# Table of Contents

Timing Studies, i.e, how many Kalmans can be squeezed into one HLT sequence ........................................................................1  
Contents ..................................................................................................................................................1  
What's this all about? ................................................................................................................................1  
How to start? - A quick fix ..................................................................................................................1  
Software description ...........................................................................................................................2  
What we measure? ................................................................................................................................3  
Current results ....................................................................................................................................3  
How to change the default configuration of the Hlt track fitter ..................................................................3  
Current version of the software and the data sample ..............................................................................5
This page explains how to study the impact of increasing the number of Kalman filter iteration, performed in Hlt2, on the execution time of the LHCb HLT trigger.

## Contents

### What's this all about?

The best starting point is to look through Agnieszka's presentations at Time Profiling Intro and here Time Profiling Plans in general, we would like to check if it is possible to make the on-line trackig perform like the off-line one. For this we are looking at the Kalman filter procedure first. The basic configuration of the HLT fitter algorithm is kept at:

```
Tr/TrackFitter/python/TrackFitter/ConfiguredFitters
```

(class `ConfiguredHltFitter`). There is a number of differences between the on- and off-line fitters, most important the number of Kalman filter iterations (1 for on- and 10 for off-line), geometry (simplified and full), measurement information (lite clusters for on- and full clusters for off-line) etc... We started our studies from looking at the impact that the increasing number of Kalman passes (from the default for the Hlt N(Kal_it) = 1 to N(Kal_it) = 10, as for the off-line configuration) has on total execution time of the Hlt. Also, we varying the number of rejected outliers - N(out) = 1,2.

The main aim is to check if this is possible (in terms of time budget) and beneficial to include more Kalman iterations in the Hlt code. This study should be treated as very important in perspective of the LHCb future plans to keep only high level objects and discard the raw data. Getting high quality tracking at the HLT is of critical importance for these plans.

### How to start? - A quick fix...

Please follow these steps and you will be all right... The tarball with all the necessary software is attached to this page - see the "Current version" section of this TWiki page. The software suite, for timing studies, contains the following:

```
Timing Studies, i.e, how many Kalmans can be squeezed into one HLT sequence...
```
Specially tailored Moore project (presently v23r2)
- Script for creating and submitting jobs to local HLT node - `run_moore_hlt.py`
- Another script to analyse the output data and make plots - `moore_time_ana_hlt.py`

Scripts are optimised for Hlt usage, but it is possible to run them on lxplus farm as well (just mind the limited resources). The first step is to untar the archive:

```bash
$> tar -zxvf timing_suite.tar.gz
```

Next, put the Moore_v23r2 folder at its usual place and recompile:

```bash
$> cp -r Moore_v23r2 $User_release_area
$> SetupProject Moore v23r2 --build-env
$> SetupProject Moore v23r2
$> cd Moore_v23r2/Hlt/Moore/cmt
$> cmt br cmt make -j X
```

Note, chose the number of cores, X, to be used for the compilation process at your discretion. Both Python scripts can be put at any location. The only prerequisite for the `run_moore_hlt.py` script is that the Moore enviroment is set properly. It will create automatically required job options and submit them to the local node. Once they are finished a second script can be used to analyse the results and create plots. An example plot that is produced by the analysis script is presented below. It represents a relative increase in execution time of the fitting procedure as a function of the number of Kalman iterations.

![Example plot](image)

### Software description

In this section we give a short description of both provided scripts.

**run_moore_hlt.py**

The script works as follow: it checks the local enviroment and configuration of the Moore installation, next a special structure of folders is created to store the results (timing as provided by the Gaudi ChronoAuditor) written out in a form of csv files. The folder structure used in our studies is as follow:

```bash
# -> first locate directories with csv files
# -> repo structure:
# timing_repo |
#   | X_outliers |
#   | timing_machineXX |
#   | summary_files_kalIT.csv
```

**How to start? - A quick fix...**
It closely resembles the data structure that is then used internally by the script to store the results for further analysis. In this way we can provide, so, called horizontal scalability that allows to add easily another parameters to our study (for instance full geometry against the simplified one). The top branch name `timing_repo` is used to locate the structure, also, it is one of line arguments for the script that performs analysis. Below it there are `X_outliers` folders containing results for jobs with different number of rejected outliers (from 1 - X). In our studies we look at rejecting either one or two largest ones. Next we have folders which names contain the machine id and finally in these folders csv files will be sortet for each number of Kalman iterations used. Note, that the names of folders and summary files encode information on how many outliers were discarded, how many machines we ran jobs on and finally how many Kalman iterations were used to fit the Hlt tracks. This structure should be treated as fixed and should not be changed since the analysis script relies on it and builds the main data strucure, to store the appropriate output data, using information encoded in names of respecitve folders and files.

