IRIS-HEP

pyHEP.dev | Sep 2026

coffea-workflow

Split the dataset, switch facilities, cache partial result, retry your analysis — don't rewrite anything.

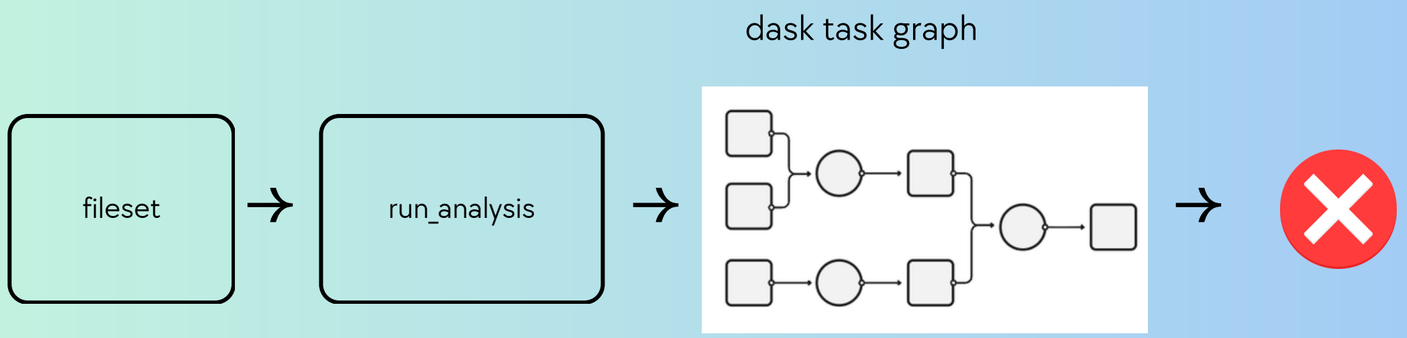
goloborodko.yana2016@gmail.com

coffea-workflow

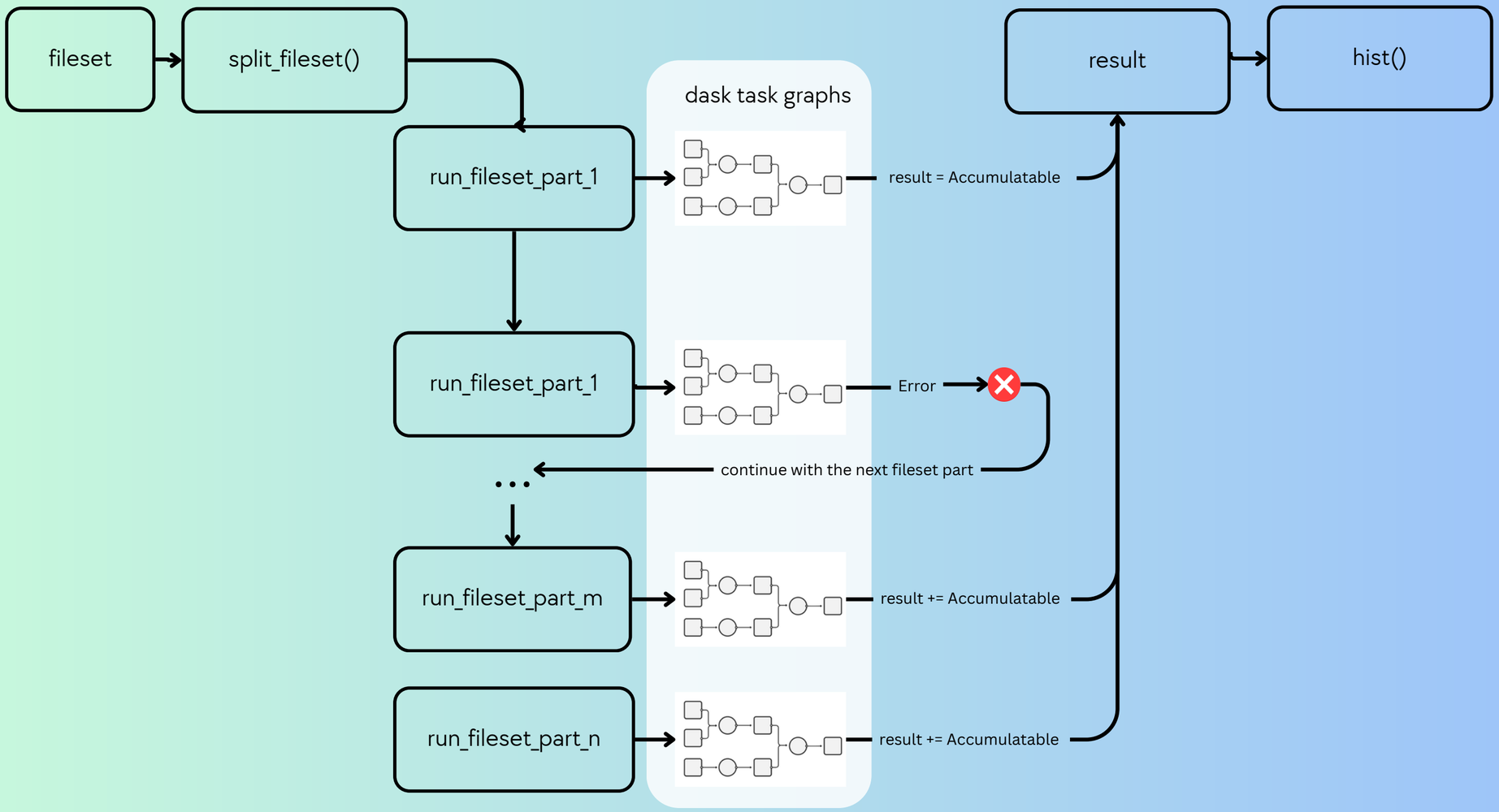
Common DAG

Fileset ⟶ **(Preprocessing)** ⟶ **Analysis** ⟶ **Plotting**

Common workflow & problem



Proposed solution



coffea-workflow v0.2.0 - now on PyPI

what's inside?

- ✔️ **pure-Python workflow API**
- ✔️ **fileset split, partial results & retry**
- ✔️ **facility factories**
  lxplus · coffea-casa · local
- ✔️ **execution knobs**
  executor type · seq/parallel
- ✔️ **caching**

```
$ pip install coffea-workflow
```

- ***requires coffea ≥ 2026.7.0*** *(also released):*
  - *Runner returns* ***Ok / Err*** *→ partial results survive*

2 layers inside

what the user sees (**external artifacts**):

Fileset → **Preprocessed** → **Analysis** → **Plotting**

changes the granularity of
chunking from files to events

what the framework does internally (invisible to user, **internal artifacts**):

