The High Level Trigger software for the CMS experiment

O. van der Aa*, C.Delaere†

Abstract

The observation of new physics will be a challenging task for the CMS experiment at the LHC, in particular for its High Level trigger (HLT). A prototype of the High Level Trigger software to be used in the filter farm of the CMS experiment and for the filtering of Monte Carlo samples will be presented. The implemented prototype heavily uses recursive processing of a HLT tree and allows dynamic trigger definition. The general architecture and design choices as well as the timing performance of the system are reviewed in the light of the DAQ constrains.

CMS DAQ AND TRIGGER DESIGN

The Compact Muon Solenoid (CMS) [1] is one of the two multi-purposes experiments at the future Large Hadron Collider. The CMS data acquisition system (DAQ) [2] is particular in the sense that it has been designed to minimize the use of custom technologies. The collaboration therefore benefits from the rapid growth of standard network and computing technologies. This choice implies that there is no special resources allocated in the design of a so called Level-2 trigger. The whole event selection is performed in two stages, respectively called *Level-1* and *High Level Trigger* (HLT). The by-product of this decision is that the selection strategies will only be defined in the HLT software. It will certainly give more flexibility for the evolution of the selection strategies that physicist will require when developing new analysis.

Another specificity of the CMS DAQ is it's modularity. It is expected that the LHC instantaneous luminosity will build up with time. Therefore it is not necessary to have the full bandwidth at the first day of beam crossing. The DAQ design allows to add bandwidth by chunks of $12.5\mathrm{kHz}$ as needed as the luminosity build up as well as the storage capacity. This modularity is reflected in the HLT cluster, build with a large amount of commercial computers.

The High Level Trigger is involved each $10\ \mu\mathrm{s}$, as soon as a Level-1 trigger accept is issued by the Level-1 Global Trigger Processor. The task of the HLT is to further reduce the data rate by a factor ~ 1000 in or order to fulfill the requirement of an output rate below $\sim 100\mathrm{Hz}$. This means that the HLT decision has to be taken within $10\ \mathrm{ms}$ while keeping efficiency as high as possible for the known physics channels. In order to achieve this data reduction, a massive computing power will be required. If one assumes a mean computing time of $\mathcal{O}(10^{-2})\ s$ for each Level-1 acceptm with a data input rate of $\mathcal{O}(10^5)$ Hz, it means that the computing cluster that will host the HLT system will be constituted of about 1000 CPUs.

The role of the CMS DAQ is to provide the events to the computing elements of the HLT system. With an estimated event size of $1\mathrm{MB}$ per event at the Level-1 output rate the DAQ system will require a total bandwidth of $100\mathrm{GB/s}$.

HLT STEERING SOFTWARE

The HLT steering software [3] has the role of applying selection criteria on reconstructed quantities. For that purpose, it uses reconstruction software [4] dedicated to each subdetector and has to bring individual responses together to build the global accept/reject . It can be seen as a logical equation involving the evaluation of several quantities that are reconstructed from the detector response. The equation can be translated into a tree where each node or element is either an operand or an logical operator. The tree representation allows to recursively evaluate the trigger.

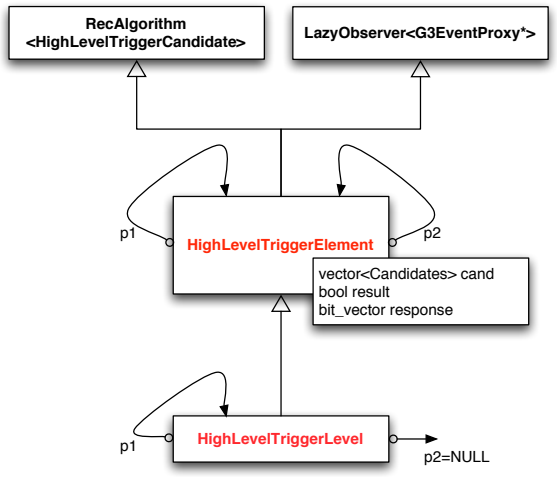


Figure 1: UML diagram of the HLT basic building blocks

The basic building blocks of the HLT are shown in Fig 1. The HighLevelTriggerElement represents each element (node) of the HLT tree. An element of a binary tree is by definition connected to two other elements that we will call "daughters". It outputs three objects

1. `bool decision`, stating the result of the selection performed in the element,

* Université Catholique de Louvain, olivier.van.der.aa@cern.ch
† Université Catholique de Louvain, christophe.delaere@cern.ch

2. `vector<bool> response`, used to give a detailed view of how the event was selected by the element,

3. `vector<Candidate> cand`, list of particles (Lorentz vector, vertex) of the particles that passes the selection.

