Table of Contents

Analysis with FWLite TSelectors

Complete:

Introduction

TSelector is a ROOT utility that allow running analysis under parallel PROOF farms. Documentation on TSelector can be found in Chapter 12 of ROOT users manual.

CMSSW Framework defines a specialized version of TSelector, FWLite TSelector, that allows direct use of the EDM Event structure, instead of dealing explicitly with ROOT branches. This technique makes porting code from interactive analysis to a batch EDAnalyzer much easier.

The following section explains how to write an analysis algorithm that can run under a FWLite TSelector and can be used with an EDAnalyzer.

The code from this example can be found in:

- CMS.PhysicsTools/ParallelAnalysis V02-00-01

Fast Start

The fastest way to get started is to use the mktsel command line tool

```
mktsel <name>
```

which will create a directory named which has the proper CMSSW package structure and includes skeletons of all the necessary source files, the CMS.BuildFile and the dictionary generation files (`classes.h` and `classes_def.xml`).

Writing the Analysis Algorithm

The algorithm interface should define the following methods:

```
TrackAnalysisAlgorithm( const TList *, TList & out );
void process( const edm::Event&  event );
void postProcess( TList & out );
```

where:

- The constructor's first argument is a possibly null pointer to a TList where that TList is sent from the TSelector's `begin(...)` method (e.g.: histogram list).
- The constructor should fill the TList `out` with the objects to be filled during the event loop (e.g: histograms). The TList `out` will be send from each algorithm on a parallel node and then a merged TList will be passed to the TSelector's `terminate(...)` method.
- The method `process(...)` is called for every event (e.g.: histograms should be filled).
- The method `postProcess(...)` is called at the end of processing in each parallel node. The argument `out` is the same TList which was passed to the constructor.

For instance, the structure could be the following:

```
struct TrackAnalysisAlgorithm {
  TrackAnalysisAlgorithm( const TList *, TList& );
```

```
  void process( const edm::Event&  );
  void postProcess( TList & );
  TH1F * h_pt, * h_eta;
  static const char * kPt, * kEta;
};
```

where we added:

- histogram pointers (`h_pt`, `h_eta`) to be used during event processing
- histogram names (`kPt`, `kEta`) for convenience, in order to easily share this information among different classes, as seen in the following.

The complete code can be found in the following source files:

- `interface/TrackAnalysisAlgorithm.h`
- `src/TrackAnalysisAlgorithm.cc`

Writing the TSelector Class

The TSelector should inherit from `TFWLiteSelector<TrackAnalysisAlgorithm>`, and define the following methods:

```
void begin( TList * & );
void terminate( TList & );
```

where:

- The method `begin(...)` is called once for each `TTree::Process(...)` (or `TChain::Process(...)`) call. The argument `TList * &` is a pointer to a TList you can create and then have passed to each algorithm on a parallel node.
- The method `terminate(...)` is called once processing has finished on all nodes. The TList argument is the merged results from the algorithms on all the parallel nodes.

The complete code can be found in the following source files:

- `interface/TrackTSelector.h`
- `src/TrackTSelector.cc`

Defining REFLEX Dictionary for TSelector class

The TSelector subclass defined above should be compiled, together with the algorithm class, and a REFLEX dictionary has to be generated in order to make the resulting library usable from ROOT interactive prompt. This is done as following:

- include the header file of the class in the file `classes.h`
- add the class name to the directives in the file `classes_def.h`:

```
<class name="examples::TrackTSelector" />
```

The actual code can be found in the following files:

- `src/classes.h`
- `src/classes_def.xml`

Using the TSelector Interactively

Once the new TSelector with its dictionary is compiled, it can be used interactively from ROOT prompt.

- start ROOT:

```
> root
```

- Enable automatic data formats library loading (FWLite):

```
gSystem->Load("libFWCoreFWLite");
AutoLibraryLoader::enable();
```

- Load the library defining the new TSelector class:

```
gSystem->Load( "libPhysicsToolsParallelAnalysis" );
```

- Setup the event data file chain:

```
TChain events("Events");
events.Add("aod.root");
```

- loop over events and process the chain:

```
events.Process( selector );
```

The full macro can be found below:

- `test/trackExample.C`

Reusing Analysis Algorithm in EDAnalyzer

Since the analysis algorithm classes used by FWLite TSelector use the common CMSSW event interface, they can be reused to write a framework EDAnalyzer module.

An algorithm object of the class `TrackAnalysisAlgorithm` becomes a private data member of the EDAnalyzer class, and is initialized at constructor time. The method `process(...)` of the algorithm is invoked in the method `analyze(...)` of the EDAnalyzer:

```
void TrackTSelectorAnalyzer::analyze( const edm::Event & event, const edm::EventSetup & ) {
  algo_.process( event );
}
```

The complete source code can be found below:

- src/TrackTSelectorAnalyzer.h
- src/TrackTSelectorAnalyzer.cc
- src/SealModule.cc

Such analyzer can be used in frameworks batch application, as according to the following job configuration script:

- test/trackExample.cfg

This approach could be further generalized in the future. Using a template class it could be possible to further

reduce the code that users need to write to define the EDAnalyzer.

Review Status

Reviewer/Editor and Date (copy from screen)	Comments
LucaLista - 31 Oct 2006	page created
BenediktHegner - 31 Jan 2007	page content last edited
JennyWilliams - 07 Feb 2007	editing to include in SWGuide

Responsible: LucaLista
Last reviewed by: Reviewer

This topic: CMSPublic > SWGuideTSelectorAnalysis
Topic revision: r13 - 2007-02-21 - JennyWilliams

Copyright &© 2008-2019 by the contributing authors. All material on this collaboration platform is the property of the contributing authors.
Ideas, requests, problems regarding TWiki? Send feedback