Summary of Short I/O Investigation

September 2, 2011
Brian Bockelman

General ROOT I/O Issues

CMS uses a heavily object-based data model. This is often at odds with best-practices design for ROOT I/O data structures. None of the items below are likely new to experts, but I found it useful to highlight a few to focus our efforts on.

Common patterns:

- **Using class inheritance**: When there is a specialization of a data product class, CMS tends to create a sub-class. This is recommended practice for C++, but causes issues for ROOT I/O; currently, it is 10 bytes of extra data per base class (decreasing to 6 in CMSSW 5).
    - This issue is most clearly highlighted in the children of the DetId class. The most extreme example I found was one that had 5 layers of inheritance; 54 bytes to store a 4 byte ID.
    - **Recommendation**: Avoid when possible, especially for small data structures and when the specialization is not needed.
- **Polymorphism**: When polymorphism is used, ROOT is unable to determine the data structure ahead of time and optimize accordingly.
  Recommendation: If not necessary, do not use. Avoid as possible.
- **Per-object metadata**: ROOT writes out metadata per-object, including metadata per parent class (as noted above). It is extremely unlikely this data will change rapidly within a file. In the RECO tier, per-object metadata takes about 33% of the uncompressed size. In AOD, it is about 10%.
  Recommendation: Ask ROOT to change the per-object metadata to be per-buffer or per-file.
- **Using STL containers**: STL containers (std::vector) take an extra malloc and several pointers to create (40 bytes of memory overhead on 64-bit systems), making deserialization slower. While they provide important usability benefits, these overheads are too high price when the number of objects held are constant or small. As a rule of thumb, use a fixed-size container if the memory overhead of creating a STL vector is greater than 30% of the total. In our test AOD file,
    - 98% of embedded vector<int> instances would be considered "too small" under this rule.
    - 96% of embedded vector<float> instances are too small.

- 100% of embedded vector<double> instances are too small.

  Recommendation: Survey the authors of the most popular objects, and ask them to remove as possible. Have the framework group provide a solution optimized for the small-container case (perhaps a small amount of fixed-size data with an "overflow" pointer?).

- **High-dimension containers**: ROOT is increasingly ineffective at serializing containers the higher the dimension. 1D containers are fine (as they can be top-level branches). 2D containers are tough (best to replace them with fixed size if reasonable). 3D containers are horribly inefficient.

  - **Recommendation**: Replace any triply nested container with a doubly-nested container using a technique similar to DetSetVectorNew or by creating a reference to another product.

  - **Recommendation:** We should consider switching to a Ref any object over 300 bytes and nested in a container.

Specific ROOT I/O Issues and Fixes

In the appendix, I include a table of all the classes I found embedded (ROOT was unable to split into a separate branch) in AOD, and their frequency.

A few notes:

- The following classes are embedded often in our data; individually, they are small, but sum up to be a significant amount of our embedded objects:

  - **Any 3-vector** or **4-vector**: These classes have an inheritance structure, typically meaning 20 byte overheads per vector. Thus, we often pay at least 50% overhead for 3-vectors in serialization. **Recommendation**: rewrite the CMS vector class to remove class inheritance. Then, the highest overhead would be 30% for a 3-vector of floats.

  - **DetID descendents**: DetID takes 14 bytes to store a 4 byte object; for each layer of inheritance, add another 10 bytes (in CMSSW 5, this is 10 bytes and 6 bytes, respectively). **Recommendation**: Remove some layers of the class, or use a custom streamer.

  - **BoolCache** and **DoNoPostReadFixup**: The latter is unused, and the former can be replaced with the new schema evolution rules. **Recommendation**: We should eliminate both, replacing them with schema evolution rules.

  - **String**: There are about 180 std::strings serialized per AOD event. Around 150 are due to L1 trigger, noted below. Another handful are due to the triggerResults object. The rest appear to come from CompositeCandidates. **Recommendation**: Remove all of these.

- **RangeMap** and **std::map**: ROOT is unable to serialize std::map effectively; it has to deserialize the contents once to a vector of pairs, then insert into a separate map. **Recommendation**: Remove; where used, replace with DetSetVectorNew (or equivalent).

- The following classes are problematic in terms of their complexity and deserialization speed:

  - **PFTau**: PFTau contains many Refs and RefVectors (the Refs are guaranteed to be inside the RefVectors; several of the RefVectors are redundant), plus contains a PiZero object. **Recommendation**: Reduce the number of Refs/RefVectors (at least 6 can be removed easily) and turn the PiZero into a ref.

  - **CSCRecHit2D**: This contains 7 std::vectors and a std::map. **Recommendation**: After consultation with the maintainer, all these can be either removed or changed to a fixed-size array.

