30.09.2020

Muon Collider simulation package

Software framework overview

P. Andreetto [a], N. Bartosik [b], L. Buonincontri [a,d], M. Casarsa [c], A. Gianelle [a], S. Jindariani [e], D. Lucchesi [a,d], S. Pagan Griso [f], L. Sestini [a]

[a] INFN Padova, [b] INFN Torino, [c] INFN Trieste,
[d] University of Padova, [e] FNAL, [f] LBNL

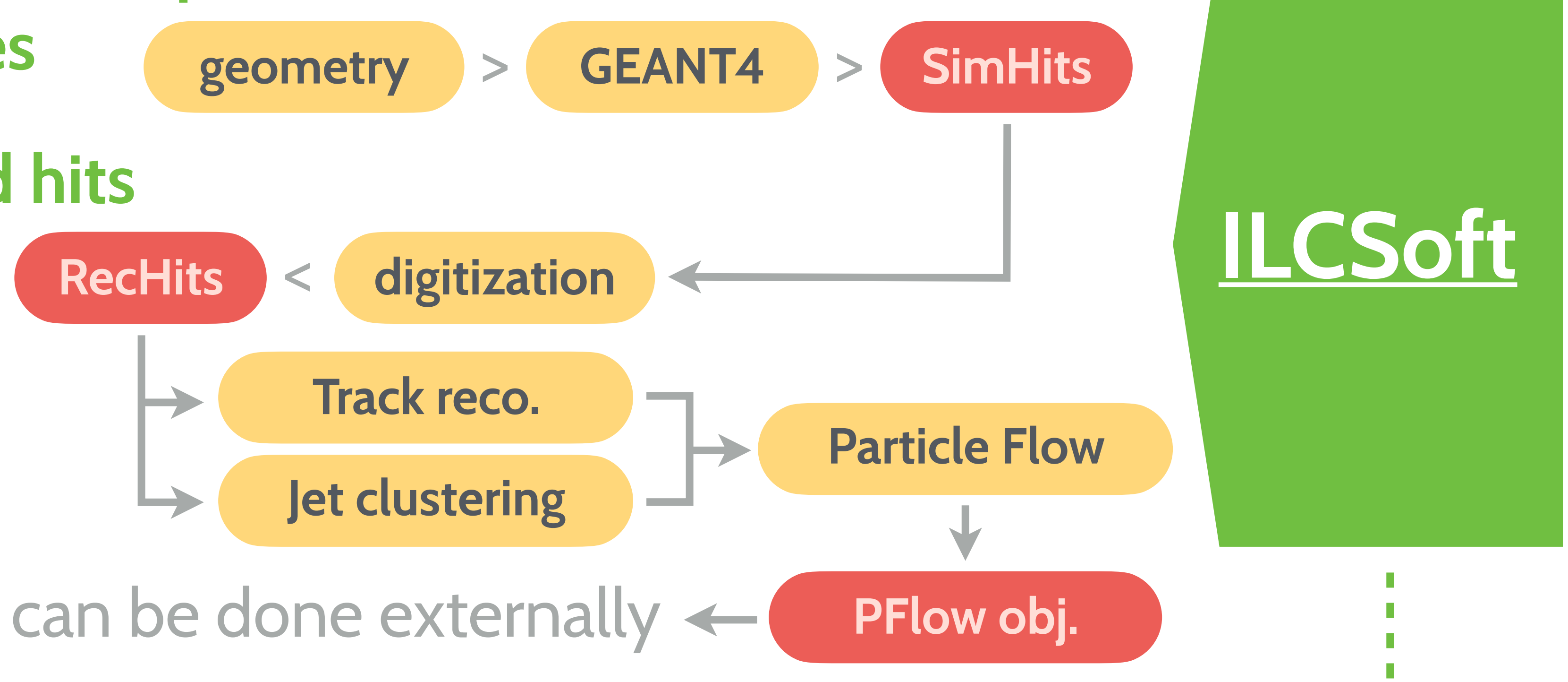
Full simulation strategy

BIB affects the detector performance in a non-trivial way

↳ we need full detector simulation to properly take into account all the effects

Key components of a physics analysis using full simulation:

1. generation of the process of interest (ME + PS) ← done externally
2. simulation of the detector response to the incoming particles
3. conversion of simulated hits to reconstructed hits
4. reconstruction of tracks/jets/particles
5. higher-level analysis ← can be done externally



All the simulation and reconstruction done within a single framework

A number of modifications and additions specific to the Muon Collider case are maintained in a separate public Muon Collider Software **repository**

Key components of ILCSoft

1. LCIO **[L**inear **C**ollider **I/O]**
   Provides consistent storage of event data (**MCParticles, SimHits/RecHits, higher-level and custom objects**) using the ***.slcio** file format
   - the most generic and basic part with no user intervention needed

