Managing Workflows Within HUBzero: How to Use Pegasus to Execute Computational Pipelines

Ewa Deelman

USC Information Sciences Institute

Acknowledgement:

Steven Clark, Derrick Kearney, Michael McLennan (HUBzero)

Frank McKenna (OpenSees)

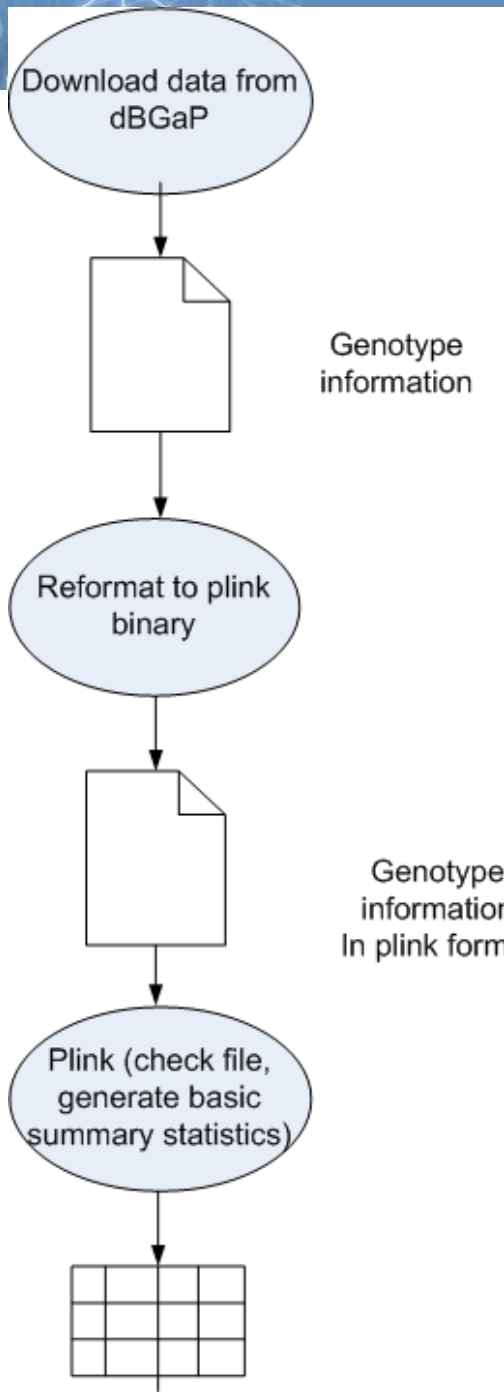
Gideon Juve, Gaurang Mehta, Mats Rynge, Karan Vahi (Pegasus)

Outline

- Introduction to Pegasus and workflows
- HUB Integration
  - Rappture and Pegasus
  - Submit command and Pegasus
- Example: OpenSEES / NEESHub
- Future directions

Computational workflows

- Help express multi-step computations in a declarative way
- Can support automation, minimize human involvement
  - Makes analyses easier to run
- Can be high-level and portable across execution platforms
- Keep track of provenance to support reproducibility
- Foster collaboration—code and data sharing



Workflow Management

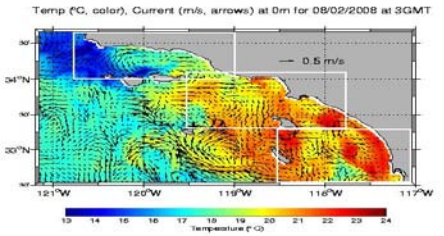
- You may want to use different resources within a workflow or over time
  - Need a high-level workflow specification
  - Need a planning capability to map from high-level to executable workflow
  - Need to manage the task dependencies
  - Need to manage the execution of tasks on the remote resources
- Need to provide scalability, performance, reliability

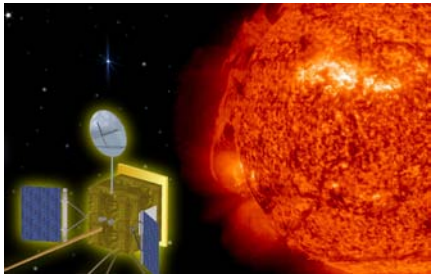
Our Approach

- Analysis Representation
  - Support a declarative representation for the workflow (dataflow)
  - Represent the workflow structure as a Directed Acyclic Graph (DAG) in a resource-independent way
  - Use recursion to achieve scalability
- System (Plan for the resources, Execute the Plan, Manage tasks)
  - Layered architecture, each layer is responsible for a particular function (Pegasus Planner, DAGMan, Condor schedd)
  - Mask errors at different levels of the system
  - Modular, composed of well-defined components, where different components can be swapped in
  - Use and adapt existing graph and other relevant algorithms
  - Can be embedded into

Pegasus Workflow Management System (est. 2001)

- A collaboration with University of Wisconsin Madison
- Used by a number of applications in a variety of domains
- Provides reliability—can retry computations from the point of failure
- Provides scalability—can handle large data and many computations (kbytes-TB of data, 1-10^6 tasks)
- Optimizes workflows for performance
- Automatically captures provenance information
- Runs workflows on distributed resources: laptop, campus cluster, Grids (DiaGrid, OSG, XSEDE), Clouds (FutureGrid, EC2, etc..)





