Stntuple tutorial

Kenichi Hatakeyama
Rockefeller University

Before we start…

- The tutorial is just meant to be an introduction to Stntuple - not a rigorous explanation of all the details (charge from the organizer).

- I have been working on Run 1 analyses until the last summer, and I'm using C++, CDF Run 2 software, root, Stntuples less than a year. I won't be able to answer all the questions you have, but at least I will be happy to find answers to those questions together with you!

Contents:

- Introduction
- Getting started
- Finding Stntuples of your interest
- How to analyze
  - Analysis module example
  - DataBlocks
  - Trigger and DB information
  - Good Run list
- Debegging

What is Stntuple?

- Stntuple is a set of data containers which simplify / standardize the data access.
- Stntuple is formatted in a standard root tree.
- It also provides an interface for analysis codes (e.g. event loop).

Stntuple web page

- http://fcdfhome.fnal.gov/usr/cdfopr/Stntuple/Stntuple.html

Stntuple version 2 - Mozilla

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop http://fcdfhome.fnal.gov/usr/cdfopr/Stntuple/Stntuple.html Search Print

Home Bookmarks Google Fermilab Fermi Linux

STNTUPLE

new user: First read THIS how to report a bug

STNTUPLE manual (Jun 04 2004): ps | ps 2-page | pdf

How to run stnmaker_prod.exe on CAF

STNTUPLE primers (primers migrate into the manual)

STNTUPLE analysis scripts: working examples

How to add a new branch to STNTUPLE

How to use secondary vertex fitter (CTVMFT) with Stntuple

STNTUPLE FAQ Prioritized list of jobs

Talks on STNTUPLE

Optimising DB access for STNTUPLE jobs

history: dev_240 | dev_239 | dev_238 | dev_237 | Other News

STNTUPLE datasets NEW

dev_240 | dev_241 | how to print a dataset

Latest tag: dev_242

base release: 5.3.3

status: STABLE

tagged: **Mar 14 2004**,

last updated: **May 13 2004**,

build instructions

update instructions

timing

validation

event size

known problems

Warning: don't use the development version (taken from the head of the repository)

STNTUPLE News

Jul 13 2004 dev_242 retagged, no changes in formats

Getting Started

What you need for your analysis

- Stntuple executables such as stnmaker.exe, stnmaker_prod.exe (if you need to create your own Stntuple)
- Stntuple shared libraries such as libStntuple_alg.so, libStntuple_loop.so etc
- Stntuple file to be analyzed (many sets of ntuples are already available. http://fcdfwww.fnal.gov/~cdfopr/Stntuple/cafdfc/, http://physics.rockefeller.edu/hatake/phys/stnqcd/)
- Your own analysis code

Getting Started Cont'd

- Prepare your working directory:
  (look at the Stntuple web page for the latest info)

```
] setup cdfsoft2 XXXreleaseXXX(e.g. 5.3.3)
] newrel –t XXXreleaseXXX directory_name
] cd directory_name
] unsetenv USESHLIBS   ( or ] export USESHLIBS= )
] addpkg Stntuple dev_242
```

- Prepare the shared libraries:

```
] gmake Stntuple.shared
```

Getting Started Cont'd

- Prepare executables if you want to make your own ntuples
  - stnmaker.exe (] gmake Stntuple.stnmaker)
    - runs on the output of ProductionExe
    - produces output in Stntuple format
    - doesn't recalculate parameters of input objects
  - stnmaker_prod.exe (] gmake Stntuple._prod)
    - runs on the output of ProductionExe
    - produces output in Stntuple format
    - remakes high-level objects (such as jets)
    - runs some analysis modules that are not part of Production e.g. SecVtxModule, JetProbModule

Access to Stntuples

- Setup

  ] cp Stntuple/scripts/.rootrc .

  ] cp Stntuple/scripts/rootlogon.C .

