USC Viterbi
School of Engineering
Information Sciences Institute

Pegasus

Pegasus Workflow Management System

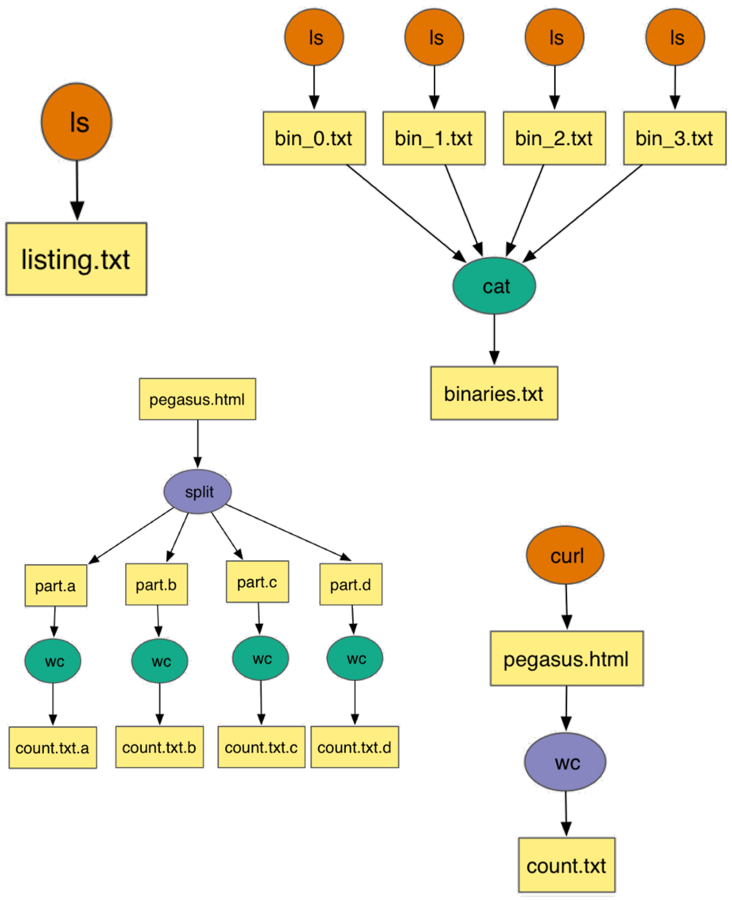
Karan Vahi

USC Information Sciences Institute

HTCondor Week 2020

Benefits of Scientific Workflows (from the point of view of an application scientist)

Pegasus



- Conducts a series of computational tasks.
  - Resources distributed across Internet.
- Chaining (outputs become inputs) replaces manual hand-offs.
  - Accelerated creation of products.
- Ease of use - gives non-developers access to sophisticated codes.
  - Avoids need to download-install-learn how to use someone else's code.
- Provides framework to host or assemble community set of applications.
  - Honors original codes. Allows for heterogeneous coding styles.
- Framework to define common formats or standards when useful.
  - Promotes exchange of data, products, codes. Community metadata.
- Multi-disciplinary workflows can promote even broader collaborations.
  - E.g., ground motions fed into simulation of building shaking.
- Certain rules or guidelines make it easier to add a code into a workflow.

Slide courtesy of David Okaya, SCEC, USC

Challenges of Workflow Management

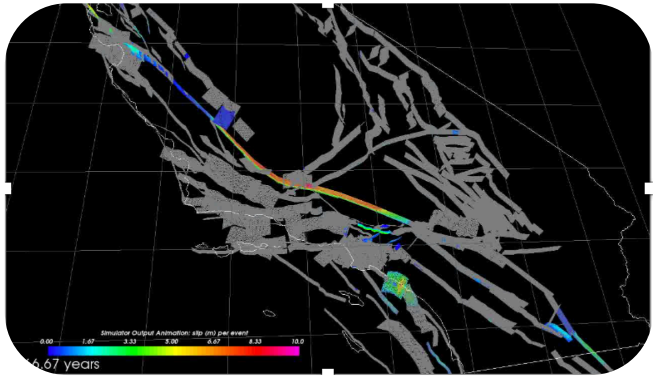
Challenges across domains

- Need to describe complex workflows in a simple way
- Need to access distributed, heterogeneous data and resources (heterogeneous interfaces)
- Need to deal with resources/software that change over time

Our focus

- Separation between workflow description and workflow execution
- Workflow planning and scheduling (scalability, performance)
- Task execution (monitoring, fault tolerance, debugging)
- Provide additional assurances that a scientific workflow is not accidentally or maliciously tampered with during its execution.

Sky mosaic, IPAC, Caltech

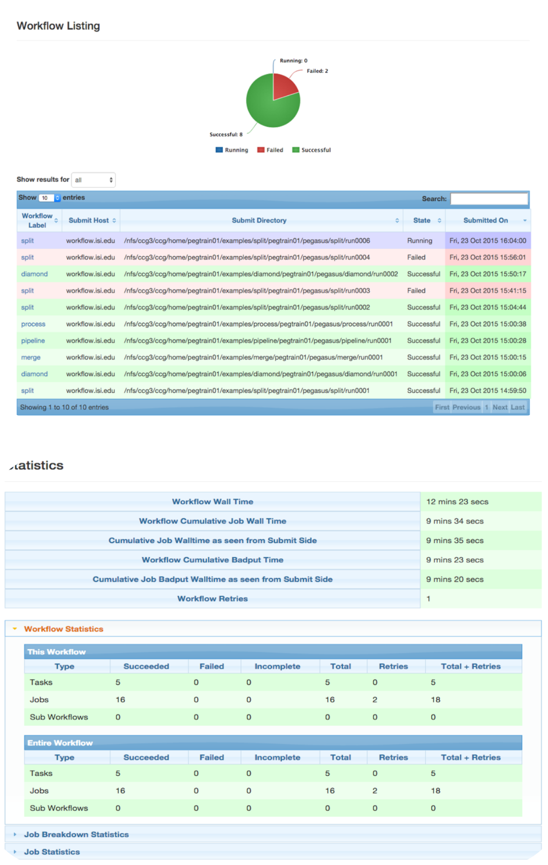


Earthquake simulation, SCEC, USC

Pegasus Workflow Management System

Pegasus

- Operates at the level of files and individual applications
- Allows scientists to describe their computational processes (workflows) at a logical level
- Without including details of target heterogeneous CI (portability)
- Scalable to $O(10^6)$ tasks, TBs of data
- Captures provenance and supports reproducibility
- Includes monitoring and debugging tools



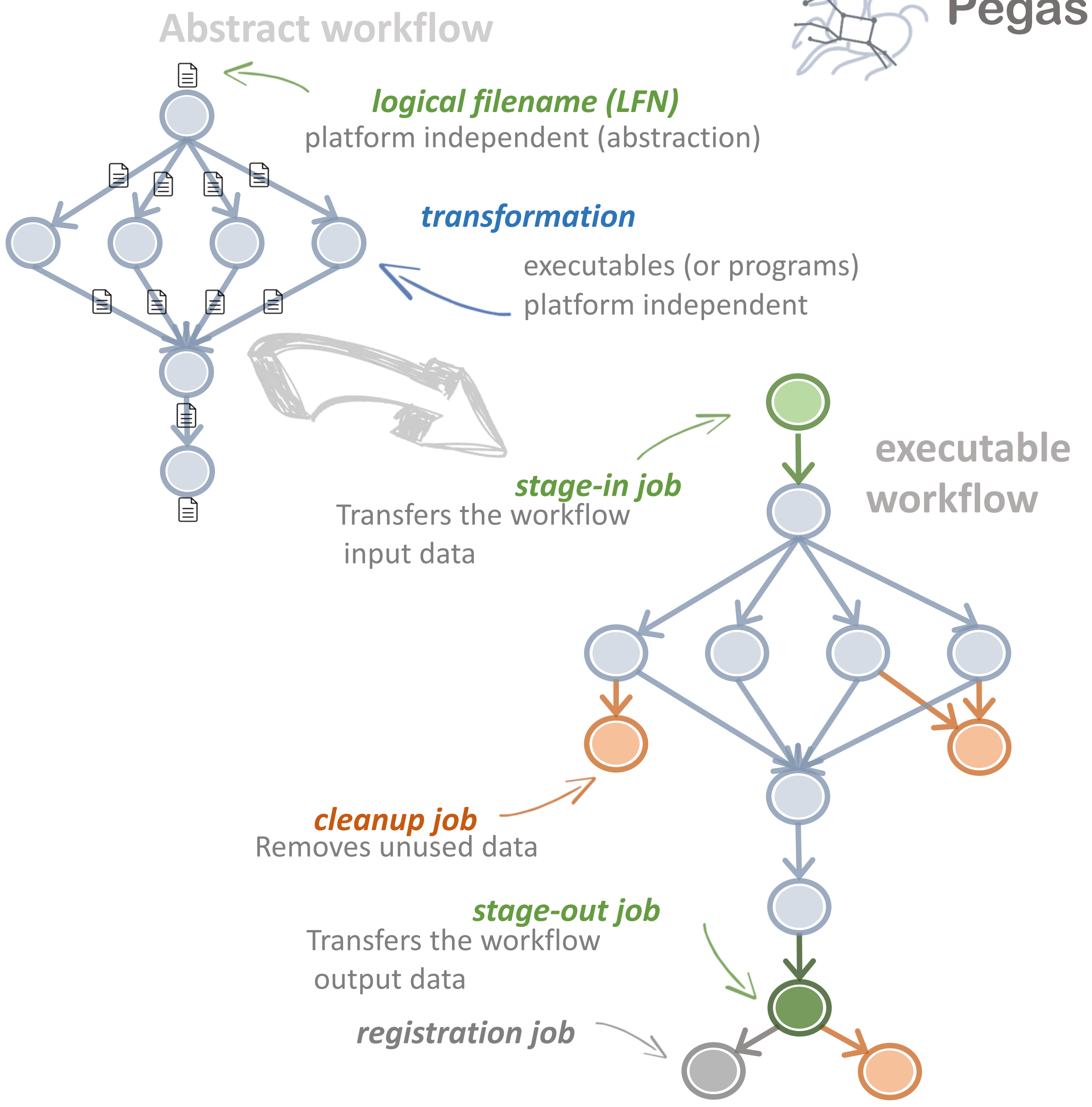
Composition in Python, R, Java, Perl, Jupyter Notebook