http://pegasus.isi.edu

Planning Process

- Assume data may be distributed in the Environment
- Assume you may want to use local and/or remote resources

- Pegasus needs information about the environment
  - data, executables, execution and data storage sites
- Pegasus generates an executable workflow

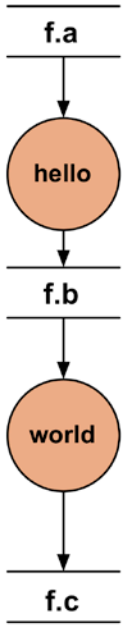
- Data transfer protocols
  - Gridftp, Condor I/O, HTTP, scp, S3, iRods, SRM, FDT (partial)
- Scheduling to interfaces
  - Local, Gram, Condor, Condor-C (for remote Condor pools), via Condor Glideins – PBS, LSF, SGE

Generating executable workflows

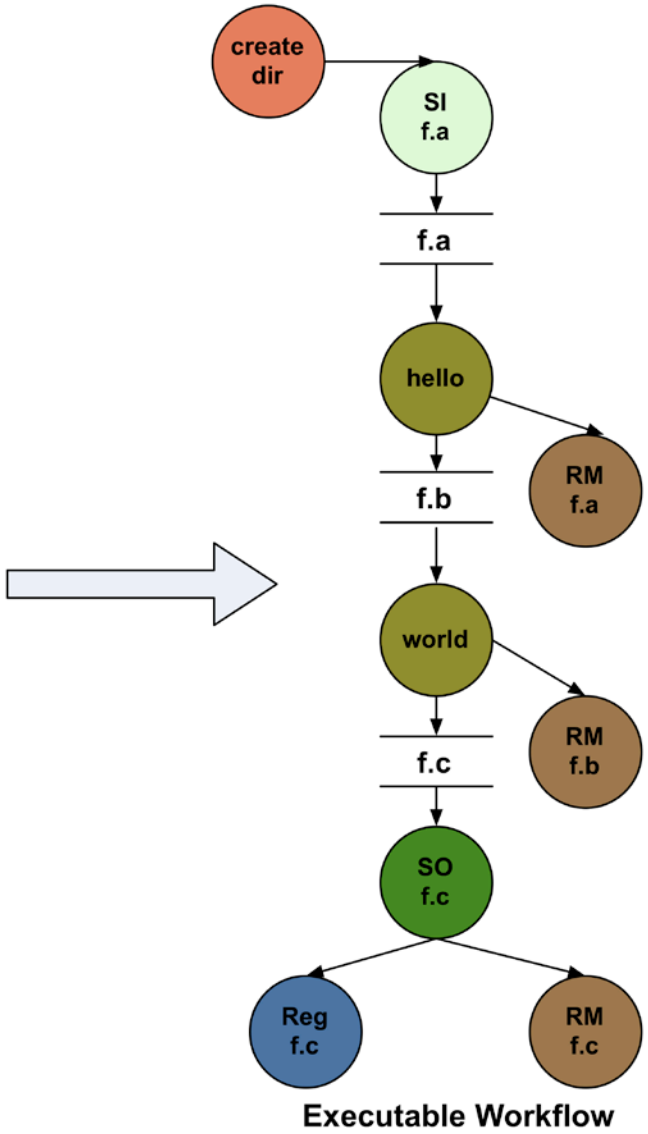
APIs for workflow specification (DAX--- DAG in XML)

Java, Perl, Python

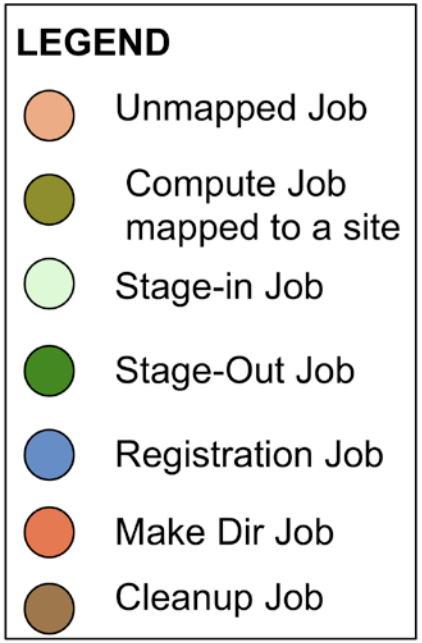
(DAX)



Abstract Workflow

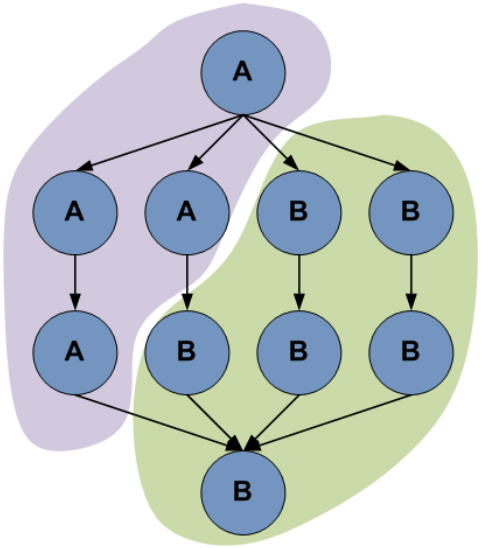


Executable Workflow



Advanced features

- Performs data reuse
- Registers data in data catalogs
- Manages storage—deletes data no longer needed
- Can cluster tasks together for performance
- Can manage complex data architectures (shared and non-shared filesystem, distributed data sources)
- Different execution modes which leverage different computing architectures (Condor pools, HPC resources, etc..)



