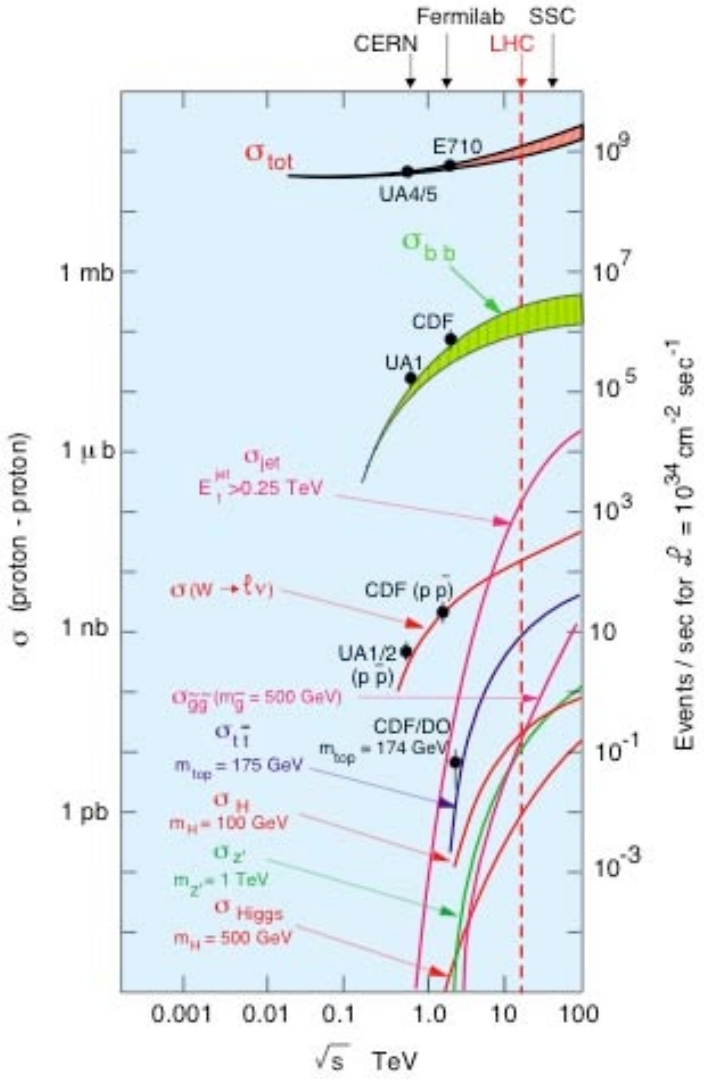
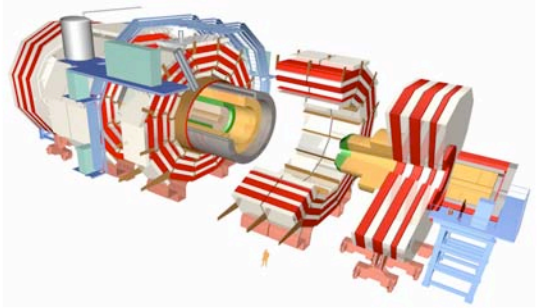
High Level Trigger of CMS

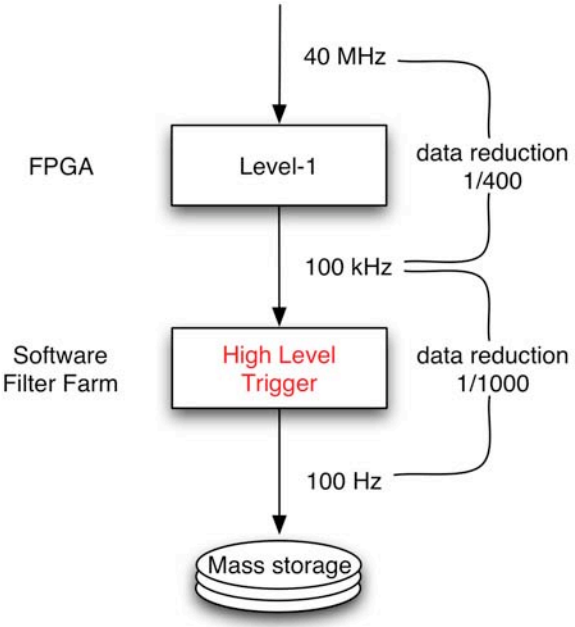
- HLT requirements
- General Architecture
- Evaluation sequence
- HLT timing
- Configuration of HLT
- Building the HLT Tree
- Existing implementations
- Conclusion

