FuncADL: Functional Analysis Description Language

Mason Proffitt and Gordon Watts
(University of Washington)

W
UNIVERSITY *of*
WASHINGTON

2021-05-18
vCHEP 2021

iris
hep
Institute for Research & Innovation
in Software for High Energy Physics

Motivation

- Query languages:
  - Database management systems help to address, among other issues[1]:
    - data redundancy
    - data independence
  - A key aspect of database management is query languages, such as SQL
- Functional languages:
  - Functional programming offers several desirable features for physics analyses:
    - Declarative
    - Stateless
    - Lazy
- Both of these concepts lead to more modular code:
  - Insulate analysis code from data storage location and file format
  - Insulate each section of code from other parts of the code

[1] https://opentextbc.ca/dbdesign01/chapter/chapter-3-characteristics-and-benefits-of-a-database/

Interface (front end)

- FuncADL is:
  - a functional query interface
  - modeled after Language INtegrated Query (LINQ[2], part of C#)
  - using Python as a host language
- Queries are built from a set of basic operators like Select, Where, Count, etc.
- Example:
  - To retrieve E_T^{miss} in all events with at least two jets with p_T > 40 GeV:

```
EventDataset(dataset_identifier) \

    .Where(lambda event: event.Jet_pt.Where(lambda pt: pt > 40).Count() >= 2) \

    .Select(lambda event: event.MET_pt)
```

[2] https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/

Interface (front end)

```
EventDataset(dataset_identifier) \
    .Where(
        lambda event: event.Jet_pt.Where(
            lambda pt: pt > 40
        )
        .Count() >= 2
    ) \
    .Select(
        lambda event: event.MET_pt
    )
```

`EventDataset()` yields a sequence of events

`Where()` applies a filter function to each sequence element

`Jet_pt` is a sequence within each event

`Count()` reduces a sequence to an integer (its length)

`Select()` applies a transformation to each sequence element

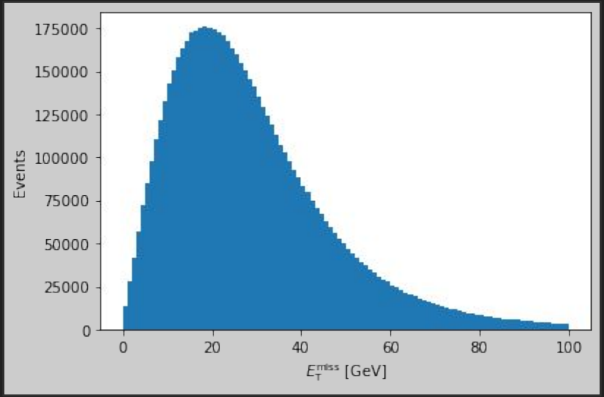
`MET_pt` is a single value in each event

Back end

- Back end translates FuncADL query into appropriate native code for execution on underlying file format
- Code generation is done by traversing the Python abstract syntax tree of the FuncADL query and forming a native representation of each tree node
- Currently three implementations:
  - Uproot back end
    - Generates Python code
    - Can operate on any flat ROOT ntuple
      - For example: CMS NanoAOD or ATLAS DAOD_PHYSLITE
  - xAOD (ATLAS) back end
    - Generates C++ code
  - CMS Run 1 AOD back end
    - Generates C++ code
  - More to come!

Full standalone example

```
>>> from func_adl_uproot import UprootDataset
>>> ds = UprootDataset('root://eospublic.cern.ch//eos/root-eos/benchmark/Run2012B_SingleMu.root')
>>> filtered_missing_ET = ds.Where(lambda event: event.Jet_pt.Where(lambda pT: pT > 40).Count() >=
2).Select(lambda event: event.MET_pt).value()
>>> filtered_missing_ET
<Array [15, 44.7, 30.5, ... 123, 30.3, 20.4] type='6665702 * float32'>
>>> import matplotlib.pyplot as plt
>>> plt.hist(filtered_missing_ET, bins=100, range=(0, 100))
>>> plt.xlabel(r'$E_\mathrm{T}^\mathrm{miss}$ [GeV]')
>>> plt.ylabel('Events')
>>> plt.show()
```



Software ecosystem

- FuncADL is connected to multiple other IRIS-HEP projects, including:
  - ServiceX
    - A high-performance data delivery service
    - Provides a centralized and highly scalable platform to run FuncADL queries
    - Can be used to efficiently query large LHC Grid datasets
    - Talk by KyungEon Wednesday morning:
      - https://indico.cern.ch/event/948465/contributions/4323965/
  - `hep_tables`
    - Essentially FuncADL under the hood, but with a DataFrame interface
    - Provides a consistent numpy/awkward/pandas-like interface across data sources
    - Can use ServiceX to provide the back end operations
    - Talk by Gordon Wednesday evening:
      - https://indico.cern.ch/event/948465/contributions/4324133/

Summary

- FuncADL is a data query language aimed at porting some of the most important advantages of database management systems and functional programming into the realm of physics analyses
- Current back end implementations already support many use cases, and broader support is on the way
- Being used as a vital component of ServiceX (see KyungEon's talk on Wed.)

