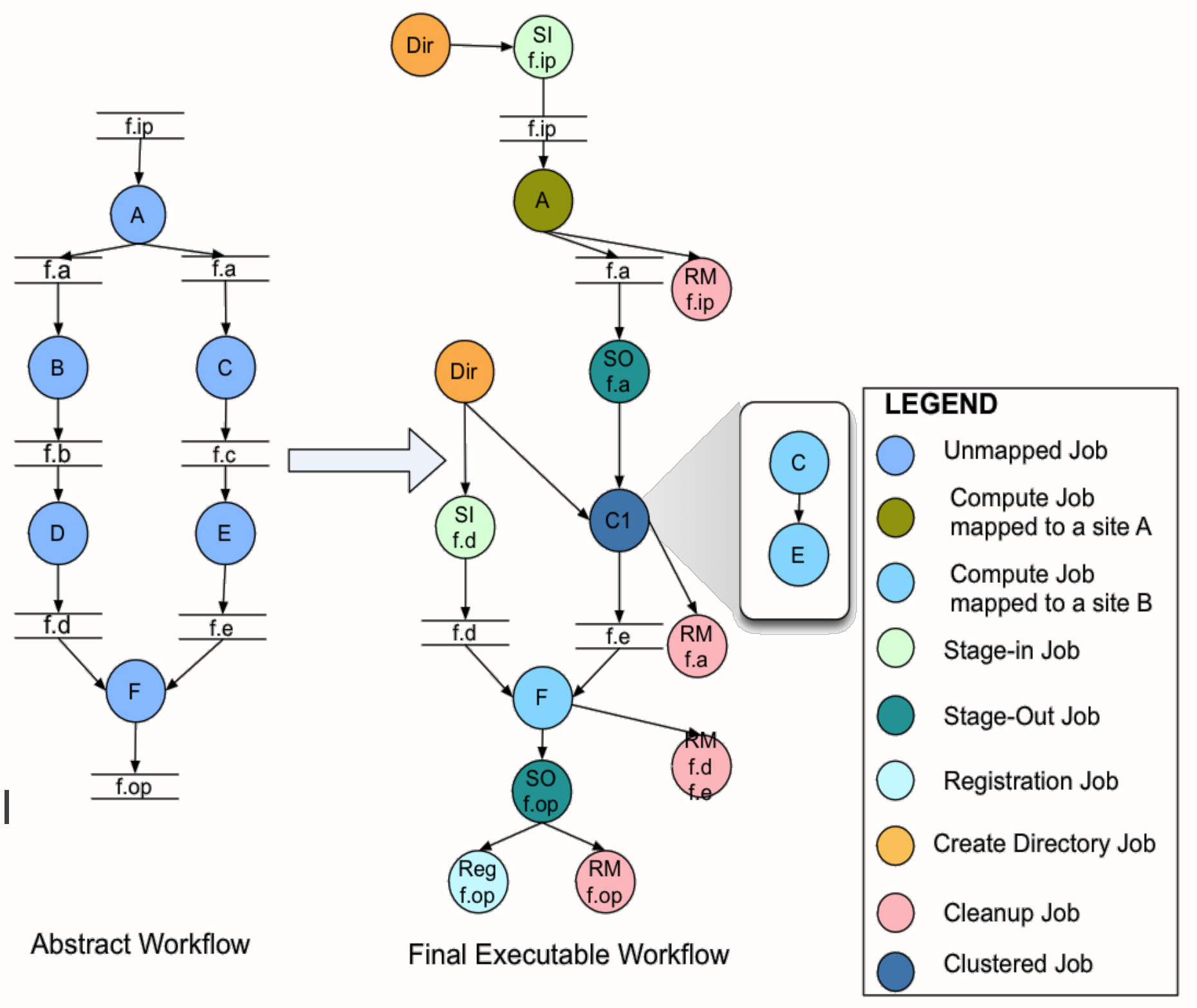
*From Abstract to Executable Workflows*

### Capabilities Highlighted

- Data Management
- Data Reuse
- Job Clustering
- Cross Site Runs

### Dashboard

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.



