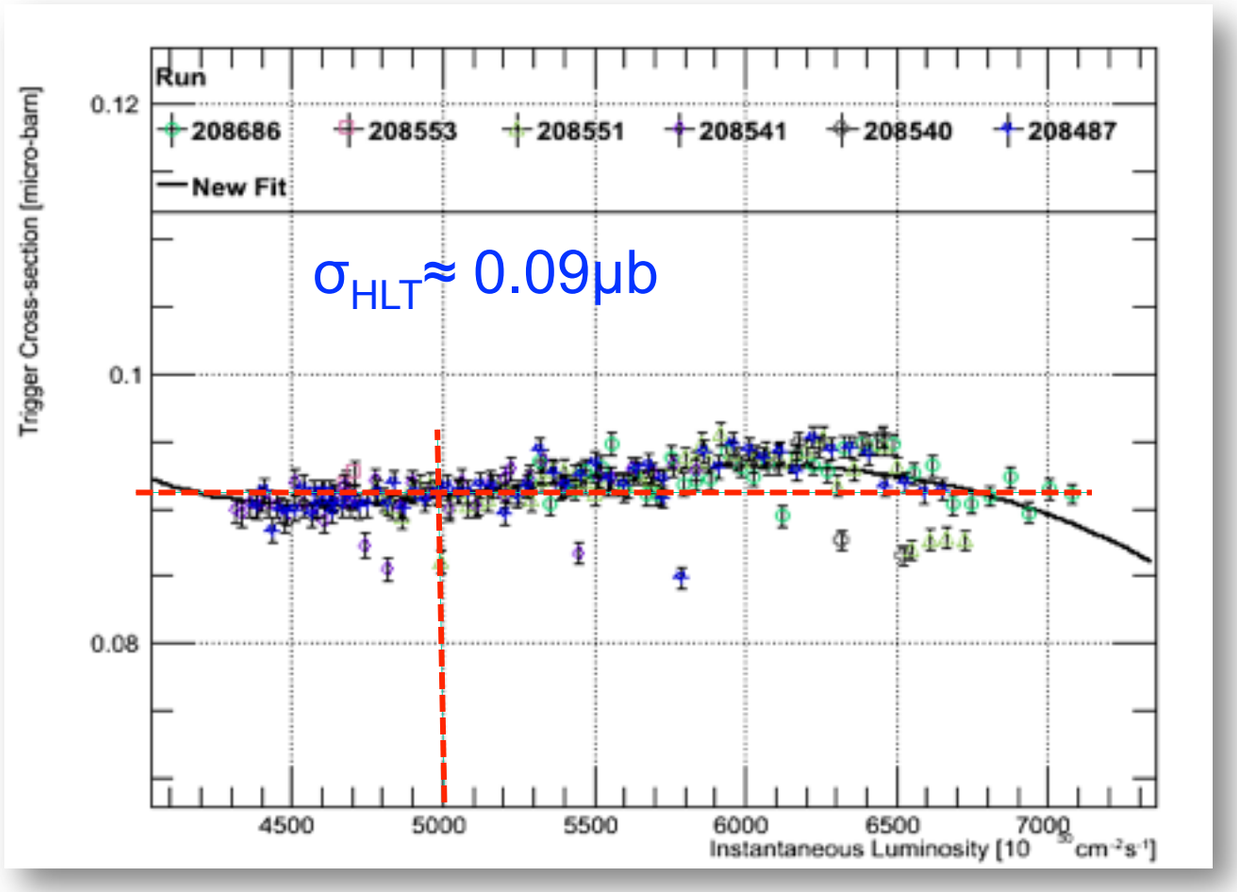
Threshold Optimization && Rate Calculation && Timing Measurement

Zeynep Demiragli
Brown University

Why do we have to reduce the rates?

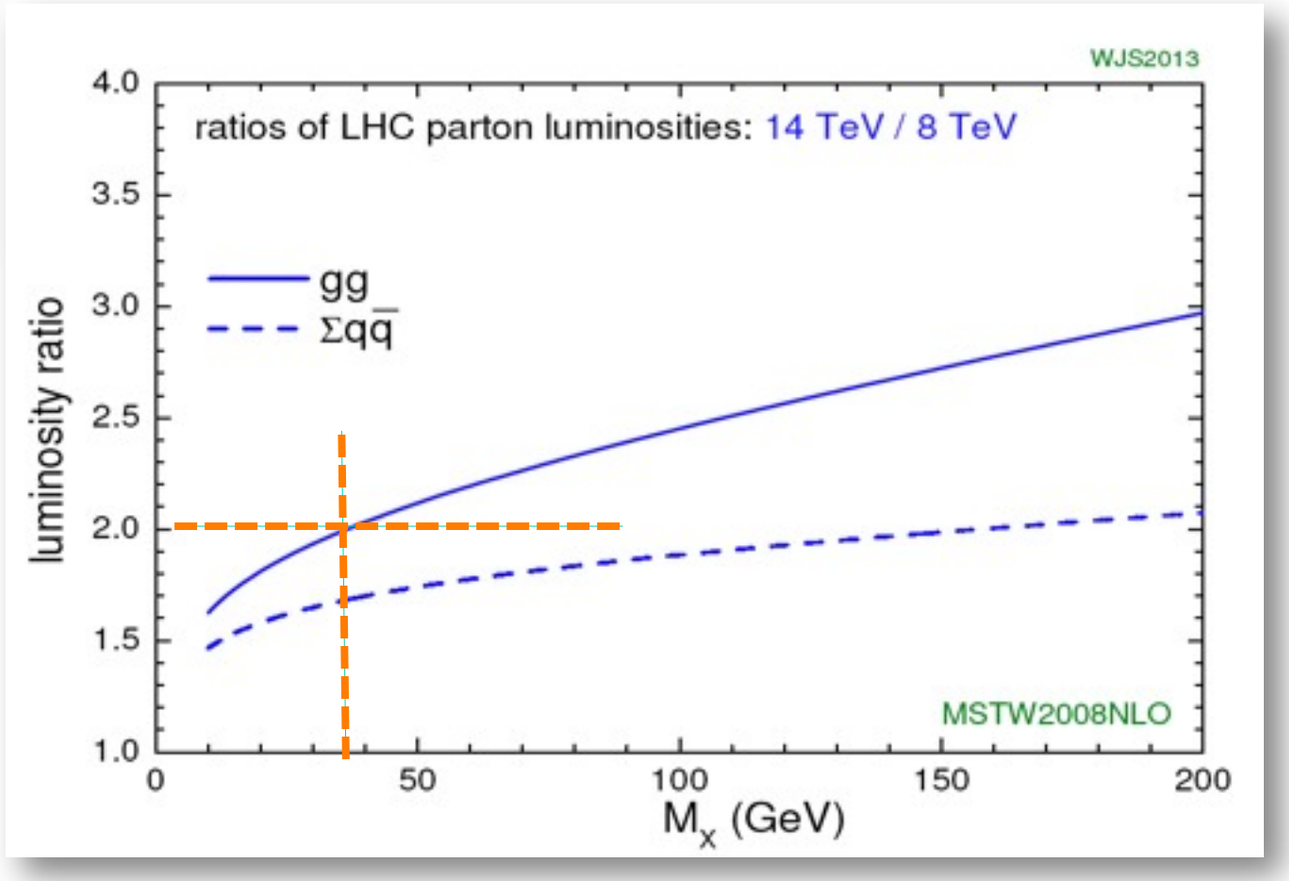


from Joe Incandela's talk

- 2012: 8 TeV HLT xsec ~0.09 μb
  - PU=25, small dependence on PU
- 8 TeV→ 14TeV ⇒ rates double
  - Average output rate of ~ 1.2kHz at $10^{34}cm^{-2}s^{-1}$ if menu untouched.

Also less parking options given no long shutdown.

If no other solution is found, simplest way of reducing the rate is increasing the pt & Et thresholds. (See the half rate menu)



HLT

How HLT Works ?

