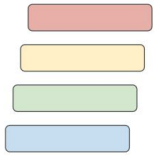
Interactive, Reproducible Notebook Workflows at Scale

Tanu Malik, on behalf of the Floability team:

Douglas Thain, Kevin Lannon, Shaowen Wang
Furquan Baig, Raza Ahmad, Saiful Islam
Shahadat Hossain, Talha Azaz

Python Harmony Workshop 2025
Chicago, IL Sept 2025



floability

The Workflow Advantage: From small scale to large scale

Work with “toy” datasets
Computing on one node

```python
import xarray as xr
import matplotlib.pyplot as plt

# Load a small NetCDF climate dataset (monthly means, 10 years)
ds = xr.tutorial.open_dataset("air_temperature")

# Compute mean temperature anomaly
climatology = ds.air.groupby("time.month").mean("time")
anomaly = ds.air.groupby("time.month") - climatology

# Plot anomaly at one grid point
anomaly.sel(lon=250, lat=60, method="nearest").plot()
plt.show()
```

WS (C, D, E)

Scale to terabytes of raw input
with tightly integrated HPC software

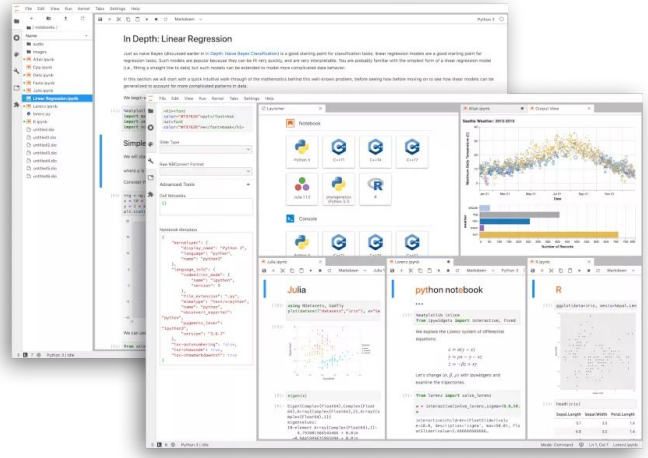
```python
import xarray as xr
from dask_jobqueue import SLURMCluster
from dask.distributed import Client

# Start a Dask cluster on HPC
cluster = SLURMCluster(
    queue='compute',
    cores=32,
    memory='128GB',
    walltime='02:00:00'
)
cluster.scale(10)  # request 10 workers
client = Client(cluster)

# Open large ensemble dataset lazily with Dask
ds = xr.open_mfdataset(
    "/path/to/ensemble/*.nc",
    chunks={"time": 365, "lat": 50, "lon": 50}
)

# Compute climatology and anomaly in parallel
climatology = ds.air.groupby("time.month").mean("time")
anomaly = ds.air.groupby("time.month") - climatology

# Persist results across workers
anomaly = anomaly.persist()

# Example: save output for post-processing
anomaly.to_netcdf("anomaly_ensemble.nc")
```

time

Small scale exploration is increasingly performed via Notebooks

Notebooks

Interactive, Graphical, Personal



?

```python
import xarray as xr
from dask_jobqueue import SLURMCluster
from dask.distributed import Client

# Start a Dask cluster on HPC
cluster = SLURMCluster(
    queue='compute',
    cores=32,
    memory='128GB',
    walltime='02:00:00'
)
cluster.scale(10)  # request 10 workers
client = Client(cluster)

# Open large ensemble dataset lazily with Dask
ds = xr.open_mfdataset(
    "/path/to/ensemble/*.nc",
    chunks={"time": 365, "lat": 50, "lon": 50}
)

# Compute climatology and anomaly in parallel
climatology = ds.air.groupby("time.month").mean("time")
anomaly = ds.air.groupby("time.month") - climatology

# Persist results across workers
anomaly = anomaly.persist()

# Example: save output for post-processing
anomaly.to_netcdf("anomaly_ensemble.nc")
```

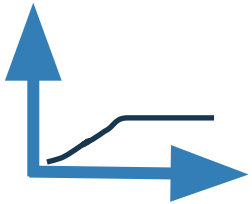
time

The Workflow Advantage: From small scale to large scale

Work with “toy” datasets
Computing on one node

```python
import xarray as xr
import matplotlib.pyplot as plt

# Load a small NetCDF climate dataset (monthly means, 10 years)
ds = xr.tutorial.open_dataset("air_temperature")

# Compute mean temperature anomaly
climatology = ds.air.groupby("time.month").mean("time")
anomaly = ds.air.groupby("time.month") - climatology

# Plot anomaly at one grid point
anomaly.sel(lon=250, lat=60, method="nearest").plot()
plt.show()
```