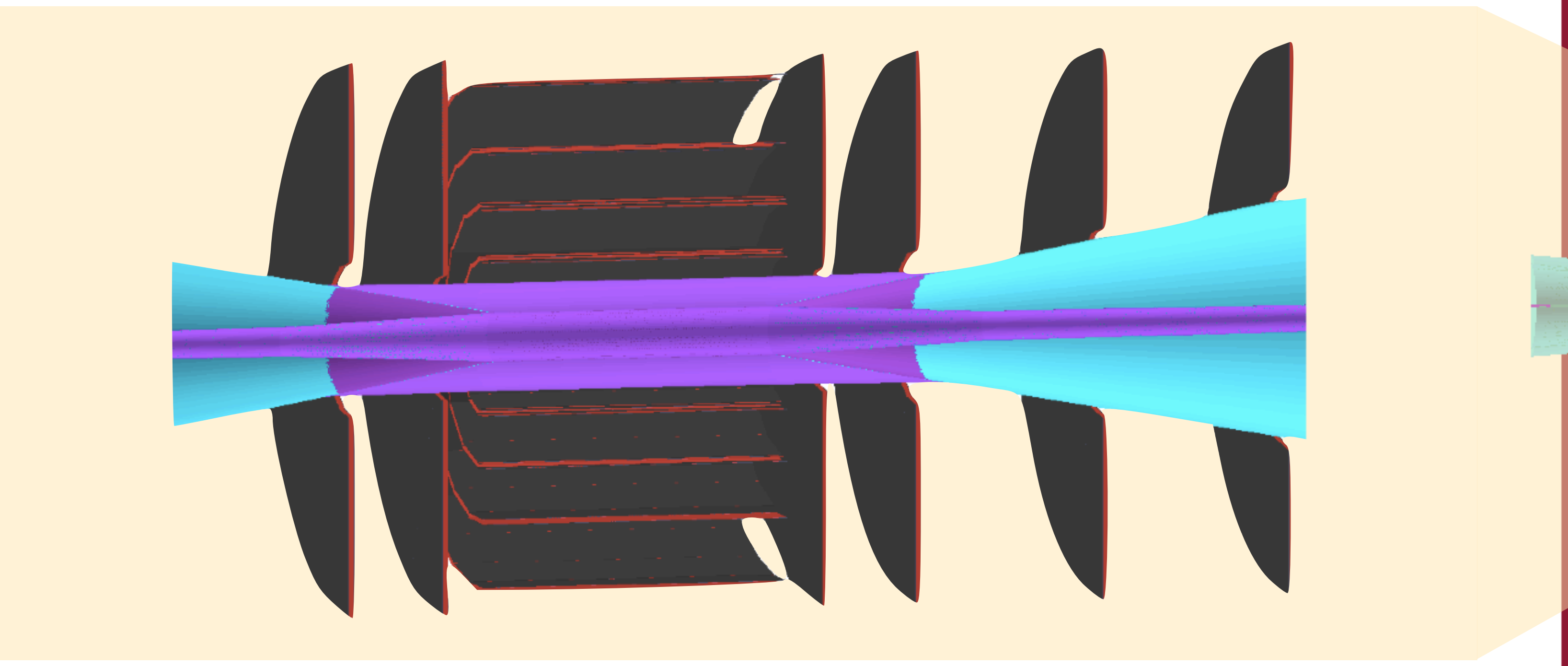
2. DD4hep **[D**etector **D**escription **for H**igh **E**nergy **P**hysics**]**
   Efficient and flexible detector geometry description with the interface to GEANT4 and simulation/reconstruction software
   - consists of C++ implementations of detector components assembled together via flexible XML configuration files
   - conceptual changes in the detector design require corresponding extensions of the underlying C++ code

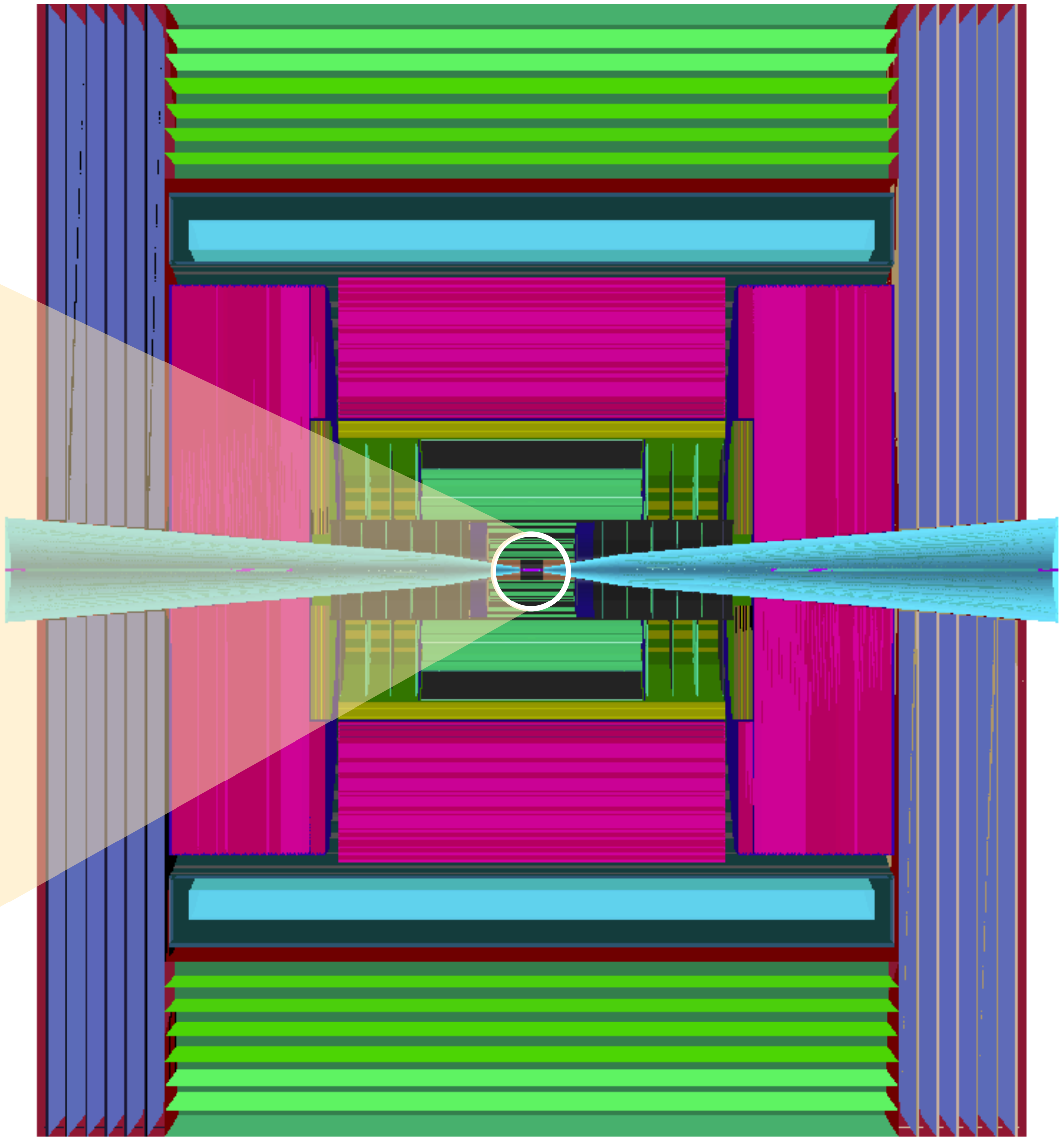
3. Marlin **[M**odular **A**nalysis & **R**econstruction for the **Lin**ear collider**]**
   Collection of processors for isolated tasks that can be chained into the necessary workflow by means of XML configuration files
   - everything after hits simulated by GEANT4 is handled by processors within the Marlin framework: ***digitization, track/jet reconstruction, b-tagging, etc.***

