Pegasus 5.0 Workflows
Workflow Management System

Karan Vahi, Ryan Tanaka
University of Southern California, School of Engineering Information Sciences Institute
vahi@isi.edu, tanaka@isi.edu

https://pegasus.isi.edu
1. Introduction
What are Scientific Workflows

- 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.
  - Resources distributed across Internet.

- 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.

Workflow Building Blocks

Slide Content Courtesy of David Okaya, SCEC, USC
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**
- Ensures **Data Integrity** during workflow execution

**NSF funded project since 2001, with close collaboration with HTCondor team**
Workflow Challenges Across Domains

- Describe complex workflows in a simple way
- Access distributed, heterogeneous data and resources (heterogeneous interfaces)
- Deal with resources/software that change over time
- Ease of use. Ability to debug and monitor large workflows

Our Focus

Separation between workflow description and workflow execution

Workflow planning and scheduling (scalability, performance)

Task execution (monitoring, fault tolerance, debugging, web dashboard)

Provide additional assurances that a scientific workflow is not accidentally or maliciously tampered with during its execution.
Key Pegasus Concepts

- **Pegasus WMS ==** Pegasus planner (mapper) + DAGMan workflow engine + HTCondor scheduler/broker
  - Pegasus maps workflows to infrastructure
  - DAGMan manages dependencies and reliability
  - HTCondor is used as a broker to interface with different schedulers

- **Workflows are DAGs**
  - Nodes: jobs, edges: dependencies
  - No while loops, no conditional branches
  - Jobs are standalone executables

- **Planning occurs ahead of execution**

- **Planning converts an abstract workflow into a concrete, executable workflow**
  - Planner is like a compiler

https://pegasus.isi.edu
**Input Workflow Specification** YAML formatted

**Portable Description**
Users do not worry about low level execution details

**Directed-Acyclic Graphs**

**Logical Filename (LFN)**
Platform independent (abstraction)

**Transformation**
Executables (or programs) platform independent

**Stage-in Job**
Transfers the workflow input data

**Cleanup Job**
Removes unused data

**Stage-out Job**
Stage-out generated output data

**Registration Job**
Registers the workflow output data

[https://pegasus.isi.edu](https://pegasus.isi.edu)
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

---

**LEGEND**
- Task flow + Checksums
- Data Stagein Job
- Data Stageout Job
- Check Integrity Job
- Checksum Generation Job
- Pegasus Lite Compute Job
- Worker Node (WN)

---

**Diagram Notes**
- SUBMIT HOST
- Compute Site 1
- Compute Site n
- Pegasus Lite Instance
- J1, J2
- T1, T2
- F.in, F.int, F.out
Pegasus-transfer

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

- 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)

Available Protocols:
- HTTP
- SCP
- GridFTP
- Globus
- Online
- iRods
- Amazon S3
- Google Storage
- SRM
- FDT
- Stashcp
- Rucio
- cp
- ln -s

https://pegasus.isi.edu
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.

https://pegasus.isi.edu
Provenance Data can be Summarized
Pegasus-Statistics or Used for Debugging Pegasus-Analyzer

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

Workflow wall time : 2 mins, 6 secs
Workflow cumulative job wall time as seen from submit side : 38 secs
Workflow cumulative job badput wall time :
Cumulative job badput wall time as seen from submit side :
And if a job fails?

**Postscript**
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

https://pegasus.isi.edu
Pegasus 5.0

- New and fresh Python3 API to compose, submit and monitor workflows, and configure catalogs
- New Catalog Formats
- Python 3 Support
  - All Pegasus tools are Python 3 compliant
  - 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 hierarchical workflows
- Reworked Documentation and Tutorial
  - https://pegasus.isi.edu/documentation/