- Check contents of Stntuple:

  print “branch” information after reading “nevents”

  ] root.exe

  - When Stntuple is on your local node

    root [x] TStnAna x(“stntuple_file.root”);

  - When you access to Stntuple via network (rootd), e.g.

    root [x] x = new TStnAna("root://fcdfdataxxx//cdf/scratch/…/stntuple_file.root");

    root [x] x.PrintStat(nevents)

Typical results from PrintStat

..... branch name	<event size>	<sigma size>	TotBytes	ZipBytes	CompFactor	% of File
HeaderBlock	96.000	0.000	1535941	302897	5.07	0.0
CalDataBlock	15809.850	6820.047	450506252	150171624	3.00	15.7
TriggerBlock	4930.960	155.865	137760523	32698000	4.21	3.4
TrigSimBlock	248.000	0.000	6974241	149572	46.63	0.0
L3SummaryBlock	1225.790	497.406	35018906	17312213	2.02	1.8
FwdDetDataBlock	998.000	0.000	27813280	7617416	3.65	0.8
DcasDataBlock	3474.000	0.000	96727584	11212868	8.63	1.2
VertexBlock	367.400	96.020	10348656	2620127	3.95	0.3
PROD@PrimeVtxFinder	426.200	179.102	12127349	3203433	3.79	0.3
ZVertexBlock	438.450	187.533	12379461	1023577	12.09	0.1
TrackBlock	14995.760	8271.102	427312894	275530850	1.55	28.9
PROD@Phoenix_Tracking	331.880	331.652	11321231	2840789	3.99	0.3
PROD@PhoenixSI_Tracking	291.400	465.135	9525621	3964881	2.40	0.4
JetBlock	2188.940	1081.113	60845921	23988367	2.54	2.5
PROD@JetCluModule-cone0.7	1849.100	812.427	51636378	20214459	2.55	2.1
PROD@JetCluModule-cone1.0	1694.750	698.547	47711009	18379134	2.60	1.9
ElectronBlock	1047.570	757.461	29670504	11223350	2.64	1.2
Phoenix_Electrons	517.480	316.192	16827634	4303355	3.91	0.5
PhotonBlock	946.710	731.992	26472640	13841497	1.91	1.4
ClusterBlock	5487.390	2287.028	157099043	60398426	2.60	6.3
MuonBlock	1887.610	1051.911	58130298	9779189	5.94	1.0
Pi0Block	993.140	631.474	29558445	14908968	1.98	1.6
TauBlock	456.400	368.994	12426381	3399985	3.65	0.4
ConversionBlock	98.000	0.000	2854991	109445	26.09	0.0
CosmicBlock	404.000	0.000	13125379	2762520	4.75	0.3
MetBlock	124.000	0.000	2938310	1201156	2.45	0.1
SecVtxTagBlock	911.570	305.990	25291133	1980230	12.77	0.2
JetProbBlock	4199.480	2771.976	115447290	32079286	3.60	3.4

Cone 0.4 → JetBlock

For details of each data block, read the Stntuple manual.

Stntuple datasets

- Many sets of Stntuples are already made and available in public.
- Stntuples are stored in multiple static fileservers. Some files are stored in the DH system recently.
- They are usually “cataloged”. The standard catalog is available at http://fcdfwww.fnal.gov/~cdfopr/Stntuple/cafdfc/
- You can have your own private catalog, run rootd on your own machine and share the dataset with anybody else. (rootd uses ~2% of CPU usage per reader on 1.5GHz machine. No real extra load.)

How to use “cataloged” datasets

Make sure that .rootrc contains the following lines:

```
#-----------------------------------------------------------------------------
# plugin handlers (to enable handling of files stored in DCACHE)
#-----------------------------------------------------------------------------
+Plugin.TFile:  ^dcache:  TDCacheFile   DCache  "TDCacheFile(const char*,Option_t*,const char*,Int_t)"
+Plugin.TFile:  ^dcap:    TDCacheFile   DCache  "TDCacheFile(const char*,Option_t*,const char*,Int_t)"
#-----------------------------------------------------------------------------
# STNTUPLE catalog servers
#-----------------------------------------------------------------------------
Stntuple.Catalog     txt://fcdflnx3.fnal.gov/cdf/home/cdfopr/cafdfc
+Stntuple.Catalog    txt://rock02.fnal.gov/home/hatake/cdf2/cafdfc.new
+Stntuple.Catalog    txt://ncdf131.fnal.gov/home/murat/cafdfc
+Stntuple.Catalog    txt://fcdflnx2.fnal.gov/cdf/home/cdfopr/cafdfc
#-----------------------------------------------------------------------------
```

