Toward Ten-Minute Turnaround with TopEFT and TaskVine (Work Queue) The View from Notre Dame

Ben Tovar

ND CMS Cooperative Computing Lab

May 4, 2023

UNIVERSITY OF NOTRE DAME

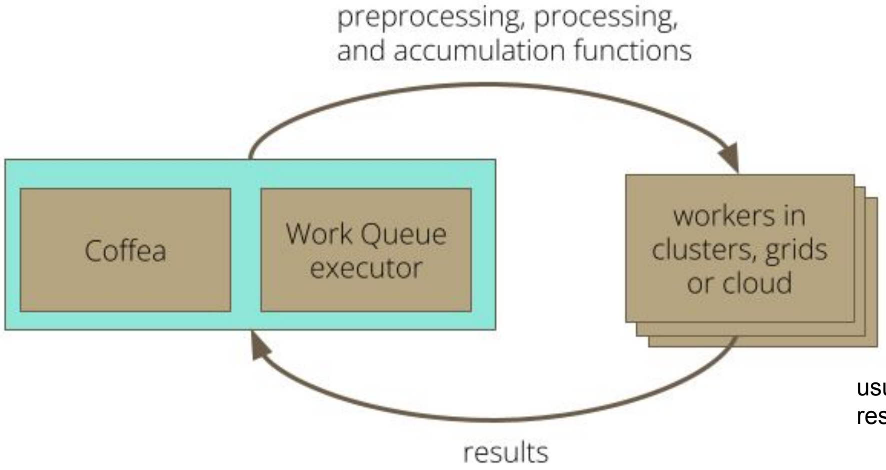
CCTools

Outline

1. Current results with Work Queue Executor and its bottlenecks
2. TaskVine and the problems it solves
3. Connections of TaskVine with AGC

Simplified View

Work Queue is an execution engine part of CCTools, a suite of programs by the ND Cooperative Computing Lab (distribute research computing).


preprocessing, processing, and accumulation functions
Coffea
Work Queue executor
workers in clusters, grids or cloud
results

usually opportunistic resources

Analysis: TopEFT Framework

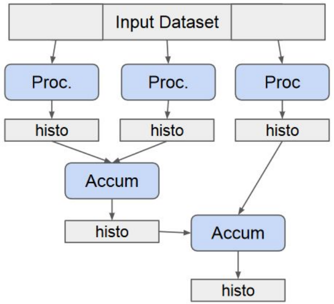
- Use TopEFT (topCoffea) analysis to test current framework
  - Full Run 2 analysis (~150/fb, HL-LHC~3000/fb)
- Designed to analyze CMS data in order to search for new physics using the framework of Effective Field Theory (EFT) [CMS-PAS-22-006]()

