Session 1: Roll up your sleeves…

Chaouki Boulahouache
University of Pittsburgh/CERN

- Introduction…
- Definitions/Information…
- Hands on Exercises…

Introduction…

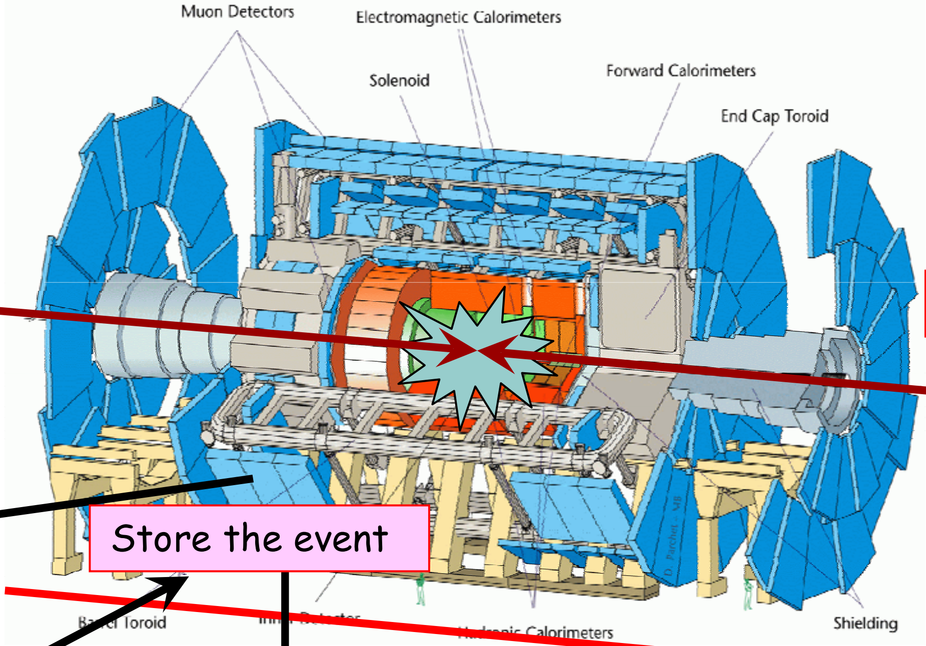
- We will try to cover the following topics:

1. Setup a work area.
2. Setup a particular version of the ATLAS software releases
3. Monte-Carlo
4. Trigger
5. Physics analysis objects.
6. Use the physics objects in an AOD analysis.
7. Use the GRID (PAnda, Ganga)
    1. Analyze data
    2. Generate MC samples,
8. …

Few Requirements…

- We expect that most of you have an lxplus account.
- At least some of you have valid grid certificate.

The Basics: A simplistic picture for a Complicated Detector



Beam 2

Beam 1

+z

Trigger(L1+HLT)

Store the event

L1A Fired?

NO

YES

Discard the event

- Events are streamed and stored into RDO (Raw Data Objects) files: only ByteStream format or output of digitization.
- Then these files are reconstructed producing the ESD (Event Summary Data/Full output) and AOD (Analysis Object Data/slimmed version of ESDs) files.

- **Today: we will use an AOD (unfortunately not from REAL data), but from Monte-Carlo (MC).**

Information…

- We will setup release 14.2.21
- You have the possibility to choose different release later on, but need to make sure the access to physics objects is the same or modify the code accordingly.
- We will try to do actual work and use ROOT to make plots…
- We will post the information of each session according to the schedule in this twiki: https://twiki.cern.ch/twiki/bin/view/Main/IctpTutorial

- I will not go into complicated explanations of the ATLAS software, but the aim is to make you start an analysis as quick as possible (during the 1h:30' per session).

Ready?