hubzero

HTCondor Week 2020

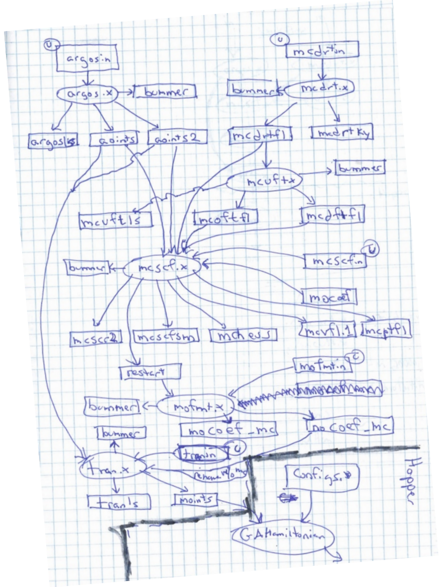
Pegasus Concepts

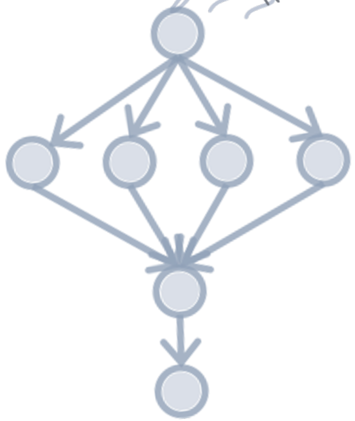
- Users describe their pipelines in a portable format called Abstract Workflow, without worrying about low level execution details.
- Workflows are DAGs
  - Nodes: jobs, edges: dependencies
  - No while loops, no conditional branches
  - Jobs are standalone executables
- Pegasus takes this and generates an executable workflow that
  - has data management tasks added
  - transforms the workflow for performance and reliability



Pegasus

Pegasus also provides tools to generate the workflow descriptions





```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)
```

