Is it an HPC Workflow Assistant? Is it a Framework? It's Drona Workflow Engine

Marinus Pennings
Andrii Kryvenko, Duy Pham, Honggao Liu
11/16/2025

High Performance
Research Computing
DIVISION OF RESEARCH

Outline

📜 History of GUI Composers at HPRC

🧱 Limits & Bottlenecks of our GUI composer

🎯 Introducing Drona, design Goals

🤖 Drona: an HPRC Workflow Assistant

⚙️ Drona: a Framework

🧘 Monitoring and Managing Workflows

🔍 Some Examples of Non Traditional HPC Workflows

✅ Final thoughts, Questions and it’s a wrap

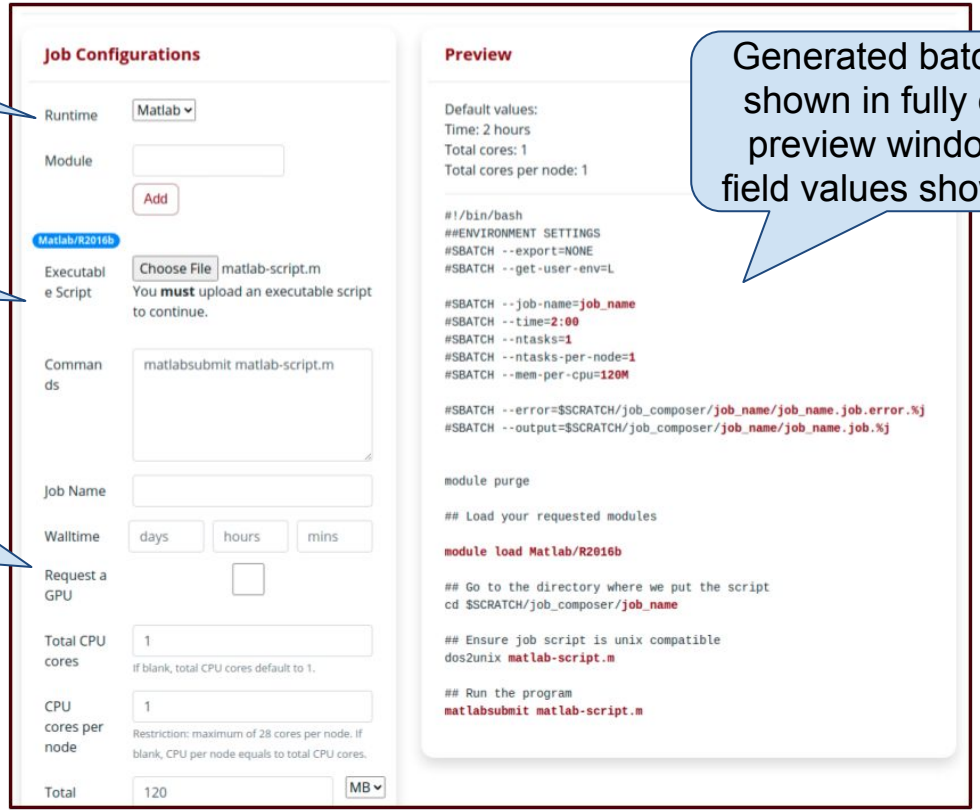
History of GUI Composers at HPRC

Select runtime

Runtime specific fields

Batch scheduler fields

Generated batch script, shown in fully editable preview window, form field values shown in red



PEARC22: Extending Functionalities on a Web-based Portal for Research Computing

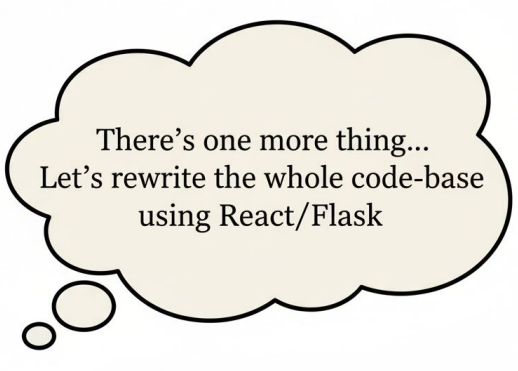
DOI: /10.1145/3491418.3535182

Limitations

- User's don't need to know scheduler directive syntax ✅ … but
  - Still need to be aware of HPC concepts
- Our dashboard didn't work well for certain workflows
  - Workflows consisting of multiple jobs, pipelines in general
  - Dynamic workflows, depending on provided input
  - Limited to single script jobs
- Our Form fields mostly static
  - Sometimes need to retrieve information dynamically (e.g. accounts)
- Limited options for checking input
  - Mostly sanity check for individual fields
- Sysadmin (or OOD/app maintainer) in charge
  - They decide what environments to provide
  - Also makes them responsible to create and maintain those environments

