CDF Multi-Branch Event I/O

-

Introduction to Current

Design and Transition Plans

Robert D. Kennedy

05 September 2001

***Event Data Model*: Manages reading/writing of events to/from data files, and manages access to objects in an event.**

To date, users could only access entire events, and only recently could access events non-sequentially. We now have a working prototype (below AC++) that can access *pieces* of *selected* events, and which restores the responsibility boundary between the EDM and the I/O modules.

This talk is a brief outline of the Multi-Branch Event I/O model implemented in this prototype, and how we propose to make this feature available to users soon.

1) Past and Present "Event Model": Structure, Capabilities

2) Multi-Branch Event Model: Structure, User Interface

3) "Branch Set" Definitions: Conceptual groups of branches

4) Status and Transition Plans

Past and Present "Event Model"

☞ **SeqRoot (RDK) - All event data stored in one ROOT TTree, in one TBranch. All event data accessed sequentially.** This meets our minimal defined requirements for ROOT event data I/O from "format decision". It serves our Online needs well, as well as our Production Farm input needs. It is clearly not optimal for all Production output, secondary/tertiary data set, and end-user use-cases.

☞ **RootFileStream (Fedor Ratnikov) - Like SeqRoot. Adds event selection criteria to second TBranch ("EventInfo"), and capability to add non-EDM branches. Users can now, for instance, randomly access events by (run,trigger).**
This adds capabilities requested by users to skip events, move backwards and forwards through an event file, select events by (run,trigger), manage BoR events efficiently during random access. Still, all of event is read, whether the user needs it or not.

☞ **ROOT solution - multi-branch events**
We have known this since the "format decision", and are now working towards this at top priority. Most other implementations (at other experiments) differ since they assume a very static event data model: exactly N classes are stored, all defined long before data taking begins in earnest. We have maintained a dynamic development model, and have had to develop a CDF-compatible approach to efficiently implementing multi-branch events.

Multi-Branch Event Model

☞ Event Data (and only that) is stored in one TTree "EventData", which contains:

a) One "Primary" data branch containing the EventRecord instance + any "branchless" objects. This branch is always enabled for I/O.

b) "Secondary" data branches containing one "SubEvent" each.

☞ Exactly one Storable Object class is mapped to one secondary data branch. If the branch "XYZ" corresponding to class "XYZ" is enabled, then that class's object go there. Otherwise "XYZ" "remains" in the Primary branch.

☞ Events could contain many branches, overwhelming users (who knows what algorithm relies on what classes?). *Branch Sets* are conceptual groups of branches/classes with similar use-cases. Most users will work with these 10-16 relatively stable branch sets. The intrepid can fine-time access by branch.

☞ I/O Modules will provide commands for users to enable/disable BranchSets, Branches, etc. This is in addition to the existing I/O filtering capabilities, which (mostly) complement branch selection.

☞ Event Selection information is stored in a separate TTree "EventInfo", and is not "visible" to users. This is only used to efficiently implement random access/event selection features.

Branch Set Definitions (example)

☞ **Note: This have not been fully reconciled PADS plans, and are presented as an illustration of what the definitions *MIGHT* be.**

BranchSet	Examples of classes assigned
Simulation	SimulationObjects classes
SiliconRawData	SIXD, ISLD, Si*Set, ...
TrackingRawData	COTD
CalorRawData	Calorimetry D banks
MuonRawData	Muon detector D banks
OtherRawData	TL2D, TFRD, TMPD, ...
Tracking	CdfTrack* , ...
Calorimetry	CalData, *Cluster*, ...
MuonRecon	Muon*Collm, CM*E, ...
SqueezedCOT	COTQ
SqueezedTracks	TRKQ
Vertex	Vertex, VertexColl, ...
EmObject	CdfEmObjectColl
Jet	CdfJetColl
Muon	CdfMuon, CdfMuonColl
OtherHLO	CdfMet
GrabBag	(anything not assigned above)

Status and Transition Plans

☞ Implementation at the EDM level for event mode I/O (objects into memory) is working.

Some conflicts exist with DHMods/EventInfo which will be addressed to allow the working implementation to be reposited (some parts now commented out in what is in development.

☞ Yet-to-implement: Multi-branch Block I/O

Design exists. Implementation after EventInfo issue fixed.

☞ Document EDM-I/O module interaction

Restores EDM responsibility for content/structure of event data files. This doc will be needed by I/O module maintainers as they rework I/O modules to use Multi-Branch File classes.

☞ Multi-Branch I/O Performance tests

☞ Rework, re-certify I/O modules, File classes.

Done "off to the side" to insure continuous service for users. This will insure that there is once again only one implementation of the CDF data file format. SeqRoot will be re-implemented in the new classes with its Online extensions retained intact.

☞ Switch in certified DH I/O Mods2.

☞ Future work and potential extensions:

Performance overall. Replace "hiding" by multi-mode event I/O.
Performance and per-branch compression (looks promising).
Ntuple-like EventCuts that can be extended for Physics cuts.
User-defined TTrees (no sync guarantee) stored in EDM files.

[Discussion Summary]

☞ **Online can continue to use SeqRoot (single Tree/single Branch) format.** In fact there is no need for anything else until after Production input. The SeqRoot File class may be re-implemented some day with the multi-branch File class, but its interface and behavior will remain as is.

☞ **All data branches for a single event will continue to be stored in a single data file.** Splitting branches between different data files, while technically possible with ROOT, requires much more development work to be done in the Infrastructure, I/O modules, and in the Data Handling system. All files containing pieces of the event to be read must be delivered to the user's job before that event can be read. Early design discussions (just after the "format decision", some quite some time ago) concluded that this was either not feasible or involved more work than benefits.

☞ **If Muons require Tracks to be present in order to function, what protects users from enabling Muons, disabling Tracks, and getting different results.** In the current plan, nothing. The concern here is about objects that depend on others for their behavior, not about Link restoring-on-demand. (*Post-meeting*: we might allow branch sets to overlap. Then, consider recording class behavior dependencies in class descriptions in addition to module/algorithm's listing class dependencies to automate setup of optimal input performance.)