Olivier van der Aa, Co Author: C. Delaere.
Intitut de Physique Nucléaire
Université Catholique de Louvain

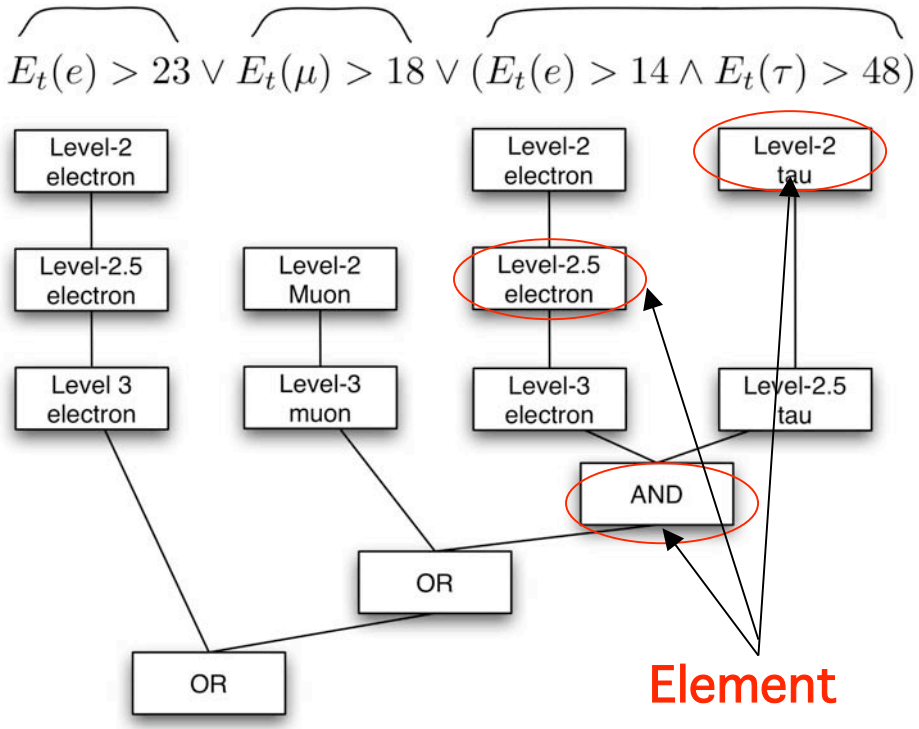
HLT data reduction requirements







HLT principles



- HLT trigger table is equivalent to a logical equation involving tests on reconstructed quantities (particle Pt, angle between particles etc...)
- The logical equation can be represented as a tree where each **Element** is a either a logical operator or an operand.

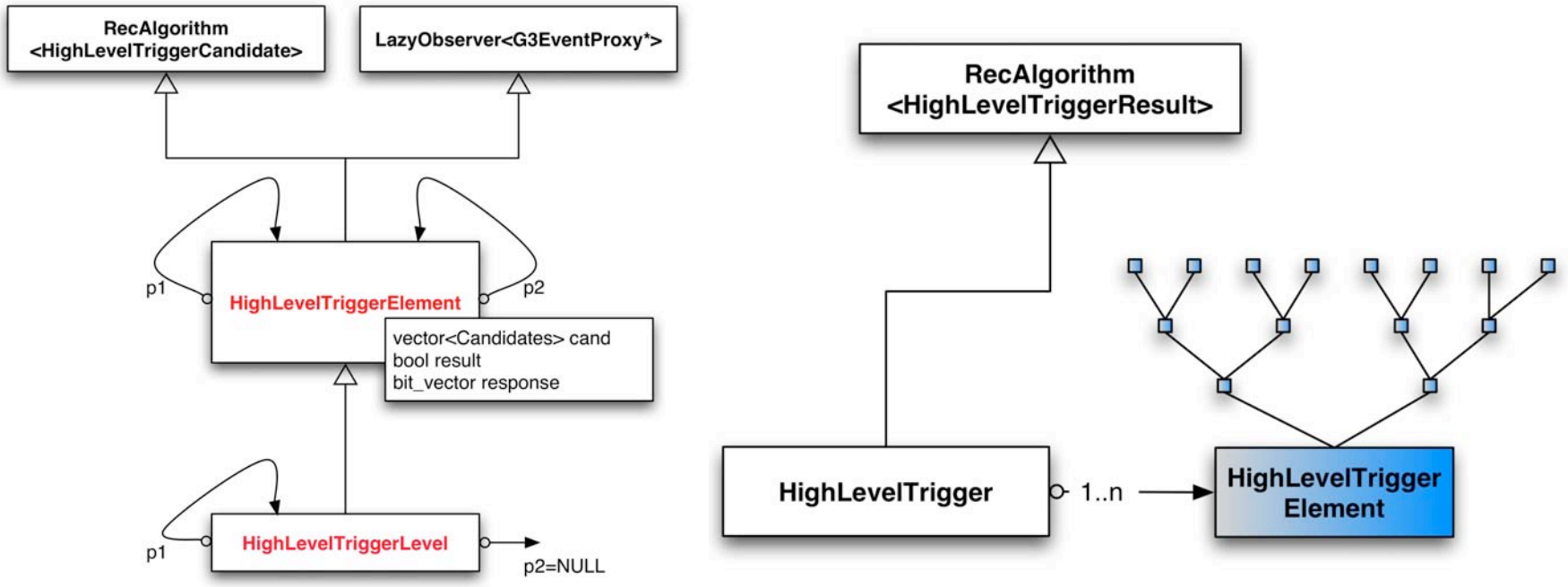
- Generally, the selection of one particle involves several steps called Levels.

