CMSSW IO Updates

Brian Bockelman

ROOT I/O Workshop, Round II

Slide from last time

Preamble

- **I'm pretty happy**. Our CPU efficiency has been improving significantly in the last year. Remote I/O is a reasonable thing to do in CMS.
- Tests with Google ProtoBufs show they are indeed faster … in a very limited use case (no compression, all data in RAM cache, no column-wise access).
  - To me, this shows we aren't completely crazy.

First, an Apology

- The last two months have been exciting...
  - ... just not in this area. These activities have suffered, and we haven't made the sort of progress I had envisioned.
- To some extent, it's a "renegade" activity - Such is life.

Progress in CMSSW IO

- CMS is in the transition between 2011 and 2012 - an opportune time to break backward compatibility.
- We've recently completed an IO survey on what can be done for 2012 data structures.

CMS IO Survey

- Focus on analysis data (RECO IO time is not relevant currently).
- Instrument ROOT IO so we can determine the number of embedded objects in the file (https://indico.cern.ch/getFile.py/access?contribId=59&sessionId=15&resId=0&materialId=paper&confId=149356).
- Illuminating for what behaves really poorly, and what is used more often.

Finding "Badness"

- We are still cavemen at quantifying and ordering issues in our data formats.
  - *Sample metric*: the number of lines of printout at gDEBUG=1 to serialize a default instance of an object.
  - *Sample metric*: adding a "cout" for each embedded object in an event, and counting the number of cout's.
  - While these provide ad-hoc illumination, it's a hack. We can't share it with other experiments, and we can't do a thorough job.

Finding "Badness"

- We still have no great tool for measuring ROOT deserialization time!
  - Especially for measuring inflate and deserialization time separately per branch.
  - We have ~300 data objects, and a vanishingly small amount of personnel time. I need this guidance to know where to direct my efforts.

Data Formats Changes (?)

- Candidates for "fixing":
  - DetId (4-byte ID of a piece of the detector)
  - Ref's (reference to a serialized element of a collection).
  - 3D-points
  - "Small" vector (std::vector<foo> of length < 5)
- Also identified physics formats that behave poorly with ROOT I/O.

Candidates for “Fixing”

- Current Plan:
  - DetID and Point3D are going to be fixed in-application.
  - Small vectors - bad for other reasons, but how much are we paying in CPU time? We’ll be providing better tools for users to use.
  - Ref’s are largely unsolved.

What is the deal with Refs?

- CMS has several ways to refer to a persisted data product.
  - "Equivalent" to C++ references, pointers, optimizations for collections of refs.
  - There is a RefCore object that does "most of" the functionality, this is used by Ref, Ptr, others. Templated Ref & Ptr provide type-safety.
  - Because people love references (especially in unsplittable collections), and each Reference has a layer of inheritance over a small amount of data, we have an issue - performance is critical.
- For this one, we see little option besides custom streamers.
  - If performance tests show that speed of custom streamers are an insufficient gain, we will likely be happy with "just" removing the byte count.

Progress in Framework Thinking

- The CMSSW framework is currently gravitating toward work-queue/consumer-producer models of concurrency.
- (Reference discussion from yesterday)
- Requires either compatibility - or not outright hostility - from ROOT.

Progress in ROOT Contributions

(Little)

Progress in ROOT (Continued)

- Some of the schema evolution work has been put on hold as the primary drivers are now custom streamer (maybe? C.f. discussion on RefCore)
- Algorithmic, but not code progress in OptimizeBaskets
- Algorithmic, but not code progress in fast merging.

ROOT 5.30 tests

- ROOT 5.30 with CMSSW has been unfortunately delayed by about 1-2 weeks.
  - CMSSW issues in merging files => no files to test ROOT 5.30 performance.
  - New pre-release today => hopefully files for testing in 48 hours.
  - (Recall that our target was to test AsyncPrefetch with CMS's TTreeCache setup).

Top Requests for ROOT

- In order of importance/time:
  - Remove byte count from serialization.
  - Better tools for measuring "badness".
  - Hooks for task-queues (or at least have certain minimal thread safety guarantees for serialization/unzipping/reading).
  - ROOT 6: Remove big endian, per-buffer class versions, object maps by offset, stop reseting of compression engine.

I think the first and third items are new.