### Software Availability

- **Release Schedule**
  - Major Release every 9 months; Minor releases every 4 months
- **Download Options**
  - Source Code publicly hosted on GitHub
  - Binary packages for Linux and MAC
  - YUM/APT repositories with RPM/DEB packages
- **Documentation / Training Materials**
  - *Tutorials* via Virtual Machine, EC2, and Docker images
  - *Support* via Email lists and online chat rooms

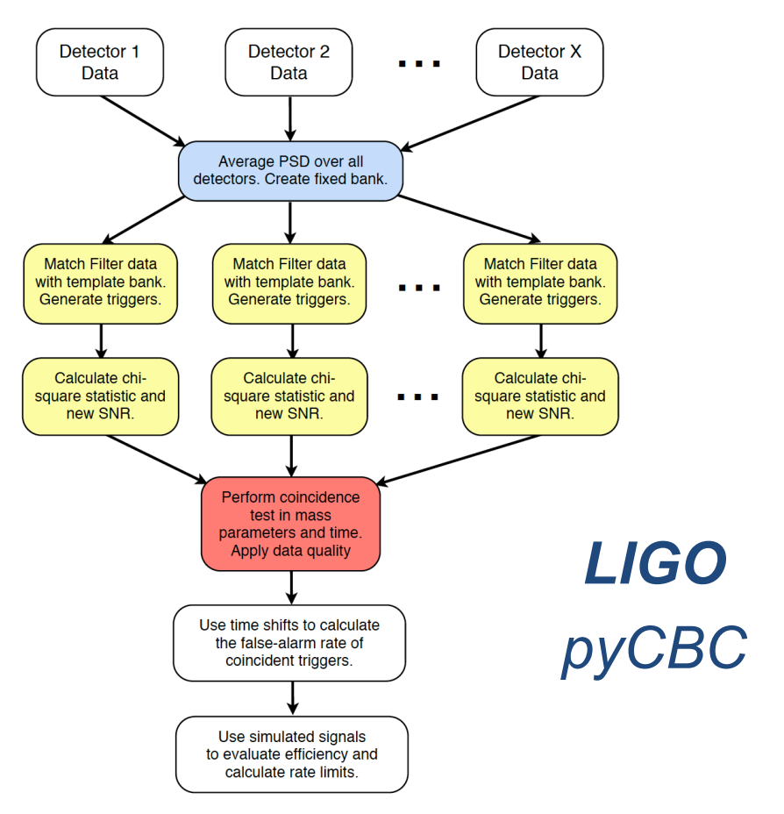
*Downloads and Usage*

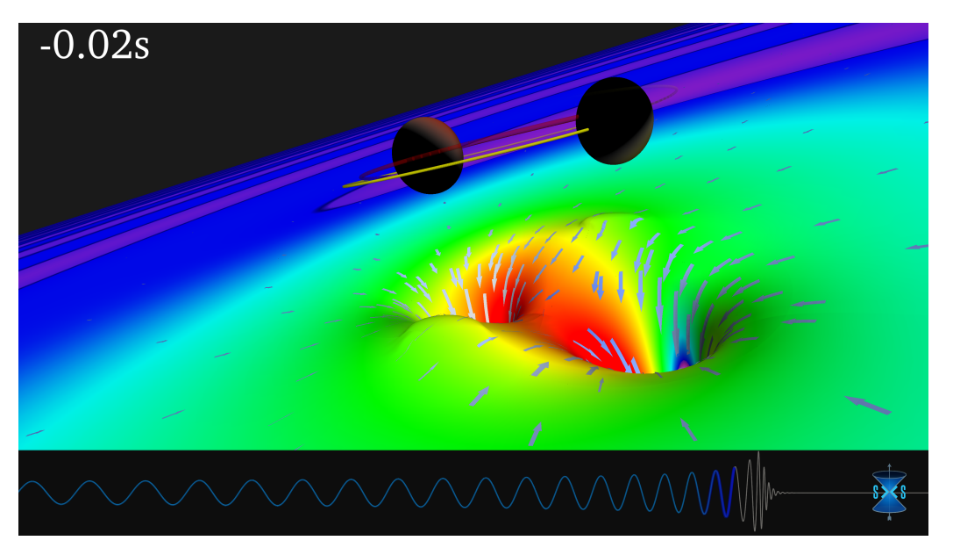
## SCIENTIFIC APPLICATIONS

*Highlighted Applications - https://pegasus.isi.edu/applications*

### Astronomy and Physics

**Pegasus powered LIGO analysis workflows to detect gravitational waves**





*0.2 Second before the black holes collide. Image credit: SXS/LIGO*

**Periodogram** workflows help detect extra solar planets

**Galactic Plane** workflow generates mosaics for astronomy surveys

### Bioinformatics

**Quality control** workflows for data submissions to NRGR repository and PAGE consortium

**Imputation** workflows on PAGE data

Workflows for **Genome and Transcriptome** free analysis of RSEQ

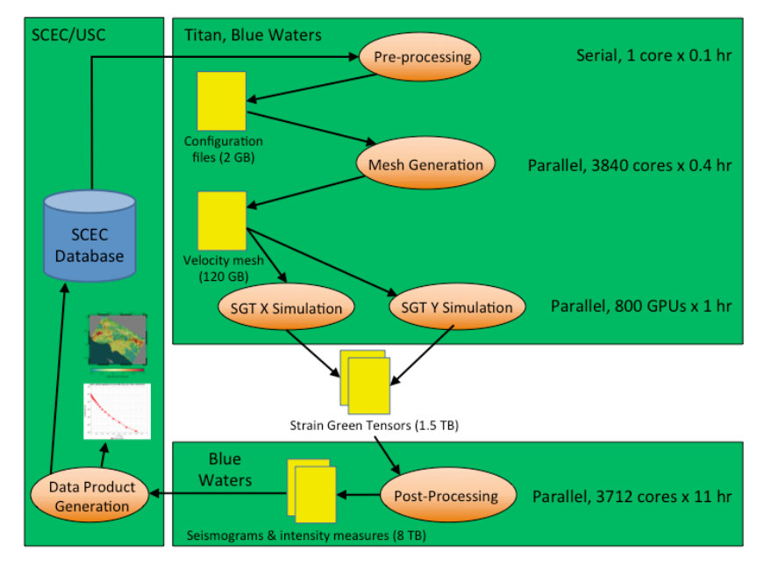
**Brain span** workflows help study gene expression in the brain

**RNA Sequencing** workflows for generating Cancer Genome Atlas

**Soybean Knowledge base** (SoyKB) workflow for resequencing soy-bean germplasm lines

### Seismology

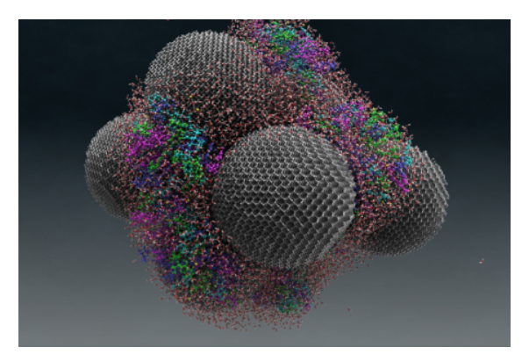
CyberShake workflows for **seismic hazard** analysis of LA basin



Broadband workflows for accurate predictions of ground motions

### Materials Science

**Spallation Neutron Source** Workflows study molecular dynamics and neutron scattering intensity calculations



Neutrons, simulation analysis of tRNA-nanodiamond combo could transform drug delivery design principles. Image credit: OLCF, ORNL