⇒ Example: Level-2 electron find a calorimetric cluster above threshold. Level-2.5 verifies that there is a corresponding hit in the pixel detector. Level-3 check that the track passes an H/E threshold to reject pions.

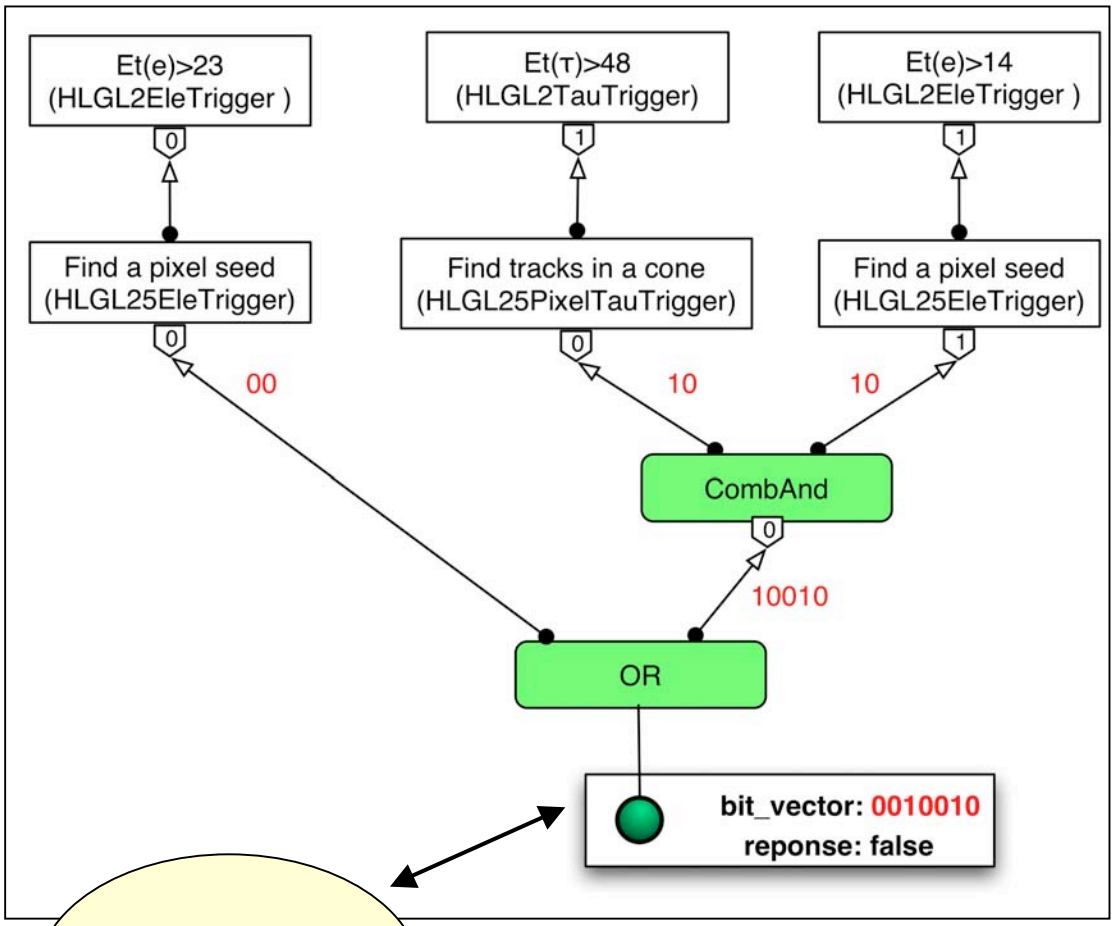
HLT output and building blocks

1. Each Tree Element has three output
   - Bool: specify the node outcome
   - Bit_vector: detail the selection state.
   - vector of TriggerCandidates: list of particles that passes the selection.