Detector geometry: derived from CLIC

Current geometry is derived from the CLIC detector with a few modifications:

- inserted BIB-absorbing tungsten nozzles developed by MAP
- inner openings of endcap detectors increased to fit the nozzles
- optimised layout of the Vertex detector to reduce occupancy at the tips of the nozzles
- Vertex segmentation along the beamline





Using the forked version of lcgeo to support the modified geometry components:

- ZSegmentedPlanarTracker, GenericCalEndcap_o2_v01

Flexible and modular configuration via XML:

CLIC_o3_v14.xml

```xml
<include ref="Beampipe_o1_v01_02.xml"/>
<include ref="Nozzle_10deg_v0.xml"/>

<include ref="Vertex_o2_v06_01.xml"/>

<include ref="InnerTracker_o2_v06_01.xml"/>
<include ref="OuterTracker_o2_v06_01.xml"/>
```

Vertex_o2_v06_01.xml

constants

```xml
<define>
    <constant name="VertexBarrel_zmax" value="65*mm"/>
    <constant name="VertexBarrel_nmodules" value="5"/>
    <constant name="VertexBarrel_r1" value="30*mm"/>
    <constant name="VertexBarrel_r2" value="51*mm"/>
    <constant name="VertexBarrel_r3" value="74*mm"/>
    <constant name="VertexBarrel_r4" value="102*mm"/>

    <constant name="VertexBarrel_Sensitive_Thickness"   value="50e-03*mm"/>
    <constant name="VertexBarrel_Support_Thickness"     value="140e-03*mm"/>
    <constant name="VertexBarrel_DoubleLayer_Gap"       value="2.0*mm"/>

    <constant name="VertexBarrel_Layer1_width" value="13*mm"/>
    <constant name="VertexBarrel_Layer2_width" value="23*mm"/>
    <constant name="VertexBarrel_Layer3_width" value="24*mm"/>
    <constant name="VertexBarrel_Layer4_width" value="24*mm"/>
```

readout collections

```xml
<readouts>
    <readout name="VertexBarrelCollection">
        <id>${GlobalTrackerReadoutID}</id>
    </readout>
    <readout name="VertexEndcapCollection">
        <id>${GlobalTrackerReadoutID}</id>
    </readout>
</readouts>
```

composition of the detector layer by layer

```xml
<detectors>
    <detector name="VertexBarrel" type="ZSegmentedPlanarTracker" vis="VXDVis" id="DetID_VXD_Barrel" readout="VertexBarrelCollection"  region="
    VertexBarrelRegion">

        <type_flags type=" DetType_TRACKER + DetType_PIXEL + DetType_VERTEX + DetType_BARREL"/>

        <layer nLadders="VertexBarrel_Layer1_Staves" phi0="0" id="0">
            <ladder    distance="VertexBarrel_r1" thickness="VertexBarrel_Support_Thickness" width="VertexBarrel_Layer1_width" length="VertexBarrel_zmax"
            offset="VertexBarrel_Layer1_offset"    material="Silicon"  vis="SiVertexPassiveVis"/>
            <sensitive nmodules="VertexBarrel_nmodules" distance="VertexBarrel_r1+VertexBarrel_Support_Thickness" thickness="VertexBarrel_Sensitive_Thickness"
            width="VertexBarrel_Layer1_width" length="VertexBarrel_zmax" offset="VertexBarrel_Layer1_offset" material="Silicon" vis="SiVertexSensitiveVis" />
        </layer>
        <layer nLadders="VertexBarrel_Layer1_Staves" phi0="0" id="1">
            <sensitive nmodules="VertexBarrel_nmodules" distance="
            VertexBarrel_r1+VertexBarrel_Support_Thickness+VertexBarrel_Sensitive_Thickness+VertexBarrel_DoubleLayer_Gap" thickness="
            VertexBarrel_Sensitive_Thickness" width="VertexBarrel_Layer1_width" length="VertexBarrel_zmax" offset="VertexBarrel_Layer1_offset" material="
            Silicon" vis="SiVertexSensitiveVis" />
            <ladder    distance="
            VertexBarrel_r1+VertexBarrel_Support_Thickness+VertexBarrel_Sensitive_Thickness+VertexBarrel_DoubleLayer_Gap+VertexBarrel_Sensitive_Thickness"
            thickness="VertexBarrel_Support_Thickness" width="VertexBarrel_Layer1_width" length="VertexBarrel_zmax" offset="VertexBarrel_Layer1_offset"
            material="Silicon"  vis="SiVertexPassiveVis" />
        </layer>
```

Many parameters can be changed just by editing the XML file:

- # of layers, layer dimension/position, sensor width/thickness, etc.

Geometry implementation: dd4hep

There is no magic. XML is translated to actual geometry objects via C++ classes

- new class was created in **lcgeo** to support segmentation along Z axis

`ZSegmentedPlanarTracker_geo.cpp`

materials of the support & sensitive layers

```
Material supp_mat     = theDetector.material( supp_matS ) ;
Material sens_mat     = theDetector.material( sens_matS ) ;
```

logical volumes for the support & sensors

```
// Creating the logical volumes for the support and sensors
Box supp_box( supp_thickness / 2., supp_width / 2., supp_zhalf );
Box sens_box( sens_thickness/2., sens_width/2., sens_modlength/2.0 - 1e-03 * dd4hep::um );

Volume supp_vol( layername+"_support", supp_box, supp_mat  );
Volume sens_vol( layername+"_sensor", sens_box, sens_mat );
Assembly ladder_assembly( layername + "_ladder" );

sens_vol.setAttributes( theDetector, x_det.regionStr(), x_det.limitsStr(), sens_vis );
supp_vol.setAttributes( theDetector, x_det.regionStr(), x_det.limitsStr(), supp_vis );

sens_vol.setSensitiveDetector(sens);
```

