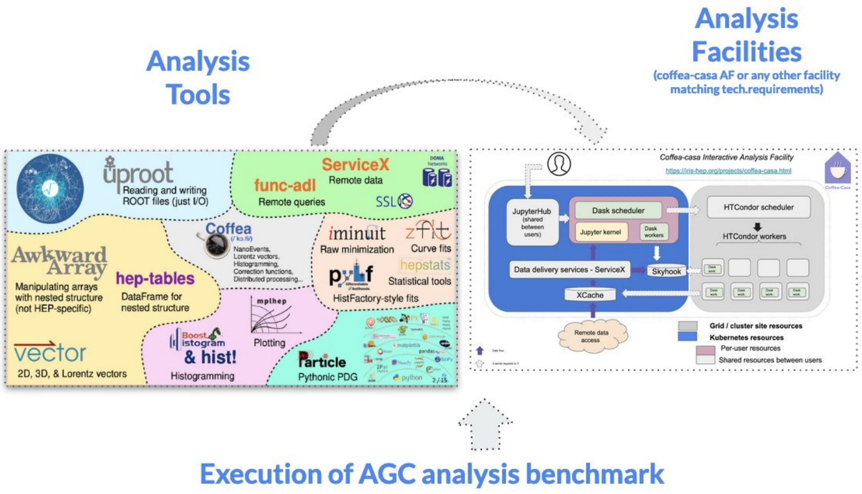
Performance Benchmarks for Analysis Grand Challenge

Mentee: Holly Wingren (UIUC)
Mentor: Carl Lundstedt (UNL)

Introduction

- Analysis Grand Challenge (AGC) of IRIS-HEP
  - Binned analysis, reinterpretation and end-to-end optimization of physics analysis use cases
  - Includes development of the required cyber infrastructure to execute them to demonstrate technologies envisioned for HL-LHC
- Capabilities include:
  - New user interfaces
  - Data access
  - Event selection
  - Statistical model building and fitting
  - Reinterpretation / analysis preservation
- Our goal: use pieces of an example physics analysis to study the performance of different system components



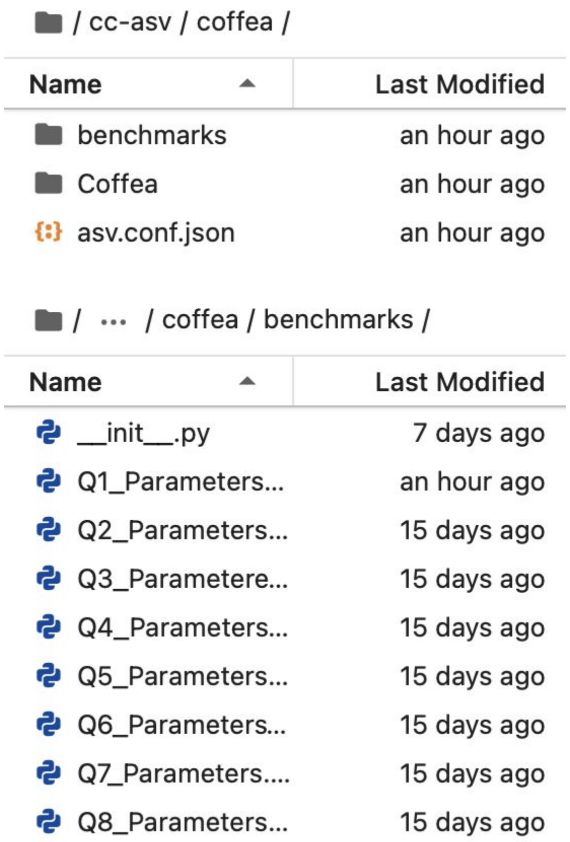
https://iris-hep.org/projects/agc

Purpose of Benchmarking

- "Why care how fast a system goes?"
  - Is it going as fast as it can? Is it going faster than other configurations?
- Measurement or a set of measurements related to the performance of a piece of code in an application
- Compares the performance between different system configurations and different running conditions
- Purpose: innovate, identify bottlenecks, compare techniques and technologies
- How do we Benchmark?
  - Execute benchmarks in Python
  - Use asv to automatically generate results and publish to web pages (GitHub)
  - Compare different versions of different packages
  - In case of performance regression, try to identify the commit which caused slow down of benchmark