HUBzero Integration

Pegasus with

https://hubzero/resources/pegtut

hubzero PLATFORM FOR SCIENTIFIC COLLABORATION

search

MATS RYNGE (RYNGE) | MY ACCOUNT | LOGOUT | HELP

get started | download | documentation | support | community | hosting | about

HOME › TOOLS › PEGASUS TUTORIAL › ABOUT

Pegasus Tutorial

Launch Tool

Version **1.0** - published on 22 Sep 2012

Open source: license | download

View All Supporting Documents

By Steven Clark[1], Derrick Kearney, Mats Rynge[2], Karan Vahi

1. Purdue University 2. USC Information Sciences Institute

Demonstration of simple Pegasus examples combined with Rappture interface.

0 Citation(s)
0 review(s) (Review this)
0 questions (Ask a question)
user(s), detailed usage
0 wish(es) (Add a new wish)
Add to your favorites!
Share:

SEE ALSO

No results found.

About | Citations | Reviews | Questions | Usage | Wish List | Supporting Docs

Category Tools

Abstract **Goal:** The goal of this tutorial is to demonstrate how to use the Pegasus Workflow Management System together with a Rappture interface to present the users an easy way to run complex computations in the Hub.

The workflow in this tutorial is a simple Hello World example. Attached to this tool is a PDF which shows how the tool was created.

To see the code for the Hub integration, please use a Workspace and check out a copy:

Benefits of Pegasus for HUB Users

- **Provides Support for Complex Computations**
  - Can connect the existing HUB models into larger computations
- **Portability / Reuse**
  - User created workflows can easily be run in different environments without alteration (today DiaGrid, OSG)
- **Performance**
  - The Pegasus mapper can reorder, group, and prioritize tasks in order to increase the overall workflow performance.
- **Scalability**
  - Pegasus can easily scale both the size of the workflow, and the resources that the workflow is distributed over.

Benefits of Pegasus for HUB Users

- **Provenance**
  - Performance and provenance data is collected in a database, and the data can be summaries with tools such as **pegasus-statistics**, **pegasus-plots**, or directly with SQL queries.
- **Reliability**
  - Jobs and data transfers are automatically retried in case of failures. Debugging tools such as **pegasus-analyzer** helps the user to debug the workflow in case of non-recoverable failures.

Pegasus in HUBzero

- Pegasus as a backend to the *submit* command
- Pegasus workflows composed in *Rappture*
  - *Build workflow within Rappture*
  - *Have Rappture collect inputs, call a workflow generator, and collect outputs*
- Pegasus Tutorial tool now available in HUBzero

