Table of Contents

Analysing Outer Tracker Threshold Scans

Introduction

The Outer Tracker (OT) threshold layer scan is used to measure gain variations in the OT. Threshold scans are performed during LHC operation, using the LHC beam. Gain variations are monitored by studying hit efficiency as a function of the OT electronics amplifier threshold. This Twiki contains instructions on how to analyse the data from such a threshold scan.

The idea of the procedure is as follows: For every layer of the OT (12 in total, 4 layers per OT station) the amplifier threshold of the OT read-out electronics is changed in 10 steps. The steps are defined as

[800 mV, 1000 mV, 1200 mV, 1250 mV, 1300 mV, 1350 mV, 1400 mV, 1450 mV, 1600 mV, 1800 mV]
Notice that the threshold scans recorded before June 2011, did not contain the last two thresholds.

As the threshold in the layer under consideration is changed, all other layers are operated at the nominal threshold value of 800 mV, to reconstruct charged particle tracks. The hit efficiency is defined as the number of found hits, divided by the total number of predicted hits, for tracks passing within 1.25\,mm from the wire. For a particular threshold and corresponding layer under study, the hit efficiency is measured in 85\,mm wide bins of the horizontal coordinate x and 56 mm high bins of the vertical coordinate y. The bin size in x corresponds to one quarter of the width of an OT module.

Package structure

Thresholdscans

- Scripts: all common functions needed for the analyses
- Compare: comparison of scan to reference scans
- TS_201X_XX: this folder exists for each scan

TS_201X_XX:

- Process: contains the script needed for processing the raw data
- MergedRootFiles: contains the steering scripts for merging the rootfiles created when processing the raw data
- Analysis: contains the steering scripts for the first step of the analysis.

Compare: Each scan is compared with a reference scan. Usually, the scan taken in August 2010 is the reference.

- A folder containing the results is automatically created for each comparison between two scans
- Compare.py: steering script for the comparison
- Plottrend.py: to make a trendplot of all past scan results

Process the raw data

General info

The data taken during a threshold scan is written to disk in the .RAW format. Typically we select 150 000 events per threshold. In total we have 10 thresholds times 12 layers, which comes down to 120*150 000 events ~ 18 M events. More information about the operational procedure of a threshold layers scan can be found here: https://lbtwiki.cern.ch/bin/view/OT/HowToDoThresholdLayerScan.

A C++ Brunel algorithm is responsible for using the OTHitEfficiencyMonitor per ODIN calibration step, where we require at least 20 OT hits for the track to be used in the efficiency calculation. This algorithm together with the adjustment in the OTHitEfficiencyMonitor can be found at /afs/cern.ch/work/l/ldufour/public/ThresholdScan/

Database Tags

We perform a dump of the online database inmediately after taking the Threshold Scan.

```
SetupProject LHCb --use-grid
# or the equivalent lb-dev variant, followed by ./run $SHELL)

# get grid proxy
lhcb-proxy-init

# secret "handshake" to make the Oracle DB available to us
source /sw/oracle/set_oraenv.sh
export CORAL_DBLOOKUP_PATH=/group/online/conddbserver/
export CORAL_AUTH_PATH=/group/online/conddbserver

# make the actual snapshot
CondDBAdmin_MakeSnapshot.py --options \
$SQLDDDBROOT/options/SQLDDDB-Oracle.py -s 2015-01-01 -u 2015-12-31 \
ONLINE sqlite_file:ONLINE-2015.db.tmp/ONLINE && \
mv -f ONLINE-2015.db.tmp ONLINE-2015.db
```

This online snapshot needs to be somewhere in a public afs folder when submitting to the grid, for some reason.

(Outdated: All scans taken during Run 2 use the default recipe in which the OT times correspond to the ones during normal data taking. It is not necessary anymore, from the scan taken on June 4th 2015 onwards, to use the dedicated database.)

The OT readout gate was changed in May 2011. However, when performing a threshold scan, the recipe still uses the old configuration (same readout gate for all OT stations instead of interspaced by 2ns). This means when anaylyzing the scans after May 15 2011, a dedicated database with module t0's is used when running the reconstruction:

```
from Configurables import ( CondDBAccessSvc, CondDB )
AlignmentCondition = CondDBAccessSvc("AlignmentCondition")
AlignmentCondition.ConnectionString = "sqlite_file:2011-05.v2-scan.db/LHCBCOND"
CondDB().addLayer(AlignmentCondition)
```

Veto Hlt Errors

The Hlt Error Filter checks whether there are Hlt2 decisions. Since this is not the case here, and we're not interested in them, the Hlt Error should be vetoed by setting the Brunel property

```
VetoHltErrorEvents = False
```

Local testing

(outdated) For this, the 'job steering' part of the Bender script needs a PFN of one raw file in the threshold scan run (in this case running only 100 events). This raw file should be available locally. **Note: Take a file in the middle of the run.** The first N files have only step 0 in the first 100 events and therefore your histograms will be empty!! :

```
files = [
        "DATAFILE='castor:/castor/cern.ch/grid/lhcb/data/2011/RAW/FULL/LHCb/CALIBRATION11/91638/0
        ]
run(100)
```

For local testing of the script, do

```
SetupProject Bender --use-grid
python -i [Bender script]
```

Interactive backend of Ganga

Once the script runs OK locally, it's a useful test to run it on the Interactive backend of Ganga, to see if the LFN's are read correctly.

There are some important lines in the submission script, since we want to send along our own installation of Bender. Notice that for using the Interactive backend of Ganga, it's important not to forget the --use-grid option (since the Interactive backend simply starts a subshell and sets up the projects as speciefied in the ganga script):

```
b = Bender(version = 'vXrYpZ')
b.setupProjectOptions = '--use-grid'
b.user_release_area = '/afs/cern.ch/user/N/Name/cmtuser'
b.module = {PATHTOBENDERSCRIPTS}
```

For running interactively a pfn is required. There are two possiblities. Either you download a raw file to your local machine. You can add this file to your job by doing (in ganga)

```
j.inputdata = ["pfn:full_path_filename.raw"]
```

Or, you can run interactively on lxplus, using the usual ways of importing the data. This is only possible on cern nodes!

Submit Jobs to the Grid

With everything set up properly, and after assuring yourself the python script works, it is time to submit all the jobs to the Grid using the Dirac backend of Ganga. You should keep in mind that Dirac can only handle 100 input files per job. So split all the RAW files in option files of 100 each, and submit the jobs (with 100 subjobs each) to the Grid. The subjobs typically take about 1.5 days to finish.

```
lhcb-proxy-init
SetupProject Ganga
ganga SubmitScript.py
```

An example of a script to submit the jobs to the Grid is attached to this Twiki.

Merging the ROOT Files

Single Scan Analysis

Comparing Scans

-- DaanVanEijk - 15-Mar-2011

This topic: Sandbox > AnalyseOTThresholdScan
Topic revision: r19 - 2016-01-05 - LaurentDufour