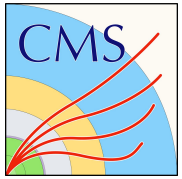


coffea-casa: Alpha Testing
&
ServiceX

Final Presentation for the IRIS-HEP Fellowship
Mentor: Dr. Oksana Shadura

IRIS-HEP Fellowship Goals

- Contribute to the further development of the Coffea-Casa Analysis Facility (AF) at University of Nebraska-Lincoln (UNL)
- Expand a gallery of Coffea-Casa analysis samples with existing analysis from CMS adapted to be executed in AF@UNL
- Facilitate the use of Coffea-Casa AF for Boston University and UNL CMS physicists currently working with NanoAOD datasets
- Investigation of possibility to integrate Coffea-Casa with Skyhook DM.

Fake Rates Analysis for Coffea-Casa

- Coffea-Casa Notebook
- Fake Rates Repository

Understanding the Big Picture Physics Question and the Detailed Steps

- Motivation for the fake rates analysis is to reduce the background for the Flavor-Changing Neutral Current studies
- Kaitlin Salyer from BU Group created slides for the steps of the cuts, definition of control region and specific goals

Familiarizing with the Coffea framework with Awkward Arrays

- Selected cuts based on the slides with reference of coffea examples
- Some remaining redundancy in the code
- Plotting 2D ratio histogram took more time than expected

Fake Rates Analysis for Coffea-Casa

- Coffea-Casa Notebook
- Fake Rates Repository

Understanding the Big Picture Physics Question and the Detailed Steps

- Motivation for the fake rates analysis is to reduce the background for the Flavor-Changing Neutral Current studies
- Kaitlin Salyer from BU Group created slides for the steps of the cuts, definition of control region and specific goals

Familiarizing with the Coffea framework with Awkward Arrays

- Selected cuts based on the slides with reference of coffea examples
- Some remaining redundancy in the code
- Plotting 2D ratio histogram took more time than expected

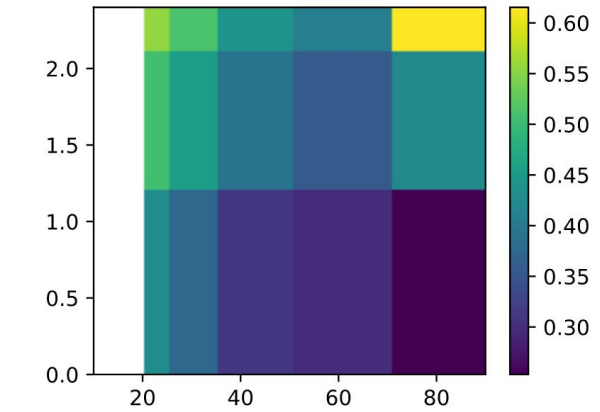
Fake Rates Analysis for Coffea-Casa

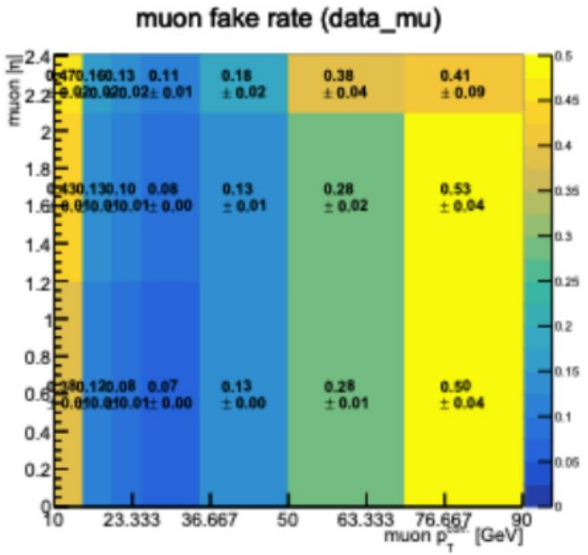
Continuing work:

- Test the fake rates on signal region
- BU graduate student to build on current work with additional physics cuts
- Performance benchmarking

```
Tight_Muon.divide(Loose_Muon)
```

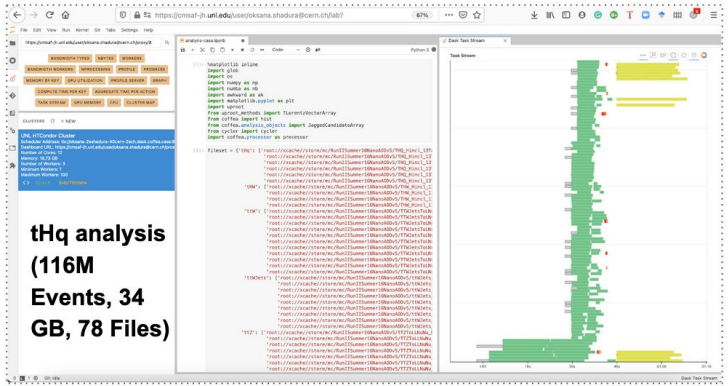
total count: 6.343185609032659, **metadata**: {}



