Before we start

<table>
<thead>
<tr>
<th>Hands on Exercises Notes</th>
<th><a href="https://pegasus.isi.edu/tutorial/wrangler/">https://pegasus.isi.edu/tutorial/wrangler/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System</strong></td>
<td>wrangler.tacc.utexas.edu</td>
</tr>
<tr>
<td><strong>Training Accounts</strong></td>
<td>Pick up from the instructor</td>
</tr>
</tbody>
</table>
**Why are we here?**

**Science Automation Technologies at USC/Information Science Institute**
- Ewa Deelman
- Pegasus WMS is a cornerstone

**XSEDE**
- Mats is an ECSS consultant focusing on workflows

**Open Science Grid**
- XSEDE service provider, workloads against other XSEDE resources

**Model Integration through Knowledge-Rich Data and Process Composition (MINT)**
- A higher level graphical user interface for automatic workflow construction

**DesignSafe / SimCenter**
- Driver for the HTCondor install on Wrangler / AGAVE

**Cyverse / IPlant**
<table>
<thead>
<tr>
<th>Outline</th>
<th>Introduction</th>
<th>Scientific Workflows</th>
<th>Pegasus Overview</th>
<th>Successful Stories</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Pegasus Overview</strong></td>
<td>Basic Concepts</td>
<td>Features</td>
<td>System Architecture</td>
</tr>
<tr>
<td></td>
<td><strong>Hands-on Tutorial</strong></td>
<td>Submitting a Workflow</td>
<td>Monitoring and Debugging</td>
<td>Generating the Workflow</td>
</tr>
<tr>
<td></td>
<td><strong>Understanding Pegasus Features</strong></td>
<td>Information Catalogs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Hands-on Tutorial</strong></td>
<td>Information Catalogs</td>
<td>Configuring Pegasus</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>More Features</strong></td>
<td>Data Staging</td>
<td>Jupyter Notebooks</td>
<td>Metadata, Hierarchal Workflows, Data Reuse</td>
</tr>
<tr>
<td></td>
<td><strong>Hands-on Tutorial</strong></td>
<td>High Throughput Jobs as MPI Master/Worker Cluster Jobs</td>
<td>MPI Jobs</td>
<td></td>
</tr>
</tbody>
</table>

https://pegasus.isi.edu
Compute Pipelines

Building Blocks

Compute Pipelines
- 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

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 campus clusters have one

Debug and Monitor Computations
- Correlate data across lots of tasks / metadata / log files
- Need to know what host a job ran on, how it was invoked, and in what environment

Restructure Workflows for Improved Performance
- Short running tasks / Data placement and management / …

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**

NSF funded project since 2001, with close collaboration with HTCondor team
Some successful stories...
60,000 compute tasks
Input Data: 5,000 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
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
Soykb Workflow
TACC Wrangler as Execution Environment

HTCondor 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 great on Wrangler due to the flash filesystem, and the memory per core (48 cores, 128 GB RAM)
# 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>Monitoring and Debugging</td>
</tr>
<tr>
<td></td>
<td>Generating the Workflow</td>
</tr>
</tbody>
</table>

## Understanding Pegasus Features

- Information Catalogs

## More Features

- Data Staging
- Jupyter Notebooks
- Metadata, Hierarchal Workflows, Data Reuse

## Hands-on Tutorial

- High Throughput Jobs as MPI Master/Worker Cluster Jobs
- MPI Jobs
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
Portable Description

Users do not worry about low level execution details

DAG in XMI

logical filename (LFN)
platform independent (abstraction)

abstract workflow

transformation
platform independent

executeables (or programs)

DAG in XML

Portable Description

Users do not worry about low level execution details

DAG

directed-acyclic graphs

stage-in job
Transfers the workflow input data

stage-out job
Transfers the workflow output data

registration job
Registers the workflow output data

cleanup job
Removes unused data
Pegasus also provides tools to generate the abstract workflow.
#!/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"))
w.addArguments("-l",part)
w.setStdout(count)
w.uses(part, link=Link.INPUT)
w.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()
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.
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.
Provenance data can be summarized with `pegasus-statistics` or used for debugging with `pegasus-analyzer`.
### Introduction
- Scientific Workflows
- Pegasus Overview
- Successful Stories