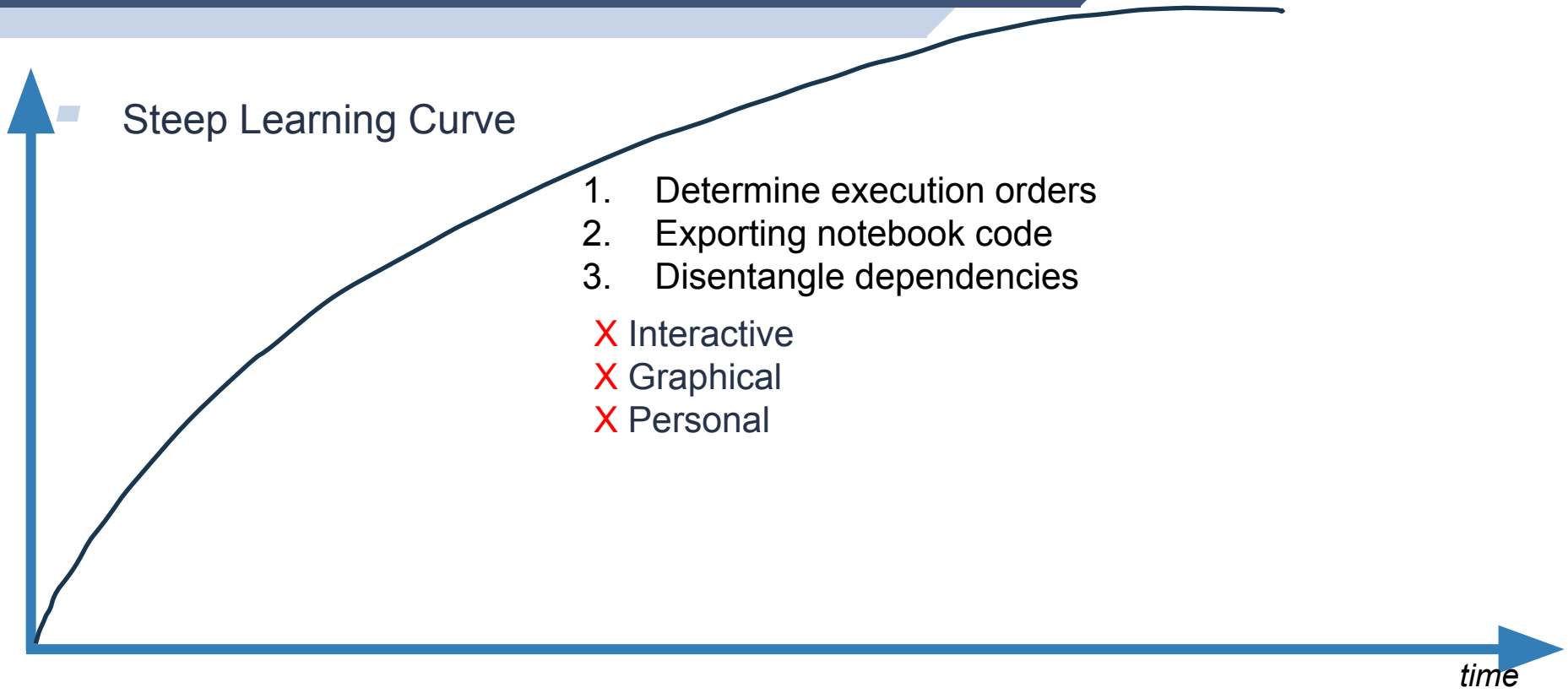
WS (C, D, E)



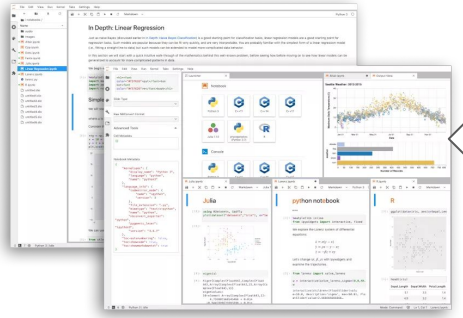
Scale to terabytes of raw input
with tightly integrated HPC software

```python
import xarray as xr
from dask_jobqueue import SLURMCluster
from dask.distributed import Client

# Start a Dask cluster on HPC
cluster = SLURMCluster(
    queue='compute',
    cores=32,
    memory='128GB',
    walltime='02:00:00'
)
cluster.scale(10)  # request 10 workers
client = Client(cluster)

# Open large ensemble dataset lazily with Dask
ds = xr.open_mfdataset(
    "/path/to/ensemble/*.nc",
    chunks={"time": 365, "lat": 50, "lon": 50}
)

# Compute climatology and anomaly in parallel
climatology = ds.air.groupby("time.month").mean("time")
anomaly = ds.air.groupby("time.month") - climatology

# Persist results across workers
anomaly = anomaly.persist()

# Example: save output for post-processing
anomaly.to_netcdf("anomaly_ensemble.nc")
```

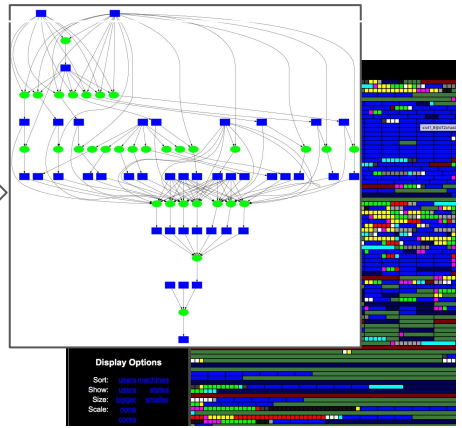
The Notebook Scalability Problem



The Floability Project (floability.github.io) aims to make notebooks interactive, reproducible and scalable, enabling portable deployment of complex scientific workflows across varied cyberinfrastructure.



floability



CSSI Frameworks: From Notebook to Workflow and Back Again

NSF

UNIVERSITY OF NOTRE DAME

Mizzou
University of Missouri

ILLINOIS

Challenges of an Integrated Infrastructure

- Portability Challenge
- Resource Provisioning Challenge
- Concurrency Challenges
- Interactivity-Scalability Challenge
- Data Challenges

The Reproducibility Challenge

You cannot deploy a notebook workflow effectively without all the supporting environment.

