Online Event Selection at the LHC

Exercises

Norbert Neumeister, Teddy Todorov

CERN School of Computing 2003
Krems an der Donau, Austria

Outline

- Exercise 1: Trivial
  - Create a local ORCA workspace
  - Run an example ORCA program
- Exercise 2: Simple
  - Write a regional track reconstruction algorithm
  - Start from muons found by the Level-1 muon trigger
  - Optimize: signal efficiency, background rejection, CPU time
- Exercise 3: Advanced
  - Implement a track isolation algorithm
  - Discriminate between $W \to \mu\nu$ and $bb \to 1\mu + X$ events
  - Optimize: signal efficiency, background rejection, CPU time

ORCA

- **Object Oriented Reconstruction for CMS Analysis**
  - CMS software
  - written in C++
  - based on
    - COBRA (**C**oherent **O**bject-oriented **B**ase for **R**econstruction, **A**nalysis and simulation)
    - CARF (**C**MS **A**nalysis and **R**econstruction **F**ramework)
- Documentation:
  - The main ORCA page is at: **http://cmsdoc.cern.ch/orca**
  - **User Guide**
    Introduction, general description
  - **Reference Manual**
    all classes documented; where you can delve into the details
    http://maincsc.donau-uni.ac.at/orcadoc or
    file:///opt/cms/Releases/ORCA/ORCA_7_2_4/doc/ReferenceManual/html/

Exercise 1

Creating a Workspace

- `cd your_home_directory`
- `scram project ORCA ORCA_7_2_4`
  - new directory ORCA_7_2_4 with subdirectories: src, config, tmp, logs
- `cd ORCA_7_2_4/src`
- `cp -r /home/neumeist/ORCA_Exercise_1 .`
- `cd ORCA_Exercise_1`
- `ls -a`
  - `ORCAExercise.cpp`
  - `BuildFile`
  - `.orcarc`
- `scram build bin`
- ``eval `scram runtime -sh` ``
- `ORCAExercise`

ORCAExercise.cpp

- Program prints:
  - simulated (GEANT) muons above threshold
  - Level-1 trigger decision
- Level-1 Trigger setting: Single Muon Trigger with p_T threshold = 12 GeV/c
- Event observer class

```
class MyEventAnalyser : public EventAnalyser {
```

- Registration of MyEventAnalyser to the framework

```
PkBuilder<MyEventAnalyser> eventAnalyser("MyEventAnalyser");
```

- Event loop

```
analysis(G3EventProxy*);
```

- **`G3EventProxy`**
  - Proxy for event data
- **`SimTrack, RecTrack`**
  - Classes for simulated and reconstructed tracks

BuildFile

selects the libraries

```
<environment>

  <group name=RecReader>
  <external ref=COBRA Use=CARF>

  <group name=L1GLOBAL>
  <use name=Trigger>

  <group name=CaloRecHitReader>
  <use name=Calorimetry>

  <group name=MuonDigiReader>
  <use name=Muon>

  <group name=TkDigiReader>
  <use name=Tracker>

  <group name=TkTracks>
  <use name=TrackerReco>

  <bin file=ORCAExcerise.cpp name=ORCAExercise></bin>

</environment>
```

Reading of Digis

Name of executable

.orcarc Datacards

```
FirstEvent = 0
LastEvent = -1

FilePath = /opt/cms/ORCADATA/Digis

# Signal
InputCollections = /System/Pileup1034/W_munu_1mu/W_munu_1mu

# Background
//InputCollections = /System/Pileup1034/b_1mu/b_1mu

# Trigger threshold [GeV/c]
ORCAExercise:ptThreshold = 12
```

Exercise 2

First Step

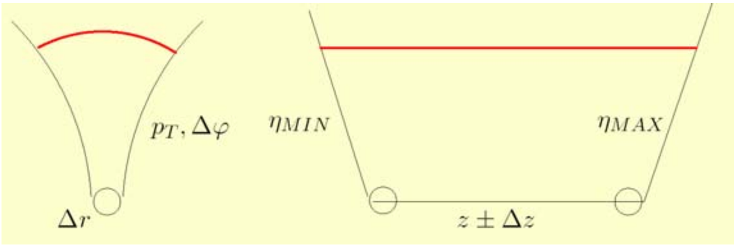
- cd your_home_directory
- cd ORCA_7_2_4/src
- cp -r /home/neumeist/ORCA_Exercise_2 .
- cd ORCA_Exercise_2
- ls -a
  - BuildFile
  - L1MuonTrackingRegionBuilder.h
  - L1MuonTrackingRegionBuilder.cc
  - ORCAExercise.cpp
  - .orcarc
- scram build bin
- eval `scram runtime -sh`
- ORCAExercise

Regional Track Reconstruction

- **Implement the method:**
  bool MyEventAnalyser::regionalTrackReconstruction()
  – Reject/Accept events based on the presence of **at least** one reconstructed track above threshold in the region of the Level-1 trigger object!
- **Use:**

```
L1MuonTrackingRegionBuilder trb;
vector<TrackingRegion*> regions = trb.regions();

RectangularEtaPhiTrackingRegion(const GlobalVector& dir,
                                const GlobalPoint& vtxPos,
                                float ptMin,
                                float rVtx, float zVtx,
                                float deltaEta, float deltaPhi)
```



