Table of Contents

HLT CPU-Timing FAQ

This is a sandbox for HLT CPU-Timing FAQ twiki page. This page is under development.

How do I run HLT CPU-timing?

Please refer to the HLTCpuTiming twiki for instructions on how to run cpu-timing studies on data events and on Monte Carlo (MC) simulated events

What are the important aspects to consider when preparing a trigger approval talk for the TSG?

This section refers mainly to the plots that come out as a result of running the hltTimingSummary macro.

1. Your proposed path should not add a significant amount of time to the whole menu. This can be shown by comparing the totalTime plot with and without your new path. The hltTimingSummary macro can help you make the comparison quickly and easily. See this section for instructions on how to do that.
2. Your proposed path should not add to the timing tail of the total time distribution. This can be checked by looking at the overflows in the totalTime plot. The hltTimingSummary can help you check which paths contribute to the overflow. Take a look at this section for instructions on how to do that. In case needed, you can refer to this section to learn how to select problematic (time consuming) events that can help you debug your code.
3. You can also show some additional timing plots (e.g. average path time, average module running time for modules in the path).
4. The timing study should be performed on a `MinBias` sample which is skimmed on representative L1 seeds. Ideally one needs at least ~20K events in order to perform the test.
5. In general, a new (or any) path should comply with the CPU-timing requirements for the online HLT menu.

What are the CPU-timing requirements for a trigger path?

These are the general guidelines concerning the CPU-timing of a trigger path:

- The target path average time should not exeed 10 ms.
- The abolute maximum average time cannot exceed 40 ms.
- The absolute maximum for execution of a path under any circumstances should not exceed 100 ms.
- No path should increase (by itself) the total time taken by the whole HLT menu by more than 5 ms.

What is the hltTimingSummary macro, and where do I get more information?

The **`hltTimingSummary`** is a utility designed to summarize timing information for trigger paths in a High Level Trigger configuration. More information can be found in the HLTTimingSummary twiki.

In order use the macro, **`HLTPerformanceInfo`** objects need to be available in your `root` input file. In order to achieve this, you will need to first perform a CPU-timing measurement. Please refer to the HLTCpuTimingInstructions twiki for guidelines on how to do this using data and/or Monte Carlo simulations.

What sample should I use for timing studies?

The timing study should be performed on a `MinBias` sample which is skimmed on representative L1 seeds. When using data, it should also be done with a reasonably high instantaneous luminosity run, since performance scales with luminosity (L1 input rate). In case of doubt, you can contact the Trigger Menu Development team for an appropiate sample.

Data Samples

2011 Data

For 2011 data, samples are obtained by skimming the complete MinimumBias PD (for a given run) by requiring only events that fired the HLT trigger `HLT_Physics`, which mimics reasonably well the L1 input rate to the HLT as seen on-line.

The names quote the Run number, the skim type, make note of the product content of the files (only the `FEDRawDataCollection` is kept), and the number of events it contains. Ideally one needs at least ~20K events in order to perform the test (this will be done when statistics permit):

```
/castor/cern.ch/user/e/ecarrera/hlt_debug_skims/forHilton/run163592_HLTPhysicsSkim_RAW001_30K.roo
/castor/cern.ch/user/e/ecarrera/hlt_debug_skims/forHilton/run163592_HLTPhysicsSkim_RAW002_29K.roo
```

For other runs, possibly more recent, you can check this twiki .

NOTE: For some older releases, it is not possible to re-run the HLT across many runs due to some bug in the HLT muon code. This has been fixed by middle July 2011 or so.

2010 Data

For 2010 data, the following samples were obtained by skimming the complete MinimumBias PD (for a given run) by requiring only events that fired the L1 technical trigger bit 4 (ZeroBias) **AND** any L1 enabled trigger bit to mimic reasonably well the L1 input rate to the HLT as seen on-line (see for example these comparison plots).

