Subject: Some issues with transient members of PndEmcCluster
Posted by Elwin Dijck on Thu, 06 Aug 2009 14:42:53 GMT
View Forum Message <> Reply to Message

Hello all,

Recently I got some weird results while analyzing cluster information from EMC simulations when using the PndEmcXClMoments class. There are some issues with the transient elements in the PndEmcCluster class when reading from ROOT files, which might be a well known problem, but I didn't see it mentioned anywhere and it took me some time to figure out what was really happening from the results I got. So I think it might be good to change some things or at least document the behaviour. For instance, running the following macro will show the problem I ran into (full file attached):

Code[code]TFile file("emc.root");
TTree *tree = (TTree *)file.Get("cbmsim");

TClonesArray *clusters = new TClonesArray("PndEmcCluster");
tree->SetBranchAddress("EmcCluster", &clusters);

for (Long64_t i = 0; i < tree->GetEntriesFast(); ++i)
{
  tree->GetEntry(i);
  cout << endl << "----- Event #" << i << endl;

  PndEmcCluster *cluster = (PndEmcCluster *)clusters->At(0);

  cout << "Xmoments used directly - " << cluster->Xmoments().AbsZernikeMoment(2, 0) << endl;

  PndEmcXClMoments xmoments(*cluster);
  cout << "Separate Xmoments - " << xmoments.AbsZernikeMoment(2, 0) << endl;
}[/pre][/align]

The two different ways of using a PndEmcXClMoments object should just return the same value, but for instance running this code for the attached simulation file (containing two 10 GeV pi0 events), you actually get:

----- Event #0
Xmoments used directly - 0.965463
Separate Xmoments - 0.965463

----- Event #1
Xmoments used directly - 517.327
Separate Xmoments - 0.987504

For the first event both results are the same, but for all later events directly using the PndEmcCluster::Xmoments() function will give garbled results.

When loading an event from a split branch from a tree, ROOT I/O only seems to overwrite the data that's actually stored in the file (also without using streamers), so not the transient pointer that is stored after the first call to PndEmcCluster::Xmoments() and these pointers just keep

their old values because the TClonesArray reuses memory without destruction/construction.

It's actually rather easy to 'avoid' the problem, by copying the PndEmcXClMoments object before using it, because in this case the copy-constructor doesn't just copy things but recalculates them (which I think shouldn't be necessary):
PndEmcXClMoments xmoments = cluster->Xmoments();
cout << xmoments.AbsZernikeMoment(2, 0) << endl;

For the transient elements in some other classes like the TCI's stored in PndEmcDigi or PndEmcWaveform there are ValidateTCI() functions that force a recalculation, but in case of these cluster information objects there aren't any. Note that this actually only leads to problems with PndEmcXClMoments (Edit: PndEmcClusterDistances should have the same problems, though I haven't used that class), as these are the only PndEmcAbsClusterProperty objects storing information that becomes invalid when their parent cluster object is overwritten.

Instead of relying on transient elements not storing data or a copy-constructor that doesn't just copy, I could think of several ways to do something about this:

Don't store any data (that can become invalid) in the transient cluster elements (not so efficient for the Zernike moments, defeats the purpose of storing PndEmcAbsClusterProperty pointers in the clusters).
Don't store those transient pointers in PndEmcCluster at all: just return a new object each time. Only the PndEmcXClMoments objects are non-trivial to construct so in most cases it wouldn't matter having to construct new objects each call anyway.
Require to call TClonesArray::Delete() for the cluster array before loading each event from the tree (rather against the idea of using TClonesArrays and should be somewhat slow, but since the number of clusters is never that big this hardly matters).
Implement a function Clear() in PndEmcCluster that resets the transient pointers to 0 and deletes any existing PndEmcAbsClusterProperty objects, and require to call TClonesArray::Clear("C") for the clusters array before loading an event from the tree (similar to the previous one, but without destructing/constructing the cluster objects themselves).
(Not a solution) I noticed PndEmcDigi or PndEmcWaveform have custom streamers and thought that maybe that could also be used to reset the transient elements of clusters. But it seems this only works when the cluster branch is not split.
Actually this also means that the custom streamers of PndEmcDigi and PndEmcWaveform are never called for those branches, although the streamer of PndEmcDigi is called for the digis stored in the fDigiList of clusters (since digis are stored twice, see this thread). Why are those custom streamers there if they're not used in most cases? ROOT does give some warnings about this: "Warning in <TTree::Bronch>: Using split mode on a class: PndEmcWaveform with a custom Streamer".

What are the opinions on this? Is there a better solution? Sorry about the long story.

Best regards,
Elwin Dijck

File Attachments

1) test_xmoments.C, downloaded 533 times
2) emc.root, downloaded 534 times

Subject: Re: Some issues with transient members of PndEmcCluster
Posted by StefanoSpataro on Thu, 06 Aug 2009 16:00:59 GMT
View Forum Message <> Reply to Message

Hi,
I do not like so much the actual transient structure of the emc properties, I think some fixed numbers could be maybe much easier to handle.
However, Dima is the "expert" of the cluster structure, then maybe he could comment about your proposals.
Thanks for your checks.