Chunking → ChunkAnalysis × N → merge

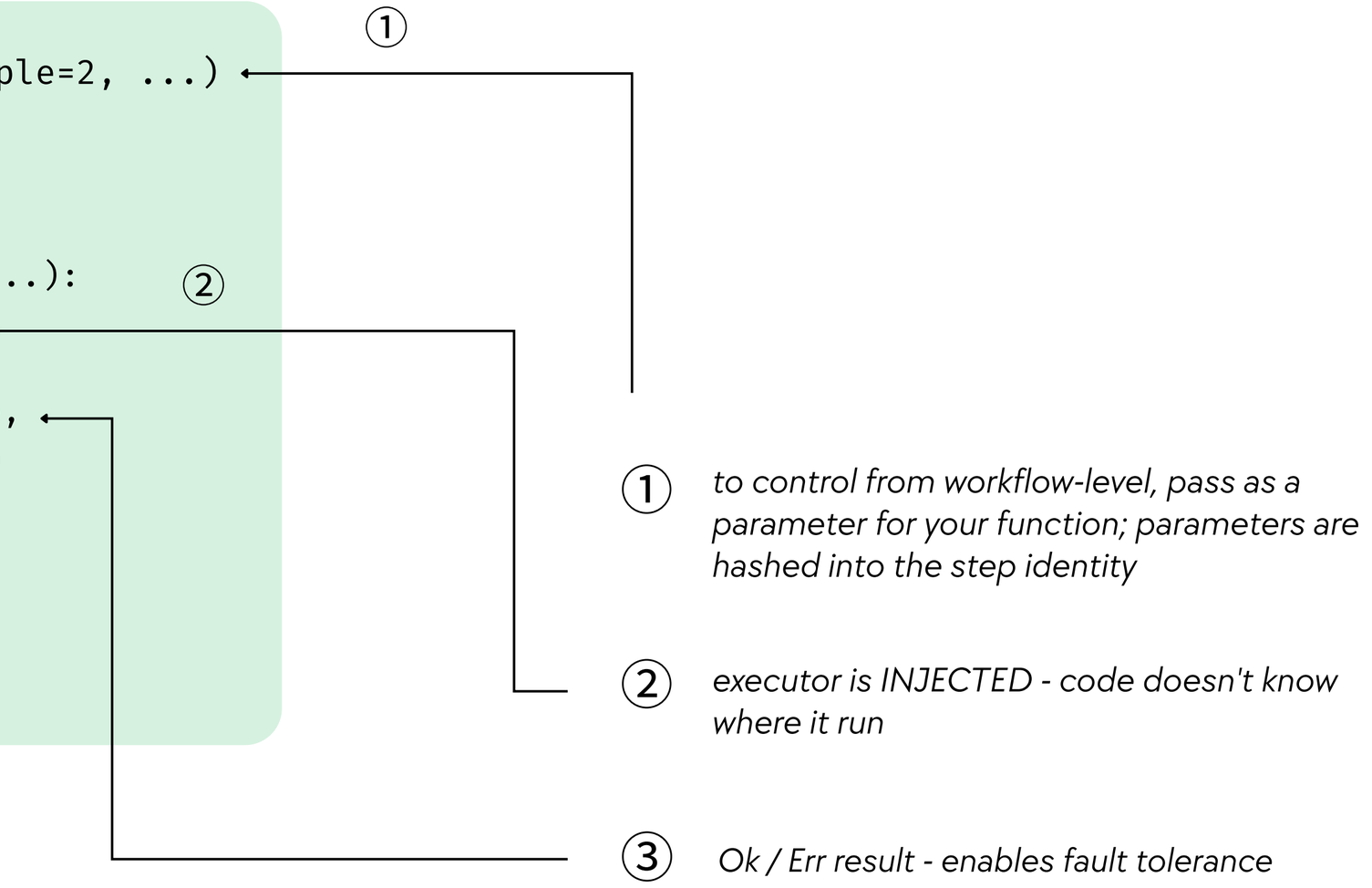
↕

[skip failures, track them]

what coffea-workflow needs from your analysis?

```
def get_fileset(n_files_max_per_sample=2, ...)



def run_analysis(fileset, executor, ...):
    run = Runner(executor=executor,
                 skipbadfiles=True,
                 use_result_type=True,
                 ...)


def plotting_1(result, ...)
```

enough to cover AGC analysis



Step & RunConfig

One function → one step of the workflow

```
Step(
    name="Analysis_ttbar",
    step_type=Analysis,
    builder=run_analysis,
    builder_params={
          "use_inference"= False},
    input=...,
    output=...,
    )
```