2. HLT steering building blocks.



HLT evaluation sequence

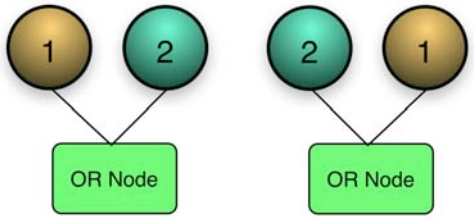


- Trigger response request is propagated from the roots to the leaves.
- The response is computed going from top to bottom and cached at each Element.
- Each Element produces a list of candidates that have passed the selection criteria.

Optimisation of the evaluation sequence

- The evaluation sequence can be ordered to minimise the computation time.

In the case of a OR, first evaluate 1 or 2 ?

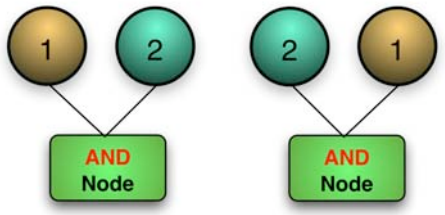


- Depends on the trigger probabilities **p**, the mean time to accept (**ta**) and to reject (**tr**) an event.
- Find the ordering that minimise the mean time to accept an event

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

$$\langle Ta_{21} \rangle = p_2 ta_2 + (1 - p_2) p_1 (tr_2 + ta_1)$$

In the case of a AND, first evaluate 1 or 2 ?



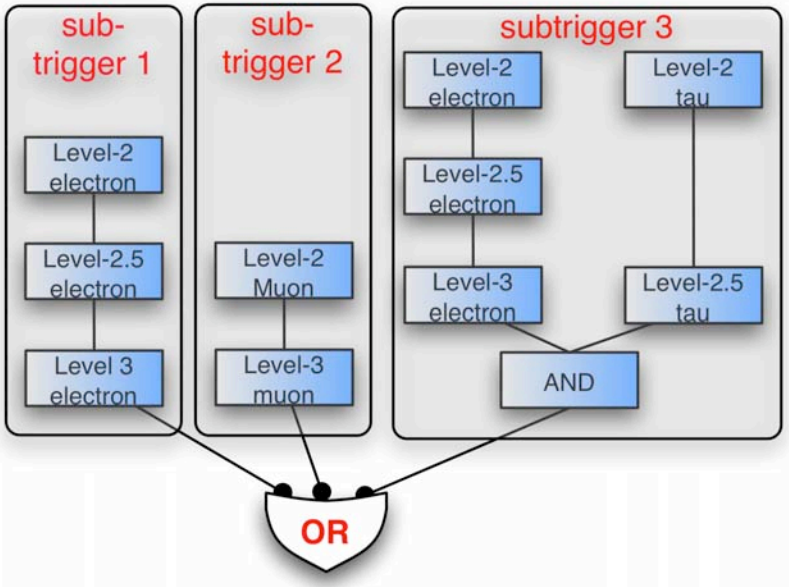
- Find the ordering that minimise the mean time to reject an event

$$\langle Tr_{12} \rangle = (1 - p_1) tr_1 + p_1 (1 - p_2)(ta_1 + tr_2)$$

$$\langle Tr_{21} \rangle = (1 - p_2) tr_2 + p_2 (1 - p_1)(ta_2 + tr_1)$$

Optimisation of the evaluation sequence

- The complete trigger is a OR of several sub-triggers.
- How to order the sub-triggers to minimise the mean time to accept an event ?



- Reject time is fixed since all sub-triggers have to be evaluated to state on the rejection of an event
- One can order the sub-triggers to optimise the mean accept time **<Ta>**

