What do you need to write a PRL

- STAR Data (exists)
- Good Physics Idea (hopefully exists)
- Analysis Maker (I'll tell you how)

A common-MuDst tutorial

or

Disappearance of back-to-back high-p_T hadron correlations with <200 lines of code

~1 citation per 5 lines of code!

This Talk was created by

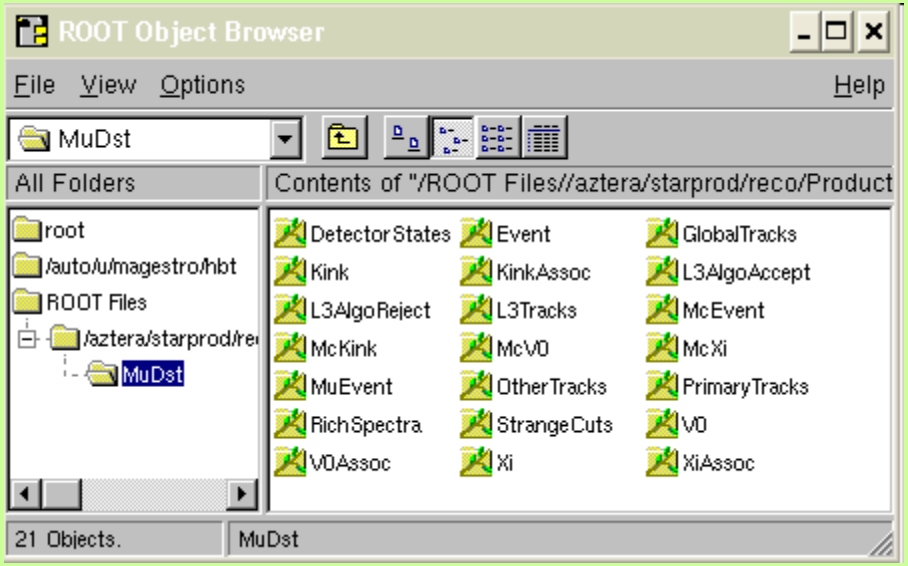
Dan Magestro

The Ohio State University

for the software/computing group

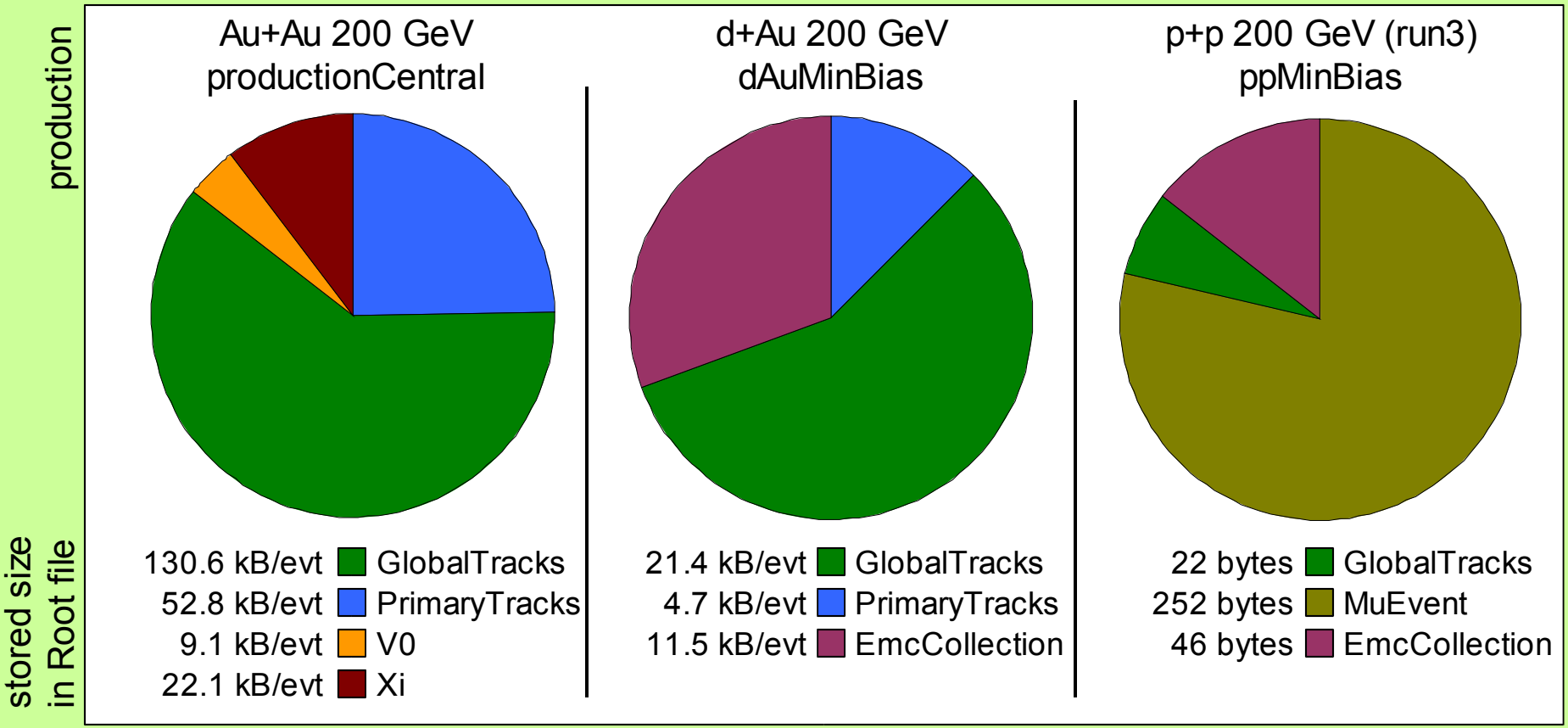
What is the common MuDst?

- Overview
  - Standardized, cross-pwg format for data storage
  - Developed by Frank Laue, Winter/Spring 2002
  - Data format based on native ROOT classes, can be read without root4star
- MuDst tree organized into branches
  - Event information, track & V0 collections, Detector & trigger systems, etc.
  - ROOT provides possibility to read subset of branches



MuDst structure

- Flexible format, composition depends on dataset:

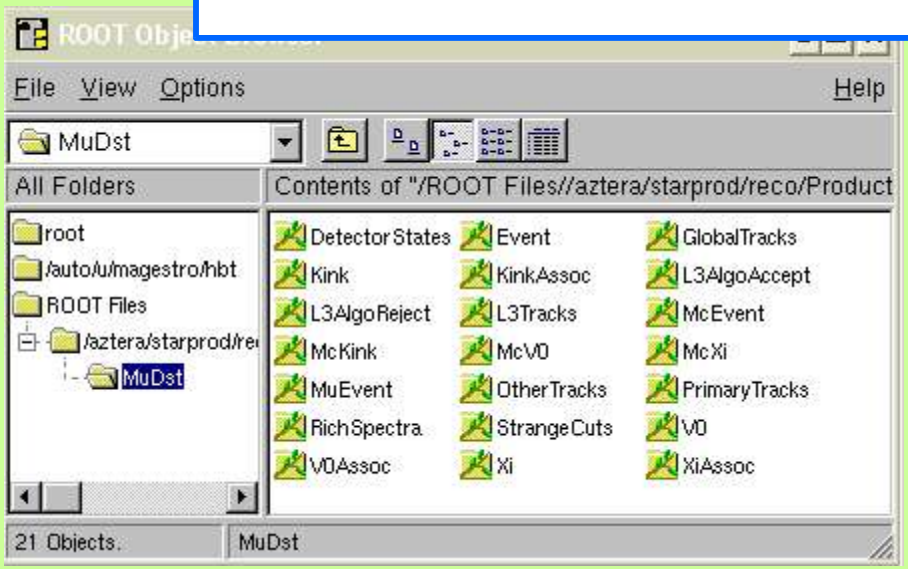


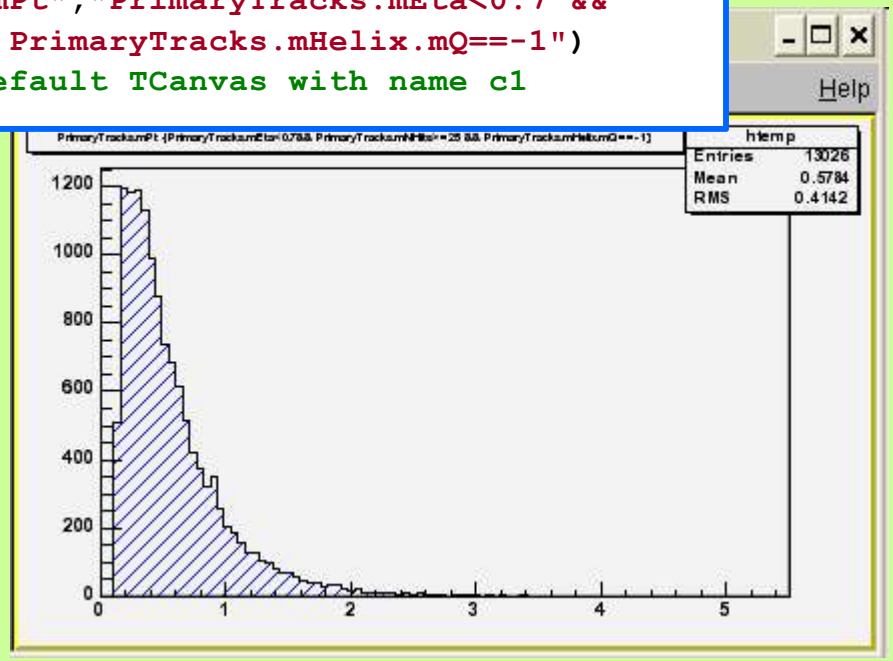
Some ways to read a MuDst (1/3)

1. Open a MuDst file, a TBrowser, and off you go

- Fast way to read the data (just requested quantities are read)
- Good for quick checks, data sanity, etc.
- TChains of TFiles can be used to increase stats (not shown here)
- Be aware that some quantities are "packed" (i.e. Pid, NSigma's in tracks)

```
root[0] TFile f("st_physics_2269018_raw_0020.MuDst.root")
root[1] TBrowser b
root[2] MuDst->Draw("PrimaryTracks.mPt","PrimaryTracks.mEta<0.7 &&
        PrimaryTracks.mNHits>=25 && PrimaryTracks.mHelix.mQ==-1")
<TCanvas::MakeDefCanvas>: created default TCanvas with name c1
```





Some ways to read a MuDst (2/3)

1. Use native Root commands to process TTree/TBranch's
   - MuDst: Data are stored in native Root collections (unlike StEvent)
   - Individual branches can be read (less i/o)

Assign TClonesArray to the PrimaryTracks branch

Read PrimaryTracks branch, fill array with GetEntry()

Loop over tracks, get pointers to each track

Fill histogram with phi difference using StMuTrack method

```
TFile mFile("st_physics_2269018_raw_0020.MuDst.root");
TTree *tree = (TTree*) mFile.Get("MuDst");

TClonesArray *array = new TClonesArray("StMuTrack",10000);

TBranch *brTracks = tree->GetBranch("PrimaryTracks");
tree->SetBranchAddress("PrimaryTracks",&array);

TH1F *dPhi = new TH1F("dPhi","dPhi",100,-3.15,3.15);

int iEvent = 0;
while( brTracks->GetEntry(iEvent++) ) {

   int nTracks = array->GetEntriesFast();

   for(int i=0; i<nTracks-1; i++) {
      StMuTrack *track1 = (StMuTrack*) array[](i);

      if(track1->pt()>2.0 && track1->flag()>0) {
         for(int j=i+1; j<nTracks; j++) {
            StMuTrack *track2 = (StMuTrack*) array[](j);

            if(track2->pt()>2.0 && track2->flag()>0)
               dPhi->Fill( track1->phi() - track2->phi
() );
         }
      }
   }
}
```