python™ Java R perl jupyter

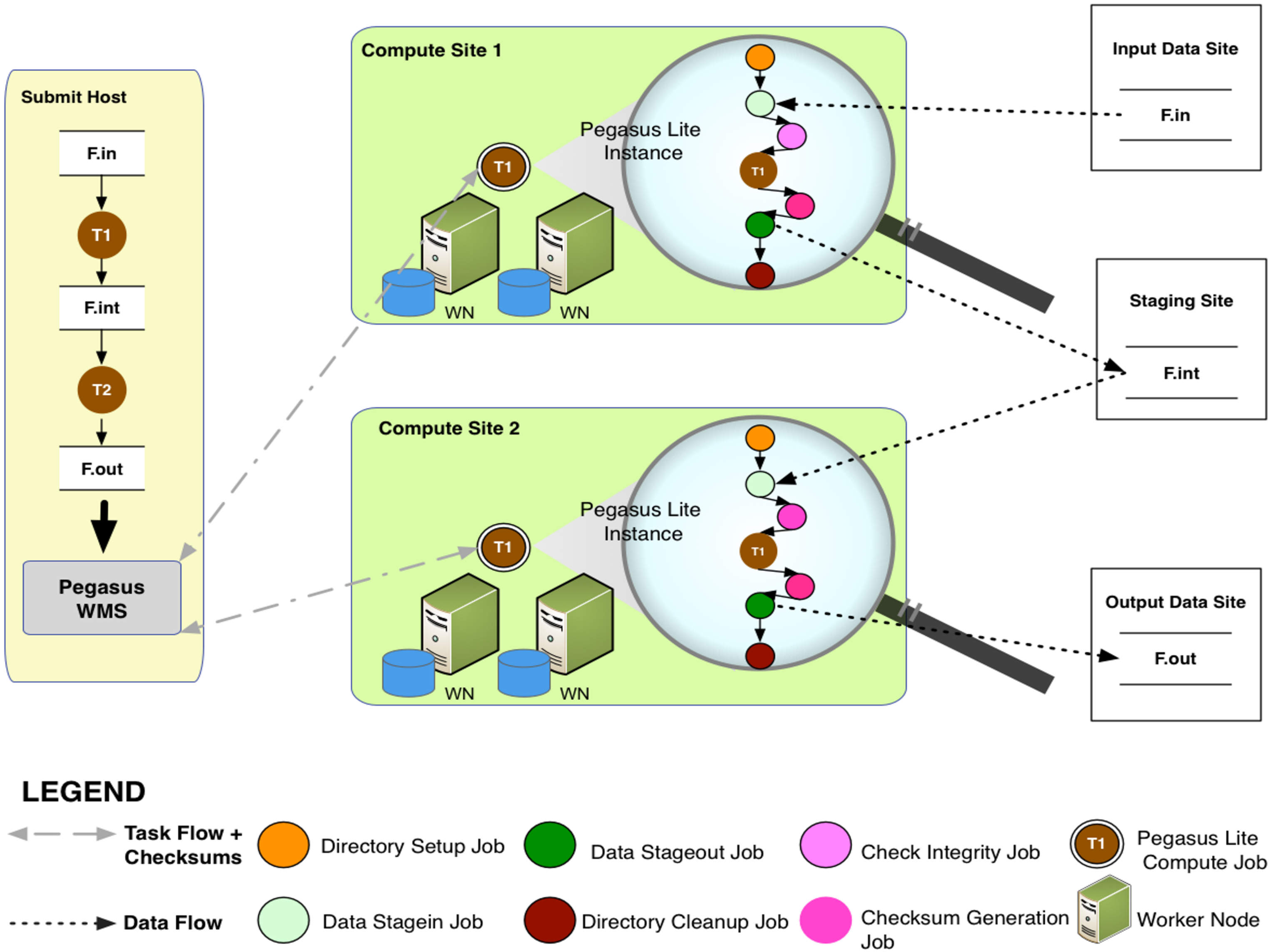
```xml
<?xml version="1.0" encoding="UTF-8"?>

<!-- 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

Pegasus Deployment

- Workflow Submit Node
  - Pegasus WMS
  - HTCondor
- One or more Compute Sites
  - Compute Clusters
  - Cloud
  - OSG
- Input Sites
  - Host Input Data
- Data Staging Site
  - Coordinate data movement for workflow
- Output Site
  - Where output data is placed

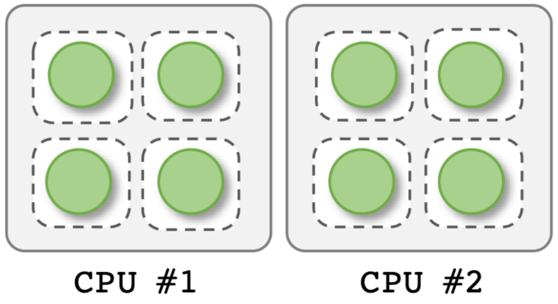


Pegasus Optimizations

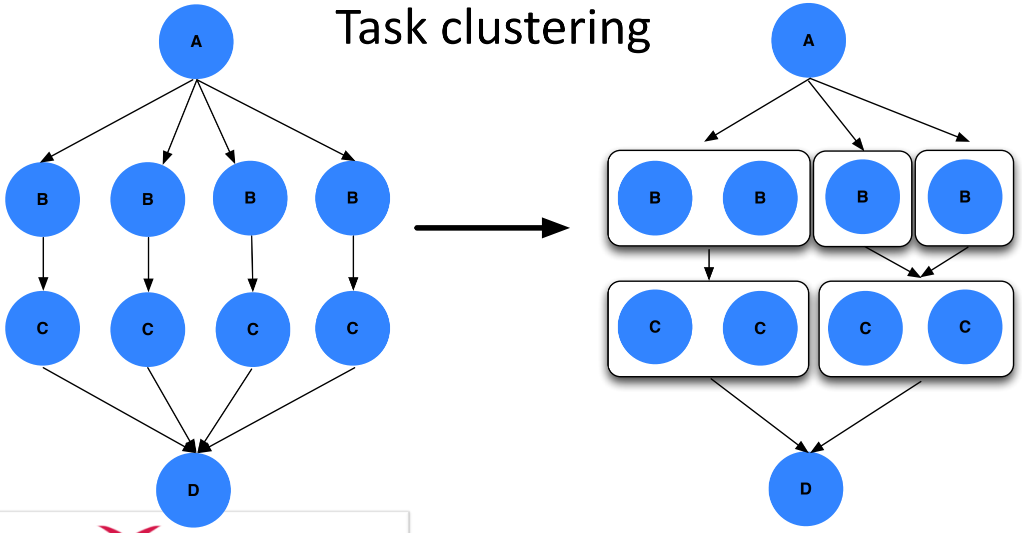
Pegasus

well-known optimizations

Task-resource co-allocation



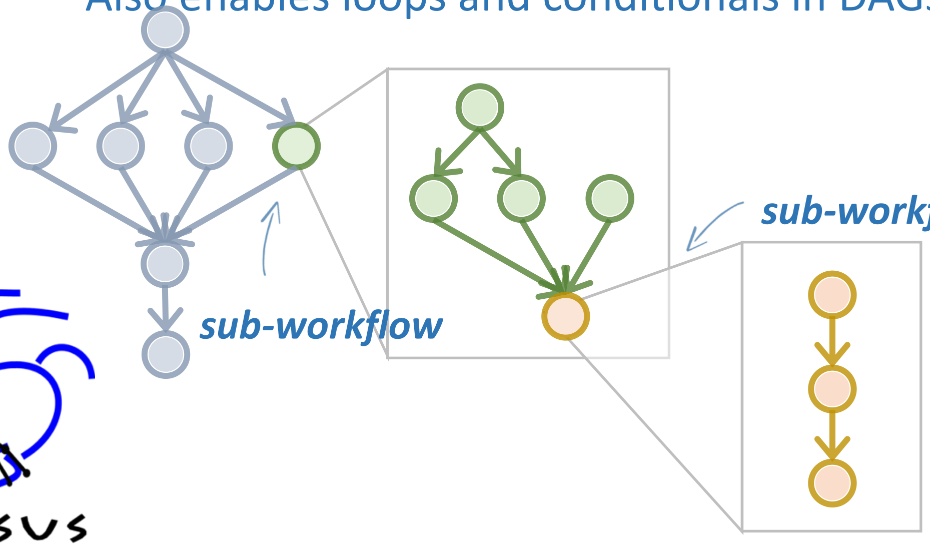
Task clustering



Hierarchical workflows

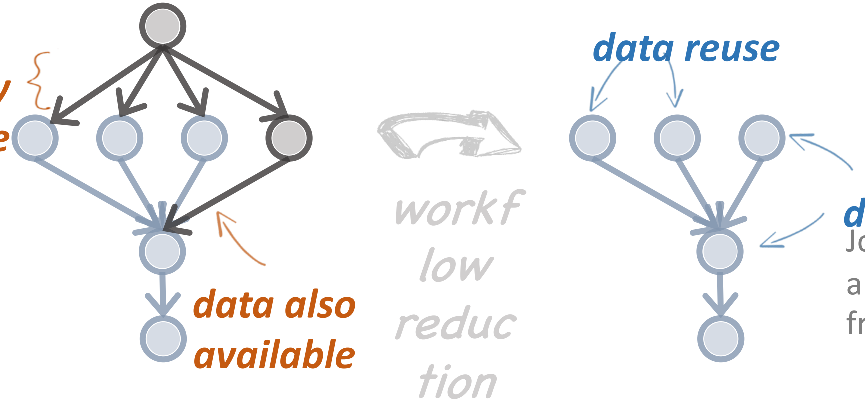
Enacts the execution of **millions of tasks**

Also enables loops and conditionals in DAGs



recursion ends when Pegasus Workflow with only compute jobs is encountered

modern workflow optimizations



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

Data Staging Configurations

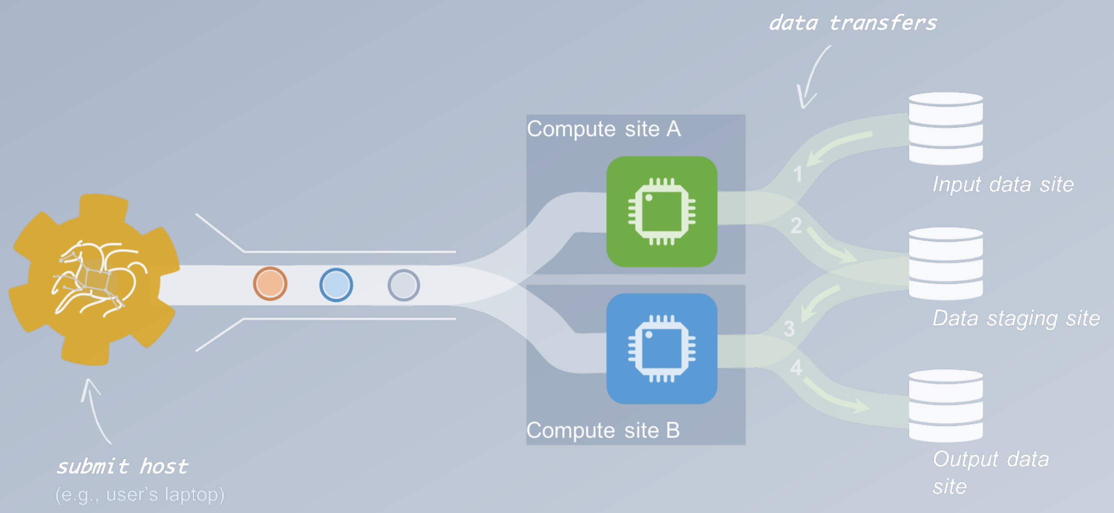
Pegasus

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

pegasus-transfer

Pegasus' internal data transfer tool with support for a number of different protocols

Pegasus

Directory creation, file removal

If protocol can support it, also used for cleanup

Two stage transfers

e.g., GridFTP to S3 = GridFTP to local file, local file to S3

Parallel transfers

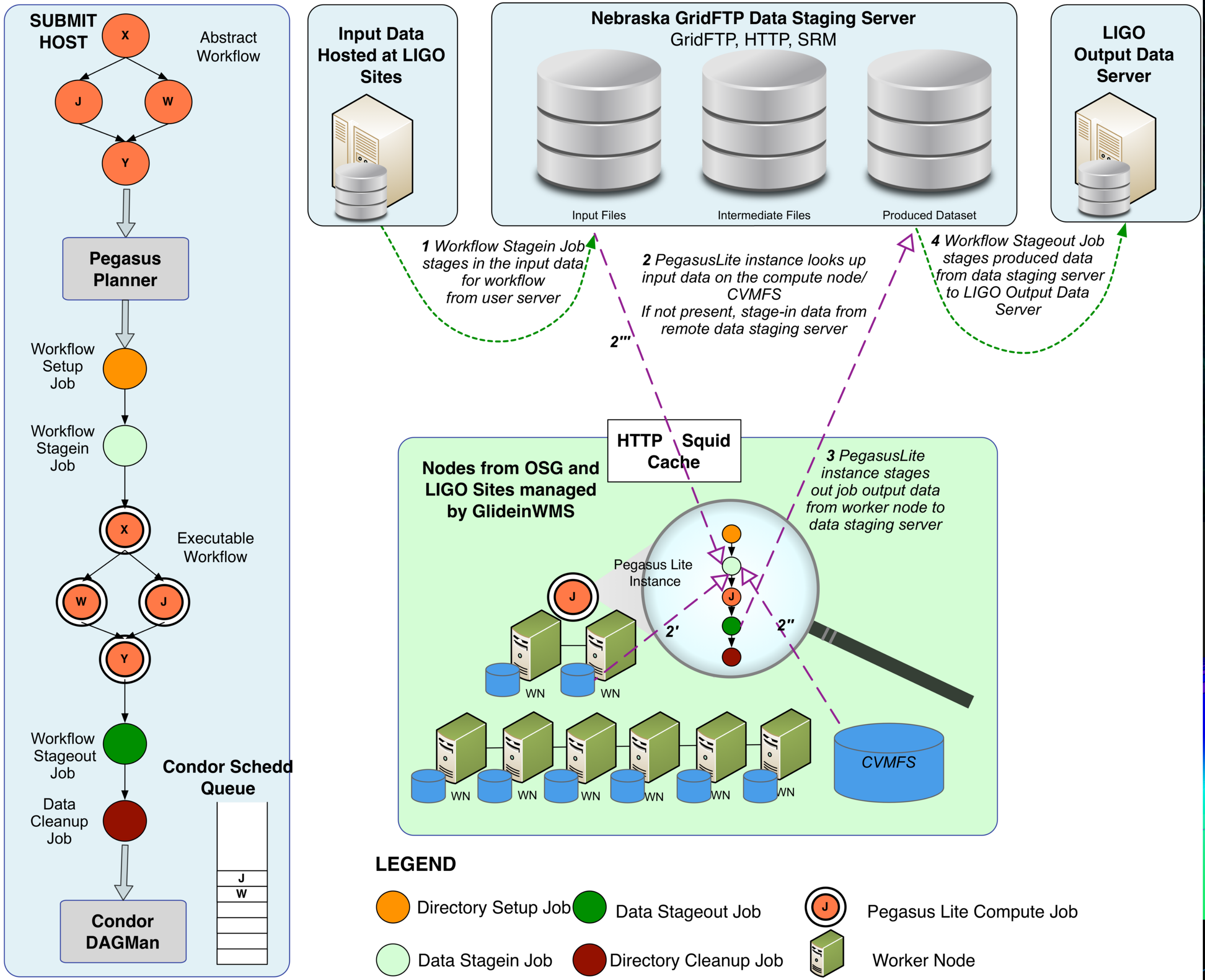
Automatic retries

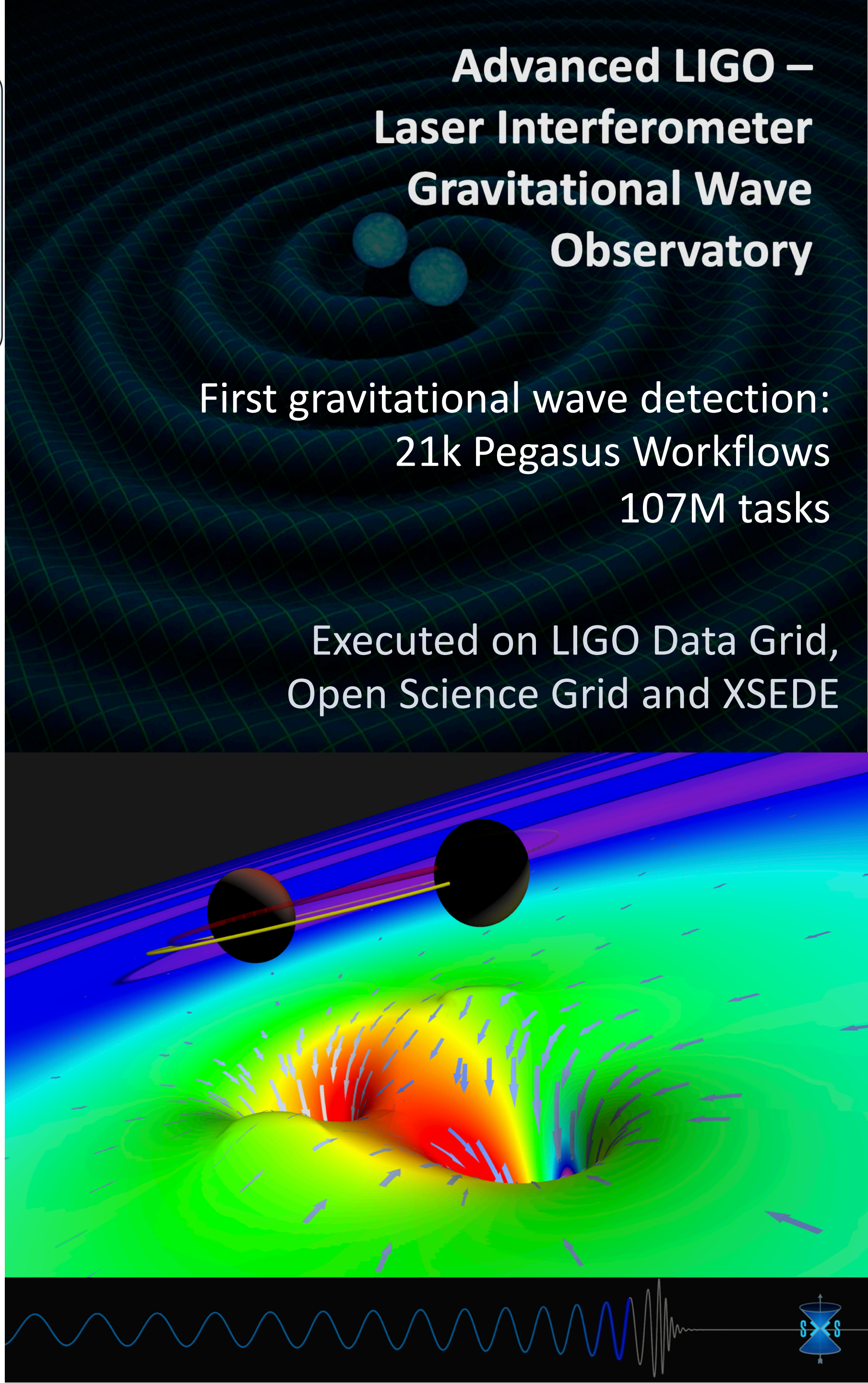
Credential management

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