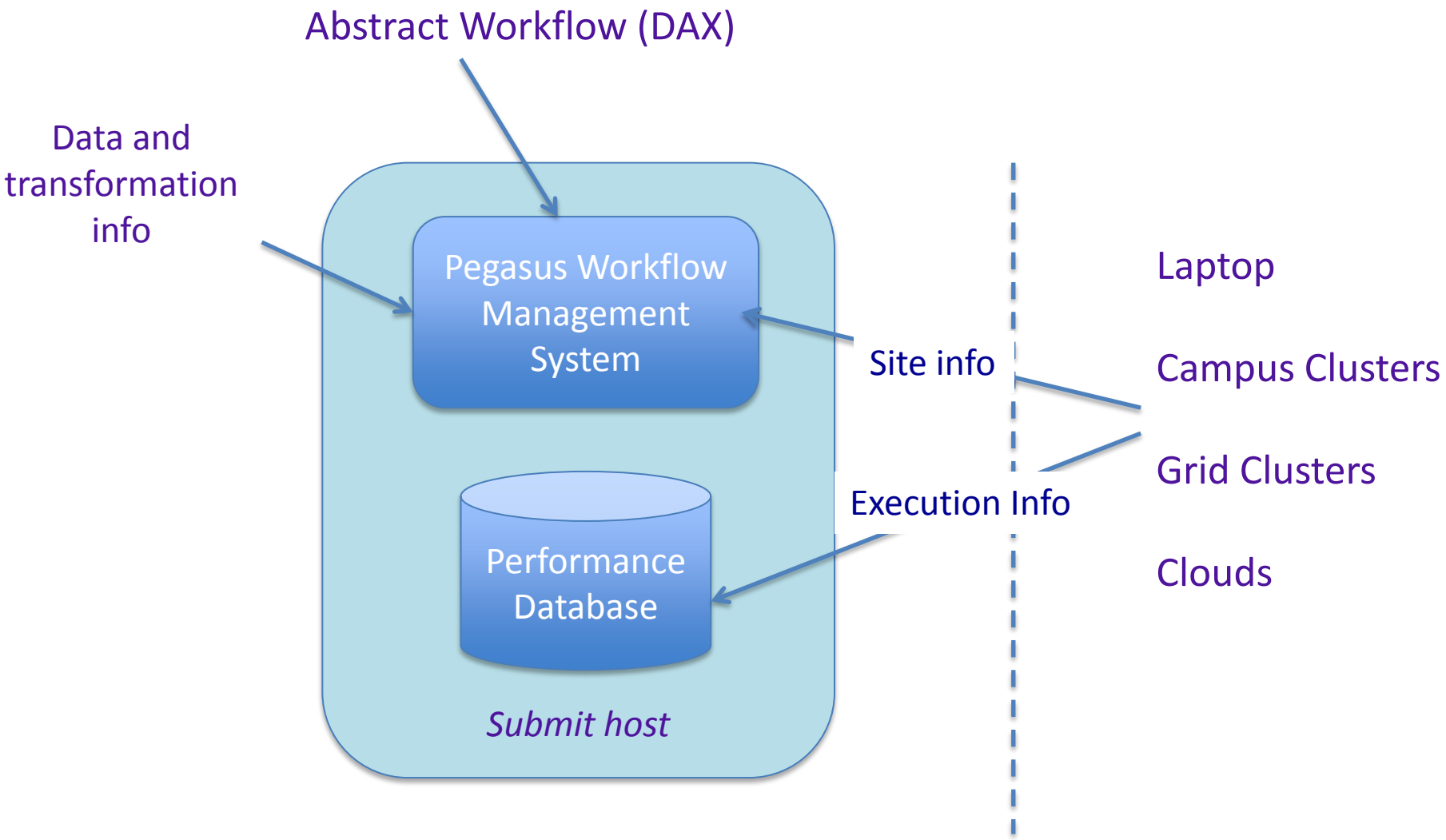
http://hubzero.org/tools/pegtut

- Session that includes Pegasus on Tuesday 1:30 – 5:30

  Room 206 **#2 Creating and Deploying Scientific Tools (part 2)**

  "… Scientific Workflows with Pegasus" by George Howlett & Derrick Kearney, Purdue University

Acknowledgements: Steven Clark and Derrick Kearney, Purdue University

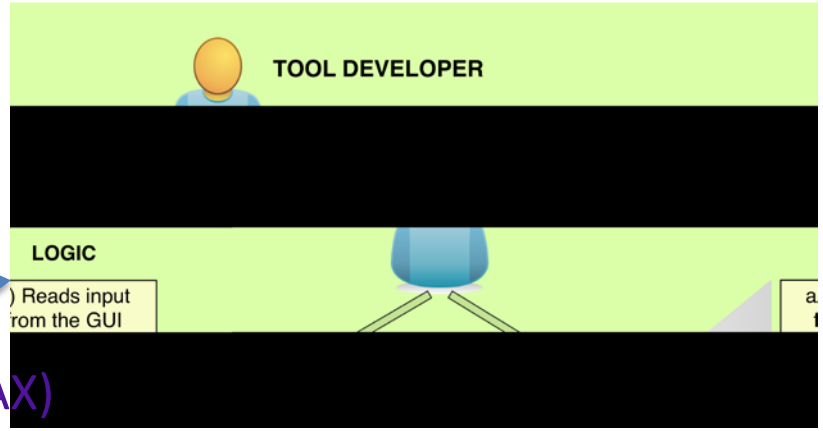

Abstract Workflow (DAX)
Data and transformation info
Pegasus Workflow Management System
Performance Database
Submit host
Site info
Execution Info
Laptop
Campus Clusters
Grid Clusters
Clouds

Use of Pegasus with Submit Command

- Used by Rappture interface to submit the workflow
- Submits the workflow through Pegasus to
  - OSG
  - DIAGRID
- Prepares the site catalog and other configuration files for Pegasus
- Uses pegasus-status to track the workflow
- Generates statistics and report about job failures using

Acknowledgements: Steven Clark and Derrick Kearney, Purdue University

Data and transformation info



Abstract Workflow (DAX)

Hub

Site info

Pegasus Workflow Management System

Performance Database

Submit host

Execution Info

Laptop

Campus Clusters

Grid Clusters

Clouds

Pegasus Workflows in the HUB

Rappture (data definitions)

Tool description

Inputs

Outputs

```
<?xml version="1.0"?>
<run>
    <tool>
        <title>Hello World Workflow</title>
        <about>Simple interface for Pegasus Hello World example</about>
        <command>python @tool/wrapper.py @driver</command>
    </tool>
    <input>
        <note>
            <contents>file://workflow.html</contents>
        </note>
        <string id="textinput">
            <about>
                <label>Your Name</label>
                <description>Enter your name</description>
            </about>
            <default>Pete</default>
        </string>
    </input>
    <output>
        <string id="greeting">
            <about>
                <label>Greeting</label>
                <description>Greeting generated by the two jobs submitted to the grid through
            </about>
        </string>
        <string id="fa">
            <about>
                <label>f.a</label>
                <description>Input to Hello job</description>
            </about>
        </string>
        <string id="fb">
            <about>
                <label>f.b</label>
                <description>Output from sayhi job</description>
            </about>
```