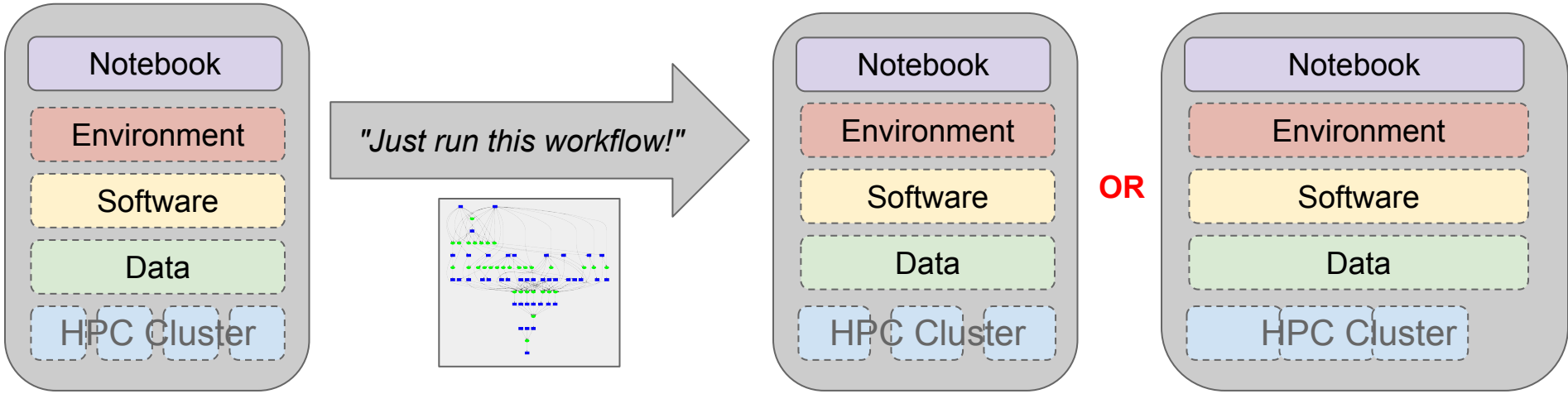
Notebook Runs at Facility A

Notebook Fails at Facility B

The Provisioning Challenge-II

User must estimate and encapsulate "Whole Workflow" Requirements

Find more ways to say NO prior to execution: available disk space, worker capacity, network utilization, cluster architecture…



Notebook Runs at Facility A

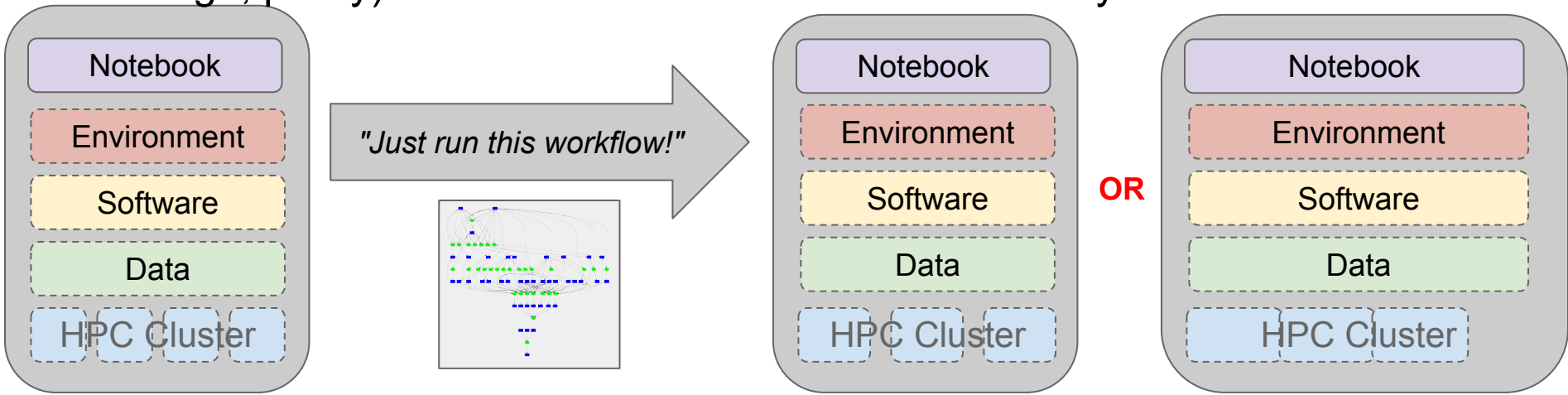
Node Type I at Facility B

Node Type II at Facility B

The Provisioning Challenge-III

Capture Site Specializations

Every facility is a snowflake. Can we capture functional differences (network, storage, policy) between facilities in a constructive way?