cache label & results key

pre-implemented behaviour

your function

workflow-level knobs

HOW TO SPLIT the fileset, what part of fileset to process

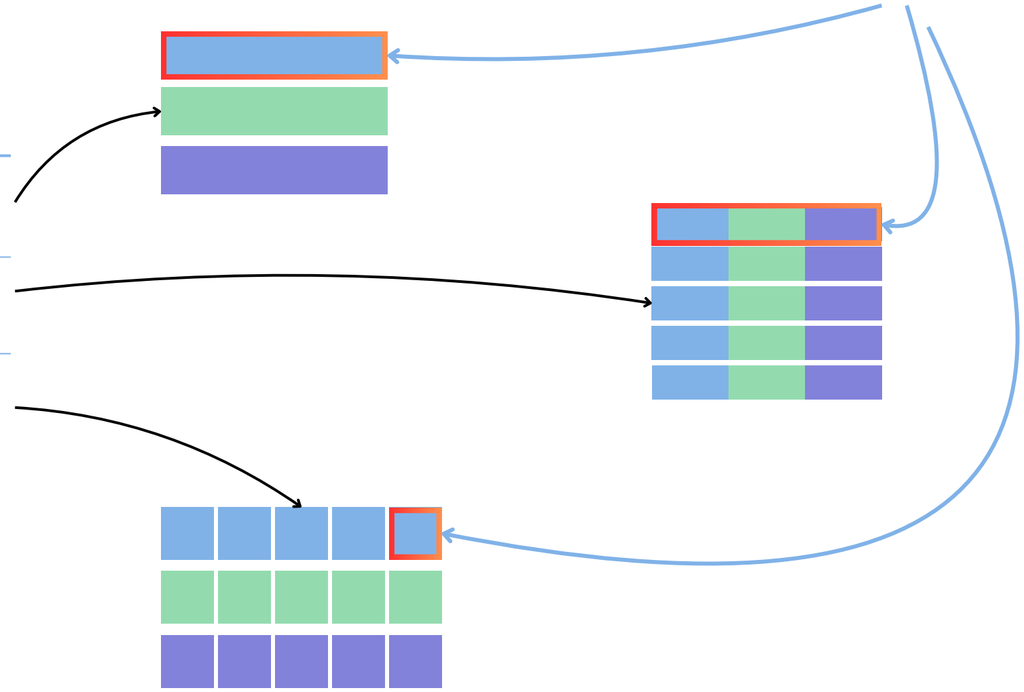
WHERE & HOW to run

```
RunConfig(

    strategy="by_dataset",
    percentage=50,
    datasets=[..],
    chunk_fraction=0.3,

    facility=facilities.coffea_casa,
    executor_config=(
       executor_type="DaskExecutor",
       parallel_chunks=True,
       )
    )
```

RunConfig knobs: split strategies

different test run variations without any additional code; control via chunk_fraction

Strategy	What it does	Example
`strategy="by_dataset"`	One chunk per dataset	3 datasets → 3 chunks
`percentage=20`	20% of each dataset per chunk	5 mixed chunks
`strategy + percentage`	Both combined	3 datasets × 5 = 15 chunks



*`datasets=[...]` - apply strategy only to selected datasets

RunConfig knobs: facilities

FacilityFactory protocol:

`preflight()`
check + fail early

`build()`
cluster + executor

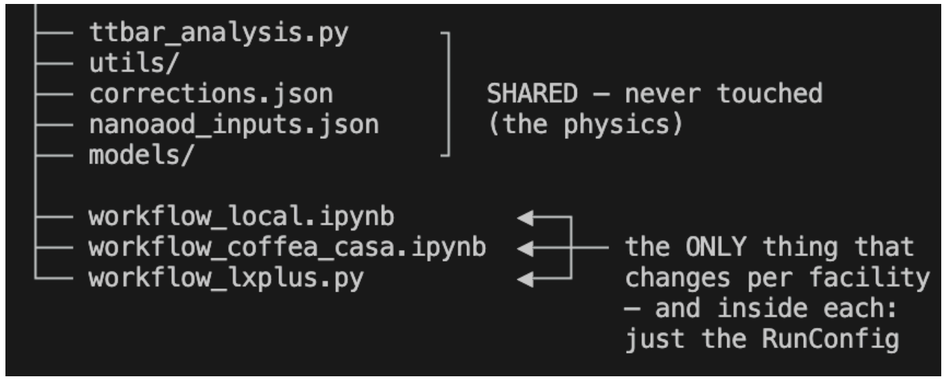
`close()`
tear down

	LOCAL	COFFEA-CASA	LXPLUS
you set:	workers	packages, files	image, queue, N workers
covered:	Futures/Iterative	executor initiation, Dask client, worker provisioning	HTCondor cluster, Apptainer creation guidence and assistance, VOMS proxy

