Pegasus Workflow Management System

The Pegasus project encompasses a set of technologies that help workflow-based applications execute in a number of different environments including desktops, campus clusters, grids, and clouds. Pegasus bridges the scientific domain and the execution environment by automatically mapping high-level workflow descriptions onto distributed resources. It automatically locates the necessary input data and computational resources necessary for workflow execution. Pegasus enables scientists to construct workflows in abstract terms without worrying about the details of the underlying execution environment or the particulars of the low-level specifications required by the middleware. Some of the advantages of using Pegasus includes:

- **Portability / Reuse** - User created workflows can easily be run in different environments without alteration. Pegasus currently runs workflows on top of Condor, Grid infrastructures such as Open Science Grid and TeraGrid, Amazon EC2, Nimbus, and many campus clusters. The same workflow can run on a single system or across a heterogeneous set of resources.

- **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. Pegasus runs workflows ranging from just a few computational tasks up to 1 million. The number of resources involved in executing a workflow can scale as needed without any impediments to performance.

- **Provenance** - By default, all jobs in Pegasus are launched via the kickstart process that captures runtime provenance of the job and helps in debugging. The 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.

- **Data Management** - Pegasus handles replica selection, data transfers and output registrations in data catalogs. These tasks are added to a workflow as auxiliary jobs by the Pegasus planner.

- **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.

- **Error Recovery** - When errors occur, Pegasus tries to recover when possible by retrying tasks, by retrying the entire workflow, by providing workflow-level checkpointing, by re-mapping portions of the workflow, by trying alternative data sources for staging data, and, when all else fails, by providing a rescue workflow containing a description of only the work that remains to be done.
Pegasus workflows have 4 components:

- **DAX** - Abstract workflow description containing compute steps and dependencies between the steps. This is called abstract because it does not contain data locations and available software. The DAX format is XML, but it is most commonly generated via the provided APIs (documentation). Python, Java, and Perl APIs are available.

- **Transformation Catalog** - Specifies locations of software used by the workflow

- **Replica Catalog** - Specifies locations of input data

- **Site Catalog** - Describes the execution environment

However, for simple workflows, the transformation and replica catalog can be contained inside the DAX, and to further simplify the setup, the following examples generate the site catalog on the fly. This means that the user really only has to be concerned about creating the DAX.

For details, please refer to the Pegasus documentation

### Montage Workflow

The Montage engine is a workflow application that is highly parallelizable, and performs all the tasks needed to assemble a set of input images into a mosaic: processing the input images to the required spatial scale, coordinate system, image projection; rectifying the background emission across the images to a common level, and co-adding the processed, rectified images to make the final output mosaic.
In the following exercises, we will be using Pegasus and Montage to create a 1x1 degree mosaic.

**Exercise 1:** The first step is to launch a new instance which will become our new submit node. We will then log in to this instance and create and run our workflow.

Important: verify that your security group accepts traffic from itself (the security group id) and port 80 and 8080. It should look something like this in the Amazon web console, with the difference being that you should have your security group id in the rule:

![Security Group Configuration](image)

Then start an instance of type **m2.4xlarge**, using the prepared image ami-f4e47cc4 in the Oregon region. Once the instance has started, use ssh to connect as the montage user:

```bash
$ ssh -i myec2key.pem montage@?????.compute.amazonaws.com
```
Once logged in, you should be able to use the HTCondor query commands to verify that you have a pool to run jobs in. The condor_status command should list the jobs slots (one per core), and condor_q should show an empty job queue.

```
$ condor_status
Name OpSys Arch State Activity LoadAv Mem ActvtyTime
slot1@ip-10-217-17 LINUX X86_64 Unclaimed Idle 0.000 8543 0+00:44:47
slot2@ip-10-217-17 LINUX X86_64 Unclaimed Idle 0.000 8543 0+00:45:05
slot3@ip-10-217-17 LINUX X86_64 Unclaimed Idle 0.000 8543 0+00:45:06
slot4@ip-10-217-17 LINUX X86_64 Unclaimed Idle 0.000 8543 0+00:45:07
Total Owner Claimed Unclaimed Matched Preempting Backfill
X86_64/LINUX 4 0 0 4 0 0 0
Total 4 0 0 4 0 0 0
```

```
$ condor_q
ID OWNER SUBMITTED RUN_TIME ST PRI SIZE CMD
0 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended
```

**Exercise 2:** Generate a Montage abstract workflow (DAX) by creating a working directory and running the mDAG command:

```
$ mkdir workflow
$ cd workflow
$ mDAG 2mass j M17 0.5 0.5 0.0002777778 . file://$PWD file://$PWD/inputs
```

Behind the scene, the mDAG command contacts IPACs data find services and queries for available files for the given survey, band, location and size of the output mosaic (0.5 x 0.5 in this case). The command generates a set files describing the input data, and workflow with the necessary Montage commands to generate the mosaic.

Open up the dag.xml file and note how the DAX is devoid of data movement and job details. These are added by Pegasus when the DAX is planned to an executable workflow, and provides the higher level abstraction mentioned earlier.