

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

Job Configurations

Runtime:

Module:

Matlab/R2016b

Executable Script: matlab-script.m
You **must** upload an executable script to continue.

Commands:

Job Name:

Walltime:

Request a GPU: ☐

Total CPU cores:
If blank, total CPU cores default to 1.

CPU cores per node:
Restriction: maximum of 28 cores per node. If blank, CPU per node equals to total CPU cores.

Total:

Preview

Default values:
Time: 2 hours
Total cores: 1
Total cores per node: 1

```
#!/bin/bash
##ENVIRONMENT SETTINGS
#SBATCH --export=NONE
#SBATCH --get-user-env=L

#SBATCH --job-name=job_name
#SBATCH --time=2:00
#SBATCH --ntasks=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem-per-cpu=120M

#SBATCH --error=$SCRATCH/job_composer/job_name/job_name.job.error.%j
#SBATCH --output=$SCRATCH/job_composer/job_name/job_name.job.%j

module purge

## Load your requested modules

module load Matlab/R2016b

## Go to the directory where we put the script
cd $SCRATCH/job_composer/job_name

## Ensure job script is unix compatible
dos2unix matlab-script.m

## Run the program
matlabsubmit matlab-script.m
```

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



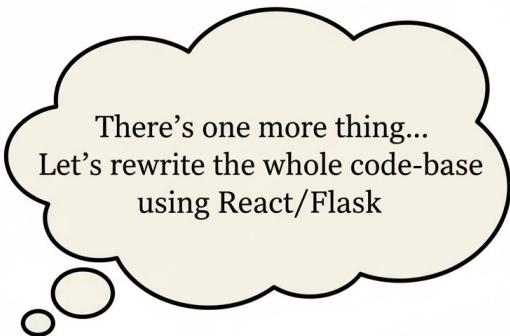
Limitations

- 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



There's one more thing...
Let's rewrite the whole code-base
using React/Flask



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.

Job Name

Location

Environments

Cautions: Job files will overwrite existing files

Import Environments					
Search environments...		All Categories	All Organizations		
Environment	Description	Category	Organization	Version	Action
MATLAB	Environment assists users in generating and submitting MATLAB jobs. This environment uses matlabsubmit as the submission tool	Scientific	Unknown	1.0.0	Add
tamulauncher	Tool to run a large number of sequential or multi-threaded commands in a single job	Scientific	TAMU	1.0.0	Add
Generic	This env is used to generate batch files for the Slurm scheduler. The form contains most of the common Slurm parameters and an element to add a module. The env is very general and should work on any cluster running Slurm	General	TAMU	1.0.0	Add
AlphaFold	This env allows researchers to generate and submit AlphaFold workflows. It can generate both AlphaFold2 and AlphaFold3 jobs.	Bio	TAMU	1.0.0	Add
LAMMPS	This EXPERIMENTAL LAMMPS environment automatically generates a LAMMPS job by analyzing the LAMMPS input file. It will provide the researcher a list of accelerator strategies and LAMMPS modules compatible with the input file. This environment is currently only available for the TAMU AFS cluster	Scientific	TAMU	1.0.0	Add

**FLASH!
! EXTRA!**

environment is a
n of a (scientific)
rkflow.

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

The screenshot displays the 'Step 2: Provide information' form in the Drona HPC Workflow Assistant. The form is organized into several sections:

- Job Information:** Includes fields for 'Job Name' (AlphaFold3Test), 'Location' (with a 'Change' button and path /scratch/user/u.mp108705/drona_composer/runs/AlphaFold3Test), 'Environments' (AlphaFold), and 'AlphaFold version' (radio buttons for AlphaFold3 and AlphaFold2).
- Terms of Use:** A grey box with text: 'To download the AlphaFold3 model file, you have to agree to the terms of use. Please fill out the AlphaFold3 Google Form to request the model file'.
- Input Fields:** Includes 'JSON input' (with a 'select' button), 'Directory to model file' (with a 'Select' button), and 'Number of recycles (default: 3)' (a text input with value 3).
- Template and Accelerator:** Includes 'Template Date (default today)' (a date input) and 'Accelerator' (a dropdown menu with '-- Choose an option --').
- Project Account:** A dropdown menu showing a list of project accounts with their IDs and CPU times, such as '154669186753 (3147336.28)'.
- Expected time:** Two sections for 'Expected time (cpu part)' and 'Expected time (gpu part)', each with dropdowns for 'Days', 'Hours', and 'Minuti'.
- Email notification options:** A dropdown menu with '-- Choose an option --'.
- Buttons:** 'Preview' and 'Show History' buttons are located at the bottom right.
- Cautions:** A small text at the bottom states: 'Cautions: Job files will overwrite existing files with the same name. The same principle applies for your executable scripts.'