– **dir:** the direction around which the region is constructed
– **vtxPos:** the position of the vertex (origin) of the of the region.
The vtxPos is supposed to be placed on the beam line, i.e. GlobalPoint(0,0,float)
– **ptMin:** minimal p_T of interest
– **rVtx:** radius of the cylinder around beam line where the tracks of interest should point to
– **zVtx:** half height of the cylinder around the beam line where the tracks of interest should point to
– **deltaEta:** allowed deviation of the initial direction of particle in η with respect to direction of the region
– **deltaPhi:** allowed deviation of the initial direction of particle in φ with respect to direction of the region

Components

- Seed Generator:

```
CombinatorialSeedGeneratorFromPixel theSG;
vector<TrajectorySeed> s = theSG.seeds();
vector<TrajectorySeed> s = theSG.seeds(const TrackingRegion&);
```

returns all seeds in a region

- Trajectory Builder:

```
CombinatorialTrajectoryBuilder theTB;
vector<Trajectory> t = theTB.trajectories(const TrajectorySeed&);
```

returns all trajectories built from a seed (with ambiguities)

- Trajectory Cleaner:

```
TrajectoryCleanerBySharedHits theTC;
theTC.clean(vector<Trajectory>&);
vector<Trajectory> t;
theTC.clean(t);
```

resolves the ambiguities of a collection of trajectories (invalidates the removed ones)

- Trajectory Smoother:

```
KFFittingSmoother theSmoother;
vector<Trajectory> t = theSmoother.trajectories(const Trajectory&);
```

smooth a trajectory using the already found hits

Important Classes

- Trajectory:
  - A trajectory is an ordered sequence of `TrajectoryMeasurement` objects (RecHits). The measurements are added to the Trajectory in the order of increasing precision.

```
Trajectory t;
TrajectoryMeasurement m = t.firstMeasurement();
PropagationDirection dir = t.direction();
```

PropagationDirection: `alongMomentum` (outwards) or `oppositeToMomentum` (inwards)

- RecTrack:
  - The result of track reconstruction is a **`vector<RecTrack>`**

```
RecTrack track(const Trajectory&); // convert Trajectory → RecTrack
TrajectoryStateOnSurface tsos = track.impactPointState();
GlobalVector mom = tsos.globalMomentum();
GlobalPoint pos = tsos.globalPosition();
float pt = mom.perp();   // transverse momentum
float eta = mom.eta();   // pseudorapidity
int charge = tsos.charge();
int hits = track.foundHits();
```

Results

- Optimize for speed and efficiency

- Run on signal and background sample (1000 events each):
    - **Signal: W** $\rightarrow \mu\nu$
    - **Background: bb** $\rightarrow$ **1**μ **+ X**

- Competition:
  - **The program will automatically send an e-mail with:**
    - username
    - signal efficiency
    - background efficiency
    - CPU time

  - **Final results can be found on:**

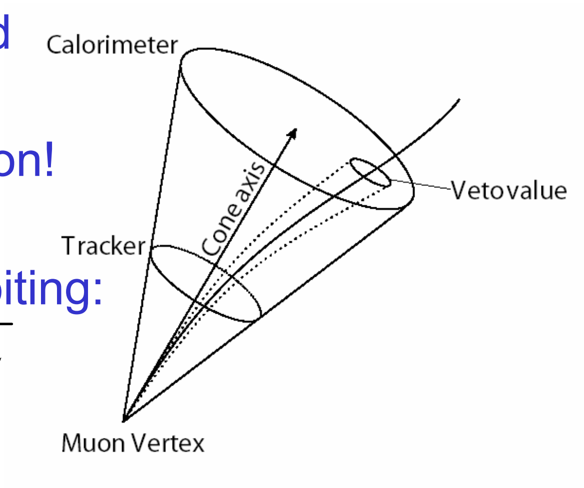
    http://maincsc.donau-uni.ac.at/~orca/

Exercise 3

Track Isolation

- **Use the program from Exercise 2 and implement the method:**
  bool MyEventAnalyser::trackIsolation(const RecTrack&)
  - Reject muons with high "activity" in their neighborhood
  - Perform regional tracking in region around muon
  - Create tracking region using vertex information of muon!
  - Reuse code from Exercise 2
  - Calculate ΣP_T of tracks in a cone around muon, exploiting:
    - Cone size: $\Delta R < 0.2$
    - Veto region: $\Delta R < 0.005$
    - $p_T^{min} \sim 0.8$ GeV
    - Isolation threshold: $\Sigma P_T < 2.0$ GeV $\rightarrow$ muon is isolated
  - Optimize cone size and threshold

$$\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\varphi)^2}$$



- Run on signal and background sample as in Exercise 2:
  - **Signal: W** $\rightarrow \mu\nu$
  - **Background: bb** $\rightarrow$ **1**μ **+ X**

The END

- Solutions can be found under:

  /home/neumeist/ORCAExercise_Solution

- You can find this document under:

  /home/neumeist/Exercises.pdf

- Final results of competition can be found under:

  http://maincsc.donau-uni.ac.at/~orca/

- Have fun