Calls an external DAX generator

Acknowledgements: Steven Clark and Derrick Kearney, Purdue University

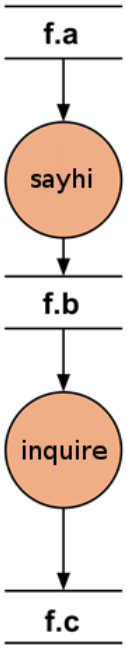
wrapper.py

- Python script
- Collects the data from the Rappture interface
- Generates the DAX
- Runs the workflow
- Presents the outputs to Rappture

```
# ------------------------------------------------------------
#  MAIN PROGRAM - generated by the Rappture Builder
# ------------------------------------------------------------
import sys
import os
from math import *

import Rappture
from Rappture.tools import getCommandOutput as RapptureExec
from Pegasus.DAX3 import *

# open the XML file containing the run parameters
io = Rappture.library(sys.argv[1])

# setup paths to our executables
scriptpath = os.path.realpath(__file__)
scriptdir = os.path.dirname(scriptpath)
tooldir = os.path.dirname(scriptdir)
sayhipath = os.path.join(tooldir,'bin','sayhi.sh')
inquirepath = os.path.join(tooldir,'bin','inquire.sh')

################################################################
# Get input values from Rappture
################################################################

# get input value for input.string(textinput)
textinput = io.get('input.string(textinput).current')
if not textinput:
   sys.stderr.write("Input data is missing\n")
   sys.exit(1)

################################################################
#  Add your code here for the main body of your program
################################################################

fp = open('f.a','w')
if fp:
   fp.write(textinput + '\n')
   fp.close()
else:
   sys.stderr.write("Could not create datafile\n")
```

Acknowledgements: Steven Clark and Derrick Kearney, Purdue University

Workflow generation



Abstract Workflow

```
# Create a abstract dag
dax = ADAG("sayhi_inquire")

# Add input file to the DAX-level replica catalog
a = File("f.a")
a.addPFN(PFN("file://" + os.getcwd() + "/f.a", "local"))
dax.addFile(a)

# Add executables to the DAX-level replica catalog
e_sayhi = Executable(namespace="sayhi_inquire", name="sayhi", version="1.0", \
                     os="linux", arch="x86_64", installed=False)
e_sayhi.addPFN(PFN("file://" + sayhipath, "condorpool"))
dax.addExecutable(e_sayhi)

e_inquire = Executable(namespace="sayhi_inquire", name="inquire", version="1.0", \
                     os="linux", arch="x86_64", installed=False)
e_inquire.addPFN(PFN("file://" + inquirepath, "condorpool"))
dax.addExecutable(e_inquire)

# Add the sayhi job
sayhi = Job(namespace="sayhi_inquire", name="sayhi", version="1.0")
sayhi.addArguments('f.a')
b = File("f.b")
sayhi.uses(a, link=Link.INPUT)
sayhi.uses(b, link=Link.OUTPUT)
dax.addJob(sayhi)

# Add the inquire job (depends on the sayhi job)
inquire = Job(namespace="sayhi_inquire", name="inquire", version="1.0")
inquire.addArguments('f.b')
c = File("f.c")
inquire.uses(b, link=Link.INPUT)
inquire.uses(c, link=Link.OUTPUT)
```

Acknowledgements: Steven Clark and Derrick Kearney, Purdue University

User provides inputs to the workflow and clicks the “Submit” button

Hello World with Pegasus

With Pegasus, you can hook together multiple executables and pass data between them. In this example, two executables are used to build a greeting.

To start, enter your name below. After pressing the Simulate button, the information is saved in the file named f.a, and passed to the **sayhi** program. **sayhi** prints a specialized hello message for you. The greeting is saved in the file named f.b and sent to the **inquiry** program. **inquiry** finishes the greeting by asking how you are doing.

This job is executed on the grid using Pegasus and Submit!

Learn about software that powers this tool:

- Pegasus Workflow Management System
- Rappture Toolkit
- Submit

Give it a try:

