Ghub Pegasus WMS Workflow with Python Example

Ghub

- Ghub scientific gateway is built on the HUBzero platform for scientific collaboration.
- Ghub provides you and other members of the Ghub community a single community place, accessible from anywhere in the world via a web browser, and hosts analytical tools, data, and other shared resources.
- Who develops the analytical tools on Ghub? You and others member of the Ghub community.
- Here we outline the procedure for developing a Pegasus WMS computational workflow analytical tool on Ghub.

Pegasus WMS

- Computational workflows are a formalization of the manual computational workflow job steps that a scientist performs to obtain scientific results.

- A Workflow Management System (WMS) comprises software that manages the distribution and execution of computational workflows.

- The Pegasus WMS was selected as the best general purpose WMS to provide the structured platform required for implementing computational workflows on the HUBzero platform [1, 2, 3].

Example Pegasus WMS Architecture

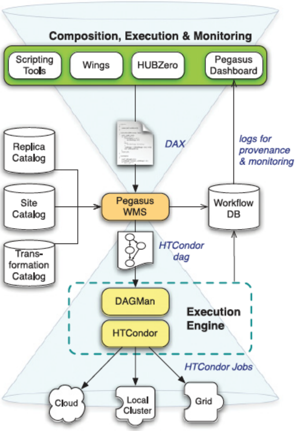


Figure 1: Pegasus WMS Architecture Example[4]

Pegasus WMS + DAGMan + HTCondor

- HTCondor is a workload scheduling system for computational jobs.
- HTCondor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management.
- DAGMan is a HTCondor tool that allows multiple jobs to be organized as a workflow, represented as a directed acyclic graph (DAG) in which the nodes represent computational tasks and edges represent the dependencies of those tasks [5].
- DAGMan automatically submits workflow jobs such that certain jobs need to be complete before other jobs start running.
- DAGMan provides the workflow engine for Pegasus.

Pegasus WMS Information Catalogs

- Pegasus requires three information catalogs to plan a workflow. These are the Site Catalog, the Transformation Catalog and the Replica Catalog.
- The Site Catalog describes the site where the workflow jobs are to be executed. For Ghub, workflow jobs are executed on CCR's UB-HPC cluster general-compute nodes by interfacing with CCR's SLURM (Simple Linux Utility for Resource Management) Workload Manager.
- The Transformation Catalog describes the launch scripts invoked by the workflow jobs. The launch scripts loads and then runs executable modules available on CCR. Execution of workflow jobs on UB-HPC requires access to executable modules available on CCR.
- The Replica Catalog tells Pegasus where to find the input files required by the workflow jobs.

Ghub Pegasus WMS Interface

- Pegasus comprises a set of system components which can be invoked via command line tools or an API interface.
- On Ghub, a Jupyter Notebook provides the interface which allows the user to interface with Pegasus, via Pegasus Python API commands, to create a YAML formatted file defining the workflow jobs, the input and output for the jobs, and the job dependencies.
- The Ghub submit command line tool allows Pegasus jobs to be executed remotely.
- The Ghub submit command line tool is used to plan and launch Pegasus workflow jobs via the Pegasus kickstart process.

Compiling and Running a Pegasus WMS Workflow

- Pegasus processes the YAML file, resolving data and software locations and all required data movements, and creates the required information catalogs and compiles a DAG file. DAGMan, as directed by the DAG file, orders the jobs according to their dependencies, and submits the jobs ready for execution to HTCondor.
- SLURM provides the framework for queuing jobs, allocating compute nodes, and starting the execution of jobs. When a SLURM job execution completes, the final status of the finished job is returned to Pegasus.

Pegasus WMS Workflow Process Steps I

Step 1: Get user's workflow parameters.

Step 2: Create a workflow using Pegasus API commands.

Step 3: For each executable required by the workflow:
Create a launch script for the executable.
Add the launch script to the Transformation Catalog.

Step 4: For each parallel run of the launch script(s):
Create a job for the launch script.
Add arguments for the job based on user's parameters.
Add inputs and outputs for the job.
Add dependencies for the job.
Add the job to the workflow.
Add the inputs for the job to the Replica Catalog.

Pegasus WMS Workflow Process Steps II

Step 5: Create the YAML file for the workflow.

Step 6: Plan and submit the YAML file.

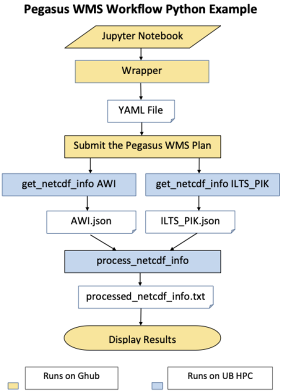
Step 7: Wait for the workflow to complete.

Step 8: View and analyze the workflow's output.