Drona: An HPC Workflow Assistant

Step 3: Preview / live output

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

The screenshot displays the 'Job Preview' window in Drona. The interface is divided into three main sections: a top file browser, a central code editor, and a right-hand live output panel.

Files: driver.sh, alphafold3-cpu.job, alphafold3-gpu.job, **alphafold3-input.json**

Notes:

- Available nodes with gpus: a30=1, h100=3. Selecting h100

Code Editor (Syntax-highlighted JSON):

```
1 {
2   {
3     "name": "promo",
4     "modelSeeds": [
5       1,
6       2,
7       3,
8       4,
9       5
10    ],
11    "dialect": "alphafold3",
12    "version": 1,
13    "sequences": [
14      {
15        "protein": {
16          "id": "A",
17          "sequence":
18            "MDQNSLPPYAQGLASPGAMTPGIPFSPMPYGTGLTPQPIQNTNSLSILEEQRRQQQQQQQQQQQQQQQQQQQ
19            QQQQQQQQQQQQQQVAAAVQSTSQATQGTSGQAPQLFHSQTLTAPLPGTTPLYSPMTPTIPATPASE
20            SSGIVPQLQINIVSTVNLGCKLDLKTIALRARNAYNPKRFAAVIMRIRERTTALIFSSGKMVCTGAKSEEQSLRAARK
21            YARVQVLGFPKFLDFKIQNMVGSQDVKFFIRLEGLVLTHQQFSSYEPELFPGLIYRMKPRIVLLIFVSGKVVLTGA
22            KVRAEIYEAFENIYPIKGRKTT"
23        }
24      },
25      {
26        "protein": {
27          "id": "B",
28          "sequence":
29            "MDQNSLPPYAQGLASPGAMTPGIPFSPMPYGTGLTPQPIQNTNSLSILEEQRRQQQQQQQQQQQQQQQQQQQ
30            QQQQQQQQQQQQQQVAAAVQSTSQATQGTSGQAPQLFHSQTLTAPLPGTTPLYSPMTPTIPATPASE
31            SSGIVPQLQINIVSTVNLGCKLDLKTIALRARNAYNPKRFAAVIMRIRERTTALIFSSGKMVCTGAKSEEQSLRAARK
32            YARVQVLGFPKFLDFKIQNMVGSQDVKFFIRLEGLVLTHQQFSSYEPELFPGLIYRMKPRIVLLIFVSGKVVLTGA
33            KVRAEIYEAFENIYPIKGRKTT"
34        }
35      }
36    ]
37  }
38 }
```

Live Output: COMPLETED

Starting job submission...
Job process started.
shatch: Job submitted with no account set
Submitted alphafold3-cpu.job, jobid: 1193253
shatch: Job submitted with no account set
Submitted alphafold3-gpu.job, jobid: 1193254, depending on successful completion of job 1193253

Job completed successfully.

Tip: Configure your job on the left, then submit to see live output on the right. Drag the center divider to resize panes.

Buttons: Submit Job, Close

Drona Workflow Engine

Step 4: start Workflow

- Live output streaming

The screenshot displays the Drona Workflow Engine interface. On the left, the 'Job Preview' panel shows a JSON configuration for an 'alphafold3-input.json' job. The configuration includes a 'name' of 'promo', 'modelSeeds' (an array of 5 integers), 'dialect' of 'alphafold3', 'version' of 1, and a 'sequences' object with a protein ID 'A' and a long sequence. A 'Notes' box above the code indicates 'Available nodes with gpus: a30=1, h100=3. Selecting h100'. On the right, the 'Live Output' panel shows the job submission process, including job IDs 1193253 and 1193254, and a final 'Job completed successfully' message. A 'COMPLETED' status is shown in a green box. At the bottom right, there are 'Submit Job' and 'Close' buttons. A tip at the bottom left suggests configuring the job on the left and submitting to see live output on the right.

