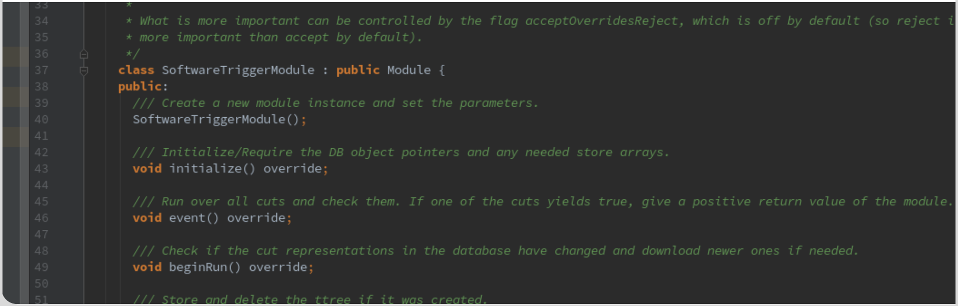
Software Trigger Module (and Cut Framework)

Trigger Meeting.

Nils Braun, Thomas Hauth | September 7, 2016

IEKP - KIT



```
 *
 * What is more important can be controlled by the flag acceptOverridesReject, which is off by default (so reject i
 * more important than accept by default).
 */
class SoftwareTriggerModule : public Module {
public:
  /// Create a new module instance and set the parameters.
  SoftwareTriggerModule();

  /// Initialize/Require the DB object pointers and any needed store arrays.
  void initialize() override;

  /// Run over all cuts and check them. If one of the cuts yields true, give a positive return value of the module.
  void event() override;

  /// Check if the cut representations in the database have changed and download newer ones if needed.
  void beginRun() override;

  /// Store and delete the ttree if it was created.
```

Introduction

The Software Trigger Module

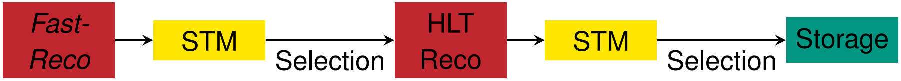
The software trigger module is a general module to perform cut decisions needed in the HLT software.

- Calculates predefined variables and applies predefined, downloaded cuts (from the database) on the event.
- Each event is classified as rejected or accepted.

Major features of the Software Trigger Module

- Centralized framework for all cuts and decisions applied for the HLT software stack
- Database up- and download of the cut settings and tag names
- Versioning of all cuts through the condition database
- Easy to extend calculation framework for variables needed in the cuts
- Cuts based on the well-tested `GeneralCut` from the framework
- Easy to use python interface for quick cut development
- Nearly 100% unittest coverage

Usage in the HLT path



- Different Cuts on different positions in the path
- Different variables to calculate - but using the same framework

Usage as a Cut Module

A Software Trigger Cut (the main entity used in the Software Trigger Module) is defined by five properties:

The cut condition A string in a format known by the `GeneralCut` with variables defined by the calculator you choose (see below). *Example:* `[visible_energy < 1.8438] and [highest_3_ecl <= 0.3999]`

The cut type A reject or accept cut. See the next slide for the difference.

The prescale An accept cut can be statistically scaled down with this factor (will only lead to a positive result in one of *N* cases).

Usage as a Cut Module

A Software Trigger Cut (the main entity used in the Software Trigger Module) is defined by five properties:

The base identifier This string defines, which variables are calculated for the event content. Also, you can only choose between cuts with the same base identifier in one run of the software trigger module.
Example: `fast_reco` or `hlt`

The cut identifier This is the identifier making the cut downloadable from the database. Two cuts with the same identifier but different base identifiers are stored separately.
Example: `reject_bkg` or `accept_ee`

The three possible results

Each cut can either be an accept or a reject cut. The overall output of the module depends on this. There are different possibilities:

- `acceptOverridesReject` is set to False (default)
  - **One of the chosen reject cuts is true:** the whole event is tagged with ”rejected” and the module gives -1 as a return value
  - **One of the chosen accept cuts is true and no reject cut is true:** the whole event is tagged as ”accepted” and the module gives +1 as a return value
  - **No cut gives a true result:** the whole event is tagged as ”don’t know” and the module gives 0 as a return value
- `acceptOverridesReject` is set to True
  - **One of the chosen accept cuts is true:** the whole event is tagged with ”accepted” and the module gives +1 as a return value
  - **One of the chosen reject cuts is true and no accept cut is true:** the whole event is tagged as ”rejected” and the module gives -1 as a return value
  - **No cut gives a true result:** the whole event is tagged as ”don’t know” and the module gives 0 as a return value

A cut gives a true result if its cut condition is true and, in cases of accept cuts, the pre scale (drawn from a uniform distribution) also leads to a true result.

