Before we start

Hands on Exercises Notes  
https://pegasus.isi.edu/tutorial/chtc/

System  
learn.chtc.wisc.edu

Training Accounts  
Pick up from the instructor
Pegasus Scientific Workflows with Containers

Pegasus Workflow Management System

Karan Vahi
## OUTLINE

<table>
<thead>
<tr>
<th>Introduction</th>
<th>Scientific Workflows</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pegasus Overview</td>
</tr>
<tr>
<td></td>
<td>Successful Stories</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pegasus Overview</th>
<th>Basic Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Features</td>
</tr>
<tr>
<td></td>
<td>System Architecture</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hands-on Tutorial</th>
<th>Submitting a Workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Workflow Dashboard and Monitoring</td>
</tr>
<tr>
<td></td>
<td>Generating the Workflow</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Understanding Pegasus Features</th>
<th>Information Catalogs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Containers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hands-on Tutorial</th>
<th>Workflows with Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clustering</td>
</tr>
<tr>
<td></td>
<td>Fault-Tolerance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other Features</th>
<th>Data Staging</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Jupyter Notebooks</td>
</tr>
<tr>
<td></td>
<td>Metadata, Hierarchal Workflows, Data Reuse</td>
</tr>
</tbody>
</table>
Compute Pipelines

Building Blocks

Allows scientists to connect different codes together and execute their analysis.

Pipelines can be very simple (independent or parallel) jobs or complex represented as DAG’s.

Helps users to automate scale up.

However, it is still up-to user to figure out.

Data Management
How do you ship in the small/large amounts data required by your pipeline and protocols to use?

How best to leverage different infrastructure setups?
OSG has no shared filesystem while XSEDE and your local campus cluster has one!

Debug and Monitor Computations
Correlate data across lots of log files
Need to know what host a job ran on and how it was invoked.

Restructure Workflows for Improved Performance
Short running tasks? Data placement.

http://pegasus.isi.edu
Why Pegasus?

**Automate** 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**

---

NSF funded project since 2001, with close collaboration with HTCondor team

[Http://pegasus.isi.edu](http://pegasus.isi.edu)
Some of the successful stories...
60,000 compute tasks
Input Data: 5000 files (10GB total)
Output Data: 60,000 files (60GB total)

executed on LIGO Data Grid, Open Science Grid and XSEDE
Advanced LIGO PyCBC Workflow

One of the main pipelines to measure the statistical significance of data needed for discovery

Contains **100’s of thousands of jobs** and accesses on order of **terabytes of data**

Uses data from multiple detectors

For the detection, the pipeline was executed on Syracuse and Albert Einstein Institute Hannover

A single run of the binary black hole + binary neutron star search through the O1 data (about 3 calendar months of data with 50% duty cycle) requires a **workflow** with **194,364 jobs**

Generating the final O1 results with all the review required for the first discovery took about **20 million core hours**
Southern California Earthquake Center’s CyberShake

 Builders ask seismologists: What will the peak ground motion be at my new building in the next 50 years?

Seismologists answer this question using Probabilistic Seismic Hazard Analysis (PSHA)

CPU jobs (Mesh generation, seismogram synthesis): 1,094,000 node-hours

GPU jobs: 439,000 node-hours

AWP-ODC finite-difference code

5 billion points per volume, 23000 timesteps

200 GPUs for 1 hour

Titan:

421,000 CPU node-hours, 110,000 GPU node-hours

Blue Waters:

673,000 CPU node-hours, 329,000 GPU node-hours

286 sites, 4 models each workflow has 420,000 tasks
Soybean Workflow

TACC Wrangler as Execution Environment

Flash Based Shared Storage

Switched to glideins (pilot jobs) - Brings in remote compute nodes and joins them to the HTCondor pool on the submit host - Workflow runs at a finer granularity

Works well on Wrangler due to more cores and memory per node (48 cores, 128 GB RAM)
# OUTLINE

| Introduction | Scientific Workflows  
|             | Pegasus Overview  
|             | Successful Stories  |

| Pegasus Overview | Basic Concepts  
|                  | Features  
|                  | System Architecture  |

| Hands-on Tutorial | Submitting a Workflow  
|                   | Workflow Dashboard and Monitoring  
|                   | Generating the Workflow  |

| Understanding Pegasus Features | Information Catalogs  
|                                | Containers  |

| Hands-on Tutorial | Workflows with Containers  
|                   | Clustering  
|                   | Fault-Tolerance  |

| Other Features | Data Staging  
|               | Jupyter Notebooks  
|               | Metadata, Hierarchal Workflows, Data Reuse  |
Basic concepts...
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
**Portable Description**

Users do not worry about low level execution details.