Python Example

- The procedure to implement a simple diamond Pegasus workflow is demonstrated with the Ghub_Pegasus_WMS_Python_Example project.
- This project's workflow executes Python scripts on CCR to analyze time series data from experiments files contained within predetermined UB CCR's mapped collection's modeling groups.
- Clone the Ghub_Pegasus_WMS_Python_Example on Ghub and follow the steps outlined by the README.md file.

Diamond Pipeline Workflow Architecture



Pegasus WMS Workflow Python Example

Jupyter Notebook

Wrapper

YAML File

Submit the Pegasus WMS Plan

get_netcdf_info AWI

get_netcdf_info ILTS_PIK

AWI.json

ILTS_PIK.json

process_netcdf_info

processed_netcdf_info.txt

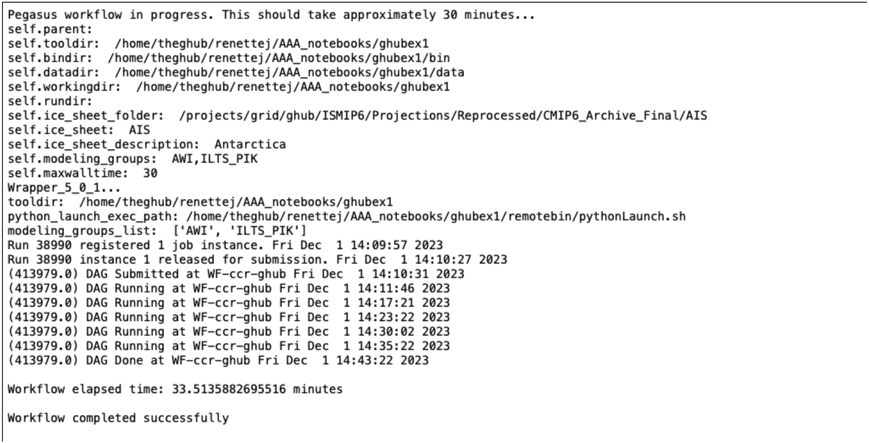
Display Results

🟨 Runs on Ghub

🟦 Runs on UB HPC

Figure 2: User Interface

Workflow's Output



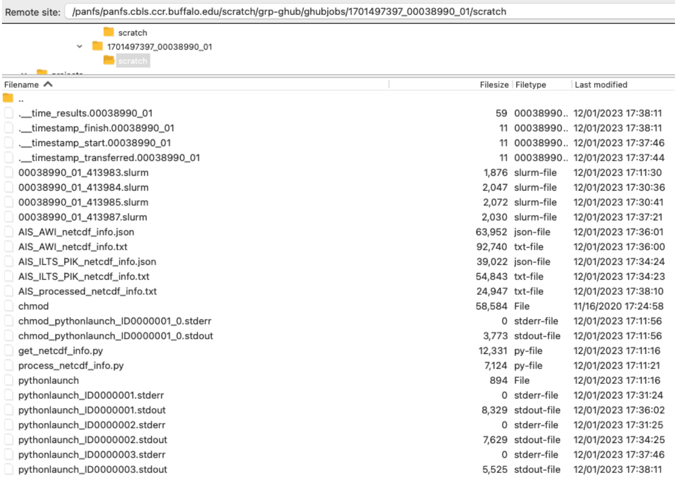
```
Pegasus workflow in progress. This should take approximately 30 minutes...
self.parent:
self.tooldir:  /home/theghub/renettej/AAA_notebooks/ghubex1
self.bindir:  /home/theghub/renettej/AAA_notebooks/ghubex1/bin
self.datadir:  /home/theghub/renettej/AAA_notebooks/ghubex1/data
self.workingdir:  /home/theghub/renettej/AAA_notebooks/ghubex1
self.rundir:
self.ice_sheet_folder:  /projects/grid/ghub/ISMIP6/Projections/Reprocessed/CMIP6_Archive_Final/AIS
self.ice_sheet:  AIS
self.ice_sheet_description:  Antarctica
self.modeling_groups:  AWI,ILTS_PIK
self.maxwalltime:  30
Wrapper_5_0_1...
tooldir:  /home/theghub/renettej/AAA_notebooks/ghubex1
python_launch_exec_path: /home/theghub/renettej/AAA_notebooks/ghubex1/remotebin/pythonLaunch.sh
modeling_groups_list:  ['AWI', 'ILTS_PIK']
Run 38990 registered 1 job instance. Fri Dec  1 14:09:57 2023
Run 38990 instance 1 released for submission. Fri Dec  1 14:10:27 2023
(413979.0) DAG Submitted at WF-ccr-ghub Fri Dec  1 14:10:31 2023
(413979.0) DAG Running at WF-ccr-ghub Fri Dec  1 14:11:46 2023
(413979.0) DAG Running at WF-ccr-ghub Fri Dec  1 14:17:21 2023
(413979.0) DAG Running at WF-ccr-ghub Fri Dec  1 14:23:22 2023
(413979.0) DAG Running at WF-ccr-ghub Fri Dec  1 14:30:02 2023
(413979.0) DAG Running at WF-ccr-ghub Fri Dec  1 14:35:22 2023
(413979.0) DAG Done at WF-ccr-ghub Fri Dec  1 14:43:22 2023

Workflow elapsed time: 33.5135882695516 minutes

Workflow completed successfully
```