```
HTTP
SCP
GridFTP
Globus
Online
iRods
Amazon S3
Google
Storage
SRM
FDT
Stashcp
Rucio
Webdav
cp
ln -s
```

Data Flow for LIGO Pegasus Workflows in OSG





Challenges to Scientific Data Integrity

Modern IT systems are not perfect - errors creep in.

At modern "Big Data" sizes we are starting to see checksums breaking down.

Plus there is the threat of intentional changes: malicious attackers, insider threats, etc.

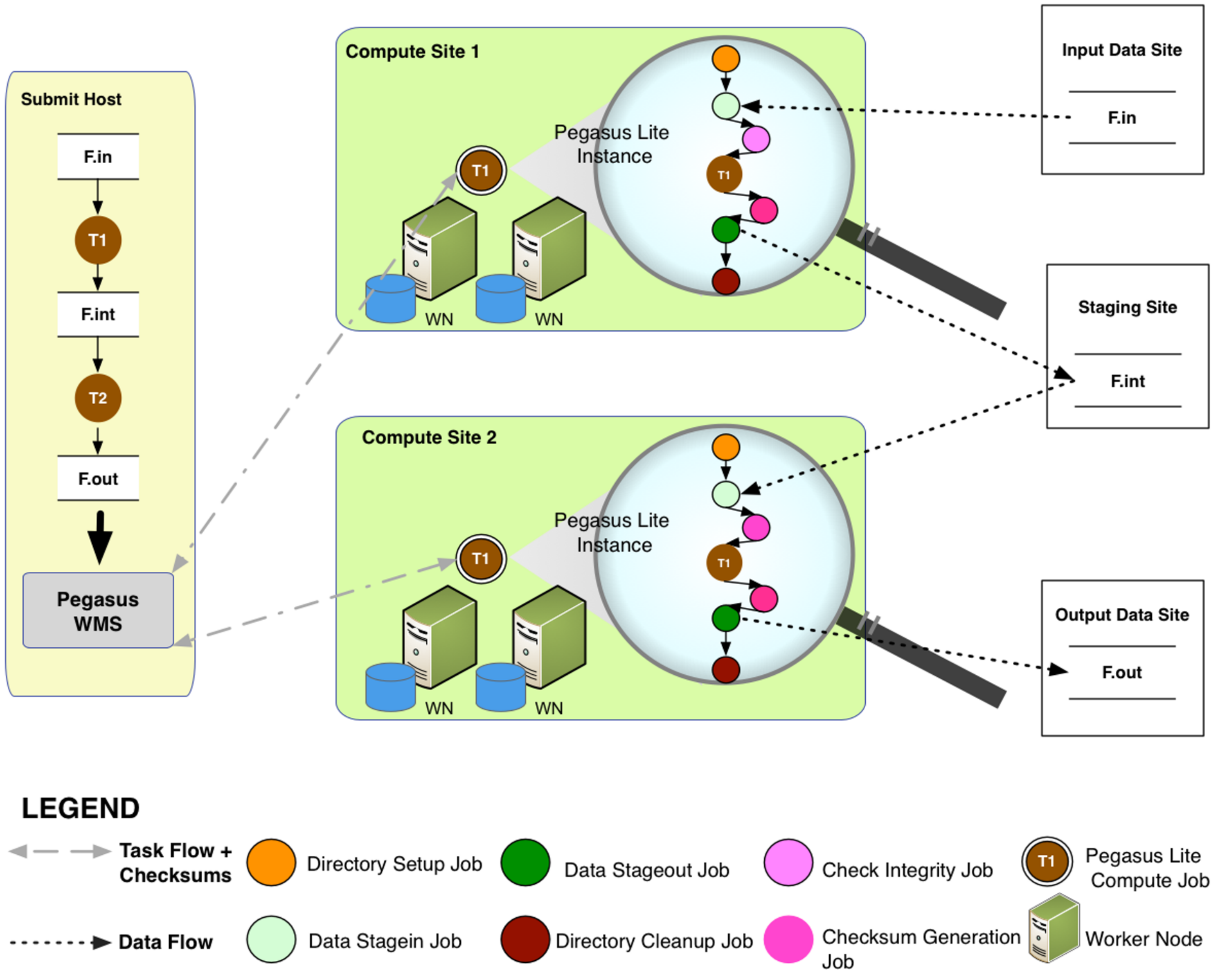
User Perception: "Am I not already protected? I have heard about TCP checksums, encrypted transfers, checksum validation, RAID and erasure coding – is that not enough?"

Automatic Integrity Checking in Pegasus

Pegasus performs integrity checksums on input files right before a job starts on the remote node.

- For raw inputs, checksums specified in the input replica catalog along with file locations
- All intermediate and output files checksums are generated and tracked within the system.
- Support for sha256 checksums

Job failure is triggered if checksums fail



Pegasus: Containers Data Management

- Treat containers as input data dependency
  - Needs to be staged to compute node if not present

- Users can refer to container images as
  - Docker Hub or Singularity Library URL's
  - Docker Image exported as a TAR file and available at a server , just like any other input dataset.

- If an image is specified to be residing in a hub
  - The image is pulled down as a tar file as part of data stage-in jobs in the workflow
  - The exported tar file is then shipped with the workflow and made available to the jobs
  - Motivation: Avoid hitting Docker Hub/Singularity Library repeatedly for large workflows

- Symlink against a container image if available on shared fileystem
  - For e.g. CVMFS hosted images on Open Science Grid

Pegasus: Container Representation

Described in Transformation Catalog

- Maps logical transformations to physical executables on a particular system

container

Reference to the container to use. Multiple transformation can refer to same container

type

Can be either docker or singularity or shifter

image

URL to image in a docker|singularity hub OR to an existing docker image exported as a tar file or singularity image

mount

Mount information to mount host directories into container

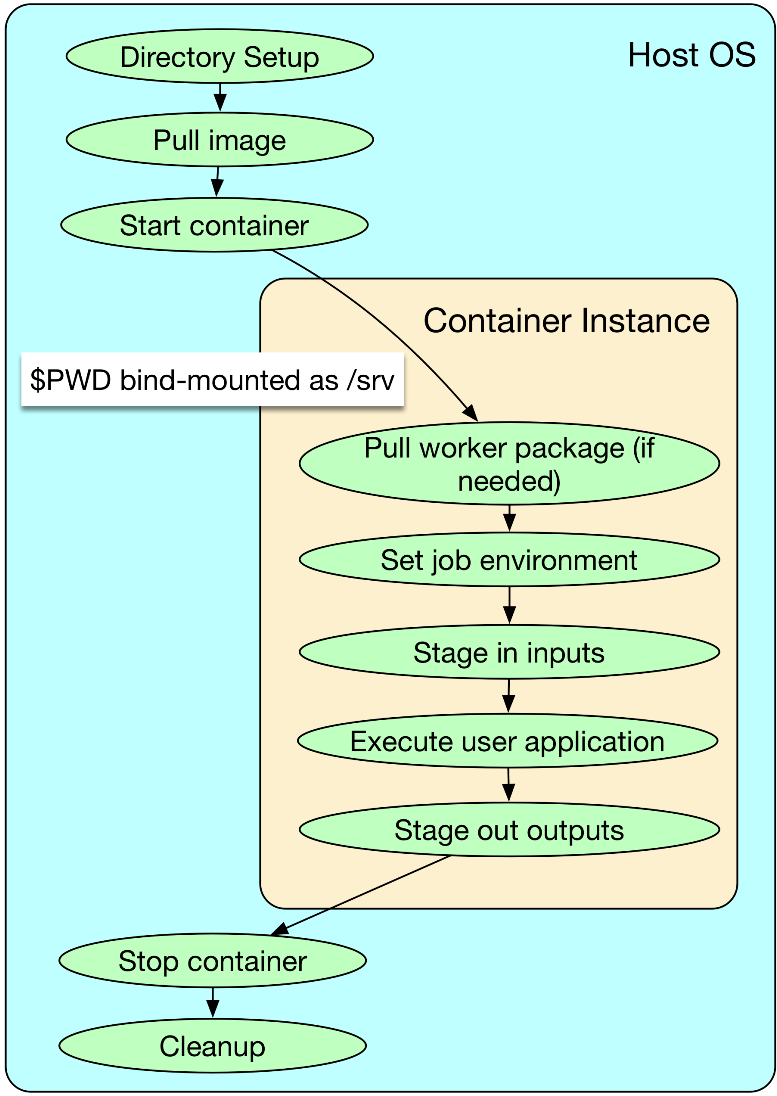
```
- transformations
    - namespace: "example"
      name: "keg"
      version: 1.0
    site:
    - name: "isi"
      arch: "x86
      os "linux"
      pfn "/usr/bin/pegasus-keg
      container "centos-pegasus"