Air Speed Velocity (asv)

- Tool for benchmarking Python packages over their lifetime
  - Runtime, memory consumption and even custom-computed values may be tracked
- Installed from PyPI using “pip3 install asv”
- Results displayed in an interactive web frontend
  - Requires only a basic static web server to host
- Benchmarks stored in a Python package
  - Collection of .py files in benchmark suite’s benchmark directory (as defined by benchmark_dir in the asv.conf.json file)
  - Within each .py file, each benchmark is a function or method



Benchmarks Tracked

- For each query, we tracked:
  - Walltime
  - Average Number of Threads
  - Bytes per Second
  - Chunksize per Second
  - Bytes per Thread per Second
  - Chunksize per Thread per Second
- For Coffea ADL Benchmarks and tt bar analysis, we were able to parameterize different variables (chunksize, max number of files, etc)

Coffea ADL Benchmarks

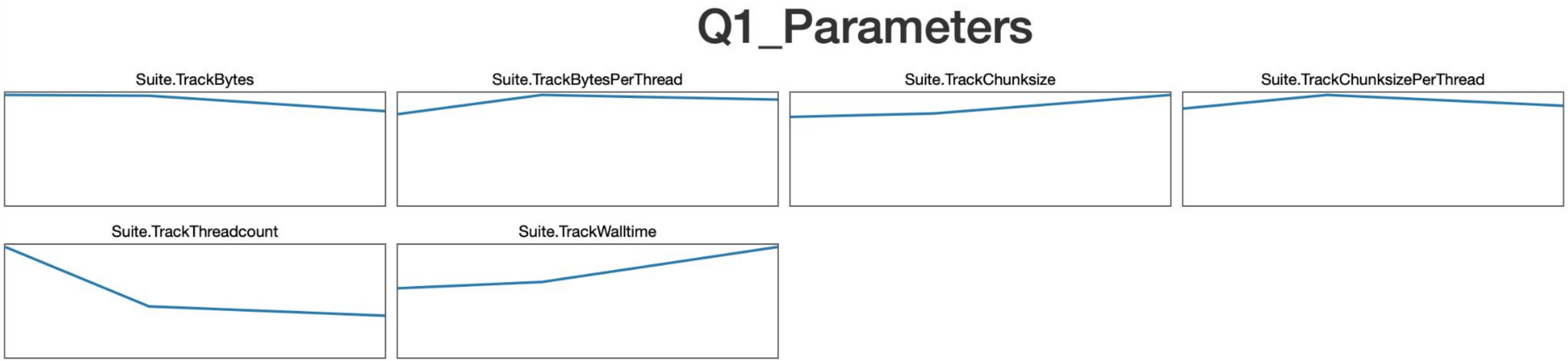
- Q1: All missing transverse energy (E_T^{miss}) in all the events
- Q2: Plot transverse momentum (p_T) of ALL the jets in all events
- Q3: Q2 but only for central jets (eta < 1)
- Q4: Plot E_T^{miss} for events that have at least two jets with Pt > 40 GeV
- Q5: Plot E_T^{miss} for events with opposite-charge muon pair with invariant mass between 60 and 120 GeV
- Q6: For events with at least 3 jets, plot the p_T of the trijet system four-momentum that has an invariant mass closest to 172.5 GeV, in each event plot maximum b-tagging discriminant value among the jets in this trijet.
- Q7: Plot the scalar sum in each event of the p_T of the jets with p_T > 30 GeV that are not with 0.4 in delta R of any light lepton with p_T > 10 GeV (Jets not aligned with lepton).
- Q8: For events with at least 3 light leptons and same-flavor, opposite-charge light lepton pair find such a pair that has a transverse mass closest to 91.2 GeV (Z boson) and plot the transverse mass of the system consisting of the E_T^{miss} and highest p_T lepton NOT in the Z pair.