Your Name: Pete

Simulate new input parameters

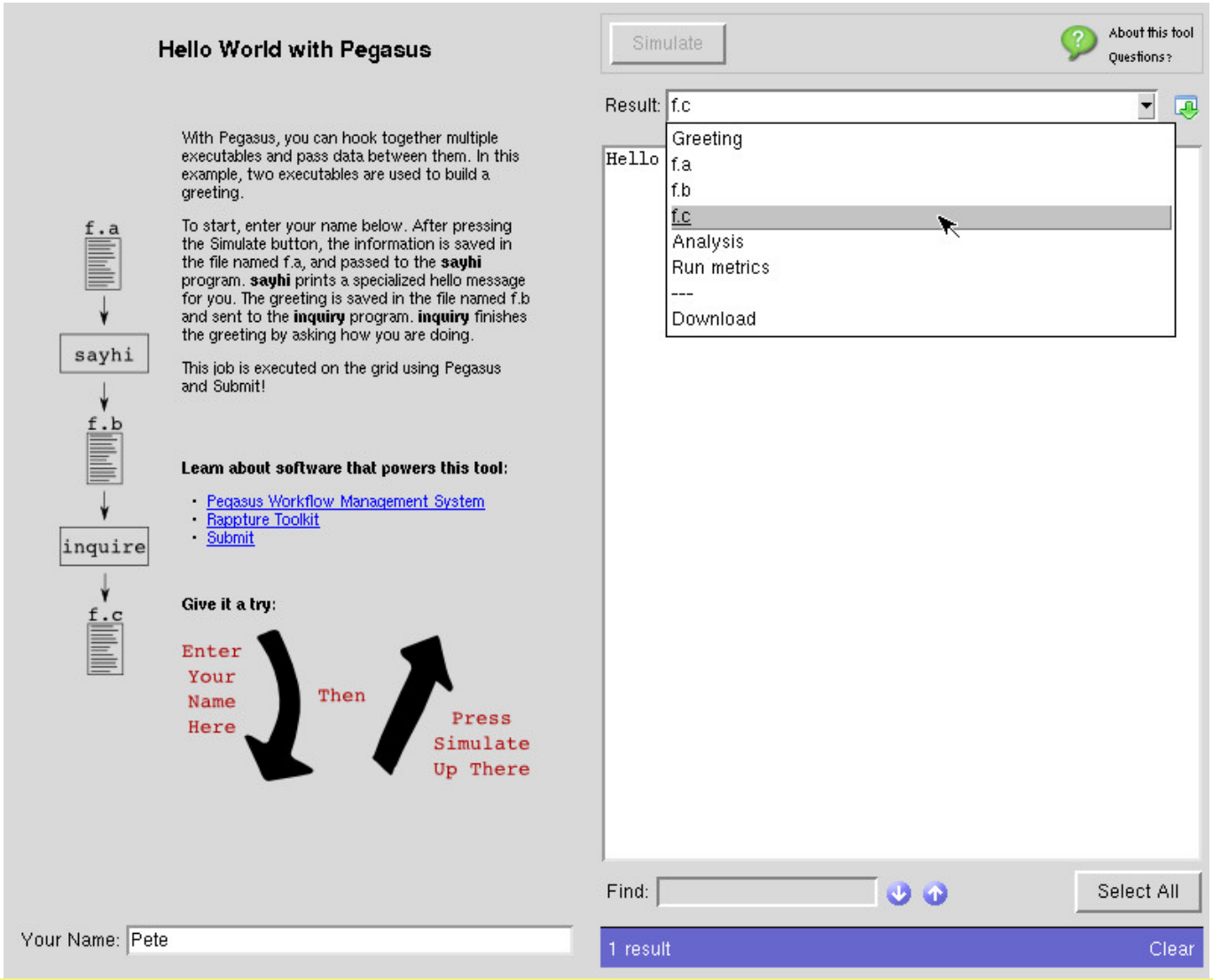
About this tool
Questions?

Hello World Workflow

Simple interface for Pegasus Hello World example

Acknowledgements: Steven Clark and Derrick Kearney, Purdue University

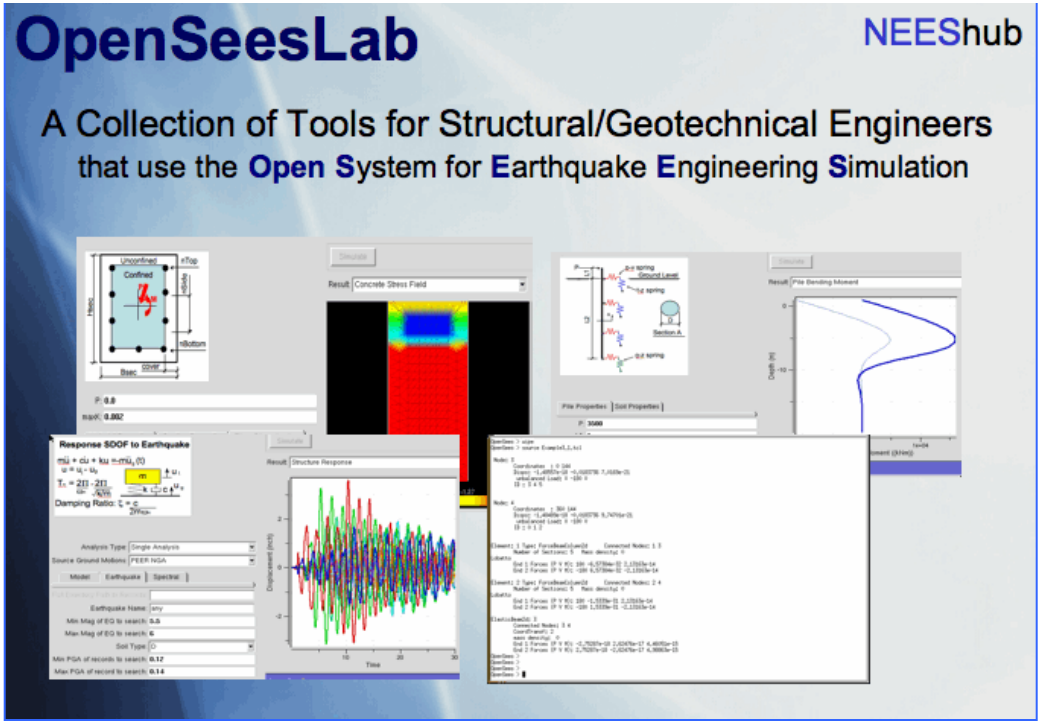
Workflow has completed. Outputs are available for browsing/downloading



