Monte Carlo simulations in High Energy Physics

Marco Giacalone

06.11.2025

Lecture 1

1st ALICE Experiment and Heavy-Ion Physics School - Shanghai

Photo: @zhangkaiyv on unsplash.com

Monte Carlo: history

- Monte Carlo methods refer to a wide class of computational methods based on random sampling to obtain numerical approximations

- Stanisław Ulam is considered the father of the modern version of Monte Carlo methods → while working on nuclear weapons (Los Alamos)

↓

Name comes from his uncle gambling habits at the famous Monaco casino

Principality of Monaco

Monte Carlo Casino

Monte Carlo: the core idea

p(x) = probability density function (PDF)

Monte Carlo methods use probability to calculate estimates
Example:

$$p(x) = \begin{cases} \frac{1}{b-a} & \text{for } x \in [a,b] \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{I} = \int_a^b f(x)dx \longrightarrow I = (b-a)\int_a^b p(x)f(x)dx$$

In probability theory, with a random variable X following p(x):

$$I = (b-a)\mathbb{E}[f(X)]$$

Using MC, our estimate is:

$$S_N = \underbrace{\frac{1}{N}\sum_{i=1}^{N} f(X_i)}_{\text{MCestimate}} \longrightarrow \text{with } N \to \infty \longrightarrow I = \frac{b-a}{N}\sum_{i=1}^{N} f(X_i) + O\left(\frac{1}{\sqrt{N}}\right)$$

Central Limit Theorem

Law of Large Numbers

Monte Carlo: the core idea

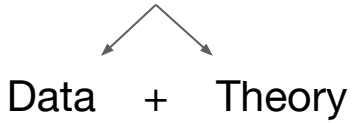
- The law of large numbers guarantees that the sampled average converges to the integral expected value with *N* samples→∞
- Uncertainty decreases as $1/\sqrt{N}$ → convergence rate quite slow for 1D or 2D integrals, but <u>no dimension scaling!</u>

$$I = V \int_{\Omega} p\left(x_1, \ldots, x_n\right) f\left(x_1, \ldots, x_n\right) dx_1 \ldots dx_n$$
$$= \frac{V}{N} \sum_{i=1}^{N} f\left(X_{1i}, \ldots, X_{ni}\right) + O\left(\frac{1}{\sqrt{N}}\right)$$

Monte Carlo has a significant advantage compared to other integral calculation methods (such as trapezoidal rule scaling: $N^{-2/d}$)

High energy physics application

- In MC the system is described by modeled PDFs → equations solutions are not analytical



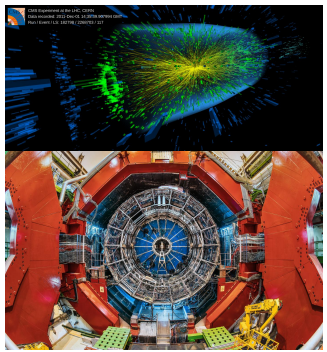
- In particle physics MC is used for:
  - **Event Selection** ⇒ algorithms are tested simulating various scenarios and conditions, isolating for example rare processes and signals
  - **Background Estimation** ⇒ detailed processes modelling and understanding
  - **Efficiency Assessment** ⇒ different components of the experimental setup, including the detector itself and the algorithms used for data analysis, are evaluated ⇒ understanding biases or limitations in the detector data
  - **Systematic Uncertainty Evaluation** ⇒ quantification of systematic uncertainties in physics measurements ⇒ changing input parameters and conditions in MC simulations make it possible to identify and quantify sources of systematic uncertainty
  - **Data Validation and Calibration** ⇒ comparison with experimental data let us study the accuracy of measurements and detectors can be calibrated accordingly making data more accurate

- Particles processes and detectors (geometry + response) are studied: **HEP MC simulation**

Classical pipeline in a HEP experiment: Experimental Data

~TB/s

Real particle collisions



sensor data

Reconstruction

Takes sensor data and reconstructs state of particles right after collisions

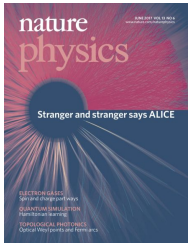
PetaBytes

AOD data

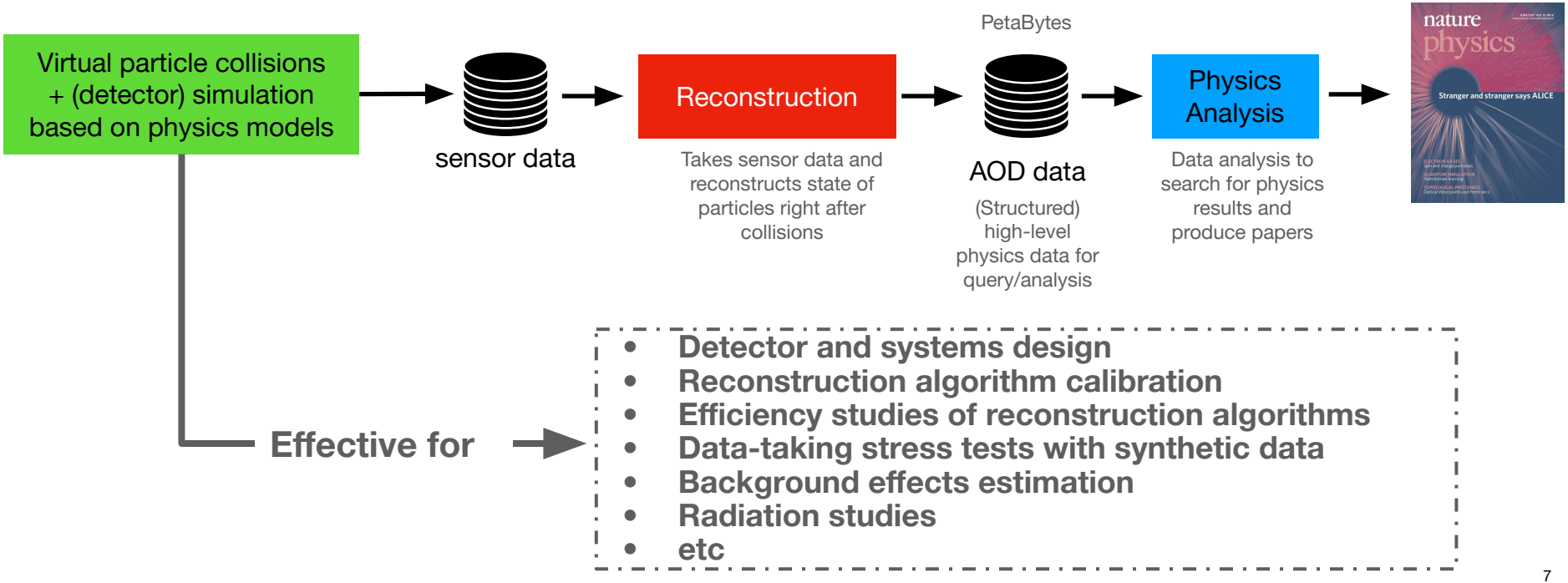
(Structured) high-level physics data for query/analysis