asv Coffea ADL Benchmarks Output

- 131074, 262144, 524288 are a parametrized values of chunksize

```
[  0.69%] ··· Q1_Parameters.Suite.TrackBytes
[  0.69%] ··· ================== ===================
               Bytes per Second
              ------------------ -------------------
                    131072         7340137.559497975
                    262144          10609057.0794363
                    524288         8889645.535759974
              ================== ===================

[  1.39%] ··· Q1_Parameters.Suite.TrackBytesPerThread
[  1.39%] ··· ================== ====================
               Bytes per Thread
              ------------------ --------------------
                    131072         4550522.4118167125
                    262144         4045014.458580545
                    524288         3721829.442479468
              ================== ====================

[  2.08%] ··· Q1_Parameters.Suite.TrackChunksize
[  2.08%] ··· ====================== ====================
               Chunksize per Second
              ---------------------- --------------------
                      131072           4792.209902909048
                      262144           16556.790885231017
                      524288            31962.9542402445
              ====================== ====================

[  2.78%] ··· Q1_Parameters.Suite.TrackChunksizePerThread
[  2.78%] ··· ====================== ===================
               Chunksize per Thread
              ---------------------- -------------------
                      131072           958.9314505294694
                      262144           2409.355647460271
                      524288           5643.808664292671
              ====================== ===================

[  3.47%] ··· Q1_Parameters.Suite.TrackThreadcount
[  3.47%] ··· =========================== ===================
               Average Number of Threads
              --------------------------- -------------------
                         131072              4.6891098343871
                         262144             4.115758281753771
                         524288             2.261038003211384
              =========================== ===================

[  4.17%] ··· Q1_Parameters.Suite.TrackWalltime
[  4.17%] ··· ========== ====================
               Walltime
              ---------- --------------------
                131072     104.28907497040927
                262144     14.740117965266109
                524288     10.344016100279987
              ========== ====================
```

Coffea ADL Plots

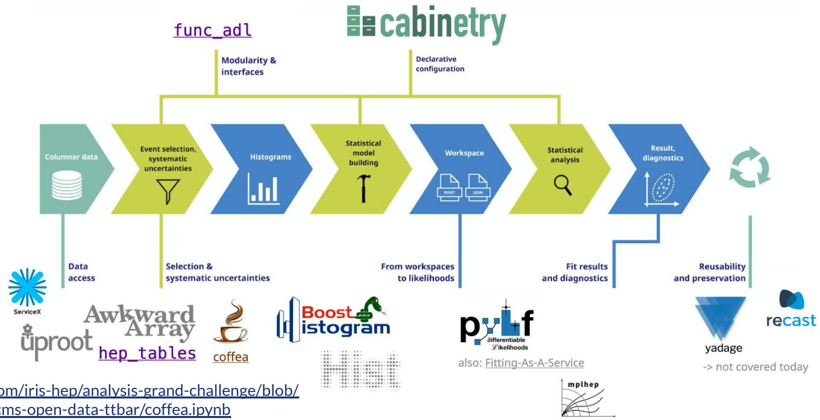


Coffea ADL Issues

- The way we had written our benchmarks, it ran the entire block of code each time
  - Took hours to fully run
- Tried a different approach that took less time to complete
  - Couldn’t parameterize