deltaPhiWithRootCommands.C

Some ways to read a MuDst (3/3)

1. Let StMuDstMaker do the work for you! ⟵ ***This talk***

- StMuDstMaker handles all i/o
- All event, track info accessed with class methods
- THIS IS THE WAY TO GO!

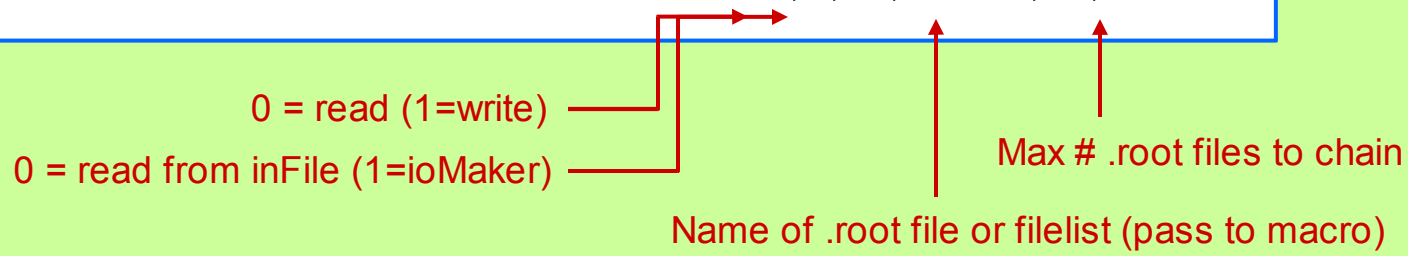
Instantiating StMuDstMaker

- StMuDstMaker can take file lists in many formats
  - Contents of directory
  - Filename filtering possible
  - Lists of files

```
StMuDstMaker *muDstMaker = new StMuDstMaker(     int mode,
                                                 int nameMode,
                                                 const char *dirName="./",
                                                 const char *fileName="",
                                                 const char *filter=".",
                                                 int maxfiles=10,
                                                 const char
*name="MuDst" )
```

- Using scheduler (.list) or filelist (.lis):
  - Instantiate after creating an StChain:

```
StMuDstMaker *muDstMaker = new StMuDstMaker(0,0,"",inFile,"",100)
```



Commonly used branches, methods

GlobalTracks, PrimaryTracks

```
StMuTrack *muTrack = (StMuTrack*) next();

muTrack->eta()          // pseudorapidity
muTrack->phi()          // az.angle (radians)
muTrack->pt()           // transverse mom.
muTrack->dEdx()         // energy loss in TPC
muTrack->charge()       // +1 or -1
muTrack->nHits()        // TPC hits
muTrack->nHitsFit()     // TPC hits used in fit
[...]

muTrack->dca()          // StThreeVectorF
muTrack->momentum()     // StThreeVectorF
muTrack->helix()        // StPhysicalHelixD
[...]

muTrack->pidProbPion()  // 0. <= pidProb <= 1.0
muTrack->nSigmaKaon()   // nSigma (bethe-bloch)
[...]
```

track quantities

vectors and helices

calculated pid quantities

MuEvent

```
StMuEvent *muEvent = muDst->event();

muEvent->refMult()
muEvent->primaryVertexPosition().z()
muEvent->magneticField()
muEvent->ctbMultiplicity()

muEvent->triggerIdCollection()
muEvent->fpdCollection()
```

event-wise quantities

detector- and trigger-wise collections

Using StMuDstMaker in macro (i.e. no analysis "maker")