=== in your analysis driver script, xxx.C ===

To use Stntuples stored on the static file server(s):

```
TChain* chain = new TChain("STNTUPLE");
TStnCatalog* catalog = new TStnCatalog();
catalog->InitChain(chain,” stntuple/dev_242:gmbs0d”,150000,160000);  // Run number from 150000 to 160000
TStnAna* x = new TStnAna(chain);
```

To use Stntuples stored in the DH system:

```
TStnCatalog* catalog = new TStnCatalog();
TStnDataset* dataset = new TStnDataset();
catalog->InitDataset(dataset,"stntuple", "dev_241/gqcd1c“ ,"","",150000,160000);  // gqcd1c, 5.1.0 production MB data
TStnAna* x = new TStnAna(dataset);
```

Event Loop and Analysis Modules

- Stntuple event loop is implemented in TStnAna class.
- Analysis codes (modules) to be executed within the event loop should be inserted into the event loop by the user.
- Analysis modules should be implemented in the class TStnModule.

Example (in e.g. your analysis driver script)

```
TStnAna* x = new TStnAna(xxx);                  // xxx can be a root file, TChain, TStnDataset.
TJetAnaModule* m = new TJetAnaModule(); // if your analysis module is TJetAnaModule
                                                 // implemented in the class TStnModule
x.AddModule(m);                                 // TJetAnaModule is inserted into event loop
x.Run(1000,150000,160000);                      // process 1000 events, from run 150000 to 160000
x.SaveHist("yyyy.root");                        // save all histograms using methods of TStnModule
```

Analysis Module Examples

- You can find many analysis module examples in the Stntuple/ana(/ana) directory.
  - TJetAnaModule.cc, TJetAnaModule.hh
  - TCalAnaModule.cc, TCalAnaModule.hh
  - …
- You can find some example analysis driver scripts in Stntuple/ana.
  - cal_ana.C, ces_ana.C etc

Example: (at the top of your test release)

```
] cp Stntuple/ana/cal_ana.C .
] root
root[x] .x cal_ana.C("your_ntuple.root")     //  TBrowser window pops up
                                             // find histogrmas in root/ana/CalAna/Hist
```

Analysis Module (Class)

- The class and its method are *declared* in the header file (e.g. TJetAnaModule.hh)
- They are *defined* in the implementation file (e.g. TJetAnaModule.cc)
- Analysis classes provide the methods:
  - BeginJob() - register data blocks, book histograms…
  - BeginRun() - performs run specific actions, obtain luminosity, beam position etc.
  - Event(int ientry) - called for every event
  - EndRun() - performs run specific actions
  - EndJob() - clean up…

Analysis Module Examples

- Modify the analysis modules for your analysis, store histograms in a root file.