Choosing the correct database

It is very important to use the correct database tags for downloading the cuts. We will first use the local database, then use the central database with the software trigger cuts and in the end use the production database for the other entries.

```
import basf2

basf2.reset_database()
basf2.use_database_chain()
basf2.use_local_database()
basf2.use_central_database("software_trigger_test")
basf2.use_central_database("production")
```

Using the module

Make sure to include the database code snippet also in your steering file!

```
import basf2
path = basf2.create_path()
# Import the simulated events (or something analogous)
path.add_module("RootInput")
# Reconstruct the information
# needed for the fast reco decision
add_reconstruction(path, trigger_mode="fast_reco")
# Add the cut module to the path
cut_module = path.add_module("SoftwareTrigger",
  baseIdentifier="fast_reco",
  cutIdentifiers=["reject_ee", "accept_ee", "reject_bkg"])
# Create three new paths (filling is not shown)
events_accepted_path = basf2.create_path()
events_rejected_path = basf2.create_path()
events_dont_know_path = basf2.create_path()
# Do something with the result of the module
cut_module.if_value("==-1", events_rejected_path)
cut_module.if_value("==1", events_accepted_path)
path.add_path(events_dont_know_path)
```

From where to get the result

There are two possibilities to get the result of a Software Trigger Module event:

1. The return value of the module is set to -1, 0 or +1, depending on the results of the cuts.
2. The module writes the results of the individual cuts as well as the total result to a `StoreObj` with the type `SoftwareTriggerResult`.
   - If you have more than one Software Trigger Module in your path, the different cut results are all added to the result object.
   - You can get the results of the single cuts with the `getResult` function.
   - The total result can be calculated again using the `getTotalResult` function, which needs the value of `acceptOverridesReject` as an input.

The `SoftwareTriggerResult` may replace the `HLTTag` as an MDST object, as it does not contain hard-coded cut tags.

Debug Output

The Software Trigger Module can output all calculated variables for each event into a ROOT TNTuple file

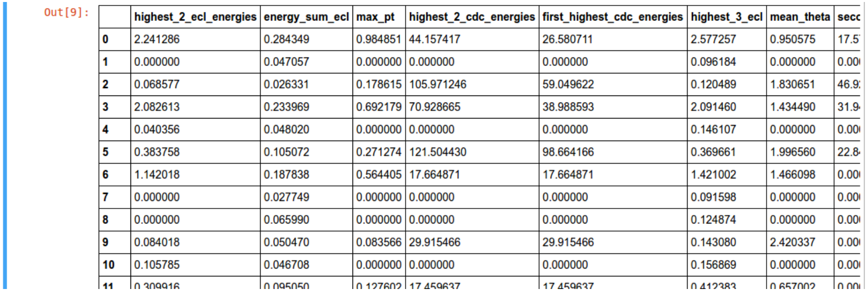
```
import basf2
path = basf2.create_path()

# Add the SoftwareTrigger
path.add_module("SoftwareTrigger",
                baseIdentifier="fast_reco",
                cut_identifiers=[],
                storeDebugOutput=True,
                debugOutputFileName="variables.root")
```

It is important to choose the correct base indentifier, because this defines which variables are calculated and stored!

Debug Output - continue

After processing the path, a new ROOT file ”variables.root” are created with a single TTree containing the calculated variables with as many rows as there were events in the process.

```
from root_pandas import read_root
df = read_root("variables.root")
```



Out[9]:

	highest_2_ecl_energies	energy_sum_ecl	max_pt	highest_2_cdc_energies	first_highest_cdc_energies	highest_3_ecl	mean_theta	secc
0	2.241286	0.284349	0.984851	44.157417	26.580711	2.577257	0.950575	17.5
1	0.000000	0.047057	0.000000	0.000000	0.000000	0.096184	0.000000	0.00
2	0.068577	0.026331	0.178615	105.971246	59.049622	0.120489	1.830651	46.9
3	2.082613	0.233969	0.692179	70.928665	38.988593	2.091460	1.434490	31.9
4	0.040356	0.048020	0.000000	0.000000	0.000000	0.146107	0.000000	0.00
5	0.383758	0.105072	0.271274	121.504430	98.664166	0.369661	1.996560	22.8
6	1.142018	0.187838	0.564405	17.664871	17.664871	1.421002	1.466098	0.00
7	0.000000	0.027749	0.000000	0.000000	0.000000	0.091598	0.000000	0.00
8	0.000000	0.065990	0.000000	0.000000	0.000000	0.124874	0.000000	0.00
9	0.084018	0.050470	0.083566	29.915466	29.915466	0.143080	2.420337	0.00
10	0.105785	0.046708	0.000000	0.000000	0.000000	0.156869	0.000000	0.00
11	0.309916	0.095050	0.127602	17.459637	17.459637	0.412383	0.657002	0.00