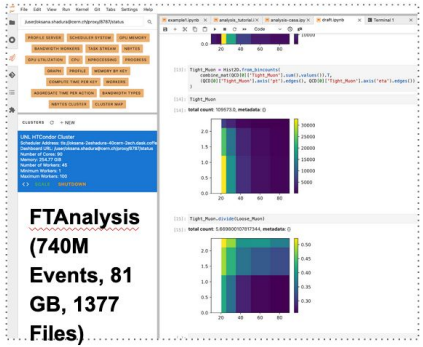


Fake Rates Analysis for Coffea-Casa

Inviting first users



Mat Adamec
(UNL undergrad)



Zora Che
(BU CS
undergrad)

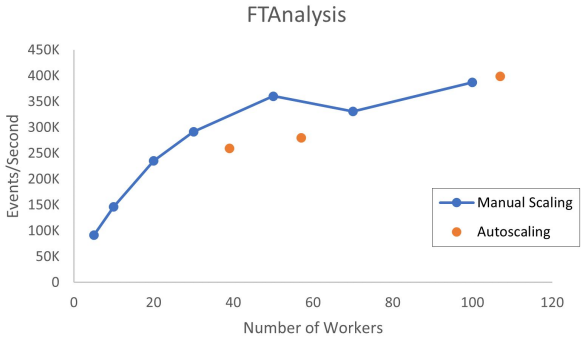
Top quark analyses using the Coffea framework
https://github.com/TopEFT/topcoffea

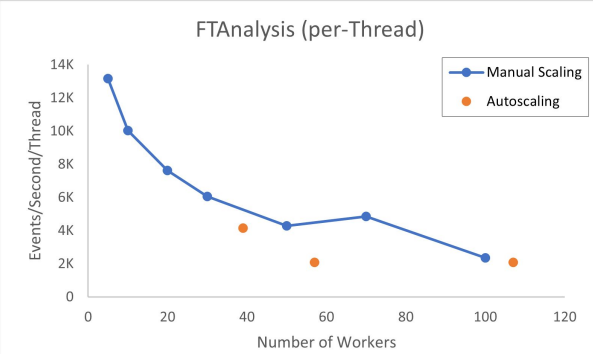
Users already running their analysis on Coffea-casa:

- Some of the examples are **done by undergraduate students**
- **Approachable** (even with all complexity of system behind) and **interactive**

Coffea-casa: an analysis facility prototype' vCHEP plenary

Fake Rates Analysis for Coffea-Casa





Performance on Dask

- 912 files on xcache, ~ 720 M events, 74 Gigabytes read

(Study by Mat Adamec: see more details [Coffea-casa: an analysis facility prototype' vCHEP plenary])

Notes on Coffea-Casa Analysis Conversion

Plotting Issues

- ROOT has some custom histograms that is not in coffea.hist. Current replication relied on a custom package
  - Dependent on this specific analysis

Dasgoclient Request

- Would be convenient to add a new feature to allow physicists to search for root file names on coffea-casa using dasgoclient within its terminal interface
- *dasgoclient* can be accessed after *voms-proxy-init* with valid certificate

TopCoffea

- Topcoffea PR

Steps in conversion

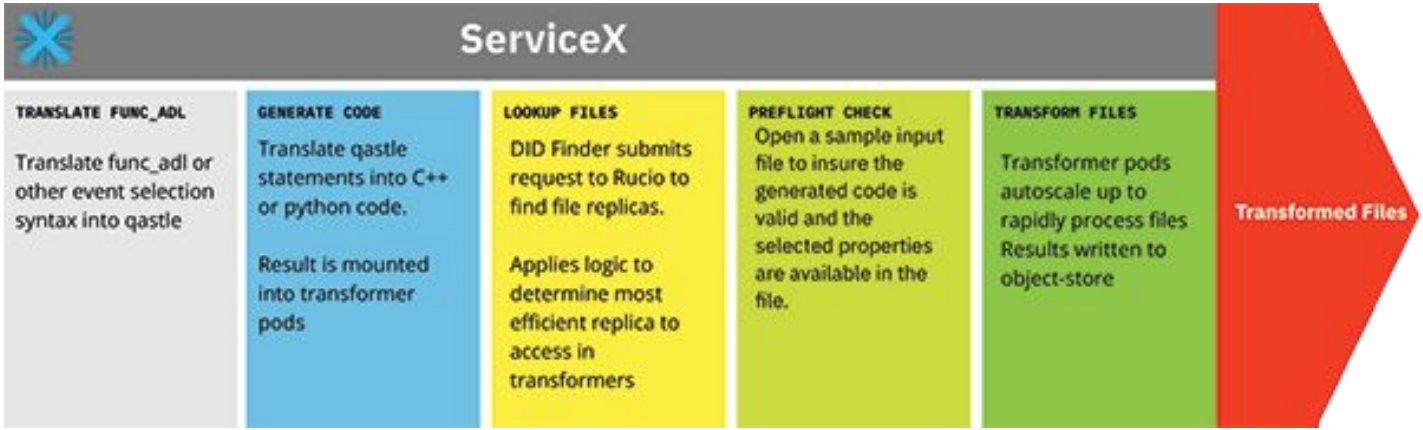
- Coffea based repository
- Adapted the command line input of chosen configuration to a notebook
- Notebook functions generate dictionary from configuration which is fed to skeleton processor class
- Experienced similar issues for plotting, added new notebook