Design Goals for a new GUI Composer

- Address all the issues mentioned in the previous slide (duh )
  - Abstract hardware as much as possible
  - Better support for alternative workflows
  - Better checking and Feedback opportunities
  - Better form support, more options out of the box, allow for dynamic input
- Researcher should be able to focus on running their workflows
  - not worry about HPC specifics
  - Even more important now, with a large variety of accelerators
- Researcher should be in control of their workflows
  - create/customize/share
  - Ability to easily rerun and reproduce results



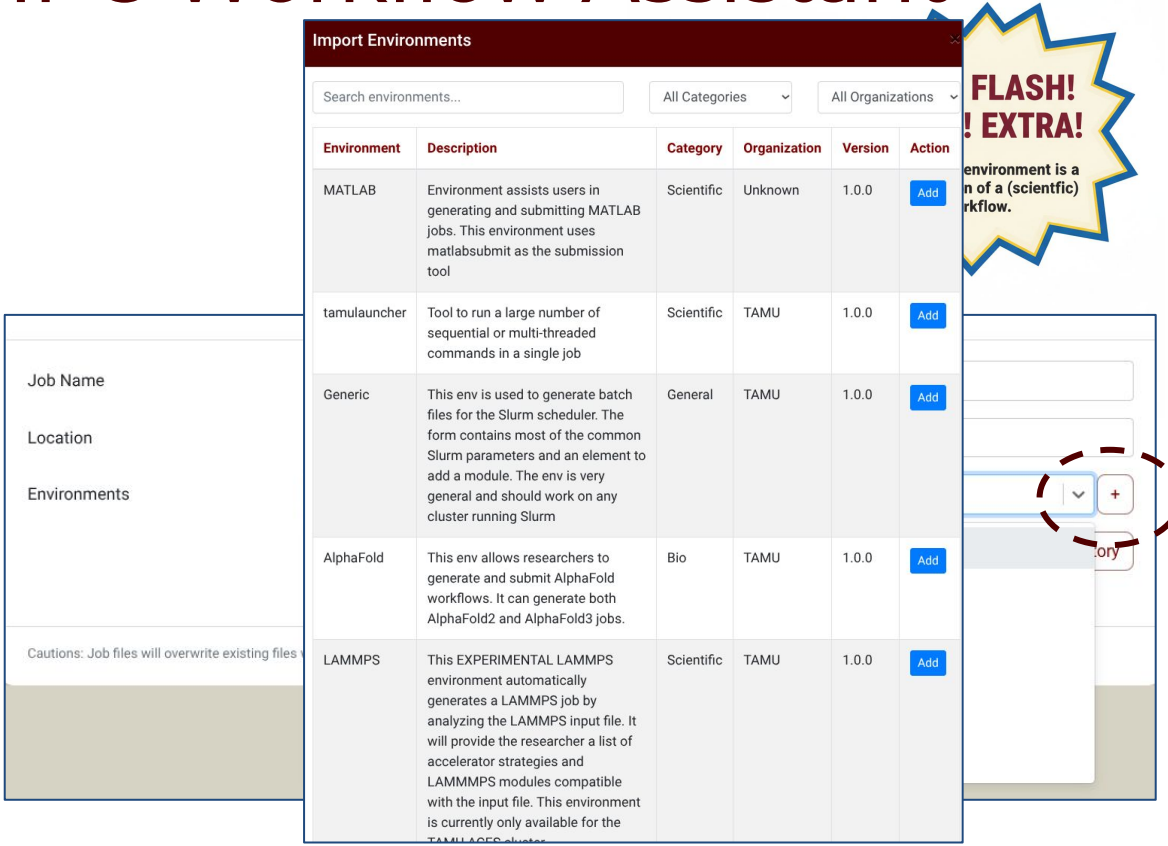
Introducing Drona Workflow Engine

- Drona → An HPC Workflow Assistant
  - A powerful GUI
  - Rich set of form elements, including dynamic retrievers
  - Instant Feedback
  - Editable Preview window
  - Live output window
- Drona → A Framework to create Workflows
  - Workflows completely independent of Drona Workflow Engine
  - Defined in mostly declarative manner
  - Flexible enough to easily create simple workflows as well as complex, multi script workflows
  - Not limited to just submitting standard batch jobs