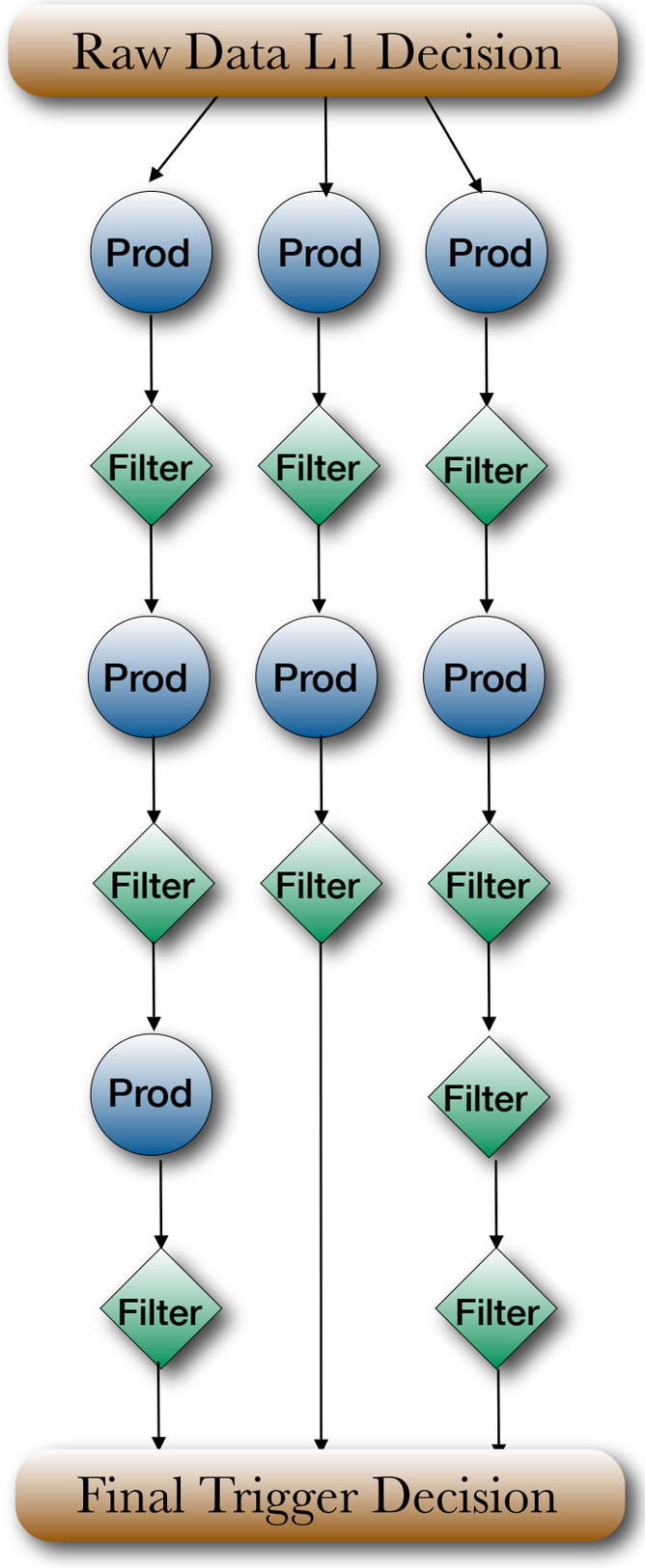
The HLT is made of many trigger paths, each path consists of several software modules which are either a “Producer” or a “Filter”:

Producers:

Perform unpacking of the raw data and the reconstruction of physics objects. These modules are taken from standard offline software, and therefore the reconstructed objects are similar to those used for the physics analysis. ***The producers account for the large majority of a path’s processing time.***

Filters:

Filters make an intermediate or final decision for the path based on the results of the producers. If an event passes a given filter, the next step of the reconstruction is performed. Once an event passes every filter in the path the event is accepted.



Open HLT

What is OpenHLT?

The OpenHLT tool is used to determine the optimal parameters for the filters of the HLT paths. The design works in two modes, Production and Filtering, which allows for faster trigger development.

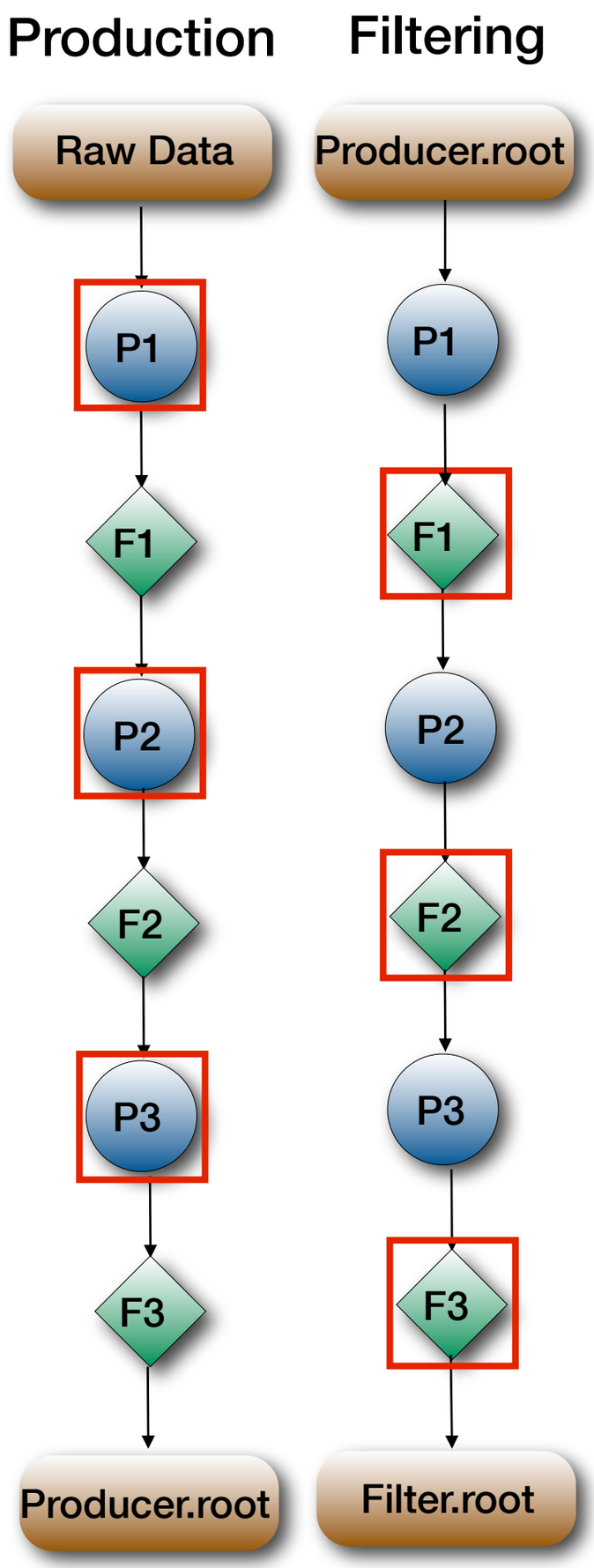
Production Stage:

In the production stage, the raw data is taken as input and all the HLT objects are produced. This time consuming step is typically run only once and no filtering is applied.

Filtering Stage:

In the filtering stage, the output of the production step is taken as input and all the filters are run on the already reconstructed objects. In this step, the user has direct access to the filter modules and can change the parameters to optimize the path of interest. The output of this step is in the standard CMSSW format.

https://twiki.cern.ch/twiki/bin/viewauth/CMS/NewOpenHLT



Framework in detail..

Advantage ?

The advantage is, you run the OpenHLT in the producer mode only once. Producing the product is the time consuming step. After you have all the producers, you can run the OpenHLT on the filtering mode as much as you want. And you can change the thresholds until the optimum rate is achieved.

Customizable !

OpenHLT framework has 4 list that will help you customize your output. These lists are:

prodWhiteList - that keeps track of the producers that needs to be rerun during filtering

pathBlackList - that keeps track of the paths that you don't want to run over

filterBlackList - that keeps track of the filters that you don't want to run over

productsToKeep - that keeps track of the producers that you would like to produce.

As default the tool is capable of running on all the paths that is currently in the menu.

Hands On Example!