sensitive surface attached to the sensor volume

```
// -------- create a measurement plane for the tracking surface attached to the sensitive volume -----
Vector3D u( 0. , 1. , 0. ) ;
Vector3D v( 0. , 0. , 1. ) ;
Vector3D n( 1. , 0. , 0. ) ;
Vector3D o( 0. , 0. , 0. ) ;

// compute the inner and outer thicknesses that need to be assigned to the tracking surf
// depending on wether the support is above or below the sensor
double inner_thickness = ( sens_distance > supp_distance ?  ( sens_distance - supp_dista
double outer_thickness = ( sens_distance > supp_distance ?    sens_thickness/2  :  ( sup

SurfaceType type( SurfaceType::Sensitive ) ;

if( isStripDetector )
  type.setProperty( SurfaceType::Measurement1D , true ) ;

VolPlane surf( sens_vol , type , inner_thickness , outer_thickness , u, v, n, o ) ;


// Calculating Sensor placements inside a Ladder sensor envelope
std::vector<PlacedVolume> pv_sensor( sens_nmodules );
for(int s=0; s<sens_nmodules; ++s) {
  double zshift = -0.5*(sens_nmodules*sens_modlength) + (s+0.5)*sens_modlength;
  pv = ladder_assembly.placeVolume( sens_vol, Position(0., 0., zshift) );
  pv.addPhysVolID("sensor", s );
  pv_sensor.at(s) = pv;
}
```

calculating rotation of each ladder

```
//--------- loop over ladders ---------------------------

for(int j=0; j<nLadders; ++j) {

  double phi = phi0 + j * dphi;
  RotationZYX rot(phi, 0, 0);

  // --- place support -----
  double lthick = supp_thickness ;
  double radius = supp_distance ;
  double offset = supp_offset ;

  layer_assembly.placeVolume( supp_vol, Transform3D( rot, Position(( radius + lthick/2. ) * cos(phi)  - offset * sin(phi),
                                                                   ( radius + lthick/2. ) * sin(phi)  + offset * cos(phi),
                                                                   0. ) ));

  // Creating the Ladder to which sensors will be assigned
  std::string laddername = layername + _toString(j,"_ladder%d");
  DetElement ladderDE( layerDE ,  laddername , x_det.id() );

  //--------- Placing sensors with relative shifts along Z inside the sensitive envelope -------------------------
  for(int s=0; s<sens_nmodules; ++s) {

    std::string sensorname = laddername + _toString(s, "_sensor%d");
    DetElement   sensorDE( ladderDE ,  sensorname , x_det.id() );
    sensorDE.setPlacement( pv_sensor.at(s) );

    volSurfaceList( sensorDE )->push_back( surf );
```

placing sensors inside ladder

All this is inside a loop over layers

Simulation/Reconstruction tools

The two most important commands: **ddsim** and **Marlin**

2. simulation of the detector response
   to the incoming particles

geometry > GEANT4 > SimHits > FILE_sim.slcio

from MC generator → `stdhep` MC file

PGun, BIB → `lcio` MC file

MC file

```
anajob <file.slcio>

///////////////////////////////////
EVENT: 3
RUN: 0
DETECTOR: CLIC_o3_v14_mod4
COLLECTIONS: (see below)
///////////////////////////////////

---------------------------------------------------------------------------
COLLECTION NAME               COLLECTION TYPE          NUMBER OF ELEMENTS
===========================================================================
ECalBarrelCollection          SimCalorimeterHit            28377
ECalEndcapCollection          SimCalorimeterHit            14854
HCalBarrelCollection          SimCalorimeterHit            33832
HCalEndcapCollection          SimCalorimeterHit            32719
HCalRingCollection            SimCalorimeterHit             1761
InnerTrackerBarrelCollection  SimTrackerHit                 2705
InnerTrackerEndcapCollection  SimTrackerHit                 1829
MCParticle                    MCParticle                   64426
OuterTrackerBarrelCollection  SimTrackerHit                 2084
OuterTrackerEndcapCollection  SimTrackerHit                 1414
VertexBarrelCollection        SimTrackerHit                 4950
VertexEndcapCollection        SimTrackerHit                 2365
YokeBarrelCollection          SimCalorimeterHit             3320
YokeEndcapCollection          SimCalorimeterHit             3613
---------------------------------------------------------------------------
```

Simulation/Reconstruction tools

The two most important commands: ddsim and Marlin

2. simulation of the detector response to the incoming particles

geometry > GEANT4 > SimHits > FILE_sim.slcio

from MC generator → `stdhep` MC file

PGun, BIB → `lcio` MC file

MC file

must be used consistently throughout the chain

geometry

3. conversion of simulated hits to reconstructed hits

RecHits < Digitization

4. reconstruction of tracks/jets/particles

Track reco.

Jet clustering

Particle Flow

PFlow obj.