Drona: An HPC Workflow Assistant

Step 1: Select Environment

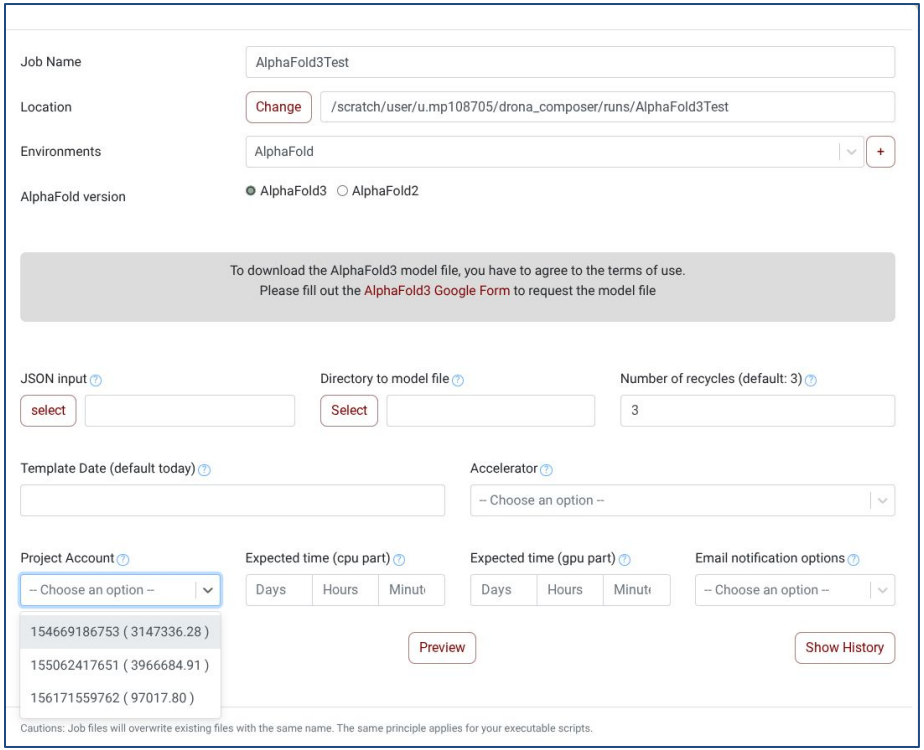
- Select Environment from dropdown
- Environments color coded
  - System envs → black
  - User envs → blue
- Option to Import additional environments.



Drona: An HPC Workflow Assistant

Step 2: Provide information

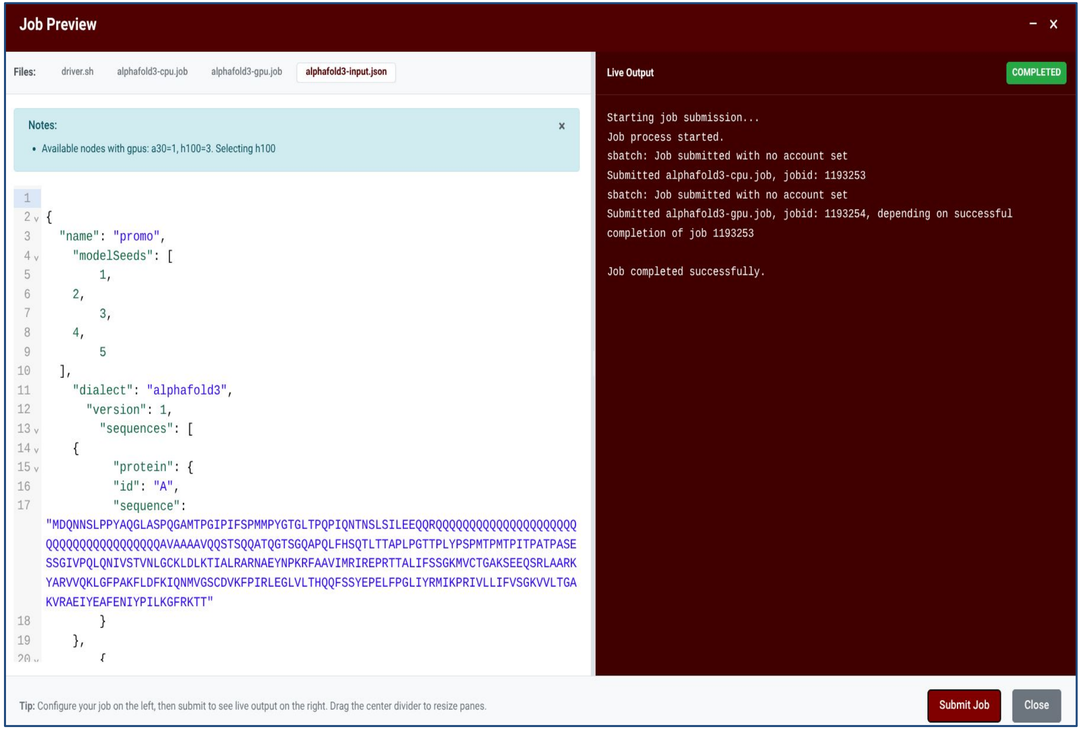
- Form expands with environment specific fields
- Rich set of elements
  - Dynamic HTML
- Dynamic retrievers
- Conditional form fields