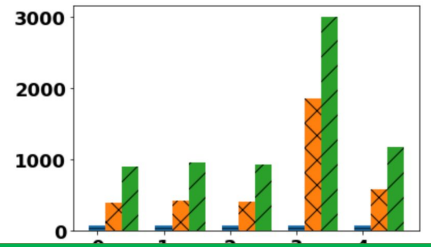
Notebook Runs at **Facility A**

Node Type I at **Facility B**

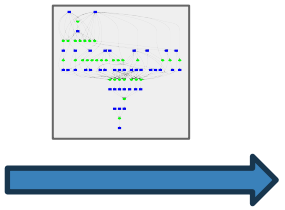
Node Type II at Facility B

The Concurrency Challenge

```
[ ]: plt.subplots(figsize =(12, 8))
     x = np.arange(len(df2.index))

[16]: plt.bar(x, df2['Size 2'], width=bar_width, hatch='-')
      plt.bar(x+(1*bar_width), df2['Size 1'], width=bar_width, hatch='x')
      plt.bar(x+(2*bar_width), df2['Size 3'], width=bar_width, hatch='/')

[16]: <BarContainer object of 5 artists>
```




```
[ ]: plt.ylabel('Container Size (MBs)', fontsize=18, fontweight='bold')
     plt.xticks([r + bar_width for r in range(len(x))], df2.index)
     plt.legend(labels=labels)

[18]: plt.savefig('export_size_containers_all.png', dpi=500)
      df2['Size 2'].head(3)

[18]: Use Case
      FlowFromSnow    72.7
      HAND            72.7
      NatGas          72.7
      Name: Size 2, dtype: float64
      <Figure size 432x288 with 0 Axes>
```

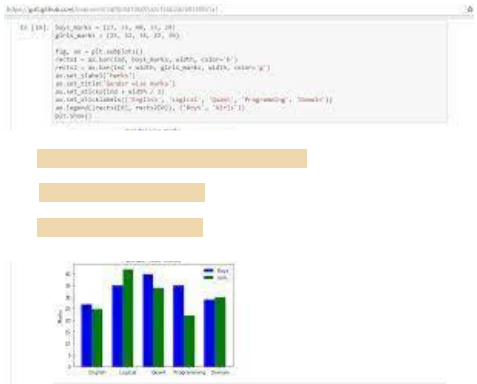


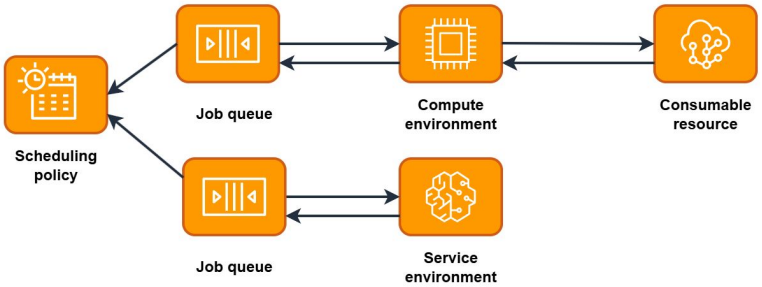
HPC Cluster

- Instead of writing in a parallel framework, can we infer independent tasks from the cell structure itself?

The Interactive-Scalability Challenge

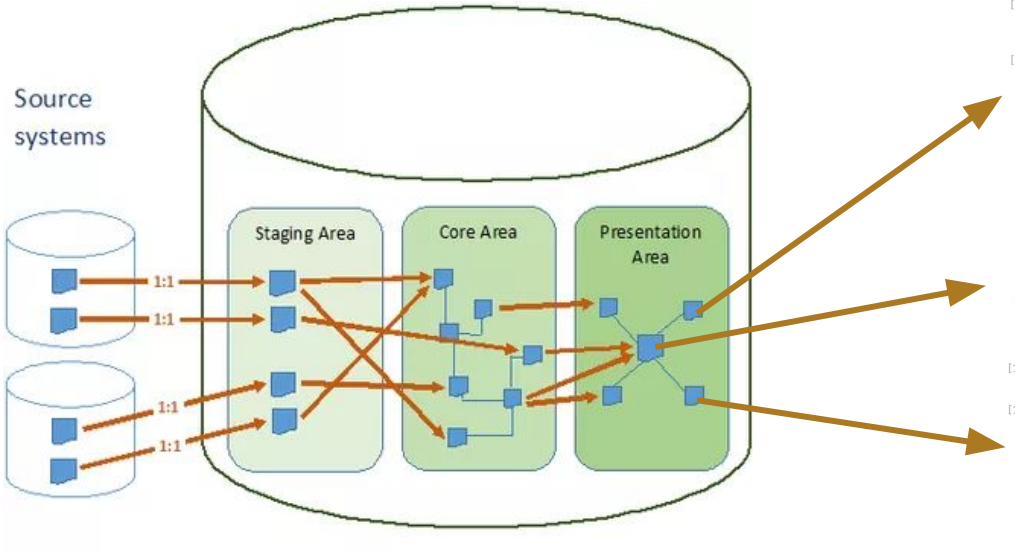
- Interactive and batch allocations are interdependent—neither is effective in isolation. Yet both must operate within limited capacity, making coordinated management essential.



