Analysis Facilities for HL-LHC (NSF)

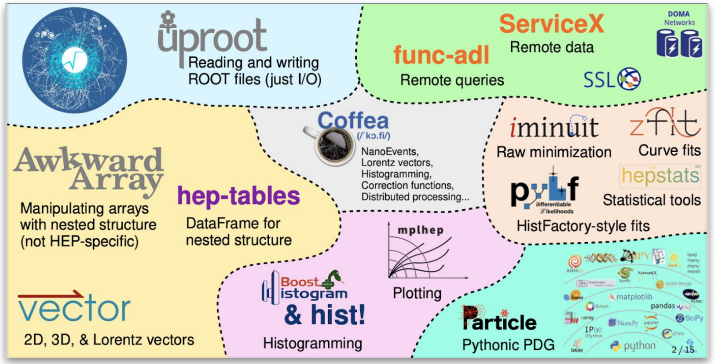
L. Bryant, R. Gardner, F. Golnaraghi, F. Hu, D. Jordan, J. Stephen, S. Thapa, I. Vukotic (UChicago)
O. Shadura, K.Bloom, C.Lundstedt, G.Attebury, J.Thiltges (UNL)
B.Bockelman (Morgridge Institute for Research)
Snowmass CompF4 Workshop
07 Apr 2022

Flexible platforms for user analysis

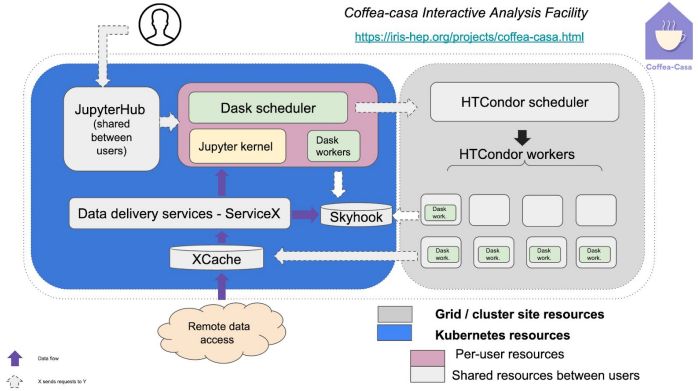
- Developers and system operators are leveraging cloud native technologies to build innovative new platforms for user analysis.
- There is a strong focus on building systems that:
  - Use declarative infrastructure
  - Scale on demand
  - Are resilient to failure
  - Are transparent and reproducible
- The front-ends of many of these systems also present interfaces to the Python ecosystem, which is receiving much attention and likely relevant for the near and longer term

Presenting a wealth of new tools to users

Novel analysis tools



Integrated into analysis facilities



Reproducible on premise or in clouds

Analysis Facility building blocks

- Modern authentication (AIM/OIDC) – **SciTokens, CI Logon, CERN idP**
- Efficient data delivery and data management tools – **Rucio, ServiceX**
- Support for object storage – **Ceph/Rook, MinIO**
- Efficient data caching solutions – **XCache**
- Columnar analysis and supporting a new Pythonic ecosystem – **Coffea Casa**
- Modern deployment and integration techniques – **K8S, GitOps, SLATE**
- Easy integration with existing HPC/HTC resources – **HTCondor, Dask**

CERN->FABRIC->Analysis Facility->Notebook

IRIS-HEP Analysis Grand Challenge Tools Workshop Example

```
[1]: from func_adl_servicex import ServiceXSourceUpROOT
     from hist import Hist
     import awkward as ak
```

We will process only one file from one of the samples. File is accessed using root protocol.

```
[2]: input_files = ['root://eospublic.cern.ch//eos/opendata/atlas/OutreachDatasets/2020-01-22/4lep/MC/mc_345060.ggH125_ZZ4lep.4lep.root']
     treename='mini'
```

CERN data source

The following command does almost everything.

First, it specifies data source by calling ServiceXSourceUpROOT and giving it filepath, root tree containing data, and a name of servicex
service to use has to be listed in the file servicex.yaml. In this repo there are two servicex instances that can process this data: uproot-af

Secondly, for every event it gets lepton pT.