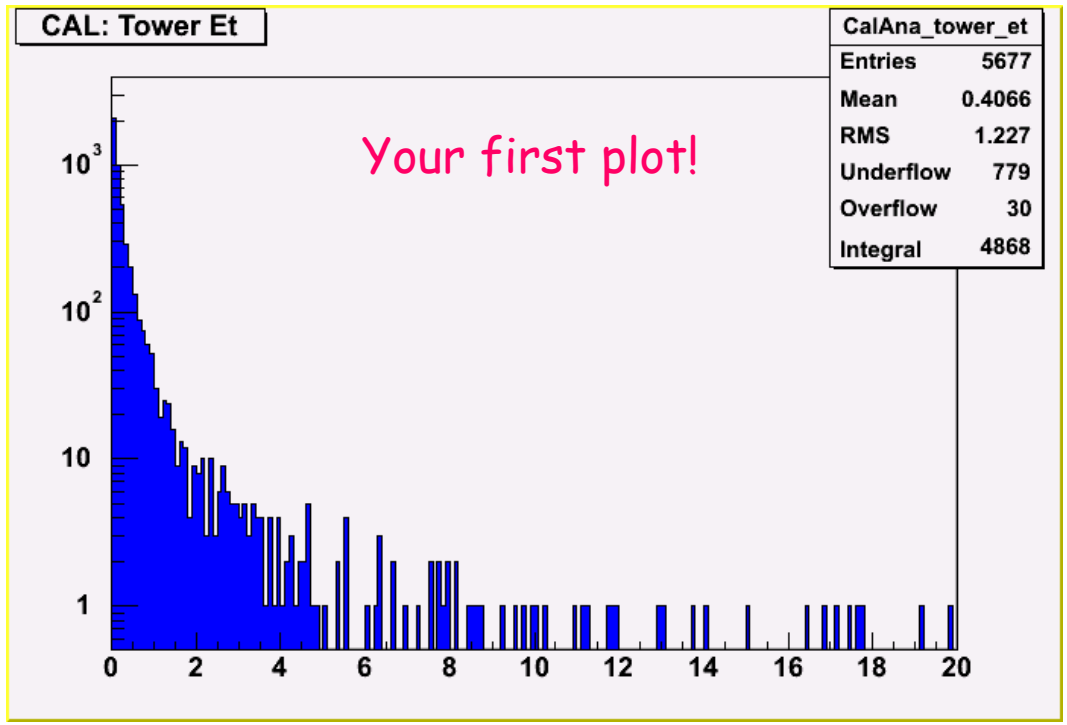
  Example: (at the top of your test release)

```
] cp Stntuple/ana/TCalAnaModule.cc .
] cp Stntuple/ana/ana/TCalAnaModule.hh .
(edit them)
] cp Stntuple/ana/cal_ana.C .
] xemacs cal_ana.C
//-----
void cal_ana(const char* Filename = "results/stnmaker_103566.root", Int_t NEvents = 10)
{
  gSystem->CompileMacro("TCalAnaModule.cc","k");  // compile
  TStnAna* x = new TStnAna(Filename);
  TCalAnaModule* m = new TCalAnaModule();
  x->AddModule(m);                                  // insert TCalAnaModule to the event loop
  x.Run(NEvents);                                   // loop over nevents
  x.SaveHist("output_histogram.root");              // outout root file containing histograms
}
//----
] xemacs rootlogon.C   // comment out gSystem->Load("./shlib/$BFARCH/libStntuple_ana.so");
] root.exe –b –q 'cal_ana.C("your_ntuple.root")'
```

Make your first plot

Example:

```
] root
root[x] TFile fileA=TFile("output_histogram.root");
root[x] fileA.cd("Ana;1");                    // Change the “”
root[x] fileA.ls();                           // Print a list of histograms
root[x] CalAna_tower_et->SetFillColor(4);
root[x] CalAna_tower_et->Draw();
root[x] gPad->SetLogy(1);
```



Use multiple data blocks in a single module

- Every module explicitly requests data blocks it's going to use, and the rest is not read in.
- Example TJetAnaModule.cc and TCalAnaModule.cc read basically only one data block, but of course you can use multiple data blocks in a single analysis module.

```
In BeginJob():
  RegisterDataBlock("HeaderBlock", "TStnHeaderBlock“,&fHeaderBlock);
  RegisterDataBlock("TriggerBlock", "TStnTriggerBlock“, &fTriggerBlock);
  RegisterDataBlock("ZVertexBlock","TStnVertexBlock“,  &fZVertexBlock);
  RegisterDataBlock("TrackBlock",    "TStnTrackBlock“,   &fTrackBlock);
  RegisterDataBlock("CalDataBlock","TCalDataBlock“,    &fCalDataBlock);
  RegisterDataBlock("MetBlock",       "TStnMetBlock“,      &fMetBlock);
RegisterDataBlock("GenpBlock",    "TGenpBlock“,         &fGenpBlock);
  RegisterDataBlock("PhotonBlock", "TStnPhotonBlock“, &fPhotonBlock);
  RegisterDataBlock("SecVtxTagBlock", "TStnSecVtxTagBlock“, &fSecVtxTagBlock);
  RegisterDataBlock("PROD@JetCluModulecone0.7","TStnJetBlock“, &fJetBlock);
```

```
In Event(int ientry):
  fHeaderBlock->GetEntry(ientry);
  fTriggerBlock->GetEntry(ientry);
  fZVertexBlock->GetEntry(ientry);
  fTrackBlock->GetEntry(ientry);
  fCalDataBlock->GetEntry(ientry);
  fMetBlock->GetEntry(ientry);
  fGenpBlock->GetEntry(ientry);
  fPhotonBlock->GetEntry(ientry);
  fJetBlock->GetEntry(ientry);
  fSecVtxTagBlock->GetEntry(ientry);
```