START…

Definition of few Terms

- Algorithm: user application controlled by framework.
  - inherits from Algorithm class
  - Implements three methods for invocation by framework: initialize(), execute(), finalize()
- Data Object: generally produced/accessed by Algorithms, managed by TDS.
  - Examples of these objects will be given in sessions 2/3.
- Services
  - Globally available software components providing specific framework capabilities, e.g., Message service, Histogram service, etc
- Job Options files
  - used to control an Athena application configuration at run-time, load all the needed algorithms needed in your analysis together with the required configuration.

Services And Message Streams

- Within the Algorithm services are readily accessible, the most common are:
  - messageService()
  - histoSvc( )
  - ntupleSvc( )
- more…

- MsgStream → Used for a more convenient way to print out different levels of information.
  - Most important message outputs are:
    - DEBUG (OutputLevel = 2)
    - INFO (OutputLevel = 3)
    - WARNING (OutputLevel = 4)
    - ERROR (OutputLevel = 5)
    - FATAL (OutputLevel = 6)
  - Severity level is defined per object instance/Algorithm:
    - Z_Analysis.OutputLevel = 2
    - ServiceMgr.MessageSvc.OutputLevel = 3

- Print messages with MsgStream

1. Include message stream header

   #include "GaudiKernel/MsgStream.h"

2. For example, in initialize()

```
MsgStream mLog( messageService(), name() );
mLog << MSG::INFO << "Initializing Z_Analysis" << endreq;
```

How do you use your algorithm ?

- See example: "run/jobOptions_Z_Analysis.py"

```
from AthenaCommon.AlgSequence import AlgSequence
topSequence = AlgSequence()
from Z_Analysis.Z_AnalysisConf import Z_Analysis
topSequence += Z_Analysis()
```

- Then you can specify the algorithm properties (see example)

```
Z_Analysis = Z_Analysis()
Z_Analysis.ElectronContainer     = "ElectronAODCollection"
```

Book and Fill Ntuples

❖ See example: "run/jobOptions_Z_Analysis.py".

1. In the header file, you need to specify the ntuple variables:

```
#include "GaudiKernel/NTuple.h"

NTuple::Item<double> nt_ZeeInvMass;
```

2. In initialize():

```
NTuple::Tuple* Ztoee = ntupleSvc()->book ("/NTUPLES/FILE1/Ztoee", CLID_ColumnWiseTuple, "ntuple");
Ztoee->addItem ("ZeeInvMass", nt_ZeeInvMass) ;
```

3. In execute(), somewhere after you fill the variable "nt_ZeeInvMass":

```
ntupleSvc()->writeRecord("/NTUPLES/FILE1/Ztoee");
```

4. Ntuple Persistency: in the jobOption file add the following:

```
from GaudiSvc.GaudiSvcConf import NTupleSvc
ServiceMgr += NTupleSvc()
ServiceMgr.NTupleSvc.Output  = [ "FILE1 DATAFILE='ZAnalysis_ntuple.root' TYP='ROOT' OPT='NEW'"
    ]
```

Book and Fill Histograms

❖ Add it yourself to the existing code.

1. In the header file, you need to specify the ntuple variables:

```
#include "GaudiKernel/ITHistSvc.h"
#include "TH1.h"

ITHistSvc * m_thistSvc;
TH1F* my1DH;
```

2. In initialize():

```
/// Retreive a pointer to THistSvc
sc = service("THistSvc", m_thistSvc);
my1DH = new TH1F("ePt", "Electron Pt",100,0,200.);
sc = m_thistSvc->regHist("/FILE1/Electron", my1DH );
```

3. In execute(), somewhere after you fill the variable "nt_ZeeInvMass":

```
my1DH->Fill(e1_pt, 1.);
```

4. Histogram Persistency: in the jobOption file add the following:

```
theApp.HistogramPersistency = "ROOT"
from GaudiSvc.GaudiSvcConf import THistSvc
ServiceMgr += THistSvc()
ServiceMgr.THistSvc.Output = ["FILE1 DATAFILE='ZAnalysis_hist.root' OPT='NEW'"]
```