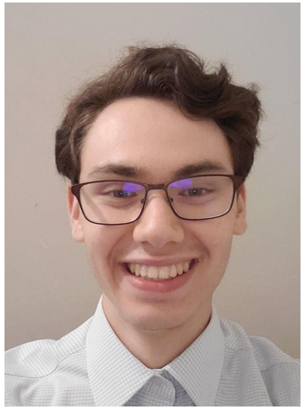
Development of Experiment-Specific Data Schemas for Coffea



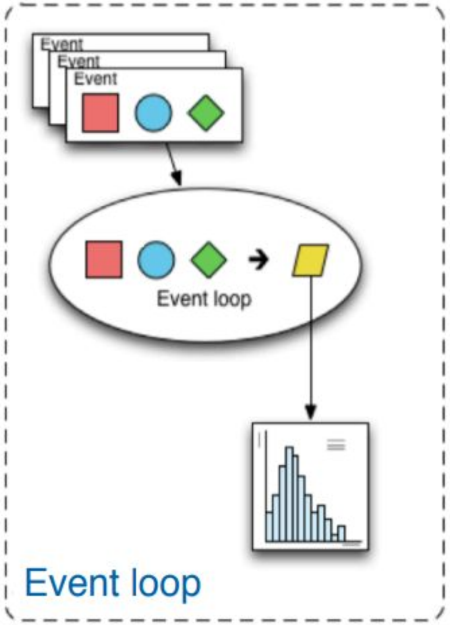
By Sam Kelson

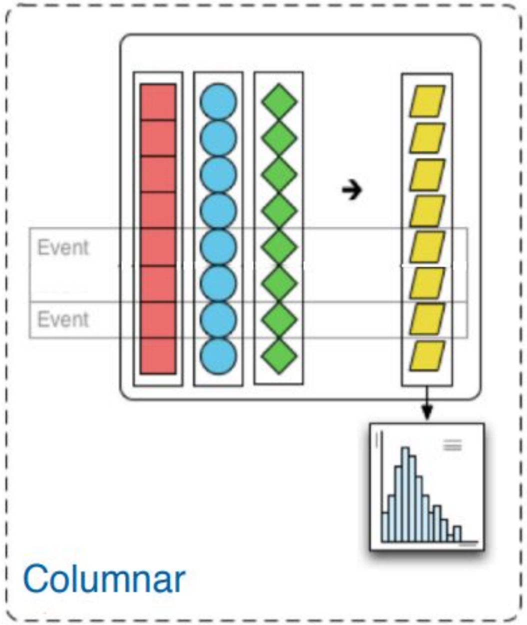
A little about me …

- IRIS-HEP Mentors: Lindsey Gray, Nick Smith, Matthew Feickert, Giordon Stark

Intro: What is Coffea?