      # INSTALLED means pfn refers to path in the container.
      # STAGEABLE means the executable can be staged into the container
      type "INSTALLED"

- cont:
    - name: "centos-pegasus"

      # can be docker, singularity or shifter
      type: "docker"

      # URL to image in docker|singularity hub or shifter repo URL or
      # URL to an existing image exported as a tar file or singularity image file
      image: "docker:///centos:7"

      # mount information to mount host directories into
      # container format src-dir:dest-dir[:options]
      mount:
         - "/Volumes/Work/lfs1:/shared-data/:ro"

      # environment to be set when the job is run in the container
      # only env profiles are supported
      profile:
      - env:
            "JAVA_HOME" "/opt/java/1.6"
```

docker

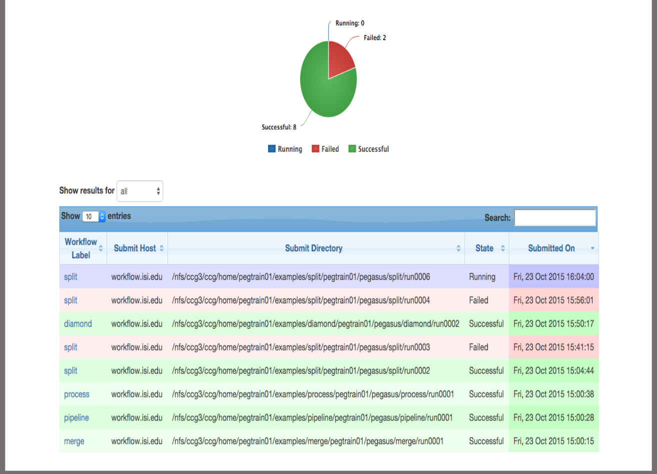
SHIFTER

Pegasus: Container Execution Model

- Containerized jobs are launched via Pegasus Lite
  - Container image is put in the job directory along with input data.
  - Loads the container if required on the node (applicable for Docker)
  - Run a script in the container that sets up Pegasus in the container and job environment
  - Stage-in job input data
  - Launches user application
  - Ship out the output data generated by the application
  - Shut down the container (applicable for Docker)
  - Cleanup the job directory

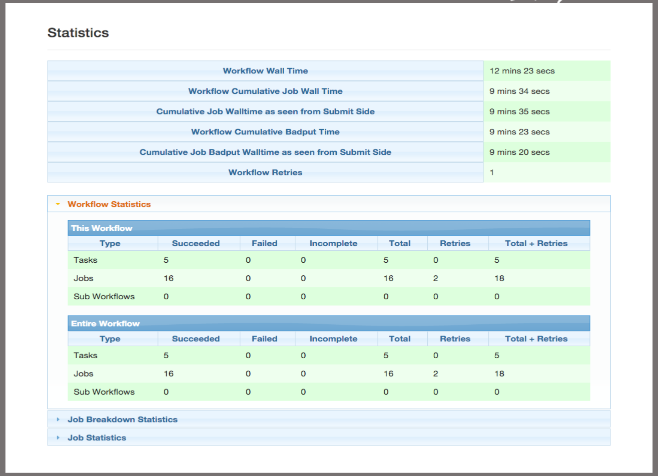


Pegasus

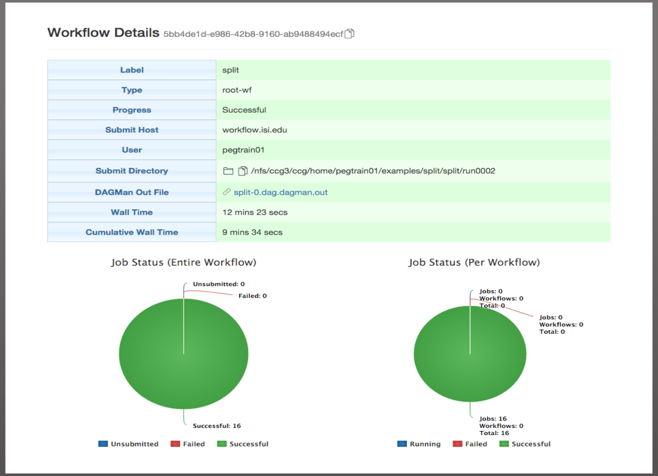


Pegasus
dashboard

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

Pegasus
dashboard

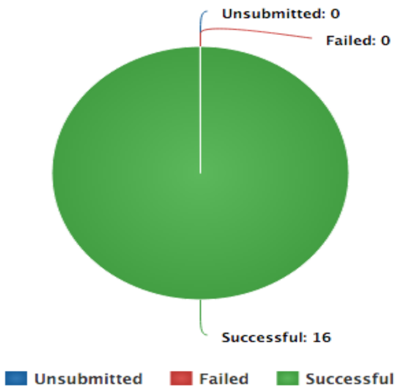
web interface for monitoring and debugging workflows

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.

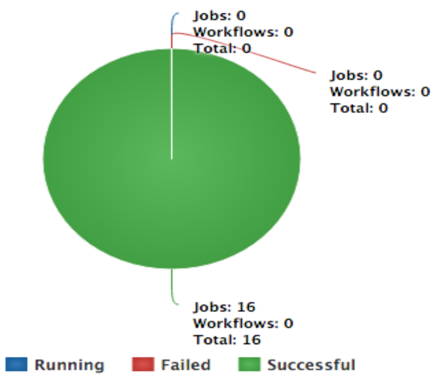
Workflow Details 5bb4de1d-e986-42b8-9160-ab9488494ecf

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

Job Status (Entire Workflow)



Job Status (Per Workflow)



Pegasus 5.0

Automate, recover, and debug scientific computations

- Reworked Python API to compose, submit and monitor workflows and configure catalogs
- New Catalog Formats
- Python 3 support
  - All Pegasus tools are Python 3 compliant.
  - 5.0 release will require Python 3 on workflow submit node
  - Python PIP packages for workflow composition and monitoring
- Zero configuration required to submit to local HTCondor pool.
- Data Management Improvements
  - New output replica catalog that registers outputs including file metadata such as size and checksums
  - Improved support for hierarchal workflows
- Revamped Documentation

```
#!/usr/bin/env python3
import logging
import sys

from Pegasus.api import *

# logs to be sent to stdout
logging.basicConfig(level=logging.DEBUG, stream=sys.stdout)

# --- Transformations ---------------------------------------
echo = Transformation(
        "echo",
        pfn="/bin/echo",
        site="condorpool"
    )

tc = TransformationCatalog()\
        .add_transformations(echo)