Job Preview

Files: driver.sh alphafold3-cpu.job alphafold3-gpu.job **alphafold3-input.json**

Notes:

- Available nodes with gpus: a30=1, h100=3. Selecting h100

```
1 {
2   {
3     "name": "promo",
4     "modelSeeds": [
5       1,
6       2,
7       3,
8       4,
9       5
10    ],
11    "dialect": "alphafold3",
12    "version": 1,
13    "sequences": [
14      {
15        "protein": {
16          "id": "A",
17          "sequence":
18            "MDQNSLPYYAQLASPGAMTPGIPFSPMPYGTGLTPQPIQNTNSLSILEEQRRQQQQQQQQQQQQQQQQQQQ
19            QQQQQQQQQQQQQQVAAAVQSTSQATQGTSGQAPQLFHSQTLTAPLPGTTPLYSPMTPTMITPATPASE
20            SSGIVPQLQINIVSTVNLGCKLDLKTIALRARNAYNPKRFAAVIMRIREPRTTALIFSSGKMVCTGAKSEEQSRLAARK
21            YARVVQKLGFPKFLDFKIQNMVSGCDVKFPIRLEGLVLTHQFSSYEPELFPGLIYRMKIPRIVLLIFVSGKVLTGA
22            KVRAEIYEAFENIYILKGRKTT"
23        }
24      }
25    ]
26  }
27 }
```

Live Output COMPLETED

Starting job submission...
Job process started.
shatch: Job submitted with no account set
Submitted alphafold3-cpu.job, jobid: 1193253
shatch: Job submitted with no account set
Submitted alphafold3-gpu.job, jobid: 1193254, depending on successful completion of job 1193253

Job completed successfully.

Tip: Configure your job on the left, then submit to see live output on the right. Drag the center divider to resize panes.

Submit Job **Close**

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



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



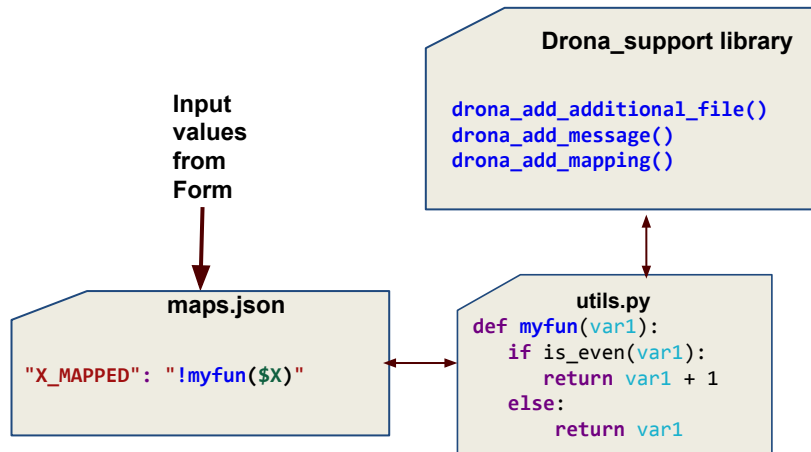
<https://tamu-edu.github.io/dor-hprc-drona-composer/>



Drona: A Framework to Create Workflows

Step 2: Mapping step

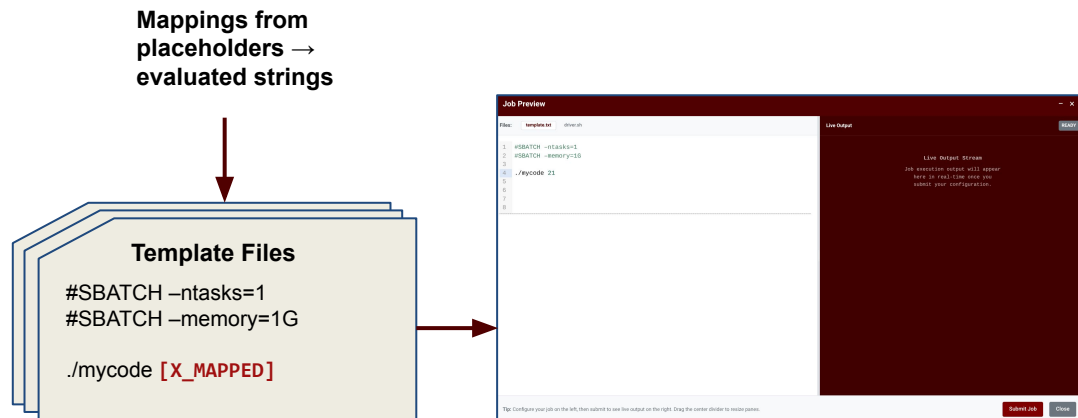
- Drona WFE maps researcher provided input values to placeholder strings using scheme in maps.json
- Format of elements
 - "LHS": "!fun(\$val) text \$val"
 - ! represents function call
 - \$ represents input variable
 - Function calls defined in [utils.py](#)
- Drona provides support library to create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



