COFFEA (ˈkôfē):
Columnar Object Framework For Effective Analysis

Matteo Cremonesi [FNAL]
On behalf of the COFFEA team

June 1st, 2021
Scientific Method in HEP

• From an hypothesis derive predictions, test the predictions in the real world
• In HEP: generate simulations based on theory, compare simulations with data

Discovery of the Higgs boson at the Large Hadron Collider

- Black dots: recorded data
- Blue shape: simulation
- Red shape: simulation of new theory (in this case the Higgs)
Organizational Aspects

• Large collaborations:
  - Thousands of particle physicists from hundreds institutes and universities from more than 40 countries

• Central production:
  - Large volume of simulation/data
    • Billions of events
  - Grid computing model
    • 300k+ CPU cores over 70+ sites spread all over the world

• Individual analysis
  - 100+ teams, all using different analysis software
  - Almost 1:1 correspondency between published papers and PhD students
    • Analysis are usually lead by the most inexperienced
Data Volume

• Extract physics results require to handle/analyze a large datasets
  - Hundreds of PBs
  - Will increase to EBs in the next decade

• Inefficiencies result in:
  - Waste of storage space
  - Large time-to-insight
    • ~ days to weeks

• Already unsustainable
Inefficiencies of a Typical Analysis Code

• Waste of storage space
  - Each step of the analysis workflow writes intermediate output

• Large time to insight
  - Each step of the analysis workflow takes significant time to be completed

• Why?
  - Same data representation and computing paradigm of central production are used, but for individual applications

Centrally produced ROOT files (~100 TB)
- Ntupling
  - ~4 x year
    - on grid, ~1 week

Group ROOT ntuples (~100 TB)
- Skimming & Slimming
  - ~1 x week
    - on batch, ~2 days

Analysis ROOT ntuples (~10 TB)
- Analysis
  - Several a day
    - on laptop, ~few minutes

plots and tables
Event Loop Analysis of ROOT Files

• File-based data representation in ROOT format
  - Each file is a collection of events

• Event loop analysis of a ROOT file
  - Load relevant values for a specific event into local variables
  - Evaluate several expression
  - Store derived values in new ROOT files
    • Duplicating the variables that were not manipulated, but that will be needed later on
  - Repeat
What is COFFEA?

A package in the scientific python ecosystem that provides a user interface for columnar analysis in HEP
Columnar Analysis: A Paradigm Shift

• Columnar data representation
  - Load relevant values for many events into contiguous columns
  - Events are rows

• Columnar analysis
  - Evaluate array programming expressions
    • Simple vector operations to act on an entire columns at once
    • No explicit loops
  - Store derived values in new contiguous columns
    • No new files written on disk
Main Benefits of COFFEA

• Ease of use and readability
  - Column analysis is a higher-level description of manipulations than an event loops
  - Code is human-readable

• Efficient code
  - Columnar analysis aligns with strengths of modern CPUs
  - Make it easy to write computationally efficient code

• Community support
  - Take advantage of off-the-shelf tools from data science
What COFFEA Provides

• Physicist friendly tools for column based analysis
  - Implements typical recipes needed to operate on NanoAOD-like ntuples
    • histogramming, plotting, and look-up table functionalities for weights and MC corrections
  - Supplies facilities for horizontally scaling

• Currently in https://github.com/CoffeaTeam/coffea
  - pip install coffea

• Realized using:
  - Scientific python ecosystem:
    • numpy, numba, scipy, matplotlib
  - Awkward-array:
    • array programming primitives to handle “Jagged Arrays”
HEP Data in Columnar Form: Jagged Arrays

HEP data is not “rectangular”:

- Cannot be represented as a flat table
  - different numbers of muons/electrons/jets etc in each event
- Can be represented as arrays of variable-length (jagged arrays)
  - https://github.com/scikit-hep/awkward-array

Muon pt: table

<table>
<thead>
<tr>
<th>Event</th>
<th>Muon pt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event 1</td>
<td>40.2, 25.6, 10.2</td>
</tr>
<tr>
<td>Event 2</td>
<td>71.1, 35.7</td>
</tr>
<tr>
<td>Event 3</td>
<td>52.3</td>
</tr>
<tr>
<td>Event 4</td>
<td>34.5, 15.7</td>
</tr>
</tbody>
</table>