Set up your workspace and check out the relevant packages:

```
cmsrel CMSSW_6_2_0
cd CMSSW_6_2_0/src
cmsenv

cvs co -r  V04-01-16      HLTrigger/btau
cp /afs/cern.ch/user/z/zdemirag/public/
forOpenHLT/* .

checkdeps -a
scram b -j4

hash -r   # using bash
rehash   # using [t]csh
```

Get the menu you would like to use (YOUR MENU):

```
hltGetConfiguration /users/jfernan2/Tutorial/MyHLT/V22
--full --offline --no-output --data --process TEST --globaltag
auto:hltonline --unprescale > hlt.py
```

To run your own table containing only HLT paths (and eventually additional ESModules), using the services and event setup from the master configuration, use:

```
edmConfigFromDB --cff --configName /dev/CMSSW_6_2_0/GRun --nopaths > setup_cff.py
```

Add to hlt.py this line, just after process = cms.Process("HLT")

```
process.load("setup_cff")
```

CAUTION: Use the global tag “PRE_P62_V8” both in setup_cff and hlt.py

Hands On Example!

You now want to change the filters in your path to achieve the optimum rate and timing while still maintaining good efficiency for your signal!

You first need to decide on the thresholds you would like to change, in our example we can change 3 obvious thresholds:

HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_v1

We can change the Muon Pt:

process.hltL3fL1sMu16Eta2p1L1f0L2f16QL3Filtered25Q = cms.EDFilter("HLTMuonL3PreFilter",

MinPt = cms.double(25.0)

We can change the Jet Pt:

process.hltIsoMu252p1CentralPFNoPUJet30MuCleaned = cms.EDFilter("HLTPFJetCollectionsFilter",

MinJetPt = cms.double(30.0)

We can change the B Tagging Discriminant:

process.hltSingleJetBLifetimeL3FilterMyBtagCSV = cms.EDFilter("HLTCaloJetTag",

MinTag = cms.double(0.7)

HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_v1

```
process.hltL3fL1sMu16Eta2p1L1f0L2f16QL3Filtered25Q = cms.EDFilter( "HLTMuonL3PreFilter",
    MaxNormalizedChi2 = cms.double( 20.0 ),
    saveTags = cms.bool( True ),
    PreviousCandTag = cms.InputTag( "hltL2fL1sMu16Eta2p1L1f0L2Filtered16Q" ),
    MinNmuonHits = cms.int32( 0 ),
    MinN = cms.int32( 1 ),
    MinTrackPt = cms.double( 0.0 ),
    MaxEta = cms.double( 2.1 ),
    MaxDXYBeamSpot = cms.double( 0.1 ),
    MinNhits = cms.int32( 0 ),
    MinDxySig = cms.double( -1.0 ),
    NSigmaPt = cms.double( 0.0 ),
    MaxDz = cms.double( 9999.0 ),
    MaxPtDifference = cms.double( 9999.0 ),
    MaxDr = cms.double( 2.0 ),
    CandTag = cms.InputTag( "hltL3MuonCandidates" ),
    MinDr = cms.double( -1.0 ),
    BeamSpotTag = cms.InputTag( "hltOnlineBeamSpot" ),
    MinPt = cms.double( 25.0 )
)
```

You can change other parameters too!

```
process.hltIsoMu252p1CentralPFNoPUJet30MuCleaned = cms.EDFilter( "HLTPFJetCollectionsFilter",
    saveTags = cms.bool( True ),
    originalTag = cms.InputTag( "hltAK5PFJetL1FastL2L3CorrectedNoPU" ),
    inputTag = cms.InputTag( "hltIsoMu252p1JetCollectionsForLeptonPlusPFJetsNoPU" ),
    MinJetPt = cms.double( 30.0 ),
    triggerType = cms.int32( 0 ),
    MinNJets = cms.uint32( 1 ),
    MaxAbsJetEta = cms.double( 2.6 )
)

process.hltSingleJetBLifetimeL3FilterMyBtagCSV = cms.EDFilter( "HLTCaloJetTag",
    saveTags = cms.bool( True ),
    MinJets = cms.int32( 1 ),
    JetTags = cms.InputTag( "hltL3CombinedSecondaryVertexBJetTags" ),
    TriggerType = cms.int32( 86 ),
    Jets = cms.InputTag( "hltSelectorJets20L1FastJet" ),
    MinTag = cms.double( 0.7 ),
    MaxTag = cms.double( 99999.0 )
)
```

Hands On Example!

We know what we want to change! Before we can use OpenHLT tools we need to make sure that non of the filters is an input for a producer.

Remember, we only want to run the production stage once so we need to make sure we decouple the stages appropriately.

Track down each filter to make sure no module uses a producer dependent on the filter:

```
process.hltIsoMu252p1CentralPFNoPUJet30MuCleaned = cms.EDFilter( "HLTPFJetCollectionsFilter",
    saveTags = cms.bool( True ),
    originalTag = cms.InputTag( "hltAK5PFJetL1FastL2L3CorrectedNoPU" ),
    inputTag = cms.InputTag( "hltIsoMu252p1JetCollectionsForLeptonPlusPFJetsNoPU" ),
    MinJetPt = cms.double( 30.0 ),
    triggerType = cms.int32( 0 ),
    MinNJets = cms.uint32( 1 ),
    MaxAbsJetEta = cms.double( 2.6 )
)

process.hltIsoMu252p1JetCollectionsForLeptonPlusPFJetsNoPU = cms.EDProducer( "HLTPFJetCollectionsForLeptonPlusJets",
    SourceJetTag = cms.InputTag( "hltAK5PFJetL1FastL2L3CorrectedNoPU" ),
    minDeltaR = cms.double( 0.3 ),
    HltLeptonTag = cms.InputTag( "hltL3crIsoL1sMu16Eta2p1L1f0L2f16QL3f25QL3crIsoRhoFiltered0p15" )
)

process.hltL3crIsoL1sMu16Eta2p1L1f0L2f16QL3f25QL3crIsoRhoFiltered0p15 = cms.EDFilter( "HLTMuonIsoFilter",
    saveTags = cms.bool( True ),
    PreviousCandTag = cms.InputTag( "hltL3fL1sMu16Eta2p1L1f0L2f16QL3Filtered25Q" ),
    MinN = cms.int32( 1 ),
    IsolatorPSet = cms.PSet(  ),
    CandTag = cms.InputTag( "hltL3MuonCandidates" ),
    DepTag = cms.VInputTag( 'hltL3MuonCombRelIsolations' )
)
```

This is a producer!

This is a filter we are changing!

Hands On Example!

What does this mean? What do I do now?

This simply means that you would have to re-run the producer whenever you want to change the cut for the Jet for this path. The jet cleaning producer bases its cleaning on the selected leptons, therefore each time you change the lepton pt cut, you should re-produce the associated jet cleaning producer.

We can get around this by generating as many paths as we want with different Muon Pt thresholds before we produce the producers! For this we will use "path_maker.py"

```
python path_maker.py -i hlt.py -o hlt_modified.py -p HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_v1
-c "hltL3fL1sMu16Eta2p1L1f0L2f16QL3Filtered25Q.MinPt = cms.double( 28.0 )" -r Changed_Mu28
```

The filter I would like to change (and the value I want to change it to)

The name of the path I want to modify

Output:

```
base HLT menu: top_trig.py
trigger path: HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_v1
requested changes: ['hltL3fL1sMu16Eta2p1L1f0L2f16QL3Filtered25Q.MinPt = cms.double( 28.0 )']
globaltag = PRE_P62_V8::All

Paths in the menu:
----------------------
HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_Changed_Mu28_v1
HLT_IsoMu24_eta2p1_v16
HLT_IsoMu24_v18
HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_v1
----------------------
```

Hands On Example!

```
usage: path_maker.py [-h] [-i FILE] [-o FILE] -p NAME -c CHANGES [CHANGES ...]
                     [-r NAME] [-v LEVEL]

This is a CMSSW tool to clone, modify and add a trigger path back into an HLT menu

optional arguments:
-h, --help            show this help message and exit
-i FILE, --input-hlt-menu FILE
                      name of the base hlt configuration file
-o FILE, --output-hlt-menu FILE
                      name of the hlt configuration file which will have the
                      new path
-p NAME, --base-trigger-path NAME
                      name of an existing trigger path you want to study
-c CHANGES [CHANGES ...], --trigger-path-changes CHANGES [CHANGES ...]
                      a space seperated list of strings describing the
                      changes to the base trigger path. example:
                      "hltEightJet35eta3p0L1FastJet.MinPt =
                      cms.double(60.0)"
-r NAME, --rename NAME
                      this name will be appended to the name of the current
                      trigger
-v LEVEL, --verbose LEVEL
                      set the verbosity level: 0:quiet, 1:normal, 2:debug
                      (default=1)
```

Hands On Example!