# --- Workflow -----------------------------------------------
Workflow("hello-world", infer_dependencies=True)\
    .add_jobs(
        Job(echo)
            .add_args("Hello World")
            .set_stdout("hello.out")
    ).add_transformation_catalog(tc)\
    .plan(submit=True)\
    .wait()
```

And if a job fails?

Pegasus

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

Job Submissions

Local

Submit Machine

Personal HTCondor

Local Campus Cluster accessible via Submit Machine **

HTCondor via BLAHP

Remote

BOSCO + SSH**

Each node in executable workflow submitted via SSH connection to remote cluster

BOSCO based Glideins**

SSH based submission of glideins

PyGlidein

IceCube glidein service

OSG using glideinWMS

Infrastructure provisioned glideins

CREAMCE

Uses CondorG

Globus GRAM

Uses CondorG

*** Both Glite and BOSCO build on HTCondor BLAHP*

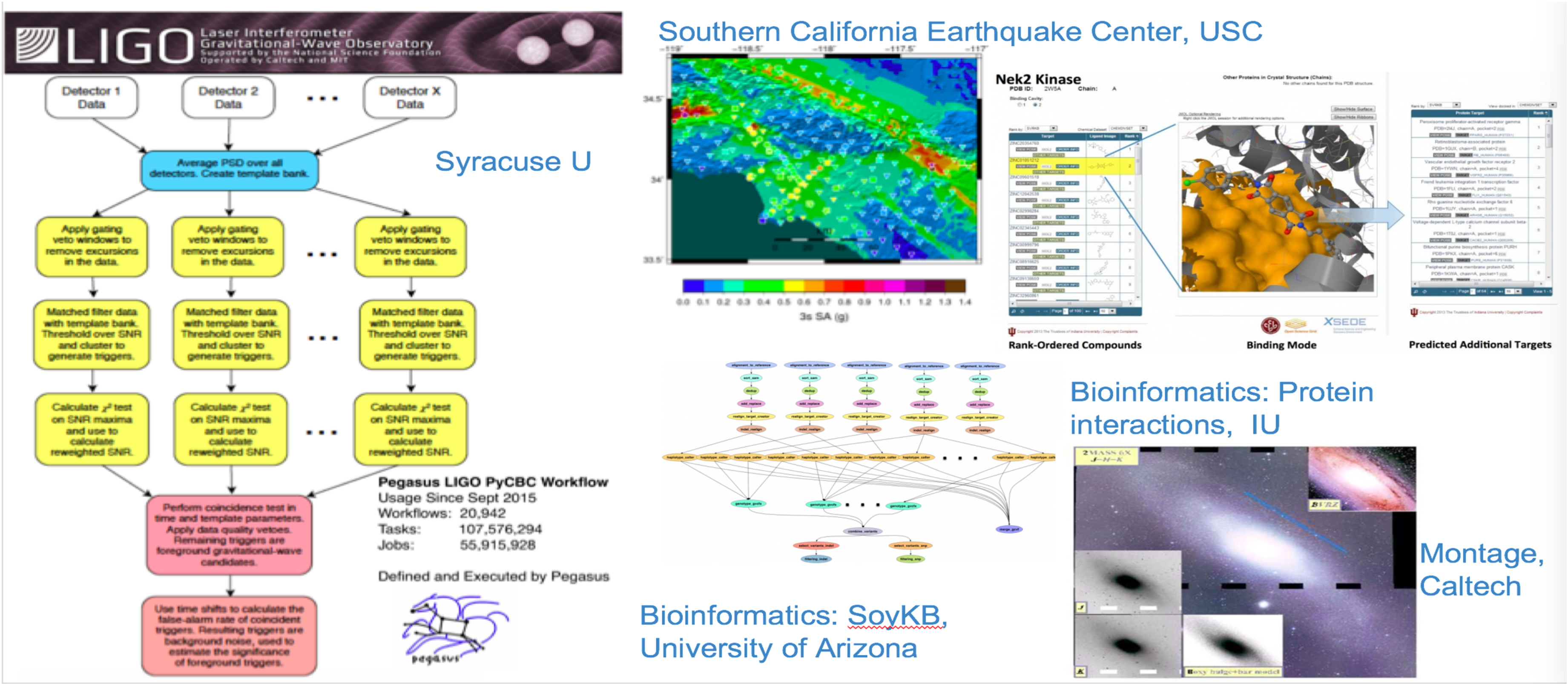
Currenty supported schedulers:

SLURM SGE PBS MOAB

Pegasus Workflow Management System, Production Use

Last 12 months: Pegasus users ran 240K workflows, 145M jobs

Majority of these include data transfers, using LAN, the Internet, local and remote storage



https://pegasus.isi.edu/

Questions?

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

Pegasus Online Office Hours

https://pegasus.isi.edu/blog/online-pegasus-office-hours/

Bi-monthly basis on second Friday of the month, where we address user questions and also apprise the community of new developments

Extra Slides: User Stories

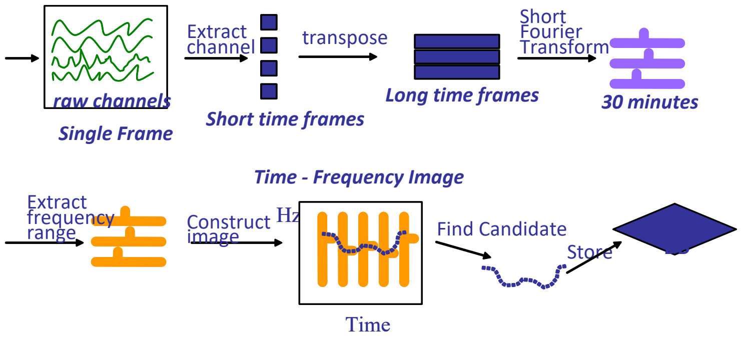
Pegasus: Grounding Research and Development

Pegasus

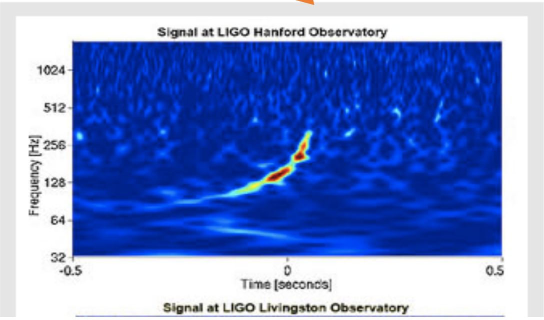
Nobel Prize

Working with LIGO

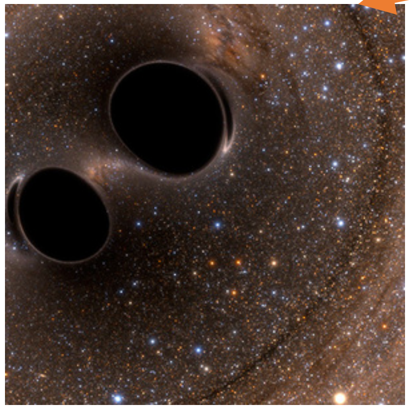
2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017



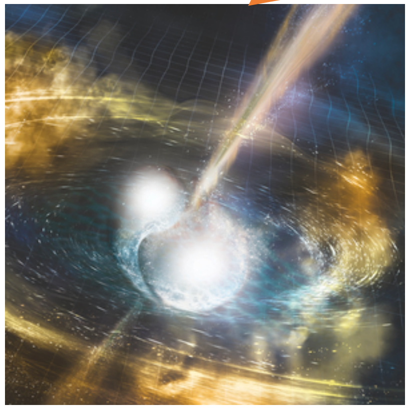
First Pegasus prototype for LIGO Pulsar Searches



Blind injection detection



First detection of black hole collision



Multi-messenger neutron star merger observation

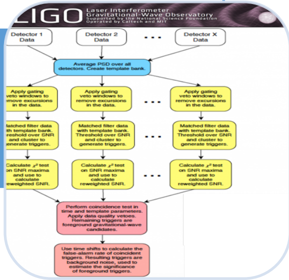
Image credit: LIGO Scientific Collaboration

Complexity of LIGO Workflows

First GW detection: ~ 21K Pegasus workflows, ~ 107M tasks

