Table of Contents

NTUPtoNTUPOld

Introduction (17.2.3.1)

This page describes how to skim/slim NTUP (D3PD) in athena with a view to run jobs in the central production system of ATLAS.

See also this tutorial :SoftwareTutorialAnalyzingD3PDsInAthena for the details of reading/writing D3PDs in athena.

<!--
* Set inputNTUP_SMWZFile = root://eosatlas//eos/atlas/atlaslocalgroupdisk/scratch/mkaneda/data/valid1.106046.PythiaZee_no_filter.merge.NTUP_S
* Set inputNTUP_SMWZFile_2 = root://eosatlas//eos/atlas/atlaslocalgroupdisk/scratch/mkaneda/data/valid1.105200.T1_McAtNlo_Jimmy.merge.NTUP_

-->

Athena setup

Please follow this instruction.

In this page, we use release AtlasPhysics-17.2.3.7.1. Make test area like:

```
mkdir -p $HOME/testarea/17.2.3.7.1
cd $HOME/testarea/17.2.3.7.1
export AtlasSetup=/afs/cern.ch/atlas/software/dist/AtlasSetup
alias asetup='source $AtlasSetup/scripts/asetup.sh'
asetup AtlasPhysics,17.2.3.7.1, here
```

<!--
In addition, check out PyUtils-00-12-06 for scripts to make D3PDObjects below.
ï¿½pre1ï¿½
-->

Checking out core package and example package

Check out and make pacakges: NTUPtoNTUPCore and NTUPtoNTUPExample

```
cd $TestArea
cmt co -r NTUPtoNTUPCore-00-00-01 PhysicsAnalysis/NTUPtoNTUP/NTUPtoNTUPCore
cmt co -r NTUPtoNTUPExample-00-00-03 PhysicsAnalysis/NTUPtoNTUP/NTUPtoNTUPExample
setupWorkArea.py
cd WorkArea/cmt/
cmt broadcast
cmt config
cmt broadcast source setup.sh
cmt broadcast make
```

There are two different methods of skim/slim.

- **NTUPtoNTUP method**
  - It uses the interface of root reading in athena (AthenaRootComps) .
  - Main script is: NTUPtoNTUPCore/scripts/NTUPtoNTUP_trf.py
  - Outputs are defined in: NTUPtoNTUPCore/python/NTUPtoNTUPProdFlags.py
  - It can use may athena infrastructures for monitoring or debuging, etc...
  - Known problem
    - CutFlowTree can't be stored
    - It can't run on recent data...
- **SkimNTUP method**
  - It uses non-athena event loop.
  - Main script is: NTUPtoNTUPCore/scripts/SkimNTUP_trf.py
  - Outputs are defined in: NTUPtoNTUPCore/python/SkimNTUPProdFlags.py

Running example

Running example 1: NTUPtoNTUP method, using jobOptions file

```
mkdir -p $TestArea/run/test1
cd $TestArea/run/test1
get_files NTUPtoMyNTUP.py
athena NTUPtoMyNTUP.py >log 2>&1
```

Check: myNtup.root is new file

"NTUPtoNTUPExample/MyNTUP_prodJobOFragment.py" is the key file to setup output file.

Running example 2: NTUPtoNTUP method, using job transform script

```
mkdir -p $TestArea/run/test2
cd $TestArea/run/test2
NTUPtoNTUP_trf.py maxEvents=-1 inputNTUP_SMWZFile=root://castoratlas//castor/cern.ch/grid/atlas/a
```

Check: There should be two files: myNtup.root and myNtup2.root

"NTUPtoNTUPExample/MyNTUP_prodJobOFragment.py" adn
"NTUPtoNTUPExample/MyNTUP2_prodJobOFragment.py" are the key files to setup output files.

