Some thoughts on protocol generators

Jan C. Bernauer for the organizers

Streaming Readout VII, 2020

RBRC
RIKEN BNL Research Center

Stony Brook
University

Why protocol generators

- We need to define a series of data structures for data exchange on the wire / on disk
- Ideally use common "framing" along most parts of the chain – close to the FE to disk
- Need to specify and document formats
- Integrity check during deserialization
- Can autogenerate interface for many languages

Wishlist

- Low overhead – we often hit I/O limits
- Implementable on FPGA
- Can contain anonymous data
- Can contain hierarchical description
- Can describe foreign formats
- Compatible with HPC?
- Automatic tool generation

Data types I checked

- (u) int 8,16,24,32,64
- float, double
- bitfield
- string
- vector of...
- maps (this is probably never in-place)

Google protocol buffers

- Languages: All the languages
- Not in-place, has to unpack
- Wire format is forward/backward compatible, not self-describing
- Doesn't have: (u)int24, bitfield. No bit packing.
- Instead, packs ints, so int64==int8.
- Has a RPC standard.

Google Flatbuffers

- In-place, low overhead
- Evolution: Can add fields, can mark fields as deprecated (in tables). Can not remove fields
- Has some accomodations for reflection.
- Doesn't have (u)int24, bitfield, No bit packing. Vectors only one level (or via table). No maps, but keys for bin search.
- Has nested structure, but full definition required at compile time.

Capt'n Proto

- In-place, low overhead
- Wire format is forward/backward compatible, not self-describing
- Doesn't have: (u)int24, bitfield. No bit packing. No maps. List size has to be known at alloc time.
- Has a RPC standard.

Apache Avro

- Every file has schema definition as the header
- Dynamically typed, untagged data
- Not in place
- Types: No (u)int16,24, bitfield. No bit packing.

Apache Thrift

- Languages: All the languages
- Mainly aimed at RPC
- Not in place.
- But supports different encodings, transports.
- Types: No (u)int16,24, bitfield. No bit packing.

Some simple prototypes

- Cap'n proto:
  `git@github.com:JanCBernauer/capnproto_test.git`
- Flatbuffers:
  `git@github.com:JanCBernauer/flatbuffers_test.git`
- I liked flatbuffers more, so I made a prototype sPHENIX event container (with serialized root as payload):
  `git@github.com:JanCBernauer/storage.git`

Roll our own

- None of the exisiting solutions are perfect, for us.
- They let us define a logical data structure, and will generate the representation.
- We likely want to define the representation:
  - Would really be great if I could specify: bit 20 to 24 in this DWORD is "channel_num"
    And then get a proper accessor in all the languages we want to support.
  - HPC/GPU suitable?
- Automatically generate debug code – range checks, bitfield decoders
- We need to document data formats anyway – why not make it work for us?