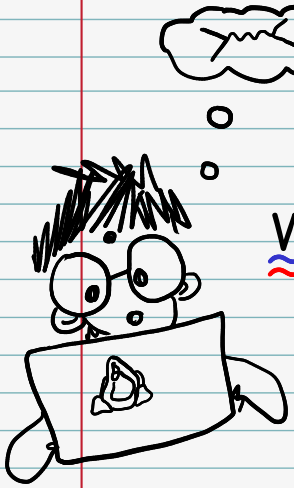


What I Did This Summer!

Analysis Grand Challenge 2024

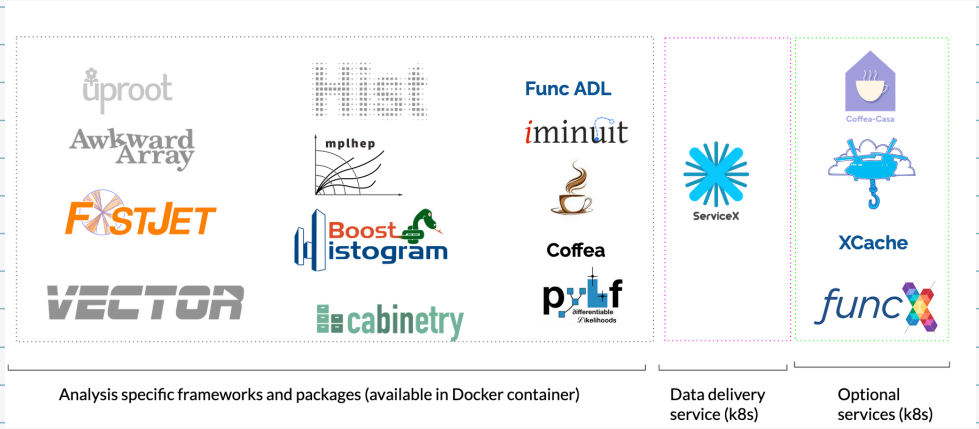
11 September 2024

Sneha Vireshwar Dixit
University of Nebraska-Lincoln

PHYSICS AND ASTRONOMY

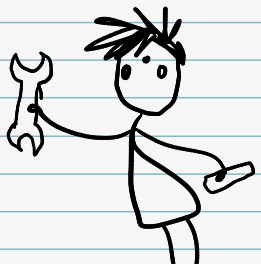
Analysis Grand Challenge (AGC) by IRIS-HEP

- Optimize physics analysis end-to-end
- Develop cyber infrastructure to prepare for HL-LHC

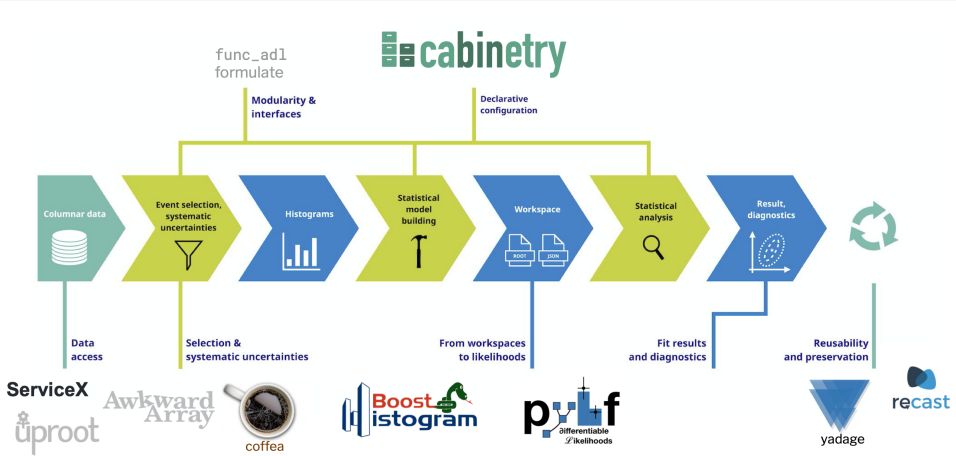


New Tools!