## CONTAINERS

*Using Containers for Running User Applications*

Application containers provides a solution to package software with **complex dependencies** to be used during workflow execution

Pegasus has support for application containers in the **non-shared filesystem** or condorio data configurations using PegasusLite.

docker

Containers currently can only be specified in the Transformation Catalog

Users have the option of either using a **different container for each executable** or **same container for all executables**

### Container Execution Model

- Sets up a directory to run a user job in
- Pulls in all the relevant input data, executables, and the container image to execution directory
- *Optionally*, loads the container from the container image file and sets up the user to run as in the container (only applicable for Docker containers)
- Mounts the job directory into the container as `/scratch` for Docker containers, while as `/srv` for Singularity containers
- Container will run a job specific script that figures out the appropriate Pegasus worker to use in the container if not already installed, and sets up the job environment to use it, before launching the user application using *pegasus-kickstart.*
- Optionally, shuts down the container (only applicable for Docker containers)
- Ships out the output data to the staging site
- Cleans up the directory on the worker node

## JUPYTER NOTEBOOKS

*Executing Scientific Workflows using the Pegasus Jupyter Python API*

The Pegasus-Jupyter integration aims to facilitate the usage of Pegasus via Jupyter notebooks. In addition to easiness of usage, notebooks foster **reproducibility** (all the information to run an experiment is in a unique place) and **reuse** (notebooks are portable if running in equivalent environments)