CMS Open Data $t\bar{t}$: from data delivery to statistical inference

- 2015 CMS Open Data - showcases an analysis pipeline
  - Features data delivery / processing, histogram construction / visualization, and statistical inference
- Technical demonstration
  - Includes relevant workflow aspects that physicists need, but isn't focused on making every piece physically meaningful
  - Particular systematic uncertainties: capture the workflow, but actual implementations are more complex in practice
- Three different data pipelines:
  - pure coffea - process data and aggregate histograms
  - coffea w/ Servicex processors - sends data to coffea, processors start running asynchronously
  - ServiceX followed by coffea - standalone ServiceX, data transfer, allowed by standalone coffea processing



https://github.com/iris-hep/analysis-grand-challenge/blob/main/analyses/cms-open-data-ttbar/coffea.ipynb

CMS Open Data $t\bar{t}$ Output

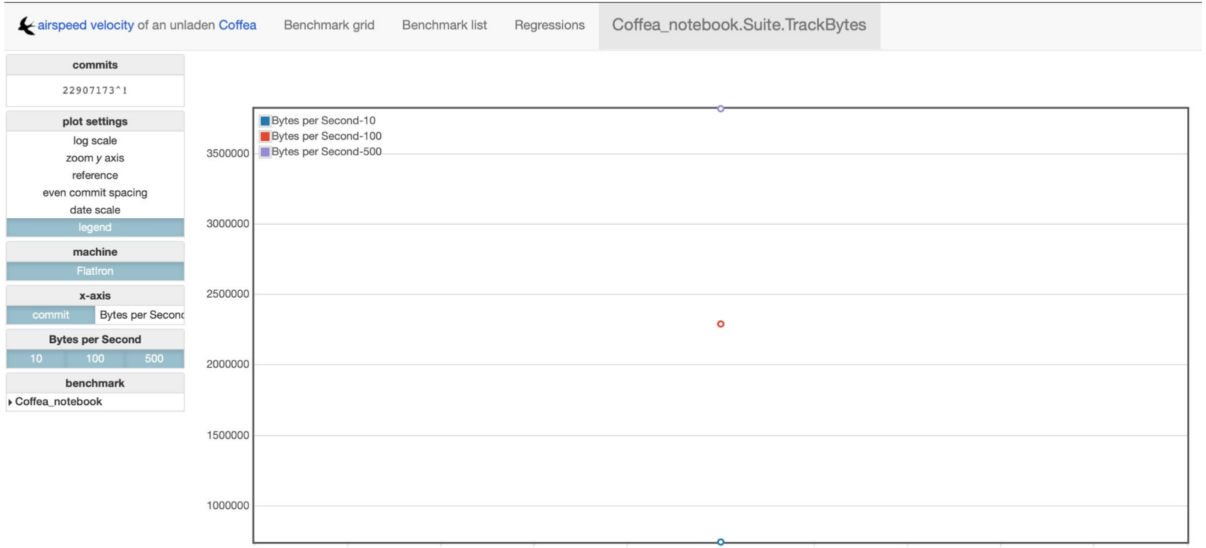
- 10, 100, 500 are the parameterized values of number of files

```
[ 25.00%] ··· Coffea_notebook.Suite.TrackBytes
[ 25.00%] ··· =================== =====================
               Bytes per Second
              ------------------- ---------------------
                       10           745541.8535327591
                      100           2291260.5771551155
                      500           3816971.075015612
              =================== =====================

[ 50.00%] ··· Coffea_notebook.Suite.TrackBytesPerThread
[ 50.00%] ··· =================== =====================
               Bytes per Thread
              ------------------- ---------------------
                       10           119555.69400317766
                      100           104553.4279952741
                      500           98564.61067434892
              =================== =====================
```

```
[ 75.00%] ··· Coffea_notebook.Suite.TrackThreadcount
[ 75.00%] ··· ============================ =====================
               Average Number of Threads
              ---------------------------- ---------------------
                           10                4.444336225021103
                          100                14.252010918084896
                          500                33.71500969104301
              ============================ =====================

[100.00%] ··· Coffea_notebook.Suite.TrackWalltime
[100.00%] ··· ========== =====================
               walltime
              ---------- ---------------------
                  10       16.9016535282135
                 100      64.12671256065369
                 500      1415.4470806121826
              ========== =====================
```