- vCHEP paper:
    - https://arxiv.org/abs/2103.02432

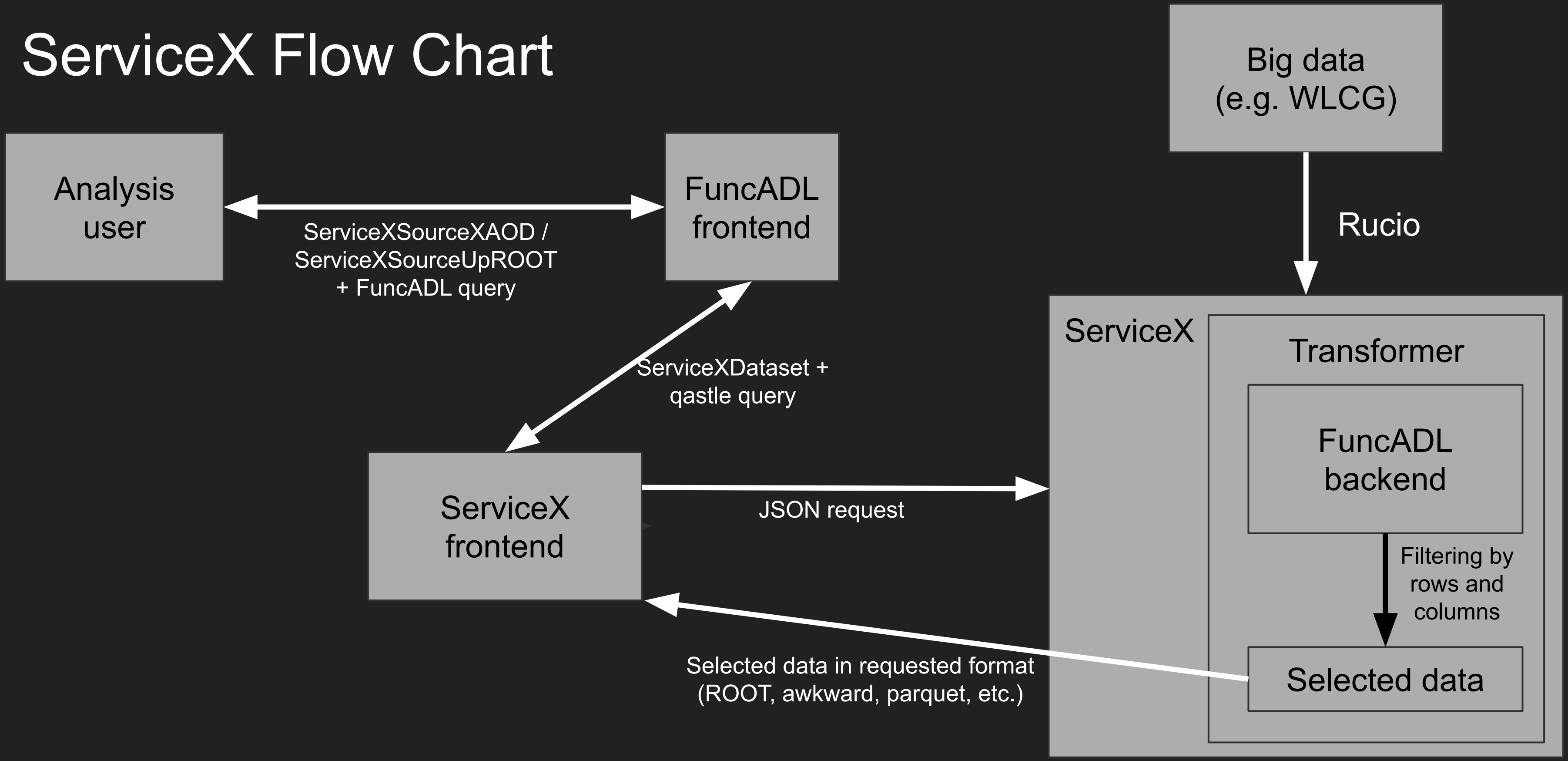
This work was supported by the National Science Foundation under Cooperative Agreement OAC-1836650.

Backup

Links

- FuncADL GitHub repositories:
  - https://github.com/iris-hep/func_adl
  - https://github.com/iris-hep/func_adl_servicex
  - https://github.com/iris-hep/func_adl_uproot
  - https://github.com/iris-hep/func_adl_xAOD
- ServiceX documentation, which includes FuncADL examples:
  - https://servicex.readthedocs.io/en/latest/user/getting-started/

ServiceX Flow Chart


Big data
(e.g. WLCG)
Analysis user
FuncADL frontend
ServiceXSourceXAOD /
ServiceXSourceUpROOT
+ FuncADL query
Rucio
ServiceXDataset +
qastle query
ServiceX
Transformer
FuncADL backend
ServiceX frontend
JSON request
Filtering by
rows and
columns
Selected data
Selected data in requested format
(ROOT, awkward, parquet, etc.)