**logical filename (LFN)**
platform independent (abstraction)

**transformation**
executables (or programs)
platform independent

**directed-acyclic graphs (DAG)**

**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

**executable workflow**
Pegasus also provides tools to generate the abstract workflow.
An example
Split Workflow

```python
#!/usr/bin/env python
import os, pwd, sys, time
from Pegasus.DAX3 import *

# Create an abstract dag
dax = ADAG("split")

webpage = File("pegasus.html")

# 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)
# associate the label with the job. all jobs with same label
# are run with PMC when doing job clustering
split.addProfile(Profile("pegasus","label","p1"))
dax.addJob(split)

# we do a parameter 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)

f = open("split.dax", "w")
dax.writeXML(f)
f.close()
```

Visualization Tools:
- pegasus-graphviz
- pegasus-plots

https://pegasus.isi.edu/documentation/tutorial_submitting_wf.php
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
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.
Provenance data can be summarized with `pegasus-statistics` or used for debugging with `pegasus-analyzer`.

Example command-line:

```plaintext
$ pegasus-status pegasus/examples/split/run0001

$ pegasus-analyzer pegasus/examples/split/run0001
PEGASUS-ANALYZER: initializing...

PEGASUS-ANALYZER: summary...
```

```
<table>
<thead>
<tr>
<th>Type</th>
<th>Succeeded</th>
<th>Failed</th>
<th>Incomplete</th>
<th>Total Retries</th>
<th>Total+Retries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Jobs</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>17</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: 2 mins, 6 secs
Workflow cumulative job wall time: 38 secs
Cumulative job wall time as seen from submit side: 42 secs
Workflow cumulative job badput wall time:
Cumulative job badput wall time as seen from submit side:
<table>
<thead>
<tr>
<th><strong>OUTLINE</strong></th>
</tr>
</thead>
</table>
| **Introduction**  | *Scientific Workflows*  
 |        | *Pegasus Overview*  
 |        | *Successful Stories*  |
| **Pegasus Overview**  | *Basic Concepts*  
 |        | *Features*  
 |        | *System Architecture*  |
| **Hands-on Tutorial**  | *Submitting a Workflow*  
 |        | *Workflow Dashboard and Monitoring*  
 |        | *Generating the Workflow*  |
| **Understanding Pegasus Features**  | *Information Catalogs*  
 |        | *Containers*  |
| **Hands-on Tutorial**  | *Workflows with Containers*  
 |        | *Clustering*  
 |        | *Fault-Tolerance*  |
| **Other Features**  | *Data Staging*  
 |        | *Jupyter Notebooks*  
 |        | *Metadata, Hierarchical Workflows, Data Reuse*  |
Hands-on Pegasus Tutorial...
Hands On Tutorial

• SSH to our training machine
  • Login with your user’s tutorial login and password
  • ssh trainXX@learn.chtc.wisc.edu

• Open exercise notes in your browser
  • https://pegasus.isi.edu/tutorial/chtc/tutorial.php
Understanding Pegasus features...
So, what information does Pegasus need?

- **Site Catalog**: describes the sites where the workflow jobs are to be executed.
- **Transformation Catalog**: describes all of the executables (called “transformations”) used by the workflow.
- **Replica Catalog**: describes all of the input data stored on external servers.
How does Pegasus decide where to execute?

- **site description**
  - describes the compute resources

- **scratch**
  - tells where temporary data is stored

- **storage**
  - tells where output data is stored

- **profiles**
  - key-pair values associated per job level

---