Drona: A Framework to Create Workflows

Step 3: Replacing placeholders

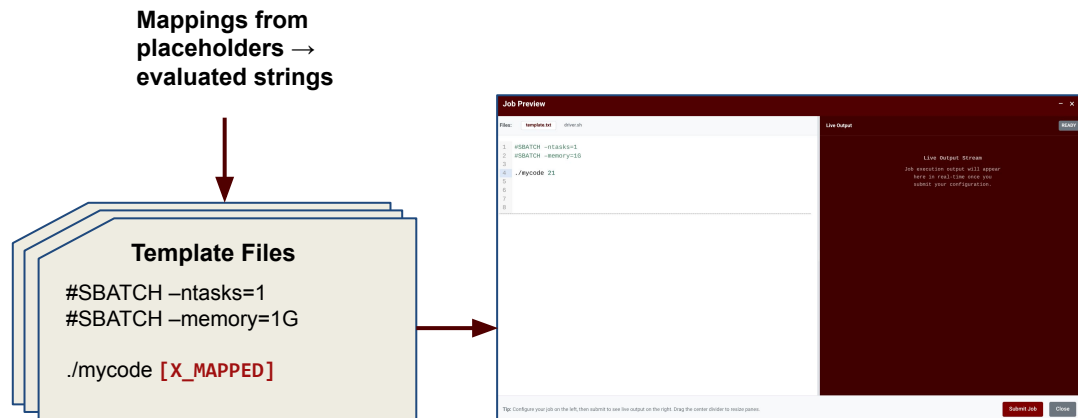
- Drona WFE iterates over all template files to replace placeholders with string values, generated in mapping step.
 - Placeholder are defined as strings enclosed by []
- Generate preview window for researcher to review and make adjustments if needed



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

Job Submission History					
From		10/13/2025	To		11/12/2025
					Filter
ID	Name	Location	Environment	Date	Actions
390893...	myAlphaFold	drona_composer/runs/myAlphaFold	AlphaFold	2025-11-07 14:22:18	Actions
681320...	unnamed	u.mp108705/drona_composer/runs	Generic	2025-10-22 16:41:06	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

Run PopTorch on IPU

This module demonstrates running a pre-configured PyTorch example on the IPU.

It will:

- Set up a PyTorch virtual environment
- Install the PopTorch package (Graphcore's PyTorch extension)
- Run a complete MNIST example on the IPU

Use this module to see PyTorch running on IPU hardware without needing to modify any code.