Physics Analysis

Data analysis to search for physics results and produce papers

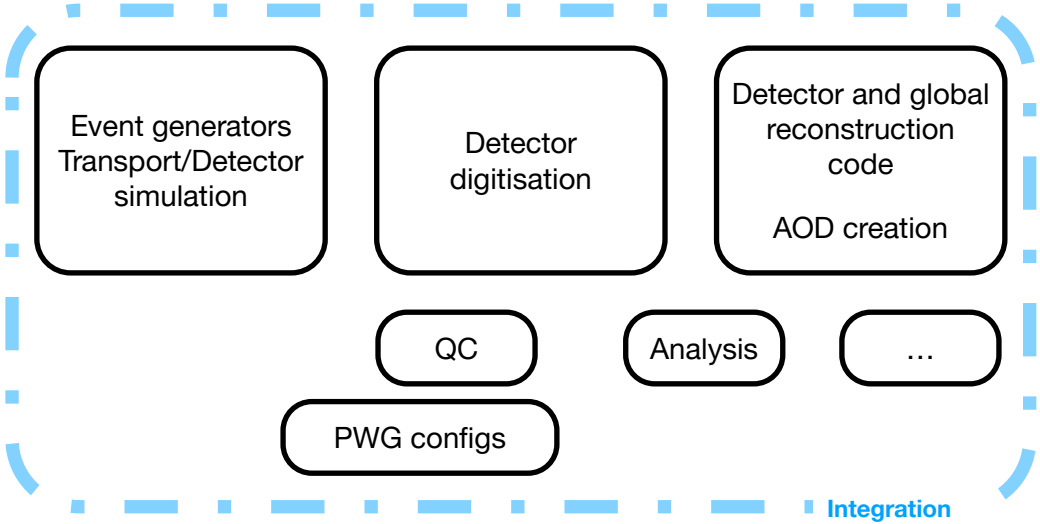


Classical pipeline in a HEP experiment: Simulation



The ALICE Run3 simulation ecosystem

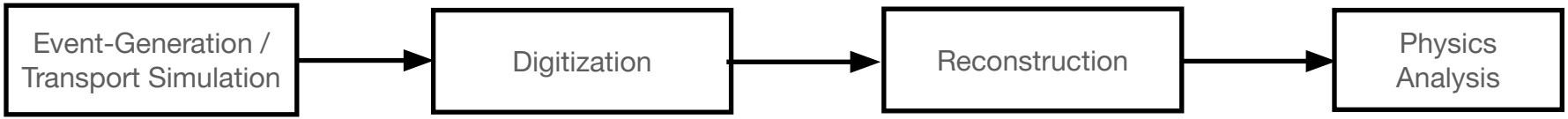
- Simulation ecosystem comprised of various components
- Core simulation part:
  - Event generation
  - Transport simulation
  - Digitization
- In addition, MC workflows may exercise all of
  - Reconstruction, QC, Analysis, etc.
- Individual parts maintained in O2 and O2Physics repositories



Integration and configuration of all parts into coherent workflows, done with:

- O2DPG repository (mainly for physics studies on GRID)
- full-system-test (mainly for data taking oriented simulations)

Data products in simulation pipeline

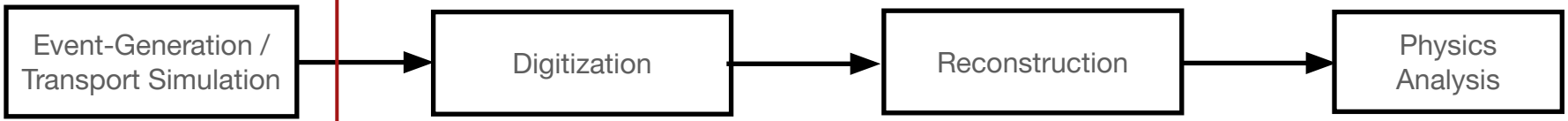


- Geometry file
- Kinematics file
- Detector response files (hits)

- Digits == detector sub-timeframes
- Comparable or close to raw detector output

- Global reconstructed tracks
- Primary + Secondary Vertices
- etc.
- **AOD (analysis object data)**

Data products in simulation pipeline



- Geometry file
- Kinematics file
- Detector response files (hits)

Lectures highlight

- Digits == detector sub-timeframes
- Comparable or close to raw detector output

- Global reconstructed tracks
- Primary + Secondary Vertices
- etc.
- **AOD (analysis object data)**

Event generation

- ALICE collects a series of event generators in the **AliGenO2** package
- Current available generators:
  - DPMJET
  - POWHEG
  - PYTHIA 8
  - ThePEG
  - SHERPA
  - JETSCAPE
  - CRMC
  - EPOS4
  - EPOS4HQ
  - STARlight
  - Upcgen
  - Graniitti
  - nOOn
  - EvtGen
  - Herwig7
- More in the future (based on users requests)

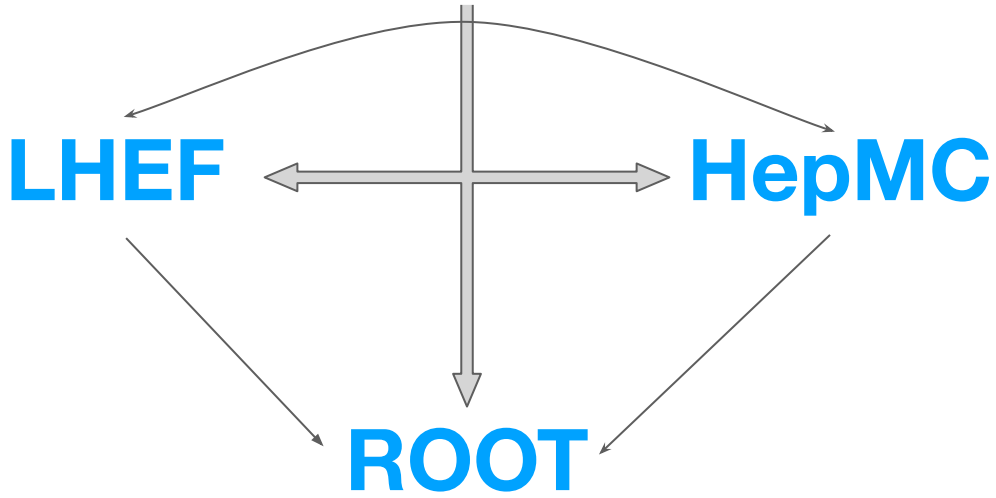
- More in the next slides

An important remark: output

- Each generator natively produces a specific output, which may or not be compatible with our framework

BUT FORTUNATELY…

Most of the recent generators are compatible to standard output formats



Various conversion tools are available

Les Houches Event file

- It originated from the Les Houches Accords (2001) as a simple, text-based format designed to facilitate interoperability between matrix element generators and parton shower or experimental frameworks

- Basically XML with extra steps → text-based format primarily designed for **parton-level (matrix element) event information**.