CMS Open Data $t\bar{t}$ Plots

- https://hollywingren.github.io/HollyWingren-cc-adl/#/



CMS Open Data $t\bar{t}$ Issues

- Very large files for 1000 and -1
  - Caused crash
- Needed to restart Dask cluster
- Possible issue with asv and/or Coffea not in the correct environment

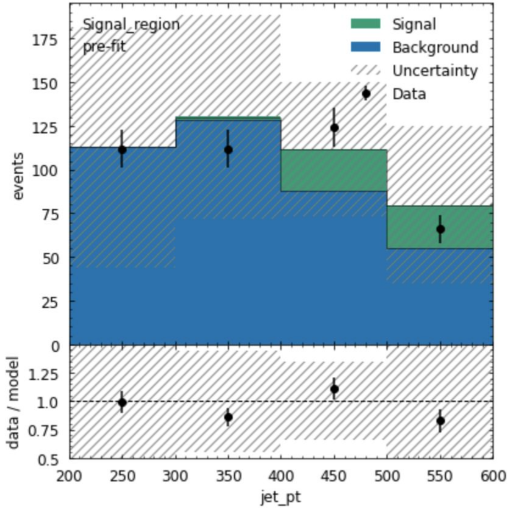
setting	number of files	total size
10	90	15.6 GB
100	850	150 GB
500	3545	649 GB
1000	5864	1.05 TB
-1	22635	3.44 TB

cabinetry

- Python library for building and steering binned template fits
  - Written with applications in High Energy Physics in mind
- Interfaces many other powerful libraries to make it easy for an analyzer to run their statistical inference pipeline
- Statistical model building and fitting
- Requires configuration file w/ 4 blocks of settings and 3 systematic uncertainties:
  - General, Regions, Samples, NormFactors
  - Luminosity, Modeling, WeightBasedModeling
- Example contains workspace building, maximum likelihood fitting, visualization, ranking

```
model_pred = cabinetry.model_utils.prediction(model)
figures = cabinetry.visualize.data_mc(model_pred, data, config=cabinetry_config)
```

```
DEBUG - cabinetry.model_utils - total stdev is [[69, 58.3, 38.2, 45.3]]
DEBUG - cabinetry.model_utils - total stdev per channel is [137]
DEBUG - cabinetry.visualize.utils - saving figure as figures/Signal_region_prefit.pdf
```

