Pegasus 5.0 + Ensemble Manager
Workflow Management System

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

https://pegasus.isi.edu
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

https://pegasus.isi.edu
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
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: Orange
- Directory Setup Job: Light Blue
- Data Stageout Job: Green
- Check Integrity Job: Red
- Pegasus Lite Compute Job: Gray
- Worker Node (WN): Purple
- Data Stagein Job: Pink
- Directory Cleanup Job: Dark Blue
- Checksum Generation Job: Maroon

---

**Diagram**

- **F.in**
  - T1
  - F.int
  - T2
  - F.out

- **Input Data Site**: F.in
- **Staging Site**: F.int
- **Output Data Site**: F.out
Advanced LIGO

Data Flow for LIGO Pegasus Workflows in OSG

- **Nodes from OSG & LIGO Sites managed By Glidein WMS**
- **HTTP Squid Cache**
- **Nodes from OSG & LIGO Sites managed By Glidein WMS**

**Legend**
- Directory Setup Job
- Data Stageout Job
- Directory Cleanup Job
- Pegasus Lite Compute Job
- Worker Node

**Key Points**
- **60,000 Compute Tasks**
- **Input Data:** 5000 files (10GB total)
- **Output Data:** 60,000 files (60GB total)
- **Processed Data:** 725 GB

**Execution Sites:**
- LIGO Data Grid
- EGI
- Open Science Grid
- XSEDE

**Details**
- Input data hosted at LIGO Sites
- Nebraska GridFTP Data staging server
- GridFTP, HTTP, SRM
- LIGO output data server

**Workflow Phases**
1. Workflow stagein Job stages in the input data for workflow from user server.
2. Pegasus lite instance looks up input data on the compute node/CVMFS. If not present, stage-in data from remote data staging server.
3. Workflow stageout Job stages produced data from data staging server to a LIGO output data server.
4. Pegasus lite instance stages out job output data from worker node to data staging server.
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 are generated and tracked within the system.

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 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/

```python
#!/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()
```
Pegasus 5.0

• Zero configuration required to submit to local HTCondor pool.
  • The "hello world" example on the right will work out of the box.
  • Pegasus will automatically create sensible defaults for sites
    • local
    • condorpool
  • By default, site "condorpool" is used as execution site.
  • Site "local" still designates the submit node, and is used to run Pegasus auxilliary jobs.

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

from Pegasus.api import *

logging.basicConfig(level=logging.DEBUG, stream=sys.stdout)

# --- Transforms --------------------------------------------

trans = Transformation(
    "echo",
    pfn="/bin/echo",
    site="condorpool"
)

tc = TransformationCatalog()
    .add_transformation(trans)

# --- 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()
```
Ensemble Manager

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

Trigger submission of workflows

Properties
Workflow 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

https://pegasus.isi.edu
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

Pegasus Ensemble Manager

Ensemble 1
Priority: 100
max_running = 2

Ensemble 2
Priority: 100
max_running = 1

Priority: 90

individual workflow
planning, submission,
& monitoring

workflow execution
Ensemble Manager: Rest API

- Exposing the ensemble manager as a REST endpoint

- Provide create, read, update, delete operations on ensembles, workflows, and triggers

- Provide Python and Java client code

- Improve support for integrating ensemble manager into larger systems
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

YouTube Channel

https://www.youtube.com/channel/UCwJQln1CqBvTJqiNr9X9F1Q/featured

---

**Pegasus in 5 Minutes**