- Stored info:
  - initial beams
  - event weights
  - cross sections
  - particles

powheg.lhe

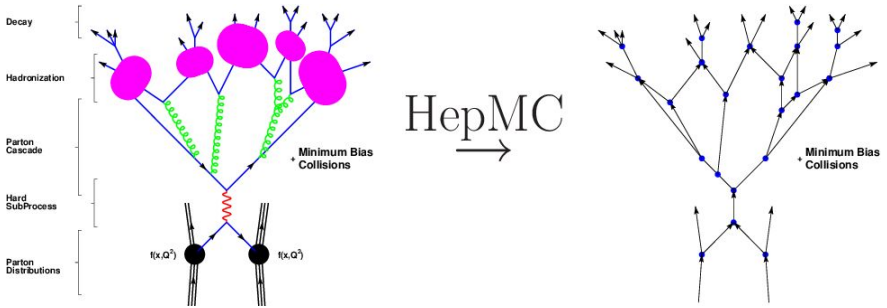
```
<LesHouchesEvents version="3.0">
<!--
file generated with POWHEG-BOX-V2
... Info about configuration file here ...
<init>
    2212     2212  6.80000E+03  6.80000E+03    -1    -1    -1    -1    -4     1
 4.44226E+08  4.95381E+05  1.00000E+00   1005
</init>
<event>
     5   1005  4.48703E+08  7.22898E+00 -1.00000E+00  2.13591E-01
    21    -1    0    0   504   505  0.000000000E+00  0.000000000E+00  1.127519888E+02  1.127519888E+02  0.000000000E+00  0.00000E+00  9.000E+00
    21    -1    0    0   502   503  0.000000000E+00  0.000000000E+00 -4.149880724E+00  4.149880724E+00  0.000000000E+00  0.00000E+00  9.000E+00
     5    1    1    2   502     0  6.866791512E+00 -3.426695095E+00  1.625574357E+00  9.170608319E+00  4.750000000E+00  0.00000E+00  9.000E+00
    -5    1    1    2     0   505 -6.765662753E-02  9.711072900E-01  2.728102216E+01  2.770856000E+01  4.750000000E+00  0.00000E+00  9.000E+00
    21    1    1    2   504   503 -6.799134885E+00  2.455587805E+00  7.969551155E+01  8.002270121E+01  1.348699152E-06  0.00000E+00  9.000E+00
#rwgt         1         1   543113996.83294153       107813255     22805060           0
</event>
</LesHouchesEvents>
```

- Allows storage of metadata blocks (<header>, <init>) and individual <event> blocks.

HepMC

- A flexible, object-oriented C++ library format for fully simulated Monte Carlo event records after showering and hadronization
- Stores particles, vertices, their relationships, and detailed event metadata
- ASCII readable user output

Physicist view



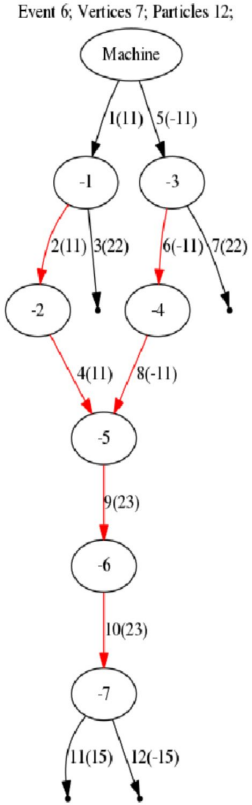

```
HepMC :: Version 3.0.0
HepMC :: Ascii3 - START_EVENT_LISTING
W 0
E 0 7 12
U GEV MM
W 1. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 00000 e +00
A 0 GenCrossSection 2.64422551 e +03 2.64422551 e +03 -1 -1
A 0 GenPdfInfo 11 -11 9.97420767 e -01 9.99999975 e -01 9.18812775 e +01 1.56824725 e +01 2.82148362 e +06 0 0
A 0 alphaQCD 0.129844
A 0 alphaQED 0.007818181
A 0 event_scale 91.88128
A 0 mpi 0
A 0 signal_process_id 221
A 0 signal_process_vertex 0
P 1 0 11 0.0000000000000000 e +00 0.0000000000000000 e +00 4.5999999997161737 e +01 4.6000000000000007 e +01 5.1099999999999995 e -04 4
P 2 1 11 0.0000000000000000 e +00 0.0000000000000000 e +00 4.5881355265109356 e +01 4.5881355265109356 e +01 0.0000000000000000 e +00 61
P 3 1 22 0.0000000000000000 e +00 0.0000000000000000 e +00 1.1864473489064407 e -01 1.1864473489064407 e -01 0.0000000000000000 e +00 1
P 4 0 -11 0.0000000000000000 e +00 0.0000000000000000 e +00 -4.5999999997161737 e +01 4.6000000000000007 e +01 5.1099999999999995 e -04 4
P 5 4 -11 0.0000000000000000 e +00 0.0000000000000000 e +00 -4.5999998855671230 e +01 4.5999998855671230 e +01 0.0000000000000000 e +00 61
P 6 4 22 0.0000000000000000 e +00 0.0000000000000000 e +00 -1.1443287704082650 e -06 1.1443287704082650 e -06 0.0000000000000000 e +00 1
P 7 2 11 0.0000000000000000 e +00 0.0000000000000000 e +00 4.5881355265109356 e +01 4.5881355265109356 e +01 0.0000000000000000 e +00 21
```

example.hepmc

HepMC

- A flexible, object-oriented C++ library format for full simulated Monte Carlo event records after showering and hadronization
- Stores particles, vertices, their relationships, and detailed event metadata
- ASCII readable user output
- Widely used throughout the HEP community
- Natively supports LHEF handling and ROOT I/O for easier analysis
- Provides a basic event browser →