Analysis measures the statistical significance of collected data

Science Workflow



Efficient, scalable, and robust execution of tasks and data access

Automation

LIGO, Open Science Grid, XSEDE, Blue Waters

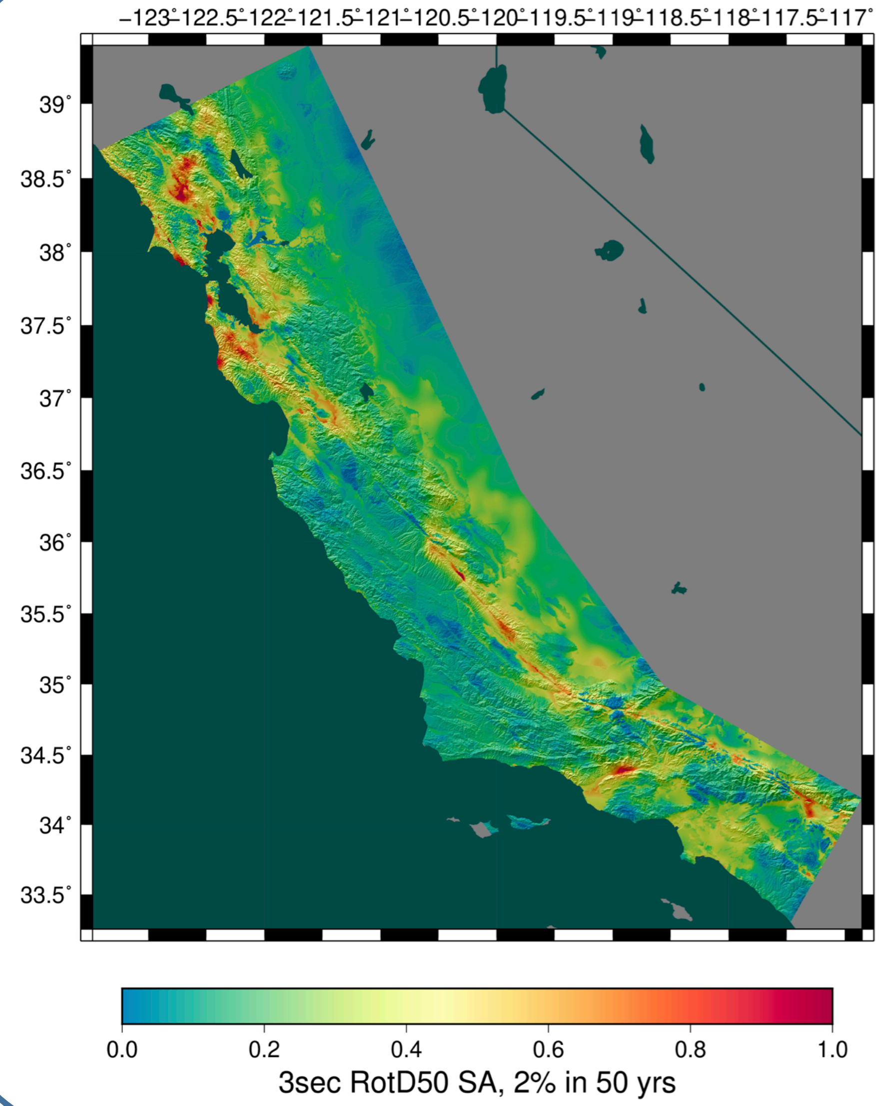
Distributed Power

2015/16

Supporting Heterogeneous Workflows

Pegasus

SCEC's CyberShake: What will the peak earthquake motion be over the next 50 years?



Useful information for:

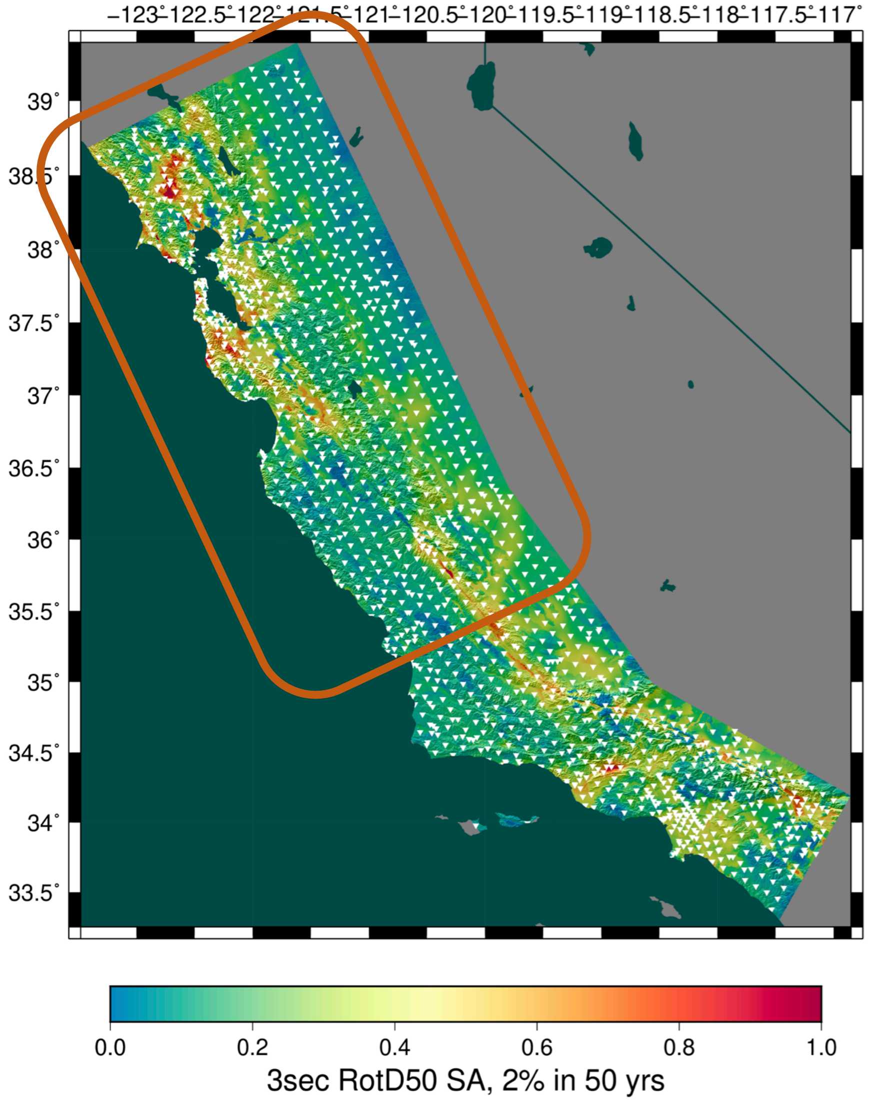
- Building engineers
- Disaster planners
- Insurance agencies

Slide credit: Southern California Earthquake Center

Pegasus

Supporting Heterogeneous Workflows

2018-2019 Mapping Northern California



- 120 million core-hours
- 39,285 jobs
- 1.2 PB of data managed
- 157 TB of data automatically transferred
- 14.4 TB of output data archived

- NCSA *Blue Waters*
- OLCF *Titan*

Total map:
170 million core hours
> 19,407 core years

Slide credit: Southern California Earthquake Center

HTCondor Week 2020

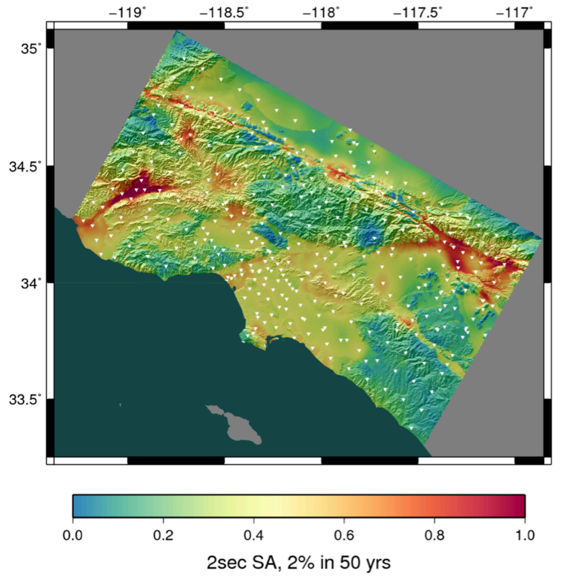
Mix Workloads on Heterogeneous/ Changing CI

Pegasus

Since 2007: 215 million core-hours (24,543 years)

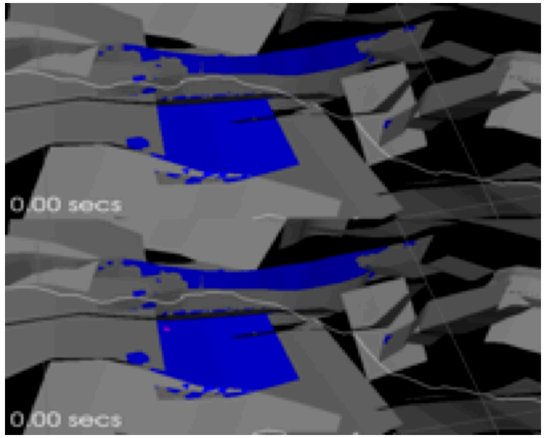
9 different supercomputers

Pegasus Optimizations:
- Task clustering
- MPI-based workflow engine



2010: World's first physics-based probabilistic seismic hazard map

Application Optimizations:
- Workflow restructuring
- MPI/code tuning
- Porting to GPUs



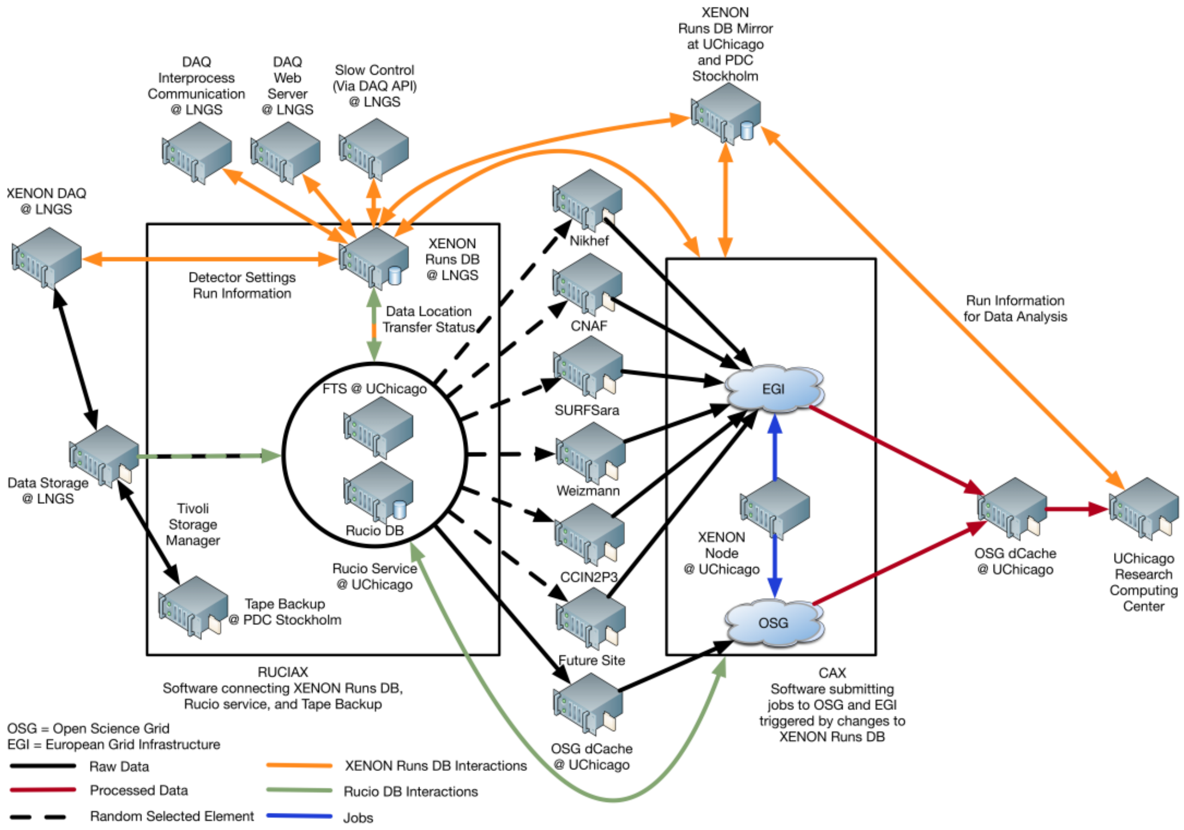
2018: Incorporating earthquake simulator with a 1 million-year catalog of California seismicity

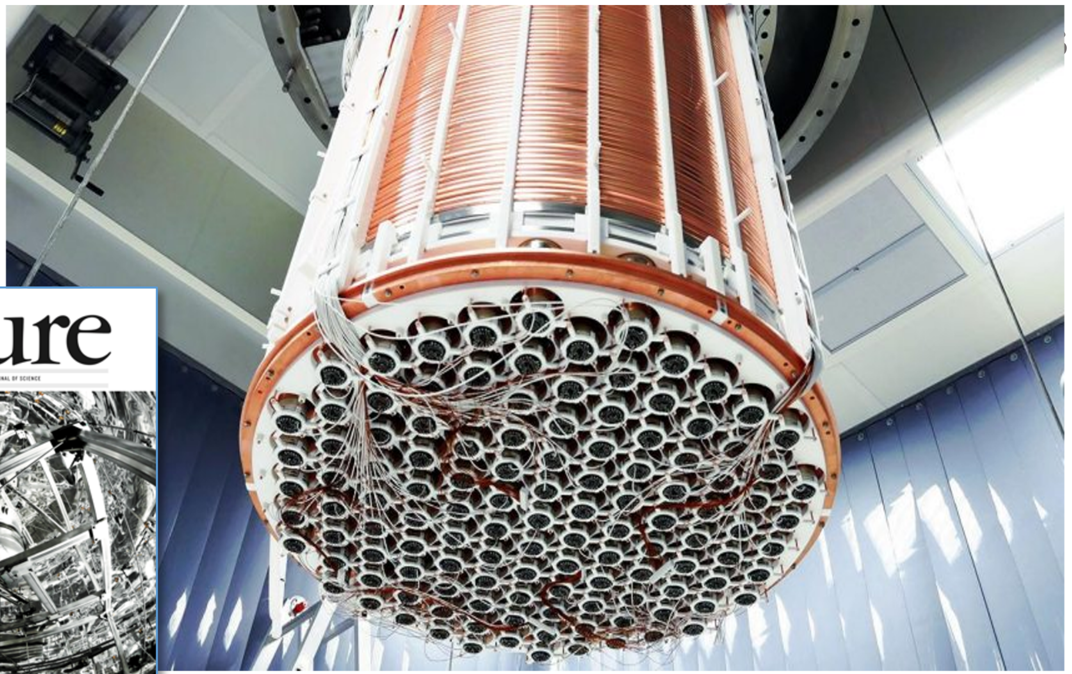
- USC HPCC
- NCSA Mercury
- TACC Ranger
- NICS Kraken
- TACC Stampede
- NCSA Abe
- ORNL Titan
- NCSA Blue Waters
- SDSC DataStar
- Summit
- Frontera

Slide credit: Southern California Earthquake Center

XENONnT - Dark Matter Search

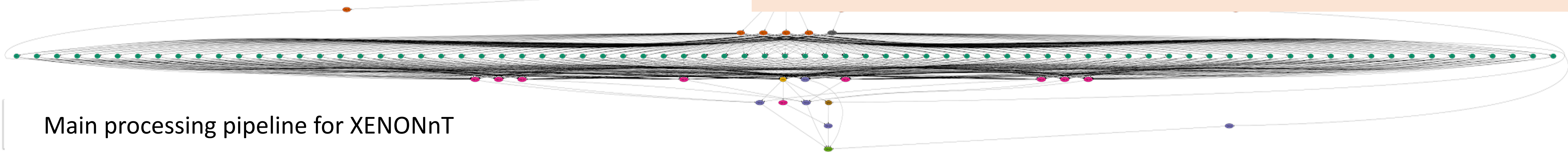
Detector at Laboratori Nazionali del Gran Sass (LNGS) in Italy. Data is distributed world-wide with Rucio. Workflows execute across Open Science Grid (OSG) and European Grid Infrastructure (EGI)





Type	Succeeded	Failed	Incomplete	Total	Retries	Total+Retries
Tasks	4000	0	0	4000	267	4267
Jobs	4484	0	0	4484	267	4751
Sub-Workflows	0	0	0	0	0	0

