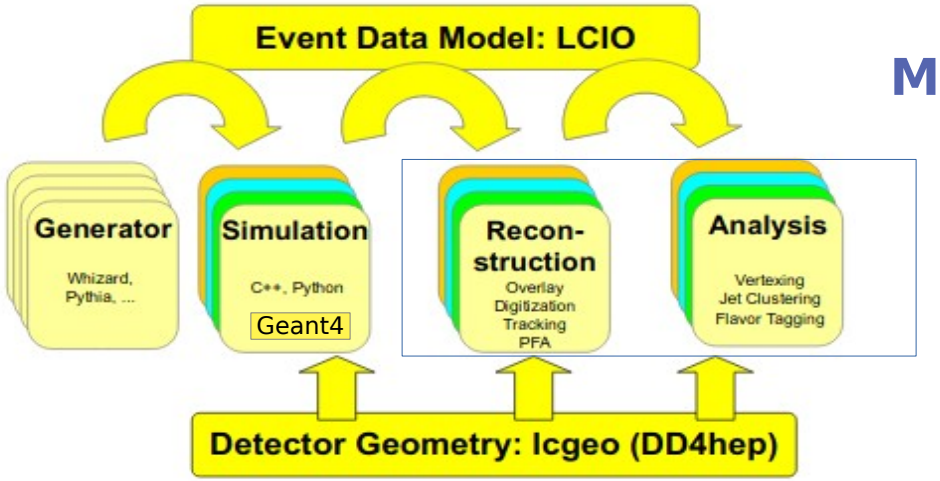
New Simulation Framework: event reconstruction

Workshop on Detector and Physics simulation at a Muon Collider
23-24 Jan 2020

Laura Buonincontri – University of Padova

ILCSoftware

ILCSoftware is the common software framework for Linear Collider detector studies



Marlin framework

- LCIO as event data model (unique data format for both simulation and analysis)
- DD4HEP for the detector geometry description

Marlin

- Marlin is a C++ software framework for analysis and reconstruction code based on LCIO
- The simulated file can be reconstructed by running the **steering file** :

  Marlin steering_file.xml

- Main idea: every computing task is implemented as a **processor** (module). There are different processors in the steering file with different goal: mix of signal and background, track reconstruction, jet clustering…
- The steering file defines the order in which the processors are called

Execute processor (in backup)

- As we will see in the Hands On section, the steering file is divided into three types of sections

1) **Execute Section**: the names of the processors which are to be executed are listed

```
<execute>
 <processor name="MyAIDAProcessor"/>
 <processor name="MyTestProcessor"/>
 <processor name="MyLCIOOutputProcessor"/>
</execute>
```

2) **Global Section**: the LCIO input files, the number of events to be run,... are specified

```
<global>
 <parameter name="LCIOInputFiles"> simjob.slcio </parameter>
 <parameter name="MaxRecordNumber" value="5001" />
 <parameter name="SupressCheck" value="false" />
 </global>
```

3) **Processor Section**: here all the available processors are configured, and parameters are set

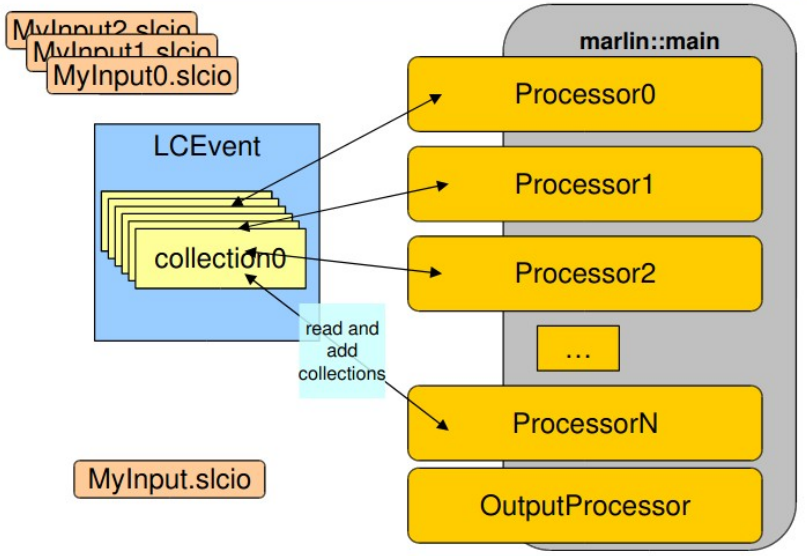
```
<processor name="MyAIDAProcessor" type="AIDAProcessor">
<!--Processor that handles AIDA files. Creates on directory per processor.
 <!-- compression of output file 0: false >0: true (default) -->
 <parameter name="Compress" type="int">1 </parameter>
 <!-- filename without extension-->
 <parameter name="FileName" type="string">aida_file </parameter>
 <!-- type of output file xml (default) or root ( only OpenScientist)-->
 <parameter name="FileType" type="string">xml </parameter>
</processor>
```

Processors: new collections creation

- The event data is stored in collections of different types

Input Collections types:

SimCalorimeterHit
SimTrackerHit
MCParticle



Output collections types:

Vertex
ReconstructedParticle
TrackerHit
Cluster
Track
…

- Processors analyze data in an event and create additional output collections that are added to the event

Access to Collections

- For each collection type, different kind of collections can be defined, and contain the output variables we are interested in:
- TrackerHit collection type defines the collections of the hits in the Tracker or the Vertex
- Each collection type is associated to a C++ class (see http://lcio.desy.de/v02-09/doc/doxygen_api/html/annotated.html)
- The public member functions of the class return the variables which describe the collections

virtual	**~TrackerHit** () Destructor.
virtual int	**getCellID0** () const =0 Same name as in **CalorimeterHit**, even though there are no 'cells' in this case.
virtual int	**getCellID1** () const =0 Same name as in **CalorimeterHit**, even though there are no 'cells' in this case Optional, check/set flag(LCIO::RTHBIT_ID1)==1.
virtual const double *	**getPosition** () const =0 The hit position in [mm].
virtual const **FloatVec** &	**getCovMatrix** () const =0 Covariance of the position (x,y,z), stored as lower triangle matrix.
virtual float	**getdEdx** () const =0 The dE/dx of the hit in [GeV].
virtual float	**getTime** () const =0 The time of the hit in [ns].
virtual int	**getType** () const =0

LCTuple processor creates a root file with these variables

Brief overview of processors

- **DDPlanarDigiProcessor**: creates TrackerHitPlane collections by smearing the simulated hits, and the TrackerHit-SimTrackHit relation collection (LCRelation) (Digitization for tracker and vertex detectors not implemented yet)
- **DDCaloDigi**: performs digitization of the calorimeter
- **Conformal Tracking, Refit**: Start from smeared hits and by mean of pattern recognition algorithms reconstruct tracks
- **DDPandoraPFANewProcessor**: for particle reconstruction. Combines information from tracks, calorimeter clusters and hits in the muon system.
- **LcfiplusProcessor and FastJetProcessor**: Primary and secondary vertex finding, jet reconstruction, and flavor tagging (to be optimized)

Other processors and commands

- **AIDAProcessor, LCTuple**: Input/Output processors
- **Overlay**: mix signal and background, here number of bunch train and bunch crossing can be set

USEFUL COMMANDS:

- To initialize the iLCSoft environment with a command like this:

```
source /data/ILCSoftware/init_ilcsoft.sh
```

- To run Marlin:

```
Marlin steering_file.xml
```

- Commands to check the events, see collections:

```
anajob file_name.slcio
```

```
dumpevent file_name.slcio n | less
```