Muon pt: jagged array

```
[ [ 40.2, 25.6, 10.2 ], [ 71.1, 35.7 ], [ 52.3 ], [ 34.5, 15.7 ] ]
```
To apply selections, one uses a *mask*:

\[
\begin{align*}
\text{mu\_pt} & = \begin{bmatrix}
40.2 & 25.6 & 10.2 \\
71.1 & 35.7 & 52.3 \\
34.5 & 15.7
\end{bmatrix} \\
\text{mask} = (\text{mu\_pt} > 30) & = \begin{bmatrix}
T & F & F \\
T & T & T \\
T & F
\end{bmatrix} \\
\text{mu\_pt}[\text{mask}] & = \begin{bmatrix}
40.2 & 71.1 & 35.7 & 52.3 & 34.5
\end{bmatrix}
\end{align*}
\]

Note that there was no explicit for loop over the events, and the mask was applied to each muon in each event.
**Coffea processor**

- Abstraction to encapsulate analysis code
- Keep it separate from input column delivery and output reduction (i.e. histogramming)
- Defines the analysis selections, weights, and output histograms
  - Input: dataframe of awkward arrays
  - Output: histograms, counters, small arrays

```python
from coffea import hist, processor

class MyProcessor(processor.ProcessorABC):
    def __init__(self, flag=False):
        self._flag = flag
        self._accumulator = processor.dict_accumulator({
            # Define histograms
        })

    @property
def accumulator(self):
        return self._accumulator

    def process(self, df):
        output = self.accumulator.identity()

        # PHYSICS GOES HERE

        return output

    def postprocess(self, accumulator):
        return accumulator

p = MyProcessor()
```
Coffea executor

• Handles the interaction with the column delivery mechanism
  - communicating with back-end scale-out systems
    • Dask, Spark, Parsl, HTCondor
• Once defined, your processor can be passed to different executors with a single line change
NanoEvents

- Coffea utility to wrap the CMS NanoAOD format into a single awkward array, with:
  - appropriate object methods, such as Lorentz vector methods
  - cross references
  - nested objects
- Instantiate an event object reading a NanoAOD file:

```python
import awkward as ak
from coffea.nanoevents import NanoEventsFactory, NanoAODSchema

fname = "https://raw.githubusercontent.com/CoffeaTeam/coffea/master/tests/samples/nano_dy.root"
events = NanoEventsFactory.from_root(fname, schemaclass=NanoAODSchema).events()
```

- Access the energy of the GenJets:

```python
events.GenJet.energy
<Array [[217, 670, 258], ... 16], [76.9]] type='40 * var * float32'>
Processor Code Examples

- Python allows very flexible interface, under-the-hood data structure is columnar

- One line of code to define analysis objects with NanoEvents:

  ```python
  electrons = events.Electron
  ```

- One line of code to define the mask to select tight electrons:

  ```python
  electronSelectTight = ((electrons.pt>35) &
  (abs(electrons.eta)<2.1) &
  (abs(electrons.eta) < 1.4442) | (abs(electrons.eta) > 1.566) &
  (electrons.cutBased>=4)
  )
  ```

- One line of code to select tight electrons from all events - **no** explicit for loop over electrons!

  ```python
  tightElectron = electrons[electronSelectTight]
  ```

- One line of code to define events passing tight electron requirements - **no** explicit for loop over events!

  ```python
  eventSelection = (ak.num(tightElectron) ==1)
  ```
Using COFFEA for CMS Analysis

• Tens of analysis in CMS have already adopted COFFEA
  - User community is growing, ~40/50 people contributing at some extent
  - Some analyses go from centrally produced NanoAOD directly to plots, with no usage of standard tools

• Results
  - No intermediate output written on disk
    • Directly from inputs to plots
  - Analysis turn-around time reduced by more than two order of magnitudes
    • From days to hours
Conclusions

• An innovative tool has been developed for data analysis in particle physics
  - It pioneers the utilization of columnar analysis

• It addresses the main issues that affect the current way of doing analysis
  - Shortage of disk space
  - Long time-to-insight, limited interactivity

• It is a real-world solution
  - It takes into account the constraints, does not require organizational changes or additional resources
  - Already used for publishable (or already published) results
## Documentation

- Coffea documentation
  - [https://coffeateam.github.io/coffea/](https://coffeateam.github.io/coffea/)
- Simple examples (with comments) for IRIS-HEP benchmarks*
  - [https://github.com/mat-adamec/coffea-benchmarks/tree/master/benchmarks](https://github.com/mat-adamec/coffea-benchmarks/tree/master/benchmarks)
  - *Set of tasks designed to demonstrate and compare usability against other analysis systems
- Coffea users egroup: [cms-coffea-users.cern.ch](https://cms-coffea-users.cern.ch)
  - Biweekly coffea users meeting on Mondays
Backup
Baby Ecosystem

• Coffea serves as incubator for rapid prototyping of missing pieces in our ecosystem. Good abstractions are factored out.