In order to use the script and submit the jobs just execute at your chosen location:

```bash
$> python run_moore_hlt.py
```

`moore_time_ana_hlt.py`

Once the jobs are finished second script can be used to analyse the results. First the entire three of the `timing_repo` storage is scanned in order to create data structure to hold the output info. This is entirely dynamic process - the structure will correspond to the actual work that has been performed. So, one can run more or less jobs on a chosen number of machines and the data structure will be created automatically reflecting precisely the work that has been done for a given timing study. This feature adds flexibility to our software suite. One just needs to just guarantee that the number of machines and Kalman iterations will be the same for each value of rejected outliers. Since, we want to compare them it makes sense to assert this requirement. Stored csv files are then parsed, filtered and appropriated times of execution and other necessary parameters are fetched in order to calculate timing metrics used in our studies (see also point "What we measure?"). Results are plotted as a function of the number of Kalman filter iterations used to fit Hlt tracks. Results for different number of outliers are superimposed on the same plots.

To obtain the timing plots run it as follow:

```bash
$> python moore_time_ana_hlt.py -p path2data_repo/ -e number_of_events_used
```

**What we measure?**

(Coming soon)

**Current results**

(Coming soon)

**How to change the default configuration of the Hlt track fitter**

A private version of Moore is necessary because we want to have means of changing the ConfiguredHltFitter settings on the level of options file. At the moment it is not possible without editing Python and recompling. Thus, we decided to instrument the top Moore configurable with new two slots (fileds):

```python
... class Moore(LHCbConfigurableUser):
...
```

Software description
__slots__ = {
    "EvtMax": -1,  # Maximum number of events to process
    "SkipEvents": 0,
    "Simulation": False,  # True implies use SimCond
    "DataType": '2012',  # Data type, can be ['DC06', '2008']
    "DDDBtag": 'default',  # default as set in DDDBConf for DataType
    "CondDBtag": 'default',  # default as set in DDDBConf for DataType
    "NumberFitIterations": 1,
    "MaxNumberOutliers": 2
}

The respective values for

NumberFitIterations

and

MaxNumberOutliers

are then propagated to Hlt2Conf configurable:

from Configurables import Hlt2Conf
iter_max = self.getProp('NumberFitIterations')
out_max = self.getProp('MaxNumberOutliers')
Hlt2Conf().setProp('NumberFitIterations', iter_max)
Hlt2Conf().setProp('MaxNumberOutliers', out_max)

These are used to change the default tracking configurations as follow:

def configureReconstruction(self):
    iter_max = self.getProp('NumberFitIterations')
    out_max = self.getProp('MaxNumberOutliers')

    Hlt2BiKalmanFittedDownstreamTracking().NumberFitIterations = iter_max
    Hlt2BiKalmanFittedDownstreamTracking().MaxNumberOutliers = out_max
    Hlt2BiKalmanFittedForwardTracking().NumberFitIterations = iter_max
    Hlt2BiKalmanFittedForwardTracking().MaxNumberOutliers = out_max
    Hlt2BiKalmanFittedRichForProtonsForwardTracking().NumberFitIterations = iter_max
    Hlt2BiKalmanFittedRichForProtonsForwardTracking().MaxNumberOutliers = out_max
    Hlt2BiKalmanFittedRichForLowPTParticlesForwardTracking().NumberFitIterations = iter_max
    Hlt2BiKalmanFittedRichForLowPTParticlesForwardTracking().MaxNumberOutliers = out_max

Finally, new values for the number of Kalman iterations performed in the Hlt trigger and rejected outliers are used in the Hlt2Tracking configurable to updated the fitter algorithm settings:

... from TrackFitter.ConfiguredFitters import ConfiguredHltFitter
fitter = ConfiguredHltFitter( getattr(Hlt2StagedFastFit, 'Fitter')

How to change the default configuration of the Hlt trackfitter
Having provided such mechanism a script can be used for creating options files and submitting jobs for chosen range of configuration parameters.

**Current version of the software and the data sample**

The current version of the timing software suite is attached below as a tarball - `timing_suite.tar.gz`. It contains the latest release of the Moore_v23r2. This will be updated constantly as needed after each new release. Enjoy using it and leave feedback.

The data sample that was used in these studies is Bd->KstarMuMu (11114001) produced according to `Beam6500GeV-RunII-MagDown-Nu1.9-25ns-Pythia8` settings. Simulation stepping was `Sim08e`. Data type is `XDIGI`.

-- TomaszSzumlak - 2014-11-14