Running example 3: method, using job transform script

```
mkdir -p $TestArea/run/testSkim1
cd $TestArea/run/testSkim1
SkimNTUP_trf.py maxEvents=-1 inputNTUP_SMWZFile=root://castoratlas//castor/cern.ch/grid/atlas/atl
```

The script file for skimming/slimming is PhysicsAnalysis/NTUPtoNTUP/NTUPtoNTUPExample/python/skim.py, which has PyROOT function, doSkim().

!JobOFragment file is PhysicsAnalysis/NTUPtoNTUP/NTUPtoNTUPExample/python/MySkimNTUP_prodJobOFragment.py, which calls doSkim().

To add new type, add new definition to PhysicsAnalysis/NTUPtoNTUP/NTUPtoNTUPExample/python/SkimNTUP_ProdFlags.py as described above for NTUPtoNTUP.

Tutorial for NTUPtoNTUP method

Adding new output definition

Add your output definition at the bottom of NTUPtoNTUPExample/python/NTUPtoNTUPProdFlags.py:

```
<br />class WriteMyTestNTUP (JobProperty):<br />"""test NTUP""" <br />    statusOn = True<br />
```

- MyTestNTUP/MyTestNTUP_prodJobOFragment.py: package name and job fragment files which will be made in the next section
- TreeNames: tree name of ntuple
- StreamName: Stream name. It is also used to define the argument name for NTUPtoNTUP_trf.py. argument name is "outputNTUP_MYTESTNTUPFile" in this case.

Preparing new package and job fragment file

Make new package to construct your new ntuple:

```
cd $TestArea
acmd.py cmt new-pkg Tutorial/MyTestNTUP
```

Make jobFragment file: Tutorial/MyTestNTUP/share/MyNTUPTest_prodJobFragment.py

```
cd Tutorial/MyTestNTUP/share/
```

- MyNTUPTest_prodJobFragment.py (simple example to copy a few branches from original ntuple)

```
<br />## This jobO should not be included more than once:
include.block( "MyNTUPTest/MyNTUPTest_prodJobFragment.py" )
## Common import(s):
from AthenaCommon.AlgSequence import AlgSequence
topSequence = AlgSequence()
from AthenaCommon.JobProperties import jobproperties
prodFlags = jobproperties.NTUPtoNTUPProdFlags
from PrimaryDPDMaker.PrimaryDPDHelpers import buildFileName

mytestntup=prodFlags.WriteMyNTUPTest

## Set up a logger:
from AthenaCommon.Logging import logging
MyNTUPTestStream_msg = logging.getLogger( 'MyNTUPTest_prodJobOFragment' )

## Construct the stream and file names for the SUSY NTUP:
streamName = mytestntup.StreamName
fileName   = buildFileName( mytestntup )
MyNTUPTestStream_msg.info( "Configuring MyNTUPTest with streamName '%s' and fileName '%s'" % \
                           ( streamName, fileName ) )

## set input tree:
from AthenaCommon.JobProperties import jobproperties
```

```
ntupFlags=jobproperties.NTUPtoNTUPProdFlags
tree_name=ntupFlags.TreeName()

## Create the NTUP streams:
from NTUPtoNTUPExample.MultipleNTUPStreamManager import MNSMgr
MyNTUPTestStream = MNSMgr.NewNTUPStream( streamName, fileName, tree_name)
MyNTUPTestStream.AddItem([
    "el_n", # add el_n
    "el_pt", # add el_pt
    "el_Etcone*", # add any branches matching el_Etcone*
    "-el_Etcone15", # remove el_Etcone15
    ])
```

The branches in new ntuple is defined by the line "MyNTUPTestStream.AddItem([...])".

Compile the package:

```
cd ../cmt/
sed -i "s/apply_pattern component_library/#apply_pattern component_library/g" requirements # comm
make
```

And test new output:

```
mkdir -p $TestArea/run/test3
cd $TestArea/run/test3
NTUPtoNTUP_trf.py maxEvents=-1 inputNTUP_SMWZFile=root://castoratlas//castor/cern.ch/grid/atlas/a
```

Check output file:

```
 acmd.py dump-root filtered.myTestNtup.root -t physics
```

Skimming/Adding new branches

This section shows example of Z->ee selection.