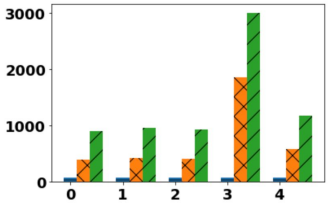


The Data Challenge

- Should notebooks encapsulate data or not for scalable computation?

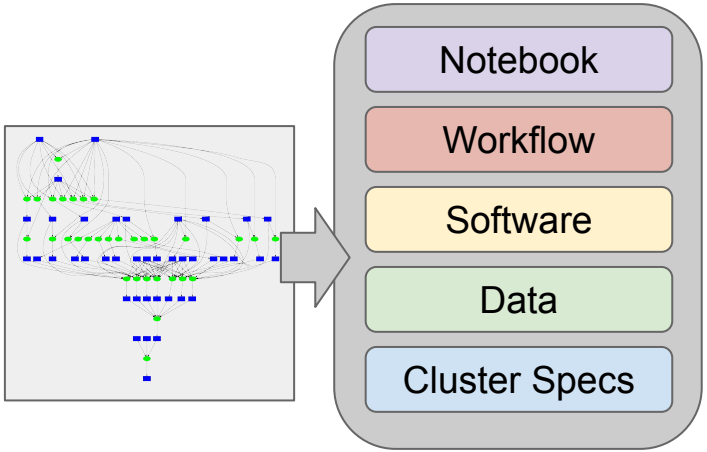



```
[ ]: plt.subplots(figsize =(12, 8))
     x = np.arange(len(df2.index))

[16]: plt.bar(x, df2['Size 2'], width=bar_width, hatch='-')
      plt.bar(x+(1*bar_width), df2['Size 1'], width=bar_width, hatch='x')
      plt.bar(x+(2*bar_width), df2['Size 3'], width=bar_width, hatch='/')

[16]: <BarContainer object of 5 artists>
```



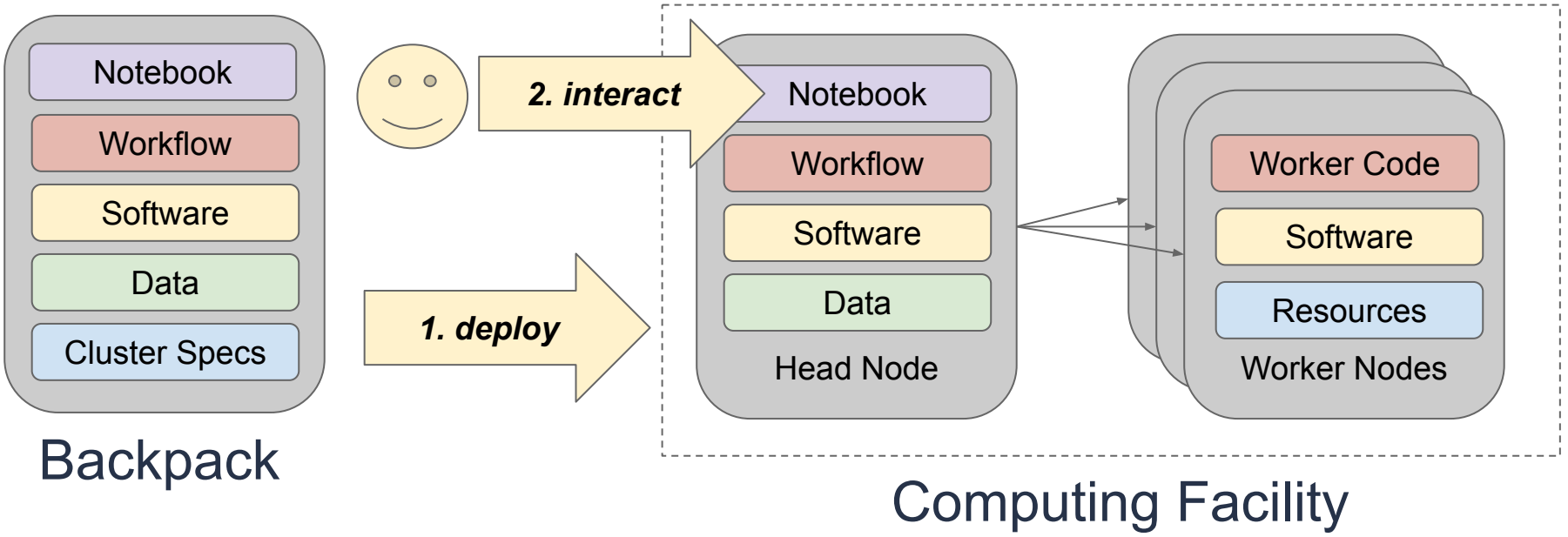

```
[ ]: plt.ylabel('Container Size (MBs)', fontsize=18, fontweight='bold')
     plt.xticks([r + bar_width for r in range(len(x))], df2.index)
     plt.legend(labels=labels)

[18]: plt.savefig('export_size_containers_all.png', dpi=500)
      df2['Size 2'].head(3)

[18]: Use Case
      FlowFromSnow    72.7
      HAND            72.7
      NatGas          72.7
      Name: Size 2, dtype: float64
      <Figure size 432x288 with 0 Axes>
```

A **backpack** contains everything needed to deploy a notebook workflow at large scale.



