Event I/O and ROOT

CDF Offline Meeting: 31-Mar-1999
Rob Kennedy

- **US HENP ROOT Users Workshop - 23-25 March 1999**
  CDF Site Report: http://runIIcomputing.fnal.gov/root/reports/cdf
  ROOT I/O: http://runIIcomputing.fnal.gov/root/reports/rootreport.txt

- **ROOT I/O at CDF - 24 March 1999: Talk by RDK**
  Plans/concerns for Data Handling, Offline Classes, & ROOT:
  http://runIIcomputing.fnal.gov/root/reports/kennedy.ps

- **Event I/O and ROOT Meeting - 29 March 1999**

→ **Working Meeting with Rene' Brun, Philippe Canal, FNAL C++ Gurus, CDF Offline, Data Handling, Infrastructure, EDM folks.**
→ **Goal: produce a plan for CDF-ROOT issues and a prioritized list of requests for ROOT development by ROOT team.**
→ **Summary of meeting will be written soon, prioritized list is done.**

Implementing I/O in Classes

➠ Schema Evolution: For now, no change to plans. Developers use rootcint or hand-code Streamer ‘clauses’, which must be inserted by hand into the existing Streamer if/else-if construct. Improvements?
➠ Data Member Support: ROOT Team agrees support for STL containers, string in rootcint for I/O is high priority (not for interpreted macros)... next few weeks? Still will be some limitation on what rootcint will support: inheritance, nested classes, etc.

Four models for enabling use of ROOT I/O by a Class

➠ Adapter - For each class A, adapter (:TObject) has a A* pointer
➠ Decorator - One class (:TObject) calls Streamer in class A
➠ Traditional - StorableObject:TObject - Let’s reconsider. Most ROOT code linked in via TFile, TTree, TBranch use in I/O mods
➠ Hot-wired - *Nothing* derives from TObject. Must provide a dictionary to "new default constructor" an instance while reading.

➠ For I/O each class needs: ClassDef(), ClassImp(), Streamer(), and "new". These methods vary by where Streamer,new are located.

CDF ROOT Event Format

➠ I/O modules read/write this 'ROOT' event, using as output/input an EDM event. User algorithms do not see 'ROOT' format, but do make I/O requests of AC++ based on CDF 'ROOT' event format.
➠ ROOT default: TBranches in a TTree are all filled at one time.

➠ CDF ROOT event = EventId (run, event) and a TList of Tree1s.
➠ Each Tree1 represents a distinct stage of data processing: Raw, Level3, Pad1, Pad2 (if reprocessing), ..., Dst-B1 (Bpads), ... User-X0.
➠ Each Tree1 is read-only once written except perhaps User Tree1s.
➠ A Tree1 contains a TList of Tree2s. A Tree2 stores a group of classes with similar use-patterns, is reasonably easy to extend.
➠ Each Tree2 is a TList of TBranches, also somewhat extendable.
➠ Each TBranch stores one specific class of objects.
➠ Policy undecided on whether ntuples/histos stored in event, too.

➠ Output of Level3, Production is multi-branch, single file mode.
➠ DST and later output will be multi-branch split-file mode files, with each split-file containing a TTree2.

Data Handling, CDF ROOT Event

➠ Two conceptual levels for specifying data I/O for user algorithms:
1) Input DataSet: data stream, data files, group of classes stored
   Relates to which tape file to spool to disk (if not already there)
2) I/O Filter: individual classes to drop or keep for consideration
   Relates to which TBranch to select for I/O from diskfile

➠ Example: For stage = Run 2 Dst-B, Stream = JPsi, Tree = Tracks:
1) Input DataSet = Run 2 Dst-B JPsi_Tracking, Nevents = 1000
2) Input Keep Group = Dst-B::Tracking, Dst-B::Primary_Vertices
3) Input Drop Group = Dst-B::Secondary_Vertices, ...
... one could also specify individual classes in the keep/drop list.

➠ Users who wish to access vastly different data (Dst vs. Raw) or in vastly different patterns (streaming or pick-event) are encouraged to break up job into units with more uniform data access needs.
➠ Users see one ‘Logical’ event. Using the EventId of an event, one can access the event data from any stage of processing for that event. Its not clear yet whether we consider ntuples/histos as part of event.

CDF: ROOT Development by June

➠ ROOT needs a hook for end-of-file handling to pass control from ROOT to the Data Handling system and back again. The Data Handling system at least needs to be able to specify the name of the new file to be opened once a split-file is to be closed, new file opened.

➠ We would like to see an EventId concept integrated with TTrees to support searches for pieces of an event across split-files. For us, the minimum EventId would be (run number, event number) pair. The "stream identifier" concept at CDF would part of the dataset selection (file selection) by the user, and not a part of EventId.

➠ We need a mechanism to isolate errors in schema to the offending class. This would be addressed at least in part by the concept of byte counts being used in the ROOT I/O system, as described by Rene' Brun. (Ie, suppose you read 64 bytes for an object when 66 written.)

➠ We request support for I/O of STL containers, std::string by rootcint. We are not requested interactive use of the STL.

CDF: ROOT Development by June

➠ We would like to see as much as is feasible a reduction in the physical coupling of ROOT classes and the physical (link-time) dependency of ROOT libraries. ...We would like to link in as few ROOT libraries as possible to support this. In particular, we do not wish to link in Graphics and Graphics-related libraries as long as we support the use of ROOT only for Event I/O. Philippe has looked into this.... In some cases, re-arrangement of classes amongst libraries may permit some libraries to be dropped from the link list. In other cases, we may use a null implementation version of a ROOT class which cannot easily be isolated by re-arrangement.

➠ (Lower Priority)More documentation of Event Models and I/O
➠ (Lower Priority) Support for Adapter approach: changes needed?

➠ (Request of CDF) Create a 1 page or so code example of how data will be accessed and used by users. This will be part of the Edm package to be delivered as part of the May/June Milestone.

➠ These items are mostly high priorities for ROOT team anyway...