Table of Contents

TauTriggerValidationTutorial......1

The Basics......2

Setting up your account......2
Setting up CMT......2
Setting up DQ2......3
Express Setup......3

TTP packages Installation......5

Downloading Packages......5
Compiling the packages......5

Searching Datasets......6

AMI......6
Downloading the dataset with DQ2......6

Creating an Ntuple......7

Editing the Ntuple......7
Compiling the Ntuple......7

Making Tables and Plots......8

TauTriggerValidationTutorial

The Basics

Setting up your account

It is assumed that you have an account on LXPLUS already, and have about 100MB of spare quota. You can check this by typing;

```
fs lq
```

the result is given in kilobytes.

The first thing we have to do is entry to lxplus, in this case we will specify at the end of the command control keys to access to graphical modes (for root framework and emacs). In this tutorial we need computers with SL4, we don't have to specify a SL4 machine because lxplus connect to this kind of O.S by default.

```
ssh yourlogin@lxplus.cern.ch -X -Y
```

Setting up CMT

This is intended for people who doesn't have a cmthome and want to know the whole process of setting up his account. If you already have done this before and want a quicker way to do this, please go to the express point.

1. Create a cmthome directory in your home directory:

```
cd $HOME
mkdir cmthome
cd cmthome
```

2. Make your login requirements file - use your favorite editor to create a file called requirements and paste into it these lines:

```
set   CMTSITE  CERN
set   SITEROOT /afs/cern.ch
macro ATLAS_DIST_AREA ${SITEROOT}/atlas/software/dist
macro ATLAS_TEST_AREA ${HOME}/scratch0/
use AtlasLogin AtlasLogin-* $(ATLAS_DIST_AREA)
```

Important notes:

* Line 4: change the path after ${HOME} so that it points to wherever you want to work. If you have scratch space it is recommended that you work in there. Of course if you want to make a new directory inside this space, that is fine.

3. Save and exit this file

4. Now set up CMT as follows:

```
source /afs/cern.ch/sw/contrib/CMT/v1r20p20090520/mgr/setup.sh
cmt config
```

The first line sets the CMT version. This changes from time to time. When it changes you need to re-do these two lines. Check from time to time that your CMT is compatible with the software release you're using, with the ATLAS release status page. It is particularly important that you use this version now, due to the change to SVN

Note: Steps (1)-(4) above only have to be done once. Step (5) has to be done every time you log into a new session.

5. Now set up the ATLAS software. For this part we'll use release 15.6.9.

```
cd $HOME
source cmthome/setup.sh -tag=15.6.9,setup,32
```

(Note that you'll get a complaint about directories not existing - don't worry about this - it will only happen once)

```
echo $TestArea
mkdir $TestArea
```

6. Go into your $TestArea, and get ready to run the ATLAS software!

```
cd $TestArea
```

Setting up DQ2

1. Set up the LXPLUS UI:

```
source /afs/cern.ch/project/gd/LCG-share/current/etc/profile.d/grid_env.sh
```

(please note that later you won't need to do this step anymore, once you get introduced to the DQ2 framework)

2. Make your proxy certificate (you'll be asked for a password):

```
voms-proxy-init --voms atlas
```

3. To load the environment on lxplus do:

```
source /afs/cern.ch/atlas/offline/external/GRID/ddm/DQ2Clients/setup.sh
```

or

```
source /afs/cern.ch/atlas/offline/external/GRID/ddm/DQ2Clients/setup.zsh
```

depending on your shell.

Express Setup

A quicker way to do this is making two requirements files for setting up athena-cmt and DQ2.

1. Create a directory called myscripts in your home directory:

```
cd $HOME
mkdir myscripts
cd myscripts
```

2. Make your athena-cmt requirements file - use your favorite editor to create a file called g15.6.9.sh and paste into it these lines:

```
export ATHENA_VERSION=15.6.9
cd ~/cmthome
```

```
cp rg${ATHENA_VERSION} requirements
source /afs/cern.ch/sw/contrib/CMT/v1r20p20090520/mgr/setup.sh
cmt config
source ~/cmthome/setup.sh -tag=${ATHENA_VERSION},setup,32
echo $CMTCONFIG
echo $CMTPATH
cmt show path
cd $HOME/myscripts
```

3. Make your DQ2 requirements file - use your favorite editor to create a file called setdq2.sh and paste into it these lines:

```
source /afs/cern.ch/project/gd/LCG-share/current/etc/profile.d/grid_env.sh
source /afs/cern.ch/atlas/offline/external/GRID/ddm/DQ2Clients/setup.sh
voms-proxy-init --voms atlas
```

4. Run both files:

```
source g15.6.9.sh
source setdq2.sh
```

Note: We only have to run the dq2 file when we want to download something from the grid.

TTP packages Installation

Downloading Packages

First we have to look for the packages tags compatibles with our athena release (15.6.9) in this twiki page (on the latest tags section): https://twiki.cern.ch/twiki/bin/view/Atlas/TrigTauPerformNtuple.

