Pegasus Workflow Manager on Perlmutter

Data Day 2022

Nick Tyler
Data Science Engagement Group
Oct 25, 2022
What is a Workflow?

• Term “Workflow” is used a lot
• Different parts of analysis
• Steps in a data intensive process
  o Sbatch job script
  o Custom infrastructure
• Workflow Tools and Engines
  o Pegasus
  o Parsl/FuncX
  o Snakemake
  o Many more!
What is a Workflow?

- Term “Workflow” is used a lot
- Different parts of analysis
- Steps in a data intensive process
  - Sbatch job script
  - Custom infrastructure
- Workflow Tools and Engines
  - Pegasus
  - Parsl/FuncX
  - Snakemake
  - Many more!
Goals of workflow tools

- Automation
- Reproducibility
- Share work with others
- Processed newest data
- Track data in the pipeline
- Use resources efficiently
- Get results faster
- Share work with others
- Less human in the loop
Pegasus Workflow Manager
What is pegasus?

• Workflow manager
• Define workflow using yaml files
  o replicas.yml
  o sites.yml
  o transformations.yml
  o workflow.yml
• There are APIs to create these yaml files
  o Python, Java, R
• Show using Python API
• Example on data day github for perlmutter
What is pegasus?

- Pegasus WMS
  - Pegasus APIs plan DAG
- Directed Acyclic Graph
  - Graph representing the work to be done
  - Nodes are executions
  - Edges show dataflow
  - Dependencies
- HTCondor Scheduler
  - DAGMan
  - Handling DAG execution
  - Scheduler manages execution of workflow
Writing the workflow

• Things to consider when building the workflow
  • What executables are we going to run?
    o Are we using a container?
  • What data do we have?
    o What are the inputs?
    o What are the outputs?
  • What are the dependencies?
    o What tasks depend on outputs from previous tasks?
    o How are they connected?
# --- Transformation Catalog (Executables and Containers) -------------------

def create_transformation_catalog(self, exec_site_name="perlmutter"):
    self.tc = TransformationCatalog()

    # Create a container to run exes in
    ubuntu = Container(
        "ubuntu",
        Container.SHIFTER,
        image="shifter://ubuntu:latest"
    )

    # Add it to the yml file
    self.tc.add_containers(ubuntu)

    # Create transforms or exes
    wc = Transformation(
        "wc", site=exec_site_name, pfn="/usr/bin/wc", is_stageable=False,
    )

    # The split command will be run in the container
    split = Transformation(
        "split", site=exec_site_name, pfn="/usr/bin/split", is_stageable=False,
        container=ubuntu
    )

    # Add the exes to the yml file
    self.tc.add_transformations(split, wc)
Transforms

• Define Executables

```python
# --- Transformation Catalog (Executables and Containers) ---------------

def create_transformation_catalog(self, exec_site_name="perlmutter"):  
    self.tc = TransformationCatalog()

    # Create transforms for exes
    wc = Transformation(
        "wc", site=exec_site_name, pfn="/usr/bin/wc", is_stageable=False,
    )
    split = Transformation(
        "split", site=exec_site_name, pfn="/usr/bin/split", is_stageable=False
    )

    # Add the exes to the yml file
    self.tc.add_transformations(split, wc)
```
Replica

• Defines Data

```python
# --- Replica Catalog ---------------------------------------------

def create_replica_catalog(self):
    self.rc = ReplicaCatalog()

    # This is the input data we will be using
    self.rc.add_replica(
        "local", "test.csv", os.path.join(
            self.wf_dir, "input", "test.csv")
    )

    # Output data is added to the replica with
    # the register_replica options in the workflow section
```
Using the transforms and replicas let's build the workflow.

```python
# --- Create Workflow -----------------------------------------------

def create_workflow(self):
    self.wf = Workflow(self.wf_name, infer_dependencies=True)
    # Defines the test file
    test_file = File("test.csv")
    num_splits = 4
    # the split job that splits the test file into smaller chunks
    split = (      
        Job("split")
        .add_args("-n", num_splits, "-d", "-a", 1, test_file, "part."")
        .add_inputs(test_file)
        .add_pegasus_profile(label="p1")
    )
    self.wf.add_jobs(split)
```
Workflow

• Using the transforms and replicas let's build the workflow

```python
# we do a parameter sweep on the first 4 chunks created
for c in range(num_splits):
    part = File("part.%s" % c)
    split.add_outputs(part, stage_out=True, register_replica=True)
    count = File("count.txt.%s" % c)
    wc = (
        Job("wc")
        .add_args("-l", part)
        .add_inputs(part)
        .set_stdout(count, stage_out=True, register_replica=True)
        .add_pegasus_profile(label="p1")
    )

    self.wf.add_jobs(wc)
```
Generating the workflow