- Select only electrons pt > 20GeV
- Select events with more than one electrons and 60GeV<m_Z(ee)<120GeV

Making new algorithm to skim events and add new branches

```
cd $TestArea/Tutorial/MyTestNTUP/src
#Make skeleton files (MyTestNTUPAlg.h/cxx)
acmd.py gen-klass  --klass MyTestZeeAlg --pkg MyTestNTUP --type alg -o MyTestZeeAlg
#make load file
mkdir -p components
cat >| components/MyTestNTUP_load.cxx << EOF
#include "GaudiKernel/LoadFactoryEntries.h"
LOAD_FACTORY_ENTRIES( MyTestNTUP )
EOF
#make entries file
cat >| components/MyTestNTUP_entries.cxx << EOF
#include "GaudiKernel/DeclareFactoryEntries.h"
#include "../MyTestZeeAlg.h"
DECLARE_ALGORITHM_FACTORY (MyTestZeeAlg)
DECLARE_FACTORY_ENTRIES( MyTestNTUP ) {
  DECLARE_ALGORITHM(MyTestZeeAlg)
}
```

```
EOF
```

Preparing D3PDObject

Prepare D3PDObject code from D3PD:

```
cd $TestArea
mkdir -p run/d3pd
cd run/d3pd
atl-gen-athena-d3pd-reader root://castoratlas//castor/cern.ch/grid/atlas/atlt3/scratch/mkaneda/da
```

There will be codes: code/*D3PDObject.cxx/h.

Copy ElectronD3PDObject to MyTestNTUP package

```
cd $TestArea/Tutorial/MyTestNTUP/src
cp $TestArea/run/d3pd/code/ElectronD3PDObject* .
```

Writing Z->ee algorithm

```
cd $TestArea/Tutorial/MyTestNTUP/src
```

Edit MyTestZeeAlg.* like: MyTestZeeAlg.cxx, MyTestZeeAlg.h

- RVar/WVar is used to read/write single variable from/to StoreGate
- el_* variables are retrieved as ElectronD3PDObject: const ElectronD3PDObject el("el_"), and can be used like: el.pt(i)
- New electrons are stored as "myEl_", by using ElectronD3PDObject
- In the function execute(), setFilterPassed function is used to decide whether the event should be passed or not.

Go to share directory and edit MyTestNTUP _prodJobOFragment.py.

```
cd $TestArea/Tutorial/MyTestNTUP/share
```

Add "myEl_*" and "my_mZ" to AddItem list and add filter algorithm:

```
<br />...<br />MyNTUPTestStream.AddItem([<br />    "el_n", # add el_n
    "el_pt", # add el_pt
    "el_Etcone*", # add any branches matching el_Etcone*
    "-el_Etcone15", # remove el_Etcone15
    "myEl_*", # new electron
    "my_mZ", # Z mass
    ])
## Algorithm for filter and new variables
from MyTestNTUP.MyTestNTUPConf import MyTestZeeAlg
ZeeAlg=MyTestZeeAlg("MyTestZeeAlg",
                            MinNumberPassed      = 2,
                            PtMin                = 20.*GeV,
                            MZMin                = 60.*GeV,
                            MZMax                = 160.*GeV,
                            )
topSequence+=ZeeAlg
MyTestNTUPStream.AddRequireAlgs(ZeeAlg.getName())
```

Go to cmt directory, and udpate requirements file: requirements, and make.

```
cd $TestArea/Tutorial/MyTestNTUP/cmt
make
```

And test new output:

```
mkdir -p $TestArea/run/test4
cd $TestArea/run/test4
NTUPtoNTUP_trf.py maxEvents=-1 inputNTUP_SMWZFile=root://castoratlas//castor/cern.ch/grid/atlas/a
```

In the log file, you can find a result of the filter

```
MyTestZeeAlg         INFO Finalizing MyTestZeeAlg...
MyTestZeeAlg         INFO ******Summary******
MyTestZeeAlg         INFO Number of processed events:          10000
MyTestZeeAlg         INFO Number of    passed events:          7287
MyTestZeeAlg         INFO Average number of electrons (all) :  4.5943
MyTestZeeAlg         INFO Average number of electrons (rem):  2.04556
MyTestZeeAlg         INFO ******************
```

And check the contents if it has correctly values "myEl_" and "my_mZ"

Introduction(17.2.1)

This page describes how to skim/slim NTUP (D3PD) in athena with a view to run jobs in the central production system of ATLAS.