FILE_reco.slcio

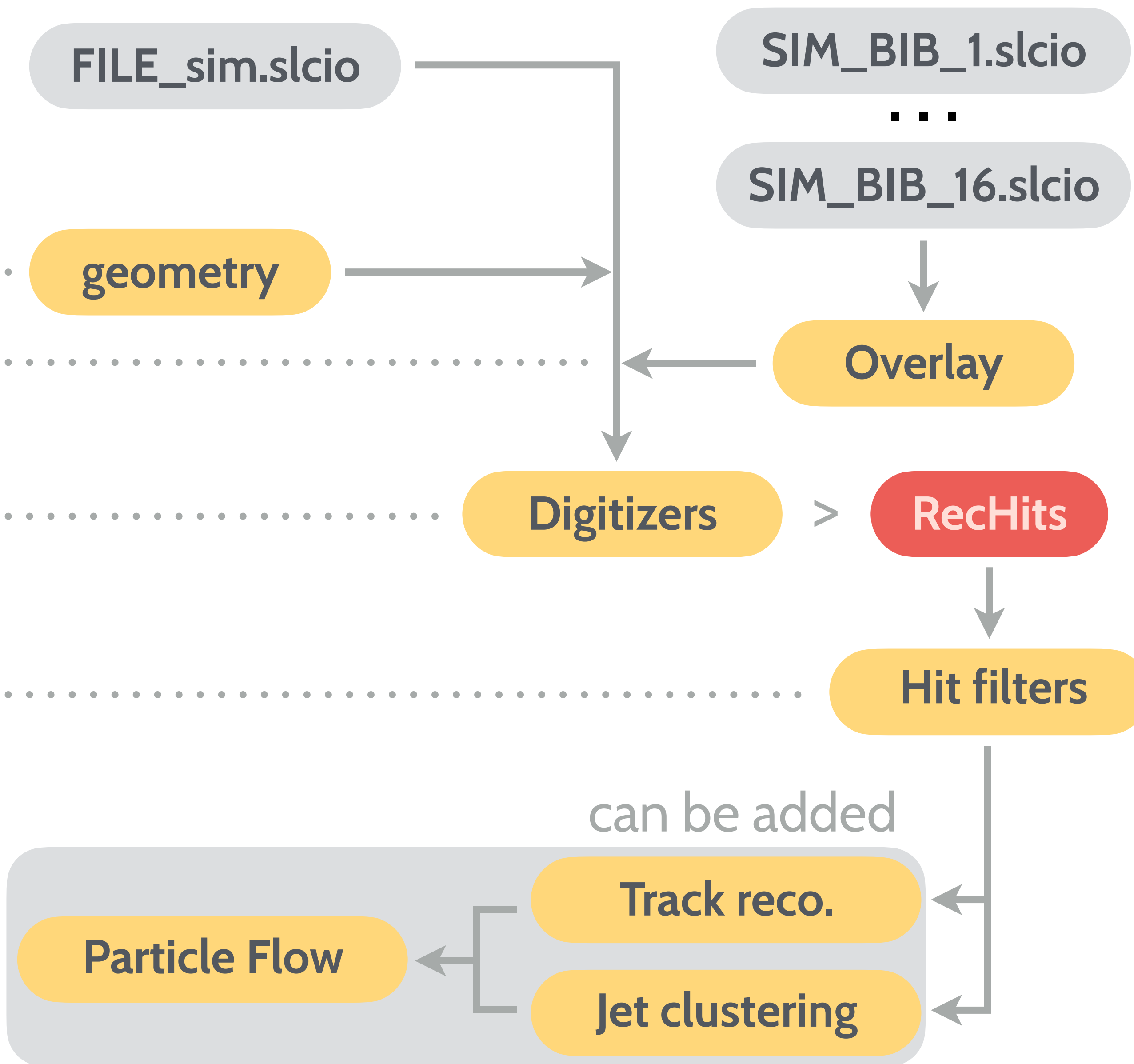
Every Operation **performed by a Marlin processor**
set up in the XML configuration file of the Marlin job

The full chain of Marlin processors can be split into smaller steps, writing intermediate output to new files, and reading them back in the following steps

Basic Marlin workflow

A Marlin job is configured through an XML file: chain of individual processors

```xml
<execute>

  <!-- ========== Setup ========== -->
  <processor name="MyAIDAProcessor"/>
  <processor name="EventNumber" />
  <processor name="Config" />

  <!-- ========== Geometry initialization ========== -->
  <processor name="InitDD4hep_mod4"/>

  <!-- ========== Overlay ========== -->
  <processor name="OverlayBIB"/>

  <!-- ========== Tracker Digitization ========== -->
  <processor name="VXDBDigitiser"/>
  <processor name="VXDEDigitiser"/>
  <processor name="IBDigitizer"/>
  <processor name="IEDigitizer"/>
  <processor name="OBDigitizer"/>
  <processor name="OEDigitizer"/>

  <processor name="FilterDL_VXDB" />
  <processor name="FilterDL_VXDE" />

  <!-- ========== Output ========== -->
  <processor name="Output_REC"/>

</execute>
```



Processor class

```xml
<processor name="VXDBDigitiser" type="DDPlanarDigiProcessor">
  <parameter name="SubDetectorName" type="string"> Vertex </parameter>
  <parameter name="IsStrip" type="bool">false </parameter>
  <parameter name="ResolutionU" type="float"> 0.005 </parameter>
  <parameter name="ResolutionV" type="float"> 0.005 </parameter>
  <parameter name="SimTrackHitCollectionName" type="string" lcioInType="SimTrackerHit"> VertexBarrelCollection
  <parameter name="SimTrkHitRelCollection" type="string" lcioOutType="LCRelation"> VXDBTrackerHitsRelations </p
  <parameter name="TrackerHitCollectionName" type="string" lcioOutType="TrackerHitPlane"> VXDBTrackerHits </par
  <parameter name="ResolutionT" type="FloatVec"> 0.05  </parameter>
</processor>
```

Each processor's instance has its own block with corresponding configuration parameters

BIB performance impact

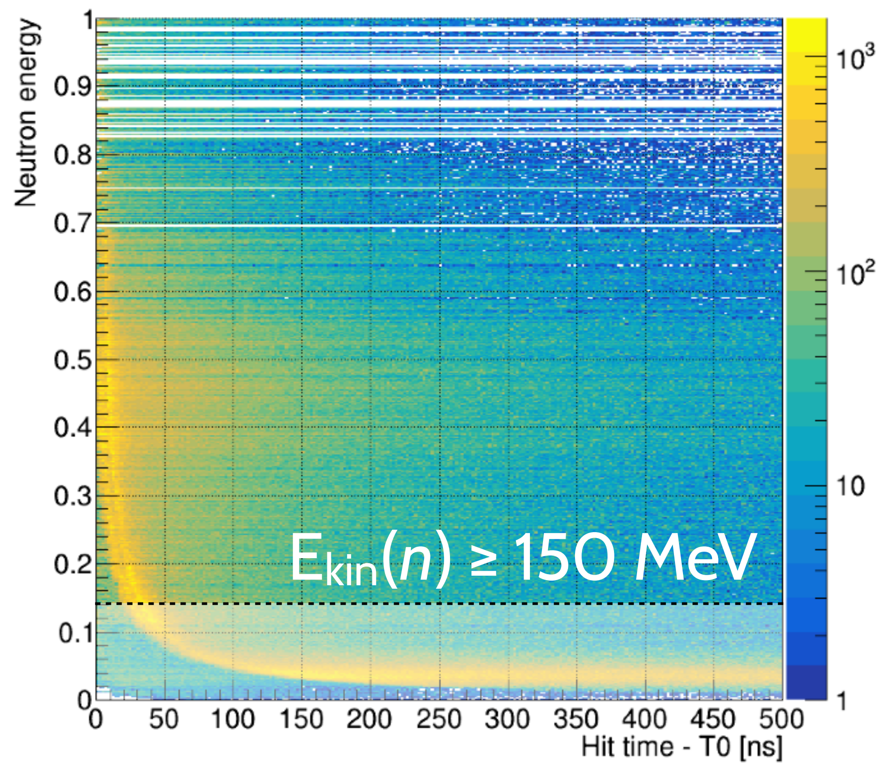
We start with 380M particles from μ⁺ and μ⁻ beams in a single bunch crossing