topEFT

topCoffea

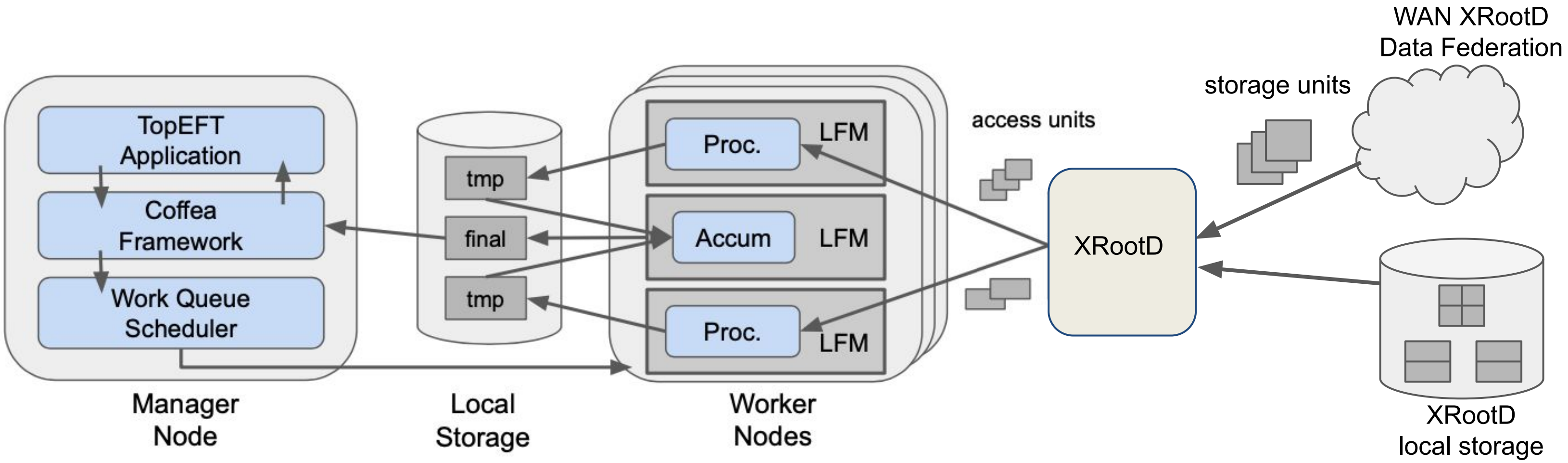
Coffea

Work Queue



UNIVERSITY OF NOTRE DAME

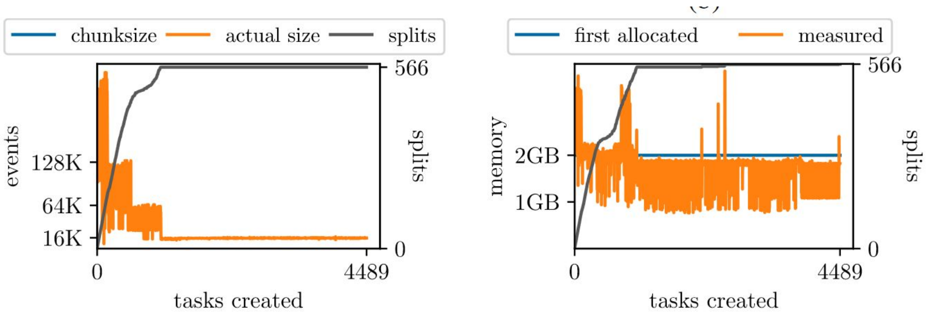
Scaling out TopEFT with Work Queue



- Work Queue is a system for creating and managing scalable manager-worker style programs
- To efficiently utilize distributed resources, TopEFT employs the Work Queue executor
- The Work Queue manager accepts task definitions from Coffea (for processing and accumulation tasks)
- Schedules the tasks to remote workers
- Sends along the relevant python environment with the task

TopEFT + WorkQueue

- Automatic resource allocation.
  - WQ finds cores and memory values per category of tasks to maximize throughput.
- Automatic python environment delivery
  - TopCoffea checks if there are new git commits of topCoffea or Coffea and constructs an environment tarball based on conda-pack. Environment cached at workers.
- Dynamic accumulation tasks creation
- Dynamic chunksize for given memory or time limits



Dynamic Task Shaping for High Throughput Data Analysis Applications in High Energy Physics, *IEEE International Parallel and Distributed Processing Symposium*, 2022 10.1109/IPDPS53621.2022.00041

TopEFT performance today at ND Tier-3

cpu needs

runtime:	100min
cores:	up to 1000
mem total:	18.5 TB
disk total:	7.2 TB

IO needs

total root data:	1.7 TB
data actually used:	0.75 TB
IO temp files:	**0.25 TB**
origin:	xrootd local

processing tasks

total:	23K
avg time:	110s
slowest:	318s
largest mem	4 GB
largest disk	0.5 GB

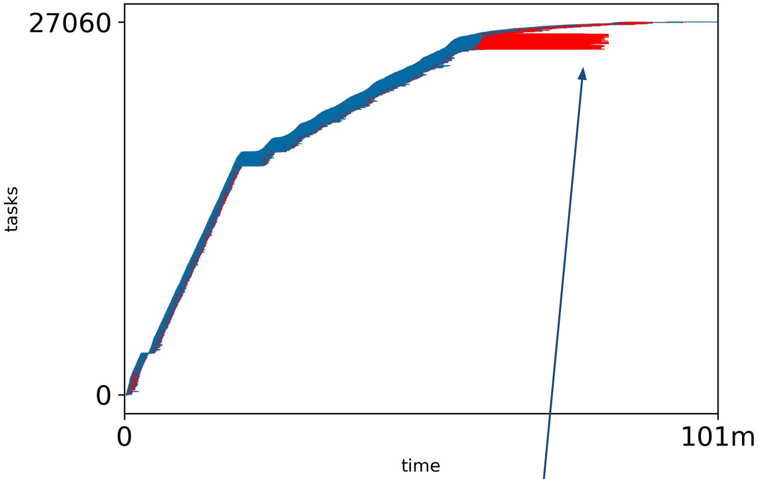
accumulation tasks

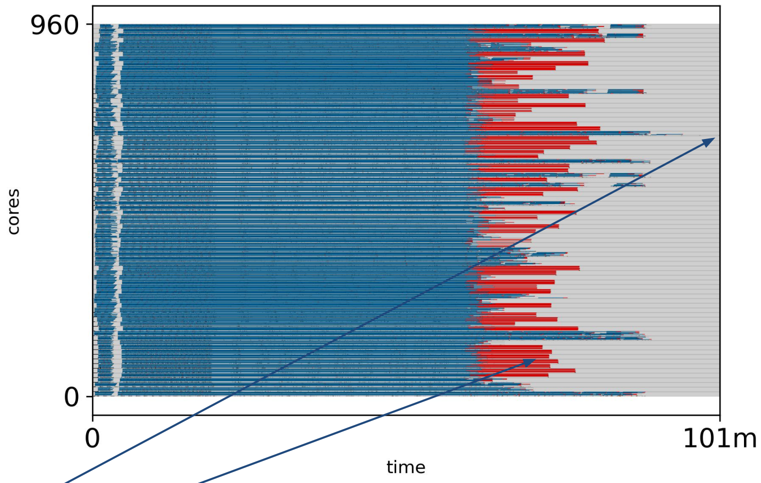
total:	1.2K
avg time:	6s
slowest:	**141s**
largest mem:	**12 GB**
largest disk:	**20 GB**

