DIANA Fellowship Proposal

Project Description

This project is based on adding functionality to uproot, a software for reading ROOT files solely in Python with the help of the Numpy library. Unlike the standard C++ ROOT implementation, uproot is only an I/O library, primarily intended to stream data into machine learning libraries in Python. Other ROOT file readers like PyROOT and root_numpy rely on the C++ ROOT implementation but uproot does not. Instead, it uses Numpy calls to rapidly cast data blocks in the ROOT file as Numpy arrays.

ROOT files can be very large so it would not be unusual to encounter a file that is too big to load entirely into memory. To solve this issue, uproot supports selective reading of individual data attributes. It also supports remote reading through XRootD, a remote file protocol for analyzing data located at any XRootD-enabled site without downloading it to one's local filesystem. If the user requires just a few arrays from a file that has hundreds, it would be much faster to read the file from the server with XRootD. This python implementation gives users a lot of control over how ROOT files are read which saves computing power as well as lets the user focus on specific portions of the ROOT file by slicing an array before loading it from the file. Reading specific portions of ROOT files makes iterating over the data more efficient.

I propose to, under the mentorship of the author of uproot Jim Pivarski, add new functionality to uproot so it can write ROOT files in addition to reading them. Currently, uproot reads data by moving TTrees into Numpy arrays, but it should be able to do the reverse and write TTrees from data stored in Numpy arrays. In the current implementation of uproot, the user has to export the data from the read ROOT files into some other format to share it with others. Once uproot is able to natively write ROOT files, the user could write histograms and modified TTrees to ROOT files in uproot itself. This would make sharing of data between people more convenient and promote collaboration as the user would not have to leave the ROOT environment or depend on some other library in order to share their work.

Proposed Timeline

Week 1 -

- Familiarise myself with the uproot codebase and design guidelines.
- Design an interface to write simple ROOT files.

Week 2 -

- Write a ROOT file which comprises of only TObjString in TDirectories.
- Check that the newly implemented interface works by testing if the written ROOT file can be read by C++ ROOT and other ROOT readers like root4j, go-hep and uproot itself.

Week 3 -

- Update the codebase to create nested TDirectories in ROOT files.

Week 4 -

- Implement functionality to write streamers to ROOT files.

Week 5 -

- Implement functionality to write Histograms to ROOT files.

Week 6 -

- Implement functionality to write TTrees which just comprise of a flat table of numbers to ROOT files.

Week 7 -

- Update the codebase to be able to write TTrees of Jagged Arrays to ROOT files.

Week 8 -

- Update the codebase to be able to write TTrees of variable-width objects like strings.

Week 9 -

- Start implementing an interface so the user can write their own custom objects to TTrees in ROOT files with streamers.

Week 10 -

- Implement functionality in uproot to copy data from ROOT files stored on the XRootD server into another ROOT file on the XRootD server.

Week 11 -

- Implement a streamlined hadd in uproot with parallel processing of data.

Week 12 -

- Update the uproot documentation.
- Prepare a technical report on the work done.
- Clean up the code and solve any lingering issues.