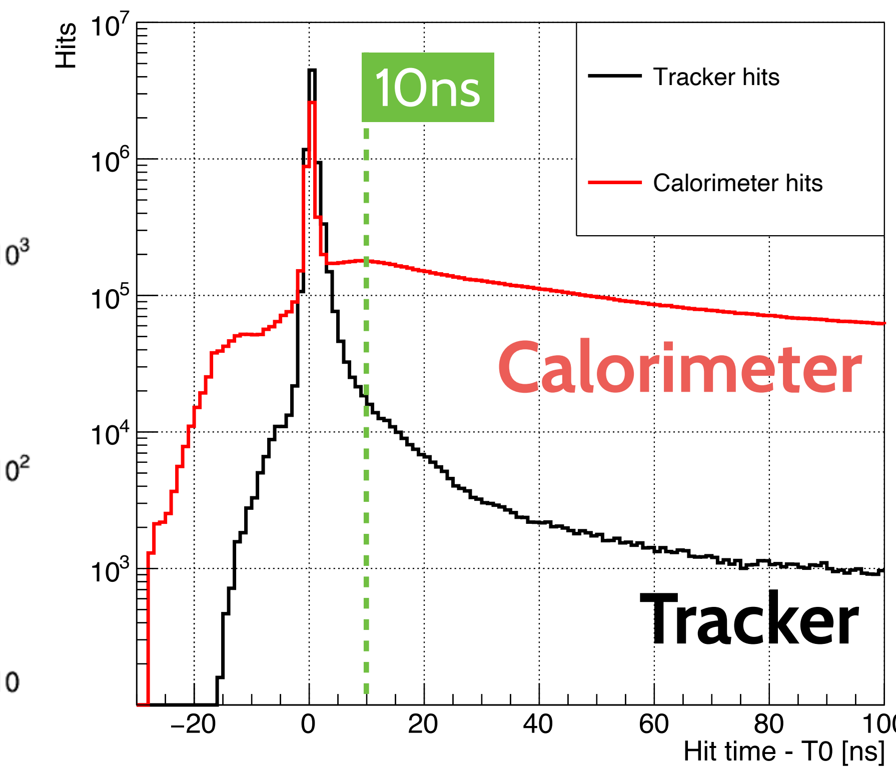
- one has to be mindful about performance at any step of the workflow

Only hits in the short readout time window are relevant [~0.1-10 ns]

Slow neutrons create calorimeter hits very late after the bunch crossing

	# of particles	*CPU time*
all	380M	380 h
timing t < 25ns	98M (26%)	60 h (18%)
$E_{kin}(n)$ cut	78M (20%)	25 h (6.6%)
lower n precision	78M (20%)	3 h (0.7%)





Significant speed up of the BIB simulation by skipping irrelevant particles at the earliest stage possible →

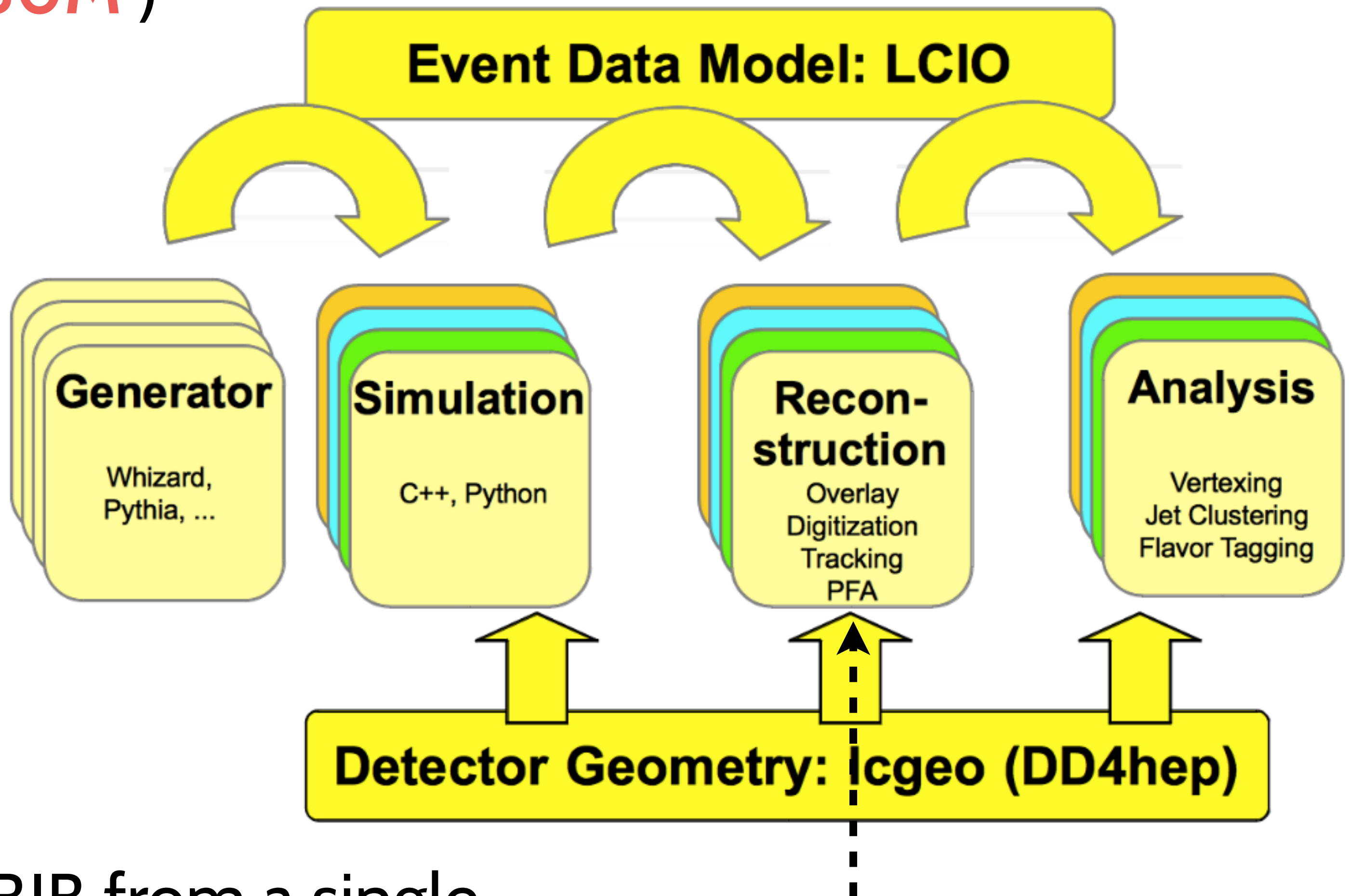
Adding BIB to the event

In a specific physics analysis the process of interest is generated by a dedicated event generator (outside the **ILCSoft** framework)

