Table of Contents

PAT Examples: Trigger Example

Contents

Introduction

Trigger information stored in the CMS EDM is optimzed for space, which comes with a lack of user friendlyness in accessing this information. The PAT provides the opportunity to access this information easily and especially to deal with the cross links between the different items (objects, filters/modules and paths).

The aim of this PAT Workbook example is to complete the information given during the PAT tutorial exercises. It is therefor necessary for the understanding of the exercises at least to follow the tutorial first:

- **Necessary**: A tutorial about PAT trigger information is found here.
- **Recommended**: The PAT trigger information is described in detail in the corresponding Software Guide.

How to get the code

- Prepare your work area for **`CMSSW_3_5_4`**:

```
cmsrel CMSSW_3_5_4
cd CMSSW_3_5_4/src/
cmsenv
```

- Check out the PAT core as it comes with the release:

```
addpkg PhysicsTools/PatAlgos
```

- Compile:

```
scram b -j 5
```

Find out more about the details

The production of PAT trigger information consists of up to four steps:

1. Production of PAT trigger objects, filters and paths:
    - The corresponding configuration file is PhysicsTools/PatAlgos/python/triggerLayer1/triggerProducer_cfi.py.
    - The provided default configuration serves the normal use case and does usually not need to be changed.
2. Matching between PAT and PAT trigger objects:
    - The corresponding configuration file is PhysicsTools/PatAlgos/python/triggerLayer1/triggerMatcher_cfi.py.
    - The provided default configurations cannot cover all use cases due to their wide variety. They rather serve as configuration examples. The user is expected to define her/his own matches and to incorporate them into the work flow.
3. Creation of the PAT trigger event:
    - The corresponding configuration file is PhysicsTools/PatAlgos/python/triggerLayer1/triggerEventProducer_cfi.py.
    - The provided default configuration has to be adapted to the matches the user defined. The provided defaults refer only to the provided default example matches.
4. Embedding of matched PAT trigger objects into the PAT objects:
    - The corresponding configuration file is PhysicsTools/PatAlgos/python/triggerLayer1/triggerMatchEmbedder_cfi.py.
    - Again, the provided default configurations refer to the example matches and need to be adapted to the user defined matches.
    - Trigger match embedding is not discussed on this page.

The production of trigger information is not part of the default PAT work flow. To switch it on easily, Python tools are provided in PhysicsTools/PatAlgos/python/tools/trigTools.py.

Exercises

Get the code skeletons

In your work area:

- Check out the code:

```
cvs co -r patTutorial_mar10_module4_exercise PhysicsTools/PatExamples
```

- Do **not** compile!

This is not needed for exercise 1.

Exercise 1 - trigger information production

The problem to solve

Modify an existing PAT configuration file so, that it

- contains the **full** PAT trigger information;
- contains **no** trigger matches.

Hints:

- Use an existing configuration that works.
- Python tools can and should be used.

The solution

- To switch on the default PAT trigger and trigger event production, append these lines to you PAT configuration file:

```
from CMS.PhysicsTools.PatAlgos.tools.trigTools import switchOnTrigger
switchOnTrigger( process )
```

- To remove the default (example) trigger matches from the work flow, also append these lines:

```
process.patTriggerSequence.remove( process.patTriggerMatcher )
process.patTriggerEvent.patTriggerMatches = []
```

Exercise 2 - trigger information production with matching

The problem to solve

Modify an existing PAT configuration file so, that it

- contains the **selected** PAT objects
  - s. PAT tutorial on cleaning;
- contains the **full** PAT trigger information;
- contains the (and only this) match of PAT muons to HLT muons with:
  - match by $\Delta\mathbf{R}$,
  - best match in $\Delta\mathbf{R}$ only.

Hints:

- Use the configuration skeleton file PhysicsTool/PatExamples/test/producePatTrigger_cfg.py only:
  - Append all needed changes to this file.
  - Do not change any file in the *CMS.PhysicsTools/PatAlgos* package itself.
- Python tools can and should be used.
- Reduce the number of event for testing in

```
process.maxEvents.input = 1000
```

- The output file of this exercise serves also as input file for exercise 3.

The solution

- To switch to **selected** PAT objects, add these lines:

```
from CMS.PhysicsTools.PatAlgos.tools.coreTools import removeCleaning
removeCleaning( process )
```

- The described match is defined as follows:
  - ♦ Match by ΔR only with ranking also by ΔR:

```
process.muonTriggerMatchHLTMuons = cms.EDFilter( "PATTriggerMatcherDRLessByR",
```

- ♦ Match **selected** PAT **muons** now:

```
src     = cms.InputTag( "selectedPatMuons" ),
```

This is the input tag as found of the according muon selection module.

- ♦ Match to PAT trigger objects:

```
matched = cms.InputTag( "patTrigger" ),
```

This is the **default** input tag as found of the according trigger producer.

- ♦ Use the **AND** of the selector configurables:

```
andOr          = cms.bool( False ),
```

This allows for wild cards.