• **python generate_workflow.py**
  - Creates the yml files
• **replicas.yml**
  - Defines the storage and data
• **sites.yml**
  - Defines the job parameters for the site sbatch
• **transformations.yml**
  - Defines the executables and their parameters
• **workflow.yml**
  - Defines the workflow
  - Workflows use transformations of replicas
Getting setup to run your workflow

• For running on perlmutter we’ll setup HTCondor as a workflow job
  o Uses scrontab to setup longer running workflow jobs
• HTCondor is a job scheduler
  o Built for High Throughput workloads
  o 100s-1000s of small jobs
  o Small job requirements (<< 1Node)
• Pegasus uses HTCondor to run workflows
  o HTCondor is a scheduler
Getting setup to run your workflow

- Check that we have HTCondor working
  - `condor_status -any`

<table>
<thead>
<tr>
<th>MyType</th>
<th>TargetType</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collector</td>
<td>None</td>
<td>My Pool - login32-perlmutter@login32</td>
</tr>
<tr>
<td>Submitter</td>
<td>None</td>
<td>condor_pool@jaws-condor</td>
</tr>
<tr>
<td>DaemonMaster</td>
<td>None</td>
<td>tylern@login32</td>
</tr>
<tr>
<td>Negotiator</td>
<td>None</td>
<td>tylern@login32</td>
</tr>
<tr>
<td>Scheduler</td>
<td>None</td>
<td>tylern@login32-perlmutter</td>
</tr>
<tr>
<td>Accounting</td>
<td>none</td>
<td>&lt;none&gt;</td>
</tr>
</tbody>
</table>
Pegasus Commands

- **pegasus-plan --submit**
Pegasus Commands

• `pegasus-plan --submit`

```bash
(pegasus)[perlmutter-login15:....-pegasus-example/perlmutter]$ pegasus-plan --submit
2022.10.21 14:02:12.237 PDT: File for submitting this DAG to HTCondor: split-0.dag.condor.sub
2022.10.21 14:02:12.245 PDT: Log of DAGMan debugging messages: split-0.dag.dagman.out
2022.10.21 14:02:12.250 PDT: Log of HTCondor library output: split-0.dag.lib.out
2022.10.21 14:02:12.255 PDT: Log of HTCondor library error messages: split-0.dag.lib.err
2022.10.21 14:02:12.260 PDT: Log of the life of condor_dagman itself: split-0.dag.dagman.log
2022.10.21 14:02:12.265 PDT: -no_submit given, not submitting DAG to HTCondor. You can do this with:
2022.10.21 14:02:12.270 PDT: /tylerm/peg
2022.10.21 14:02:12.275 PDT: utter/tylerm
2022.10.21 14:02:12.281 PDT: /global/u/t/tylerm/nersc-pegasus-example/perlmutter/tylerm/pegasus/split/run0003
2022.10.21 14:02:12.286 PDT: Your workflow has been started and is running in the base directory: /global/u/t/tylerm/nersc-pegasus-example/perlmutter/tylerm/pegasus/split/run0003
2022.10.21 14:02:12.291 PDT: To monitor the workflow you can run ***
2022.10.21 14:02:12.296 PDT: pegasus-status -l /global/u/t/tylerm/nersc-pegasus-example/perlmutter/tylerm/pegasus/split/run0003
2022.10.21 14:02:12.301 PDT: To remove your workflow run ***
2022.10.21 14:02:12.311 PDT: Time taken to execute is 4.18 seconds
```
Pegasus Commands

- `pegasus-analyzer path/to/workflow/run0001`
Watching the jobs progress

• **condor_q**

```
(pegasus)[perlmutter-login15:...-pegasus-example/perlmutter]$ condor_q

-- Schedd: tylnr@login32 : <10.252.1.147:9876>... @ 10/21/22 14:03:29
OWNER BATCH_NAME SUBMITTED DONE RUN IDLE TOTAL JOB_IDS
condor_pool split-0.dag+27 10/21 14:02 _ _ 1 1 28.0

Total for query: 1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
Total for all users: 1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
```

• **sq**

```
(pegasus)[perlmutter-login15:...-pegasus-example/perlmutter]$ sq

<table>
<thead>
<tr>
<th>JOBID</th>
<th>ST</th>
<th>USER</th>
<th>NAME</th>
<th>NODES</th>
<th>TIME</th>
<th>QOS</th>
<th>FEATURES</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3378786</td>
<td>R</td>
<td>tylnr</td>
<td>htcondor_workflow_node</td>
<td>1</td>
<td>6-22:34:09</td>
<td>workflow</td>
<td>cron</td>
<td>login32</td>
</tr>
<tr>
<td>3470264</td>
<td>PD</td>
<td>tylnr</td>
<td>stageinremotepe</td>
<td>1</td>
<td>0:00</td>
<td>debug</td>
<td>cpu</td>
<td>(Priority)</td>
</tr>
</tbody>
</table>
```
Watching the jobs progress
Questions?

• Pegasus is just one of many workflow tools
• Each has its advantages and disadvantages
• Checkout all the tools we have on our docs