RNTuple API Review

Amit Bashyal
06/26/2024

Use of RNTuple in CAF

- Writing CAF Objects
- Reading CAF Objects
- Writing CAF Objects as flat ntuples
- Reading CAF Objects as flat ntuples

API’s besides reading and Writing

- RNTupleDescriptor → Get information on all top level Fields

- RNTupleView → Could be implemented in CAF Reader for selective I/O
  - Only reading certain SR Objects

- RNTupleImporter →FlatCAF naming scheme uses “.” and “..”
  - Existing FlatCAF needs to use RNTupleImporter with SetConvertDotsInBranchNames

A look at *art* (Data processing Framework)

- Currently used by ProtoDUNE
  - DUNE will have different framework but could anticipate several (good) features of art going into the new framework
- Went through the component of art framework that handles I/O
- Uses ROOT::TTree for I/O
- What would it take (in general) for art to adopt ROOT::RNTuple
- DISCLAIMER: This is mostly for educational purpose

Writing art::Events (General Overview)

Manage access to the ROOT files and related TTrees

Add/Edit data products. **Provenance** of the data products is also created/edited using **Provenance/BranchDescription**

Serves as foundational class for all data product types that are added in **art::Event**

User defined c++ class data products are serialized/deserialized as EDProducts.

*Has virtual functions and pointers

Data products along with the provenance are persisted in ROOT::TTree as ROOT::TBranch

Each data product persisted in unique TBranch.

Writing art::Events (General Overview)

Manage access to the ROOT files and related TTrees

Add/Edit data products. **Provenance** of the data products is also created/edited using **Provenance/BranchDescription**

Serves as foundational class for all data product types that are added in **art::Event**

User defined c++ class data products are serialized/deserialized as EDProducts.

*Has virtual functions and pointers

Data products along with the provenance are persisted in ROOT::TTree as ROOT::TBranch

Each data product persisted in unique TBranch.

Writing art::Events in RNTuple (General Overview)

RNTupleWriter/RNTupleParallelWriter to create an input file

Data model is persisted as RNTupleModel.

RNTupleModel is a collection of serialized C++ types.

Serves as foundational class for all data product types that are added in **art::Event**

User defined c++ class data products are serialized/deserialized as EDProducts.

Need to make sure that data models are supported by RNTuple.

Data products along with the provenance could be persisted in RNTupleModel as RFields and (de) serialized into columns.

Certain C++ types could be mapped into multiple columns.

**Shown for illustration purpose. Actual implement could be quite different*

Writing art Events (RootOutputFile)

```
RootOutputFile::RootOutputFile(OutputModule* om,
                               string const& fileName,
                               ClosingCriteria const& fileSwitchCriteria,
                               int const compressionLevel,
                               int64_t const saveMemoryObjectThreshold,
                               int64_t const treeMaxVirtualSize,
                               int const splitLevel,
                               int const basketSize,
                               DropMetaData dropMetaData,
                               bool const dropMetaDataForDroppedData)
  : om_{om}
  , file_{fileName}
  , fileSwitchCriteria_{fileSwitchCriteria}
  , compressionLevel_{compressionLevel}
  , saveMemoryObjectThreshold_{saveMemoryObjectThreshold}
  , treeMaxVirtualSize_{treeMaxVirtualSize}
  , splitLevel_{splitLevel}
  , basketSize_{basketSize}
  , dropMetaData_{dropMetaData}
  , dropMetaDataForDroppedData_{dropMetaDataForDroppedData}
  , filePtr_{TFile::Open(file_.c_str(), "recreate", "", compressionLevel)}
{
  using std::make_unique;
  // Don't split metadata tree or event description tree
  metaDataTree_ = RootOutputTree::makeTTree(
    filePtr_.get(), rootNames::metaDataTreeName(), 0);
  fileIndexTree_ = RootOutputTree::makeTTree(
    filePtr_.get(), rootNames::fileIndexTreeName(), 0);
  parentageTree_ = RootOutputTree::makeTTree(
    filePtr_.get(), rootNames::parentageTreeName(), 0);
  treePointers_[0] =
```

Split-level does not exist.

Basket-Size → Page-Size

- API currently unavailable but has come up during ATLAS/CMS RNTuple review.
- Could be more significant for DUNE compared to ATLAS/CMS.

Writing art::Event (RootOutputTree)

```
TTree*
RootOutputTree::makeTTree(TFile* filePtr,
                          std::string const& name,
                          int const splitLevel)
{
  TTree* tree = new TTree(name.c_str(), "", splitLevel);
  if (!tree) {
    throw Exception{errors::FatalRootError}
      << "Failed to create the tree: " << name << "\n";
  }
```

TTree → RNTupleModel

```
namespace {
  void
  fillBranches(std::vector<TBranch*> const& branches,
               bool const saveMemory,
               int64_t const threshold)
  {
    for (auto b : branches) {
      auto bytesWritten = b->Fill();
      if (saveMemory and bytesWritten > threshold) {
        b->FlushBaskets();
        b->DropBaskets("all");
      }
    }
  }
}
```

```
void fillFields(std::shared_ptr<RNTupleWriter> ntupleWriter,
bool const saveMemory, int64_t const threshold)
{
    int64_t totalBytesWritten = 0;
    if(saveMemory){
        totalBytesWritten += ntupleWriter->Fill();
        if(totalBytesWritten>threshold){
            ntupleWriter->CommitCluster();
            totalBytesWritten = 0;
        }
    }
    else{
    ntupleWriter->Fill();
    }

}
```

A possible implementation of art::fillBranches() in RNtuple

Reading art::Events (General Overview)

Manages opening, closing and accessing events from multiple ROOT files in specific order.
Ensures continuous access of events from multiple ROOT files.

Read event, identify next event or seek random event based on event identifiers from a given ROOT file.

Manages art::Event for Reading.

Provides pointers to read events from run, subrun, file, run-range etc.

Access to the data product from an art::Event using DataProduct Handle

Allows access to DP provenance

*Also allows inserting new data-products in existing art::Event

Data products are de-serialized from TBranch.

Reading art::Event in RNTuple (General Overview)

Read event, identify next event or seek random event based on event identifiers from a given ROOT file.

RNTupleReader supports sequential reading of ROOT Files.

Manages art::Event for Reading.

Provides pointers to read events from run, subrun, file, run-range etc.

Access to the data product from an art::Event using DataProduct Handle

Allows access to DP provenance

*Also allows inserting new data-products in existing art::Event

C++ type Data products are de-serialized from one or multiple columns of RField.

Shown for illustration purpose. Actual implement could be quite different