See also this tutorial :SoftwareTutorialAnalyzingD3PDsInAthena for the details of reading/writing D3PDs in athena.

Athena setup

Please follow this instruction.

In this page, we use release 17.2.1. Make test area like:

```
mkdir -p $HOME/testarea/17.2.1
cd $HOME/testarea/17.2.1
export AtlasSetup=/afs/cern.ch/atlas/software/dist/AtlasSetup
alias asetup='source $AtlasSetup/scripts/asetup.sh'
asetup 17.2.1, here
```

In addition, check out PyUtils-00-12-06 for scripts to make D3PDObjects below.

```
cmt co -r PyUtils-00-12-06 Tools/PyUtils
cd Tools/PyUtils/cmt
make
```

Checking out main package and running examples

Checking out main pacakge: NTUPtoNTUPExample

```
cd $TestArea
cmt co -r NTUPtoNTUPExample-00-00-03 -o svn+ssh://svn.cern.ch/reps/atlasusr/mkaneda/atlasoff/ Phy
PhysicsAnalysis/NTUPtoNTUP/NTUPtoNTUPExample/cmt/
make
```

Running example 1: using jobOptions file

```
mkdir -p $TestArea/run/test1
cd $TestArea/run/test1
get_files NTUPtoMyNTUP.py
athena NTUPtoMyNTUP.py >log 2>&1
```

Check: myNtup.root is new file

Running example 2: using job transform script

```
mkdir -p $TestArea/run/test2
cd $TestArea/run/test2
NTUPtoNTUP_trf.py maxEvents=-1 inputNTUP_SMWZFile=root://castoratlas//castor/cern.ch/grid/atlas/a
```

Check: There should be two files: myNtup.root and myNtup2.root

In both cases, "NTUPtoNTUPExample/MyNTUP_prodJobOFragment.py" is the key file to setup output file (myNtup.root).

Adding new output definition

Add your output definition at the bottom of NTUPtoNTUPExample/python/NTUPtoNTUPProdFlags.py:

```
class WriteMyTestNTUP (JobProperty):
"""test NTUP"""
    statusOn = True
    allowedTypes = ['bool']
    StoredValue = False
    StreamName = 'StreamNTUP_MYTESTNTUP'
    FileName = '' NTUPScript = "MyTestNTUP/MyTestNTUP_prodJobOFragment.py"
    TreeNames = ['physics']
    SubSteps = ['n2n']
prodFlags.add_JobProperty (WriteMyTestNTUP)
listAllKnownNTUPtoNTUP.append (prodFlags.WriteMyTestNTUP)
```

- MyTestNTUP/MyTestNTUP_prodJobOFragment.py: package name and job fragment files which will be made in the next section
- TreeNames: tree name of ntuple
- StreamName: Stream name. It is also used to define the argument name for NTUPtoNTUP_trf.py. argument name is "outputNTUP_MYTESTNTUPFile" in this case.