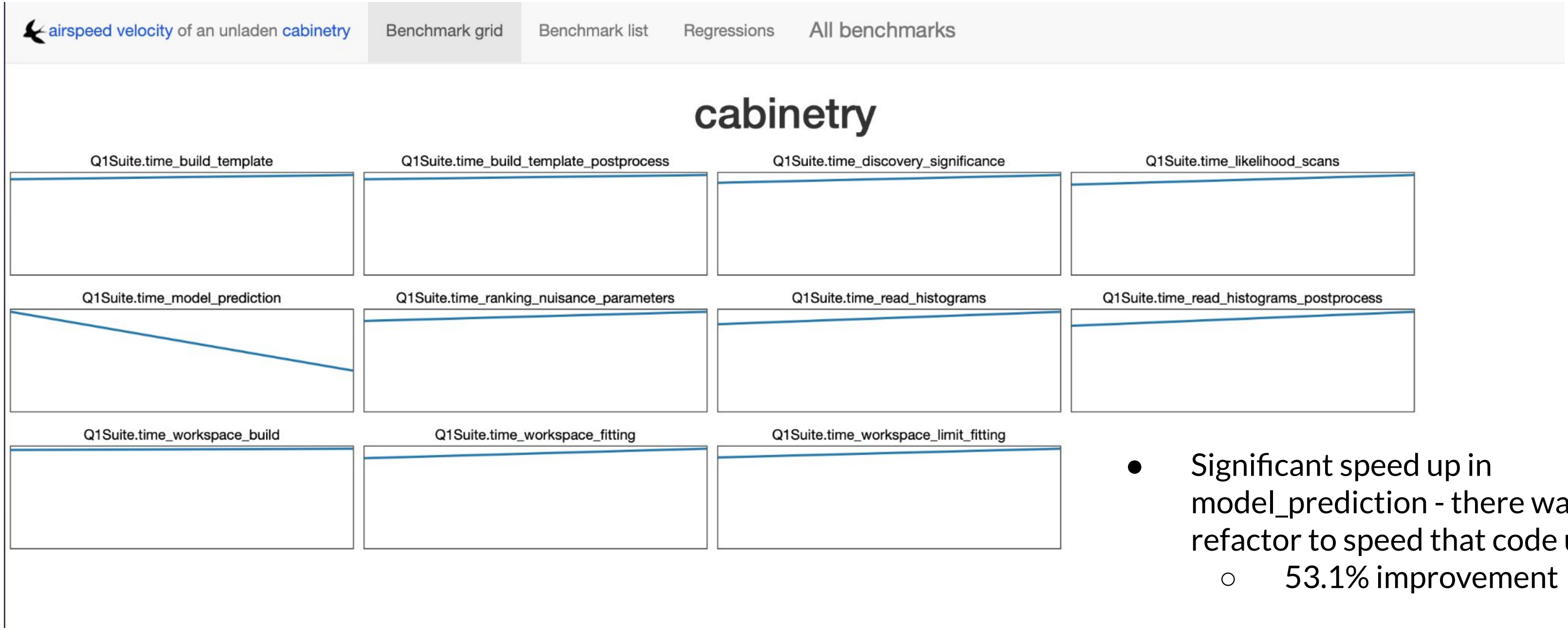


https://github.com/cabinetry/cabinetry-tutorials/blob/master/example.ipynb

cabinetry Output

```
· Running 22 total benchmarks (2 commits * 1 environments * 11 benchmarks)
[  0.00%] · For cabinetry commit ae7444cd <v0.4.0^0>:
[  0.00%] ·· Benchmarking virtualenv-py3.8-pip+wget-uproot
[  2.27%] ··· Running (cabinetry.Q1Suite.time_build_template--)........
[ 20.45%] ··· Running (cabinetry.Q1Suite.time_workspace_build--)...
[ 27.27%] ··· cabinetry.Q1Suite.time_build_template                                31.3±0.3ms
[ 29.55%] ··· cabinetry.Q1Suite.time_build_template_postprocess                    39.5±0.2ms
[ 31.82%] ··· cabinetry.Q1Suite.time_discovery_significance                         236±0.2ms
[ 34.09%] ··· cabinetry.Q1Suite.time_likelihood_scans                                 519±2ms
[ 36.36%] ··· cabinetry.Q1Suite.time_model_prediction                                 474±1μs
[ 38.64%] ··· cabinetry.Q1Suite.time_ranking_nuisance_parameters                      1.45±0s
[ 40.91%] ··· cabinetry.Q1Suite.time_read_histograms                              11.9±0.05ms
[ 43.18%] ··· cabinetry.Q1Suite.time_read_histograms_postprocess                  20.2±0.04ms
[ 45.45%] ··· cabinetry.Q1Suite.time_workspace_build                                  252±3μs
[ 47.73%] ··· cabinetry.Q1Suite.time_workspace_fitting                             62.7±0.2ms
[ 50.00%] ··· cabinetry.Q1Suite.time_workspace_limit_fitting                          9.05±0s
```

cabinetry Plots



- Significant speed up in model_prediction - there was a refactor to speed that code up
  - 53.1% improvement