3 steps

1. Make an StChain
2. Instantiate StMuDstMaker
3. Get pointer to event and tracks within event loop

event loop

accessing event, tracks

iterate over tracks with TIter

Warning: analyzing from uncompiled macros is much slower than from compiled code!

```
void anaMuDstWithoutMaker(TString inFile, int nEvents=10000)
{

  //--------- Instantiate chain and MuDst reader --------//

  StChain *chain = new StChain;
  StMuDstMaker* muDstMaker
           = new StMuDstMaker(0,0,"",inFile,"");

  StMuEvent *ev = 0;
  StMuTrack *track = 0;
  TClonesArray *tracks = 0;

  //------------ The STAR chain Event loop -----------//

  chain->Init();

  for (int iev=0; iev<nEvents; iev++) {
    chain->Clear();
    int iret = chain->Make(iev);

    ev = muDstMaker->muDst()->event();
    tracks = muDstMaker->muDst()->primaryTracks();

    //---- Apply event cuts ----//
    if(fabs(ev->primaryVertexPosition().z())<25.) {

      TIter next(tracks);
      while( track = (StMuTrack*) next() ) {
        // Put track cuts and track-wise analysis here
        cout << "track pt = " << track->pt() << endl;
      }
    }
  }

  chain->Finish();
  delete chain;
}
```

Example: A simple Maker for analysis (1/4)

- Overview of StAnalysisMaker
  - Part of StMaker framework for analyzing data in a chain
  - Overrides Init(), Make(), Clear(), and Finish() methods
  - Can be placed in its own local "shared object" library for fast editing/compiling
  - (by the way, you can call it whatever you want – we'll call ours **DeltaPhiMaker**)

- Our strategy for **DeltaPhiMaker**

constructor: Pass pointer to StMuDstMaker object

Init(): Instantiate & define histograms

every event:

Make(): Get pointers to StMuEvent & StMuTrack for current event, apply event cuts, loop over tracks, apply track cuts, & fill histograms

Clear(): Do nothing (some analyses may clear collections, etc.)

Finish(): Write histograms to file, print some information to stdout

Example: A simple Maker for analysis (2/4)

DeltaPhiMaker class definition:

Inherited from StMaker

Classes for applying track/event cuts

```
#ifndef DeltaPhiMaker_hh
#define DeltaPhiMaker_hh

#include "StMaker.h"
#include "TString.h"

class TFile;
class StMuDstMaker;
class TH1F;

class DeltaPhiMaker : public StMaker {

public:
  DeltaPhiMaker(StMuDstMaker* maker);
  ~DeltaPhiMaker() {;}

  void   Clear(Option_t *option="");
  Int_t  Init();
  Int_t  Make();
  Int_t  Finish();

  void SetFileName(TString name) {mFileName =
name;}

private:
  bool accept(StMuDst*);
  bool accept(StMuTrack*);

  StMuDstMaker* mMuDstMaker;

  int        mNEventsPassed;
  int        mNEventsFailed;
  TString    mFileName;

  //-------- Define histograms here --------//
  TH1F *hDeltaPhi;
  TH1F *hPhiTrig;
  TH1F *hPhiAssoc;
  TH1F *hPt;
  TH1F *hRefMult;
  TH1F *hVz;
```

deltaPhiMaker.h

Example: A simple Maker for analysis (3/4)

DeltaPhiMaker implementation:

Constructor

```
DeltaPhiMaker::DeltaPhiMaker(StMuDstMaker *maker) : StMaker() {
  mNEventsPassed = mNEventsFailed = 0;
  mFileName = "";
  mRefMult[0] = 0;
  mRefMult[1] = 9999;

  mMuDstMaker  = maker;
}
```