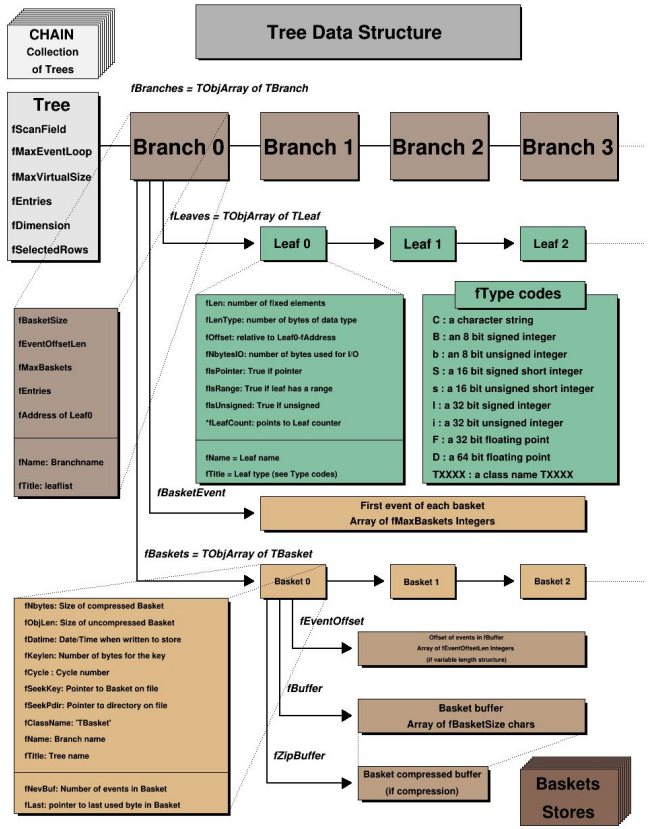
Pythia 8.2.40 simulated $e^+e^- \rightarrow \tau^+\tau^-$ with initial state radiation and w/o hadronisation and decays. Event info are shown, Ellipses are vertices and arrow are particles..



ROOT

- ROOT is a C++ data analysis framework developed at CERN
- Stores simulation and experimental data in highly compressed, tree-like structures called TTrees and TNtuples (float based branches) →
- Supports efficient data handling and analysis of large datasets (advanced math functions available
- Fully interfaced with operating system (files, network, threads…)
- Has in-built graphical tools for TTree browsing and fast histogramming

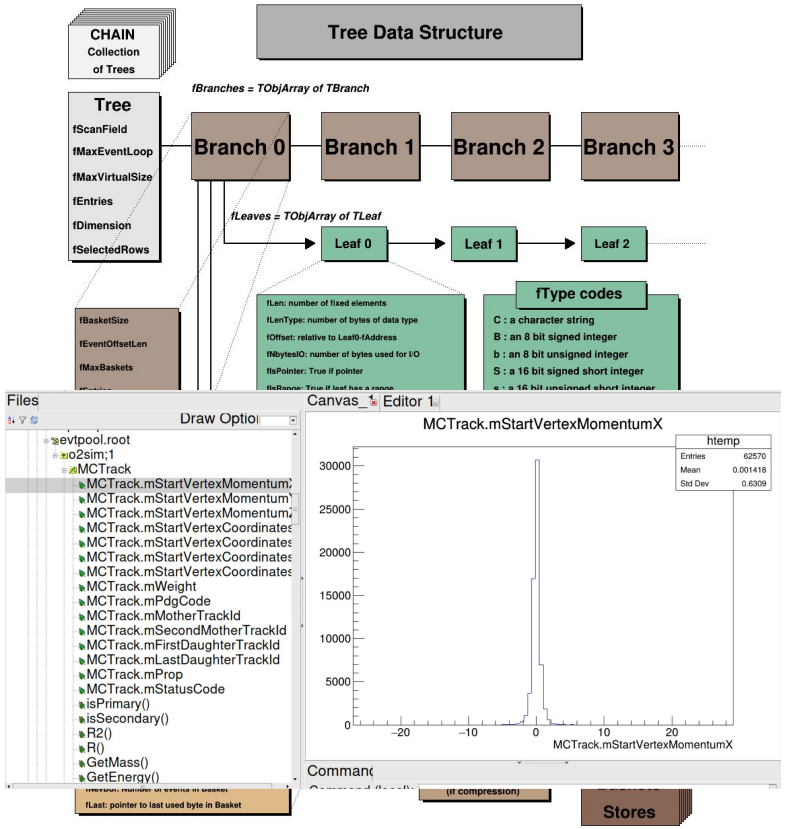
ROOT is the standard at the LHC since 2008
→ ALICE framework is based on it



ROOT

- ROOT is a C++ data analysis framework developed at CERN
- Stores simulation and experimental data in highly compressed, tree-like structures called TTrees and TNtuples (float based branches)
- Supports efficient data handling and analysis of large datasets (advanced math functions available
- Fully interfaced with operating system (files, network, threads…)
- Has in-built graphical tools for TTree browsing and fast histogramming →

ROOT is the standard at the LHC since 2008
→ ALICE framework is based on it



Generators native support

Generator	LHEF	HepMC	ROOT
DPMJET	No	Yes	No
POWHEG	Yes	No	No
PYTHIA 8	R/W	Yes	Yes
ThePEG	Yes	Yes	No
SHERPA	Yes	Yes	No
JETSCAPE	No	Yes	No
CRMC	No	Yes	Yes
EPOS4	No	Yes	Yes

Generator	LHEF	HepMC	ROOT
STARlight	No	Yes	No
Upcgen	No	Yes	No
Graniitti	Yes	Yes	Yes
nOOn	No	No	Yes
EvtGen	No	No	Yes
Herwig7	Yes	Yes	No

However, ROOT format is obtainable by all generators via analysis tools → automatised in ALICE framework

Why so many generators?

- **Physics phenomena are extremely various** ⇒ Different generators (generally developed by different groups) specialize in different processes or approximations

- **No theory is perfect (so far…)** ⇒ different theoretical models and approaches on how to simulate parton showers, hadronization, heavy flavor treatment, etc. are implemented in each generator

- **Some generators are collision or process specific** and **legacy tools are still used as reference**

- **Validation and cross-checking** ⇒ Comparing results from several generators reduces systematic uncertainties and ensures robust theoretical predictions

- **Computational Performance Trade-offs** ⇒ some generators prioritize speed, others accuracy (also model based)

Let’s talk about some models…

Extremely detailed recent review

Pythia8

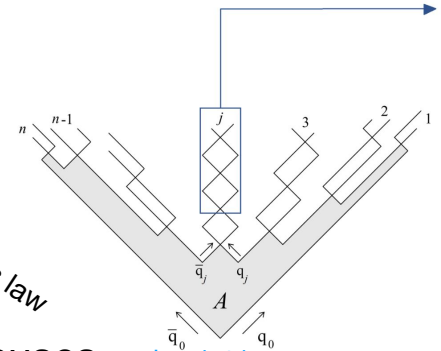
- PYTHIA 8 is a general-purpose Monte Carlo event generator written in C++

- Partons hadronization, in the default Monash 2013 tune, is based on the Lund string fragmentation model → interaction between two partons occurs through coloured fields represented by strings

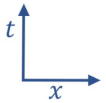
- A quark-antiquark pair interaction can be described by the Cornell potential

