Pegasus - a dHTC friendly workflow manager

Mats Rynge
rynge@isi.edu

https://pegasus.isi.edu
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
- All data is tracked

Pegasus takes this and generates an executable workflow
- Data management tasks added
- Transforms the workflow for performance and reliability
- HTCondor DAGMan DAG

Planning occurs before execution
- New and fresh Python3 API to compose, submit and monitor workflows, and configure catalogs
- New Catalog Formats
- Python 3
  - 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
- Major documentation improvements
  - [https://pegasus.isi.edu/docs/5.0.0dev/index.html](https://pegasus.isi.edu/docs/5.0.0dev/index.html)
Optimizations

Hierarchical workflows
- Enacts the execution of millions of tasks
- Also enables loops and conditionals in DAGs

Task clustering
- Original Workflow
- Workflow After Clustering

Task-resource co-allocation
- CPU #1
- CPU #2

Data Reuse
- Data already available
- Data also available
- workflow reduction
- data reuse
- Jobs which output data is already available are pruned from the DAG
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

[Image of workflow diagrams]

[Link: https://pegasus.isi.edu/]
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
**pegasus-transfer**

Directory creation, file removal
- If protocol can support it, also used for cleanup

Two stage transfers between incompatible protocols
- e.g., GridFTP to S3 is executed as: 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)
Containers are data too!

Users can specify to use images from Docker Hub, Singularity Library, or a file using URLs

The image is pulled down as a tar file as part of data stage-in jobs in the workflow

- The exported tar file / image file is then transferred to the job as any other piece of data
- Motivation: Avoid overwhelming Docker Hub/Singularity Library/… with by repeatedly requesting the same image
- Motivation: Optimize workflow data placement and movement

Symlink against a container image if available on shared file systems. For example, CVMFS hosted images on Open Science Grid
Advanced LIGO – Laser Interferometer Gravitational Wave Observatory

40,000 compute tasks
Inputs files: 1,100
Output files: 63
Processed Data: 725 GB

Executing on LIGO Data Grid, EGI, Open Science Grid and XSEDE
Pegasus performs integrity checksums on input files right before a job starts, ensuring the computation is on the expected piece of data.

- For inputs from external sources, checksums specified in the input replica catalog along with file locations, or generated first time we encounter the file.
- All intermediate and output files checksums are generated and tracked within the system.

Checksums validation failures is a job failure.
VERITAS / Nepomuk Otte, GATech

Seeing very small, but steady stream of corrected integrity errors from reporting back to Pegasus dashboard.

For VERITAS, \(\sim 0.04\%\) of transfers fail with integrity errors. (~1 / 2,500 transfers)

Cause uncertain (diagnosis is harder than detection).

Possibly errors in http based transfers (s3 protocol against CEPH)
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*
Initial Results with Integrity Checking on

• OSG-KINC workflow (50,606 jobs) encountered 60 integrity errors in the wild (production OSG). The problematic jobs were automatically retried and the workflow finished successfully.

• The 60 errors took place on 3 different hosts. The first one at UColorado, and group 2 and 3 at UNL hosts.

• Error Analysis (by hand)
  
  • 1 input file error at University of Colorado.
  
  • 3 input file (kinc executable) errors on one node at University of Nebraska. The timespan across the failures was 16 seconds. We suspect that the node level cache got corrupted.
  
  • 56 input file errors on a different compute nodes at University of Nebraska. The timespan across the failures was 1,752 seconds. We suspect that the site level cache got corrupted.
Abstract

Portable Description
Users do not worry about low level execution details

Executable

DAGMan

- **stage-in job**: Transfers the workflow input data
- **cleanup job**: Removes unused data
- **stage-out job**: Transfers the workflow output data
- **registration job**: Registers the workflow output data

**Portable Description**

- **logical filename (LFN)**: platform independent (abstraction)
- **transformation**: executables (or programs) platform independent
**XENONnT - Dark Matter Search**

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Succeeded</th>
<th>Failed</th>
<th>Incomplete</th>
<th>Total</th>
<th>Retries</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total+Retries</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tasks</td>
<td>4000</td>
<td>0</td>
<td>0</td>
<td>4000</td>
<td>267</td>
</tr>
<tr>
<td>Jobs</td>
<td>4484</td>
<td>0</td>
<td>0</td>
<td>4484</td>
<td>267</td>
</tr>
<tr>
<td>Sub-Workflows</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

---

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 output failure wall time: 1 day, 2 hrs
Cumulative job output failure wall time as seen from submit side: 4 days, 20 hrs
Why Pegasus?

Automates complex, multi-stage processing pipelines

Enables parallel, distributed or remote 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 data integrity

NSF funded project since 2001, with close collaboration with HTCondor team
Running Pegasus workflows with Jupyter

![Diagram showing Pegasus workflows with Jupyter and different network connections (WAN, LAN) between Campus Cluster, HPC/HTC, and Clouds.]
Provenance data can be summarized with `pegasus-statistics` or used for debugging with `pegasus-analyzer`.
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 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.