- **C**olumnar **O**bject **F**ramework **F**or **E**ffective **A**nalysis



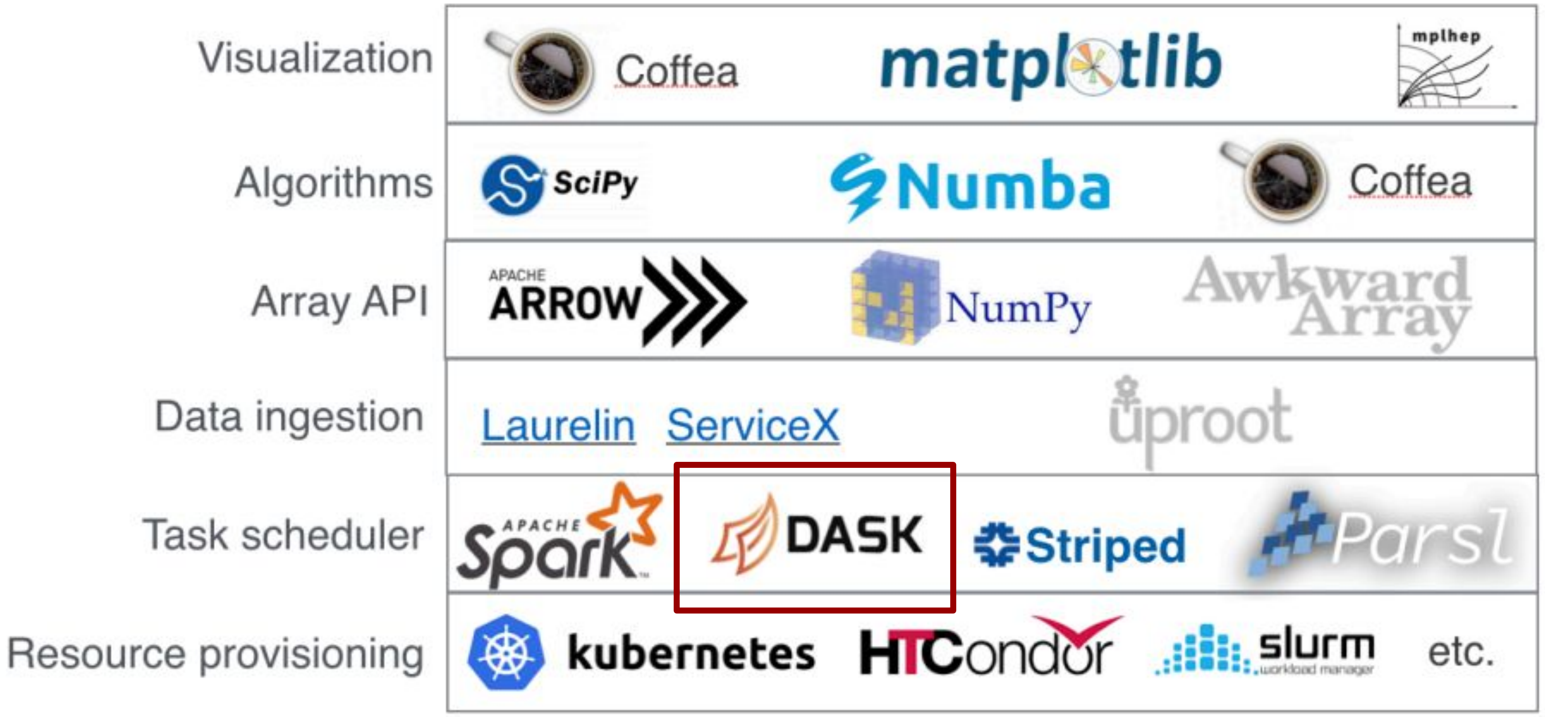


ex: ROOT’s RDataFrame

ex: coffea

img credit: Nick Smith

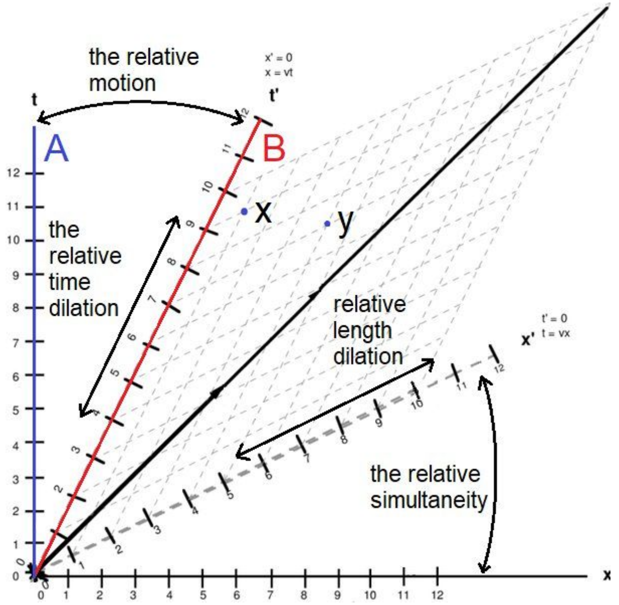
Intro: Coffea in the PyHEP Ecosystem



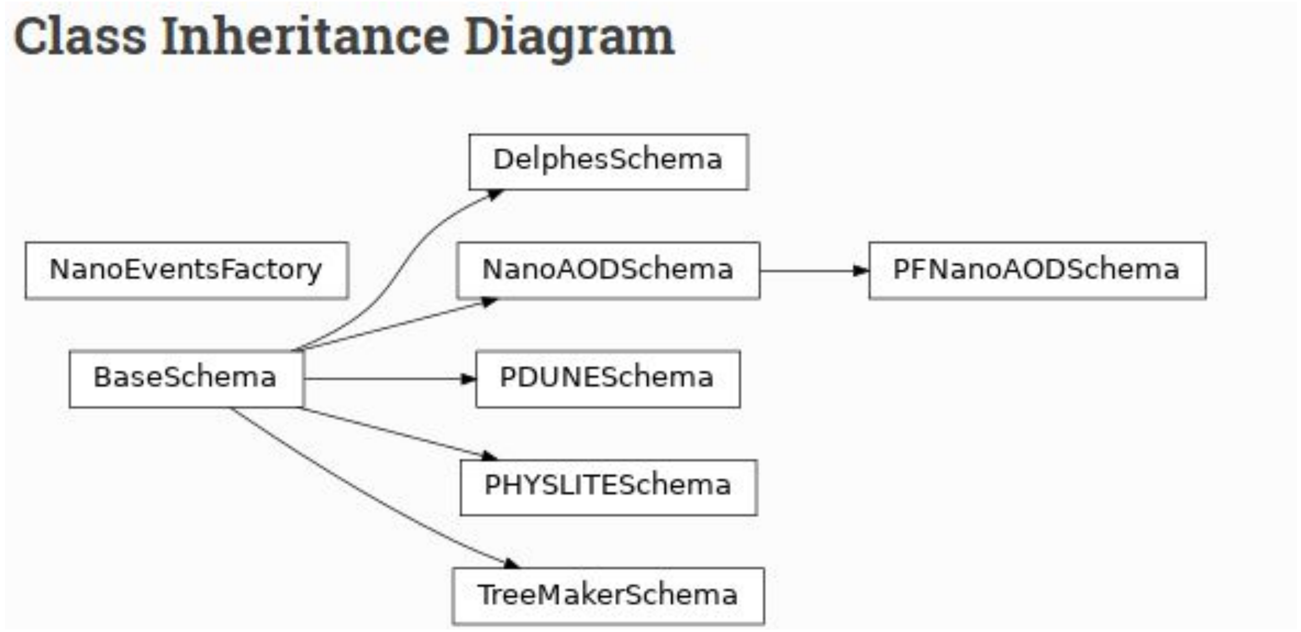
Visualization	Coffea	matplotlib	mplhep	
Algorithms	SciPy	Numba	Coffea	
Array API	APACHE ARROW	NumPy	Awkward Array	
Data ingestion	Laurelin	ServiceX	uproot	
Task scheduler	APACHE Spark	DASK	Striped	Parsl
Resource provisioning	kubernetes	HTCondor	slurm workload manager	etc.

Intro: Coffea uses Awkward Array Behaviors

- A data structure is defined by the information it encodes and the ways in which it can be used.
- Ex: Lorentz vectors

