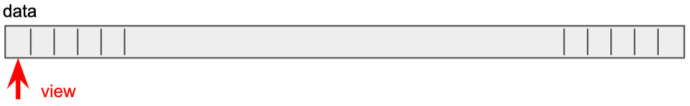
Awkward Arrays to RDataFrame and back

Ianna Osborne, Jim Pivarski
Princeton University

Awkward Arrays and RDataFrame provide two very different ways of performing calculations at scale. By adding the ability to zero-copy convert between them, users get the best of both. It gives users a better flexibility in mixing different packages and languages in their analysis.

The *ak.to_rdataframe* function presents a view of an Awkward Array as an RDataFrame source. This view is generated on demand and the data is not copied. The column readers are generated based on the run-time type of the views. The readers are passed to a generated source derived from *ROOT::RDF::RDataSource*.

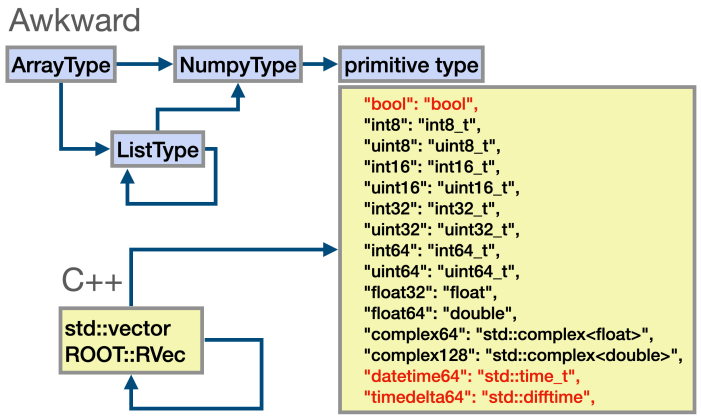
- Generated AwkwardArray RDataSource takes pointers into the original array data: a 40-byte ArrayView object is allocated on the stack
- The large-scale array data are not copied
- The views are transient, their lifetime is defined by the lifetime of their lookup Python object



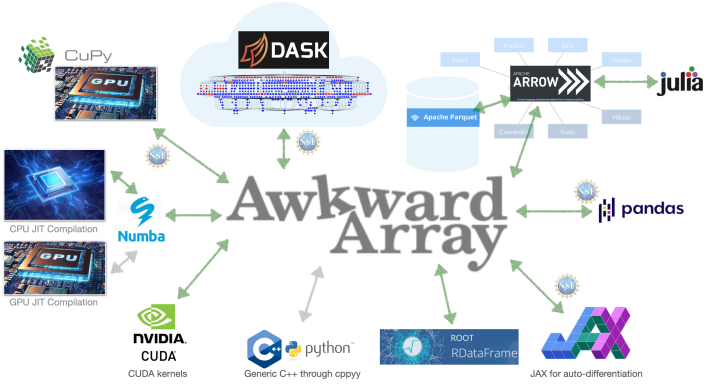
- ArrayViews keep a pointer to their lookup Python object to facilitate a no-copy conversion back to an Awkward Array

The implementation is exploiting JIT techniques. Constructing the Awkward Arrays from the RDataFrame result pointers is fairly independent from ROOT.

The *ak.from_rdataframe* function converts the selected columns as native Awkward Arrays.



- A Pythonized by cppyy C++ header-only class converts the data to the Numpy-like buffers
- The C++ templated header-only implementation constructs the Form - i.e. the description of an Awkward Array - from the data types



See Manasvi's talk The Awkward World of Python and C++

Awkward RDataFrame Tutorial presented at the PyHEP 2022 Workshop shows examples of the column definition, applying user-defined filters written in C++, and plotting or extracting the columnar data as Awkward Arrays.

Support for this work was provided by NSF cooperative agreement OAC-1836650 (IRIS-HEP)