```xml
<site description="site description" handle="local" arch="x86_64" os="LINUX">
  <!-- These are the paths on the submit host where Pegasus stores data -->
  <directory type="shared-scratch" path="/home/tutorial/run">
    <file-server operation="all" url="file:///home/tutorial/run"/>
  </directory>

  <!-- Scratch is where temporary files go -->
  <directory type="local-storage" path="/home/tutorial/outputs">
    <file-server operation="all" url="file:///home/tutorial/outputs"/>
  </directory>

  <!-- This profile tells Pegasus where to find the user's private key -->
  <profile namespace="env" key="SSH_PRIVATE_KEY">
    /home/tutorial/.ssh/id_rsa
  </profile>
</site>
```

http://pegasus.isi.edu
How does it know where the executables are or which ones to use?

executables description
list of executables locations per site

physical executables
mapped from logical transformations

transformation type
whether it is installed or available to stage

# This is the transformation catalog. It lists information about each of the executables that are used by the workflow.

```bash
tr ls {
  site PegasusVM {
    pfn "/bin/ls"
    arch "x86_64"
    os "linux"
    type "INSTALLED"
  }
}
...
```
What if data is not local to the submit host?

# This is the replica catalog. It lists information about each of the input files used by the workflow. You can use this to specify locations to input files present on external servers.

# The format is:  
# LFN PFN site="SITE"

f.a file:///home/tutorial/examples/diamond/input/f.a site="local"

logical filename  
abstract data name

physical filename  
data physical location on site different transfer protocols can be used (e.g., scp, http, ftp, gridFTP, etc.)

site name  
in which site the file is available
Replica catalog

multiple sources

**pegasus.conf**

# Add Replica selection options so that it will try URLs first, then
# XrootD for OSG, then gridftp, then anything else
pegasus.selector.replica.regex.rank.1=file:///cvmfs/.*
pegasus.selector.replica.regex.rank.2=file://.*
pegasus.selector.replica.regex.rank.3=root://.*
pegasus.selector.replica.regex.rank.4=gridftp://.*
pegasus.selector.replica.regex.rank.5=.*

**rc.data**

# This is the replica catalog. It lists information about each of the
# input files used by the workflow. You can use this to specify locations
# to input files present on external servers.

# The format is:
# LFN PFN site="SITE"

f.a file:///cvmfs/oasis.opensciencegrid.org/diamond/input/f.a site="cvmfs"
f.a file:///local-storage/diamond/input/f.a site="prestaged"
f.a gridftp://storage.mysite/edu/examples/diamond/input/f.a site="storage"
Pegasus Container Support

• Support for
  • Docker
  • Singularity – Widely supported on OSG

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

• Users can refer to a container they want to use. However, they let Pegasus stage their executable 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 at tar file and ship them around* (planned for 4.8.X)
Data Management for Containers

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

- We want to avoid hitting Docker/Singularity Hub repeatedly for large workflows
  - Extend pegasus-transfer to pull image from Docker Hub and then export it as tar file, that can be shipped around in the workflow.

- Ensure pegasus worker package gets installed at runtime inside the user container.
**OUTLINE**

<table>
<thead>
<tr>
<th>Introduction</th>
<th>Scientific Workflows</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pegasus Overview</td>
</tr>
<tr>
<td></td>
<td>Successful Stories</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pegasus Overview</th>
<th>Basic Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Features</td>
</tr>
<tr>
<td></td>
<td>System Architecture</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hands-on Tutorial</th>
<th>Submitting a Workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Workflow Dashboard and Monitoring</td>
</tr>
<tr>
<td></td>
<td>Generating the Workflow</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Understanding Pegasus Features</th>
<th>Information Catalogs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Containers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hands-on Tutorial</th>
<th>Workflows with Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clustering</td>
</tr>
<tr>
<td></td>
<td>Fault-Tolerance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other Features</th>
<th>Data Staging</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Jupyter Notebooks</td>
</tr>
<tr>
<td></td>
<td>Metadata, Hierarchical Workflows, Data Reuse</td>