The HighLevelTriggerLevel is a specialisation of the HighLevelTriggerElement having only one daughter. In general the selection of a particle proceeds in several steps being implementations of the HighLevelTriggerElement.

The Element observes events as they are dispatched by the CMS reconstruction framework and reconstruct a list of trigger candidates.

In general the HLT trigger table consists of several sub-triggers that are combined with a logical "OR". The owner of the root of the HLT is designed to hold several roots. When the HLT response is requested, the response of each sub-trigger is queried and proceed recursively up to the tree leaves. The evaluation of a tree branch stops as soon as an element fails in the chain avoiding unnecessary computing.

To illustrate the evaluation sequence assume a trigger tree as shown on Fig. 2, which is equivalent to the logical equation:

$$E_t(e) > 23 \vee (E_t(e) > 14 \wedge E_t(\tau) > 48).$$

The selection of particles proceeds in levels, here for the sake of the example the selection of electrons and taus proceed in two steps implemented in HighLevelTriggerLevel derived classes. The logical operators are HighLevelTriggerElement connected to their two daughters.

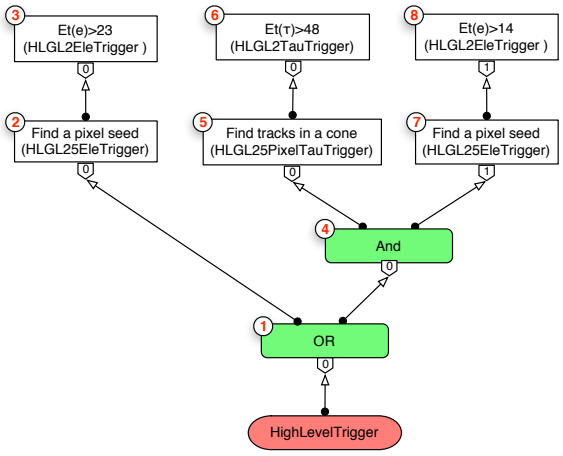


Figure 2: Example of a HLT tree. Names in parenthesis are class names.

When the HighLevelTrigger is queried for it's response the request is recursively propagated through the tree in the following sequence:

1. The request will go up to the element 3 , via the "or" node and element 2.

2. Element 3 will be evaluated and return false.

3. Since the element 3 is false, the attached element 2 (HLGL25EleTrigger) will not be evaluated and will return false.

4. The request will then go up to element 6, via the "or" node, the "And" element and the element 5

5. Element 6 It will be evaluated and return false.

6. Element 5 (HLGL25PixelTauTrigger) must now be evaluated, but since the previous level returned false it will be skipped and return false.

7. The request will then go up to element 8, via the "or" node and the "And" element and the attached element 7.

8. Element 8 It will be evaluated and return true as well as element 7.

9. The "And" element will be evaluated and return false.

10. The "or" element (the root node) will be evaluated and return false.

At the end of the evaluation, each Element response is concatenated to form a list of bits detailing the HLT decision.

EVALUATION SEQUENCE OPTIMIZATION

The evaluation sequence can be optimized in order to minimize the mean computing time required to select events online. It is clear that not all Elements have to be evaluated if one of them gives a positive response.

In the case of a "or" node, if the first daughter gives a positive response then it is not necessary to evaluate the second daughter. One has then to find the best order of the two daughter in order to minimize the mean computing time to accept an event. In the case of a "or" node, the mean accept time is given by:

$$\langle Ta_{12} \rangle = p_1 ta_1 + (1 - p_1) p_2 (tr_1 + ta_2), \qquad (1)$$

where $ta_{(1,2)}$ is the mean accept time for each element, $tr_{(1,2)}$ is the mean reject time and $p_{(1,2)}$ is the probability for each element to be positively evaluated. If one reverse the order of the two Elements, then indices in Eq. 1 just have to be permuted. Although the mean accept time Ta_{12} can be optimized, the reject time is fixed since both elements have to be evaluated to be sure that the "or" element gives a negative response. In the case of a "and" Element, the situation is reversed, one can find an ordering that minimize the mean reject time but not the mean accept time.