Drona: An HPC Workflow Assistant

Step 3: Preview / live output

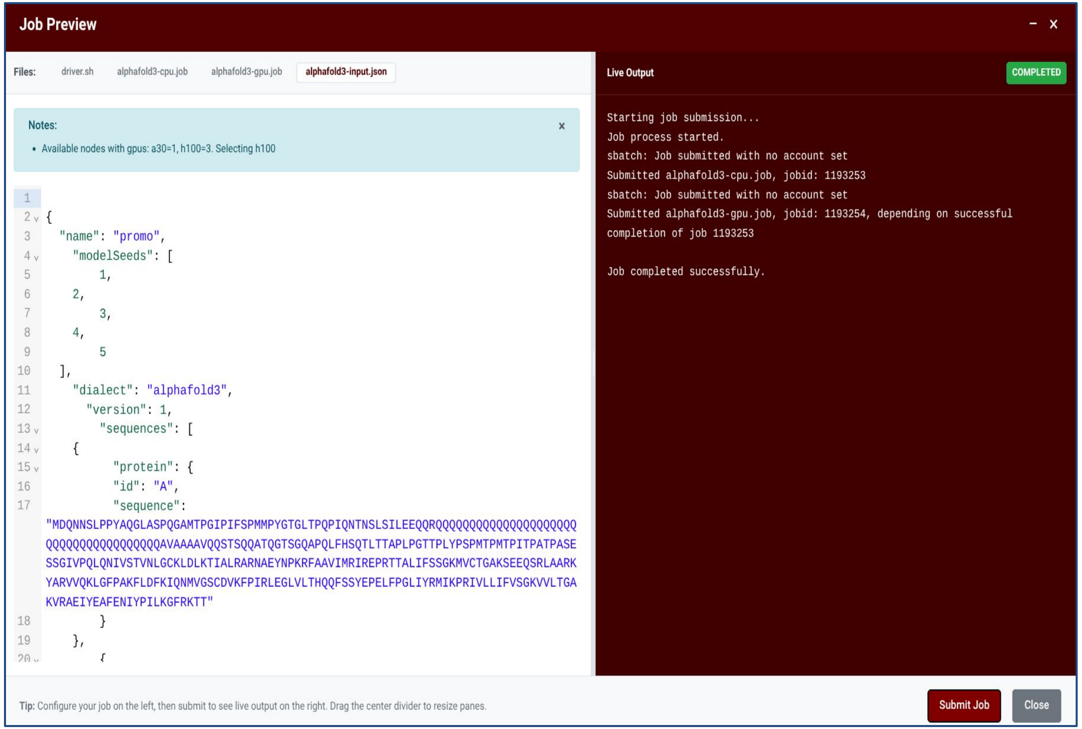
- Editable preview window
- Syntax-highlighting
- Notes and Warnings



Drona Workflow Engine

Step 4: start Workflow

- Live output streaming



Introducing Drona Workflow Engine

- Drona → An HPC Workflow Assistant
  - A powerful GUI
  - Rich set of form elements, including dynamic retrievers
  - Instant Feedback
  - Editable Preview window
  - Live output window
- Drona → A Framework to create Workflows
  - Workflows completely independent of Drona Workflow Engine
  - Defined in mostly declarative manner
  - Flexible enough to easily create simple workflows as well as complex, multi script workflows
  - Not limited to just submitting standard batch jobs

Drona: A Framework to Create Workflows

A Drona environment is:

- Completely independent from Drona Workflow Engine
- Defined in mostly declarative manner:
  - schema.json →defines the form
  - maps.json → advanced mapping
  - driver.sh → script to setup and start the workflow
  - Template files → templated scripts
  - utils.py → optional user functions
- Stored in researcher's local directory

schema.json

```
"X": {
    "type": "number",
    "label": "Input param",
    "help" : "Enter input"
  }
    :
```

map.json

```
"X_MAPPED": "!myfun($X)"
```

utils.py

```
def myfun(var1):
   if is_even(var1):
      return var1 + 1
   else:
       return var1
```

Template Files

```
#SBATCH –ntasks=1
#SBATCH –memory=1G

./mycode [X_MAPPED]
```

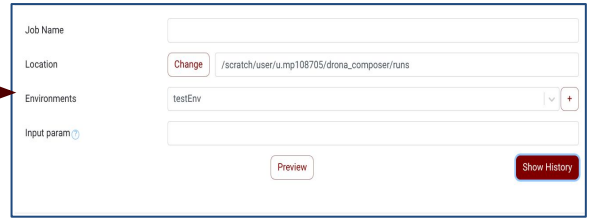
Drona: A Framework to Create Workflows

Step 1: Render the form

- Once researcher selects a workflow, Drona WFE reads the schemas.json file and dynamically renders the form

schema.json

```
"X": {
    "type": "number",
    "label": "Input param",
    "help" : "Enter input"
  }
    :
```



schemas.json is a json formatted file containing a list of form elements. (see above for an example partial form element)