Figure 3: Workflow Output

Workflow’s UB CCR Scratch Directory



Remote site: /panfs/panfs.cbls.ccr.buffalo.edu/scratch/grp-ghub/ghubjobs/1701497397_00038990_01/scratch

Filename	Filesize	Filetype	Last modified
..			
.__time_results.00038990_01	59	00038990..	12/01/2023 17:38:11
.__timestamp_finish.00038990_01	11	00038990..	12/01/2023 17:38:11
.__timestamp_start.00038990_01	11	00038990..	12/01/2023 17:37:46
.__timestamp_transferred.00038990_01	11	00038990..	12/01/2023 17:37:44
00038990_01_413983.slurm	1,876	slurm-file	12/01/2023 17:11:30
00038990_01_413984.slurm	2,047	slurm-file	12/01/2023 17:30:36
00038990_01_413985.slurm	2,072	slurm-file	12/01/2023 17:30:41
00038990_01_413987.slurm	2,030	slurm-file	12/01/2023 17:37:21
AIS_AWI_netcdf_info.json	63,952	json-file	12/01/2023 17:36:01
AIS_AWI_netcdf_info.txt	92,740	txt-file	12/01/2023 17:36:00
AIS_ILTS_PIK_netcdf_info.json	39,022	json-file	12/01/2023 17:34:24
AIS_ILTS_PIK_netcdf_info.txt	54,843	txt-file	12/01/2023 17:34:23
AIS_processed_netcdf_info.txt	24,947	txt-file	12/01/2023 17:38:10
chmod	58,584	File	11/16/2020 17:24:58
chmod_pythonlaunch_ID0000001_0.stderr	0	stderr-file	12/01/2023 17:11:56
chmod_pythonlaunch_ID0000001_0.stdout	3,773	stdout-file	12/01/2023 17:11:56
get_netcdf_info.py	12,331	py-file	12/01/2023 17:11:16
process_netcdf_info.py	7,124	py-file	12/01/2023 17:11:21
pythonlaunch	894	File	12/01/2023 17:11:16
pythonlaunch_ID0000001.stderr	0	stderr-file	12/01/2023 17:31:24
pythonlaunch_ID0000001.stdout	8,329	stdout-file	12/01/2023 17:36:02
pythonlaunch_ID0000002.stderr	0	stderr-file	12/01/2023 17:31:25
pythonlaunch_ID0000002.stdout	7,629	stdout-file	12/01/2023 17:34:25
pythonlaunch_ID0000003.stderr	0	stderr-file	12/01/2023 17:37:46
pythonlaunch_ID0000003.stdout	5,525	stdout-file	12/01/2023 17:38:11

Figure 4: UB CCR Scratch Directory

References I

- Michael McLennan, Steven Clark, Ewa Deelman, Mats Rynge, Karan Vahi, Frank McKenna, and Derrick Kearney. Bringing Scientific Workflow to the Masses via Pegasus and HUBzero. *SemanticScholar*, 2013.
- Michael McLennan, Steven Clark, Ewa Deelman, Mats Rynge, Karan Vahi, Frank McKenna, Derrick Kearney, and Carol Song. HUBzero and Pegasus: integrating scientific workflows into science gateways. *Wiley*, 2014.

References II

- Ewa Deelman, Karan Vahi, Gideon Juve, Mats Rynge, Scott Callaghan, Philip J. Maechling, Rajiv Mayani, Weiwei Chen, Rafael Ferreira da Silva, Miron Livny, and Kent Wenger. Pegasus, a workflow management system for science automation. *Future Generation Computer Systems*, 46:17–35, 2015. ISSN: 0167-739X. DOI: `https://doi.org/10.1016/j.future.2014.10.008`. URL: `https://www.sciencedirect.com/science/article/pii/S0167739X14002015`.
- C. S. Liew, M. P. Atkinson, M. Galea, T. F. Ang, P. Martin, and J. I. V. Hemert. Scientific workflows. *Computing Surveys*, 49(4):1–39, 2016. DOI: `https://doi.org/10.1145/3012429`.

References III

Ryan Mitchell, Loïc Pottier, Steve Jacobs, Rafael Ferreira da Silva, Mats Rynge, Karan Vahi, and Ewa Deelman. Exploration of Workflow Management Systems Emerging Features from Users Perspectives. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4537–4544, 2019. DOI: `10.1109/BigData47090.2019.9005494`.