File formats for physics data

Jim Pivarski

Princeton – DIANA

November 7, 2016

The obvious answer: ROOT, right?

Advantages of ROOT

- Flexible format can encode arbitrarily complex C++ data structures.
- Encoding ("streaming") can be hand-tuned.
- May be record-major or columnar ("split") with individually compressed columns ("branches").
- Familiar to physics community, and developers are very responsive to physics needs.

Disadvantages of ROOT

- Lack of native libraries for other runtimes (e.g. Spark runs on JVM).
- Custom encoding can only be read with original code.
- File format requires a lot of seeking, can't (efficiently) be streamed (e.g. via HTTP).
- No familiarity outside of physics. Big Data and Machine Learning communities use other formats.

Some popular formats

	binary	schema	structures	columnar	compressable
XML/JSON	no	external	trees	no	external
MessagePack, UBJSON, CBOR, etc.	yes	no	trees	no	some are
Protobuf, Thrift, Avro	yes	yes	trees	no	yes
Parquet	yes	yes	trees	yes	yes
Arrow/Feather (in-memory)	yes	yes	trees	yes	no
HDF5	yes	yes	arrays/trees	manually?	yes
Numpy	yes	yes	flat arrays	can be	yes
ROOT	yes	yes	graphs	can be	yes

http://en.wikipedia.org/wiki/Comparison_of_data_serialization_formats

Abstract type systems

Every serialization format must have a type system. Most are not glued to a particular language.

Different type systems, but the reuse the same concepts:

1. boolean, integers, floats, strings, maybe null (singleton)
2. variable-length arrays of X, maybe key-value maps of X
3. record structures with named, typed fields (product types)
4. maybe tagged unions of the above (sum types)

"Flat arrays" (Numpy; TNtuple) are fixed-length arrays of #1 only.

"Graphs" (ROOT) describe a full object graph using indexes that point into an array.

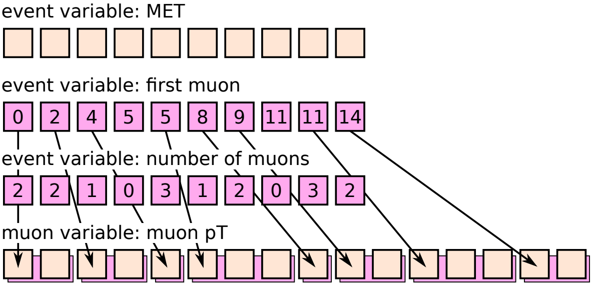
Tagged unions can encode nullable types and are more general than class hierarchies.

Logical type systems

Type systems are not limiting: conventions and interpretations on top of the "physical type system" can provide high-level features.

For example,

- flat arrays (Numpy) can encode structures with index arrays:



- Parquet uses logical type systems to provide Avro objects.
- ROOT's physical types are streamers, logical types are C++.

Transient: getting data into another system

- Necessary in many cases: the C++/Java barrier is particularly difficult to overcome.

End-stage of analysis in another system

- If we're making ntuples anyway, the format should be chosen with the analysis tool in mind: ROOT or otherwise.

Releasing open data to the public

- Current format probably inhibits use. Avro/Parquet on popular analysis clouds (AWS, Google Cloud, Azure, etc.)?

New primary data storage format?

- Would require *considerable* thought.

Today's talks

Avro and Numpy for Spark and machine learning

7 Nov 2016, 17:30 → 18:55 Europe/Zurich

40-R-B10 (CERN)

Description Experiences writing from CMSSW to industry-standard file formats for interoperability with big data and machine learning tools.

17:30 → 17:40 **Overview of file formats** 10m
Speaker: Jim Pivarski (Princeton University)

17:40 → 18:10 **Numpy and Avro files for machine learning** 30m
Speaker: Valentin Y Kuznetsov (Cornell University (US))

18:10 → 18:20 **Structured Avro files for Apache Spark** 10m
Speaker: Nhan Viet Tran (Fermi National Accelerator Lab. (US))

18:20 → 18:30 **Reading ROOT files with a pure-Java I/O library** 10m
Speaker: Viktor Khristenko (The University of Iowa (US))