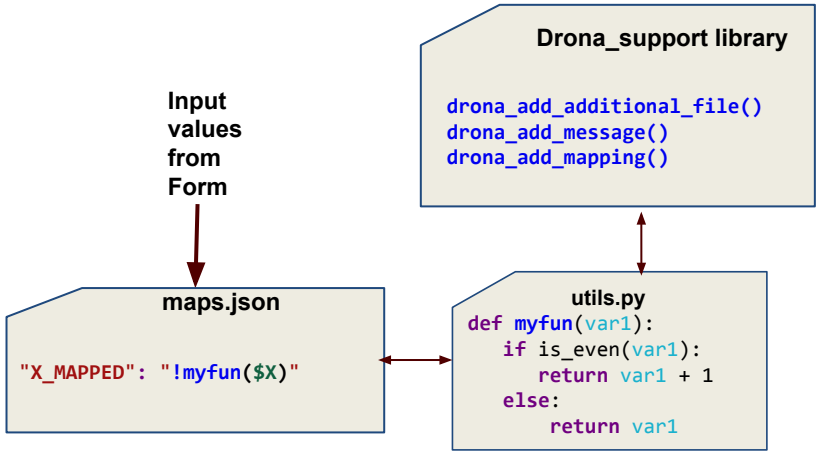
For a list of form elements

https://tamu-edu.github.io/dor-hprc-drona-composer/

Drona: A Framework to Create Workflows

Step 2: Mapping step

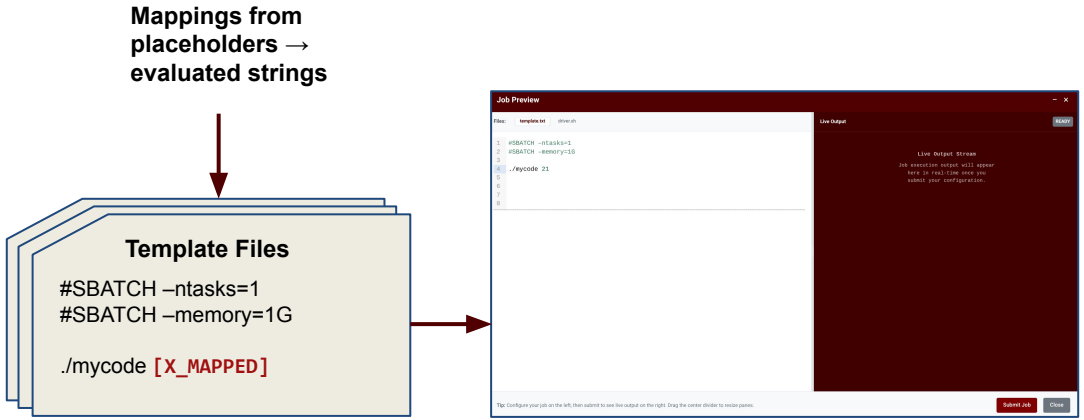
- Drona WFE maps researcher provided input values to placeholder strings using scheme in maps.json
- Format of elements
  - "LHS": “!fun($val) text $val”
  - ! represents function call
  - $ represents input variable
  - Function calls defined in utils.py
- Drona provides support library to create messages, dynamically add files, dynamically add mappings



Drona: A Framework to Create Workflows

Step 3: Replacing placeholders

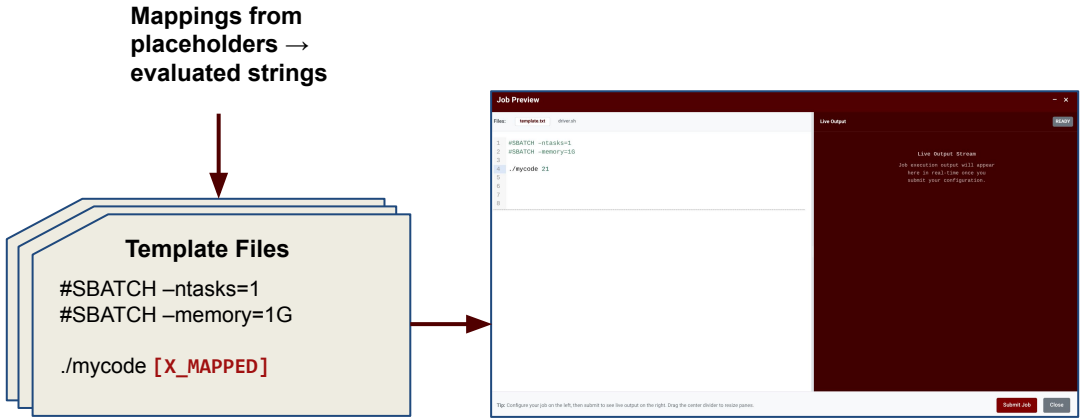
- Drona WFE iterates over all template files to replace placeholders with string values, generated in mapping step.
  - Placeholder are defined as strings enclosed by []
- Generate preview window for researcher to review and make adjustments if needed