Preparing new package and job fragment file

Make new package to construct your new ntuple:

```
cd $TestArea
acmd.py cmt new-pkg Tutorial/MyTestNTUP
```

Make jobFragment file: Tutorial/MyTestNTUP/share/MyNTUPTest_prodJobFragment.py

```
cd Tutorial/MyTestNTUP/share/
```

- MyNTUPTest_prodJobFragment.py (simple example to copy a few branches from original ntuple)

```
## This job0 should not be included more than once:
include.block( "MyNTUPTest/MyNTUPTest_prodJobFragment.py" )
## Common import(s):
from AthenaCommon.AlgSequence import AlgSequence
topSequence = AlgSequence()
from AthenaCommon.JobProperties import jobproperties
prodFlags = jobproperties.NTUPtoNTUPProdFlags
from PrimaryDPDMaker.PrimaryDPDHelpers import buildFileName

mytestntup=prodFlags.WriteMyNTUPTest

## Set up a logger:
from AthenaCommon.Logging import logging
MyNTUPTestStream_msg = logging.getLogger( 'MyNTUPTest_prodJobOFragment' )

## Construct the stream and file names for the SUSY NTUP:
streamName = mytestntup.StreamName
fileName   = buildFileName( mytestntup )
MyNTUPTestStream_msg.info( "Configuring MyNTUPTest with streamName '%s' and fileName '%s'" % \
                           ( streamName, fileName ) )

## set input tree:
from AthenaCommon.JobProperties import jobproperties
ntupFlags=jobproperties.NTUPtoNTUPProdFlags
tree_name=ntupFlags.TreeName()

## Create the NTUP streams:
from NTUPtoNTUPExample.MultipleNTUPStreamManager import MNSMgr
MyNTUPTestStream = MNSMgr.NewNTUPStream( streamName, fileName, tree_name)
MyNTUPTestStream.AddItem([
    "el_n", # add el_n
    "el_pt", # add el_pt
    "el_Etcone*", # add any branches matching el_Etcone*
    "-el_Etcone15", # remove el_Etcone15
    ])
```

The branches in new ntuple is defined by the line "MyNTUPTestStream.AddItem([...])".

Compile the package:

```
cd ../cmt/
sed -i "s/apply_pattern component_library/#apply_pattern component_library/g" requirements # comm
make
```

And test new output:

```
mkdir -p $TestArea/run/test3
cd $TestArea/run/test3
NTUPtoNTUP_trf.py maxEvents=-1 inputNTUP_SMWZFile=root://castoratlas//castor/cern.ch/grid/atlas/a
```

Check output file:

```
acmd.py dump-root filtered.myTestNtup.root -t physics
```

Skimming/Adding new branches

This section shows example of Z->ee selection.

- Select only electrons pt > 20GeV
- Select events with more than one electrons and 60GeV<m_Z(ee)<120GeV

Making new algorithm to skim events and add new branches

```
cd $TestArea/Tutorial/MyTestNTUP/src
#Make skeleton files (MyTestNTUPAlg.h/cxx)
acmd.py gen-klass  --klass MyTestZeeAlg --pkg MyTestNTUP --type alg -o MyTestZeeAlg
#make load file
mkdir -p components
cat >| components/MyTestNTUP_load.cxx << EOF
#include "GaudiKernel/LoadFactoryEntries.h"
LOAD_FACTORY_ENTRIES( MyTestNTUP )
EOF
#make entries file
cat >| components/MyTestNTUP_entries.cxx << EOF
#include "GaudiKernel/DeclareFactoryEntries.h"
#include "../MyTestZeeAlg.h"
DECLARE_ALGORITHM_FACTORY (MyTestZeeAlg)
DECLARE_FACTORY_ENTRIES( MyTestNTUP ) {
  DECLARE_ALGORITHM(MyTestZeeAlg)
}
EOF
```

Preparing D3PDObject

Prepare D3PDObject code from D3PD:

```
cd $TestArea
mkdir -p run/d3pd
cd run/d3pd
atl-gen-athena-d3pd-reader root://castoratlas//castor/cern.ch/grid/atlas/atlt3/scratch/mkaneda/da
```

root://castoratlas//castor/cern.ch/grid/atlas/atlt3/scratch/mkaneda/data/valid1.106046.PythiaZee_no_filter.merge.NTUP

There will be codes: code/*D3PDObject.cxx/h.