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

from Pegasus.api import *

go

# 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()
```
2. Hands on Exercises

https://pegasus.isi.edu
Hands on Tutorial Exercises: Setup

It is the same (but hosted) as the self-guided tutorial available in the Pegasus documentation: [https://pegasus.isi.edu/documentation/user-guide/tutorial.html](https://pegasus.isi.edu/documentation/user-guide/tutorial.html)

Please claim an instance by putting you name next to an unused instance in: [shorturl.at/oxIO6](shorturl.at/oxIO6) (see Zoom chat for clickable link!)

Follow the link next to your name.
Docker Container / Jupyter Notebook

Container is for tutorial purposes - most production workflows have dedicated submit hosts

Jupyter is optional. You can choose to use just the workflow abstraction API, the full workflow management API, inside or outside Jupyter.
Docker Container / Jupyter Notebook

Production Setup

Host Machine (Your Laptop)

File System

File1

job1

file2

job2

file3

Workflow

HTCondor

Pegasus

aws

/var

usr

tmp

/HTCondor/2019-05-06

/var

usr

tmp

/var

usr

tmp

/var

usr

tmp

/var

usr

tmp
Docker Container / Jupyter Notebook

Development Setup

Host Machine (Your Laptop)

Development Container

Pegasus

HTCondor

Workflow

File System

localhost:8888
2.1 API
Key Pegasus Concepts

▲ **Pegasus WMS ==** Pegasus planner (mapper) + DAGMan workflow engine + HTCondor scheduler/broker

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

▲ **Workflows are DAGs**

- Nodes: jobs, edges: dependencies
- No while loops, no conditional branches
- Jobs are standalone executables

▲ **Planning occurs ahead of execution**

▲ **Planning converts an abstract workflow into a concrete, executable workflow**

- Planner is like a compiler
2.2 Debugging
2.3 Command Line Tools

https://pegasus.isi.edu
2.4 Summary
15 Minute Break
3. Advanced Topics
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...

Typically Most HPC Sites

Submit Host

Shared Filesystem

Input data site
Data staging site
Output data site

https://pegasus.isi.edu
Cloud Computing
High-scalable object storages

Typical cloud computing deployment (Amazon S3, Google Storage)
Grid Computing
Local data management

Submit Host

Compute Site

Typical OSG sites
Open Science Grid
Running fine-grained workflows on HPC systems...

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
Performance.
Why not improve it?

Clustered Job
Groups small jobs together to improve performance

Task
Small granularity
Pegasus also handles large-scale workflows

Recursion ends
When abstract workflow with only compute jobs is encountered

https://pegasus.isi.edu
Data Reuse prune jobs if output data already exists

Jobs which output data is already available are pruned from the DAG
And if a job fails?

Postscript
- 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

https://pegasus.isi.edu
Metadata
Can associate arbitrary key-value pairs with workflows, jobs, and files

Data Registration
Output files get tagged with metadata on registration in the workflow database

Static and Runtime Metadata

Static: application parameters
Runtime: performance metrics

x-pegasus:
apilang: python
createdBy: vahi
createdOn: 2023-08-20T10:08:48Z
pegasus: "5.0"
name: diamond
metadata:
  experiment: "par_all27_prot_lipid"
  jobs:
    - type: "job"
      name: "namd"
      id: "2D00000001"
    arguments: ["equilibrate.conf"]
  uses:
    - lfn: "Q42.psf"
      metadata:
        type: "psf"
        charge: "-42"
        type: "input"
    - lfn: "eq.restart.coord"
      type: "output"
      metadata:
        type: "coordinates"
        stageOut: true
        registerReplica: true
      metadata:
        timesteps: 500000
        temperature: 200
        pressure: 1.01353

Select Data Based On Metadata

Register Data With Metadata

Workflow, Job, File

Related Tools:
- Netlogger: Events
- Runtime Metadata
- Static metadata from DAX and catalogs
- Collected from Kickstart records
- Python Metadata API
- S3
- IRODS
- Pegasus dashboards
- Pegasus: metadata (command line tool)

USER

https://pegasus.isi.edu
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 Container Support

Users can refer to containers in the Transformation Catalog with their executable preinstalled.

Users can refer to a container they want to use – Pegasus stages their executables and containers to the node.