Now we are ready to produce all the necessary producers using the openHLT.py

```
usage: openHLT.py [-h] [-p] -i FILES [FILES ...] -o FILE [-t FILE]
                  [-c CHANGES [CHANGES ...]] [-n N] [-j] [-v] [-l FILE]
                  [-g FILE] [--go]
```

1) Run on production mode and produce all the products! This stage takes a long time! You can submit crab jobs for this stage using the -c option!

For interactive running, select a good run from Web Based Monitoring and pick up a root file for testing purposes:

python openHLT.py -p -i /store/data/Run2012D/SingleMu/RAW/v1/000/208/487/F0FDFDB5-E63D-E211-BDF5-5404A63886D4.root -o Prod.root -t hlt_modified.py -n 5000 --go

2) Run on filtering mode to get the # of events that has passed your trigger.

python openHLT.py -i Prod.root -o Filt.root -t hlt_modified.py -n 5000 --go

Hands On Example!

```
optional arguments:
-h, --help            show this help message and exit
-p, --run-producers   run configuration in producer mode (default: False)
-i FILES [FILES ...], --input-root-files FILES [FILES ...]
                      a space seperated list of input root file(s) (raw
                      data) (required)
-o FILE, --output-root-file FILE
                      openHLT output root file (required)
-t FILE, --hlt-config FILE
                      hlt configuration file (default: hlt.py)
-c CHANGES [CHANGES ...], --other-changes CHANGES [CHANGES ...]
                      a space seperated list of python code strings (each
                      string must be in quotes) to appear in the openHLT go
                      file. Intended as a quick&dirty way to modify module
                      parameters. Example pseudocode: "process.[module
                      name].[parameter]=cms.[data type]( value )"
-n N, --n-events N    maximum number of events to process (default: 1000)
                      Hint: -1 works when in quotes "-1"
-j, --crab-job        configuration file will run in a crab job
-v, --verbose         increase verbosity
-l FILE, --openhlt-template-file FILE
                      openHLT template file (default: openHLT.TEMPLATE)
-g FILE, --openhlt-go-file FILE
                      openHLT "go" file to be written out (default:
                      openhlt_go.py)
--go                  start cmsRun with the "go" file
```

You can modify paths further!!

python path_maker.py -i hlt_modified.py -o jet_35.py -p HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_v1 -c "hltIsoMu252p1CentralPFNoPUJet30MuCleaned.MinJetPt = cms.double(35.0)" -r Changed_Jet35

```
----------------------
HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_Changed_Mu28_v1
HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_Changed_Jet35_v1
HLT_IsoMu24_eta2p1_v16
HLT_IsoMu24_v18
HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_v1
----------------------
```

python path_maker.py -i jet_35.py -o hlt_final.py -p HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_Changed_Jet35_v1 -c "hltSingleJetBLifetimeL3FilterMyBtagCSV. MinTag = cms.double(0.9)" -r Changed_Btag09

```
----------------------
HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_Changed_Mu28_v1
HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_Changed_Jet35_v1
HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_Changed_Jet35_Changed_Btag09_v1
HLT_IsoMu24_eta2p1_v16
HLT_IsoMu24_v18
HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_v1
----------------------
```

When you are satisfied with your list of triggers do:

python openHLT.py -i /store/group/comm_trigger/TriggerStudiesGroup/OpenHLT/Prod_Top_Trig_Tutorial.root -o Filt.root -t hlt_final.py -n 5000 --go

How to calculate the rates?

1) If there is a looser path already present in one of the previously used menus when collecting data:

Then you are LUCKY! Your work is very little when you want to estimate the rates!

All you need to do is a select a good run(s) with a well defined luminosity regions (TSG usually wants rates in few different luminosities) and run the OpenHLT in the filtering mode!

Make sure you add the "looser path" to the list of trigger you are running over.

For our case, we have IsoMu24 as our "looser trigger"

Run	Passed	Failed	Error	Name
5000	52	4948	0	HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_v1
5000	2848	2152	0	HLT_IsoMu24_v18
5000	2639	2361	0	HLT_IsoMu24_eta2p1_v16
5000	46	4954	0	HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_Changed_Mu28_v1
5000	47	4953	0	HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_Changed_Jet35_v1
5000	26	4974	0	HLT_IsoMu25_eta2p1_CentralPFNoPUJet30_BTagCSVd07_Changed_Jet35_Changed_Btag09_

All we need to do is the relative ratio of our new trigger to the loose trigger and multiply this ratio with the rate of the loose trigger for the run we have used!

How to calculate the rates?

WBM is your friend! https://cmswbm.web.cern.ch/cmswbm/

1)

Core Services

RunSummary [24h] [24h&1+trig]
RunTimeSummary [LHC Fills] Deadtime
FillReport [Latest Fill] DataSummary

2)

RunSummary for Specific Runs

Enter a RunNumber *or* LHC Fill and press **return**; Recent LHC Fills

CMS RunNumber: 208487 Submit

3)

All times are in UTC

RUN	LUMI_NB_LIVE_DELIV	SEQUENCE	TRIGGER_MODE	
208487	74.774977 76.264262	GLOBAL-RUN toppro	l1_hlt_collisions_2012/v150	1

<208486|208493> **Sub-system specific WBM services:** CSC | DT | ECAL | HC RPC | TRACKER

BField	3.799 Tesla
InitialPrescaleIndex	3
Tier0Transfer	true
TriggerMode	l1_hlt_collisions_2012/v150
TTC MI Key	beam1-manual-20110413
L1 Key	TSC_20121110_002952_collisions_BASE
HLT Key	/cdaq/physics/Run2012/8e33/v2.1/HLT/V7
HLT Version	CMSSW_5_2_7_HLT3

4)

243	HLT_IsoMu24_v17 (101261)	635 (635)	114511601	114511601	544204	36.77

HLT trigger rates for HLT_IsoMu24_v17: run 208487, path 101261

LS number from [ ] to [ ]

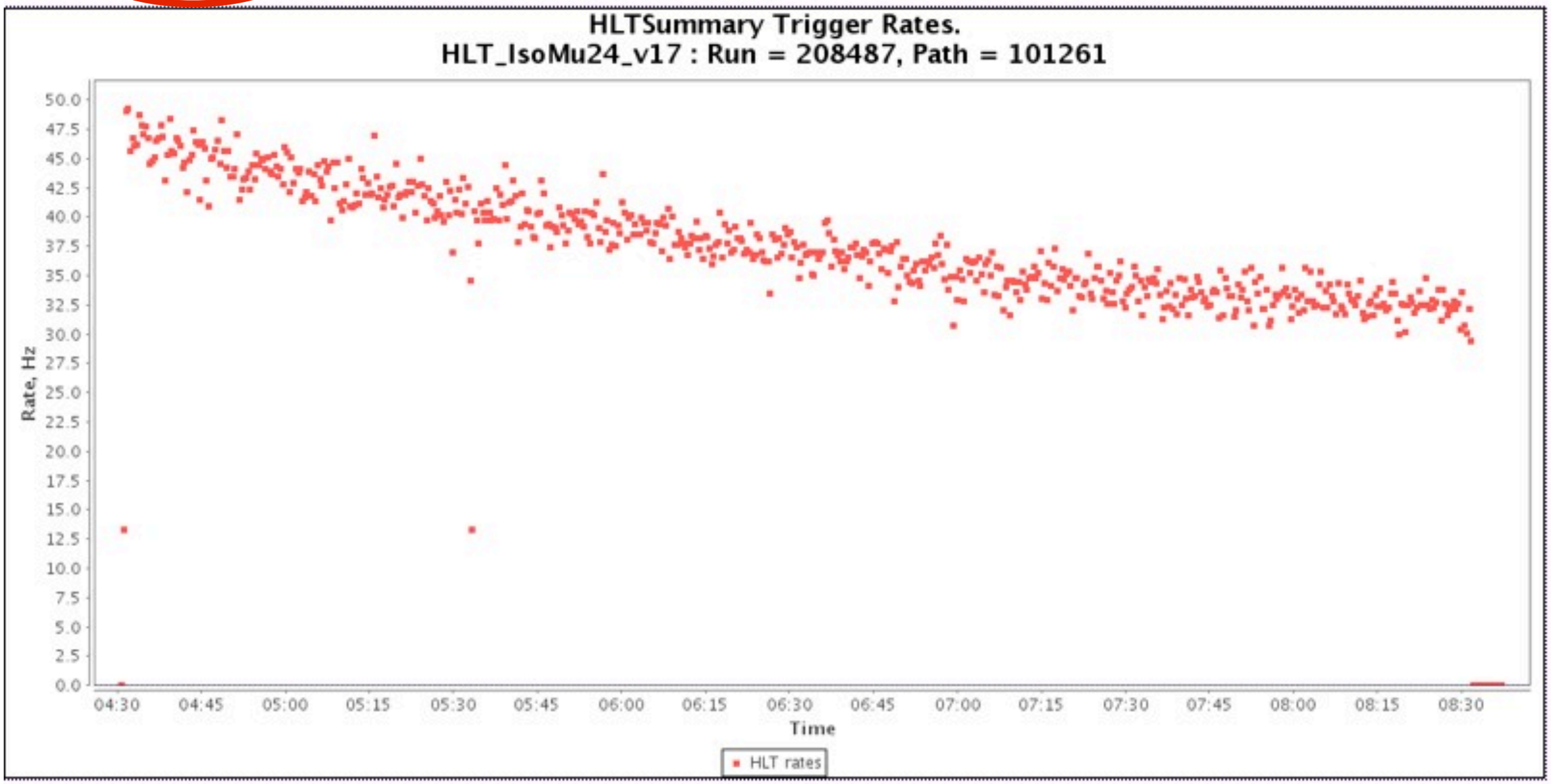
Rate, Hz: Min. [ ] Max. [ ]