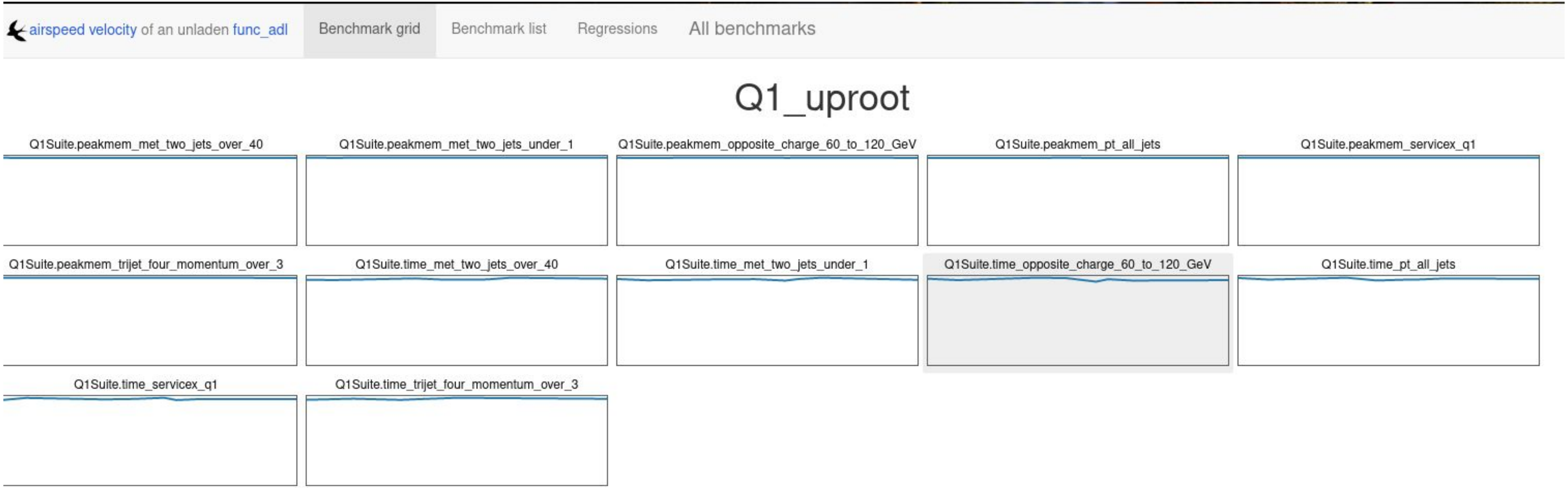
func_adl

- Query languages
  - Database management systems help to address:
    - data independence
    - data redundancy
- Functional languages
  - Functional programming offers several desirable features for physics analyses:
    - Declarative
    - Stateless
    - Lazy
- Both of these concepts (query languages and functional languages) lead to more modular code:
  - Insulate analysis code from data storage location and file format
  - Insulate each section of code from other parts of the code
- 6 tasks:
  - Task 1: Plot the E_T^{miss} of all events
  - Task 2: Plot the p_T of all jets
  - Task 3: Plot the p_T of jets with $|\eta| < 1$
  - Task 4: Plot the E_T^{miss} of events that have at least two jets with pT > 40 GeV
  - Task 5: Plot the E_T^{miss} of events that have an opposite-charge muon pair with an invariant mass between 60 and 120 GeV
  - Task 6: For events with at least three jets, plot the p_T of the trijet four-momentum that has the invariant mass closest to 172.5 GeV in each event