IPU System Monitor

▲ Hide IPU System Monitor

gc-monitor Partition: p17 [active] has 16 reconfigurable IPU's									
IPU-M	Serial	IPU-M SW[Server version]	ICU FW	Type	ID	IPU#	[Routing]		
10.5.5.1	0019.0002.8222521	2.6.0	1.11.0	2.5.9	M2000	0	3	DNC	
10.5.5.1	0019.0002.8222521	2.6.0	1.11.0	2.5.9	M2000	1	2	DNC	
10.5.5.1	0019.0001.8222521	2.6.0	1.11.0	2.5.9	M2000	2	1	DNC	
10.5.5.1	0019.0001.8222521	2.6.0	1.11.0	2.5.9	M2000	3	0	DNC	
10.5.5.2	0021.0002.8222521	2.6.0	1.11.0	2.5.9	M2000	4	3	DNC	
10.5.5.2	0021.0002.8222521	2.6.0	1.11.0	2.5.9	M2000	5	2	DNC	
10.5.5.2	0021.0001.8222521	2.6.0	1.11.0	2.5.9	M2000	6	1	DNC	
10.5.5.2	0021.0001.8222521	2.6.0	1.11.0	2.5.9	M2000	7	0	DNC	
10.5.5.3	0013.0002.8222521	2.6.0	1.11.0	2.5.9	M2000	8	3	DNC	
10.5.5.3	0013.0002.8222521	2.6.0	1.11.0	2.5.9	M2000	9	2	DNC	
10.5.5.3	0013.0001.8222521	2.6.0	1.11.0	2.5.9	M2000	10	1	DNC	
10.5.5.3	0013.0001.8222521	2.6.0	1.11.0	2.5.9	M2000	11	0	DNC	
10.5.5.4	0016.0002.8222521	2.6.0	1.11.0	2.5.9	M2000	12	3	DNC	
10.5.5.4	0016.0002.8222521	2.6.0	1.11.0	2.5.9	M2000	13	2	DNC	
10.5.5.4	0016.0001.8222521	2.6.0	1.11.0	2.5.9	M2000	14	1	DNC	
10.5.5.4	0016.0001.8222521	2.6.0	1.11.0	2.5.9	M2000	15	0	DNC	
Attached processes in partition p17									
				IPU		Board			
PID	Command	Time	User	ID	Clock	Temp	Temp	Power	
1724420	python	14s	u.sp100705	0	1850MHz	24.7 C	20.7 C	114.3 W	



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

The screenshot shows a web interface for monitoring AlphaFold workflows. At the top, there's a section for 'Environments' with a dropdown set to 'AlphaFold'. Below this, the 'AlphaFold Workflow' section contains three radio buttons: 'New AlphaFold3', 'New AlphaFold2', and 'Current Workflow' (which is selected). To the right of these is a dropdown menu for 'Select workflow' showing 'MyAFWorkflow (id 20250726-115634) 07/25/2025'. Below the workflow selection is a table of jobs with columns: JOB ID, JOB NAME, ELAPSED TIME, STATUS, and CANCEL JOB. The table lists two jobs: '1203999' (AF3-CPU.job) with a status of 'RUNNING' and '1204000' (AF3-GPU.job) with a status of 'PENDING'. Each job has a progress bar and a 'CANCEL JOB' button with a checkmark icon.

JOB ID	JOB NAME	ELAPSED TIME	STATUS	CANCEL JOB
1203999	AF3-CPU.job	18:00/24:00 (hh:mm)	RUNNING	☒
1204000	AF3-GPU.job	00:00/48:00 (hh:mm)	PENDING	☒



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

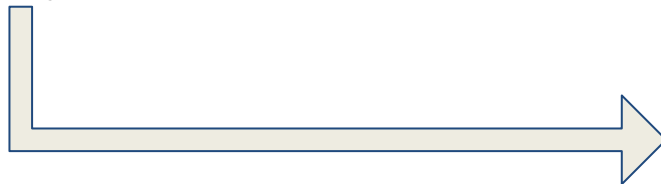


Some non Traditional Examples

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 IPU

● 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

IPU Tutorial

Welcome to the IPU tutorial organized by HPRC

This tutorial consists of a number of modules. Select the next module.

The System status element shows the status of the IPU system. You can find the accompanying documentation in the IPU System Monitor.

Module

- Run PyTorch on IPU
- Setup training materials
- Run Tensorflow on IPU
- Run PyTorch on IPU
- Porting Tensorflow on IPU
- Model Replication
- Model Pipelining

Porting PyTorch to IPU

In this hands-on module, you'll learn how to port a PyTorch model to the IPU. You can use the Playground to experiment with the code.

Using a Fashion-MNIST classification task

- Import and use the PyTorch package
- Implement loss computation for IPU
- Configure IPU options
- Set up PyTorch DataLoaders
- Use PopTorch optimizers
- Create training and inference model
- Manage IPU resources with detach

Check the 'Hints' section if you need guidance.

