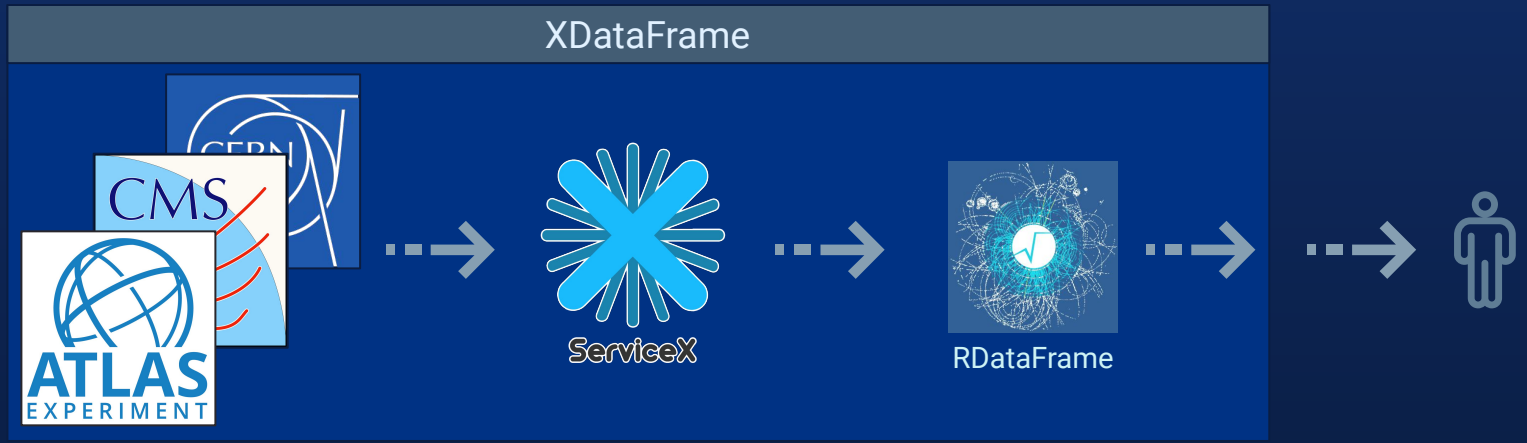
Implementing ServiceX data source for ROOT's RDataFrame

Nicholas Decheine
Mentor: Gordon Watts

The Goal:

Create a C++ program, XDataFrame, to act as a data pipeline between experiment data, fetched by ServiceX, and an RDataFrame, a ROOT analysis object.

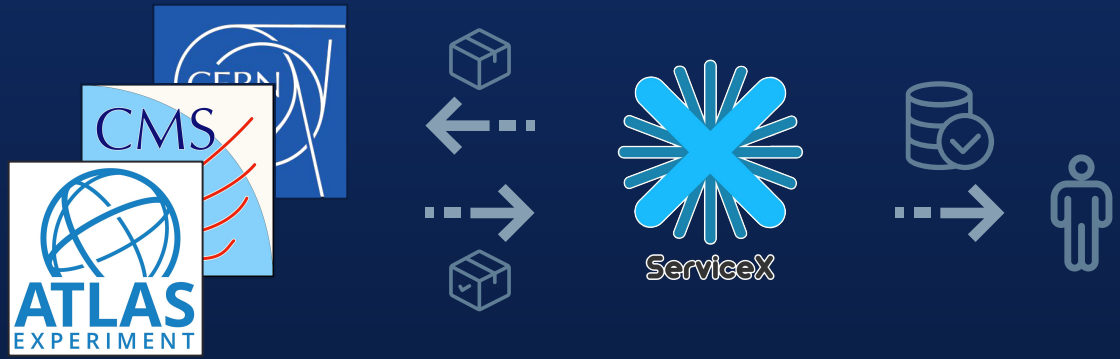


Use Case

- This library can be used by high energy physicists in their programs to streamline data acquisition to analysis.
- Researchers can spend less time downloading their data they want to use and get straight to coding the analysis, the stuff they actually care about.

