Refactoring AwkwardForth Generation in Uproot

Seth Bendigo[1]
Mentors: Ioana Ifrim[2] Jim Pivarski[2]

2023

[1]South Dakota School of Mines and Technology

[2]Princeton University

Intro to Uproot

Intro to Uproot

Uproot is an I/O library for reading and writing ROOT files for use in Python [1]. To keep it lightweight and portable, it is kept strictly as an I/O library, and does not depend on ROOT.

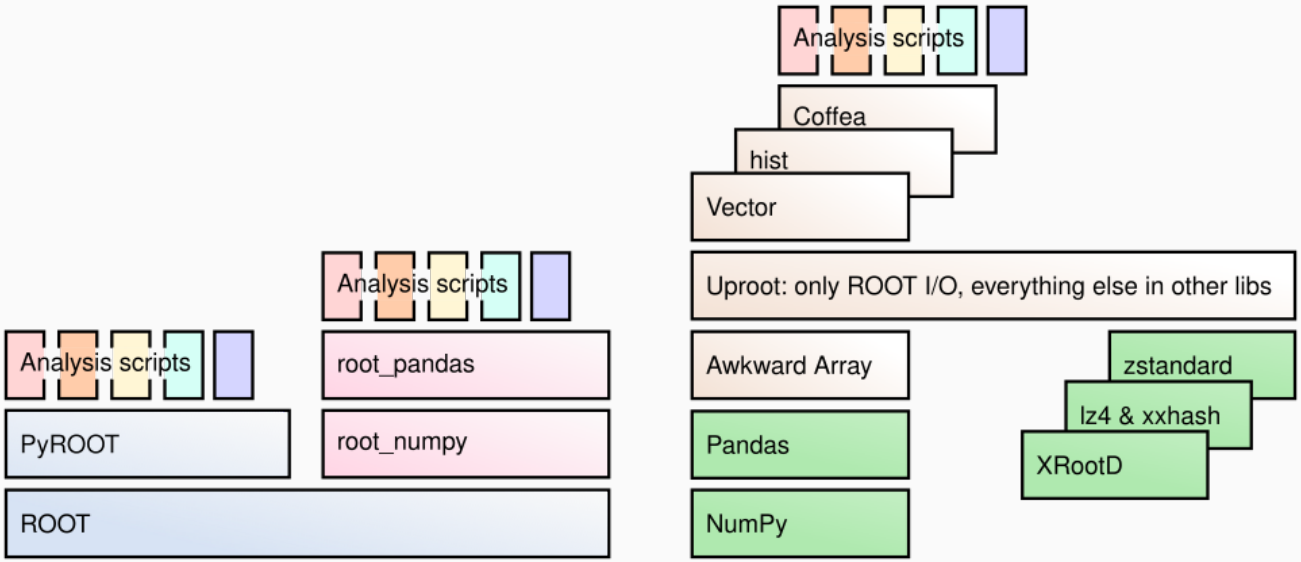

Figure 1: Abstraction layers of various methods to use ROOT files in Python.

Intro to Uproot

When reading in non-columnar data types, iteration is required, and Python loops are slow compared to compiled languages.

- A compiled language cannot be used since, at compile time, the byte-for-byte layout in ROOT files with complex data types is unknown.
- Just-in-time compilation could be used, but this affects portability.

AwkwardForth

AwkwardForth

Uproot instead implements the use of AwkwardForth, an internal domain specific language [2].

By generating AwkwardForth code to read in the incoming complex data types, Uproot is significantly optimized. In the case of std::vector<std::vector<float», AwkwardForth is faster than Python by a factor of about 400.

AwkwardForth

Unfortunately, the current implementation of AwkwardForth generation in Uproot has some problems.

Issues

- Excessively mutable: Objects that change their attributes in arbitrary ways as information needed to generate AwkwardForth accumulates.
- Readability: Dead code, nondescript attribute names.

Refactor

Refactor

To fix these issues, the refactor will focus on rewriting the generation to be more aligned with **functional** programming. This will help by explicitly avoiding the mutability that has caused issues.

The refactor will also utilize test-driven development extensively. There are $\sim$140 tests relating to AwkwardForth in the current implementation.

Refactor

So far, I have removed the code that relied on AwkwardForth. My next step is to understand in depth how AwkwardForth *currently* generates for a simple case (such as std::vector<std::vector<float»), and then redesign it.

Then, I can expand on the new design to cover other data types using the (already written) tests to guide me.

Bibliography

Jim Pivarski, Henry Schreiner, Angus Hollands, Pratyush Das, Kush Kothari, Aryan Roy, Jerry Ling, Nicholas Smith, Chris Burr, and Giordon Stark.
Uproot, June 2023.

Jim Pivarski, Ianna Osborne, Pratyush Das, David Lange, and Peter Elmer.
AwkwardForth: accelerating uproot with an internal DSL.
EPJ Web of Conferences, 251:03002, 2021.