executors: Iterative · Futures · Dask

Demo: same workflow, three facilities

Analysis Grand Challenge




```
RunConfig(

    strategy="by_dataset",
    percentage=50,
    datasets=[..],
    chunk_fraction=0.3,

    cache_dir=".cache_local",

    facility=facilities.local,
    executor_config=(
       executor_type="FuturesExecutor",
       )
    )
```

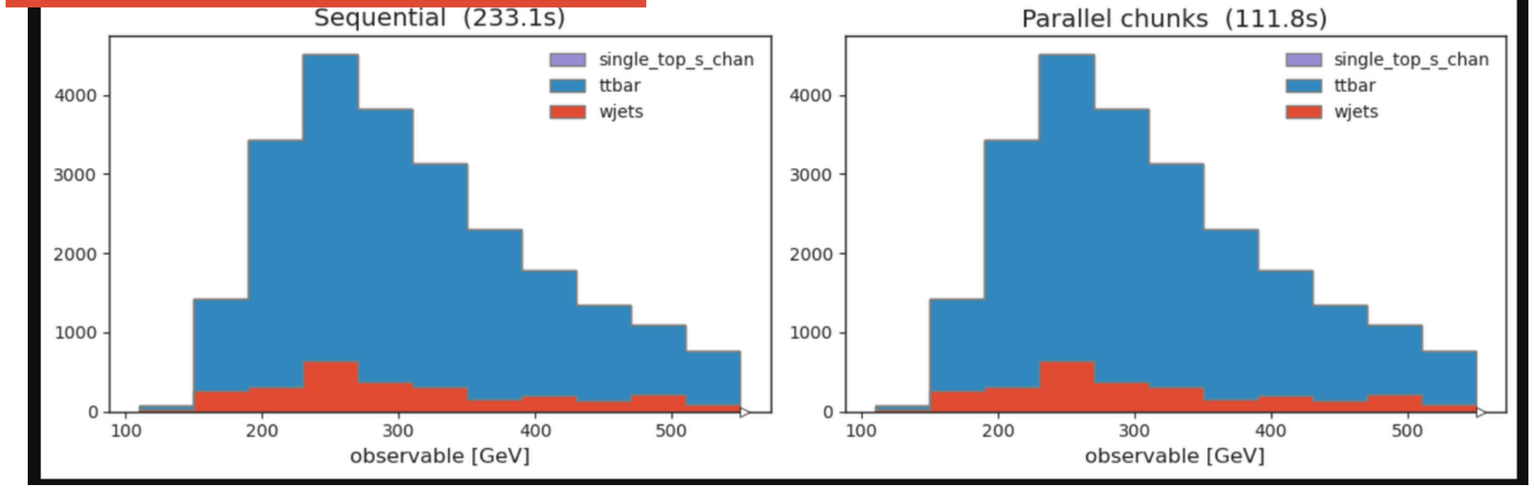
```
RunConfig(

    strategy="by_dataset",
    percentage=50,
    datasets=[..],
    chunk_fraction=0.3,

    cache_dir=".cache_coffea_casa",

    facility=facilities.CoffeaCasaFactory(...),
    executor_config=(
       executor_type="DaskExecutor",
       parallel_chunks=True,
       )
    )
```

```
RunConfig(

    strategy="by_dataset",
    percentage=50,
    datasets=[..],
    chunk_fraction=0.3,

    cache_dir=".cache_lxplus",

    facility=facilities.LxplusFactory(...),
    executor_config=(
       executor_type="DaskExecutor",
       parallel_chunks=True,
       )
    )
```