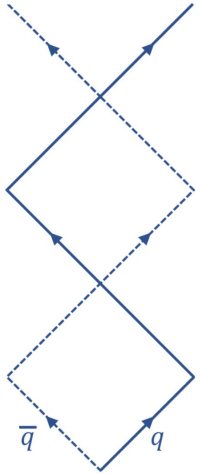
$$V(r) = \kappa r - \alpha r^{-1} \xrightarrow{\text{with increasing distance}} V(r) = \kappa r$$

Hooke's law



Lund strings fragmentation





yo-yo process

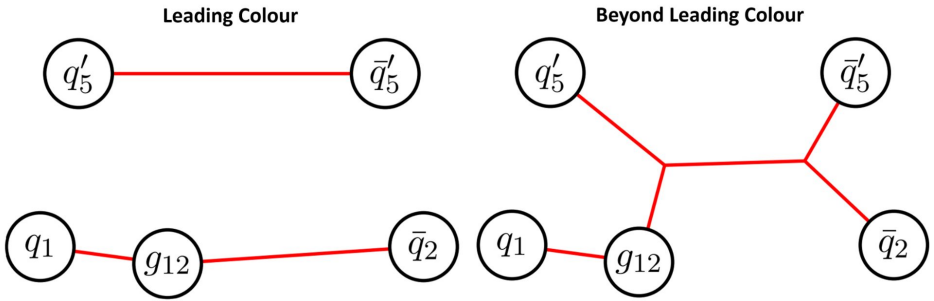
- Increasing the distance between the partons causes the colour string to stretch → the creation of a new pair will become more favourable than further extending the colour string (energy linear relation)

- Both pairs are forced together by the linear potential when distance increases in a motion called yo-yo → quarks can recombine with other pairs producing hadrons

Pythia8 color reconnection (CR)

- The Lund string model alone does not describe the production of some heavy-flavour hadrons in pp collisions at the LHC → other theoretical approaches are needed, especially for baryons

- PYTHIA uses a leading-colour approximation to trace the colour flow for each event
  a. quarks in partonic final state are connected uniquely to a single other parton in the event
  b. gluons carry both colour and anticolour → they are connected to two partons by two colour strings

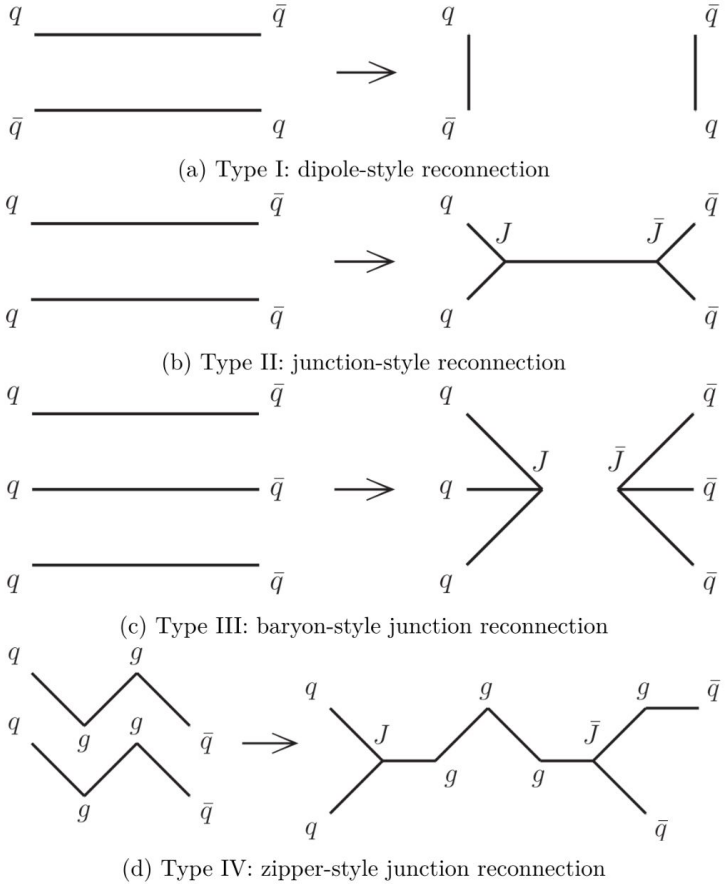


simple junction topologies

Pythia8 enhanced CR

arXiv:1505.01681

- In pp collisions events with multiple parton interactions can occur → all the partonic systems generated are to be considered
- Recombination of quarks is allowed both among partons arising from MPIs and from beam remnants → more string topologies that will hadronize
- New complex junctions are available →
- The achievable topologies in an event are ruled by the minimisation of the string potential energy
- Different model modes are available by tuning various parameters → hadrons production affectec

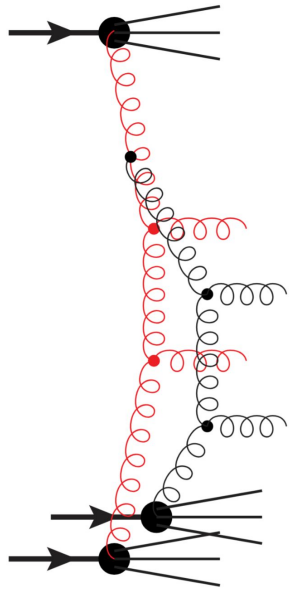


Pythia8 Angantyr

Angantyr model

- Angantyr extends PYTHIA8 compatibility to heavy-ion collisions by stacking multiple nucleon-nucleon collisions
- Glauber formalism with Gribov fluctuations for projectile nuclei
- Wounded (collision participating) nucleons are tagged with the type of collisions
- Pythia machinery is used to generate multiple nucleon-nucleon sub-collisions
- Multiple scattering between one projectile and two target nucleons → cannot be simulated directly by Pythia

Second interaction is directly colour connected to the first one

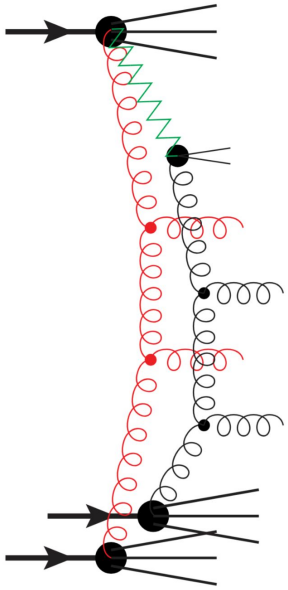


Pythia8 Angantyr

Angantyr model

- Angantyr extends PYTHIA8 compatibility to heavy-ion collisions by stacking multiple nucleon-nucleon collisions
- Glauber formalism with Gribov fluctuations for projectile nuclei
- Wounded (collision participating) nucleons are tagged with the type of collisions
- Pythia machinery is used to generate multiple nucleon-nucleon sub-collisions
- Multiple scattering between one projectile and two target nucleons cannot be simulated directly by Pythia
  ⇒ single diffraction machinery is modified and the scattering is generated as two pp events stacked together →