Current Bottlenecks Visualized

🟦 tasks executing
🟥 results waiting retrieval

Accumulation
Data Returned
TopEFT
+ Work Queue





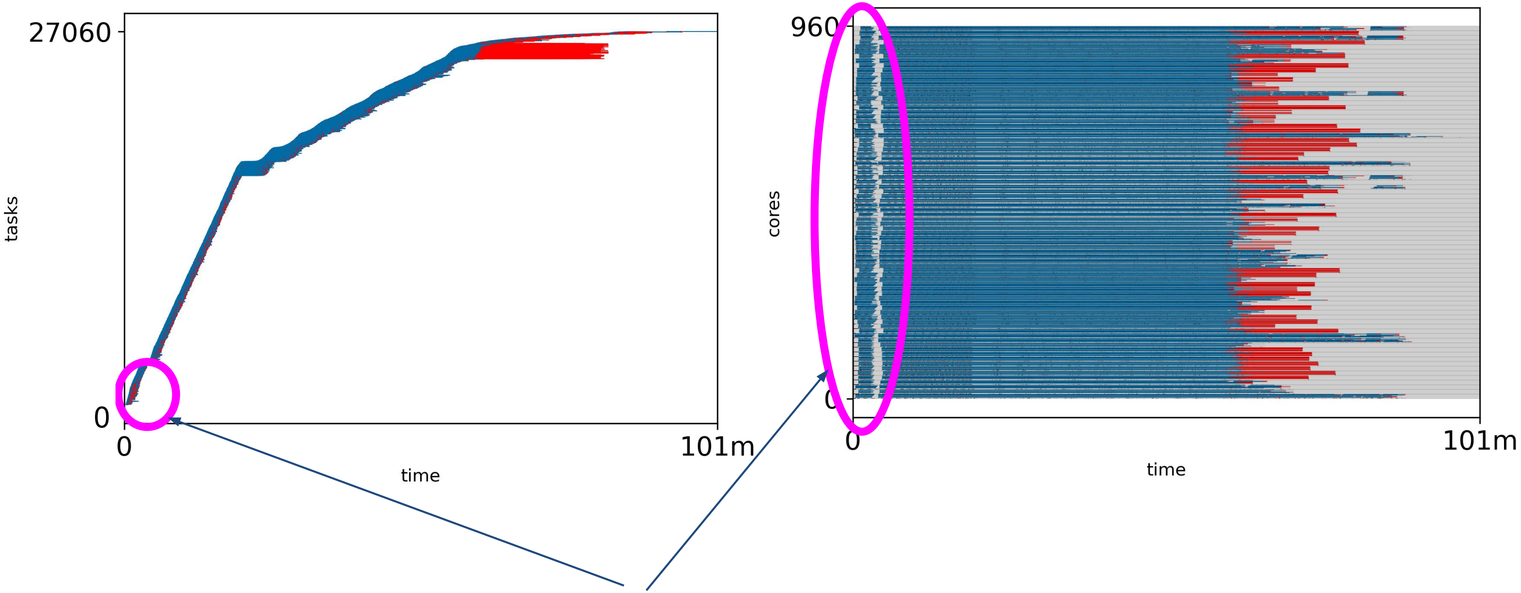
Long worker down time

Long accumulation tail

80 12-core workers
Whole Analysis: 101 minutes
Run over all Run 2 data and Monte Carlo

Current Bottlenecks Visualized

Accumulation
Data Returned
TopEFT
+ Work Queue



Preprocessing (~2 minutes)
XRootD requests and asset management

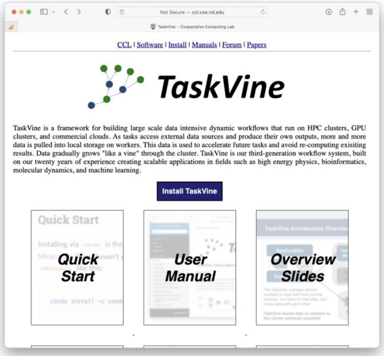
Current Performance Bottlenecks

In order of impact:

1. All partial results are returned to the manager, and sent back to workers for accumulation
2. XRootD servers on top of HDFS old spinning disks, which greatly limits bandwidth
3. Extra data read by the XRootD protocol that is not part of the read requests
4. Accumulation tasks may need tens of GB of memory, which reduces parallelism
5. Manager does not efficiently hand out tasks to workers or obtain workers