- ♦ Select HLT muons:

```
filterIdsEnum  = cms.vstring( 'TriggerMuon' ),
filterIds      = cms.vint32( 0 ),
```

or

```
filterIdsEnum  = cms.vstring( '*' ),
filterIds      = cms.vint32( 83 ),
```

The corresponding codes are found in DataFormats/HLTReco/interface/TriggerTypeDefs.h.

- ♦ Skip further selection criteria with wild cards:

```
filterLabels   = cms.vstring( '*' ),
pathNames      = cms.vstring( '*' ),
collectionTags = cms.vstring( '*' ),
```

- ♦ Use the default selection limits for muons:

```
maxDPtRel = cms.double( 0.5 ),
maxDeltaR = cms.double( 0.5 ),
```

These are described in the Software Guide - PATTriggerMatcher Module Configuration

- ♦ Select only one match:

```
resolveAmbiguities    = cms.bool( True ),
```

- ♦ Select the match by the defined ranking:

```
resolveByMatchQuality = cms.bool( True )
)
```

The ranking has been defined already by the choice of the concrete matcher module (s. above).

- The new match (and only this one) is included in the work flow by:
  - ♦ Load the PAT trigger information into your process:

```
process.load( "CMS.PhysicsTools.PatAlgos.triggerLayer1.triggerProducer_cff" )
```

Modify the trigger matcher sequence, which is found in PhysicsTools/PatAlgos/python/triggerLayer1/triggerMatcher_cfi.py:

- ♦ ◊ Add the new matcher module **first**:

```
process.patTriggerMatcher += process.muonTriggerMatchHLTMuons
```

- ♦ ◊ Remove anything else found to be present in the matcher sequence:

```
process.patTriggerMatcher.remove( process.patTriggerElectronMatcher )
process.patTriggerMatcher.remove( process.patTriggerMuonMatcher )
process.patTriggerMatcher.remove( process.patTriggerTauMatcher )
```

- ♦ ◊ Let the PAT trigger event being aware of the (and only this) new match:

```
process.patTriggerEvent.patTriggerMatches = [ "muonTriggerMatchHLTMuons" ]
```

- To finally switch on the PAT trigger information:

```
from CMS.PhysicsTools.PatAlgos.tools.trigTools import switchOnTrigger
switchOnTrigger( process )
```

Since the Python tool **`switchOnTrigger( process )`** needs the final list of trigger matches to be included, it has to come **last** in the whole procedure. This is also the reason, why the PAT trigger information has to be loaded explicitly before to be modified (otherwise the Python tool does this). This differs from the approach in exercise 1.

Exercise 3 - trigger information analysis

The problem to solve

Modify an existing CMSSW analyzer skeleton so, that the analyzer:

- compares *pt*, and of the PAT/trigger object matches found in exercise 2 in 2-dim histograms;
- fills a histogram with the mean value of *pt* for all present trigger filter IDs of the trigger objects (as defined in DataFormats/HLTReco/interface/TriggerTypeDefs.h).

Hints:

- Use the checked out analyser's code and configuration skeleton files:
  - ♦ PhysicsTools/PatExamples/plugins/PatTriggerAnalyzer.cc (sufficient for the first part of the exercise)
  - ♦ PhysicsTools/PatExamples/plugins/PatTriggerAnalyzer.h
  - ♦ PhysicsTools/PatExamples/test/analyzePatTrigger_cfg.py
- The code skeleton does not compile as it comes.
- The output file of exercise 2 serves as input file for this exercise.

- The analyzer uses the **`TFileService`**. Histograms can be defined in ther **`beginJob()`** method. An example is given.
- Handles to access all PAT trigger collections, the PAT trigger event and the PAT muon collection are pre-defined in the **`analyze()`** method. Not all of them are necessarily needed.
- Also the PAT trigger match helper to access cross links between different PAT trigger items is already predefined:

```
const TriggerMatchHelper matchHelper;
```

- The loop over the PAT muons and the access to them in the **needed format** is pre-defined:

```
for ( size_t iMuon = 0; iMuon < muons->size(); ++iMuon ) {
// loop over muon references (PAT muons have been used in the matcher in task 3)
  const reco::CandidateBaseRef candBaseRef( MuonRef( muons, iMuon ) );
} // iMuon
```

- An empty **`endJob()`** method to deal with the *pt* mean values is already present.

The solution

- To compare PAT and trigger objects:
  - The required histograms can be defined in the **`beginJob()`** method e.g. like this:

```
histos2D_[ "ptTrigCand" ] = fileService->make< TH2D >
( "ptTrigCand", "Object vs. candidate p_{T} (GeV)", 60, 0., 300., 60, 0., 300. );
histos2D_[ "ptTrigCand" ]->SetXTitle( "candidate p_{T} (GeV)" );
histos2D_[ "ptTrigCand" ]->SetYTitle( "object p_{T} (GeV)" );
histos2D_[ "etaTrigCand" ] = fileService->make< TH2D >
( "etaTrigCand", "Object vs. candidate #eta", 50, -2.5, 2.5, 50, -2.5, 2.5 );
histos2D_[ "etaTrigCand" ]->SetXTitle( "candidate #eta" );
histos2D_[ "etaTrigCand" ]->SetYTitle( "object #eta" );
histos2D_[ "phiTrigCand" ] = fileService->make< TH2D >(
"phiTrigCand", "Object vs. candidate #phi", 60, -Pi(), Pi(), 60, -Pi(), Pi() );
histos2D_[ "phiTrigCand" ]->SetXTitle( "candidate #phi" );
histos2D_[ "phiTrigCand" ]->SetYTitle( "object #phi" );
```