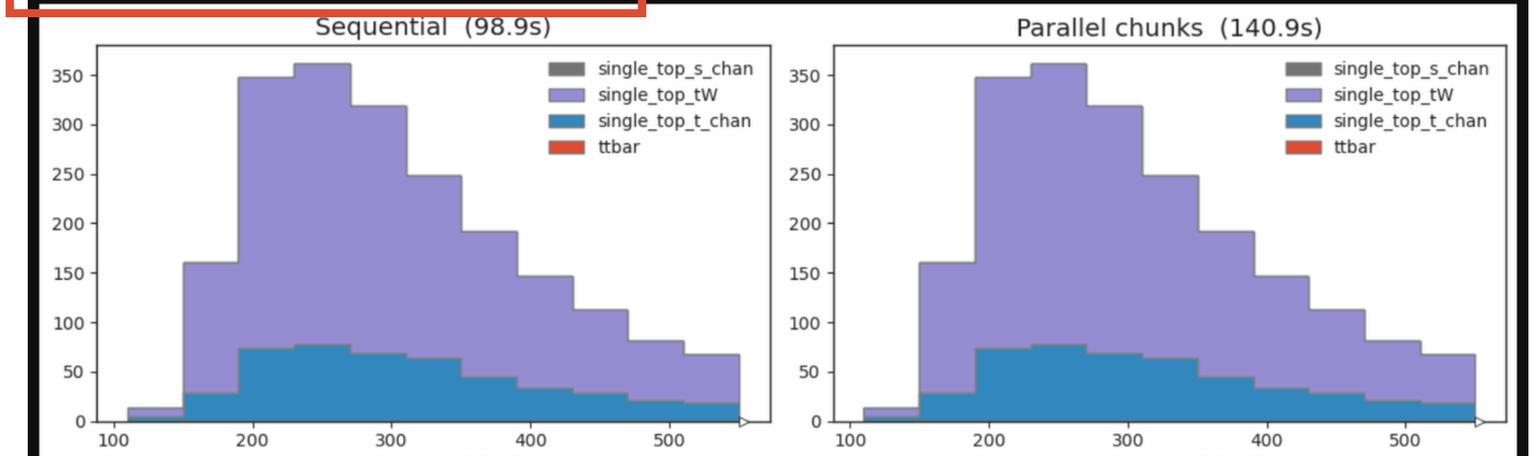
demo facilities

○ RunConfig knobs: sequential vs parallel dispatch

```
========================================
Sequential wall time :  233.1s
Parallel wall time   :  111.8s
Speed-up             : 2.08x
========================================
```




```
========================================
Sequential wall time :   98.9s
Parallel wall time   :  140.9s
Speed-up             : 0.70x
========================================
```



SEQUENTIAL (default)

one subset at a time, ALL workers work on one subset

s1 ▶ [W1 W2 W3 W4]

s2 ▶ [W1 W2 W3 W4]

s3 ▶ [W1 W2 W3 W4]

PARALLEL

all subsets at once, ONE worker per subset, no idle gaps between subsets

s1 ▶ W1 s2 ▶ W2

s3 ▶ W3 s4 ▶ W4

s5 ▶ W1 ...

Interoperability: one DAG, two backends

FOR NOW

run(workflow, config) executes the DAG in-process

IDEA

snakemake_render(workflow, config, out_dir) compiles the same DAG into a runnable Snakefile, hands scheduling to Snakemake

coffea-workflow owns:

- what a Fileset/Preprocessed/Analysis/Plotting is;
- the split strategies;
- coffea Runner/executor construction;
- the Ok/Err accumulatable-merge and partial-result;
- facility environment (VOMS/x509, Apptainer image, worker packages/files)

Snakemake owns:

- dependency resolution and scheduling;
- the checkpoint/dynamic-fan-out over chunks;
- per-rule retries, resources (mem_mb, runtime), timeouts;
- cluster submission via executor plugins (HTCondor, Slurm, Kubernetes, cloud);
- incremental re-runs

Links:

- **repository**: https://github.com/CoffeaTeam/coffea-workflow/blob/main/README.md
- **PyPI**: https://pypi.org/project/coffea-workflow/

THANK YOU!