A backpack encapsulates a notebook workflow with structured specifications of all its dependencies—data, software,and computational resources—needed for seamless execution across diverse environments.

Floability deploys a **backpack** into a facility.



So What's Actually in a Backpack? (V1)

https://github.com/floability/floability-examples/tree/main/cms-physics-dv5

```
➔ workflow
   ◆ cms-physics-dv5.ipynb
➔ software
   ◆ environment.yml
   ◆ worker-environment.yml
➔ data
   ◆ data.yml
   ◆ samples/qcd/.../nano_mc2017_11.root
   ◆ samples/diboson/.../nanomc2017_6.root
➔ compute
   ◆ compute.yml
```

So What's Actually in a Backpack? (V1)

https://github.com/floability/floability-examples/tree/main/cms-physics-dv5

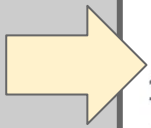
- **workflow**
  - **cms-physics-dv5.ipynb**
- software
  - environment.yml
  - worker-environment.yml
- data
  - data.yml
  - samples/qcd/.../nano_mc2017_11
  - samples/diboson/.../nanomc2017
- compute
  - compute.yml

```
def analysis(events):
    warnings.filterwarnings("ignore", module="coffea.*"

    dataset = events.metadata["dataset"]
    events["PFCands", "pt"] = events.PFCands.pt * event

    cut_to_fix_softdrop = ak.num(events.FatJet.constitu
    events = events[ak.all(cut_to_fix_softdrop, axis=1)

    trigger = ak.zeros_like(ak.firsts(events.FatJet.pt)
    for t in triggers["2017"]:
        if t in events.HLT.fields:
            trigger = trigger | events.HLT[t]
    trigger = ak.fill_none(trigger, False)

    events["FatJet", "num_fatjets"] = ak.num(events.Fat

    goodmuon = (
        (events.Muon.pt > 10)
        & (abs(events.Muon.eta) < 2.4)
        & (events.Muon.pfRelIso04_all < 0.25)
        & events.Muon.looseId
    )

    nmuons = ak.sum(goodmuon, axis=1)
```

So What's Actually in a Backpack?

https://github.com/floability/floability-examples/tree/main/cms-physics-dv5

- workflow
  - cms-physics-dv5.ipynb
- software
  - **environment.yml**
  - worker-environment.yml
- data
  - data.yml
  - samples/qcd/.../nano_mc2017_11.
  - samples/diboson/.../nanomc2017_
- compute
  - compute.yml

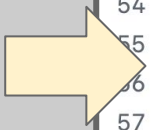


```yaml
name: my_mdv5_env

channels:
  - conda-forge

dependencies:
  - python=3.12
  - cloudpickle
  - ndcctools
  - coffea<2024.4.2
  - dask<2024.5.2
  - dask-awkward<2024.5.0
  - dask-core<2024.5.0
  - dask-histogram<2024.5.0
  - fsspec
  - pip:
    - fastjet<3.4.2.2
```

So What's Actually in a Backpack? (V1)

https://github.com/floability/floability-examples/tree/main/cms-physics-dv5

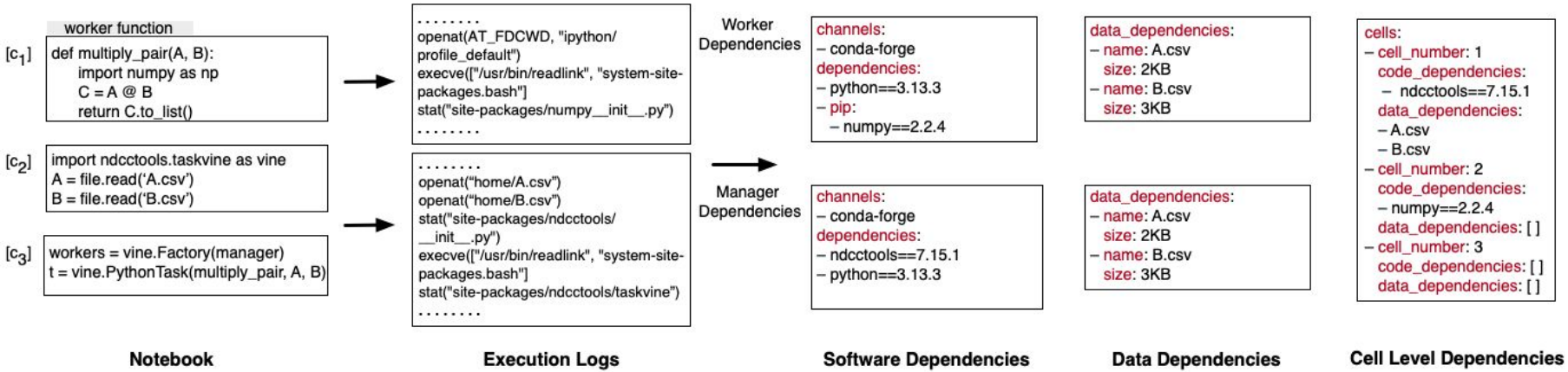
- workflow
  - cms-physics-dv5.ipynb
- software
  - environment.yml
  - **worker-environment.yml**
- data
  - data.yml
  - samples/qcd/.../nano_mc2017_11.root
  - samples/diboson/.../nanomc2017_6.rc
- compute
  - compute.yml



```
41    - conda-forge::pytz==2025.2
42    - conda-forge::rich==14.0.0
43    - conda-forge::scipy==1.15.2
44    - conda-forge::setuptools==78.1.0
45    - conda-forge::tblib==3.1.0
46    - conda-forge::toolz==1.0.0
47    - conda-forge::tqdm==4.67.1
48    - conda-forge::uhi==0.5.0
49    - conda-forge::uproot==5.6.0
50    - conda-forge::wheel==0.45.1
51    - conda-forge::xxhash==0.8.3
52    - conda-forge::yaml==0.2.5
53    - conda-forge::zstandard==0.23.0
54    - pip==25.0.1
55    - python==3.12.10
56    - pip:
57      - debugpy==1.8.14
58      - fastjet==3.4.2.1
59      - ipython-pygments-lexers==1.1.1
60      - ipython==9.1.0
61      - jsonschema==4.23.0
62      - jupyter-client==8.6.3
63      - jupyter-core==5.7.2
64      - matplotlib-inline==0.1.7
65      - nbformat==5.10.4
66      - vector==1.6.1
```