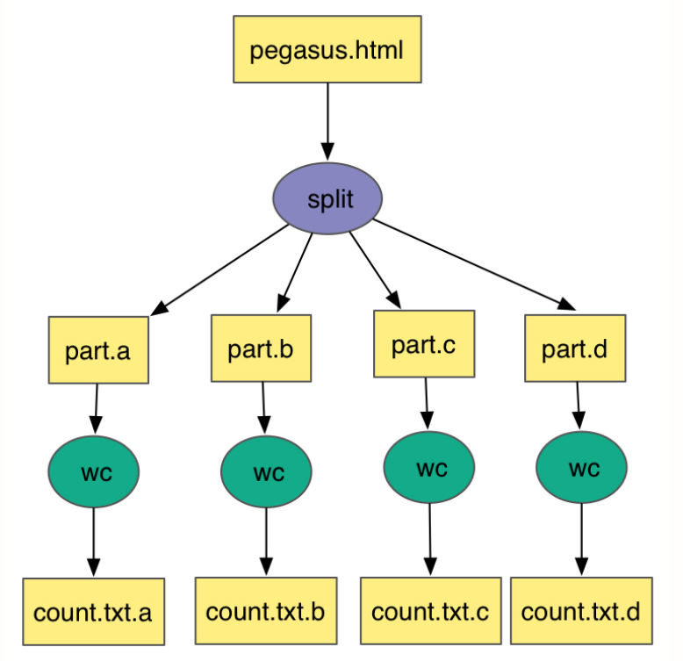
### Pegasus-Jupyter Python API

The first step to enable Jupyter to use the Pegasus API is to import the Python Pegasus Jupyter API. The instance module will automatically load the **Pegasus DAX3** API and the **catalogs** APIs.

```
In [ ]: from Pegasus.jupyter.instance import *
```

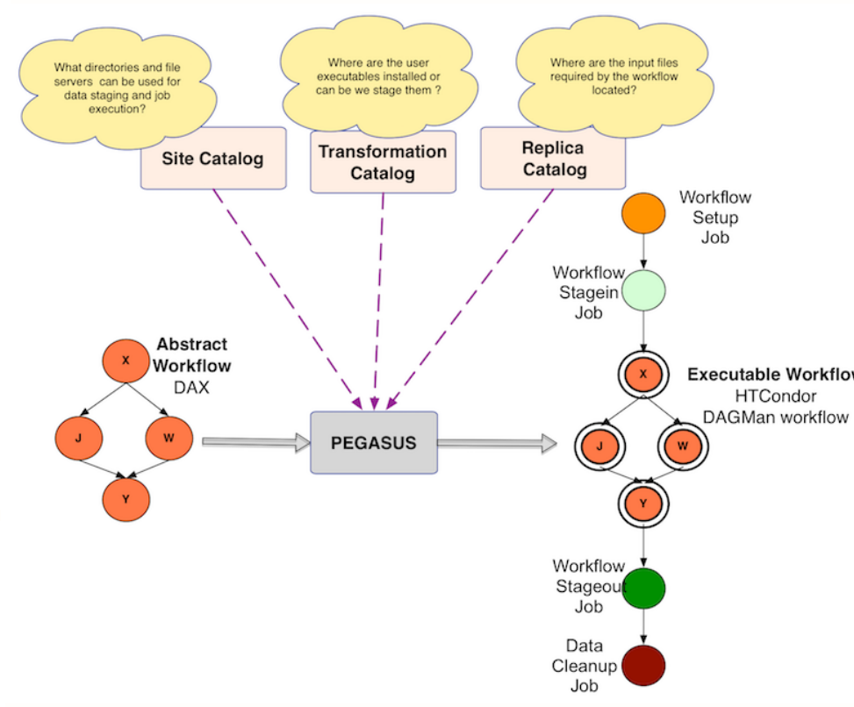
Pegasus reads workflow descriptions from DAX files. The term "DAX" is short for "Directed Acyclic Graph in XML". DAX is an XML file format that has syntax for expressing **jobs**, **arguments**, **files**, and **dependencies**.

```
In [ ]: # Create an abstract dag
        dax = ADAG('split')

In [ ]: webpage = File('pegasus.html')
        dax.addFile(webpage)

        # 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)
        dax.addJob(split)

        # we do a parmeter sweep on the first 4 chunks created
        for c in "abcd":
            part = File("part.%s" % c)
            split.uses(part, link=Link.OUTPUT, transfer=False, register=False)

            count = File("count.txt.%s" % c)

            wc = Job("wc")
            wc.addProfile( Profile("pegasus","label","p1"))
            wc.addArguments("-l",part)
            wc.setStdout(count)
            wc.uses(part, link=Link.INPUT)
            wc.uses(count, link=Link.OUTPUT, transfer=True, register=True)
            dax.addJob(wc)

            #adding dependency
            dax.depends(wc, split)
```