Job Preview

```
#!/usr/bin/env python3
import torch
import torchvision
import argparse

parser = argparse.ArgumentParser(
    description='Fashion-MNIST classification task'
)
parser.add_argument('--batch-size', type=int, default=64)
parser.add_argument('--test-batch-size', type=int, default=100)
parser.add_argument('--device-id', type=int, default=0)
parser.add_argument('--help', action='help')

args = parser.parse_args()

# Create a PyTorch model
model = torchvision.models.resnet18()

# Create a PyTorch Data Loader
train_loader = torchvision.data.DataLoader(
    dataset=torchvision.data.MNIST('./data', train=True, transform=torchvision.transforms.Compose([
        torchvision.transforms.ToTensor(),
    ])),
    batch_size=args.batch_size,
    shuffle=True,
    num_workers=0,
)

# Create a PyTorch Data Loader
test_loader = torchvision.data.DataLoader(
    dataset=torchvision.data.MNIST('./data', train=False, transform=torchvision.transforms.Compose([
        torchvision.transforms.ToTensor(),
    ])),
    batch_size=args.test_batch_size,
    shuffle=False,
    num_workers=0,
)

# Create a PyTorch optimizer
optimizer = optim.Adam(model.parameters())

# Create a PyTorch trainer
trainer = FashionMNISTTrainer(model, train_loader, test_loader, optimizer, args.device_id)

Hints for PyTorch Porting



Check the boxes below to reveal hints



☒ Hint 1: Import the PyTorch



Show Hint for Step 2 (Loss Computation)



☒ Hint 2: In training mode, use



Show Hint for Step 3 (PopTorch Options)



Show Hint for Step 4 (Training DataLoader)



IPU System Monitor



Hide IPU System Monitor

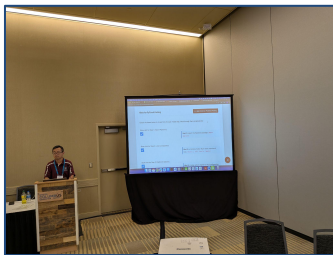


| IPU ID   | Serial            | IPU-M (Hardware version) | GPU ID | Type  | ID | IPU-M (Mounting) |
|----------|-------------------|--------------------------|--------|-------|----|------------------|
| 10.5.5.1 | 0015.0002.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 0  | 0                |
| 10.5.5.1 | 0015.0002.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 1  | 1                |
| 10.5.5.1 | 0015.0001.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 2  | 2                |
| 10.5.5.1 | 0015.0001.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 3  | 3                |
| 10.5.5.2 | 0015.0002.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 4  | 4                |
| 10.5.5.2 | 0015.0002.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 5  | 5                |
| 10.5.5.2 | 0015.0001.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 6  | 6                |
| 10.5.5.2 | 0015.0001.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 7  | 7                |
| 10.5.5.3 | 0015.0002.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 8  | 8                |
| 10.5.5.3 | 0015.0002.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 9  | 9                |
| 10.5.5.3 | 0015.0001.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 10 | 10               |
| 10.5.5.3 | 0015.0001.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 11 | 11               |
| 10.5.5.4 | 0015.0002.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 12 | 12               |
| 10.5.5.4 | 0015.0002.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 13 | 13               |
| 10.5.5.4 | 0015.0001.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 14 | 14               |
| 10.5.5.4 | 0015.0001.0220201 | 2.0.0                    | 1.11.0 | 2.5.0 | 15 | 15               |



| Attached processes on partition 027 |         |      |            | IPU |         |          |       | Board |       |      |       |
|-------------------------------------|---------|------|------------|-----|---------|----------|-------|-------|-------|------|-------|
| PID                                 | Command | Time | User       | ID  | Clock   | Temp     | Power | ID    | Clock | Temp | Power |
| 1058807                             | Python  | 100  | u-mg108780 | 1   | 1058807 | 10.5.5.1 | 25.2  | C     | 102.9 | W    |       |


```



Hugging Face Workflow

Job Name

JobName

Location

Change

/scratch/user/u.ak202494/drona_composer/runs/JobName

Environments

HuggingFaceDrona

▼

+

 **HuggingFace Training & Inference**

Scalable workflows for model inference and fine-tuning with automated resource optimization

Workflow Type

☐ Inference

☒ Fine-tuning

☐ Model Management

☐ Dataset Management

Task Type ?

☐ Text Classification

☐ Text Generation

☐ Question Answering



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