Cut Creation and Uploading

Create a reject cut (to reject background events after *FastReco*) and upload it to the local database:

```
from ROOT import Belle2
from Belle2.SoftwareTrigger import SoftwareTriggerCut
from softwaretrigger import db_access

bkg_cut_st = SoftwareTriggerCut.compile(
  "[[visible_energy < 1.8438] and
    [highest_3_ecl <= 0.3999] and [max_pt <= 0.3152]]",
  1, True)
db_access.upload_cut_to_db(bkg_cut_st,
                           "fast_reco", "reject_bkg")
```

After that, the cut is stored in the same folder as you called the python code from in a folder called ”localdb”. If you run your steering file accessing this cut in this folder, you can use this new cut.

Cut Creation and Uploading - continue

If you are happy with the cut, you can go and upload it to the central database. In the moment, I do not push to the production cut but rather have created my own global tag (`software_trigger_test`). If you also want to push to this tag, call

```
upload_localdatabase --tag software_trigger_test  \
                     localdb/database.txt \
                     --final-exp 0
```

Of course, you are free to create your own global tag and push to this (or push to another already present global tag). You just have to remember to change to this cut also in your other steering files.

Where to find more information

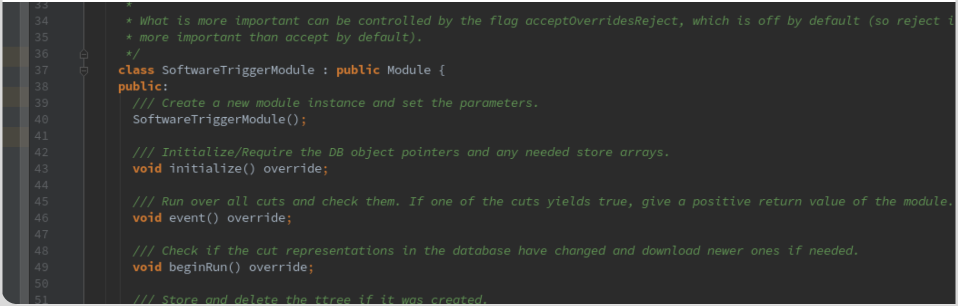
- All the code related to the Software Trigger can be found in `skim/softwaretrigger`.
- An example file of the usage is currently under construction in `skim/softwaretrigger/scripts/softwaretrigger/__init__.py`.
- The shown information plus some more can be found at confluence `https://confluence.desy.de/display/BI/The+Software+Trigger+Module`.
- If you are interested in updates, please follow the JIRA ticket BII-1622 (`https://agira.desy.de/browse/BII-1622`).
- In case of questions, please write a mail to `nils.braun@kit.edu`.

Runtime Analysis and *FastReco* on real HLT Hardware

Trigger Meeting.

Nils Braun, Thomas Hauth | September 7, 2016

IEKP - KIT

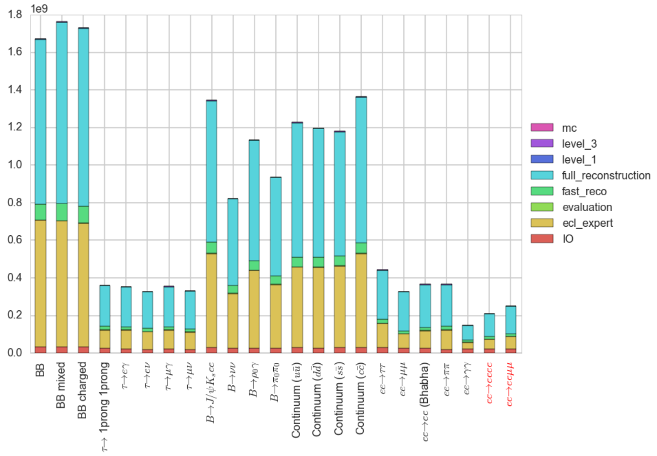


```
 *
 * What is more important can be controlled by the flag acceptOverridesReject, which is off by default (so reject i
 * more important than accept by default).
 */
class SoftwareTriggerModule : public Module {
public:
  /// Create a new module instance and set the parameters.
  SoftwareTriggerModule();

  /// Initialize/Require the DB object pointers and any needed store arrays.
  void initialize() override;

  /// Run over all cuts and check them. If one of the cuts yields true, give a positive return value of the module.
  void event() override;

  /// Check if the cut representations in the database have changed and download newer ones if needed.
  void beginRun() override;

  /// Store and delete the ttree if it was created.
```