Drona: A Framework to Create Workflows

Step 4: Run the workflow

- Start workflow
  - Create run directory (if needed)
  - Copy all generated files to that directory
  - Execute the driver.sh script
- Output streamed to output window



Monitoring/Managing Workflows

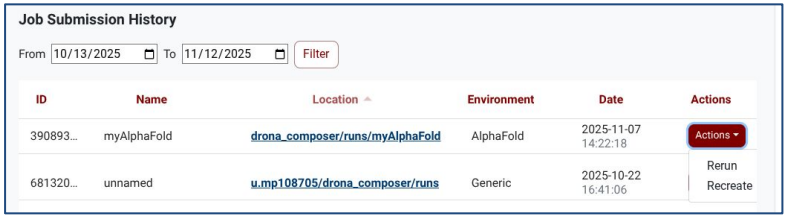
Now, we submitted a workflow, what's next???

- Monitoring is mostly a manual process
  - Check output logs and files on the command line
  - Check gpu utilization, either command line, or indirectly using external tools
  - Other domain-specific external tools
- Some tools provide options to "manage" a job
  - Mostly batch job options
    - Job information, e.g. walltime, resource info, job location ✅
    - Options to cancel a job ✅
  - Ability to rerun a job ✅
  - Not a very elegant solution for multi-step jobs (pipelines) ❌
  - No domain specific information ❌

Workflow Monitoring/Managing
(the Drona way)

Legacy Drona Management functionality:

- Workflow history (might be incorporated in other tools)
  - Shows basic info → id, name, location, start date.
  - Reproducibility options:
    - Rerun
    - Recreate

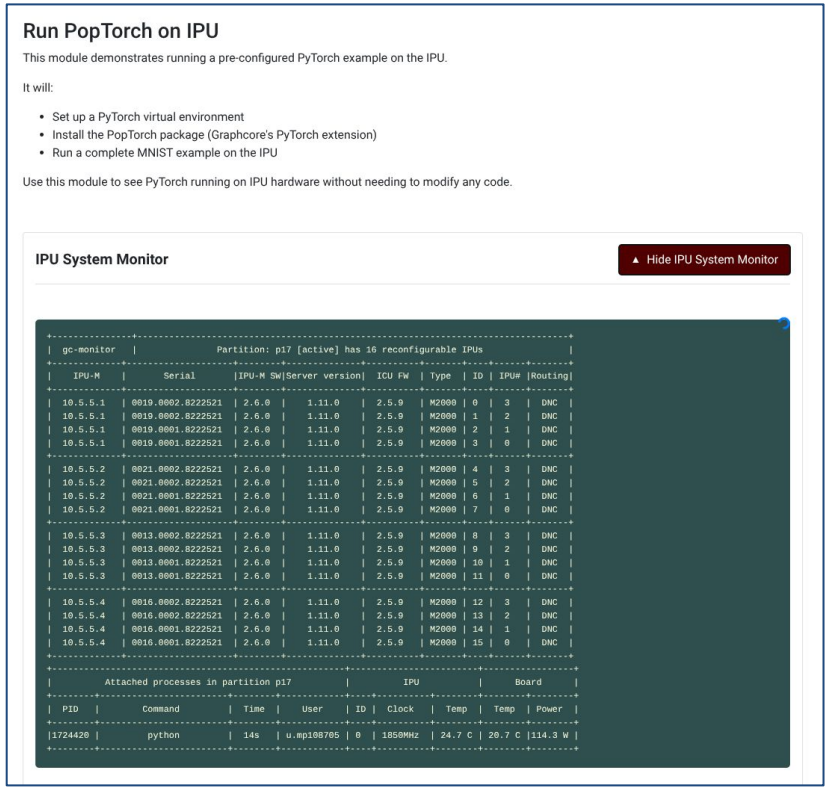


However ……..

A Drona environment has a ton of information about its workflow, why not leverage that information to provide truly useful information to the researcher

Workflow Monitoring/Managing
(the Drona way)

- Managing a workflow that has been submitted can be part of that Drona environment
  - Typical: create → submit
  - Drona: create → submit → manage
- Drona provides the tools to manage workflow
  - Specialized form elements & retriever functions
  - Tools to access/update Workflow database
  - Environment variables
    - DRONA_WF_ID
    - DRONA_WF_LOCATION
    - DRONA_WF_RUNTIME_DIR
- Example on the right: Drona IPU training env ⏩
  - Monitor IPU utilization while running exercise