Find an order {k1…kn} of the sub trigger for which,

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

Is minimal

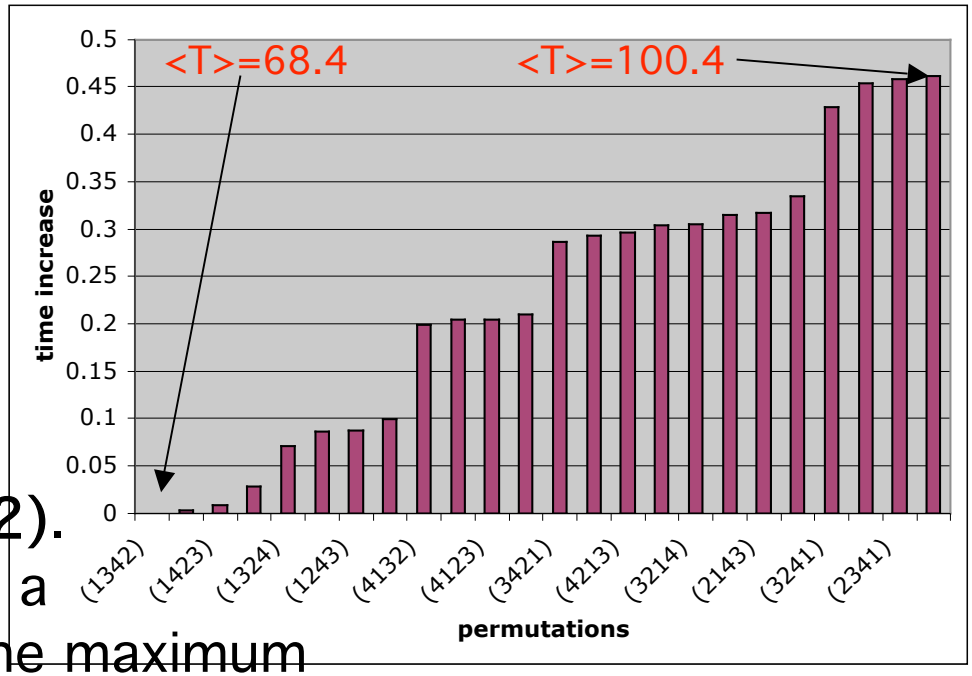
Optimisation of the evaluation sequence

- Evaluation of all possible combination grows as **n!**
- Define an order between two sub trigger as:

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

- Sort the sequence $\{k_1 \ldots k_n\}$ according to the defined order.

bit	p_i	t_a	t_r
1	0.5	60	12
2	0.3	60	50
3	0.1	20	10
4	0.6	90	40



- Best combination is **(1342)**.

Other combination result in a time increase of 50% for the maximum

Optimisation of the evaluation sequence

- Optimisation at each Trigger node has been implemented
- A procedure to optimise the multi-or that holds the sub-triggers can be enabled.

- The mean accept time optimisation can reduce the computation time for signal events for which p_i>>1E-3.
- The mean rejection time dominates the total mean time to process events since in most of the cases events events are rejected to reduce the rate by a factor 1000.

$$\langle T \rangle = \langle Ta \rangle_{\{k_1 \cdots k_n\}} + \langle Tr \rangle$$