### The Catalogs API

```
In [ ]: rc = ReplicaCatalog(workflow_dir)
        rc.add('pegasus.html', 'file:///nfs/v5/rafsilva/pegasus.html', site='local'

In [ ]: e_split = Executable('split', arch=Arch.X86_64, os=OSType.LINUX, installed=
        e_split.addPFN(PFN('file:///usr/bin/split', 'condorpool'))

        e_wc = Executable('wc', arch=Arch.X86_64, os=OSType.LINUX, installed=True)
        e_wc.addPFN(PFN('file:///usr/bin/wc', 'condorpool'))

In [ ]: tc = TransformationCatalog(workflow_dir)
        tc.add(e_split)
        tc.add(e_wc)

In [ ]: sc = SitesCatalog(workflow_dir)
        sc.add_site('condorpool', arch=Arch.X86_64, os=OSType.LINUX)
        sc.add_site_profile('condorpool', namespace=Namespace.PEGASUS, key='style',
        sc.add_site_profile('condorpool', namespace=Namespace.CONDOR, key='universe
```



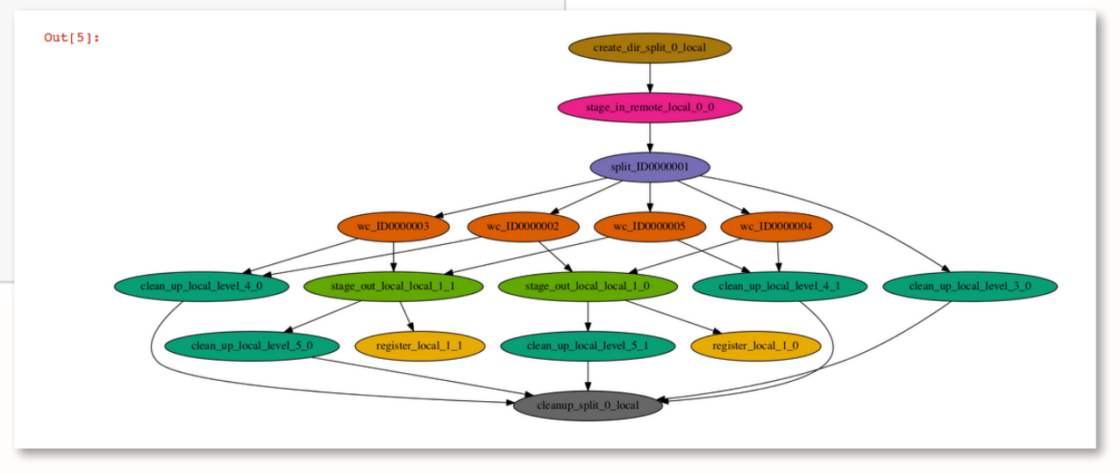
### Running and Monitoring Workflows

```
In [ ]: instance = Instance(dax, replica_catalog=rc, transformation_catalog=tc, sit

In [ ]: instance.run(site='condorpool', force=True)

In [ ]: instance.status(loop=True, delay=5)

         Progress: 23.1% (Running)        (Completed: 3, Queued: 0, Running: 2, Failed: 0)
```

### Additional Capabilities

*Visualizing the Executable Workflow*

```
In [ ]: wf_image_exe = instance.view(abstract=False)

        # IPython package for visualizing images
        from IPython.display import Image
        Image(wf_image_exe)
```



*Workflow statistics*

```
In [ ]: instance.statistics()

         Workflow Wall Time: 47 min, 23 secs
```

## LEARN MORE

**Get in Touch**
https://pegasus.isi.edu – pegasus-support@isi.edu

Pegasus is funded by the National Science Foundation under the OAC SI2-SSI program, grant #1664162