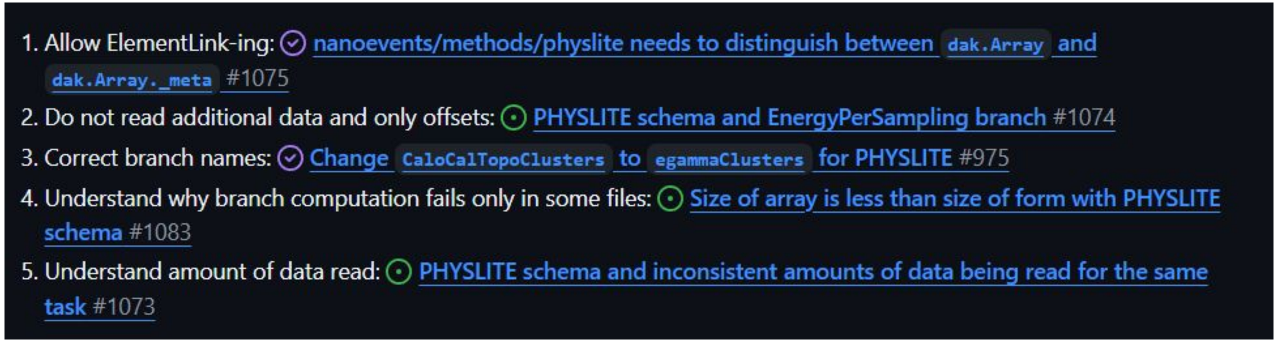


Goal: Experiment-Independent Functionality for Coffea



PHYSLITE Updates

- Mostly bug hunting…

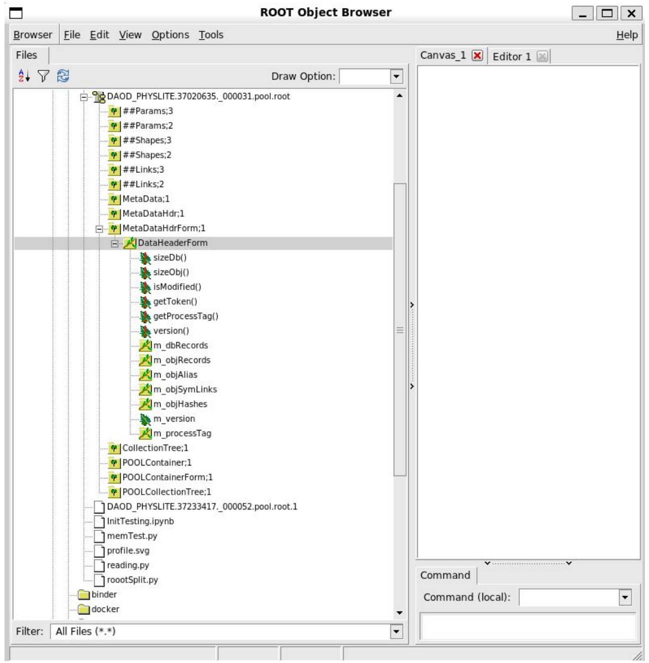


1. Allow ElementLink-ing: nanoevents/methods/physlite needs to distinguish between `dak.Array` and `dak.Array._meta` #1075
2. Do not read additional data and only offsets: PHYSLITE schema and EnergyPerSampling branch #1074
3. Correct branch names: Change `CaloCalTopoClusters` to `egammaClusters` for PHYSLITE #975
4. Understand why branch computation fails only in some files: Size of array is less than size of form with PHYSLITE schema #1083
5. Understand amount of data read: PHYSLITE schema and inconsistent amounts of data being read for the same task #1073

- Also improved element linking

Incoming New Features: Improved Metadata Reading

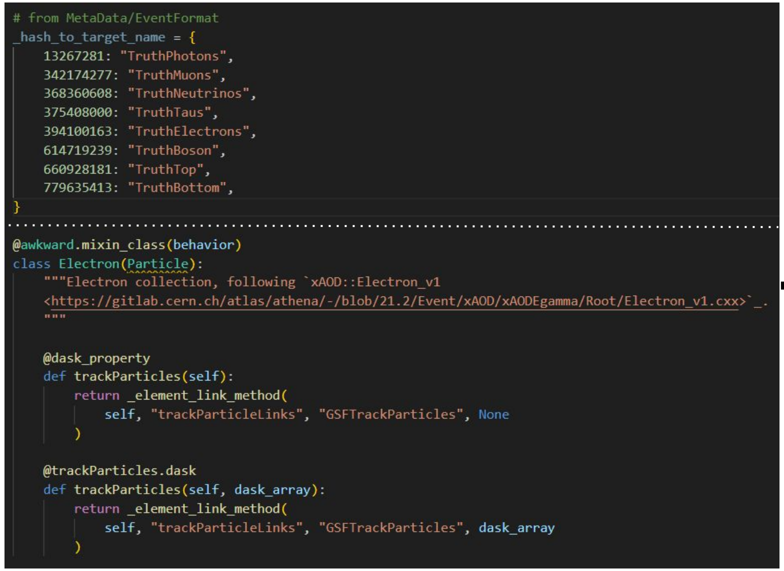
- Uproot can’t read everything, including metadata
- Metadata can help with element linking and object identification



```
DAOD_PHYSLITE.37020635._000031.pool.root
  ##Params;3
  ##Params;2
  ##Shapes;3
  ##Shapes;2
  ##Links;3
  ##Links;2
  MetaData;1
  MetaDataHdr;1
  MetaDataHdrForm;1
    DataHeaderForm
      sizeDb()
      sizeObj()
      isModified()
      getToken()
      getProcessTag()
      version()
      m_dbRecords
      m_objRecords
      m_objAlias
      m_objSymLinks
      m_objHashes
      m_version
      m_processTag
  CollectionTree;1
  POOLContainer;1
  POOLContainerForm;1
  POOLCollectionTree;1
DAOD_PHYSLITE.37233417._000052.pool.root.1
InitTesting.ipynb
memTest.py
profile.svg
reading.py
roootSplit.py
binder
docker
```

What it Could Do:

Hard Coded:



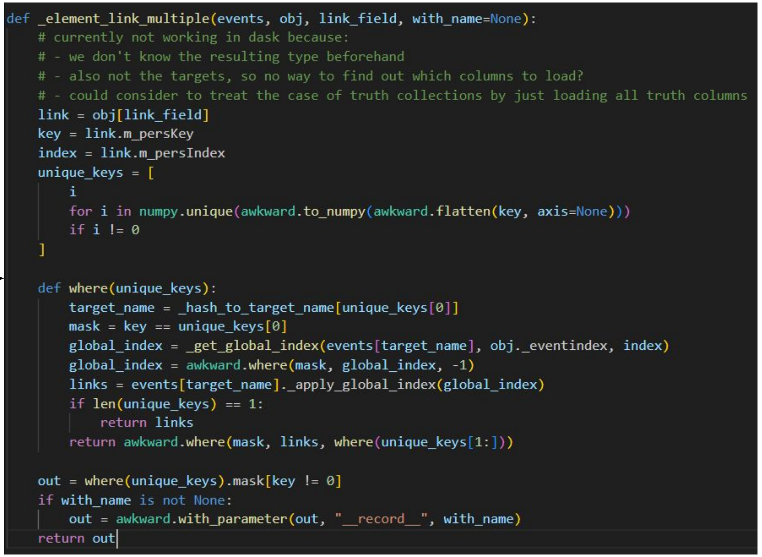
```
# from MetaData/EventFormat
_hash_to_target_name = {
    13267281: "TruthPhotons",
    342174277: "TruthMuons",
    368360608: "TruthNeutrinos",
    375408000: "TruthTaus",
    394100163: "TruthElectrons",
    614719239: "TruthBoson",
    660928181: "TruthTop",
    779635413: "TruthBottom",
}
```

```
@awkward.mixin_class(behavior)
class Electron(Particle):
    """Electron collection, following `xAOD::Electron_v1
    <https://gitlab.cern.ch/atlas/athena/-/blob/21.2/Event/xAOD/xAODEgamma/Root/Electron_v1.cxx>`_.
    """

    @dask_property
    def trackParticles(self):
        return _element_link_method(
            self, "trackParticleLinks", "GSFTrackParticles", None
        )

    @trackParticles.dask
    def trackParticles(self, dask_array):
        return _element_link_method(
            self, "trackParticleLinks", "GSFTrackParticles", dask_array
        )
```

Metadata Reading:



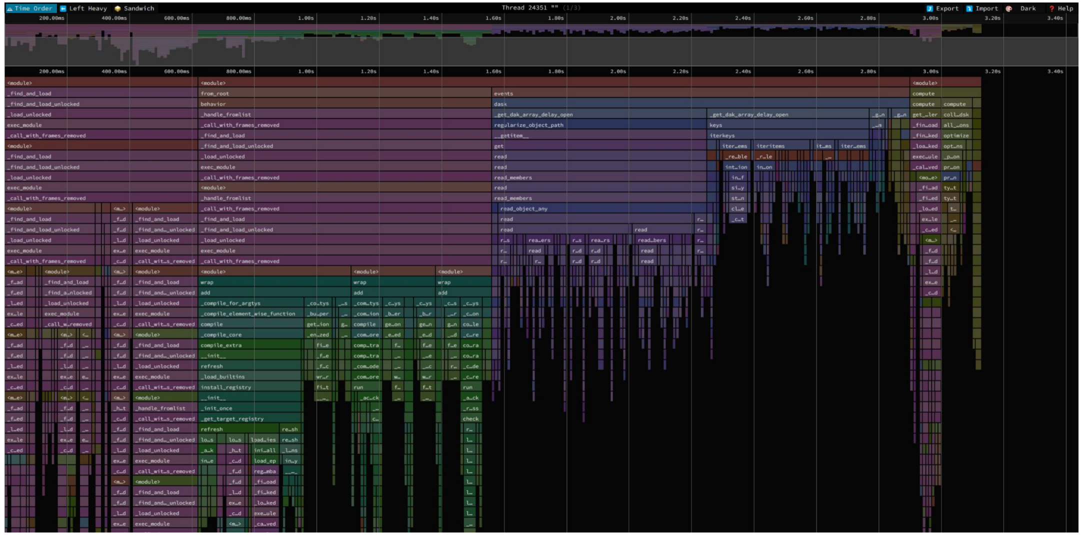
```
def _element_link_multiple(events, obj, link_field, with_name=None):
    # currently not working in dask because:
    # - we don't know the resulting type beforehand
    # - also not the targets, so no way to find out which columns to load?
    # - could consider to treat the case of truth collections by just loading all truth columns
    link = obj[link_field]
    key = link.m_persKey
    index = link.m_persIndex
    unique_keys = [
        i
        for i in numpy.unique(awkward.to_numpy(awkward.flatten(key, axis=None)))
        if i != 0
    ]

    def where(unique_keys):
        target_name = _hash_to_target_name[unique_keys[0]]
        mask = key == unique_keys[0]
        global_index = _get_global_index(events[target_name], obj._eventindex, index)
        global_index = awkward.where(mask, global_index, -1)
        links = events[target_name]._apply_global_index(global_index)
        if len(unique_keys) == 1:
            return links
        return awkward.where(mask, links, where(unique_keys[1:]))

    out = where(unique_keys).mask[key != 0]
    if with_name is not None:
        out = awkward.with_parameter(out, "__record__", with_name)
    return out
```

Whats Next for PHYSLITE

- Performance testing
  - Memory
  - Computation time/usage
  - W / WO Dask



Special Thanks Too:

Lindsey Gray, Nick Smith, Matthew Feickert, Giordon Stark, and Evangelos Kourlitis