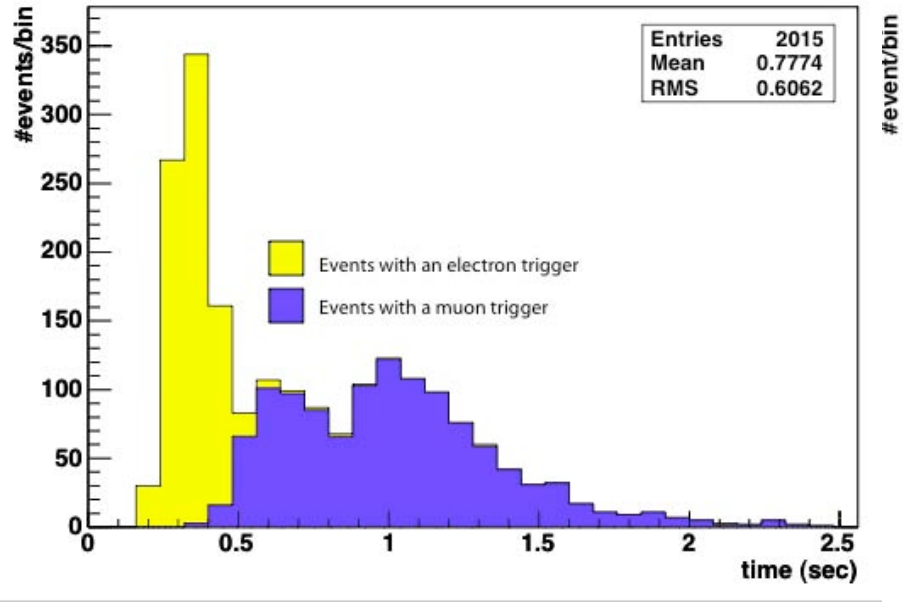
Dominates for background events

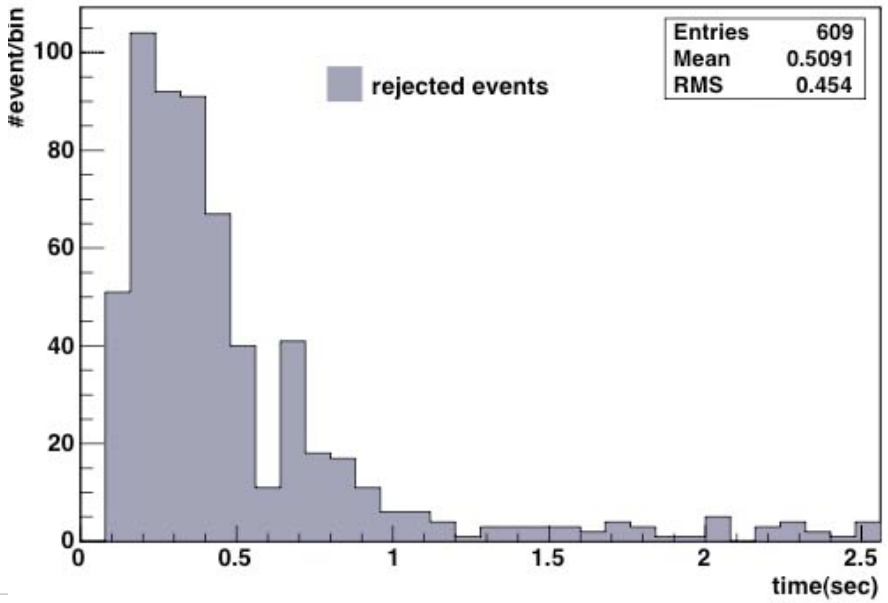
$$\langle Tr \rangle = \prod_{i=1}^{n} (1 - p_i) \sum_{i=1}^{n} tr_i$$

Invariant under permutation

→ Time optimisation will reduce tails in the computing time distribution

Time distribution





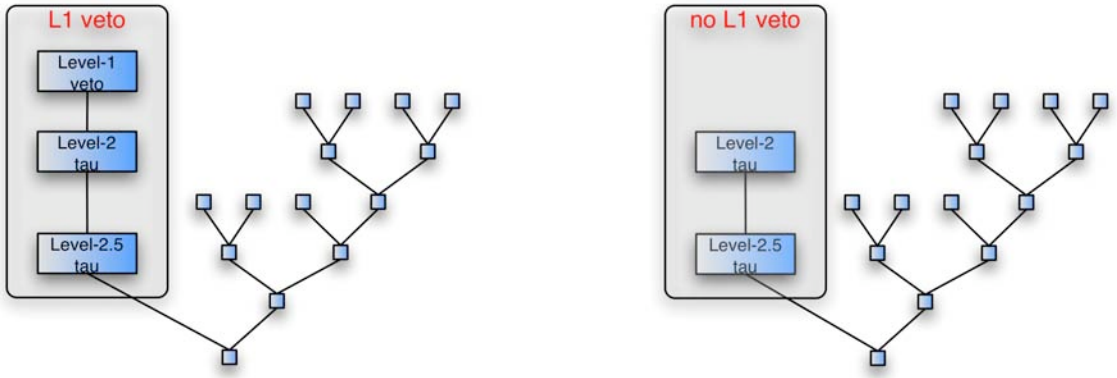
- DY$\rightarrow l^+l^-$ with no Pileup in non optimised mode
- Intel Xeon CPU 2.8GHz, 512kB L2 cache, 2GB RAM
- Mean time per L1 accept
  1. For HLT accepted events: 777 ms
  2. For HLT rejected events: 500 ms

HLT modes

- HLT has been designed to work in two modes:

1. **Veto mode:** High Level sub-triggers are computed only if there is a corresponding L1 accept.
2. **Non veto:** All HLT algorithms are computed even if the corresponding Level-1 as not been fired.