Experimental Setup

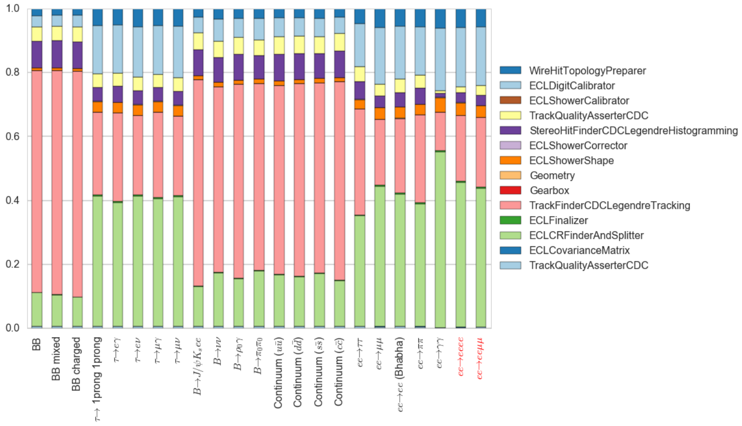
- Measurements on the Runtime of the full reconstruction was performed on the HLT worker nodes.
- In this talk: only single processing will be shown, multiprocessing is under analysis in the moment.
- All events were processed - cuts on the events were performed in retrospect.

Average Event Processing Time (in ns)



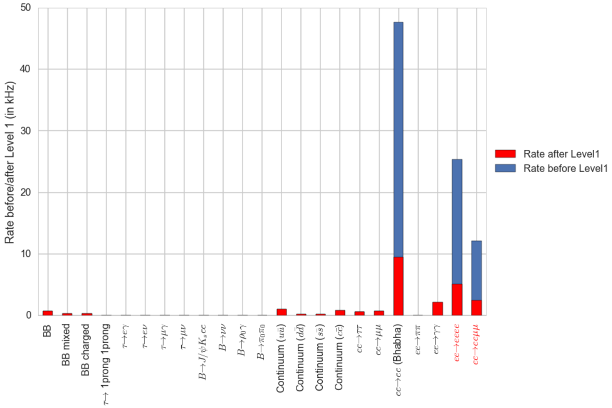
- Processing time depends heavily on the channel.
- ECLExpert will not be taken into account in the following (under construction).

Average Ratio of different *FastReco* Modules



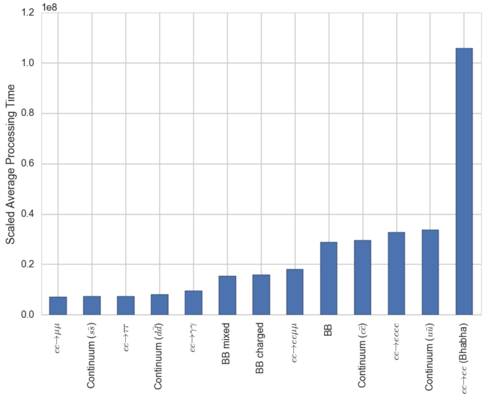
Optimization is currently done in the Legendre Track Finder and the Quality Asserter.

Assumed Cross Section after Level 1



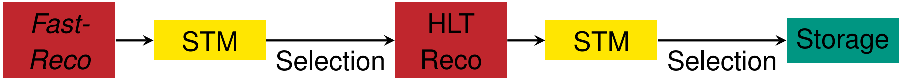
- For calculating the average event time, a ratio of the different channels has to be calculated.
- The numbers are taken from '*Overview of the Belle II Physics Generators*' by P. Urquijo and T. Ferber.
- It is assumed, that the Level 1 has a background (and ee) rejection of 80 %.