1. Go into your $TestArea.

```
cd $TestArea
```

2. For Validation we will download three packages: TrigTauPerformNtuple, TrigTauPerformAnalysis, TrigTauPerformValidation.

```
cmt co Trigger/TrigAnalysis/TrigTauAnalysis/TrigTauPerformNtuple
cmt co Trigger/TrigAnalysis/TrigTauAnalysis/TrigTauPerformAnalysis
cmt co Trigger/TrigAnalysis/TrigTauAnalysis/TrigTauPerformValidation
cmt co Trigger/TrigAnalysis/TrigBunchCrossingTool
cmt co Trigger/TrigAnalysis/TrigTauAnalysis/TrigTauPerformAthena
cmt co PhysicsAnalysis/TauID/TauDiscriminant
```

* -XX-YY-ZZ: packages tags.

Compiling the packages

1. Compile each package in this order: Ntuple->Analysis->Validation.

```
cd Trigger/TrigAnalysis/TrigTauAnalysis/TrigTauPerformNtuple/cmt
gmake -f Makefile.standalone
```

Note: You have to do the same with the other two packages.

2. Create a run directory in the Trigger area (we will use this folder later for the python code).

```
cd Trigger/TrigAnalysis/TrigTauAnalysis/
mkdir run
```

Searching Datasets

Datasets are produced with diferent kind of variables and athena releases. When the datasets are done they will have a specific tag, in this tutorial we will use a dataset for athena release 15.6.9.

AMI

AMI (ATLAS Metadata Interface) is a generic cataloging framework used in ATLAS for dataset discovery (also a Tag Collector + one or two other things).

1. The first thing we have to do is entry the AMI web page (you need a grid certificate correctly loaded into your browser):
http://ami.in2p3.fr:8080/AMI/servlet/net.hep.atlas.Database.Bookkeeping.AMI.Servlet.Command?linkId=512

2. In the first rectangle we can write any option we want to look for, in this case we will search a medium pT signal coming from a Z boson (write: ztautau).

3. On the top of the page (just under Home) you will find a "clickeable" rectangle with the kinds of dataset we can find for search, in this case we will use the mc09-production.

4. On the third column (dataType) click on the Grupby button (left icon) and select Browse only AOD's.

5. We will use a sample with ~10.000 events and with a transformation package higher that 15.6.9.0, so let's find it.

6. When we already find a good dataset, then you have to click on the details hyperlink on the left.

7. In this web page you can see the details of the dataset we are looking for. We can check the athena version, the number of AOD's in it (40 is a good statistic) and something we have to write for a future step: the Geometry Version (in our case ATLAS-GEO-10-00-00).

8. Click on the DQ2 hyperlink, just down the dataset name.

9. On the right we have three rows. The first one, Existing Replicas, tell us in which sites our dataset is (there have to be at least one for download it). In the third row, Contained datasets, there is a hyperlink to the dataset location, click it.

10. Now we have almost the same page, but the difference is that now it appears a new button: dq2-get, click it. This will show you the command you have to write on lxplus to download the dataset from the grid, copy it.

Downloading the dataset with DQ2

1. Go to your temporally folder.

```
cd /tmp/username
```

2. Paste the dq2-get command you just copy from AMI and wait untill the download is done.

```
dq2-get valid1.105188.A3_Ztautau_filter.recon.AOD.e380_s764_r1266_tid131286_00
```

Creating an Ntuple

We can make an Ntuple from the AOD's that are inside de dataset, we can also do the same with RDO's files.

Editing the Ntuple

1. Create a directory in the temporary area where we will put the ntuples (the ntuples can be quite heavier ~150M).

```
cd /tmp/yourusername
mkdir ntuples
```

2. Download the python file that create Ntuples from the dataset.

```
cd ntuples
get_files TTPNtuple.py
```

3. Edit the file - use your favorite editor to edit the following lines:

* Uncomment the first line and replace the DetDescrVersion with the Geometry Version we wrote before (ATLAS-GEO-10-00-00). This is the detector geometry used by the dataset creation.

* In line 66, change EvtMax from 1000 to -1, this will runs over all events that are available.

* In line 70, change: include("TrigTauPerformAthena/tauPerform_RDO_topOptions.py"), to: include("TrigTauPerformAthena/tauPerform_AOD_topOptions.py").

* Copy this two lines on line 64. This way is easier to access to a large quantity of files:

```
  from glob import glob
  PoolAODInput= glob('/tmp/yourlogin/data/datasetfolder/AOD*')
```

* In line 75 change the root file name to: trigtau.15691.AOD.root

Compiling the Ntuple

Just run it typing:

```
athena TTPNtuple.py
```

Note: This will last ~30min, so we will leave the first part of the tutorial up here.

I suggest you to save que ntuple in your own PC or Server, because the file weight ~150Mb and most of the lxplus accounts have 300Mb.