UNIVERSITY OF NOTRE DAME

Next: TaskVine Workflow Scheduler



Not Secure — ccl.cse.nd.edu

TaskVine: - Cooperative Computing Lab

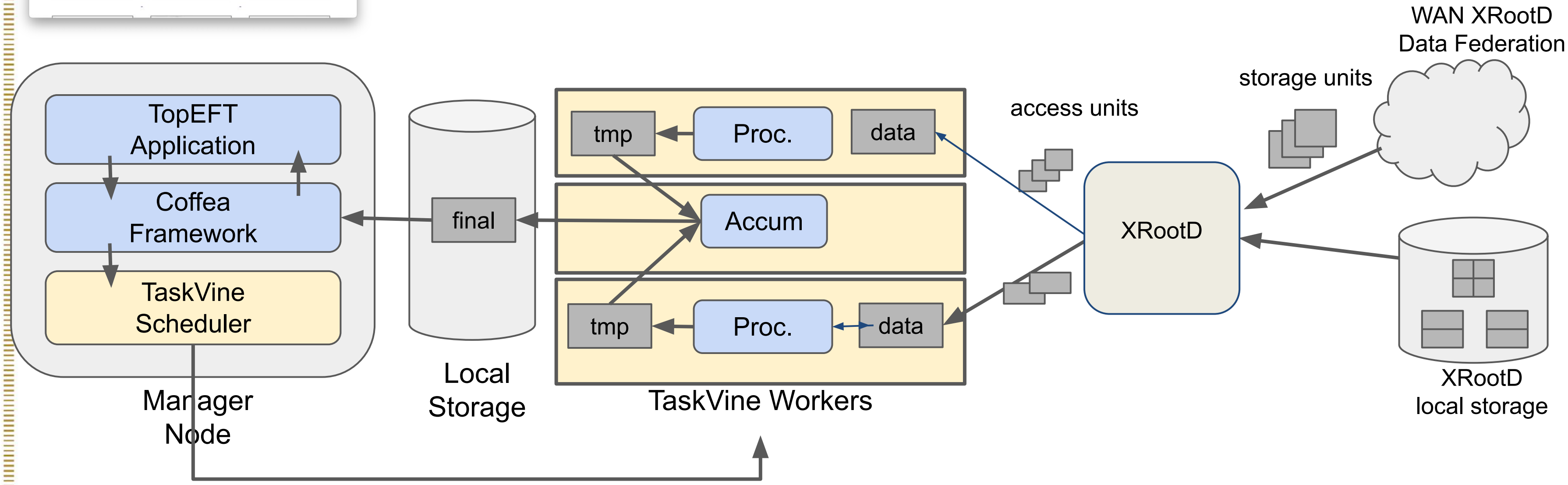
CCL | Software | Install | Manuals | Forum | Papers

TaskVine

TaskVine is a framework for building large scale data intensive dynamic workflows that run on HPC clusters, GPU clusters, and commercial clouds. As tasks access external data sources and produce their own outputs, more and more data is pulled into local storage on workers. This data is used to accelerate future tasks and avoid re-computing exisiting results. Data gradually grows "like a vine" through the cluster. TaskVine is our third-generation workflow system, built on our twenty years of experience creating scalable applications in fields such as high energy physics, bioinformatics, molecular dynamics, and machine learning.

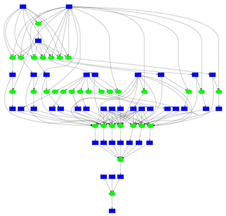
TaskVine is our next generation of workflow scheduler that improves upon Work Queue. Key idea: **data stays in the cluster** where it is accessed or created, so that tasks can simply use data in place, rather than moving it around. Our prototype of TopEFT running on TaskVine eliminates the "long tail" of accumulation tasks by keeping the intermediate data in place.

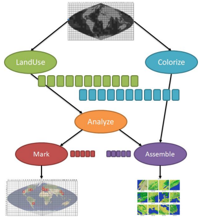
http://ccl.cse.nd.edu/software/taskvine

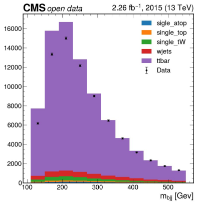


TaskVine Application Stack

```
import taskvine

file = URL(www)
m.submit(task)
task = m.wait(5)
```







Custom App

C or Python

Dask

Python

Parsl

Python

Coffea

Python

TaskVine Manager

TaskVine Worker

TaskVine Worker

TaskVine Worker

TaskVine Worker

TaskVine Worker

HPC/University Cluster

Changes Needed

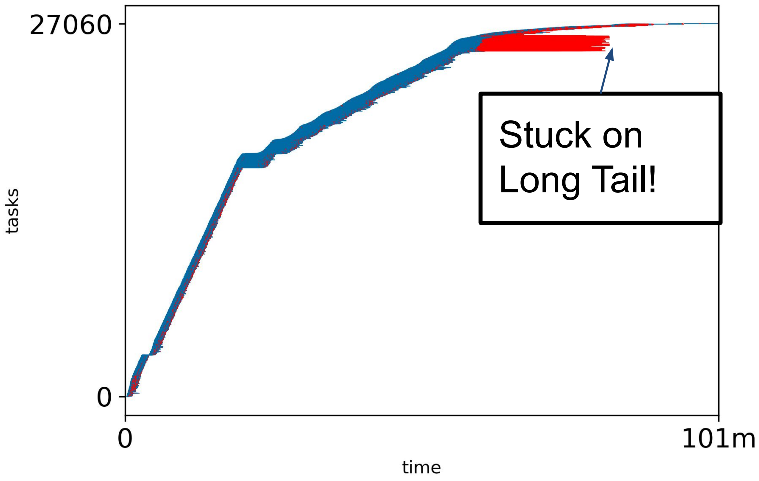
- **Data Storage System:** Every task in the system reads out a different selection of data. Need a data storage system that provides low latency (from open to first read) and high throughput (many clients reading separate data at once.)
  - Migrate away from HDFS on spinning disk cluster to Ceph, ServiceX on experimental NVMe cluster.
- **Managing Assets for Startup:** A significant amount of turnaround time is lost to startup: allocating nodes, transferring software environments, establishing connections.
  - Retain as much as possible on each cluster node, and design systems to exploit assets already present.
- ***Managing Data Reduction**: TopEFT in particular produces large quantities of intermediate data: transferring it back to a central point results in exponential growth of network traffic:
  - Leave data where it is created in the cluster, and dispatch accumulation tasks to consume it in place. (Requires closer attention to failure and recovery.)

