Pegasus is funded by the National Science Foundation under grant #1664162
Pegasus Ensemble Manager

Ryan Tanaka
Programmer Analyst

February 23, 2021
Workflow Ensembles

- workflows for large computational problems are often composed of several inter-related workflows grouped into ensembles.
- ensemble workflows typically have a similar structure, but they differ in:
  - input data
  - number of tasks
  - individual task sizes
Workflow Ensembles: SCEC & CyberShake

- application developed by the Southern California Earthquake Center to generate seismic hazard maps
- 2013 study, Cybershake ran an **ensemble of 2,288 workflows** to generate hazard maps for over 286 sites (https://strike.scec.org/scecpedia/CyberShake_Study_13.4)

CyberShake [1], for example, uses ensembles to generate seismic hazard maps. Each workflow in a CyberShake ensemble generates a hazard curve for a particular geographic location, and several hazard curves are combined to create a hazard map. In a 2013 study, CyberShake was used to generate a set of hazard maps over 286 sites that required an ensemble of 2288 workflows [2].
Workflow Ensembles: Galactic Plane

- developed by NASA/IPAC Infrared Science Archive
- ensemble consisting of 17 workflows, each of which contains 900 sub-workflows
- input data set roughly 18 million images
- uses Montage engine to generate a multiwavelength image atlas of the galactic plane

source: http://montage.ipac.caltech.edu/gallery.html
Outline

- Workflow Ensembles
- **Pegasus Ensemble Manager**
  - Using the Ensemble Manager
  - Coming Soon: REST API
- Summary
The Pegasus Ensemble Manager

- service for managing collections of workflows (ensembles)
- eliminates the need to keep track of multiple submit directories
- provides **ensemble level configuration**
  - max_planning
  - max_running
  - workflow priority
- provides **monitoring and debugging** capabilities
  - pegasus-status
  - pegasus-analyzer
The Pegasus Ensemble Manager cont.

**Pegasus Ensemble Manager**

Ensemble 1
- max_running = 2
- Priority: 100

Ensemble 2
- max_running = 1
- Priority: 100
- Priority: 90

Ensemble management

Individual workflow planning, submission, & monitoring

**Workflow execution**

**Pegasus WMS**
The Pegasus Ensemble Manager cont.

Ensemble 1
max_running = 2
Priority: 100

Ensemble 2
max_running = 1
Priority: 100
Priority: 90

Individual workflow planning, submission, & monitoring

Workflow execution
The Pegasus Ensemble Manager cont.

Ensemble 1
max_running = 2
Priority: 100

Ensemble 2
max_running = 1
Priority: 100
Priority: 90

Pegasus WMS
Priority: 100

Pegasus Ensemble Manager

Priority: 100
Priority: 100

ensemble management
dividual workflow planning, submission, & monitoring
workflow execution
Outline

- Workflow Ensembles
- Pegasus Ensemble Manager
- Using the Ensemble Manager
- Coming Soon: REST API
- Summary
Using Ensemble Manager

1. start the ensemble manager on the submit node
   ○ pegasus-em server
Using Ensemble Manager cont.

1. Start the ensemble manager on the submit node
   - pegasus-em server
2. Create the ensemble
   - pegasus-em create myruns
Using Ensemble Manager cont.

1. start the ensemble manager on the submit node
   - `pegasus-em server`
2. create the ensemble
   - `pegasus-em create myruns`
3. configure ensemble (optional)
   - `pegasus-em config myruns --max_planning=2 --max_running=2`
Using Ensemble Manager cont.

1. start the ensemble manager on the submit node
   - pegasus-em server
2. create the ensemble
   - pegasus-em create myruns
3. configure ensemble (optional)
   - pegasus-em config myruns --max_planning=2 --max_running=2
4. add workflows to ensemble
   - pegasus-em submit myruns.wf1 /wf1.py
   - pegasus-em submit myruns.wf2 /wf2.py
5. configure priority (optional)
   - pegasus-em priority myruns.wf1 -p 10
Using Ensemble Manager cont.

1. start the ensemble manager on the submit node
   - `pegasus-em server`
2. create an ensemble
   - `pegasus-em create myruns`
3. configure ensemble (optional)
   - `pegasus-em config myruns --max_planning=2 --max_running=2`
4. add workflows to ensemble
   - `pegasus-em submit myruns.wf1 /wf1.py`
   - `pegasus-em submit myruns.wf2 /wf2.py`
5. configure priority (optional)
   - `pegasus-em priority myruns.wf1 -p 10`
6. # do not set `submit=True`
7. `wf.plan()`
Using Ensemble Manager cont.

6. check the status of submitted workflows
   ○ `pegasus-em workflows myruns`

7. debug a failed workflow
   ○ `pegasus-em analyze myruns.wf1`
Using the Ensemble Manager: Triggers

- **cron based workflow trigger**
  - add the same workflow to the ensemble manager at a specified time interval
  - example: `pegasus-em cron-trigger myruns mytrigger 1h /home/ryan/workflow.py -t 1d`

- **file pattern + cron based workflow trigger**
  - at the given time interval, the specified workflow will be added to the ensemble manager such that all new input files matching the given file pattern(s) will be passed as input to the workflow script using the --inputs flag
  - example: `pegasus-em file-pattern-trigger myruns 10s_txt 10s /home/ryan/workflow.py /home/ryan/input/*.txt --timeout 40s`
Outline

- Workflow Ensembles
- Pegasus Ensemble Manager
- Using the Ensemble Manager
- Coming Soon: REST API
- Summary
Coming Soon: REST API

- exposing the ensemble manager as a REST endpoint
- provide **CRUD operations on Ensembles, Workflows, and Triggers**
- provide **Python and Java client code**
- improve support for integrating ensemble manager into larger systems
Outline

- Workflow Ensembles
- Pegasus Ensemble Manager
- Using the Ensemble Manager
- Coming Soon: REST API
- Summary
Pegasus Ensemble Manager: Recap

- **service that resides on the submit node** along with Pegasus and HTCondor
- useful for **managing groups of workflows** (ensembles)
- provides **monitoring and debugging capabilities** through pegasus-status and pegasus-analyzer
- provides **workflow triggering capabilities**
- **REST API** being developed for upcoming release
Thank You!