Processing Time Scaled with Cross Section

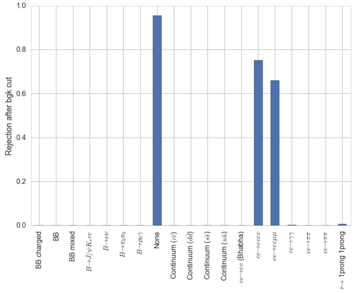


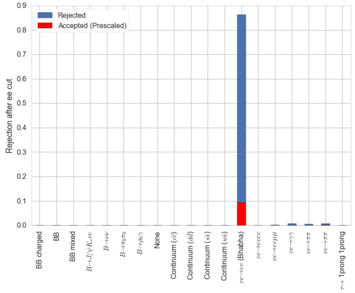
- With these ration, the average processing time (without cuts) for one average event is: **0.30 s**
- Assuming 6400 cores (without degregation because of hypertreading, IO, etc.) with 20 kHz (30 kHz), the limit is **0.32 s** (0.21 s).

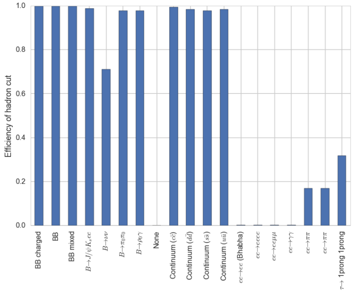
Usage in the HLT path



Cuts applied after *FastReco*



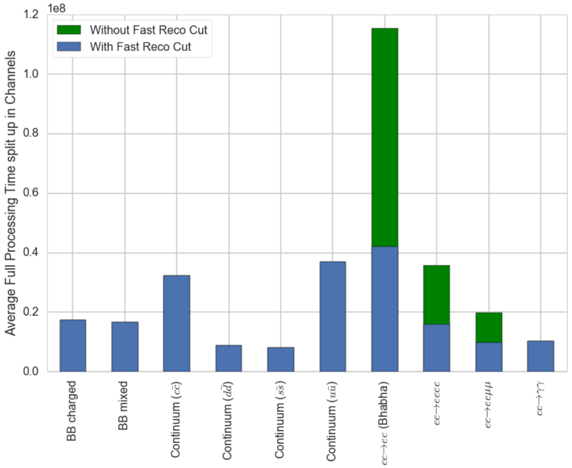




The shown cuts are:

- Background Cut
- EE Cut
- Hadron Cut

Average Processing Time After *FastReco* Cuts



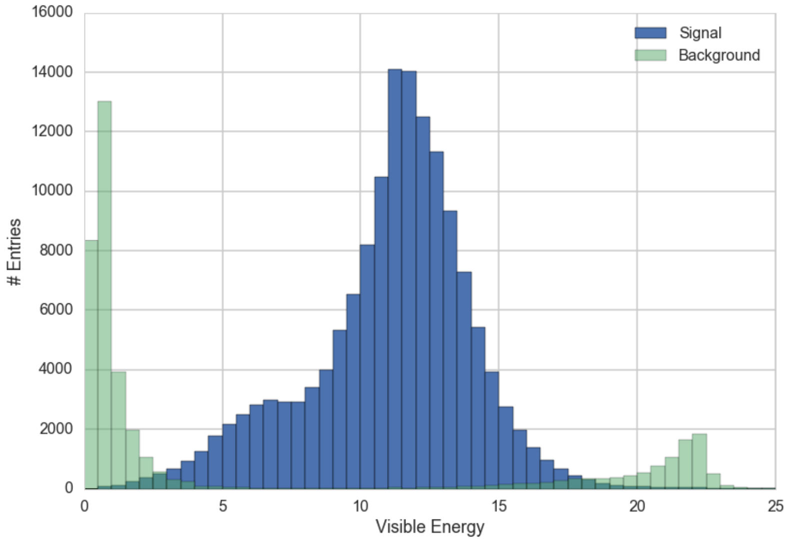
With these cuts, the average processing time is **0.198 s** (was 0.30 s).

Summary

- Software Trigger Framework was introduced in basf2; some further developments under way.
- Single-Core measurements are promising - *FastReco* works quite well.
- Multi-Core measurements are preformed in the moment - network streaming measurements are under construction.

Backup

Example for one variable: visible energy



Open Issues

- Specialized prescale for Bhabha-Events according to event variables.
- Variable Calculator for final HLT cuts not finished and committed yet.