Second nucleon is only diffractively excited by a Pomeron exchange



EPOS4

Website

- It models **soft and hard QCD processes**, including multi-parton interactions, string formation, and hadronic cascades

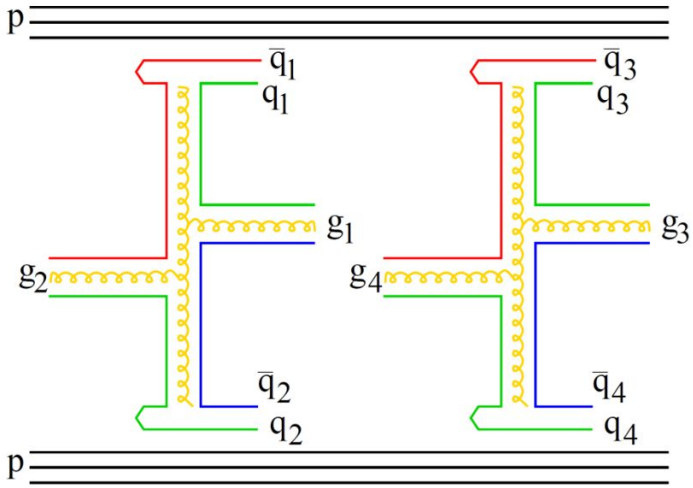
- The model is rooted in a **Gribov-Regge multiple scattering framework**, with parallel “primary scatters” and energy sharing between them → earliest stage of collision (t = 0) includes many parton-parton scatterings

- The generator provides a hydrodynamical evolution in pp, p–A and heavy-ion collisions through parton ladders which show up as flux tubes (strings)

- Possibility to use the Ultra-relativistic Quantum Molecular Dynamics (UrQMD) model for final-state hadronic transport → system evolution after hydrodynamic freeze-out, the system evolves via UrQMD to account for late-stage hadronic rescattering

EPOS4: a small example

- A *gg* → *gg* process is useful to understand better the concepts:
  - Two pomerons (parton ladders)
  - Interaction between quarks and antiquarks for each pomeron
  - Kinky strings can be identified following color flows (q_1 - g_1 - $\bar{q}_2$, q_2 - g_2 - $\bar{q}_1$, etc.)
    → 1D color flux tubes



Simplified pp scattering, with double-pomeron exchange, showing color flow

- Quark-antiquarks pairs are generated when the strings break creating hadrons and resonances → similarly jets are generated through segmentation of high-pT hard partons

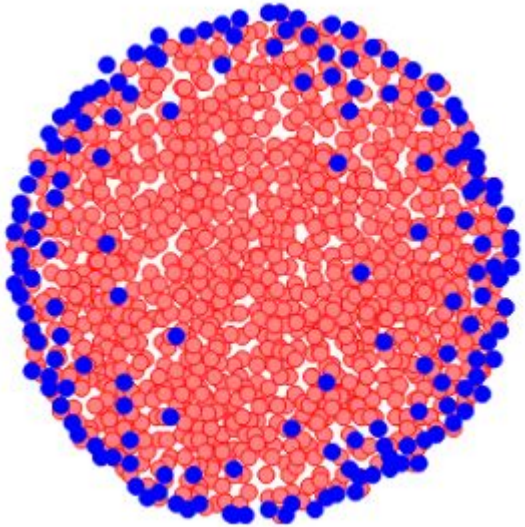
Is this always valid?

EPOS4: core-corona

- In AA and high multiplicity p-A/pp scatterings at high energy, the previous mechanism is not sufficient to describe particle production → high string density → segmentation only decay is not possible

- Flux tubes make thermalising bulk matter that expands collectively, called *core*

- String segments close to the surface of the bulk matter or with a high p_T can leave the core and hadronise → *corona* region



Corona = Blue, Core = Red

String segments form bulk matter based on p_T and local density → corona particles appear as hadrons while the core ones provide the initial condition for hydrodynamical evolution → they will hadronise later during medium freeze-out

HERWIG7

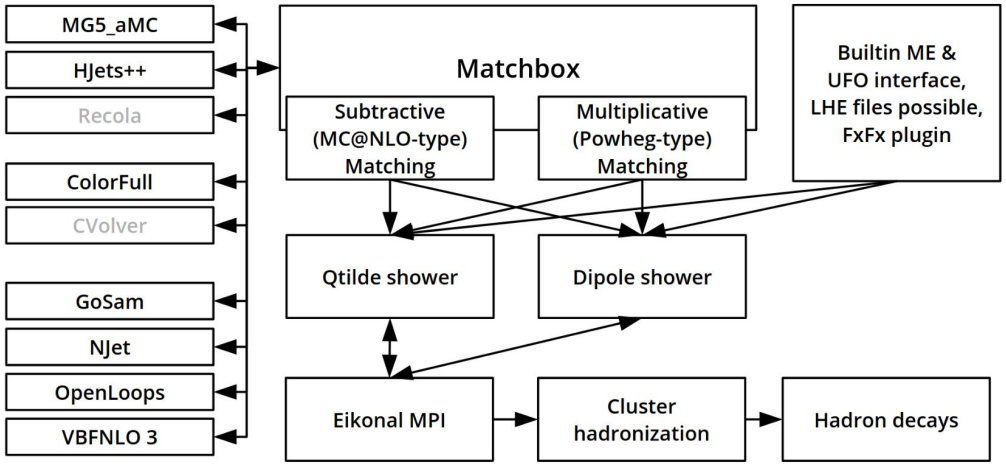
Website

- The multi-purpose generator is used for e^+e^-, ep and pp collisions → no heavy-ion collisions available

- It provides QCD simulations at next-to-leading order for virtually all SM processes → many external libraries fully integrated in the generator

- Parton showers developed with complex mechanism via the coherent branching algorithm

- Uses a cluster model (no strings) for hadronization → executed after the parton showering for quarks and gluons to combine



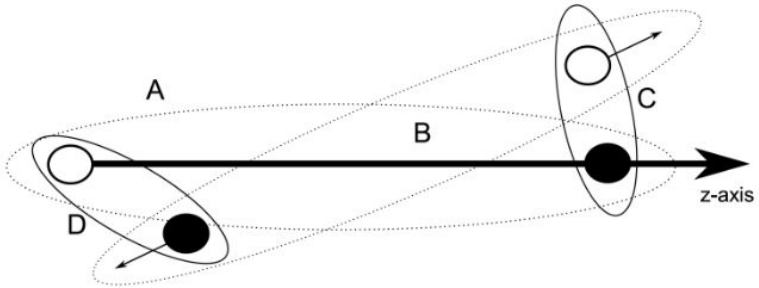
Simple scheme leading to hadron decays from amplitude evaluation in Herwig7