  - **L1 Triggers**: The current data structure contains a triply-nested vectors of ints (vector<vector<vector<int> > >), causing a major slowdown in memory and in deserialization. Additionally, there are a huge number of strings (150+). Trigger information averages 16KB uncompressed per event, and contributes significantly to the deserialization time (all of this has to be deserialized to read any of it). **Recommendation**: At the minimum, move all the strings to lumi information. The triply-nested vector should be replaced with a flatter data structure.

  The maintainers of the data structures mentioned above have been informed.

- We should consider custom streamers for heavily used classes that are serialized poorly, unlikely to change, or need schema evolution. In CMSSW 5, the minimum overhead is 6 bytes (2 bytes class version, 4 bytes for byte count; add another 4 bytes for virtual classes). This could be reduced to 2 for classes where we may want to evolve in the future, or 0 for classes we will never evolve within a release series. Recommended are:

  - **3-vectors**: These will never evolve.

  - **DetId children**: I do not believe these will evolve – they contain 4 bytes of data, and up to 50 bytes of metadata, depending on the class hierarchy.

  - **Ref, Ptr, and friends**: These are somewhat complicated and probably will evolve. However, there are thousands per event; this is likely worth the pain. I can provide a sample implementation.

Suggested Work Plan

Here's my recommended work plan and suggested. In order of importance:

1. Fix 3-vectors and 4-vectors; custom streamers. (framework group)
2. Remove BoolCache and DoNoPostReadFixup (framework group)
3. Fix L1 triggers (L1 and performance group)
4. Custom streamers for Ref/Ptr (framework group)
5. Remove RangeMap (performance group)
6. Custom streamers for DetId children (performance or framework group)
7. Fix CSCRecHit2d (maintainers)
8. Fix PFTau (maintainers)
9. Consider a "small vector" specialization (performance group)

Appendix A

Below is a list of all the embedded objects in the first 159 events of an AOD file. The first column is

426899 edm::BoolCache
246580 edm::RefCore
201114 DetId
72552 edm::helpers::DoNoPostReadFixup
39941 string
29829 edm::Ref<vector<reco::Track>,reco::Track,edm::refhelper::FindUsingAdvance<vector<reco::Track>,reco::Track>>
28941 edm::reftobase::BaseHolder<reco::Track>
28516
edm::reftobase::Holder<reco::Track,edm::Ref<vector<reco::Track>,reco::Track,edm::refhelper::FindUsingAdvance<vector<rec
o::Track>,reco::Track>>
17690 ROOT::Math::Cartesian3D<double>
17166 vector<float>
15054 reco::isodeposit::Direction::Distance
13725 vector<int>
13052 TNamed
11765 vector<unsigned int>
6475 pair<unsigned int,unsigned int>
5472 ROOT::Math::Cartesian3D<Double32 t>
5428 Basic3DVector<float>
5404 Point3DBase<float,LocalTag>
5404 PV3DBase<float,PointTag,LocalTag>
5396 TAttFill
5390 TBranchElement
5380 LocalError
5294 TrackingRecHit
5207 edm::reftobase::RefHolderBase
5199 TLeafElement
5092 ROOT::Math::PositionVector3D<ROOT::Math::Cartesian3D<Double32 t>,ROOT::Math::DefaultCoordinateSystemTag>
4892 edm::ProductID
4685 vector<edm::Ptr<reco::Candidate>>
4681 edm::Hash<1>
4614 edm::reftobase::IndirectHolder<reco::Jet>
4614 edm::reftobase::BaseHolder<reco::Jet>
4292 vector<pair<edm::ProductID,unsigned int>>
4190 edm::ParameterSetBlob
3816 vector<bool>
3339 edm::ValueMap<bool>
3245 edm::Timestamp
3242 edm::Hash<2>
3239 edm::EventID
3204 vector<double>
3076
edm::reftobase::RefHolder<edm::Ref<vector<reco::CaloJet>,reco::CaloJet,edm::refhelper::FindUsingAdvance<vector<reco::Cal
oJet>,reco::CaloJet>>
3076 edm::Ref<vector<reco::CaloJet>,reco::CaloJet,edm::refhelper::FindUsingAdvance<vector<reco::CaloJet>,reco::CaloJet>>
2946 reco::CaloID
2946 edm::Ptr<reco::CaloCluster>
2946 ROOT::Math::PositionVector3D<ROOT::Math::Cartesian3D<double>,ROOT::Math::DefaultCoordinateSystemTag>
2668 RecHit2DLocalPos
2628 DTWireId
2628 DTSuperLayerId
2628 DTLayerId
2628 DTChamberId
2602 RecHit1D
2602 DTRecHit1D
2199 map<int,pair<unsigned int,unsigned int>>
2199 edm::RangeMap<int,vector<float>,edm::CopyPolicy<float>>
2115 CSCRecHit2D
1907 vector<CorrMETData>
1621 ROOT::Math::Cartesian3D<float>

