Pegasus
Automate, recover, and debug scientific computations.

Mats Rynge
rynge@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
Taking a closer look into a workflow…

- **job**: Command-line programs
- **dependency**: Usually data dependencies
- **split**
- **merge**
- **pipeline**

**directed-acyclic graphs**

**DAG in XML**

**Pegasus**

http://pegasus.isi.edu
From the abstraction to execution!

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

http://pegasus.isi.edu
Optimizing storage usage...

cleanup job
Removes unused data

http://pegasus.isi.edu
Pegasus also provides tools to generate the abstract workflow

```python
dax = ADAG("test_dax")
firstJob = Job(name="first_job")
firstInputFile = File("input.txt")
firstOutputFile = File("tmp.txt")
firstJob.addArgument("input=input.txt", "output=tmp.txt")
firstJob.uses(firstInputFile, link=Link.INPUT)
firstJob.uses(firstOutputFile, link=Link.OUTPUT)
dax.addJob(firstJob)
for i in range(0, 5):
    simulJob = Job(id="%d" % (i+1), name="simul_job")
    simulInputFile = File("tmp.txt")
    simulOutputFile = File("output.%d.dat" % i)
    simulJob.addArgument("parameter=%d" % i, "input=tmp.txt", output=simulOutputFile.getName())
    simulJob.uses(simulInputFile, link=Link.INPUT)
    simulJob.uses(simulOutputFile, link=Link.OUTPUT)
dax.addJob(simulJob)
dax.depends(parent=firstJob, child=simulJob)
fp = open("test.dax", "w")
dax.writeXML(fp)
fp.close()
```

DAG in XML
While you wait...

...or the execution is finished.

Does everything executed successfully?

Statistics
Workflow execution and job performance metrics

Past executions?

Web-based interface
Real-time monitoring, graphs, provenance, etc.

How my workflow behaves?

Debug
Set of debugging tools to unveil issues

Command-line tools
Tools for monitor and debug workflows

RESTful API
Monitoring and reporting information on your own application interface

http://pegasus.isi.edu
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
But, if you prefer the 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

<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>10323</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>10323</td>
</tr>
<tr>
<td>Jobs</td>
<td>172</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>172</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 : 58 mins, 6 secs
Workflow cumulative job wall time : 145 hours, 38 mins
Cumulative job wall time as seen from submit side : 148 hours, 2 mins
Workflow cumulative job badput wall time :
Cumulative job badput wall time as seen from submit side :

...Pegasus provides a set of concise and powerful tools
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
Worried about data?
Let Pegasus manage it for you
How we handle it:

Pegasus

Input data site
Data staging site
Output data site

Submit host (e.g., user’s laptop)

Compute site A

Compute site B
However, there are several possible configurations for data sites...

submit host
(e.g., user’s laptop)

Typically most HPC sites
Pegasus also handles high-scalable object storages

Typical cloud computing deployment (Amazon S3, Google Storage)

submit host (e.g., user’s laptop)
Pegasus can also manage data over the submit host...
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
A few more features...
Performance, why not improve it?

- clustered job
  Groups small jobs together to improve performance

- task
  small granularity

https://pegasus.isi.edu

Workflow restructuring
Workflow reduction
Hierarchical workflows
Pegasus-mpi-cluster
What about **data reuse**?

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

 workflow restructuring
 workflow reduction
 hierarchical workflows

---

recursion ends when DAX with only compute jobs is encountered

http://pegasus.isi.edu
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
Pegasus’ flow at a glance

Data Reuse
- Replica Catalog

Task Clustering
- Transformation Catalog

Directory Creation and File Cleanup
- Site Catalog

Remote Workflow Engine
- Site Catalog
- Transformation Catalog

Site Selection
- Site Selector
- Site Catalog
- Transformation Catalog
- Replica Catalog

Transfer Refiner
- Replica Selector
- Replica Catalog

Code Generation
- executable workflow

abstract workflow

http://pegasus.isi.edu
Applications...
Multi-wavelength image atlas of the Galactic Plane, with coverage of 360° along the galactic plane and ±20° on either side

16 different wavelengths from 1 to 24 µm

Each output image is 5° by 5° in size, and have an overlap of 1° with neighboring tiles

Processed so that they appear to have been measured with a single instrument observing all 16 wavelengths - Cartesian projection

18 million input images (~2.5TB)

16 workflows, each of which contains 1,001 sub-workflows (hierarchical workflows)

10.5 million tasks
Amazon Web Services contributed the computations and storage

**hi1.4xlarge instance**
- Memory optimized, with 2 x SSD ephemeral drives
- 318,000 core hours
- Spot instance price: $5,950

Note: this is from 2013!
60,000 compute tasks
Input Data: 5000 files (10GB total)
Output Data: 60,000 files (60GB total)

Executed on LSC Data Grid, Open Science Grid and XSEDE

PyCBC Paper: An improved pipeline to search for gravitational waves from compact binary coalescence. Samantha Usman, Duncan Brown et al.

PyCBC Detection GW150914: First results from the search for binary black hole coalescence with Advanced LIGO. B. P. Abbott et al.
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)
http://soykb.org

XSEDE Allocation
PI: Dong Xu
Trupti Joshi, Saad Kahn, Yang Liu, Juexin Wang, Badu Valliyodan, Jiaojiao Wang

https://github.com/pegasus-isi/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 in the submit host - Workflow runs at a finer granularity

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

HipChat
Pegasus est. 2001
Automate, recover, and debug scientific computations.

Thank You
Questions?

Mats Rynge
rynge@isi.edu

Meet our team
Ewa Deelman
Karan Vahi
Mats Rynge
Rajiv Mayani
Rafael Ferreira da Silva

USC Viterbi
School of Engineering
Informatics & Computer Science
Extra...
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
... <!-- The local site contains information about the submit host -->
  <!-- The arch and os keywords are used to match binaries in the transformation catalog -->
  <site handle="local" arch="x86_64" os="LINUX">
      <!-- These are the paths on the submit host where Pegasus stores data -->
      <!-- Scratch is where temporary files go -->
      <directory type="shared-scratch" path="/home/tutorial/run">
          <file-server operation="all" url="file:///home/tutorial/run"/>
      </directory>
    <!-- Storage is where pegasus stores output files -->
    <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 for SCP transfers -->
    <profile namespace="env" key="SSH_PRIVATE_KEY">/home/tutorial/.ssh/id_rsa</profile>
  </site>
...
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.

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