HERWIG7 CR

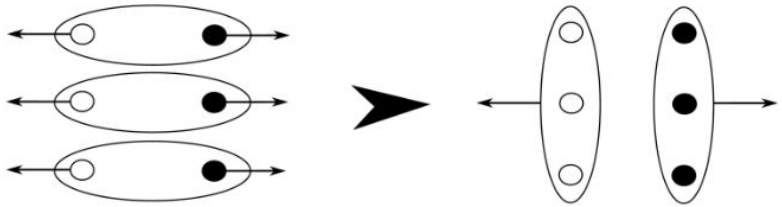
3 color reconnection models are implemented (and tunable) in HERWIG:

Mesonic

- Plain reconnection model ⇒ a quark can reconnect with all other clusters existing at that time, but new generated clusters invariant mass sum must be lower (p_{reco})
- Statistical model ⇒ finding a cluster configuration with a preferably small colour length $\lambda = \sum_{i=1}^{N_{cl}} m_i^2$



Mesonic reconnection



Baryonic reconnection

- Baryonic reconnection model ⇒ no reduction of the cluster masses (due to large baryonic mass) → geometric reconnection → quarks in same phase space region based on rapidity → two baryonic clusters out of three mesonic ones

What about transport?

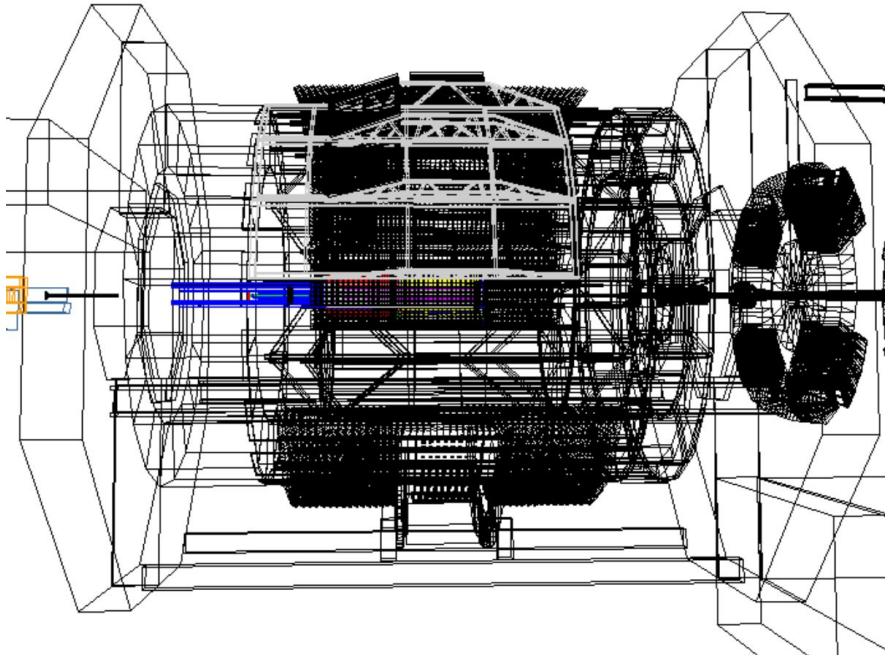
- Simulation transport == particles propagation through a virtual representation of detector ⇒ simulate material interaction ⇒ generate new particles ⇒ deposit energy in the detector and create “hits”

- ALICE geometry for simulation is defined via TGeo from ROOT ⇒ used also for reconstruction and event display

Library has native visualisation and browsing capabilities

Subdetectors teams provide geometries as C++ code + parameters (usually in XML/JSON)

- Geometries are used in transport engines which take care of particles decays and interactions using physics lists

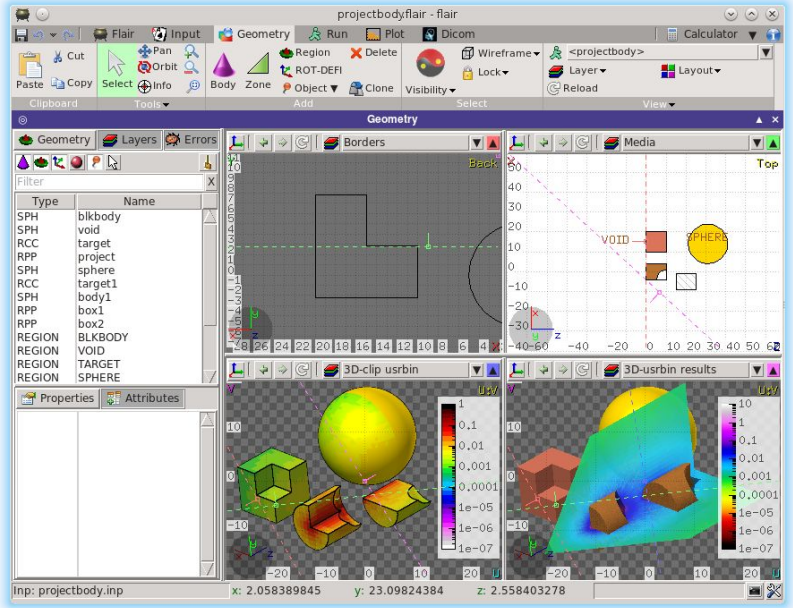


Current simulated ALICE central barrel geometry

FLUKA

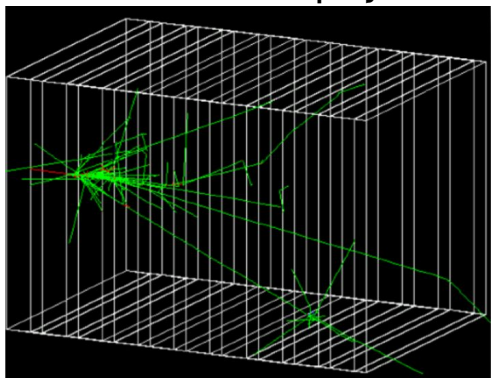
Resource Website

- FLUKA is a general purpose Monte Carlo tool for particle interaction and transport → mostly used for **accelerator design, shielding** and **radiation protection**
- Born in the 60s by J. Ranft → modern version released under CERN copyright → latest FLUKA 4.5.1 was released on 24/09/2025
- Usage of fully integrated and highly optimized models for em and hadronic interactions
- Source available under restrictive licenses → CERN or institutions with FLUKA license
- Provides an advanced GUI → Flair



Website

- **Transport standard in ALICE** and in many other HEP experiments
- GEANT4 is a C++ transport engine → previous version was in Fortran
- Completely open source → very large user base in many fields
- Physics processes and interactions are customisable → users can write their own physics lists or choose one of the many already available
- Well maintained and supported → latest release 28/04/2025
- Integration with ROOT, Qt, OpenGL for easy visualization and geometry creation → supports live event display →
- Natively allows multi-threading



50 MeV e^- entering calorimeter

How is everything integrated in ALICE?