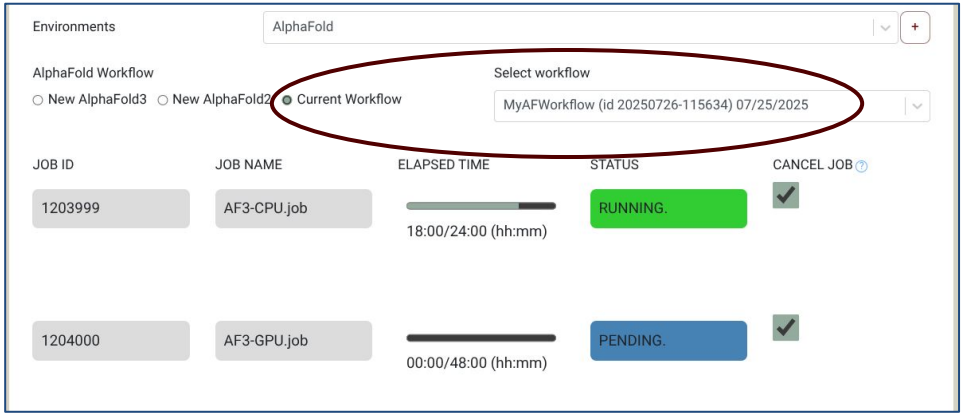


Workflow Monitoring/Managing
(the Drona way)

Example regular “traditional” workflow

Add job management to AlphaFold workflow

- Additional radio button “current workflow”
- Dropdown with all workflows
  - Select the workflow
  - dynamicSelect form element
  - Drona function to access database
- Associated jobs info
  - Shows info for all jobs in workflow
  - Option to cancel any of the jobs
  - Uses standard Form elements
  - Alternatively, staticText with retriever
- Other potentially useful info
  - Pickle, utilization, specific messages from logs, etc



Workflow Monitoring/Managing
(the Drona way)

- Our responsibility: We provide the tools

- Drona environment developer's responsibility: Create an intuitive workflow monitoring/management experience
  - We do both; provide tools and create some of the Drona environments
  - Domain expert knows better what is useful information than a sysadmin
    - For running workflows and completed workflows.
- Many other options for workflow/job management
  - E.g. a Drona Job listing environment, showing all jobs (mimics traditional job listing features)
  - Up to the creativity of the Drona environment creator

Some non Traditional Examples

Typically, (GUI) job composer tools (and Drona) are used for:

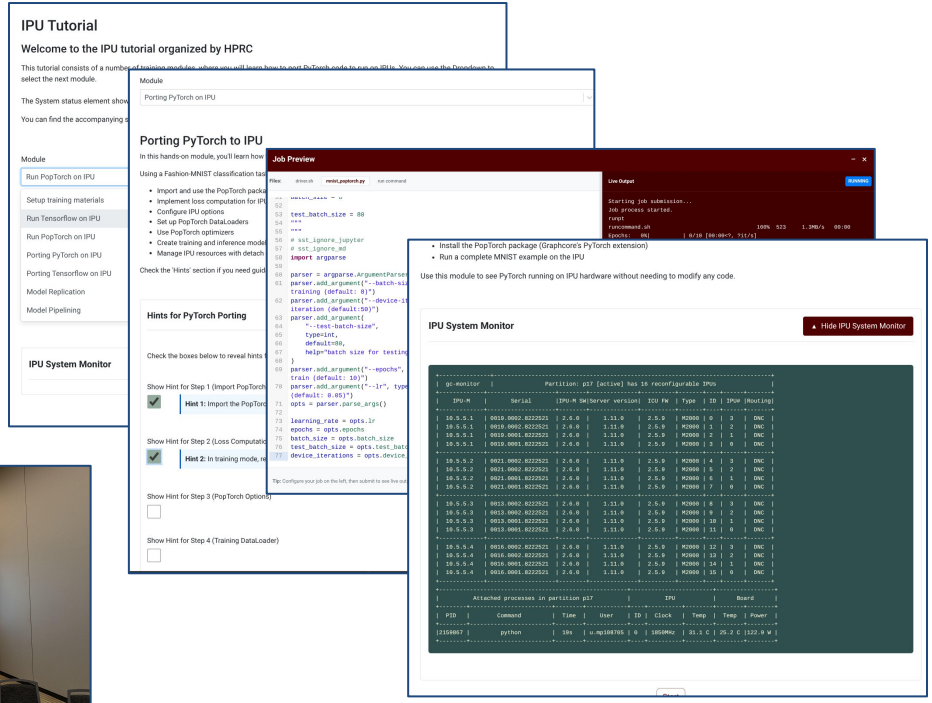
- “traditional” workflows
  - E.g. AlphaFold (from example)
- Submitted to HPC batch schedulers
  - E.g. Slurm

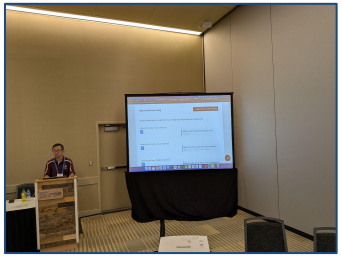
Drona can be used for so much more than just that