- Useful if you want to use a site recommended/standard container image.
- Users are using generic image with executable staging.

Future Plans

- Users can specify an image buildfile for their jobs.
- Pegasus will build the Docker image as separate jobs in the executable workflow, export them as a tar file and ship them around.

Containers Execution Model

- Directory Setup
- Pull image
- Start container
- $PWD bind-mounted as/srv
- Pull worker package (if needed)
- Set job environment
- Stage in inputs
- Execute user application
- Stage out outputs
- Stop container
- Cleanup

https://pegasus.isi.edu
Data Management for Containers

Containers are data too!

Pegasus treats containers as input data dependency
- Staged to compute node if not present
- Docker or Singularity Hub URL’s
- Docker Image exported as a TAR file and available at a server, just like any other input dataset

Scaling up for larger workflows
- 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
- Pricing considerations. You are now charged if you exceed a certain rate of pulls from Hubs

Other Optimizations
- Symlink against existing images on shared file system such as CVMFS
- The exported tar file is then shipped with the workflow and made available to the jobs
Job Submissions

LOCAL

Submit Machine
Personal HTCondor

Local Campus Cluster accessible via Submit Machine**
HTCondor via BLAHP

** Both Glite and BOSCO build on HTCondor BLAHP

Currently supported schedulers:
SLURM SGE PBS MOAB

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

https://pegasus.isi.edu
Credentials Management

Credentials required for two purposes
- Job Submission
- Data transfers to stage-in input and stage-out generated outputs when a job executes

Specifying Credentials
- Users can specify credentials in a generic credentials file on submit host
- Associate credentials with sites in site catalog

Approach
- Planner will automatically associate the required credentials with each job
- The credentials are transferred along with the job
- Usually available only for the duration of the job execution

Supported Credentials
- X.509 grid proxies
- Amazon AWS S3 keys
- Google Cloud Platform OAuth token (.boto file)
- iRods password
- SSH keys
- Web Dav

https://pegasus.isi.edu
Amazon AWS Batch

AWS Batch

Container based, dynamically scaled and efficient batch computing service

Automatically launches compute nodes in Amazon based on demand in the associated job queue

Users can specify compute environment that dictates what type of VM’s are launched

Pegasus will allow clusters of jobs to be run on Amazon EC2 using AWS Batch Service

New command line tool: `pegasus-aws-batch`

Automates most of the batch setup programmatically

- Sets up and Deprovisions
  - Compute Environment
  - Job Queues
- Follows AWS Batch HTTP specification
Ensemble Manager

Allow users to submit a collection of workflows (ensembles)
Automatically spawn and manage collections of workflows

Trigger submission of workflows

Properties
Workflows within an ensemble may have different priorities
> Priorities can also be changed at runtime
Ensembles may limit the number of concurrent planned and running workflows

Additional Actions
Ensembles can be paused, resumed, removed, re-planned, and re-executed
A debugging mechanism is also provided to investigate failures in workflow runs
Actions can be performed both to ensembles and single workflows within ensembles
Ensemble Manager Triggers

**Cron workflow trigger**
Automatically submit workflows to the ensemble manager at regularly occurring time intervals

**File pattern workflow trigger**
Cron trigger functionality
New input files matching a given file pattern(s) will be passed as input
Ideal for regular batch processing of data as it arrives in one or more given directories
Ensemble Manager Overview
Ensemble Manager Overview

Ensemble 1
- Priority: 100
- max_running = 2

Ensemble 2
- Priority: 100
- max_running = 1

Ensemble management

Individual workflow planning, submission, & monitoring

Workflow execution

https://pegasus.isi.edu
Ensemble Manager Overview
Get Started

- **Pegasus Website**
  https://pegasus.isi.edu

- **Users Mailing List**
  pegasus-users@isi.edu

- **Support**
  pegasus-support@isi.edu

- **Slack**
  Ask for an invite by trying to join pegasus-users.slack.com in the Slack app

- **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.**