Motivation

- Several different code interfaces exist for analyzing HEP data, and new ones are being created.
- Each of these analysis description languages (ADLs) needs to be capable of performing common analysis tasks.
- The most accessible documentation of an ADL is by working examples inspired by real analyses.
- Fair comparison of ADLs requires code samples of each performing the same tasks on the same input.

Index repository

- Ideas for common analysis tasks to use as reference were discussed in the HSF Data Analysis Working Group.
- An initial list of data query challenges or functionality benchmarks was established.
- A new GitHub repository was created to document the functionality benchmarks.
- Contributors have implemented solutions to the challenges with different ADLs.
- The new repository provides an index to these ADLs and links to the implementations of the benchmark tasks.

Language catalog

- The functionality benchmarks have been implemented in the following languages so far:

<table>
<thead>
<tr>
<th>Analysis description language</th>
<th>Programming language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDataFrame</td>
<td>C++</td>
<td>High-level interface within ROOT for declarative analysis</td>
</tr>
<tr>
<td>NAIL (Natural Analysis Implementation Language)</td>
<td>Python</td>
<td>Interface for declarative analysis using RDataFrame as a backend</td>
</tr>
<tr>
<td>groot</td>
<td>Go</td>
<td>Pure Go package that provides read/write access to ROOT files</td>
</tr>
<tr>
<td>coffea (Colomna Object Frame-Work For Effective Analysis)</td>
<td>Python</td>
<td>Builds on numply and awkward-array for columnar data analysis</td>
</tr>
</tbody>
</table>

Functionality benchmarks

- The current list of functionality benchmarks is shown in the table below.
- The right column explains the generic type of functionality being exemplified.
- This will be expanded as new challenges are suggested as GitHub issues to cover missing analysis use cases.

<table>
<thead>
<tr>
<th>Benchmark description</th>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plot the missing E_T of all events.</td>
<td>Loop over events and get an event-level variable.</td>
</tr>
<tr>
<td>Plot p_T of all jets in each event.</td>
<td>Loop over an array in each event.</td>
</tr>
<tr>
<td>Plot p_T of jets with</td>
<td></td>
</tr>
<tr>
<td>Loop over an array and aggregate the results to filter at the event level.</td>
<td></td>
</tr>
<tr>
<td>Plot the missing E_T of events that have at least two jets with p_T &gt; 40 GeV.</td>
<td>Loop on pairs of objects in one collection and do four-vector algebra.</td>
</tr>
</tbody>
</table>

Challenge implementation examples

- An implementation of the first functionality benchmark for each ADL is illustrated below.
- All implementations produce a histogram similar to the one at the bottom-right.

```
coffe [4]

class METProcessor ( processor . ProcessorABC ):
    def __init__(self):
        self . MET_axis = hist . Bin ( "MET\_GeV\" , 50 , 0 , 100 )
        self . accumulator = processor . dict_ accumulator ( [ "MET" , hist . Bin ( "Counts" , self . MET_axis , "MET\_axis" ) ] )

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

def process(self , df):
    output = self . accumulator . identity ( )
    output [ "MET\" , "MET\_GeV\" ] = df . MET
    return output

def postprocess ( self , accumulator ):
    output = processor . run_uproot_job ( files , treename = "Events" , processor_instance = METProcessor ( ) , executor = processor . futures . executor , executor_args = [ "workers" : 4 , chunksize = 50000 ] )
    hist . plot1d ( output [ "MET\" ] . overlay = "dataset" , fillopts = [ "edgecolor" : ( 0 , 0 , 0 , 0.3 ) , "alpha" : 0.8 ] )
```

Index repository

- A list of functionality benchmarks designed to demonstrate how to perform common analysis tasks in an analysis description language has been created.
- Several contributors have written implementations of these benchmark tasks in different analysis languages.
- A repository documenting the functionality benchmarks, as well as indexing analysis languages and their implementations of the benchmarks has been created at https://github.com/iris-hep/adl-benchmarks-index

References


Acknowledgements

This material is based upon work supported by the National Science Foundation under Cooperative Agreement OAC-1836650. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.