```
Workflow wall time                                       : 5 hrs, 2 mins
Cumulative job wall time                                 : 136 days, 9 hrs
Cumulative job wall time as seen from submit side        : 141 days, 16 hrs
Cumulative job badput wall time                          : 1 day, 2 hrs
Cumulative job badput wall time as seen from submit side : 4 days, 20 hrs
```



Main processing pipeline for XENONnT

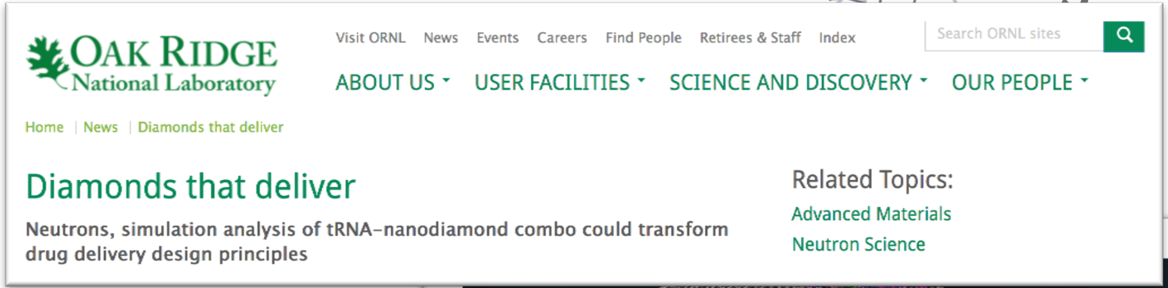
Impact on DOE Science

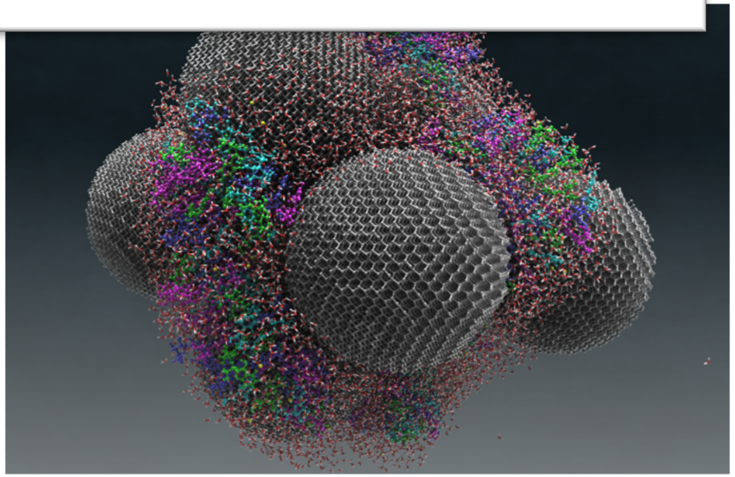
Enabled cutting-edge domain science (e.g., drug delivery) through collaboration with scientists at the DoE **Spallation Neutron Source (SNS)** facility

A Pegasus workflow was developed that confirmed that ***nanodiamonds*** can enhance the dynamics of tRNA

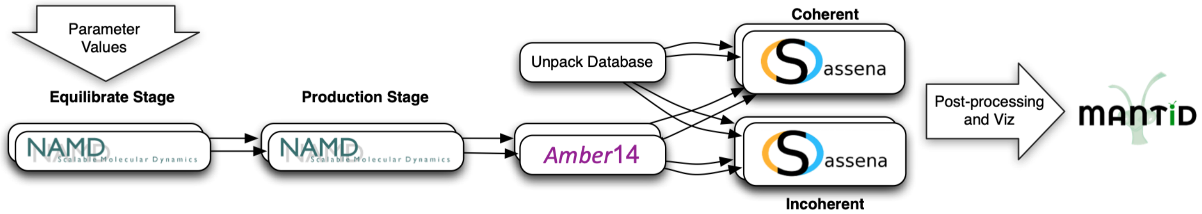
- It compared SNS neutron scattering data with MD simulations by calculating the epsilon that best matches experimental data

Ran on a Cray XE6 at NERSC using 400,000 CPU hours, and generated 3TB of data.





Water is seen as small red and white molecules on large nanodiamond spheres. The colored tRNA can be seen on the nanodiamond surface. (Image Credit: Michael Mattheson, OLCF, ORNL)



Predictive Modeling and Diagnostic Monitoring of Extreme Science Workflows

PANORAMA

An automated analysis workflow for optimization of force-field parameters using neutron scattering data*. V. E. Lynch, J. M. Borreguero, D. Bhowmik, P. Ganesh, B. G. Sumpter, T. E. Proffen, M. Goswami, Journal of Computational Physics, July 2017.*