Also, the following lines in e.g. your header file:

```
#include <Stntuple/loop/TStnModule.hh>
#include <Stntuple/obj/TStnHeaderBlock.hh>
#include <Stntuple/obj/TStnTriggerBlock.hh>
#include <Stntuple/obj/TStnVertexBlock.hh>
......
protected:
TStnHeaderBlock*  fHeaderBlock;
TStnTriggerBlock* fTriggerBlock;
TStnVertexBlock*  fZVertexBlock;
……
```

Data Blocks

- As for details of the contents of each DataBlock, look at chapter 18 of the Stntuple manual.
- Shown below is an example of looping over jets in JetBlock

```
int NMaxJets=40;
_njet          = fJetBlock->NJets();              // int, number of jets in the event
_zVertexJet = fJetBlock->ZVertex();               // double, Zvertex used in jet clustering
for (int i=0; i<TMath::Min(NMaxJets,_njet); i++) {
   TStnJet* jet = fJetBlock->Jet(i);
   _P2Jet[i].setX(jet->Momentum()->Px());     // Hep2Vector _P2Jet[40];
   _P2Jet[i].setY(jet->Momentum()->Py());     //
   _PtJet[i] =_P2Jet[i].mag();                // double, jet Pt
   _EtJet[i] =jet->Et();                      // double, jet Et
   _EtaJet[i]=jet->Momentum()->Eta();         // double, jet eta
   _RapJet[i]=jet->Momentum()->Rapidity();    // double, jet rapidity
   _DetJet[i]=jet->DetEta();                  // double, detector eta
   _EmfJet[i]=jet->Emfr();                    // double, EM fraction
   _ChfJet[i]=jet->EOverP();                  // double, E/p
   _PhiJet[i]=jet->Momentum()->Phi();         // double, jet azimuthal angle
   _NtracksJet[i]=jet->NTracks();             // int, number of tracks in the jet
   _NtowersJet[i]=jet->NTowers();             // int, number of towers in the jet
}
```

Trigger Information

- Trigger information is stored in TriggerBlock in Sntuple.
- You can test L3 bits inside the Stntuple analysis module (in every analysis module).
- For more details, see chapter 8 of the Stntuple manual.

Example

In your analysis driver script:

```
TJetAnaModule* m = new TJetAnaModule;       // if your analysis module is TJetAnaModule
m->AddL3TriggerName("JET_20_CENTRAL"); // select only JET_20_CENTRAL
```

In BeginJob():

```
RegisterDataBlock("HeaderBlock","TStnHeaderBlock“, &fHeaderBlock);
RegisterDataBlock("TriggerBlock","TStnTriggerBlock“, &fTriggerBlock );
```

In BeginRun():

```
TStnDBManager*    dbm = TStnDBManager::Instance();
tt    = (TStnTriggerTable*) dbm->GetTable("TriggerTable");
dbm->GetTable("TriggerTable")->Print();  // print the trigger table
int run = GetHeaderBlock()->RunNumber();
TStntuple::Init(run);
TStntuple::InitListOfL3Triggers(this);
```

In Event():

```
fTriggerBlock->GetEntry(ientry);
TStntuple::PrintListOfPassedTriggers(fTriggerBlock,0);   // print the passed triggers
int passed = TStntuple::CheckL3TriggerPath(fTriggerBlock,GetListOfL3Triggers());
if (! passed) .... // leave the module
```