Making Tables and Plots

There is a file in the TriggerTauPerformValidation directory that have some examples about the creation of tables or plots from the ntuple we just compile. You can access to this file typing:

```
cd $TestArea/Trigger/TrigAnalysis/TrigTauAnalysis/TrigTauPerformValidation
```

the file name is "howto", it should be look like the following:

```
gSystem->Load("libHist.so");
gSystem->Load("libTree.so");
gSystem->Load("libGpad.so");
gSystem->Load("libCore.so");

gSystem->Load("lib/TrigTauPerformNtuple/libTrigTauPerformNtuple.so");
gSystem->Load("lib/TrigTauPerformAnalysis/libTrigTauPerformAnalysis.so");
gSystem->Load("lib/TrigTauPerformValidation/libTrigTauPerformValidation.so");

TTPValCutStudy study("ntuples/TTP09/tmp.*");
study.analyze();
study.print();

TTPValEffTableMaker maker("ntuples/TTP09/tmp.*");
maker.analyze();
maker.print();


//-------------------------------------------------------------------

//Here are two examples of how to use the validation classes:
(the usage of other tools in this package is almost the same, please have a look at the 'manual')

//========TTPValCutStudy======
//This class prints out relative and absolute efficiency curves after each cut. This is plotted a

TTPValCutStudy study;  //create CutStudy object
study.setFile("ntuples/TTP09/tmp.*"); //define which file(s) is/are used
//also possible:    TTPValCutStudy study("ntuplest/TTP09/tmp.*");

//The following settings are all optional
study.setType(0);                       //Type referes to the type of the efficiencies; 0 - absol
study.setNprong(3);                     //Only look at 3-prong decays; default is 0 - no selectio
study.setMenu(TrigTauFlags::TAU35I);    //defines which menu is used; default is TAU20I
study.setCutLevel(TrigTauSelectionManager::TIGHT); //cutlevel is set to TIGHT; default is LOOSE
study.setOfflineType(TrigTauMCTauList::BOTHTAU); //offline matching set to BOTHTAU; default is EI
study.setChainType(TrigTauFlags::SITRK); //default is IDTRK; for other values see TrigTauFlags.h
study.setMaxVisEta(1.);                 //default is 2.5
study.setMinLeadingPt(9.0);             //default is 6.0

//Now, start analyzing and printing the result
study.analyze();
study.print();                          //creates file ntuple.eps with all efficiency histograms
study.save("ntuple.root");              //saves histograms in ntuple.root
study.compare("ntuple.root");           //plots analyzed histograms with the histograms from ntup


======TTPValEffTableMaker======
//This class prints out relative and absolute efficiencies for each trigger level for different o

TTPValEffTableMaker maker;
maker.setFile("ntuples/TTP09/tmp.*");

//The following settings are all optional
maker.setMenu(TrigTauFlags::TAU35I);                     //defines which menu is used; default is
maker.setCutLevel(TrigTauSelectionManager::TIGHT);      //cutlevel is set to TIGHT; default is LO
maker.setMaxVisEta(2.3);                                //default is 2.5
maker.setMinLeadingPt(5.0);                             //default is 6.0
maker.setChainType(TrigTauFlags::SITRK);                //default is IDTRK; for other values see

//Now, start analyzing and printing the result
maker.analyze();
maker.print(0);                                         //printing absolute efficiencies; without
maker.print(1);                                         //printing relative efficiencies


=====TTPValBackgroundStudy=====
//Prints efficiency with respect of pt, eta, phi. Prints also rates for given luminosity and x-se

TTPValBackgroundRate brate("../data/ntuples/TTP14/5010/tmp*");
brate.analyze();
brate.print(1E31, 1.38E-27);    //prints rates for lum=10^31 and x-section 1.38x10^-27 (which is
```

You have to call also the following libraries just after the first four lines:

```
gSystem->Load("libPhysics");
gSystem->Load("libPostscript.so");
```

Lines 6 to 8 call the libraries we have recently download and compile, so you have to put the path to they.

You should also add to lines 10 and 14 the path to the ntuple we have recently made.

The following code have more examples of how we can call differents functions, for the moment we will only use the first two ones. You can also see other functions in the file "manual" that is in the same directory, for example we will use also (copy it before line 17 changing the path to the ntuple):

```
TTPValMultiPlots mplots("ntuples/TTP09/tmp.*");
mplots.analyze();
mplots.print();
```

Now you have to comment all the lines after the function we just add and save the file.

Go to the run directory and copy the "howto" file we modify before.

```
cd $TestArea/Trigger/TrigAnalysis/TrigTauAnalysis/run
cp /scratch0/AtlasOffline-15.6.9/Trigger/TrigAnalysis/TrigTauAnalysis/TrigTauPerformValidation/ho
```

Open this file with your favorite editor and save it as "ptt.C". Then you have to run it in root:

```
root -l
.x ptt.C
```

Major updates:
-- Main.Sebastian Olivares - 27 May 2010

RESPONSIBLE SebastianOlivares
REVIEW **Never reviewed**

This topic: Main > TauTriggerValidationTutorial
Topic revision: r17 - 2011-02-14 - FernandoAndresQuinonezGranados