In the previous section, it has been described that the HLT can hold several sub-triggers that are specialized for different type of selection. The HLT has to sequentially evaluate each sub trigger. Evaluation stops either if a positive response is computed or if all sub-triggers have given a negative response, in which case the event is rejected.

The optimization of such a configuration is a generalization of the "or" element in which they are n daughter attached to the only High Level Trigger object. The mean time for such a configuration to be evaluated is $\langle T \rangle = \langle Ta \rangle + \langle Tr \rangle$, where $\langle Ta \rangle$ is the mean accept time and $\langle Tr \rangle$ is the mean reject time. Let $k_1 \cdots k_n$ be the order of the sub-trigger, with $k_i = \{1 \cdots n\}$ and $k_i \neq k_j \; \forall \; i \neq j$. The mean time for a reject is given by:

$$\langle Tr \rangle_{k_1 \cdots k_n} = \prod_{i=1}^{n} (1 - p_{k_i}) \sum_{i=1}^{n} tr_{k_i}, \qquad (2)$$

which is invariant under any permutation of the elements. This is simply seen since all sub-trigger have to be evaluated to reject the event. The mean accept time $\langle Ta \rangle_{k_1 \cdots k_n}$ is given by:

$$\langle Tr \rangle_{k_1 \cdots k_n} = \\ p_{k_1} ta_{k_1} + \sum_{i=2}^{n} \left[p_{k_i} \prod_{j=1}^{i-1} (1 - p_{k_j}) \right] \left[ta_{k_i} + \sum_{l=1}^{i-1} tr_{k_l} \right]. \qquad (3)$$

It is clear that all possible arrangements cannot be evaluated since it grows as $n!$ with the trigger size. It can be shown that the arrangement that minimizes expression of Eq. 3 is found if the order between two elements is defined as for the simple "or" case, i.e.:

$$||kl|| < ||lk|| \equiv \\ pa_k t_k + (1 - pa_k) pa_l (tr_k + ta_l) < \\ pa_l t_l + (1 - pa_l) pa_k (tr_l + ta_k). \qquad (4)$$

The gain in ordering sub-triggers in such a sequence is illustrated in the following, where one assumes to have four sub-triggers that have high triggering probabilities. Let choose four triggers with $(p_i, ta, tr) = \{(0.5, 60, 12), (0.3, 60, 50), (0.1, 20, 10), (0.6, 90, 40)\}$, the time units are arbitrary. On Fig. 3 one can see that the sub-trigger order has a significant impact on the mean computing time. In this particular case, the total time increase is 50% when going from the (1342) permutation to the (2431) one.

It is important to notice that the total computing time is the mean accept time in addition to the mean reject time. Since the HLT has to reject most of the events accepted at Level-1, a small fraction of the time will be used in accepting events. The above probabilities will be 10^{-3} such that $\langle Tr \rangle$ will dominate and cannot be optimized. Still the tails in the timing distribution will be reduced if one uses the specified method to reduce the mean accept time.

TRIGGER PARAMETERS DEFINITION

The computation of the trigger response has to be handled coherently when a change in the selection criteria occurs. It should also be possible to store the trigger settings

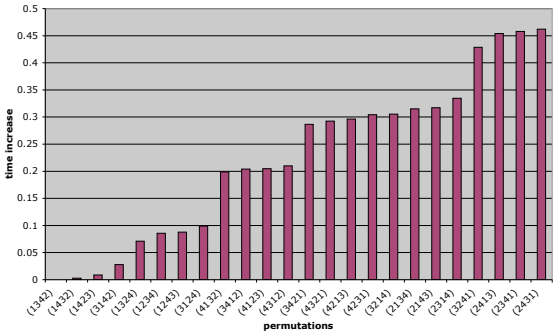
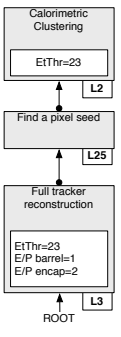


Figure 3: Average accept time for each permutation of the triggers elements.

with the trigger outcome. A mechanism is provided by the CARF framework [5] to address these issues. Each algorithm, which in the HLT is an element, should announce what are the parameters and default values defining it's behaviour. When the Element is queried to compute it's response, a list of the parameters and their values is provided and the element computation is fired. The result of the Element and it's corresponding query can be stored for future analysis. This mechanism allows to cache an algorithm result and return the cached value if the same query is issued twice for the same event.