The names quote the Run number, the skim type, make note of the product content of the files (only the `FEDRawDataCollection` is kept), and the number of events it contains. Ideally one needs at least ~20K events in order to perform the test, but this is no longer possible with the current conditions.

```
/castor/cern.ch/user/e/ecarrera/hlt_debug_skims/forHilton/run147451_L1ZBSkim_RAW000_36K.root
/castor/cern.ch/user/e/ecarrera/hlt_debug_skims/forHilton/run147451_L1ZBSkim_RAW001_33K.root
/castor/cern.ch/user/e/ecarrera/hlt_debug_skims/forHilton/run147451_L1ZBSkim_RAW002_5K.root
/castor/cern.ch/user/e/ecarrera/hlt_debug_skims/forHilton/run148002_L1ZBSkim_RAW_0.6K.root
/castor/cern.ch/user/e/ecarrera/hlt_debug_skims/forHilton/run148819_L1ZBSkim_RAW_3.3K.root
/castor/cern.ch/user/e/ecarrera/hlt_debug_skims/forHilton/run148862_L1ZBSkim_RAW_5.8K.root
/castor/cern.ch/user/e/ecarrera/hlt_debug_skims/forHilton/run149181_L1ZBSkim_RAW_10K.root
```

Monte Carlo Samples

The Trigger Menu Development team provided L1 skims for the 5E32 Hz/cm² menu, which are now available for timing studies. The skims are based on the new L1 menu, `L1Menu_Collisions2011_v0`, and include the L1 jet corrections. Two skims are available from the new Spring11 MinBias MC production samples: one with no pileup, and one with 10 PU events/BX. The files are located at CERN at:

```
  /castor/cern.ch/user/a/apana/l1skims/5e32/MB_NoPU_v2
  /castor/cern.ch/user/a/apana/l1skims/5e32/MB_PU10_v2
```

There are some specific instructions on how to use these samples for timing studies. Please refer to the HLTCpuTiming twiki, particularly the running on MC events section.

The L1 bits used for the skimming and their prescale values are listed in `/afs/cern.ch/user/a/apana/public/L1Skims_5e32/l1bits_prescales_v2.txt`.

For completeness, we mention an earlier version of these skims:

```
  /castor/cern.ch/user/a/apana/l1skims/5e32/MB_NoPU
  /castor/cern.ch/user/a/apana/l1skims/5e32/MB_PU10
```

The L1 bits used for the skimming and their prescale values are listed in `/afs/cern.ch/user/a/apana/public/L1Skims_5e32/l1bits_prescales.txt`.

Which machine should I use for timing studies?

Please refer to this page for instructions.

Out of the many plots produced by the hltTimingSummary macro, what are the most important to consider?

All plots produced by the `hltTimingSummary` macro can be very useful to understand the performance of the HLT menu and its components, but the two most important are the **`Total time for all modules per event`** and the **`Average time per path`** plots. For a brief description of the plots given by the `hltTimingSummary` macro and the pdf maker please refer to the [[][pdf Summary]] page.

In the following, examples plots are shown. The actual plots may change due to L1 composition of data or monte carlo simulation events and/or the HLT menu.

How should I interpret/read the total time distribution plot?

The **`Total time for all modules per event`** plot, plot found in the main pdf file and the specific pdf file, shows the distribution of the event total time in a run. A total time of an event is the sum of the time by the executed modules.

The `total time` plot, like the one in the image above, can be divided into four regions. The distribution may change due to L1 composition of data or monte carlo simulation events and/or the HLT menu..

1. **First peak (from 0 to 27 msec)**: The first peak consists of events where the signals fail very quickly after it enters the HLT. These events correspond to events that only execute some fast and basic modules, like checking trigger type of the L1 seeds and checking if there is L1 seed. When an event has an undesired trigger type or it has no information from the L1, the event fails immediately with a very short total time. These fast events contribute to the first peaks.
2. **Second peak (from 27 to 46 msec)**: The second peak consists of events where signals are further analyzed by executing the related modules. Once the HLT recognizes the L1 seeds, the signals then triggers modules to unpack the detector information. Depending on the features of the signals, different modules are called to execute the HLT trigger path, which aims at identifying the corresponding particles/physics. For example, to ensure an electron candidate is an electron, one has to first unpack the information in the surrounding ECAL.
3. **Tail (from 46 to 200 msec)**: However, some signals require longer time for analysis; these long events contribute to the tail. For example, while regional unpacking is possible for certain objects (like electrons), other candidates may be identified by unpacking information from several regions. Jet candidates, for instance, need to unpack both ECAL and HCAL information because jets cover both regions in the detector.
4. **Overflow (from 200 msec to infinity)**: In order to show a clear distribution on the total time plot, the macro selects, by default, the best time ranges to plot. Events that take very long time falls into the overflow, which is shown in the legend. Note that the macro takes into account the overflow cases when calculating the mean total time.