</tr>
</tbody>
</table>
Hands-on Pegasus Tutorial...
OUTLINE

Introduction  
Scientific Workflows  
Pegasus Overview  
Successful Stories

Pegasus Overview  
Basic Concepts  
Features  
System Architecture

Hands-on Tutorial  
Submitting a Workflow  
Workflow Dashboard and Monitoring  
Generating the Workflow

Understanding Pegasus Features  
Information Catalogs  
Containers

Hands-on Tutorial  
Workflows with Containers  
Clustering  
Fault-Tolerance

Other Features  
Data Staging  
Jupyter Notebooks  
Metadata, Hierarchal Workflows, Data Reuse
A few more features...
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
There are several possible configurations...

Submit host (e.g., user’s laptop)

Typically most HPC sites
**Cloud Computing**

Typical cloud computing deployment (Amazon S3, Google Storage)

Submit host (e.g., user's laptop)

Compute Site

Object storage

Input data site
Data staging site
Output data site

Staging Site

High-scalable object storages
Grid Computing

local data management

submit host
(e.g., user’s laptop)

Typical OSG sites
Open Science Grid

http://pegasus.isi.edu
And yes... you can mix everything!
pegasus-transfer

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

Directory creation, file removal
   If protocol supports, 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 3\textsuperscript{rd} party transfers)
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

HPC System

Master (rank 0)

worker
rank 1

rank n-1

workflow restructuring
workflow reduction
hierarchical workflows

pegasus-mpi-cluster

http://pegasus.isi.edu
Performance, why not improve it?

- clustered job
  - Groups small jobs together to improve performance

- task
  - small granularity

Workflow restructuring
Workflow reduction
Hierarchical workflows
Pegasus-mpi-cluster

http://pegasus.isi.edu
And if a job fails?

**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
Running Pegasus workflows with Jupyter
importing the API

from Pegasus.jupyter.instance import *

creating an instance of the DAX

instance = Instance(dax)

running a workflow

instance.run(site='condorpool')

using the Pegasus DAX3 API to write a workflow

# Create an abstract dag
dax = ADAG("split")

# 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)

# associate the label with the job. All jobs with same label
# are run with PMC when doing job clustering
split.addProfile(Profile("pegasus","label","p1"))
dax.addJob(split)

monitoring a workflow execution

instance.status(loop=True, delay=5)

Progress: 100.0% (Success) (Completed: 17, Queued: 0, Running: 0, Failed: 0)
Data registration

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

Static and runtime metadata

Static: application parameters
Runtime: performance metrics

Metadata

Can associate arbitrary key-value pairs with workflows, jobs, and files

<adag ...
 <metadata key="experiment">par_all27_prot_lipid</metadata>
 <job id="ID0000001" name="namd">
   <argument><file name="equilibrate.conf"/></argument>
   <metadata key="timesteps">500000</metadata>
   <metadata key="temperature">200</metadata>
   <metadata key="pressure">1.01325</metadata>
   <uses name="O42.psf" link="input">
     <metadata key="type">psf</metadata>
     <metadata key="charge">42</metadata>
   </uses>
   ...
   <uses name="eq.restart.coord" link="output" transfer="false">
     <metadata key="type">coordinates</metadata>
   </uses>
   ...
 </job>
</adag>

workflow, job, file

select data based on metadata

register data with metadata

Pegasus
http://pegasus.isi.edu

since Pegasus 4.6
What about **data reuse**?

- **Data already available**
- **Data also available**

Jobs which output data is already available are pruned from the DAG.
Pegasus also handles large-scale workflows

Workflow restructuring
Workflow reduction
Hierarchical workflows

Pegasus-mpi-cluster

Recursion ends when DAX with only compute jobs is encountered

http://pegasus.isi.edu
Job Submissions

**Local**

Submit Machine
   Personal HTCondor

Local Campus Cluster accessible via
   Submit Machine *
   HTCondor via Glite

**Remote**

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

BOSCO based Glideins**
   SSH based submission of Glideins

PyGlidein
   ICE Cube Glidein service

OSG using glideinWMS

CREAMCE
   Uses CondorG

Globus GRAM
   Uses CondorG

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

Supported schedulers

PBS   SGE   SLURM   MOAB

https://pegasus.isi.edu
Pegasus est. 2001

Automate, recover, and debug scientific computations.

Get Started

Pegasus Website
http://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