IPU Tutorial

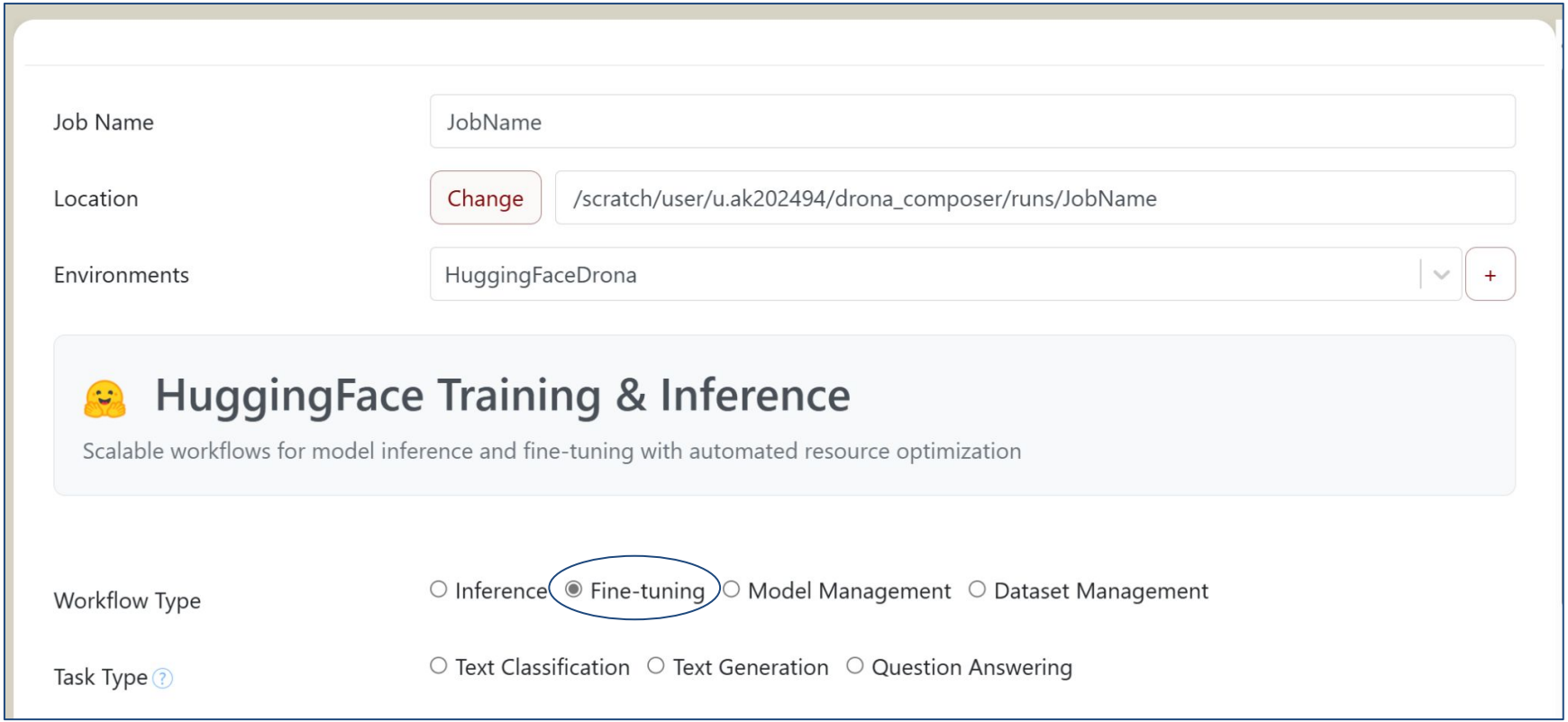
Tutorial to teach participants to port PyTorch/Tensorflow code to Graphcore IPUs

- Workflow
  - Pick training module
  - Show module goals / hints
  - Generate boilerplate code for preview
  - Participant works on exercise in preview editor (TODO items)
  - Run the exercise
  - Check live output
  - Check utilization
  - Pick next training module
- Used at PEARC25 tutorial
- ACCESS short course





Hugging Face Workflow



Final thoughts???

We believe the Drona Engine is quite stable, build upon a solid foundation to create complex workflows. Next step: build it out, and hopefully make this a community effort

- Create/share environments
- Create/share driver templates
- Create/share retriever functions
  - Like IPU and GPU utilization retriever, job info, etc

Questions???

Contact us at help@hprc.tamu.edu

Thank you

Drona Homepage

Drona GitHub page

Stop by our booth #1143 to say Hi and win some prizes ……. and to talk Drona of course

Supported by:
National Science Foundation (NSF) award number 2112356 ACES - Accelerating Computing for Emerging Sciences