- User Interfaces, eg. Jupyter
- Data access
- Event Selection
- Histogramming / summary statistics
- Statistical model building / fit making
- Reinterpretation / analysis preservation



AGC Pipeline 2024



How do we do this?

1. Physics analysis w. OpenData (like LHC physics analyses)
2. Reference Implementation - conduct analysis with AGC workflow tools at scale.

} successful demo of AGC pipeline

Our sample notebook needs :

- Successful implementation of physics analysis.
- Up-to-date utilization of tools in AGC pipeline.
- Clearly written + commented code which is easily reproducible.

} easy-to-follow example for users

- Mechanism to record performance metrics.

} important for testing/development

Implementation

config/utils

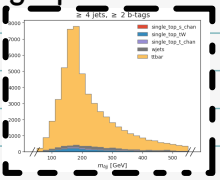
```
config = {
    "global": {
        # ServiceX: ignore cache with repeated queries
        "SERVICEX_IGNORE_CACHE": False,
        # analysis facility: set to "coffea_casa" for coffea-
        "AF": "coffea_casa",
        # number of bins for standard histograms in processor
        "NUM_BINS": 25,
        # lower end of standard histograms in processor
        "BIN_LOW": 50,
        # upper end of standard histograms in processor
        "BIN_HIGH": 550,

    },
    "benchmarking": {
        # chunk size to use
        "CHUNKSIZE": 250000,
        # read files from public EOS (thanks to the CMS DPOA te
        # note that they are likely only available temporarily
        # and not part of an official CMS Open Data release
        "INPUT_FROM_EOS": False,
        # prefix for URIs for ATLAS-style xcache use
        # e.g. "root://xcache.af.uchicago.edu//" for UChicago
        "XCACHE_ATLAS_PREFIX": None,
        ### metadata to propagate through to metrics ###
        # "ssl-dev" allows for the switch to local data on /d
        "AF_NAME": "coffea_casa",
        # currently has no effect
        "SYSTEMATICS": "all",
        # does not do anything, only used for metric gatherin
        "CORES_PER_WORKER": 1,
        # scaling for local setups with FuturesExecutor
        "NUM_CORES": 100,
        # only I/O, all other processing disabled
        "DISABLE_PROCESSING": False,
```

(customize options)

processor

```
class TtbarAnalysis(processor.ProcessorABC):
    def __init__(self, use_inference, use_triton):

        # initialize dictionary of hists for signal and control region
        self.hist_dict = {}
        for region in ["4j1b", "4j2b"]:
            self.hist_dict[region] = (
                hist.Hist.new.Reg(utils.config["global"]["NUM_BINS"],
                                  utils.config["global"]["BIN_LOW"],
                                  utils.config["global"]["BIN_HIGH"],
                                  name="observable",
                                  label="observable [GeV]")
                .StrCat([], name="process", label="Process", growth=True)
                .StrCat([], name="variation", label="Systematic variation", growth=True)
                .Weight()
            )

        self.cset = correctionlib.CorrectionSet.from_file("corrections.json")
        self.use_inference = use_inference

        # set up attributes only needed if USE_INFERENCE=True
        if self.use_inference:

            # initialize dictionary of hists for ML observables
            self.ml_hist_dict = {}
            for i in range(len(utils.config["ml"]["FEATURE_NAMES"])):
                self.ml_hist_dict[utils.config["ml"]["FEATURE_NAMES"][i]] = (
                    hist.Hist.new.Reg(utils.config["global"]["NUM_BINS"],
                                      utils.config["ml"]["BIN_LOW"][i],
                                      utils.config["ml"]["BIN_HIGH"][i],
                                      name="observable",
                                      label=utils.config["ml"]["FEATURE_DESCRIPTIONS"][i])
                    .StrCat([], name="process", label="Process", growth=True)
                    .StrCat([], name="variation", label="Systematic variation", growth=True)
                    .Weight()
                )

            self.use_triton = use_triton

    def only_do_IDs(self, events):
        for branch in utils.config["benchmarking"]["IO_BRANCHES"][
            utils.config["benchmarking"]["IO_FILE_PERCENT"]
        ]:
            if "_" in branch:
                split = branch.split("_")
                object_type = split[0]
                property_name = "_".join(split[1:])
                ak.materialized(events[object_type][property_name])
            else:
                ak.materialized(events[branch])
        return {"hist": {}}
```