Installing users' custom package for Dask workers is a working issue

- Currently using a dependency installer

ServiceX

on-demand service delivering data from the grid to high energy physics analysts in an easy, flexible, and highly performant manner



- Issue: Option to write transformer results to persistent volume

BenGalewsky commented on Feb 22 Member

Story

As a coffea-casa user I want to have my transformer results written to a POSIX persistent volume so other tools in the facility can work with the data

Assumptions

1. New `result-destination` option: `volume`
2. New settings in `values.yaml` for helm chart

- transformer.persistence.existingClaim
- transformer.persistence.storageClass
- transformer.persistence.size
- transformer.persistence.annotations
- transformer.persistence.subdir

3. If existingClaim is None, TransformerManager in app will create a read-write-many PVC
4. The new or existing PVC will be mounted into each of the transformer pods
5. Results will be written to a directory under the PVC mount/subdir directory named after the transform request UID

Acceptance Criteria

1. Given I have a transform request with `result-destination: volume` and I have provided a valid persistent volume claim when I submit the transform then it should run without errors and the transform files are available in the provided `subdir` under a directory named after the transform request's UID

👍 1 🎉 1 🚀 1

ServiceX - Transformer Result to POSIX FS

ssl-hep / **ServiceX** PR

Installed in helm chart in development space

ssl-hep / **ServiceX_App** PR

Flask app deployed by helm chart, can start another pod deployment using the docker image of Uproot Transformer

ssl-hep / **ServiceX_transformer** PR

Builds a transformer that writes results to indicated result destination

ssl-hep / **ServiceX_Uproot_Transformer** PR

Uproot version of the transformer

ServiceX - Transformer Result to POSIX FS

ssl-hep / ServiceX

PR

Installed in helm chart in development space

- Changed configmap to correspond to new choices in the values.yaml configuration for the deployment
- Updated RBAC role and rolebinding rules for ability to generate persistent volume claims when no existing claim is given
- Updated default values.yaml configuration, and added an example configuration using volume as result destination
- In developing space, installed a local helm chart with the above changes

ServiceX - Transformer Result to POSIX FS

ssl-hep / **ServiceX_App** PR

Flask app deployed by helm chart, can start another pod deployment using the docker image of Uproot Transformer

- In transformation manager when given valid persistent volume claim, generate transformation pod using PVC binding to PV
- In transformation manager dynamically generate PVC with specified storage class for the transformation pod binding to PV when no claim is given
- Passed subdirectory to the uproot transformer for writing results
- In developing space, app image is pulled from Docker Hub zche/servicex_app

ServiceX - Transformer Result to POSIX FS

ssl-hep / **ServiceX_transformer** PR

Builds a transformer that writes results to indicated result destination

ssl-hep / **ServiceX_Uproot_Transformer** PR

Uproot version of the transformer

- Added new parsing option in the base transformer
- Imported the updated base transformer locally in uproot transformer
    - Once the base transformer library is updated, local temporary import should be reverted
- Updated output path in uproot transformer
- In developing space, when a json request is submitted to the app via Postman, app invokes transformer by pulling from image zche/servicex_func_adl_uproot_transformer

ServiceX - Transformer Result to POSIX FS

Status: Transformer pods are started and running for both the case with valid claim and no claim

Development Process

- Building Docker images on UNL Anvil, developing in a custom namespace on cmsaf

Learnings

- Understanding Kubernetes and helm interaction
  - Details such as configmap doesn't get updated after helm upgrade if it wasn't deleted
- Iteratively testing when working on an application with many components
- Finding a debugging friendly environment and try to yak shave less
- Creating workflows with remote development (automating part of the git pull and image push)

Thank you!

Thank you to the coffea-casa and ServiceX teams, and especially to Oksana, Ben, and Mat for your sustained support.