Acknowledgements: Steven Clark and Derrick Kearney, Purdue University

OpenSEES / NEEShub

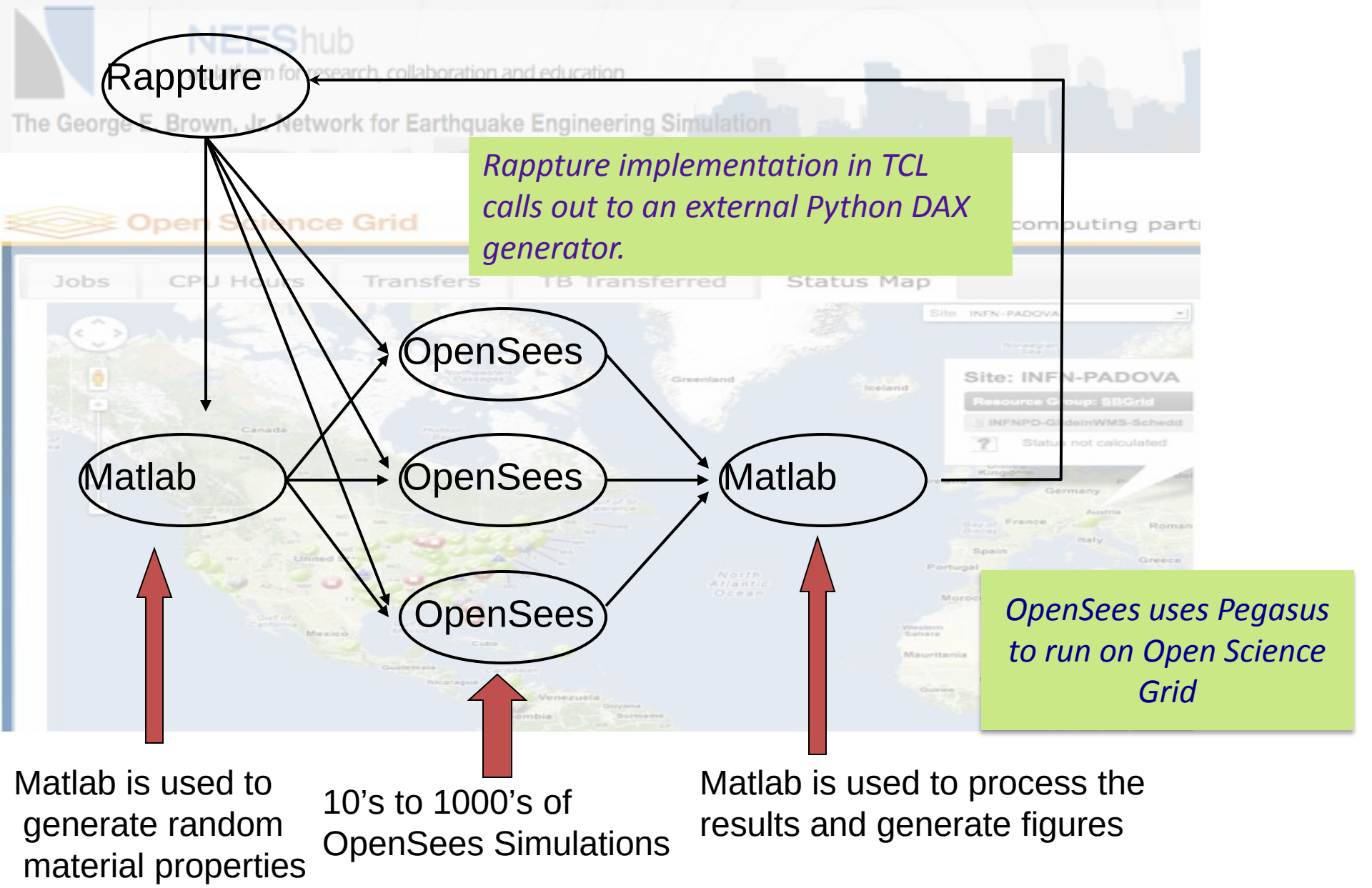
The OpenSeesLab tool: http://nees.org/resources/tools/openseeslab



Is a suite of Simulation Tools powered by OpenSees for:

1. Submitting OpenSees scripts to NEEShub resources
2. Educating students and practicing engineers

Acknowledgements: Frank McKenna from UC Berkeley

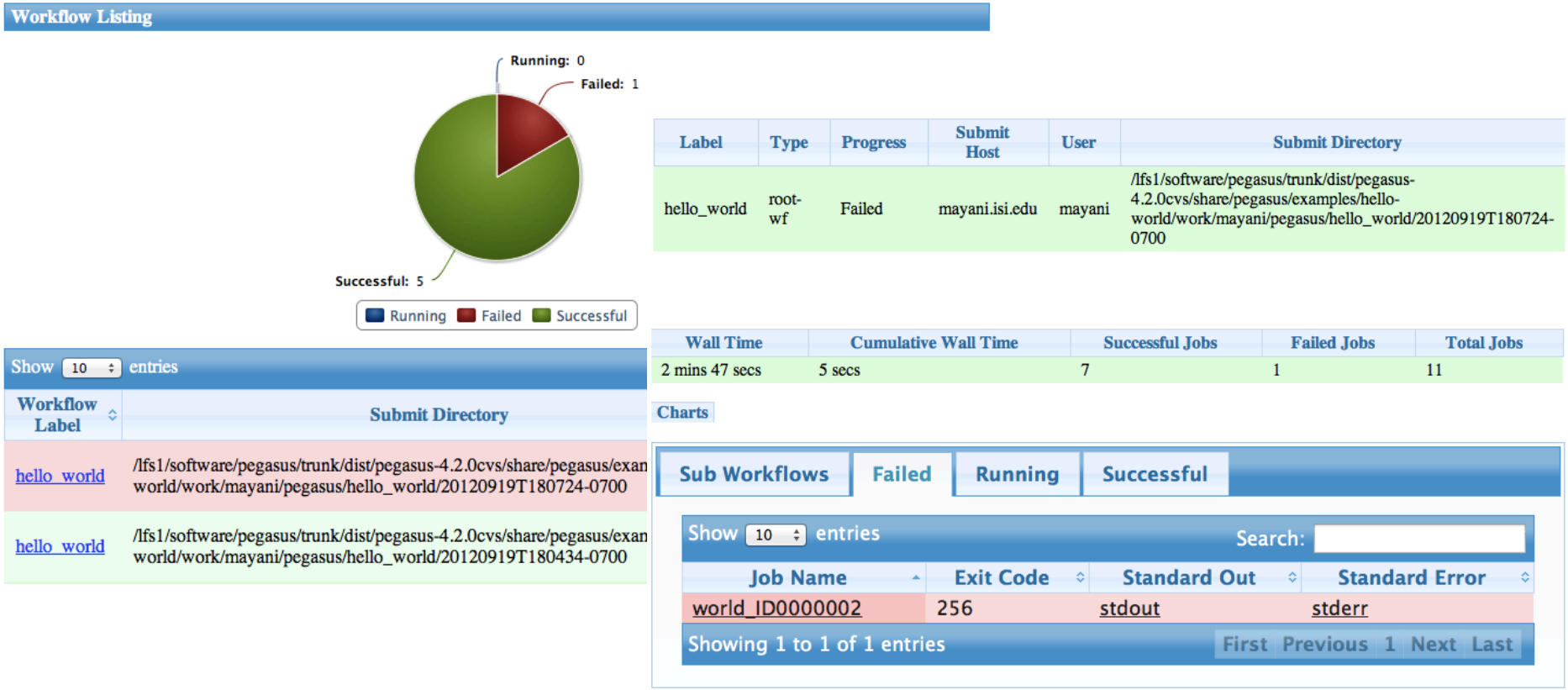


Pegasus is Responsible for moving the data from the NEEShub to the OSG, orchestrating the workflow and returning the results to NEEShub.

Acknowledgements: Frank McKenna from UC Berkeley

Future Directions

- Submit to manage parameter sweep computations (now only on HUBzer0)
- Web-based monitoring



Benefits of workflows in the HUB

- Support for complex applications/ builds on existing domain tools
- Clean separations for users/developers/operator
  - User: Nice high level interface via Rappture
  - Tool developer: Only has to build/provide a description of the workflow (DAX)
  - Hub operator: Ties the Hub to an existing distributed computing infrastructure (DiaGrid, OSG, …)
- The Hub and Pegasus handle low level details
  - Job scheduling to various execution environments
  - Data staging in a distributed environment
  - Job retries
  - Workflow analysis
  - Support for large workflows

Benefits of the HUB to Pegasus

- Provides a nice, easy to use interface to Pegasus workflows
- Broadens the user base
- Improves the software based on user's feedback
- Drives innovation—new deployment scenarios, use cases

- I look forward to a continued collaboration

Further Information

- Session that includes Pegasus on Tuesday 1:30 – 5:30
  - Room 206 **#2 Creating and Deploying Scientific Tools (part 2)**
  - "… Scientific Workflows with Pegasus" by George Howlett & Derrick Kearney, Purdue University
- Pegasus Tutorial on the HUB
  - https://hubzero.org/tools/pegtut
- General Pegasus Information http://pegasus.isi.edu
- Pegasus in a VM—allows you to develop DAXes
  - http://pegasus.isi.edu/downloads
  - **We are happy to help!**
- Support mailing lists pegasus-support@isi.edu pegasus-users@isi.edu,, pegasus-announce@isi.edu
- Contact me deelman@isi.edu

Big Thank You to the HUBzero and OpenSees teams!