Subject: Overview of Different Methods of Running Over TTree / TNTuple
Posted by Andreas Herten on Mon, 23 Jun 2014 12:47:52 GMT
View Forum Message <> Reply to Message

There are different methods you can employ when running over ROOT's TNTuples and TTrees to generate plots for your work: Loop over every single branch with tree->SetBranchAddress, or just take all entries and project them onto a histogram, either with tree->Draw() or tree->Project().

I was interested in the pros, the cons, and the performances.
So I dug into analyzing TTrees and compared different approaches.
Here's my result: http://static.andreasherten.de/2014/06/23/ROOT-NTuple-Analysis.html -- a summary follows.

Do you do anything differently when analyzing trees? Do you know of any cool shortcuts? Or ways to do things more efficiently? I love to hear them in the replies.

Summary:
All links go to the corresponding sections on the detailed page.
The most explicit way to get to the data in a TTree's branch is this:

```
float value;
tree->SetBranchAddress("branchValue", &value);
for (int i = 0; i < tree->GetEntries(); i++) {
    tree->GetEntry(i);
    std::cout << value << std::endl;
}
```

In the loop, value can be used to fill a histogram, to print, to create a TLorentzVector for further analysis, ... This requires a lot of temporary variables, has a loop and is, in general, quite a lot of code. A speed-up can be gained by disabling individual, not-used branches with SetBranchStatus, more structured code can be achieved when creating small structs as data containers for your variables (e.g. for all daughter particles).

ROOT offers a shortcut to get data from a branch to ones own histogram. Project and Draw. The following two statements are equal:

```
TH1F * hist = new TH1F("hist", "Very Histogram", 100, 0, 1.0);
tree->Project("branchValue", "hist");
tree->Draw("branchValue >> hist");
```

If you don't want a TCanvas popup when invoking Draw, add a "goff" as a third parameter. Accessing values in branches like this is very efficient and has only few lines of code. You can Project/Draw under conditions (Project("branchValue", "hist", "branchValue > 23 && branchValue < 1337") and do more sophisticated stuff when also employing TEntryLists.

Performance: When running in macro (uncompiled) mode, Project/Draw is two times faster then the other approach. When running compiled (with root macro.C+), the explicit variable declaration is two times faster. At least for my tested example case.