☐ Logarithmic scale ☐ Lines Draw: [Rates] per [Timestamp]

[Submit Query]

Run	Path ID	Path Name	LS length, sec.	First Timestamp	Last Timestamp	First LS	Last LS
208487	101261	HLT_IsoMu24_v17	23.3104	2012.12.04 04:30:50	2012.12.04 08:37:09	1	635

Entries	Avg. Rate, Hz	RMS Rate, Hz	Min. Rate, Hz	Max. Rate, Hz	Avg. Count	RMS Count	Min. Count	Max. Count
635	36.7653	37.4857	0.0000	49.2055	857.0142	873.8059	0	1147



You can specify the Lumi Sections!!

How to calculate the rates?

If there is no looser path already present in one of the previously used menus when collecting data then you have to skim over the loosest possible trigger in the menu: HLT_Physics which is an or of all the L1 in the menu!

This is done by adding following few lines in the beginning of hlt.py :

```
process.load('HLTrigger/HLTfilters/hltHighLevel_cfi')
process.hltHighLevel.TriggerResultsTag = cms.InputTag("TriggerResults","","HLT")
process.hltHighLevel.HLTPaths = ("HLT_Physics_vXX",)
```

Make sure to use a consistent version

And adding:

```
process.YOURPATH = cms.Path( process.hltHighLevel + process.HLTBeginSequence + ...)
```

Log file and it will tell you if that particular event has fired the HLT_Physics online or not. And if it fired the HLT_Physics it will see if it will fire the your path or not.

Looking at the number of events fired for your path and for HLT_Physics you can estimate the rate of your path by simply multiplying the rate of the HLT_Physics path with the ratio you get from # of events that fire HLT_Physics and your path.

How to calculate the rates?

When you are proposing a trigger to the TSG, make sure you present rates for different pile up scenarios to show the behavior of your path as a function of pile up. You can add the root files to the hlt.py and add the necessary lumi mask.

Example (PU 20 and PU 10 recommendation is on the backup):

###30PU
lumisToProcess = cms.untracked.VLuminosityBlockRange('207454:139-207454:165'),

PU 30:
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/BA7406CD-AC30-E211-9D49-5404A63886CC.root
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/AA7326E2-AB30-E211-8694-001D09F244DE.root
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/8CBE3C51-AE30-E211-9AA4-BCAEC532971D.root
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/8C630651-AE30-E211-B1D8-BCAEC5364CFB.root
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/DE8A7DDB-AB30-E211-850B-003048D2BC5C.root

```
process.source = cms.Source( "PoolSource",
    fileNames = cms.untracked.vstring(
'/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/BA7406CD-AC30-E211-9D49-5404A63886CC.root',
'/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/AA7326E2-AB30-E211-8694-001D09F244DE.root',
'/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/8CBE3C51-AE30-E211-9AA4-BCAEC532971D.root',
'/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/8C630651-AE30-E211-B1D8-BCAEC5364CFB.root',
'/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/DE8A7DDB-AB30-E211-850B-003048D2BC5C.root',
    ),
    secondaryFileNames = cms.untracked.vstring(
    ),
    inputCommands = cms.untracked.vstring(
        'keep *'
    ),
    lumisToProcess = cms.untracked.VLuminosityBlockRange('207454:139-207454:165'),

)
```

How to estimate the timing of the path?

Why do we have to estimate the timing of our path?

The general idea is to measure the CPU time of your path and make sure it doesn't add too much time to the menu, because there is a hard limit on the processing time per event for the HLT (or else we would get dead time).

Also, there are a lot of variables that affect the measurement so we try to all do the same thing (use **vocms110**, standard skims) so we can compare accurately against other peoples results.

To log into the **vocms110** simply do:

ssh -X username@lxplus5
ssh -X username@vocms110

For this exercise only **DO NOT** use **vocms** machines. Keep working in the lxplus machines.

How to estimate the timing of the path?

Set up the appropriate CMSSW release (620 in our case) you wish to run in.

Check out the most recent version of the FastTimerService (https://twiki.cern.ch/twiki/bin/view/CMS/FastTimerService) and compile.

addpkg HLTrigger/Timer V01-13-02

scram b

Copy a recent online menu compatible with the release to your area in confDBD (you know how to import menus!) in this case /online/collisions/2012/8e33/v2.2/HLT/V7. Then import your path into that menu, and save. This is also a good check to see if your path will be able to integrate well to the rest of the menu.

To get your menu with the timing services now do:

hltGetConfiguration /users/zdemirag/timing/hlt_online8e33_v2pt2_v7_testpath/V2 --full --offline --data --timing --globaltag auto:hltonline --no-output > hlt_8e33_v2pt2_v7_noutput_withtestpath.py

Be sure to run with the proper prescales (as this will effect the timing!) and use the **--timing** option to load the FastTimerService.

How to estimate the timing of the path?

Look into the menu you dumped. In my case it is: hlt_8e33_v2pt2_v7_noutput_withtestpath.py

1) Modify the input files: Use the recommended files

(Ask for the recent ones from Grant Christopher / Aram Avestiyan)

2) Add a lumi mask: The lumi mask is also provided by them.

3) Put the path you are interested in at the end of the list of cms.Paths in your hlt file. For this example it will be like this already.

4) Change the maximum number of events (approx 30K is usually for timing..)

You are ready to RUN!! (it will take about an hour, do nohup and pipe the output to a log file)