1603
edm::Ref<edm::RangeMap<DTChamberId,edm::OwnVector<DTRecSegment4D,edm::ClonePolicy<DTRecSegment4D>>,edm::C
onePolicy<DTRecSegment4D>
1603
edm::Ref<edm::RangeMap<CSCDetId,edm::OwnVector<CSCSegment,edm::ClonePolicy<CSCSegment>>,edm::ClonePolicy<CSCS
egment>
1550 multimap<reco::isodeposit::Direction::Distance,float>
1538
edm::reftobase::RefHolder<edm::Ref<vector<reco::JPTJet>,reco::JPTJet,edm::refhelper::FindUsingAdvance<vector<reco::JPTJe
t>,reco::JPTJet>>
1538 edm::Ref<vector<reco::JPTJet>,reco::JPTJet,edm::refhelper::FindUsingAdvance<vector<reco::JPTJet>,reco::JPTJet>>
1427 edm::BranchID
1140 reco::HitPattern
1108 vector<reco::MuonSegmentMatch>
975 vector<string>
936 vector<reco::RecoTauPiZero>
882 vector<pair<DetId,float>>
795 edm::ValueMap<float>
793 vector<unsigned short>
740 vector<unsigned long>
707 vector<unsigned char>
676 set<string>
676 edm::Transient<edm::BranchDescription::Transients>
676 edm::BranchKey
676 edm::BranchDescription
636 vector<char>
635 edm::DoNotRecordParents
632 vector<reco::PreshowerCluster>
553 RPCRecHit
553 RPCDetId
548 vector<edm::RefToBase<reco::Track>>
477 vector<edm::HLTPathStatus>
477 vector<EcalRecHit>
477 edm::TriggerResults
477 edm::SortedCollection<EcalRecHit,edm::StrictWeakOrdering<EcalRecHit>>
477 edm::HLTGlobalStatus
436 vector<unsigned long
432 vector<reco::Track>
426 reco::LeafCandidate
426 reco::Candidate
416 edm::reftobase::BaseHolder<TrajectorySeed>
416
edm::Ref<vector<TrajectorySeed>,TrajectorySeed,edm::refhelper::FindUsingAdvance<vector<TrajectorySeed>,TrajectorySee
d>>
380 edm::reftobase::IndirectHolder<reco::Track>
380
edm::Ref<vector<reco::TrackExtra>,reco::TrackExtra,edm::refhelper::FindUsingAdvance<vector<reco::TrackExtra>,reco::Trac
kExtra>>
380 ROOT::Math::DisplacementVector3D<ROOT::Math::Cartesian3D<Double32
t>,ROOT::Math::DefaultCoordinateSystemTag>
373 vector<edm::BranchID>
373 edm::Transient<edm::Parentage::Transients>
366 reco::RecoChargedCandidate
366 reco::RecoCandidate
340 vector<reco::Candidate*>
335
edm::reftobase::RefHolder<edm::Ref<vector<reco::Track>,reco::Track,edm::refhelper::FindUsingAdvance<vector<reco::Track
>,reco::Track>>
331 vector<reco::MuonChamberMatch>
331
map<reco::Muon::MuonTrackType,edm::Ref<vector<reco::Track>,reco::Track,edm::refhelper::FindUsingAdvance<vector<reco
::Track>,reco::Track>>
318 edm::DataFrameContainer
318 EcalDigiCollection
316 vector<reco::PreshowerClusterShape>
236
edm::Ref<vector<reco::SuperCluster>,reco::SuperCluster,edm::refhelper::FindUsingAdvance<vector<reco::SuperCluster>,reco
::SuperCluster>>
232 vector<ROOT::Math::PositionVector3D<ROOT::Math::Cartesian3D<float>,ROOT::Math::DefaultCoordinateSystemTag>>