### Pegasus Overview
- Basic Concepts
- Features
- System Architecture

### Hands-on Tutorial
- Submitting a Workflow
- Monitoring and Debugging
- Generating the Workflow

### Understanding Pegasus Features
- Information Catalogs

### More Features
- Data Staging
- Jupyter Notebooks
- Metadata, Hierarchal Workflows, Data Reuse

### Hands-on Tutorial
- High Throughput Jobs as MPI Master/Worker Cluster Jobs
- MPI Jobs
Using Shared FileSystem for Data Access

XSEDE Wrangler

Shared Disk such as Lustre Filesystem

LOGIN NODE

J
w

C1
Local Disk

Cn
Local Disk

LEGEND

Directory Setup Job
Data Stageout Job
Data Stagein Job
Directory Cleanup Job
Hands on

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

**System**  
wrangler.tacc.utexas.edu

**Training Accounts**  
*Pick up from the instructor*
<table>
<thead>
<tr>
<th>Outline</th>
</tr>
</thead>
</table>
| **Introduction** | Scientific Workflows  
Pegasus Overview  
Successful Stories |
| **Pegasus Overview** | Basic Concepts  
Features  
System Architecture |
| **Hands-on Tutorial** | Submitting a Workflow  
Monitoring and Debugging  
Generating the Workflow |
| **Understanding Pegasus Features** | Information Catalogs |
| **Hands-on Tutorial** | Information Catalogs  
Configuring Pegasus |
| **More Features** | Data Staging  
Jupyter Notebooks  
Metadata, Hierarchal Workflows, Data Reuse |
| **Hands-on Tutorial** | High Throughput Jobs as MPI Master/Worker Cluster Jobs  
MPI Jobs |
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 handle="local" arch="x86_64" os="LINUX">
  <!-- The local site contains information about the submit host -->
  <!-- The arch and os keywords are used to match binaries in the -->
  <!-- transformation catalog -->
  <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>

  <!-- 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

```bash
# 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
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
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"
```
• Support for
  • Docker
  • Singularity
  • Shifter (coming soon)

• 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
  • Images exported as a TAR file and available on a filesystem/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.
<table>
<thead>
<tr>
<th>Outline</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>Scientific Workflows</td>
</tr>
<tr>
<td></td>
<td>Pegasus Overview</td>
</tr>
<tr>
<td></td>
<td>Successful Stories</td>
</tr>
<tr>
<td>Pegasus Overview</td>
<td>Basic Concepts</td>
</tr>
<tr>
<td></td>
<td>Features</td>
</tr>
<tr>
<td></td>
<td>System Architecture</td>
</tr>
<tr>
<td>Hands-on Tutorial</td>
<td>Submitting a Workflow</td>
</tr>
<tr>
<td></td>
<td>Monitoring and Debugging</td>
</tr>
<tr>
<td></td>
<td>Generating the Workflow</td>
</tr>
<tr>
<td>Understanding Pegasus Features</td>
<td>Information Catalogs</td>
</tr>
<tr>
<td>Hands-on Tutorial</td>
<td>Information Catalogs</td>
</tr>
<tr>
<td></td>
<td>Configuring Pegasus</td>
</tr>
<tr>
<td>More Features</td>
<td>Data Staging</td>
</tr>
<tr>
<td></td>
<td>Jupyter Notebooks</td>
</tr>
<tr>
<td></td>
<td>Metadata, Hierarchal Workflows, Data Reuse</td>
</tr>
<tr>
<td>Hands-on Tutorial</td>
<td>High Throughput Jobs as MPI Master/Worker Cluster Jobs</td>
</tr>
<tr>
<td></td>
<td>MPI Jobs</td>
</tr>
</tbody>
</table>