UNIVERSITY OF NOTRE DAME

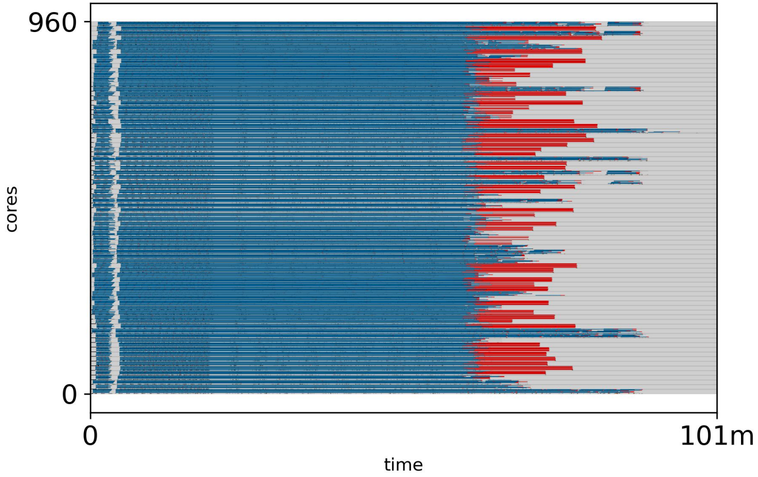
Old: Accumulation Data Returned

TopEFT
+ Work Queue

102 minutes



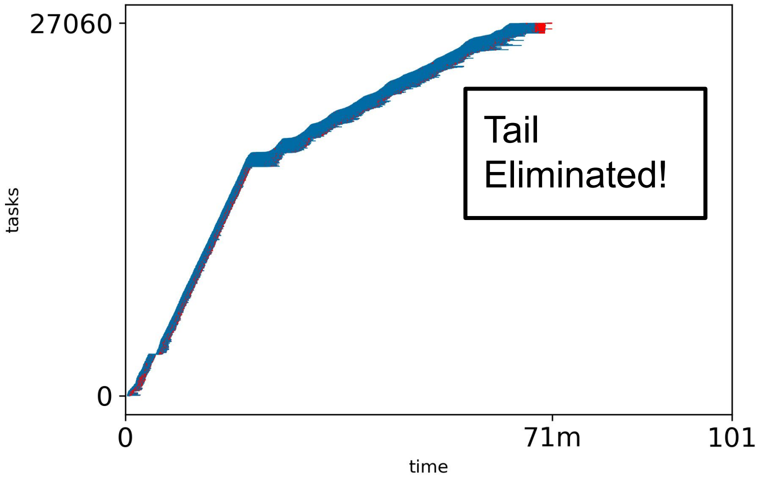
Stuck on Long Tail!



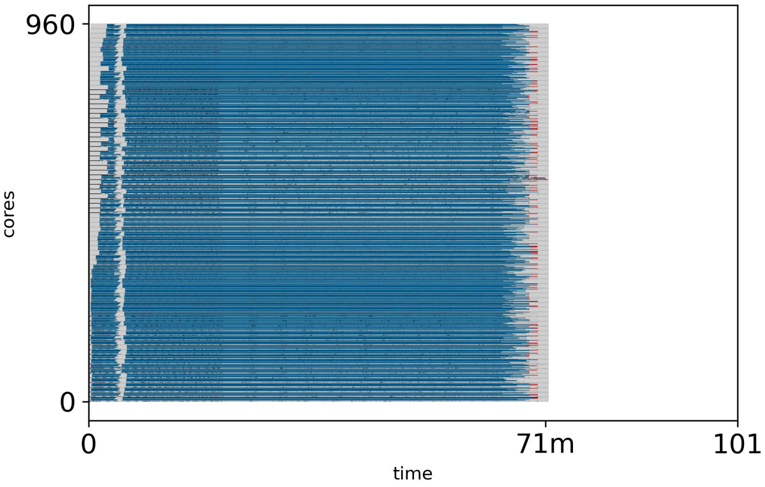
New: In-Cluster Accumulation

TopEFT
+ TaskVine

83 minutes



Tail Eliminated!