Init()

```
Int_t DeltaPhiMaker::Init() {

  //---- Track-wise histograms ----//
  hDeltaPhi = new TH1F("hDeltaPhi","dPhi",64,-3.2,3.2);
  hPhi =  new TH1F("hPhi","Phi",64,-3.2,3.2);
  hPt = new TH1F("hPt","Pt",100,2.,10.);

  //---- Event-wise histograms ----//
  hRefMult = new TH1F("hRefMult","Ref Mult",100,0,1000);
  hVz = new TH1F("hVz","Primary Vz",100,-25.,25.);

  return StMaker::Init();
}
```

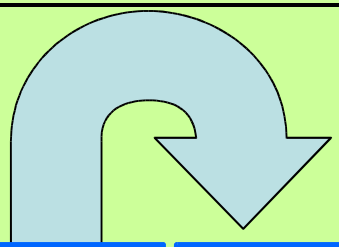
Finish()

```
Int_t DeltaPhiMaker::Finish() {

  //  Output file
  TFile *mFile =  new TFile(mFileName, "RECREATE");
  cout << "The output filename is " << mFileName.Data() <<
endl;

  cout << "Events passed: " << mNEventsPassed << endl;
  cout << "Events failed: " << mNEventsFailed << endl;

  hPhi->Write();
  hPt->Write();
  hDeltaPhi->Write();
  hRefMult->Write();
  hVz->Write();
  mFile->Close();
  return kStOK;
```

deltaPhiMaker.cxx

Example: A simple Maker for analysis (4/4)

DeltaPhiMaker Make():



```
Int_t DeltaPhiMaker::Make() {

  StMuEvent *muEvent=mMuDstMaker->muDst()->event
();

  if(!accept(muEvent)) {
    mNEventsFailed++;
    return kStOk;
  }

  //------ Fill event-wise histograms here -----//
  hVz->Fill(muEvent->primaryVertexPosition().z());
  hRefMult->Fill( muEvent->refMult() );

  TClonesArray *tracks =
       mMuDstMaker->muDst()->primaryTracks();
  TIter next(tracks);

  TList trigTracks, assocTracks;

  StMuTrack *track=0;
  while ( (track = (StMuTrack*)next()) ) {

    if(!accept(track)) continue;

    //---- Fill TList's with high-pT tracks ----//
    assocTracks.Add(track);
    if(track->pt()>=4.0) trigTracks.Add(track);

    //------- Fill track-wise histograms -------//
    hPhi->Fill(track->phi());
    hPt->Fill(track->pt());
  }

  StMuTrack *trigPart = 0;
  StMuTrack *assocPart = 0;

  TIter nextTrig(&trigTracks);
  TIter nextAssoc(&assocTracks);

  //-------- Loop over  --------//
  while ( (trigPart = (StMuTrack*)nextTrig()) ) {

    nextAssoc.Reset();
    while ( (assocPart = (StMuTrack*)nextAssoc()) ) {

      if( trigPart->pt() > assocPart->pt() ) {

        float dPhi = assocPart->phi() - trigPart->phi
();
        if (dPhi>TMath::Pi()) dPhi-=2*(TMath::Pi());
        if (dPhi<-TMath::Pi()) dPhi+=2*(TMath::Pi());
        hDeltaPhi->Fill( dPhi );
      }
    }
  }

  mNEventsPassed++;
  return kStOK;
}
```