Software environment

Important:

Users without an AliEn certificate will experience a limited usage of the next presented tools

simplest local build (basic generators such as PYTHIA8)

```
aliBuild build O2 O2DPG --defaults o2

alienv enter O2/latest,O2DPG/latest
```

full local build (all generators with AliGenO2, QC and O2Physics included)

```
aliBuild build O2sim --defaults o2

alienv enter O2sim/latest
```

MC stable releases → listed here

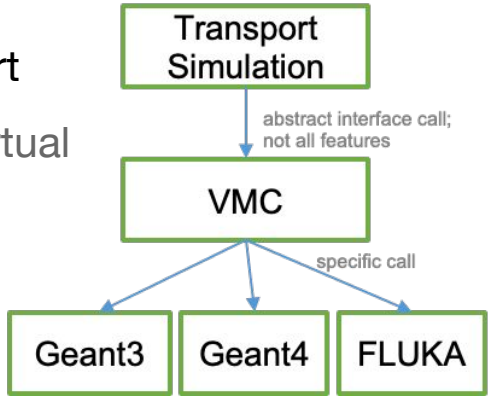
```
/cvmfs/alice.cern.ch/bin/alienv enter O2PDPSuite::MC-prod-2025-v13-1
```

o2-sim: ALICE Run3 simulation tool

- **o2-sim is the particle-detector simulator** for ALICE Run3 provided by the O2sim package

Implements ALICE detector on top of well known particle-transport engines that implement actual physics models and particle transport

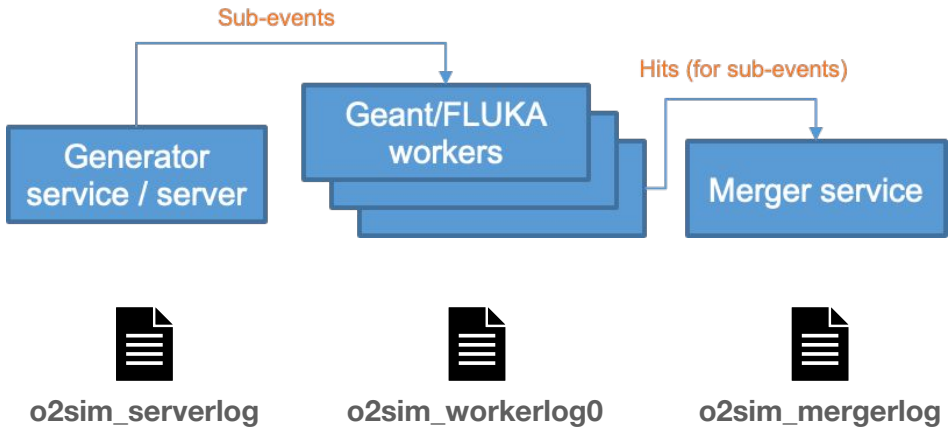
- Geant4, Geant3 and FLUKA interchangeably through use of Virtual Monte Carlo API



- **Main tasks of o2-sim**:
  - ALICE geometry creation
  - Event generation (primary particle generation)
  - Simulation of physics interaction of particles with detector material (secondary creation, etc.) and transport of particles until they exit detector or stop
  - Creation of hits (energy deposits) as a pre-stage of detector response after particle passage

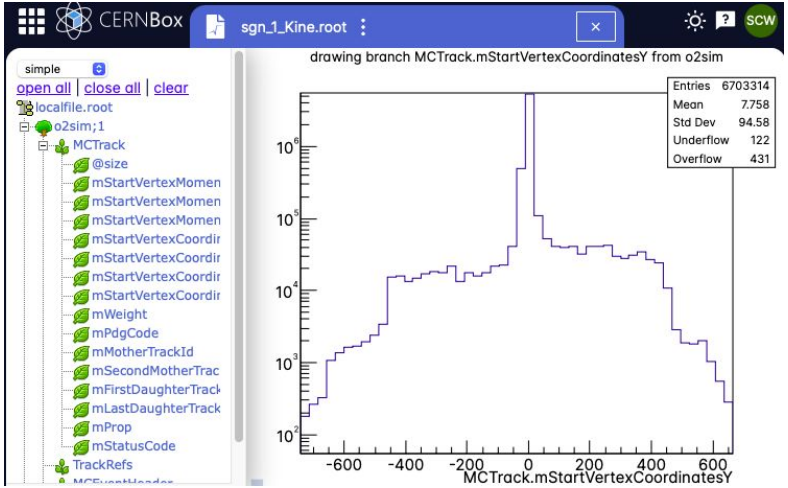
o2-sim: ALICE Run3 simulation tool

- New in Run3: **scalable multi-core simulation with sub-event parallelism**
  → allows to use big servers and obtain results for individual large events quickly
- Important: o2-sim treats **events in complete isolation** - **no timeframe concept** (enters during digitization)
- o2-sim produces 3 internal log files → in-depth description of each process and debug



o2-sim: Kinematics output

- Kinematics output (default file o2sim_**Kine.root**) from transport simulation likely most interesting for physics analysis
  - contains creation vertices, momenta, etc of **primary (generator)** and **secondary (transport) particles** created in simulation
  - information on physics creation process, provenance (mother-daughter), etc.
  - Based on o2::MCTrack class, which is basically a more lightweight TParticle
- For each event, there is one entry of vector<MCTracks> in a TTree
- By default, kinematics is pruned (only relevant particles kept)
- In addition, event-level meta-information about each generated event is available in a separate file (o2sim_**MCHeader.root**)
  - for instance impact parameter of PbPb collision



"histogram of production vertex-y of all MCtracks (primary and secondary)"

👉 Example: Run/SimExamples/Jet_Embedding_Pythia8

Helper classes to access MC kinematics

- Reading and navigating manually through kinematics can be cumbersome (“ROOT-IO boilerplate”)
- Offer 2 main utility classes making this easy for user
  - **MCKinematicsReader** - Class to easily read and retrieve tracks for given event or a Monte Carlo label
  - **MCTrackNavigator** - Class to navigate through mother-daughter tree of MC tracks and to query physics properties

```
using o2::steer;
using o2;

// access kinematics file with simulation prefix o2sim
MCKinematicsReader reader("o2sim",MCKinematicsReader::Mode::kMCKine);

// get all Monte Carlo tracks for this event
std::vector<MCTrack> const& tracks = reader.getTracks(event);

for (auto& t : tracks) {
  // analyse tracks; fetch mother track of each track (in the pool of all tracks)
  auto mother = o2::mcutil::MCTrackNavigator::getMother(t, tracks);
  if (mother) {
    std::cout << "This track has a mother\n";
  }
  // fetch the (backward first) primary particle from which this track derives
   auto primary = o2::mcutil::MCTrackNavigator::getFirstPrimary(t, tracks);
}
```

“Read all Monte Carlo tracks from stored kinematics file for event id 1. Then loop over all tracks and determine the direct mother particle and the primary ancestor in each case”

More in the next lecture