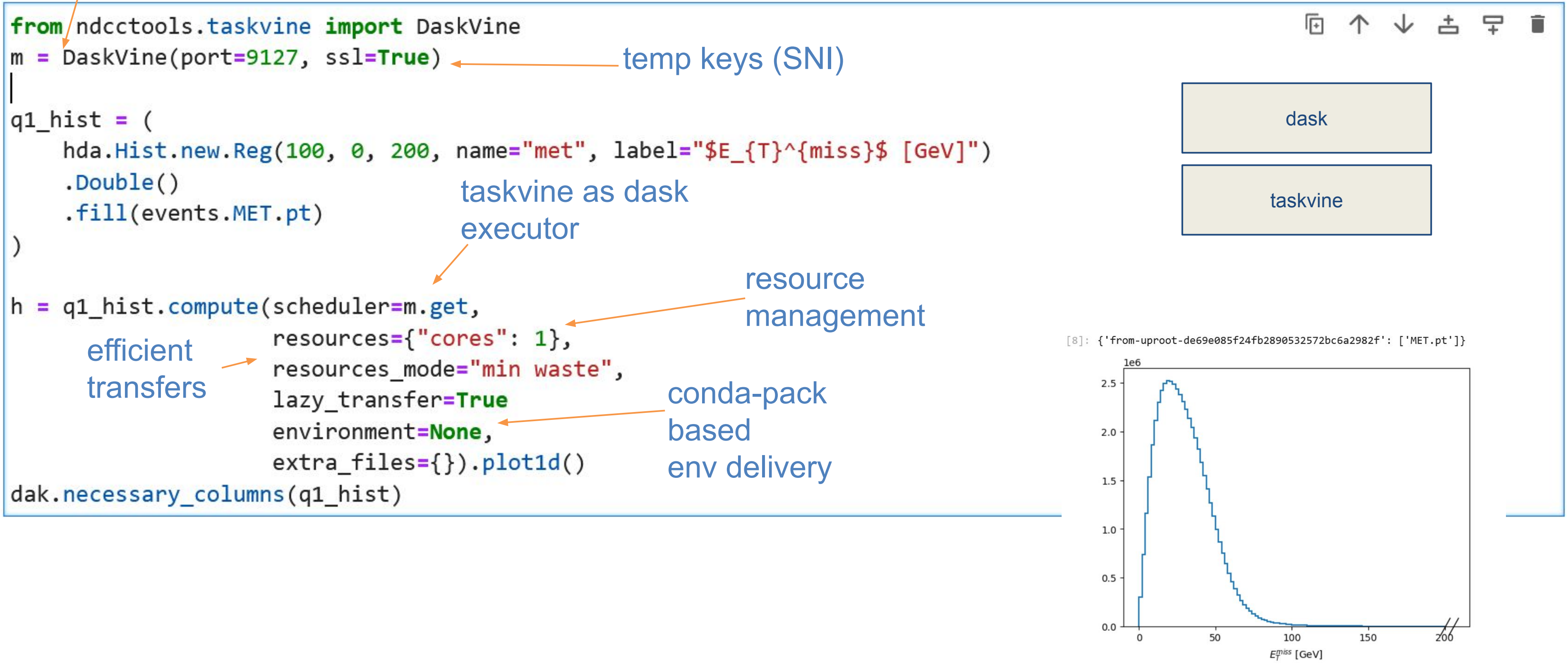


Work Queue / TaskVine and AGC connections

TaskVine executing Dask

(experimental on cctools 7.5.4)

specialized manager to execute dask
only final results loaded into memory



```
from ndcctools.taskvine import DaskVine
m = DaskVine(port=9127, ssl=True)

q1_hist = (
    hda.Hist.new.Reg(100, 0, 200, name="met", label="$E_{T}^{miss}$ [GeV]")
    .Double()
    .fill(events.MET.pt)
)

h = q1_hist.compute(scheduler=m.get,
                    resources={"cores": 1},
                    resources_mode="min waste",
                    lazy_transfer=True
                    environment=None,
                    extra_files={}).plot1d()
dak.necessary_columns(q1_hist)
```

```
[8]: {'from-uproot-de69e085f24fb2890532572bc6a2982f': ['MET.pt']}
```

temp keys (SNI)

taskvine as dask executor

resource management

efficient transfers

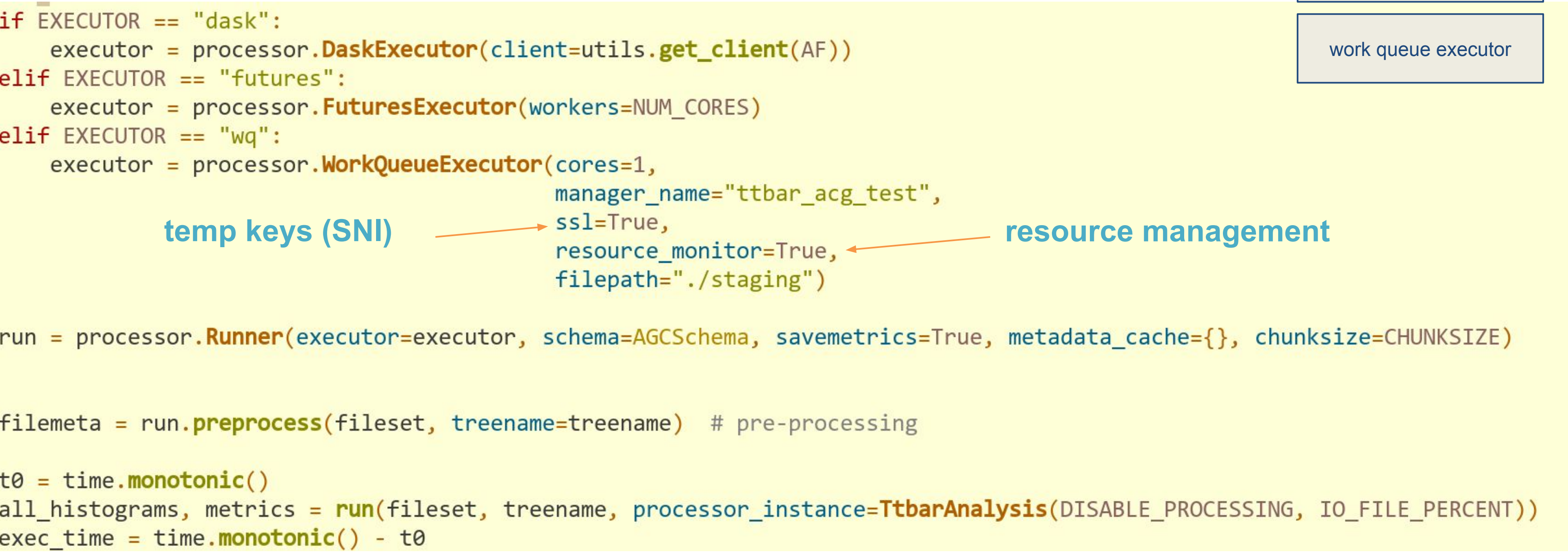
conda-pack based env delivery

WQ executing ACG analysis (coffea pre-2023)

