Toolkit for data reduction to tuples for the ATLAS experiment
Scott Snyder (BNL) <snyder@bnl.gov>
Attila Krasznahorkay (NYU) <Attila.Krasznahorkay@cern.ch>
(For the ATLAS Collaboration)

INTRODUCTION

The final stage of a physics analysis is most often performed from a tuple-based data format using tools such as ROOT [1]. ATLAS grew to have many such tuples, each produced by a different group with different conventions. One size doesn’t fit all, but ATLAS provided a modular toolkit for producing such tuples. This allows sharing implementations, and more importantly, ensures that all tuples made this way share common naming and conventions.

OVERVIEW

File: egamma.root
Tree: egamma
Object: el_
Block: Electrons
el_eta: vector<float>
el_phi: vector<float>
Block: Cluster
eL_cl_eta: vector<float>
Block: Track
eL_trk_pt: vector<float>
eL_trk_index: vector<int>
Object: trk_

ASSOCIATIONS

Tools can associate either from one object to another (single association) or from one object to a set of objects (multiple association). Associations can be represented in the tuple either by adding an index to another object in the tuple ("el_trk_index" above) or by directly “containing” the target object within the source object ("el_el_cl_eta"). Multiple contained associations can use either a nested vector ("el_trk_pt") or enter additional tuple rows.

BLOCK FILLER TOOLS

Should take a single object and copy data from it to the tuple; looping over containers of objects is handled by the caller.

VIRTUAL BlockFillerTool:
struct FourMomFillerTool:
    public BlockFillerTool<FourMom> {
        ... Boilerplate omitted.
        // Variables being filled.
        // Class types may also be used.
        float *m_pt, *m_eta, *m_phi;
        // Called once to declare variables to fill.
        virtual StatusCode fill (const FourMom & p) { ...
            return StatusCode::SUCCESS;
        }
        // Called once to declare variables to fill.
        virtual StatusCode book() { ...
            return StatusCode::SUCCESS;
        }
    } // Called once to declare variables to fill.

GENERIC COMPONENTS

Block filler and association tools depend on the types of objects being manipulated. However, the core tools for retrieving objects from the event data store, looping over collections, calling block filler tools, and formatting the data into the tuple are mostly independent of the types being manipulated. The standard C++ RTTI is augmented with information about class inheritance relations; this allows generic pointers to be properly converted.

ALTERNATE TUPLE FORMATS

While all physics analyses so far use ROOT tuples, a prototype exists for writing HDF5 [2]. No changes are needed beyond selecting the alternate backend implementation.

REFERENCES


ACKNOWLEDGMENT

This work is supported in part by the U.S. Department of Energy under contract DE-AC02-98CH10886 with BNL.