How do you find software/data deps?

Use the **SciUnit** technology to instrument a running notebook to observe the Python dependencies of the manager/worker.



Fils, Gabriel, Zhihao Yuan, and Tanu Malik. "Sciunits: Reusable research objects."
In *2017 IEEE 13th International Conference on e-Science (e-Science)*, pp. 374-383. IEEE, 2017.

So What's Actually in a Backpack? (V1)

https://github.com/floability/floability-examples/tree/main/cms-physics-dv5

- workflow
  - cms-physics-
- software
  - environment.
  - worker-envir
- **data**
  - **data.yml**
  - **samples/qcd/.../nano_mc2017_11.root**
  - **samples/diboson/.../nanomc2017_6.root**
- compute
  - compute.yml

```yaml
data:
  - name: "diboson_zz_6"
    source_type: "backpack"
    source: "data/samples/diboson/zz/nano_mc2017_6.root"
    target_location: "data/samples/diboson/zz/nano_mc2017_6.root"

  - name: "diboson_zz_9"
    source_type: "backpack"
    source: "data/samples/diboson/zz/nano_mc2017_9.root"
    target_location: "data/samples/diboson/zz/nano_mc2017_9.root"
```

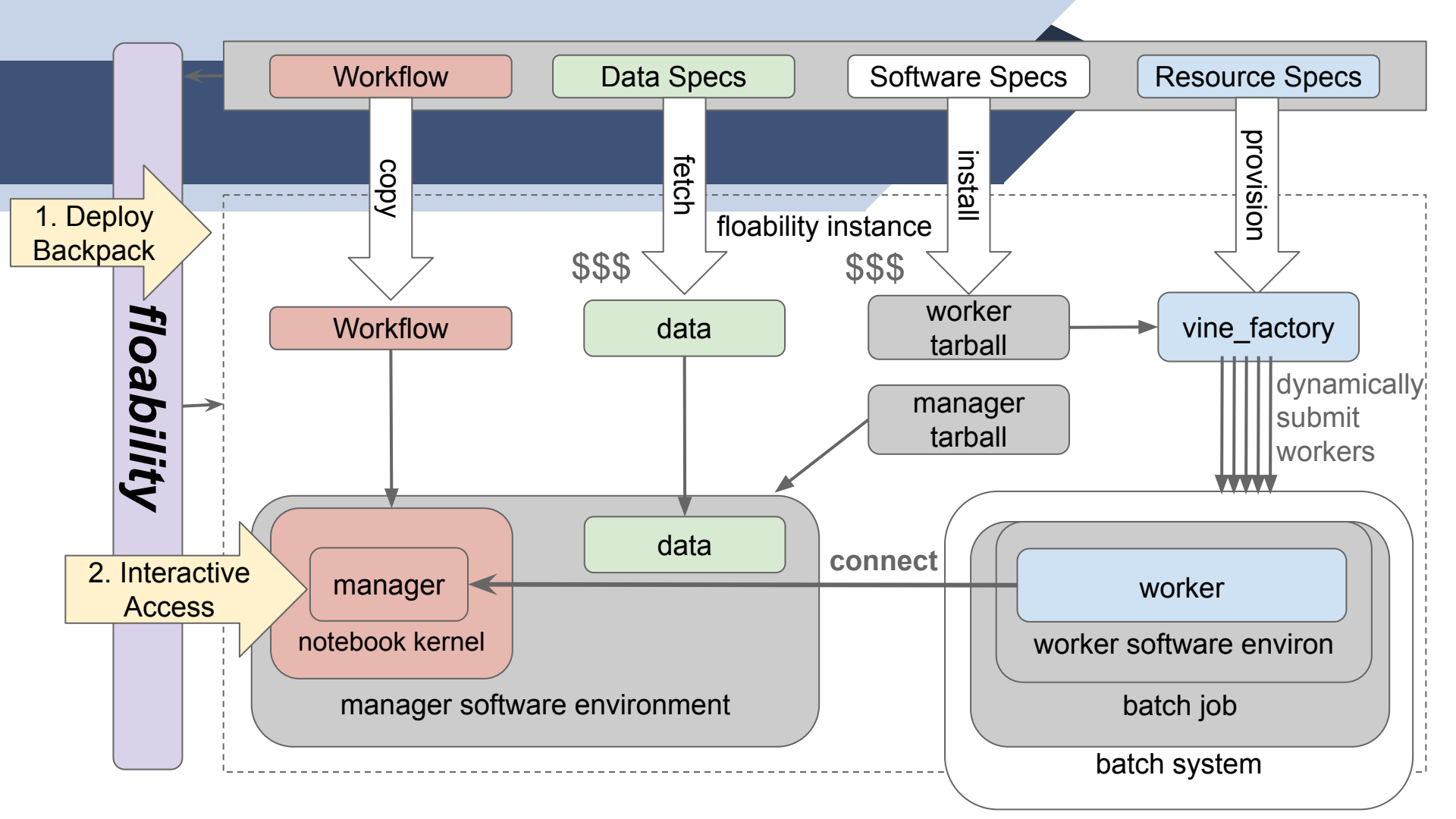
Opportunity to connect Pelican!