https://pegasus.isi.edu
<table>
<thead>
<tr>
<th>Hands on Exercises Notes</th>
<th><a href="https://pegasus.isi.edu/tutorial/wrangler/">https://pegasus.isi.edu/tutorial/wrangler/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System</strong></td>
<td>wrangler.tacc.utexas.edu</td>
</tr>
<tr>
<td><strong>Training Accounts</strong></td>
<td>Pick up from the instructor</td>
</tr>
<tr>
<td>Outline</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
| **Introduction** | Scientific Workflows  
Pegasus Overview  
Successful Stories |
| **Pegasus Overview** | Basic Concepts  
Features  
System Architecture |
| **Hands-on Tutorial** | Submitting a Workflow  
Monitoring and Debugging  
Generating the Workflow |
| **Understanding Pegasus Features** | Information Catalogs |
| **Hands-on Tutorial** | Information Catalogs  
Configuring Pegasus |
| **More Features** | Data Staging  
Jupyter Notebooks  
Metadata, Hierarchal Workflows, Data Reuse |
| **Hands-on Tutorial** | High Throughput Jobs as MPI Master/Worker Cluster Jobs  
MPI Jobs |

https://pegasus.isi.edu
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
High Performance Computing

There are several possible configurations...

submit host
(e.g., user’s laptop)

shared filesystem

Input data site
Data staging site
Output data site

typically most HPC sites
Cloud Computing

Typical cloud computing deployment (Amazon S3, Google Storage)

submit host
(e.g., user’s laptop)
Grid Computing

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

local data management

Typical OSG sites
Open Science Grid
And yes... you can mix everything!

submit host
(e.g., user’s laptop)

Compute site A

Compute site B

Output data site

Input data site
(Data staging site)

Input data site
(Data staging site)

Input data site
(Output data site)

shared filesystem

object storage
 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)
Performance, why not improve it?

*clustered job*
Groups small jobs together to improve performance

*task*
small granularity

https://pegasus.isi.edu
Running **fine-grained** workflows on HPC systems...

submit host (e.g., user’s laptop)

workflow wrapped as an MPI job

Allowing sub-graphs of a Pegasus workflow to be submitted as monolithic jobs to remote resources
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

WAN LAN

Campus Cluster  HPC/HTC  Clouds

https://pegasus.isi.edu
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)

instance = Instance(dax)

running a workflow

instance.run(site='condorpool')

monitoring a workflow execution

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

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

![Diagram showing metadata workflow]

```
<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="Q42.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>
```

Pegasus
https://pegasus.isi.edu
What about **data reuse**?

Jobs which output data is already available are pruned from the DAG.

*data already available*

*data also available*

*workflow reduction*

*data reuse*

*workflow restructuring*

*hierarchical workflows*

*pegasus-mpi-cluster*
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.
Job Submissions

**Local**
Submit Machine
Personal HTCondor

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

**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

** Both Glite and BOSCO build on HTCondor BLAHP

Currently supported schedulers:

SLURM  SGE  PBS  MOAB

https://pegasus.isi.edu
Using Shared Filesystem for Data Access

XSEDE Wrangler

LOGIN NODE

Abstract Workflow

Workflow Setup Job

Workflow Stagein Job

Executable Workflow

Workflow Stageout Job

Data Cleanup Job

BLAHPS

CondoR DAGMan

CondoR Schedd

Shared Disk such as Lustre Filesystem

C1

Local Disk

Cn

Local Disk

LEGEND

Orange: Directory Setup Job

Green: Data Stagein Job

Blue: Data Stageout Job

Red: Directory Cleanup Job
<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*  
*Monitoring and Debugging*  
*Generating the Workflow* |
| **Understanding Pegasus Features** | *Information Catalogs* |
| **Hands-on Tutorial** | *Information Catalogs*  
*Configuring Pegasus* |
| **More Features** | *Data Staging*  
*Jupyter Notebooks*  
*Metadata, Hierarchal Workflows, Data Reuse* |
| **Hands-on Tutorial** | *High Throughput Jobs as MPI Master/Worker Cluster Jobs*  
*MPI Jobs* |

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

HipChat