The analysis macro

instantiate StMuDstMaker

instantiate DeltaPhiMaker, pass StMuDstMaker pointer

```
void anaDeltaPhiMaker(TString inFile,
                      TString outFile, int nEvents) {

   StChain *chain = new StChain;

   StMuDstMaker *muDstMaker
            = new StMuDstMaker(0,0,"",inFile,".",50);

   DeltaPhiMaker *anaMaker = new DeltaPhiMaker(muDstMaker);
   anaMaker->SetRefMult(20,1000);
   anaMaker->SetFileName(outFile);

   chain->Init();
   chain->EventLoop(1,nEvents);
   chain->Finish();

   delete chain;
}
```

anaDeltaPhiFromMuDst.C

A complex example: StHbtMaker

- StHbtMaker converts StMuDst into StHbtEvent using StMuDstMaker methods

```
StHbtEvent::StHbtEvent(const StMuDst* dst, int trackType) {
  StMuEvent* ev = dst->event();
  mEventNumber = ev->eventNumber();
  mRunNumber = ev->runNumber();
  mTpcNhits = 0;
  mNumberOfTracks = ev->eventSummary().numberOfTracks();
  mNumberOfGoodTracks = ev->eventSummary().numberOfGoodTracks
  mCtbMultiplicity = (unsigned short) ev->ctbMultiplicity();
  mZdcAdc[0] = (unsigned short) ev->zdcAdcAttentuatedSumWest()
  mZdcAdc[1] = (unsigned short) ev->zdcAdcAttentuatedSumEast()
  mUncorrectedNumberOfPrimaries = ev->refMult();
  mPrimVertPos = ev->eventSummary().primaryVertexPosition();
  mMagneticField = ev->magneticField();

  mTriggerWord = ev->l0Trigger().triggerWord();
  mTriggerActionWord = ev->l0Trigger().triggerActionWord();
  [...]

  // copy track collection
  TClonesArray* tracks=0;
  switch (trackType) {
    case 0: tracks = dst->globalTracks(); break;
    case 1: tracks = dst->primaryTracks(); break;
  }
  if (tracks) {
    int nTracks = tracks->GetEntries();
    for ( int i=0; i<nTracks; i++) {
      StHbtTrack* trackCopy
          =new StHbtTrack(dst, (StMuTrack*) tracks->UncheckedA
      mTrackCollection->push_back(trackCopy);
    }
  }
  [...]
```

StHbtEve

```
StHbtTrack::StHbtTrack(const StMuDst* dst, const StMuTrack* t) {
  StMuEvent* ev = dst->event();
  mTrackType = t->type();
  mTrackId = t->id();
  mNHits = t->nHits();
  mNHitsPoss = t->nHitsPoss();
  mNHitsDedx = t->nHitsDedx();
  mNSigmaElectron = t->nSigmaElectron();
  mNSigmaPion = t->nSigmaPion();
  mNSigmaKaon = t->nSigmaKaon();
  mNSigmaProton = t->nSigmaProton();
  mPidProbElectron = t->pidProbElectron();
  mPidProbPion = t->pidProbPion();
  mPidProbKaon = t->pidProbKaon();
  mPidProbProton = t->pidProbProton();
  mdEdx = t->dEdx();
  mChiSqXY = t->chi2xy();
  mChiSqZ = t->chi2z();
  mMap[0] = t->topologyMap().data(0);
  mMap[1] = t->topologyMap().data(1);
  mHelix = t->helix();
  if(t->globalTrack()) mHelixGlobal = t->globalTrack()->helix();
  mCharge = t->charge();


  double pathlength = mHelix.pathLength(ev->primaryVertexPosition());
  mP = mHelix.momentumAt(pathlength,ev->magneticField()*kilogauss);
  mPt = mP.perp();
  StThreeVectorD
      dca(mHelix.at(pathlength) - ev->primaryVertexPosition());
```

StHbtTrack.cc

Other resources

- $STAR/StRoot/StMuDSTMaker/COMMON/macros/
  - Contains several examples (including scheduler example)
  - loadSharedLibraries.C: up-to-date list of libraries needed by StMuDstMaker
- MuDst Hypernews
  - Specialized forum for discussing MuDst issues & questions
- Root manual
  - http://root.cern.ch/root/html/
- And of course...
  - Ask those working with & around you!
- The code from this presentation is available at RCF:

  /star/u/magestro/muDstTutorial/