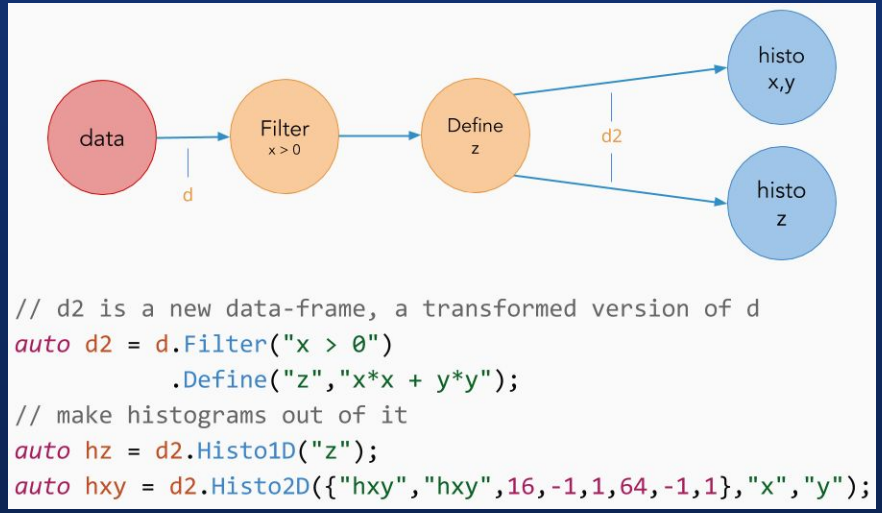
ServiceX

- ServiceX is a data extraction and delivery delivery service.
- Users provide a dataset identifier and a selection statement that specifies filters and columns.
- ServiceX brings up parallel workers to open the files in the dataset and uses experiment approved frameworks to extract the data and store it in popular columnar file formats for easy analysis using familiar tooling.
- A dashboard of current and past jobs is easily browsable on the React web app: cmsopendata.servicex.ssl-hep.org



RDataFrame

- ROOT's RDataFrame is an ideal analysis object that is commonly used by high energy physicists for its wide variety of tools and operations to use on large data sets.
- RDataFrame has implicit multithreading.
- Easy filtering and definitions of new transformed data-frames




```
// d2 is a new data-frame, a transformed version of d
auto d2 = d.Filter("x > 0")
           .Define("z","x*x + y*y");
// make histograms out of it
auto hz = d2.Histo1D("z");
auto hxy = d2.Histo2D({"hxy","hxy",16,-1,1,64,-1,1},"x","y");
```

Figure 1: *RDataFrame filter diagram. A graph composed of two branches, one starting with a filter and one with a define. Source: CERN, RDataFrame class reference page*

Technologies Used by XDataFrame

ROOT

For RDataFrame object

ServiceX

Data delivery service

CMake

Build system

MinIO

High performance, Kubernetes native, cloud object storage used by ServiceX to store delivered files

AWS C++ SDK

S3 package to fetch data from S3-based MinIO cloud storage

Dependencies Note

All external dependencies are handled and fetched by CMake.

Using XDataFrame

- The library can be included in a C++ project. First set a `requestString` including at least the following