taskset -c 0 cmsRun hlt_8e33_v2pt2_v7_noutput_withtestpath.py

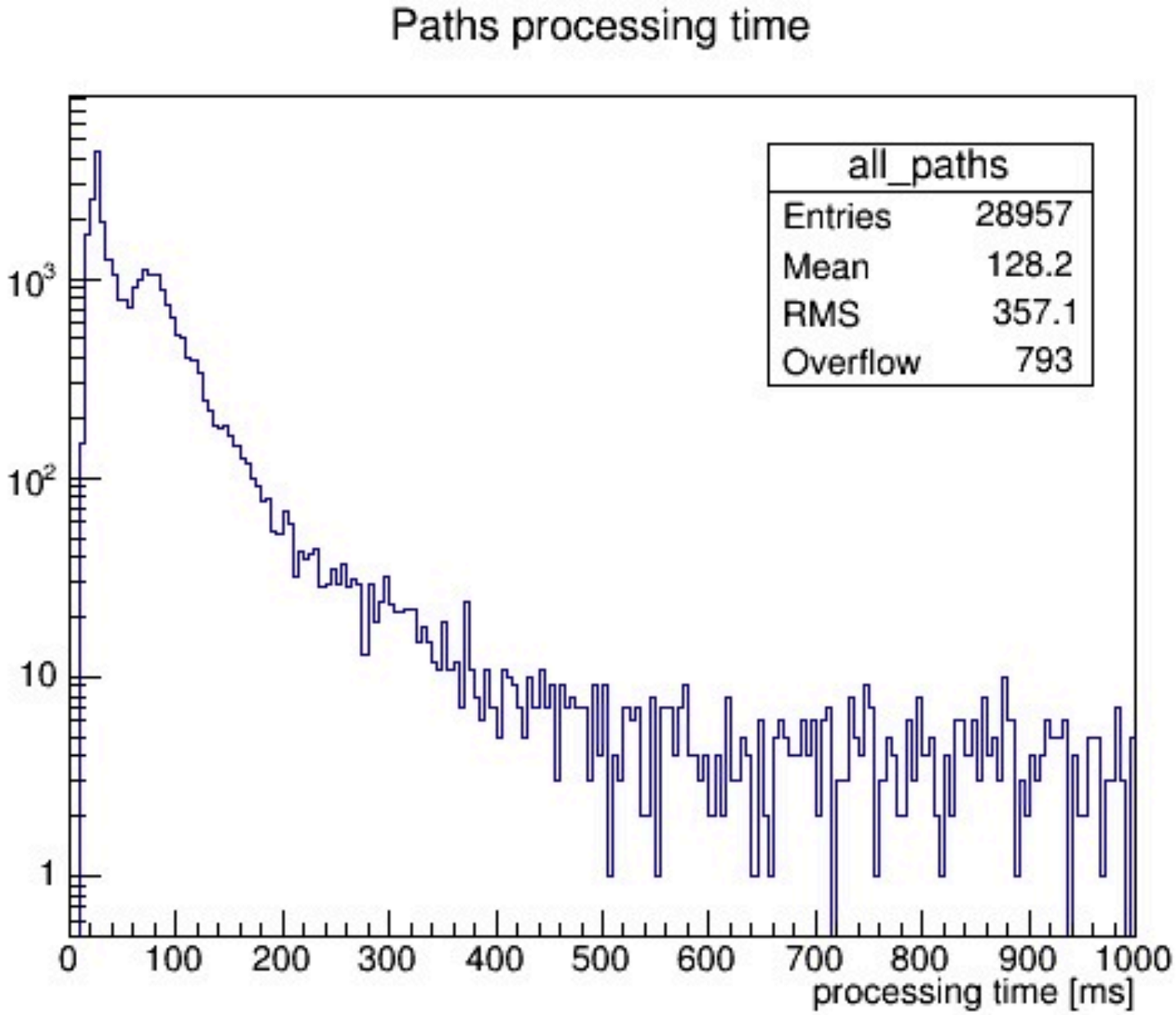
You should end up with a file called:

DQM_V0001_R000207454__HLT__FastTimerService__All.root. Look in the folder "DQMData/Run 207454/HLT/Run summary/TimerService/"

How to estimate the timing of the path?

You can pick up the output from this directory if you haven't run one for yourself: /afs/cern.ch/user/z/zdemirag/public/TopTriggerTutorial

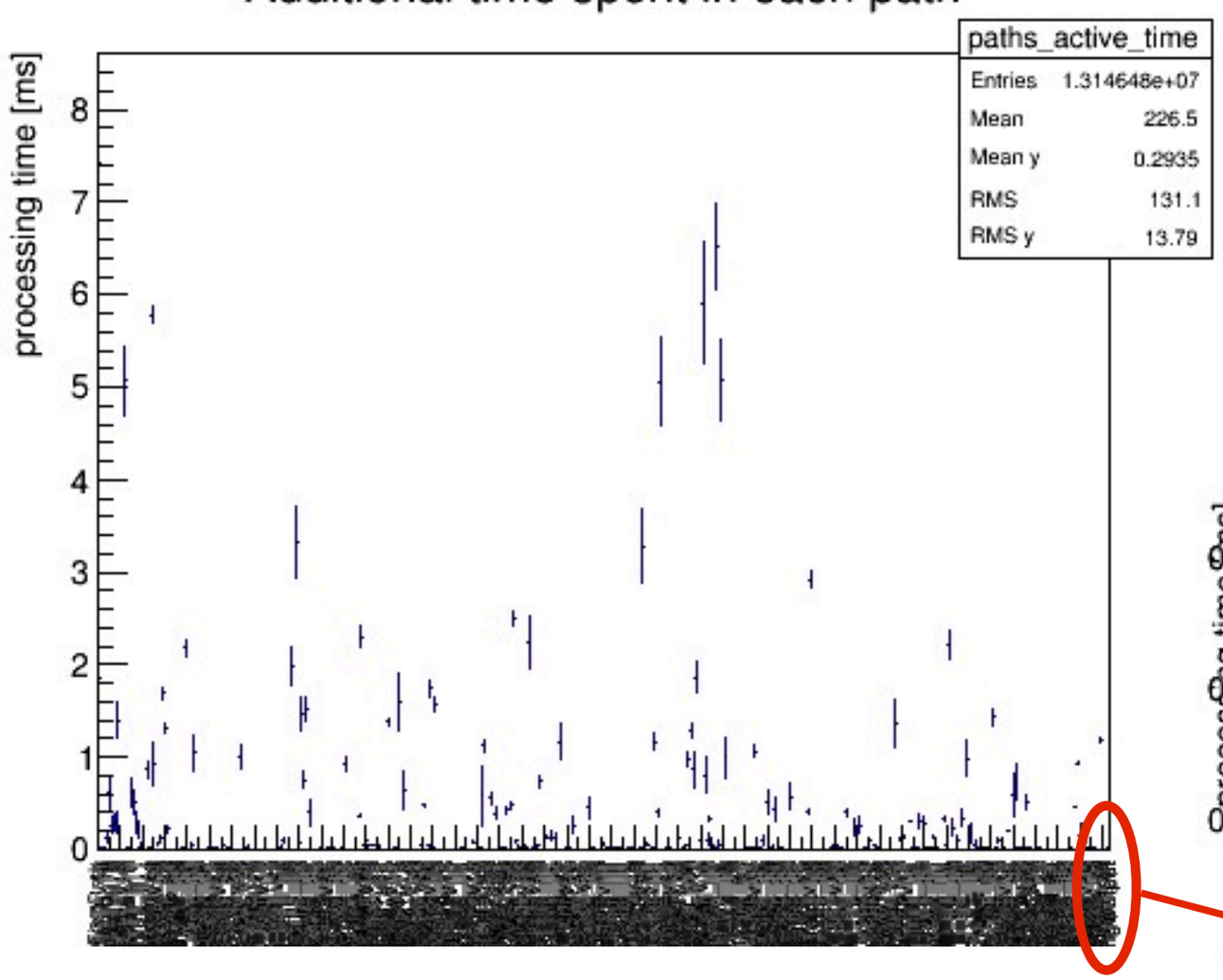
The total menu time per event given by the histogram "all_paths".