graphs



metrics


```
▾ root:
    bytesread: 7748406064
  ▸ columns: [] 24 items
    entries: 47996231
    processtime: 5178.281707763672
    chunks: 239
    walltime: 1279.532303282991
    num_workers: 50
    af: "coffea_casa"
    dataset_source: "UNL"
    use_dask: true
    use_servicex: false
    systematics: "all"
    n_files_max_per_sample: 5
    cores_per_worker: 1
    chunksize: 200000
    disable_processing: false
    io_file_percent: "4.1"
    agc_version: "main"
    use_inference: true
    use_triton: false
```


(report performance)

Coffea 0.7 vs Coffea 2024

Coffea 0.7

- Pandas + Numpy based.
- Isolated chunks (nonrepresentative mean)
- Can access only locally stored files.

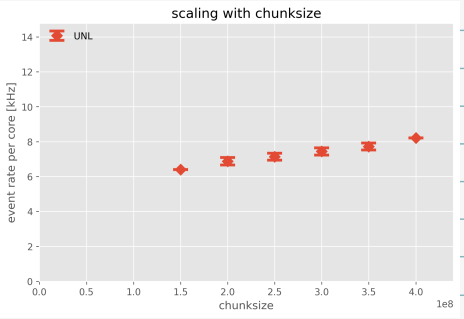
Coffea 2024

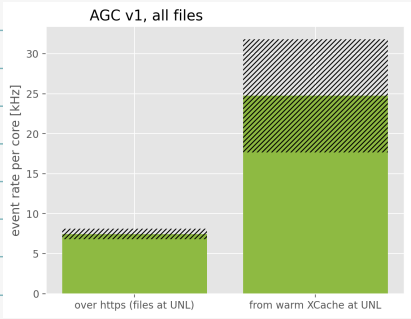
- Fully Awkward based.
- Non isolated chunks (representative mean)
- Can access local + external files.
- Allows partial extraction of files.

Coffea 2024 is not backwards compatible! We need to build a new sample implementation.

Spring 2024

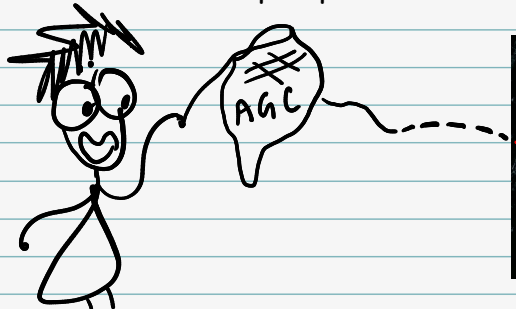
- CMS $t\bar{t}$ sample implementation Coffea 0.7 $\rightarrow$ Coffea 2024.
- Implemented + tested new tools - DASK Client, XCache.
- Measured + compared efficiency of new tools.



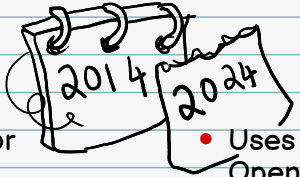


Tasks for Summer 2024 :

- ATLAS HZZ implementation Coffea 0.7$\rightarrow$ Coffea 2024.
- Build utils file for easy configuration.
- Build metrics file to report performance data.