- - To fill the histograms, add inside the loop over the PAT muons in the **`analyze()`** method this to
    - access the mathed trigger muon to the given PAT muon using the PAT trigger match helper:

```
const TriggerObjectRef trigRef(
matchHelper.triggerMatchObject(candBaseRef,triggerMatch,iEvent,*triggerEvent)
);
```

- - - fill the histograms including a necessary check for the validity of the retrieved objects:

```
if ( trigRef.isAvailable() ) { // check references (necessary!)
  histos2D_[ "ptTrigCand" ]->Fill( candBaseRef->pt(), trigRef->pt() );
  histos2D_[ "etaTrigCand" ]->Fill( candBaseRef->eta(), trigRef->eta() );
  histos2D_[ "phiTrigCand" ]->Fill( candBaseRef->phi(), trigRef->phi() );
}
```

- To analyze the mean *pt* of all trigger objects depending on their filter ID,
  - Define the min. and max. filter ID in:
    - the analyzer's class definition in PhysicsTools/PatExamples/plugins/PatTriggerAnalyzer.h

```
unsigned minID_;
unsigned maxID_;
```

- ♦ ◊ the analyzer's constructor in PhysicsTools/PatExamples/plugins/PatTriggerAnalyzer.cc

```
minID_( iConfig.getParameter< unsigned >( "minID" ) ),
maxID_( iConfig.getParameter< unsigned >( "maxID" ) ),
```

- ♦ ◊ the analyzer's configuration in PhysicsTools/PatExamples/test/analyzePatTrigger_cfg.py

```
minID = cms.uint32( 81 ),
maxID = cms.uint32( 96 ),
```

This is, where the exact numbers as found in DataFormats/HLTReco/interface/TriggerTypeDefs.h go.

- ♦ Define maps to sum up the objects' counts and *pt* depending on the filter ID in PhysicsTools/PatExamples/plugins/PatTriggerAnalyzer.cc

```
std::map< unsigned, unsigned > sumN_;
std::map< unsigned, double >   sumPt_;
```

- ♦ In the **beginJob()** method of the analyzer in PhysicsTools/PatExamples/plugins/PatTriggerAnalyzer.cc:
  - ◊ define the histogram:

```
histos1D_[ "ptMean" ] = fileService->make< TH1D >
( "ptMean", "Mean p_{T} (GeV) per filter ID",
maxID_ - minID_ + 1, minID_ - 0.5, maxID_ + 0.5);
histos1D_[ "ptMean" ]->SetXTitle( "filter ID" );
histos1D_[ "ptMean" ]->SetYTitle( "mean p_{T} (GeV)" );
```

and

- ♦ ◊ initialize the maps:

```
for ( unsigned id = minID_; id <= maxID_; ++id ) {
  sumN_[ id ] = 0;
  sumPt_[ id ] = 0.;
}
```

- ♦ In the **analyze()** method of the analyzer in PhysicsTools/PatExamples/plugins/PatTriggerAnalyzer.cc, accumulate the sums depending on the objects' filter IDs:

```
for ( unsigned id = minID_; id <= maxID_; ++id ) {
  const TriggerObjectRefVector objRefs( triggerEvent->objects( id ) );
  sumN_[ id ] += objRefs.size();
  for ( TriggerObjectRefVector::const_iterator iRef = objRefs.begin();
 iRef != objRefs.end(); ++iRef ) {
    sumPt_[ id ] += ( *iRef )->pt();
  }
}
```

- ♦ In the **endJob()** method of the analyzer in PhysicsTools/PatExamples/plugins/PatTriggerAnalyzer.cc, fill the histogram with the mean values:

```
for ( unsigned id = minID_; id <= maxID_; ++id ) {
  if ( sumN_[ id ] != 0 ) histos1D_[ "ptMean" ]->Fill( id, sumPt_[ id ]/sumN_[ id ] );
}
```

Get the code solutions

In your work area:

- Protect your own code by renaming, e.g. by appending a `.tmp` at the end of each file you have worked on.
- Update the code:

```
cvs up -r patTutorial_mar10_module4_solution3 PhysicsTools/PatExamples
```

- Compile:

```
scram b -j 5
```

Review status

Reviewer/Editor and Date	Comments
RogerWolf - 13 May 2009	Created the template page
VolkerAdler - 18 Jan 2010	Version to publish, based on tutorial in Dec. 2009
VolkerAdler - 16-Mar-2010	Updated for tutorial in Mar. 2010

Responsible: VolkerAdler
Last reviewed by: VolkerAdler - 18 Jan 2010

-- RogerWolf - 11 Jun 2009

This topic: CMSPublic > WorkBookPATExampleTrigger
Topic revision: r6 - 2010-04-28 - RogerWolf