In your header file:

```
#include <Stntuple/alg/TStntuple.hh>
#include <Stntuple/obj/TStnHeaderBlock.hh>
#include <Stntuple/obj/TStnTriggerBlock.hh>
#include <Stntuple/obj/TStnDBManager.hh>

protected:
TStnTriggerBlock* fTriggerBlock;
TStnHeaderBlock*  fHeaderBlock;
TStnTriggerTable*  tt;
```

DB Constants

- Run-dependent constants (such as luminosities, beam positions etc) are stored in Stntuple files once per run.
- They are accessible via Stntuple database manager.
- See chapter 7 of the Stntuple manual for details.

Also, the following lines in e.g your header file:

```
#include <Stntuple/obj/TStnDBManager.hh>
#include <Stntuple/obj/TStnTriggerBlock.hh>
#include <Stntuple/obj/TStnRunSummary.hh>
#include <Stntuple/data/TStnBeamPos.hh>

protected:
TStnTriggerBlock* fTriggerBlock;
TStnTriggerTable*  tt;
TStnRunSummary* runSummary;
TStnBeamPos* bpCOT;
TStnBeamPos* bpSVX;
```

Example: (in BeginRun of your analysis module)

```
TStnDBManager*    dbm = TStnDBManager::Instance();

  tt    = (TStnTriggerTable*) dbm->GetTable("TriggerTable");
  std::cout << " Getting trigger Table " << std::endl;

  runSummary = (TStnRunSummary*)                    // Luminosity information
    TStnDBManager::Instance()->GetTable("RunSummary");
  lum[0]= runSummary->LumiTev();
  lum[1]= runSummary->LumiTape();
  lum[2]= runSummary->OfflineLumiRS();

  TStnBeamPos* bp = (TStnBeamPos*) dbm->GetTable("SvxBeamPos");       // SvxBeamPosition
  svxBeamPos[0]=bp->X0(); svxBeamPos[1]=bp->DxDz();
  svxBeamPos[2]=bp->Y0(); svxBeamPos[3]=bp->DyDz();

  TStnBeamPos* cotbp = (TStnBeamPos*) dbm->GetTable("CotBeamPos");   // CotBeamPosition
  cotBeamPos[0]=cotbp->X0(); cotBeamPos[1]=cotbp->DxDz();
  cotBeamPos[2]=cotbp->Y0(); cotBeamPos[3]=cotbp->DyDz();
```

Good Run List

- You have to check if the run which you are analyzing is usable for your analysis. (Is the run in your “good run list”?)
- Good Run List
  - Electroweak group good run list
    http://www-cdf.fnal.gov/internal/physics/ewk/tools_and_datasets/good_run_list.html
    See chapter 15 of the Stntuple manual about how to use this list with Stntuple.
  - DQM (Data Quality Monitoring) group good run lists (for top, EWK, exotics and QCD physics analyses)
    http://www-cdf.fnal.gov/internal/dqm/goodrun/good.html
    See e.g.
    http://physics.rockefeller.edu/hatake/phys/stnqcd/index.html#goodrun
    about how to use DQM good run lists with Stntuple.

Debugging - GDB

GDB, the GNU project debugger, allows you to see what is going on `inside‘ another program while it executes.
Useful links:
http://www.gnu.org/software/gdb/gdb.html
http://www.delorie.com/gnu/docs/gdb/gdb_toc.html

How to use GDB under emacs: take a look at the Stntuple manual

How to use GDB under unix shell:
] gdb
(gdb) file root.exe
(gdb) run (to start up root.exe)
when the process crashes
(gdb) bt
when you have a “core” file
] gdb root.exe [core file]

for further details, read GDB manual!

Support Model

http://fcdfwww.fnal.gov/~cdfopr/Stntuple/disclaimer.html
http://fcdfwww.fnal.gov/~cdfopr/Stntuple/howto/report_a_bug.html

- It is nobody's MOU.
- It is developed/supported by physicists for themselves.
- If the data of your interest is missing, please volunteer.

Please enjoy using Stntuple!