

Proposed Changes to the ORCA Calorimeter Software

Vladimir Litvin

Caltech

Chris Tully

Princeton

July 3, 2001

Abstract

Proposals are given for improvements to the ORCA calorimetry code. A recommendation is made to stage the changes. At each stage the accuracy and robustness of the changes can be tested and verified. The purpose of this note is to circulate the details to gain feedback and modifications to this draft.

Introduction

Several specific requests are given on how to improve the calorimetry code. Each of these requests tries to satisfy the need for speed optimization while maintaining or increasing the accuracy of the formatting process. Additional functionality to enable test beam data processing is also introduced. The creation of an associated structure for Monte Carlo information is proposed.

The proposal tries to avoid the problem of doing too much too fast. It first identifies key steps and in what order to introduce and test them ¹.

Step 1

The bulk of the processing time can be removed by a simplification of the noise injection without loss of accuracy. Currently, the cell formatting procedure creates empty `CaloTimeSamples` for cells which contain no simulation hits. Each of these objects then has noise injected into the time samples and is formatted into `CaloDataFrames`.

The first step is to skip the creation of empty `CaloTimeSamples` or subsequent `CaloDataFrames` for cells with no simulation hits. These cells have a predictable distribution of derived `CaloRecHit` properties. The `CaloRecHit` objects for these empty cells can be created in the `CaloRecHit` formatting stage. If zero suppression is enabled, the empty `CaloRecHit` objects can be tested before creation so that only objects above threshold are kept. This means specifically to throw a single MC trial for the amplitude and then test it against the threshold before creating the object. For empty `CaloRecHits` above threshold, the remain quantities need to be derived from the appropriate probability density functions. For instance, the χ^2 values can be chosen from a distribution such that, for the corresponding number of degrees of freedom from the fit, the χ^2 probability is flat.

Included in this step will be the replacement of analytical pulse shape functions with pulse shape histograms in one nanosecond steps. One nanosecond is the storage accuracy of the simulation hits. This method is already used for the HCAL, and the same procedure can be used for the ECAL and preshower. Basically, the calculation time for special functions such as exponentials is time consuming. Instead of passing a pointer to a class which contains an analytical function, the class will access a histogram which is

¹No modifications will be made to the `CaloDetailed` classes.

created once in the channel setup procedure. Only one histogram will exist per region. The ECAL barrel, for instance, can be serviced by a single histogram.

Step 1 has implications for the trigger primitive calculation as the first step in the calculation is to digitally add the linearized CaloDataFrames when more than one cell exists per trigger element. To compensate for missing CaloDataFrames, the noise contribution appropriate for the missing number of cells needs to be added to the derived quantities. Despite what was just mentioned, the trigger primitives are currently computed from CaloTimeSamples. This is addressed in step 2.

The benefit of Step 1 is mainly to increase the formatting speed. A side result of this step is that the total size of all the CaloDataFrames from the calorimetry is now manageable. A complete raw set from the ECAL constitutes about 2.4 MBytes when empty cells are included. This size is reduced to 120 kBytes when only hit cells are kept. This size is compatible with the expected size from detector data after selective readout techniques are applied and should be manageable for the persistent store. This point is returned to in step 3. This step addresses the main inefficiency of the calorimetry code. However, the modification to the trigger primitive processing will need substantial testing.

Step 2

The navigation needs to be simplified for the calculation of the trigger primitive information of the ECAL. At the moment, the coarse grain object, which is called the EcalTriggerTower, is created with a list of finer granularity constituents, the crystals. Instead of using the navigation in the finer granularity base to construct strips, the EcalTriggerTower should contain precomputed lists of crystals for each of the strips, the number of strips and the strips should be ordered in azimuthal angle. This will retain the flexibility to change the strip definitions while avoiding the need for finer granularity navigation for a default set of strips.

Currently, trigger primitives are computed from CaloTimeSamples. This needs to be switched to operating on CaloDataFrames which means the introduction of linearization before the trigger primitive generation is performed. This step is needed for step 3 to follow.

These modifications to the trigger primitive generation code needs to be

well tested and verified.

Step 3

In order to be able to process test beam data with the calorimetry code, the CaloDataFrames must be made persistent. The formatter for this object can then be used to retrieve test beam CaloDataFrames in place of simulated CaloDataFrames². It is also proposed at this step to create an associated object for each cell which contains hits. This associated object would contain a list of Monte Carlo hits for the cell. This list may be confined to in-time hits and separated specifically into signal hits and pile-up hits. Depending on the allowable size, even in-time pile-up hits may need to be suppressed.

By allowing the CaloDataFrames to be persistent, all calculations that run from these objects must not depend on CaloTimeSamples, i.e. the CaloDataFrame formatting can be run as a separate production and will not include trigger primitive processing.

As this step involves the creation of new formatters and new objects into the Objectivity schema, it can only be introduced when modifications to the database can be tested.

Step 4

There are some methods of the readout class which need to remain separate from the offline use. The first of these is the pulse shape analyzer. In the readout class, the pulse shape analyzer is meant only for the zero suppression algorithm. The constraints on this algorithm come from the choice made in the digital readout hardware. The linearization process used in the front-end for the trigger primitive calculation must also remain separate from the offline linearization process. The online linearization is limited by the RAM lookup table techniques.

This step requires low level modifications in the calorimetry code and can potentially cause many problems if not implemented correctly.

²For the processing of test beam data, a calibration database is needed.

Conclusions

A set of proposed changes have been made for the calorimetry software. These changes are partitioned into steps. This draft proposal spells out the steps which will occur and the benefits to the software from each step. These are not expected to be easy modifications and need to be reviewed first for their content before implemented in a standard ORCA release.