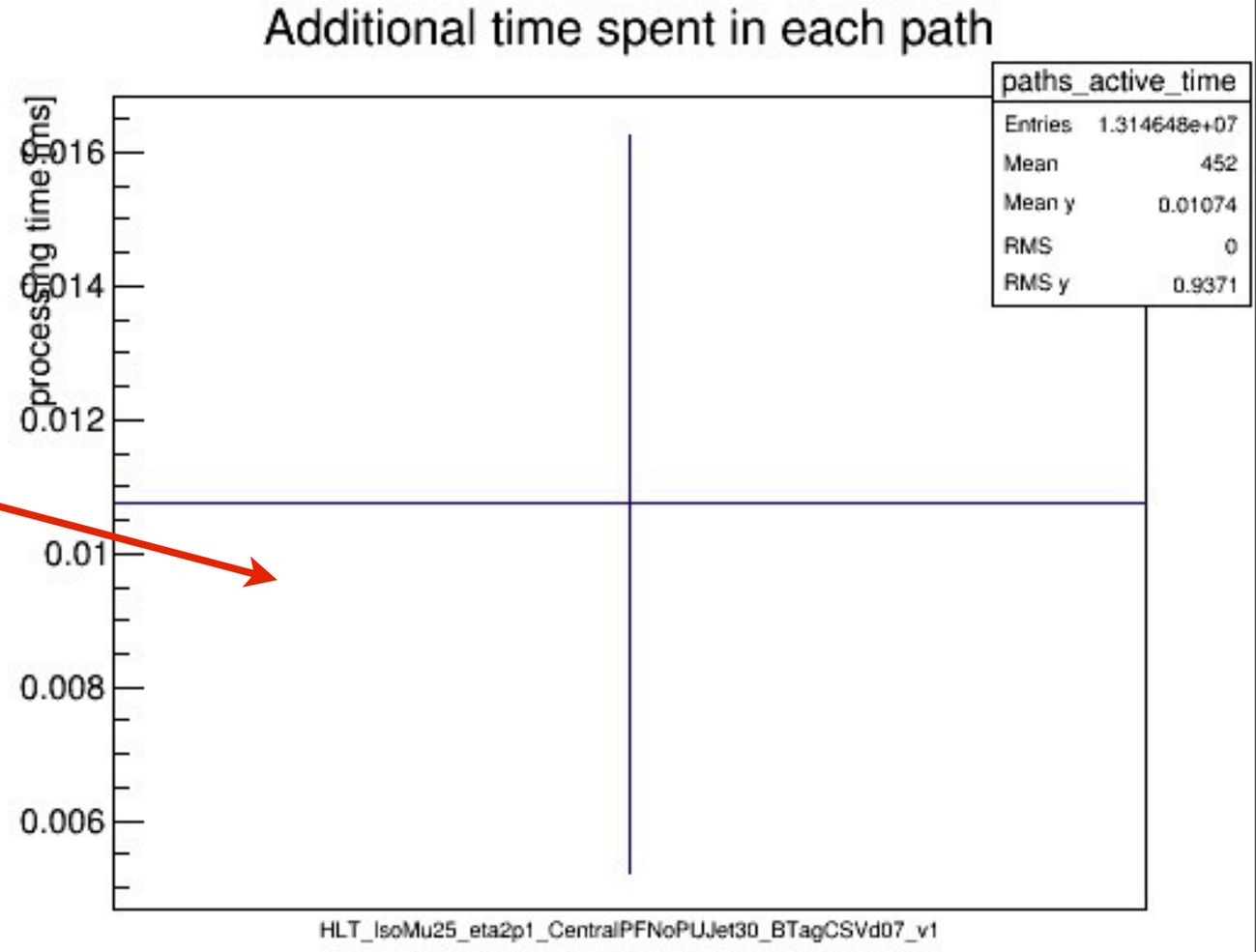
This distribution is good to compare against other people's results to make sure you are doing everything correct.

This distribution excludes the time it takes to open and close the files.

How to estimate the timing of the path?

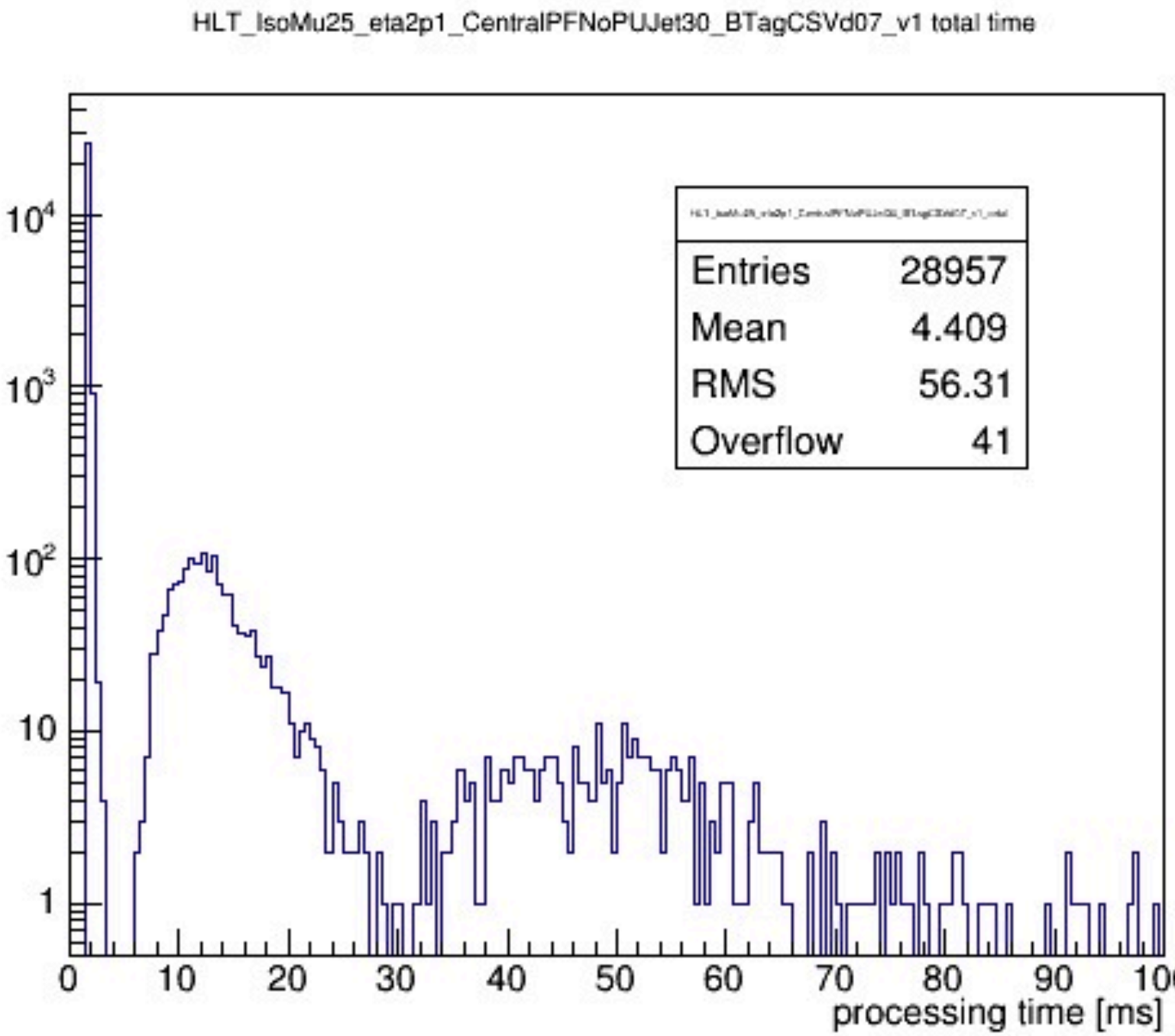


Now look at path's active time. This distribution counts the additional time spent in each path as they run. Only the last path in here has much meaning.



