Status Report

2015. 10. 14

Nam Jong Woo

Action Item

- ROOT
  - TTree
  - TBranch
  - TChain

TTree

- Store large quantities of **same-classs** objects
- **TNtuple** : TTree with only floating-point numbers
- Fill branch buffers with leaf data
- Buffers are written to disk when it is full
- Objects are collected and written a bunch at a time
- Take advantage of compression
- Reduces header of each objects
- Optimize data access
- Independent from each other branches
- Do not need to read entire event every time

TTree

- **TTree *tree = new TTree(name, title)**
  - Creates a Tree with name and title.
- TBranch *branch = tree->Branch(branchname,className,object, bufsize, splitlevel)
- tree->Fill()
  - loops on all defined branches and for each branch invokes the Fill function.

TBranch

- To add a TBranch to a TTree use TTree::Branch()
  - Branch ("branch name", address of first variable, string leaf list);
  - eg. tree -> Branch ("staff", &staff.dat, "cat/I:division:flag:age:…)
- leaf list: <name>/<type>:<name>/<type>
- buffer size
- split-level
  - 0: no split, 1-99
  - split branch is faster to read slower to write

Writing the Tree

```
TFile f("tree1.root","recreate");
TTree t1("t1","a simple Tree with simple variables");
Float_t px, py, pz;
Double_t random;
Int_t ev;
t1.Branch("px",&px,"px/F");
t1.Branch("py",&py,"py/F");
t1.Branch("pz",&pz,"pz/F");
t1.Branch("random",&random,"random/D");
t1.Branch("ev",&ev,"ev/I");

//fill the tree
for (Int_t i=0;i<10000;i++) {
  gRandom->Rannor(px,py);
  pz = px*px + py*py;
  random = gRandom->Rndm();
  ev = i;
  t1.Fill();
```

Reading the Tree

```
TFile *f = new TFile("tree1.root");
TTree *t1 = (TTree*)f->Get("t1");
Float_t px, py, pz;
Double_t random;
Int_t ev;
t1->SetBranchAddress("px",&px);
t1->SetBranchAddress("py",&py);
t1->SetBranchAddress("pz",&pz);
t1->SetBranchAddress("random",&random);
t1->SetBranchAddress("ev",&ev);

//create two histograms
TH1F *hpx   = new TH1F("hpx","px distribution",100,-3,3);
TH2F *hpxpy = new TH2F("hpxpy","py vs px",30,-3,3,30,-3,3);

//read all entries and fill the histograms
Long64_t nentries = t1->GetEntries();
for (Long64_t i=0;i<nentries;i++) {
  t1->GetEntry(i);
  hpx->Fill(px);
  hpxpy->Fill(px,py);
}
```

Friends

- friendship : unrestricted access to the friends data
  - like adding branch without risk of damage
- # of entries in friends must be equal or greater to original tree
- tree1->AddFriend(“tree2”, “file2.root”);
- tree1.Draw(“v1:tree2.v2”); // v1 from tree1, v2 from tree2

TChain

- List of ROOT files containing same tree
- TChain chain(“T”);
  - chain.Add(“file1.root”);
  - chain.Add(“file2.root”);
  - chain.Add(“file3.root”); // each file contains tree “T”
  - chain.SetBranchAddress(branchname, address)
- TChain::AddFriend
  - TChain ch(“t”);
  - TChain ch1(“t1”);
  - ch.AddFriend(“t1”);
  - ch.Draw(“var:t1.v1”); // var from t, v1 from t1