ATLAS vs. CMS implementations :



ATLAS

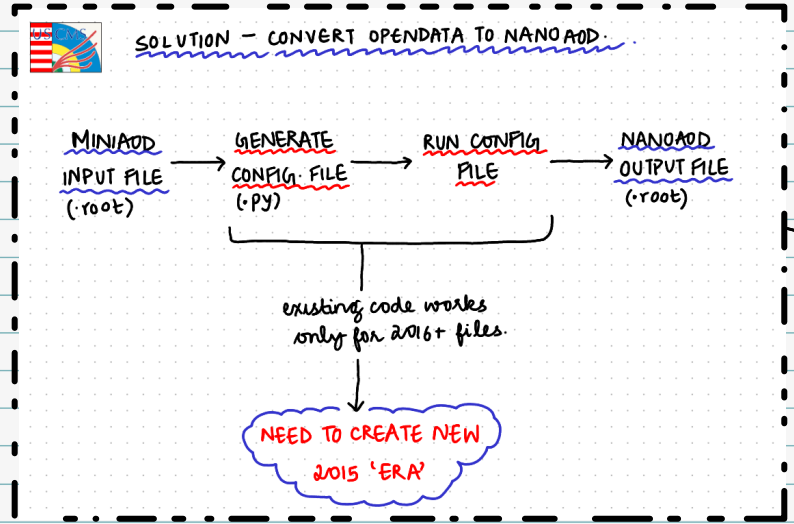
- Uses MiniAOD datatype for OpenData.
  - Older, larger files, slower
  - Less native compatibility with Coffea
- Uses BaseSchema (unchanged from root files)

CMS

- Uses NanoAOD datatype for OpenData.
  - Newer, smaller files, faster
  - More native compatibility with Coffea
- Uses NanoAODSchema (more intuitive to me)

ATLAS implementation using AGC pipeline has less documentation to rely on (mainly a CMS effort)

Fun Fact...



worked on generating CMS Opendata NanoAODs used as part of 2022 USCMS PURSUE with Nick Smith and Oksana Shadura @ UNL

(presentation slide from 2022)

Results

```python
class HZZProcessor(processor.ProcessorABC):
    def __init__(self):
        pass

    def process(self, events):
        dataset_category = events.metadata['dataset']
        isRealData = "genWeight" not in events.fields
        sumw = 0. if isRealData else ak.sum(events.genWeight, axis=0)
        cutflow = {"start": ak.num(events, axis=0)}

#         if isRealData:
#             events = events[
#                 corrections.lumimask(events.run, events.luminosityBlock)
#             ]
#             cutflow["lumimask"] = ak.num(events, axis=0)

        # apply a cut to events, based on lepton charge and lepton type
        events["lepfiltered"] = events[lepton_filter(events.lep_charge, events.lep_type)]

        # construct lepton four-vectors
        leptons = ak.zip(
            {"pt": events.lep_pt,
             "eta": events.lep_eta,
             "phi": events.lep_phi,
             "energy": events.lep_E}, #not lep_energy
            with_name="Momentum4D",
        )

        cutflow["ossf"] = ak.num(events, axis=0)

        mllll = (leptons[:, 0] + leptons[:, 1] + leptons[:, 2] + leptons[:, 3]
        ).mass / 1000

        # creat histogram holding outputs, for data just binned in m4l
        mllllhist_data = hist.Hist.new.Reg(
            num_bins,
            bin_edge_low,
            bin_edge_high,
            name="mllll",
            label="$\mathrm{m_{4l}}$ [GeV]",
        ).Weight()  # using weighted storage here for plotting later, but not needed

        # three histogram axes for MC: m4l, category, and variation (nominal and
        # systematic variations)
        mllllhist_MC = (
            hist.Hist.new.Reg(
                num_bins,
                bin_edge_low,
                bin_edge_high,
```

wrote ATLAS HZZ processor

added metrics


```
▾ root:
    walltime: 1632.4484504340217
    num_workers: 100
    af: "coffea_casa"
    systematics: "all"
    n_files_max_per_sample: 1
    cores_per_worker: 1
    chunksize: 250000
    io_file_percent: "4.1"
    agc_version: "main"
```


https://github.com/iris-hep/calver-coffea-agc-demo/pull/4

(pull request on Calver Coffea demo)

Questions?

(special thanks to Ken Bloom, Oksana Shadura, Alexander Held)