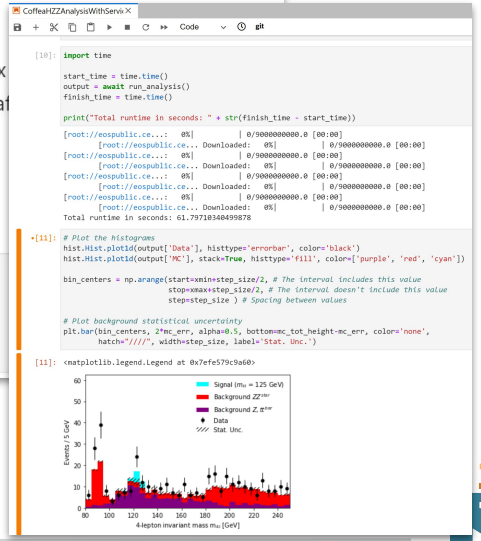
Finally, it specifies that the data should be returned as an Awkward Array.

ServiceX on FABRIC-NCSA

```
[3]: data = ServiceXSourceUpROOT(input_files, treename, backend_name='uproot-fabric') \
         .Select("lambda e: {'lep_pt': e['lep_pt']}")
         .AsAwkwardArray() \
         .value()
```

Query

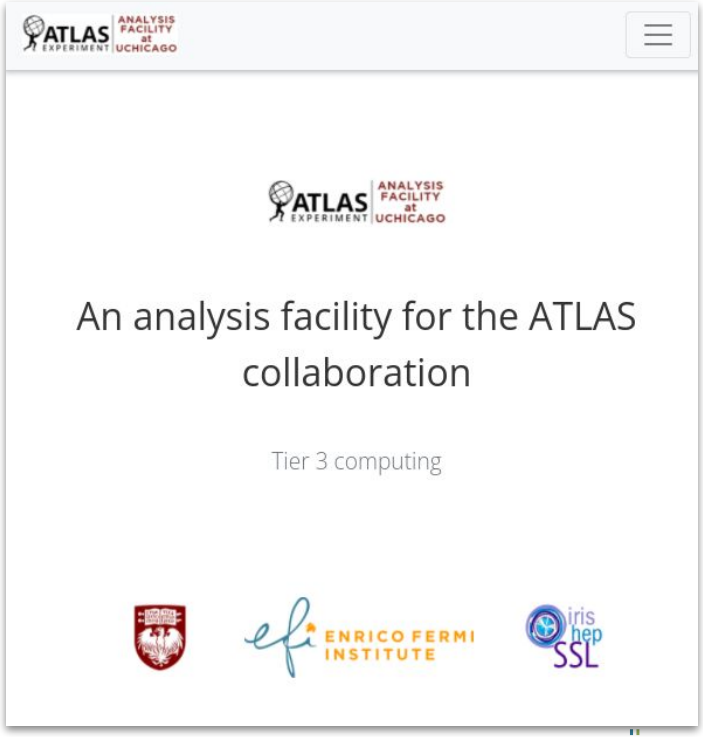
Output



All code can be found here.

AFs today

- **University of Chicago**
  - Kubernetes-based facility
  - Notebook-driven analysis and traditional batch analysis (Unix logins, local storage)
  - ATLAS IdP and CILogon Identity
- **University of Nebraska-Lincoln**
  - Kubernetes-based facility
  - Notebook-driven analysis via Coffea Casa, Dask and HTCondor
  - Integrated with CMS IdP



Evolving our broader computing model

- In the Open Science Grid and PATh, service deployment model leverages declarative infrastructure via Kubernetes (K8S)
- In US ATLAS, data caches (XCache) and software caches (Squid) deployed at Tier 2s via SLATE / Federated Operations (FedOps) – all leveraging Kubernetes
- US CMS building OpenShift-based services at the Tier 1, preliminary explorations of K8S elsewhere

Thank you!

Questions?

(Special thanks to all who contributed slides and feedback)