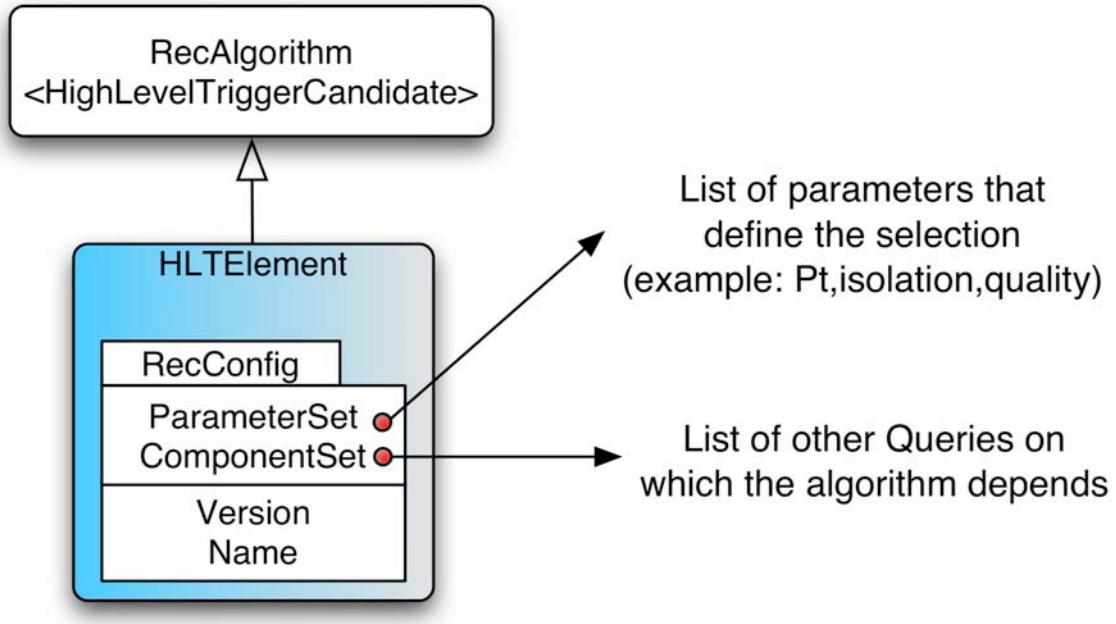
- Natural implementation. At the building of the trigger tree, an additional element is added on top of the tree leaves.



HLT parameter definition

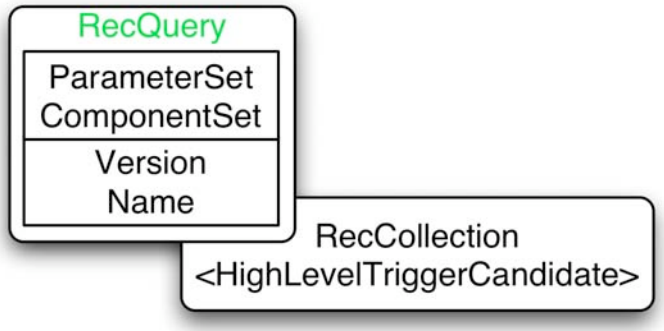
- At each HLT element, the parameters defining the selection are configurable. Example: Pt threshold, Isolation cut.
- It uses the mechanism of RecConfig and RecQuery provided by the CARF framework.

RecConfig: Specify what are the parameters on which the algorithm depends and their default values.



HLT parameter definition

RecQuery: Used to specify the values of the parameters needed to obtain the RecObjects produced by the corresponding RecAlgorithm



When requesting a collection of object:

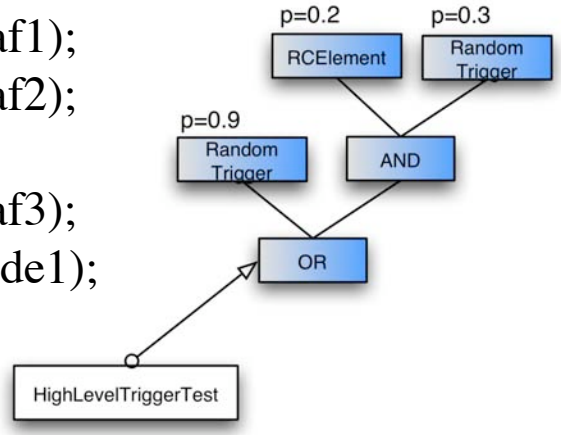
- If no RecQuery is given, the default values are used.
- Otherwise the parameters of the RecQuery are used to construct the collection.

Guarantee same result (RecObj) for the same configuration.

