Pointer Skims

Refereneces

From Core Mnutes March 20, 2007:

ROOT data indexing: (Eric C.) ROOT-based metadata scheme (used by BABAR) containing pointers to locate all known data for a given event. The basic idea involves storing (File, Tree, Entry) in an external table, pointed to by a single pointer in an NTuple (index file), but with some additional flexibility (works with xrootd, including when data has been migrated to tape). Tools to read & copy the event data also provided. See https://confluence.slac.stanford.edu/download/attachments/20011/Event_Collections.pdf. BABAR code is all online in their CVS repository, module KanEvent. This scheme might find application in GLAST for interleave, skimming, and analysis scenarios.

Relational DB option

In a nutshell: while at the collaboration meeting it became apparent that the goals for this "system" were rapidly changing and the scope of the concept is increasing dramatically. For example, the pipeline people would like to be able to categorize events (pass CNO filter, pass MIP filter, GCR event, etc.) and they think it would be natural to just write out an event/run number to do this, then use the "system" to read the events back. In thinking about it a bit more, it seems to me that the problem neatly divides into two pieces. One piece is the part that given a run and event number (which I am told are the unique identifiers for all events) returns the information on where to find the actual data. The other piece is the code that, given the run and event number and the information on where to find the data, returns the actual data requested (which can be various root trees - mc, relation, recon, digi, etc., or ntuples, or...).
My argument is that the first piece is best done with a relational database. With a relational database you would use the run/event number as the key and the enter, only once, all the information on where to find the various bits of data associated with it. In addition, you can also have a few more bits of information which will further categorize the event which can be used during a query to identify the event in some particular way. I think this type of system will be far more extensible and much easier to manage than a pile of root files which would try to do this same sort of thing. In addition, it would be straightforward for the pipeline guys to hook into this automatically to fill it. I also think it would be very easy to transport the database to other installations which might be repositories of large datasets (e.g. Lyon).
In any case, I think this approach also neatly divides the problem into two pieces which may well make it much easier to implement.
And, finally, I also discovered that Joanne has already provided all the tools necessary for implementing such an approach, including some very nice gui tools for looking at stuff that is in there. There would be cost involved in understanding her stuff and then wrapping what we wanted to do around the outside of it but this would be far simpler than trying to invent something ourselves.
I have the BaBar code downloaded on my laptop. I had started to think about building it but realized I needed some include files. When I asked Tom where I might find them he cautioned me that I was beginning to pull on a very long string and it might be best to not try to do that. So... whatever we decide to do, it sounds like our best approach may be to take the concept but do our own implementation.
Tracy

Use Cases

Level 0 Interleave

Feed the simulation a library of background events it can randomly access in full from the original data files. So would we envision Interleave querying the DB directly during a run to get the next background event?For the interleave we might try to do it like the ntuple interleave, where you set up to read all events within a given bin, on the assumption that several events will be generated from that bin before going to a new bin. But I think there are few enough interleave events that querying the db each time you want one is not so much overhead. I suppose we will see!

Pruner Skimming

A search is performed and N events are found that satisfy the criteria - with run/event ids in hand, the system finds the data files containing the events, pulls them out and ships them off for delivery to the user doing the search.

User Analysis

User Joe has his favorite N events and he wants to keep a run/event list handy to share with his friends. His friends can then use that list to retrieve the full data themselves.right. the db produces a list which the root reading code uses to return the events as you need them.