So What's Actually in a Backpack? (V1)

https://github.com/floability/floability-examples/tree/main/cms-physics-dv5

- workflow
  - cms-physics-dv5.ipynb
- software
  - environment.yml
  - worker-environment.yml
- data
  - data.yml
  - samples/qcd/.../nano_mc2017_
  - samples/diboson/.../nanomc20
- **compute**
  - **compute.yml**

```
1    vine_factory_config:
2      min-workers: 2
3      max-workers: 400
4      cores: 4
5      memory: 1024
6      disk: 2000
```


Workflow
Data Specs
Software Specs
Resource Specs
copy
fetch
install
provision
1. Deploy Backpack
floability
floability instance
$$$
$$$
Workflow
data
worker tarball
vine_factory
manager tarball
dynamically submit workers
2. Interactive Access
manager
notebook kernel
data
connect
worker
worker software environ
manager software environment
batch job
batch system

Deployment is a Work in Progress

Workflow	ND CRC	Purdue Anvil	UT Stampede3	AWS Cluster	OSPool
DV5	✔	✔	✔	✔	
DConv	✔	✔	✔	✔	
CTrend	✔	✔	✔	✔	
CESM Ocean Heat	✔	✔	✔	✔	
Montage	✔				

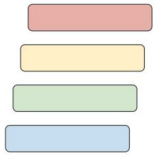
https://github.com/floability/floability-examples

Deployment to Multiple Sites: floability run

Workflow	Metric	ND CRC		Purdue Anvil		UT Stampede3		AWS Cluster	
		First	Repeat	First	Repeat	First	Repeat	First	Repeat
DConv	Total Runtime	292	136	744	255	684	330	216	88
	Env Creation (M + W)	132	0	516	0	307	0	107	0
	Env Extraction	26	29	36	39	116	116	26	26
	Notebook Execution	116	98	174	198	229	171	66	51
CTrend	Total Runtime	265	107	1052	304	405	201	302	216
	Env Creation (M + W)	159	0	763	0	217	0	133	0
	Env Extraction	30	29	44	44	97	90	30	32
	Notebook Execution	57	64	225	240	75	101	127	172
DV5	Total Runtime	370	70	1212	281	827	218	500	89
	Env Creation (M + W)	303	0	973	0	598	0	242	0
	Env Extraction	28	29	43	44	105	83	29	37
	Notebook Execution	25	28	172	217	103	115	215	40

eScience 2025 Talk by Saiful Islam

- Backpacks for Notebooks: Enabling Containerized Notebook Workflows in Distributed Environments
  - **Md Saiful Islam**, Talha Azaz, Raza Ahmad, A S M Shahadat Hossain, Furqan Baig, Shaowen Wang, Kevin Lannon, Tanu Malik, Douglas Thain
- Wednesday morning session



floability

floability.github.io

Overall Project Status:

- Year two of a four year translational project.
- MVP of the Floability tool is published via Conda:
  - `conda install -c conda-forge floability`
- Currently gathering and testing applications at multiple sites.
  - OSPool – Need to be able to connect from worker to manager.
- Looking for applications and (brave) initial users.

floability

floability.github.io

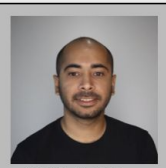
```
conda install -c conda-forge floability
```

Prof. Douglas Thain
Principal Investigator
University of Notre Dame

Prof. Tanu Malik
co-PI
University of Missouri

Prof. Kevin Lannon
co-PI
University of Notre Dame

Prof. Shaowen Wang
co-PI
University of Illinois



Md Saiful Islam
Ph.D. Student
University of Notre Dame

Talha Azaz
MS. Student
DePaul University

Ben Tovar
Research Software Engineer
University of Notre Dame

A S M Shahadat Hossain
Ph.D. Student
University of Missouri

Raza Ahmad
Ph.D. Student
DePaul University

Dr. Furqan Baig
Research Programmer
University of Illinois Urbana Champaign

This work was supported in part by NSF grant 2411436 CSSI Frameworks: From Notebook to Workflow and Back Again