Building the HLT Tree

- The HighLevelTrigger derived class need to implement a setup method in which the construction of a set of RecQuery will define the HLT tree.

```
class HighLevelTriggerTest : public HighLevelTrigger
{
   virtual void setup() {
            RecQuery leaf1("RCTrigger");  leaf1.setParameter("probability",0.2);
            RecQuery leaf2("RandomTrigger");  leaf1.setParameter("probability",0.3);
            RecQuery leaf3("RandomTrigger"); leaf3.setParameter( ``probability", 0.9);
            RecQuery node1("ANDTrigger");
                     node1.setComponent("mother1",leaf1);
                     node1.setComponent("mother2",leaf2);
            RecQuery racine("ORTrigger");
                     racine.setComponent("mother1",leaf3);
                     racine.setComponent("mother2",node1);
            addRootTriggerElement(racine);
            }
};
```



XML specification of the HLT tree

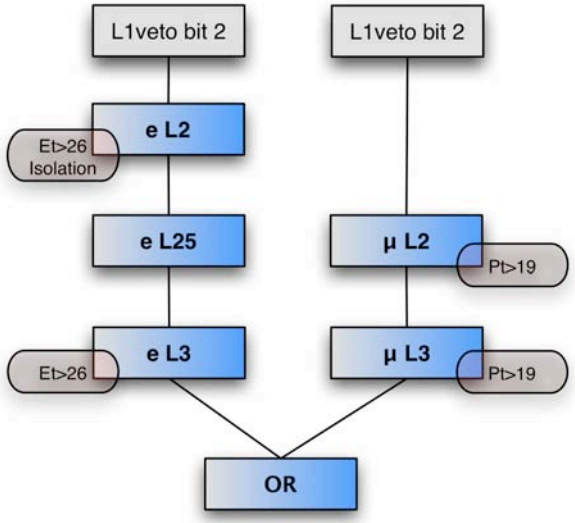
- XML can be used to specify the whole tree with all the parameters that defines the trigger behaviour.
- No need to recompile to change the trigger configuration.
- HighLevelTriggerXML builds the set of RecQuery out of an XML file

```
<?xml version="1.0" encoding="Latin-1" standalone="no"?>
<!DOCTYPE GlobalTrigger SYSTEM "./HighLevelTrigger.dtd">
<GlobalTrigger>
      <OR_Node vetoBit="-1" >

            <L3_electron_trigger  name="HLTelectrons">
                  <L25_electron_trigger>
                        <L2_electron_trigger vetoBit="2">
                          <Parameter value="26" name=”EtThr"/>
                          <Parameter value= "1" name="Isolated"/>
                        </L2_electron_trigger>
                  </L25_electron_trigger>
                <Parameter value="26" name="EtThr"/>
            </L3_electron_trigger>

            <L3_single_muon_trigger name="HLTmuons">
                  <L2_single_muon_trigger vetoBit="0">
                    <Parameter name="PtThr" value="19"/>
                  </L2_single_muon_trigger>
                <Parameter name="PtThr" value="19"/>
            </L3_single_muon_trigger>

      </OR_Node>
</GlobalTrigger>
```



Trigger Elements implementations

Element class	Description
HLGL2TauTrigger	Level 2 single tau Trigger
HLGL25PixelTauTrigger	Level 2.5 tau validation with pixel
HLGL25TrackerTauTrigger	Level 2.5 tau validation with tracker
HLGL2EleTrigger	Level 2 single electron Trigger
HLGL25EleTrigger	Level 2.5 single electron Trigger
HLGL2L25DoubleEleTrigger	Level 2 or Level 2.5 double electron Trigger
HLGL2L25PhotonTrigger	Level 2 or Level 2.5 single photon Trigger
HLGL2MuTrigger	Level 2 single muon Trigger
HLGL3MuTrigger	Level 3 single muon Trigger
HLGL2JetTrigger	1234 Jet Trigger (calorimetric)
HLGL2MetTrigger	Missing Et Trigger
HLGL3EleTrigger	Level 3 single electron Trigger
HLGL3PhotonTrigger	Level 3 photon Trigger
HLGL3BJetTrigger	Level 3 TrackCounting b tagger
HLGL2JPsiTrigger	Specific J/ tagger
ttHJetTagging	ttH Trigger
HighLevelTriggerAndCombNode	Logical and between 2 Trigger, Candidates can be requested to be separated.
HighLevelTriggerAndNode	Logical and between 2 Trigger
HighLevelTriggerNot	Logical not
HighLevelTriggerOrNode	Logical or between 2 Trigger
HighLevelTriggerRandomElement	random Trigger

17 Trigger Elements are implemented.

- Selection of:
  (e/γ,$\mu\tau$)
  Missing Et,
  B Jet,
  1,2,3,4 Jets

- Can be combined with the logical Elements:
  AND
  OR
  CombAnd: ex, find a pair that has same vertex.

Conclusion

HLT steering software has been developed,

- It uses reconstruction algorithms that are combined to form the decision logic.
- The decision logic is implemented in a tree.

The evaluation sequence can be ordered to minimise the time the evaluate the trigger response.

The decision logic can be specified via a XML file.

All selection parameters are programmable in a coherent way.