How do I check the total timing for events accepted and rejected by the HLT?

Sometimes, it is useful to separate the total average time for those events that were rejected by the HLT from those that were accepted. This is possible if you run the timing test with a minor modification to the timing python configuration (check the HLTCpuTiming twiki for for details on how to obtain such config file). The trick consists in changing the `hltBoolTrue` module in the CMSSW sequence for the `HLTriggerFinalPath` to `hltBoolFalse`, this will prevent all the events to be "accepted" by the system.

Newer versions of the `hltTimingSummary` macro create plots, as the one just above for the total timing, but separating events that were accepted and rejected by the system. If your output root file of the `hltTimingSummary` does not contain these plots, there is a developing version of the `hltTimingSummary` macro that you can use. Please take a look at this twiki in order to take advantage of this improvement.

How should I interpret/read the average time per path plot?

The "average time per path" plot shows average path time for each path in a run. Average path time of a path is the sum of time taken by the path in all events divided by the total number of events. The plot shows, in average, which paths take the longest time in an event.

In the given example, the first path (from the left) has the largest average path time of about 8.8 msec. Note that, because this example was obtained using a MinBias dataset, the paths with zero path time are the paths related to the Calibration events and other events like Randoms or DTErrors, which don' t populate the MinBias dataset.

How can I determine which paths contribute to the different regions in the total time distribution plot?

One can use the option `-q` to plot the average path time for all the overflow events, but this information is only available from the output root file.

For example, in a given run, the events with total time exceeding 500 msec fall into the overflow category. One can add the option `-q 500,99999` to check the paths that contribute to that region:

The `-q` option can also take multiple time ranges, separated by the letter `A`, to plot the average path time for different selection of events. For example:

* regionsTotalTime.jpg:

Also, by using the `-q` option, the **`Per event time`** plots (for all trigger paths) are shown as stacked histograms with events corresponding to the different total timing regions:

How do I exclude trigger paths from the timing results?

There are two ways of doing this. Method A excludes the timing of that particular path in the `hltTimingSummary` marco outpu, while method B re-runs the measurement without considering the particular path. There is a small timing discrepancy due to the intrinsic resolution of the timing measurement.

Method A - Skipping myPath from timing calculation (faster than method B)

One can use `-x` option to exclude a particular path from timing results.

For example:

```
-x myPath
```

Or, one can use `-x` to exclude a list of paths in a txt file. For example:

```
-x skipPaths.txt
```

where skipPaths.txt is:

```
myPath1
myPath2
myPath3
```

Method B - Re-running the measurement

1. Open the configuration file, e.g. offline_data.py, and comment out the unwanted path, e.g. myPath:

```
process.Path1 = cms.Path(process.HLTBeginSequenceBPTX + process.hltLevel1Activity)
process.Path2 = cms.Path(process.HLTDoLocalPixelSequence + process.hltPixelActivityFilter + proce
#process.myPath = cms.Path( process.HLTBeginSequence + process.hltDTTFUnpacker + process.hltDTAct
process.Path3 = cms.Path(process.hltDTActivityFilterTuned + process.HLTEndSequence )
...
```