ttbar_analysis_pipeline

coffea

work queue executor

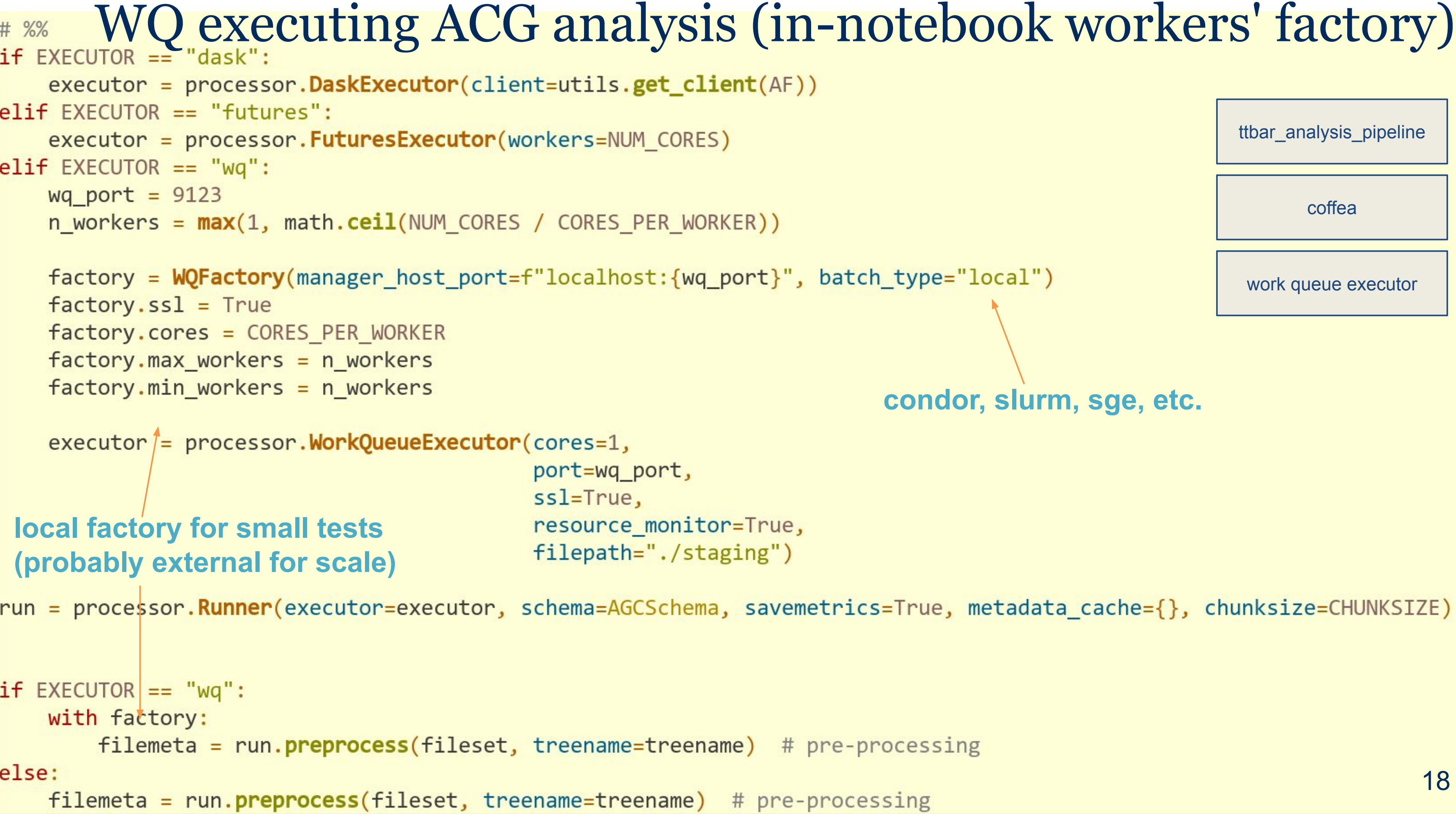


```
if EXECUTOR == "dask":
    executor = processor.DaskExecutor(client=utils.get_client(AF))
elif EXECUTOR == "futures":
    executor = processor.FuturesExecutor(workers=NUM_CORES)
elif EXECUTOR == "wq":
    executor = processor.WorkQueueExecutor(cores=1,
                                           manager_name="ttbar_acg_test",
                                           ssl=True,
                                           resource_monitor=True,
                                           filepath="./staging")

run = processor.Runner(executor=executor, schema=AGCSchema, savemetrics=True, metadata_cache={}, chunksize=CHUNKSIZE)


filemeta = run.preprocess(fileset, treename=treename)  # pre-processing

t0 = time.monotonic()
all_histograms, metrics = run(fileset, treename, processor_instance=TtbarAnalysis(DISABLE_PROCESSING, IO_FILE_PERCENT))
exec_time = time.monotonic() - t0
```

temp keys (SNI)