232
vector<ROOT::Math::DisplacementVector3D<ROOT::Math::Cartesian3D<float>,ROOT::Math::DefaultCoordinateSystemTag>>
230
vector<ROOT::Math::PositionVector3D<ROOT::Math::Cartesian3D<double>,ROOT::Math::DefaultCoordinateSystemTag>>
213
edm::reftobase::RefHolder<edm::Ref<vector<TrajectorySeed>,TrajectorySeed,edm::refhelper::FindUsingAdvance<vector<Traj
ectorySeed>,TrajectorySeed>>
213 edm::reftobase::IndirectHolder<TrajectorySeed>
203
edm::reftobase::Holder<TrajectorySeed,edm::Ref<vector<TrajectorySeed>,TrajectorySeed,edm::refhelper::FindUsingAdvance
<vector<TrajectorySeed>,TrajectorySeed>>
159 vector<vector<vector<int>>
159 vector<edm::Hash<1>>
159 vector<TrackingRecHit*>
159 vector<L1GtPsbWord>
159 vector<L1GtLogicParser::OperandToken>
159 vector<L1GtFdlWord>
159 vector<L1GlobalTriggerObjectMap>
159 vector<HORecHit>
159 vector<HFRecHit>
159 vector<HBHERecHit>
159 vector<DetId>
159 vector<CastorRecHit>
159 edm::SortedCollection<HORecHit,edm::StrictWeakOrdering<HORecHit>>
159 edm::SortedCollection<HFRecHit,edm::StrictWeakOrdering<HFRecHit>>
159 edm::SortedCollection<HBHERecHit,edm::StrictWeakOrdering<HBHERecHit>>
159 edm::SortedCollection<CastorRecHit,edm::StrictWeakOrdering<CastorRecHit>>
159 edm::RefProd<L1MuGMTReadoutCollection>
159 L1GtfeWord
159 L1GlobalTriggerReadoutRecord
159 L1GlobalTriggerObjectMapRecord
159 EEDigiCollection
159 EBDigiCollection
158 vector<trigger::TriggerObject>
158 vector<trigger::TriggerEvent::TriggerFilterObject>
158 trigger::TriggerEvent
158 edm::ValueMap<unsigned int>
157
vector<pair<ROOT::Math::PositionVector3D<ROOT::Math::Cartesian3D<double>,ROOT::Math::DefaultCoordinateSystemTag>,
float>>
157
vector<edm::Ref<vector<reco::CaloCluster>,reco::CaloCluster,edm::refhelper::FindUsingAdvance<vector<reco::CaloCluster>,r
eco::CaloCluster>>
156
vector<ROOT::Math::DisplacementVector3D<ROOT::Math::Cartesian3D<double>,ROOT::Math::DefaultCoordinateSystemTag>
>
116
vector<edm::Ref<vector<reco::Track>,reco::Track,edm::refhelper::FindUsingAdvance<vector<reco::Track>,reco::Track>>
116 vector<edm::Ptr<reco::CaloCluster>>
116 vector<Measurement1DFloat>
90
edm::Ref<vector<reco::GsfTrack>,reco::GsfTrack,edm::refhelper::FindUsingAdvance<vector<reco::GsfTrack>,reco::GsfTrack>>
72 vector<CaloTowerDetId>
60 reco::PFCandidate
60 reco::CompositeCandidate
60 edm::RefVectorBase<unsigned int>
60
edm::RefVector<vector<reco::PFBlock>,reco::PFBlock,edm::refhelper::FindUsingAdvance<vector<reco::PFBlock>,reco::PFBloc
k>>
60 edm::Ptr<reco::PFCandidate>
60 edm::OwnVector<reco::Candidate,edm::ClonePolicy<reco::Candidate>>
60 ROOT::Math::PositionVector3D<ROOT::Math::Cartesian3D<float>,ROOT::Math::DefaultCoordinateSystemTag>
45
edm::reftobase::RefHolder<edm::Ref<vector<reco::GsfTrack>,reco::GsfTrack,edm::refhelper::FindUsingAdvance<vector<reco::
GsfTrack>,reco::GsfTrack>>
45
edm::reftobase::Holder<reco::Track,edm::Ref<vector<reco::GsfTrack>,reco::GsfTrack,edm::refhelper::FindUsingAdvance<vect
or<reco::GsfTrack>,reco::GsfTrack>>
24 Vector3DBase<float,LocalTag>

24 RecSegment
24 PV3DBase<float,VectorTag,LocalTag>
24 CLHEP::HepSymMatrix
24 CLHEP::HepGenMatrix
15 vector<CSCSegment>
15 vector<CSCRecHit2D>
15 CSCSegment
14 edm::Transient<edm::ProcessConfiguration::Transients>
8 TObjString
6 vector<DTRecHit1D>
6 TStreamerObjectAnyPointer
6 TAttMarker
6 TAttLine
6 DTRecSegment2D
3 map<unsigned int,string>
3 edm::Transient<edm::ProcessHistory::Transients>
3 DTSLRecSegment2D
3 DTRecSegment4D
3 DTChamberRecSegment2D
2 vector<vector<int>>
2 vector<edm::ProcessConfiguration>
2 edm::LuminosityBlockID
1 vector<vector<unsigned int>>
1 vector<edm::ProcessHistory>
1 vector<edm::IndexIntoFile::RunOrLumiEntry>
1 vector<edm::Hash<2>>
1 set<edm::BranchID>
1 map<edm::BranchKey,edm::BranchDescription>
1 map<edm::BranchID,set<edm::BranchID>>
1 edm::Transient<edm::ProductRegistry::Transients>
1 edm::Transient<edm::IndexIntoFile::Transients>
1 edm::RunID
1 L1GtTriggerMenuLite