func_adl Output

```
• Running 96 total benchmarks (8 commits * 1 environments * 12 benchmarks)
[  0.00%] · For func_adl commit 61b35593 <maint>:
[  0.00%] ·· Benchmarking conda-py3.8
[  3.65%] ··· Running (Q1_uproot.Q1Suite.time_met_two_jets_over_40--)......
[  6.77%] ··· Q1_uproot.Q1Suite.peakmem_met_two_jets_over_40                          81.6M
[  7.29%] ··· Q1_uproot.Q1Suite.peakmem_met_two_jets_under_1                          82.1M
[  7.81%] ··· Q1_uproot.Q1Suite.peakmem_opposite_charge_60_to_120_GeV                 85.2M
[  8.33%] ··· Q1_uproot.Q1Suite.peakmem_pt_all_jets                                   80.6M
[  8.85%] ··· Q1_uproot.Q1Suite.peakmem_servicex_q1                                   79.4M
[  9.38%] ··· Q1_uproot.Q1Suite.peakmem_trijet_four_momentum_over_3                   89.7M
[  9.90%] ··· Q1_uproot.Q1Suite.time_met_two_jets_over_40                           469±6ms
[ 10.42%] ··· Q1_uproot.Q1Suite.time_met_two_jets_under_1                           571±8ms
[ 10.94%] ··· Q1_uproot.Q1Suite.time_opposite_charge_60_to_120_GeV                  853±7ms
[ 11.46%] ··· Q1_uproot.Q1Suite.time_pt_all_jets                                    463±6ms
[ 11.98%] ··· Q1_uproot.Q1Suite.time_servicex_q1                                    286±3ms
[ 12.50%] ··· Q1_uproot.Q1Suite.time_trijet_four_momentum_over_3                    920±8ms
```

func_adl Plots



Conclusions and Future Work

- Successfully converted ADL, FUNC_ADL, CABINETRY and Coffea benchmarks to ASV and can publish the results to github pages
- Future benchmarks could include new I/O products such as ServiceX and Skyhook
- Implement the t-tbar benchmarks at various scales once problems with Coffea's interaction with ASV are addressed

References

Cranmer, Kyle, and Alexander Held. “Cabinetry.” *Institute for Research and Innovation in Software for High Energy Physics*, 28 June 2022, https://iris-hep.org/projects/cabinetry.html.

Droettboom, Michael, and Pauli Virtanen. “Airspeed Velocity.” *Airspeed Velocity 0.5.1 Documentation*, 2018, https://asv.readthedocs.io/en/stable/index.html.

Graur, Dan, et al. “Evaluating Query Languages and Systems for High-Energy Physics Data [Extended Version].” *Inspire HEP*, 26 Apr. 2021, https://inspirehep.net/literature/1860769.

Held, Alexander, and Oksana Shadura. “The Analysis Grand Challenge.” *Institute for Research and Innovation in Software for High Energy Physics*, https://iris-hep.org/projects/agc.

Held, Alexander, et al. “Scikit-Hep/Cabinetry.” *GitHub*, 11 Feb. 2022, https://github.com/scikit-hep/cabinetry.

Held, Alexander. “Analysis-Grand-Challenge/Coffea.ipynb.” *GitHub*, 9 Aug. 2022, https://github.com/iris-hep/analysis-grand-challenge/blob/main/analyses/cms-open-data-ttbar/coffea.ipynb.

Lundstedt , Carl. “Clundst/CL-Bench.github.io.” *GitHub*, 10 Aug. 2022, https://github.com/clundst/cl-bench.github.io.

Proffitt, Mason, et al. “Iris-HEP/ADL-Benchmarks-Index.” *GitHub*, 25 Jan. 2022, https://github.com/iris-hep/adl-benchmarks-index.

Proffitt, Mason. “Func-Adl-Demo/Demo.ipynb.” *GitHub*, 3 Nov. 2021, https://github.com/masonproffitt/func-adl-demo/blob/master/demo.ipynb.

Shadura, Oksana, and Alexander Held. “Oshadura/CC-ASV.” *GitHub*, 4 Aug. 2022, https://github.com/oshadura/cc-asv.

Smith, Nicholas. “Coffea-Benchmarks/Coffea-ADL-Benchmarks.ipynb.” *GitHub*, 15 Nov. 2021, https://github.com/CoffeaTeam/coffea-benchmarks/blob/master/coffea-adl-benchmarks.ipynb.

Watts, Gordon, et al. “Iris-Hep/FUNC_ADL.” *GitHub*, 22 Mar. 2022, https://github.com/iris-hep/func_adl.