- Simulation step is handled by **DD4hep**: `ddsim --steeringFile clic_steer.py`
- **stdhep** or **slcio** formats for are supported as input

BIB particles can be added in either of the two places:

- **Simulation** (as **MCParticles** ~380M)
  - fluctuations in **GEANT4** simulation are not significant
  - takes a lot of CPU time
- **Reconstruction** (as **SimHits**)
  - entering at **digitization** step
  - much more efficient



Optimal solution for now:

- perform **GEANT4** simulation of BIB from a single bunch crossing and overlay **SimHits** on every signal event (Overlay processor)

Treatment of timing

Timing is very important at a Muon Collider: BIB arrives later wrt BX

- we remove particles/hits outside the time window of interest to save CPU time

Looking at the Vertex detector hits:

- assuming $\sigma_t = 50\text{ps}$
- interested in RecHits with
  $-3\sigma_t < t_{hit} < +3\sigma_t$

SimHits > Overlay > Digitization > RecHits

Overlay:
1. calculate $t^* = t - TOF$
2. add hits to the event if
   $-6\sigma_t < t^* < +6\sigma_t$

Digitization:
1. smear position by $\sigma_{U/V}$
2. smear time by σ_t
3. correct time by TOF
4. keep hits within $\pm 3\sigma_t$

300ps

OverlayBIB processor

```
<parameter name="Collection_IntegrationTimes"
  VertexBarrelCollection        0.3
  VertexEndcapCollection        0.3

  InnerTrackerBarrelCollection  0.6
  InnerTrackerEndcapCollection  0.6

  OuterTrackerBarrelCollection  0.6
  OuterTrackerEndcapCollection  0.6

</parameter>
```

150ps

VXDBDigitiser processor

```
<parameter name="ResolutionT" type="FloatVec"> 0.05  </parameter>
<parameter name="UseTimeWindow" type="bool"> true </parameter>
<parameter name="CorrectTimesForPropagation" type="bool" value="true"/>
<parameter name="TimeWindowMin" type="float"> -0.15 </parameter>
<parameter name="TimeWindowMax" type="float"> 0.15 </parameter>
```

Calorimeter showers take more time to develop

↳ timing windows are asymmetric
$-1\text{ns} < t_{hit} < 10\text{ns}$

```
<parameter name="UseEcalTiming" type="int">1 </parameter>
<parameter name="ECALCorrectTimesForPropagation" type="int">1 </parameter>
<parameter name="ECALTimeWindowMin" type="float">-1 </parameter>
<parameter name="ECALBarrelTimeWindowMax" type="float">10 </parameter>
```

Useful commands

Show SLCIO file contents:

- prints parameters used to run the job
- prints collection stats. for each event

`anajob <file.slcio>`

```
///////////////////////////////////
EVENT: 3
RUN: 0
DETECTOR: CLIC_o3_v14_mod4
COLLECTIONS: (see below)
///////////////////////////////////

---------------------------------------------------------------------------
COLLECTION NAME               COLLECTION TYPE          NUMBER OF ELEMENTS
===========================================================================
ECalBarrelCollection          SimCalorimeterHit            28377
ECalEndcapCollection          SimCalorimeterHit            14854
HCalBarrelCollection          SimCalorimeterHit            33832
HCalEndcapCollection          SimCalorimeterHit            32719
HCalRingCollection            SimCalorimeterHit             1761
InnerTrackerBarrelCollection  SimTrackerHit                 2705
InnerTrackerEndcapCollection  SimTrackerHit                 1829
MCParticle                    MCParticle                   64426
```

Dump event contents in text format:

- set LCIO_READ_COL_NAMES env. variable to only dump specific collections of interest

`dumpevent <file.slcio> <event #>`

```
============================================================================
        Event  : 1 - run:  0 - timestamp 0 - weight 1
============================================================================
 date:       01.01.1970  00:00:00.000000000
 detector : CLIC_o3_v14_mod4
 event parameters:

 collection name : VXDBTrackerHits
 parameters:

--------------- print out of TrackerHitPlane collection ---------------

  flag:  0x80000000
 parameter CellIDEncoding [string]: system:5,side:-2,layer:6,module:11,sensor:8,
     LCIO::THBIT_BARREL : 1

 [   id   ] |cellId0 |cellId1 | position (x,y,z)            | time    |[type]|[qual.]|
EDep    |EDepError|   du    |   dv    |q|  u (theta, phi)    |  v (theta, phi)
------------|--------|--------|-----------------------------|---------|------|------|--
-------|---------|---------|---------|-|------------------|-------------------|
 [00003271]  |33660929|00000000|+1.33e+01,-2.71e+01,-6.72e+00|+8.54e-02|[0000]|[0000]|
+1.36e-05|+0.00e+00|+5.00e-03|+5.00e-03|+0|+1.57e+00,+3.93e-01|+0.00e+00,+0.00e+00|
    id-fields: (system:1,side:0,layer:0,module:13,sensor:2)
```

Count number of events in SLCIO file(s)