Copy ElectronD3PDObject to MyTestNTUP package

```
cd $TestArea/Tutorial/MyTestNTUP/src
cp $TestArea/run/d3pd/code/ElectronD3PDObject* .
```

Writing Z->ee algorithm

```
cd $TestArea/Tutorial/MyTestNTUP/src
```

Edit MyTestZeeAlg.* like: MyTestZeeAlg.cxx, MyTestZeeAlg.h

- RVar/WVar is used to read/write single variable from/to StoreGate
- el_* variables are retrieved as ElectronD3PDObject: const ElectronD3PDObject el("el_"), and can be used like: el.pt(i)

- New electrons are stored as "myEl_", by using ElectronD3PDObject
- In the function execute(), setFilterPassed function is used to decide whether the event should be passed or not.

Go to share directory and edit MyTestNTUP _prodJobOFragment.py.

```
cd $TestArea/Tutorial/MyTestNTUP/share
```

Add "myEl_*" and "my_mZ" to AddItem list and add filter algorithm:

```
...
MyNTUPTestStream.AddItem([
    "el_n", # add el_n
    "el_pt", # add el_pt
    "el_Etcone*", # add any branches matching el_Etcone*
    "-el_Etcone15", # remove el_Etcone15
    "myEl_*", # new electron
    "my_mZ", # Z mass
    ])
## Algorithm for filter and new variables
from MyTestNTUP.MyTestNTUPConf import MyTestZeeAlg
ZeeAlg=MyTestZeeAlg("MyTestZeeAlg",
                            MinNumberPassed      = 2,
                            PtMin                = 20.*GeV,
                            MZMin                = 60.*GeV,
                            MZMax                = 160.*GeV,
                            )
topSequence+=ZeeAlg
MyTestNTUPStream.AddRequireAlgs(ZeeAlg.getName())
```

Go to cmt directory, and udpate requirements file: requirements, and make.

```
cd $TestArea/Tutorial/MyTestNTUP/cmt
make
```

And test new output:

```
mkdir -p $TestArea/run/test4
cd $TestArea/run/test4
NTUPtoNTUP_trf.py maxEvents=-1 inputNTUP_SMWZFile=%inputNTUP_SMWZFile%  outputNTUP_MYTTESTNTUPFil
```

In the log file, you can find a result of the filter

```
MyTestZeeAlg         INFO Finalizing MyTestZeeAlg...
MyTestZeeAlg         INFO ******Summary******
MyTestZeeAlg         INFO Number of processed events:          10000
MyTestZeeAlg         INFO Number of    passed events:          7287
MyTestZeeAlg         INFO Average number of electrons (all) :  4.5943
MyTestZeeAlg         INFO Average number of electrons (rem):  2.04556
MyTestZeeAlg         INFO ******************
```

And check the contents if it has correctly values "myEl_" and "my_mZ"

Another method of->D3PD

NTUPtoNTUPExample has another method to skim/slim D3PD by using simple script.

Run example:

```
mkdir -p $TestArea/run/test5
cd $TestArea/run/test5
SkimNTUP_trf.py maxEvents=-1 inputNTUP_SMWZFile=root://castoratlas//castor/cern.ch/grid/atlas/atl
```

The script file for skimming/slimming is PhysicsAnalysis/NTUPtoNTUP/NTUPtoNTUPExample/python/slim.py, which has PyROOT function, doSkim().

JobOFragment file is PhysicsAnalysis/NTUPtoNTUP/NTUPtoNTUPExample/python/MySkimNTUP_prodJobOFragment.py, which calls doSkim().

To add new type, add new definition to PhysicsAnalysis/NTUPtoNTUP/NTUPtoNTUPExample/python/SkimNTUP_ProdFlags.py as described above for NTUPtoNTUP.

Major updates:
-- MichiruKaneda - 05-Apr-2012

%RESPONSIBLE% MichiruKaneda
%REVIEW% **Never reviewed**

This topic: Main > NTUPtoNTUPOld
Topic revision: r2 - 2012-08-13 - MichiruKaneda