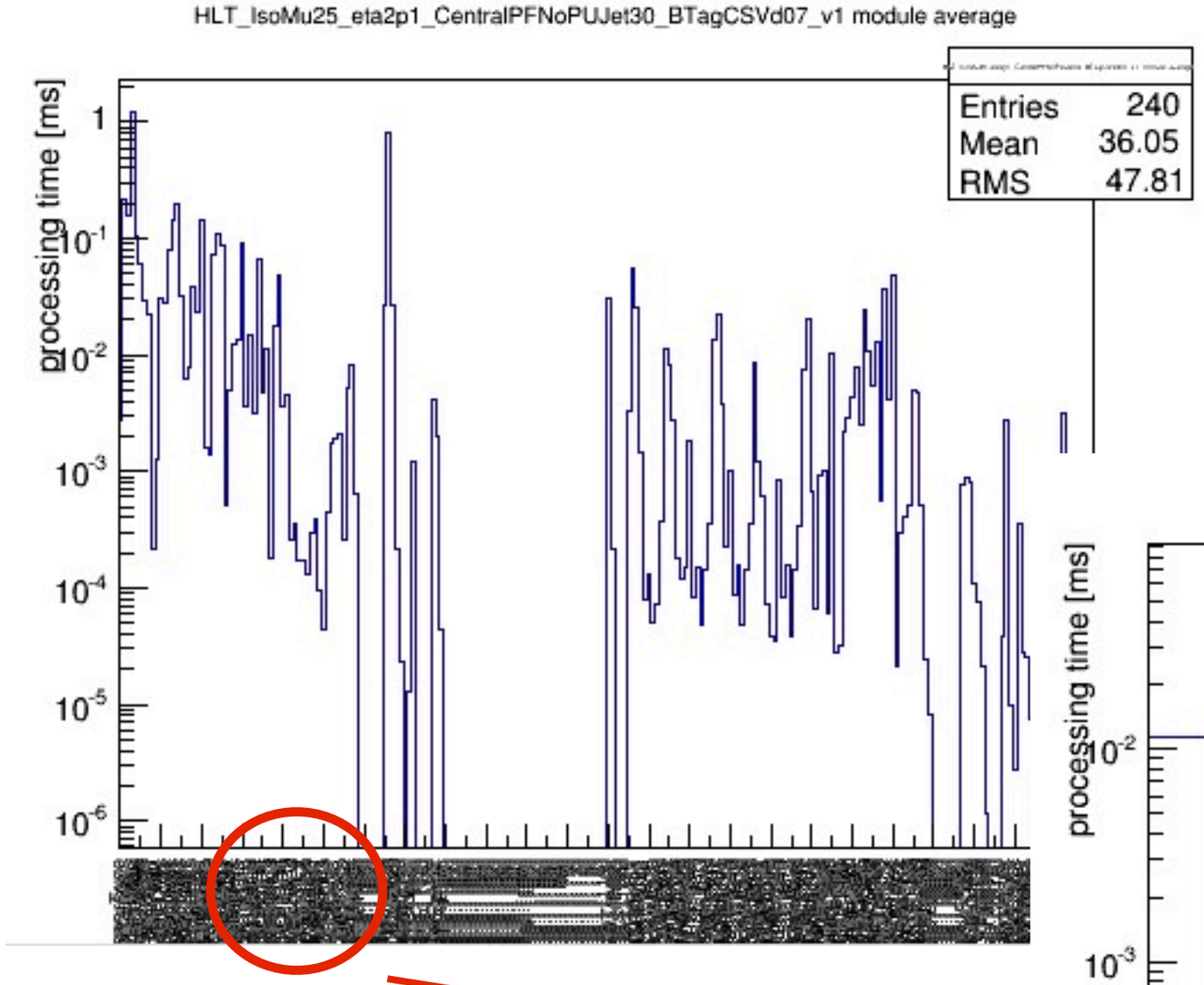
Since we put our path last, we can read-off how much it added to the menu. In this case 0.01 ms! (GREAT!)

How to estimate the timing of the path?

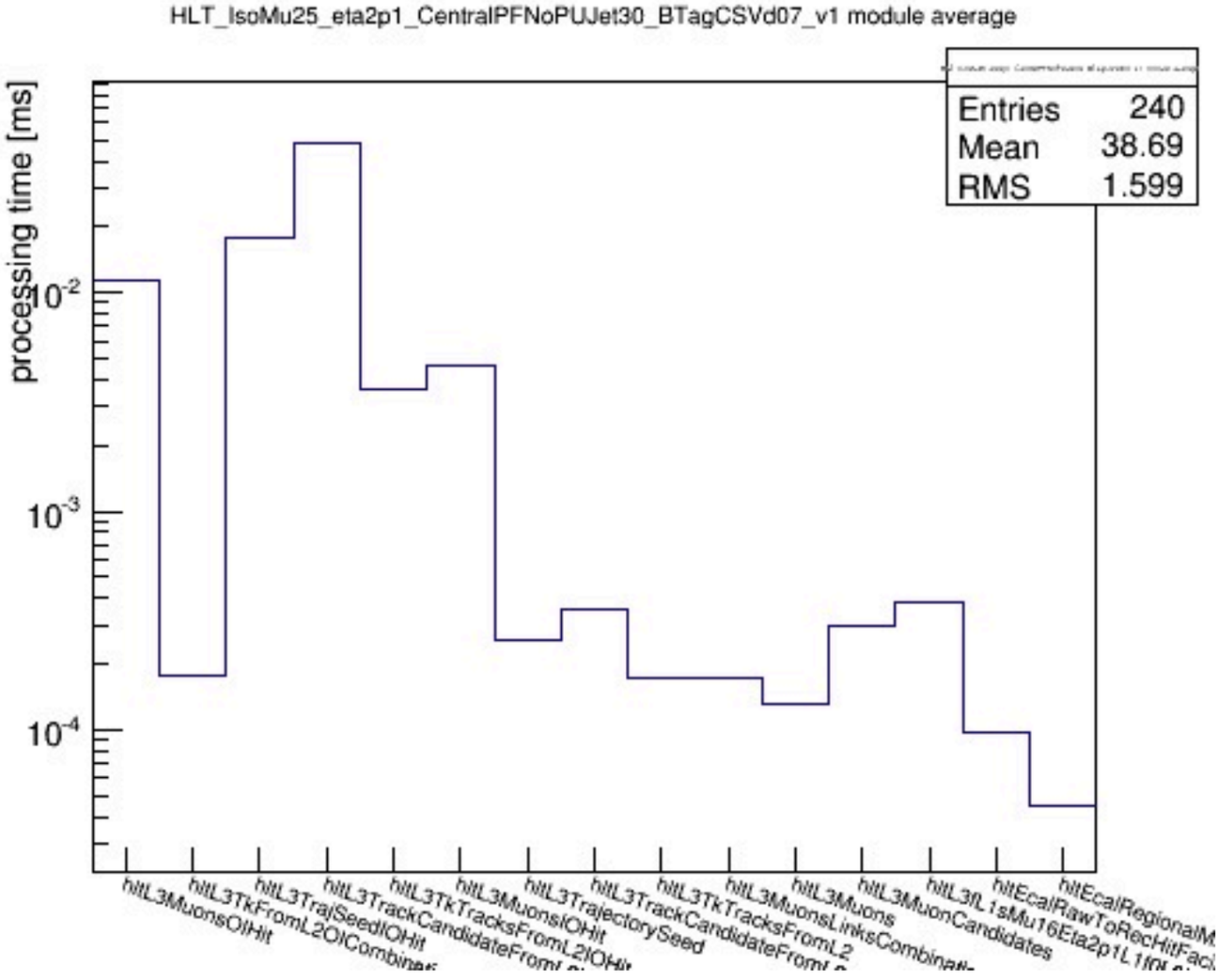


Now look into the Paths directory and find the trigger you are interested in. Look for the total suffix. This distribution will give you the average time it took for your path to run. In this case it is 4.4 ms which is greater than the exclusive time, because it shares modules with the other paths with the menu.

How to estimate the timing of the path?



If your path does take a lot of time, you can use the module_average plot to see which module is taking the most time.



Conclusion

You now know how to optimize the thresholds for your trigger to achieve the optimum balance between rate, timing and acceptance!

Once you are happy and the TSG is happy with your trigger you will follow the steps for integration tests to make sure your trigger is compatible with the rest of the menu!

Once the trigger integration is done and your trigger is put in online menus, your trigger will start collecting data which will be available to be used by everyone in the collaboration in their respective physics analysis!

Your job is not done! You will have to provide necessary monitoring for your path (next talk!) and provide help to people who would like to use your trigger if they need any help..

How to calculate the rates?

###30PU
lumisToProcess = cms.untracked.VLuminosityBlockRange('207454:139-207454:165'),

###20PU
lumisToProcess = cms.untracked.VLuminosityBlockRange('207454:730-207454:800'),

###10PU
lumisToProcess = cms.untracked.VLuminosityBlockRange('207454:1820-207454:1880'),

PU 30:
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/BA7406CD-AC30-E211-9D49-5404A63886CC.root
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/AA7326E2-AB30-E211-8694-001D09F244DE.root
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/8CBE3C51-AE30-E211-9AA4-BCAEC532971D.root
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/8C630651-AE30-E211-B1D8-BCAEC5364CFB.root
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/DE8A7DDB-AB30-E211-850B-003048D2BC5C.root

PU 20:
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/F66F7F66-D130-E211-907A-003048CF99BA.root
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/4EE46365-D130-E211-AF2D-002481E0DEC6.root
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/AA929065-D130-E211-B5F7-003048D2C108.root

PU 10:
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/3C7CBED9-FE30-E211-9EED-5404A63886BB.root
/store/data/Run2012D/HLTPhysicsParked/RAW/v1/000/207/454/CE6AA9F5-FE30-E211-9742-003048D2BC4C.root