1. Comment out the prescale of myPath (if any):

```
process.PrescaleService = cms.Service( "PrescaleService",
    lvl1DefaultLabel = cms.untracked.string( "0" ),
    lvl1Labels = cms.vstring( '0' ),
    prescaleTable = cms.VPSet(
      cms.PSet(  pathName = cms.string( "Path1" ),
        prescales = cms.vuint32( 100 )
      ),
      cms.PSet(  pathName = cms.string( "Path2" ),
        prescales = cms.vuint32( 100 )
      ),
#     cms.PSet(  pathName = cms.string( "myPath" ),
#       prescales = cms.vuint32( 40 )
#     ),
      cms.PSet(  pathName = cms.string( "Path3" ),
        prescales = cms.vuint32( 100 )
      ),
      ...
  )
```

1. cmsRun the configuration file to produce hte input file HLT.root (please look at the HLTCpuTiming instructions).

   ```
   cmsRun offline_data.py >&! full.log
   ```
2. Use `hltTimingSummary` macro to analyze the timing results.

How do I obtain a list of problematic events (that take too much time)?

One can use `-l`, `-a`, and `-m` options (the latter can also be combined with the `-g` option) to obtain a list of run and event numbers based on total time, path time, and module time respectively. Please see below for instructions.

Based on the total time?

For example, in a given run, the events with total time exceeding 200 msec fall into the overflow category. One can add the option `-l 200` to exctract the information for those events. The run and event numbers for the overflow events are printed in the output summary text file in a python format, which can be copied and pasted directly to do CMSSW job:

```
#The following 43 events took longer than 200 msec to run:
#(formatted for usage within the PoolSource module,i.e, Run:Event)

eventsToProcess = cms.untracked.VEventRange(
'135735:22055809',
'135735:22112642',
'135735:22131036',
...
)
```

Based on path time?

For example, in a given run, one can get the information for events where any path time exceeds 10 msec by adding the option `-a 10`. The run and event numbers for those events are printed in the output summary text file in a python format, which can be copied and pasted directly to do CMSSW job:

```
#The following 114 events took longer than 10 msec to run:
#(formatted for usage within the PoolSource module,i.e, Run:Event)

eventsToProcess = cms.untracked.VEventRange(
'142513:20170202',
'142513:20184378',
'142513:20188658',
...
)
```

Based on module time?

For example, in a given run, one can get the information for events where any module time exceeds 50 msec by adding the option `-m 50`. The run and event numbers for those events are printed in the output summary text file in a python format, which can be copied and pasted directly to do CMSSW job:

```
#The following 102 events took longer than 50 msec to run:
#(formatted for usage within the PoolSource module,i.e, Run:Event)

eventsToProcess = cms.untracked.VEventRange(
```

```
'142513:20161232',
'142513:20171568',
'142513:20172865',
...
)
```

Based on module time of specific module(s)?

One can also get the information for events where any given module(s) run longer than a given module time. For example, in a given run, one can obtain a list of events where `myModule1` and `myModule2` exceeds 50 msec by adding the option `-g myModule1,myModule2 -m 50` or `-g myModules.txt -m 50`, where `myModules.txt` is a text file:

```
myModule1
myModule2
```

The run and event numbers for those events are printed in the output summary text file in a python format, which can be copied and pasted directly to do CMSSW job:

```
#The following 53 events where myModule1 took longer than 50 msec to run:
#(formatted for usage within the PoolSource module,i.e, Run:Event)

eventsToProcess = cms.untracked.VEventRange(
'142513:20193242',
'142513:20253122',
'142513:20254693',
...
)

#The following 13 events where myModule2 took longer than 50 msec to run:
#(formatted for usage within the PoolSource module,i.e, Run:Event)

eventsToProcess = cms.untracked.VEventRange(
'142513:20212345',
'142513:20254233',
'142513:20256324',
...
)
```

Where can I find CPU-timing validation and monitoring reports?

Please take a look at this page.

-- EdgarCarrera - 31-Aug-2010

%RESPONSIBLE% EdgarCarrera

-- AlejandroGomez - 12-Sep-2011

This topic: Sandbox > AlejandroGomezSandbox_HLTCpuTimingFAQ
Topic revision: r1 - 2011-09-12 - AlejandroGomez

Based on module time? 10