The definition of the HLT tree takes the form of a series of query's (RecQuery) constructed in by a HighLevelTrigger derived classes which follows a factory design pattern [6]. An example of such a definition is given in Fig. 4. In the implementation of the HLT setup() method, the queries and parameters associated with each element is defined. The RecQuery name defines which element will be build and the setParameter method defines the parameters name and values.




```
virtual  void HighLevelTriggerTest::setup() {
    // L2 Element
    RecQuery l2("HLGL2EleTrigger");
      l2.setParameter("EtThr",23);
    // L25 connected to the L2
    RecQuery l25("HLGL25EleTrigger");
      l25.setComponent("mother1",l2);
    // L3 connected with the L25
    RecQuery l3("HLGL3EleTrigger");
      l3.setParameters("EtThr",23);
      l3.setParameter("EovP_barrel",1);
      l3.setParameter("EovP_endcap",2);
      l3.setComponent("mother1",l3);
// finally add the tree to the list of root's
   addRootTriggerElement(l3);
};
```

Figure 4: Example of the specification for a single electron trigger with three levels. The required code to construct the trigger tree is shown on the right

The factory based design is particularly useful to support the construction of the HLT tree from a condition database. The example shown on Fig. 4 defines the tree statically. To dynamically define the tree, one has implemented a factory

that reads it's configuration from an XML [7] file specifying the elements, their parameters and how they are connected. Fig 5 detail the XML content giving an equivalent trigger tree as defined on Fig. 4. As in the static example, each XML tag define the algorithm to be used and it's associated parameters.

```
<?xml version="1.0" encoding="Latin-1" standalone="no"?>
<!DOCTYPE GlobalTrigger SYSTEM "./HighLevelTrigger.dtd">
<GlobalTrigger>
 <L3EleTrigger>
   <L25EleTrigger>
     <L2EleTrigger>
       <Parameter value="23" name="EtThr"/>
     </L2EleTrigger>
   </L25EleTrigger>
   <Parameter value="23" name="EtThr"/>
   <Parameter value="1" name="EovP_barrel"/>
   <Parameter value="2" name="EovP_endcap"/>
 </L3EleTrigger>
</GlobalTrigger>
```

Figure 5: XML specification of the trigger tree for the selection of an electron.

CONCLUSIONS

The CMS experiment has defined reconstruction algorithms [2] for the selection of electrons, muons, taus, missing E_t, jets which are implemented in the ORCA software. The HLT steering software, also part of ORCA, eases the combination of the selection algorithms to define the HLT selection logic. The selection logic is represented in a tree and the definition of the tree can be done via an XML file. Optimisation of the tree evaluation sequence as been pursued and show significant improvement in the mean accept time for events selected with high efficiency.

ACKNOWLEDGMENTS

C.D. would like to thank the Belgian FNRS for supporting him as research fellow. Thanks to our supervisor V. Lemaître for supporting this work. We would like to thank E. Burton for her help in the design of the optimisation algorithm.

REFERENCES

[1] The Compact Muon Solenoid – Technical Proposal, CERN/LHCC **94**-38

[2] CMS Collaboration, *"The Trigger and Data Acquisition project, Volume II.Data Acquisition & High-Level Trigger. Technical Design Report"*, CERN/LHCC **2002**-26

[3] HLT web page, http://www.fynu.ucl.ac.be/he/hlt/

[4] ORCA web page, http://cmsdoc.cern.ch/orca/

[5] CMS Collaboration, V. Innocente, *CMS reconstruction and analysis: an object oriented approach*, Elsevier, Computer Physics Communications **110** (1998) 192-197

[6] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns*, Addison-Wesley Professional; 1st edition (January 15, **1995**)

[7] XML specification, http://www.w3.org/TR/REC-xml/