Experiment ID

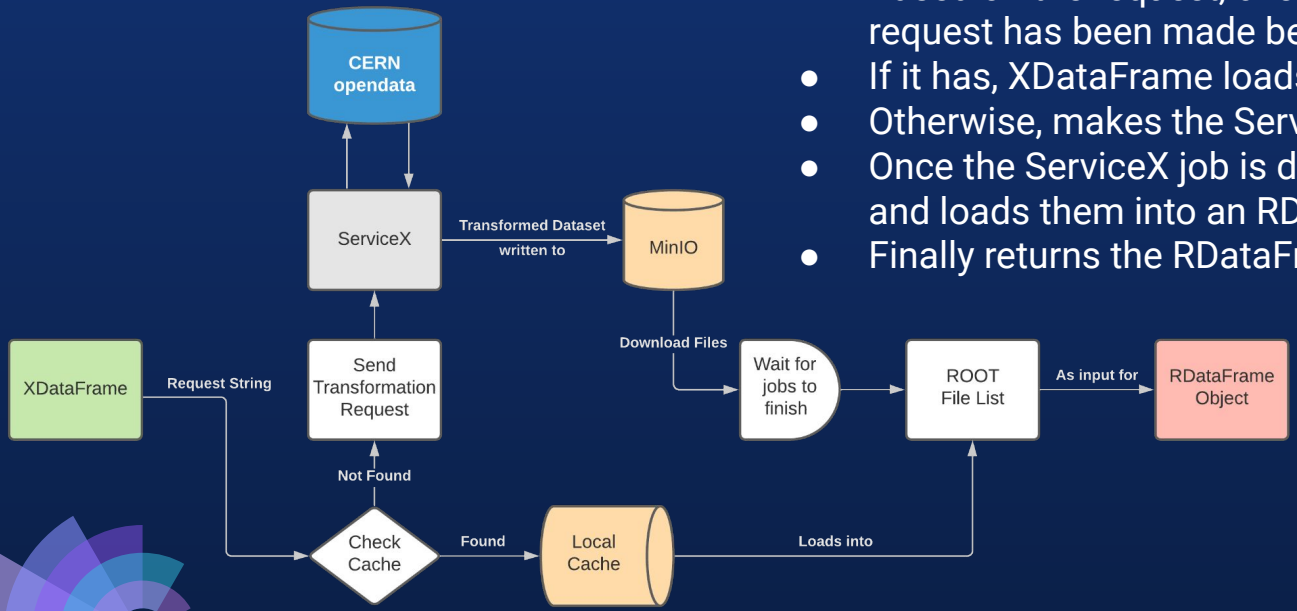
Selection string. Written in func_adl, an SQL like language for extracting columnar data from a ROOT file or ATLAS xAOD file.

```
{
    "did": "cernopendata://1507",
    "selection": "(call ResultTTree (call Select (call SelectMany (call
EventDataset 'ServiceXSourceCMSRun1AOD') (lambda (list e) (call (attr e
'TrackMuons') 'globalMuons'))) (lambda (list m) (call (attr m 'pt')))) (list
'mu_pt') 'treeme' 'file.root')",
    "result-destination": "object-store",
    "result-format": "root-file",
    "chunk-size": "1000",
    "workers": "20"
}
```

How a call to XDataFrame will look used in code.

```
ROOT::RDataFrame mydf = XDataFrame(requestString);
```

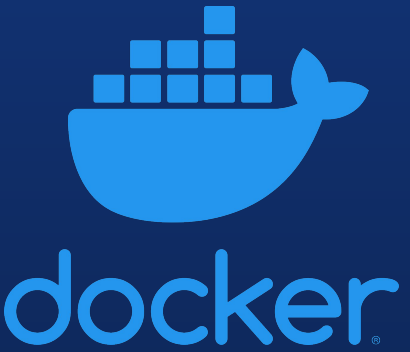
How It Works



- Takes in a string specifying the request.
- Based on the request, checks it's cache if the same request has been made before.
- If it has, XDataFrame loads the cached files.
- Otherwise, makes the ServiceX request.
- Once the ServiceX job is done, it fetches the ROOT files and loads them into an RDataFrame.
- Finally returns the RDataFrame to the user.

Demo

- As part of this project I created a Docker image hosted on Docker Hub with an installation of XDataFrame and a demo project that can be run immediately after entering the Docker image.
- User's can try out XDataFrame without having to install it.
- The demo .cxx file contains an example request_string, creates a ROOT canvas, creates an RDataFrame using XDataFrame, and applies a few simple filters to it. Then draws the histogram and prints it to a pdf file.
- This demo program serves as a useful starting point for those using XDataFrame for the first time.



Issues and Improvements

- Improvements
  - XDataFrame could be extended to take a file location containing an input string
- Issues
  - First build time can take a few minutes, as downloading and building the AWS SDK S3 package can be demanding. Once files are downloaded and it is built, it does not need to be built again subsequently.
  - There are a few different ways one can incorporate XDataFrame into their project
    - Add the repository to a project's directory tree and use `add_subdirectory(XDataFrame)` to add the source code directly to the project.
      - This has the least work on the user's part, but will add all the dependencies of XDataFrame to the user's CMake project. This method was chosen because it works the best so far.
      - `FetchContent` could also work well
    - Install XDataFrame and have a project use `find_package(XDataFrame)`, but this has run into complications and requires more steps on the user's part. This method was not chosen for these reasons.

Thank you!

Questions?