resource management

WQ executing ACG analysis (in-notebook workers' factory)



```
# %%
if EXECUTOR == "dask":
    executor = processor.DaskExecutor(client=utils.get_client(AF))
elif EXECUTOR == "futures":
    executor = processor.FuturesExecutor(workers=NUM_CORES)
elif EXECUTOR == "wq":
    wq_port = 9123
    n_workers = max(1, math.ceil(NUM_CORES / CORES_PER_WORKER))

    factory = WQFactory(manager_host_port=f"localhost:{wq_port}", batch_type="local")
    factory.ssl = True
    factory.cores = CORES_PER_WORKER
    factory.max_workers = n_workers
    factory.min_workers = n_workers

    executor = processor.WorkQueueExecutor(cores=1,
                                           port=wq_port,
                                           ssl=True,
                                           resource_monitor=True,
                                           filepath="./staging")

run = processor.Runner(executor=executor, schema=AGCSchema, savemetrics=True, metadata_cache={}, chunksize=CHUNKSIZE)


if EXECUTOR == "wq":
    with factory:
        filemeta = run.preprocess(fileset, treename=treename)  # pre-processing
else:
    filemeta = run.preprocess(fileset, treename=treename)  # pre-processing
```

ttbar_analysis_pipeline

coffea

work queue executor

condor, slurm, sge, etc.

local factory for small tests
(probably external for scale)

TaskVine as pool for FuturesExecutor (probably already obsolete)

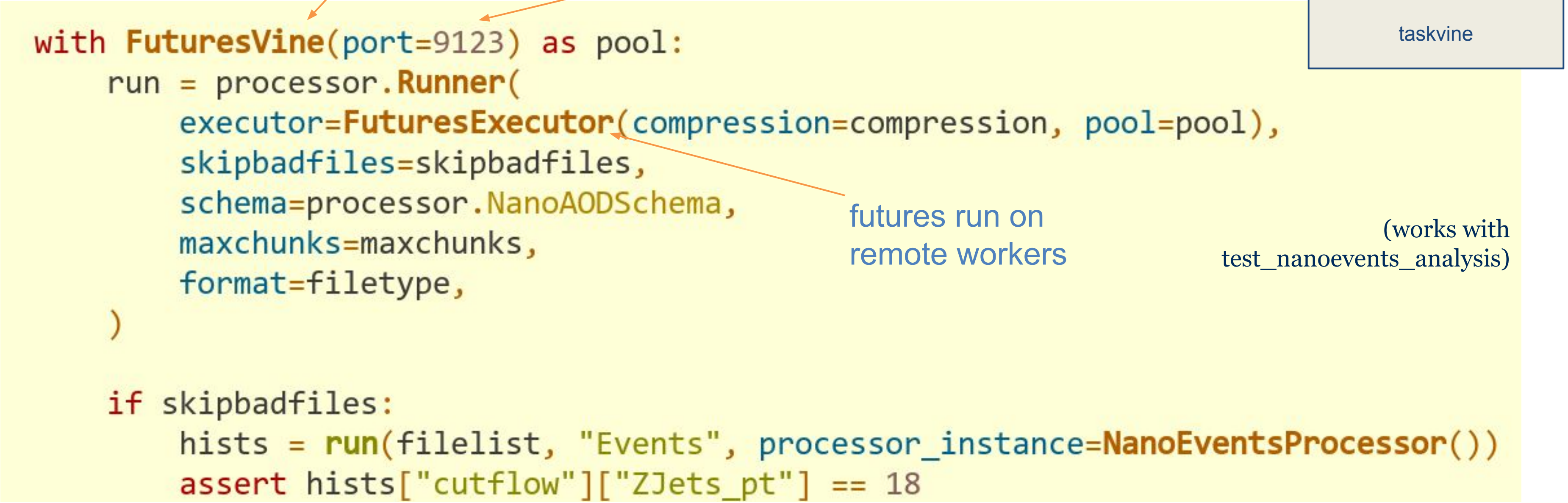
coffea

FuturesExecutor

taskvine

specialized manager that implements concurrent.futures.Executor

initialize as any other TaskVine manager



```
with FuturesVine(port=9123) as pool:
    run = processor.Runner(
        executor=FuturesExecutor(compression=compression, pool=pool),
        skipbadfiles=skipbadfiles,
        schema=processor.NanoAODSchema,
        maxchunks=maxchunks,
        format=filetype,
    )

    if skipbadfiles:
        hists = run(filelist, "Events", processor_instance=NanoEventsProcessor())
        assert hists["cutflow"]["ZJets_pt"] == 18
```

futures run on remote workers

(works with test_nanoevents_analysis)

https://cctools.readthedocs.io
https://github.com/cooperative-computing-lab/cctools
https://github.com/TopEFT/topcoffea

```
conda install -c conda-forge ndcctools
```

Thanks to team and collaborators

ND CMS and topCoffea

Prof. Kevin Lannon
Kelci Mohrman (Dr.!)
John Lawrence
Brent R. Yates
Andrew Wightman
Andrea Trapote

Other Collaborators

Parsl team

CCL

Prof. Douglas Thain
Benjamin Tovar (RSE)
Thanh Son Phung
Barry Sly-Delgado
Colin Thomas
David Simonetti
Joe Duggan
Andrew Hennessee
Jachob Dolak

This work was supported by NSF Award OAC-1931348