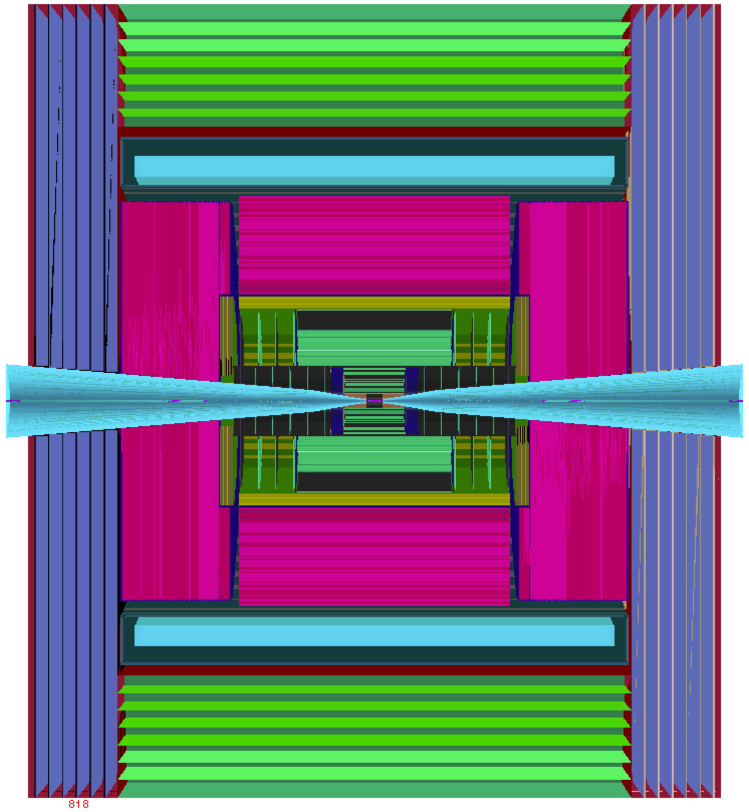
`lcio_event_counter [<file.slcio>]`

Display collection stats. in SLCIO file(s)

`lcio_check_col_elements <collection> [<file.slcio>]`

Other lcio_* commands also available

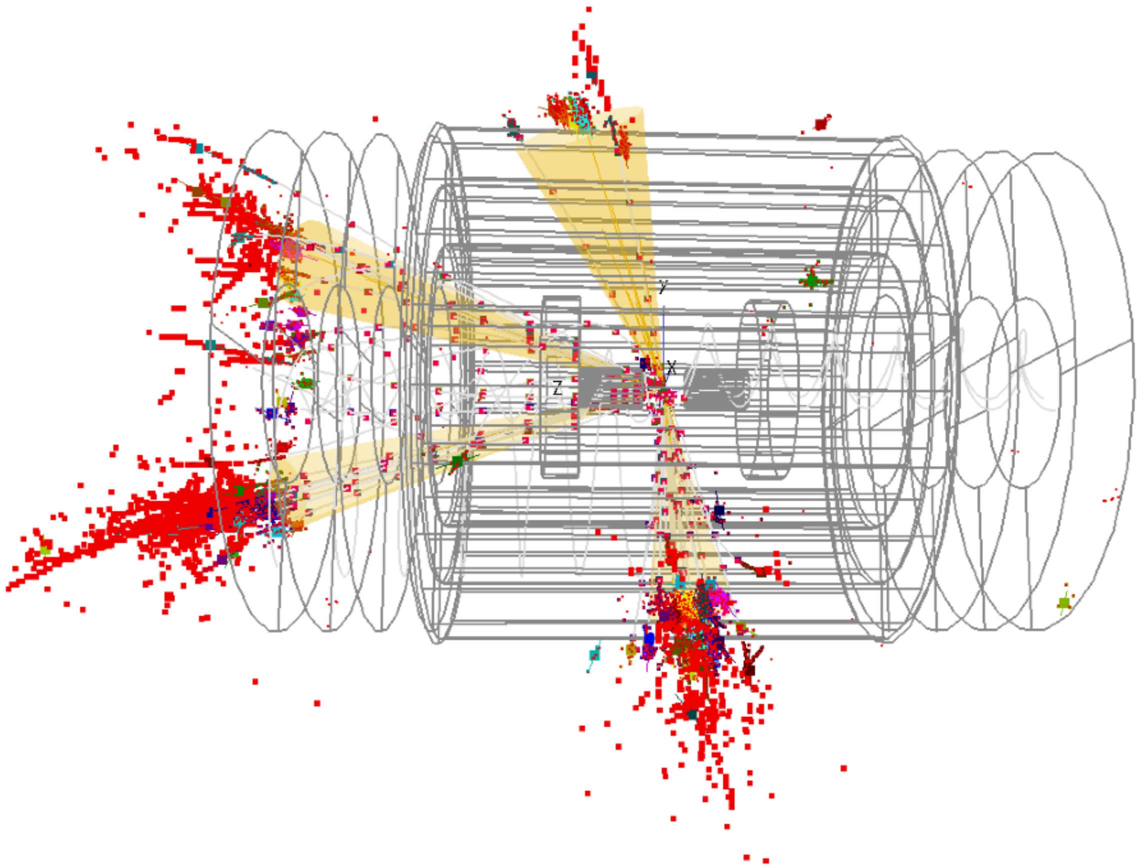
Useful commands



Visualise a geometry in full detail

```
teveDisplay -compact <geometry.xml>
```

- takes long to load, but useful for examining the precise layout
- comment unneeded subdetectors in the XML to make it faster



Visualise an event from SLCIO file with simplified geometry rendering

```
ced2go -d <geometry.xml> <file.slcio>
```

- loads faster, emphasis on examining the event content

Additional materials

DD4hep User's Manual

DD4hep and Shareable Detector Geometry Description A.Sailer

iLCSoft tutorial F.Gaede

Analysis in python: pyLCIO examples CLIC TWiki

- particularly useful to play around with slcio files using Python (*or Jupyter*)

BACKUP

BIB particles: from MARS15

BIB provided my MAP as a text file: list of particles from MARS15 simulation

- each line represents a single particle crossing the outer detector/nozzle surface
- only a fraction of all particles actually included
  - each particle has an associated weight to calculate the proper normalisation

Dedicated [C++ macro] converts text files to slcio files, compatible with **ILCSoft**

- **1 line → 1 MCParticle** with corresponding position, momentum, pdgId, etc.
- \+ N copies of the particle randomly distributed in φ to account for the weight
- **particles split in multiple events** (default: 2000 lines/event → **2993 events**)
  - can use a fraction of all particles in the simulation (< 2993 events)
  - to run the GEANT4 simulation in parallel over fixed batches of events

Possible to exclude particles based on certain selection criteria

- time of arrival of the particle
- energy of the particle if it's a neutron *(relevant for performance)*